



UNIVERSIDAD POLITÉCNICA DE MADRID

FACULTAD DE INFORMÁTICA

Departamento de Lenguajes y Sistemas Informáticos  
e Ingeniería del Software

PHD THESIS

CHARACTERISATION SCHEMA  
FOR SELECTING  
SOFTWARE TESTING TECHNIQUES

AUTHOR: SIRA VEGAS

ADVISORS: VICTOR R. BASILI  
(University of Maryland)  
NATALIA JURISTO  
(Universidad Politécnica de Madrid)

Madrid, February 2002



To mum, dad, Isa and Santi.



# Acknowledgments

The author of this dissertation is indebted to many people, without whom this volume would never have seen the light of day.

Firstly, I would like to thank my dissertation supervisors, Dr. Victor Basili and Dr. Natalia Juristo, for the time and the guidance they have given me throughout. I am thankful to Dr. Victor Basili for the trust he placed in Dr. Natalia Juristo and in me when we embarked on this adventure of joint supervision. I will never be able to thank him enough for the opportunities he has given me to learn the working philosophy at the Experimental Software Engineering Group of the University of Maryland and for impressing upon me the importance of experimentation as a fundamental part of research work within the field of software engineering. I have so many things to thank Dr. Natalia Juristo for that I am afraid they would be too long to list. So, I will try to summarise them all by thanking her for what she has taught me professionally in relation to teaching and research and the valuable advice she has given me within the social province. Thanks for always finding time for me whenever I needed it.

Secondly, I would like to thank all the people who have given their views about the problem for their co-operation: all those who have acted as producers and consumers. I would like to thank Silvia Daiqui, Diego Daiqui, Brian Deyaert, Jeffrey Eng, Santiago Graña, Paul Jordan, Robert Moore, Ioana Rus, Guilherme Travassos, Karina Villela and Jeff Voas for their time. Antonia Bertolino, Dick Hamlet, José Carlos Maldonado and Dolores Wallace, who supervised and came up with highly useful remarks on the proposed solution, deserve a special mention.

I am also grateful to all the people at the School of Computer Science of the Technical University of Madrid and the Experimental software Group of the University of Maryland who have contributed in any way to the development of this PhD dissertation. In particular, I would like to thank: Ana and scar, for all the favours they have done for me over the years and thanks to which this dissertation has taken shape; Rachel, for her hard work, without which the English version of this dissertation would not have been a shade of what it is; and Luis de Ledesma, Cristina López, Begoña Portes and Beatriz Díaz, for their efficiency and obligingness when I had trouble with paperwork.

Finally, I would like to thank my Spanish and American families and Santi for the support they have given me during the gestation of this dissertation. My mother, for always looking out for me and, especially, my eating habits. My father and my sister for trying to make

sure I never lost contact with reality (always from different perspectives). Santi, for cordially putting up with me and my moods and playing secretary whenever he could. And John and Tony for playing mother and looking after me during my time at Maryland.

# Abstract

The importance of properly selecting testing techniques is widely accepted in the software engineering community. However, there are chiefly two reasons why the selections now made by software developers cannot be classed as correct. Firstly, they have limited knowledge of all the techniques now available, which means that there are a lot of techniques with which the average developer is unfamiliar. Secondly, the information now available with regard to the different existing testing techniques is mostly procedural (that is, focused on how to use the technique), whereas there is almost no pragmatic information (focused on the result of using the technique). The open problem addressed in this research is precisely how to help developers improve software testing technique selection.

A testing technique characterisation schema is proposed to achieve this objective. Being instantiated for multiple techniques, the schema can be used to build a repository containing information on testing techniques. This schema systematically describes all the testing techniques, focusing mainly on pragmatic aspects of the techniques, leading to more objective selections. The proposed characterisation schema is composed of a non-flat set of attributes, grouped around the elements of the testing process to which they refer. These elements are then grouped around the different testing process stages. This logical grouping makes the schema information coherent.

An empirical and iterative process was followed to arrive at this schema. An empirical process was used, because testing techniques are not founded on a solid theoretical basis. This means that the schema needs to be founded not only on testing theory, but also on what the different subjects related to software testing know about techniques. It is iterative, because a schema based on the theory now existing on testing will be created. This schema is gradually refined with the knowledge of developers, researchers and experts in the area. The completed characterisation schema was evaluated in two different ways. Firstly, the schema is verified empirically by instantiating it for multiple testing techniques. Secondly, an experiment is carried out, in which the repository created in the earlier verification is put into use to select testing techniques for different projects.

Finally, the original contribution of this research is a conceptual tool that can be used by developers to systematically and objectively select the testing techniques to be used in a software project.





# Contents

<b>Part I</b>	<b>INTRODUCTION TO THE RESEARCH</b>	<b>1</b>
<b>1</b>	<b>Introduction</b>	<b>3</b>
1.1	Research Area . . . . .	3
1.2	Some Software Testing Definitions . . . . .	4
1.3	The Problem: Characterisation of Testing Techniques . . . . .	7
1.4	Importance of the Problem . . . . .	9
1.5	Problem-Solving Approach . . . . .	10
1.6	Organisation of the Document . . . . .	12
<b>2</b>	<b>State of the Art</b>	<b>15</b>
2.1	Scope . . . . .	15
2.2	Testing Area . . . . .	18
2.3	Area of Characterisation . . . . .	32
2.4	Conclusions on the State of the Art . . . . .	46
<b>3</b>	<b>Problem Statement</b>	<b>47</b>
3.1	Description of the Problem . . . . .	49
3.2	Problem-Solving Approach . . . . .	51
3.3	Hypothesis Research Goals . . . . .	53
<b>Part II</b>	<b>RESOLUTION</b>	<b>59</b>
<b>4</b>	<b>Problem Solving Process</b>	<b>61</b>
4.1	Objectives of Technological Research . . . . .	61
4.2	Application of the Scientific Method to Technological Research . . . . .	65
4.3	Expert Peer Review versus Experimental Testing . . . . .	69
4.4	Problem-Solving Process . . . . .	71
<b>5</b>	<b>First Generative Iteration: Deductive Theoretical Schema</b>	<b>77</b>
5.1	Testing Process . . . . .	77
5.2	Stratification of Testing-Related Information . . . . .	79
5.3	Testing Process Elements . . . . .	82
5.4	Attributes of the Theoretical Schema . . . . .	84

5.5	Result of the Construction of the Theoretical Schema . . . . .	98
5.6	Use and Evolution of the Characterisation Schema . . . . .	98
<b>6</b>	<b>Second Generative Iteration: Inductive Empirical Schema</b>	<b>109</b>
6.1	Data Collection . . . . .	110
6.2	Data Analysis . . . . .	114
6.3	Result of Building the Empirical Schema . . . . .	136
6.4	Study of the Evolution of the Empirical Schema . . . . .	136
<b>7</b>	<b>Synthesis of Perspectives: Proposal of the Preliminary Schema</b>	<b>149</b>
7.1	Rules of Synthesis . . . . .	149
7.2	Synthesis of the Theoretical and Empirical Schemas . . . . .	151
7.3	Result of Schema Synthesis . . . . .	155
7.4	Study of Schema Synthesis . . . . .	155
<b>8</b>	<b>Improvement of the Schema: Expert Peer Review</b>	<b>159</b>
8.1	Questionnaire for Experts . . . . .	160
8.2	Questionnaire Analysis Method . . . . .	160
8.3	Analysis of Responses . . . . .	163
8.4	Schema after Peer Review . . . . .	175
<b>Part III</b>	<b>VALIDATION AND CONCLUSIONS</b>	<b>179</b>
<b>9</b>	<b>Empirical Evaluation</b>	<b>181</b>
9.1	Choice of the Workload . . . . .	181
9.2	Analysis of the Results . . . . .	183
9.3	Conclusions on the Empirical Evaluation . . . . .	189
<b>10</b>	<b>Experimental Evaluation</b>	<b>191</b>
10.1	Objective of the Experiment . . . . .	191
10.2	Experiment Planning . . . . .	192
10.3	Experimental Design . . . . .	201
10.4	Data Analysis . . . . .	207
10.5	Conclusions on Experimental Evaluation . . . . .	250
10.6	Characterisation Schema Improvement . . . . .	251
<b>11</b>	<b>Conclusions and Future Research Lines</b>	<b>255</b>
11.1	Conclusions . . . . .	255
11.2	Future Lines of Research . . . . .	257

<b>Part IV</b>	<b>BIBLIOGRAPHY AND APPENDIXES</b>	<b>259</b>
	<b>Bibliography</b>	<b>261</b>
<b>A</b>	<b>Terminology</b>	<b>273</b>
<b>B</b>	<b>Details on Software Testing Techniques</b>	<b>275</b>
B.1	Software Testing Techniques . . . . .	275
B.2	Metrics for Software Testing Techniques Comparison . . . . .	276
<b>C</b>	<b>Related Work in the Testing Area</b>	<b>281</b>
C.1	Studies on the Relationships between Structural Testing Techniques . . . . .	281
C.2	Studies on Data Flow Testing Techniques . . . . .	283
C.3	Comparisons between Random and Partition Testing . . . . .	285
C.4	Comparisons between Functional and Structural Testing Techniques . . . . .	287
C.5	Studies on Mutation Testing Techniques . . . . .	289
C.6	Studies on Regression Testing Techniques . . . . .	290
C.7	Studies on Minimisation Testing Techniques . . . . .	292
C.8	Other Studies . . . . .	293
<b>D</b>	<b>Additional Information on the Empirical Schema</b>	<b>295</b>
D.1	Forms Used . . . . .	295
D.2	Answers from Respondents . . . . .	295
D.3	Data on Respondents Answers . . . . .	295
D.4	Information Supplied by Respondents . . . . .	316
<b>E</b>	<b>Additional Information on Expert Peer Review</b>	<b>327</b>
E.1	Questionnaires used in Expert Peer Review . . . . .	327
E.2	Answers Supplied by the Experts . . . . .	327
<b>F</b>	<b>Schema Instantiation</b>	<b>347</b>
<b>G</b>	<b>Additional Information on the Experiment</b>	<b>361</b>
G.1	Project Contexts . . . . .	361
G.2	Forms Used . . . . .	363
G.3	ANOVA Validity Testing . . . . .	364



# Details on Contents

<b>Part I</b>	<b>INTRODUCTION TO THE RESEARCH</b>	<b>1</b>
<b>1</b>	<b>Introduction</b>	<b>3</b>
1.1	Research Area . . . . .	3
1.2	Some Software Testing Definitions . . . . .	4
1.2.1	Terminology . . . . .	4
1.2.2	Testing Process . . . . .	5
1.3	The Problem: Characterisation of Testing Techniques . . . . .	7
1.4	Importance of the Problem . . . . .	9
1.5	Problem-Solving Approach . . . . .	10
1.6	Organisation of the Document . . . . .	12
<b>2</b>	<b>State of the Art</b>	<b>15</b>
2.1	Scope . . . . .	15
2.1.1	Testing . . . . .	16
2.1.2	Reuse . . . . .	16
2.1.3	Software Architectures . . . . .	17
2.1.4	Technology Selection . . . . .	17
2.1.5	Classification of the Study Areas . . . . .	18
2.2	Testing Area . . . . .	18
2.2.1	Descriptive Criteria . . . . .	18
2.2.2	Related Work in the Testing Area . . . . .	22
2.2.2.1	Studies on Relationships between Structural Techniques . . .	22
2.2.2.2	Studies on Data Flow Testing Techniques . . . . .	23
2.2.2.3	Comparisons between Random and Partition Testing Tech- niques . . . . .	25
2.2.2.4	Comparisons between Functional and Structural Techniques	26
2.2.2.5	Studies on Mutation Testing Techniques . . . . .	27
2.2.2.6	Studies on Regression Testing Techniques . . . . .	28
2.2.2.7	Studies on Minimisation Testing Techniques . . . . .	29
2.2.2.8	Other Studies . . . . .	31
2.2.3	Conclusions on the Testing Area . . . . .	32
2.3	Area of Characterisation . . . . .	32

2.3.1	Descriptive Criteria . . . . .	32
2.3.2	Related Work in the Characterisation Area . . . . .	38
2.3.2.1	Studies on Software Reuse . . . . .	38
2.3.2.2	Studies on Software Architectures . . . . .	41
2.3.2.3	Studies on Technology Selection . . . . .	42
2.3.3	Conclusions on the Area of Characterisation . . . . .	45
2.4	Conclusions on the State of the Art . . . . .	46
<b>3</b>	<b>Problem Statement</b>	<b>47</b>
3.1	Description of the Problem . . . . .	49
3.2	Problem-Solving Approach . . . . .	51
3.3	Hypothesis Research Goals . . . . .	53
3.3.1	Working Hypothesis . . . . .	54
3.3.2	Testing the Working Hypotheses . . . . .	55
<b>Part II</b>	<b>RESOLUTION</b>	<b>59</b>
<b>4</b>	<b>Problem Solving Process</b>	<b>61</b>
4.1	Objectives of Technological Research . . . . .	61
4.2	Application of the Scientific Method to Technological Research . . . . .	65
4.3	Expert Peer Review versus Experimental Testing . . . . .	69
4.4	Problem-Solving Process . . . . .	71
4.4.1	First Generative Iteration: Deductive Theoretical Schema . . . . .	72
4.4.2	Second Generative Iteration: Inductive Empirical Schema . . . . .	72
4.4.3	Synthesis of Perspectives: Proposal of the Preliminary Schema . . . . .	74
4.4.4	Improvement of the Schema: Expert Peer Review . . . . .	74
4.4.5	Empirical Evaluation . . . . .	75
4.4.6	Experimental Evaluation . . . . .	75
<b>5</b>	<b>First Generative Iteration: Deductive Theoretical Schema</b>	<b>77</b>
5.1	Testing Process . . . . .	77
5.2	Stratification of Testing-Related Information . . . . .	79
5.3	Testing Process Elements . . . . .	82
5.3.1	Tactical Level . . . . .	82
5.3.2	Operational Level . . . . .	83
5.4	Attributes of the Theoretical Schema . . . . .	84
5.4.1	Operational Level . . . . .	85
5.4.1.1	Agents . . . . .	85
5.4.1.2	Technique . . . . .	87
5.4.1.3	Results (or product) . . . . .	91
5.4.1.4	Object . . . . .	93
5.4.2	Tactical Level . . . . .	95

5.4.2.1	Objective . . . . .	96
5.4.2.2	Scope . . . . .	97
5.5	Result of the Construction of the Theoretical Schema . . . . .	98
5.6	Use and Evolution of the Characterisation Schema . . . . .	98
5.6.1	Primary Use of the Repository: Selection . . . . .	98
5.6.2	Secondary Uses of the Repository: Evolution . . . . .	103
5.6.2.1	Post-Selection Evolution . . . . .	105
5.6.2.2	Post-Use Evolution . . . . .	106
5.6.2.3	Evolution by Producer Feed . . . . .	106
5.6.3	Secondary Uses of the Repository: Research . . . . .	107
5.6.4	The Role of the Librarian in the Repository . . . . .	107
<b>6</b>	<b>Second Generative Iteration: Inductive Empirical Schema</b>	<b>109</b>
6.1	Data Collection . . . . .	110
6.1.1	Characterisation of the Population of Respondents . . . . .	110
6.1.2	Survey Coverage . . . . .	111
6.1.3	Form Building . . . . .	113
6.2	Data Analysis . . . . .	114
6.2.1	Creation of the Reference Set . . . . .	115
6.2.2	Analysis of the Reference Set . . . . .	115
6.2.2.1	Levels of the Empirical Schema . . . . .	115
6.2.2.2	Elements of the Empirical Schema . . . . .	121
6.2.2.3	Attributes of the Empirical Schema . . . . .	124
6.3	Result of Building the Empirical Schema . . . . .	136
6.4	Study of the Evolution of the Empirical Schema . . . . .	136
6.4.1	Schema Growth . . . . .	136
6.4.2	Importance of each Schema Attribute . . . . .	140
6.4.3	Schema Evolution . . . . .	146
<b>7</b>	<b>Synthesis of Perspectives: Proposal of the Preliminary Schema</b>	<b>149</b>
7.1	Rules of Synthesis . . . . .	149
7.2	Synthesis of the Theoretical and Empirical Schemas . . . . .	151
7.2.1	First Rule of Synthesis: Levels and Elements of the Preliminary Schema	151
7.2.2	Second Rule of Synthesis: Attributes of a Schema . . . . .	152
7.2.3	Third Rule of Synthesis: Equal Attributes of Two Schemas . . . . .	152
7.2.4	Fourth Rule of Synthesis: Similar Attributes in Two Schemas . . . . .	153
7.2.4.1	Identifier versus Tools . . . . .	153
7.2.4.2	Number of Generated Cases versus Cost of Execution . . . . .	154
7.2.4.3	Coverage versus Completeness . . . . .	154
7.3	Result of Schema Synthesis . . . . .	155
7.4	Study of Schema Synthesis . . . . .	155

<b>8</b>	<b>Improvement of the Schema: Expert Peer Review</b>	<b>159</b>
8.1	Questionnaire for Experts . . . . .	160
8.2	Questionnaire Analysis Method . . . . .	160
8.3	Analysis of Responses . . . . .	163
8.3.1	Part 1: Respondents . . . . .	163
8.3.2	Part 2: Generic Questions . . . . .	163
8.3.2.1	Utility . . . . .	163
8.3.2.2	Effectiveness . . . . .	164
8.3.2.3	Understandability . . . . .	164
8.3.3	Part 3: Attributes . . . . .	165
8.3.3.1	Redundancy . . . . .	165
8.3.3.2	Importance . . . . .	166
8.3.3.3	Superfluousness . . . . .	167
8.3.3.4	Completeness . . . . .	168
8.3.3.5	Names . . . . .	169
8.3.4	Part 4: Elements . . . . .	169
8.3.4.1	Groupings . . . . .	169
8.3.4.2	Attribute/Element Relationship . . . . .	170
8.3.4.3	Superfluousness . . . . .	170
8.3.4.4	Completeness . . . . .	170
8.3.4.5	Names . . . . .	170
8.3.5	Part 5: Levels . . . . .	171
8.3.5.1	Groupings . . . . .	171
8.3.5.2	Elements/Level Relationship . . . . .	171
8.3.5.3	Superfluousness . . . . .	171
8.3.5.4	Completeness . . . . .	171
8.3.5.5	Names . . . . .	172
8.3.6	Part 6: Other Remarks . . . . .	172
8.4	Schema after Peer Review . . . . .	175

## Part III VALIDATION AND CONCLUSIONS 179

<b>9</b>	<b>Empirical Evaluation</b>	<b>181</b>
9.1	Choice of the Workload . . . . .	181
9.1.1	Schema Instantiation . . . . .	182
9.1.2	Schema Use . . . . .	182
9.2	Analysis of the Results . . . . .	183
9.2.1	Feasibility . . . . .	183
9.2.1.1	Producer Viewpoint . . . . .	183
9.2.1.2	Consumer Viewpoint . . . . .	188
9.2.2	Schema Flexibility . . . . .	189



9.3	Conclusions on the Empirical Evaluation . . . . .	189
<b>10</b>	<b>Experimental Evaluation</b>	<b>191</b>
10.1	Objective of the Experiment . . . . .	191
10.2	Experiment Planning . . . . .	192
10.2.1	Parameters . . . . .	195
10.2.2	Factors and their Alternatives . . . . .	197
10.2.2.1	Method of Selection . . . . .	197
10.2.2.2	Software Project . . . . .	198
10.2.3	Response Variables . . . . .	199
10.3	Experimental Design . . . . .	201
10.3.1	Two-Factor Design with Replication . . . . .	203
10.3.2	Experimental procedure . . . . .	203
10.3.3	Threats to validity . . . . .	205
10.3.3.1	Internal Threats . . . . .	205
10.3.3.2	External Threats . . . . .	206
10.4	Data Analysis . . . . .	207
10.4.1	Characteristics of the Subjects . . . . .	209
10.4.2	Schema Efficiency . . . . .	210
10.4.2.1	Testing Technique Study Time . . . . .	210
10.4.2.2	Testing Technique Selection Time . . . . .	212
10.4.2.3	Time Consulting Doubts about the Schema . . . . .	213
10.4.2.4	Conclusions on Schema Efficiency . . . . .	214
10.4.3	Schema Usability . . . . .	215
10.4.3.1	Number of Problems Encountered . . . . .	215
10.4.3.2	Sort of Problems Encountered . . . . .	216
10.4.3.3	Number of Doubts about the Schema . . . . .	221
10.4.3.4	Sort of Attributes of the Schema Consulted . . . . .	222
10.4.3.5	Conclusions on Schema Usability . . . . .	224
10.4.4	Schema Completeness . . . . .	225
10.4.4.1	Amount of Information Used for Selection . . . . .	225
10.4.4.2	Sort of Information Used in Selection . . . . .	226
10.4.4.3	Amount of Information Missing for Selection . . . . .	231
10.4.4.4	Sort of Information that is Missing for Selection . . . . .	233
10.4.4.5	Conclusions on Schema Completeness . . . . .	237
10.4.5	Schema Effectiveness . . . . .	237
10.4.5.1	Number of Techniques Considered . . . . .	237
10.4.5.2	Number of Selected Techniques . . . . .	239
10.4.5.3	Sort of Techniques Selected . . . . .	241
10.4.5.4	Conclusions on Schema Effectiveness . . . . .	245
10.4.6	Schema Satisfaction . . . . .	245

10.4.7	Conclusions on Groups . . . . .	246
10.4.7.1	Analysis of the Learning Effect in Groups 1 and 3 . . . . .	247
10.4.7.2	Analysis of the Effect of the Method in Groups 2 and 4 . . . . .	248
10.5	Conclusions on Experimental Evaluation . . . . .	250
10.6	Characterisation Schema Improvement . . . . .	251
<b>11</b>	<b>Conclusions and Future Research Lines</b>	<b>255</b>
11.1	Conclusions . . . . .	255
11.2	Future Lines of Research . . . . .	257
<b>Part IV</b>	<b>BIBLIOGRAPHY AND APPENDIXES</b>	<b>259</b>
<b>Bibliography</b>		<b>261</b>
<b>A</b>	<b>Terminology</b>	<b>273</b>
<b>B</b>	<b>Details on Software Testing Techniques</b>	<b>275</b>
B.1	Software Testing Techniques . . . . .	275
B.2	Metrics for Software Testing Techniques Comparison . . . . .	276
<b>C</b>	<b>Related Work in the Testing Area</b>	<b>281</b>
C.1	Studies on the Relationships between Structural Testing Techniques . . . . .	281
C.1.1	Rapps and Weyuker . . . . .	281
C.1.2	Weiser, Gannon and Mc Mullin . . . . .	281
C.1.3	Clarke, Podgurski, Richardson and Zeil . . . . .	282
C.1.4	Ntafos . . . . .	282
C.1.5	Zeil . . . . .	282
C.1.6	Frankl and Weyuker . . . . .	283
C.1.7	Frankl and Weyuker . . . . .	283
C.2	Studies on Data Flow Testing Techniques . . . . .	283
C.2.1	Weyuker . . . . .	283
C.2.2	Frankl and Weiss . . . . .	284
C.2.3	Bieman and Schultz . . . . .	284
C.2.4	Hutchins, Foster, Goradia and Ostrand . . . . .	284
C.2.5	Frankl and Iakounenko . . . . .	285
C.3	Comparisons between Random and Partition Testing . . . . .	285
C.3.1	Duran and Ntafos . . . . .	285
C.3.2	Hamlet . . . . .	286
C.3.3	Hamlet and Taylor . . . . .	286
C.3.4	Weyuker and Jung . . . . .	286
C.3.5	Chen and Yu . . . . .	287
C.3.6	Chen and Yu . . . . .	287

C.3.7	Ntafos . . . . .	287
C.4	Comparisons between Functional and Structural Testing Techniques . . . . .	287
C.4.1	Myers . . . . .	287
C.4.2	Basili and Selby . . . . .	288
C.4.3	Kamsties and Lott . . . . .	288
C.4.4	Wood, Roper, Brooks and Miller . . . . .	288
C.5	Studies on Mutation Testing Techniques . . . . .	289
C.5.1	Offut and Lee . . . . .	289
C.5.2	Offut, Rothermel and Zapf . . . . .	289
C.5.3	Frankl, Weiss and Hu . . . . .	289
C.5.4	Wong and Mathur . . . . .	290
C.6	Studies on Regression Testing Techniques . . . . .	290
C.6.1	Rothermel and Harrold . . . . .	290
C.6.2	Rosenblum and Rothermel . . . . .	290
C.6.3	Graves, Harrold, Kim, Porter and Rothermel . . . . .	291
C.6.4	Vokolos and Frankl . . . . .	291
C.6.5	Wong, Horgan, London and Agrawal . . . . .	291
C.6.6	Rothermel, Untch, Chu and Harrold . . . . .	291
C.6.7	Elbaum, Mailshevsky and Rothermel . . . . .	292
C.6.8	Kim, Porter and Rothermel . . . . .	292
C.6.9	Rothermel and Harrold . . . . .	292
C.6.10	Rosenblum and Weyuker . . . . .	292
C.7	Studies on Minimisation Testing Techniques . . . . .	292
C.7.1	Wong, Horgan, London and Mathur . . . . .	292
C.7.2	Rothermel, Harrold, Ostril and Hong . . . . .	293
C.8	Other Studies . . . . .	293
C.8.1	Frankl, Hamlet, Littlewood and Strigini . . . . .	293
C.8.2	Ntafos . . . . .	293
C.8.3	Frankl and Deng . . . . .	293
C.8.4	Lauterbach and Randall . . . . .	294
<b>D</b>	<b>Additional Information on the Empirical Schema</b>	<b>295</b>
D.1	Forms Used . . . . .	295
D.2	Answers from Respondents . . . . .	295
D.3	Data on Respondents Answers . . . . .	295
D.4	Information Supplied by Respondents . . . . .	316
<b>E</b>	<b>Additional Information on Expert Peer Review</b>	<b>327</b>
E.1	Questionnaires used in Expert Peer Review . . . . .	327
E.2	Answers Supplied by the Experts . . . . .	327
<b>F</b>	<b>Schema Instantiation</b>	<b>347</b>

<b>G</b>	<b>Additional Information on the Experiment</b>	<b>361</b>
G.1	Project Contexts . . . . .	361
G.1.1	Parking Garage Project Context . . . . .	361
G.1.2	Loan Arranger Project Context . . . . .	362
G.1.3	Video Project Context . . . . .	362
G.1.4	WLMS Project Context . . . . .	363
G.2	Forms Used . . . . .	363
G.3	ANOVA Validity Testing . . . . .	364
G.3.1	Schema Efficiency . . . . .	378
G.3.2	Schema Usability . . . . .	378
G.3.3	Schema Completeness . . . . .	379
G.3.4	Schema Effectiveness . . . . .	379

# List of Figures

1.1	Instantiation of the characterisation schema. . . . .	11
2.1	Testing techniques by families. . . . .	21
3.1	Repository environment. . . . .	51
4.1	Preliminary problem-solving process. . . . .	65
4.2	Traditional classification of the sciences with respect to scientific research. . .	67
4.3	Deductive-inductive problem-solving process, with abduction. . . . .	69
4.4	Deductive/inductive problem-solving process with expert peer review. . . . .	70
4.5	Design of the i-th empirical schema. . . . .	73
5.1	Process of generating test cases. . . . .	78
5.2	Decomposition of the characterisation schema into levels. . . . .	81
5.3	Decomposition of the tactical level into elements. . . . .	83
5.4	Process of test case generation. . . . .	83
5.5	Decomposition of the operational level into elements. . . . .	84
5.6	Different types of adequacy criteria. . . . .	91
5.7	Determination of the bounded variables in the schema according to the project characteristics. . . . .	101
5.8	Pre-selection of techniques from the repository considering the bounded variables.	102
5.9	Relaxation of some bounded variables. The value of some variables is changed in order to allow the selection of a testing technique. . . . .	103
5.10	Final selection of techniques. The pre-selected techniques are studies in order to determine which is the most convenient for the current project. . . . .	104
5.11	Characterisation schema evolution. New techniques can be added to the repository, or new information can be provided. . . . .	105
5.12	Repository uses. . . . .	108
6.1	Company or institution at which the respondents work. . . . .	111
6.2	Position held by each of the respondents. . . . .	112
6.3	Qualifications of each of the respondents. . . . .	113
6.4	Consumer experience in software development. . . . .	113
6.5	Attributes each respondent contributed to the empirical schema. . . . .	138

6.6	Rate of characterisation schema growth. . . . .	139
6.7	Speed of characterisation growth. . . . .	140
6.8	Accumulated speed of growth of the characterisation schema. . . . .	141
6.9	Absolute importance of each empirical schema attribute. . . . .	142
6.10	Relative importance of each empirical schema attribute. . . . .	144
6.11	Absolute importance of each empirical schema element. . . . .	145
6.12	Relative importance of each empirical schema element. . . . .	145
6.13	Importance of each empirical schema level. . . . .	146
6.14	Time at which each empirical scheme attribute appeared. . . . .	147
7.1	Theoretical and empirical schemas. . . . .	150
7.2	Source of synthesised schema attributes. . . . .	157
8.1	Representation of the schema as a chart. . . . .	173
10.1	Estimated values for mean study time. . . . .	211
10.2	Estimated values for mean selection time. . . . .	213
10.3	Estimated values for number of problems encountered. . . . .	217
10.4	Estimated values for the frequency of problems encountered. . . . .	218
10.5	Column and row points for the method and problem variables. . . . .	219
10.6	Estimated values for the mean frequency of consultation of each attribute. . .	223
10.7	Estimated values for the mean frequency of consultation of each attribute. . .	224
10.8	Estimated values of the mean amount of information used. . . . .	227
10.9	Estimated values for the frequency of use of information. . . . .	229
10.10	Column and row points for the method and information variables. . . . .	230
10.11	Estimated values for the amount of missing information. . . . .	233
10.12	Estimated values for the mean frequency of missing information. . . . .	235
10.13	Estimated values for the mean frequency of missing information. . . . .	236
10.14	Column and row points for the method and missing information variables. . .	236
10.15	Estimated values for the number of techniques studied. . . . .	239
10.16	Estimated values for the mean number of techniques selected. . . . .	240
10.17	Estimated values for the mean frequency of technique selection. . . . .	243
10.18	Column and row points for the method and technique variables. . . . .	243
10.19	Use procedure for the characterization schema. . . . .	253
D.1	Consumer form. . . . .	296
D.2	Producer form. . . . .	297
D.3	Results of the interview with the first consumer. . . . .	298
D.4	Results of the interview with the second consumer. . . . .	299
D.5	Results of the interview with the third consumer. . . . .	300
D.6	Results of the interview with the fourth consumer. . . . .	301
D.7	Results of the interview with the fifth consumer. . . . .	302
D.8	Results of the interview with the sixth consumer. . . . .	303

D.9	Results of the interview with the seventh consumer. . . . .	304
D.10	Results of the interview with the eighth consumer. . . . .	305
D.11	Results of the interview with the ninth consumer. . . . .	306
D.12	Results of the interview with the tenth consumer. . . . .	307
D.13	Results of the interview with the eleventh consumer. . . . .	308
D.14	Results of the interview with the twelfth consumer. . . . .	309
D.15	Results of the interview with the thirteenth consumer. . . . .	310
D.16	Results of the interview with the first producer. . . . .	311
D.17	Results of the interview with the second producer. . . . .	312
D.18	Results of the interview with the third producer. . . . .	313
D.19	Information supplied by respondents (1/8). . . . .	318
D.20	Information supplied by respondents (2/8). . . . .	319
D.21	Information supplied by respondents (3/8). . . . .	320
D.22	Information supplied by respondents (4/8). . . . .	321
D.23	Information supplied by respondents (5/8). . . . .	322
D.24	Information supplied by respondents (6/8). . . . .	323
D.25	Information supplied by respondents (7/8). . . . .	324
D.26	Information supplied by respondents (8/8). . . . .	325
E.1	Questionnaire used for the expert peer review (1/5). . . . .	328
E.2	Questionnaire used for the expert peer review (2/5). . . . .	329
E.3	Questionnaire used for the expert peer review (3/5). . . . .	330
E.4	Questionnaire used for the expert peer review (4/5). . . . .	331
E.5	Questionnaire used for the expert peer review (5/5). . . . .	332
E.6	Answers given by expert 1 (1/4). . . . .	333
E.7	Answers given by expert 1 (2/4). . . . .	334
E.8	Answers given by expert 1 (3/4). . . . .	335
E.9	Answers given by expert 1 (4/4). . . . .	336
E.10	Answers given by expert 2 (1/3). . . . .	337
E.11	Answers given by expert 2 (2/3). . . . .	338
E.12	Answers given by expert 2 (3/3). . . . .	339
E.13	Answers given by expert 3 (1/3). . . . .	340
E.14	Answers given by expert 3 (2/3). . . . .	341
E.15	Answers given by expert 3 (3/3). . . . .	342
E.16	Answers given by expert 4 (1/3). . . . .	343
E.17	Answers given by expert 4 (2/3). . . . .	344
E.18	Answers given by expert 4 (3/3). . . . .	345
G.1	Form E0 (1/2). . . . .	365
G.2	Form E0 (2/2). . . . .	366
G.3	Form E1 (1/2). . . . .	367
G.4	Form E1 (2/2). . . . .	368

G.5 Form E2 (1/2).	369
G.6 Form E2 (2/2).	370
G.7 Form E3.	371
G.8 Form E4.	372
G.9 Form E5.	373
G.10 Form E6.	374
G.11 Form E7.	375
G.12 Form E8.	376
G.13 Form E9.	377
G.14 Validity of the ANOVA results for study time.	378
G.15 Validity of the ANOVA results for selection time.	378
G.16 Validity of the ANOVA results for the time spent consulting doubts about the schema.	379
G.17 Validity of the ANOVA results for number of problems encountered.	379
G.18 Validity of the ANOVA results for frequency of problems encountered.	380
G.19 Validity of the ANOVA results for the number of doubts about the schema.	380
G.20 Validity of the ANOVA results for the sort of doubts about the schema.	380
G.21 Validity of the ANOVA results for the amount of information used in selection.	381
G.22 Validity of the ANOVA results for the frequency of use of information.	381
G.23 Validity of the ANOVA results for the amount of missing information.	381
G.24 Validity of the ANOVA results for the frequency of appearance of missing information.	382
G.25 Validity of the ANOVA results for number of techniques considered.	382
G.26 Validity of the ANOVA results for the number of techniques selected.	383
G.27 Validity of the ANOVA results for the frequency of selection of each technique.	383



# List of Tables

1.1	Cost (effort) to repairing software in relationship to development stage. . . .	5
2.1	Value of the criteria of comparison for studies of relationships between structural techniques. . . . .	23
2.2	Value of the criteria of comparison for the studies on data flow testing techniques. 24	
2.3	Value of the criteria of comparison for comparisons between random and partition testing techniques. . . . .	26
2.4	Value of the criteria of comparison for functional and structural techniques. .	27
2.5	Value of the criteria of comparison for the studies on mutation testing techniques.	28
2.6	Value of the criteria of comparison for studies on regression testing techniques.	30
2.7	Value of the criteria of comparison for studies on minimisation testing techniques.	30
2.8	Value of the criteria of comparison for other studies on testing techniques. . .	31
2.9	Values of the criteria of comparison in the testing area (1/3). . . . .	33
2.10	Values of the criteria of comparison in the testing area (2/3). . . . .	34
2.11	Values of the criteria of comparison in the testing area (3/3). . . . .	35
2.12	Values of the criteria of comparison for the family of reuse. . . . .	41
2.13	Value of the comparison criteria for the family of technology selection. . . . .	43
2.14	Value of the comparison criteria in the area of characterisation. . . . .	44
3.1	Aspects of the characterisation schema for evaluation. . . . .	55
3.2	Questions obtained using the GQM. . . . .	58
5.1	Attributes of the agent element. . . . .	87
5.2	Attributes of the technique element. . . . .	91
5.3	Attributes of the results element. . . . .	94
5.4	Attributes of the object element. . . . .	96
5.5	Attributes of the objective element. . . . .	97
5.6	Attribute of the scope element. . . . .	98
5.7	Theoretical schema. . . . .	99
6.1	Content of the reference set (1/5). . . . .	116
6.2	Content of the reference set (2/5). . . . .	117
6.3	Content of the reference set (3/5). . . . .	118
6.4	Content of the reference set (4/5). . . . .	119

6.5	Content of the reference set (5/5).	120
6.6	Questions raised by respondents and rejected.	121
6.7	Grouping of the questions as levels.	122
6.8	Grouping of the tactical level questions as elements.	123
6.9	Grouping of the operational level questions as elements.	123
6.10	Grouping of the use level questions as elements.	124
6.11	Attributes generated by the objective element.	125
6.12	Attributes generated by the scope element.	126
6.13	Attributes generated by the agents element.	127
6.14	Attributes generated by the tools element (1/2).	128
6.15	Attributes generated by the tools element (2/2).	129
6.16	Attributes generated by the technique element (1/3).	130
6.17	Attributes generated by the technique element (2/3).	131
6.18	Attributes generated by the technique element (3/3).	132
6.19	Attributes generated by the results element.	133
6.20	Attributes generated by the object element.	134
6.21	Attributes generated by the project element.	135
6.22	Attributes generated by the satisfaction element.	135
6.23	Empirical schema.	137
7.1	Result of applying the first rule of synthesis.	152
7.2	Result of applying the second rule of synthesis.	153
7.3	Result of applying the third rule of synthesis.	154
7.4	Result of applying the fourth rule of synthesis.	155
7.5	Result of perspective synthesis: preliminary schema.	156
8.1	Schema after expert peer review (1/3).	176
8.2	Schema after expert peer review (2/3).	177
8.3	Schema after expert peer review (3/3).	178
9.1	Questions to be answered during empirical evaluation.	181
9.2	Associated variables.	188
10.1	Questions to be answered during the experiment.	192
10.2	Sources of variation when selecting testing techniques.	196
10.3	Characterisation of the software projects used for the experiment.	199
10.4	Experiment response variables.	200
10.5	Allocation of selection methods to groups.	201
10.6	Project allocation to subgroups.	202
10.7	Allocation of subjects to groups and subgroups.	203
10.8	Experimental procedure.	203
10.9	Mean and standard deviation for study time.	210
10.10	ANOVA for study time.	211

10.11	Estimated values for mean study time. . . . .	211
10.12	Mean and standard deviation for selection time. . . . .	212
10.13	ANOVA for selection time. . . . .	212
10.14	Estimated values for selection time. . . . .	213
10.15	Mean and standard deviation of the time spent consulting doubts about the schema. . . . .	214
10.16	ANOVA for the time spent consulting doubts about the schema. . . . .	214
10.17	Mean and standard deviation for number of problems encountered. . . . .	215
10.18	ANOVA for number of problems encountered. . . . .	216
10.19	Estimated values for number of problems encountered. . . . .	216
10.20	Description of the problems encountered during selection. . . . .	217
10.21	ANOVA for frequency of problems encountered. . . . .	218
10.22	Analysis of correspondence between the method and problem variables. . . .	219
10.23	Mean and standard deviation for the number of doubts about the schema. . .	221
10.24	ANOVA for the number of times schema help has been consulted. . . . .	221
10.25	Schema attributes. . . . .	222
10.26	ANOVA for the frequency of schema attribute consultation. . . . .	223
10.27	Amount of information considered in selection. . . . .	226
10.28	ANOVA for the amount of information used in selection. . . . .	226
10.29	Estimated values of the mean amount of information used. . . . .	226
10.30	Description of information used for selection. . . . .	228
10.31	ANOVA for the frequency of use of information. . . . .	229
10.32	Analysis of correspondences between the method and information variables. .	230
10.33	Mean and standard deviation for the amount of missing information. . . . .	232
10.34	ANOVA for the amount of missing information. . . . .	232
10.35	Estimated values for the amount of missing information. . . . .	232
10.36	Description of the missing information. . . . .	234
10.37	ANOVA for the frequency of appearance of missing information. . . . .	235
10.38	Analysis of correspondences between the method and missing information variables. . . . .	235
10.39	Mean and standard deviation for the number of techniques studied. . . . .	238
10.40	ANOVA for the number of techniques studied. . . . .	238
10.41	Estimated values of the number of techniques studied. . . . .	239
10.42	Mean and standard deviation for the number of techniques selected. . . . .	239
10.43	ANOVA for the number of techniques selected. . . . .	240
10.44	Estimated values for the mean number of techniques selected. . . . .	240
10.45	Description of the selected techniques. . . . .	241
10.46	ANOVA for the frequency of selection of each technique. . . . .	242
10.47	Analysis of correspondences between the method and technique variables. . .	242
10.48	Values of each response variable per session for groups 1 and 3. . . . .	247
10.49	Values of each response variable per session for groups 2 and 4. . . . .	249

D.1	Data on respondents answers. . . . .	314
D.2	Number of attributes incorporated by each respondent to the schema. . . . .	315
D.3	Growth rate and speed rate of the empirical schema. . . . .	315
D.4	Importance of each attribute in the empirical schema. . . . .	316
D.5	Importance of each element in the empirical schema. . . . .	317
D.6	Importance of each level in the empirical schema. . . . .	317
F.1	Boundary value analysis technique . . . . .	348
F.2	Random testing technique . . . . .	349
F.3	Sentence coverage technique . . . . .	350
F.4	Decision coverage technique . . . . .	351
F.5	Path coverage technique . . . . .	352
F.6	Threads coverage technique . . . . .	353
F.7	All-possible-rendezvous technique . . . . .	354
F.8	All-c-uses technique . . . . .	355
F.9	All-p-uses technique . . . . .	356
F.10	All-uses technique . . . . .	357
F.11	All-du-paths technique . . . . .	358
F.12	Mutation technique . . . . .	359
F.13	Selective mutation technique . . . . .	360

Part I

**INTRODUCTION TO THE  
RESEARCH**



# Chapter 1

## Introduction

### 1.1 Research Area

One of the hobbyhorses of software systems development has been and still is the construction of quality products that do not exceed the originally budgeted price and are delivered according to a pre-established schedule [Blum, 1992]. The endemic impossibility of achieving this objective was historically baptised as the *software crisis* [Naur and Randell, 1969] and mainly strikes complex projects. This problem first became apparent in the 60s, when computers started to become popular. Far from disappearing, the problem has become yet more serious, as the computer industry has continued to expand and the market demanded increasingly more complex systems.

Of the three objectives mentioned above software systems quality, construction within budget and prompt delivery-, **this piece of research focuses on software systems quality**. Quality control and assurance activities not only affect code, but also each and every one of the intermediate products generated during software construction. As conceived by Beizer [Beizer, 1990], the actions for producing quality systems can be generically divided into two groups: preventive and corrective. The main difference between the two is that preventive actions define a series of activities to be performed *a priori*, that is, during software construction, to ensure that the product output is not of poor quality, whereas corrective actions take place *a posteriori*, that is, once the software product has been constructed, when its quality is evaluated. In this case, if the value of software output is lower than desired, improvements to the software product (or the software construction process) are proposed. Preventive actions include software development good practice guidelines. Corrective actions encompass the activities proper to software evaluation. **This piece of research addresses research on corrective or software evaluation measures**.

As Harrold [Harrold, 2000] claims, evaluation is a highly important process, as it is directed at assuring software quality. Evaluation works by studying the software system to gather information about its quality. According to Juristo [Juristo, 1998], evaluation involves examining the developed product (code or other intermediate products) under evaluation and judging whether it meets the desired quality level (if it does, development can continue;

otherwise, the evaluated product should be reworked to raise its quality). The products obtained during software development can be evaluated according to two different strategies:

- Static analysis.
- Dynamic analysis, or testing.

These two strategies are often confused and both are mistakenly grouped under the term *testing* [Bertolino, 2001]. However, there are significant differences between static and dynamic analyses of a product. The difference between the two strategies is best understood by making a distinction between the aspects of the software products to be evaluated. On the one hand, there are static criteria, which are related to visible properties with the system at rest; on the other hand, there are dynamic criteria, which are related to properties that are only manifest when the system is operational.

Static analysis can be used to evaluate static criteria and involves examining the product under evaluation at rest. Dynamic analysis, or testing, can be used to evaluate dynamic criteria, which means that it examines the result of operating the system as opposed to the product directly. Therefore, the dynamic analysis of a software product implies execution, as only by studying the result of this execution is it possible to decide whether or not (or to what extent) the quality levels set for the dynamic aspects judged are met. The task performed by a system during execution for the purposes of running a dynamic analysis is what is known as workload or simply testing.

At present, the executable software product *par excellence* is code (although there are some executable specification and design languages, their use is not very widespread). Therefore, any product obtained during development (including code) can be evaluated by means of static analysis. However, dynamic analysis (or testing) almost exclusively evaluates codes. **This piece of research focuses on software testing or the dynamic evaluation of code.**

## 1.2 Some Software Testing Definitions

Before moving on to the research problem posed within the area of software testing, it is necessary to explain a series of concepts that can lead to confusion. In particular, this piece of research provides explanations of the terminology and testing process.

### 1.2.1 Terminology

Firstly, a distinction has to be made between terms that have a host of definitions, which can lead to confusion about their meaning. This refers in particular to the difference between fault, failure and error. This dissertation will use the definitions given by the IEEE [IEEE, 1983]:

- *Error*. People make errors when they reason incorrectly to try to solve a problem.



- *Fault*. An error becomes a fault when it is written (included) in any of the developed software products.
- *Failure*. Failures occur when a software system does not behave as desired (does not do what it should do), which reveals a fault in the software.
- The term *defect* will be generally used to refer to any of these concepts.

Developers make **errors** for any number of reasons: from a conceptual error, through a misinterpretation, to a simple distraction. Any of these errors can lead the developer to enter a **fault** in the software at some point. A fault can take many forms. At the very simplest, these faults can be classed into three groups: omission (something that should have been done has been overlooked), excess (something that should not have been done has been done) or commission (something has been done incorrectly). Two things can happen if the fault is propagated undetected through all the intermediate products to code: the fault either shows up or does not show up when the program is run. If it does show up, a **failure** will be said to have occurred, as the actual response does not coincide with the expected response.

The question of whether a fault is detected earlier or later is critical. As shows Table 1.1 [Davis, 1993], the later a fault is detected, the more costly it is to repair. This applies both after delivery and during development.

STAGE	RELATIVE COST OF REPAIR
Requirements	0.1–0.2
Design	0.5
Coding	1
Unit testing	2
Acceptance testing	5
Maintenance	20

Table 1.1: Cost (effort) to repairing software in relationship to development stage.

Table 1.1 shows that while it is true that a fault detected during unit testing is 10 times more expensive to repair than the same fault detected during requirements specification, this same fault is up to a hundred times more costly if detected during system operation. This explains why research is so important in the area of evaluation, generally, and, particularly, in testing. There is a need for methods to support defect detection during development and before the system is put into operation. Indeed, testing is the last chance during development to detect and correct faults that would be associated with all too costly failures if detected during system operation.

### 1.2.2 Testing Process

A process has to be followed to satisfactorily achieve the objective pursued by software testing. Objective achievement will depend on how good this process is.

There is sometimes some confusion as to the testing process and it is mistakenly thought that a testing technique outputs faults, when it really outputs test cases. On other occasions, one finds that the testing process in routine practice boils down to executing the software as many times as deemed necessary (or as often as there is time to), testing with randomly selected inputs. Neither the generated test cases, nor the stop testing condition are ever reported. Neither is a proper estimate of the time and resources that will be necessary to run the tests ever made. Bearing in mind that it is necessary to have a common view of the process to understand the remainder of the research, the parts of a testing process are summarised below.

Being a software process, the activities to be performed during testing are well defined. This piece of research takes the view of the PSS-05 standard [PSS-05, 1991]:

1. *Test planning.* The purpose of this activity is to identify the characteristics or quality attributes of the software to be tested, the rigour with which they are to be tested, any characteristics to not to be tested and the software elements that are to be tested. Also resources (personnel, tools, etc.) will be allocated to each of the tasks to be performed, and the testing techniques to be used will be selected.
2. *Test design.* For each test identified in the above activity, the selected testing techniques will be applied and the generated test cases will be identified.
3. *Test case specification.* For each generated test case, the elements it affects must be identified, and both the inputs required to execute the test case and the expected outputs and resources (software or hardware) required to run the test cases must be specified. Also the test cases will be ordered depending on how they are to be executed.
4. *Test procedure definition.* For each generated test case, the steps to be taken to correctly execute each generated test case must be specified.
5. *Test procedure execution.* Each generated test case will be executed according to the associated test procedure defined.
6. *Analysis of the results.* For each test procedure executed, the version of the software elements involved in the test, as well as the characteristics of the environment in which the test was run must be referenced. A description will be given of test execution (author, test starting and finishing time), as well as the results of the test (success or failure).

**This dissertation focuses on the second step or test design**, as it is the critical step in the testing process.

The testing process (the six above-mentioned steps) is considered as one of the most costly development processes [Beizer, 1990], sometimes exceeding fifty per cent of total development costs. Bearing in mind the testing process defined above, it can be found that one of

the many factors that influence the cost of testing is the number of test cases designed (step 2). This means that the more test cases are generated, the longer it will take to specify and execute the tests and, therefore, the more costly they will be. It is well known that exhaustive testing, or the execution of all the possible combinations of input values is unfeasible [Myers, 1979]. For example, it would take as many test cases as there are possible combinations for summing any two numbers to test a program as simple as a sum of two numbers, and this would consume precious time. Therefore, the tests will be run on a relatively small set [Beizer, 1990] previously selected from all the possible cases. The choice of test cases is of utmost importance, not only because the resulting set has to be reduced to the minimum, but also because this set must allow the software system to comprehensively demonstrate the aspect under evaluation [Beizer, 1990], [Clarke *et al.*, 1989]. If this aspect cannot be sufficiently exhibited using the selected set of test cases, the tests will have failed to achieve their objective, because they will not have examined the software properly.

Testing techniques actually focus on the choice of a set from all the possible test cases, an activity that conforms to the third point of the process explained above. This means that the problem of choosing test cases for a software project is actually the selection of one or more testing techniques. **Therefore, this piece of research focuses precisely on the selection of testing techniques.**

### 1.3 The Problem: Characterisation of Testing Techniques

As mentioned above, what are referred to as *testing techniques* have emerged to find a satisfactory set of test cases. Tens of techniques now exist [Frankl and Weyuker, 1991], [Frankl and Weyuker, 1993b], [Chen and Yu, 1996], which poses the following question. What differences are there among the techniques? The most evident differences refer to characteristics explained in the technique description. For example, they differ both as regards the inputs they require to develop the test cases, the form in which they choose the set of test cases, etc. However, paraphrasing Hamlet [Hamlet, 1989], distinctions of this sort are based on mechanical aspects (that is, the mechanics of the technique) and not on the efficacy or efficiency of the techniques. Nevertheless, a classification on the basis of the strengths and weaknesses of each technique (that is, on the basis of its efficiency or productivity, that is, on other theoretical, technical and pragmatic aspects) would be much more useful than classifications based on mechanical or operational considerations. Technique strengths and weaknesses provide useful information for developers by means of which they can distinguish under what project circumstances they should choose one or other technique.

Distinctions concerning both the mechanical aspects and the theoretical, technical and pragmatic aspects of the techniques can be found in the literature. Nevertheless, whereas the mechanical distinctions are well established and appear in any textbook on testing, the theoretical, technical and pragmatic questions are proper to research papers. Indeed, various studies concerning the distinctions based on theoretical, technical and pragmatic aspects can be found in the literature, ranging from theoretical studies, through simulations, to

experiments [Frankl and Weyuker, 1991]. These studies investigate the differences between certain techniques on the basis of a variety of criteria (like efficiency or effectiveness depending on the type of software being tested) and are motivated by the need to find out how a given technique behaves within a given project environment [Howden, 1977], [Shimeall and Leveson, 1988], [Ntafos, 1988], [Wong *et al.*, 1995], etc. One reason why studies of this sort are proper to research and have not yet become part of the body of knowledge is that there are not very many and, therefore, their findings are not solid enough. Hamlet [Hamlet, 1989] explains why these studies are so few and far between:

- It is difficult to compare the techniques, as they do not have a solid theoretical foundation.
- It is difficult to determine which variables are of interest.

Nonetheless, these are precisely the two points on which the studies can shed light. In other words, they can help to investigate and improve the theoretical basis underlying the techniques, making it possible to establish uniform criteria for comparison (for which there is a need [Frankl and Weyuker, 1991]). The second point addresses the problem of subjectivity in technique assessment. Assessment subjectivity occurs at two levels:

1. Different metrics can be chosen to measure an aspect for assessment.
2. Different people can consider different aspects as important.

The first level is dangerous when reflecting the findings of a study, although it can be remedied relatively easily by clearly explaining the metrics used to measure the aspect in question. The second is the level that is likely to pose more problems as regards technique comparison, as the term *better* can mean *less costly to use* for one person, whereas it can mean *more efficient* for another [Ntafos, 1988], [Weiss, 1990]. Some people [Ntafos, 1988] consider some techniques to be preferable to others because they are easy to use and have tools that automate part of the work rather than because they are effective or efficient. This means that all the potentially interesting attributes should be assessed for each technique, that is, they should include all the possible meanings of the word *better* for different developers.

What are the best-suited (best) techniques for preparing the set of test cases to be used to evaluate a given system aspect in a particular situation? It follows from the above that there is no easy answer to this question. Indeed, Bertolino [Bertolino, 2001] claims that this remains an open issue. One sure thing is that while there are comparative studies between techniques, there are no studies that examine the conditions of applicability of a technique at length or assess the relevant attributes for each technique. Additionally, existing studies show contradictory results [Rowland and Zuyuan, 1989].

Although the question of which techniques are best suited for generating the set of test cases for testing a given system appears to pose significant challenges, it is a question

often faced by developers, especially practitioners in charge of the testing process. How is it answered today? Neither systematically, nor following pre-established guidelines, of course. If we consult different project histories and reckon up, we find that of all the existing testing techniques, many are never even considered for use and others are used over again in different projects without even examining, after use, whether or not they were really useful. The decisions made by developers are not so much haphazard as limited insofar as their knowledge of the techniques is. There are two main reasons why developers do not make good choices. Both refer to their knowledge about existing testing techniques and their properties [Vegas, 2001]:

- The information available about the techniques is normally distributed across different sources of information (books, articles and even people). This means that developers do not have an overall idea of what techniques are available and of all the information of interest about each testing technique.
- There is no access to the pragmatic information concerning each testing technique unless they have used it before. Developers do not tend to share the knowledge they acquire by using a testing technique with others. This means that they miss out on the chance of learning about the experiences of others.

**This piece of research focuses on identifying relevant information for selecting test cases** so as to overcome the problem of selecting testing techniques for a software project.

## 1.4 Importance of the Problem

Several authors have generically addressed the importance of developers having access to information about testing techniques so that they can decide which ones they should use in a given context and the lack of such information. Thus, Kamsties and Lott [Kamsties and Lott, 1995] underline the need for evidence to help developers to decide which fault detection technique to apply under given circumstances. Wong *et al.* [Wong *et al.*, 1995] speak of the need to know the costs and benefits of applying different testing techniques. Shimeall and Leveson [Shimeall and Leveson, 1988] remark on how useful it would be to know to what extent two testing techniques are complementary or redundant. Again, Bertolino [Bertolino, 2001] stresses the importance of the problem of identifying the best technique under given circumstances. On the other hand, Howden [Howden, 1978] refers to the fact that a very common testing problem is the lack of well-defined techniques for selecting test cases.

Moreover, Beizer [Beizer, 1990] stresses the importance of proper selection, claiming that different software systems have different aspects or characteristics to be tested depending on how and for what purpose they were developed. On this basis, he claims that the selection of an unsuitable technique can lead to an inappropriate set of test cases, which will bring with it an inaccurate (if not erroneous) evaluation of the software aspect being tested.

Other authors put the emphasis on the terms in which the techniques should be described. Ntafos [Ntafos, 1988] notes that there is a lack of generally accepted models for measuring the cost and effectiveness of the techniques. Hamlet [Hamlet, 1989] discusses the importance of technique comparisons being based on their quality. Basili and Selby [Basili and Selby, 1987] state that it would be of interest to characterise the effectiveness of a technique with development factors. Lauterbach and Randall [Lauterbach and Randall, 1989] go a step further and speak of proper storage of the information gathered.

Apart from all the above, it can be added that if software systems development is to be an engineering discipline rather than a craft, a formalised basis is required upon which to make the decision of which technique to apply rather than this being done by mere intuition. Rombach [Rombach, 1992] claims that both the software processes and the principles, techniques and tools used in systems construction are not generally applicable and, therefore, their limitations should be investigated as far as project contexts are concerned.

Therefore, it can be said that **the problem addressed in this piece of research, the characterisation of testing techniques as an aid for selection, is an open question in Software Engineering (SE) and a matter of notable importance**, as stated by several authors.

## 1.5 Problem-Solving Approach

In solving the problem posed (determination of the information required to select testing techniques), there is one main objective: to help developers to choose the best suited testing techniques for each project during the testing technique selection process, in accordance with assessment criteria of their own or of the organisation for which they work. Indeed, the supposed benefit is that it will not be necessary to be acquainted with the technique to the extent of having used it or knowing how it is applied to make the selection.

It might appear, in principle, that it would suffice to identify which variables, criteria or simply information are useful for developers selecting the testing techniques to use in a given project. However, this is not enough if the objective is the proposed solution to really help developers. As mentioned above, one selection problem is where developers can find the information they require. So, the solution proposed here will not be confined to identifying useful selection criteria, but will also provide support for the selection process. Therefore, this dissertation also provides an infrastructure for storing the information identified and specified for each technique. The proposed solution is called **characterisation schema**.

It is important to stress that although the primary objective of the characterisation schema is to help software developers with selection, a schema of this sort will also benefit testing research. This is due to the fact it will help testing researchers to direct their utility studies about testing techniques. Thus, developers are provided with useful information about both the existing techniques and new techniques of interest for current software needs.

The characterisation schema should include the following: **a non-flat set of characteristics, in which the characteristics will be grouped according to the**

**referenced test element.** This set will be invariant for the techniques (that is, there will be no optional characteristics depending on the technique to be represented), although the value of more or fewer characteristics will be known depending on the information available about the technique. The schema should be able to fully describe the properties of the technique so that a decision can be made on whether or not to use the technique without having to be acquainted with or having used it before.

The schema will be instantiated once for each technique to be represented, as shown in Figure 1.1. Thus, it will be possible to build a repository that contains all the techniques of interest to a given organisation.

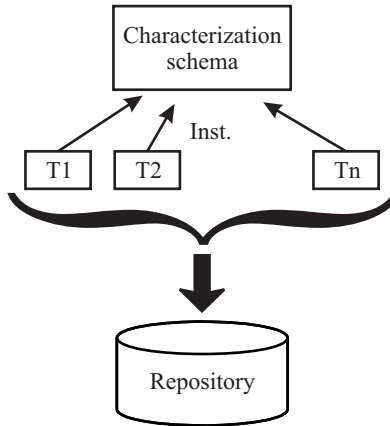


Figure 1.1: Instantiation of the characterisation schema.

If it is to be useful, this characterisation schema should have the following attributes:

- *It should account for all the information of potential interest.* The fact that the evaluation function of the soundness of a technique depends on the subject means that all the characteristics of potential interest to different developers have to be investigated. This means that the validity of the proposed solution will not be confined to a particular view of what is relevant information.
- *It should be independent of the test case selection strategy.* This aims to assure that the schema is generic enough to represent all the testing techniques now in existence, from techniques that are fully defined to techniques that represent undocumented knowledge.
- *It should be systematic.* It has to be systematic enough for two different people using the same evaluation function under the same circumstance to arrive at the same conclusions.
- *It should be efficient.* This assures that the technique can be selected using few resources.
- *It should be effective.* Test case selection using the schema has to lead to choices that are at least as satisfactory as if the selection were made without the schema

and should, preferably, improve such choices as far as meeting the objectives of the test are concerned.

- *It should be easy to use.* This will motivate consumers to use the schema.

Having defined the characteristics of the schema to be developed, the next question is how to do this. The procedure followed in this investigation was as follows. Two generative cycles were used to build a preliminary schema. This made it possible to build the schema by stages, where a different viewpoint was incorporated in each stage: first of the author of the dissertation, (hereinafter investigator), later of software developers and software testing researchers. The preliminary schema was then revised by experts in software testing, which led to its refinement. Finally, the revised schema was evaluated. This evaluation was run with respect to the investigator and potential schema users (students).

**An empirical approach has been adopted mainly because theoretical justification of the proposed solution was out of the question owing to the immaturity of the knowledge about testing.** Nonetheless, this immaturity could not be allowed to put off any attempt at problem solving, as developers are in need of an instrument of this sort. Therefore, in this dissertation an empirical approach has been adopted: a hypothesis, fruit of the perception of reality by the investigator, developers and testing researchers, is matched with its operating environment in the software project.

## 1.6 Organisation of the Document

This document, describing the development of the research, has been divided into four parts:

- The first part is an introduction to the research area and to the problem to be solved. It has been divided into three chapters. Chapter 1 is an introduction to the research area, which sets out both the problem to be addressed and the problem-solving approach. Chapter 2 explores work related to the planned research, putting it into context. Chapter 3 focuses more specifically on the stated problem, laying the foundations of the research to be conducted.
- The second part is focused on solving the problem posed. This part is composed of five chapters organised as follows. Chapter 4 details the process followed during the research to arrive at the solution. Chapter 5 and Chapter 6 include the perceptions of the investigator and of developers and researchers, respectively, of the problem. Each chapter presents a schema output as a result of these realities. Chapter 7 defines the preliminary schema, which is the fruit of the union of these realities and Chapter 8 presents a refinement of the preliminary schema, bearing in mind the opinion of testing experts.
- The third part focuses on the evaluation of the proposed solution and its implications. It has been organised as follows. Chapter 9 and Chapter 10 focus



on the evaluation of the proposed solution. Chapter 9 describes the empirical evaluation of the schema by the investigator and Chapter 10 details an experimental evaluation and subsequent schema refinement. Chapter 11 discusses the findings of the research and addresses future lines of research.

- The fourth part contains the bibliography and the necessary appendixes.



## Chapter 2

# State of the Art

### 2.1 Scope

As this chapter shows, there is no approach as yet that specifically deals with the problem of testing technique characterisation. It is clear from Chapter 1 that there have been no attempts at solving this problem, not because it is not of interest but because it is difficult to solve. However, there are approaches that are closely related to this subject of research. On the one hand, testing techniques have been studied comparatively, that is, there are investigations that endeavour, by a variety of means, to gather information that can be used to make distinctions between testing techniques from a technical/pragmatic viewpoint. On the other hand, there are studies that seek to characterise different software artefacts. This research does not deal with testing techniques, but does partly share the goal of this piece of research, which is to develop information frameworks that can be used to select a software artefact (techniques, products, etc.).

Four areas have been considered to be of interest for the research proposed here:

- *Testing*. There is a range of research in this area given over to comparing testing techniques. Several branches of work focusing on studies on different testing techniques are identified here.
- *Reuse*. There have been various efforts in this area to characterise artefacts for the purpose of reuse. Reuse usually implies the existence of a repository of reusable artefacts. An efficient selection of the artefacts best suited to a given situation within a software project depends on these artefacts having been characterised.
- *Software architectures*. Within the area of software architectures, there is a research line that aims to define the parameters necessary to be able to class the different architectural styles. The purpose of this classification is to enable developers to formally describe the architectural designs of software systems and create a catalogue <sup>1</sup> that stores information on styles.

---

<sup>1</sup>In this piece of research the word *catalogue* is used as a synonym of *repository*.

- *Technology selection.* This area encompasses different work related to the characterisation of techniques, methods and tools for software development. The main purpose is to help developers select the techniques, methods and tools they need to build a software system.

The relationship of each of the above areas with the research described here is explained below.

### 2.1.1 Testing

The testing area is an essential component of this piece of research, which deals with the problem of testing technique selection. Although nobody in the testing area has done any research into the problem of technique characterisation at the level of specificity presented here, there are, as mentioned above, a range of studies in this area that are directly related to the problem of testing technique selection. These studies can be divided into several groups:

- Studies on the relationships between structural techniques.
- Studies on data flow testing techniques.
- Comparisons between random and partition testing techniques.
- Comparisons between functional and structural testing techniques.
- Studies on mutation testing techniques.
- Studies on regression testing techniques.
- Studies on minimisation testing techniques.
- Other studies.

Each of these will be studied in more detail later.

### 2.1.2 Reuse

Reuse involves using artefacts that have been developed earlier to build a new software system instead of building this system from scratch. Reuse can take place at several levels within a project:

1. *Code.* An important point in relation to this idea is the development of catalogues (similar to those existing in other industries) that illustrate the main features of the software components and can be used by developers to select the components that are in their best interests.

Quite a lot of work has been done on module cataloguing and code repository construction. These approaches will be revised to find out whether any could solve or help to solve the problem of testing technique characterisation.

2. *Designs, requirements, etc.* None of the work in this area is related to characterisation, as this approach is based on the development of generic requirements and design documents for a specific product, supported by a component repository. Software development has still not reached the point of combining requirements, design and code taken from a repository of reusable artefacts. Therefore, there are no approaches related to requirements and/or design pattern characterisation could be of use for the purposes of this piece of research.
3. *Experiences.* There are several approaches related to this idea of reusing anything potentially of interest for future software projects. They focus on populating repositories with different artefacts apart from code. The problem of characterisation again arises in these approaches, and they will, therefore, be revised to find out whether any such solutions could be of interest.

### 2.1.3 Software Architectures

One of the key points when developing a software system is the design of the system architecture. Design is related to the decomposition of the system into computational objects and their interactions [Garlan *et al.*, 1995]. At present there is no formal notation for describing designs in SE, which means that the descriptions are informal and the names used sometimes lead to confusion. A clear example is the client-server organisation, a widely accepted term that can, however, be interpreted differently by different people.

Within the area of software architectures, the approaches whose purpose is to establish a classification of architectural styles will be examined. These approaches have two goals. Firstly, they aim to formalise the design descriptions so that anyone reading a design can interpret it unambiguously and, secondly, they endeavour to create a catalogue that provides information about the existing styles and aids developers with selection.

The problem of establishing a classification of architectural styles is of interest to the research presented here in that it is directly related to the generic problem of characterisation, since the development of a catalogue of architectural styles implies their characterisation.

### 2.1.4 Technology Selection

There are now a wide variety of techniques and tools to support software systems construction. Developers tend to find it difficult to select which tools or techniques to use in a given project, as there are no formal guidelines to support selection. Developers usually have to base their decisions on their own experience and knowledge or even on mere speculations by others.

This area aims to help developers to choose the techniques, methods and tools for use in a software project. Conceptually, it is very close to the problem at hand, as testing technique selection is a particular instance of the selection of techniques for use in a software development. For this reason, the work done in this area will be closely examined.

### 2.1.5 Classification of the Study Areas

Section 2.1 described the areas of interest for this research. However, the set of criteria for description differs from area to area. For the purpose of these criteria, the four areas will be grouped as follows:

- *Testing area.* This encompasses studies concerned with the similarities and/or differences between different testing techniques.
- *Characterisation area.* This includes the areas of reuse, software architectures and technology selection. These areas have been grouped as one, because they all share the same goal of characterising software artefacts, irrespective of whether this is for the purpose of selection (as in the case of reuse or technologies) or description (as in the case of software architectures).

The reason why two different sets of criteria are established is that there are points of interest within the characterisation area, like, for example, the process followed to arrive at the characterisation, which there are not in the testing area, as no such process exists.

## 2.2 Testing Area

### 2.2.1 Descriptive Criteria

The following criteria were considered of interest for the testing area: study type, the aspect(s) of the techniques analysed by the comparative study, the techniques studied, as well as the generality or exhaustiveness of the study. Each of these descriptive criteria is explained in detail below.

- **Study type.** There are several different sorts of possible studies for ascertaining the uses to which different testing techniques are put. There are different sources for the recommendations and suggestions on the use of testing techniques: theoretical studies, simulations and empirical studies. This criterion reflects the method followed to compare the different techniques. This information is of great interest. On the one hand, it can be used to ascertain how powerful, meaning how generalisable and applicable, the findings are in practice. On the other hand, the inexistence of theoretical studies is an indicator of how developed the theoretical basis of testing techniques is. The possible values for this criterion are:
  - *Theoretical.* The studies analyse the chosen set of features bearing in mind the scientific basis of the techniques.
  - *Empirical.* The studies analyse the results obtained when applying (using) the techniques. It is interesting to underline that not all empirical studies are the same. Some of these studies are based on the design of experiments

and statistical analysis. The goal here is not to study the validity of a study on the basis of the type of statistical techniques used. Therefore, for the purposes of this research, they will all be considered the same, irrespective of how formal they are.

- *Simulation.* These studies are midway between theoretical and empirical studies. They use the theoretical basis of the technique to derive (usually represented as mathematical formulas) a series of numerical values that should be obtained empirically if the technique were to be used. In other words, they simulate situations in which the technique is used.

- **Aspects.** Different studies focus on different aspects of the testing techniques. The purpose of this criterion is to take into account the aspects on which the study has focused to find out which are of interest to the scientific community, as well as to ascertain the breadth of the study with respect to the set of possible features of the techniques that have been analysed.

The possible values for this criterion throughout this study are both the different aspects and the relationships between these aspects. The different aspects are listed below, although their interrelationships are not taken into account.

- *Inclusion.*
- *Selectivity.*
- *Probability of one fault.*
- *Narrowing.*
- *Cover.*
- *Appropriate covers.*
- *Partitioning.*
- *Appropriate partitioning.*
- *Number of faults.*
- *Fault rate.*
- *Size* (number of generated cases).
- *Coverage.*
- *Visibility* (number of times a set of test cases detects a fault).
- *Probability of there being faults not detected by the technique.*
- *Cost of not detecting faults.*
- *Impact of the use of groups.*
- *Time* (generation, execution, computational).
- *Fault type.*

- *Software type.*
- *Skills.*
- *Number of killed mutants.*
- *Number of generated mutants.*
- *Accuracy.*
- *Memory.*
- *Reliability.*
- *Generality.*
- *Accountability.*

The possible values of this criterion are explained in Appendix B.2.

- **Techniques.** This criterion contemplates the coverage of the study, expressing the types of techniques that have been taken into account by the study. The values found for this criterion are the existing techniques:

- *Control flow testing.*
- *Data flow testing.*
- *Expressions testing.*
- *Data functions testing.*
- *Partition testing.*
- *Random testing.*
- *Structural testing.*
- *Mutation testing.*
- *Regression testing.*
- *Minimisation testing.*
- *Prioritisation testing.*
- *Operational testing.*
- *Required elements testing.*
- *Debug testing.*

The values for this criterion, and other existing values not addressed by any study, are explained in Appendix B.1.

- **Exhaustiveness.** The families of techniques mentioned in the above criterion have variants. The idea of introducing this criterion is to specify whether the study is exhaustive for all the techniques of which the family is composed. For this purpose,



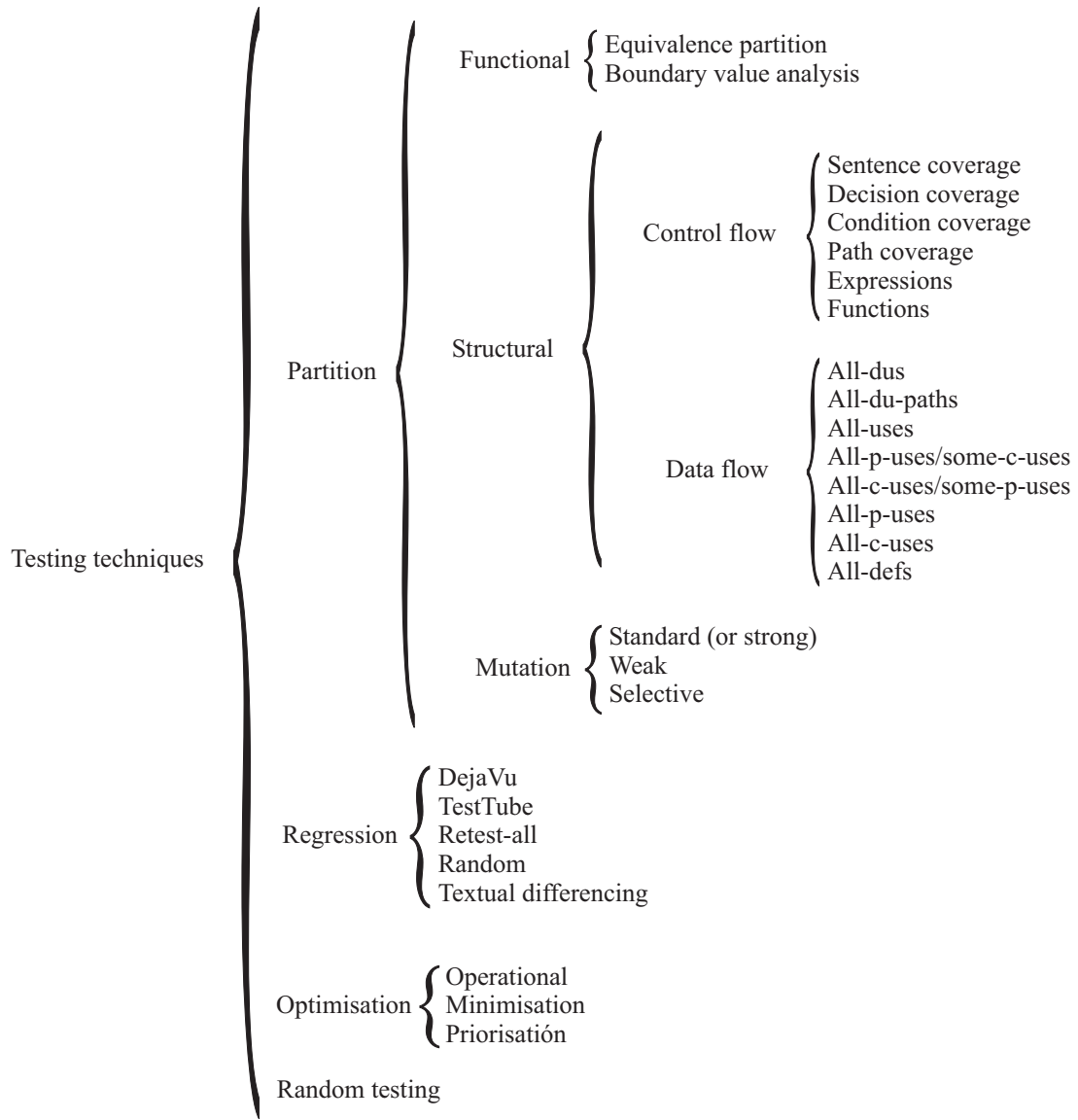


Figure 2.1: Testing techniques by families.

the composition of the families of testing techniques is illustrated in Figure 2.1 below, to which this criterion refers.

The possible values for this criterion are:

- *No*. One or several techniques of the family are studied, but it is not covered in its entirety.
- *Family*. The study is run for all the techniques of the family.
- *Yes*. The study is valid for all testing techniques.

### 2.2.2 Related Work in the Testing Area

The studies in the testing area that are pertinent to this piece of research are presented below. They have been grouped around a series of studies or lines of research to ease their study.

The basic concepts and terms are in Appendix B.1.

#### 2.2.2.1 Studies on Relationships between Structural Techniques

The goal of this research line is to establish a series of relationships between different testing techniques. This family encompasses seven studies by Rapps and Weyuker [Rapps and Weyuker, 1982], [Rapps and Weyuker, 1985], Weiser *et al.* [Weiser *et al.*, 1985], Clarke *et al.* [Clarke *et al.*, 1985], [Clarke *et al.*, 1989], Ntafos [Ntafos, 1988], Zeil [Zeil, 1988] and Frankl and Weyuker [Frankl and Weyuker, 1991], [Frankl and Weyuker, 1993a], [Frankl and Weyuker, 1993b].

They mainly focus on structural techniques, especially on data flow and control flow testing techniques. Although one study (Weiser *et al.*) does include other structural techniques, like data functions or expressions testing, and one of the studies by Frankl and Weyuker is generic for all partition testing techniques. The goal of these studies is to order the techniques on the basis of defined relationships.

Generally, most of these studies are theoretical, which means that their findings are more reliable for ratifying claims about a technique. However, they are also much harder to perform insofar as they require a much more solid and formalised theoretical foundation. Experimentation is used, as in Weiser *et al.*'s study, when the theory does not explain the problem at issue.

As mentioned above, this family of studies focuses on the relationships between the different techniques for the purposes of establishing a partial order. Most of the studies (Rapps and Weyuker, Weiser, Clarke *et al.* and Ntafos) focus on studying the relationship of inclusion. Interestingly, these are the oldest studies, which explains why they focus on a relationship that has later been rated as not being very useful as it is not very significant. The latest studies (chronologically speaking) were performed by Zeil and Frankl and Weyuker, and, contrariwise, set out to search for potentially more useful relationships like selectivity, narrowing, covering or partitioning. The studies by Frankl and Weyuker attempt to establish connections between these relationships and the expected number of faults that the technique will discover or the probability of the technique finding at least one fault.

Only one of the inclusion studies (by Ntafos) is exhaustive with respect to the families it studies, the others are not. This is because these are a succession of studies, each of which endeavours to complete the preceding one by adding more techniques to the study. The others are exhaustive with respect to the family, except the second study by Frankl and Weyuker, which is a refinement or particularisation of the first.

Table 2.1 shows the differences between the different studies of each family, which are described in more detail in Appendix C.1.

By way of a summary, the main problems with the family of studies on relationships

STUDY	TYPE	ASPECTS	TECHNIQUES	EXHAUS
Rapps & Weyuker	Theoretical	Inclusion	Control flow Data flow	No
Weiser, Gannon & McMullin	Theoretical	Inclusion	Data flow Control flow Expressions Functions	No
	Empirical	Inclusion	Data flow Expressions	No
Clarke, Podgurski, Richardson & Zeil	Theoretical	Inclusion	Data flow Control flow	No
Ntafos	Theoretical	Inclusion	Data flow Control flow	Family
Zeil	Theoretical	Selectivity	Data flow Control flow	Family
Frankl & Weyuker	Theoretical	Prob. 1 fault/narrowing Prob. 1 fault/covers Prob. 1 fault/appropriate covers Prob. 1 fault/partitioning Prob. 1 fault/appropriate part.	Partition	Family
Frankl & Weyuker	Empirical	No. faults/appropriate covers	Data flow Control flow	No

Table 2.1: Value of the criteria of comparison for studies of relationships between structural techniques.

between structural techniques can be said to be that:

- They are not exhaustive studies in the universe of testing techniques, although some are for the set of techniques of which the family is composed.
- The aspects that they study are not usually oriented to pragmatic usefulness for the developer, as they do not establish comparisons on the basis of something that is potentially useful from the pragmatic viewpoint.
- In any case, and even if the aspect under investigation is of interest, it is usually insufficient for developers, as it is only one of several aspects (which would give a more complete idea of the relationship between the techniques).

### 2.2.2.2 Studies on Data Flow Testing Techniques

This series of studies more or less follows on from the studies discussed in Section 2.2.2.1, dealing in this case with data flow testing techniques and seeking to observe the differences between these techniques to better understand their application domain. These studies differ from the others in that they directly study certain aspects of interest rather than the relationships between techniques.

The studies by Weyuker [Weyuker, 1988], [Weyuker, 1990], Frankl and Weiss [Frankl and Weiss, 1991a], [Frankl and Weiss, 1991b], [Frankl and Weiss, 1993], Bieman and Schultz [Bieman and Schultz, 1992], Hutchins *et al.* [Hutchins *et al.*, 1994] and Frankl and Iakounenko [Frankl and Iakounenko, 1998] have been examined. The study by Bieman and Schultz ratifies some of the results obtained by Weyuker. Likewise, the studies by Frankl and Weyuker and Frankl and Iakounenko are complementary, as they ratify each other.

The main characteristics of these studies are that they are all empirical, which means that they are not as powerful as the theoretical investigations, because their results are not as easy to generalise.

Rather than addressing the relationships between different techniques, they focus on more pragmatic aspects, such as the number of generated cases (Weyuker and Bieman and Schultz), coverage (Frankl and Weiss, Hutchins *et al.* and Frankl and Iakounenko), the rate of fault detection (Hutchins *et al.* and Frankl and Iakounenko) or the probability of finding at least one fault (Frankl and Weiss). However, the number of aspects addressed is still insufficient for selection purposes, and some of these aspects (probability of finding at least one fault) are not useful for developers.

They focus on a very small group of techniques, and none of them are exhaustive at the family level. Some study one technique from one family (as is the case of Bieman and Schultz), others focus on several techniques from a family (Weyuker) and others address one technique from several families (Frankl and Weiss, Hutchins *et al.*, and Frankl and Iakounenko).

Table 2.2 shows the differences between the different studies of each family, which are described in Appendix C.2.

STUDY	TYPE	ASPECTS	TECHNIQUES	EXHAUS
Weyuker	Empirical	Size	Data flow	No
Frankl & Weiss	Empirical	Prob. 1 fault Size/Prob. 1 fault Cover/Prob. 1 fault	Data flow Control flow Random	No
Bieman & Schultz	Empirical	Size	Data flow	No
Hutchins, Foster, Goradia & Ostrand	Empirical	Fault rate Size Rate/Coverage Size/Coverage	Data flow Control flow	No
Frankl & Iakounenko	Empirical	No. faults No. faults/coverage	Data flow Control flow Random	No

Table 2.2: Value of the criteria of comparison for the studies on data flow testing techniques.

The findings after examining this family were:

- As they are empirical studies, their results are more difficult to generalise, although there are studies that help to generalise or ratify the results of others.

- They usually address one or too few aspects for the selection to be effective.
- The studies are not exhaustive either separately or combined.

### 2.2.2.3 Comparisons between Random and Partition Testing Techniques

It has always been thought that the random testing technique is one of the worst (if not the worst) testing technique with regard to the number of faults detected. This line of research presents a series of studies whose authors have examined whether or not this statement is at all founded and, if so, under what conditions it is true.

Within this line of research, the studies by Duran and Ntafos [Duran and Ntafos, 1984], Hamlet [Hamlet, 1987], Hamlet and Taylor [Hamlet and Taylor, 1988], [Hamlet and Taylor, 1990], Weyuker and Jeng [Weyuker and Jeng, 1991], Chen and Yu [Chen and Yu, 1994], [Chen and Yu, 1996] and Ntafos [Ntafos, 1998] have been taken into account.

The research by Duran and Ntafos was the pioneer of this series of studies. Surprisingly, they demonstrated that the random testing technique was better than partition testing techniques with regard to the number of faults detected. On this basis, the following studies endeavour to analyse these results in an attempt to define in what situations one technique behaves better than the others do. All these studies are an extension or continuation of each other. For example, the studies by Weyuker and Jeng and by Hamlet follow on from the study by Duran and Ntafos. The study by Hamlet and Taylor is a continuation of the study by Hamlet and the studies by Chen and Yu follow on from work by Weyuker and Jeng.

The studies performed within this line range from theoretical (Hamlet, Hamlet and Taylor, Weyuker and Jeng and Chen and Yu), through simulations (Duran and Ntafos, Ntafos), to empirical studies (Duran and Ntafos and Chen and Yu). They all focus on the study of partition and random testing techniques, although part of one (namely, Duran and Ntafos) refers solely to the random testing technique. They are all exhaustive at family level, except the empirical studies, which are not exhaustive.

With regard to the aspects studied, most focus on the probability of detecting at least one fault (Duran and Ntafos, Hamlet and Taylor, Weyuker and Jeng or Chen and Yu), although other aspects of interest are the number of faults detected (Duran and Ntafos), the probability of there being undetected faults at the end of testing, the number of times a set finds a fault (Duran and Ntafos) or the monetary cost of not finding a fault (Ntafos).

Table 2.3 shows the differences between the different studies of each family, which are described in Appendix C.3.

The findings after studying this family are:

- They focus on a very few aspects, which means that the understanding gained of the techniques under consideration is limited.
- Although they focus on the study of two families of techniques, the results cannot be extrapolated to any testing technique.

STUDY	TYPE	ASPECTS	TECHNIQUES	EXHAUS
Duran & Ntafos	Simulation	No. Faults Prob. 1 fault	Random Partition	Family
	Empirical	Visibility	Random	No
	Empirical	Coverage	Random	No
Hamlet	Theoretical	Prob. of undetected faults	Random Partition	Family
Hamlet & Taylor	Theoretical	Prob. 1 fault	Random	Family
		Prob. of undetected faults	Partition	
Weyuker & Jeng	Theoretical	Prob. 1 fault	Random Partition	Family
Chen & Yu	Theoretical	Prob. 1 fault	Random Partition	No
Chen & Yu	Empirical	No. faults	Random Partition	Family
Ntafos	Simulation	Cost of not detecting faults	Random Partition	Family

Table 2.3: Value of the criteria of comparison for comparisons between random and partition testing techniques.

- Some of the aspects examined, such as the probability of finding at least one fault, are not of interest to developers.

However, these studies have established when it is preferable to use the random testing technique rather than the partition testing technique.

#### 2.2.2.4 Comparisons between Functional and Structural Techniques

The four studies presented in this section correspond to a series of empirical studies whose authors investigate the differences between functional and structural testing techniques (including code analysis, which will not be taken into account for the purposes of this study) with regard to a series of development issues.

The studies that will be taken into account are: Myers [Myers, 1978], Basili and Selby [Basili and Selby, 1985], [Selby and Basili, 1984], [Basili and Selby, 1987], Kamsties and Lott [Kamsties and Lott, 1995] and Wood *et al.* [Wood *et al.*, 1997].

The main difference between these studies are the aspects that they address (they vary slightly from one to another) and the techniques they study. In the study by Myers, the subjects are left to choose their preferred technique or combination of techniques from each family, whereas they are obliged to use a given technique in the others. Although they always use the same functional testing technique, the structural techniques used differ. Obviously, as they focus on a single technique from each family, the studies are not exhaustive.

The studies by Kamsties and Lott and Wood *et al.* are replications of the experiment by Basili and Selby, which, in turn, follows on from the study by Myers.

The features they cover vary and include the number of faults detected, the fault detection rate, the type of faults detected and many others, as shown in Table 2.4. Table 2.4 shows the differences between the different studies of each family, which are described in Appendix C.4.

STUDY	TYPE	ASPECTS	TECHNIQUES	EXHAUS
Myers	Empirical	Groups No. faults Time (gen+exec)	Functional Structural	No
Basili & Selby	Empirical	No. faults Fault rate Fault type Computational time Software type Experience Coverage	Functional Control flow	No
Kamsties & Lott	Empirical	No. faults Fault rate Skills Fault type	Functional Control flow	No
Wood, Roper, Brooks & Miller	Empirical	No. faults Fault rate Software type Groups	Functional Control flow	No

Table 2.4: Value of the criteria of comparison for functional and structural techniques.

The findings after studying this family are:

- The aspects on which they focus are all useful from the pragmatic viewpoint and of interest for developers. Thus, for example, these are the only studies that consider human factors when applying testing techniques.
- They are all empirical studies; however, the fact that they complement each other means that, on the one hand, they ratify each other and, on the other, their findings can be generalised.
- They focus on very few techniques, although this means that their results are more convincing for the techniques they do study.

### 2.2.2.5 Studies on Mutation Testing Techniques

This family of studies focuses on the analysis of different mutation testing techniques. Whereas the studies by Offut and Lee [Offut and Lee, 1991], [Offut and Lee, 1994] and by Offut *et al.* [Offut *et al.*, 1993], [Offut *et al.*, 1996] focus on the study of other types of mutation testing that are less costly than the traditional technique, the research by Frankl *et al.* [Frankl *et al.*, 1994], [Frankl *et al.*, 1997] and by Wong and Mathur

[Wong and Mathur, 1995] compare mutation testing with other white box techniques (data flow and control flow testing).

All the studies are empirical, which means that they are not as generic or extrapolable as if they were theoretical. Indeed, none of the studies are exhaustive.

Generally, these studies aim to find out the costs and benefits of using the different techniques. However, the aspects that they study have the drawback of not being useful for developers. Indeed, they concentrate on the number of dead mutants (Offut and Lee, Offut *et al.* and Frank *et al.*), the number of test cases generated (Offut and Lee), the number of sentences executed (Offut and Lee), the number of mutants generated (Offut *et al.*) and the probability of at least one fault being discovered (Frankl *et al.*).

Table 2.5 shows the differences between the different studies of each family, which are described in Appendix C.5.

STUDY	TYPE	ASPECTS	TECHNIQUES	EXHAUS
Offut & Lee	Empirical	No. killed mutants Size Coverage	Mutation	No
Offut, Rothermel & Zapf	Empirical	No. generated mutants No. killed mutants	Mutation	No
Frankl, Weiss & Hu	Empirical	Prob. 1 fault No. killed mutants Coverage	Data flow Mutation	No
Wong & Mathur	Empirical	Visibility	Mutation Control flow	No

Table 2.5: Value of the criteria of comparison for the studies on mutation testing techniques.

For the study of this family, the findings are:

- It follows from the fact that all the studies are empirical that the theoretical foundation of mutation testing is not very solid.
- Furthermore, the studies are not related to each other and are, therefore, not complementary, meaning that the family as a whole is not very powerful.
- Finally, the aspects on which they focus are not of interest for developers.

#### 2.2.2.6 Studies on Regression Testing Techniques

Regression testing is fundamental to software maintenance, as it must assure that the changes made to the software have been made correctly and that, also, they have not affected parts of the software that were in order. Moreover, regression testing should take up as little time as possible in the sense of executing as few test cases as possible. This line of research covers the different studies on regression testing techniques.



Most of the studies on regression testing techniques are empirical, except the theoretical studies by Rothermel and Harrold [Rothermel and Harrold, 1994] and [Rothermel and Harrold, 1996] or the combined study by Roseblum and Weyuker [Roseblum and Weyuker, 1997a] and [Roseblum and Weyuker, 1997b].

Almost none of the studies are exhaustive, except the one by Elbaum *et al.* [Elbaum *et al.*, 2000] and the theoretical study by Rothermel and Harrold, which are exhaustive at family level.

They are quite homogeneous, insofar as they all focus on more or less the same aspects. Thus, the empirical studies by Rothermel and Harrold [Rothermel and Harrold, 1997a], [Rothermel and Harrold, 1997b], [Rothermel and Harrold, 1998], Roseblum and Rothermel [Roseblum and Rothermel, 1997], [Bible *et al.*, 1999], Vokolos and Frankl [Vokolos and Frankl, 1998] and Roseblum and Weyuker focus on the percentage of cases selected to apply the regression testing technique, as well as the test case selection and execution time. Graves *et al.* [Graves *et al.*, 1998] and Kim *et al.* [Kim *et al.*, 2000] focus on the percentage of selected cases and the number of faults detected. Rothermel *et al.* [Rothermel *et al.*, 1999] and Elbaum *et al.* look at the number of faults detected. Finally, Wong *et al.* consider not only size, but also accuracy and memory. Although there are not many, the aspects are generally pragmatically useful.

Table 2.6 shows the differences between the different studies of each family, which are described in Appendix C.6.

The findings for this family are:

- The studies are specific to certain techniques of the family of regression testing techniques, which means that their findings are not extendible to other techniques.
- The aspects of interest vary from one study to another, making it difficult to draw conclusions even within the same family of techniques.
- However, the fact that there are various studies that examine the same aspects across different techniques means that they are complementary and generate a more solid, albeit small body of knowledge.

### 2.2.2.7 Studies on Minimisation Testing Techniques

The purpose of this family of studies is to analyse a technique known as minimisation testing on sets of test cases generated by white box techniques (data flow and control flow testing).

Both studies are practically the same. They are both empirical studies that compare a minimisation testing technique with a white box technique, which means that they are not exhaustive. Both study the number of generated cases and the number of faults detected. The only difference between them is that the study by Wong *et al.* [Wong *et al.*, 1995] and [Wong *et al.*, 1999] uses a data flow testing technique for comparison, whereas the one by Rothermel *et al.* [Rothermel *et al.*, 1998] uses a control flow testing technique. This difference

STUDY	TYPE	ASPECTS	TECHNIQUES	EXHAUS
Rothermel & Harrold	Empirical	Size Time	Regression	No
Rosembaum & Rothermel	Empirical	Time Size	Regression	No
Graves, Harrold, Kim, Porter & Rothermel	Empirical	Size No. faults	Regression	No
Vokolos & Frankl	Empirical	Size Time	Regression	No
Wong, Horgan, London and Agraval	Empirical	Size Accuracy Memory	Regression Minimisation Prioritisation	No
Rothermel, Untch, Chu & Harrold	Empirical	Fault rate	Prioritisation	No
Elbaum, Mailshevsky & Rothermel	Empirical	Fault rate	Regression	Family
Kim, Port & Rothermel	Empirical	Size No. faults	Regression	No
Rothermel & Harrold	Theoretical	Inclusion Accuracy Efficiency Generality Accountability	Regression	No
Rosembaum & Weyuker	Theoretical Empirical	Size Time	Regression	No

Table 2.6: Value of the criteria of comparison for studies on regression testing techniques.

can be put down to the fact that Rothermel *et al.* seek to generalise the results obtained by Wong *et al.* for other techniques.

Table 2.7 shows the differences between the different studies of each family, which are described in Appendix C.7.

STUDY	TYPE	ASPECTS	TECHNIQUES	EXHAUS
Wong, Horgan, London & Mathur	Empirical	Size/Cover No. faults/Cover	Minimisation Data flow	No
Rothermel, Harrold, Ostril & Hong	Empirical	No. faults Size	Minimisation Control flow	No

Table 2.7: Value of the criteria of comparison for studies on minimisation testing techniques.

The findings for these two studies are:

- They focus on a small number of features, which are insufficient for selection purposes.
- They are not exhaustive, but examine a single technique.
- They are highly interesting, as they seek to generalise the results. However, they are not considered sufficient, as there are only two.

### 2.2.2.8 Other Studies

Below we present other studies that could not be classed in any of the above groups. Unlike the above, these studies do not belong to a line of research pursuing specific objectives, but are one-off contributions to research in the testing field. This is perhaps their weakest point.

Five studies have been grouped within this family. These are the studies by Frankl *et al.* [Frankl *et al.*, 1998], [Hamlet *et al.*, 1997], Ntafos [Ntafos, 1981], [Ntafos, 1984], Frankl and Deng [Frankl and Deng, 2000] and Lauterbach and Randall [Lauterbach and Randall, 1989]. Most of the studies are empirical and not exhaustive, except the one by Frankl *et al.*, which is theoretical and also exhaustive at family level. The techniques they examine vary from one study to another, as do the aspects on which the focus, which are shown in Table 2.8, illustrating the differences between the different studies of each family. These studies are described in more detail in Appendix C.8.

STUDY	TYPE	ASPECTS	TECHNIQUES	EXHAUS
Frankl, Hamlet, Littlewood & Stringini	Theoretical	Reliability	Debug testing Operational	Family
Ntafos	Empirical	No. killed mutants	Required elements Control flow Random	No
Frankl & Deng	Empirical	Reliability	Control flow Data flow Operational	No
Lauterbach & Randall	Empirical	No. faults Effort of application Human influence Coverage	Random Data flow Control flow	No

Table 2.8: Value of the criteria of comparison for other studies on testing techniques.

These studies are plagued by all the problems that have been detected earlier. They add very little (or nothing) to the knowledge on testing techniques, because none of them belong to a line of research or set of studies.

### 2.2.3 Conclusions on the Testing Area

The shortcomings found in the studies that have been studied can be summarised as follows. Table 2.9, Table 2.10 and Table 2.11 give an overview of the values of the criteria of comparison for all the studies examined in the testing area.

After studying the testing area, it is found, generically, to have the following shortcomings:

- The studies are **not exhaustive** as regards the universe of testing techniques, and only a few cases are exhaustive with regard to the family or families that they study. This means that the knowledge of the different techniques under the same conditions is not very robust and well founded.
- If there were enough studies to create a solid body of knowledge on testing techniques, this would make up for the non-exhaustiveness of the studies. However, many of the **results of the different studies cannot be compared or combined** satisfactorily even for the same technique or family. This is mainly because they use different metrics to define the same aspects under study.
- Even though the metrics used for evaluation are correct, they are not always useful for developers, mainly because they do **not contemplate the pragmatic usefulness of the technique**.

## 2.3 Area of Characterisation

### 2.3.1 Descriptive Criteria

For the area of characterisation, we considered the following issues to be of interest: the object that is to be characterised, how the object is characterised in the approach, the process followed to get the characterisation schema, the aspects of the object that have been taken into account for characterisation, how the schema is to be completed and, finally, the possibility of the schema evolving over time. These are explained in detail below.

- **Object.** Different elements have been characterised in SE. This criterion indicates the element characterised in the approach examined. A revision of the values for this criterion gives an overall idea of the objects that have been of interest for characterisation purposes. And, vice versa, one can get an idea of the objects that have not been characterised by observing this criterion.

The values that have been observed for this criterion in the approaches studied are:

- *Software modules.* Simply code.
- *Architectural styles.* These refer to all the available styles from which one can choose the styles that will be used to build the software design.

STUDY	TYPE	ASPECTS	TECHNIQUES	EXHAUS
Rapps & Weyuker	Theoretical	Inclusion	Control flow Data flow	No
Weiser, Gannon & McMullin	Theoretical	Inclusion	Data flow Control flow Expressions Functions	No
	Empirical	Inclusion	Data flow Expressions	No
Clarke, Podgurski, Richardson & Zeil	Theoretical	Inclusion	Data flow Control flow	No
Ntafos	Theoretical	Inclusion	Data flow Control flow	Family
Zeil	Theoretical	Selectivity	Data flow Control flow	Family
Frankl & Weyuker	Theoretical	Prob. 1 fault/narrowing Prob. 1 fault/covers Prob. 1 fault/appropriate covers Prob. 1 fault/partitioning Prob. 1 fault/appropriate part.	Partition	Family
Frankl & Weyuker	Empirical	No. faults/appropriate covers	Data flow Control flow	No
Weyuker	Empirical	Size	Data flow	No
Frankl & Weiss	Empirical	Prob. 1 fault Size/Prob. 1 fault Cover/Prob. 1 fault	Data flow Control flow Random	No
Bieman & Schultz	Empirical	Size	Data flow	No
Hutchins, Foster, Goradia & Ostrand	Empirical	Fault rate Size Rate/Coverage Size/Coverage	Data flow Control flow	No
Frankl & Iakounenko	Empirical	No. faults No. faults/coverage	Data flow Control flow Random	No
Duran & Ntafos	Simulation	No. Faults Prob. 1 fault	Random Partition	Family
	Empirical	Visibility	Random	No
	Empirical	Coverage	Random	No
Hamlet	Theoretical	Prob. of undetected faults	Random Partition	Family
Hamlet & Taylor	Theoretical	Prob. 1 fault Prob. of undetected faults	Random Partition	Family
Weyuker & Jeng	Theoretical	Prob. 1 fault	Random Partition	Family
Chen & Yu	Theoretical	Prob. 1 fault	Random Partition	No?

Table 2.9: Values of the criteria of comparison in the testing area (1/3).

STUDY	TYPE	ASPECTS	TECHNIQUES	EXHAUS
Chen & Yu	Empirical	No. faults	Random Partition	Family
Ntafos	Simulation	Cost of not detecting faults	Random Partition	Family
Myers	Empirical	Groups No. faults Time (gen+exec)	Functional Structural	No
Basili & Selby	Empirical	No. faults Fault rate Fault type Computational time Software type Experience Coverage	Functional Control flow	No
Kamsties & Lott	Empirical	No. faults Fault rate Skills Fault type	Functional Control flow	No
Wood, Roper, Brooks & Miller	Empirical	No. faults Fault rate Software type Groups	Functional Control flow	No
Offut & Lee	Empirical	No. killed mutants Size Coverage	Mutation	No
Offut, Rothermel & Zapf	Empirical	No. generated mutants No. killed mutants	Mutation	No
Frankl, Weiss & Hu	Empirical	Prob. 1 fault No. killed mutants Coverage	Data flow Mutation	No
Wong & Mathur	Empirical	Visibility	Mutation Control flow	No
Rothermel & Harrold	Empirical	Size Time	Regression	No
Roseblum & Rothermel	Empirical	Time Size	Regression	No
Graves, Harrold, Kim, Porter & Rothermel	Empirical	Size No. faults	Regression	No
Vokolos & Frankl	Empirical	Size Time	Regression	No
Wong, Horgan, London and Agraval	Empirical	Size Accuracy Memory	Regression Minimisation Prioritisation	No
Rothermel, Untch, Chu & Harrold	Empirical	Fault rate	Prioritisation	No

Table 2.10: Values of the criteria of comparison in the testing area (2/3).

STUDY	TYPE	ASPECTS	TECHNIQUES	EXHAUS
Elbaum, Mailshesky & Rothermel	Empirical	Fault rate	Regression	Family
Kim, Port & Rothermel	Empirical	Size No. faults	Regression	No
Rothermel & Harrold	Theoretical	Inclusion Accuracy Efficiency Generality Accountability	Regression	No
Roseblum & Weyuker	Theoretical Empirical	Size Time	Regression	No
Wong, Horgan, London & Mathur	Empirical	Size/Coverage No. faults/Coverage	Minimisation Data flow	No
Rothermel, Harrold, Ostril & Hong	Empirical	No. faults Size	Minimisation Control flow	No
Frankl, Hamlet, Littlewood & Stringini	Theoretical	Reliability	Debug testing Operational	Family
Ntafos	Empirical	No. killed mutants	Required elements Control flow Random	No
Frankl & Deng	Empirical	Reliability	Control flow Data flow Operational	No
Lauterbach & Randall	Empirical	No. faults Effort of application Human influence Coverage	Random Data flow Control flow	No

Table 2.11: Values of the criteria of comparison in the testing area (3/3).

- *Software technologies.* These encompass methods, techniques or tools useful for developing software.
  - *Requirements acquisition methods.* These refer to the methods used during the problem analysis phase of development.
  - *Development problems.* These refer to all the problems that can emerge during software development and their solution.
  - *Any software development element.* This refers to both software products (designs, requirements, testing plans, etc.) and processes (analysis, design, etc.) and techniques.
- **Type of characterisation.** This criterion studies the characterisation proposed by the research. The possible values for this criterion are:

- *Metaschema*. This contains a series of attributes that characterise the element in question. However, it is generic, which means that two things have to be done before it is used: (1) (some of) the attributes it contains must be refined for the specific element to be instantiated and (2) it has to be instantiated.
  - *Schema*. This is a specific set of attributes that characterise the element in question. Although it does not have to be refined (that is, it is specific enough for the element), this set of attributes does have to be instantiated before it can be used.
  - *Catalogue*. This contains both the attributes that characterise the element and the instantiations for all the elements. This means that the catalogue is ready for use.
- **Elaboration**. From the manner in which the characteristics of the schema have been obtained, it will be possible to ascertain whether or not the information the schema contains is justified in any way. This is important, as the information that appears in a characterisation schema should be founded and not simply appear because the person who designed the schema thought it was important.

The possible values for this criterion are:

- *Reflection by the researchers*. When the proposal (schema, metaschema or catalogue) is obtained by means of the reflection of the researchers, it is absolutely theoretical and also subjective, as it reflects their reality (of the researchers) on the problem of characterisation.
  - *Discrimination*. When the proposal is obtained by discrimination, the researchers have observed and studied the existing elements to be characterised and have built the schema on the basis of the differences and similarities between these elements. In this case, although the schema is not theoretical (it is founded on an empirical basis of observed reality), it is subjective, as it only takes into account the reality of the researchers.
  - *Surveying*. When the proposal is obtained by surveying, it also has an empirical basis, as reality is observed, but it is objective, as the proposal is made after examining more than one reality (apart from the reality of the researchers).
- **Aspects**. There are different relationships between the elements that take part in software development. For a characterisation schema to be complete, it should not only describe the object being characterised, but also take into account characteristics of other objects with which the object being characterised interacts and which can place restrictions on the use of the object being characterised. It



is also of interest to store information about past uses of the objects, as they can give people an idea of how the object behaved earlier. This criterion is an aid for measuring schema completeness.

The possible values of this criterion are:

- *Object*. This refers to the information that the schema will contain concerning the element that is being characterised.
  - *Environment*. This refers to the information that the schema will contain concerning the project restrictions on the element for it to be operational.
  - *Problem*. This refers to the part of the schema concerning the operating restrictions of the problem-related element (software to be built).
  - *Experience*. This refers to the part of the schema that will contain information related to the impressions of the users of the technique about earlier uses of the technique.
- **Instantiation.** Having made the proposal (schema, metaschema or catalogue), a series of steps have to be taken before it can be used. This criterion refers to how the proposal is instantiated, that is, when information is entered into the schema to conform a catalogue. The manner in which the schema is completed is of interest for several reasons. The information it contains must be reliable, as if it is wrong, the object could be used incorrectly, and the schema would fail to fulfil the purpose for which it was developed.

The values found in the revised literature for this criterion are:

- *Librarian*. This means a proposal of a schema or metaschema that is instantiated when the element is entered for the first time in the repository. The characteristics of this type of instantiation is that the schema is complete and completed by a person who is not the user of the repository (or producer or consumer) from the information contained in the element.
- *Researchers*. This means a proposal of a catalogue, all the elements of which are instantiated by the researchers themselves at the same time and attached to the proposal.
- *Expert in the element*. This means a proposal of a schema or metaschema, which is completely instantiated by a person who is an expert in the element.
- *Users*. This is again a proposal of a schema or metaschema in which a given element is instantiated when it is entered in the schema using information from the users of the repository. Here the instantiation is not necessarily complete and the values can also be modified.

- **Dynamism.** It is of interest for the information filed in a catalogue to be updated and also for the schema itself to evolve. This property refers to the schema itself evolving over time. This is of interest, as the information is dynamic, not static. With time, some characteristics could cease to be of interest, whereas new characteristics could appear.

This criterion is closely related to the above and, in some cases, its value depends on the value of the *instantiation* criterion. Therefore, it is important to know when the proposal under consideration suggests that the schema be completed and whether any sort of feedback is proposed. Some proposals suggest that it be instantiated once at the beginning and others propose an iterative process. Schemas in which the catalogue is completed by the researcher are static schemas, whereas the other instantiations can lead to static or dynamic schemas.

The possible values for this criterion are:

- *Dynamic.* A given catalogue, schema or metaschema proposal will be dynamic when its structure (proposed attributes) can evolve during use.
- *Static.* The proposal has been conceived in such a manner that structural changes cannot be made during use.

## 2.3.2 Related Work in the Characterisation Area

### 2.3.2.1 Studies on Software Reuse

Within the reuse area, the work on characterisation schemas carried out by Prieto-Díaz, related to the characterisation of software modules, Basili and Rombach, concerning the characterisation of any software artefact, and Henninger, who characterises problems during software development, will be studied.

#### 2.3.2.1.1 Prieto-Díaz

In [Prieto-Díaz, 1985], [Prieto-Díaz and Freeman, 1987], [Jones and Prieto-Díaz, 1988], [Prieto-Díaz, 1989], [Prieto-Díaz, 1990] and [Prieto-Díaz, 1991], Prieto-Díaz presents a characterisation schema for reusable software modules. His main objective is to aid the identification and later recovery of such modules (stored in a repository) to find the components that are less costly in terms of effort to adapt to the current project.

The schema was constructed by means of discrimination or examination of and later classification of existing reusable modules (what is called *literary warrant*), analysing the similarities and differences between these modules. This schema contemplates two aspects of the modules: (1) functionality (which would represent what) or the object and (2) the environment (which would represent where).

The schema is instantiated by the librarian when the element is entered in the repository. However, no new attributes can be added to the schema. It is the repository elements and

not the remarks or impressions of the consumers that are used to provide feedback to the schema, which means that it is static schema.

The work by Prieto-Díaz is significant mainly because it was the first on the subject of characterisation. This was the first researcher to realise the benefits of using characterisation schemas for classifying reusable artefacts. Additionally, he uses a method to obtain characteristics that is based on the study of a variety of real components and not on his own opinions. However, the characterisation is specific for reusable modules in this case, which means that it cannot be used for testing techniques. The fact that it is the librarian who carries out the instantiation causes problems, as this is not necessarily the person best acquainted with the reusable artefact. On the other hand, as the schema is static, it cannot evolve, even if information is detected that could be removed from the schema or new information appears.

### 2.3.2.1.2 Basili and Rombach

Based on the idea that anything related to development, and not just software products, is reusable [Rombach, 1992], [Basili *et al.*, 1994], Basili and Rombach present [Basili and Rombach, 1988a], [Basili and Rombach, 1988b], [Basili and Rombach, 1990], [Basili and Rombach, 1991] a characterisation metaschema for any software development element: products, processes, techniques, etc. The goal they pursue is to increase their possibilities of reuse (which they achieve by specifying the characteristics of the artefact that support decision making on whether or not it can be used under a given circumstance explicit). Owing to the generality of the metaschema, it has to be adapted to the type of artefact to be characterised before it is used.

The process they have followed to design the metaschema is reflection by the researchers, based on a reuse model, which is refined gradually through reasoning. Each step of the refinement captures the logic of the resulting schema. The schema contemplates three aspects: it should contain characteristics proper to the artefact (object) or object, it should contain characteristics of the relationships of the artefact with other artefacts (interface) or environment and it should contain characteristics of the environment in which the artefact can be used (context) or problem.

The schema is gradually instantiated by the users of the metaschema, as the element is used and, consequently, other characteristics of the element are discovered. Structural changes to the schema are also permitted with use, which means that the schema is dynamic.

The work of Basili and Rombach is important in that it shows that it is possible to reuse anything in SE. Additionally, they present an organisation in which both the schema itself and the information it contains can evolve with time, making it more flexible. Their metaschema is generic enough for any software object to be instantiated. However, it requires a refinement in which specific details proper to the elements to be characterised (modules, techniques, etc.) are added, which must be completed before the schema is used. On the other hand, the method they use to design the schema, based on the reflection of the researchers, is not sufficiently justified, as it has not been checked against reality.

Many of the aspects of the proposal by Basili and Rombach have been used as a starting point for the proposal made in this piece of research. The proposal aims to incorporate many of the strengths of the proposal by Basili and Rombach: distinction between aspects and dynamic schema incrementally completed by its users. On the other hand, it aims to overcome its limitations, which are mainly the generality of a metaschema that serves for any software element (here we construct a schema for a particular element like testing techniques) and its subjectivity, as it is created on the basis of investigations by its authors.

### 2.3.2.1.3 Henninger

In [Henninger *et al.*, 1995], [Henninger, 1995a], [Henninger, 1995b] and [Henninger, 1996], the authors propose a characterisation schema together with a support tool to capture and, thus, enable later dissemination of different problems related to software development, alongside their solution. The purpose is to help developers to find the best solutions (best suited resources) to their problems.

The process followed for creating the schema is never fully explained, from which it is inferred that the reflections of the researchers were used. The aspects included in the schema are: descriptions of problems, which are associated with resources (or solutions to the problem, possibly tools, development methods, people, process models, technology, etc.) and which constitute the object, and projects or environment associated with the object. Accordingly, one can start from any of the three aspects to arrive at any of the other two.

The repository is populated by means of its use. Each time that a user solves a problem, the solution is stored in the repository. However, the problem with this proposal, an attempt at the correction of which is explained in [Henninger, 1997], is that it does not clearly specify the information that it has to contain, which means that the information stored is often not of interest or is not useful. The schema is static, as it does not allow structural changes of the information it contains.

It follows from the above that the characterisation schema of the authors is not of use for testing techniques, as it is designed for the characterisation of problems. Additionally, the method that has been followed to get the characterisation is devoid of an empirical basis, as it has been obtained on the basis of the reflection of the researchers. The schema proposed is static, which does not allow for possible changes to the structure of the stored information.

By way of a summary, Table 2.12 reflects the values of the assessment criteria for studies on reuse.

From Table 2.12, it follows that within the family of reuse, the only approach that could be used to solve the problem in question is the work by Basili and Rombach. However, as this is a metaschema, it needs to be adapted. Another fundamental problem is that it is not sufficiently justified, which is essential. With regard to the other approaches, they are specific for other elements that are not testing techniques, which means that they cannot be used.

APPR.	OBJECT	PROPOSAL	ELABOR.	ASPECTS	INST.	DYNAM.
Prieto-Díaz	Software modules	Schema	Discrimination	Object Environment	Librarian	Static
Basili & Rombach	Any artefact	Metaschema	Research	Object Environment Problem	Users	Dynamic
Henninger	Problems	Schema	Research	Object Environment	Users	Static

Table 2.12: Values of the criteria of comparison for the family of reuse.

### 2.3.2.2 Studies on Software Architectures

As stated in [Garlan and Perry, 1995], a system software architecture is the structure of the components of a program/system, their interrelationships and the principles and guides that govern their design and evolution over time. According to this definition, it can be inferred that the architecture of a software system corresponds to one of its design levels.

The problem with software architectures today is that they are described completely informally (except some of the more advanced structural or object-oriented architectures). This is not usually of consequence for small software systems, but, in complex systems, where the team is large or when reuse is practised (in both cases the designers are not the same people as the programmers), this informal notation sometimes leads to the design being misinterpreted and, thus, a system is implemented that does not respect the design.

The area of software architectures is extensive and includes different efforts, whose aim is to make it a discipline. However, as mentioned in Section 2.1.3, only the characterisation of architectural styles is of interest for the work. This line of research has been developed by the ATA (Architecture Tradeoff Analysis) group at SEI (Software Engineering Institute).

The catalogue of publications of this group is very extensive. Therefore, only the publications that are directly related to the characterisation of architectural styles are listed here. These are [Garlan and Shaw, 1993], [Shaw, 1995a], [Shaw, 1995b], [Shaw and Garlan, 1996], [Shaw and Clements, 1997] and [Bass *et al.*, 1998].

The objectives pursued by the authors in characterising architectural styles are:

- *Establish a standard of architectural styles.* This standard will make it possible to unify the descriptions made by developers of the architecture of a software system.
- *Create a catalogue of architectural styles.* This catalogue will provide information to developers about a given style, will make it possible to discriminate between styles and will serve as a guide during the selection of the style or styles to be followed for the design of a given software system.

The authors provide a catalogue, which means that the schema is already completely instantiated. The catalogue has been designed following a process of discrimination by means

of the study and classification of numerous designs. This means that the different designs were observed and, on this basis, they deduced which characteristics differentiate one style from another.

The catalogue contemplates not only the characteristics proper to the styles (model) or object, but also characteristics of the requirements of the application or problem and characteristics of the environment in which the design is to be implemented (context) or environment, which can place restrictions on the developer when using the style.

As they provide a catalogue, it is the researchers who instantiate the proposal. This means that the schema is static, as it does not directly take into account the impressions and remarks of the users of the schema with respect to its form.

Here, the authors present a completed catalogue for architectural styles, which is not applicable for testing techniques. Although their proposal is justified, at least partially (the work of the authors was based on the discrimination of elements), it does not incorporate other viewpoints that could enrich the schema and bring it closer to the reality of developers. Being a catalogue, the proposal is static and does not, therefore, contemplate the possibility of other characteristics being added. One good point about this proposal is that it contemplates all the possible aspects of the characterisation of the element: both the object and its environment and the problem for which it is suited.

### 2.3.2.3 Studies on Technology Selection

Within this family, the approaches by Birk and Kröschel for the characterisation of software technologies and by Maiden and Rugg for characterising requirements elicitation methods will be studied.

#### 2.3.2.3.1 Birk and Kröschel

In [Birk, 1997a], [Birk, 1997b], [Birk and Krschel, 1999a] and [Birk and Krschel, 1999b], the authors propose a characterisation metaschema for characterising software technologies. This work is based on the fact that methods, techniques and tools that are not universally applicable, as stated in [Rombach, 1992], and the goal they pursue is to aid the selection of technologies for use in a software project.

The process they have followed to design the schema is not made explicit, and it is, therefore, assumed to be the result of the reflection of the researchers. And the aspect on which they focus primarily is to reflect the application domain or environment and the problem for which the technology is suited.

The metaschema is instantiated as follows. Every time a new technology is to be inserted, the metaschema is refined. It is then instantiated, asking experts. They do not explicitly take into account the impressions or possible remarks of the consumers, from which it is inferred that once the generic schema has been instantiated for the first time for a given technology, it does not change structurally, leading to a static schema.

In this case the authors present a metaschema which could, in theory, be suited for selecting testing techniques, but is not in practice, as it is too generic (techniques generally).

Furthermore, as it is obtained by reflection of the researchers, it is not justified and it is not known whether it would be adequate. Additionally, the schema presented has the drawback of being static, and cannot therefore evolve.

### 2.3.2.3.2 ACRE

Maiden and Rugg present in [Maiden and Rugg, 1996] and [ACRE Web page, ] a schema for characterising requirements elicitation methods to thus facilitate the selection of methods and help developers to prepare an acquisition programme. Apart from the schema, they propose a series of tables, which are actually the instantiation of the schema as a catalogue.

With regard to the process followed to get the schema, the authors speak of research and their own experiences. As the developers of the schema are experts in the area, one can infer that the process was based on observation and discrimination of the existing methods. However, the authors have added a stage where a series of experts validate the work they have done. The aspects reflected in the schema are the object and the problem.

As the researchers present the schema already instantiated as a catalogue, it becomes static, as they do not take into account the opinions of the schema users. From this it follows that the authors present a completed catalogue for methods of requirements acquisition, which is of no use for testing techniques. Although their proposal is to some extent justified (it is based on the discrimination of elements), they do not contemplate other viewpoints of selection apart from their own. Being a catalogue, their proposal is static, which means that they do not contemplate either possibilities of the information varying (new techniques, etc.) or other characteristics being added to the schema itself.

Table 2.13 shows the value of the assessment criteria for the family of software technologies.

APPR.	OBJECT	PROPOSAL	ELABORAT.	ASPECTS	INST.	DYNAM.
<b>Birk &amp; Kröschel</b>	Tecnologies	Metaschema	Research	Environment Problem	Experts	Static
<b>ACRE</b>	Requirements elicitation methods	Catalogue	Discrimination	Object Problem	Researchers	Static

Table 2.13: Value of the comparison criteria for the family of technology selection.

It can be deduced that the only approach that could be used to solve the problem at hand is the work by Birk and Kröschel. However, they propose a metaschema, which would have to be particularised for testing techniques. Additionally, their proposal is not justified, as it is based on the opinion of the researchers, which could lead to a schema of little use or with inadequate information for the problem of selecting testing techniques. Another problem is that Birk and Kröschel do not consider reflecting characteristics of the element, which means that information of interest would be missing. Finally, they propose a static schema, which would mean that any schema malformations could not be corrected.

APPROACH	OBJECT	PROPOSAL	ELABORATION	ASPECTS	INSTANTIATION	DYNAMISM
<b>Prieto-Díaz</b>	Software modules	Schema	Discrimination	Object Environment	Librarian	Static
<b>Basili &amp; Rombach</b>	Any software element	Metaschema	Research	Object Environment Problem	Users	Dynamic
<b>Henninger</b>	Development problems	Schema	Research	Object Environment	Users	Static
<b>ATAM</b>	Architectural styles	Catalogue	Discrimination	Object Environment Problem	Researchers	Static
<b>Birk Kröschel</b>	Software technologies	Metaschema	Research	Environment Problem	Experts	Static
<b>ACRE</b>	Requirements elicitation methods	Catalogue	Discrimination	Object Problem	Researchers	Static

Table 2.14: Value of the comparison criteria in the area of characterisation.



### 2.3.3 Conclusions on the Area of Characterisation

Table 2.14 shows an overview of the values of the comparison criteria for all the approaches studied in the area of characterisation

After studying the area of characterisation, the findings are as follows:

- There is no proposal that is specific for characterising testing techniques. On the one hand, there are proposals that suggest specific schemas for given elements and, therefore, are not applicable to testing techniques and, on the other, there are proposals that suggest schemas that are too abstract and whose instantiation would call for a huge effort on the part of schema users.
- When a catalogue is proposed, the resulting proposal is not at all flexible, whereas the metaschemas are too abstract. The ideal thing is to propose a schema that can be completed dynamically.
- The schemas are usually designed either by means of the discrimination of existing elements (which are at least justified) or, at worst, on the basis of the personal opinions of the researchers and not checked against reality. The opinions of other groups, like software developers or other researchers, are never taken into account.
- Only a few proposals take into account three of the aspects identified as desirable: object, environment and problem. Not one proposes storing information related to the possible opinions of developers gathered from their experience in using the elements.
- The schemas are usually instantiated by the researchers themselves a priori, which does not give the proposal any flexibility to grow as it is used. In other cases, it is instantiated by people who are not necessarily the best acquainted with the element to be instantiated. On other occasions, expert opinions are sought, which would be the ideal thing, if it were not for their unavailability. Practically speaking, the best option is for the schema to be instantiated dynamically as the schema information becomes available, gathering this information from as many sources as possible.
- Most of the proposals usually present static schemas with respect to instantiation (as is the case with catalogues, for example). However, even schemas whose instantiation is dynamic, do not allow for structural changes to be made, which leads to schemas that are not very flexible and which will perhaps not be very useful in the future. As stated by Basili and Rombach, missing information or information that is of no use could be detected as the schema is used. This will lead to changes in the composition of the schema that can only improve it.

## 2.4 Conclusions on the State of the Art

Hence, from the study carried out in the testing area, one can conclude that although there are numerous studies whose aim is to gain an understanding of and compare the different testing techniques, none of these is sufficiently exhaustive to form a body of knowledge on testing techniques. One might think that as there are so many, they could be combined to gather information of interest. However, most of the them are not comparable (neither for confirming other earlier studies nor for generalising the knowledge they discover), as the terms of comparison (metrics for the aspects studied) vary from one to another, making this comparison impossible. Finally, it is worth noting also that there are studies that investigate aspects that will unquestionably be of interest for researchers but not for developers, as they do not focus on the study of the pragmatic usefulness of the testing techniques. From all this it follows that: (1) it is not clear what information is of interest on testing techniques and (2) no one has yet tried to find out what this could be or gathered such information.

With regard to the study carried out in the field of characterisation, one can conclude that although there are numerous proposals related to the characterisation of different software elements, they are either too specific or too general to characterise testing techniques. Furthermore, the existing proposals are not usually justified with regard to the information that they suggest should be used to characterise the element in question, which means that they do not always contemplate all the aspects that should be used for characterisation. In many cases, it is the researcher who instantiates the schema, possibly leading to it containing incorrect information, which is aggravated by the fact that neither the contents nor the structure of the schema can be modified.

From all this, it follows that the problem of characterising software testing techniques for selection is an open one, as there is no satisfactory approach.

## Chapter 3

# Problem Statement

As discussed in Chapter 1, the problem of testing technique selection arises due to the following circumstances:

- The processes, techniques and tools used in the construction of software systems are not universally applicable [Basili and Rombach, 1991]. And this also applies to testing techniques, which are not all equally applicable for validating a given system.
- There is a wide variety of testing techniques [Chen and Yu, 1996], [Frankl and Weyuker, 1991], [Frankl and Weyuker, 1993b].
- The information about the different testing techniques is, at the best, distributed across different sources of information and, at worst, non-existent [Vegas, 2001].
- In the context of testing technique selection, the term *best* has different meanings depending on the person making the comparison [Ntafos, 1988], [Weiss, 1990].

According to Chapter 2, the state of the art with regard to the selection of testing techniques can be summarised as follows:

- The studies are neither exhaustive with regard to the universe of testing techniques nor for the family or families they study. This means that the knowledge of the different techniques under the same conditions is incomplete.
- The non-exhaustiveness of the studies could be compensated for if the results of the different studies could be compared or combined satisfactorily. Unfortunately, this is not possible either, as the findings of the different studies are not generally comparable, mainly because they use different metrics to define the same aspects under study.
- In some cases, the studies do not address pragmatic aspects, which means that they are difficult for developers to use.

- There are proposals on the characterisation of software elements. Unfortunately, however, none can be used because they are either too specific and are not applicable to testing techniques or they are too generic and do not address aspects proper to testing techniques.

As there are numerous techniques that are not universally applicable (generally in SE and particularly within software testing), it makes sense to talk of a need for selection. This can be mainly put down to the fact that performance differs depending on the situation in which the techniques are applied. The selection problem is all the more complex taking into account that the evaluation function is subjective and the theoretical knowledge of testing techniques today is not sufficient for a formal and objective evaluation function to be found.

However, the fact that it is still not possible to find a formal and objective evaluation function does not mean that the problem of testing technique selection should be abandoned or rated as unapproachable. The approach chosen in this research is to study the decision-making parameters used by developers and try to understand the process for the purpose of extracting a pattern of some sort for the evaluation function applied.

From the discussion in Chapter 1, it appears to be generally accepted that the process followed to make the selection is to compare given characteristics of the testing techniques with project needs. Having done this, the technique whose characteristics are best suited to the needs of the project will be selected. Accordingly, the problem of selection can be defined as the composition of two problems:

1. The completeness of the set of techniques on which the selection is based.
2. The appropriate identification of the characteristics that represent both the technique and the environment of the software project.

With respect to the problem of completeness, the composition of the original set of techniques will have a decisive influence on selection, which is as follows. Suppose that we have two sets of different techniques and one technique that will be considered best suited according to the evaluation function of a subject with respect to a given aspect of the software in a given project. If this technique is present in one set and not in another, the subject will choose the best suited technique to be used to evaluate the respective aspect of the software in the project in question from the set in which it is present, whereas it will not be able to be selected from the other set in which it is not present, leading to the selection of another less suited technique.

The reason why the original set of techniques can vary from one case to another is due mainly to the knowledge of the person who is going to make the selection. Obviously, two people do not necessarily have to be acquainted with the same testing techniques and even one and the same person does not necessarily have to be familiar with the same techniques at two different times in his/her lifetime (he/she may learn or even forget techniques).

There are several reasons why the needs of developers vary from project to project, but they will always be related to the characteristics of the project, like the type of software to be developed or the characteristics of the team of developers. With regard to the characteristics of the actual technique, they vary from one to another owing to the manner in which they have been conceived.

With regard to the subproblem of the connection of characteristics between project and technique, it is generally accepted that there is no optimal technique for all situations.

It follows that one has to attack on two different fronts to solve the testing technique selection problem: complete the knowledge of the person who is to make the selection about the possible options concerning existing techniques and discover the characteristics or information on the software project and on the testing techniques that have an influence on the fitness of a technique under given circumstances. These are the two points that this research aims to address by proposing a characterisation schema.

### 3.1 Description of the Problem

As concluded from the revision the state of the art, none of the proposals in the literature on characterisation schemas could be used for the purpose of this research. Although there is no solution to this problem at present, the characteristics of the solution are known:

- A characterisation schema is needed that is specific for testing techniques, does not need adaptation and is ready to be instantiated for each technique.
- The schema must reflect aspects of both the actual technique and the project environment.
- The schema will be instantiated incrementally as the information on each technique becomes known.
- Both the contents and the structure of the schema may change through use.

The problem to be solved will be precisely defined on the basis of these requirements below. The idea in this investigation is to characterise testing techniques to aid their selection. It is reasonable to expect that the actions to be taken by a developer who wants to make a systematic and balanced selection of testing techniques will coincide with those of any selection-based evaluation, which are:

1. List the techniques from which the selection is to be made.
2. Assess the characteristics of these techniques that are relevant for the project in question.
3. Assess the characteristics of the project in question.

4. Compare 2 and 3 and decide which is the best-suited technique.

However, it is not possible to complete this process in a reasonable time nowadays, which means that it is not feasible. There are two main reasons for this, which affect step one and two respectively.

- Step 1 would involve gathering all the existing testing techniques. This would mean not only gathering the techniques that are created by researchers (a task that, although costly, would be feasible, as researchers have an interest in the techniques being known and, therefore, disseminate them widely), but also gathering all the variations on a given technique designed by each developer (a task that is not feasible because it would involve gathering information indefinitely).
- Step 2 would involve the existence of a formal evaluation function that would make it possible to establish which characteristics or aspects of the techniques are to be evaluated and, additionally, the information on each technique would have to be documented.

Opposite this ideal process that would make it possible to find the best-suited techniques for a given project, there is reality. Here, there are, on the one hand, software developers, responsible for selecting the testing techniques, in accordance with their knowledge and beliefs, for later use. On the other hand, there are the inventors of the testing techniques, responsible for creating new techniques or improving existing techniques depending on the problems and shortcomings they identify. The process actually followed, which is much quicker, although much less effective, does not usually lead developers to find the best-suited techniques. This process is characterised as follows:

- At best, the developers list only the techniques with which they are acquainted and, at worst, which they have used.
- The evaluation function used during assessment is subjective and variable, not so often because of the person who makes the evaluation but because of the information available about the testing techniques.

Finally, the scenario proposed in this research, using the characterisation schema developed here, is as follows. The role of the developers (who are referred to as testing technique *consumers*) will be likewise to select the techniques to be used in a given project for later use, but the selection will be made on the basis not of their knowledge but of the information contained in a repository that is the fruit of successive instantiations of the characterisation schema for the existing techniques. They will also be responsible for feedback on the contents of the repository, in accordance with the findings they will have made as a result of using the technique. The role of the investigators (referred to as testing

technique *producers*) will be to create new techniques, as well as to study the characteristics and conditions of applicability of the techniques, in this case to satisfy the real demand for information of the repository and not their own beliefs. Finally, there will be a third role the *librarian*, who is the person who will be responsible for maintaining the repository, reflecting the information supplied by both producers and consumers. Figure 3.1 illustrates this situation.

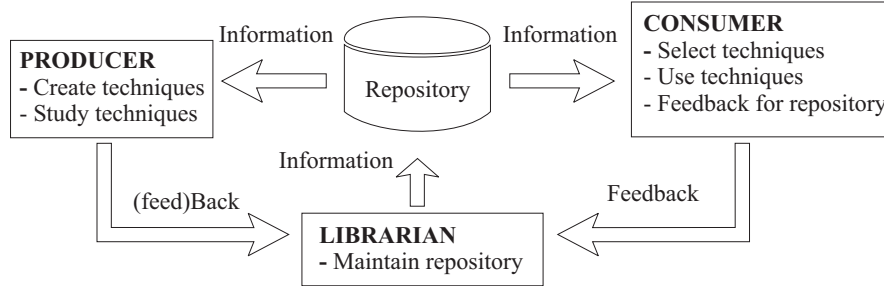


Figure 3.1: Repository environment.

The process of selection, in this case, will be as follows:

- The consumers will choose the techniques from those existing in the repository.
- The evaluation function will have been made explicit (and, therefore, objectified) so that the choice of the technique or techniques will be made in full knowledge of and not by *intuition* or ignorance of the technique or its characteristics.

From all this, it follows that the problem to be solved involves the construction of a characterisation schema that will be able to be used, on the one hand, to reflect the nature of as many testing techniques as possible and, on the other, to contemplate all the information relevant for selection.

## 3.2 Problem-Solving Approach

The solution proposed in this research to the problem of testing technique selection is a **characterisation schema for testing techniques**. Being a characterisation schema, it will be instantiated once per technique so that a repository can be built containing all the testing techniques relevant for the organisation.

The schema will be composed of a set of attributes that is invariant with respect to the testing techniques, which means that the schema provides just one means of describing the essence of all the testing techniques, all of which will be described in the same format. The attributes of which it is composed refer to both **characteristics of the actual technique** and **characteristics of the software projects for which the technique is suited**. This set of attributes is not flat, and the attributes will be grouped first as elements (which are

what are involved in the testing process), and these elements will later be grouped as levels, each of which will refer to different information on software testing.

Having designed the characterisation schema, it will be instantiated as follows. The first time that information is entered about a technique, the person responsible will instantiate the characterisation schema for the technique in question. It is unlikely that all the information required by the schema will be known at this time, in which case this information will be left blank until it is available and can be completed. As new information is gathered about the technique (either from new studies by producers or from information provided by consumers after using the technique), the information available about the technique in question will be updated. This involves the construction of a **dynamic repository**, the information of which will vary over time.

The information required by the schema is gathered from more than one source: on the one hand, from studies and producer knowledge (possibly including the person who invented the technique) and, on the other hand, consumer knowledge. They each have different knowledge. According to Robillard's ideas on knowledge in software development [Robillard, 1998], which is classed as *topic* knowledge (knowledge that refers to the meaning of the concepts) and *episodic* knowledge (knowledge that refers to the experience in using the knowledge), producers could be said to provide topic knowledge and consumers to provide episodic knowledge.

The schema provides suitable mechanisms for managing the repository, which will include the definition of procedures both to complete the repository of information and to use it as a selection aid.

The process to be followed to get the characterisation schema is iterative and empirical. It is iterative because the final schema will be the result of successive refinements of a preliminary schema proposed by the investigator. It is empirical because the preliminary schema reflects not only the perception of the problem by the investigator but also of other groups related to software testing, like developers or researchers in the area.

Ideas are checked by means of experiments in the natural sciences (physics, chemistry, biology, etc.). These sciences follow the theory+experimentation sequence to discover scientific knowledge. The scientist comes up with a hypothesis in the theory phase that is checked against reality during the experimentation phase. On the other hand, the social sciences (sociology, politics, etc.) follow the experience+experimentation sequence. The researcher formulates a hypothesis based on different realities in the experience phase, which is checked during experimentation.

The sequence followed in this piece of research is pseudoschema+experience+experimentation:

- *Pseudoschema*. Perception of the investigator concerning testing pseudoschema. This is not schema because it is a body of unchecked opinions and not of confirmed theories.
- *Experience*. Opinion of other developers and researchers in the testing area.



- *Experimentation.* Investigation of whether the solution improves selection in any way.

The main reason for following this sequence was that if the schema merely reflected the theory, then it would not necessarily improve the solution over the use of documents, as ultimately the investigator would gather the information from her own experience and thoughts (as there is no sufficiently solid theory to support this). Therefore, if the opinion of the investigator is backed by the experience of other subjects in selection, the schema will reflect more appropriate information.

This does not mean, of course, conducting a population study, where the more individuals questioned the better. Although this might appear to be the best-suited option to substitute the theory+experience sequence, it is not workable because it is impossible to get enough people to co-operate.

Thus, the problem-solving approach in this piece of research will be as follows. Firstly, a schema will be built that reflects the opinion of the researcher extracted from experience and published documents. This schema will be complemented with another that is the result of taking into account the viewpoint of developers and researchers in the testing area. This schema will be revised against the opinion of several experts in the testing area and accordingly modified. Once the schema has been generated, it will be checked twice. The first evaluation will be empirical and will involve the instantiation of the schema by the investigator to check issues of feasibility and the second will be an experiment with computer science students to observe schema use. The problem-solving process will be explained in detail in the next chapter.

### 3.3 Hypothesis Research Goals

Briefly, the hypothesis formulated in this piece of research is:

*The testing technique selection process is improved using the characterisation schema proposed in this piece of research.*

However, the proposal of using a characterisation schema to support selection is a proposal of the research presented here. Really, the schema proposed in this piece of research is just one of the possible forms of specifying this general idea. This means that this research does not intend to defend the idea that the schema proposed here is the best of all possible schemas. The goal of this research is to justify that describing testing techniques using characterisation schemas leads to a better selection process. Thus, the hypothesis really formulated here is:

*The testing technique selection process is improved using a characterisation schema as opposed to selection using books.*

This idea of improving the selection of testing techniques by means of a characterisation schema is based on a yet more general hypothesis formulated by Basili and Rombach in [Basili and Rombach, 1991]:

*The process of selecting software artefacts is improved using artefact characterisation schemas.*

Accordingly, the goal of this research and, therefore, its contribution is to check an instance of this general hypothesis, which specifically would state:

**H.** *The use of a characterisation schema improves the process of selecting the testing techniques to be used in a given software project in terms of the quality of and the time it took to get the solutions, compared against the use of books.*

### 3.3.1 Working Hypothesis

However, the term *improves* which appears in the main hypothesis is too broad for this hypothesis to be addressed as a whole. Therefore, the main hypothesis **H** will be itemised as a set of aspects that specify the term *improves*, leading to a sub-hypothesis for each of these aspects.

Thus, the general hypothesis **H** can be decomposed into the following sub-hypotheses:

- H<sub>1</sub>** The schema enables producers to describe the software testing techniques.
- H<sub>2</sub>** The schema enables consumers to decide whether or not to use a testing technique.
- H<sub>3</sub>** The schema enables all the existing types of testing techniques to be represented.
- H<sub>4</sub>** The schema describes a testing technique more fully (more and more useful information) than a book.
- H<sub>5</sub>** The schema helps to select better suited testing techniques than a book.
- H<sub>6</sub>** The schema enables techniques to be selected more rapidly than using books.
- H<sub>7</sub>** The schema is easy to understand so that it can be easily completed by producers and easily read by consumers.
- H<sub>8</sub>** The users of the schema work more readily with than without the schema.

These sub-hypotheses reflect the aspects that are to be improved in technique selection and are explained in Table 3.1.

ASPECT	IMPLICATION
Feasibility	The schema enables producers to describe the software testing techniques
	The schema enables consumers to decide whether or not to use a testing technique
Flexibility	The schema enables all the existing types of testing techniques to be represented
Completeness	The schema describes a testing technique more fully (more and more useful information) than a book
Effectiveness	The schema helps to select better-suited testing techniques than a book
Efficiency	The schema enables techniques to be selected more rapidly than using books
Usability	The schema is easy to understand so that it can be easily completed by producers and easily read by consumers
User satisfaction	The users of the schema work more readily with than without the schema

Table 3.1: Aspects of the characterisation schema for evaluation.

### 3.3.2 Testing the Working Hypotheses

For it to be possible to test the defined working hypotheses, both the metrics that are to be used for this purpose and the form in which the evaluation is to be completed for each of the aspects specified above have to be established.

1. **Feasibility.** Studying the feasibility of a proposal (in this case the proposed characterisation schema) involves examining both the operational aspect (does it work?) and the technical aspect (can it be built?). The results of this study will determine whether or not the proposed solution is feasible. In the context of this research, feasibility means:

- *Producer.* The proposed characterisation schema enables the nature of the artefact for characterisation to be reflected (it is possible to get the information required on the testing techniques).
- *Consumer.* Reading the characterisation of a testing technique, it is possible to decide whether or not to use it in a given situation.

This aspect will be evaluated by means what has been termed *empirical evaluation*. That is, the investigator will instantiate the characterisation schema for several techniques (covering a broad spectrum of technique types) to then check whether it is feasible to get the information required to complete the schema and whether it is possible to make the selection using the schema.

2. **Flexibility.** Flexibility implies a characterisation schema that reflects the relevant aspects of the different testing techniques so that none of the existing techniques are left out. This is related to the criterion of *exhaustiveness* that was mentioned in Chapter 2. The values of this criterion varied depending on how many of existing techniques were addressed. Accordingly, the schema will be more or less flexible

depending on the extent to which the different types of testing techniques can be represented.

This aspect will also be evaluated by means of *empirical evaluation*. That is, the investigator will again instantiate the schema for a set of testing techniques that covers all existing technique types.

3. **Completeness.** In this particular case, completeness refers to the fact that no relevant information is lost when characterising the testing techniques. This research will study completeness by observing whether the characterisation schema fails to contemplate information that is necessary for selection purposes.

This aspect will be evaluated by means of *experimental evaluation*. When making the selection, the subjects will be asked what information that they need for selection they think is missing from the characterisation schema.

4. **Effectiveness.** In this case, the effectiveness of the characterisation schema means that the consumer is capable of finding a set of techniques that are as *good* for solving the problem as if they had been selected using books. *Good* means that, assuming that there is an *optimum* technique for the situation in question, the set of possible techniques for use within the characterisation schema includes this optimum technique.

This aspect will be evaluated by means of *experimental evaluation*. The suitability of each of the techniques selected by the subject for the problem will be studied using both the schema and books.

5. **Efficiency.** In this case, where the aim is to minimise decision-making time, efficiency refers to the time it takes consumers to decide on which set of testing techniques to use.

This aspect will be evaluated by means of *experimental evaluation*, measuring the time it takes the subject to make the selection during the experiment using both the schema and books.

6. **Usability.** The ISO defines usability [ISO-9241, 1998] as: "The extent to which a product can be used by given users to effectively, efficiently and satisfactorily achieve specific objectives within a given context of use".

This aspect will be evaluated by means of *experimental evaluation*, analysing what problems users have when using the characterisation schema.

7. **User satisfaction.** This parameter aims to find out how well disposed people are to using the characterisation schema: whether they like it, whether they find it easy to use, and whether they would use it if they had the chance to.

This aspect will be evaluated by means of *experimental evaluation*, asking the people who have used the schema during the experiment what they think about it.

Table 3.2 shows how the hypotheses are evaluated and was generated using the GQM [Basili, 1991].

There are two interesting things with respect to the evaluation of the proposed solution. The first is that there are aspects that refer only to the schema and aspects that refer to the use of the schema as compared with the use of books. The aspects that fall within the first group are feasibility and flexibility. The aspects that fall within the second group are completeness, effectiveness, efficiency, usability and user satisfaction. The second group is orthogonal to the first group. Table 3.2 shows that two different models have been used to evaluate the schema:

- Schema use.
- User opinion.

Looking at the type of questions associated with each model, it is clear that the objective of the evaluation is different in each case:

- *Assessment*. The goal is to study a given property of the schema or of the situation using books. For this purpose, schema use will be used as the model, either by the investigator or by users. An objective observation of either the process or the result of using the schema or books will be made.
- *Improvement*. The goal is to improve the characterisation schema. For this purpose, the opinion of the user after using the schema will be used as the model.

The evaluations made by the investigator (feasibility and flexibility) are described in Chapter 9. The evaluations made by users (completeness, effectiveness, efficiency, usability and satisfaction) are described in Chapter 10.

OBJECT		QUESTION	EVALUATION
ASPECT	VIEWPOINT		
Feasibility	Producer	1. Is it possible to describe at least one testing technique?	Schema use
	Consumer	2. Is it possible to decide whether or not to use a testing technique in at least one situation?	Schema use
Flexibility	Producer	3. Can the schema be used to characterise any existing technique?	Schema use
		4. How formal does a technique have to be for it to be possible to characterise it?	
Completeness	Consumer	5. Is the information considered in the schema sufficient for selection purposes?	Schema use
		6. What information is missing from the schema?	User opinion
Effectiveness	Consumer	7. Can the schema be used to select the best-suited techniques?	Schema use
		8. Is there any case in which the schema cannot be used to decide which technique to use?	
Efficiency		9. How can effectiveness be improved?	User opinion
	Consumer	10. How long does selection take using the schema?	
		11. How many resources are required to use the schema?	Schema use
		12. How long does it take to decide whether or not to use a technique?	
Usability		13. How could efficiency be improved?	User opinion
		14. How long does it take to learn how to use the schema?	
	Consumer	15. How many explanations are required during schema use?	Schema use
		16. What sort of explanations are required?	
		17. How often is help consulted during schema use and for how long?	
		18. Are the names that appear in the schema appropriate?	User opinion
		19. How can usability be improved?	
		20. What are the advantages and disadvantages of using the schema?	
User satisfaction	Consumer	21. Would people be prepared to use it?	User opinion
		22. What improvements could be made?	
		23. What do people like and dislike about the schema?	
		24. What is a good environment for using the schema?	
		25. Is there any superfluous or redundant information?	

Table 3.2: Questions obtained using the GQM.

Part II

# RESOLUTION





## Chapter 4

# Problem Solving Process

The main difference between scientific and what is termed common knowledge lies in the methods used to generate new knowledge. The result of using this kind of methods is that the propositions formulated by a science cannot be called into question as easily as those formulated using other methods [Judd *et al.*, 1991].

This is apparently accepted by all disciplines. Physicists and chemists, mathematicians, psychologists, etc., all know that any result they present will not be valid unless they have taken an approach based on the scientific method. This method usually assures the veracity (within certain limits) of the formulated propositions.

There are different classes of sciences, which means that there is more than one variant of the scientific method. Obviously, a sociological statement is not verified in the same way as a physical statement. SE also needs to be tested empirically, as the veracity of its statements must be checked against real facts. However, SE, like psychology, has certain characteristics that differentiate it from the natural sciences (physics, chemistry, biology) and liken it to the social sciences (sociology, politics, anthropology, historiography), primarily, that many propositions on software construction refer to people rather than objects. While this makes research more complex (interactions are more involved because they do not deal with objects [Judd *et al.*, 1991]), it also suggests that methods and techniques from both classes of sciences should be used.

The method followed to solve the problem raised is explained throughout this chapter. Like the scientific method, this process must assure the fitness of the proposed solution.

### 4.1 Objectives of Technological Research

Scientific research and technological research have different means and ends. Rogers [Rogers, 1983] claims that these differences are of teleological origin, that is, are due to the different goals pursued by science and technology. Rogers actually makes four distinctions between scientific and technological research:

1. *Scientific research is broad spectrum, whereas technological investigation is narrow spectrum.*

Rogers points out that technological research is directed at serving the process of manufacturing or building things and, therefore, it has a clearly defined purpose, which may be to design a bridge using less material, a faster car, etc. In its endeavour to explain phenomena, however, scientific research is free to move away from the originally stated objective of research, as the research itself opens up new alleys.

On this basis, SE research is technological, as its purpose is specifically directed at serving the software systems construction process. Typical examples of SE objectives are to output higher quality software systems, use fewer resources, etc.

2. *Technological research is more limited than scientific research.*

This limitation refers to the stopping condition of the research. Technological research usually ends when it has reached a satisfactory solution. Scientific research, however, never ends. This is because scientific hypotheses can never be ratified, they can only be refuted. Technological research can be taken up again, either because of renewed interest in improving the result obtained in earlier research or because the technology on which it is based has advanced. On the other hand, it can also stop if another discovery appears which provides a better solution to the problem in question. These two circumstances do not occur in scientific research.

On this point, again, SE research can be said to be technological, because an investigation pursuing a particular goal (for example, a method for systematising the step from software requirements to design) will end when a method has been developed that outperforms existing approaches, even if it is not the best of all methods. This is the case primarily because it is possible to ratify that the method in question is better than existing ones (at least under certain circumstances).

3. *Science progresses by virtue of the refutation of its hypotheses, whereas technologists strive to avoid the falsification of their theories.*

Rogers attributes the interest of engineering disciplines in not refuting their theories to the disastrous consequences of failures, either for customers or for the company that built the product. Therefore, the goal of technologists is simply to run enough laboratory experiments to gain a good understanding of the behaviour of the theory formulated and be able to apply this theory routinely, rather than spending time on its refutation. From this point of view, it can again be concluded that SE research is technological, as software construction failures can earn a company a bad reputation and mean that its products are not purchased. Many important software development companies closed during the 80s due to problems with their products. Therefore, instead of trying to refute the theories they create, SE researchers endeavour to better understand their hypotheses (to find out when they should be applied).

4. *Scientists seek revolutionary changes and, although technologists also look for*

*conceptions that are revolutionary, the present technological level limits their investigations.*

With regard to the type of changes sought, the deployment of any of the accomplishments achieved by a technologist will be held back if the present state of the technology is not advanced enough for it to be used, which never happens in science.

SE research can also be said to be technological as far as this point is concerned. Theories such as GUIs facilitate man/machine interaction (and thus improve software system quality) or compilers ease systems programming were dependent on technological achievements like the mouse or improved computer performance (in terms of memory, CPU, etc.).

Insofar as they discuss the differences between technological and scientific research, these four points lay the foundations for the investigation to be carried out, which, as follows from the above, is technological, not scientific. Points 1 and 3 above, in particular, give an idea of what the research method for solving the problem stated in this dissertation should be like:

- The research discussed here is directed at facilitating or improving the process of testing technique selection in software projects and thus contributing to the construction of higher quality software systems. The purpose of this dissertation is to propose a characterisation schema that is used; that is, it should *be workable*, which means that the problem-solving method must be aimed at promoting (and even guaranteeing) its use. This focus on the use of the solution proposed led to the decision to get people related to the testing field involved in the research.

During characterisation schema design, a decision has to be made on what information it should contain. This is not an easy task, however, as the schema has to meet the information needs of a variety of people with different goals. More precisely, it must:

1. Be useful for developers when selecting the techniques required to generate the test cases and, thus, be able to evaluate code.
2. Be possible for testing technique producers to fill in the information asked for in the schema.

Having identified the constraints to be met by the information contained in the schema, the question is *What information does the developer need to select a testing technique?* One possibility is to think about what one believes developers would like to know when deciding on the technique or techniques to use and even gather a collection of information that appears to be more or less coherent. But, would this collection of information be the real solution to the selection problem? This

problem is far from trivial. If the inclusion of the information that appears in the schema is not justified on the basis of a theory (and no such theory exists today) or is incomplete with respect to the set required to make the selection, the fitness of the characterisation schema, or even its validity, could be questioned. Following this model, the schema generated would possibly be of little use and it would take longer to reach a satisfactory solution.

The goal in this piece of research is to be pragmatic and have the solution used (this is the only way of improving testing technique selection). So, in the absence of a theory that confirms why some information facilitates or is necessary for selection and other information is not, the schema should reflect the opinion of developers and testing researchers (future schema users). But, being a matter of opinion, there is a risk of the schema being a mere collection of non-convergent information. The process is, therefore, subject to two restrictions: (1) the thoughts of the investigator are used as a basis upon which the opinions of the participants take shape; (2) a convergence study is carried out to find out whether or not there is a solution to the stated problem. If this study were to find that the opinions did not converge, it would mean that there is not enough common ground between opinions, that is, there is neither a theory nor empirical knowledge enough about the subject to address the problem.

- Owing to the risk involved in deploying the product created, an experimental evaluation must be run by means of which to improve rather than refute the hypothesis (product).

The best way of examining product validity is to put it into operation and observe how well it fits in with development: the problems up against which users come and how the product could be improved to make it useful for developers. For this purpose, once the schema based on the opinion of producers and consumers has been built, the schema will be first instantiated for a range of techniques, that is, evaluated empirically, and, then, an experiment will be run with final-year (year six, second four-month period) computer science students. These students will have to use the schema under different circumstances. The use of the schema will provide feedback to the investigator, which can be used to improve the schema.

Bearing in mind that the proposed solution has to be based on the opinion of the people involved in the problem of selecting testing techniques, all the opinions should be taken into account during the problem-solving process. The relevant classes of people are developers and testing researchers. As there is also a need to check the result obtained, the solution should take into account the practical use of the schema. Pursuant to the ideas raised in Chapter 3, the schema will be evaluated empirically and experimentally. Figure 4.1 illustrates the problem-solving process to be followed. This process would be composed of an opinion poll among producers and consumers concerning the relevant information for selection, on

the basis of which a preliminary schema would be designed. This schema would then be evaluated and improved on the basis of an experiment with students, by means of which the operation of the schema built could be observed.

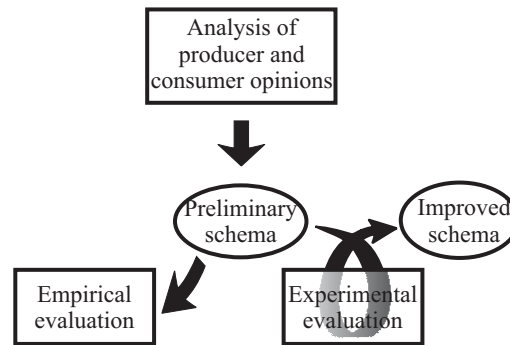


Figure 4.1: Preliminary problem-solving process.

## 4.2 Application of the Scientific Method to Technological Research

The above-mentioned differences between scientific and technological research made no reference to the method or process to be followed for solving a problem (although the stopping condition was mentioned). One might think, in principle, that, as the two branches of research are different, the traditional method of investigation, the scientific method, is not valid. Despite the observed differences, however, the scientific method is indeed applicable in the technological field. That is, the basic steps of the method followed to generate new knowledge do not differ in scientific and technological research.

The **classical scientific method**, proposed by Descartes [Descartes, 1637], is composed of the following steps:

1. Observation and recording of facts.
2. Analysis and classification of observed facts.
3. Inductive derivation of generalisations (hypotheses) from the facts observed in the preceding step.
4. Later testing of the generalisations (hypotheses).

This **classical scientific method** or **inductive scientific method** means that after the facts have been recorded, classified and analysed, a hypothesis is formulated (step 3) that responds to the problem to be solved and which is later tested.

However, the inductive method has been slightly modified by the new trends in the philosophy of science. Hempel [Hempel, 1966] argues that the inductive method is not strictly

true, as the first step implies observing and recording *all* the facts available to date, which is impossible because these are infinite. It is more likely that this first step refers to the search for facts *relevant* to a given hypothesis. This means that hypotheses must exist prior to step 3 of induction, and these hypotheses would serve as a guide for collecting the facts for the investigation. Thus, Hempel arrives at the following conclusion:

*Hypotheses are not derived from the observed facts, they are invented to account for these facts.*

And, ultimately, resolves that knowledge is discovered by what is called the **method of hypotheses**, which involves:

*Inventing hypotheses to try to solve a problem and later subjecting these hypotheses to testing.*

Testing in this **method of hypotheses** or **deductive scientific method** would be composed of two parts: one in which the hypotheses are tested against the data collected before the hypothesis was formulated (which is similar to the first step of the classical method), and another that reproduces the experimentation that appears in the classical scientific method (step four), in which the hypothesis is tested against facts in an attempt to reveal all its implications.

From this it follows that the first step of the method is to formulate a hypothesis that serves as a guide for preliminary data collection. This would be the start of a series of successive cycles of hypothesis generation and refutation.

The **deductive scientific method** would be as follows [Hempel, 1966]:

1. Establish a hypothesis on the basis of the personal reflection of the researcher.
2. Observe and record facts relevant to the stated hypothesis.
3. Test hypothesis. This will involve:
  - (a) Testing whether the recorded facts support the hypothesis.
  - (b) Testing the hypothesis and its implications against new facts.

However, this deductive method, which should theoretically serve not only for any sort of scientific research but also for technological research, has not been equally satisfactory in all branches of scientific research [Glaser and Strauss, 1967]. The problems this method raises for some branches of research are discussed below.

The traditional classification of the sciences [Hempel, 1966], [Judd *et al.*, 1991] is shown in Figure 4.2.

The first distinction between the branches of science that appears in Figure 4.2 (empirical and non-empirical sciences) is made according to the type of testing performed. The purpose of the empirical sciences [Hempel, 1966] is to:

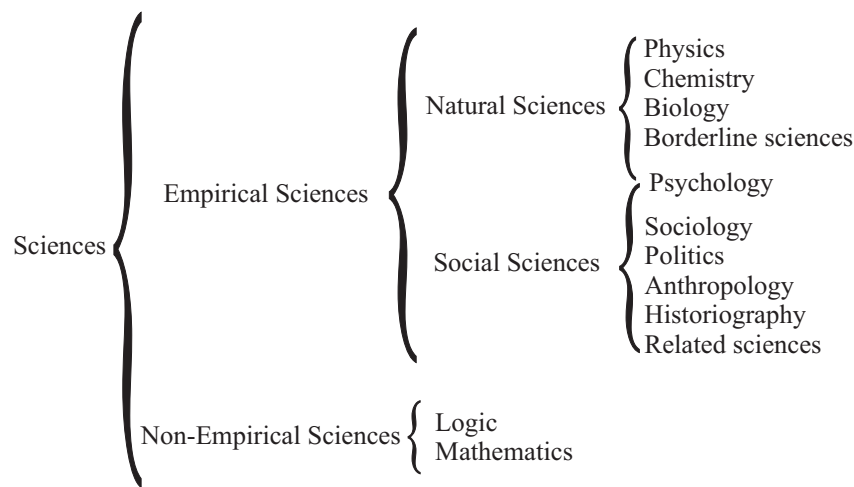


Figure 4.2: Traditional classification of the sciences with respect to scientific research.

*Explore, describe, explain and predict events that take place in the world  
in which we live.*

The statements of the empirical sciences must, therefore, be verified against experience, which can be done using a range of methods: experimentation, systematic observation, interviewing, document examination, etc. However, the propositions of the non-empirical sciences are proven without an essential reference to empirical data.

With regard to the natural sciences and the social sciences, their differences refer to what they study: nature and human beings. And although human beings are part of nature, they differ, today, with respect to the rules by which they are governed. Nature has objective general rules which research has to discover. Human beings do not appear to obey objective rules or, at least, not all are *discoverable*. SE appears, like psychology, to be a mixture of physical elements in which there is human participation.

According to Glaser and Strauss [Glaser and Strauss, 1967], the problem of the social sciences is conditioned by the fact that historically the hypotheses have been extracted from the perceptions of researchers rather than being generated from empirical data. Therefore, these hypotheses have had to be verified (or tested) at length, leading successive generations of researchers to focus on testing rather than on the generation of new hypotheses.

The main problem of the social sciences appears to be the difficulty of generating hypotheses from qualitative data (the data which are handled in most of these areas). Accordingly, the deductive scientific method, which justifies the creation of a preliminary invented hypothesis, leads the social sciences to take the mistaken view of associating qualitative data with verification and quantitative data with deduction, when both data types should be used for both verification and deduction.

However, this does not appear to be the only hitch in using the deductive scientific method in the social sciences. According to Glaser and Strauss, the hypotheses are not fully independent of the process that generated them. This means that when the soundness of

a hypothesis is judged, attention should be paid not only to the test results, but also to the process followed to generate the hypothesis. Glaser and Strauss uphold that hypotheses that are not based on data do not normally conform to reality, do not work or are not understandable enough to be used. Therefore, they are useless for both research and practical application. Moreover, as testing is inevitable, these thought-based hypotheses must be related to data, and the risk of the theory and the empirical world not agreeing is greater in the cases of hypotheses formulated by logical deduction.

Accordingly, Glaser and Strauss provide a variation on the scientific method for application in social sciences research. The **grounded scientific method** is actually the inductive scientific method, discussed earlier, where the first hypothesis arises as a result of the pure observation of the facts.

Furthermore, Peirce [Peirce, 1931 1958] discusses the differences between using induction or deduction, while introducing a new concept called *abduction*. Thus, he claims that:

*Abduction is the process of forming an explanatory hypothesis. It is, therefore, the only logical option for introducing a new idea, because induction does nothing more than determine a value and deduction merely unfolds the necessary consequences of a pure hypothesis. Deduction proves that something must be; induction shows that something is really operational; whereas abduction is confined to suggesting that something may be.*

This dissertation proposes a method midway between deduction and induction, employing abduction. The investigator formulates a preliminary deductive hypothesis used to collect the preliminary data and be able to better understand the problem, thus making later data analysis easier. A second inductive hypothesis is formulated from the data collected and compared with the first one. From this comparison, a new abductive hypothesis emerges, which will be tested against the empirical facts.

The steps to be taken are:

1. Establish a hypothesis by personal reflection of the researcher or *deductive hypothesis* (deduced from the facts known by the investigator and which led her to detect the problem).
2. Establish a hypothesis grounded on the data or *inductive hypothesis*. This will be done by:
  - (a) Observing and recording the facts relevant to the theoretical hypothesis formulated.
  - (b) Analysing and classifying the observed facts.
  - (c) Inductively deriving generalisation from the facts observed in the preceding step.



3. Formulate a new *abduced hypothesis* by comparing the earlier two hypotheses.
4. Test the abduced hypothesis.

Figure 4.3 shows the proposed process tailored to the problem of testing techniques selection. First, a deduced theoretical schema is built and then an induced empirical schema is then designed. These two schemas are synthesised by means of abduction to formulate a preliminary schema that will then be evaluated empirically and refined by experimentation.

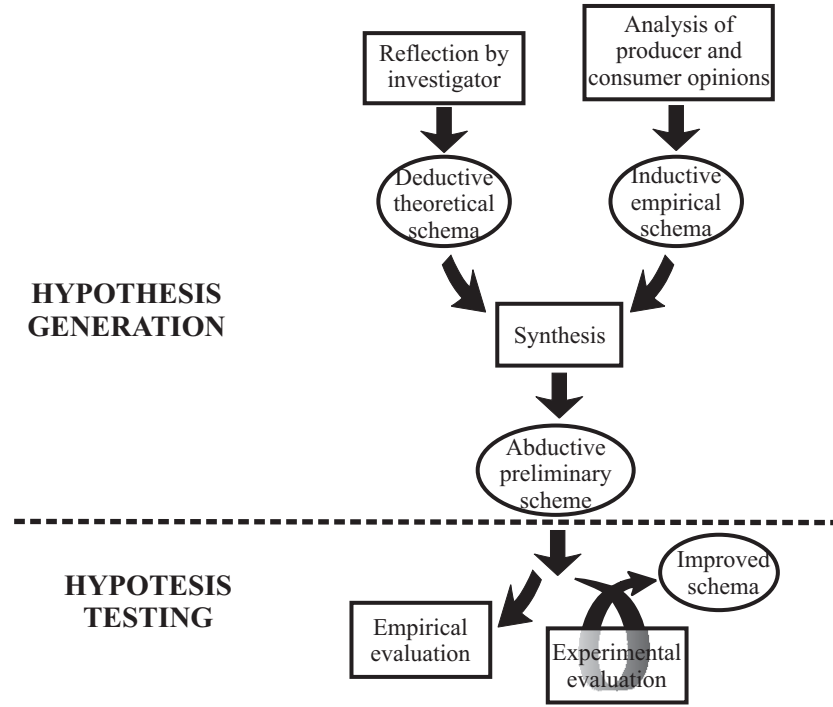


Figure 4.3: Deductive-inductive problem-solving process, with abduction.

It is clear that this process for building the schema is an empirical and iterative approach to solving the problem of testing technique selection, which is to determine a necessary and sufficient collection of information for selecting a testing technique, albeit based on a preliminary hypothesis obtained by deduction. Thus, the preliminary hypothesis will guide later data collection and analysis, whereas this hypothesis, very weak from the viewpoint of its theoretical justification, will be strengthened and reformulated by the data collected, even before testing. Finally, the new deductive-inductive hypothesis is tested.

### 4.3 Expert Peer Review versus Experimental Testing

The schema obtained after the synthesis of the theoretical and empirical schemas reflects the viewpoint of the investigator, developers and researchers concerning the selection problem. However, according to the philosophy of some sciences (for example, medicine), it is interesting to get a second opinion with regard to a given complex problem. Therefore,

a series of experts in the testing field have been asked to give their opinion on the abducted preliminary schema prior to evaluation. The goal of this expert peer review is to correct possible schema defects caused by the way in which it was derived. The possible defects of the schema obtained prior to the review by experts are as follows:

- *Defects of form.* Both producers and consumers have given their particular view of the information they believe to be relevant for selecting testing techniques. However, the investigator alone created the structure that reflects this information. It would not be amiss to get a second opinion on this structure.
- *Defects of substance.* The information for the preliminary schema was gathered indiscriminately. It may contain errors involuntary introduced by the investigator or by the people participating in the research. For example, there may be redundant information (dependencies between information contained in the schema), missing or unworkable information not detected by the investigator.

This is why it was decided to include a peer review by testing experts after the synthesis of the two schemas and before evaluation. This revision will take place prior to testing to correct any schema malformations beforehand. Thus, the schema construction process would be as shown in Figure 4.4.

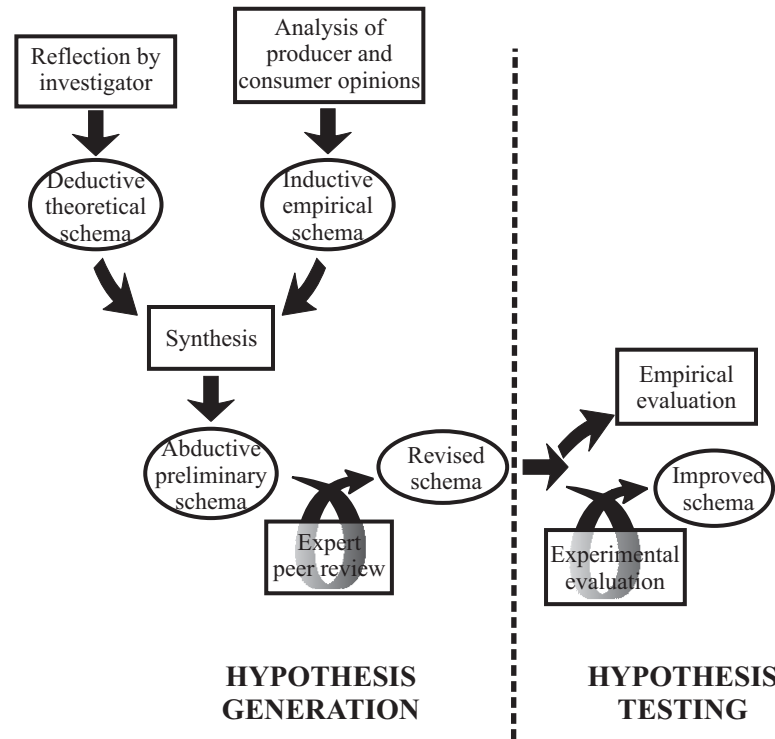


Figure 4.4: Deductive/inductive problem-solving process with expert peer review.

## 4.4 Problem-Solving Process

By way of a summary, the final problem-solving process followed is divided into two parts: hypothesis generation and hypothesis testing.

- *Hypothesis Generation.* Hypothesis generation has been divided into four different stages. This was done in order to explicitly state the different sources of information used to formulate the hypothesis and because each iteration aims to gather different information types. The phases of generation are:
  1. *First Generative Iteration: Deductive Theoretical Schema.* In this phase, a preliminary characterisation schema is built that will serve as a starting point and will guide preliminary data collection. The objective here is to build a schema on the basis of an analysis of the testing techniques to be characterised. So, the construction of this schema was guided by deductive reasoning concerning available testing techniques and what relevant characteristics they all have in common.
  2. *Second Generative Iteration: Inductive Empirical Schema.* The purpose of this phase is to create a schema that reflects the opinions of developers and researchers on the relevant information for characterising/selecting a testing technique. So, the aim is to contemplate all the information needed by consumers to make the selection and the information that producers believe to be necessary for understanding the nature of a testing technique. The process is, therefore, inductive, producing a schema containing the characteristics desired by producers and consumers.
  3. *Synthesis of Perspectives: Proposal of the Preliminary Schema.* In this phase, the two schemas built earlier are merged to provide a single view of the information that is relevant for selection. This abductive process synthesises the information reflected in the earlier schemas, guaranteeing that there is no loss of information.
  4. *Improvement of the Schema: Expert Peer Review.* In this phase advice is sought from experts in the testing field, who revise both the form and the substance of the schema. The results of the assessment are used to create a new refined version of the schema.
- *Hypothesis Testing.* As is the case with hypothesis generation, testing has been divided into two parts. Again, this was because each test was directed at different population groups examining the schema and at assessing different aspects of the schema. The phases for testing were:

1. *Empirical Evaluation.* The aim of this evaluation is to examine basic characteristics of the schema, like its feasibility and flexibility.
2. *Experimental Evaluation.* This evaluation involves running an experiment with computer science students. The primary aim of this experiment is to observe the completeness, efficacy, efficiency and usability of the schema. Another refined version of the schema is created on the basis of the results of the data analysis.

Each of these phases of the schema construction process are explained in more detail below.

#### 4.4.1 First Generative Iteration: Deductive Theoretical Schema

As the process of schema design has been defined, the first step is to build a theoretical schema that will be added to and improved in later iterations, that is, a starting point.

This schema will be based on the experience and knowledge of the investigator and her thoughts on what information is involved in decision-making on which testing techniques to use. It is important to note that this does not pretend to be a solution to the problem. Its purpose is to serve as a starting point on the basis of which to start to build the characterisation schema. This first schema was built according to the procedure followed by Prieto-Díaz [Prieto-Díaz, 1991], which involves studying existing testing techniques.

A strategy of decomposition will be followed to build this preliminary schema. First, the different objectives or levels of testing will be identified, along with the different elements that participate in the testing process within each level. Finally, the different characteristics of the schema will be generated.

Chapter 5 describes the construction of the deductive theoretical schema. It also prescribes the use and evolution process of the repository originated by the instantiation of the characterisation schema for different techniques.

#### 4.4.2 Second Generative Iteration: Inductive Empirical Schema

The schema obtained in the first iteration reflects the opinion of the investigator on the information that can influence decision-making on which testing techniques should be used in a given project. However, this schema does not necessarily respond, at least completely, to the opinion of producers and consumers on the selection problem.

The objective of this phase is to ascertain the information that the characterisation schema should contain to satisfy producers and consumers. This outputs a schema that responds to their opinions.

Figure 4.5 shows the activities to be performed to get the  $i$ -th empirical schema.

The empirical schema has been obtained incrementally. A set of opinions (questions or information respectively) about the information required to completely select/define a testing technique was obtained for each consumer/producer surveyed. The sets of

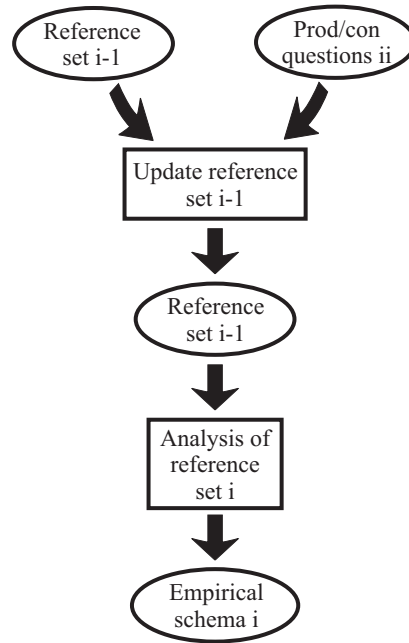


Figure 4.5: Design of the  $i$ -th empirical schema.

questions/information obtained have been analysed incrementally, not in parallel. This means that the producers/consumers have been gradually incorporated, which has made it possible to cover the total set of possible producers/consumers according to their characteristics.

More exactly, the iteration for running the analysis was as follows. Taking a reference set (originally empty) and the opinions of the producer/consumer, the reference set is updated to include any opinions not included before, and the respective empirical schema is obtained. The reference set can be updated in several ways: either by adding new questions or reformulating others to make them more generic or more specific (never by deletion).

An interesting point here is to know the characteristics of the participants, as it is important to know what type of producers/consumers are represented by the schema to be built.

Another point, which is not as important as accounting for all types of producers/consumers, is the number of people that have to participate in this phase. The sample of people does not need to be of a minimum size to achieve statistical significance. The number is not essential, as Glaser and Strauss [Glaser and Strauss, 1967] state that the number of data collected during research is relevant for testing and not for generating the hypothesis. So, the number of individuals involved will be important at that point and, as such, will be taken into account later on.

The stopping condition for this activity is the stability of the characterisation schema. It will not be possible to stop gathering information from different people until the rate of change of the schema is low enough. Therefore, the evolution and change of the characterisation schema as new producers/consumers are incorporated is examined in this phase.

Producers are used because developers/consumers are not experts in testing technique theory (although they are in the testing process) insofar as they do not necessarily have to thoroughly understand or be acquainted with all testing techniques. It appears, therefore, that another viewpoint on selection should be taken into account to complement the one held by consumers, and this is the viewpoint of the researcher/producer. A synthesis of the two viewpoints is possibly the best way of empirically ascertaining the information to be included in the schema.

Chapter 6 explains the construction of the inductive empirical schema.

#### 4.4.3 Synthesis of Perspectives: Proposal of the Preliminary Schema

In this phase, the two characterisation schemas created earlier are taken and synthesised into a single characterisation schema. For this purpose, the attributes of which the schemas are composed are analysed one by one, and their inclusion in the preliminary schema is studied according to the following rules:

1. Any equal attributes that appear in at least one schema will be directly entered in the preliminary schema.
2. If there are similar attributes or some attributes are more generic or more specific than others, the best way of adding them to the schema to assure that no information is lost during synthesis and there is no redundancy will be studied.

Once the preliminary schema has been built, the source of the attributes of which it is composed is examined so as to analyse the different viewpoints of the subject types that have contributed to creating this preliminary schema.

Chapter 7 addresses the synthesis of the two views of the schema.

#### 4.4.4 Improvement of the Schema: Expert Peer Review

At this point, there is a schema whose information responds to both the thoughts of the investigator and to the opinion of consumers and producers.

It would appear, however, appropriate for someone else to revise the schema and give an opinion on it, as neither the consumers nor the producers have so far seen the schema (they were asked for their opinion on selection, but they were never shown what information had been input).

This stage involves sending the characterisation schema to a series of experts in the testing field, who will review it and give their opinion on both its form (structure and organisation) and substance (its contents).

The preliminary schema will be modified on the basis of the analysis of the opinions of the experts to incorporate their suggestions, giving rise to a new, improved schema.

Chapter 8 reflects the expert revision.

#### 4.4.5 Empirical Evaluation

This first testing phase is carried out on the revised schema. Its purpose is to evaluate the feasibility (from the viewpoint of producers and consumers) and the flexibility of the characterisation schema.

For this purpose, the investigator will instantiate the characterisation schema over again to study these aspects. Also, an example is given for the usage of the schema for selection purposes.

The empirical evaluation is explained in Chapter 9.

#### 4.4.6 Experimental Evaluation

This second testing phase is again carried out on the version of the schema that includes the opinions of both experts and producers and consumers. In this testing phase, a series of subjects will act as consumers and use the schema to select testing techniques.

This evaluation takes the form of a controlled experiment, in which the different characteristics of the schema, like completeness, effectiveness, efficiency, usability and satisfaction, will be evaluated.

Both quantitative and qualitative data will be collected during the experiment, which, after analysis, will be used to again modify and improve the schema.

Chapter 10 reflects the experimental evaluation of the characterisation schema.





## Chapter 5

# First Generative Iteration: Deductive Theoretical Schema

This chapter addresses the construction of the theoretical characterisation schema. As mentioned earlier, the characterisation schema reflects only information relevant for selecting testing techniques and its purpose is to help users to make the selection without having to have an exhaustive and practical knowledge of all the techniques. As there are wide variety of techniques, the schema has to be designed to be able to reflect the nature of all the techniques: both similarities and differences. Indeed, the goal of this chapter is precisely to build a characterisation schema that can be used to reflect the differences and similarities between the testing techniques on the basis of the principal investigator's view of software testing.

### 5.1 Testing Process

After formulating the question "what information is required to satisfactorily select the set of testing techniques that will be used in a software project?", one has to look for sources of information to find an answer to this question, and these sources are: testing (observe the testing process) and testing techniques (what they are like, what characteristics they have, etc.).

As an important source is the testing process, the role played by existing testing techniques in this process will be briefly recalled. As discussed in Chapter 1, the main difference between the static and dynamic methods of evaluation is that the static methods evaluate the software when the system is at rest, whereas the dynamic methods evaluate the software by examining the behaviour of the operational system. The anomalous behaviour of the software system is indicative of the existence of a fault in the system.

Inputs have to be supplied to put the software system into operation. Therefore, the dynamic evaluation of the software (or testing) is composed of four separate stages:

1. Generate a set of test cases that contain the inputs that will be supplied to the

system for the purposes of testing.

2. Run the test cases and examine the behaviour of the software system or outputs.
3. Analyse the behaviour of the software system to determine whether or not it was as expected.
4. Search for the software fault that caused the system to behave irregularly in order to correct the fault. This step will only take place if the evaluation is corrective.

Figure 5.1 shows the succession of these stages.



Figure 5.1: Process of generating test cases.

It is important to note that test cases play an essential role in software testing. If the goal of the software testing is to evaluate a given quality attribute (correctness, performance, reliability, etc.), the test cases must be such that they can be used to reflect the aspect of interest of the software as exhaustively as possible. The generation of test cases is, therefore, a critical task. In view of the time constraints in software projects, however, the set of test cases should be the minimum to cover the aspect under consideration.

This takes us back to stage 1 of the testing process, which addresses test case generation. There are now a series of techniques, known as testing techniques, designed for the purpose of generating test cases. These techniques are based on different software elements for generating the cases. Traditionally, and as discussed later in this chapter, testing techniques can be classed according to different parameters, which are typically:

- The sources of information or inputs required to be able to apply the technique. With respect to this parameter, inputs are classed as specifications-, program- or interface-based.
- The generality of the techniques. There are techniques that are applicable to all sorts of software and there are specialised techniques for concurrent software, object-oriented software, real-time software, etc.
- The quality attribute of the software to be tested: reliability, usability, correctness, etc.

It is this type of parameters, which distinguish one technique from another, that will be studied to extract the aspects potentially of interest for selecting testing techniques.

Before concluding this section and moving on to the construction of the theoretical schema, it remains to make just one final remark with respect to the process reflected in Figure 5.1. This section has focused on steps 1 and 2 of the process, leaving aside 3 and 4, which address

the analysis of the outputs and the location and correction of the fault or faults in the software, as these steps are outside the scope of testing techniques and, therefore, of this research.

## 5.2 Stratification of Testing-Related Information

Applying the testing process described above to a given software system, one finds that it is divided as follows:

1. Selection of the quality attributes that are to be tested, as well as the expected values for each attribute, depending on the operational characteristics that the software should have (including things like the problem to be solved, the software application domain, the characteristics of the end users, etc.).
2. For each of the attributes identified in step 1, one should:
  - (a) Identify the project environment (both technological and human) in which the test is to take place.
  - (b) Select a testing technique to generate the test cases, depending on the operational characteristics of the software identified in step 1 and the project environment identified in step 2(a).
3. For each generated test case, and provided there are changes that affect the part of the software in question:
  - (a) Execution of the specified test cases.
  - (b) Evaluation of the results obtained. Depending on the outcome of the evaluation, a decision will be made on whether to stay in this step, return to step 2 or end.

Although decisions have to be made at all three points, the decisions that are relevant for this research and, therefore, on which it will focus are the first two. This is mainly because the selection of testing techniques is related precisely to these two points.

The main difference between points 1 and 2 lies in the fact that the purpose of point 1 is to establish a generic framework within which the testing of the software in question will take place. This step is necessary because not all software systems built today are the same, and a decision must therefore be made on which is the best way to evaluate each system. As regards step 2, it is necessary because not all projects are the same (even if they are building the same software). This means that neither the characteristics of the developer company, nor the employees, nor the technologies will be the same, and the tests to be run must therefore be carried out differently.

The characterisation schema must capture all this to assure that selection can be made optimally. This means that the schema must include information about:

- The characteristics of the software within which the test is to be take place (point 1).
- The conditions of the project within which the test is to take place (point 2).

More formally, these types of information are named and defined as follows:

- *Tactical Information.* The testing framework that will be used to evaluate the software code under development. The underlying idea here is to identify the characteristics that the test cases to be generated should have to assure that the test will be successful. This level focuses on what could be termed generic project characteristics.
- *Operational Information.* The testing technique or techniques that are used to get a set of test cases with the desired characteristics. The idea is to discover the testing technique or techniques that behave best under the project circumstances in question. It focuses on what could be referred to as the particular or specific project characteristics.

From the nomenclature used, one might think that a third level, *strategic level*, is missing from this division. This no less important level does indeed exist in software testing. However, this level is so high (it corresponds to the objectives of the software tests: construction of a quality software system) that it is not taken into account in this research, as it would affect all the testing processes equally, irrespective of how they have been defined.

The information contained in the **Tactical Level** is related to the initial or tactical planning that will be followed to run the tests. Faced with a given software, the first thing to be done as far as testing is concerned is to identify when each of the code evaluations is to be run during development, what parts of the code will be affected and what aspect or aspects of the software are to be evaluated in each evaluation.

As is the case in the industrial manufacturing of some materials, where the characteristics that the material should have are established by analysing the uses to which the material is to be put, the use to which the test cases to be generated will be put will be what determines the characteristics they should have for testing purposes. Take a plastic, for example, the fact that is to be used either to manufacture the inside of a car, to make plastic bags, to fabricate bottles, etc., will determine how flexible, how resistant and how malleable it has to be. Likewise, the fact that a set of test cases is to be used to test the security of a software system or the correctness of an algorithm implementation will determine whether the cases should be such that they exhaustively test all sorts of inputs, only the most common inputs or perhaps the inputs that entail anomalous behaviour on the part of the user.

Finally, it is important to explain that just as a given material cannot be used on all occasions and some of its properties have to vary depending on its use (leading to variations or versions of the material), when a set of test cases is generated for a given purpose it is very likely that they will not be useful in other circumstances.

Accordingly, the tactical level will reflect the information related to the use to which the generated test cases will be put, that is, whether they are going to be used to test an algorithm, a subsystem, the entire system, and whether they will be used to discover defects in software or to evaluate a given quality attribute of the software.

The information contained in the **Operational Level** is related to the optimal conditions of testing techniques operativeness, once given characteristics of the environment in which the technique is to be applied have been determined. Just as certain pressure and temperature conditions are required for a chemical reaction to take place, the conditions of application of a technique have to be as conducive as possible for the expected test cases to be generated effectively (in terms of time and resources) and efficiently during software testing. This means that it may or may not be appropriate to apply a given technique depending on the knowledge and experience of the personnel and whether or not the available tools are suitable. This is equivalent to the reaction not taking place or to the products obtained being of poor quality.

In other words, the operational level reflects the characteristics of both the technique and the project environment: tools, knowledge of the personnel, characteristics of technique applicability, etc., from which it will be possible to deduce whether or not the technique in question is the best suited for the project situation in question.

These types of knowledge have been grouped around different levels because they refer to different matters. Whereas the tactical level establishes whether the testing process is suited for the project in question in terms of what aspects of the system to test and when (that is, what tests to run on the software), the operational level is a guide for selecting the best-suited techniques to assure that the individual tests selected are successful. Software testing will be satisfactory when the tasks assigned at each level have been successfully executed.

Thus, one may conclude thus by saying that the testing process is essential for establishing the purpose for which the test cases generated will be used, as it determines what characteristics the test cases should have (focus on test cases). Having done this, the choice of the technique or techniques will be based on determining which of these (given the working environment of the project) can be used to assure that the set of test cases have the desired characteristics. Figure 5.2 shows how the characterisation schema is decomposed into levels.

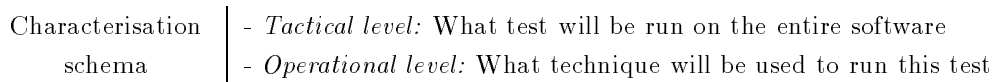


Figure 5.2: Decomposition of the characterisation schema into levels.

## 5.3 Testing Process Elements

So far the parts of testing process relevant for the characterisation schema have been identified. However, the granularity of the information of interest about these parts is still too coarse to be used in the schema. Therefore, this information has to be decomposed.

### 5.3.1 Tactical Level

As mentioned above, the aim of this level is to identify the test to which the code will be subjected or to choose the tactic to be followed to test the code. The following things have to be taken into account to identify the test to be run: the aspect of the software to be evaluated, which parts of the code are affected by the aspect in question, etc. Therefore, the use to which the test cases are to be put has to be specified.

Thus, this research has to establish the parameters that best define both the test and what is to be achieved using the test. There would appear to be two parameters:

1. The *purpose* or *objective* of the test, which defines what is to be tested, that is, the goal or aim of the test.

The set of cases generated when applying a given testing technique cannot be used to test any software quality attribute or to test the same attribute in the same way. For example, a set of test cases generated to test whether an algorithm is correctly implemented is not generally useful for checking whether the implementation of this algorithm is efficient or whether the system is acceptable. Suppose that one wants to check, on the one hand, system security and, on the other, system usability. The best way to test security is to use test cases that do not correspond to the routine use of the system but which represent attacks or unlikely situations. To test usability, on the other hand, one looks for test cases that represent the usual or common uses of the system. And, again, if one wants to test the correctness of an algorithm, one must use test cases that test both the normal actions of the algorithm and the exceptional cases (whether or not they are erroneous).

Furthermore, a technique that generates cases to test security in a safety-critical system is of no use for generating cases in a non-safety-critical system. And this is precisely what will be reflected by the purpose of the test: the software attribute that is to be evaluated using the test and how rigorously or with what degree of confidence this is to be done.

2. The *scope* of the test, which can be defined by saying what part of the software system is to be tested, when the test is to be run and the components of the software system that are affected by the test.

Depending which test is run, it will affect different parts of the software, ranging from an algorithm, through an entire module, a group of modules that perform a system function, to a subsystem and even the entire system. Also, depending on

how the development has been organised, the test will take place at a one or another time within the process. Also the part of the functionality offered by the system that needs to be tested should also be specified.

The scope, then, refers to the part of the system involved in the test. Figure 5.3 shows the decomposition of the tactical level.

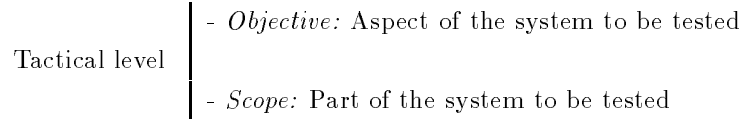


Figure 5.3: Decomposition of the tactical level into elements.

### 5.3.2 Operational Level

As mentioned in Section 5.2, the aim at this level is for the application (or use) of the technique to be as effective as possible, as well as efficient. This will involve a series of factors, which are discussed below.

Being a software process, the generation of test cases can be represented generically as shown in Figure 5.4(a). As shown in Figure 5.4(a), a software process generates a software product, where the techniques used, on the one hand, and the resources used, on the other, are the controllers of the process. If this generic view is specified for the case at hand, the process would be the generation of test cases, the input would be the software (generally, as each testing technique calls for specific inputs that vary from one technique to another), the output would be the generated test cases and the controllers would be, on the one hand, the technique or techniques used and, on the other, the tools and personnel, as shown in Figure 5.4(b).

In other words, the test case generation technique that is applied to the software outputs a series of test cases within an environment that is determined by the tools available for performing the task and the personnel who carry out the task in question.

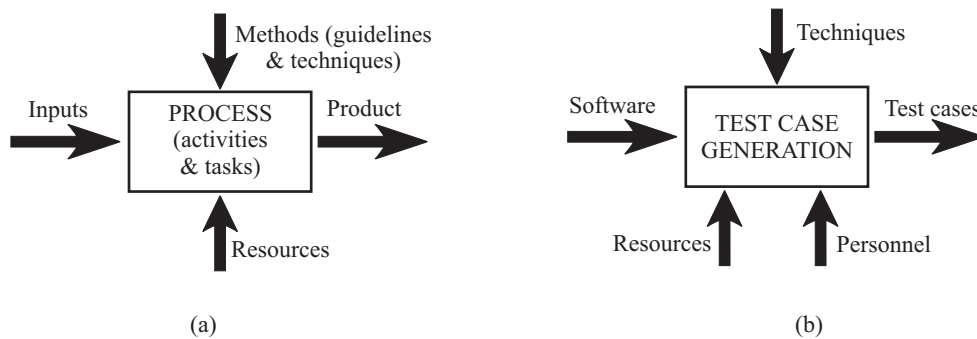


Figure 5.4: Process of test case generation.

Therefore, according to Figure 5.4(b), it can be said that the information that the operational level of the characterisation schema should contain has to refer to:

1. The people who are to use the technique or **agents**. The characteristics of these people can lead to one or another technique being chosen. If the testing personnel are not very experienced in one technique and there is no time for training, another is likely to be selected.
2. The **tools** that should or could be used. The fact that a company does or does not own a given tool that supports the use of a given technique can lead to the selection of one technique rather than another.
3. The software (code) to be tested or **object**. The code has certain characteristics that can determine the use or rejection of a technique. For example, the type of programming language used, the code size, etc.
4. The **technique**. Depending on the characteristics of the technique, a decision can be made on whether or not to use it at a given time. Characteristics like complexity, effectiveness, maturity, usability, etc., will be decisive for deciding on its use.
5. The generated test cases; that is, the **results** (and/or consequences) of using the technique. Some characteristics of the technique are environment dependent, and these are precisely the ones that reflect its behaviour. How good a technique is when applied can be ascertained from the generated test cases and not from the technique. Thus, some characteristics of these test cases will be of interest for selection purposes.

Figure 5.5 shows the decomposition of the operational level.

Operational level	<ul style="list-style-type: none"> <li>- <i>Agents</i>: People who are to use the technique</li> <li>- <i>Tools</i>: Artefacts that can be used with the technique</li> <li>- <i>Object</i>: Code to be tested</li> <li>- <i>Technique</i>: Technique characteristics that influence selection</li> <li>- <i>Results</i>: Test cases generated with the technique</li> </ul>
-------------------	--

Figure 5.5: Decomposition of the operational level into elements.

## 5.4 Attributes of the Theoretical Schema

Briefly, it was found first that the information relevant to the selection of testing techniques can be grouped at two levels. Then, different testing process elements were identified within each level.



This, however, is not the lowest level of schema decomposition, which can be further refined by reducing the granularity of the information identified so far (agents, tools, technique, results, object, purpose and scope) to the level of the attributes of these elements. In other words, it remains to determine the information of interest about the identified concepts. The attributes of the schema are identified in the following sections. For this purpose, each of the elements identified at each level will be examined in more detail to shape the atomic characteristic or characteristics representing all the information of interest.

### 5.4.1 Operational Level

At the operational level, the elements that represent the characteristics of the people who apply the technique to generate the test cases, the tools available for applying the technique, the technique, the test cases generated by applying the technique, and the software undergoing tests have so far been identified. Below, the attributes that supply the necessary information for selection are identified.

#### 5.4.1.1 Agents

This element takes into account the characteristics of the people who are to use the technique. This group of attributes is significant at the operational level, as the characteristics of the people who apply the technique have an influence on the test cases that will be generated.

Human factors are not normally taken into account when comparing the effectiveness and/or efficiency of testing techniques. For example, Weyuker *et al.* state [Weyuker *et al.*, 1991] that their study (like many others) does not take into account practical criteria such as people's behaviour when using the testing technique. In order to illustrate that this sort of criteria should be taken into account, suppose that two subjects A and B separately apply the same technique to the same program and find a set of faults Fa and Fb.

With regard to effectiveness, if the human factors are not taken into account, one is assuming that Fa and Fb are exactly the same, which, as several experiments have shown [Basili and Selby, 1987], [Kamsties and Lott, 1995], [Wood *et al.*, 1997], is not true. However, it is important to bear in mind that before getting Fa and Fb, the subjects had to generate a set of test cases, which we will term Ca and Cb. Assuming that Fa and Fb are different, there are two possibilities:

1.  $Ca=Cb$ , that is, both subjects generate the same set of test cases.
2.  $Ca \neq Cb$ , that is, the subjects generate different sets of test cases.

Case 1, where Ca and Cb are equal, means that if Fa and Fb are different, it is because A and B have a different capability of detecting (observing) system faults. This means that any comparison between two techniques, T1 and T2, assumes that, given a technique and a program (or software), there is a single set of associated test cases, and, therefore, any

attribute or quality of these test cases is due solely and exclusively to the characteristics of the technique.

The second case, where the individuals generate different test cases, would indicate that the techniques are for some reason not equally applicable by all people and that any characteristics that the test cases have will be due not only to the technique but also to the person who applied it.

However, it has been demonstrated experimentally that two people applying the same technique generate different test cases [Basili and Selby, 1987], [Kamsties and Lott, 1995], [Wood *et al.*, 1997] (which means that case 1 cannot occur). Indeed, in his book [Myers, 1979], Myers recommends the use by experienced people of the error guessing technique, which means that the experience of the person in applying the technique is something to be taken into account when deciding on whether or not to use a given testing technique. On the other hand, other studies [Myers, 1978], [Basili and Selby, 1987] show how experience in testing has an influence on the effectiveness of a technique, although these studies are not binding on any technique in particular. Nor do they explain whether the effectiveness is due to the effectiveness when generating test cases, the effectiveness in detecting faults in the software or both. The characteristics of agents that are of interest for the schema are those that have an influence on the application of the technique: on the one hand, their experience and on the other their knowledge. Knowledge is important because this will have an influence on whether or not they are acquainted with the technique (and if they are not acquainted with the technique whether or not it will be easy to learn and how well), and experience is important because it refers to putting their knowledge of the techniques into practice.

#### 5.4.1.1.1 Experience of the person who is to apply the technique

This characteristic is related to the experience of the person who is to apply the technique. But experience measured in absolute terms says very little, as the person who is going to generate the test cases may be experienced in other activities of software development and not in testing or be experienced in other techniques but not in the one in question.

Accordingly, it is not only important to state how much experience is required to apply a technique, but also in what subject in particular experience is needed. The application of a technique may merely call for theoretical experience in testing in some cases, experience in other techniques may suffice in others and it may be necessary for the subject to be experienced in the technique in question in others.

This information will be taken into account by creating an attribute in the schema, which will be termed *experience* and which will reflect the type of experience needed by the person who is going to use the technique to guarantee, if not an optimum, at least an acceptable application of the technique.

#### 5.4.1.1.2 Knowledge required to be able to understand or apply the technique

This information refers to the training of testing personnel or, more precisely, the training

of the people responsible for generating test cases (the subject of interest for selecting testing techniques). This characteristic of technique users focuses not so much on theoretical training in the technique or techniques of interest as on the agent's knowledge of everything involved in the use of the technique.

For example, testers have to be acquainted with the programming language in which the code is written in order to be able to use the basic path technique (and, generally, white box tests). Additionally, there may be a tool that automates some parts of the technique. For example, the flow chart and cyclomatic complexity (in basic path testing) can be generated by a tool, with which the person should be acquainted in this case.

Accordingly, with regard to agent training, both the knowledge of the object, that is, the code, specifications, etc., and the technological knowledge, that is, about the testing tools, should be specified.

This information is of interest primarily because if the personnel need knowledge that they do not possess, the training to be provided has to be taken into account. It is important to remember that the provision of training, if necessary, implies a financial investment and takes time.

This information will be reflected in the schema by creating an attribute that will be termed *knowledge*, which should reflect the advanced knowledge that the people who are going to use the technique should have. As mentioned earlier, this knowledge will consist not only of concepts of the technique but also of other related knowledge.

Table 5.1 shows the decomposition of the agent element.

ELEMENT	ATTRIBUTE	DESCRIPTION
Agents	Experience	Experience required to use the technique
	Knowledge	Knowledge required to be able to apply the technique

Table 5.1: Attributes of the agent element.

#### 5.4.1.2 Technique

This element contains information that represents the characteristics of the testing technique that have an impact on the decision of whether or not to select the technique. It is important to stress that, for selection purposes, the information about a technique should focus not on specific aspects of the application of the technique (for example, the sequence of steps to be followed to generate the test cases), but on pragmatic aspects (for example, the benefits of using the technique in a given situation).

The relevant information about the testing technique is: the tools available to support technique use; whether the technique is easy to understand and use; whether it is a technique under experimentation or has already been tested; the relationship with other techniques; where information about the technique can be found; whether the results the technique generates are repeatable; and, generally, what impression people who have used the technique

have.

#### 5.4.1.2.1 Tools

In [Hamlet, 1988], Hamlet speaks of the importance of taking advantage of tools in SE, and how merely using such tools can contribute to making SE an engineering discipline. According to Hamlet, tools should be used whenever possible.

The information to be reflected here refers to any useful tools that can be used to make the job of generating test cases when applying the technique easier. There are no tools that fully automate any technique at present, but there are general-purpose tools that can be used to automate some parts of the application of some techniques (that is, they are associated more with the testing process than with the actual techniques). Therefore, the only information relevant for selecting techniques will be the tools available for use for applying the technique. Therefore, it has been decided to include this information in the technique element rather than in an attribute termed tools (it does not make any sense having an element with only one attribute).

The interest in finding out whether tools exist is conditioned by the fact that both the generation and execution, not to mention the maintenance, of test cases usually takes a lot of effort. The existence of a tool that automates part of the test case generating technique can have different implications. For example, it will reduce the time it takes to generate test cases, or, perhaps, some knowledge that would be needed if the technique was applied manually will be obviated, or, alternatively, knowledge of the testing tool will be needed, etc.

This information is taken into account by creating a new attribute in the schema that is termed *tools* and that will identify all the tools that automate at least part of the technique. A tool will be identified, for example, by means of the name of the tool and the company that manufactures the tool.

#### 5.4.1.2.2 Ease of understanding

This information is especially relevant when the person who is to apply the technique is not experienced in the technique. If this is the case, the ease of understanding or comprehensibility of the technique will play an important role in selection. In order to improve comprehensibility, it is essential that the technique is explained as well as possible.

The attribute generated in this case is *ease of understanding* or *comprehensibility*. Obviously, simpler techniques will be preferred to techniques that are more complicated to understand and, therefore, apply.

#### 5.4.1.2.3 Effort for applying the technique

This information is related to the fact that, even if the agents are familiar with the theory of the technique (that is, has understood how it works), it will take more effort to apply the technique from the concepts learnt if these are difficult or complex. This attribute reflects the need to ascertain the resources that are consumed when applying the technique. Obviously, an attempt will be made to avoid techniques that consume too many resources and an effort

will always be made to use techniques whenever possible that do not involve too much effort on the part of the people who are going to apply the technique (supposing that two techniques are equally effective).

This information can be taken into account by adding an attribute to the characterisation schema that will be termed *cost of application* and which will indicate the effort required to apply the technique.

Looking at this and the above attribute (comprehensibility), one finds that both describe two aspects of what could be the same characteristic: technique complexity. A technique can be complex because it is difficult to understand (which means that once it has been learnt, it is not costly to apply), because it is difficult to apply (in which case not until one is very experienced will one develop the necessary instinct to apply the technique) or for both reasons (in which case the technique would have to be really effective for it to be worthwhile using it). As the values of these two attributes are independent (the value of one does not condition the other), we have opted not to unite them in a single attribute termed complexity.

#### 5.4.1.2.4 References to information on the technique

This attribute reflects the interest that consumers may have in knowing what sources of information there are associated with a technique. Faced with a given technique, it is essential for the consumer to know from what documentation they can extract information about the technique, should they have any doubt about its application. Reflecting at length on this subject, however, not only the documentation in which information about the technique can be found is of interest. Generally, any source from which information about the technique can be gathered will be of interest to consumers.

Accordingly this information is introduced as accurately and as generically as possible in the characterisation schema by adding an attribute termed *sources of information*. This attribute contains pointers to additional information about the techniques, whether this be documentation, people, web pages, or generally any kind of source of information.

#### 5.4.1.2.5 Dependencies among techniques

This information reflects the need to know what relationships there are between the different testing techniques for the purposes of technique selection. In particular, consumers will be most interested in this respect in knowing what techniques can or should be combined with another and whether they should be applied in a particular order. The use of one technique can possibly mean that another should have been used beforehand or that there is a specific technique for use afterwards. This is quite logical, as a range of studies [Basili and Selby, 1987], [Kamsties and Lott, 1995], [Wood *et al.*, 1997] have shown that the best way of testing software is by complementing various techniques, as it appears that different techniques tend to detect different defects.

Accordingly, this information is taken into account by adding an attribute to the characterisation schema that is termed *dependencies* and which specifies which techniques it is recommended to combine with a given one and how they should be combined.

#### 5.4.1.2.6 Degree of formality of the technique

The degree of formality of a technique is important when considering its possible application. On the one hand, if agents who are not very experienced or not very well acquainted with the technique use a technique that is not very formal, they run the risk of applying it incorrectly, as the steps to be followed during technique application are not as well defined as in the formal techniques. On the other hand, the more experienced the people are, the more liberties they are likely to take when applying a technique. In either of the two cases, if the technique did not achieve the expected results (its use was not satisfactory), it will be more difficult to find out whether the users employed it incorrectly or whether the technique itself is not satisfactory if it is not formally defined.

Associated with the degree of formality of a technique is technique repeatability, and this refers to the fact that the efficacy of the technique is independent of the person using the technique. Although the same technique applied by two different people on the same software can lead to different results, the more formalised the technique is, the more likely two people are to arrive at the same results after application.

This information is introduced in the characterisation schema by creating an attribute that will be termed *repeatability* and which will indicate whether the technique is systematised, formalised or neither of the two. The degree of formality of a technique can vary from mere guides or recipes (neither systematic nor formalised) to a series of well-defined and organised steps (which will be systematic and may or may not be formalised).

#### 5.4.1.2.7 Adequacy criterion of the technique

The criterion used in the literature to classify testing techniques is what is referred to as *adequacy criterion*, which is considered a stopping rule of test case generation. Zhu *et al.* [Zhu *et al.*, 1997] study existing adequacy criteria and define two forms:

- *Generators*: Serve to select new test cases and are, therefore, test case selection criteria.
- *Acceptors*: Serve to determine the quality of the set of test cases and are, therefore, adequacy criteria for the set of test cases.

Zhu *et al.* [Zhu *et al.*, 1997] show that both definitions are equivalent from the mathematical viewpoint and that the term *adequacy criterion* can be used in both cases.

Bearing this in mind, the relationship between the adequacy criterion and a testing technique is defined in this piece of research as follows. The testing technique can be used to generate a set of test cases, indicating how they have to be selected. The adequacy criterion serves to guide the selection of test cases, stating which is the next case to be generated and when to stop generating test cases. It follows that the essence of a testing technique is the adequacy criterion.

Hence, it can also be inferred that, as the essence of a testing technique is the adequacy criterion, what one is really doing when selecting a technique is to select a given *adequacy*

*criterion*. This means that the adequacy criterion is a critical characteristic for inclusion in the schema.

Figure 5.6 shows the different types of adequacy criteria defined in [Zhu *et al.*, 1997]. The adequacy criteria will be values of the *adequacy criterion* attribute.

Adequacy criteria	Sources of information	Specifications based Program based Interface based
	Implementation knowledge	White box Black box
	Focus	Structural Fault based Error based

Figure 5.6: Different types of adequacy criteria.

Table 5.2 shows the decomposition of the technique element.

ELEMENT	ATTRIBUTE	DESCRIPTION
Technique	Tools	Available tools that ease the use of the technique
	Comprehensibility	Whether or not the technique is easy to understand
	Cost of application	How much effort is needed to apply the technique
	Sources of information	Where to find information about the technique
	Dependencies	Relationships of one technique with another
	Repeatability	Whether two people generate the same test cases
	Adequacy criterion	Test case generation and stopping rule of the technique

Table 5.2: Attributes of the technique element.

#### 5.4.1.3 Results (or product)

The information contained in this element refers to what characteristics the test cases generated by applying the technique will have. It is to be expected that the test cases will have different characteristics depending on the technique applied, who applies the technique, to what software it is applied and the type of tools that are used, and these characteristics need to be known for the tests to be as successful as expected.

The information of interest about the test cases includes knowledge of the cover of the technique, the number of test cases generated, the type of defects it helps to describe, the capability of the set of cases to detect defects and the number of cases of the set that are not useful.

##### 5.4.1.3.1 Number of generated test cases

Having discovered that one of the essential differences between techniques is due to the underlying adequacy criterion, it is only logical that one way of comparing the techniques is by means of their adequacy criterion. As discussed above, there are different adequacy criteria. Each technique obeys a adequacy criterion. So, once the best-suited adequacy criterion has been established, the number of techniques that can be used at a given time will be limited.

Although Zhu *et al.* [Zhu *et al.*, 1997] touch upon the subject of comparison of adequacy criteria, it is further discussed in [Ntafos, 1988] and [Weyuker *et al.*, 1991]. One of the aspects proposed for comparing adequacy criteria is cost. When speaking of the cost associated with a adequacy criterion, the measure usually used [Weyuker *et al.*, 1991] is the size of the set of test cases generated. However, the number of generated test cases is not an indicator of the cost of running the tests, as not all the test cases take as long to execute, which means that a better measure would be the *cost of execution* of all the test cases.

#### 5.4.1.3.2 Type of defects discovered

One fairly widespread hypothesis in the software testing area is that different testing techniques find different types of defects. Indeed, studies have been carried out [Basili and Selby, 1987] to observe the effect of applying several techniques on the same software and have discovered that more defects would be discovered in the software than if just one technique were used.

Each set of generated test cases can help to discover different types of defects. Therefore, an attribute into should be added to the characterisation schema termed type of defects discovered.

In principle, one might think that this attribute should be associated with the technique and not with the test cases. However, it is important to bear in mind that it is not the technique but the test cases that *show up* faults. Accordingly, this attribute will be considered to refer to the generated test cases.

#### 5.4.1.3.3 Fault detection capability

The capability of the set of generated test cases to detect defects is the same as the effectiveness of the technique. Not all the sets of test cases serve to detect the same defects. This is why an attribute will be added to the schema that will be termed *effectiveness*.

As in the case of the type of defects, one might think that this attribute refers to the technique. It has been associated with results for the same reason as given above for this attribute.

#### 5.4.1.3.4 Coverage provided

Coverage measures the amount of system inputs covered by the set of test cases generated by the technique. Traditionally, this measure is linked to code, although it could also be applied to requirements or designs. Indeed, the coverage or capability of the set of test cases generated to cover the input is very much related to the completeness of the set of cases. The ideal situation would be not to have to add test cases to the set generated by the technique. On this basis of this ideal case, one aims to have to add as few as possible.



Indeed, Wallace's software error analysis technique [Wallace, 1988] (which suggests using different metrics as an aid for collecting historical data on errors and their later analysis) speaks of the importance of collecting the total number of test cases required and not generated. This metric is related to the fact that the technique omits generating useful test cases, that is, with the completeness of the set of test cases.

The coverage provided by the test cases is taken into account by adding a characteristic to the characterisation schema termed *completeness*.

#### 5.4.1.3.5 Number of useful test cases in the set

The application of a testing technique can result in the set of test cases not being complete (as discussed above) or in there being redundant cases (repeated cases). This information is related to the redundancies in the test cases. Going back to Wallace's error analysis technique [Wallace, 1988], it also suggests the use of the total number of test cases executed with respect to the generated cases. This metric is related to the fact that the technique generates useless test cases.

This information is taken into account by adding an attribute to the characterisation schema termed *correctness*.

#### 5.4.1.3.6 Adequacy degree of the technique

The *adequacy criterion* of the technique has already been proposed as an element of the schema. However, not only the adequacy criterion is important for the schema. As stated in [Zhu *et al.*, 1997], a set of test cases can be seen from the viewpoint of whether or not it fulfils a given adequacy criterion, which is measured within a range rather than as yes or no, so that one would be measuring the extent to which a given criterion is fulfilled. This would lead to also considering the *degree of adequacy* as a characteristic of the schema. Thus, for example, if one takes sentence coverage as the adequacy criterion of a technique, the adequacy degree would specify the percentage of the number of sentences that the set of generated test cases covers.

Viewed from this angle, if the *adequacy criterion* is a characteristic inherent to the technique, is this the case with the *degree of adequacy*? In principle, one might think it is, that is, that the degree of adequacy is also a characteristic of the technique. However, going back to the definitions of the adequacy criterion, one finds that the adequacy criterion defines the technique as such, whereas the degree of adequacy is something that is measured on the set of test cases that has been generated by the technique. Thus, by definition, the degree of adequacy is assumed to refer to the test cases and not the technique in this research.

Table 5.3 shows the decomposition of the results element.

#### 5.4.1.4 Object

This element contains the information related to the characteristics of the code that can have an influence on the use of a testing technique. This group of characteristics is interesting and important in that code, being the end product of software development, is the reflection

ELEMENT	ATTRIBUTE	DESCRIPTION
Results	Completeness	Coverage provided by the set of test cases
	Cost of execution	Time the technique takes to execute the test set
	Type of defects	Type of defects the technique helps to discover
	Effectiveness	Capability of the set of test cases to detect defects
	Correctness	Test cases to be deleted from the set
	Adequacy degree	Extent to which the adequacy criterion is achieved

Table 5.3: Attributes of the results element.

both of all the decisions made during development (design or implementation decisions) and of the characteristics of the application environment of the software, such as the type of user who is to use the software, the software application domain or the task to be performed.

With regard to design decisions, software development can be seen as series of successive transformations in which the level of formality rises, starting from requirements specification and ending with the writing of the source code. It is because of this series of refinements that the characteristics and/or development decisions can be said to be reflected in code. For example, if the requirements call for a complex system (with many interactions), the resulting design will also be complex (as the chosen software architecture will reflect the interaction of the different system modules), as will the code. The type of development paradigm selected will influence the type of programming language chosen (functional, concurrent, imperative, object oriented) etc.

Briefly, when studying the characteristics of the code that possibly have an influence on the use of a testing technique, one will not have to focus or be guided only and exclusively by the characteristics of the code, but also by all the characteristics of software development that are somehow reflected in the code.

The information associated with this element is: whether the testing technique is linked with any software development paradigm, whether it can be used with any type of software, whether it can be used with any programming language, whether it can be used with software of any size and whether it is linked to any particular life cycle or methodology.

#### 5.4.1.4.1 Software architecture

Not all techniques are applicable to all the software development paradigms, as their underlying architectures have different testing needs. Thus, certain techniques will have to be used depending on the architecture of the software to be tested. This point is illustrated taking the example of structured (or call and return) architectures and object-oriented architectures.

Object orientation makes it possible to build more modular systems (favouring encapsulation) and includes inheritance and polymorphism. Inheritance enables the derived classes to inherit attributes and behaviours from the base classes without having to be redefined, and polymorphism makes it possible to redefine the attributes and behaviours

that are different to that of the base class. Object orientation means that both the design and coding vary with respect to the structured paradigm: the programming languages include new features which will have to be tested-, as do the designs. These characteristics are, as mentioned above, inheritance and polymorphism, and will mainly affect integration testing. Indeed, the modular tests are simpler but the integration tests are more complex in object orientation.

Therefore, if this information is to be reflected in the characterisation schema, a new attribute will have to be defined termed *software architecture*.

#### 5.4.1.4.2 Software type

As software systems become a part of people's lives, more and more different systems are built. When computers started to be used, they were mainly used for intensive mathematical calculations. Today, there are fully integrated into everyday life, and can be used for tasks as varied as book publishing, education, or the control of what are known as intelligent buildings.

Of course, the main characteristics of a software system are determined by the characteristics of the task they are to be perform, which in turn will have special characteristics to be tested. It is not the same to test a real-time system as to test a management or expert system. Therefore, the schema needs an attribute to reflect this information, termed *software type*.

#### 5.4.1.4.3 Programming language

The type of programming language used in coding will have an influence on the technique to be chosen, as the typical constructions to be tested usually vary from one type of programming language to another: it is not the same to test an imperative programming as it is to test a functional program. An attribute termed *programming language* is added to account for this information.

#### 5.4.1.4.4 Development method (life cycle or methodology)

There are now different methods of development: traditional, knowledge-based, with reuse, etc. These development methods are associated with a particular way of running tests and a technique that is useful one of the methods is not necessarily suitable for another.

The attribute that takes this information to account in the schema is termed *development method*.

Table 5.4 shows the decomposition of the object element.

### 5.4.2 Tactical Level

The information related to the purpose of the test and the part of the software being tested is located at the tactical level. Below, the characteristics of interest associated with each of these elements are proposed.

ELEMENT	ATTRIBUTE	DESCRIPTION
Object	Software architecture	Development paradigm to which the technique is linked
	Software type	Type of software the technique can test
	Programming language	Programming language with which the technique can be used
	Development method	Development method or life cycle to which the technique is linked

Table 5.4: Attributes of the object element.

#### 5.4.2.1 Objective

This information is related to the objectives that are expected to be achieved after testing. As discussed above, the objectives of the tests do not necessarily always have to be the same, although the generic objective will be the evaluation of the software.

The information associated with this element is related to the objective, that is, the quality attribute to be evaluated by the test and with the rigour of the test.

##### 5.4.2.1.1 Quality attribute

Software quality is a multidimensional aspect and is a function of a series of attributes, which will depend on the characteristics of the software system to be built. Thus, when selecting testing techniques, the quality attributes associated with the techniques are important for deciding which techniques can be used to evaluate a given attribute. An attribute is added to the schema termed *quality attribute* to take this into account. Of course, this means that the quality attributes that are of interest for the software to be tested have to have previously been determined.

However, it is not enough merely to include the software quality attribute to completely reflect the user's need for information, the schema attribute must also reflect the metric used. This is logical, as there are usually different metrics associated with one and the same software quality attribute, which can be used by the developer to interpret the values obtained in the evaluation. Accordingly, the consumer will be able to find out both the quality attributes which the technique can be used to evaluate and how the results obtained should be interpreted.

##### 5.4.2.1.2 Rigour of the test

Given a series of software quality attributes, it is very likely that some are more important when running the tests than others (this prioritisation does not emerge automatically, it has to be established by developers because some quality attributes are incompatible). This means that the most important attributes will be the ones on which more emphasis has to be placed, that is, where the test will have to be more exhaustive. Of course, it is not the same to test the safety of a system that controls a nuclear power station as it is to test a system that manages the contents of a warehouse.

It is evident that the degree of rigorousness with which the same attribute is to be evaluated will vary from system to system, giving rise to the schema attribute test *rigour*.

Table 5.5 shows the decomposition of the objective element.

ELEMENT	ATTRIBUTE	DESCRIPTION
Objective	Quality attribute	Quality attribute of the system to be tested
	Rigour	Intensity of the test

Table 5.5: Attributes of the objective element.

#### 5.4.2.2 Scope

The information to which the element refers is related to the definition of the scope of the test, which means setting the test in space and time, specifying which part of the software it affects and the development phase to which it refers.

The relevant information for this element includes: when during development the test will take place, the elements of the code on which it will act and the part of the software that will be tested.

##### 5.4.2.2.1 Time during development when the technique is to be used

This question refers to the temporal location or when during development (phase, activity or task) the test cases will be generated. There are several points during software development when testing techniques are applied: during the requirements phase, where system and acceptance techniques are applied, in the design phase, where integration techniques are used, during coding, where unit testing techniques are used, and during maintenance, where regression techniques are used.

Thus, this information is taken into account by creating an attribute in the schema termed *phase*, which corresponds to the development phase (requirements, design, etc.) in which the technique is used.

##### 5.4.2.2.2 Elements on which the technique acts

Whenever a test is run it will affect a different part of the system: either a module, a set of modules or the entire system.

This information gives rise to the attribute *element* to be tested.

##### 5.4.2.2.3 Part of the system to be tested

The measurement or evaluation of a quality attribute can be reflected in different parts of the software (for example, security can be measured on access controls, on the network software used, etc.) or the hardware.

Thus, the attribute to which this information would give rise would be the *aspect* of the software to be tested. Thus, this attribute refers to the functionality or part of the system to be tested: communications software, connection with hardware, database, etc.

Table 5.6 shows the decomposition of the scope element.

ELEMENT	ATTRIBUTE	DESCRIPTION
Scope	Phase	Time of development when the test is to be run
	Element	Elements of the system on which the test acts
	Aspect	Functionality of the system to be tested

Table 5.6: Attribute of the scope element.

## 5.5 Result of the Construction of the Theoretical Schema

Throughout this chapter, a preliminary version of what will be the characterisation schema for selecting software testing techniques has been built. Of course, there is no guarantee that this schema is complete. This is why this result is considered as just a preliminary hypothesis obtained by deduction based on the characteristics of the testing process and testing techniques.

By way of a summary of this chapter, Table 5.7 presents the composition of the schema: the first column indicates the level to which the attribute belongs, the second column reflects the element to which it refers, the third states the attribute and the fourth contains a brief explanation of the attribute.

## 5.6 Use and Evolution of the Characterisation Schema

The purpose of the characterisation schema is for it to be instantiated for multiple testing techniques to build a repository containing information on as many testing techniques as are of interest for improving testing technique selection. Having presented the preliminary approach of the characterisation schema, procedures associated with the use and evolution of the schema need to be prescribed. Each of these is explained below.

As mentioned in Chapter 4, the repository will be used directly by producers and consumers and indirectly by the librarian. The producers will be able to provide new information for the repository and also to establish new research lines or goals. On the other hand, consumers will be able to select testing techniques for the projects on which they are working. They will also be able to provide feedback on the schema contents and structure, based on the results they get when using the repository and the selected techniques. Finally, the librarian will update the repository on the basis of the information supplied by producers and consumers, taking care to preserve the coherence of the information in the schema. There will, therefore, be five procedures, each associated with the uses of the repository, which will also evolve the schema.

The procedures for using the schema are detailed below.

### 5.6.1 Primary Use of the Repository: Selection

The main objective for which the repository was conceived is to select the testing techniques to be used in the software projects, on the basis of the information it contains. The process

LEVEL	ELEMENT	ATTRIBUTE	DESCRIPTION
Tactical	Objective	Quality attribute	Quality attribute of the system to be tested
		Rigour	Intensity of the test
	Scope	Phase	Time of development at which the test is to be run
		Element	Elements of the system on which the test acts
Operational	Agents	Aspect	Functionality of the system to be tested
		Experience	Experience required to use the technique
		Knowledge	Knowledge required to be able to apply the technique
		Tools	Available tools that ease the use of the technique
	Technique	Comprehensibility	Whether or not the technique is easy to understand
		Cost of application	How much effort is needed to apply the technique
		Sources of information	Where to find information about the technique
		Dependencies	Relationships of one technique with another
	Results	Repeatability	Whether two people generate the same test cases
		Adequacy criterion	Test case generation and stopping rule of the technique
		Completeness	Coverage provided by the set of test cases
		Cost of execution	Time the technique takes to execute the test set
	Object	Type of defects	Type of defects the technique helps to discover
		Effectiveness	Capability of the set of test cases to detect defects
		Correctness	Test cases to be deleted from the set
		Adequacy degree	Extent to which the adequacy criterion is achieved
		Software architecture	Development paradigm to which the technique is linked
		Software type	Type of software the technique can test
		Programming language	Programming language with which the technique can be used
		Development method	Development method or life cycle to which the technique is linked

Table 5.7: Theoretical schema.

associated with the use of the repository for selection purposes is detailed in this section.

Two concepts need to be introduced in order to explain the process to be followed to select testing techniques. These are:

- *Bounded variables.* These are the schema attributes whose value is imposed by the project. Their value cannot be changed during the selection process. For example, the development method used for the software project, the type of software under development or the software quality attributes for evaluation are usually pre-established for the project within which testing techniques are being selected, and the consumer responsible for making the selection cannot change their value at his or her convenience.
- *Free variables.* These are the schema attributes whose value is not imposed by the project and which, therefore, can vary depending on current selection needs and/or availability. For example, the characteristics of the people who are to apply the testing techniques (if they have not been determined by the time the selection is made and the person making the selection is given the freedom to choose) or the tools that can be used (if the company has or is prepared to buy the tools) might not be pre-established for the project within which testing techniques are being selected, and, therefore, the consumer responsible for making the selection will be able to change their value at will.

Obviously, the more variables there are that are free (and, therefore, the fewer there are that are bounded) the better, as this will mean that the consumer has more freedom when selecting techniques. By way of an example, suppose a consumer wants to select the most effective technique, but the candidate technique calls for a lot of effort of application (time) and the required time is not available, this person can opt to choose a less effective technique whose application calls for less effort. The result of the selection will depend on whether the attributes effort of application and effectiveness are free or bounded variables. If the two variables are free, it will be possible to select either of the two techniques. If only one of the two variables were bounded, the most effective technique or the technique taking less time to apply (depending on which was the bounded variable) would have been chosen. If both attributes were bounded, and it was not possible to change the value of either, it would have been impossible to choose either of the two techniques.

The steps of the process for the consumer to use the schema for selection purposes are established on the basis of the above definitions as follows:

1. **Determination of the bounded variables.** The objective of this step is to identify the constraints imposed by the software project for selection purposes. It is reflected in Figure 5.7.

- *Input.* Empty characterisation schema, set of software project characteris-



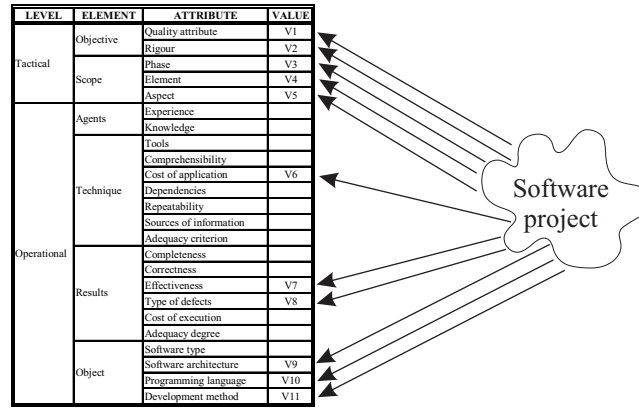


Figure 5.7: Determination of the bounded variables in the schema according to the project characteristics.

tics.

- *Process.* The consumer responsible for making the selection will identify whether there is any value within the project for each characterisation schema attribute. If so, the attribute is a bounded variable and is assigned a value. Otherwise, the attribute is a free variable and does not have any value yet.
- *Output.* Characterisation of the software project constraints as a set of bounded variables and their associated values.

**2. Pre-selection of a set of techniques depending on the value of the bounded variables.** The purpose of this step is to identify the set of techniques that satisfy the constraints imposed by the project. This step is reflected in Figure 5.8.

- *Input.* Set of bounded variables with their associated value, repository.
- *Process.* Search the repository for techniques whose attribute values for the bounded variables coincide with the value of the bounded variables. This step will be performed by the repository or the consumer responsible for selection depending on whether or not an automated search procedure is available.
- *Output.* Set of testing techniques that satisfy the constraints imposed by the bounded variables.

**3. If the set of pre-selected techniques is empty, relax some of the bounded variables and go to step 2.** If this point is reached, it means that it is impossible to find a testing technique in the repository that satisfies the project constraints. However, if the software is to be tested, there has to be a technique that meets the project constraints. The purpose of this step is to relax some of the project constraints so that it is possible to select a technique. Figure 5.9 reflects this step.

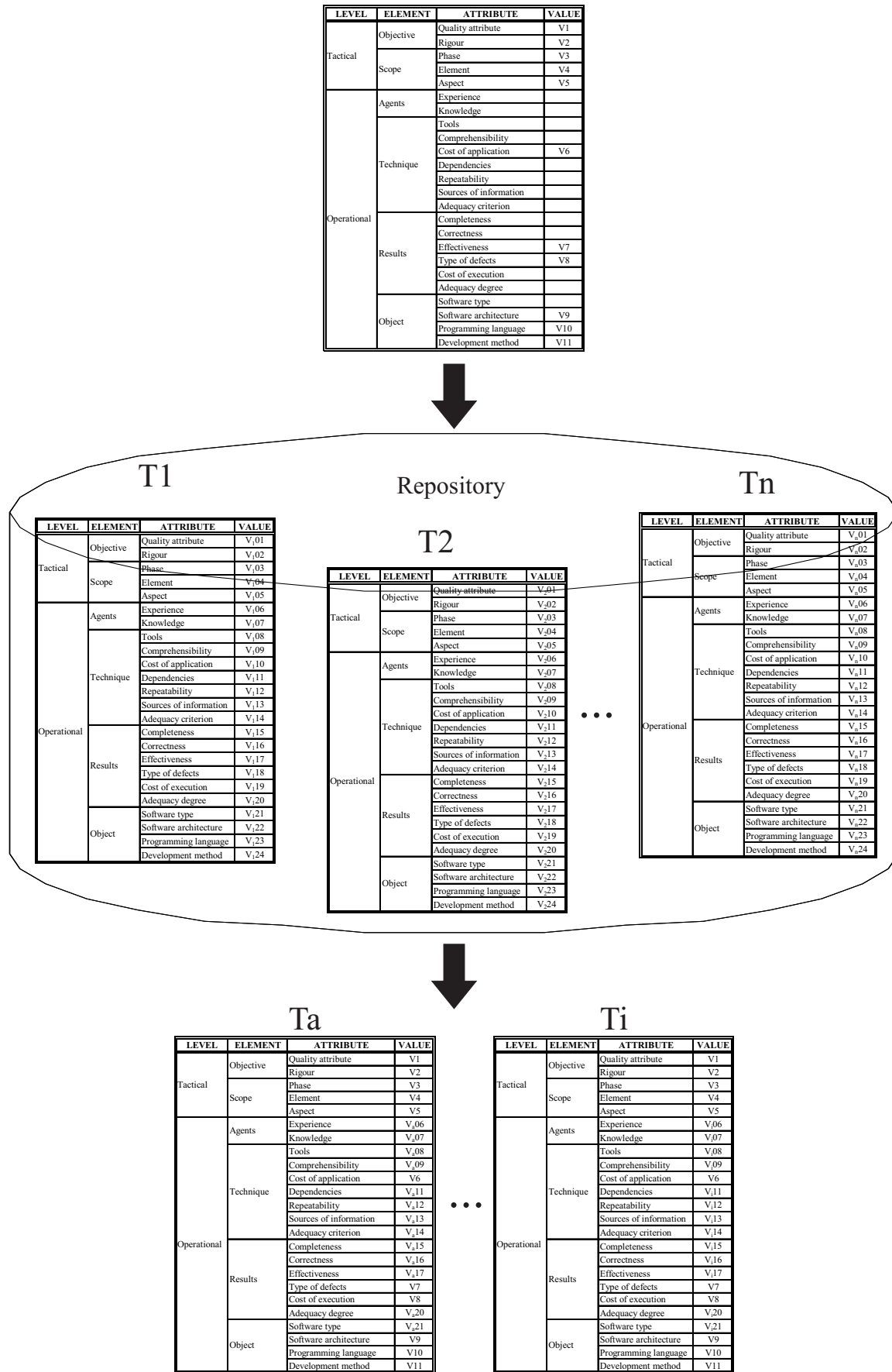


Figure 5.8: Pre-selection of techniques from the repository considering the bounded variables.

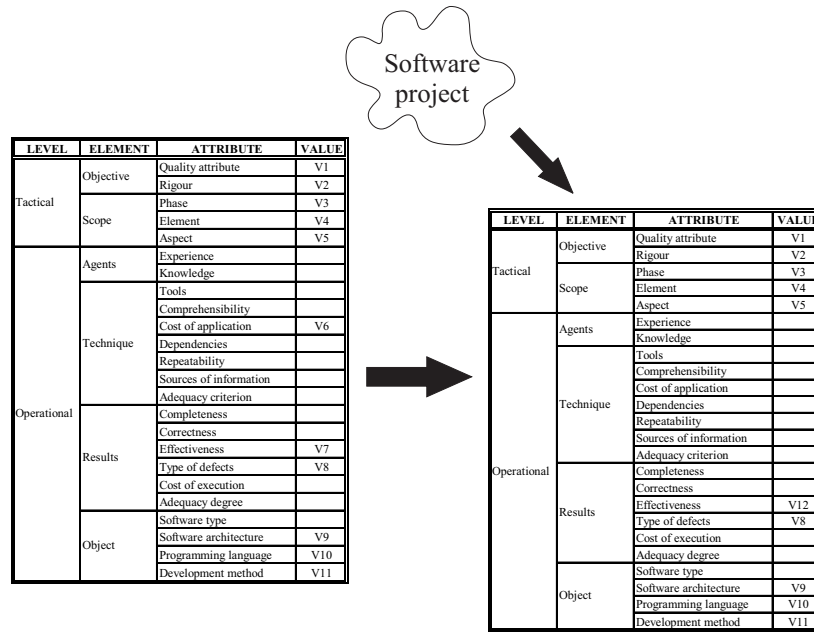


Figure 5.9: Relaxation of some bounded variables. The value of some variables is changed in order to allow the selection of a testing technique.

- *Input.* Set of bounded variables and their values, project characteristics.
  - *Process.* The consumer responsible for making the selection has to reassign the value to as many bounded variables as possible to assure that it is possible to select at least one testing technique.
  - *Output.* Modified set of bounded variables and their values.
4. **If the set of techniques obtained in step 2 is not empty, sift the values of the free variables until the best-suited techniques for selection have been identified..** The objective of this step is to make a comparative study of the techniques that satisfy the project constraints in order to select the best suited of the possible techniques. Figure 5.10 reflects this step.

- *Input.* Set of pre-selected techniques, set of software project characteristics.
- *Process.* The consumer responsible for the selection has to compare each of the pre-selected techniques on the basis of the values of the free variables and select the techniques considered best suited for the characteristics of the project in question.
- *Output.* Selected technique(s) for the project in question.

### 5.6.2 Secondary Uses of the Repository: Evolution

Although the primary objective of the repository is to ease the selection of testing techniques, it is also true that it would be useless without relevant, useful contents. In this section, the

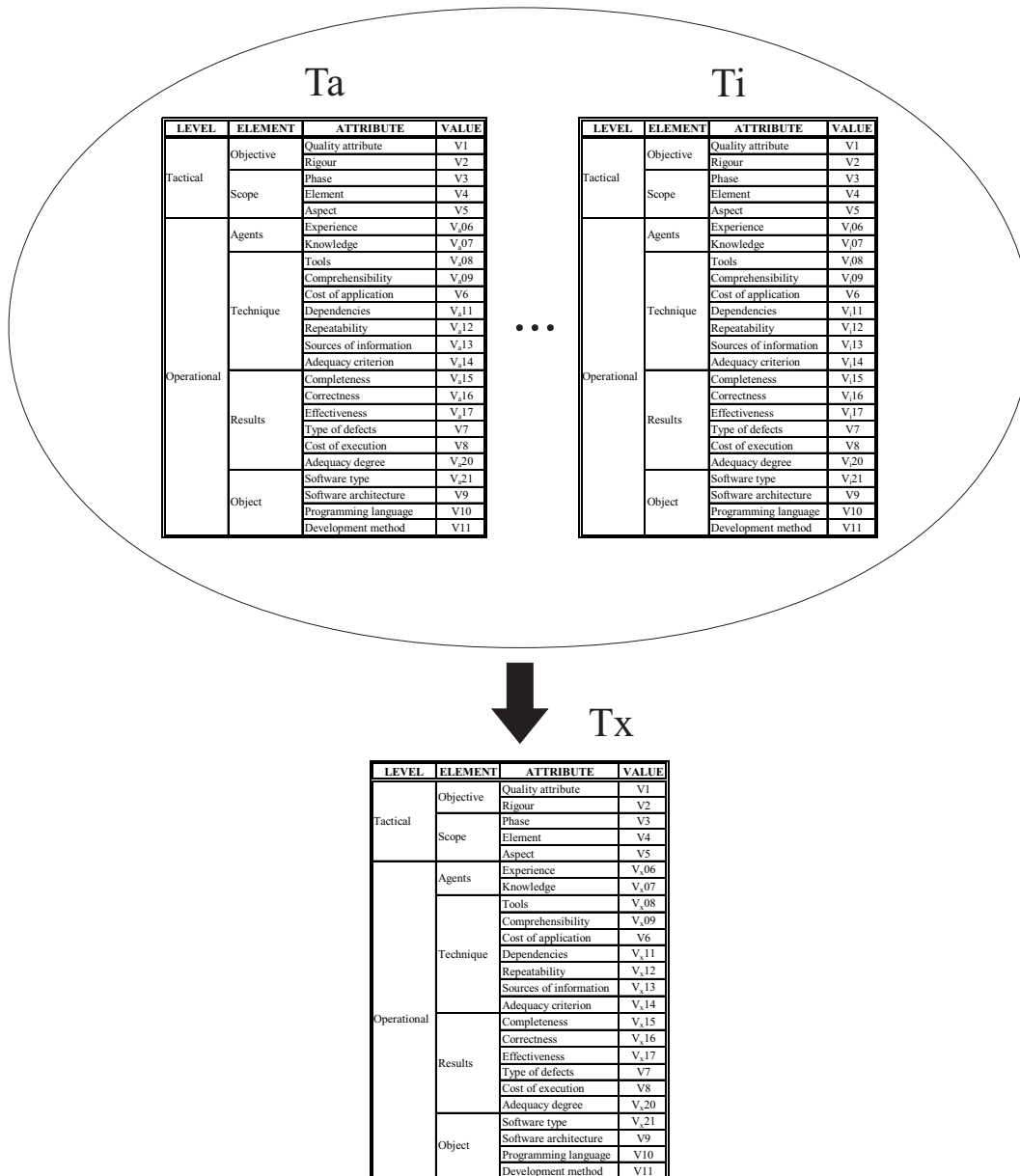


Figure 5.10: Final selection of techniques. The pre-selected techniques are studies in order to determine which is the most convenient for the current project.

procedures for both producers and the consumers associated with the use of the repository for the purposes of evolution are explained in detail.

There are several possible types of repository evolution, as shown in Figure 5.11.

- Entry of new instantiated techniques into (or deletion of existing techniques from) the repository.
- Entry of missing information about techniques already in the repository.
- Update of technique information already in the repository.

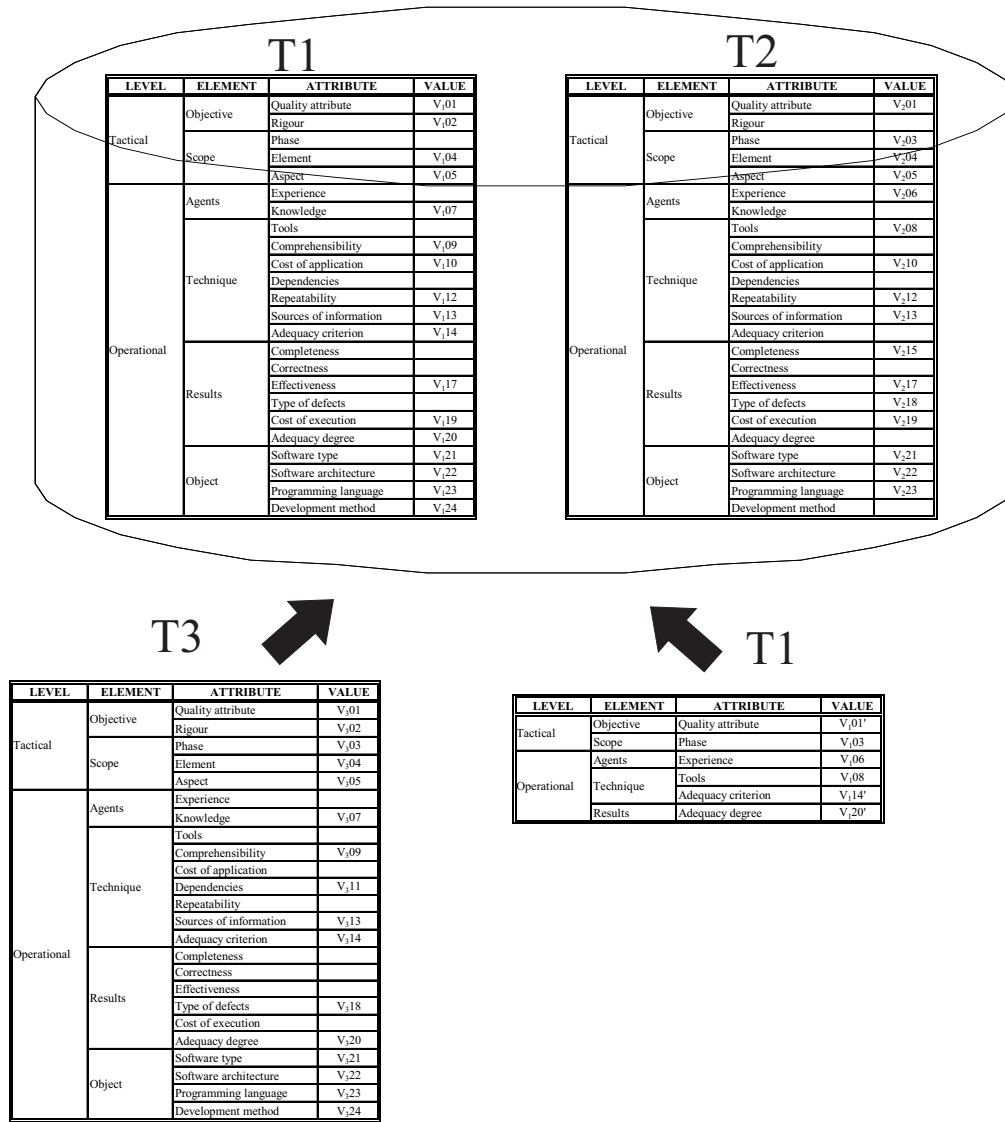


Figure 5.11: Characterisation schema evolution. New techniques can be added to the repository, or new information can be provided.

- Entry of changes to the structure of the characterisation schema (addition or deletion of schema attributes).

All the changes to the repository due to schema evolution will be made by the librarian, who will be responsible for preserving the coherence of the information in the repository at all times. However, the librarian will receive information from the following sources.

### 5.6.2.1 Post-Selection Evolution

After selection, statistics will be collected (by means of a questionnaire) about the attributes of interest for the consumer (both attributes that were already in the schema and attributes that did not appear), the number of techniques examined, the technique(s) selected and

whether any technique was missing from the schema. This statistics collection procedure will make it possible for the consumer to provide feedback on the structure of the characterisation schema and on the repository techniques.

The statistics on the attributes of interest for the consumer can be used to update the schema structure. This is done either by adding the new attributes suggested by the consumer or removing attributes not consulted after a given period of time (deletion is not prescriptive, it may be decided to follow the heuristic of not deleting information).

The statistics on the techniques of interest for the consumer can be used to update the techniques contained in the repository. This is done either by adding any new techniques that the consumer thought were missing or removing techniques not selected after a given time period (again, deletion is not prescriptive, it may be decided to follow the heuristic of not deleting information).

Post-selection evolution will benefit consumers in that it will assure that the techniques contained in the repository and the information about the techniques are the best suited for the projects to be carried out at the company.

#### 5.6.2.2 Post-Use Evolution

Once the technique has been used, information will be gathered from the consumer (again by means of a questionnaire) about the value of the attributes of the selected technique(s). The consumer will have gained this information from experience in applying the technique. The information gathered can be: (1) the same as the information that the repository already contains, in which case the repository will remain unchanged; (2) new (which means that the values of the attributes related to the information in question were empty), in which case the librarian will enter the information into the repository; and (3) already in the repository and different, which means that the librarian will have to refine the gathered and the existing information, before making a decision on the new value of the attributes concerned. The goal of post-use evolution is to improve the quality of the information contained in the repository to make it as complete and accurate as possible and make it possible to select the best-suited techniques.

#### 5.6.2.3 Evolution by Producer Feed

There are two possible types of information supplied by producers for the repository:

- *On demand by the librarian.* Every time the librarian detects that a consumer wants a new technique (evolution due to step 1 of selection), a producer will be asked to supply all the information it now has about the technique. Normally, this information will be supplied as pointers to sources of information, such as research papers, books, Web pages or even information directly from the mouth of the producer. Similarly, if the structure of the schema is modified by adding new attributes (evolution again due to step 1 of selection), the librarian will ask the

producer for the respective information, which will be supplied in the same format as above.

- *On the initiative of the producer.* Similarly, as a result of research activity, the producer can provide the librarian with information to modify the contents of the repository, either by supplying information on new testing techniques or updating information on existing techniques or suggesting changes to the schema structure.

### 5.6.3 Secondary Uses of the Repository: Research

This section deals with the use of the repository by the producer for research purposes. Basically, this use involves the producer periodically consulting the contents of the repository for the purpose of undertaking new research concerning both information related to already existing techniques and research into new techniques that can fill any gaps now left by the existing testing techniques.

### 5.6.4 The Role of the Librarian in the Repository

As it is planned to deploy the repository on the basis of multiple instantiations of the characterisation schema for different techniques by producers and its evolution by having both producers and consumers update its contents, there is a problem. As different people (different producers and consumers) enter the information into the repository, these people could enter any information they like in a disorderly and conflicting manner, leading to chaos. Indeed, the producers and consumers could misinterpret schema attributes when entering the information into the repository and, therefore, enter inappropriate information into the repository. On the other hand, when updating existing information, producers and consumers could misinterpret this information or not know how to combine it with the information they have, and the update could lead to a loss of important information or the entry of incorrect information.

There should be a librarian (systems type) and a content knowledge expert. These people may be the same person, who will analyse the information gathered from different sources to instantiate the testing techniques. This person will also be responsible for interpreting the information supplied by producers and consumers to assure that the contents of the repository are reliable. As this person will be perfectly acquainted with the schema, he or she will not make mistakes due to misinterpretation.

Figure 5.12 gives a summary of the uses presented in this section.

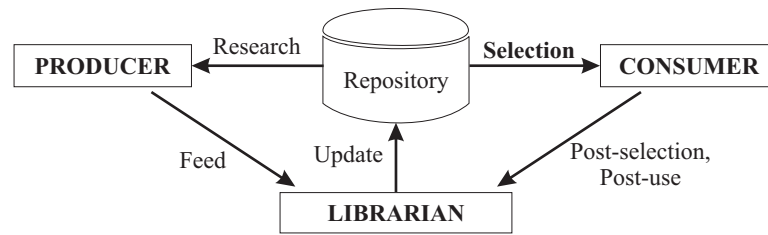


Figure 5.12: Repository uses.



## Chapter 6

# Second Generative Iteration: Inductive Empirical Schema

At this point, we have a characterisation schema that is the fruit of the personal reflection of the investigator to determine the information relevant for selecting testing techniques in a given project. In the absence of a solid theory on software testing upon which to build the characterisation schema, this first schema was obtained by studying the state of the art of testing techniques (books and research papers), as well as looking for the differences and similarities between different techniques. However, it is necessary to find out what information the people directly related to software testing (developers and researchers) think is relevant for selection purposes and thus complement the views held by different types of subjects of the problem. That is, there is no guarantee that all the information that developers and researchers consider to be relevant for selecting testing techniques is now present in the theoretical schema. As explained in Chapter 4, the aim of consulting consumers, or the people responsible for selection, and producers, or the people responsible for fabricating new testing techniques, is to get a clearer idea of what information a consumer or producer considers to be relevant for selection purposes.

As indicated in Chapter 4, the objective of this phase is to build a characterisation schema of software testing techniques that reflects both the needs of consumers when selecting testing techniques, as well as the information that producers believe to be necessary to fully reflect the nature of a testing technique. The tasks to be carried out to get the empirical schema include sending out two different questionnaires to respondents: a questionnaire on the basis of which the information that the consumer believes to be relevant for selection is gathered and another that is used to gather the information that the producer believes to be necessary to define a testing technique. This information is then analysed to produce a characterisation schema that will reflect the opinions of both consumers and producers about the selection problem.

The empirical schema has been built incrementally, as described in Chapter 4. This means that each time information was gathered from a new person (producer or consumer), a new version of the empirical schema was created. However, the process of building the empirical

schema is presented sequentially in this chapter, which means it is built once for the total set of respondents. This was done to make it easier to read.

## 6.1 Data Collection

The co-operation of a series of respondents (term that will be used hereinafter to refer to the people, both producers and consumers, who have participated in building the empirical schema) will be required to complete the empirical schema. These people will be given a form, which will be different depending on the role they play: producer or consumer. Both forms are divided into two parts. The first part requests personal information that will be used to class the subject within the set of respondents; the second part varies depending on the role. For consumers, the second part of the questionnaire sets out a situation on the basis of which respondents should state, in the form of questions, the information they would like to know about the testing techniques for the purposes of selection. For producers, they are asked directly what information they believe to be necessary to fully define the nature of a testing technique for selection purposes.

### 6.1.1 Characterisation of the Population of Respondents

One of the key tasks for designing the empirical schema was the selection of the respondents. The characteristics of the people involved in the construction of the empirical schema can have a significant influence on the resulting schema. The people involved should be as heterogeneous as possible to assure that the schema does not reflect a unilateral viewpoint. For this purpose, an attempt was made to include respondents with a wide variety of characteristics.

However, the search for volunteers was an arduous task. The type of respondents required for this study (company developers and company and university researchers) are very busy people who have hardly any time for this sort of occupations. Even so, fourteen respondents from a range of fields, with varying experience and of different nationality participated. As the set of participant subjects had to be as heterogeneous as possible, we looked for people who played different roles in the testing area. Each of the participant subjects was described for the purpose of examining schema coverage with respect to the type of respondents. The parameters used to describe the respondents are:

- *Current position.* Description of the position now held by the respondents within the company or institution for which they work. This parameter is necessary to find out how the respondents are now related to software testing.
- *Years of service.* How many years the respondents have held their current position. This parameter is necessary to find out how experienced the subjects are in the tasks mentioned in the above parameter.

- *Company/Institution.* Company or institution for which the respondents work. This parameter is necessary, as it provides information about the branch of the software industry in which the respondents work.
- *Qualifications.* Qualifications of the respondents. This parameter is necessary to give an idea of what theoretical knowledge the respondents have in the field of software.
- *Experience in Testing.* Experience of respondents in software testing. Thanks to this parameter, it is possible to ascertain both what theoretical knowledge and practical experience the respondents have in software testing.

Also, depending on the role played by the respondents (producer or consumer), they were asked respectively:

- *Area of interest in software testing.* This parameter can be used to find out the area or aspect of testing in which the producer is specialised. This is important as people from different areas can provide a variety of information.
- *Experience in software development.* How many years the respondents have worked in the field of software development. This parameter is necessary to find out how experienced the subjects are in software development.

### 6.1.2 Survey Coverage

This section discusses the different aspects of the characterisation that have been chosen for the coverage study.

With regard to the companies at which the respondents work, Figure 6.1 shows that there are two respondents who are faculty members, two respondents who work for centres associated to universities and twelve respondents who belong to different software companies (both small, medium-sized and large enterprises).

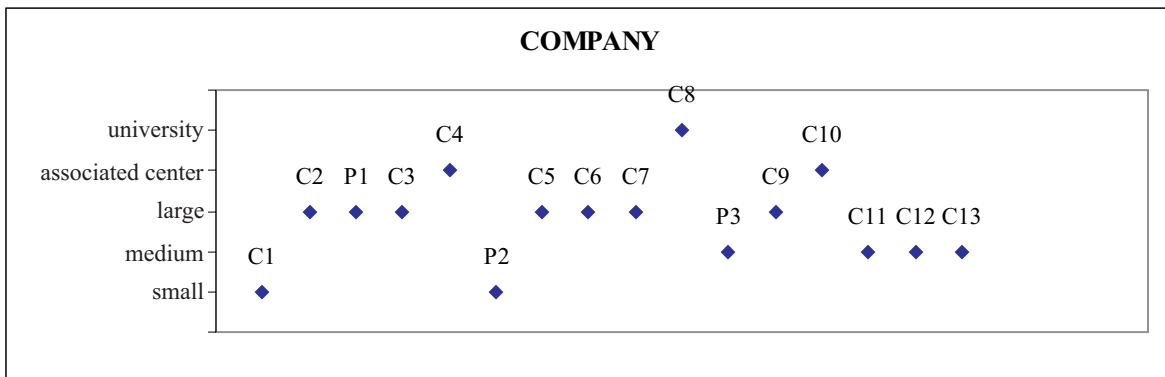


Figure 6.1: Company or institution at which the respondents work.

As shown in Figure 6.2, the group includes two university professors, five software department managers (four from development departments and one from a research department), two project managers, six developers and one scientist. The years of service is variable. The experience of the group in software testing also varies and, interestingly, in accordance with the positions held by the respondents. For consumers: the project managers are experienced in planning the testing process within the project, as well as in system testing and acceptance; the software engineers are experienced in unitary tests and some in integration tests; the professor is experienced in test planning. The producers are experienced in research.

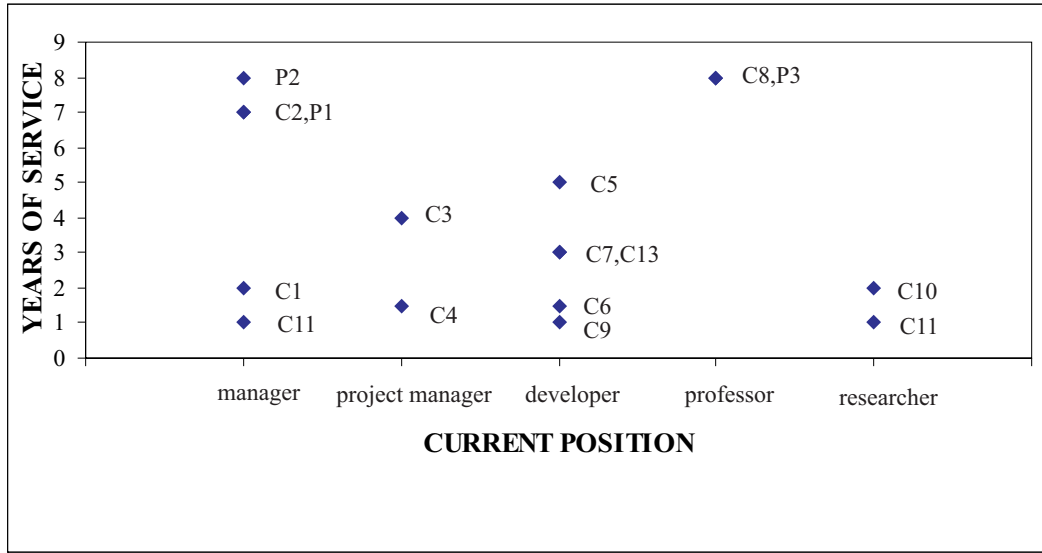


Figure 6.2: Position held by each of the respondents.

As shown in Figure 6.3, they are all in possession of qualifications ranging from bachelor degrees in computer science, through master degrees in software engineering or computer science, to doctorates in computer science, except two, one of whom is a bachelor in physics (now taking a doctorate in computer science) and the other is a master in electrical engineering.

The experience of the consumers in software development is shown in Figure 6.4. It varies from 3 to 22 years, the mean being about 12 years.

The areas of interest within software testing for producers are: the study of technique applicability for one and new techniques for the other two.

One possible question is how to deal with respondents who class themselves as both producers and consumers. Of the respondents, there are two who fall within this category. They were given two forms to fill in on the condition that each questionnaire be completed on different days (if possible a couple of days apart) so that the result of the first questionnaire does not have an influence on the second. In this particular case, P1 is C2 and P3 is C8. Interestingly, both respondents gave different responses on the producer and consumer questionnaires they returned.

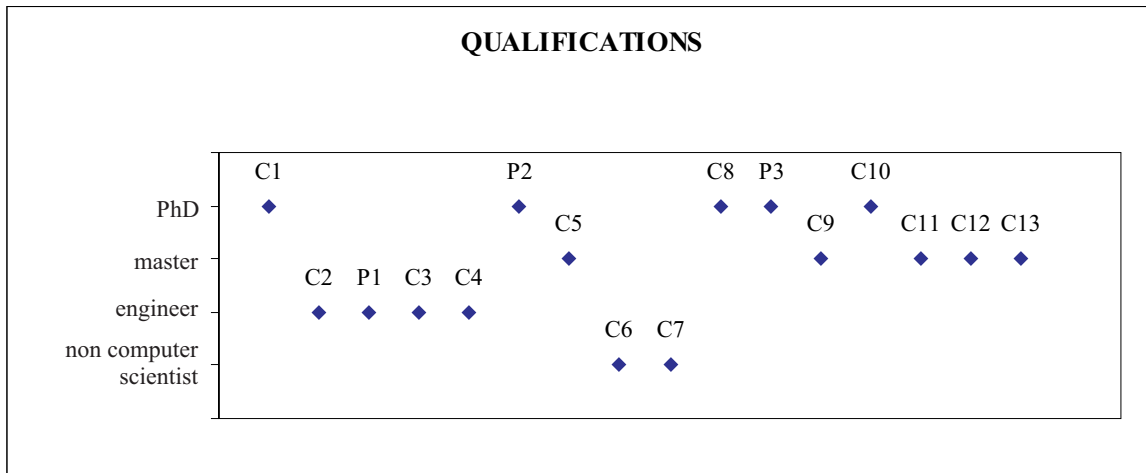


Figure 6.3: Qualifications of each of the respondents.

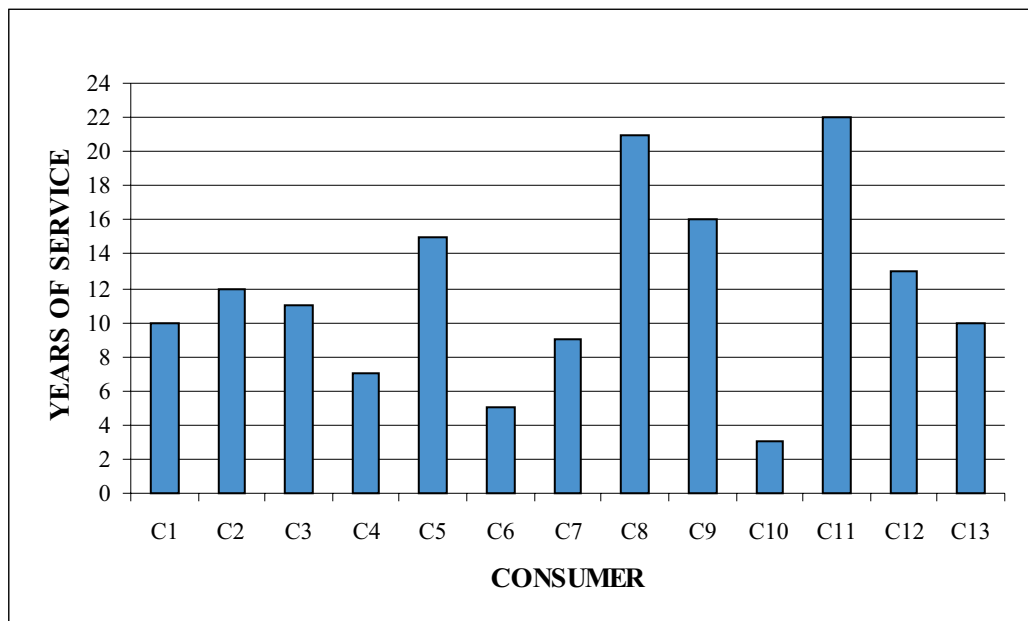


Figure 6.4: Consumer experience in software development.

### 6.1.3 Form Building

It was thought that the best way of understanding consumer needs is for consumers to write a list containing the questions they usually ask (or would ask if they were asked to make a rigorous selection) themselves about the testing technique when selecting techniques for use in a given project. However, taking into account the issues of characterisation discussed above, it was essential for the form delivered to consumers to include, apart from the information relevant for selection, their characterisation.

Therefore, the composition of the form delivered to consumers, which is shown in Appendix D.1, is as follows. First, information is requested that will help to describe the

consumer. Then a situation is set out upon the basis of which the generic question of what information the consumer would like to know about the testing technique to decide whether or not to apply it in a given software project is asked. The goal is to have consumers write as detailed a list of questions as possible about the information they believe to be relevant for selection purposes.

Consumers were given two possibilities for completing the questionnaire. They could fill the questionnaire either in private or with the investigator (in which case it was the investigator who took the notes). There was no pre-established time limit for filling in the forms (this was done at the respondent's convenience). A limit was placed on questions related to the internal operation of the technique, as one of the premises of this piece of research is that it should not be necessary to be acquainted with the internal operation of the technique for selection purposes.

As far as producers are concerned, it was thought that the simplest thing to be able to easily understand their knowledge was for them to directly make a list of the information they believe to be relevant for selecting testing techniques.

However, as with consumers, it is also important gather characterisation information at this point. Therefore, the form used for this purpose, which appears in Appendix D.1, will again be divided into two parts: one which gathers information on the sample and another which sets out the selection problem for producers and asks them to indicate what they believe to be relevant information.

Producers filled in the form in the same way as consumers. They were given the possibility of completing the questionnaire in private or with the investigator, and they were not allotted a pre-established time limit either.

## 6.2 Data Analysis

Again, it is important to stress that the empirical schema was really built incrementally, creating a new version of the schema every time a new form was received. However, this chapter presents the empirical schema as if it was obtained on all the forms collected in order make it easier to read and understand.

Accordingly, the tasks to be performed every time a respondent completed a new form as set out in Chapter 4 are completed just once here. These tasks as specified in Chapter 4 are:

- *Update reference set i-1.* This task is referred to in this chapter as *creation of the reference set* and is described in Section 6.2.1. Any equivalent information supplied by the respondents is grouped in this step. However, and as follows from the form that appears in Appendix D.1, there is one type of information that the respondents were not allowed to mention, and this is information related to the internal operation of the technique. Nevertheless, it was found that some respondents did mention this and it was decided to remove any such inadmissible information. This information was deleted because, as mentioned earlier, the characterisation schema is based on

the idea that it is not necessary to understand the internal operation of a testing technique for selection purposes.

- *Analysis of reference set i.* This task is referred to in this chapter as *analysis of the reference set* and is described in Section 6.2.2. Having grouped the equivalent information in the questions, the questions are analysed, adding the appropriate information to the schema to assure that the schema responds to all the questions.

### 6.2.1 Creation of the Reference Set

Table 6.1, Table 6.2, Table 6.3, Table 6.4 and Table 6.5 reflect what was termed in Chapter 4 *reference set* and which encompasses the information of interest supplied by producers and consumers. The information is presented grouped around questions, depending on their meaning, and any inadmissible information was removed. The information reflected in these tables is as follows: column 1 indicates the number assigned to the question and the column 2 formulates the question, also indicating its source. The source is indicated by means of a letter followed by a number, a colon and another number. The letter indicates whether the question was raised by a producer (P) or a consumer (C); the first number identifies the individual who raised the question (from 01 to 13 for consumers and from 01 to 03 for producers); the second number identifies the information supplied (question if it is a consumer, statement if it is a producer) by the respondent. This way, the information a respondent provided can be easily identified in Appendix D.2, where the forms collected from the subjects appear.

Some of the questions raised by the respondents have not been taken into account for several reasons. Table 6.6 shows the rejected questions.

As shown in Table 6.6, the questions not used vary from questions that refer to the internal operation of the technique, through questions that are too abstract to be analysed and get reliable results, to questions that are not related to the specific part of testing dealt with in this research.

### 6.2.2 Analysis of the Reference Set

The strategy followed to analyse the reference set was to analyse each of the questions of which it was composed one by one to examine how they should be addressed in the schema. This is the task discussed in this section. First, a high-level classification will be made of the information, identifying the testing process levels and the elements to which they belong. Then each piece of information will be examined in detail. Section 6.2.2.1 addresses the levels identified in the empirical schema, Section 6.2.2.2 discusses the elements identified for the empirical schema and, finally, Section 6.2.2.3 studies how the questions in the reference set should be added to the empirical schema.

#### 6.2.2.1 Levels of the Empirical Schema

The grouping of the information collected from the respondents is shown in Table 6.7.

N	QUESTION AND SOURCE
1	<p>Can the technique be used with any programming language?</p> <p>C01:01 Does it depend on the programming language of the code to be tested?</p> <p>C03:04 What programming language can it be used with?</p> <p>C06:16 What programming languages does the technique work for?</p> <p>C11:04 What languages does the technique support?</p> <p>C13:03 What is the environment where it can be used?</p>
2	<p>Are there tools that ease the use of the technique?</p> <p>C01:02 Are there tools to support it?</p> <p>C02:10 What are the available tools?</p> <p>P01:06 If it can be automated, at least partially</p> <p>C03:02 What are the resources needed (machines, people, etc.)?</p> <p>P02:01 Automation of the technique</p> <p>C06:05 Are there tools (executables or documents) to guide the user in using the technique?</p> <p>C08:05 What is the support (tool) the technique has?</p> <p>P03:05 Automation support with tools</p> <p>C09:01 Is this technique automatic or manual?</p> <p>C10:03 Is there tool support?</p> <p>C10:04 Does it allow automation?</p> <p>C11:06 Can the technique be used to perform automated testing?</p> <p>C13:06 What tools can be used to support the technique?</p>
3	<p>Has the technique ever been used before?</p> <p>C01:03 Are there users or projects of reference?</p> <p>C02:03 Has somebody used it?</p> <p>C08:10 Has the technique ever been used before?</p> <p>C08:14 Who has used the technique? If nobody, why?</p>
4	<p>Can the technique be used with any development method?</p> <p>C01:04 Is it bounded to a life-cycle or a methodology?</p> <p>C02:05 For what life-cycle can it be used?</p> <p>P01:13 Type of development method the technique can be used with</p> <p>C07:01 What software development process can the technique be used with? (i.e. incremental, XP, spiral, waterfall, etc.)</p> <p>C13:03 What is the environment where it can be used?</p>
5	<p>At what moment of the development can the technique be used?</p> <p>C01:05 In what phase of the development process can it be used?</p> <p>C06:09 How early in the development cycle can a test case be constructed?</p> <p>C09:02 Is this technique available to be used by all levels of testing for this product?</p> <p>C09:05 Does this technique provide regression testing?</p> <p>C12:01 What types of testing are supported by each technique? (unit testing, integration testing, etc.)</p> <p>C12:06 Which are the development activities to which one can apply the technique?</p> <p>C13:05 When can the technique be applied?</p>

Table 6.1: Content of the reference set (1/5).



N	QUESTION AND SOURCE
6	<p>Do people need specific knowledge in order to use the technique?</p> <p>C01:06 Do people have to be trained?</p> <p>C01:13 Are people with experience or special knowledge required?</p> <p>C01:17 What type of people can use it?</p> <p>C02:01 What is the knowledge people should have in order to use it?</p> <p>C02:07 What is the estimate time for training people who will use it?</p> <p>C02:08 What knowledge should people have in order to use it?</p> <p>P01:04 If it can be used by average developers</p> <p>P01:10 Knowledge of the people that have to use it</p> <p>C03:02 What are the resources needed (machines, people, etc.)?</p> <p>C08:06 Will people have to be trained using the tool?</p> <p>C08:08 Does the use of the tool imply a cultural change?</p> <p>P03:02 Whether the technique is usable for mid-developers</p> <p>C10:07 Do testers need to have special knowledge to use it?</p>
7	<p>Is the technique experimental or has it been used before?</p> <p>C01:07 Is it experimental or has it been tested?</p> <p>C08:11 Has the technique been evaluated or validated?</p>
8	<p>Can the technique be used with any type of software?</p> <p>C01:08 In what type of projects has it been used? (real-time, management, etc.)</p> <p>C02:04 What type of software it can be used with?</p> <p>P01:09 Type of software the technique can be used with</p> <p>C03:05 Is it adequate for the type of software that is being developed?</p> <p>C04:01 For what kind of software product is this technique more adequate?</p> <p>C08:01 What type of software is the technique suitable for?</p>
9	<p>What is the effectiveness of the technique?</p> <p>C01:09 What is its effectiveness? (number of errors found)</p> <p>P01:02 Which are the benefits the technique should produce (number of errors)</p> <p>C03:01 What are its results (objectives)?</p> <p>C04:03 What are the results this technique has got? (Is it effective?)</p> <p>C08:12 Is it cost-effective?</p> <p>C13:04 How effective is this technique generally?</p>
10	<p>What is the effort in terms of time and resources to apply the technique?</p> <p>C01:10 How long does it take test case generation?</p> <p>C02:06 What is the estimate time when test cases generation?</p> <p>P01:03 Resources the technique uses</p> <p>P01:15 Time that will be needed in order to apply the technique</p> <p>C04:04 How much effort do I need to apply this technique?</p> <p>C06:02 How much time does it take to produce a test case?</p> <p>C08:16 How easy is it to apply?</p> <p>P03:06 Application cost, in terms of time</p> <p>C11:08 If the technique instruments the software under test, what is the overhead for the instrumentation? (memory, cpu time...)</p> <p>C13:01 How easy is it to use?</p> <p>C13:02 What is the cost of using it?</p>

Table 6.2: Content of the reference set (2/5).

N	QUESTION AND SOURCE
11	<p>How long does it take to obtain the test cases?</p> <p>C01:10 How long does it take test case generation?</p> <p>C02:06 What is the estimate time when test cases generation?</p> <p>C06:02 How long does it take to implement and re-implement a test case?</p> <p>C08:15 How straightforward is producing the test cases?</p>
12	<p>What is the formality level of the technique?</p> <p>C01:11 Is it objective?</p> <p>C05:02 Does the technique create repeatable tests?</p>
13	<p>Does the use of the technique imply the use of other techniques?</p> <p>C01:12 Is it enough using that technique or has to be complemented with other(s)?</p> <p>P01:08 If it can be complemented with other techniques</p> <p>P02:07 Whether you are duplicating effort when using several techniques</p> <p>C10:09 How does it compare to other similar techniques?</p>
14	<p>Does the technique require experienced people?</p> <p>C01:13 Are people with experience or special knowledge required?</p> <p>C01:17 What type of people can use it?</p> <p>C03:02 What are the resources needed (machines, people, etc.)?</p> <p>C08:04 What is the level of experience of the people with the technique?</p>
15	<p>How many test cases does it generate?</p> <p>C01:14 Does it generate a minimum set of test cases?</p>
16	<p>What software objects does it test?</p> <p>C01:15 Can it be used n software components, or the whole system is needed?</p> <p>C01:18 Can it be used for testing hardware?</p> <p>C04:02 What is possible to test with this technique?</p> <p>C05:01 Does the technique test the requirements?</p> <p>C05:08 Does the technique test the current system design and methodology?</p> <p>C07:03 What dependencies this product has on other products in order to allow for staging and integration testing?</p>
17	<p>What type of defects does it look for?</p> <p>C01:16 What type of errors does it look for?</p> <p>C05:03 Does the technique test for logic problems?</p> <p>C05:04 Does the technique test for application errors?</p> <p>C05:05 Does the technique check for grammatical/syntax errors?</p>
18	<p>Does it need inputs in a special format?</p> <p>C01:19 Does it require requirements/design/etc. in any special format?</p> <p>C07:02 What should be the format of the development documents produced?</p> <p>C10:05 How much do you have to change the development previous to testing to apply this technique?</p>
19	<p>What is the documentation available?</p> <p>C02:02 What is the available documentation?</p> <p>C06:05 Are there tools (executables or documents) to guide the user in using the technique?</p> <p>C10:08 What kind of support exists for introducing the technique (training, etc)?</p>
20	<p>Are the people who have used the technique satisfied with it?</p> <p>C02:03 Are people who have used it satisfied with it? Will they use it again?</p>
21	<p>What have been the results (costs and benefits) of using the technique?</p> <p>C02:03 What have been the advantages of using the technique?</p> <p>C02:03 What has been the return on investment of the technique?</p> <p>C06:06 What kind or results have you seen when used on a project like mine?</p>

Table 6.3: Content of the reference set (3/5).

N	QUESTION AND SOURCE
22	Which tools have been used with the technique? C02:03 Have tools been used with the technique? Which ones? How was the experience?
23	What part of the technique do tools automate? C02:11 What part of the techniques do available tools automates?
24	What is the cost of buying and maintaining the tools? C02:12 What is the acquisition and maintenance cost of the tools?
25	What are the restrictions of the tool? C02:13 What hardware and operating system does the technique work in? C02:14 What programming language and dialect does the technique work with? C08:07 What is the platform (software and hardware) the tool needs? C11:05 What platform does the technique run on, and what are its resource requirements (memory, disk, cpu time...)?
26	What type of support does the tool have? C02:15 Does the technique have support? (hot-line) C08:09 What is the existing support for the tool?
27	Can the technique be used for software systems of any size? C02:16 What is its scalability? Can it be used for big and small systems?
28	What quality attributes can be tested with the technique? P01:01 What the technique measures C04:02 What is possible to test with this technique? C12:02 Which of the following quality attributes are testable by the technique? Reliability, safety, reusability, maintainability, scalability C12:04 What metrics are used to evaluate the quality attributes listed in question C12:02?
29	Regression cost P01:14 Easiness for regression
30	What aspects of the software does the technique test? C04:02 What is possible to test with this technique? C05:09 Does it test for functionality? C06:04 What aspects of the software is the technique aimed at verifying? C06:15 Is the methodology capable of generating test cases for GUIs? C09:06 Does this technique provide functional testing? C09:07 Does this technique test a user's interaction to this product? C09:09 Does this technique help test the scalability and load balancing for the product? C10:02 Does it test functional or non-functional requirements?
31	What are the inputs to the technique? P02:03 Whether the technique requires access to source code C06:11 What are the inputs to the technique? C06:14 Does the test methodology use black box, white box, or both methods? P03:03 Inputs. Type of artefacts needed to apply it C09:03 Does this technique provide white-box testing? C09:04 Does this technique provide black-box testing? C10:01 What are the inputs required by the technique? C11:01 Does the technique require source code? C11:02 What is my interface with the technique?
32	What is the coverage provided by the generated test cases? P02:05 Whether they have been designed to ensure that code is covered C08:13 What is the coverage? P03:01 Coverage (regarding the aspect being tested: sentences, models, scenarios, etc.)

Table 6.4: Content of the reference set (4/5).

N	QUESTION AND SOURCE
33	<p>What is the adequacy criteria of the technique?</p> <p>P02:06 Stopping criteria of the technique</p> <p>C06:14 Does the test methodology use black box, white box, or both methods?</p> <p>C09:03 Does this technique provide white-box testing?</p> <p>C09:04 Does this technique provide black-box testing?</p> <p>C12:05 How would you classify the technique? (black-box, white-box, dynamic, static, etc)</p>
34	<p>Is the technique bounded to any development paradigm?</p> <p>C06:12 Does the technique assume a certain type of architecture or modelling paradigm?</p> <p>C08:02 What development paradigm can it be used with?</p> <p>C12:03 What development paradigms are supported by each of the techniques?</p> <p>C13:03 What is the environment where it can be used?</p>
35	<p>What software application domain can the technique be used with?</p> <p>C08:03 What application domain can it be used with? (for risks identification)</p>
36	<p>Is it easy to understand?</p> <p>C08:17 How easy is it to understand?</p> <p>C10:06 How much does the adoption of the technique take?</p>
37	<p>What testing level can the technique be used at?</p> <p>C09:02 Is this technique available to be used by all levels of testing for this product?</p> <p>C09:05 Does this technique provide regression testing?</p> <p>C12:01 What types of testing are supported by each technique? (unit testing, integration testing, etc.)</p>

Table 6.5: Content of the reference set (5/5).

As shown in Table 6.7, the information can be grouped as three levels: tactical, operational and use. The meaning of each level is as follows:

1. *Tactical Level.* This level coincides with the idea of tactical level that appeared in the theoretical schema. Therefore, the information contained in this level refers to information related to what is to be tested. It would encompass questions: 5, 16, 28, 30, 35 and 37 from the reference set.
2. *Operational Level.* This level coincides with the idea of operational level that appeared in the theoretical schema. That is, the information contained in this level is related to information about the project within which the test is to be run. Questions 1, 2, 4, 6-15, 17-19, 23-27, 29, 31-34 and 36 of the reference set appear at this level.
3. *Use Level.* It was not possible to associate the information contained in this level with any of the two levels in the theoretical schema. Therefore, a new level was created: the use level. The group of questions of which this new level is composed refers to the personal experiences of people who have used the technique. It encompasses questions 3 and 20-22 of the reference set.

QUESTION
C02:09 Does it have scientific basis?
P01:05 Whether the technique has a scientific background.
P01:07 Return on the investment (cost and benefits) of the technique
P01:11 Type of hardware the technique can be used with.
P01:12 Type of company the technique can be used with.
C03:03 What is the base underlying?
C04:05 Is it possible to do things in a simultaneous way?
C04:06 How can it decrease the testing effort?
P02:02 Costs versus return-on-investment for each technique.
P02:04 If they are static or dynamic.
C05:06 Do any techniques provoke other errors in the system?
C05:07 Do any techniques isolate problems in the software?
C06:01 What is the cost of buying the techniques?
C06:03 Are the tests that are produced automate able?
C06:07 What are the important factors for determining whether to use a technique and how to tailor it?
C06:08 Will the technique help pinpoint high risk areas of the project?
C06:10 Is there traceability from feature to test?
C06:13 Does the technique use the existing system GUIs to test the system or are "internal" test scripts (code) written?
C07:04 How were the unit tests conducted during development?
C07:05 How do the unit tests relate to the requirements?
C07:06 Is there a "testing" framework integrated with the product?
C07:07 Are the tests going to be part of the installation?
C07:08 What Test Plans have been written?
C07:09 How often are the tests to be run? Manual vs. Automated?
P03:04 Validation. Show that it works
C09:08 Does this technique produce any results, logs or statistics from the tests?
C11:03 What data does the technique generate?
C11:07 Does the technique instrument the software under test? If yes, how?
C13:07 What assumptions are made for the technique to be effective?

Table 6.6: Questions raised by respondents and rejected.

### 6.2.2.2 Elements of the Empirical Schema

Having identified the levels of which the empirical schema is composed, the information was again grouped within each level, but this time analysing the element of the tests to which it refers. The elements identified within each level are presented below.

#### 6.2.2.2.1 Tactical Level

Table 6.8 shows the grouping of the questions at the tactical level as elements.

Two elements were identified at the tactical level, which coincide with the two identified in the theoretical schema. These are:

LEVEL	QUESTION
Tactical	5. At what moment of the development can the technique be used? 16. What software objects does it test? 28. What quality attributes can be tested with the technique? 30. What aspects of the software does the technique test? 35. What software application domain can the technique be used with? 37. What testing level can the technique be used at?
Operational	1. Can the technique be used with any programming language? 2. Are there tools that ease the use of the technique? 4. Can the technique be used with any development method? 6. Do people need specific knowledge in order to use the technique? 7. Is the technique experimental or has it been used before? 8. Can the technique be used with any type of software? 9. What is the effectiveness of the technique? 10. What is the effort in terms of time and resources to apply the technique? 11. How long does it take to obtain the test cases? 12. What is the formality level of the technique? 13. Does the use of the technique imply the use of other techniques? 14. Does the technique require experienced people? 15. How many test cases does it generate? 17. What type of defects does it look for? 18. Does it need inputs in a special format? 19. What is the documentation available? 23. What part of the technique do tools automate? 24. What is the cost of buying and maintaining the tools? 25. What are the restrictions of the tool? 26. What type of support does the tool have? 27. Can the technique be used for software systems of any size? 29. Regression cost 31. What are the inputs to the technique? 32. What is the coverage provided by the generated test cases? 33. What is the adequacy criteria of the technique? 34. Is the technique bounded to any development paradigm? 36. Is it easy to understand?
Use	3. Has the technique ever been used before? 20. Are the people who have used the technique satisfied with it? 21. What have been the results (costs and benefits) of using the technique? 22. Which tools have been used with the technique?

Table 6.7: Grouping of the questions as levels.

1. *Objective.* The information covered in this element refers to the goal of the test. It encompasses questions 28 and 35 of the reference set.
2. *Scope.* The information covered by this element refers to the scope of the test. It encompasses questions 5, 16, 30 and 37.

ELEMENT	QUESTION
Objective	28. What quality attributes can be tested with the technique?
	35. What software application domain can the technique be used with?
Scope	5. At what moment of the development can the technique be used?
	16. What software objects does it test?
	30. What aspects of the software does the technique test?
	37. What testing level can the technique be used at?

Table 6.8: Grouping of the tactical level questions as elements.

#### 6.2.2.2.2 Operational Level

Table 6.9 shows the grouping of the operational level questions as elements.

ELEMENT	QUESTION
Agents	6. Do people need specific knowledge in order to use the technique?
	14. Does the technique require experienced people?
Object	1. Can the technique be used with any programming language?
	4. Can the technique be used with any development method?
	8. Can the technique be used with any type of software?
	27. Can the technique be used for software systems of any size?
	34. Is the technique bounded to any development paradigm?
Tools	2. Are there tools that ease the use of the technique?
	23. What part of the technique do tools automate?
	24. What is the cost of buying and maintaining the tools?
	25. What are the restrictions of the tool?
	26. What type of support does the tool have?
Technique	7. Is the technique experimental or has it been used before?
	10. What is the effort in terms of time and resources to apply the technique?
	11. How long does it take to obtain the test cases?
	12. What is the formality level of the technique?
	13. Does the use of the technique imply the use of other techniques?
	18. Does it need inputs in a special format?
	19. What is the documentation available?
	29. Regression cost
	31. What are the inputs to the technique?
	33. What is the adequacy criteria of the technique?
	36. Is it easy to understand?
Results	9. What is the effectiveness of the technique?
	15. How many test cases does it generate?
	17. What type of defects does it look for?
	32. What is the coverage provided by the generated test cases?

Table 6.9: Grouping of the operational level questions as elements.

Five elements were identified within the operational level, one more than in the theoretical schema. These elements are:

1. *Agents*. The information covered in this element refers the characteristics of the people who are to apply the technique. It would encompass questions 6 and 14.
2. *Object*. The information covered in this element refers to the characteristics that the software should have to be able to apply the technique. It would encompass questions 1, 4, 8, 27 and 34.
3. *Tools*. The information covered in the tools element refers to the characteristics of the tools that can be used when applying the technique. It would encompass questions 2 and 23-26.
4. *Technique*. The information covered here refers to the characteristics of the actual technique that possibly have an influence on its selection. It would encompass questions 7, 10, 11-13, 18, 19, 29, 31, 33 and 36.
5. *Results*. The information covered in this element refers to the characteristics of the test cases generated by the technique (or results of applying the technique). It would encompass questions 9, 15, 17 and 32.

#### 6.2.2.2.3 Use Level

Table 6.10 shows the grouping of the use level questions as elements.

ELEMENT	QUESTION
Project	3. Has the technique ever been used before?
	20. Are the people who have used the technique satisfied with it?
	22. Which tools have been used with the technique?
Satisfaction	21. What have been the results (costs and benefits) of using the technique?

Table 6.10: Grouping of the use level questions as elements.

Two elements were identified within this level, which are:

1. *Project*. The information covered in this element refers to the interest of respondents in learning about and characterising software projects in which the technique has been applied in order to compare these earlier projects with the current situation. It would encompass questions 3, 20 and 22.
2. *Satisfaction*. The information covered in this element complements the above information on earlier projects. The respondents are also interested in knowing the results of using the technique in the project from the viewpoint of what impression it caused on the person who used the technique. It would encompass question 21.

#### 6.2.2.3 Attributes of the Empirical Schema

This section addresses the attributes that the empirical schema should contain to respond to the questions raised by consumers. They are reflected in Table 6.11 through Table 6.22.



QUESTION	SOURCE	EXPLANATION	SCHEMA ATTRIBUTE
28. What quality attributes can be tested with the technique?	P01:01 What the technique measures C04:02 What is possible to test with this technique? C12:02 Which of the following quality attributes are testable by the technique? Reliability, safety, reusability, maintainability, scalability. C12:04 What metrics are used to evaluate the quality attributes listed in question C12:02?	Questions C04:02 and C12:02 show that respondents would like to know which software quality attributes the technique can be used to evaluate. So, respondents want to know the quality attributes associated with each technique in order to be able to decide which techniques they can use to evaluate a given attribute. However, it is not enough merely to include the software quality attribute to completely reflect the user's need for information, the schema attribute must also reflect the metric used. This accounts for the specification of C12:04. This is logical, as there are usually different metrics associated with one and the same software quality attribute, which can be used by the developer to interpret the values obtained in the evaluation. Accordingly, the consumer will be able to find out both the quality attributes that the technique can be used to evaluate and how the results obtained should be interpreted.	Quality attribute
35. What software application domain can the technique be used with?	C08:03 What application domain can it be used with? (for risks identification).	This question addresses the fact that some software systems require more exhaustive tests than others do. For example, a safety-critical system needs to pass more rigorous tests of some quality attributes than other systems do. Indeed, the characteristics of the software to be built make it possible to identify the risk areas to which the respondent refers. Asking about the application domain of the technique, the respondents aims to find out for which of the above-mentioned risk areas the technique can be used. The information to be stored in the schema represents how rigorous or precise the evaluation technique is.	Quality attribute Rigour

Table 6.11: Attributes generated by the objective element.

QUESTION	SOURCE	EXPLANATION	SCHEMA ATTRIBUTE
5. At what moment of the development can the technique be used?	<p>C01:05 In what phase of the development process can it be used?</p> <p>C06:09 How early in the development cycle can a test case be constructed?</p> <p>C09:02 Is this technique available to be used by all levels of testing for this product?</p> <p>C09:05 Does this technique provide regression testing?</p> <p>C12:01 What types of testing are supported by each technique? (unit testing, integration testing, etc.)</p> <p>C12:06 Which are the development activities to which one can apply the technique?</p> <p>C13:05 When can the technique be applied?</p>	<p>This group of questions refers to the time of development (phase, activity or task) when the technique can be used. There are several points during software development when testing techniques are applied: during the requirements phase, where system and acceptance techniques are applied, in the design phase, where integration techniques are used, during coding, where unit testing techniques are used, and during maintenance, where regression techniques are used.</p>	Phase
16. What software objects does it test?	<p>C01:15 Can it be used on software components, or the whole system is needed?</p> <p>C01:18 Can it be used for testing hardware?</p> <p>C04:02 What is possible to test with this technique?</p> <p>C05:01 Does the technique test the requirements?</p> <p>C05:08 Does the technique test the current system design and methodology?</p> <p>C07:03 What dependencies this product has on other products in order to allow for staging and integration testing?</p>	<p>These questions refer to the elements of the software system that are affected by the test: algorithms, subsystems, modules, etc. Faced with a given technique, it is important to know to which system elements it can be applied.</p>	Element
30. What aspects of the software does the technique test?	<p>C04:02 What is possible to test with this technique?</p> <p>C05:09 Does it test for functionality?</p> <p>C06:04 What aspects of the software is the technique aimed at verifying?</p> <p>C06:15 Is the methodology capable of generating test cases for GUIs?</p> <p>C09:06 Does this technique provide functional testing?</p> <p>C09:07 Does this technique test a user's interaction to this product?</p> <p>C09:09 Does this technique help test the scalability and load balancing for the product?</p> <p>C10:02 Does it test functional or non-functional requirements?</p>	<p>This set of questions refer to the functionality of the software system to be tested, as different techniques can be used to test different parts of the software, such as security, communications, user interface, etc.</p>	Aspect
37. What testing level can the technique be used at?	<p>C09:02 Is this technique available to be used by all levels of testing for this product?</p> <p>C09:05 Does this technique provide regression testing?</p> <p>C12:01 What types of testing are supported by each technique? (unit testing, integration testing, etc.)</p>	<p>This group of questions refers to the testing level for which the test is suited. However, the level is determined on the one hand by the development phase in which the test is to take place and, on the other, by the elements involved in the test.</p>	Phase Element

Table 6.12: Attributes generated by the scope element.

QUESTION	SOURCE	EXPLANATION	SCHEMA ATTRIBUTE
6. Do people need specific knowledge in order to use the technique?	<p>C01:06 Do people have to be trained?</p> <p>C01:13 Are people with experience or special knowledge required?</p> <p>C01:17 What type of people can use it?</p> <p>C02:01 What is the knowledge people should have in order to use it?</p> <p>C02:07 What is the estimate time for training people who will use it?</p> <p>C02:08 What knowledge should people have in order to use it?</p> <p>P01:04 If it can be used by average developers</p> <p>P01:10 Knowledge of the people that have to use it</p> <p>C03:02 What are the resources needed (machines, people, etc.)?</p> <p>C08:06 Will people have to be trained using the tool?</p> <p>C08:08 Does the use of the tool imply a cultural change?</p> <p>P03:02 Whether the technique is usable for mid-developers</p> <p>C10:07 Do testers need to have special knowledge to use it?</p>	<p>All these respondent questions reflect their concern about special or specific knowledge that the people responsible for applying or using the testing technique should have. It follows from these questions that this information is of interest primarily because if the personnel need knowledge that they do not possess, the training to be provided has to be taken into account. It is important to remember that the provision of training, if necessary, implies a financial investment and takes time.</p>	Knowledge
14. Does the technique require experienced people?	<p>C01:13 Are people with experience or special knowledge required?</p> <p>C01:17 What type of people can use it?</p> <p>C03:02 What are the resources needed (machines, people, etc.)?</p> <p>C08:04 What is the level of experience of the people with the technique?</p>	<p>These questions reflect the concern of respondents about the experience that the people who are to use the technique should have. But, of course, it is not only important to state how much experience is required to apply a technique, but also in what subject in particular experience is needed. The application of a technique may merely call for theoretical experience in testing in some cases, experience in other techniques may suffice in others and it may be necessary for the subject to be experienced in the technique in question in others.</p>	Experience

Table 6.13: Attributes generated by the agents element.

QUESTION	SOURCE	EXPLANATION	SCHEMA ATTRIBUTE
2. Are there tools that ease the use of the technique?	<p>C01:02 Are there tools to support it?</p> <p>C02:10 What are the available tools?</p> <p>P01:06 If it can be automated, at least partially.</p> <p>C03:02 What are the resources needed (machines, people, etc.)?</p> <p>P02:01 Automation of the technique.</p> <p>C06:05 Are there tools (executables or documents) to guide the user in using the technique?</p> <p>C08:05 What is the support (tool) the technique has?</p> <p>P03:05 Automation support with tools</p> <p>C09:01 Is this technique automatic or manual?</p> <p>C10:03 Is there tool support?</p> <p>C10:04 Does it allow automation?</p> <p>C11:06 Can the technique be used to perform automated testing?</p> <p>C13:06 What tools can be used to support the technique?</p>	<p>All these questions show how important it is for respondents for there to be a support that can be used to automate the application or use of the testing technique. This interest is related to the fact that both the generation and execution, not to mention the maintenance, of test cases usually takes a lot of effort. The existence of a tool that automates part of the test case generating technique can have different implications. For example, it will reduce the time it takes to generate test cases, or, perhaps, some knowledge that would be needed if the technique was applied manually will be obviated, or, alternatively, knowledge of the testing tool will be needed, etc.</p>	Identifier
23. What part of the technique do tools automate?	<p>C02:11 What part of the techniques do available tools automate?</p>	<p>This question reflects the fact that there are no tools that fully automate any technique, but which automate some parts of the application of some techniques.</p>	Automation
24. What is the cost of buying and maintaining the tools?	<p>C02:12 What is the acquisition and maintenance cost of the tools?</p>	<p>The question reflects the interest of respondents in the cost of testing tools. This information has an influence on the selection process because software projects are subject not only to time but also to budget constraints. The fact that an organisation already has a testing tool can lead the testing team to opt for one technique as opposed to another of similar characteristics for which they do not own a tool. But not only the purchase price of the tool is important, the maintenance cost is also a significant factor that will have an influence when opting for one tool or another (and possibly by extension for one technique or another).</p>	Cost

Table 6.14: Attributes generated by the tools element (1/2).

QUESTION	SOURCE	EXPLANATION	SCHEMA ATTRIBUTE
25.What are the restrictions of the tool?	C02:13 What hardware and operating system does the technique work in? C02:14 What programming language and dialect does the technique work with? C08:07 What is the platform (software and hardware) the tool needs? C11:05 What platform does the technique run on, and what are its resource requirements (memory, disk, cpu time....)?	This question reflects the interest of respondents in knowing the technological environment in which the tool works. In particular, the data that appear to be of interest for respondents are the hardware required, the operating system and the programming language (including dialect) with which it operates.	Environment
26.What type of support does the tool have?	C02:15 Does the technique have support? (hot-line). C08:09 What is the existing support for the tool?	Another matter that appears to be of interest for respondents with respect to testing tools is the type of support provided. It is essential to know what type of help is built into the tool in question. For example, if a tool purchased in the USA has a 24-hour hotline, there will be no need for concern about the time difference should any problem occur.	Support

Table 6.15: Attributes generated by the tools element (2/2).

QUESTION	SOURCE	EXPLANATION	SCHEMA ATTRIBUTE
7. Is the technique experimental or has it been used before?	C01:07 Is it experimental or has it been tested? C08:11 Has the technique been evaluated or validated?	These questions reflect the concern of respondents about the extent to which the technique has been tested or validated. It is logical that they should be concerned about this, as a technique that is experimental or has not been thoroughly tested will not offer as many guarantees as a technique that has been used previously. However, there is a range of variations between the two extremes that will represent how experimental the technique is.	Maturity level
10. What is the effort in terms of time and resources to apply the technique?	C01:10 How long does it take test case generation? C02:06 What is the estimate time when test cases generation? P01:03 Resources the technique uses. P01:15 Time that will be needed in order to apply the technique C04:04 How much effort do I need to apply this technique? C06:02 How much time does it take to produce a test case? C08:16 How easy is it to apply? P03:06 Application cost, in terms of time C11:08 If the technique instruments the software under test, what is the overhead for the instrumentation? (memory, cpu time...) C13:01 How easy is it to use? C13:02 What is the cost of using it?	This question encompasses the questions raised by respondents related to their concern about the difficulty of applying the technique from the viewpoint of the time it takes. Obviously, respondents will attempt to avoid techniques that consume too many resources, and an effort will always be made to use, whenever possible, techniques that do not involve too much effort on the part of the people who are going to apply the technique.	Cost of application
11. How long does it take to obtain the test cases?	C01:10 How long does it take test case generation? C02:06 What is the estimate time when test cases generation? C06:02 How long does it take to implement and re-implement a test case? C08:15 How straightforward is producing the test cases?	These questions are related to the fact that many techniques do not output test cases. This means that after applying the technique an additional step has to be taken to generate the test data. This is the case, for example, with structural tests. The techniques output the conditions of the code that are to be covered, but it is still not known what inputs are to be supplied to the program.	Test data cost

Table 6.16: Attributes generated by the technique element (1/3).

QUESTION	SOURCE	EXPLANATION	SCHEMA ATTRIBUTE
12. What is the formality level of the technique?	C01:11 Is it objective? C05:02 Does the technique create repeatable tests?	The questions covered by question number 6 reflect the fact that the same technique applied to the same program by two different people will often produce different results. This is related to how formal the technique is. The more formalised the technique is, the more likely two people are to arrive at the same results after application than if it were not very formalised. Repeatability is what will indicate whether or not the technique is formalised. The degree of formality of a technique can vary from mere guides or recipes to a series of well-defined and organised steps.	Repeatability
13. Does the use of the technique imply the use of other techniques?	C01:12 Is it enough using that technique or has to be complemented with other(s)? P01:08 If it can be complemented with other techniques P02:07 Whether you are duplicating effort when using several techniques C10:09 How does it compare to other similar techniques?	The questions covered by question number twelve reflect respondents' need for information about the relationships between different testing techniques. In particular, respondents will be most interested in this respect in knowing what techniques can or should be combined with another and whether they should be applied in a particular order. This is quite logical, as it has been shown that the best way of testing software is by complementing various techniques, as it appears that different techniques tend to detect different defects.	Dependencies
18. Does it need inputs in a special format?	C01:19 Does it require requirements/design/etc. in any special format? C07:02 What should be the format of the development documents produced? C10:05 How much do you have to change the development previous to testing to apply this technique?	The questions covered by question 18 reflect the interest of respondents in knowing the inputs that will have to be supplied to the technique and, especially, their format. It is reasonable for respondents to want to know this, as if a given technique requires the input documents in a format that is not being used in the project, it does not make much sense to use the technique. For example, if the requirements specification has to be made in formal notation and it has been done in natural language, it is unlikely to be worth the effort changing the notation of the requirements document just to apply the technique.	Inputs

Table 6.17: Attributes generated by the technique element (2/3).

QUESTION	SOURCE	EXPLANATION	SCHEMA ATTRIBUTE
19. What is the documentation available?	C02:02 What is the available documentation? C06:05 Are there tools (executables or documents) to guide the user in using the technique? C10:08 What kind of support exists for introducing the technique (training, etc)?	These questions reflect the interest of respondents in knowing what sources of information are associated with a given technique. Faced with a given technique, it is essential for respondents to know from what documentation they can extract information about the technique, should they have any doubt about its application. Reflecting at length on these questions, however, not only the documentation in which information about the technique can be found is of interest. Generally, any source from which information about the technique can be gathered will be of interest to respondents.	Sources of information
29. What is its regression cost?	P01:14 Easiness for regression.	There are specific techniques for regression, therefore, this question does not make sense as regards the schema.	No attribute
31. What are the inputs to the technique?	P02:03 Whether the technique requires access to source code C06:11 What are the inputs to the technique? C06:14 Does the test methodology use black box, white box, or both methods? P03:03 Inputs. Type of artefacts needed to apply it C09:03 Does this technique provide white-box testing? C09:04 Does this technique provide black-box testing? C10:01 What are the inputs required by the technique? C11:01 Does the technique require source code? C11:02 What is my interface with the technique?	The questions associated with this question are equivalent to those associated with question number 18.	Inputs
33. What is the adequacy criteria of the technique?	P02:06 Stopping criteria of the technique. C06:14 Does the test methodology use black box, white box, or both methods? C09:03 Does this technique provide white-box testing? C09:04 Does this technique provide black-box testing? C12:05 How would you classify the technique? (black-box, white-box, dynamic, static, etc)	These questions correspond to the definition that has been given for adequacy criterion in the previous chapter.	Adequacy criterion
36. Is it easy to understand?	C08:17 How easy is it to understand? C10:06 How much does the adoption of the technique take?	This question reflects the interest of respondents in knowing what difficulties the people who are going to use the technique, especially, if they have never used it before, are likely to encounter. Obviously, simpler techniques will be preferred to techniques that are more complicated to understand and, therefore, apply.	Comprehensibility

Table 6.18: Attributes generated by the technique element (3/3).



QUESTION	SOURCE	EXPLANATION	SCHEMA ATTRIBUTE
9. What is the effectiveness of the technique?	C01:09 What is its effectiveness? (number of errors found) P01:02 Which are the benefits the technique should produce (number of errors) C03:01 What are its results (objectives)? C04:03 What are the results this technique has got? (Is it effective?) C08:12 Is it cost-effective? C13:04 How effective is this technique generally?	This series of questions reflects the interest of respondents in knowing whether or not the technique really fulfils the objectives of the test, that is, knowing whether or to what extent it is effective. It might appear at first glance that this attribute should be grouped under the technique element. However, it is the test cases that show up faults and, therefore, it has been decided to group this attribute under the results element.	Effectiveness
15. How many test cases does it generate?	C01:14 Does it generate a minimum set of test cases?	This question refers to the number of test cases that the technique generates. This is often viewed as the measure of the cost of the technique. The number of test cases generated by a technique is directly related to the duration of the tests for two main reasons: the execution of the test cases and the time it takes to generate and maintain the test cases.	Number of cases generated
17. What type of defects does it look for?	C01:16 What type of errors does it look for? C05:03 Does the technique test for logic problems? C05:04 Does the technique test for application errors? C05:05 Does the technique check for grammatical/syntax errors?	This set of respondent questions clearly refers to the type of defects (problems, errors, faults, etc.) detected by the technique. This information is also very useful, as depending on the errors traditionally made at the company (or by the group of developers), one technique or another will be of more interest. This information can also be an aid for complementing the technique.	Type of defects
32. What is the coverage provided by the generated test cases?	P02:05 Whether they have been designed to ensure that code is covered C08:13 What is the coverage? P03:01 Coverage (regarding the aspect being tested: sentences, models, scenarios, etc.).	This user question evidently refers to the cover the technique. Coverage is important, as it establishes how complete the technique is with respect to the objective of the test.	Coverage

Table 6.19: Attributes generated by the results element.

QUESTION	SOURCE	EXPLANATION	SCHEMA ATTRIBUTE
1. Can the technique be used with any programming language?	C01:01 Does it depend on the programming language of the code to be tested? C03:04 What programming language can it be used with? C06:16 What programming languages does the technique work for? C11:04 What languages does the technique support? C13:03 What is the environment where it can be used?	These questions reflect respondents' interest in knowing the programming language with which the technique can be used. Each type of programming language is characterised by typical constructions. Thus, functional languages do not use the same type of constructions as structured, real-time and logical languages, etc.	Programming language
4. Can the technique be used with any development method?	C01:04 Is it bounded to a life-cycle or a methodology? C02:05 For what life-cycle can it be used? P01:13 Type of development method the technique can be used with. C07:01 What software development process can the technique be used with? (i.e. incremental, XP, spiral, waterfall, etc.) C13:03 What is the environment where it can be used?	This series of questions accounts for the respondents' need to know with what development method the testing technique can be used (prototyping, reuse, etc.).	Development method
8. Can the technique be used with any type of software?	C01:08 In what type of projects has it been used? (real-time, management, etc.) C02:04 What type of software it can be used with? P01:09 Type of software the technique can be used with. C03:05 Is it adequate for the type of software that is being developed? C04:01 For what kind of software product is this technique more adequate? C08:01 What type of software is the technique suitable for?	The type of software (management, control, real-time, etc.) with which the technique can be used appears to be important for respondents.	Software type
27. Can the technique be used for software systems of any size?	C02:16 What is its scalability? Can it be used for big and small systems?	This question reflects the interest of respondents in knowing the size of the software product with which the technique can be used. Some techniques will possibly only be of use for small development projects or small parts of the software.	Size
34. Is the technique bounded to any development paradigm?	C06:12 Does the technique assume a certain type of architecture or modelling paradigm? C08:02 What development paradigm can it be used with? C12:03 What development paradigms are supported by each of the techniques? C13:03 What is the environment where it can be used?	These respondent questions reflect their interest in knowing the development paradigm with which the technique can be used. The technique may have limitations, because it is for structured software, object-oriented software, etc. The objective of this information is to find out precisely what limitations the technique has.	Software architecture

Table 6.20: Attributes generated by the object element.

QUESTION	SOURCE	EXPLANATION	SCHEMA ATTRIBUTE
3. Has the technique ever been used before?	C01:03 Are there users or projects of reference? C02:03 Has somebody used it? C08:10 Has the technique ever been used before? C08:14 Who has used the technique? If nobody, why?	The questions covered by question number three represent the interest of respondents in knowing whether the technique has been used before and, if so, for what projects it has been used. The interest in finding out about earlier applications of the technique is logical, as it makes it possible to compare the earlier projects with the project at hand.	Reference projects
20. Are the people who have used the technique satisfied with it?	C02:03 Are people who have used it satisfied with it? Will they use it again?	These questions reflect respondent concern about what impression the technique caused on the people who used it. Irrespective of what is said about the technique at the operational level, it appears that respondents still want to know what people who have used the technique think about it.	Personnel
22. Which tools have been used with the technique?	C02:03 Have tools been used with the technique? Which ones? How was the experience?	This question reflects the interest of respondents in knowing which tools were used with the technique in earlier projects	Tools used

Table 6.21: Attributes generated by the project element.

QUESTION	SOURCE	EXPLANATION	SCHEMA ATTRIBUTE
21. What have been the results (costs and benefits) of using the technique?	C02:03 What have been the advantages of using the technique? C02:03 What has been the return on investment of the technique? C06:06 What kind of results have you seen when used on a project like mine?	The questions covered by question number twenty-one refer to what people think about the application of the technique in a given project. The opinions will normally reflect both positive and negative aspects of technique use. These questions will be answered by entering three different attributes in the characterisation schema, each of which represents the strengths, weaknesses and opinion about technique application.	Benefits Problems Opinion

Table 6.22: Attributes generated by the satisfaction element.

## 6.3 Result of Building the Empirical Schema

This chapter addressed the construction of the empirical schema. This schema is composed of thirty-six attributes, divided into three levels. Table 6.23 illustrates its composition. Column 1 shows the level to which the attribute belongs, column 2 reflects the element to which it refers, column 3 shows the attribute and column 4 gives a brief explanation of each attribute.

## 6.4 Study of the Evolution of the Empirical Schema

The empirical schema underwent a series of transformations as the reference set evolved. Information of interest about the empirical schema, such as schema stability, what are the most important attributes within the schema or when each schema attribute appeared, will be able to be deduced from the evolution of the reference set.

Indeed, the following studies will be described in this section:

- *Schema growth*. Study of the impact of adding each new respondent to the schema in terms of the number of attributes added.
- *Importance of each schema attribute*. Study of the number of respondents who mentioned a given schema attribute as being of interest.
- *Schema evolution (when each attribute was added)*. The time at which each schema attribute appeared and possible relationship with its importance.

The data for the graphs that appear in this section are given in Appendix D.3.

### 6.4.1 Schema Growth

Thanks to the way in which the schema was built, it is possible to find out the status of the schema as each respondent was added. As the information provided by new respondents was analysed, the schema was open to possible changes. The aim of examining schema growth is to find out how big these changes were in terms of the how many new attributes a respondent added to the characterisation schema.

The reason for studying schema growth is to analyse its stability. As the growth of the schema slows down, its stability grows, and this reaches its height when growth is zero. As mentioned at the start of this chapter, respondents continued to be interviewed until the stability of the schema reached an acceptable level.

Figure 6.5 reflects the changes that the empirical schema underwent. The x-axis represents the respondents interviewed. The y-axis represents the number of attributes each consumer contributed to the characterisation schema. The darker region represents the number of new attributes entered by each consumer into the schema, whereas the lighter region represents the number of attributes already existing in the schema that the consumer would like to see in the schema.

LEVEL	ELEMENT	ATTRIBUTE	DESCRIPTION
Tactical	Objective	Quality attribute	Quality attribute of the system to be tested
		Rigour	Intensity of the test
	Scope	Phase	Time of development at which the test is to be run
		Element	Elements of the system on which the test acts
		Aspect	Functionality of the system to be tested
Operational	Agents	Experience	Experience required to use the technique
		Knowledge	Knowledge required to be able to apply the technique
	Tools	Identifier	Name of the tool and the manufacturer
		Automation	Part of the technique automated by the tool
		Cost	Cost of tool purchase and maintenance
		Environment	Platform (SW and HW) and programming language with which the tool operates
		Support	Support provided by the tool manufacturer
	Technique	Comprehensibility	Whether or not the technique is easy to understand
		Maturity level	How experimental and/or how well validated the technique is
		Cost of application	How much effort is needed to apply the technique
		Inputs	Inputs required to apply the technique
		Adequacy criterion	Test case generation and stopping rule of the technique
	Results	Test data cost	Cost of identifying the test data
		Dependencies	Relationships of one technique with another
		Repeatability	Whether two people generate the same test cases
		Sources of information	Where to find information about the technique
		Coverage	Coverage provided by the set of test cases
Use	Object	Effectiveness	Capability of the set of cases to detect defects
		Type of defects	Type of defects the technique helps to discover
		Number of generated cases	Number of test cases generated per software size unit
		Software type	Type of software that can be tested using the technique
		Software architecture	Development paradigm to which the technique is linked
	Project	Programming language	Programming language with which the technique can be used
		Development method	Development method or life cycle to which the technique is linked
		Size	What size the software should be to be able to use the technique
		Reference projects	Earlier projects in which the technique has been used
		Tools used	Tools used with the technique in earlier projects
	Satisfaction	Personnel	Personnel who used the technique on earlier projects
		Opinion	General opinion about the technique after having used it
		Benefits	Benefits of using the technique
		Problems	Problems with using the technique

Table 6.23: Empirical schema.

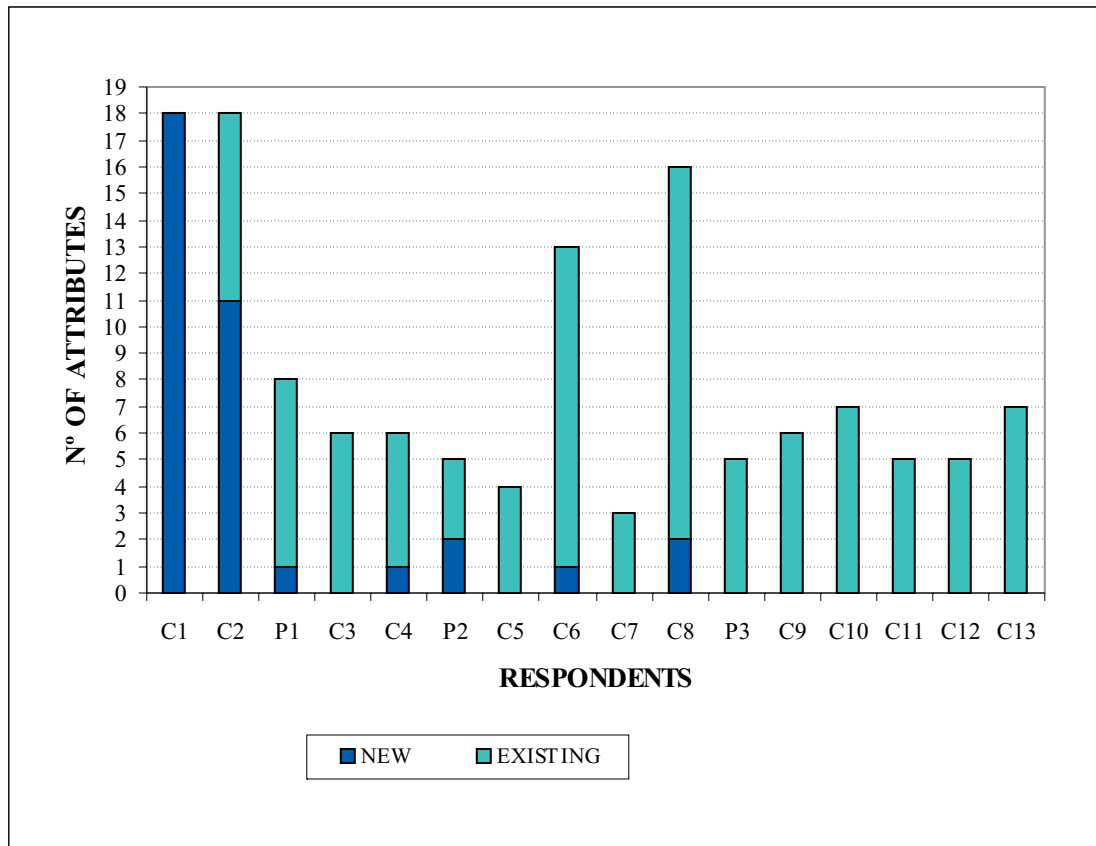


Figure 6.5: Attributes each respondent contributed to the empirical schema.

It is clear that all the attributes were new for the first respondent, as the schema did not exist beforehand. The second respondent also has the chance to contribute quite a lot of new attributes (eleven). However, as of respondent number three, there are nine subjects of fourteen who provide no new attributes (although they did ask for seven attributes). Only five of fourteen subjects do provide anything new, albeit only one or two attributes.

It is also clear that, as of respondent number 10 (consumer 8), no new attribute is generated in the schema, which means that the last six respondents would have found the schema to be satisfactory.

This means that when interviewing stopped, the stability of the schema was already very high. Schema stability cannot be said to be total, because a new respondent could always find new attributes. However, it can be said that the stability is high enough to guarantee a schema of some quality and for the search for information to be stopped.

Continuing with the study of schema growth, and this time for the purpose of reflecting the changes that took place in the characterisation schema as new respondents were added, two metrics were defined in terms of number of attributes of which it is composed:

- *Rate of schema growth.* This expresses the number of new attributes added by a given consumer to the characterisation schema, divided by the size of the schema before this consumer was added. Intuitively, this metric reflects how much the

schema grows with the consumer in question.

- *Speed of schema growth.* This expresses the number of new attributes added by a given consumer to the characterisation schema, divided by the final size of empirical schema. Intuitively, this metric reflects how rapidly the schema grows to reach its final size.

These two metrics are reflected in the graphs below. Figure 6.6 reflects the rate of schema growth, whereas Figure 6.7 and Figure 6.8 represent the speed (simple and accumulated) of schema growth. The x-axis of all three graphs reflects each of the respondents and the y-axis the percentage schema growth (albeit in different terms).

Figure 6.6 reflects the growth rate of the schema, that is, what percentage of the size of the schema at the time is represented by the attributes added by a given respondent. As is to be expected, the rate of growth of the first respondent is infinite (the schema goes from 0 to 18 attributes). For the second respondent, it is also large, around 60% growth. However, as of the third respondent, the rate of growth falls drastically (note that it is already less than 10% for the third respondent) and is never over 10%, dropping to 0% by respondent 11 and never rising again.

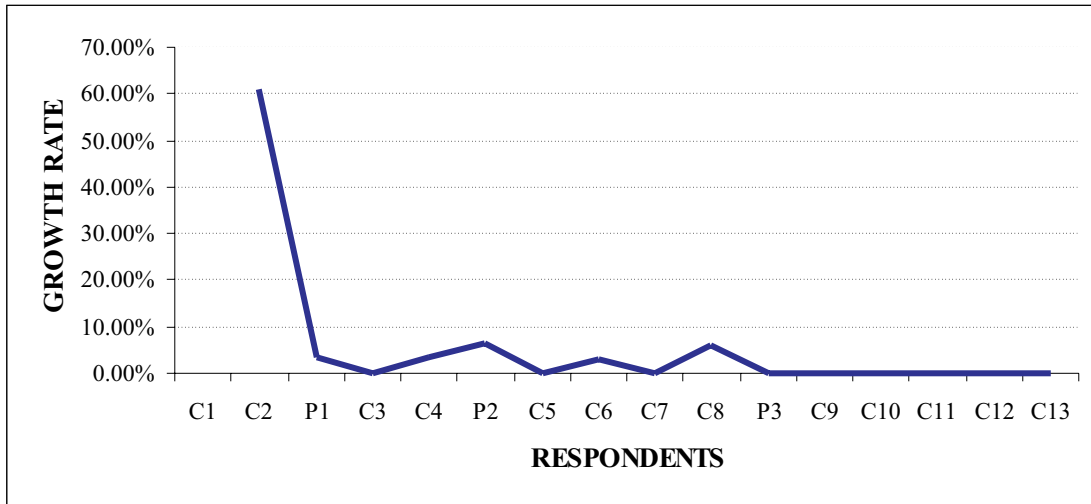


Figure 6.6: Rate of characterisation schema growth.

The shape of this graph confirms the above idea that the proposed schema is stable. Moreover, it also says that this stability was reached quite early on in terms of the number of respondents, as the slope of the graph is always negative and quite marked.

However, this graph could be deceptive. It is important to remember that it compares the current size of the schema with the number of new attributes entered by the respondent in question, and the percentage is calculated with respect to the size of the schema before entering the attributes. As the schema gradually grows, even if two subjects enter the same number of attributes, the second has a lower growth rate than the first. Accordingly, one might think that this does not give an accurate idea of schema stability. However, this is

not true for the following reason: the schema grows little as of the third respondent. Were growth to be large, the rate would indeed be more deceptive, however, one is talking about growths of at most two attributes, which is not significant for the total size of the schema, which already has twenty-nine attributes (out of thirty-six) at this point.

However, this graph is complemented by Figure 6.7, which reflects how much the schema grows with each respondent, this time with respect to the final size of the characterisation schema.

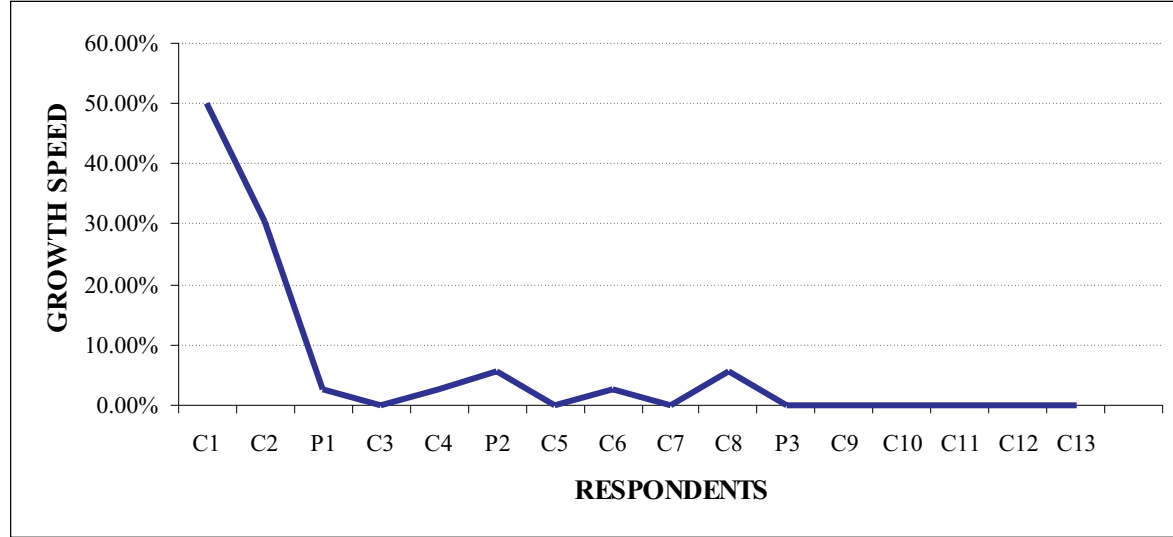


Figure 6.7: Speed of characterisation growth.

Figure 6.7 shows a very similar behaviour to Figure 6.6. With the first respondent, the schema grows to 50% of its final size, which is gradually completed until the schema reaches its final size with respondent 10. As is to be expected, the speed of growth of the schema is decreasing throughout the entire domain, which is another sign of its stability and that it has been reached gradually and not suddenly (if it had been reached suddenly, the graph could even have been increasing and then drop suddenly). This trend can be better appreciated in Figure 6.8, which represents the same values as Figure 6.7, but on an accumulated basis.

#### 6.4.2 Importance of each Schema Attribute

Having reached the conclusion that the empirical schema is sufficiently stable to stop data collection, it makes sense to start studying its content. The content of the schema will be examined by analysing the importance of each item of information that the schema contains. Two metrics have been defined for this purpose:

- *Absolute importance.* This measure represents the percentage of respondents that have mentioned the importance of a given type of information contained in the schema. It reflects the absolute importance of the information contained with



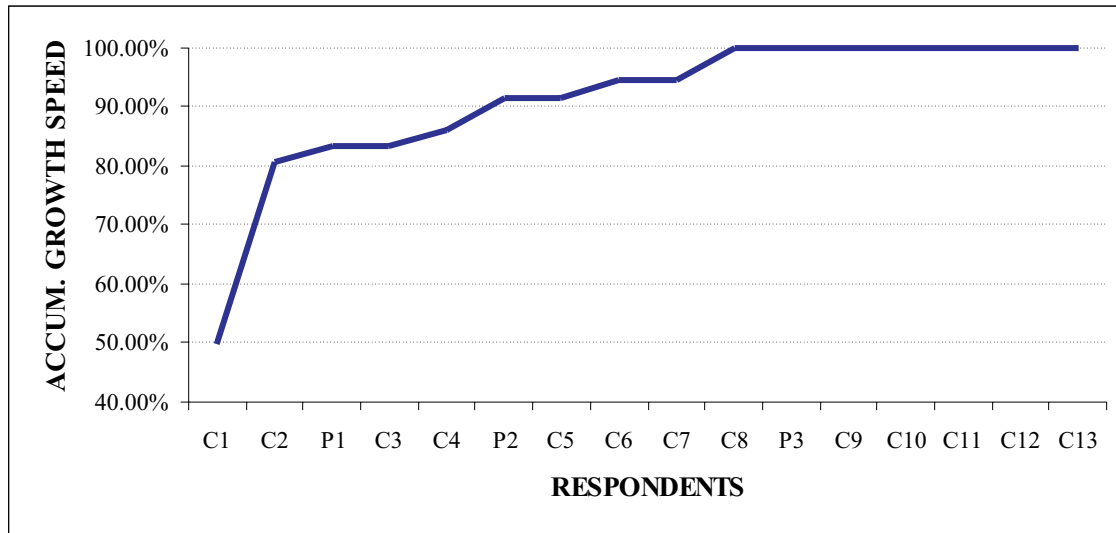


Figure 6.8: Accumulated speed of growth of the characterisation schema.

respect to the respondents. Intuitively, this metric means how important a given type of information is for the population.

- *Relative importance.* This represents the importance of a given type of information contained in the schema with respect to the other information it contains. It is calculated by weighting the value of each vote cast by a respondent by the number of votes cast by a consumer. Intuitively, this metric means how important given information is for the population, but this time with respect to the other information the schema contains.

The importance of the information contained in the characterisation schema will be examined at three levels:

- For schema attributes.
- For schema elements.
- For schema levels.

Starting with the schema attributes, Figure 6.9 reflects the absolute importance of each schema attribute, whereas Figure 6.10 reflects the relative importance of each schema attribute.

The y-axis of Figure 6.9 represents each of the schema attributes, whereas the x-axis represents the percentage of respondents who took the attribute in question into account for selection purposes.

It is clear that the attributes of most interest to the respondents (have been considered by over half of the population) are in descending order of votes: the existence of tools (*identifier*),

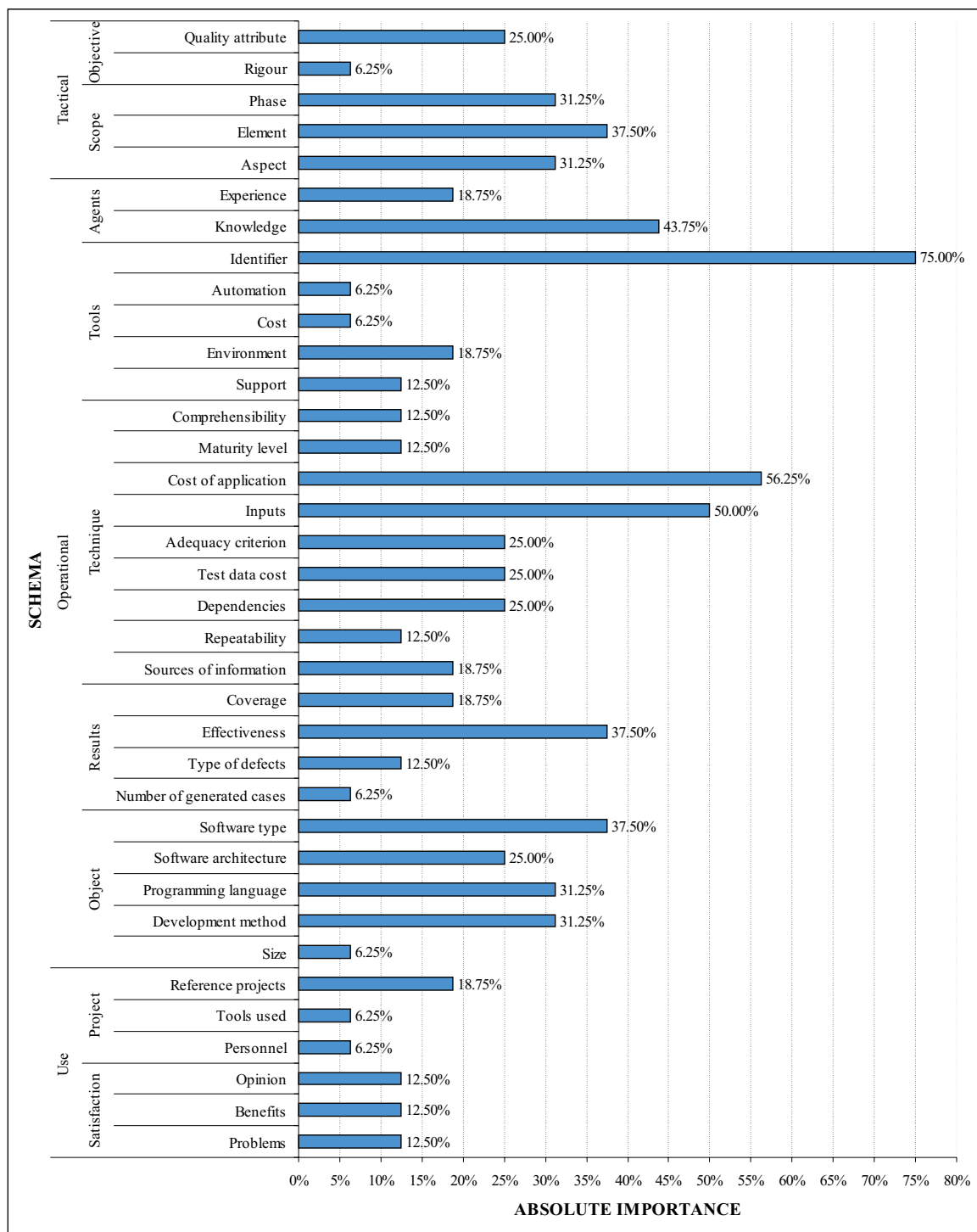


Figure 6.9: Absolute importance of each empirical schema attribute.

the *cost of application* of the technique and the *inputs* of the technique. This means that, on the one hand, the costs of using the technique and, on the other, the things required to apply the technique are taken into account. This makes sense, although it would have been logical for the respondents to focus on the benefits of the technique and not only on the costs. This

could indicate that they think that all techniques are equally effective or that they put the costs of a technique before the benefits of using it.

Other attributes of interest (considering the range of up to thirty-five per cent) are: the *knowledge* required to apply the technique, what *element* the technique tests, the *type of software* and the *effectiveness* of the technique. This means, on the one hand, that they have taken into account the benefits of using the technique (considered in terms of its effectiveness) and any possible limitations its application may have in terms of the people who are to apply it, what is to be tested, and how the project is being developed. Again, it makes sense that they should focus on these points, as they reveal that the respondents do not overlook the benefits and limitations of using a technique, although they do not consider them as important.

Finally, the least voted attributes include attributes related to earlier experiences using the technique, along with the rigour of the technique, the cost of the tools, the part of the technique tools automate, the number of cases the technique generates of the size the software should have. This is strange, because, as discussed later in Chapter 10, the attributes referred to experiences are the highest valued when using the schema. Additionally, the omission of the limitations of the technique appears to indicate that the respondents overlook the fact that some testing techniques can be more efficient (consume less resources) than others.

It is not worth discussing the other attributes, as their scores were intermediate and not extreme.

Continuing with the importance of schema attributes, the y-axis of the Figure 6.10 represents each of the schema attributes and the x-axis represents the percentage weight (in terms of importance) of each of the attributes that the empirical schema contains.

The purpose of using this graph was to normalise Figure 6.9, making the sum of the percentages 100%. However, compared with the Figure 6.9, it is clear that the numbers do not differ, at least to an extent that can be considered significant.

Now focusing on the elements of the schema, Figure 6.11 reflects the absolute importance of each schema element, whereas Figure 6.12 reflects the relative importance of each element.

The y-axis of Figure 6.11 represents each element of the schema, whereas the x-axis represents the percentage of respondents who took into account the element in question for selection purposes. For this purpose, respondents were considered to have taken into account an element if they mentioned at least one of the attributes each element contains.

With regard to elements, the most voted is *technique*, followed by *tools*, *object*, *scope*, *results* and *agents*. They all appear to be of great interest to the respondents. However, the elements considered least important are the *use level* and the *objective* elements. This trend confirms what was revealed by the study of the schema attributes, and is again surprising. It was to be expected that the respondents would have placed considerable emphasis on knowing about earlier uses of the technique and in finding out what exactly the technique is useful for.

Continuing with the importance of schema elements, the y-axis of Figure 6.12 represents each of the schema elements, whereas the x-axis represents the percentage weight (in terms

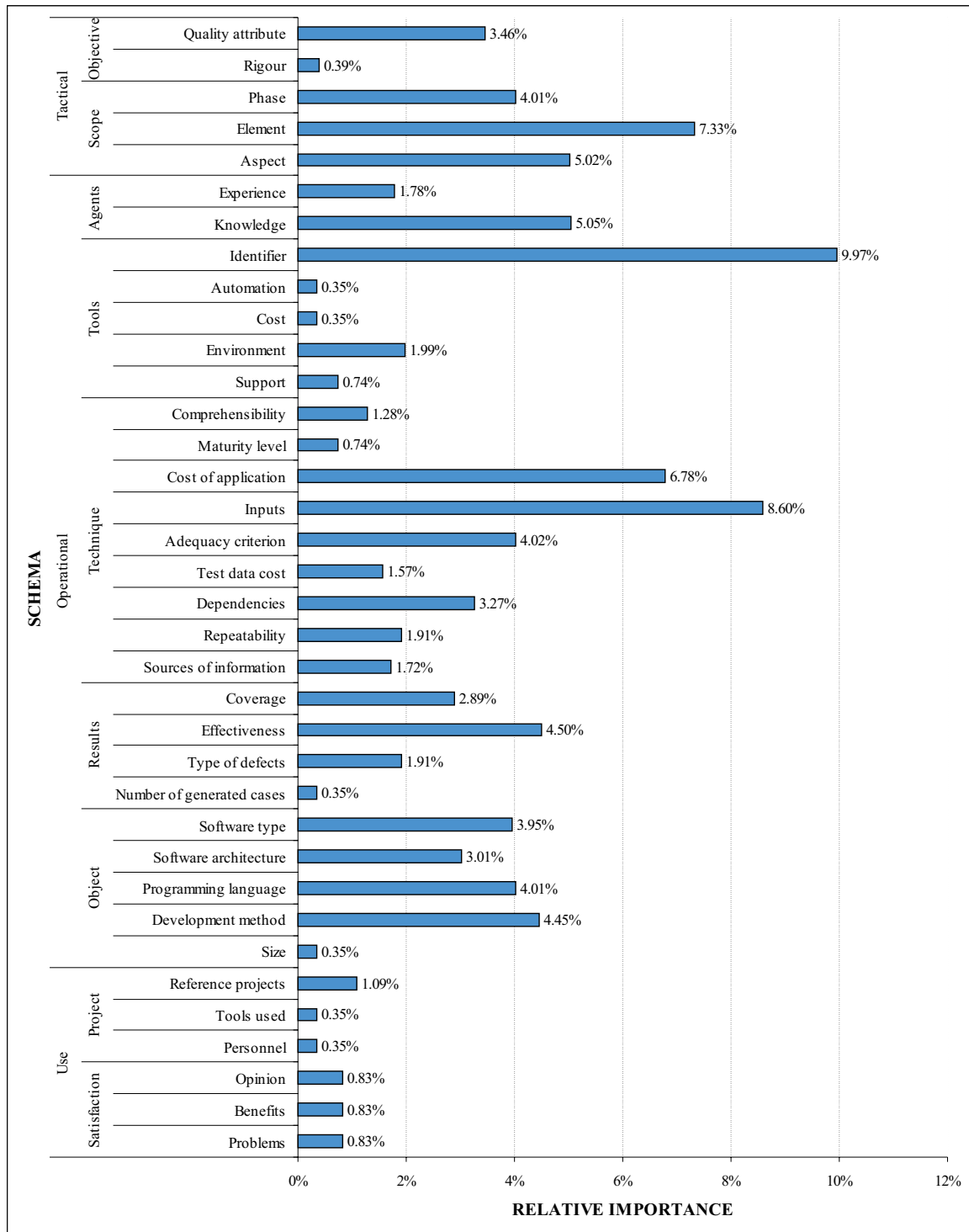


Figure 6.10: Relative importance of each empirical schema attribute.

of importance) of each of the elements the empirical schema contains.

As was the case with the attributes, there are no differences with regard to the proportion between this and the above graph, which means that no further remarks are necessary.

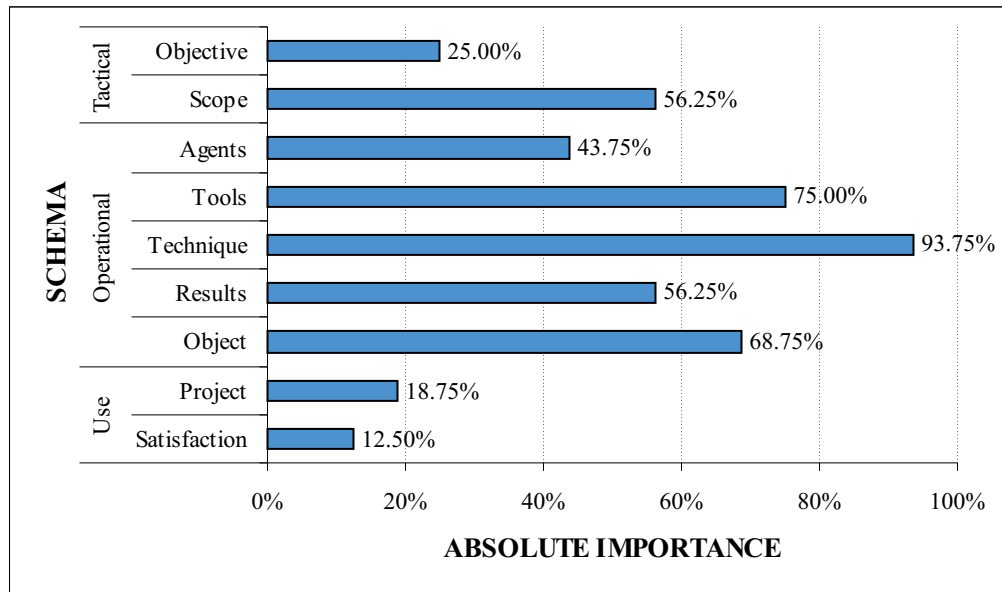


Figure 6.11: Absolute importance of each empirical schema element.

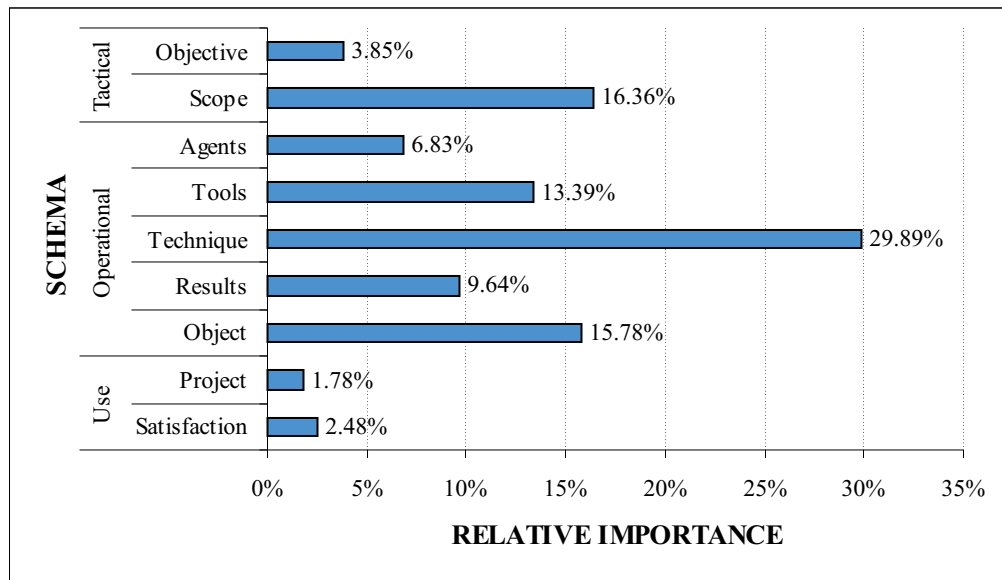


Figure 6.12: Relative importance of each empirical schema element.

Finally, this analysis is concluded with the schema levels. Figure 6.13 reflects both the absolute and relative importance of each level of the schema. The x-axes of both graphs represent the different schema levels, and the y-axes, each of the schema levels by order of importance (percentage). Respondents were considered to have taken into account a level, if they mentioned at least one attribute from the level in question. The main difference between the two graphs is again that the relative importance means that each level can be compared with the others on equal terms, as the sum of percentages is 100% in this case.

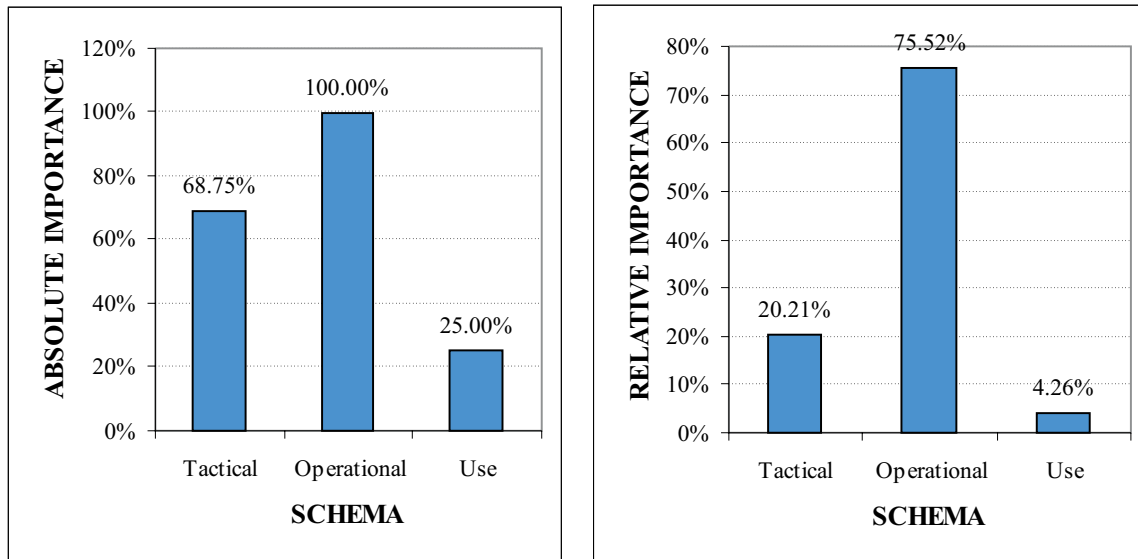


Figure 6.13: Importance of each empirical schema level.

It is clear that both the *operational* and the *tactical* levels are the most voted levels, followed very far behind by the *use* level. This is again interesting for the same reasons as before. It could be justified by formulating the hypothesis that the information contained in the *use* level is very appreciated when it is observed to exist, but people do not seem to be aware that they need and use it for selection purposes.

### 6.4.3 Schema Evolution

Finally, Figure 6.14 reflects the time at which each empirical schema attribute appeared. For more information, readers are referred to Appendix D.4, which outlines the attributes each respondent contributed to the empirical schema.

Recalling the conclusions drawn with regard to the absolute importance of schema attributes, the most voted attributes were: *identifier* of the tools available for using the technique, the *cost of application* of the technique and the *inputs* for the technique, which appear with the first respondent. In this particular case, it can be inferred that the most voted attributes (remember that these were mentioned by over 50% of the respondents) are the first to appear.

As regards the attributes that had been quite often voted (over 35% of respondents considered them to be of interest) the following were prominent: the *knowledge* required to apply the technique, the *element* evaluated by the technique, the *type of software* and *effectiveness*, which appear with the first respondent. Again, it can be deduced that the most voted attributes are the first to appear.

Finally, as regards the least voted attributes, the prominent ones were *use level* attributes, where only one attribute was considered by 20% of the respondents and the *rigour* of the technique, the *cost* and *automation* of the tools, the *number of cases* the technique generates

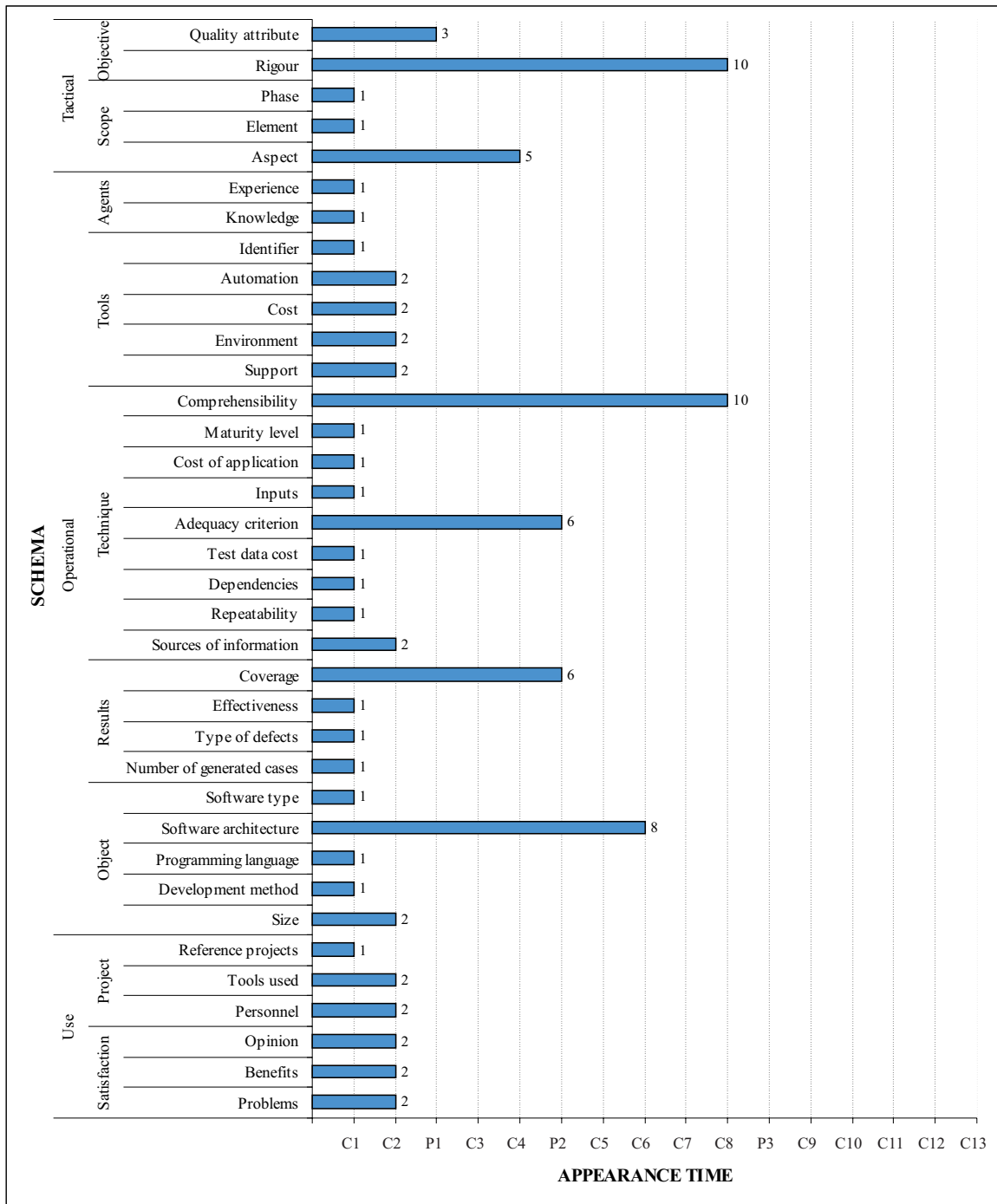


Figure 6.14: Time at which each empirical scheme attribute appeared.

or the *size* the software should have, which were considered by under 7% of respondents.

In the case of the rigour attribute, it appears with the tenth respondent, which provides a possible explanation of why they received so few votes. However, rest of attributes all appeared with the first or the second respondent, which does not explain why they received so few votes. This is especially extrange in the case of the attributes that belong to the use level, considering that during schema use they were found to be the attributes that were most

appreciated (this will be discussed later in Chapter 10).



## Chapter 7

# Synthesis of Perspectives: Proposal of the Preliminary Schema

At this point, we have two characterisation schemas (a theoretical and an empirical schema) that reflect different viewpoints or perspectives of the problem of selecting testing techniques in a software projects. These are: theory, represented by the investigator, and practice, represented by software developers and researchers in the testing area. The next step is to synthesise these two perspectives into one.

The heuristic to be followed for the synthesis is based on the preservation of information: (1) if an attribute appears in the two schemas, it will appear in the synthesised schema and (2) if an attribute appears in only one of the two schemas, it will also appear in the synthesised schema. In no case will the possibility of removing information from the characterisation schema be considered in this phase. The reason is that the fact the investigator has not been able to deduce any attribute mentioned by any respondent from the theory (or vice versa) does not necessarily mean that this attribute is not important or necessary. The omission may be due to a mistake or oversight. Likewise, as there is no way of knowing which attributes are not necessary for selection (this information was never solicited), it is better to play safe.

### 7.1 Rules of Synthesis

The rules of synthesis fulfil the purpose of making a clear and organised synthesis. Whereas in the empirical schema the information provided by each new respondent served to feedback the reference set (assigning questions to existing groups or forming new groups), which then modified the schema, there are now two complete schemas (both having their own structure). These schemas have to be analysed to find out what information they contain and around what elements and levels this information is distributed, to then try to merge this information into a single schema.

Figure 7.1 summarises the contents of the two characterisation schemas.

As shown in Figure 7.1, there is information (where the generic term information means both elements and attributes or levels), which:

LEVEL	ELEMENT	ATTRIBUTE
Tactical	Objective	Quality attribute Rigour
	Scope	Phase Element Aspect
Operational	Agents	Experience Knowledge
	Technique	Tools Comprehensibility Cost of application Dependencies Repeatability Sources of information Adequacy criterion
	Results	Completeness Correctness Effectiveness Type of defects Cost of execution Adequacy degree
	Object	Software type Software architecture Programming language Development method

THEORETICAL SCHEMA

ATTRIBUTE	ELEMENT	LEVEL
Quality attribute Rigour	Objective	Tactical
Phase Element Aspect	Scope	
Experience Knowledge	Agents	Operational
Identifier Automation Cost Environment Support	Tools	
Comprehensibility Maturity level Cost of application Inputs Adequacy criterion Test data cost Dependencies Repeatability Sources of information	Technique	
Coverage Effectiveness Type of defects Number of generated cases	Results	
Software type Software architecture Programming language Development method Size	Object	
Reference projects Tools used Personnel	Project	
Opinion Benefits Problems	Satisfaction	

EMPIRICAL SCHEMA

Figure 7.1: Theoretical and empirical schemas.

1. Appears in only one of the characterisation schemas.
2. Appears in both schemas.

As far as levels and elements are concerned, when one level or element appears in more than one schema, it always appears under the same name. This is due to the fact that as the empirical schema was built, an attempt was made to reuse the names of levels and elements from the theoretical schema. However, this is not the case with the attributes, which means that the type of information they represent has to be examined in more detail to be able to decide whether or not it is really the same.

Before defining the rules of synthesis, two fundamental concepts related to these rules must be defined:

- *Equality.* Two attributes are equal if they bear the same name and belong to the same element and level.
- *Similarity.* Two attributes are similar if they do not bear the same name or do not belong to the same element or same level, although they represent the same or similar concepts.

Accordingly, the following rules are defined for synthesis:

1. The levels and elements of the synthesised schema will be the union of the levels and elements of the two original schemas.
2. Any attributes that appear in just one of the characterisation schemas will appear unchanged in the synthesised schema.
3. Any attributes that appear in both schemas and are equal will appear unchanged in the synthesised schema.
4. Any attributes that appear in the two schemas and are similar will be studied to decide whether they are used to generate one or several attributes.
5. In no case will information be deleted from the characterisation schema.

On the basis of the above rules, the two original characterisation schemas will be synthesised into what will be termed hereinafter the preliminary schema.

## 7.2 Synthesis of the Theoretical and Empirical Schemas

When synthesising the two characterisation schemas, the rules defined above are followed one by one to put together the preliminary schema.

### 7.2.1 First Rule of Synthesis: Levels and Elements of the Preliminary Schema

According to the first rule of synthesis, the levels and elements of the preliminary schema will be the union of the levels and elements of the two schemas.

The theoretical schema is composed of two levels: *tactical* and *operational*, whereas the empirical schema is composed of three levels: *tactical*, *operational* and *use*. After the application of the first rule of synthesis, the levels of the preliminary schema would be: *tactical*, *operational* and *use*.

As regards elements, the theoretical schema is composed of six elements: *objective* and *scope* (tactical level) and *agents*, *technique*, *results* and *object* (operational level). On the other hand, the empirical schema is composed of nine elements: *objective* and *scope* (tactical level), *agents*, *tools*, *technique*, *results* and *object* (operational level) and *project* and *satisfaction* (use level). After the application of the first rule of synthesis, the elements of the schema would be distributed as: *objective* and *scope* (tactical level), *agents*, *tools*, *technique*, *results* and *object* (operational level) and *project* and *satisfaction* (use level).

The result of applying the first rule of synthesis is reflected in Table 7.1.

LEVEL	ELEMENT
Tactical	Objective
	Scope
Operational	Agent
	Tools
	Technique
	Results
	Object
Use	Project
	Satisfaction

Table 7.1: Result of applying the first rule of synthesis.

### 7.2.2 Second Rule of Synthesis: Attributes of a Schema

According to the second rule of synthesis, all attributes that appear once in one of the two schemas will be transferred unchanged to the preliminary schema.

- The attributes that appear in only the theoretical schema are: *correctness* and *adequacy degree* (results element).
- The attributes that appear in only the empirical schema are: *automation*, *cost*, *environment* and *support* (tool element); *maturity level*, *inputs* and *test data cost* (technique element); *size* (object element); *reference projects*, *tools used* and *personnel* (project element); *opinion*, *benefits* and *problems* (satisfaction element).

After applying this rule, the resulting preliminary schema would be as shown in Table 7.2.

### 7.2.3 Third Rule of Synthesis: Equal Attributes of Two Schemas

According to the third rule of synthesis, any attributes that appear in the two schemas and are the same will appear unchanged in the preliminary schema.

These attributes can be located by simply consulting each of the two original schemas and looking for the attributes with the same name that belong to the same element and the same

LEVEL	ELEMENT	ATTRIBUTE
Operational	Tools	Automation Cost Environment Support
	Technique	Maturity level Inputs Test data cost
	Results	Correctness Adequacy degree
	Object	Size
Use	Project	Reference projects Tools used Personnel
	Satisfaction	Opinion Benefits Problems

Table 7.2: Result of applying the second rule of synthesis.

level. These attributes are: *quality attribute* and *rigour* (purpose element); *phase*, *element* and *aspect* (scope element); *experience* and *knowledge* (agents element); *comprehensibility*, *cost of application*, *dependencies*, *repeatability*, *sources of information* and *adequacy criterion* (technique element); *effectiveness* and *type of defects* (results element); *software type*, *software architecture*, *programming language* and *development method* (object element).

The result of applying this rule would be as shown in Table 7.3.

#### 7.2.4 Fourth Rule of Synthesis: Similar Attributes in Two Schemas

According to the fourth rule of synthesis, any attributes that appear in both schemas and are similar will appear in the preliminary schema after examining their meaning. This case calls for a more detailed discussion. The similar attributes are as follows.

##### 7.2.4.1 Identifier versus Tools

Conceptually, these two attributes represent the same thing, namely, the name of the tools available for the technique. The concept to be represented here is both the name and the brand or manufacturer who markets the tool (so that it is completely defined). The tools attribute belongs to technique, whereas the identifier belongs to the tools element. The reason for this is that the theoretical schema does not have any such element. As the element to which this attribute is to belong is tools (it does not make sense for it to belong to technique if there is an element termed thus), it cannot bear the same name so the valid option will be the name *identifier*.

LEVEL	ELEMENT	ATTRIBUTE
Tactical	Objective	Quality attribute Rigour
	Scope	Phase Element Aspect
Operational	Agents	Experience Knowledge
	Technique	Comprehensibility Cost of application Dependencies Repeatability Sources of information Adequacy criterion
	Results	Effectiveness Type of defects
	Object	Software type Software architecture Programming language Development method

Table 7.3: Result of applying the third rule of synthesis.

#### 7.2.4.2 Number of Generated Cases versus Cost of Execution

The concept represented by these two attributes is exactly the same and refers to the effort it will take to execute the test cases that have been generated. According to this definition, it would appear to be more logical to use the attribute *cost of execution*. However, it is important to bear in mind that often the effort of executing a set of test cases depends not only on the number of cases generated, but also on the software in question, the automation of test execution, etc. Therefore, the name *number of generated cases* was thought to be a better option for the preliminary schema.

#### 7.2.4.3 Coverage versus Completeness

As in the two above cases, these two attributes are again found to represent the same concept, namely, the extent to which the objective of the test is achieved. The name coverage is perhaps more intuitive, although it has the drawback of being very closely related to code coverage and, therefore, to a particular type of techniques. Therefore, the name *completeness* (albeit less intuitive) will be used.

The result of applying this rule is shown in Table 7.4.

LEVEL	ELEMENT	ATTRIBUTE
Operational	Tools	Identifier
	Results	Completeness
		Number of generated cases

Table 7.4: Result of applying the fourth rule of synthesis.

### 7.3 Result of Schema Synthesis

Table 7.5 presents the composition of the preliminary schema produced after the synthesis of the theoretical and empirical schemas. Column 1 shows the level to which the attribute belongs, column 2 reflects the element to which it refers, column 3 states the attribute and column 4 gives a brief explanation of the attribute.

### 7.4 Study of Schema Synthesis

Figure 7.2 shows the source of the attributes of the preliminary characterisation schema. Columns 1 to 3 show the schema itself (column 1 indicates the levels, column 2, the elements and column 3, the attributes). The next two columns indicate whether the information represented by an attribute is present in either of the two schemas: theoretical (column labelled T) and empirical (column labelled E). Accordingly, the original composition of the two schemas can be traced back from Figure 7.2.

It follows from Figure 7.2 that the theoretical schema (the schema built on the basis of the literature through the perceptions of the investigator) is composed of two levels, six elements and twenty-four attributes. It also shows that the empirical schema (which was built on the basis of the opinions of different subjects related to software testing) is composed of three levels (the same as appear in the preliminary schema), nine elements (the same as in the preliminary schema) and thirty-six attributes. The preliminary schema is composed of three levels, nine elements and thirty-eight attributes.

It is interesting to note that 14 of the attributes present in the preliminary schema do not appear in the theoretical schema. On the other hand, there are only two attributes that are present in the preliminary schema and not in the empirical schema. This means that the empirical and the preliminary schema are almost exactly the same, except for two attributes. In other words, 58% of the attributes of the preliminary schema are common to the two original schemas, whereas 5% of the other 42% are supplied by the theoretical schema and 37% by the empirical schema. This is an interesting point that it is worthwhile analysing in more detail.

The major omissions of the theoretical schema are the *use* level and the *tools* element. As regards the *use* level, one reason why it is not present is possibly that it was assumed during the investigation that the information provided by the producers with respect to a testing technique is complete enough for consumers not to have to look for other sources of

LEVEL	ELEMENT	ATTRIBUTE	DESCRIPTION
Tactical	Objective	Quality attribute	Quality attribute to be tested in the system
		Rigour	Intensity of the test
	Scope	Phase	Time of development at which the test is to be run
		Element	Elements of the system on which the test acts
Operational	Agents	Aspect	Functionality of the system to be tested
		Experience	Experience required to use the technique
		Knowledge	Knowledge required to be able to apply the technique
		Identifier	Name of the tool and the manufacturer
	Tools	Automation	Part of the technique automated by the tool
		Cost	Cost of tool purchase and maintenance
		Environment	Platform (SW and HW) and programming language with which the tool operates
		Support	Support provided by the tool manufacturer
	Technique	Comprehensibility	Whether or not the technique is easy to understand
		Maturity level	How experimental and/or how well validated the technique is
		Cost of application	How much effort is needed to apply the technique
		Inputs	Inputs required to apply the technique
		Adequacy criterion	Test case generation and stopping rule
		Test data cost	Cost of identifying the test data
		Dependencies	Relationships of one technique with another
		Repeatability	Whether two people generate the same test cases
	Results	Sources of information	Where to find information about the technique
		Completeness	Coverage provided by the set of cases
		Correctness	Test cases to be deleted from the set
		Effectiveness	Capability of the set of cases to detect defects
		Type of defects	Type of defects the technique helps to discover
		Number of generated cases	Number of cases generated per software size unit
		Adequacy degree	Degree to which the adequacy criterion is achieved
		Software type	Type of software that can be tested using the technique
	Object	Software architecture	Development paradigm to which the technique is linked
		Programming language	Programming language with which the technique can be used
		Development method	Development method or life cycle to which the technique is linked
		Size	Size that the software should have to be able to use the technique
Use	Project	Reference projects	Earlier projects in which the technique has been used
		Tools used	Tools used in earlier projects
		Personnel	Personnel who worked on earlier projects
		Opinion	General opinion about the technique after having used the technique
	Satisfaction	Benefits	Benefits of using the technique
		Problems	Problems with using the technique

Table 7.5: Result of perspective synthesis: preliminary schema.



NIVEL	ELEMENTO	ATRIBUTO	T	E
Tactical	Objective	Quality attribute		
		Rigour		
	Scope	Phase		
		Element		
		Aspect		
Operational	Agents	Experience		
		Knowledge		
	Tools	Identifier		
		Automation		
		Cost		
		Environment		
		Support		
	Technique	Comprehensibility		
		Maturity level		
		Cost of application		
		Inputs		
		Test data cost		
		Dependencies		
		Repeatability		
		Sources of information		
		Adequacy criterion		
	Results	Completeness		
		Correctness		
		Effectiveness		
		Type of defects		
		Number of generated cases		
		Adequacy degree		
	Object	Software type		
		Software Architecture		
		Programming language		
		Development method		
		Size		
Use	Project	Reference projects		
		Tools used		
		Personnel		
	Satisfaction	Opinion		
		Benefits		
		Problems		

Figure 7.2: Source of synthesised schema attributes.

information. As regards the *tools* element, they were considered important, but details like their *automation* (part of the technique automated by the tool), their *cost*, the *support* provided by the tool vendor, or the *platform* (hardware and software) and programming language (*environment*) that support the tools were not taken into account. This could be due to the fact that pragmatic aspects of the techniques were overlooked. The minor omissions of the theoretical schema are some attributes of the *technique* element (*maturity level*, *inputs* and *test data cost*) and the *object* attribute (*size*), which corroborate the above supposition that pragmatic aspects of the testing techniques were overlooked when building the theoretical schema.

The empirical schema, on the other hand, has only minor omissions, as the respondents failed to detect only two attributes of the final schema: *adequacy degree* and *correctness*, both belonging to the *results* element. The absence of these concepts in the empirical schema is

probably due to the fact that not enough people were interviewed or that the set of possible respondents was not satisfactorily covered. If all this information is linked to the respective perspectives upon which the schemas were based, the following conclusions can be drawn.

- The investigator's view of the selection problem leads to both the use and the pragmatic aspects of the testing techniques having been omitted from the schema.
- The respondents have a fuller view of the selection problem, as they only missed two schema attributes, which correspond to the theoretical aspects of the technique. This means that the coverage of the set of possible respondents was not optimal.

## Chapter 8

# Improvement of the Schema: Expert Peer Review

At this point, a schema has been built that reflects, on the one hand, the investigator's view of the testing technique selection problem and, on the other, both the information needs of consumers for the purposes of selection and the provision of information required according to producers to define the properties of a testing technique.

However, although the schema includes a range of opinions, the grouping of information as elements and levels was furnished by the investigator, as were the names of the schema attributes. This harks back to the fact that, at this stage of problem solving, although many people have expressed their opinion about the information that the schema should contain, nobody has actually seen yet the result, that is, the schema itself. For this purpose, a series of experts was selected, to whom the characterisation schema was sent. They were asked to examine and give their opinion on the schema.

It was decided to use testing experts, as they respond to the profile of the type of reviewer required. Thanks to their lengthy experience in both the theory and practice of the testing process and their knowledge of the people involved in this process, the experts are in a position to judge the schema. Furthermore, testing experts can be considered to represent another level involved in the selection problem, apart from consumers and producers. They are people with lengthy experience in the testing area, often encompassing both viewpoints.

The goal of this review is for these experts to examine the characterisation schema and give their opinions on both *form* and *substance*. The opinions on *form* include making judgements about the suitability of schema organisation or of the names that appear in the schema. The opinions on *substance* include making judgements about the existence of possible redundancies in the schema or missing information, etc.

This chapter is organised around the questionnaire that was sent to these experts. Section 8.1 reflects the objectives and organisation of the questionnaire, Section 8.2 presents the method followed to analyse the judgements of the experts, Section 8.3 presents the analysis of the judgements made by the experts, as well as the changes made to the schema as a result, and Section 8.4 presents a summary of how the preliminary schema changed after adding the

suggestions of the experts, as well as the actual schema obtained after including these changes.

## 8.1 Questionnaire for Experts

The questionnaire sent to the testing experts appears in Appendix E.1. This questionnaire is composed of six parts:

1. *Personal particulars.* Following on from the above questionnaires, the aim of this section is to be able to classify the set of individuals who completed the questionnaire. For this purpose, questions are asked concerning the position they now held, the place at which they work and the experience they have in the testing area.
2. *Generic questions.* This section is composed of three questions, designed to be able to find out what the respondent thinks about the use of a characterisation schema to solve the testing technique selection problem, as well as whether the proposed schema can achieve this goal.
3. *Questions related to the Schema Attributes.* This section is composed of five questions. These questions aim to discover whether there is missing or surplus information in the characterisation schema, which are the most and least relevant attributes, as well as any possible redundancies or mistaken names within the characterisation schema.
4. *Questions related to the Schema Elements.* This section is also composed of five questions. These questions aim to discover whether the grouping of attributes as elements in the schema makes sense, whether the schema attributes are properly assigned to elements, whether there are missing or surplus elements within the schema, or whether there are mistaken element names.
5. *Questions related to Schema Levels.* This section is composed of the same five questions as under the above point. In this case, however, they refer to characterisation schema levels instead of elements.
6. *Other remarks.* In this section, the respondents are given the opportunity to enter any remarks or suggestions that they were unable to express earlier and believe to be of interest.

## 8.2 Questionnaire Analysis Method

Taking into account that these are open-ended questionnaires, in which the expert response is a description rather than a quantification, the opinions are analysed critically. This means that the opinions of all the experts on a particular subject are read and understood. Then, the investigator checks whether the opinions are contradictory or coincident and, finally, she

makes a decision on whether or not to accept the suggestion, and, if it is accepted, how this suggestion can be included. The decision on whether or not to accept the experts' suggestions is made according to a series of rules, which are:

1. If the experts disagree, the majority view will be respected.
2. If more than one expert recommends a given change, the recommendation will be taken into account.
3. If only one expert recommends a change, this change will be accepted provided the proposed change is not due to a misinterpretation of the schema, its logic or its contents. In this respect, there are four possible types of unaccepted suggestions:

(a) *Misinterpretation of the objective of the schema.* The schema is designed as an aid for testing technique selection and, as such, will contain information exclusively related to technique selection. Therefore, it is contrary to the objective of the schema to enter information on elements involved in the testing process if this information is not directly related to the technique. The schema now contains information about three of the above-mentioned elements: agents, software and tools. As regards agents and the software to be tested, the only information reflected in the schema is information that limits or restricts the use of the technique. That is, the schema includes the characteristics of the agents who will be capable of applying the technique and of the software on which it will be possible to apply the technique. With respect to tools, the name of the tools that can be used to improve technique use, information related to the constraints on tool use (price, part of the technique that it can automate, software and hardware platform on which the tool can be used and tool support) have also been added. However, the information about tools, which refers to the constraints on their use, has been included in case the use of a testing tool is considered within the project. This information is not oriented to a future selection or evaluation of testing tools (generically), as the information provided in the schema would be insufficient for this purpose. Neither does this research aim to include more information, as the objective of the schema is to select testing techniques, not tools. Therefore, suggestions that aim to add information that is not directly related to the technique will not be accepted.

(b) *Misinterpretation of the logic of the schema.* The schema attributes are grouped around elements that are involved in the testing process, to which they refer. Similarly, the elements are grouped around different testing levels or times. Therefore, it is contradictory to the logic of the schema to move an attribute from one element to another or add a new attribute and (re)assign it to an element to which it does not refer. For example, take the

*number of test cases* belonging to the results element. The number of test cases generated by a technique is a characteristic of the result of applying the technique, so it would not make sense to move it to the technique element, for instance. Therefore, suggestions that aim to assign (new or existing) attributes to elements to which these attributes do not refer will not be accepted.

- (c) *Misinterpretation of the schema contents.* Any suggestions that reveal that the expert has misinterpreted the function or meaning of any schema attribute will not be taken into account. For example, someone might suggest that the attributes *environment* (tools) and *programming language* (object) are the same. Here, the content of the schema has been misinterpreted, as *environment* refers to the specific programming language for which a given tool can be used, whereas *programming language* refers to the programming language for which the technique can be used. It is well known that testing tools are designed specifically for a particular programming language, which is not usually the case with techniques. Hence, the value of these two attributes will not usually be the same. Therefore, it is contradictory to the schema contents to modify the schema if the modification stems from the misinterpretation of its contents.
- (d) When only one expert recommends a given change, this change is not always as evident as when it is recommended by several experts. In this case, it is the opinion of the expert versus the opinion of the investigator. It is sometimes impossible to reconcile the two viewpoints, and it was decided that the opinion of the investigator should take precedence. One such the case is the suggestion to replace the attribute *cost of application* (technique) by *complexity*, as the investigator is of the opinion that a technique can be easy and still take a long time to use. Another is the suggestion to remove the attribute *experience* (agents), as the investigator is of the opinion that not all types of knowledge have to be backed up by experience. Another is the suggestion that the *cost of application* (technique) is the same as the *number of generated cases* (results), as the investigator is of the opinion that the number of cases generated by a technique is not necessarily influenced by the time it takes to apply the technique.

Therefore, it is contradictory to the research project to make modifications in which the investigator does not believe or about which she is not sure.

4. In any case, if the solution of the problem stated by the expert goes beyond structural changes to the schema (for example, build a tool to improve schema use), the suggestion will be accepted, but the solution will be left for future research.

## 8.3 Analysis of Responses

This section analyses the responses provided by the experts. The appropriateness of including the changes suggested by the expert in the characterisation schema is analysed for each response, following the rules explained above. When it is advisable to add a suggestion to the schema, it is also specified how the change will be made.

### 8.3.1 Part 1: Respondents

The questionnaire was originally sent to twelve testing experts, of which only four responded. Therefore, the schema was reviewed by four experts, whose particulars (in alphabetical order are):

- *Expert 1.* Professor of ICMC/USP; expert in testing.
- *Expert 2.* NASA systems analyst at GSFC/Unisys. Her experience in testing includes ten years as a developer and eighteen years as a researcher within the area.
- *Expert 3.* Senior researcher at IEI/CNR (Italy); ten years' experience as a researcher in the software testing area.
- *Expert 4.* Full professor at Portland State University; thirty years' experience as a researcher and professor within the software testing area.

It is clear that all the respondents have lengthy experience mainly as researchers, although some are also experienced as developers within the testing area. Additionally, it is worth mentioning that they are people of international reputation within the testing area. For the exact details on the opinions of each expert, readers are referred to Appendix E.2.

### 8.3.2 Part 2: Generic Questions

As mentioned above, this part contains the generic questions about the schema and its usefulness. It is composed of three questions, the responses to which are analysed below.

#### 8.3.2.1 Utility

The first question refers to whether it is useful for a company to have a repository containing information about testing techniques for the purpose of easing their selection.

Experts 1, 2, and 3 responded that it would be useful and added no further remarks. Although of the opinion that it would be useful, expert 4 doubts that any such repository can be built at present. The problems pointed out by this expert are: (1) that the repository would have to contain too much information (otherwise it would oversimplify reality) and (2) it is not feasible to gather the information that such a repository would have to contain.

There are two currents of opinion, which means that there is no agreement on whether the solution is feasible. As more experts affirm that it is feasible, and the investigator agrees with this opinion, the schema is accepted as feasible. The remark is, however, taken into account, because it advises as to the problem of the amount of information required for selection purposes.

### 8.3.2.2 Effectiveness

The second question refers to whether or not the proposed characterisation schema can achieve the pursued objective.

Although of the opinion that it does fulfil the objective, expert 1 adds that it should be accompanied by a glossary to prevent misunderstandings regarding terminology. Expert 2 responds that it does achieve the objective, and adds no further remarks. Expert 3 responds that it does not fulfil the objectives, because the schema contains too much information, and, if it is to be effective, it should be easy to use and, especially, to maintain. Additionally, this expert also mentions the fact that it is not feasible to get values for some schema attributes, that there are redundancies and that it would be a good idea to provide the person who is to instantiate the schema with a list of possible values from which to choose one. Expert 4 did not answer this question.

Some of the remarks proposed here have led to modifications of the schema:

- Add a glossary of terms to the characterisation schema that explains the terminology used for software testing.
- Add the possible values for each schema attribute. This will make it easier to instantiate for producers.

The other remarks (removal of redundant and unfeasible information) will be addressed later, when examining this aspect of the schema.

Expert 4 stated that the schema was not feasible because too much information was required. Expert 3 now speaks of reducing the information. However, the investigator (and the consulted population) believes that the information it contains is strictly necessary. It does not include superfluous information. However, note is taken of this problem (which is again the amount of information), which is considered as one of the risks to the solution proposed here being finally used. The use of a tool could perhaps be an aid for more easily handling the information contained in the schema.

### 8.3.2.3 Understandability

The third question refers to whether or not the proposed characterisation schema is easy to understand.

Expert 1 responds that it is, adding no further remarks. Expert 2 responds that it is not fully comprehensible, although it is easy enough to understand after having examined the



schema thoroughly a couple of times. Expert 3 responds that it is not easy to understand, giving the same reasons as for the above question: it contains too much information and some correlated attributes, and it is not always clear what the value of the attribute will be. Expert 4 responds that some parts are easy to understand and others are not. Additionally, this expert states that the parts that are easy to understand do not provide useful information for selection purposes, whereas as the parts that are difficult to understand say nothing at all.

Therefore, most of the experts consider that the schema is understandable (to a varying degree). The problems of there being too much information, of some attributes being correlated and of the attribute value not always being evident were stressed.

As mentioned above, the problems of correlation will be dealt with in Section 8.3.3.1.

The problem of there being surplus information was discussed in the preceding section. The proposed solution for this problem is the use of a tool (schema automation).

The problem of the possible attribute values was also dealt with in the preceding section, where it was suggested that the possible schema values for each attribute should be added to the schema.

As of this point, expert 4 did not complete any further sections of the questionnaire, except for the last part. It is assumed that this expert believed that the schema is of no use for its purpose and cannot be understood. So, hereinafter we will continue with experts one to three.

### 8.3.3 Part 3: Attributes

This part, as mentioned above, contains the questions related to the characterisation schema attributes. This part of the questionnaire is composed of five questions.

#### 8.3.3.1 Redundancy

This question refers to the existence of redundant (or correlated) information in the schema.

Expert 1 says that the *maturity level (technique)* and *repeatability (technique)* attributes are really the same characteristic. Expert 2 makes no comment. Expert 3 says that the *cost of application (technique)* and *number of generated cases (results)* attributes are the same, the *quality attribute (objective)* and *defect type (results)* attributes are the same, the *aspect (scope)* and *software type (object)* attributes are the same, the *experience (agents)* and *knowledge (agents)* attributes are the same, and the *environment (tools)* and *programming language (object)* attributes are the same.

These opinions have been dealt with as follows:

- *Maturity level* and *repeatability (technique)*. The expert suggests that the technique maturity level determines repeatability. Furthermore, the objective of this attribute was to find out whether or not a technique is already in use; however, this information can be gleaned from the use level. Therefore, following the advice of

the expert, it was decided to remove one of the attributes. Specifically, the *maturity level* attribute was removed.

- *Cost of application (technique)* and *number of generated cases (results)*. In this case, the objective pursued by the two attributes is not the same. The cost of application of the technique is not directly related to the number of cases generated by the technique. Whether the technique is easy or difficult to understand also influences cost. Therefore, this remark did not lead to a modification of the schema.
- *Quality attribute (objective)* and *defect type (results)*. In this case, the expert suggests that the *quality attribute* attribute is the same as the *defect type*, attribute, and appears to be right. The information reflected by *quality attribute* is the objective of the test. For techniques that are used to detect errors (not for evaluation), the defect type identified by the technique does indeed match the quality attribute to be evaluated. Therefore, the *quality attribute* attribute is replaced by *defect type* which better reflects the information about the defects that the technique can be used to detect (usability, control, reliability, etc.).
- *Aspect (scope)* and *software type (object)*. In this case, the attributes do not appear to be related. *Aspect* refers to the functionality of the software to be tested (communications, DB interaction, etc.), whereas *software type* refers to whether the system is a control, management, real-time system, etc. The attributes are considered to reflect different things, but not correlated. Therefore, this remark does not lead to a modification of the schema.
- *Experience* and *knowledge (agents)*. A lot of people do indeed consider knowledge to be backed by experience. However, theoretical knowledge should be kept separate from the practical experience of the subjects (distinction between topic and episodic knowledge already discussed in Chapter 3) using two different schema attributes. Therefore, the suggestion of this expert will not be taken into account, and both attributes will be retained.
- *Environment (tools)* and *programming language (object)*. These two attributes are not the same, as even if a technique is useful for any structured language (C, Modula-2, Pascal, etc.), a given tool may not cover all the existing structured programming languages. Accordingly, the remark does not lead to any modification of the schema.

### 8.3.3.2 Importance

This time the experts are asked which characterisation schema attributes they consider to be more or less important. The objective of this question is to compare the expert responses with the study of importance carried out for the empirical schema.

The most important attributes in the empirical schema were, in decreasing order of votes, the existence of tools (*identifier*), the *cost of application* of the technique, the technique

*inputs*, the *knowledge* required to apply the technique, the *element* tested by the technique, the *type of software*, and technique *effectiveness*.

Expert 1 considers the most important attributes to be the *cost of application*, which appears in second place in the above ranking, *effectiveness*, which appears in sixth place in the empirical schema, and the *defect type*, which is not among the most voted attributes in the empirical schema. Expert 2 considers the most important attributes to be the *technique* element, some of the attributes of which appear in the ranking of the most voted attributes in the empirical schema, the attribute *software architecture*, which is not among the most voted attributes of the empirical schema, and *automation*, which does not appear in the ranking of the most voted attributes of the empirical schema. Expert 3 considers the most important attribute to be the *cost of application*, which appears in second place in the empirical schema.

It follows from this that both experts and producers and consumers have more or less the same idea as regards the most interesting schema attributes, although the opinions do not fully coincide.

The least voted attributes in the empirical schema include the attributes related to earlier experiences using the technique along with the *rigour* of the technique, tool *cost* and *automation*, the *number of cases* generated by the technique or the software *size*.

Expert 1 mentions the *tools* attributes, which appears as the second most voted element in the empirical schema, and *agents*, which is the fourth least voted element of the empirical schema. Expert 2 mentions *rigour*, which does appear in the ranking of the least voted attributes in the empirical schema, and the *adequacy degree*, which does not appear in the lists for the empirical schema. Expert 3 mentions the *aspect* attributes and *correctness*, which do not appear in either of the two empirical schema lists.

Again, it is found that both producers and consumers and experts again share some of the opinions with respect to the least important attributes of the schema, although they do not fully coincide.

It is important to note, on the other hand, that some of the least important attributes for the experts have already been removed on the recommendation of the testing experts. This information is analysed in the following section.

### 8.3.3.3 Superfluousness

This time the experts were asked which schema attributes they believed to be superfluous.

Experts 1 and 2 do not believe that any characterisation schema attribute is superfluous. Expert 3 thinks that the following attributes are superfluous for the following reasons: *rigour (objective)*, because it does not depend on the testing technique but on the person running the test; *adequacy degree (results)*, because it is not known in most cases and is not important for all techniques; and *correctness (results)*, because it is impossible to ascertain and is confusing. As regards the latter attribute, the expert mentions that perhaps precision was meant, that is, avoidance of redundant test cases, which is not the same thing.

As regards these opinions, the decisions taken were:

- *Rigour (objective)*. Indeed, as the expert suggests, the rigour of a technique is more closely related to the project or the situation than to the technique. Additionally, the information to be reflected by rigour (test exhaustiveness or rigour) can be extracted from technique effectiveness. The process is really as follows: the person who is to apply the technique knows what the rigour of the test should be and, on this basis, selects the technique that behaves best or is more effective. Therefore, this attribute is removed and the *effectiveness* attribute is moved to the *objective* element to substitute the *rigour* attribute.
- *Adequacy degree (results)*. According to one of the experts, it is not possible at present to find out the adequacy degree of a testing technique for the project in question. Additionally, the adequacy degree is not a significant characteristic for all testing techniques. One of the characteristics that the schema was to have was to represent the testing techniques using an invariant set of attributes. If this attribute were to be retained, the schema would not be invariant for the techniques. Therefore, this attribute is removed.
- *Correctness (results)*. According to one of the experts, correctness is not something that can be measured, as it is not possible to predict how many test cases will have to be deleted from a generated set. The expert also claims that this attribute is confusing, and suggests changing it for *precision*. Following the expert's advice, the attribute *correctness* will be called *precision*.

#### 8.3.3.4 Completeness

The aim of this question was to find out whether the experts believe that the schema does not account for all the information required to select testing techniques.

Expert 1 would add: *support for other activities: debugging, maintenance, etc.*, to the *tools* element, because software tests are closely related to debugging and maintenance; *ease of integration* to the *tools* element, because a testing tool should be easy to integrate with other development tools; *complexity* to the *technique* element, because it would provide information about how easy or difficult it is to apply the technique; *laboratory package* to the *technique* element, because it would provide facilities for replicating experiments and evaluating the technique, and *pilot study availability* to the *technique* element, because it provides information about aspects of technique application and provides means for comparison with other techniques. Expert 2 would add the *purpose* attribute to the *objective* element, as the utility of the technique should be apparent as soon as possible. Expert 3 would not add any new information.

- *Purpose (objective)*. In the analysis of earlier questions, it was decided to move the *defect type* attribute from *results* to *objective* and remove the *quality attribute* attribute (*objective*). However, following this suggestion, the *purpose* attribute will

be added to the *objective* element to define whether the purpose of the technique is to predict or correct.

- *Support for other activities: debugging, maintenance, etc. (tools)*. It was decided not to add this new attribute to the schema, mainly because the purpose of the schema is to characterise testing techniques and not tools. So much information is required to select tools that another schema could be built entirely devoted to characterising testing tools.
- *Ease of integration (tools)*. It was decided not to add this new attribute to the schema for the same reason as the above attribute was not added.
- *Complexity (technique)*. It was decided not to add this attribute to the schema, as the *cost of application* of the technique is thought to reflect the same concept, which is the effort required to apply the technique. More complex techniques will call for a bigger application effort than less complex techniques.
- *Laboratory package (technique)*. It was decided not to add this attribute to the schema, as it is considered that the requested information can be obtained either from the *sources of information* attribute or the use level for another project type.
- *Pilot study availability (technique)*. It was decided not to add this attribute to the schema for the same reason as above.

#### 8.3.3.5 Names

In this case, the experts were asked whether they thought that the name of any of the characterisation schema attributes should be changed, either because it was not meaningful or not clear enough.

Expert 1 suggests changing the name of the *sources of information* attribute and calling it *sources of information and training material*. This is not considered necessary, as the schema names should not be too long and *sources of information* is generic enough to also contain this sort of information. It would suffice perhaps to explain in the attribute description that the attribute can also contain training material. Experts 2 and 3 do not suggest changing any name.

### 8.3.4 Part 4: Elements

As mentioned above, this part contains the questions related to the elements of the characterisation schema. This part of the questionnaire is composed of five questions.

#### 8.3.4.1 Groupings

This question aims to find out whether the grouping of the different schema attributes as elements is coherent and makes sense.

All the experts, including expert 1, 2 and 3, believe that it is coherent to group the different attributes as elements, and make no further remarks.

#### 8.3.4.2 Attribute/Element Relationship

The purpose of this question is to find out whether the experts think that the current grouping of attributes as elements is correct.

Expert 1 would move the *development method* attribute of the *object* element to the *objective* element. Expert 2, on the other hand, considers that the grouping is sound. Expert 3 would move the *cost of application* attribute from the *technique* element to the *results* element.

The decisions made with regard to these remarks are:

- *Development method* (from *object* to *objective*). The change suggested by the expert will not be made, as the *objective* element is related to the purpose of the test and not the characteristics of the software to be tested, which are to be found in the *object* element.
- *Cost of application* (from *technique* to *results*). The change suggested by the expert will not be made, as the *cost of application* refers to the technique, not to the generated test cases. Additionally, the cost of application of the test cases can be deduced from the *number of cases*, in the *results element*.

#### 8.3.4.3 Superfluousness

This time the experts were asked whether they believed that any of the elements of the characterisation schema should be removed. In this case, none of the three indicated that any of the schema elements should be removed.

#### 8.3.4.4 Completeness

This question asked the experts whether they would add any new element to the schema.

Expert 1 believed the *pilot project* element should be added, because it would provide information about the aspects of technique application, as well as means for comparison with other techniques. Experts 2 and 3 did not believe it necessary to add any new element to the schema.

It was decided not to add this element to the schema, as this information is covered by the existing *project* element as mentioned for the attributes.

#### 8.3.4.5 Names

This question reflects the possibility of there being elements in the schema whose name is not meaningful, clear or correct.

Expert 1 did think that the *technique* element should be renamed as *technique/criterion* and the *results* element as *results of application*. However, experts 2 and 3 did not believe it to be necessary to rename any schema element.

The decisions made were:

- *Technique* as *technique/criterion*. It was not considered necessary to change the name, as the *technique* element contains an attribute termed *adequacy criterion* of the technique and the suggested change could lead to confusion.
- *Results* as *results of application*. The term *results* appears not to be clear enough, as results of application could refer to the actual technique or to the generated test cases. Therefore, the name is changed to *test cases*, which is clearer.

### 8.3.5 Part 5: Levels

As mentioned above, this part contains the questions related to the characterisation schema levels. This part of the questionnaire is composed of five questions.

#### 8.3.5.1 Groupings

Experts 1, 2 and 3 all believe that it is coherent to group the elements as levels and make no further remarks.

#### 8.3.5.2 Elements/Level Relationship

The purpose of this question is to find out whether the elements are incorrectly grouped, that is, whether any of the experts would move any element from one level to another.

None of the experts expressed the wish to move elements from one level to another.

#### 8.3.5.3 Superfluosness

This question aims to find out whether there are redundant or correlated levels in the characterisation schema.

As for the above question, none of the experts thought it necessary to remove any of the three levels that the characterisation schema now contains.

#### 8.3.5.4 Completeness

This question aims to shed light upon other levels that the experts believe to be of interest for selection purposes.

As for the above questions, none of the experts thought it necessary to add any new level to the schema.

### 8.3.5.5 Names

This question aims to find out whether the experts would change the name of any of the levels in the schema.

Expert 1 does not consider that any name should be changed. On the other hand, experts 2 and 3 agree that the *use* level should be renamed and called *historical*, as the term use is confusing. The term use appears to evoke finding out details of how to use the technique and not information about past uses of the technique. Therefore, the *use* level has been renamed *historical* level.

### 8.3.6 Part 6: Other Remarks

In this part of the questionnaire, the experts were asked to indicate any other type of remark that they were unable to make when answering the above questions. In this part of the questionnaire, the experts stated the major problems or weaknesses they detected in the schema. Although they have been addressed earlier, there follows a summary of the experts' indications in this part of the questionnaire and how they were dealt with.

#### 1. *Expert 1.*

- It is necessary to specify the terminology used. The expert recommends adding a glossary to the schema.

A glossary of terms was added to the characterisation schema, explaining the terminology followed for software testing.

- The completeness and correctness of the set of test cases are part of the adequacy criterion.

This remark did not lead to any change in the schema. Completeness and correctness are viewed as attributes of the set of test cases generated by the technique, not of the actual technique.

- The number of test cases of results should be replaced by cost of application. This remark did not lead to any modification of the schema. This is because the cost of application of a technique is not considered to reflect the same information as the number of generated cases.

- The application domain is not very well characterised. Only one type of software is accounted for.

The application domain is the area for which the software is developed (banks, insurance, management, etc.). A lot of information can indeed be given on the application domain of a technique (type, problem-solving type, whether there is software of this type for the domain in question, etc.). However, it is not believed to be relevant.

#### 2. *Expert 2*



- The tactical level is not well defined. At this level it should be clear what the purpose of the test is. The expert suggests adding an attribute termed purpose.

As regards this remark, the *quality attribute* attribute (*objective*) has been removed and replaced by the *defect type* attribute (*results*). The *purpose* attribute has also been added to the *objective* element.

- Rigour does not make much sense in the schema, as it is something that is associated with the person using the technique not with the technique. This should not be taken to mean that knowing the rigour with which the software is to be tested is not important, it is simply not related to the technique.

Therefore, the *rigour* attribute has been removed and has been replaced by the *effectiveness* attribute.

- The schema is not represented very well. The expert suggests representing it as layers, where the first layer is the tactical level, followed by the operational level and then the use (historical) level (representation as a table rather than a chart).

The representation with which the experts were supplied was as a chart as shows Figure 8.1. This representation has been changed to tabular format following this advice.

<i><b>TACTICAL LEVEL</b></i>		<i><b>USE LEVEL</b></i>	
<b>OBJECTIVE</b>	<b>SCOPE</b>	<b>PROJECT</b>	<b>SATISFACTION</b>
- Quality attribute - Rigor	- Phase - Element - Aspect	- Reference projects - Tools used - Personnel	- Opinion - Benefits - Problems
<i><b>OPERATIONAL LEVEL</b></i>			
<b>TECHNIQUE</b>	<b>OBJECT</b>		<b>RESULTS</b>
- Comprehensibility - Maturity level - Cost of application - Dependencies - Repeatability - Sources of information - Inputs - Test data cost - Adequacy criterion	- Software type - Software architecture - Programming language - Size - Development method		- Completeness - Correctness - Effectiveness - Type of defects - N. of test cases - Adequacy degree
	<b>AGENTS</b>		<b>TOOLS</b>
	- Experience - Knowledge		- Identifier - Automation - Cost - Environment - Support

Figure 8.1: Representation of the schema as a chart.

### 3. Expert 3.

- There are too many attributes in the schema, which makes it not only

difficult to use but also to maintain (which is worse).

One possible solution to this is to automate the schema as a tool, as none of the experts wanted to delete attributes from the schema.

- The expert suggests defining two different sorts of attributes -essential and optional attributes- in order to discriminate the type of information.

In principle, all the attributes are optional, that is, during schema instantiation, an attempt will be made to instantiate all the known attributes. It is not essential to instantiate the schema in its entirety.

- There is information in the schema that is not feasible to gather.

This refers to the *adequacy degree*. This attribute was removed as a result of this remark.

- For attributes whose value is not clear (non-numerical attributes, like development method, etc.), the expert suggests providing a list of values to make it easier to complete.

As regards this remark, the possible values for each attribute were added to the characterisation schema.

- There are some attributes that are not very precise (it is not easy to see what their value could be).

This remark was solved in the same way as the above.

- Generally, it is recommended not to include attributes in the characterisation schema whose value is difficult to find.

This remark will be addressed by letting the schema evolve with use. Any attributes that are not used will be removed.

#### 4. Expert 4.

- This expert states that it is not feasible to complete a schema of this sort.
- The expert states that the testing technique selection problem is so complex that if all the schema variables were to be taken into account it would be unmanageable and any simplification would lead to an unreliable schema.
- It is not possible to get the information asked for in the schema.
- People do not agree on what the attribute values for techniques are, which means it is impossible to get anything reliable.

As discussed in Chapter 1 of this research, the investigator is perfectly aware that it is not easy to solve the problem of gathering all the right information for selecting testing techniques. However, it was decided not to put off the attempt at solving the problem any longer in view of the grave consequences of a poor selection. The set of information that has been obtained as relevant for selection is stable, a lot

of people have been involved in its construction, and the other experts think that it is necessary and sufficient. Therefore, the schema is considered to be feasible. The investigator also is aware that there is information in the schema that cannot be gathered at present. However, it is important to bear in mind that one of the objectives of the schema is to provide testing researchers with new research goals that are in line with developers' needs. Finally, it is important to stress that the proposed characterisation schema aims to reflect the knowledge there is about techniques. The problem of contradictory information is inherent to current knowledge about testing techniques, and this has to be addressed with or without a schema (this problem is beyond the scope of this research). However, the schema can help people to solve this problem insofar as it will make it more patent than it is at present.

## 8.4 Schema after Peer Review

Briefly, the changes made to the preliminary schema can be summarised as follows:

- Five attributes have been deleted: three from the *tactical* level (*quality attribute*, *rigour* and *phase*) and two from the *operational* level (*maturity level*, and *adequacy degree*).
- The *correctness* attribute of the *operational* level was replaced by another named *precision*.
- Two attributes have been moved from the *operational* level to the *tactical* level (*effectiveness* and *defect type*).
- A new attribute, termed *purpose*, was created and placed in the *objective* element.
- The *results* element has been renamed as *test cases*.
- The *use* level has been renamed as *historical* level.

Table 8.1, Table 8.2 and Table 8.3 reflect the schema after the expert peer review.

LEVEL	ELEMENT	ATTRIBUTE	DESCRIPTION POSSIBLE VALUES
Tactical	Objective	Purpose	Type of evaluation and quality attribute to be tested in the system Two values: find defects, assess software
		Defect type	Defect type Defect types detected in the system (control, assignation,initialisation, usability, performance, etc.)
		Effectiveness	What capability the set of cases should have to detect defects Percentage
	Scope	Element	Elements of the system on which the test acts (function, procedure, system, subsystem, etc.)
		Aspect	Functionality of the system to be tested (communications, database, security, etc.)
	Agents	Knowledge	Knowledge required to be able to apply the technique (cyclomatic complexity, flow charts, etc.)
Experience		Experience required to be able to apply the technique (tool understanding, etc.)	
Identifier		Name of the tool and the manufacturer Two values: [tool name] and [company name]	
Automation		Part of the technique automated by the tool (flow chart, mutant generation, test case generation, etc.)	
Operational	Tools	Cost	Cost of tool purchase and maintenance Two values: [purchase cost] and [maintenance]
		Environment	Platform (SW and HW) and programming language with which the tool operates Three values: [SW requirements], [HW requirements] and [programming language]
		Support	Support provided by the tool manufacturer (24-hour hotline, technical assistance, etc.)

Table 8.1: Schema after expert peer review (1/3).

LEVEL	ELEMENT	ATTRIBUTE	DESCRIPTION POSSIBLE VALUES
Operational	Technique	Comprehensibility	Whether or not the technique is easy to understand (high, medium, low)
		Cost of application	How much effort it takes to apply the technique (high, medium, low)
		Inputs	Inputs required to apply the technique (requirements, code, design, etc.)
		Adequacy criterion	Test case generation and stopping rule family (data flow, flow control, etc.) and technique (sentence coverage, etc.)
		Test data cost	Cost of identifying the test data (high, medium, low)
		Dependencies	Relationships of one technique with another
		Repeatability	Two values: [technique] and dependency type (should be applied before, after, should never be used with, etc.)
		Sources of information	Whether two people generate the same test cases (yes, no)
		Completeness	Where to find information about the technique (a person, a book, an article, an experiment, etc.)
		Precision	Coverage provided by the set of cases Percentage
	Test cases	Number of generated cases	How many repeated test cases the technique generates Percentage
		Software type	Number of cases generated per software size unit Formula
		Software architecture	Type of software that can be tested using the technique (real time, batch, iterative, expert system, etc.)
	Object	Programming language	Development paradigm to which it is linked (call and return, OO, etc.)
		Development method	Programming language with which it can be used (structured, functional, logical, real time, concurrent, etc.)
		Size	Development method or life cycle to which it is linked (prototyping, reuse, waterfall, knowledge-based system, etc.)
			Size that the software should have to be able to use the technique (number in KLOC)

Table 8.2: Schema after expert peer review (2/3).

LEVEL	ELEMENT	ATTRIBUTE	DESCRIPTION POSSIBLE VALUES
Historical	Project	Reference projects	Earlier projects in which the technique has been used [project name]
		Tools used	Tools used in earlier projects [tool name]
		Personnel	Personnel who worked on earlier projects [people's names]
	Satisfaction	Opinion	General opinion about the technique after having used it [sentence or paragraph explaining the opinion]
		Benefits	Benefits of using the technique [sentence or paragraph explaining the benefits of the technique]
		Problems	Problems with using the technique [sentence or paragraph explaining the drawbacks of the technique]

Table 8.3: Schema after expert peer review (3/3).

## Part III

# VALIDATION AND CONCLUSIONS





## Chapter 9

# Empirical Evaluation

The objective of this first evaluation of the characterisation schema is to study some of the hypotheses formulated in Chapter 3. In this case, the characterisation schema will be verified to evaluate *static* aspects of the schema, like feasibility from the viewpoint of producers and consumers and schema flexibility.

The primary objective of this empirical evaluation is to find out whether it is actually possible to find the information contained in the characterisation schema, how this information can be used during selection and whether it is possible to instantiate the schema for any of the existing testing techniques.

For the purpose of checking schema flexibility and schema feasibility from the producer viewpoint, a group of techniques will be selected and instantiated. In order to check schema feasibility from the consumer viewpoint, a situation will be set out in which the schema has to be used. That is, a decision will have to be made as to which technique or techniques are to be selected.

Table 9.1 indicates the questions to be answered during empirical evaluation.

FOCUS		QUESTION
ASPECT	VIEWPOINT	
Feasibility	Producer	1. Is it possible to describe at least one testing technique?
	Consumer	2. Is it possible to decide whether or not to use a testing technique in at least one situation?
Flexibility	Producer	3. Can the schema be used to characterise any existing technique?
		4. How formal does a technique have to be for it to be possible to characterise it?

Table 9.1: Questions to be answered during empirical evaluation.

### 9.1 Choice of the Workload

One important decision to be made before instantiating the schema is the choice of the set of techniques for which the schema is to be instantiated. Also, for the purposes of checking

schema feasibility from the consumer viewpoint, a project has to be chosen on the basis of which the selection can be made.

### 9.1.1 Schema Instantiation

The set of techniques chosen must make it possible to check both schema feasibility from the producer viewpoint and schema flexibility. For this purpose, it was decided to select a number of technique families, which covers the variety of techniques between families, and a number of techniques within each family, which covers the variety of techniques within each family. Additionally, it was resolved to choose well-known techniques, as this gives a better understanding of how the schema is instantiated.

Accordingly, the chosen techniques were:

- *Functional testing techniques:* Boundary value analysis and random testing.
- *Control flow testing techniques:* Sentence coverage, decision coverage, path coverage and thread coverage.
- *Data flow testing techniques:* All-c-uses, all-p-uses, all-uses, all-du-paths, and all-possible-rendezvous.
- *Mutation testing techniques:* Mutation and selective mutation.

### 9.1.2 Schema Use

Below, a testing technique selection problem is suggested:

*A system is to be built to manage a car park (concurrent system). At this stage of the project, the quality assurance team has identified the key quality attributes of this software system. These were deduced by examining the characteristics of the software to be developed, as well as its application domain. In this particular case, the essential attributes are correctness, security and timing.*

*Having examined the quality attributes of interest, the question is to decide which techniques would be best suited to evaluate the correctness of the above-mentioned software system, bearing in mind the following project situation. The system is to be coded in ADA, the development team is quite experienced in developing similar systems and, also, it has been found that almost all the errors that the developers make are proper to concurrent programs. The testing team is also experienced in testing this type of systems.*

## 9.2 Analysis of the Results

This section describes the results of instantiating the schema for the techniques mentioned in Section 9.1.1 and trying to solve the above problem.

### 9.2.1 Feasibility

In order to decide whether the characterisation schema is feasible, its feasibility from both the consumer and producer viewpoints has to be checked.

#### 9.2.1.1 Producer Viewpoint

The chosen techniques were instantiated using the literature examined in Chapter 2 as a source of information. The instantiation of one of the techniques is explained in detail below. However, the instantiation of all the techniques that were chosen is shown in Appendix F.

The *decision coverage* technique has been chosen for instantiation as an example. For this purpose, every attribute of the schema will be revised and assigned a value:

#### 1. Tactical level.

##### (a) *Objective.*

- *Purpose.* The purpose of the technique is to detect faults [Beizer, 1990]. So, the value associated with this attribute will be *find faults*.
- *Defect type.* According to Beizer [Beizer, 1990], the type of defects usually detected by this type of techniques is related to program control. Therefore, the value associated with this attribute will be *control*.
- *Effectiveness.* It has been reported that this technique has a 48% probability of detecting faults for at least one project [Wood *et al.*, 1997]. The value associated with this attribute will be a probability of 48% until other data is available.

##### (b) *Scope.*

- *Element.* The family of white-box techniques is mainly used for unit tests [Sommerville, 1992]. So, the value associated with this attribute will be *units*.
- *Aspect.* No specification or constraint has been found with respect to the aspect evaluated by the technique. So, the associated value will, in principle, be *any*.

#### 2. Operational level.

(a) *Agents.*

- *Knowledge.* Having used the technique personally, the investigator is able to state that agents need to be familiar with flow charts to apply the technique (at the understanding level, when a tool is used and at the creation level when a tool is not used). So, the value of this attribute will be *flow charts*. In the absence of references, the personal experience of the investigator in using the technique was used to allocate a value to this attribute.
- *Experience.* Having used the technique personally, the investigator is able to state that experience in using a dynamic and/or static analysis tool is required, as this improves technique use. So, the value of this attribute will be *dynamic and/or static analysis tools*. In the absence of references, the personal experience of the investigator in using the technique was used to allocate a value to this attribute.

(b) *Tools.*

- *Identifier.* There is a wide range of dynamic analysers. By way of an example, Telelogic's Logiscope will be used.
- *Automation.* They automate the generation of the flow charts and measure test case coverage.
- *Cost.* From 3,000 to 6,000 Euros.
- *Environment.* Pascal, C, C++ and Ada.
- *Support.* 24-hour hotline and assistance.

(c) *Technique.*

- *Comprehensibility.* On the basis of the explanations given by Beizer and Sommerville, the technique appears to be easy to understand. The value of this attribute will be *high*. In the absence of references, the personal experience of the investigator in using the technique was used to allocate a value to this attribute.
- *Cost of application.* The cost of application of the technique is highly dependent on whether or not a tool is used. Generally, the cost of application is *medium* as compared with other techniques (white box or mutation). In the absence of references, the personal experience of the investigator in using the technique was used to allocate a value to this attribute.
- *Inputs.* The technique needs source code for application [Beizer, 1990].
- *Test data cost.* As compared with black box techniques, the test data cost is high, although it is even higher for data flow techniques.

Therefore, the value of this attribute will be *medium*. In the absence of references, the personal experience of the investigator in using the technique was used to allocate a value to this attribute.

- *Dependencies*. As the technique focuses on detecting control errors, it is recommended that it be used with techniques that detect processing errors [Beizer, 1990].
- *Repeatability*. The technique is not repeatable [Basili and Selby, 1987]. Different subjects can generate different test cases. The value associated with this attribute will be *no*.
- *Sources of information*. Information about this technique can be found in: [Sommerville, 1992], [Beizer, 1990], [Wood *et al.*, 1997], [Frankl and Weiss, 1991b], etc.
- *Adequacy criterion*. This technique belongs to the control flow family, particularly, decisions coverage [Beizer, 1990]. The values associated with the attribute will be *flow control*, *decision coverage*.

(d) *Test cases*.

- *Completeness*. No value was found for this attribute. So, there is, in principle, no value associated with this attribute.
- *Precision*. No value was found for this attribute. So, there is, in principle, no value associated with this attribute.
- *Number of generated test cases*. The number of generated test cases is exponential with respect to the number of code decisions [Beizer, 1990], [Weyuker, 1988]. So, the value of this attribute will be *exponential with respect to the number of code decisions*.

(e) *Object*.

- *Software type*. No specification or constraint has been found with respect to the type of software with which the techniques can be used. So, the associated value will, in principle, be *any*.
- *Software architecture*. No specification or constraint has been found with respect to the software architecture with which the technique can be used. So, the associated value will, in principle, be *any*.
- *Programming language*. No specification or constraint has been found with respect to the programming language with which the technique can be used. So, the associated value will, in principle, be *any*.
- *Development method*. No specification or constraint has been found with respect to the development method with which the technique can be used. So, the associated value will, in principle, be *any*.

- *Size*. As this is a white box technique, it is not scalable, which means that performance falls as the size of the software increases [Beizer, XX]. Therefore, the associated value will be *medium*.

### 3. Historical level.

#### (a) *Project*.

- *Reference projects*. The investigator did not have access to real projects (i.e. real developments other than research projects and *toy* programs for experiments), in which the technique has been used. So, there is no value associated with this attribute.
- *Tools used*. No reference has been found to tools used in real projects, in which the technique has been used. So, there is no value associated with this attribute.
- *Personnel*. No reference has been found to people who have used this technique. So, there is no value associated with this attribute.

#### (b) *Experience*.

- *Opinion*. People think that it is acceptable, although it must be supplemented by others in order to increase its effectiveness. In the absence of references, the personal experience of the investigator in using the technique was used to allocate a value to this attribute.
- *Benefits*. It is simple to use. In the absence of references, the personal experience of the investigator in using the technique was used to allocate a value to this attribute.
- *Problems*. A dynamic analyser should not be used if the technique is applied to real-time systems, because it falsifies code implementation time. In the absence of references, the personal experience of the investigator in using the technique was used to allocate a value to this attribute.

The findings of the instantiation were:

- There is information that is difficult to find, especially information related to tools and reference projects. This may be due to the fact that the sources of information consulted to instantiate the schema (books and research articles) were not as varied as they should have been. For example, with regard to the existence of tools, access to this information could have been gained by asking companies that manufacture this sort of software or using general-purpose reports, such as those published by Ovun or Gardner Group. With regard to the information related to reference projects, it would have been able to be ascertained by asking people working at

software development companies who are experienced in testing. Neither of these two things was considered necessary for a preliminary test of the schema and was left for its later deployment. However, a cultural change has to take place at companies for it to be possible to get reliable information about the past uses of a testing technique. This would involve the *post-mortem* analysis of projects to weigh up the results of using the techniques and decide, *a posteriori*, whether they were the best-suited techniques for the project.

There were also two schema attributes (*precision* and *completeness*) belonging to the *test cases* element, whose value was not found anywhere. This casts doubts upon the advisability of these two attributes appearing in the schema. However, as one of them (*completeness*) was generated in both the theoretical and empirical schemas and was not considered unsuitable by the experts and the other (*precision*) appeared at the suggestion of an expert, they will be retained at least until schema deployment in a real environment ratifies their suitability.

- Contradictory information is often found about the testing techniques. This is inevitable, because as long as the parameters that affect the use of a testing technique are not perfectly defined, some may not be studied. If this were to occur, a possible result would be that two uses of a technique appear to be equal (the values of all the parameters coincide except the one not taken into account), although they are not in actual fact. And, therefore, the results of using it in the second situation will be different to those predicted by the first. The studies carried out on testing techniques would have to be as rigorous as possible and, thus, reflect the information more correctly in order to output non-contradictory information.
- The metrics used to fill in some attributes are not easy to interpret. For example, for technique effectiveness, one often finds *probability of finding a given fault* as the associated metric. However, this attribute should really reflect the *percentage of faults that the technique can detect*. Can both metrics really be considered to reflect the same information? Or, contrariwise, do they reflect different things? This problem could be solved if the metrics expressly asked for by the schema were used every time studies were carried out on testing techniques. This would raise the confidence level of the instantiated schema.

As a final conclusion on schema feasibility, it can be said that the schema can be instantiated, in exchange, for sizeable structural changes in the organisation, e.g. the implementation of an Experience Factory [Basili *et al.*, 1994]. This involves introducing a series of activities at the end of each project, which make it possible to gather all the data required to update the repository. This has to be done by a team employed exclusively on this task, if it is to be effective. Also, the best way of making use of the characterisation schema, at least until the theory on testing is more solid, is thought to be at organisational or departmental level, rather than at community level. The use of the schema on a small scale

provides better guarantees of success, as it will be easier to reach agreement on the values of each schema attribute for the techniques of interest.

However, it is important to stress that the potential of the schema, which is now limited by the existing theory on testing techniques, is much greater. The schema is very useful as an aid for looking for information on testing techniques. This includes information that is at present very dispersed and information that is not now disseminated, namely, the opinion of other people who have used the technique.

### 9.2.1.2 Consumer Viewpoint

The steps of the process proposed in Section 5.6 will be followed one by one to make the selection on the basis of the above-mentioned problem.

1. *Determine bounded variables.* According to the problem statement, it is correctness that is to be evaluated, which means that the *purpose* would be to detect faults in any type of element. The system is to be developed in Ada, which is a *language* for real-time systems. The development team is experienced in developing this type of systems, which means that they are unlikely to make many errors. Table 9.2 shows the associated variables for the example.

LEVEL	ELEMENT	ATTRIBUTE	VALUE
Tactical	Objective	Purpose	Find faults
		Defect type	ANY
		Effectiveness	>50%
	Scope	Element	ANY
		Aspect	ANY
Operational	Object	Software type	Real time
		Software architecture	Concurrent
		Programming language	Ada
		Development method	ANY
		Size	Medium

Table 9.2: Associated variables.

2. *Pre-select an initial set of techniques.* Given the associated variables in Table 9.2, their value was compared with those of the technique contained in the repository in Appendix F. The techniques that will be selected are: F.1, F.2, F.5, F.7, F.8, F.9, F.10, F.11, F.12 and F.13. The techniques F.3 and F.4 will be rejected because their effectiveness is low, and technique F.6 will be discarded because it is for object-oriented software.
3. *Identify the best-suited techniques for selection.* Of the pre-selected techniques, there is one that is specific for Ada-style programming languages (concurrency implementation using *rendezvous*). Although there are general-purpose techniques (for all software types) that are more effective, it appears that the technique that is



specific for concurrent software detects the faults proper to concurrency better than the other techniques. Furthermore, technique F.5 states that when the technique is used with concurrent and real-time systems, a dynamic analyser cannot be used as a tool. Additionally, techniques F.8, F.9, F.10, F.11, F.12 and F.13 cannot be used without a tool (which is not available). Therefore, the *all-possible-rendezvous* techniques will be selected. However, the dependency attribute states that the technique should be supplemented with a black-box technique. Observing the black-box techniques in the pre-selected set (F.1 and F.2), it is found that the *random testing* technique is useful for people with experience in the type of tests to be run and will, therefore, also be selected.

The finding is, therefore, that **it is possible to make at least one the selection** using the characterisation schema.

### 9.2.2 Schema Flexibility

As regards schema flexibility, as can be seen from Appendix F, it was possible to satisfactorily instantiate all the testing techniques that were originally selected. This means that the schema could be used to instantiate thirteen testing techniques from four different families. Of course, this does not mean that the schema is totally flexible. It would be necessary to instantiate the schema for *all* existing testing techniques to make such a claim. However, the fact that a series of techniques that are representative of existing techniques have been able to be instantiated without any problem indicates that the schema is flexible enough to be able to instantiate the huge majority of, if not all, testing techniques.

## 9.3 Conclusions on the Empirical Evaluation

1. Is it possible to describe at least one testing technique?

The response is yes, it can be used to instantiate at least one technique, as shown in Section 9.2.1.1.

However, as can be seen from Appendix F, none of the techniques could be completely instantiated. Moreover, of all the characterisation schema attributes, there are two (completeness and precision) that could never be allocated a value. This is not related to the feasibility of the schema, it means that more research has to be done in the testing techniques area.

2. Is it possible to decide whether or not to use a testing technique in at least one situation?

As shown by the example in Section 9.2.1.2, the response to this question is *yes, in theory*. Yes, because it was possible to select techniques for the particular example proposed. (This should not be taken to mean that it is always possible to select techniques, but, it is possible in at least one situation). However, this is theoretical,

because this is an imaginary case. The use of an imaginary case means that the information is set out as a statement rather than existing in a real project. Another point to be stressed is that the selection was made by the principal investigator rather than by a third party. However, techniques will be selected by third parties later on in Chapter 10.

Finally, it is interesting to note that testing techniques can be selected even if the schema is not completely instantiated.

3. Does the schema characterise any existing technique?

Although it was not possible to completely instantiate the characterisation schema for the chosen techniques (as shown in Appendix F), no other problems were encountered when instantiating the schema. This means that no problems were found concerning the adequacy of the framework used to represent the schema to characterise a testing technique.

4. 4 How formal does a technique need to be able to be characterised?

As the only problem encountered when instantiating the characterisation schema for all the testing techniques (as shown in Appendix F) was missing information, it can be concluded that the nature of the testing techniques does not influence their characterisation. However, the availability of information it is likely to have an impact on the decision on whether or not to select a given technique.

## Chapter 10

# Experimental Evaluation

### 10.1 Objective of the Experiment

The goal of this second evaluation is to conclude the study of the hypotheses formulated in this piece of research. This time, the characterisation schema is tested to evaluate aspects of interest to consumers, like completeness, effectiveness, efficiency, usability or satisfaction.

The primary objective of this experimental evaluation is to try to understand how schema use affects the testing technique selection process. That is, whether the schema is really an improvement on selection using other resources (basically books) or, contrariwise, it is preferable to carry on using the traditional selection process, because the schema increases the workload; and whether the schema is of assistance to consumers in the sense that it improves the work they do.

Accordingly, the generic null hypothesis of the experiment is:

$H_0$ : There is no difference in the selection process using the characterisation schema and books for all software projects.

This would lead to four alternative hypotheses:

$H_1$ : The selection process improves using the characterisation schema as compared to books for all software projects.

$H_2$ : The selection process improves using books as compared to the characterisation schema for all software projects.

$H_3$ : The selection process is better, for some projects as compared to others, irrespective of the method used.

$H_4$ : Depending on the software project, the process improves using either books or the characterisation schema.

The null hypothesis will be divided into as many subhypotheses as aspects of the schema are to be examined during the experiment. Table 10.1 shows the different aspects of

the schema to be examined during the experiment, which are completeness, effectiveness, efficiency, usability and user satisfaction.

OBJECT		QUESTION
ASPECT	VIEWPOINT	
Completeness	Consumer	5. Is the information considered in the schema sufficient for selection purposes?
		6. What information is missing from the schema?
Effectiveness	Consumer	7. Can the schema be used to select the best-suited techniques?
		8. Is there any case in which the schema cannot be used to decide which technique to use?
		9. How can effectiveness be improved?
Efficiency	Consumer	10. How long does selection take using the schema?
		11. How many resources are required to use the schema?
		12. How long does it take to decide whether or not to use a technique?
		13. How could efficiency be improved?
Usability	Producer & Consumer	14. How long does it take to learn how to use the schema?
		15. How many explanations are required during schema use?
		16. What sort of explanations are required?
		17. How often is help consulted during schema use and for how long?
		18. Are the names that appear in the schema appropriate?
User satisfaction	Producer & Consumer	19. How can usability be improved?
		20. What are the advantages and disadvantages of using the schema?
		21. Would people be prepared to use it?
		22. What improvements could be made?
		23. What do people like and dislike about the schema?
		24. What is a good environment for using the schema?
		25. Is there any superfluous or redundant information?

Table 10.1: Questions to be answered during the experiment.

Table 10.1 contains one aspect (user satisfaction) that will not be considered for the purpose of establishing a hypothesis that can be refuted by means of statistical evidence. This aspect will be assessed informally, examining the opinions of each subject, rather than by means of a rigorous statistical analysis.

The situation presented in Table 10.1 leads to an experimental design in which both qualitative and quantitative data will be collected, and where the analysis of these data will make it possible to correct any schema defects and propose a new improved version of the schema.

## 10.2 Experiment Planning

Testing the hypothesis involves checking whether there is a dependency between the selection process, the method used for the purposes of selection and the project for which the selection is made. However, *improve* is a very generic term that requires further explanation. For this

purpose, the generic hypothesis will be divided into five more detailed hypotheses, each of which refers to one of the aspects in Table 10.1 that are to be evaluated, and their associated alternative hypotheses:

For **completeness**, we have:

H<sub>01</sub>: The completeness of the original information for making the selection is independent of the method used for the purpose and the project under consideration.

H<sub>11</sub>: The characterisation schema provides a more complete original set of information for all projects.

H<sub>21</sub>: Books provide a more complete original set of information for all projects.

H<sub>31</sub>: It is the project under consideration that makes the original set of information more or less complete.

H<sub>41</sub>: Depending on the project under consideration, books or the schema will provide a more complete original set of information.

For **effectiveness**, we have:

H<sub>02</sub>: The effectiveness of the selection process is independent of the method used for the purpose and the project under consideration.

H<sub>12</sub>: The characterisation schema improves selection effectiveness for all projects.

H<sub>22</sub>: Books improve selection effectiveness for all projects.

H<sub>32</sub>: It is the project under consideration that improves or worsens selection effectiveness.

H<sub>42</sub>: Depending on the project under consideration, the best effectiveness will be achieved using books or the schema.

For **efficiency**, we have:

H<sub>03</sub>: The efficiency of the selection process is independent of the method used for the purpose and the project under consideration.

H<sub>13</sub>: The characterisation schema improves selection efficiency for all projects.

H<sub>23</sub>: Books improve selection efficiency for all projects.

H<sub>33</sub>: It is the project under consideration that improves or worsens selection efficiency.

H<sub>43</sub>: H43: Depending on the project under consideration, the best selection efficiency will be achieved by means of books or the schema.

For **usability**, we have:

H<sub>04</sub>: The usability of the method of selection is independent of the method used for the purpose and the project under consideration.

H<sub>14</sub>: The usability of the characterisation schema for selection is better than the usability of books for all projects.

H<sub>24</sub>: The usability of the books for selection is better than the usability of the characterisation schema for all projects.

H<sub>34</sub>: It is the project under consideration that improves or worsens the usability of the selection method.

H<sub>44</sub>: Depending on the project under consideration, usability will be better for books or the characterisation schema.

As indicated by the above hypotheses, the response variables of the experiment are: the completeness of the method of selection or *available information* during selection; the effectiveness of the method of selection or *fitness* of the techniques selected for the software projects; the efficiency of the method of selection or *time and resources* employed in selection; and the usability of the method of selection or *problems encountered* during selection. As the aim is to check the relationships (dependency of completeness, effectiveness, efficiency and usability in selection in respect of the method used and the project under consideration), the experiment will collect data on the completeness, effectiveness, efficiency and usability for each method. Then, the statistical significance of the differences observed will be examined, and if any sort of relationship is actually detected, the characteristics of this relationship will be examined in more detail. There are a variety of possible situations when studying the relationship between selection process, method and project:

- The method of selection is significant for the selection process. In this case, it will be necessary to study which method behaves best, try to give an explanation of the reasons why this is the case and, if it is not the schema, find out how the schema could be improved.
- The project under consideration is significant for the selection process. In this case, it will be necessary to study which projects behave best, and try to give an explanation of why this is the case. This option is of no use for improving the schema. The schema cannot be improved, because this is a process/project relationship, on which the method of selection has no influence.
- The combination of the project under consideration and the selection method is significant for the selection process. In this case, it will be necessary to study which method behaves best for which projects, try to give an explanation of why this is

the case and find out whether and how the schema can be improved for the projects for which it is less effective.

- Neither the method of selection, nor the project under consideration, nor the combination of the two is significant for the selection process. In this case, it will be necessary to give an explanation of the reasons why this is the case. This option is of no use for improving the schema. The schema cannot be improved, because the method of selection has no influence in this case.

For user satisfaction, the subjects participating in the experiment will assess how satisfied they are with the schema. After getting their assessment, a non-statistical study will be conducted to find out whether the values are acceptable and how the schema could be improved to make it easier for consumers to use.

So far, we have been talking about comparing two different situations or selection methods: using the schema and using books, both for different projects. However, this is not the only possible variation in the environment within which the schema is to be tested. Indeed, there are a host of variables whose variables can vary during testing technique selection. It is impossible to consider all the variations, as this would call for a lot resources that are not available for running the experiment. Table 10.2 shows some of the existing variables.

Table 10.2 shows some of the conditions that can vary within a project when comparing the different methods for selecting the testing techniques that are to be used during development. They include, for example, the type of problem in question (real-time, synchronous, control, data processing, etc.), the experience of the people who are to select the techniques (novices, experienced or very experienced), the documentation of the project to be used for selection purposes (requirements, design or code), the type of test for which the techniques are to be selected (unit, integration, system or acceptance or regression testing), the software attribute to be tested (correctness, validity, reliability, etc.), the task to be performed (testing technique selection, preparation of a test plan or running the tests), as well as the method used to select the techniques (books, characterisation schema or knowledge of the subject in question). As mentioned above, it is not possible to take into account all the sources of variation in the experiment, and, therefore, only a few will be addressed. The choice of the project variables to be taken into account in the experiment and their permitted values are discussed below.

### 10.2.1 Parameters

As mentioned above, it is impossible to account for all the possible variations of the variables that directly affect this experiment. Some variables will not be able to be used because it is not possible to get the value in question, whereas others will be eliminated to simplify the experiment and make it easier to run.

The parameters, (characteristics of the project that remain unchanged throughout the experiment) are discussed below. These are the characteristics that either do not or are not wanted to influence the result of the experiment [Juristo and Moreno, 2000]. In the case at hand, we have:

SOURCE	VARIATIONS
Selection method	Books Schema Routine practice
Attribute	Correctness Validity Reliability Security etc.
Task	Selection Test plan Test execution
Test	Units Integration System Acceptance Regression
Documentation	Requirements Design Code
Sw. system	Real-time Synchronous Control Data processing etc.
Subject	Inexperienced Experienced Very experienced

Table 10.2: Sources of variation when selecting testing techniques.

- *Subject experience.* Subjects with differing degrees of experience can be considered, providing for the possibility of studying whether different subjects respond equally to the schema. However, the only available population at present is composed of students, who belong to the *inexperienced* category of subjects. Accordingly, the behaviour of the schema will not be explored for the *experienced* and *very experienced* categories of subjects.
- *Task to be performed.* The subjects can be asked to perform different tasks, ranging from mere *selection* of the testing techniques to be used in the project, through the preparation of a *test plan*, to *running* the tests. For reasons of both time (the preparation of the test plan and test execution are more time-consuming and more complex tasks) and the characteristics of the subjects (inexperienced people), it was decided that the task to be performed would be confined to simply selecting testing techniques. This tests schema performance with regard to its primary function.



- *Documentation.* Another possible variation is the technical documentation that the subjects have and on which they base their selection. They could be given the project *requirements* document, the *design* document and the *code* and/or a combination of any of these. In this experiment, it was decided to supply the subjects with the *requirements* document, mainly because, as they are to select but not apply the techniques, they do not need to use any other project technical document apart from requirements. They will, of course, also need generic information about the project not contained in the technical document, but this is discussed in more detail in Section 10.3.2.
- *Test.* Of the different tests that can be run on the software, the experiment focuses on unit testing and does not stop to examine integration, system or acceptance testing. There are two reasons for this: the available time and the fact that unit testing is the goal of many of the testing techniques contained in the schema.
- *Attribute.* There are many software quality attributes that can be evaluated. It was considered that the best thing would be to ask the subjects to evaluate a familiar attribute, like *correctness*, leaving aside others like, for example, *reliability*, *usability*, etc. There are two reasons for this: we have a group of inexperienced people and many of the testing techniques contained in the schema refer to correctness.

## 10.2.2 Factors and their Alternatives

The above section discussed the variables, which, for one reason or another, remain constant throughout the experiment. The variables whose value changes, i.e. whose effect on the response variable is of interest, are discussed below. These are the factors [Juristo and Moreno, 2000]. The variables whose value is to change are: the method of selection and the software project (software system and project context).

### 10.2.2.1 Method of Selection

As mentioned above, one of the objectives of the experiment is to study the relationship between the method of selection used and the efficiency and effectiveness of the selection process. As shown in Table 10.2, the three possible values of the method of selection are: using the information that appears in *testing books*, using the characterisation *schema* or following *routine practice* by the subject. However, it is important to take into account that the subjects of this experiment are inexperienced, which means that it does not make sense to use routine practice. Therefore, the factor **method of selection** will have two possible alternatives: *books* and characterisation *schema*.

- *Selection using books.* The subjects will be given copies of three books, which are the only ones that they will be able to use for selection purposes. The books chosen for this purpose are easily accessible and their authors are widely reputed in SE

or in software testing. These books are *Software Engineering, 5th edition*, by Ian Sommerville (1998), *Software Engineering: Theory and Practice* by Shari L. Pfleeger (1999) and *Software Testing Techniques, 2nd edition*, by Boris Beizer (1990). The subjects have to decide which testing techniques they will use to test their allotted software project on the basis of the information they can find in the books.

- *Selection using the schema.* The subjects will be given the repository that appears in Appendix F, which they will be asked to use to decide which testing techniques they would select for their allotted software project. The recommended process for schema use is defined in Section 5.6.

Some differences between the levels of the method of selection factor are worthy of note:

1. There is a process associated with the schema use but not with the use of books.
2. The schema contains more instantiated techniques than appear in books.
3. Information related to testing techniques (for example, types of tests or static analysis techniques) appears in books.

However, these differences are an accurate reflection of how the schema and books work. The schema includes a method of use and books do not. The schema will usually contain more techniques than books, as there are techniques that only appear in research articles or which are a personalised variant of another technique that has occurred to someone at the company after years of experience and are, therefore, not published in books. Finally, testing books will contain other information not related to selection and which can interfere with selection. However, this is how testing books are.

In short, the possible bias in favour of the schema is due more to its definition and construction than to possible fortuities inadvertently (or deliberately) introduced into the experimental design.

#### 10.2.2.2 Software Project

The software project factor includes all characteristics of the project, including not only the software system to be developed (Table 10.2), but also the budget and time restrictions of the project, personnel characteristics, etc.

From the above, it follows that there is an infinity of values for this variable. This problem was settled in this experiment by selecting four software projects. This is what is known as *random factor*. Apart from the requirements document, for each of the software projects selected, the subjects are given a project context document, which explains the actual project situation. This will give the subjects the information they need about the software system to be tested, and the project of which it is part. The four software projects selected are:

- *Video (M)*. Management-type system. The aim of this project is to build a software system to manage a video club.
- *Loan arranger (B)*. Batch-type system. The aim of this project is to build a software system for bank loan approval or rejection and to calculate loan repayments.
- *Parking (S)*. Synchronous-type system. The aim of this project is to build a software system to control the availability of car parking slots.
- *WLMS (RT)*. Real-time system. The aim of this project is to build a software system to monitor the water level in a pump.

The information about the project contexts is given in Appendix G.1. Table 10.3 gives a summary of the characteristics of each project.

CHARAC./PROJECT	Parking	Loan	Video	WLMS
Size	small	small	small	medium
Sw type	synchronous	batch	management	real time
Sw architecture	concurrent	structured	OO	Real time
Development team	experienced	experienced	no experience	experienced
Testing team	experienced	no experience	experienced	no experience
Programming language	CC-Modula	Pascal	C++	Ada
Tools available	No	Dynamic analyzer	No	Dynamic analyzer
Possibility buy tools	No	–	Yes	No
Available testing time	enough	not enough	enough	enough
N. remaining defects	less as possible	does not matter	less as possible	no defects

Table 10.3: Characterisation of the software projects used for the experiment.

### 10.2.3 Response Variables

The response variables are the experiment outputs [Juristo and Moreno, 2000] and must reflect the relationships between the values of the factors, making it possible to draw conclusions about the proposed characterisation schema. As mentioned in Section 10.1, the aspects of the schema that are considered in this experiment are completeness, usability, effectiveness, efficiency and, generally, user satisfaction. The response variables to be considered, taken from Table 10.1, are shown in Table 10.4, grouped by attribute.

A series of forms, reproduced in Appendix G.2, were used to collect the response variables. The subjects used the forms to explain the process they used to make the selection. The following is an explanation of how the response variables were gathered using the above forms:

- *Background knowledge of the subject*. This information is gathered by means of Form E0, which contains open questions.
- *User satisfaction*. This information is gathered by means of Form E2 (questions 11 to 16), which contains open questions, and by means of Form E9.

ASPECT	RESPONSE VARIABLE
Completeness	How much information was used for selection? Sort of information used for selection purposes How much information was missing for selection? Sort of information that was missing for selection
Usability	How many problems were encountered during selection? Description of problems encountered How often schema help was consulted during selection Attributes consulted in help
Effectiveness	How many techniques were considered during selection? How many techniques were selected? Techniques selected?
Efficiency	Time spent studying techniques Selection time Time spent consulting doubts about schema attributes
User satisfaction	Benefits and drawbacks of using the schema Would you be prepared to use it? Improvements you would make What did you like and dislike? Has it changed the way you view the selection problem? What have you learnt? Would you do things differently next time? Suitability of the names and organisation of the schema
Background	Work experience Experience in software testing How do you usually select testing techniques? What type of information do you think you will need? What sort of problems do you think you are going to encounter? How long do you think it should take you to make the selection?

Table 10.4: Experiment response variables.

- *Completeness.* Subjects have to fill in Form E3, stating, for each of the techniques examined, the information used for selection purposes. Hence, it is possible to find both how much and what sort of information the subjects have used for selection. The information will be separated from the technique during collection.

Subjects have to fill in Form E4, stating, for each technique, the information they would like to have had about the technique and which they were unable to locate. Hence, it is possible to find out how much and what information the subjects missed (again, it is separated from the technique for simplicity's sake).

- *Usability.* The subjects will fill in Form E5, stating, for each technique, the problems they encountered when making the selection. Hence, it is possible to find out how many and what sort of problems the subjects encountered.

The subjects will fill in Form E8, stating, for each doubt they had about the meaning

of a schema attribute, the time they spent trying to understand the attribute. Hence, it is possible to find out how often the subjects consulted the schema help and what attributes they consulted.

- *Effectiveness.* Subjects will fill in Form E6, stating the techniques they have selected for the project in question. Hence, it is possible to find out how many and what sort of techniques the subjects selected.

Also, the number of techniques considered by subjects for selection can be obtained from Form E3 (information considered for selection).

- *Efficiency.* Subjects will fill in Form E7, stating the time spent studying each testing technique (by technique). Hence, it is possible to find out the total time it took the subjects to study the techniques.

The total time the subjects have spent consulting doubts about the schema can be obtained from Form E8 (consultation of doubts about the schema).

Subjects were asked to state the time it took them to make the selection (study plus selection) on Forms E1 and E2. The actual selection time can be obtained by subtracting the study time stated on Form E7 from the time it took the subjects to complete the whole process.

### 10.3 Experimental Design

As mentioned in Section 10.1, the primary objective of this experiment is to find out what influence the characterisation schema has on the testing techniques selection process in a software project. It also aims to examine the differences with respect to using another selection method other than the schema (in this case, testing books). For the purpose of better examining these relationships, it was decided to divide the population into four groups, each of which will make the selection twice, according to the pattern shown in Table 10.5.

	<b>Group 1</b> <b>(18 pers.)</b>	<b>Group 2</b> <b>(33 pers.)</b>	<b>Group 3</b> <b>(18 pers.)</b>	<b>Group 4</b> <b>(18 pers.)</b>
<b>Selection 1</b>	Schema	Books	Books	Schema
<b>Selection 2</b>	Schema	Schema	Books	Books

Table 10.5: Allocation of selection methods to groups.

As shown in Table 10.5, group 1 will make both selections using the characterisation schema, group 2 will make the first selection with the help of books and the second using the schema, group 3 will make both selections using books and, finally, group 4 will make the first selection using the schema and the second using books. This distribution has its *raison d'être*:

- Groups 2 and 4 are appropriate for studying the effect of the method of selection. Although the situation in which the schema will really be deployed in the group 2 situation, group 4 must be included to assure a balanced design (however, group 2 contains more subjects than any of the others as preference is given to real situations).
- Groups 1 and 3 are appropriate for assessing the learning effect (whether the second selection turns out better than the first and to what extent in both cases).

Additionally, we want to know what the opinion of the different subjects is with regard to the schema and how this opinion varies from one group to another in each case. Briefly, we want to find out whether the different types of subject (depending on the group to which they belong) are happy with the method/methods they have used for selection and which they would prefer, given the choice.

As regards size, each group should contain eight subjects to assure statistical significance (eight times four is thirty-two, and thirty is the minimum number of subjects for the population to be close to a normal distribution). On the other hand, we want the size of group 2 to be considerably larger than the others, as this would be the context in which the schema would be introduced into companies, where people already have experience in selection and try out a new method that is supposed to be better. Finally, eighty-seven subjects participated in the experiment. These eighty-seven subjects were randomly divided into four groups, allocating eighteen subjects to groups 1, 3 and 4 and thirty-three to group 2.

Also, as mentioned above, each subject has to make two selections. As there are four different projects and each group has either 18 or 33 members, it will be possible to distribute the projects randomly between the different selections (1 and 2) and different groups (1 to 4), establishing replications. Table 10.6 shows the subgroups (A to H) that have been established within each group (1 to 4) for project allocation.

	Selection 1				Selection 2			
REPLIC.	M	B	S	RT	M	B	S	RT
A	X	-	-	-	-	-	X	-
B	X	-	-	-	-	-	-	X
C	-	X	-	-	-	-	X	-
D	-	X	-	-	-	-	-	X
E	-	-	X	-	X	-	-	-
F	-	-	X	-	-	X	-	-
G	-	-	-	X	X	-	-	-
H	-	-	-	X	-	X	-	-

Table 10.6: Project allocation to subgroups.

As regards the size of the subgroups (replications), for groups 1, 3 and 4, they will be

composed of two subjects, except for two subgroups, which will have 3 members. For group 2, they will be composed of 4 subjects, except for one subgroup, which will have 5 members.

### 10.3.1 Two-Factor Design with Replication

The subjects were allocated to groups, and then to projects, completely at random. The subjects were first allocated to groups by drawing lots from two bags (one contained the subject numbers and the other, group numbers for the eighty-seven subjects). Subjects were then allocated to subgroups, also by drawing lots from two bags (one contained the subject numbers for a group and the other, subgroup letters for the number of individuals in the group).

Table 10.7 shows the result of the allocation of subjects to groups and subgroups.

	Group 1	Group 2	Group 3	Group 4
A	S7,S58,S69	S11,S18,S55,S86	S31,S33,S72	S59,S61
B	S3,S46	S1,S8,S73,S74	S32,S62	S37,S81
C	S2,S83	S43,S67,S70,S84	S64,S80	S9,S71,S79
D	S26,S35	S14,S30,S77,S82	S21,S66	SS22,S50
E	S34,S38	S12,S40,S47,S75	S23,S42,S65	S13,S85
F	S10,S52,S56	S5,S49,S51,S87	S36,S57	S28,S68
G	S4,S6	S19,S24,S25,S53,S60	S39,S76	S29,S41,S63
H	S20,S78	S16,S27,S45,S48	S17,S54	S15,S44

Table 10.7: Allocation of subjects to groups and subgroups.

### 10.3.2 Experimental procedure

The experiment was organised as five sessions, as shown in Table 10.8. Each session took place once a week, when the work to be completed by students during the week and handed in at the next session was set.

	Week 1 5-11 March	Week 2 12-18 March	Week 3 19-25 March	Week 4 26-1 April	Week 5 2-8 April
Group 1	Introductory session	Schema learning	Selection with schema	Selection with schema	Reflection conclusions
Group 2		Selection with books	Schema learning		
Group 3			Selection with books	Schema learning	
Group 4		Schema learning	Selection with schema	Selection with books	

Table 10.8: Experimental procedure.

Table 10.8 has two peculiarities:

1. The schema learning session is not common for all the subjects. It was not held until the students had to use the schema. This was done to prevent subjects from groups 2 and 3 unconsciously behaving like the members of group 4 and taking advantage of what is in theory the characterisation schema to make the selection.
2. The members of group 3 learn of the existence of the schema and how it is used, although they never actually use it. This was done, on the one hand, because we wanted all the subjects to be acquainted (at least in theory) with the characterisation schema by the end of the experiment. On the other hand, returning to the generic hypothesis formulated in Section 10.1, upon which this research is based, it is supposed that the schema should be beneficial to developers, as it will help them to make better suited decisions in less time. In this case, we wanted to find out what opinion the schema would merit before it was actually used.

As reflected in Table 10.8, the experiment will be carried out as follows:

- **Session 1.** All the subjects who are to participate in the experiment are assembled, and they are given an introductory session about the work that they have to do and what is expected of them, although they are not given information on the details of the experiment, like objectives, etc. Everyone is given a form on which they are asked about their background in software testing, and techniques selection in particular, as well as their opinion on how easy or difficult these tasks are likely to be and the problems they are likely to encounter during software development. This is the form E0 reproduced in Appendix G.2.
- **Session 2.** The subjects hand in the form that they were given the week before, and two parallel sessions are held: one with the subjects belonging to groups 1 and 4, and another with the subjects belonging to groups 2 and 3. The first session is an introduction to the characterisation schema and the subjects are taught how to use the schema. In the second session, the subjects are given the material required to make the selection using books (requirements documents, project contexts, photocopies of the respective books and forms E1, E3, E4, E5, E6 and E7 reproduced in Appendix G.2). They are then told what they have to do and they are given time to examine the material and ask any questions they may have. The tasks of each group are: for groups 1 and 4, examine the schema and clear up any doubts they may have, and for groups 2 and 3, make the selection with books.
- **Session 3.** Groups 2 and 3 hand in the result of the selection with books and two parallel sessions are held: one with group 2 and another with groups 1, 3 and 4. Again, the first session is an introduction to the characterisation schema and its use. In the second, the subjects are given the material required to make the selection using books in the case of group 3 (requirements documents, project contexts, photocopies of the respective books and forms E1, E3, E4, E5, E6 and E7



reproduced in Appendix G.2) and using the schema in the case of groups 1 and 4 (requirements documents, project contexts, repository and forms E2, E3, E4, E5, E6, E7 and E8 reproduced in Appendix G.2). They are then told what they have to do and are given time to examine the material and ask any questions they may have. The tasks of each group are: for group 2, to study the schema and clear up any doubts they may have, and for groups 1, 3 and 4, make the selection.

- **Session 4.** Groups 1, 3 and 4 hand in the result of the selection and again two parallel sessions are held: one with group 3, and another with groups 1, 2 and 4. With respect to the first session, again it is an introduction to the characterisation schema and its use. However, unlike the other learning sessions, the subjects of group 3 are not going to put into practice these concepts, and they are simply given a form at the end of the session (this is form E9 reproduced in Appendix G.2), on which they are asked to give their impressions on whether the schema would solve the problems they encountered when making the earlier selections and, if so, how. As regards the second session, as always, the students are given the material required to make the selection, using books in the case of group 4 (requirements documents, project contexts, photocopies of the respective books and forms E1, E3, E4, E5, E6 and E7 reproduced in Appendix G.2) and using the schema in the case of groups 1 and 2 (requirements documents, project contexts, repository and forms E2, E3, E4, E5, E6, E7 and E8 reproduced in Appendix G.2). They are then told what they have to do and are given time to examine the material and ask any questions they may have. The tasks of each group are: for group 3, fill in the form about the schema and, for groups 1, 2 and 4, make the selection.
- **Session 5.** In this session, all the subjects who participated in the experiment are assembled and the objectives of the experiment are explained. Opinions are interchanged.

### 10.3.3 Threats to validity

When designing an experiment, there are always two types of threats to its validity: internal threats, which are represented by possible design errors, and external threats, which are related to the validity or representativeness of the findings of the experiment.

#### 10.3.3.1 Internal Threats

The following were identified:

- *Copying.* There is the possibility of students copying each other, as they do the work at home. An attempt was made to solve this problem by telling the students that they would be graded on the basis not of the techniques selected but of the effort made to do the exercise. Moreover, it should be stressed that the subjects involved

in the experiment were all volunteers, which suggests that they were interested from the very start in the sort of work they were doing.

- *Capability.* It is true that not all the subjects will have the same problem-solving ability. Randomisation should minimise this problem.
- *Method learning.* If subjects apply the same method of selection twice, they will learn from the mistakes they made the first time and will do a better job the second time round (irrespective of how good the method of selection is). This threat is explicitly taken into account in groups 1 and 3. However, we have groups 2 and 4, in which the subjects apply a different method of selection each time, to counteract its effect.
- *Object learning.* If subjects use the same project for the two selections they are to make, they will also learn from the mistakes they made the first time and will do a better job the second time round (irrespective of how good the method of selection is). This has been remedied by having each subject make the two selections for different projects.
- *Boredom.* Subjects may find the experiment boring and, therefore, their performance may be below normal. It is assumed that the grading of the exercise will motivate the students. Also, the subjects who have performed the experiment are volunteers, which means that they should have at least some interest in the subject.
- *Enthusiasm.* On other occasions, subjects are excited about the prospect of trying out a new method or technique, which means that they work harder on this technique. In this case, the subjects are inexperienced, which suggests that they will work equally as hard on selection with or without the schema.
- *Unconscious formalisation.* As one method is more formal than the other (schema as opposed to *ad hoc* process using books), the group that uses the schema first and the books afterwards may make use of the more formal model for the benefit of books. This could mean that books come off better than they really should. In case this happens (group 4), the other three groups will control the possible effect of this group.
- *Procedure.* Something that can, and actually did, occur is that the subjects do not follow the process they were told to for selection using the schema. This can lead to deviations in the results obtained using the schema, whether for better or worse.

### 10.3.3.2 External Threats

The following were identified:

- *Language.* As mentioned above, the schema is instantiated in the subjects native tongue, whereas the books are in English (a language with which many subjects are not well enough acquainted). Although it was not foreseen that the subjects would encounter this difficulty (it was thought that they would have no problem with reading technical texts in English), they did. As the subjects have to read technical texts in English in this country, this problem will mean that the results obtained here cannot be generalised to all countries (at least to English-speaking countries), as it has benefited the schema.
- *Experience.* The subjects are not experienced, which can mean that it was harder for them to understand testing books (there is the possibility that their vocabulary on the subject is not good enough). This may have been more advantageous for the schema and will mean that the results cannot be generalised to all subject types.
- *Projects.* Four projects were used and an attempt was made for them to be representative of reality. However, experiments with more projects should be run, as not all the possible situations of a software project have been accounted for.
- *Techniques used.* A set of techniques that was as varied and complete as possible was chosen to instantiate the schema. This means that at least two members of all the families of testing techniques are represented. Also some techniques that are representative of specific software (like techniques for object-oriented or real-time software) were included. However, only unit testing techniques were accounted for. This issue would have to be addressed in more detail by running experiments with integration, system and regression testing techniques. Also, more techniques for specific software should be included.

## 10.4 Data Analysis

The statistical program SPSS (version 10) is used to analyse the data collected during the experiment, alongside two different statistical methods: analysis of variance and analysis of simple correspondences. Before briefly describing the objective of each method, it is important to recall that two types of variables have been gathered during the experiment: quantitative values (for example, the number of problems encountered, the time taken to make the selection or how information was used for selection purposes) and qualitative values (for example, the sort of problems encountered during selection or the sort of information used during selection).

Analysis of variance (ANOVA) is used to study the relationships between a quantitative response variable and one or more qualitative factors. Its objective is to determine whether the differences between the means of the response variable in the groups established by the combinations of factor levels are statistically significant. The steps to be followed to interpret the result of the ANOVA are:

1. Observe the significance level of the statistical model used to check whether it is possible to reject the null hypothesis. The confidence level chosen for rejection is 95%, which means that the null hypothesis will be able to rejected provided that the significance level of the statistical model is under 0.05.
2. If the null hypothesis has been rejected, the significance level of the factors and their combinations will be observed to find out which affect the response variable. The factors (or combination of factors) whose significance level is under 0.05 will be the ones that affect the response variable.
3. The significant factors or combinations of factors will be examined to find out how each of their levels affect the response variable.
4. Model validation. For the results of the ANOVA to be valid, two conditions have to be met: the sample must have a normal distribution and be homocedastic (of equal variance). As the tests of all the models were satisfactory, they have been moved to Appendix G.3.

As mentioned above, ANOVA can only be used for quantitative response variables, and qualitative variables have also been collected in this experiment. For the qualitative variables, data analysis is non-parametric. However, non-parametric statistical tests are less powerful than parametric tests and more complex to interpret. Therefore, with the aim of simplifying the analysis of the experiment, it was decided to transform the qualitative variables into quantitative variables for the purposes of analysis. For the qualitative variables, the frequency of appearance of each type of response (as a percentage) and a three-factor ANOVA will be used: the two factors considered in the design, plus the response type. For the genuinely quantitative experiment variables, a two-factor ANOVA will be used: the factors of the experiment. Thus, for example, a two-factor ANOVA (method and project) will be carried out on the response variable to examine the number of problems encountered, whereas a three-factor ANOVA will be carried out (method, project and sort of problem) on the frequency of each problem encountered to study the sort of problems encountered. For the three-factor analyses, the interaction of order 3 has been considered negligible as per professional prescription.

Additionally, for the three-factor ANOVA, the artificial factor introduced (which represents the sort of information whose frequency is examined) has from twenty-five to forty-seven levels. When the result of the interaction of the method and sort of information, or of the project and sort of information, is significant, we will face the problem of analysing these varied and, therefore, complex interactions. For simplicitys sake, the method of analysis by simple correspondences will be used to study any interactions. This method can be used with two related qualitative variables with many levels. The objective of this method is to interpret the similarities between the levels of one variable in respect of those of the other, as well as the relationships between the different levels of the variables.

### 10.4.1 Characteristics of the Subjects

Before carrying out the experiment, a questionnaire was given to the subjects to gather information *a priori* on their characteristics and opinions about the selection problem. The data gathered from subjects related to the *experience* aspect are as shown in Table 10.4:

1. *Work experience.* The subjects used for the experiment are 87 final-year students at the School of Computer Science, Technical University of Madrid. Of the subjects, 50% have no work experience and the other 50% are experienced in the following: 50% have worked as developers, 21% have worked as analysts, 24% have been members of a testing team and 5% have worked as project managers. Of the experienced subjects, 45% have work experience of less than six months, 30% from six months to a year and 25% of over a year.
2. *Experience with software testing.* Of the subjects, 75% are acquainted with software testing only through what they have studied at university, whereas the other 25% have run tests as part of their work. As regards experience in testing, 70% routinely run tests as part of the practical exercises set for their degree course, 6% have only done run tests in small exercises and 24% have been involved in testing as part of real development projects.
3. *Selection heuristic.* None of the students have a heuristic for selecting testing techniques, either because they have never considered the problem or they do not consider selection to be necessary (which means that they always apply the same techniques).
4. *Information to be used in selection.* They think that the information relevant for selecting techniques is the technique inputs and outputs, as well as the type of software to be tested. A fewer number think that the test execution time, software architecture, deployment and development environment, development method used and software size are important. Finally, at the tail end of importance, they mention complexity, cost of application, tools, technique comprehensibility and sources of information and experiences in technique use.
5. *Problems they are likely to encounter.* The problems they think are important for selection include familiarity with the techniques, being able to find the best-suited technique, not being able to access information about the technique, shortage of time for making the selection and economic costs of a poor selection.
6. *Time it would take to make the selection.* Of the subjects, 39% don't know. Of the other 41%, 65% think not very long, 25% think a week or more and 10% say more than a month. Almost all state that it depends on the experience of the subject, the type of software and the size of the software.

### 10.4.2 Schema Efficiency

As indicated in Table 10.4, three response variables have been taken into account to study the efficiency of the characterisation schema:

- The time spent studying the techniques.
- The selection time.
- The time students who made the selection using the schema spent consulting doubts about schema attributes.

Three statistical analyses were made, one for each response variable.

#### 10.4.2.1 Testing Technique Study Time

Table 10.9 reflects the values in minutes for the mean and standard deviations of the total study times and with respect to each alternative under consideration. It is found that the mean study time is lower for the schema than for using books and that the four projects behave similarly.

Factor	Alternative	Mean	Std. Devi.	N
Method	Books	279.40	167.82	85
	Schema	84.07	63.90	87
Project	Video	162.64	156.40	45
	WLMS	187.59	155.10	41
	Parking	200.96	170.93	45
	Loan	170.98	157.65	41
	<b>Total</b>	180.60	159.63	172

Table 10.9: Mean and standard deviation for study time.

Table 10.10 shows the result of the ANOVA for study time. On the one hand, it is found that the null hypothesis of equality of study times can be rejected, because the significance level (column labelled Sig.) for the model is lower than 0.05. On the other hand, it is found that the factor that affects the response variable is method (significance level of under 0.05). Neither the project, nor the combination of method and project affect study time for a testing technique.

This is the expected result, as it says that study time is shorter using one of the two methods (books or schema). On the other hand, it indicates that the study time for testing techniques is not influenced either by the software project.

Table 10.11 and Figure 10.1 show the mean estimated value in minutes for study time using books and the schema. It is found that the schema is the method with the lower associated techniques study time, where the difference between the two is some three times.

Source	Type III square sum	gl.	Mean square	F	Sig.
Statistical model	1703652,888	7	243378,984	15,042	0,000
Intersection	5649762,731	1	5649762,731	349,180	0,000
PROJECT	31145,335	3	10381,778	0,642	0,589
METHOD	1612850,681	1	1612850,681	99,681	0,000
PROJECT * METHOD	31615,588	3	10538,529	0,651	0,583
Error	2653536,432	164	16180,100		
Total	9967131,000	172			
Corrected total	4357189,320	171			

Table 10.10: ANOVA for study time.

Method	Mean	Std. error	Confidence interval at 95%	
			Lower bound	Upper bound
Books	278,451	13,822	251,158	305,744
Schema	84,518	13,647	57,571	111,465

Table 10.11: Estimated values for mean study time.

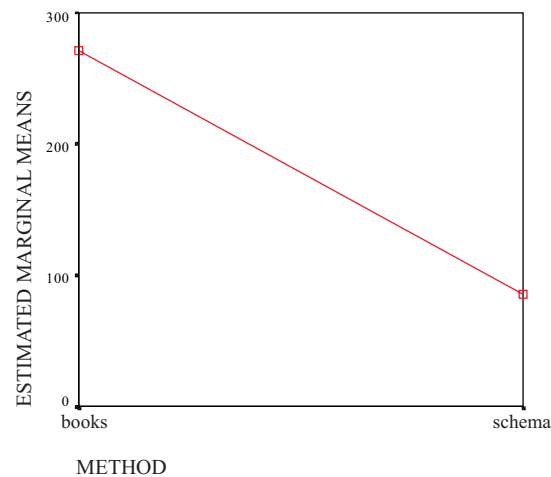


Figure 10.1: Estimated values for mean study time.

The experiment appears to indicate that **the characterisation schema reduces the time it takes to study the techniques for selection purposes**. However, it is important to remember that the testing books are in English and the characterisation schema is in the native tongue of the subjects. One might think that this result is not valid, as language also influences study time. However, it should be stressed that developers in Spain have to study books in English and that a Spanish repository would improve technique study. This leads to the conclusion that the result, although it can be taken to be valid, could not be extrapolated

to English-speaking countries.

Another point that should be mentioned is the fact that the technique study time may decrease as experience (and books/schema use) increases. Again this could threaten the validity of these results, preventing their extrapolation to other subject types.

#### 10.4.2.2 Testing Technique Selection Time

Table 10.12 reflects the values in minutes for the mean and standard deviation of the total selection times and with respect to each alternative under consideration. It is found that the mean is lower for schema use than for use of books and that the four projects behave similarly.

Factor	Alternative	Mean	Std. Devi.	N
Method	Books	247,82	191,74	85
	Schema	145,91	84,29	87
Project	Video	199,49	175,87	45
	WLMS	169,93	140,69	41
	Parking	213,67	168,94	45
	Loan	200,00	131,73	41
	Total	196,27	155,70	172

Table 10.12: Mean and standard deviation for selection time.

Table 10.13 shows the result of the ANOVA for selection time. It is found that the null hypothesis of equality of selection times is rejected, as the significance level of the model is under 0.05. The factor that affects selection time is the method, as its significance level is under 0.05. As with learning time, neither the project nor the combination of method and project influence selection time.

Source	Type III square sum	gl.	Mean square	F	Sig.
Statistical model	518992,289	7	74141,756	3,353	0,002
Intersection	6603830,921	1	6603830,921	298,630	0,000
PROJECT	40088,501	3	13362,834	0,604	0,613
METHOD	431192,516	1	431192,516	19,499	0,000
PROJECT * METHOD	32875,736	3	10958,579	0,496	0,686
Error	3626659,868	164	22113,780		
Total	10771641,000	172			
Corrected total	4145652,157	171			

Table 10.13: ANOVA for selection time.

Table 10.14 and Figure 10.2 show the mean value in minutes for selection time using books and the schema. Again, the schema is the method that, as its mean is lower, provides



a lower selection time, where the difference is less than twice in this case.

Method	Mean	Std. error	Confidence interval at 95%	
			Lower bound	Upper bound
<b>Books</b>	246,348	16,159	214,441	278,255
<b>Schema</b>	146,073	15,954	114,571	177,576

Table 10.14: Estimated values for selection time.

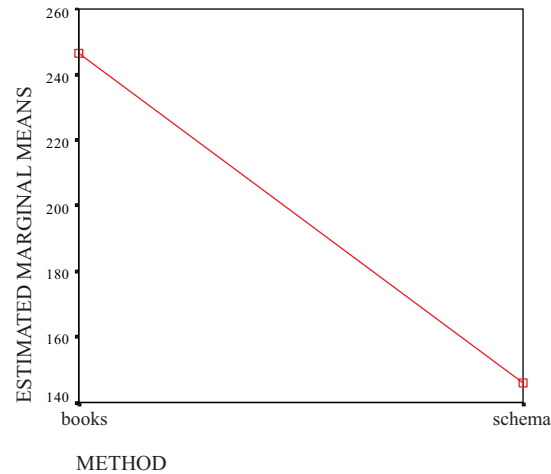


Figure 10.2: Estimated values for mean selection time.

This result is apparently surprising, as in principle, selection time should not depend on the method used. One might think that once the information about the technique has been gathered, the time it takes to make the decision should not depend on the method used to gather this information. However, a reason why this happens can be ventured, and this is that it is the sort of information gathered using the schema that speeds up decision making. The schema apparently provides more useful information for selection purposes than the details found in books. This improves decision-making and, therefore, decisions can be made quicker than if the information gathered is less useful, and has to be examined in more detail.

So, the data appear to indicate that **the characterisation schema reduces the testing technique selection time**. However, schema effectiveness has not yet been examined, and the validity of these results is subject to the results for schema effectiveness. An unfavourable result for effectiveness would nullify the results for efficiency.

#### 10.4.2.3 Time Consulting Doubts about the Schema

Table 10.15 reflects the values in minutes for the mean and standard deviation of the total time spent consulting doubts about the schema and with respect to each alternative under consideration. Note that, in this case, the only factor under consideration is the project,

as the *method* factor has a single value, which is *schema*. Table 10.15 shows that all four projects behave similarly.

Factor	Alternative	Mean	Std. Dev.	N
Sistem	Video	4.50	8.72	22
	WLMS	6.57	13.53	21
	Parking	4.23	4.16	22
	Loan	3.33	3.26	21
	<b>Total</b>	4.65	8.37	86

Table 10.15: Mean and standard deviation of the time spent consulting doubts about the schema.

Table 10.16 shows the result of the ANOVA for the time spent consulting doubts about the schema. It is found that the null hypothesis of equality of times spent settling doubts by projects cannot be rejected, as the significance level of the statistical model is over 0.05. This means that the time spent consulting doubts about the schema is independent of the project under consideration.

Source	Type III square sum	gl.	Mean square	F	Sig.
Statistical model	118.362	3	39.454	0.554	0.647
Intersection	1864.937	1	1864.937	26,207	0.000
PROJECT	118.362	3	39.454	0.554	0.647
Error	5835.173	82	71.161		
Total	7814.000	86			
Corrected total	5953.535	85			

Table 10.16: ANOVA for the time spent consulting doubts about the schema.

This result was to be expected, as the time spent settling doubts does not necessarily depend on the project in question. Another expected result is the value of the mean time spent settling doubts. The mean value, shown in Table 10.15, is 4.65 minutes. If this is compared with the time it takes to study the techniques (84.07 minutes for choice using the schema and 279.40 minutes for selection with books) and the selection time (145.91 minutes for selection using the schema and 247.82 minutes for selection with books), the data appear to indicate that **the time spent by the subjects on settling the doubts they have about the schema is negligible as compared with study and selection times**. Also, this time can be considered to decrease the more the subjects use the schema.

#### 10.4.2.4 Conclusions on Schema Efficiency

The total time required to solve the selection problem is the sum of the study time, plus the selection time and consultation time (which is zero if books were used for selection). This

experiment found that the schema helps to reduce both the study and the selection time as compared with books and that the time spent consulting the schema can be considered negligible with respect to the other two. Accordingly, it can be concluded that one of the objectives of this research is achieved, and this is the construction of a **characterisation schema that makes selection more efficient**. However, the results are subject to the following conditions: non English-speaking and inexperienced subjects.

### 10.4.3 Schema Usability

The response variables that appear in Table 10.4 were taken into account for the characterisation schema usability study. These are:

- The number of problems encountered by each subject when making the selection.
- The sort of problems encountered by each subject when making the selection.
- How often a subject making the selection using the schema had to consult the schema help.
- Which attributes were consulted in the help.

Four statistical analyses are presented below, one for each response variable.

#### 10.4.3.1 Number of Problems Encountered

Table 10.17 shows the values for the mean and standard deviation of the number of problems that each subject encountered depending on each alternative considered. Again, it is found that the mean is lower for the schema than for books and that the four projects behave similarly.

Factor	Alternative	Mean	Std. Devi.	N
Method	Books	2,67	1,30	86
	Schema	1,51	1,19	86
Project	Video	2,24	1,58	45
	WLMS	1,85	1,26	41
	Parking	2,13	1,50	45
	Loan	2,12	1,05	41
	Total	2,09	1,37	172

Table 10.17: Mean and standard deviation for number of problems encountered.

Table 10.18 shows the result of the ANOVA of the number of problems encountered. It is found that the null hypothesis is rejected (the significance level of the model is under 0.05) and that the method is the only factor that affects the response variable, as its significance

Source	Type III square sum	gl.	Mean square	F	Sig.
Statistical model	64,149	7	9,164	5,863	0,000
Intersection	747,679	1	747,679	478,305	0,000
METHOD	57,556	1	57,556	36,820	0,000
PROJECT	3,958	3	1,319	0,844	0,472
METHOD * PROJECT	2,048	3	0,683	0,437	0,727
Error	256,362	164	1,563		
Total	1074,000	172			
Corrected total	320,512	171			

Table 10.18: ANOVA for number of problems encountered.

is under 0.05. Neither the project, nor the combination of method and project affect the response variable.

This, as was to be expected, indicates that the number of problems encountered when making the selection varies depending on the method used. And it varies as shown in Table 10.19 and Figure 10.3, that is, the subjects encounter more problems when making the selection using books than using the schema, where the difference between the estimated values for the mean in each case is just under twice.

Method	Mean	Std. error	Confidence interval at 95%	
			Lower bound	Upper bound
<b>Books</b>	2,667	0,135	2,400	2,934
<b>Schema</b>	1,509	0,135	1,242	1,775

Table 10.19: Estimated values for number of problems encountered.

Accordingly, the data appear to indicate **that subjects encounter fewer problems when making the selection using the characterisation schema presented here.** However, not only the number but also the sort of problems has to be taken into account. This will be analysed below.

#### 10.4.3.2 Sort of Problems Encountered

The sort of problems encountered during selection are as reflected in Table 10.20. It is found that the subjects have encountered twenty-five different problems. As this response variable is qualitative, it will be quantified as explained earlier, by replacing the response variable with the frequency of appearance of each problem encountered and adding the problem as a factor.

Table 10.21 shows the results of the ANOVA for the percentage of subjects who encountered each problem. From this table, it can be deduced that the null hypothesis of equality of the frequency of problems encountered by each subject is rejected (the significance

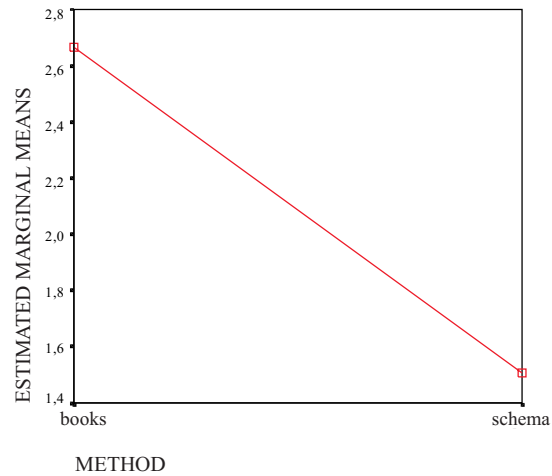


Figure 10.3: Estimated values for number of problems encountered.

N	Problem description
1	Use of different values to qualify an attribute in different techniques
2	Need for information on software architecture
3	Schema information is too brief, too general or too ambiguous
4	Difficulties with the experiment itself (forms)
5	Lack of information about technique effectiveness
6	Understandability of books or schema
7	A summary with the characteristics for selecting the technique is missing
8	Difficulties with documentation (the photocopies of books were not legible)
9	Information is insignificant for selection or the value is always the same
10	Lack of information on access or use of tools
11	Missing or insufficient historical information
12	Language difficulties
13	Problems with the ERS or problem domain
14	Inconsistencies (contradictions) between schema values
15	Inexperience of the subject making the selection
16	Technique characterisation information is missing
17	Information on project context is missing
18	Too many attributes in the schema
19	Information on precision is missing from the schema
20	Information on technique procedure is missing
21	Information on the purpose of the technique is missing
22	Information on technique application time is missing
23	Not enough time to complete the exercise
24	Information on technique completeness is missing
25	Information on technique knowledge is missing

Table 10.20: Description of the problems encountered during selection.

level of the model is under 0.05). Also it is found that the method of selection used and the problem influence the number of people who detect the problem, as does the interaction between the method and problem, as the significance is under 0.05 in all three cases.

This goes to say that the number of subjects who come across a given problem depends

Source	Type III square sum	gl.	Mean square	F	Sig.
Statistical model	31307,177	127	246,513	11,345	0,000
Intersection	11392,951	1	11392,951	524,330	0,000
METHOD	269,306	1	269,306	12,394	0,001
PROJECT	86,809	3	28,936	1,332	0,271
PROBLEM	15374,319	24	640,597	29,482	0,000
METHOD * PROJECT	19,365	3	6,455	0,297	0,827
METHOD * PROBLEM	14418,419	24	600,767	27,649	0,000
PROJECT * PROBLEM	1138,960	72	15,819	0,728	0,910
Error	1564,457	72	21,729		
Total	44264,586	200			
Corrected total	32871,635	199			

Table 10.21: ANOVA for frequency of problems encountered.

on both the problem in question and the method used to make the selection. In other words, whether or not a subject comes across a problem depends on the method used. That is, there are problems that occur more frequently when using one method than when using the other or even problems that only occur with one method. This is reflected in Figure 10.4.

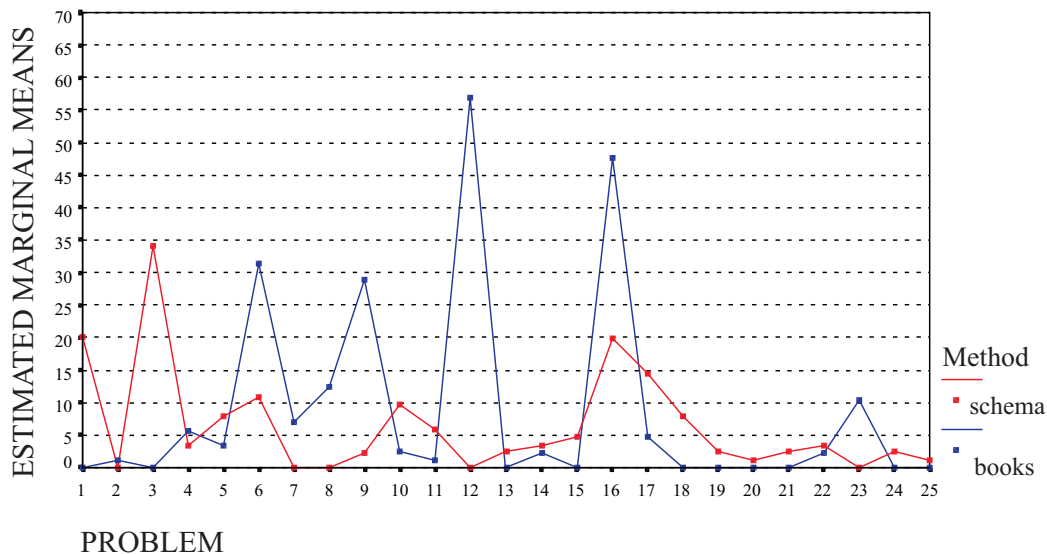


Figure 10.4: Estimated values for the frequency of problems encountered.

Table 10.22 shows the results of the analysis of simple correspondences for the method and problem variables. It is found that the value of chi-squared is very high, and the significance level is under 0.05, which means that both variables are closely related and the results of the analysis of simple correspondences are valid. Accordingly, the levels of both variables can be represented as points in a one-dimensional space (a straight line), which indicates the

distance between these levels.

Dimension	Gen value	Inertia	Chi-square	Sig.	Proportion of inertia	
					Explained	Accumulated
1	0,751	0,564			1,000	1,000
Total		0,564	181,476	0,000	1,000	1,000

Table 10.22: Analysis of correspondence between the method and problem variables.

Figure 10.5 shows the values obtained using the analysis of simple correspondences for each level of the variables of interest. The points in red represent the value of the problems and the lines in blue represent the value of the methods. Mapping the points and the two straight lines to the x-axis, we have the distance between the problems and the methods of selection.

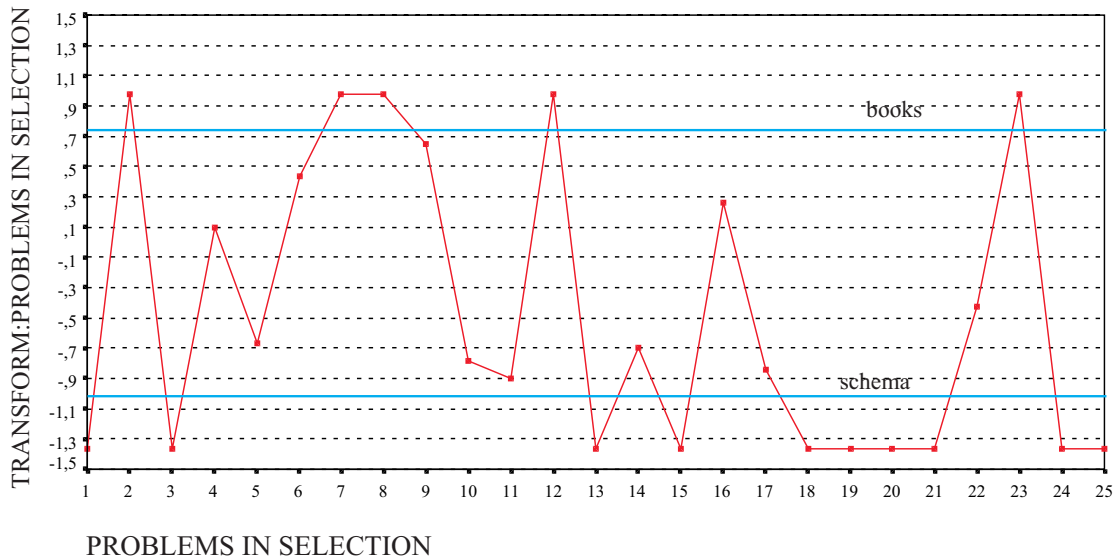


Figure 10.5: Column and row points for the method and problem variables.

Figure 10.5 shows that the points under or very close to the schema line refer to problems that will occur only when the characterisation schema is being used. The points above or very close to the books line refer to problems that will occur only when books are being used. The intermediate points that are between and not close to either of the two lines refer to problems that will occur indistinctly.

Looking at the sort of problems associated with each method of selection, we find that:

1. Problems 1, 3, 13, 15, 18, 19, 20, 21, 24 and 25 only arise when working with the schema. Looking at Table 10.20, we find that:

- Problems 1, 3, 18 and 20 are related to the schema (things to be improved

in the schema).

- Problems 13 and 15 are not related to the schema (in principle, they are not necessarily associated with the schema).
  - Problems 19, 21, 24 and 25 refer to information that is missing from the schema. Interestingly, this information does not appear in the books either, and the subjects did not notice that the information was missing until they realised that it could be of interest.
2. Problems 2, 7, 8, 12 and 23 arise only when working with books. They refer to things like the need for a summary of the technique characteristics that are useful for selection purposes, difficulties with the documentation, there not being enough time to complete the exercise, or problems with the language.
  3. The other problems arise in both situations, although there are problems, like 5, 10, 11, 14 or 17, that are more likely to occur with the schema or 6 and 9 that are more likely with books.

Problems 5, 10, 11, 16, 17 and 22 refer to missing information. Again, the information to which they refer (except for problem 22) is information that appears but is not instantiated in the schema.

The other problems are varied. Thus, for example, problem 4 refers to problems with the experiment; problem 6 refers to method comprehensibility (books or schema); problem 9 refers to the existence of unnecessary information for selection purposes and 14 refers to inconsistencies between values.

In sum, we have that:

- The frequency of appearance of each problem is lower with the schema than with books.
- The problems related to the schema refer to uninstantiated attributes, incomprehensible values for the information, the schema containing too much information (the information is sometimes considered to be superfluous because all the values are the same for all the techniques) and missing information in the schema (this will be dealt with later, under schema completeness).
- The problems related to books refer to missing information, problems of understanding (unstructured information), not enough time to complete the exercise, the existence of unnecessary information (for example, technique procedure), inconsistencies between the information provided, and problems with the language.



From the data analysis, it can be concluded that, **although schema usability is not poor (only in one case is the frequency of a problem with the schema over 15%), there is room for improvement.** Indeed, where the frequency of appearance of a problem is greater using the schema than with books, the problem always refers to missing information not instantiated in the schema (the missing information is more evident than in the case of books). This again harks back to the language problem. Figure 10.4 shows that its frequency is very high, which stresses the fact that the results on testing technique study time (efficiency) are confined to non-English-speaking countries.

#### 10.4.3.3 Number of Doubts about the Schema

Table 10.23 shows the values for the mean and standard deviation of the total number of attributes for which each subject has consulted the schema help and with respect to each alternative under consideration (project, as again the *method* factor is a single value, which is *schema*). Again, we find that all four projects behave similarly.

Factor	Alternative	Mean	Std. Dev.	N
Project	Video	2.41	2.82	22
	WLMS	3.95	7.45	21
	Parking	3.18	3.20	22
	Loan	2.57	2.34	21
	Total	3.02	4.38	86

Table 10.23: Mean and standard deviation for the number of doubts about the schema.

Table 10.24 shows the result of the ANOVA for the number of attributes that have been consulted in the schema help. It is found that the null hypothesis of equality of number of consultations for each project cannot be rejected, as the significance level of the model is over 0.05.

Source	Type III square sum	gl.	Mean square	F	Sig.
Statistical model	31.267	3	10.422	0.534	0.660
Intersection	788.443	1	788.443	40.390	0.000
PROJECT	31.267	3	10.422	0.534	0.660
Error	1600.686	82	19.521		
Total	2418.000	86			
Corrected total	1631.953	85			

Table 10.24: ANOVA for the number of times schema help has been consulted.

As was to be expected, this indicates that the number of attributes consulted in the schema help does not depend on the project in question. The mean number of attributes

consulted is 3.02 per person. This number, compared with the 33 attributes in the schema cannot be considered high. Therefore, the data appear to show that **the number of doubts each person has about the schema is not a cause for concern.**

#### 10.4.3.4 Sort of Attributes of the Schema Consulted

Table 10.25 reflects the characterisation schema attributes. As this response variable is qualitative, it will be quantified as explained above, that is, by replacing the response variable with the frequency of appearance of each attribute consulted in the help and adding the attribute as a factor.

N	Attribute
1	Software architecture
2	Aspect
3	Automation
4	Benefits
5	Number of cases generated
6	Cost of application
7	Completeness
8	Knowledge
9	Tool cost
10	Software type
11	Test data cost
12	Adequacy criteria
13	Dependencies
14	Effectiveness
15	Element
16	Cost of application
17	Tool environment
18	Inputs
19	Sources of information
20	Tools used
21	Identifier
22	Programming language
23	Development method
24	Opinion
25	Personnel
26	Precision
27	Problems
28	Purpose
29	Projects of reference
30	Repeatability
31	Size
32	Support

Table 10.25: Schema attributes.

Table 10.26 shows the results of the ANOVA for the percentage of subjects who have consulted the meaning of each schema attribute. From this table, it can be deduced that the null hypothesis of equality of frequency of the attributes consulted by each subject is rejected (as the significance level of the model is under 0.05). It is also found that the project in

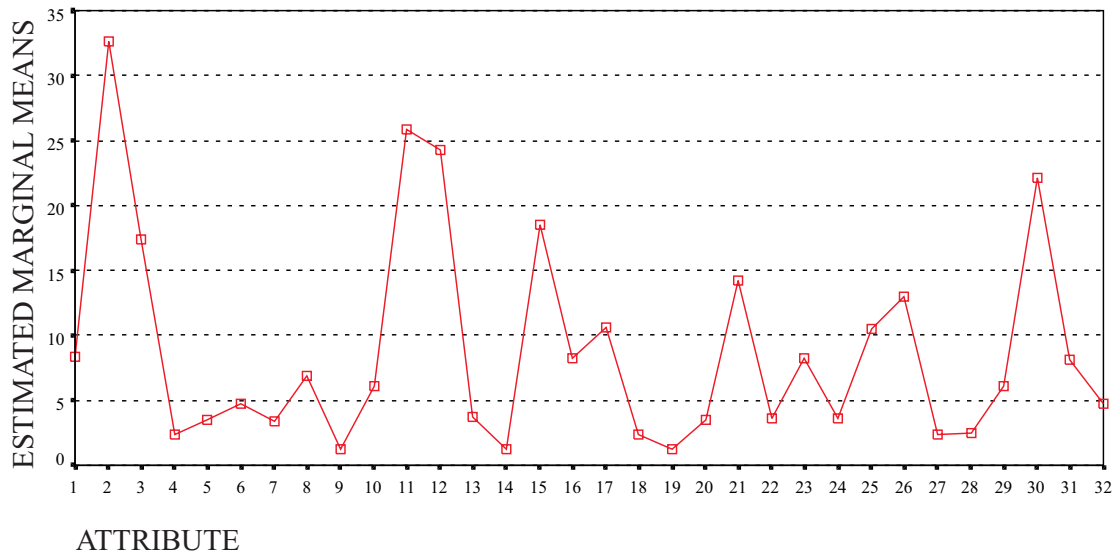
question and the attribute influence the number of people who consult the meaning of an attribute.

Source	Type III square sum	gl.	Mean square	F	Sig.
Statistical model	8896.830	34	261.671	9.529	0.000
Intersection	10194.849	1	10194.849	371.244	0.000
PROJECT	696.212	3	232.071	8.451	0.000
ATTRIBUTE	8200.618	31	264.536	9.633	0.000
Error	2553.902	93	27.461		
Total	21645.580	128			
Corrected total	11450.731	127			

Table 10.26: ANOVA for the frequency of schema attribute consultation.

This goes to say that the fact that a subject consults the meaning of an attribute depends on both the project in question and the attribute. That is, there are attributes that are consulted more often than others (by nature) and projects that appear to raise more doubts about the meaning of the schema attributes for some subjects than others.

Figure 10.6 shows the estimated values for the mean percentage of subjects who consult each attribute by attributes. It is found that, generally, the frequency is quite low for all the attributes (it is never over 35% of the subjects).



cmcm

Figure 10.6: Estimated values for the mean frequency of consultation of each attribute.

For the purposes of this study, we will focus on the most often consulted attributes, and, therefore, a frequency limit of 15% of subjects is set. Looking at Figure 10.6, the most often consulted attributes are 2, 3, 11, 12, 15 and 30, which are, respectively, aspect, automation,

data cost, adequacy criterion, element and repeatability. These attributes are considered to represent concepts that are not intuitive or easy to understand for the subjects.

On the other hand, Figure 10.7 shows the estimated values for the mean percentage of subjects who consult schema attributes by project. It is found that there is a project that appears to behave differently from the others, and this is the real-time project. This appears to indicate that more complex projects make subjects reflect at greater length on the techniques to be selected and, therefore, also on the attributes to be considered. Again, the frequency is not found to be high (in no case is it over 15%).

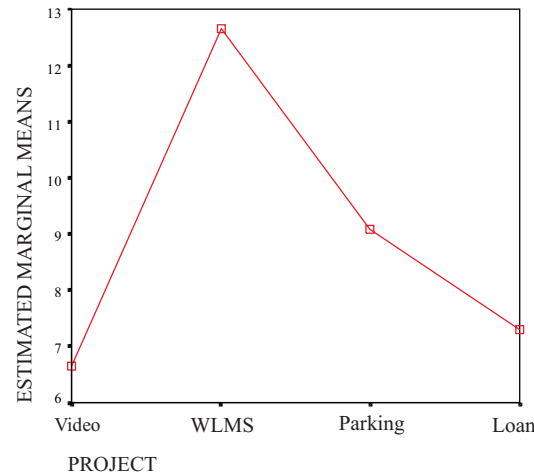


Figure 10.7: Estimated values for the mean frequency of consultation of each attribute.

In view of the analysed data, it can be deduced that **the most often consulted attributes appear to be the attributes that represent concepts that are not intuitive or are more difficult for the subjects to interpret.**

#### 10.4.3.5 Conclusions on Schema Usability

The number of problems found during selection, the sort of problems, the number of schema attributes that are problematic for selection purposes and the sort of attributes were taken into account to evaluate schema usability. The first two variables provide relative results on schema behaviour as compared with books, whereas the latter two provide absolute results, irrespective of books.

From the relative comparison of the schema against books, it was found that the subjects have fewer problems using the schema than with books. It was also discovered that the frequency of appearance of each problem will be lower and that the main problems encountered by the subjects using the schema are the result of there being attributes that are not instantiated in the schema, as well as there being too much information (a problem that had been predicted by an expert and which could be solved by building a tool). On the other hand, the problems concerning the selection with books were predicted in the introduction to this research, and these are the poor organisation of the available information, as well

as missing information of interest and the existence of information that is unnecessary for selection purposes.

From the absolute comparison, it was found that the frequency with which the meaning of attributes is consulted is low, and that the most often consulted attributes appear to be the attributes that represent concepts that are not intuitive or are difficult for the subjects to interpret.

From all this, it can be deduced that **characterisation schema usability is acceptable, although there is room for improvement**. It is acceptable insofar as the frequencies of appearance of problems are lower than for books, and the frequency with which the meaning of the attributes is consulted is also low. However, schema usability could be improved, for example, by building a tool to make the information easier to handle. It could also be improved by assuring that, every time a technique is added, the entry contains as much information as possible.

#### 10.4.4 Schema Completeness

The response variables covered for the study of schema completeness are as appear in Table 10.4:

- How much information a subject uses for selection.
- The sort of information used for selection.
- How much information for selection each subject considers to be missing.
- The sort of information that is considered to be missing.

For this purpose, four statistical analyses were carried out, one for each variable.

##### 10.4.4.1 Amount of Information Used for Selection

Table 10.27 reflects the values for the mean and standard deviation of how much information is considered by each subject for selection as a total and with respect to each alternative under consideration. It is found that the mean is lower for books than for the schema and that the four projects behave similarly.

Table 10.28 shows the result of the ANOVA for the amount of information used for selection. It is found that the null hypothesis of equality of the amount of information used in selection can be rejected (the significance level of the model is under 0.05). The only factor that affects the response variable is the method used for selection (significance level of under 0.05).

Table 10.29 and Figure 10.8 show the estimated values for the mean amount of information used in selection with and without the schema. It is found that more information is used for selection when using the schema than when using books, where the difference is almost double.

Factor	Alternative	Mean	Std. Devi.	N
Method	Books	7,79	3,59	86
	Schema	12,60	6,43	86
Project	Video	10,93	6,68	45
	WLMS	10,37	5,66	41
	Parking	9,53	4,81	45
	Loan	9,95	5,70	41
	Total	10,20	5,73	172

Table 10.27: Amount of information considered in selection.

Source	Type III square sum	gl.	Mean square	F	Sig.
Statistical model	1062,026	7	151,718	5,477	0,000
Intersection	17820,658	1	17820,658	643,281	0,000
METHOD	979,659	1	979,659	35,363	0,000
PROJECT	42,737	3	14,246	0,514	0,673
METHOD * PROJECT	21,873	3	7,291	0,263	0,852
Error	4543,253	164	27,703		
Total	23492,000	172			
Corrected total	5605,279	171			

Table 10.28: ANOVA for the amount of information used in selection.

Method	Mean	Std. error	Confidence interval at 95%	
			Lower bound	Upper bound
Books	7,803	0,568	6,681	8,925
Schema	12,582	0,568	11,460	13,705

Table 10.29: Estimated values of the mean amount of information used.

This result says that the amount of information used in the selection depends only on the method used (not on the project under consideration or a combination of the two) and more information is used with the schema than with books. That is, the data appear to indicate that **the schema provides more useful information for selection purposes than books**.

#### 10.4.4.2 Sort of Information Used in Selection

This response variable reflects the sort of information that the subjects have used to make the selection. Table 10.30 shows that the subjects have used forty-seven types of different information. Again, this is a qualitative response variable (sort of information used), which will be quantified as usual, that is, taking the percentage of people who have used the

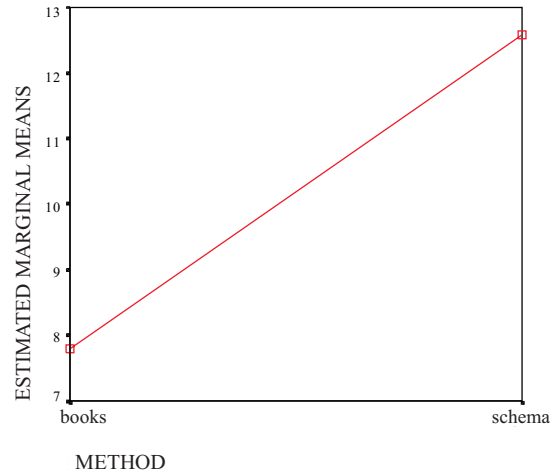


Figure 10.8: Estimated values of the mean amount of information used.

information in question as the response variable and adding the sort of information as a new factor.

Table 10.31 shows the results of the ANOVA on the frequency of use of the information. The null hypothesis of equality of frequencies for all information can be rejected, as the significance level of the model is under 0.05. Also, it is found that the factors that influence frequency are: the method used for selection, the sort of information, the combination of method and information and the combination of project and information, as their significance is under 0.05.

This means that different information is used with each method and also with each project.

As regards the project, it was not expected that the information used for the selection would vary so much as to be statistically significant. It was expected that the same information would always be consulted, irrespective of the project or, at least, that there would be a subset of information that the subjects would always consult.

The information forms used for selection purposes were inspected to gain a better understanding of the influence of the project. They show that the subjects consult the information they see fit, that is, the subjects did not follow the recommended process (the process proposed in Section 5.6) The subjects were told that, to make the selection, they should look for the techniques whose attributes coincided with the project situation in question. However, it appears that the subjects select any information they like. This can jeopardise selection, as the subjects may forget to consult the value of an attribute of interest for the technique, which means that the schema must be improved. The possible solution, which will be set out at the end of this chapter, is to add a process to the schema, detailing how the selection should be made (that is, what attributes have to be consulted).

As regards the method, it is to be expected that either the schema or the books are providing useful information for the selection that is not covered by the other method. This means that one of the two methods is providing more useful information for selection purposes

N	Description of information
1	Scope
2	Effectiveness
3	Software architecture
4	Aspect
5	Automation
6	Benefits
7	Number of generated cases
8	Life cycle
9	Cost of application
10	Completeness
11	Knowledge
12	Cost of tool
13	Cost of regression
14	Cost of data
15	Adequacy criterion
16	Dependencies
17	Element
18	Comprehensibility
19	Tool environment
20	Inputs
21	Time taken to run the test
22	Experience
23	Focus
24	Sources of information
25	Tools
26	Tools used
27	Historical
28	Identifier
29	Team independence
30	Technique type (structural or dynamic)
31	Programming language
32	Methodology
33	Object
34	Opinion
35	Personnel
36	Precision
37	Problems
38	Software type
39	Purpose
40	Project
41	Reference projects
42	Repeatability
43	Results
44	Outputs
45	Support
46	Size
47	Possible variants of a technique

Table 10.30: Description of information used for selection.

(that is, it is more complete). Figure 10.9 shows the estimated values for the mean frequency of each sort of information in each case.



Source	Type III square sum	gl.	Mean square	F	Sig.
Statistical model	214421,183	237	904,731	13,217	0,000
Intersection	142994,281	1	142994,281	2088,994	0,000
METHOD	19872,400	1	19872,400	290,315	0,000
PROJECT	415,800	3	138,600	2,025	0,113
INFUSED	130135,356	46	2829,029	41,329	0,000
METHOD * PROJECT	14,755	3	4,918	0,072	0,975
METHOD * INFUSED	50317,570	46	1093,860	15,980	0,000
PROJECT * INFUSED	13665,303	138	99,024	1,447	0,015
Error	9446,274	138	68,451		
Total	366861,738	376			
Corrected total	223867,457	375			

Table 10.31: ANOVA for the frequency of use of information.

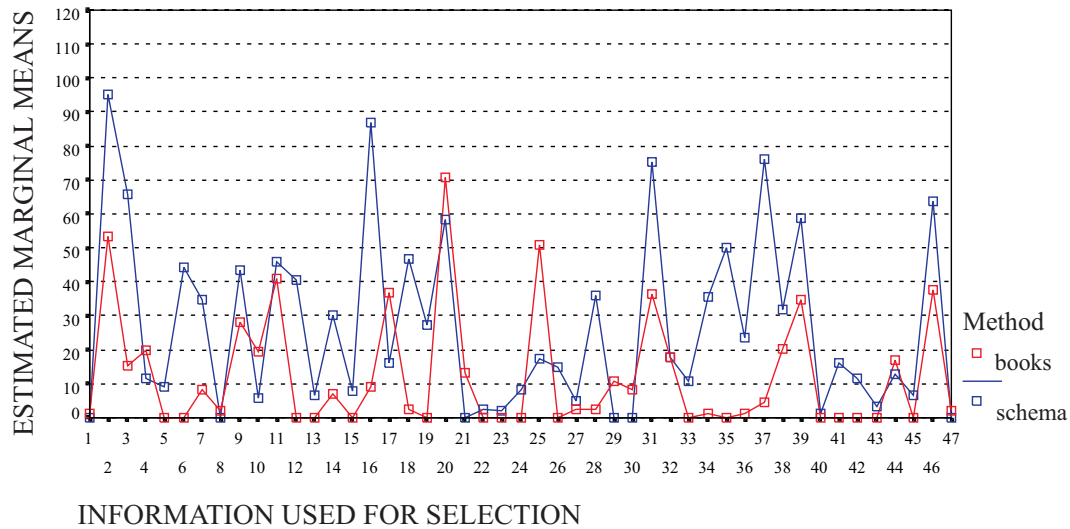


Figure 10.9: Estimated values for the frequency of use of information.

Table 10.32 shows the results of the analysis of simple correspondences for the method and information variables. It is found that the value of chi-squared is very high, and the significance level is under 0.05, which means that both variables are closely related and that the results of the analysis of simple correspondences are valid. Accordingly, the levels of the two variables can be represented as points in a one-dimensional space (straight line), which will indicate the distance between these levels.

Figure 10.10 shows the values obtained using the analysis of simple correspondences for each of the levels of the variables of interest. The points in red represent the value of the information and the lines in blue represent the value of the methods. Mapping the points and the two straight lines to the x-axis, we get the distance between the information and the

Dimension	Gen value	Inertia	Chi-square	Sig.	Proportion of inertia	
					Explained	Accumulated
1	0,528	0,279			1,000	1,000
Total		0,279	440,051	0,000	1,000	1,000

Table 10.32: Analysis of correspondences between the method and information variables.

methods of selection.

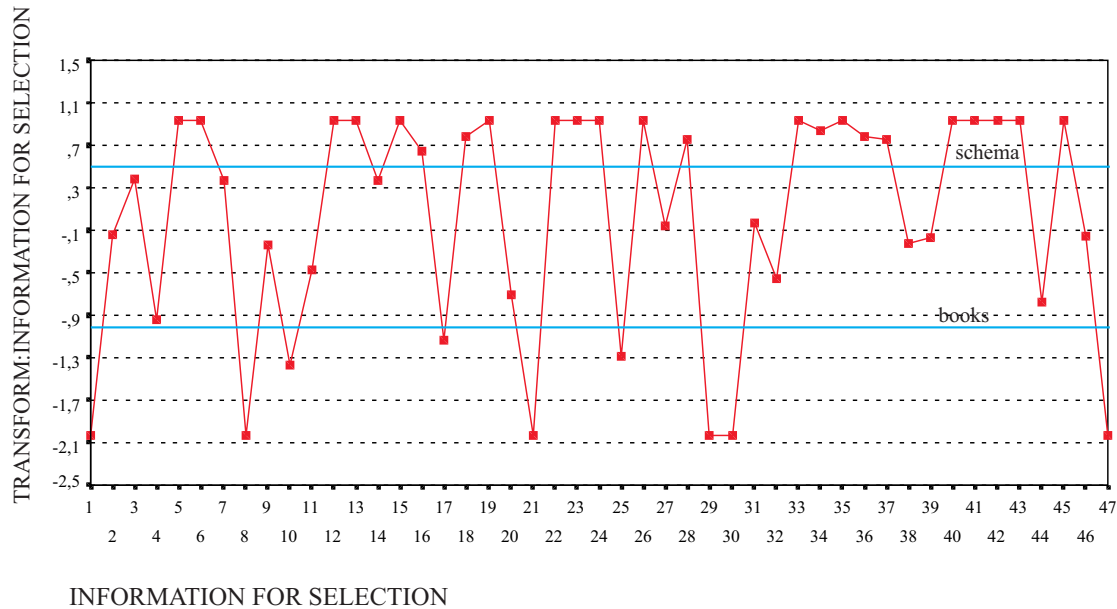


Figure 10.10: Column and row points for the method and information variables.

Figure 10.10 shows that the points above or close to the line refer to information that will be used only when the characterisation schema is being used. The points that are below or very close to the books line refer to information that will be used only when using books. The intermediate points that are between and not close to either of the two lines refer to information that will be used indistinctly for selection purposes.

Looking at the sort of information used for selection purposes, Figure 10.10 shows that there are three types: information used only in books, information used only with the schema and information used in both cases. Each one will be analysed below.

- The information used only with books is numbered 1, 8, 10, 17, 21, 25, 29, 30 and 37. This information can be divided into two types:
  - Information explicitly set out by the schema. This is the case of information 1, 8, 10, 17 and 25. This information appears explicitly in the repository provided, either grouped under several attributes or under another name.

- Information not explicitly set out in the schema. This type of information includes information that does not appear or appears implicitly in the schema.

The information that appears implicitly is the time it takes to run the test (it will be a function of the cost of application of the technique, of the characteristics of the people who are to use the technique, of whether they are going to use tools and the number of cases generated by the technique), and whether the technique is a white-box or black-box technique (it can be deduced from the inputs required by the technique).

The information not set out by the schema is the independence of the development and the testing teams and the possible variants of a technique. The subjects need to know whether the testing teams are independent to find out whether the inputs required by the technique will be available. It is not believed to be necessary to add this attribute to the schema, as the information we are really looking for is the availability of the inputs. The possible variants of a technique are useful for identifying techniques of the same family, which can be ascertained by consulting the value of the adequacy criterion attribute.

- Of the information that is only used with the schema, the information related to past uses of the technique (historical level) and pragmatic aspects of tools, such as the cost of tool purchase, support provided by the manufacturer, etc., are noteworthy.
- The information used with both is: information related to the objective of the test, the technique itself, the object it can be used to test, the test cases generated and agent knowledge. All the information belongs to the operational and tactical schema levels, and tools are not mentioned.

From the data analysis, it can be concluded that **there is information that only appears in the characterisation schema, and the information that appears only in books can be somehow deduced from the contents of the characterisation schema.**

#### 10.4.4.3 Amount of Information Missing for Selection

Table 10.33 shows the mean and standard deviation for the information that the subjects considered to be missing during selection. It is found that they consider that less information is missing when they use books than when they use the schema. The amount of information does not appear to vary from one project to another.

Table 10.34 shows the results of the ANOVA. It is found that the null hypothesis of equality of information for selection that is missing can be rejected. It is also found that the factor that influences the amount of information considered to be missing is the method, as its significance is under 0.05.

Factor	Alternative	Mean	Std. Devi.	N
Method	Books	2,12	1,91	86
	Schema	3,72	3,51	86
Project	Video	3,42	3,00	45
	WLMS	2,10	1,97	41
	Parking	2,73	2,18	45
	Loan	3,39	4,06	41
Total		2,92	2,93	172

Table 10.33: Mean and standard deviation for the amount of missing information.

Source	Type III square sum	gl.	Mean square	F	Sig.
Statistical model	165,607	7	23,658	2,973	0,006
Intersection	1449,419	1	1449,419	182,114	0,000
METHOD	108,996	1	108,996	13,695	0,000
PROJECT	46,544	3	15,515	1,949	0,124
METHOD * PROJECT	8,268	3	2,756	0,346	0,792
Error	1305,253	164	7,959		
Total	2936,000	172			
Corrected total	1470,860	171			

Table 10.34: ANOVA for the amount of missing information.

Table 10.35 and Figure 10.11 show the estimated values for the mean amount of information for selection that is missing using the schema and books. It shows that the subjects think that more information is missing when using the schema than when using books. This is an interesting point that was not expected. It was expected to find that less information was missing from the schema than from books.

Method	Mean	Std. error	Confidence interval at 95%	
			Lower bound	Upper bound
Books	2,110	0,305	1,508	2,711
Schema	3,704	0,305	3,102	4,305

Table 10.35: Estimated values for the amount of missing information.

This result cannot be explained until the sort of missing information has been examined.

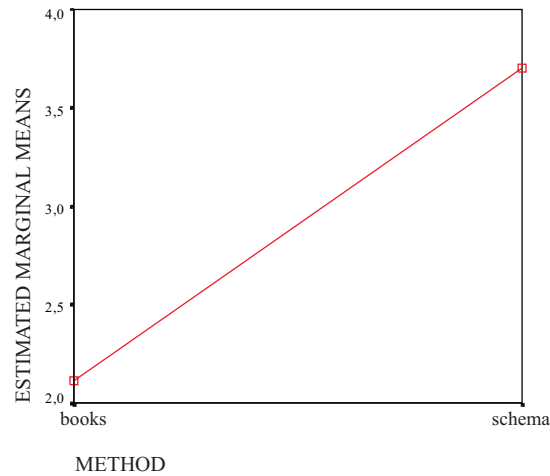


Figure 10.11: Estimated values for the amount of missing information.

#### 10.4.4.4 Sort of Information that is Missing for Selection

Table 10.36 shows the sort of information that the subjects felt was missing during selection. It is found that forty-two types of information were missing. This variable (sort of information that is missing for selection) is again qualitative and will be quantified as usual, studying the frequency of appearance of each item of information and introducing the items as a factor.

Table 10.37 shows the results of the ANOVA for the frequencies of appearance of each item of missing information. It is found that the null hypothesis of equality of frequencies can be rejected (the significance level of the model is under 0.05). The factors that influence the response variable are: method used, project under study, the information considered and the combination of method and information that is missing, as their significance is under 0.05.

This means that the information that a subject considers to be missing depends on the method used, but, irrespective of the sort of information, the subjects miss a different amount of information depending on the projects. This is shown in Figure 10.12 and Figure 10.13.

Table 10.38 shows the results of the analysis of simple correspondences for the method and missing information variables. It is found that the value of chi-squared is very high and the significance level is under 0.05, which means that both variables are closely related and that the results of the analysis of simple correspondences are valid. Accordingly, the levels of both variables can be represented as points in a one-dimensional space (a straight line), which will indicate the distance between the levels.

Figure 10.14 shows the values obtained using the analysis of simple correspondences for each of the levels of the variables of interest. The points in red represent the value of the missing information and the lines in blue represent the value of the methods. Mapping the points and the two straight lines to the x-axis, we have the distance between the missing information and the methods of selection.

Figure 10.14 shows that the points above or very close to the schema line refer to the missing information only when the characterisation schema is being used. The points below

N	Description of information
1	System architecture in which the technique can be used
2	Aspect
3	Automation
4	Benefits
5	Number of generated cases
6	Cost of application
7	Completeness
8	Knowledge
9	Cost of regression
10	Cost of tools
11	Adequacy criterion
12	Time taken to learn to use a technique
13	Dependencies
14	Effectiveness
15	Element
16	Comprehensibility
17	Environment
18	Inputs
19	Team experience
20	Tools
21	Tools used
22	Historical
23	Identifier
24	Characteristic information
25	Language
26	Methodology
27	Opinion
28	Personnel
29	Development personnel
30	Precision
31	Problems
32	Technique procedure
33	Purpose
34	Reference projects
35	Support
36	Size
37	Technique application time
38	Software type
39	Tool use
40	Effectiveness variance
41	Outputs
42	Schema

Table 10.36: Description of the missing information.

or very close to the books line refer to the missing information only when books are being used. The intermediate points that are between but are not close to either of the two lines refer to the information that will be missing during selection indistinctly.

Analysing the missing information, one might think that a lot of information is missing from the schema, as there are many points close to its line (indeed, this was the result of the study on the amount of information). However, looking at the problems it addresses

Source	Type III square sum	gl.	Mean square	F	Sig.
Statistical model	34394,909	212	162,240	5,309	0,000
Intersection	15851,795	1	15851,795	518,762	0,000
METHOD	1187,969	1	1187,969	38,877	0,000
PROJECT	444,007	3	148,002	4,843	0,003
INFOMISS	21494,614	41	524,259	17,157	0,000
METHOD * PROJECT	94,699	3	31,566	1,033	0,380
METHOD * INFOMISS	7329,706	41	178,773	5,850	0,000
PROJECT * INFOMISS	3590,322	123	29,190	0,955	0,600
Error	3758,510	123	30,557		
Total	54249,477	336			
Corrected total	38153,418	335			

Table 10.37: ANOVA for the frequency of appearance of missing information.

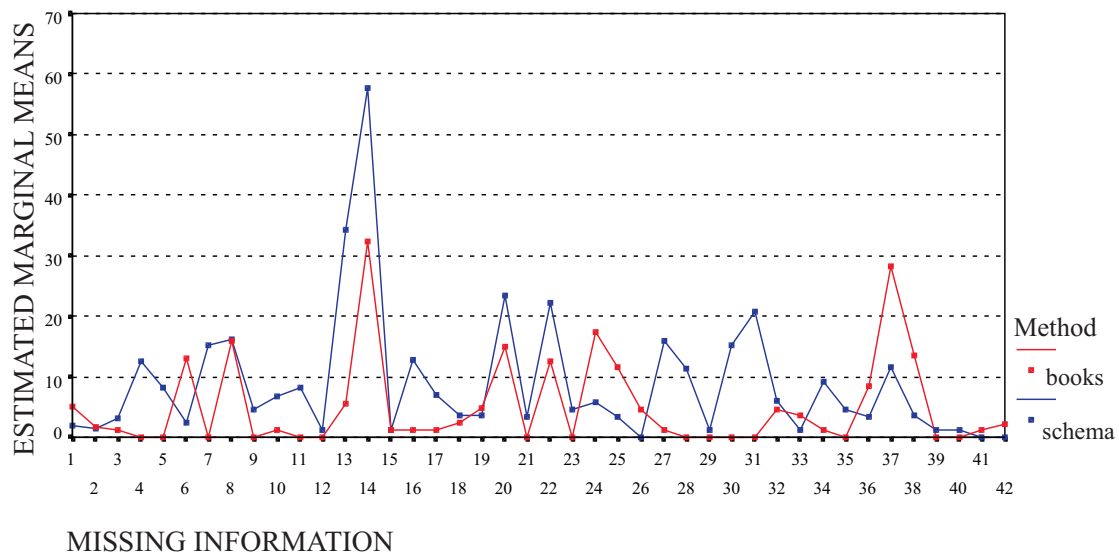


Figure 10.12: Estimated values for the mean frequency of missing information.

Dimension	Gen value	Inertia	Chi-square	Sig.	Proportion of inertia	
					Explained	Accumulated
1	0,586	0,344			1,000	1,000
Total		0,344	172,469	0,000	1,000	1,000

Table 10.38: Analysis of correspondences between the method and missing information variables.

(all, except 12, 29 and 40), it can be deduced that almost all refer to attributes present in the schema that have not been filled in, but information about which, interestingly, does not

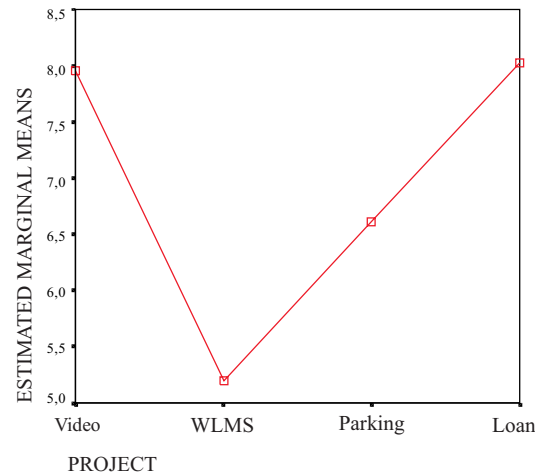


Figure 10.13: Estimated values for the mean frequency of missing information.

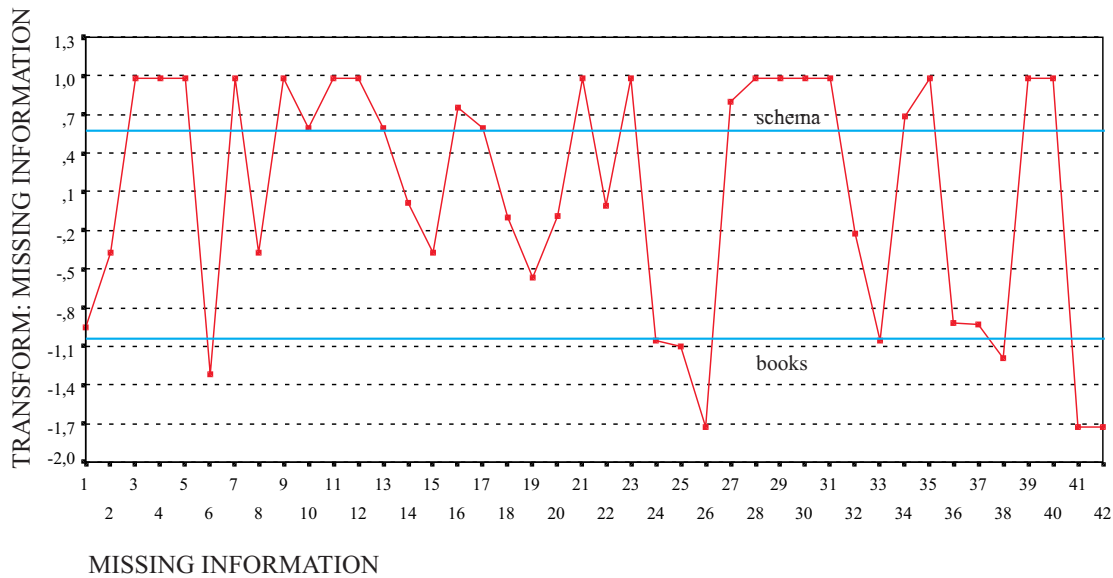


Figure 10.14: Column and row points for the method and missing information variables.

appear in books either. This phenomenon can be explained by the fact that it is much more evident that relevant information for selection purposes is missing in the characterisation schema than in books, mainly because the information is much more clearly structured in the schema.

Moreover, the only information missed during schema use is:

- Time required to learn the technique. This can be deduced from the knowledge of the agents and the effort of application of the technique.
- Characteristics of the development personnel. This information is useful for identifying the type of defects that are to be detected during testing. In this case,



the procedure should be to ascertain the defect types to be detected and look for a technique suited for these defects.

- Variance of the effectiveness. In this case, the subjects want to find out not only the mean effectiveness, but also the variance of the technique. It is a good idea, then, for effectiveness to be represented in the schema as a range and not as a single value.

The data appear to indicate that **all the information that the subjects need for selection is in the schema either implicitly or explicitly**. The only modification made is to change the type of value of one of the schema attributes.

#### 10.4.4.5 Conclusions on Schema Completeness

This section addressed both the information the subjects used during selection and the missing information. The main finding of this study is that **it is important for the characterisation schema to be completely instantiated** for users to be able to take full advantage of the schema and for them to consider it useful (this can pose a threat to its utility). Another interesting point observed is that subjects are not always able to ascertain the value of variables that do not appear in the schema, but whose values can be easily deduced from the schema. This is the case of the time it will take to apply the technique. If the cost of application of the technique, the knowledge of the people who are to use the technique, whether or not tools are to going to be used and the size of the software are known, it is easy to find out how long it will take to apply the technique.

#### 10.4.5 Schema Effectiveness

As shown in Table 10.4, there are three response variables to be taken into account to study characterisation schema effectiveness:

- The number of techniques considered to make the selection.
- The number of techniques that the subject has selected in each case.
- The techniques selected by the subjects.

Three statistical analyses are presented below, one for each response variable.

##### 10.4.5.1 Number of Techniques Considered

Table 10.39 shows the mean and standard deviation for the number of techniques considered by each subject to make the selection. It is found that more techniques were available for selection using the schema than using books and, interestingly, the standard deviation for the schema (not for books) is almost zero. This means that the subjects using the schema

Factor	Alternative	Mean	Std. Devi.	N
Method	Books	8,33	2,29	86
	Schema	12,99	0,11	86
Project	Video	10,64	2,80	45
	WLMS	10,66	2,84	41
	Parking	10,56	2,95	45
	Loan	10,78	2,88	41
<b>Total</b>		10,66	2,84	172

Table 10.39: Mean and standard deviation for the number of techniques studied.

have all considered all the techniques presented, which was not the case for books. Also, it is found that the number of techniques studied for the different projects is similar.

Table 10.40 shows the results of the ANOVA for the number of techniques studied. It is found that the null hypothesis of equality of means of the number of techniques studied can be rejected (the significance level of the model is under 0.05). It is also found that the influential factor is the method used in selection, as the significance level is under 0.05.

Source	Type III square sum	gl.	Mean square	F	Sig.
Statistical model	936,007	7	133,715	49,086	0,000
Intersection	19490,460	1	19490,460	7154,785	0,000
METHOD	930,571	1	930,571	341,605	0,000
PROJECT	0,630	3	0,210	0,077	0,972
METHOD * PROJECT	0,498	3	0,166	0,061	0,980
Error	446,755	164	2,724		
Total	20917,000	172			
Corrected total	1382,762	171			

Table 10.40: ANOVA for the number of techniques studied.

Table 10.41 and Figure 10.15 show the estimated values for the mean of number of techniques studied with the schema and using books. It is found that the mean is lower for the use of books than for the use of the schema, and variance is much lower for schema use. This is interesting, as it indicates that the starting set of techniques is easily identified using the schema, which is not the case for books, where not everyone identifies the same original set of techniques.

So, the data appear to indicate that **the schema helps to identify the available testing techniques more easily.**

Method	Mean	Std. error	Confidence interval at 95%	
			Lower bound	Upper bound
Books	8,330	0,178	7,978	8,682
Schema	12,989	0,178	12,637	13,341

Table 10.41: Estimated values of the number of techniques studied.

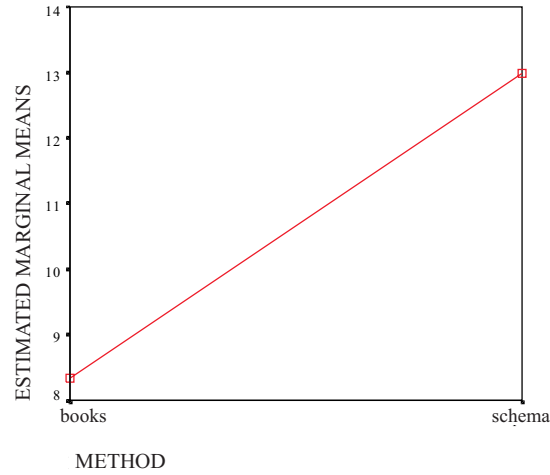


Figure 10.15: Estimated values for the number of techniques studied.

#### 10.4.5.2 Number of Selected Techniques

Table 10.42 shows the mean and standard deviation for the total number of selected techniques and with respect to each alternative of interest. It is found that the mean is lower for the use of the schema than for the use of books and is similar for all projects.

Factor	Alternative	Mean	Std. Devi.	N
Method	Books	3,21	1,31	86
	Schema	2,26	0,97	86
Project	Video	2,73	1,32	45
	WLMS	2,95	1,09	41
	Parking	2,91	1,35	45
	Loan	2,32	1,13	41
	<b>Total</b>	2,73	1,25	172

Table 10.42: Mean and standard deviation for the number of techniques selected.

Table 10.43 shows the result of the ANOVA for the number of selected techniques. It is found that the null hypothesis of equality in the number of techniques selected can be rejected (the significance of the model is under 0.05). It also shows that the only influential factor is the method, although the project would be considered if the confidence level is lowered from

95% to 90%, as the significance level is under 0.1.

Source	Type III square sum	gl.	Mean square	F	Sig.
Statistical model	51,722	7	7,389	5,663	0,000
Intersection	1278,712	1	1278,712	980,059	0,000
METHOD	38,495	1	38,495	29,504	0,000
PROJECT	9,747	3	3,249	2,490	0,062
METHOD * PROJECT	2,906	3	,969	0,742	0,528
Error	213,976	164	1,305		
Total	1550,000	172			
Corrected total	265,698	171			

Table 10.43: ANOVA for the number of techniques selected.

Table 10.44 and Figure 10.16 show the estimated values for the mean of the number of techniques selected in the case of books and the schema. It is found that fewer techniques were selected using the schema than with books.

Method	Mean	Std. error	Confidence interval at 95%	
			Lower bound	Upper bound
<b>Books</b>	3,204	0,123	2,960	3,448
<b>Schema</b>	2,257	0,123	2,013	2,500

Table 10.44: Estimated values for the mean number of techniques selected.

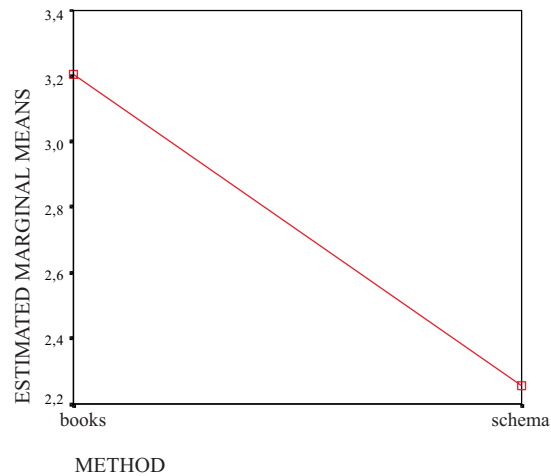


Figure 10.16: Estimated values for the mean number of techniques selected.

This goes to say that the number of techniques selected by a subject depends on the whether the schema or books are being used. It is interesting to observe how dependency

on the project is only appreciable if the significance level is lowered, as a greater influence of the project was expected. Let us take a look at what happens with the sort of techniques selected. It also indicates that fewer techniques are selected using the schema than using books. This might be because the techniques selected with the schema are better suited and the subjects feel that they do not need to select any more. It could also be due to the different sort of techniques that appear. **However, these are only conjectures that cannot be explained until the sort of selected techniques are known.**

#### 10.4.5.3 Sort of Techniques Selected

Table 10.45 shows the sort of techniques selected by subjects. It shows that 27 techniques were taken into account. This variable (sort of techniques selected) is again qualitative and will be quantified as usual, that is, studying the frequency of selection of each technique as the response variable and adding the selected techniques as a factor.

N	Technique
1	Automated testing tools
2	Branch testing
3	Black box
4	Paths, coverage
5	Threads, coverage
6	Data flow testing
7	Decisions, coverage
8	Domain testing
9	All-du-paths
10	Fault seeding
11	Flow graphs and path testing
12	Function testing
13	Interface testing
14	All-uses
15	Mutation
16	Selective mutation
17	Object oriented
18	All-possible-rendezvous
19	Random tests
20	Performance tests
21	All-p-uses
22	Code inspection
23	Sentences, coverage
24	Limit values
25	Statement testing
26	Structural testing
27	Predicates, coverage

Table 10.45: Description of the selected techniques.

Table 10.46 shows the results of the ANOVA for the frequencies of selection of each technique. It is found that the null hypothesis of equality of frequency of selection of the different techniques can be rejected, as the significance level of the model is under 0.05. The factors that influence the frequency of selection are the method used for selection, the

technique in question and the combination of method and technique, as their significance is under 0.05.

Source	Type III square sum	gl.	Mean square	F	Sig.
Statistical model	55961,336	132	423,950	4,513	0,000
Intersection	22732,934	1	22732,934	242,017	0,000
METHOD	797,877	1	797,877	8,494	0,005
PROJECT	198,755	3	66,252	0,705	0,552
TECHNSEL	15345,313	625	613,813	6,535	0,000
METHOD * PROJECT	70,466	3	23,489	0,250	0,861
METHOD * TECHNSEL	32440,341	25	1297,614	13,815	0,000
PROJECT * TECHNSEL	7108,584	75	94,781	1,009	0,484
Error	7044,834	75	93,931		
Total	85739,104	208			
Corrected total	63006,170	207			

Table 10.46: ANOVA for the frequency of selection of each technique.

This result is quite surprising, as it says that the selection of a technique depends exclusively on the method used for selection (schema or books). The project was expected to somehow influence selection, although this means that the proposed project situations are not so disparate for the purposes of selecting techniques as for a statistical difference to be appreciated.

The fact that the selection of a technique depends on the method used for selection is to be expected. This is assumed to be due to the fact that the schema contained techniques that the books did not and which, apparently, were selected as being better suited. Figure 10.17 shows the estimated values for the mean frequency of selection of each technique.

Table 10.47 shows the results of the analysis of simple correspondences for the method and technique variables. It shows that the value of chi-squared is very high and the significance level is under 0.05, which means that both variables are closely related and that the results of the analysis of simple correspondences are valid. Accordingly, the levels of both variables can be represented as points in a one-dimensional space (a straight line), which will indicate the distance between the levels.

Dimension	Gen value	Inertia	Chi-square	Sig.	Proportion of inertia	
					Explained	Accumulated
1	0,928	0,862			1,000	1,000
Total		0,862	403,346	0,000	1,000	1,000

Table 10.47: Analysis of correspondences between the method and technique variables.

Figure 10.18 shows the values obtained using the analysis of simple correspondences for each of the levels of the variables of interest. The points in red represent the value of the

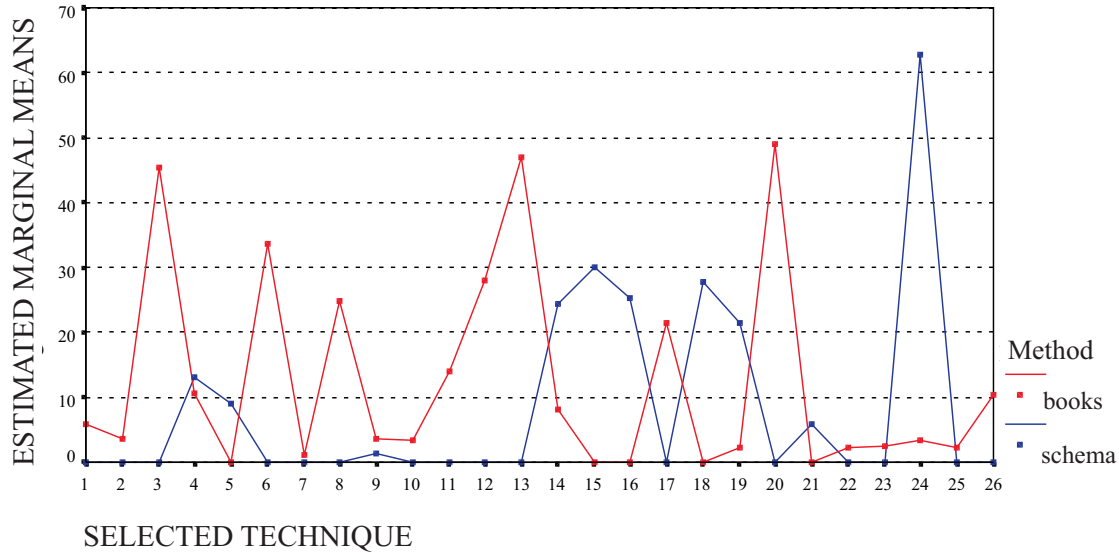


Figure 10.17: Estimated values for the mean frequency of technique selection.

techniques and the lines in blue represent the value of the methods. Mapping the points and the two straight lines to the x-axis, we have the distance between the techniques and the methods of selection.

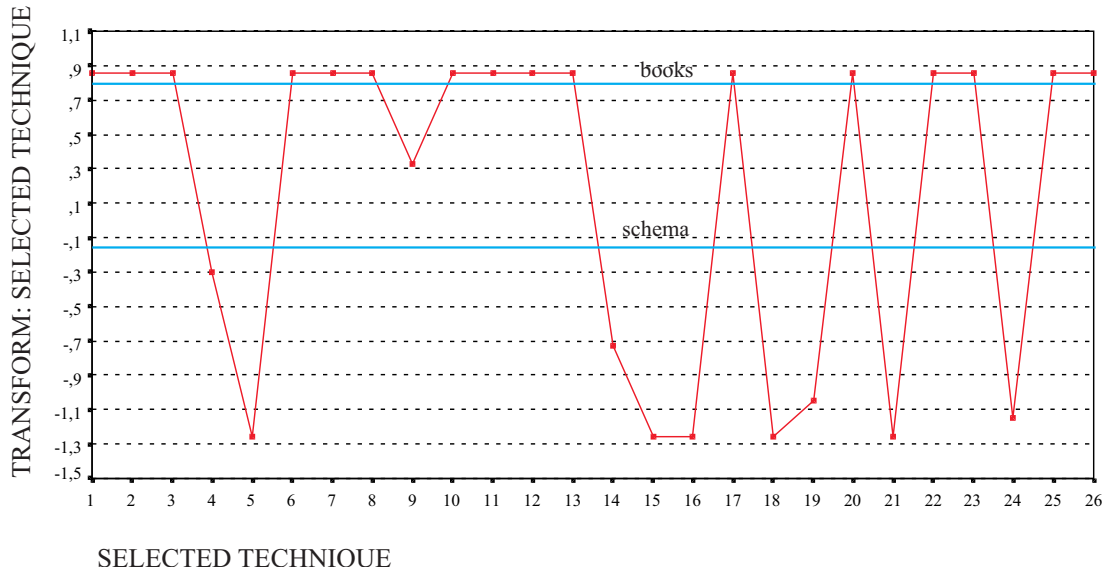


Figure 10.18: Column and row points for the method and technique variables.

Figure 10.18 shows that the points above or very close to the schema line refer to the techniques selected only when the characterisation schema is used. The points below or very close to the books line refer to the techniques selected only when books are used. The intermediate points that are between but are not close to either of the two lines refer to techniques that will be selected indistinctly.

The idea for checking the effectiveness of the characterisation schema was to classify each selected technique as suited or unsuited for the problem in question and then study the number of suited and unsuited techniques for the problem that had been selected using both the schema and books. However, this turned out to be impossible because a detailed study of the techniques selected using books revealed the following:

- The subjects confuse technique with family, as in the case of 3, 6, 8, 11, 12, 17 and 26. This means that they did not reach a granularity level that was sufficient to understand what was and what was not a technique. This very often occurred, as the mean frequencies of selection planned for each *technique* are 35%, 25%, 15%, 28%, 22% and 10%, as shown in Figure 10.18. There are two possible interpretations for this result: all the techniques of one family are equal (in which case one might wonder why they exist), or they are not equal (which has been demonstrated by several studies mentioned in Chapter 2). As there are studies that back the differences between techniques of the same family, it is deduced that what actually happened is that the subjects were unable to see the differences between the different techniques because of the way they were presented in the books.
- The subjects confuse technique with test type. This is the case of 1, 13 and 20. This is related to some extent with the above-mentioned granularity problem. Additionally, it is interesting to note that none of the test types selected are useful for evaluating correctness (which is what they were being asked to do). This points to the possibility of the goal of the techniques not being very clear in books, and that the schema achieves a better-targeted selection.

Finally, and merely out of interest (they are not relevant for the analysis), it is noteworthy that the subjects using books do not agree on the nomenclature to be applied, although it is not clear whether by choice or because they have not identified technique equality. Thus, it is found that the techniques 2 and 7 are the same, and techniques 23 and 25 are the same. Another interesting point is that the subjects who made the selection using books included a static technique (code inspection) when they were asked to choose only dynamic ones (this is a rare case where the subject decided to use complementary books apart from the ones given).

The two above-mentioned points ratify the original hypothesis that the schema achieves better suited selections insofar as they are better targeted.

Furthermore, the fact that the average developer has little knowledge of testing can be ratified, which means that the schema is beneficial in that it stops concepts being confused.

Finally, we can conclude by saying that **the hypothesis on schema effectiveness has not been able to be tested, because the state of technique selection is more immature than it was believed to be**. The only claim that can be made for the time being is that the selection using the schema is better *tuned*, which is a step prior to finding out what was to be ascertained.



Interestingly, the subjects select techniques with which they are better acquainted, which ratifies another original hypothesis that stated that developers selected the techniques with which they feel most at home.

Finally, it was found that testing books do not address selection. That is, they describe the testing techniques, focusing exclusively on their application.

#### 10.4.5.4 Conclusions on Schema Effectiveness

After studying schema effectiveness, it was found that the number of original techniques is lower for books than with the schema and varies from subject to subject; the number of selected techniques is lower for the schema than for books; and the subjects select either families of techniques, things that are not techniques or techniques with which they are very familiar.

Combining these results, the conclusion is that **the subjects using books are unable to distinguish between a technique and a family or something that is not a technique** (which is indicated by the fact that the set of original techniques is different for the subjects who made the selection using books and who select things that are not techniques), even though they were given an explanation as to what a technique is. As none of the subjects is *incompetent* for performing the task (they would also have failed in the selection using the schema), this could be explained by saying that **books are confusing as regards the information they provide**. This could also be the reason why the subjects tend to select more techniques, gaining more assurance that the tests will turn out right, and why they choose techniques with which they are very familiar. Finally, it should be stressed that **the schema leads to more precise selections and it was not possible to demonstrate whether or not they are more effective than books**. This is left for the future.

#### 10.4.6 Schema Satisfaction

To assess satisfaction with the schema, the subjects are asked to subjectively summarise their perceptions of the selection process. Therefore, they are asked about questions related to efficiency, usability, utility, etc.

- *Advantages and Disadvantages of Using the Schema.* Generally, the subjects like the schema. However, they do stress the fact that there are uninstantiated attributes. They also think that the schema contains too much information. This again suggests the need to build a tool to make the information the schema contains easier to handle.
- *Would you be prepared to use it?* Generally, all the subjects would be prepared to use the schema, provided they do not have to instantiate it. The subjects do not feel capable of instantiating the schema for the techniques.

- *Improvements you would make.* They miss some information, although, interestingly, the information they do not find either refers to things that they can deduce from the schema (like, the time it will take to apply a technique, for example) or information that they should extract from their project context for comparison with a schema attribute (as is the case of the experience of the development team, where what they are really looking for are the defect types to be detected).
- *What do you and do not you like?* They like the fact that it improves selection as compared with books. They do not like the fact that they sometimes do not understand the meaning of the attributes or that there are uninstantiated attributes for a technique. Neither do they like having to handle so much information.
- *Has the way you see the selection problem changed?* The subjects view of the problem has changed in most cases. This is mainly due to the fact that the subjects had never had to select testing techniques for a software project before.
- *What have you learned?* Mainly, they have learnt what things are and what is not important for making the selection. They have also gained knowledge on testing techniques, as they have learnt about many new ones.
- *Would you do things differently next time?* Almost all the subjects say they would. Every time you do something, you find things that you have done wrong or simply could do better next time, ranging from how to read the documentation to how to fill in the forms to organise the selection.
- *Suitability of names and schema distribution.* As regards the suitability of the names, the names that they allege not to be very intuitive are precisely the ones that refer to non-intuitive concepts about the techniques (adequacy criterion, precision, etc.), which suggests that the schema names are suitable.

In view of the results, it can be deduced that the **subjects are satisfied with the characterisation schema, although, interestingly, satisfaction varies from group to group, as discussed in the following section.**

#### 10.4.7 Conclusions on Groups

As mentioned in Section 10.3, there was a reason for the four groups. Groups 1 and 3 were designed to study the learning effect, and groups 2 and 4 were designed to study the effect of using one method of selection before the other. This section gives a brief informal analysis of each of the response variables discussed above (which means that the conclusions reached are not relevant from the statistical viewpoint), considering the groups and the days on which the selection was made.

### 10.4.7.1 Analysis of the Learning Effect in Groups 1 and 3

In order to study the learning effect, that is, the effect that the second application of the same method of selection could have had on the response variables, the variation in the values of the response variables for each session will be analysed for groups 1 and 3 (the groups that used the same method of selection, schema and books, respectively, in both sessions). For this purpose, Table 10.48 shows the values of each response variable per group and session. The table reflects the mean values for the quantitative variables (times and number of) and the set of values that were used for the qualitative variables (sort of).

ASPECTS	Response variables	GROUP 1		GROUP 3	
		Session 1	Session 2	Session 1	Session 2
EFFICIENCY	Study time	107	55	306	167
	Selection time	201	110	262	146
	Doubts time	6	3	–	–
USABILITY	N. of problems	2	1	3	2
	Sort of problems	14	12	11	7
	N. times help consulted	3	1	–	–
	Attrib. consulted	25	13	–	–
COMPLETENESS	N. info used	13	13	7	7
	Sort info used	35	37	21	16
	N. info missed	3	3	3	1
	Sort info missed	19	20	15	10
EFFECTIVENESS	N. techniques considered	13	13	8	9
	N. selected techniques	3	2	4	3
	Selected techniques	9	8	15	17

Table 10.48: Values of each response variable per session for groups 1 and 3.

1. *Efficiency.* Looking at the first three rows of Table 10.48, it is found that the difference between the times used in both studying and selecting the testing techniques and in consulting doubts about the schema falls by half from session to session. This appears to indicate that practice in using the method of selection influences the time it takes to make the selection. However, even though the times fall for both methods in the second session, it is found that, as far as technique study time is concerned, the subjects with practice in selection using books still take longer than the subjects without practice in selection using the schema. It would be interesting to study the evolution of technique study time with a third or even fourth selection, as these data appear to show that technique study time is shorter using the characterisation schema, irrespective of the experience in selection.
2. *Usability.* Looking at the four rows of Table 10.48 related to schema usability, it is found that usability does not appear to be influenced by experience in using the method of selection. There are two exceptions: the sort of problems in the case of selection using books (group 3), and the number of attributes about which

help is sought in the case of selection using the schema (group 1). This appears to indicate that: (1) the type of problems that the subjects encountered when making the selection using the schema are not remedied by experience, unlike the problems encountered by the subjects making the selection using books and (2) that the doubts of the subjects making the selection using the schema concerning the attribute meanings appear to clear up in time.

3. *Completeness.* Looking at the four rows of Table 10.48 related to schema completeness, it is found that the subjects opinion of schema completeness does not vary in time. This is logical, as the structure of the characterisation schema was unchanged from one selection to another.
4. *Effectiveness.* Looking at the three rows of Table 10.48 related to schema effectiveness, it is also found that the results do not vary as the subjects gain experience in using the method of selection. This is especially interesting in the case of the subjects who used books for the selection, as it suggests that selection does not improve substantially with experience. However, it would again be interesting to observe what happens in later selections, as this result appears to indicate that the selections using books will still be worse than with the schema, irrespective of how experienced the subject is, and this result does not appear to be very logical.
5. *User satisfaction.* As regards the subjects who performed the schema+schema selection (group 1), on the whole, it can be said that these are the subjects who were least able to appreciate the characterisation schema. The reason for this could be that this is a group of subjects who had not faced the problem of making the selection beforehand and were directly given a tool to do this. As they have not had the experience of making the selection on the basis of what little information is available in books, they are unable to appreciate the tool they have been using.

When they were shown the schema at the end, the subjects who performed the books+books selection (group 3) were delighted with the possibility of being able to use a tool like this rather than having to pass through the ordeal they had to.

From all this, it can be concluded that **there appears to be a learning effect on some response variables, especially variables referring to the efficiency of the method of selection.**

#### 10.4.7.2 Analysis of the Effect of the Method in Groups 2 and 4

In order to study the effect of the method, that is, the effect that the second application of the same method of selection could have had on the response variables, the variation in the values of the response variables for each session will be analysed for groups 2 and 4 (the groups that used a different method of selection, schema/books and books/schema, in each of the two sessions). For this purpose, Table 10.49 shows the values of each response variable

per group and session. The table reflects the mean values for the quantitative variables (times and number of) and the set of values that were used for the qualitative variables (sort of).

ASPECTS	Response variables	GROUP 2		GROUP 4	
		Session 1	Session 2	Session 1	Session 2
EFFICIENCY	Study time	310	92	74	303
	Selection time	263	133	149	306
	Doubts time	–	6	2	–
USABILITY	N. of problems	3	2	1	3
	Sort of problems	12	13	14	9
	N. times help consulted	–	4	2	–
	Attrib. consulted	–	35	13	–
COMPLETENESS	N. info used	7	13	12	10
	Sort info used	25	40	37	24
	N. info missed	3	4	4	1
	Sort info missed	25	35	28	12
EFFECTIVENESS	N. techniques considered	8	13	13	8
	N. selected techniques	3	2	2	3
	Selected techniques	15	10	8	14

Table 10.49: Values of each response variable per session for groups 2 and 4.

1. *Efficiency.* Looking at the first three rows of Table 10.49, it is found that the experience in a method of selection does not appear to affect the use of the other, as the times are very similar in both sessions for the two groups. This is good for the experiment, as it would eliminate the possible threat to the validity posed by unconscious formalisation (advantage of the subjects who used the schema first over those who did not).
2. *Usability.* Looking at the four rows of Table 10.49 related to schema usability, it is found that having used one or other method first will affect two response variables: the type of problems found and the attributes whose meaning is consulted. As regards the type of problems encountered, it appears that the subjects who use the characterisation schema first encounter fewer problems. As regards attribute consultation, interestingly, the subjects who use books first, are interested in each and every schema attribute. The investigator has not found a meaningful explanation for these results, which means that they will have to be further investigated.
3. *Completeness.* Looking at the four rows of Table 10.49 related to schema completeness, it is found that the subjects opinion on schema completeness does not appear to be influenced by the method used in first or second place. This is again logical, as the structure of the characterisation schema was unchanged from one selection to another.

4. *Effectiveness.* Looking at the three rows of Table 10.49 related to schema effectiveness, it is also found that the effectiveness of the method of selection does not appear to be influenced by the experience in using one or other method.
5. *User satisfaction.* The subjects who completed the books+schema sequence (group 2) all underlined how easy it was to make the second selection, as many of the problems they had with the selection using books disappeared. They were mainly thankful for the fact that the techniques were easily identifiable, that the documentation was in their native tongue and that the descriptions of the techniques were useful and easily understandable.

The subjects who completed the schema+books selection sequence (group 4), interestingly, made a more *guided* selection than the other individuals. As they had already used the schema, they were familiar with the characteristics of interest of a technique and went directly to the information of interest in the books instead of wasting time on useless information. However, it is worth mentioning the ordeal these subjects went through to make the second selection.

From all this, it can be concluded that **the method of selection does not appear to be influenced by the order in which it has been applied**, at least in the huge majority of cases.

## 10.5 Conclusions on Experimental Evaluation

The research questions that appear in Table 10.1 and have not been directly addressed by the analysis of the data collected during the experiment will be answered in this section.

9. How can effectiveness be improved?

It has not been possible to evaluate characterisation schema effectiveness for the reasons mentioned in Section 10.4.5.3. However, it was found that the subjects who used the characterisation schema made more *accurate* selections than those who used books for selection purposes did. Accordingly, this question cannot be answered.

11. How many resources are required to use the schema?

At this point, the schema is instantiated on paper, which means that the only resources required for its use are someone to make the selection and the printed repository (or a computer equipped with a word processor) for consultation.

13. How could efficiency be improved?

As mentioned above, a possible solution for improving the efficiency of the characterisation schema would be to develop a tool to automate its use. The tool would make the information contained in the schema easier to handle.

14. How long does it take to learn to use the schema?

According to the experimental design, the subjects were given a week to study the characterisation schema, plus a two-hour class during which they were taught how to use the schema. The estimated time for each individual to learn the characterisation schema is two working days. This is believed to be more than sufficient to learn how to use the schema (assuming that the individual is familiar with software testing).

19. How can usability be improved?

It appears from the experiment that the values allocated to the schema attributes are too brief for their meaning to be understood. It appears that it would be better to associate a sentence or short paragraph with each attribute. However, it is not very clear whether this is a result of the students not having understood the meaning of the attribute itself rather than its value.

24. What is the right environment for schema use?

For the time being, it appears that the environment defined for the experiment (experimental parameters) is suited for the characterisation schema.

## 10.6 Characterisation Schema Improvement

According to the conclusions reached during the analysis of the data collected during the experiment, a series of improvements can be made to the characterisation schema:

- Present technique effectiveness as a range rather as a number.
- Develop a tool to improve information handling.
- Define a procedure or process that tells consumers how they have to use the schema.

It is relatively easy to make the improvement mentioned under the first point. The effectiveness attribute of the technique will be changed. It is more complicated to deal with the second point, which will be added to the list of work to be carried out in the future. As regards the third point, a possible solution is discussed below.

It was discovered that the consumers used whatever information they chose to make the selection instead of using the full set of information belonging to the context of the project on which they were working. It was decided to correct this by adding the comment that one of the experts made in Chapter 8 (and which was rejected at that time): divide the schema into compulsory and optional attributes. The compulsory attributes are the attributes for which a desired value must be provided when the selection is underway; the optional attributes are attributes for which it is not necessary to provide a desired value during selection.

On the basis of these definitions, the following procedure for schema use can be defined. This procedure is reflected in Figure 10.19 and it basically the same proposed in Section 5.6 apart from steps 1 and 2. It is composed of the following steps.

1. Identify the desired technique values for the attributes that belong to the *tactical* level and *object* element of the characterisation schema.
2. Identify whether there are other attributes that could impose any sort of restriction on the technique to be selected (for example, whether the project has to use a given testing technique, etc.).
3. Compare the desired values of the attributes with the values of each technique contained in the repository. Preselect the techniques whose values coincide with the specified ones.
4. If the set of preselected techniques is empty, relax one of the restrictions and return to step 3.
5. If the set of preselected techniques is not empty, study the values of the other attributes of the techniques it contains, paying special attention to the dependencies attribute.



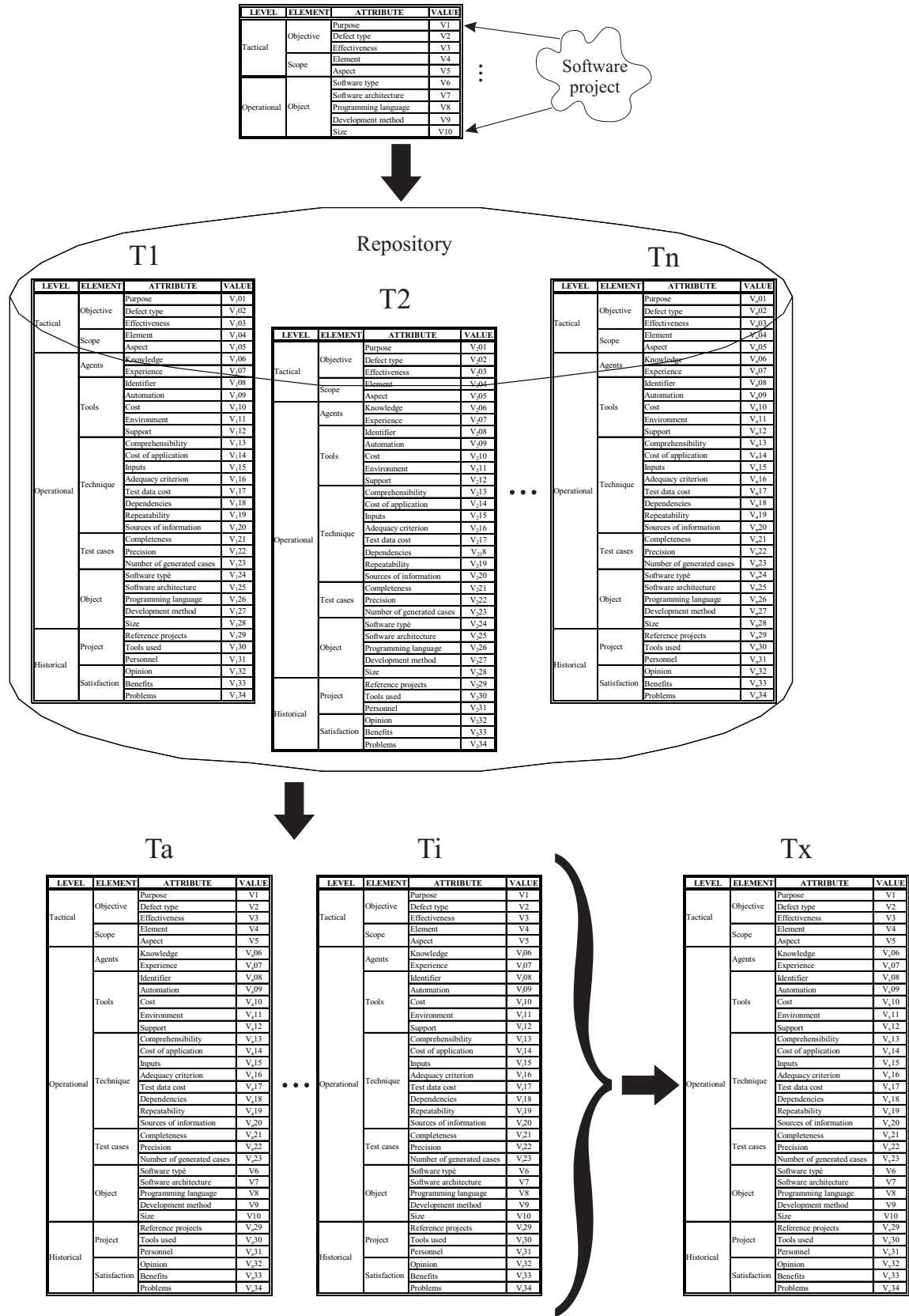


Figure 10.19: Use procedure for the characterization schema.



## Chapter 11

# Conclusions and Future Research Lines

### 11.1 Conclusions

Software testing and, particularly, the behaviour of testing techniques has been and still is a field of great interest to SE researchers. The interest in software testing is fully justified, as the final quality of which the delivered software system will depend on these tests. The interest in finding out how the different testing techniques behave is also completely justified mainly for two reasons: (1) the generation of test cases is one of the most important stages during software testing, and (2) there is a wide variety of techniques available that are not universally applicable.

The main problem which software developers encounter when selecting the best-suited testing technique or techniques for a software project is information. The information on testing techniques is, at best, distributed across different sources of information, and, at worst, non-existent. The information that can now be found about testing techniques refers to aspects concerning the mechanics of the techniques rather than to the costs and benefits of using the techniques. Two reasons are usually given to justify the non-existence of this sort of information: (1) it is difficult to compare the techniques, because their theoretical foundation is not solid enough; (2) it is difficult to determine which aspects should be studied. The approach taken in this research to the problem of gathering relevant information on software testing techniques is called a characterisation schema.

Chapter 2 discussed, on the one hand, the efforts within the software testing area to find out the costs and benefits associated with the use of different testing techniques and, on the other, the efforts within SE generally to characterise different software elements.

The efforts within the software testing area have the drawback of not being exhaustive with respect to either the set of existing testing techniques or the parameters of interest for a technique. Additionally, as these studies use different criteria to compare the same aspects of the techniques, the results of one study cannot be combined with those of another and, therefore, generalised. On other occasions, the criteria of comparison are not even relevant

for software developers and are only useful at the research level.

The main drawback of the efforts within the characterisation area is that none of the existing proposals are useful for characterising testing techniques in particular. Some proposals are too generic and, therefore, do not account for the specific aspects of software testing. Others, although they are specific enough, are not designed for selecting software tests, which means that they cannot be used for the problem addressed here either. Other secondary problems are related to the actual structure of the schema, which could be improved upon in some of the proposals.

This research proposes **using a characterisation schema to describe existing testing techniques**. Accordingly, a repository can be built that contains the description of each of the techniques of interest. The characterisation schema is proposed as a non-flat set of attributes, which will be grouped depending on the referenced test element and the level at which it is used. This set of characteristics is invariant for the techniques, although the extent of the instantiation will depend on the information available about each testing technique. The schema is based on the idea of it being possible to decide whether or not to use a technique without having to be acquainted with or having used the technique before. Accordingly, the schema can be used to fully describe the properties of any technique, so that a decision can be made on whether or not it is to be used without having to have procedural knowledge of the technique.

A combination of the deductive and inductive scientific methods was followed to search for the set of attributes that such a characterisation schema should contain. This approach to building the schema is distinct from the other characterisation schema proposals, which have always taken a purely deductive approach, based on the reasoning of the researchers.

The pure deductive method was not used in this research precisely because the testing technique area lacks a solid theoretical foundation at present. Accordingly, the steps taken to build the characterisation schema are:

1. *Generation of a deduced theoretical schema.* This schema is the fruit of the investigators perception of testing techniques and is based on her thoughts about existing testing techniques and the relevant characteristics that they have in common.
2. *Generation of an induced empirical schema.* This schema is the fruit of the search for opinions among software developers and researchers in the testing area. Sixteen respondents were interviewed and asked about the information required to characterise/select a testing technique. The analysis of the information gathered led to a second characterisation schema, which reached stability the tenth respondent.
3. *Abduction of a preliminary schema from the above two schemas.* A new schema was built, which was the result of the synthesis of these two schemas generated in the above two steps.
4. *Improvement of the preliminary schema by means of expert peer review.* Advice

was sought from experts in the testing area, who revised both the form and the substance of the preliminary schema. The results of the review were used to build an improved version of the preliminary schema.

Finally, the generated characterisation schema was evaluated. Two evaluations were carried out for the purposes of this test: an empirical evaluation, and an experimental evaluation. In the empirical evaluation, the investigator was able to check the feasibility and flexibility of the proposed characterisation schema. For this purpose, the characterisation schema was instantiated for a set of testing techniques of interest. This set has a series of properties, i.e. while accounting for sufficient variety, it also contains techniques that are very similar to each other, so as to be able to exhibit a range of schema characteristics. In the experimental evaluation, an experiment was run to compare completeness, effectiveness, efficiency, usability and, generally, user satisfaction with the schema and its behaviour against the use of other methods of selection, like books. The experiment was run with a group of eighty-seven final-year students at the School of Computer Science, Technical University of Madrid, who used both the schema and diverse documentation to select the testing techniques to be used in a given project. The experiment checked that, as was expected, the schema is more efficient, usable, and complete than the use of books for selection purposes. However, it was not possible to test characterisation schema effectiveness, as the current status of testing technique selection is more precarious than it was believed to be. Although this evaluation stopped short of testing effectiveness, it did indicate that a *finer tuned* selection can be achieved with the schema than with books.

## 11.2 Future Lines of Research

It is intended to continue the research discussed in this report and the following lines, in particular, appear to be promising:

- As it was not possible to evaluate the effectiveness of the characterisation schema in as much detail as expected, it was decided to run future experiments to study this aspect in more depth. Thus, for example, it would be interesting to study the effectiveness of the schema with subjects experienced in software testing and in the selection problem.
- The investigator has instantiated the characterisation schema for a series of techniques in this research. However, this instantiation could have been much more complete by adding many other techniques to the schema and looking at more sources of information to corroborate and/or complete the information gathered. Work is now underway to put together a repository that will contain information on all existing testing techniques. This information is being gathered from different sources: experiments run using different testing techniques, research papers on

different testing techniques and books that contain information on software testing, and will be posted on the web for the scientific community and developers.

- The schema is at present instantiated on paper, which means that its efficiency could be increased if there were a tool that acted as a repository and included the mechanisms required to enter, retrieve and update the information related to testing techniques. This work is now ongoing.
- This work focuses on a characterisation schema that is exclusive for testing techniques. It is intended to study how the schema could be generalised to cover any sort of technique or method proper to software engineering and, thus, increase the power of the schema proposed here.
- One of the contributions of this research is the problem-solving process used, which is not very commonplace in current research. It is intended to study whether this process could be standardised and be proposed as a method to be followed when building information repositories in SE. This would give people who want to build any type of characterisation schema a basis on which to found their work, enabling them to build better schemas.
- The characterisation schema proposed here, precisely because it is the first to be proposed for testing techniques, is understood not to be the best. Although this research has presented preliminary indications concerning its feasibility, effectiveness, etc., the work to be done in the near future includes putting the schema into practice within an organisation and using it for several case studies. This will make it possible to improve the contents of the schema and check what effect its use has within a real project environment.

Part IV

**BIBLIOGRAPHY AND  
APPENDIXES**





# Bibliography

- [ACRE Web page, ] <http://www.soi.city.ac.uk/~cc559/acre/welcome.html>.
- [Basili and Rombach, 1988a] V.R. Basili and H.D. Rombach. Towards a comprehensive framework for reuse: A reuse-enabling software evolution environment. Technical Report CS-TR-2158, Department of Computer Science, University of Maryland, College Park, MD 20742, December 1988.
- [Basili and Rombach, 1988b] V.R. Basili and H.D. Rombach. The TAME project: Towards improvement-oriented software environments. *IEEE Transactions on Software Engineering*, 14(6):758–773, June 1988.
- [Basili and Rombach, 1990] V.R. Basili and H.D. Rombach. Model-based reuse characterization schemes. Technical Report CS-TR-2446, Department of Computer Science, University of Maryland, College Park, MD 20742, April 1990.
- [Basili and Rombach, 1991] V.R. Basili and H.D. Rombach. Support for comprehensive reuse. *Software Engineering Journal*, pages 303–316, September 1991.
- [Basili and Selby, 1985] V.R. Basili and R.W. Selby. Comparing the effectiveness of software testing strategies. Technical Report TR-1501, Department of Computer Science. University of Maryland, College Park, May 1985.
- [Basili and Selby, 1987] V.R. Basili and R.W. Selby. Comparing the effectiveness of software testing strategies. *IEEE transactions on software engineering*, SE-13(12):1278–1296, December 1987.
- [Basili *et al.*, 1994] V.R. Basili, H.D. Rombach, and G. Caldiera. *The Experience Factory*, pages 469–476. Encyclopedia of Software Engineering - 2 Volume Set. John Wiley & Sons, Inc., 1994.
- [Basili, 1991] V.R. Basili. Software modeling and measurement: The goal/question/metric paradigm. Technical Report CS-TR-, Department of Computer Science, University of Maryland, College Park, MD 20742, December 1991.
- [Bass *et al.*, 1998] L. Bass, P. Clements, R. Kazman, and K. Bass. *Software Architecture in Practice*. SEI Series in Software Engineering. Addison-Wesley, January 1998.

- [Beizer, 1990] B. Beizer. *Software Testing Techniques*. International Thomson Computer Press, second edition, 1990.
- [Bertolino, 2001] A. Bertolino. Guide to the knowledge area of software testing. Software Engineering Body of Knowledge, February 2001.
- [Bible *et al.*, 1999] J. Bible, G. Rothermel, and D. Rosenblum. A comparative study of coarse- and fine-grained safe regression test selection. Technical Report 99-60-05, Computer Science Department, Oregon State University, 1999.
- [Bieman and Schultz, 1992] J.M. Bieman and J.L. Schultz. An empirical evaluation (and specification) of the all-du-pahs testing criterion. *Software Engineering Journal*, pages 43–51, January 1992.
- [Birk and Krschel, 1999a] A. Birk and F. Krschel. A knowledge management lifecycle for experience packages on software engineering technologies. IESE-Report 007.99/E, Fraunhofer IESE, Kaiserslautern, Germany, February 1999.
- [Birk and Krschel, 1999b] A. Birk and F. Krschel. A knowledge management lifecycle for experience packages on software engineering technologies. In *First Workshop on Learning Software Organizations*, Kaiserslautern, Germany, June 1999.
- [Birk, 1997a] A. Birk. Modelling the application domains of software engineering technologies. IESE-Report 014.97/E, Fraunhofer IESE, Kaiserslautern, Germany, August 1997.
- [Birk, 1997b] A. Birk. Modelling the application domains of software engineering technologies. In *Proceedings of the Twelfth International Conference Automated Software Engineering (ASE)*, Lake Tahoe, California, November 1997.
- [Blum, 1992] B.I. Blum. *Software Engineering: A Holistic View*. John Hopkings Applied Physics Laboratory Series in Science and Engineering. Oxford University Press, New York, May 1992.
- [Chen and Yu, 1994] T.Y. Chen and Y.T. Yu. On the relationships between partition and random testing. *IEEE Transactions on Software Engineering*, 20(12):977–980, December 1994.
- [Chen and Yu, 1996] T.Y. Chen and Y.T. Yu. On the expected number of failures detected by subdomain testing and random testing. *IEEE Transactions on Software Engineering*, 22(2):109–119, February 1996.
- [Clarke *et al.*, 1985] L.A. Clarke, A. Podgurski, D.J. Richardson, and S.J. Zeil. A comparison of data flow path selection criteria. In *Proceedings of the 8th International Conference on Software Engineering*, pages 244–251, London, UK, August 1985.
- [Clarke *et al.*, 1989] L.A. Clarke, A. Podgurski, D.J. Richardson, and S.J. Zeil. A formal evaluation of data flow path selection criteria. *IEEE Transactions on Software Engineering*, 15(11):1318–1332, November 1989.

- 
- [Davis, 1993] A.M. Davis. *Software Requirements: Objects, functions and states*. PTR Prentice Hall, 1993.
- [Descartes, 1637] R. Descartes. *Discours de la methode pour bien conduire sa raison; et chercher la vrit dans les sciences*. 1637.
- [Duran and Ntafos, 1984] J.W. Duran and S.C. Ntafos. An evaluation of random testing. *IEEE Transactions on Software Engineering*, SE-10(4):438–444, July 1984.
- [Elbaum *et al.*, 2000] S. Elbaum, A.G. Mailshevsky, and G. Rothermel. Prioritizing test cases for regression testing. In *Proceedings of the ACM SIGSOFT 2000 International Symposium on Software Testing and Analysis*, pages 102–112, Portland, Oregon, USA, August 2000. ACM.
- [Frankl and Deng, 2000] P.G. Frankl and Y. Deng. Comparison of delivered reliability of branch, data flow and operational testing. In *Proceedings of the ACM SIGSOFT 2000 International Symposium on Software Testing and Analysis*, pages 124–134, Portland, Oregon, USA, August 2000. ACM.
- [Frankl and Iakounenko, 1998] P.G. Frankl and O. Iakounenko. Further empirical studies of test effectiveness. In *Proceedings of the ACM SIGSOFT International Symposium on Foundations on Software Engineering*, pages 153–162, Lake Buena Vista, Florida, USA, November 1998. ACM.
- [Frankl and Weiss, 1991a] P.G. Frankl and S.N. Weiss. Comparison of all-uses and all-edges: Design, data, and analysis. Technical Report CS-91-03, Hunter College, Computer Science Department, March 1991.
- [Frankl and Weiss, 1991b] P.G. Frankl and S.N. Weiss. An experimental comparison of the effectiveness of the all-uses and all-edges adequacy criteria. In *Proceedings of the Symposium on Testing, Analysis and Verification*, pages 154–164, Victoria, BC, Canada, October 1991.
- [Frankl and Weiss, 1993] P.G. Frankl and S.N. Weiss. An experimental comparison of the effectiveness of branch testing and data flow testing. *IEEE Transactions on Software Engineering*, 19(8):774–787, August 1993.
- [Frankl and Weyuker, 1991] P.G. Frankl and E.J. Weyuker. Assessing the fault-detecting ability of testing methods. In *Proceedings of ACM SIGSOFT '91 Conference on Software for Critical Systems*, pages 77–91, New Orleans, LA, December 1991. ACM Press.
- [Frankl and Weyuker, 1993a] P.G. Frankl and E.J. Weyuker. An analytical comparison of the fault-detecting ability of data flow testing techniques. In *Proceedings of the 15th International Conference on Software Engineering*, pages 415–424, Baltimore, Maryland, May 1993. IEEE Computer Society Press/ACM Press.
-

- [Frankl and Weyuker, 1993b] P.G. Frankl and E.J. Weyuker. A formal analysis of the fault-detecting ability of testing methods. *IEEE Transactions on Software Engineering*, 19(3):202–213, March 1993.
- [Frankl *et al.*, 1994] P.G. Frankl, S.N. Weiss, and C. Hu. All-uses versus mutation: An experimental comparison of effectiveness. Technical Report PUCS-94-100, Polytechnic University, Computer Science Department, February 1994.
- [Frankl *et al.*, 1997] P.G. Frankl, S.N. Weiss, and C. Hu. All-uses vs mutation testing: An experimental comparison of effectiveness. *Journal of Systems and Software*, 38:235–253, September 1997.
- [Frankl *et al.*, 1998] P.G. Frankl, R.G. Hamlet, B. Littlewood, and L. Strigini. Evaluating testing methods by delivered reliability. *IEEE Transactions on Software Engineering*, 24(8):586–601, August 1998.
- [Garlan and Perry, 1995] D. Garlan and D. Perry. Introduction to the special issue on software architecture. *IEEE Transactions on Software Engineering*, 21(4), April 1995.
- [Garlan and Shaw, 1993] D. Garlan and M. Shaw. An introduction to software architecture. In V. Ambriola and G. Tortora, editors, *Advances in Software Engineering and Knowledge Engineering*, Series on Software Engineering and Knowledge Engineering, pages 1–39. Volume 2, World Scientific Publishing Company, Singapore, 1993. Also available as: Carnegie Mellon University Technical Report CMU-CS-94-166, January 1994. Reprinted in CMIS 460: Software Design and Development Faculty Course Guide, University of Maryland, Office of Instructional Development, Summer 1995.
- [Garlan *et al.*, 1995] D. Garlan, R. Allen, and J. Ockerbloom. Architectural mismatch: Why reuse is so hard. *IEEE Software*, pages 17–26, November 1995.
- [Glaser and Strauss, 1967] B.G. Glaser and A.L. Strauss. *The Discovery of Grounded Theory: Strategies for qualitative research*. Aldine de Gruyter, 1967.
- [Graves *et al.*, 1998] T.L. Graves, M.J. Harrold, J.M. Kim, A. Porter, and G. Rothermel. An empirical study of regression test selection techniques. In *Proceedings of the 1998 International Conference on Software Engineering*, pages 188–197, Kyoto, Japan, April 1998. IEEE Computer Society Press.
- [Hamlet and Taylor, 1988] D. Hamlet and R. Taylor. Partition testing does not inspire confidence. In *Proceedings 2nd Workshop on Software Testing, Verification and Analysis*, pages 206–215, Banff, Canada, July 1988.
- [Hamlet and Taylor, 1990] D. Hamlet and R. Taylor. Partition testing does not inspire confidence. *IEEE Transactions on Software Engineering*, 16(12):1402–1411, February 1990.

- [Hamlet *et al.*, 1997] P.G. Frankl D. Hamlet, B. Littlewood, and L. Strigini. Choosing a testing method to deliver reliability. In *Proceedings of the 19th International Conference on Software Engineering*, pages 68–78, Boston, Massachusetts, May 1997.
- [Hamlet, 1987] R. Hamlet. Probable correctness theory. *Info Proc Letters*, 25:17–25, April 1987.
- [Hamlet, 1988] R.G. Hamlet. Special section on software testing. *Communications of the ACM*, 31(6):662–667, 1988.
- [Hamlet, 1989] R. Hamlet. Theoretical comparison of testing methods. In *Proceedings of the ACM SIGSOFT '89 Third Symposium on Testing, Analysis and Verification*, pages 28–37, Key West, Florida, December 1989. ACM.
- [Harrold, 2000] M.J. Harrold. Testing: A roadmap. In *Proceedings of the 22nd International Conference on the Future of Software Engineering*, pages 63–72, Limerick, Ireland, May 2000. IEEE Computer Society/ACM Press.
- [Hempel, 1966] C.G. Hempel. *Philosophy of Natural Science*. Prentice-Hall, 1966.
- [Henninger *et al.*, 1995] S.R. Henninger, K. Lappala, and A. Raghavendran. An organizational learning approach to domain analysis. In *Proceedings of the 17th International Conference on Software Engineering. ICSE-17*, pages 95–104, Seattle, Washington, April 1995. ACM Press.
- [Henninger, 1995a] S.R. Henninger. Accelerating successful reuse through the domain lifecycle. In *Proceedings of the Seventh Workshop on Institutionalizing Software Reuse*, St. Charles, Illinois, August 1995. Andersen Consulting Center.
- [Henninger, 1995b] S.R. Henninger. Developing domain knowledge through the reuse of project experiences. In Mansur Samadzadeh and Mansour Zand, editors, *Proceedings of the Symposium on Software Reusability. SSR'95*, pages 186–195, Seattle, Washington, April 1995. ACM Press.
- [Henninger, 1996] S.R. Henninger. Accelerating the successful reuse of problem solving knowledge through the domain lifecycle. In *Proceedings of the Fourth International Conference on Software Reuse*, pages 124–133, Orlando, Florida, April 1996.
- [Henninger, 1997] S.R. Henninger. Tools supporting the creation and evolution of software development knowledge. In *Proceedings of the 12th International Conference on Automated Software Engineering*, pages 46–53, Lake Tahoe, NV, November 1997.
- [Howden, 1977] W.E. Howden. An evaluation of symbolic testing. Computer Science Technical Report 15, University of California, San Diego, APIS, (1977).
- [Howden, 1978] W.E. Howden. Theoretical and empirical studies of program testing. *IEEE Transactions on Software Engineering*, SE-4(4):293–298, July 1978.

- [Hutchins *et al.*, 1994] M. Hutchins, H. Foster, T. Goradia, and T. Ostrand. Experiments on the effectiveness of dataflow- and controlflow-based test adequacy criteria. In *Proceedings of the 16th International Conference on Software Engineering*, pages 191–200, Sorrento, Italy, May 1994. IEEE.
- [IEEE, 1983] *IEEE Standard Glossary of Software Engineering Terminology*. ANSI/IEEE Std, 1983. IEEE Computer Society.
- [ISO-9241, 1998] Ergonomic requirements for office work with visual display terminals. Technical Report ISO-9241-11, ISO, Geneva, 1998.
- [Jones and Prieto-Díaz, 1988] G. Jones and R. Prieto-Díaz. Building and managing software libraries. In *Proceedings of the 12th Annual International Computer Software and Applications Conference*, pages 228–236, Chicago, IL, October 1988. IEEE Computer Society Press.
- [Judd *et al.*, 1991] C.M. Judd, E.R. Smith, and L.H. Kidder. *Research Methods in Social Relations*. Harcourt Brace Jovanovich, sixth edition, 1991.
- [Juristo and Moreno, 2000] N. Juristo and A.M. Moreno. *Basics of Software Engineering Experimentation*. Kluwer Academic Publishers, 2000.
- [Juristo, 1998] N. Juristo. Editor’s introduction. *Knowledge Based Systems*, 11(2):77–85, October 1998.
- [Kamsties and Lott, 1995] E. Kamsties and C.M. Lott. An empirical evaluation of three defect-detection techniques. In *Proceedings of the Fifth European Software Engineering Conference*, Sitges, Spain, September 1995.
- [Kim *et al.*, 2000] J.M. Kim, A. Porter, and G. Rothermel. An empirical study of regression test application frequency. In *Proceedings of the 22nd International Conference on Software Engineering*, pages 126–135, Limerick, Ireland, May 2000. IEEE Computer Society Press.
- [Lauterbach and Randall, 1989] L. Lauterbach and W. Randall. Experimental evaluation of six test techniques. In *Proceedings of the 4th Annual Conference on Computer Assurance*, pages 36–40, Gaithersburg, Maryland, June 1989. National Institute of Standards & Technology, IEEE.
- [Maiden and Rugg, 1996] N.A.M. Maiden and G. Rugg. ACRE: Selecting methods for requirements acquisition. *Software Engineering Journal*, 11(3):183–192, 1996.
- [Myers, 1978] G.J. Myers. A controlled experiment in program testing and code walk-throughs/inspections. *Communications of the ACM*, 21(9):760–768, September 1978.
- [Myers, 1979] G. J. Myers. *The Art of Software Testing*. Wiley-Interscience, 1979.

- 
- [Naur and Randell, 1969] P. Naur and B. Randell, editors. *Software Engineering*, Report on a Conference Sponsored by the NATO Science Committee, 1968, Garmisch, Alemania, Bruselas, 1969.
- [Ntafos, 1981] S.C. Ntafos. On testing with required elements. In *COMPSAC'81. Proceedings of the 5th Annual International Computer Software and Applications Conference*, pages 132–139, Chicago, Illinois, November 1981.
- [Ntafos, 1984] S.C. Ntafos. An evaluation of required element testing strategies. In *Proceedings of the 7th International Conference on Software Engineering*, pages 250–256, Orlando, Florida, March 1984.
- [Ntafos, 1988] S. Ntafos. A comparison of some structural testing strategies. *IEEE transactions on software engineering*, 14(6):368–374, June 1988.
- [Ntafos, 1998] S.C. Ntafos. On random and partition testing. In *Proceedings of ACM SIGSOFT International Symposium on Software Testing and Analysis*, pages 42–48, Clearwater Beach, Florida, March 1998. ACM Press.
- [Offut and Lee, 1991] A.J. Offut and S.D. Lee. How strong is weak mutation? In *Proceedings of the Symposium on Testing, Analysis, and Verification*, pages 200–213, Victoria, BC, Canada, October 1991. ACM.
- [Offut and Lee, 1994] A.J. Offut and S.D. Lee. An empirical evaluation of weak mutation. *IEEE Transactions on Software Engineering*, 20(5):337–344, August 1994.
- [Offut *et al.*, 1993] A.J. Offut, G. Rothermel, and C. Zapf. An experimental evaluation of selective mutation. In *Proceedings of the 15th International Conference on Software Engineering*, pages 100–107, Baltimore, USA, May 1993. IEEE.
- [Offut *et al.*, 1996] A.J. Offut, A. Lee, G. Rothermel, R.H. Untch, and C. Zapf. An experimental determination of sufficient mutant operators. *ACM Transactions on Software Engineering and Methodology*, 5(2):99–118, April 1996.
- [Peirce, 1931–1958] C.S. Peirce. *Collected Papers of Charles Sander Peirce*. Eds P. Weiss and A. Burkes, Massachusetts (USA), 1931–1958.
- [Prieto-Díaz and Freeman, 1987] R. Prieto-Díaz and P. Freeman. Classifying software for reusability. *IEEE Software*, 4(1):6–16, January 1987.
- [Prieto-Díaz, 1985] R. Prieto-Díaz. *A software classification scheme*. PhD dissertation, Department of Information and Computer Science, University of California, Irvine, 1985.
- [Prieto-Díaz, 1989] R. Prieto-Díaz. *Software Reusability*, volume 1, chapter 4. Classification of Reusable Modules, pages 99–123. Addison-Wesley, 1989. T.J. Biggerstaff and A.J. Perlis editors.
-

- [Prieto-Díaz, 1990] R. Prieto-Díaz. Implementing faceted classification for software reuse (experience report). In *Proceedings of the 12th International Conference on Software Engineering*, pages 300–304, Nice, France, March 1990. IEEE Computer Society Press.
- [Prieto-Díaz, 1991] R. Prieto-Díaz. Implementing faceted classification for software reuse. *Communications of the ACM*, 34(5):89–97, May 1991.
- [PSS-05, 1991] Esa software engineering standards pss-05-0. Software Engineering Standards ISSN 0379 4059, European Space Agency (ESA), ESTEC, Noordwijk, The Netherlands, February 1991. ESA Board for Software Standardisation and Control (BSSC).
- [Rapps and Weyuker, 1982] S. Rapps and E.J. Weyuker. Data flow analysis techniques for test data selection. In *Proceedings of the 6th International Conference on Software Engineering*, pages 272–278, Tokio, Japan, September 1982.
- [Rapps and Weyuker, 1985] S. Rapps and E. Weyuker. Selecting software test data using data flow information. *IEEE Transactions on Software Engineering*, SE-11(4):367–375, April 1985.
- [Robillard, 1998] P.N. Robillard. The role of knowledge in software development. *Communications of the ACM*, 42(1):87–92, January 1998.
- [Rogers, 1983] G.F.C. Rogers. *The Nature of Engineering*. The Macmillan Press Ltd, 1983.
- [Rombach, 1992] H. D. Rombach. Systematic software technology transfer. In V.R. Basili, H.D. Rombach, and R.W. Selby, editors, *Experimental Software Engineering Issues. International Workshop*, Daugstuhl Castle, Germany, September 1992.
- [Rosenblum and Rothermel, 1997] D. Rosenblum and G. Rothermel. A comparative study of regression test selection techniques. In *Proceedings of the 2nd International Workshop on Empirical Studies of Software Maintenance*, pages 1–6, Bari, Italy, October 1997. ACM.
- [Rosenblum and Weyuker, 1996] D.S. Rosenblum and E.J. Weyuker. Predicting the cost-effectiveness of regression testing strategies. In *Proceedings of the ACM SIGSOFT'96 Fourth Symposium on the Foundations of Software Engineering*, pages 118–126, San Francisco, CA, October 1996. ACM.
- [Rosenblum and Weyuker, 1997a] D.S. Rosenblum and E.J. Weyuker. Lessons learned from a regression testing case study. *Empirical Software Engineering*, 2(3):188–191, March 1997.
- [Rosenblum and Weyuker, 1997b] D.S. Rosenblum and E.J. Weyuker. Using coverage information to predict the cost-effectiveness of regression testing strategies. *IEEE Transactions on Software Engineering*, 23(3):146–156, March 1997.
- [Rothermel and Harrold, 1994] G. Rothermel and M.J. Harrold. A framework for evaluating regression test selection techniques. In *Proceedings of the 16th International Conference on Software Engineering*, pages 201–210, Sorrento, Italy, May 1994. IEEE Computer Society/ACM Press.



- [Rothermel and Harrold, 1996] G. Rothermel and M.J. Harrold. Analyzing regression test selection techniques. *IEEE Transactions on Software Engineering*, 22(8):529–551, August 1996.
- [Rothermel and Harrold, 1997a] G. Rothermel and M.J. Harrold. Experience with regression test selection. *Empirical Software Engineering Journal*, 2(2):178–187, 1997.
- [Rothermel and Harrold, 1997b] G. Rothermel and M.J. Harrold. A safe, efficient regression test selection technique. *ACM Transactions on Software Engineering and Methodology*, 6(2):173–210, April 1997.
- [Rothermel and Harrold, 1998] G. Rothermel and M.J. Harrold. Empirical studies of a safe regression test selection technique. *IEEE Transactions on Software Engineering*, 24(6):401–419, June 1998.
- [Rothermel *et al.*, 1998] G. Rothermel, M.J. Harrold, J. Ostrin, and C. Hong. An empirical study of the effects of minimization on the fault detection capabilities of test suites. In *Proceedings of the International Conference on Software Maintenance*, pages 34–43, November 1998.
- [Rothermel *et al.*, 1999] G. Rothermel, R.H. Untch, C. Chu, and M.J. Harrold. Test case prioritization: An empirical study. In *Proceedings of the International Conference on Software Maintenance*, pages 179–188, September 1999.
- [Rowland and Zuyuan, 1989] J. Rowland and Y. Zuyuan. Experimental comparison of three system test strategies. preliminary report. In *International Symposium on Software Testing and Analysis*, pages 141–149, Key West, Florida, USA, December 1989. ACM.
- [Selby and Basili, 1984] R.W. Selby and V.R. Basili. Evaluating software engineering testing strategies. In *Proceedings of the 9th Annual Software Engineering Workshop*, pages 42–53, NASA/GSFC, Greenbelt, MD, November 1984.
- [Shaw and Clements, 1997] M. Shaw and P. Clements. A field guide to boxology: Preliminary classification of architectural styles for software systems. In *Proc. COMPSAC97, 21st Int’l Computer Software and Applications Conference*, pages 6–13, August 1997.
- [Shaw and Garlan, 1996] M. Shaw and D. Garlan. *Software Architecture: Perspectives on an Emerging Discipline*. Prentice-Hall, April 1996.
- [Shaw, 1995a] M. Shaw. Coping with heterogeneity in software architecture. In *Position paper for Dagstuhl Workshop on Software Architecture*, February 1995.
- [Shaw, 1995b] M. Shaw. Some patterns for software architectures. In John Vlissides, James Coplien, and Norman Kerth, editors, *In Pattern Languages of Program Design, Vol 2*, Second Annual Conference on Pattern Languages of Programming, pages 255–269. Addison-Wesley 1996, September 1995.

- [Shimeall and Leveson, 1988] T.J. Shimeall and N.G. Leveson. An empirical comparison of software fault tolerance and fault elimination. In *Proceedings 2nd Workshop on Software Testing, Verification and Analysis*, pages 180–187, Banff, Canada, July 1988.
- [Sommerville, 1992] I. Sommerville. *Software Engineering*. Addison-Wesley, 1992.
- [Vegas, 2001] S. Vegas. What information is relevant when selecting testing techniques? In Knowledge Systems Institute, editor, *Proceedings of the 13th International Conference on Software Engineering and Knowledge Engineering*, pages 45–52, Buenos Aires, Argentina, 2001.
- [Vokolos and Frankl, 1998] F. Vokolos and P.G. Frankl. Empirical evaluation of the textual differencing regression testing technique. In *Proceedings of the International Conference on Software Maintenance (ICSM-98)*, Bethesda, Maryland, November 1998. IEEE, Computer Society Press.
- [Wallace, 1988] D. Wallace. Software error analysis. Technical report, NIST, Gaithersburgh, MD 20742, December 1988.
- [Weiser *et al.*, 1985] M.D. Weiser, J.D. Gannon, and P.R. McMullin. Comparison of structural test coverage metrics. *IEEE Software*, 2(3):80–85, March 1985.
- [Weiss, 1990] S.N. Weiss. Methods of comparing test data adequacy criteria. In *COMPSAC'90. Proceedings of the 14th Annual International Computer Software and Applications Conference*, pages 1–6, Chicago, Illinois, October 1990.
- [Weyuker and Jeng, 1991] E.J. Weyuker and B. Jeng. Analyzing partition testing strategies. *IEEE Transactions on Software Engineering*, 17(7):703–711, July 1991.
- [Weyuker *et al.*, 1991] E.J. Weyuker, S.N. Weiss, and D. Hamlet. Comparison of program testing strategies. In ACM, editor, *Proceedings of the Symposium on Testing, Analysis, and Verification*, pages 1–10, Victoria, British Columbia, Canada., October 1991.
- [Weyuker, 1988] E. Weyuker. An empirical study of the complexity of data flow testing. In *Proceedings 2nd Workshop on Software Testing, Verification and Analysis*, pages 188–195, Banff, Canada, July 1988.
- [Weyuker, 1990] E.J. Weyuker. The cost of data flow testing: An empirical study. *IEEE Transactions on Software Engineering*, 16(2):121–128, February 1990.
- [Wong and Mathur, 1995] E. Wong and A.P. Mathur. Fault detection effectiveness of mutation and data flow testing. *Software Quality Journal*, 4:69–83, 1995.
- [Wong *et al.*, 1995] W.E. Wong, J.R. Horgan, S. London, and A.P. Mathur. Effect of test set minimization on fault detection effectiveness. In *Proceedings of the 17th International Conference on Software Engineering*, pages 41–50, Seattle, Washington, USA, April 1995. IEEE.

- [Wong *et al.*, 1998] W.E. Wong, J.R. Horgan, S. London, and H. Agrawal. A study of effective regression testing in practice. In *Proceedings of the 8th International Symposium on Software Reliability Engineering (ISSRE'97)*, pages 264–274, Bethesda, Maryland, November 1998. IEEE, Computer Society Press.
- [Wong *et al.*, 1999] W.E. Wong, J.R. Horgan, A.P. Mathur, and A. Pasquini. Test size minimization and fault detection effectiveness: A case study in a space application. *The Journal of Systems and Software*, 48(2):79–89, October 1999.
- [Wood *et al.*, 1997] M. Wood, M. Roper, A. Brooks, and J. Miller. Comparing and combining software defect detection techniques: A replicated empirical study. In *Proceedings of the 6th European Software Engineering Conference*, Zurich, Switzerland, September 1997.
- [Zeil, 1988] S.J. Zeil. Selectivity of data–flow and control–flow path criteria. In *Proceedings of the 2nd Workshop on Software Testing, Verification and Analysis*, pages 215–222, Banff, Canada, July 1988.
- [Zhu *et al.*, 1997] H. Zhu, P.A.V. Hall, and J.H.R. May. software unit test coverage and adequacy. *ACM Computing Surveys*, 29(4):366–427, December 1997.



# Appendix A

## Terminology

This appendix includes the terminology relevant to this piece of research.

**ANOVA.** Acronym that stands for ANalysis Of VAriance. Statistical technique that finds out the dependency degree between a response variable and one or more factors.

**Catalogue.** See repository.

**Characterisation schema.** Invariant and non flat set of attributes (grouped into elements and levels) which fully describes the nature of a testing techniques in order to allow its proper usage.

**Consumer.** Software developer who is responsible for selecting the testing techniques to apply in the software project.

**Expert.** Person with high experience in the software testing area. An expert can be a producer or a consumer.

**Producer.** Person who develops new software testing techniques.

**Repository.** Set of testing techniques described according to the contents of the characterisation schema. Also, set of instantiations of the characterisation schema.

**Respondent.** Subject (producer or consumer) that has been interviewed in order to know the relevant information regarding a testing technique.

**Test case.** Possible input to a software system.

**Testing technique.** Technique that allows the construction of sets of test cases.



## Appendix B

# Details on Software Testing Techniques

This appendix presents an overview of the most well-known software testing techniques, as well as the metrics that have been used in literature for its comparison.

### B.1 Software Testing Techniques

As mentioned earlier, software testing techniques aim at selecting test cases from the universe of possible inputs to a software system. When these test cases are run, they allow the identification of possible faults in the software. Nowadays, software testing techniques are grouped into *families*. There are seven families which have been the main target of software testing techniques research: functional, random testing, data-flow, control-flow, mutation, minimisation and regression. Next, each family will be briefly presented.

1. **Functional testing.** This family of testing techniques uses the program specifications to develop a set of test cases. The techniques in this family divide the universe of inputs to the system in subdomains, and obtain the test cases choosing one or more elements (depending on the concrete technique within the family) of each subdomain.
2. **Random testing.** This family of testing techniques chooses the inputs to the system (e.g. test cases) at random. Each technique in the family differs from the others in the heuristic to select the test cases. This means that the test cases are not really chosen at random, but there is any kind of knowledge or heuristic implicit that guides to the person who is applying the technique: for example, knowledge about typical faults in the software or about the operational profile of the system.
3. **Control-flow testing.** This family of testing techniques uses the structure of the control flow of the program to choose the test cases. Each technique in the family differs from the others in how exhaustively exercises the control structure of the

program. This gives rise to what are known as the different types of coverage. The most used coverage types are: sentence, decision, condition and path coverage.

4. **Data-flow testing.** This family of testing techniques looks at how data flows in the program to choose the test cases. They distinguish between definition and use of the variables of a program, and sometimes between different uses of the variables (in predicates or in expressions). Attending to how exhaustively they cover the definition and use of the variables in the program, they give rise to different types of coverage: all-uses, all-definitions, all-du-paths, etc.
5. **Mutation testing.** This family of testing techniques is based on the knowledge of the typical mistakes that people usually make when programming. These typical errors are modeled as mutation operators. The operators are applied to the program, creating this way multiple versions of the base (original) program (each version differs from the base program in one single sentence where the operator has been applied and that represents a fault in the program) named mutants. Test cases are generated in order too *kill* all mutants. A mutant is killed when the muted sentence is exercised.
6. **Regression testing.** Regression testing usually occurs whenever a fault is found and the program is modified to correct it. This may happen during testing or during maintenance. Regression testing implies that all test cases must be re-executed. However it would not be practical (sometimes even feasible) to execute all test cases, but only the ones related to the affected part of the code. Therefore, this family of techniques focuses on selecting test cases starting out of an existing set. The differences between the members of the family lies on the heuristic followed to select the test cases.
7. **Minimisation.** This family of techniques is usually used for regression purposes, but not always. The idea is to obtain a minimal set of test cases starting out of a set of test cases, so that the testing time is minimised.

## B.2 Metrics for Software Testing Techniques Comparison

Testing technique comparison has been done in research in different terms. This section presents the metrics used for comparison and used in Chapter 2.

- *Inclusion.* A technique T1 includes technique T2 (this relation has been defined for control-flow and data-flow testing techniques) if for each definition/use graph G, any set of complete paths of G that satisfy T1, also satisfy T2. This relation says that all test cases generated by T1 also satisfy T2. Two more definitions appear as a consequence of this one:



- Strict inclusion: T1 strictly includes T2 if for some graph G there is a complete set of paths in G that satisfy T2 and not T1.
  - Incomparability: T1 and T2 are incomparables if C1 does not include C2 and C2 does not include C1.
- *Selectivity.* Technique T1 is more selective than T2 regarding a testing goal if it never requires more, and sometimes requires less, test paths in order to achieve that goal. Examples of possible testing goals are: simple transference (once the sentence that contains the fault is executed, the fault is easily propagated to the end of the program), and constant/variable differencing (force a variable to take at least two different values to detect whether there are some values for which the program does not work).
  - *Probability of at least one fault.* Probability that the set of test cases generated by a technique discovers at least one fault in the program. There are two ways to calculate this probability: by means of a probability formula (this formula varies according to the testing technique) or generating a big number of sets of test cases for the program with the technique. In this case, the probability is given by the number of sets that show at least one fault in the program divided by the number of set of test cases generated.
  - *Narrow.* Technique T1 narrows T2 for program P with specification S if for each subdomain D belonging to the set of input subdomains of T2, there is a subdomain D' that belongs to the set of input subdomains of T1 such that D' is contained or is the same as D.
  - *Cover.* Technique T1 covers T2 for program P with specification S if for each subdomain D that belongs to the set of input subdomains of T2 there is a non empty set of subdomains that belong to the set of input subdomains of T1 such that the union of these subdomains is D.
  - *Appropriate cover.*
  - *Partition.* Technique T1 partitions T2 for program P with specification S if for every subdomain D belonging to the set of input subdomains of T2 there is a non empty collection of disjoint paired subdomains belonging to the set of input subdomains of T2 such that the union of these subdomains is D.
  - *Appropriate partition.*
  - *Number of faults.* Number of faults the set of test cases obtained with the technique shows. This number is usually given as a percentage on the total number of faults in the program, or on the number of faults in the program by fault type.

- *Fault rate.* Number of faults showed by the set of test cases generated by the technique per time unit. This aspect is usually measured as the amount of faults showed by hour, and this time usually includes both the time spent in applying the technique and the time spent in running the test cases generated by the technique. Sometimes this time also includes the time spent not only in seeing the failures, but also the time spent in finding the fault associated to the failure.
- *Size.* Number of test cases generated by the technique. Depending on the type of technique, the number of cases generated is given attending to different parameters. Therefore, for data flow techniques, it has been seen that it depends on the number of decisions the program has.
- *Coverage.* Percentage of the program that the set of test cases generated by the technique covers. There are different levels or degrees of coverage, being the most well-known: sentence coverage (measures the percentage of sentences of the program the test cases execute), decision coverage (measures the percentage of decisions of the program the test cases execute) and path coverage (percentage of paths the test cases execute).
- *Visibility.* This concept is applied to the faults of a program more than to the test cases generated by a technique, although it really has to do with the technique itself. It is defined as the number of times a set of test cases shows a fault.
- *Probability of faults not shown by the technique.* Sometimes it is not only important the number of faults a technique is able to show, but also the probability that there are faults in the program that have not been shown by the test cases. This is interesting mainly to know the maintenance costs of the program.
- *Cost of not detecting faults.* The fact that once testing has finished there are some faults remaining in the program, may have costs. For this reason, it is interesting to model in different ways the cost of having faults in the program not detected by the technique during testing.
- *Impact of groups use.* Due to the variability observed in the testing techniques (different subjects applying the same technique to the same program, found different faults), the effect provoked by groups has been studied. This way, the effect of having two subjects applying the same technique to the same program and the effect of having two subjects applying two different techniques to the same program have been studied. In both cases, the combination of subjects improves the amount of faults found.
- *Time.* This time might include different stages: time spent when generating test cases, time spent in executing the generated test cases, or CPU time during test case execution.

- *Fault type.* According to a given taxonomy, it can be interesting to observe whether some techniques are prone to show some faults or not. Different taxonomies have been used to study this aspect.
- *Software type.* It might happen that some testing techniques depend on the type of software that is being tested. This has been one of the parameters studied in testing techniques comparison. The type of software is measured according to specific taxonomies given by the authors of the study, since there is not a consensus about how programs can be classified.
- *Experience.* Other variable of interest when studying different testing techniques has been the influence of the experience of the people applying the techniques on the effectiveness of the technique.
- *Number of killed mutants.* In mutation testing, it has been measured the number of mutants killed by the set of test cases generated by a technique. This is usually done for two purposes: 1) compare a less powerful mutation technique (weak or selective) with a more powerful or 2) compare a non-mutation technique with a mutation technique. In both cases, two techniques are being compared in terms of one of them.
- *Number of mutants generated.* Para las tcnicas de mutacin es interesante conocer el nmero de mutantes que generan, ya que est en relacin directa con el nmero de casos de prueba que sern necesarios para probar el sistema. De algn modo, est en relacin directa con la mtrica ya estudiada anteriormente que hablaba del tamao de los casos de prueba.
- *Precision.* This term is usually used for regression testing. It refers to the number of test cases chosen with respect to the ones that ideally should have been chosen. The techniques known as *safe* have a precision of 1 (or 100%).
- *Memory.* In regression testing, it is defined as the degree in which test cases that make that the base program and the modified behave equally are chosen.
- *Reliability.*
- *Generality.* In regression testing, it refers to the ability of a technique of handling with real applications and languages.
- *Accountability.* In regression testing, this refers to the degree in which the technique can help to evaluate the adequacy of the set of test cases.



## Appendix C

# Related Work in the Testing Area

## C.1 Studies on the Relationships between Structural Testing Techniques

### C.1.1 Rapps and Weyuker

In [Rapps and Weyuker, 1982] and [Rapps and Weyuker, 1985], Rapps and Weyuker present several *data flow* testing techniques. They also perform a *theoretical study* in which they also include *control flow* techniques, defining the *inclusion* relation among techniques. This way, they establish a partial order of the studied techniques regarding this relation. The study they present *is not exhaustive regarding the families* in the study.

This is an interesting study, since the authors not only define several *data flow* testing techniques, but also intend giving an idea of their relative goodness and regarding *control flow* techniques. However it has two drawbacks: on the one hand the *inclusion* relation defined is not related to any other aspects that could be of interest, as the number of test cases generated or the effectiveness of the technique. On the other hand, it is limited, because it is *not exhaustive* regarding the set of techniques in each family.

### C.1.2 Weiser, Gannon and Mc Mullin

In [Weiser *et al.*, 1985], Weiser, Gannon and Mc Mullin perform a *theoretical study* that basically is an extension of the study performed by Rapps and Weyuker to other families of testing techniques. This way, the *inclusion* relation among techniques is studied. The families implied in this study are: *data flow*, *control flow*, *expressions* and *functions*. The study is *not exhaustive* for all families analysed.

As a result of the study, the authors discover that there are two families of techniques that cannot be compared: *data flow* and *expressions*. For this reason, they choose *a technique* from both families and perform an *empirical study* to observe whether a set of test cases that meets the criteria of one of them, also meets the criteria of the other; this is, whether the techniques *do not include* one each other because somehow they are "equal".

This is an interesting study, since it extends the study performed by Rapps and Weyuker

incorporating new families of testing techniques. However, the study is *not exhaustive* not even for the families they work with. Again, it is observed that the aspect being studied, that seems to be of interest for the researchers, is not useful for developers.

### C.1.3 Clarke, Podgurski, Richardson and Zeil

In [Clarke *et al.*, 1985] and [Clarke *et al.*, 1989], the authors present a *theoretical study* that is also a continuation of the study performed by Rapps and Weyuker. Again, a set of *data flow* and *control flow* testing techniques are examined, investigating the same *inclusion* relation defined by Rapps and Weyuker. Although they extend the *inclusion* relation to some other techniques of the *data flow* family, the study is still *not exhaustive* regarding the families it works with.

This is, like the previous ones, an interesting study, since it continues the idea of investigating the theoretical foundations of testing techniques. However, it has the same drawbacks the other studies have: the defined relation (it is the same that in the other studies) does not give a practical idea of the behavior of the techniques, and again the study is *not exhaustive* for the universe of testing techniques.

### C.1.4 Ntafos

In [Ntafos, 1988], Ntafos performs a *theoretical study* that follows the research line established by the previously described studies, again incorporating new techniques to the partial order suggested by the *inclusion* relation among the different techniques of the *data flow* and *control flow* families. This time, the study is *exhaustive* regarding the studies families.

Again, the research line followed is the same as in previous studies, which is important to better know the theoretical foundations of testing techniques. However, and although this time the whole family of *data flow* and *control flow* testing techniques are covered, there is the problem that the study cannot be extrapolated to other families of testing techniques, and also the results of using the *inclusion* relation are not practical for developers.

### C.1.5 Zeil

In [Zeil, 1988], Zeil performs a *theoretical study* in which he criticises the inclusion relation that has been used traditionally for the study of testing techniques. He states that although this relation provides information about how powerful the technique is, it does not provide any information about the cost of a technique and the type of defects it helps to find. For this reason, he defines and studies a new relation among *data flow* and *control flow* testing techniques named *selectivity*. This study is *exhaustive* to cover all techniques of both families, although it does not focus on any particular technique.

However, the study has some drawbacks: on the one hand, it has the recurrent problem (that appears again and again in all studies of testing techniques) that the study is *not exhaustive for the universe of testing techniques*. On the other hand, the author refers to *selectivity* regarding to different testing goals, although he never makes clear enough what

a testing goal should be. For this reason, the *selectivity* relation is not defined for practical effects.

### C.1.6 Frankl and Weyuker

In [Frankl and Weyuker, 1991], the authors perform a *theoretical* study in which they establish new relations *narrows*, *covers*, *properly covers*, *partitions* and *properly partitions* for *partition* testing techniques. This study aims at completing the lack of the previous studies of studying a significant aspect. In this case, the objective is to study the relation of the relationships presented with the effectiveness of the technique, defining effectiveness as the *probability of revealing at least one fault*. The type of study the authors perform is generic for the family, which means that it is *exhaustive* for the family.

### C.1.7 Frankl and Weyuker

In [Frankl and Weyuker, 1993b], the authors perform an *empirical study* in which they extend the one above, redefining effectiveness as the *expected number of revealed faults*. they also associated this aspect to the *properly covers* relation. This way, the authors study some *data flow* and *control flow* testing techniques, establishing a partial order among them. Of course, this study is *not exhaustive*.

This study focuses on such an aspect as the *expected number of faults revealed* by a technique, which is relevant for developers when deciding whether or not they should use a testing technique. However, the study is *not exhaustive*, even regarding the *data flow* and *control flow* families, and only one aspect is studied.

## C.2 Studies on Data Flow Testing Techniques

### C.2.1 Weyuker

In [Weyuker, 1988] and [Weyuker, 1990], Weyuker presents an *empirical study* to compare the cost of several *data flow* testing techniques, expressing it as the *number of test cases the technique generates*. She also compares the number of test cases generated empirically, with the theoretical limit associated to each technique. However, the study is *not exhaustive for the family of techniques*.

The study performed by Weyuker is one of the most important ones regarding useful information that can be obtained from a testing technique. This is mainly due to how interesting the studied aspect is. Although she *does not examine comprehensively all testing techniques of the family*, some interesting conclusions can be obtained for the few techniques studied. In this case, and since the only studied aspect is the *number of test cases generated*, it is useful, because this number is related to the testing time.

### C.2.2 Frankl and Weiss

Frankl and Weiss present in [Frankl and Weiss, 1991a], [Frankl and Weiss, 1991b] and [Frankl and Weiss, 1993] an *empirical study* in which they examine three techniques from three different families: *data flow*, *control flow* and *random*. This means that the study is *not exhaustive* for the families of the studied techniques. In the study, the authors investigate the effectiveness of each technique, defining it as *the probability that a set of test cases reveals at least a fault in a program*. The authors also study the relation *size/probability*, to check whether the *size* of the set of test cases has an influence in the *probability that a technique reveals at least one fault* (the authors want to know whether a technique is good just by itself or because it generates more test cases than other technique), and the *coverage/probability* relation, to observe whether or not the most effective techniques are the ones with more coverage. The study ends with a logistic regression where the *size* and the *coverage of the set of test cases* is taken into account to predict whether or not the set of test cases will reveal a fault.

Although the study is useful (it takes into account aspects of interest to get a good theoretical foundation for testing techniques), it has the same drawbacks that have been mentioned earlier. Again, the study is *not exhaustive for all families studied*. It can also be seen that the study focuses on aspects that are not very important for developers, as the *probability that a fault is revealed* which is a common aspect in studies about testing techniques. This shows a clear incommunication between developers and researchers.

### C.2.3 Bieman and Schultz

Bieman and Schultz perform in [Bieman and Schultz, 1992] an *empirical study* that focuses on studying the *number of test cases generated* by a technique from the *data flow* family. This study completes Weyuker's, since the technique they study is one of the techniques she has already studied. This way, this work is useful because it ratifies what others have already said, and can be used to generalise results (an important thing for the empirical studies). Obviously, the work is *not exhaustive even for the family of testing techniques*.

The interest of this work is mainly due to the fact that it ratifies the results of other previous studies. The weak points are that it only focuses on a technique, and therefore it is *not exhaustive*, and it focuses in one aspect when assessing the technique.

### C.2.4 Hutchins, Foster, Goradia and Ostrand

In [Hutchins *et al.*, 1994] the authors present an *empirical study* in which they compare two techniques, each from the *data flow* and *control flow* respectively. The objective of the study is to observe the effectiveness of each technique, measured as the *fault detection rate*, the *size* of the set of test cases generated, and the relations between *detection rate/coverage* and *size/coverage*. Again, the study is *not exhaustive for the families* under study.

This study is very similar to Frankl and Weiss's, although this one focuses on a more adequate aspect to measure the effectiveness of the techniques. Also, this study does not



investigate the same data flow technique as Frankl y Weiss’s study. For these two reasons, the results of these two studies are not comparable. This is something very common in testing technique studies. There is no consensus in the aspects to be studied and in the techniques to be studied. For this reason, sometimes the conclusions of the studies are not comparable. Finally, this study is *not exhaustive* and covers only one aspect.

### C.2.5 Frankl and Iakounenko

The authors present in [Frankl and Iakounenko, 1998] an *empirical study* very similar to that of Frankl and Weiss, presented in Section C.2.2. The authors compare three testing techniques from the *control flow*, *data flow* and *random* families. The objective of the study is to compare the effectiveness of the techniques, measured as the *ability to show faults*. The authors also study the relation *ability/coverage* for a technique, and for all techniques (the look at the *ability* of the technique given a coverage level). Again, it can be observed that the study is *not exhaustive for all techniques in the family*.

This study is very similar to the previous, and therefore, has the same drawbacks. Again, the study is limited because *it does not cover all techniques*, and because it deals with only a few aspects. It also focuses on different aspects from the previous studies, which implies that the studies are not comparable.

## C.3 Comparisons between Random and Partition Testing

### C.3.1 Duran and Ntafos

In [Duran and Ntafos, 1984], the authors perform several studies in which they investigate the effectiveness of *random* and *partition* testing techniques. For this purpose they perform:

- A *simulation* of the effectiveness of the *random* and *partition* testing technique. For this purpose, the authors simulate in each case the number of partitions, number of test cases and fault rate, and applying several probability formulas, they estimate the amount of times the *number of faults revealed* is higher for each technique, as well as the *probability* that the technique reveals at least one fault.
- An *empirical study* of the *effectiveness* of the *random* technique, when applying it to several programs. Each program has one only fault, and the effectiveness is interpreted in this case as the *number of times a set of test cases reveals a fault*.
- An *empirical study* of the *coverage* obtained by the *random* testing technique. For this purpose, they generate several sets of test cases and they measure the obtained *coverage* (segment coverage, branch coverage, path coverage and required pairs coverage).

As it can be seen, although the studies are *generic for partition techniques*, they are *not exhaustive for the universe of testing techniques*.

It is important to remark that this study throws some light on a fact that was not believed to be possible, and is that the *random* techniques are only slightly worse (and in some cases a little bit better) than *partition* techniques. This is an interesting result in the testing area, since a technique that takes so little time to be applied, as the *random* is, can be as effective as *partition* testing techniques. However, this study has the same drawbacks of the previously presented studies: *it cannot be generalised for all techniques*.

### C.3.2 Hamlet

Hamlet presents in [Hamlet, 1987] a *theoretical study* in which he completes the results obtained by Duran and Ntafos when comparing *partition* techniques with *random* technique. For this purpose, he studies the correctness of both techniques, measured as the *probability that there are faults not revealed by the test cases generated*.

Finally, they conclude with the identification of when the *random* technique behaves better than *partition* techniques, in what correctness is referred.

Hamlets *theoretical study* encompasses all *partition* testing techniques and *random* techniques. However, it cannot be made extensive to all *testing techniques*. The interesting thing from this study is that, starting out of the conclusions obtained in a previous study, and, apparently contradictory to what intuition says, it tries to reason out of these conclusions, their scope and justification. The drawbacks of the study is that the authors focus on an only aspect, and the results cannot be extrapolated to other testing techniques.

### C.3.3 Hamlet and Taylor

In [Hamlet and Taylor, 1988] and in [Hamlet and Taylor, 1990], Hamlet and Taylor continue studying the conditions under which the *random* technique is better than *partition* techniques. For this purpose, they perform a *theoretical study* in which they analyse the advantages of using a technique instead of the other. The aspects they study are: the *probability the technique reveals at least a fault*, and the *probability that there are not revealed faults* after applying the technique. Again, the study occurs at *family* level.

The authors conclusion is that *partition* techniques should be used to find faults and *random* techniques should be used to estimate software correctness.

### C.3.4 Weyuker and Jung

In [Weyuker and Jeng, 1991], the authors perform a *theoretical study* in which they try to guess in what cases or under which conditions the *random* testing technique is better than *partition* testing techniques. For this purpose, they perform a study of the effectiveness of both techniques, defined as the *probability the technique reveals at least one fault*. The study is *generic enough to cover the families from both techniques*.

The main drawbacks of this study are not new, and have appeared in other previous studies. On the one hand, the study *is not exhaustive for all testing techniques*, and on the

other hand, the aspect used for comparison does not seem to be very adequate, at least from the developer viewpoint, that does not want to *find at least one fault*, but all s/he is able to.

### C.3.5 Chen and Yu

Chen and Yu complete in [Chen and Yu, 1994] the study performed by Weyuker and Jeng. For this purpose, they perform a *theoretical study* in which they proof some of the conjectures revealed by Weyuker and Jeng. In this study, Chen and Yu re-examine *random* and *partition* testing techniques. The aspect the authors examine is the same Weyuker and Jeng did in their study, and is the *probability that the technique reveals at least one fault*.

Again, the study is *significant at family level*.

### C.3.6 Chen and Yu

In [Chen and Yu, 1996], the authors present an *empirical* study, in which they still try to know the *effectiveness* of *random* and *partition* testing techniques. In this case, effectiveness is redefined, since the authors have noticed about the problem of the aspect studied in the two studies above. This way, this time effectiveness is defined as the *expected number of faults the technique will reveal*. They also perform a more profound study to try to limit the conditions under which a technique is better than the other. As always in these studies, the study is *exhaustive regarding the families*.

Although this study has some of the drawbacks of other testing studies: *it does not cover all techniques* and the aspects it deals with are limited, in this case it can be seen that the performed study takes into account a relevant aspect to compare both techniques.

### C.3.7 Ntafos

In [Ntafos, 1998], the author presents a *simulation* in which he investigates the effectiveness of *random* and *partition* techniques. This time, the author considers the *monetary cost of not revealing faults*. To justify the study of this aspect, the author states that the aspects that have been used for comparison in previous studies were not relevant. Again, this study is *exhaustive regarding the families studied*.

This study, as always, does *not cover all techniques*, and although the aspect the author works with can be of interest for developers (*monetary cost*), is not enough, since testing technique selection usually depends on more than one aspect.

## C.4 Comparisons between Functional and Structural Testing Techniques

### C.4.1 Myers

Myers presents in [Myers, 1978], an *empirical study* in which he investigates the effectiveness, efficiency and human factors when using *functional* and *structural* testing techniques. The

subjects used in the study are not imposed an specific testing technique from the *functional* and *structural* families, but they can choose the one they prefer (they are allowed even to choose any sort of combination among the techniques of the family). The aspects considered in the study are *number of failures detected*, *time spent when applying the technique and running the test cases* and *variability among individuals and groups* when detecting faults.

This study *cannot be considered exhaustive regarding the testing techniques*, because we cannot know the techniques the individuals used.

#### C.4.2 Basili and Selby

Basili and Selby present in [Basili and Selby, 1985], [Selby and Basili, 1984] and [Basili and Selby, 1987] an *empirical study* in which they investigate the effectiveness and efficiency of a *functional* and a *control flow* testing techniques. They relate these aspects to other aspects from software development. The aspects the authors investigate in this study are: *the number of faults found*, the *fault detection rate*, the *type of faults found*, the *computer time* consumed by each technique, the *software type* for which a technique is more adequate, the *coverage* of the technique and the *experience* of the person who is applying the technique.

Since the study focuses on a technique per family, this study is *not exhaustive* for the families considered. However, the good thing of this study is that it uses several different aspects for technique comparison.

#### C.4.3 Kamsties and Lott

In [Kamsties and Lott, 1995] Kamsties and Lott present an *empirical study*. In this study, the authors replicate Basili and Selby's study; therefore, they investigate again the effectiveness and efficiency of a *functional* and a *control flow* testing techniques. The aspects they study are: the *number of faults detected*, the *fault detection rate*, the *type of faults found* and the *motivation and skills of the subjects who applied the technique*.

This study focuses again in a technique per family, therefore, it is *not exhaustive*.

#### C.4.4 Wood, Roper, Brooks and Miller

In [Wood *et al.*, 1997] the authors present an *empirical study* in which they again replicate Basili and Selby's one. In this study, the authors investigate the effectiveness and efficiency of a *functional* and a *control flow* technique. The aspects under study are: the *number of faults found*, the *fault detection rate*, the *type of software* and the influence of the *groups of people*. As in the rest of the studies of this family, the authors work with two techniques. For this reason, the study is *not exhaustive regarding the families* implied in the study.

## C.5 Studies on Mutation Testing Techniques

### C.5.1 Offut and Lee

In the *empirical study* presented in [Offut and Lee, 1991] and [Offut and Lee, 1994], the authors evaluate different *mutation* techniques: more precisely, they examine traditional mutation and four types of weak mutation. The objective of the study is observe the effectiveness, measured as the *percentage of mutants each technique kills*, and the cost, measured as the *number of test cases generated* and the *number of sentences executed* for each technique. The study is *not exhaustive for the family*, since it does not cover all types of mutation.

Again, the problems of this *empirical study* are the same as always: it does *not cover* all testing techniques, not even the ones corresponding to the family. In this case, the study focuses on the family, trying to know the differences among the different techniques in the family. Some of the studies aspects can be criticised as not adequate, since it is not demonstrated that the *percentage of mutants a technique kills* is related to its power to discover faults

### C.5.2 Offut, Rothermel and Zapf

In [Offut *et al.*, 1993] and [Offut *et al.*, 1996], the authors present an *empirical study* in which they investigate several *mutation* techniques (more precisely, mutation 2, 4 and 6-selective) as an alternative to standard mutation, since the number of mutants that standard mutation generates per program is very high. In this work, the authors examine the cost of the different techniques, measured as the *number of mutants generated* and the coverage, measured as the *number of mutants generated with standard mutation that kill the test cases generated with selective mutation*. The authors also *empirically* check some of the formulas for estimating the number of mutants a technique generates that have been proposed in literature. The study is *not exhaustive regarding the family*.

This study, as the one above, focuses on the analysis of the techniques in a family to compare the different options. It is interesting to note that the aspects studied here are not the same as in the previous studies, what reasserts the idea that the studies are not comparable. Again, it can be seen how the study is *not exhaustive for all techniques*.

### C.5.3 Frankl, Weiss and Hu

In [Frankl *et al.*, 1994] and [Frankl *et al.*, 1997] the authors present an *empirical study* in which they compare the use of *mutation* techniques to the use of *data flow* techniques. The study aims at examining the effectiveness of the techniques, or *probability that a set of test cases reveals a fault*, the coverage, *number of mutants killed* (for *data flow*) or *percentage of the executable duas covered* (for *control flow*, and their relations. The study is *not exhaustive* for the family.

From the previously examined studies, this is the only one that compares mutation to

other families of testing techniques. The metrics are also relevant to the aspects under study. However, it is possible to find the same problems as in the previous studies: the study does not cover the whole family, and only a few aspects are studied.

#### C.5.4 Wong and Mathur

Wong and Mathur present in [Wong and Mathur, 1995] an *empirical study* in which they investigate different *mutation techniques*, comparing them to *data flow* testing techniques. The study focuses on the *percentage of test cases that reveal at least one fault*.

The study does not cover all techniques in both families, which means that it is *not exhaustive*.

### C.6 Studies on Regression Testing Techniques

#### C.6.1 Rothermel and Harrold

Rothermel and Harrold present in [Rothermel and Harrold, 1997a], [Rothermel and Harrold, 1997b] and [Rothermel and Harrold, 1998] an *empirical study* in which they compare a selective *regression technique*: DejaVu to the traditional technique of *re-executing all test cases*. The objective of the study is compare the costs and benefits of the selective technique to those of the traditional technique. The costs are represented as the *time spent in selecting and executing the subset of test cases* on which regression testing is performed, and the benefits are represented as the *percentage of test cases selected by the technique*. Obviously, this study is *not exhaustive regarding the regression family*.

Although this study still lacks of the same coverage problems the other studies do, the aspects under study do not only make sense, but they are useful for developers.

#### C.6.2 Rosenblum and Rothermel

Extending the study presented in Section C.6.1, in [Rosenblum and Rothermel, 1997] and [Bible *et al.*, 1999] the authors present an *empirical study* in which they compare two selective *regression* techniques: DejaVu and TestTube, and a non-selective technique, which is *re-executing all test cases*. The study aims at comparing the costs and benefits of applying each technique. The costs are represented by the *time spent in analyse (decide the test cases to be chosen) and execute the set of test cases*, and the benefits are represented as the precision of the technique. The authors define the precision of a regression technique as the number of not necessary test cases that the technique selects from the original set (obviously, the higher the number of selected test cases is, the worse the precision will be), and in this study it is measured using the *reduction in the number of test cases of each set*<sup>1</sup>. Since the authors are only considering three techniques, the study is *not exhaustive for the family*.

---

<sup>1</sup>It is important to note that this simplification can be made because both techniques are "safe": this is, they guarantee that all test cases affected by a change are selected.

This study intends to be a continuation of the one above, extending it to other techniques. Although it is not exhaustive, it completes the one above.

### C.6.3 Graves, Harrold, Kim, Porter and Rothermel

In [Graves *et al.*, 1998] the authors present an *empirical study* about the costs, measured as *size of the set of test cases*, and benefits, measured as *percentage of fault reduction*, of five regression techniques. Obviously, the study is *not exhaustive for the family*.

Apart from the traditional coverage problems, it can be observed that costs and benefits have been measured differently from the previous studies (a new aspect, effectiveness, is introduced), which does not allow the comparison between this study and the previous ones.

### C.6.4 Vokolos and Frankl

In this *empirical study*, Vokolos and Frankl [Vokolos and Frankl, 1998] evaluate a safe regression technique named *textual differencing*. The objective of this study is analyse the effectiveness of the technique, measured as the *percentage of reduction of the set of test cases*, and efficiency, measured as the *time spent to get the reduction of the set of test cases*. The study is *not exhaustive for the family*.

### C.6.5 Wong, Horgan, London and Agrawal

In [Wong *et al.*, 1998] the authors present an *empirical study* in which they analyse the application of *minimisation* and *prioritisation* to the set of test cases obtained using the modifications based regression technique. For this purpose, the authors compare the results of applying only this technique, or applying it with prioritisation or minimisation (but not both things at the same time). The objective of the study consists of examining the cost and effectiveness of the different techniques. They measure the *reduction in the size of the set of test cases obtained*, the precision, defined as the *percentage of the test cases of a set in which the base program and the modified produce different outputs on the total of test cases in the set*, and memory, defined as the *percentage of test cases selected from those that need to be re-executed* (the original program and the modified program produce different output). The study is *not exhaustive*.

### C.6.6 Rothermel, Untch, Chu and Harrold

The authors present in [Rothermel *et al.*, 1999] an *empirical study* performed on nine *prioritisation* techniques. For this purpose, they distinguish between: no prioritisation, random prioritisation, optimal and six fine-grained more. When prioritising a set of test cases, it can be done regarding different objectives: find as soon as possible the most dangerous faults, the most expensive, etc. In this case, the authors choose as objective find as soon as possible all the faults the program contains. The objective of the study is analyse the effectiveness of the different techniques, measured as the *fault detection rate*. The study is *not exhaustive for the family*.

### C.6.7 Elbaum, Mailshevsky and Rothermel

In [Elbaum *et al.*, 2000], the authors present a completion of the *empirical study* presented in Section C.6.6. In this study, they compare 14 *regression techniques*, adding to the ones in the above mentioned study, 8 coarse-grained techniques. This experiment aims at studying the effectiveness, measured as *fault detection rate* of: 1) prioritisation based on version, 2) the coarse and fine grained techniques, and 3) the use of predictors of fault proneness. In this case the study *covers the whole family*.

### C.6.8 Kim, Porter and Rothermel

In [Kim *et al.*, 2000] the authors present an *empirical study* in which they compare different *regression* testing techniques: retest all, random, minimisation, DejaVu and Test Tube (the two last belong to the "safe" group). The objective of the study is analyse the effects of the frequency of testing in the costs and benefits of each technique. For this purpose, the authors study the size of the set of test cases, measured as the *percentage of reduction on the original set*, and costs are represented as the effectiveness of the techniques, measured as the absolute (number of total faults the program has) and relative (number of faults the original set of test cases detects) *number of faults found*,

### C.6.9 Rothermel and Harrold

In [Rothermel and Harrold, 1994] and in [Rothermel and Harrold, 1996], Rothermel and Harrold present a *theoretical study* in which they compare various *regression techniques*. The aspects the authors focus on are: *inclusion, precision, efficiency, generality* and *accountability*. The study is *not exhaustive for the regression testing techniques family*.

### C.6.10 Rosenblum and Weyuker

In [Rosenblum and Weyuker, 1996], [Rosenblum and Weyuker, 1997a] and [Rosenblum and Weyuker, 1997b] the authors present a *theoretical study*, completing the one above, in which they compare different *regression techniques*. The aspects the authors focus on are: *inclusion, precision, efficiency, generality* and *accountability*. The study is *exhaustive for the family of regression testing techniques*.

## C.7 Studies on Minimisation Testing Techniques

### C.7.1 Wong, Horgan, London and Mathur

In [Wong *et al.*, 1995] and [Wong *et al.*, 1999] the authors present an *empirical study* about the effects of *minimisation* on the set of test cases generated by a *data flow* testing technique. The study focuses on analysing the *number of test cases* generated and the *ratio of detected faults* for different *coverage* levels. The main objective of the study is check whether the *number of revealed faults* is related to the *size of the set of test cases* or to its *coverage*. The



question the authors raise is: what is the effect of reducing the size of a set of test cases in its effectiveness for different coverage levels? The objective is to understand the costs and benefits of the techniques from a practical viewpoint. This study is *not exhaustive*, since it does not cover the family.

### C.7.2 Rothermel, Harrold, Ostril and Hong

In [Rothermel *et al.*, 1998] the authors present an *empirical* study in which they analyse the effects of *minimising* a set of test cases generated with a *control flow* testing technique. The objective of the study is complete Wong *et al.*'s, checking whether their conclusions can be extended to other testing techniques. The authors analyse the costs and benefits of the *minimisation*. On the one hand, costs are represented according to the *reduction in the detection of faults* obtained when using minimisation. On the other hand, benefits are measured as the *percentage of reduction of the original test set*. The study is performed for different sizes of the original set. This study is *not exhaustive for the family*.

The aspects studied here are useful, but there is still the *exhaustiveness* problem, that does not allow generalising results.

## C.8 Other Studies

### C.8.1 Frankl, Hamlet, Littlewood and Strigini

In [Frankl *et al.*, 1998] and [Hamlet *et al.*, 1997], Frankl *et al.* present a *theoretical study* in which they analyse the *reliability* of two different testing techniques: *debug testing* and *operational profile*. The study is *exhaustive for both families*.

### C.8.2 Ntafos

In [Ntafos, 1981] and [Ntafos, 1984], Ntafos performs an *empirical study* in which he studies the effectiveness, measured as the *number of killed mutants* for three families of testing techniques: *required elements*, *control flow* and *random*. The study is *not exhaustive* for the families.

### C.8.3 Frankl and Deng

The authors present in [Frankl and Deng, 2000] an *empirical study* in which they compare three testing techniques that belong to the *control flow*, *data flow* and *operational* families. The study aims at observing the increment in the reliability of the program when eliminating the faults detected by the techniques, measuring reliability as the *probability that the program fails* when executing an input randomly chosen according to the operational profile. The study is *not exhaustive for the families studied*.

This study has the drawbacks other studies have regarding coverage and generalisation of the results obtained. In this case, this is mainly because this is an only study that does not follow any particular research line.

#### C.8.4 Lauterbach and Randall

In [Lauterbach and Randall, 1989] Lauterbach and Randall present an *empirical study* in which they compare different *control flow*, *functional* and *random* testing techniques regarding the: *number of faults revealed* and the *application effort* of the technique. They also study the condition *coverage* obtained by the studied techniques, as well as the influence of the *subjects* applying the techniques in their effectiveness.

## Appendix D

# Additional Information on the Empirical Schema

This appendix contains all information related to the information gathered from producers and consumers. Section D.1 presents the forms handed out to producers and consumers, Section D.2 presents the information supplied by producers and consumers, Section D.3 contains the data corresponding to the schema evolution study, and finally, Section D.4 presents the contributions of each respondent to the empirical schema.

### D.1 Forms Used

Figure D.1 and Figure D.2 show the forms used to obtain questions from consumers and information from producers.

### D.2 Answers from Respondents

Figure D.3, Figure D.4, Figure D.5, Figure D.6, Figure D.7, Figure D.8, Figure D.9, Figure D.9, Figure D.10, Figure D.11, Figure D.12, Figure D.13, Figure D.14 and Figure D.15 show the result of the interviews with consumers. Figure D.16, Figure D.17 and Figure D.18 show the result of the interviews with producers.

### D.3 Data on Respondents Answers

This section presents the data obtained from respondents' answers that have been used to perform the statistics about the evolution of the empirical schema.

Table D.1 shows the attributes of the schema proposed by each subject.

**CONSUMER FORM**

**I. PERSONAL DATA**

Name: \_\_\_\_\_

E-mail: \_\_\_\_\_

Position: \_\_\_\_\_

Years in current position : \_\_\_\_\_

Company/Institution: \_\_\_\_\_

Qualifications: \_\_\_\_\_

Experience in software development: \_\_\_\_\_

Experience in testing (brief paragraph): \_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

**II. SITUATION**

In this context, software testing is defined as dynamic evaluation of code, and a testing technique is that which helps to select test cases (inputs to the software system to be run).

Imagine that you are responsible for the testing of a certain software product (if it is easier for you, you may think in a concrete situation instead of thinking in abstract). Somebody offers you a set of testing techniques and tells you that you must choose from that set the ones you will use to test the software. Imagine that you do not know the techniques. This person tells you that s/he will not give you any kind of information regarding the procedure for using (applying) a technique, but apart from this, you may ask for any information you want about the technique.

**III. QUESTION**

What information would you like to know about these testing techniques? Answer to this question by writing the questions you would make to the person who is offering you the techniques.

Figure D.1: Consumer form.

**PRODUCER FORM**

**I. PERSONAL DATA**

Name: \_\_\_\_\_

E-mail: \_\_\_\_\_

Position: \_\_\_\_\_

Years in current position : \_\_\_\_\_

Company/Institution: \_\_\_\_\_

Qualifications: \_\_\_\_\_

Research interests in sw testing: \_\_\_\_\_

Experience in testing (brief paragraph): \_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

**II. SITUATION**

In this context, software testing is defined as dynamic evaluation of code, and a testing technique is that which helps to select test cases (inputs to the software system to be run).

As researcher in the software testing area and possible creator of new testing techniques, what information do you think is relevant when defining completely the nature of a software testing technique?

**III. ANSWER**

Figure D.2: Producer form.

## CONSUMER 1

### PERSONAL DATA

Position: Head of software development department  
 Years in current position : 2  
 Company/Institution: Abax S.L.  
 Qualifications: PhD in Computer Science  
 Experience in software development (years): 10  
 Experience in testing (brief paragraph): 6 years doing testing (real time software) and 4 years managing software testing activities

### QUESTIONS

1. Does it depend on the programming language of the code to be tested?
2. Are there tools to support it?
3. Are there users or projects of reference?
4. Is it bounded to a life-cycle or a methodology?
5. In what phase of the development process can it be used?
6. Do people have to be trained?
7. Is it experimental or has it been tested?
8. In what type of projects has it been used? (real-time, management, etc.)
9. What is its effectiveness? (number of errors found)
10. How long does it take test case generation?
11. Is it objective?
12. Is it enough using that technique or has to be complemented with other(s)?
13. Are people with experience or special knowledge required?
14. Does it generate a minimum set of test cases?
15. Can it be used n software components, or the whole system is needed?
16. What type of errors does it look for?
17. What type of people can use it?
18. Can it be used for testing hardware?
19. Does it require requirements/design/etc. in any special format?

Figure D.3: Results of the interview with the first consumer.

## CONSUMER 2

### PERSONAL DATA

Position: Director of the Software Engineering Laboratory  
 Years in current position : 7  
 Company/Institution: German Aerospace Center  
 Qualifications: Systems engineer and computer scientist  
 Experience in software development (years): 12  
 Experience in testing (brief paragraph): Test the software developed in the center, from the viewpoint of QA. Testing legacy software and OO software

### QUESTIONS

1. What is the knowledge people should have in order to use it?
2. What is the available documentation?
3. Has somebody used it?  
 Are people who have used it satisfied with it? Will they use it again?  
 Have tools been used with the technique? Which ones? How was the experience?  
 What have been the advantages of using the technique?  
 What has been the return on investment of the technique?
4. What type of software it can be used with?
5. For what life-cycle can it be used?
6. What is the estimate time when test cases generation?
7. What is the estimate time for training people who will use it?
8. What knowledge should people have in order to use it?
9. Does it have scientific basis?
10. What are the available tools?
11. What part of the techniques do available tools automates?
12. What is the acquisition and maintenance cost of the tools?
13. What hardware and operating system does the technique work in?
14. What programming language and dialect does the technique work with?
15. Does the technique have support? (hot-line).
16. What is its scalability? Can it be used for big and small systems?

Figure D.4: Results of the interview with the second consumer.

## CONSUMER 3

### PERSONAL DATA

Position: Project manager

Years in current position : 4

Company/Institution: Aerospace German Center

Qualifications: BSC in Computer Science

Experience in software development (years): 11

Experience in testing (brief paragraph): 7 years

### QUESTIONS

1. What are its results (objectives)?
2. What are the resources needed (machines, people, etc.)?
3. What is the base underlying?
4. What programming language can it be used with?
5. Is it adequate for the type of software that is being developed?

Figure D.5: Results of the interview with the third consumer.



## CONSUMER 4

### PERSONAL DATA

Position: Project manager

Years in current position : 1 and a half

Company/Institution: Fundaseo Bahiana de Cardiologia/Federal Univeristy of Bahia/Brazil

Qualifications: Master in Computer Science

Experience in software development (years): 7 years

Experience in testing (brief paragraph): A course on software testing. Took part in two testing processes - operational sytems- and followed another one-specialist system

### QUESTIONS

1. For what kind of software product is this technique more adequate?
2. What is possible to test with this technique?
3. What are the results this technique has got? (Is it effective?)
4. How much effort do I need to apply this technique?
5. Is it possible to do things in a simultaneous way?
6. How can it decrease the testing effort?

Figure D.6: Results of the interview with the fourth consumer.

## CONSUMER 5

### PERSONAL DATA

Position: Senior Software Engineer

Years in current position : 5

Company/Institution: General Dynamics

Qualifications: MS in Electrical Engineer

Experience in software development (years): 15

Experience in testing (brief paragraph): Software: 10 years (including 4 years producing consumer CDROM software). Hardware: 2 years in PCB manufacturing plant

### QUESTIONS

1. Does the technique test the requirements?
2. Does the technique create repeatable tests?
3. Does the technique test for logic problems?
4. Does the technique test for application errors?
5. Does the technique check for grammatical/syntax errors?
6. Do any techniques provoke other errors in the system?
7. Do any techniques isolate problems in the software?
8. Does the technique test the current system design and methodology?
9. Does it test for functionality?

Figure D.7: Results of the interview with the fifth consumer.

## CONSUMER 6

### PERSONAL DATA

Position: Technical Consultant I (Software Engineer)

Years in current position : 1 and a half

Company/Institution: Compaq

Qualifications: B.S. Physics (Working on PhD in Computer Science)

Experience in software development (years): 5

Experience in testing (brief paragraph): I have never been a member of a test team. As a developer I have written unit and sub-system level integration tests. I have also helped various project test team member define and their system and acceptance tests.

On the current project, I am working on we are implementing repeatable, automated unit and integration tests. Most of this is being done using Junit as a testing infrastructure. These tests also serve the purpose of validating the installation to the software once delivered (a kind of diagnostics tool). This testing has so far aminly dealt with the "back end" systems/models and has not been extend to the verification of the user interfaces (an area I am interested in as a member of the user interface team).

### QUESTIONS

1. What is the cost of buying the techniques? Cost to purchase (\$)
2. How much time does it take to produce a test case?  
How long does it take to implement and re-implement a test case?
3. Are the tests that are produced automate able (both the running and verification) or are the test cases aimed at a user running the tests and verifying the results?
4. What aspects of the software is the technique aimed at verifying? (User interfaces, database access, scalability, reliability, concurrency, etc.)
5. Are there tools (executables or documents) to guide the user in using the technique?
6. What kind or results have you seen when used on a project like mine?
7. What are the important factors for determining whether to use a technique and how to tailor it?
8. Will the methodology help pinpoint high risk areas of the project? --Areas that greater development resources should be devoted to because of either complexity, size, external interfaces, etc.
9. How early in the development cycle can a test case be constructed?
10. Will I be able to tell if a test fails due to an unimplemented feature or just that the test fails. Is there traceability from feature to test and if a test tests multiple features will the test report the results for each feature or all as one?
11. What are the inputs to the technique? Requirements, high level design, detailed design.
12. Does the technique assume a certain type of architecture or modelling paradigm? OO, client-server, distributed, monolithic application, etc.
13. Does the test methodology use the existing system GUIs to test the system or are "internal" test scripts (code) written?
14. Does the test methodology use black box, white box, or both methods? How well are the internals of the system verified or is only the interfaces to the system analyzed?
15. Is the methodology capable of generating test cases for GUIs? Just the GUI's, to make sure the user can not get into a bad state or input invalid data. Can these be automated?
16. What programming languages does the technique work for? What languages has it been used with in the past?

Figure D.8: Results of the interview with the sixth consumer.

## CONSUMER 7

### PERSONAL DATA

Position: Senior member of technical staff

Years in current position : 3

Company/Institution: General Dynamics, Electronic Systems

Qualifications: BE in Electrical Engineering, MS in Software Engineering

Experience in software development (years): 9

Experience in testing (brief paragraph): 3 systems/6 software

### QUESTIONS

1. What software development process can the technique be used with? (i.e. incremental, XP, spiral, waterfall, etc.)
2. What should be the format of the development documents produced?
3. What dependencies this product has on other products in order to allow for staging and integration testing?
4. How were the unit tests conducted during development? Were they?
5. How do the unit tests relate to the requirements?
6. Is there a "testing" framework integrated with the product?
7. Are the tests going to be part of the installation?
8. What Test Plans have been written?
9. How often are the tests to be run? Manual vs. Automated?

Figure D.9: Results of the interview with the seventh consumer.

## CONSUMER 8

### PERSONAL DATA

Position: Associated professor/ Visiting professor  
 Years in current position : 8/2  
 Company/Institution: Federal University of Brazil/ Visiting University of Maryland  
 Qualifications: PhD in Computer Science  
 Experience in software development (years): 21  
 Experience in testing (brief paragraph): Planning testing for information systems, databases, something with knowledge-based systems, control software, simulation and planning and telecommunications software.

### QUESTIONS

1. What type of software is the technique suitable for?
2. What development paradigm can it be used with?
3. What application domain can it be used with? (for risks identification).
4. What is the level of experience of the people with the technique?
5. What is the support (tool) the technique has?
6. Will people have to be trained using the tool?
7. What is the platform (software and hardware) the tool needs?
8. Does the use of the tool imply a cultural change?
9. What is the existing support for the tool?
10. Has the technique ever been used before?
11. Has the technique been evaluated or validated?
12. Is it cost-effective?
13. What is the coverage?
14. Who has used the technique? If nobody, why?
15. How straightforward is producing the test cases?
16. How easy is it to apply?
17. How easy is it to understand?

Figure D.10: Results of the interview with the eighth consumer.

## CONSUMER 9

### PERSONAL DATA

Position: Technical consultant II

Years in current position : 1

Company/Institution: Compaq Federal, LLC

Qualifications: Master in Computer Science

Experience in software development (years): 16

Experience in testing (brief paragraph): 1.5 years on formal test team, many years of unit testing

### QUESTIONS

1. Is this technique automatic or manual?
2. Is this technique available to be used by all levels of testing for this product? By all levels I mean Unit testing, sub-system testing, system testing and integration testing. Can this technique be used throughout the complete life-cycle of the product?
3. Does this technique provide white-box testing?
4. Does this technique provide black-box testing?
5. Does this technique provide regression testing?
6. Does this technique provide functional testing?
7. Does this technique test a user's interaction to this product? For instance, if this product has a user interface, does this technique provide the ability to simulate a user's actions with the user interface.
8. Does this technique produce any results, logs or statistics from the tests?
9. Does this technique help test the scalability and load balancing for the product?

Figure D.11: Results of the interview with the ninth consumer.

## CONSUMER 10

### PERSONAL DATA

Position: Scientist

Years in current position : 2

Company/Institution: Fraunhofer Center, Maryland

Qualifications: PhD in Computer Science

Experience in software development (years): 3

Experience in testing (brief paragraph): Very little

### QUESTIONS

1. What are the inputs required by the technique?
2. Does it test functional or non-functional requirements?
3. Is there tool support?
4. Does it allow automation?
5. How much do you have to change the development previous to testing to apply this technique?
6. How much does the adoption of the technique take?
7. Do testers need to have special knowledge to use it?
8. What kind of support exists for introducing the technique (training, etc)?
9. How does it compare to other similar techniques?

Figure D.12: Results of the interview with the tenth consumer.

## CONSUMER 11

### PERSONAL DATA

Position: Director of Software Engineering

Years in current position : 1

Company/Institution: Reliable Software Technologies

Qualifications: MS Computer Science

Experience in software development (years): 22

Experience in testing (brief paragraph): For 6 years I was an orbit flight-software integration lead on the Shuttle Mission Simulator (SMS) at the Johnson Space Center. This is essentially a flight-to-flight functional testing position. I was required to develop test plans for each orbit flight software lead and then oversee the execution of that plan.

### QUESTIONS

1. Does the technique require source code?
2. What is my interface with the technique?
3. What data does the technique generate?
4. What languages does the technique support?
5. What platform does the technique run on, and what are its resource requirements (memory, disk, cpu time...)?
6. Can the technique be used to perform automated testing?
7. Does the technique instrument the software under test? If yes, how?
8. If the technique instruments the software under test, what is the overhead for the instrumentation? (memory, cpu time...)

Figure D.13: Results of the interview with the eleventh consumer.



## CONSUMER 12

### PERSONAL DATA

Position: Senior Research Associate

Years in current position : 1

Company/Institution: Reliable Software Technologies

Qualifications: MS Computer Science

Experience in software development (years): 13

Experience in testing (brief paragraph): For 12 years I was involved in the development of real time flight simulations. Testing involved unit, integration and system level testing of a wide variety of software components in these highly complicated systems

### QUESTIONS

1. What types of testing are supported by each technique? (unit testing, integration testing, etc.)
2. Which of the following quality attributes are testable by the technique? Reliability, safety, reusability, maintainability, scalability.
3. What development paradigms are supported by each of the techniques? Procedural, object-oriented, functional, aspect-oriented, table-oriented.
4. What metrics are used to evaluate the quality attributes listed in question 2?
5. How would you classify the technique? (black-box, white-box, dynamic, static, etc)
6. Which are the development activities to which one can apply the technique?: requirements, design, code, maintenance

Figure D.14: Results of the interview with the twelfth consumer.

## CONSUMER 13

### PERSONAL DATA

Position: Senior Software Engineer

Years in current position : 1

Company/Institution: Reliable Software Technologies

Qualifications: MS in Computer Science

Experience in software development (years): 10 years

Experience in testing (brief paragraph): \_\_\_\_\_

### QUESTIONS

1. How easy is it to use? A developers need to know how easy or difficult it is to apply a given technique in terms of required skill level or complexity.
2. What is the cost of using it? Similar to above, but how much time/money/resources will be required?
3. What is the environment where it can be used? Testing techniques should list environments for which they are particularly well suited. environment factors like operating system, programming language, technology (COM/DCOM, CORBA, EJB, etc) should be included
4. How effective is this technique generally? What measurements are used to quantify effectiveness
5. When can the technique be applied?
6. What tools can be used to support the technique?
7. What assumptions are made for the technique to be effective?

Figure D.15: Results of the interview with the thirteenth consumer.

## PRODUCER 1

### PERSONAL DATA

Position: Director of the Software Engineering Laboratory

Years in current position : 4

Company/Institution: German Aerospace Center

Qualifications: Systems engineer and Computer Scientist

Research interests in sw testing: Take techniques already developed, study their applicability and transform and unify them in order to apply them to real projects and satisfy quality assurance restrictions

Experience in testing (brief paragraph): Test the software developed in the center, from the viewpoint of QA. Testing legacy software and OO software

### ANSWER

1. What are you trying to measure and how.
2. Which are the benefits the technique should produce (number of errors)
3. Feasibility, in the sense that it will not use too many resources.
4. That it can be used by people with medium knowledge.
5. Whether the technique has a scientific background.
6. If it can be automated, at least partially.
7. Return on the investment (cost and benefits) of the technique.
8. If it can be complemented with other techniques. Use results of ones as inputs to the following ones
9. Type of software the technique can be used with.
10. Knowledge of the people that have to use it.
11. Type of hardware the technique can be used with.
12. Type of company the technique can be used with.
13. Type of development method the technique can be used with.
14. Easiness for regression. Reexecute the test once the software has been changed.
15. Time that will be needed in order to apply the technique

Figure D.16: Results of the interview with the first producer.

## PRODUCER 2

### PERSONAL DATA

Position: Chief Scientist

Years in current position : Chief Scientist

Company/Institution: Reliable Software Technologies

Qualifications: PhD in Computer Science

Research interests in sw testing: \_\_\_\_\_

Experience in testing (brief paragraph): 10 years of experience in testing research

### ANSWER

1. Automation of the technique.
2. Costs versus return-on-investment for each technique.
3. Whether the technique requires access to source code
4. If they are static or dynamic.
5. Whether they have been designed to ensure that code is covered
6. Stopping criteria of the technique.
7. Whether you are duplicating effort when using several techniques

Figure D.17: Results of the interview with the second producer.

## PRODUCER 3

### PERSONAL DATA

Position: *Associated professor/ Visiting professor*

Years in current position : *8/2*

Company/Institution: *Federal University of Brazil/ Visiting University of Maryland*

Qualifications: *PhD in Computer Science*

Research interests in sw testing: *Integrated CASE tools, inspections and OO testing*

Experience in testing (brief paragraph): *Planning testing for information systems, databases, something with knowledge-based systems, control software, simulation and planning and telecommunications software*.

### ANSWER

1. Coverage (regarding the aspect being tested: sentences, models, scenarios, etc.).
2. Whether the technique is usable for mid-developers.
3. Inputs. Type of artefacts needed to apply it
4. Validation. Show that it works
5. Automation support with tools
6. Application cost, in terms of time

Figure D.18: Results of the interview with the third producer.

LEVEL	ELEMENT	ATTRIBUTE	C1	C2	P1	C3	C4	P2	C5	C6	C7	C8	P3	C9	C10	C11	C12	C13
Tactical	Objective	Quality attribute			X		X					X					X	
		Rigour										X						
	Scope	Phase	X							X				X			X	X
		Element	X				X		X		X			X			X	
		Aspect					X		X	X				X	X			
Operational	Agents	Experience	X			X						X						
		Knowledge	X	X	X	X						X	X	X				
	Tools	Identifier	X	X	X	X		X		X		X	X	X	X	X		X
		Automation		X														
		Cost		X														
		Environment		X								X				X		
		Support		X								X						
	Technique	Comprehensibility										X			X			
		Maturity level	X									X						
		Cost of application	X	X	X		X			X		X	X	X		X		X
		Inputs	X					X		X	X		X					
		Adequacy criterion						X		X				X			X	
		Test data cost	X	X						X					X			
		Dependencies	X		X													
		Repeatability	X						X									
		Sources of information		X						X					X			
		Coverage						X					X					
Use	Results	Effectiveness	X		X	X	X						X					X
		Type of defects	X						X									
	Object	Number of generated cases	X															
		Software type	X	X	X	X	X					X						
		Software architecture								X		X					X	X
		Programming language	X			X				X						X		X
		Development method	X	X	X						X							X
		Size		X														
	Project	Reference projects	X	X								X						
		Tools used		X														
	Satisfaction	Personnel		X														
		Opinion		X						X								
		Benefits		X						X								
		Problems		X						X								

Table D.1: Data on respondents answers.

Table D.2 shows how many attributes has incorporated each respondent to the schema.

RESPONDENT	EXISTING	NEWS	TOTAL
C1	0	18	18
C2	7	11	18
P1	7	1	8
C3	6	0	6
C4	5	1	6
P2	3	2	5
C5	4	0	4
C6	12	1	13
C7	3	0	3
C8	14	2	16
P3	5	0	5
C9	6	0	6
C10	7	0	7
C11	5	0	5
C12	5	0	5
C13	7	0	7

Table D.2: Number of attributes incorporated by each respondent to the schema.

Table D.3 shows the growth rate of the empirical schema, as well as the growth speed.

RESPONDENT	INITIAL SIZE	FINAL SIZE	GROWTH RATE	GROWTH SPEED	ACCUM SPEED
C1	0	18		50.00%	50.00%
C2	18	29	61.11%	30.56%	80.56%
P1	29	30	3.45%	2.78%	83.33%
C3	30	30	0.00%	0.00%	83.33%
C4	30	31	3.33%	2.78%	86.11%
P2	31	33	6.45%	5.56%	91.67%
C5	33	33	0.00%	0.00%	91.67%
C6	33	34	3.03%	2.78%	94.44%
C7	34	34	0.00%	0.00%	94.44%
C8	34	36	5.88%	5.56%	100.00%
P3	36	36	0.00%	0.00%	100.00%
C9	36	36	0.00%	0.00%	100.00%
C10	36	36	0.00%	0.00%	100.00%
C11	36	36	0.00%	0.00%	100.00%
C12	36	36	0.00%	0.00%	100.00%
C13	36	36	0.00%	0.00%	100.00%

Table D.3: Growth rate and speed rate of the empirical schema.

Table D.4 shows the number of votes obtained by each attribute in the schema.

Table D.5 shows the number of votes obtained by each element in the empirical schema.

LEVEL	ELEMENT	ATTRIBUTE	%VOTES	%RESPOND
Tactical	Objective	Quality attribute	3.46%	25.00%
		Rigour	0.39%	6.25%
	Scope	Phase	4.01%	31.25%
		Element	7.33%	37.50%
		Aspect	5.02%	31.25%
Operational	Agents	Experience	1.78%	18.75%
		Knowledge	5.05%	43.75%
	Tools	Identifier	9.97%	75.00%
		Automation	0.35%	6.25%
		Cost	0.35%	6.25%
		Environment	1.99%	18.75%
		Support	0.74%	12.50%
	Technique	Comprehensibility	1.28%	12.50%
		Maturity level	0.74%	12.50%
		Cost of application	6.78%	56.25%
		Inputs	8.60%	50.00%
		Adequacy criterion	4.02%	25.00%
		Test data cost	1.57%	25.00%
		Dependencies	3.27%	25.00%
		Repeatability	1.91%	12.50%
		Sources of information	1.72%	18.75%
	Results	Coverage	2.89%	18.75%
		Effectiveness	4.50%	37.50%
		Type of defects	1.91%	12.50%
		Number of generated cases	0.35%	6.25%
	Object	Software type	3.95%	37.50%
		Software architecture	3.01%	25.00%
		Programming language	4.01%	31.25%
		Development method	4.45%	31.25%
		Size	0.35%	6.25%
Use	Project	Reference projects	1.09%	18.75%
		Tools used	0.35%	6.25%
		Personnel	0.35%	6.25%
	Satisfaction	Opinion	0.83%	12.50%
		Benefits	0.83%	12.50%
		Problems	0.83%	12.50%

Table D.4: Importance of each attribute in the empirical schema.

Table D.6 shows the number of votes obtained by each level in the empirical schema.

## D.4 Information Supplied by Respondents

Figure D.19, Figure D.20, Figure D.21, Figure D.22, Figure D.23, Figure D.24, Figure D.25 and Figure D.26 show the information supplied by the respondents.

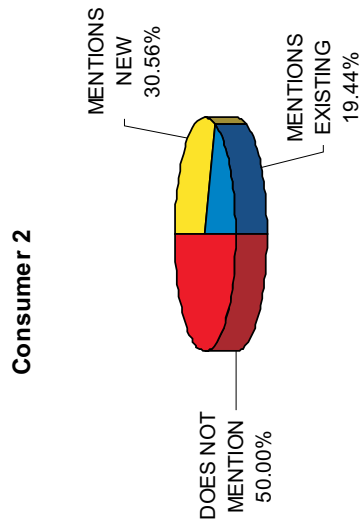
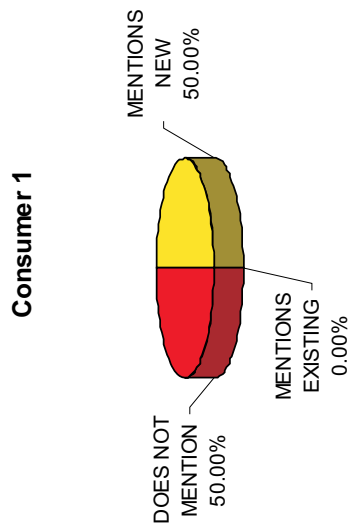


LEVEL	ELEMENT	%VOTES	%RESPOND
Tactical	Objective	3.85%	25.00%
	Scope	16.36%	56.25%
Operational	Agents	6.83%	43.75%
	Tools	13.39%	75.00%
	Technique	29.89%	93.75%
	Results	9.64%	56.25%
	Object	15.78%	68.75%
Use	Project	1.78%	18.75%
	Satisfaction	2.48%	12.50%

Table D.5: Importance of each element in the empirical schema.

LEVEL	%VOTES	%RESPOND
Tactical	20.21%	68.75%
Operational	75.52%	100.00%
Use	4.26%	25.00%

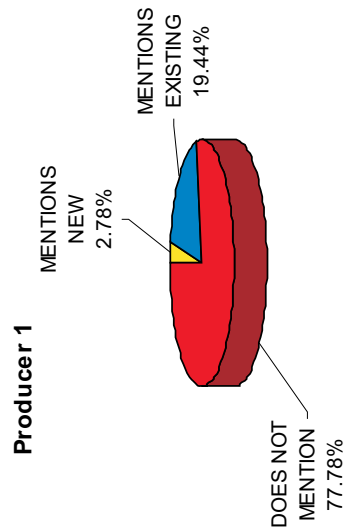
Table D.6: Importance of each level in the empirical schema.



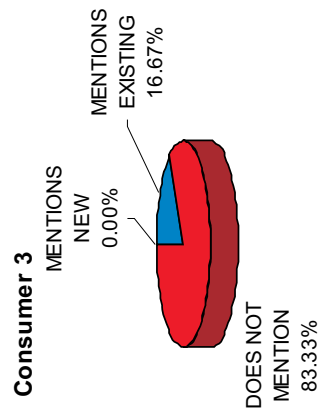
C1			
NEW		EXISTING	DOEST NOT MENTION
Phase	Dependencies		Quality attribute
Element	Repeatability		Rigour
Experience	Effectiveness		Aspect
Knowledge	Type of defects		Automation
Identifier	N of generated cases		Cost
Maturity level	Software type		Environment
Cost of application	Prog language		Support
Inputs	Development method		Comprehensibility
Test data cost	Reference projects		Adequacy criterion
			Problems

C2			
NEW		EXISTING	DOEST NOT MENTION
Automation	Problems	Knowledge	Quality attribute
Cost		Identifier	Rigour
Environment		Cost of application	Phase
Support		Test data cost	Element
Sources of info		Software type	Aspect
Tools used		Development method	Experience
Personnel		Size	Comprehensibility
Opinion		Reference projects	Maturity level
Benefits			Inputs
			Adequacy criterion
			Dependencies
			Repeatability
			Coverage
			Effectiveness
			Type of defects
			N of generated cases
			Sw architecture
			Prog language

Figure D.19: Information supplied by respondents(1/8).



P1			
NEW	EXISTING	DOEST NOT MENTION	
Quality attribute	Knowledge Identifier Cost of application Dependencies Effectiveness Software type Development method	Rigour Phase Element Aspect Experience Automation Cost Environment Support Comprehensibility	Prog language Size Reference projects Tools used Personnel Opinion Benefits Problems



C3			
NEW	EXISTING	DOEST NOT MENTION	
	Experience Knowledge Identifier Effectiveness Software type Prog language	Quality attribute Rigour Phase Element Aspect Automation Cost Environment Support Comprehensibility	N of generated cases Arq. del software Development method Size Reference projects Tools used Personnel Opinion Benefits Problems

Figure D.20: Information supplied by respondents (2/8).

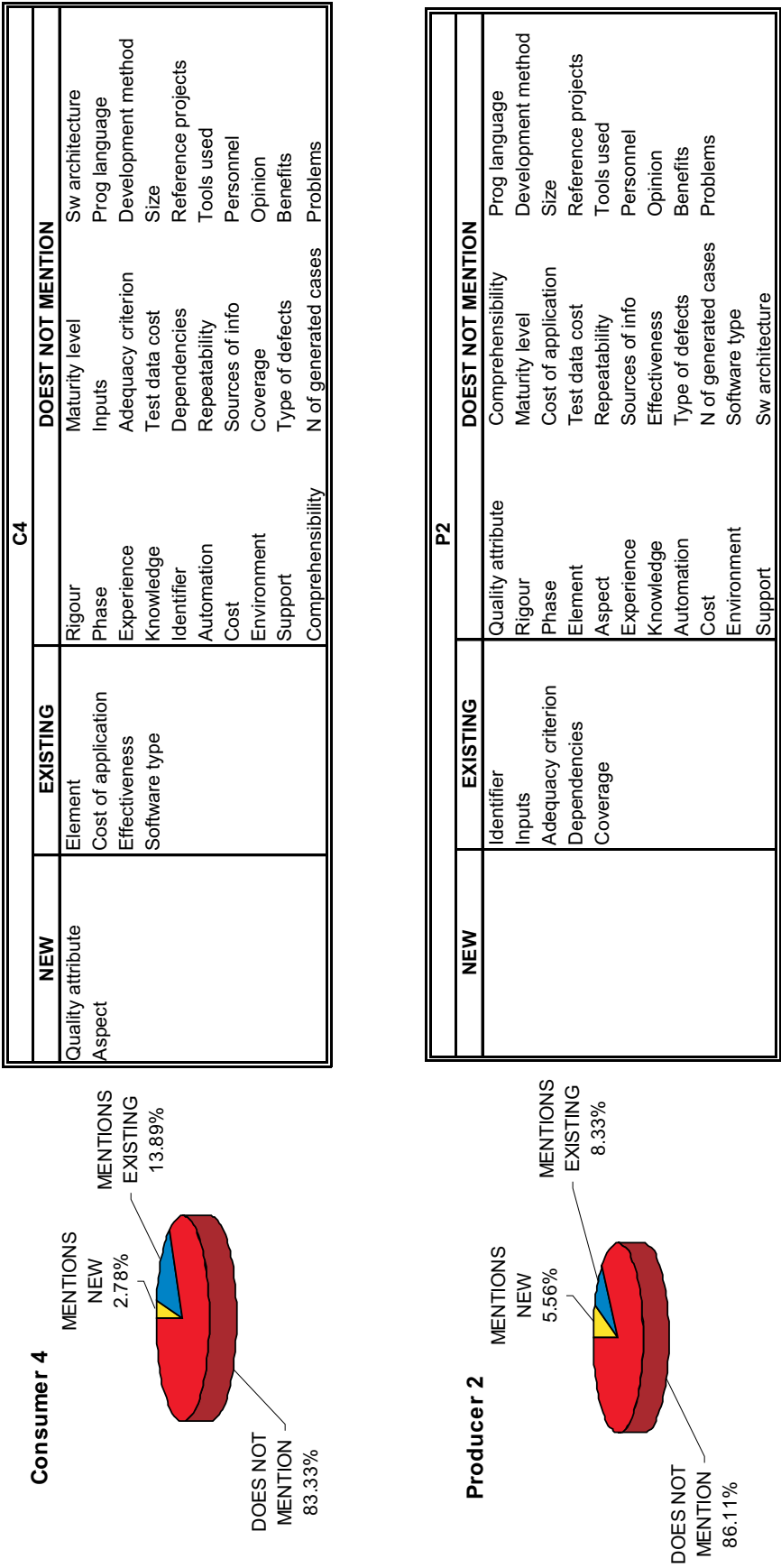


Figure D.21: Information supplied by respondents (3/8).

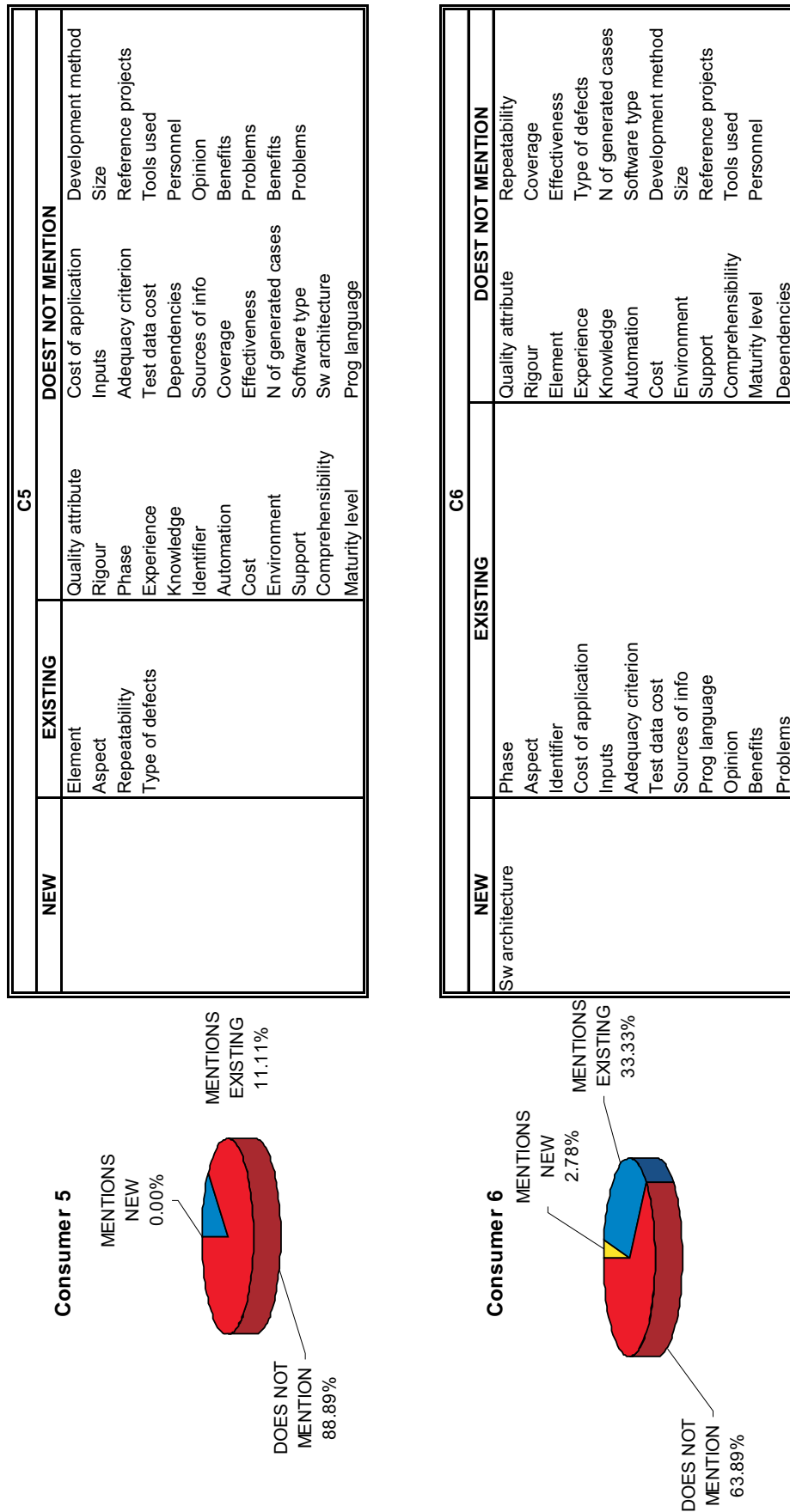
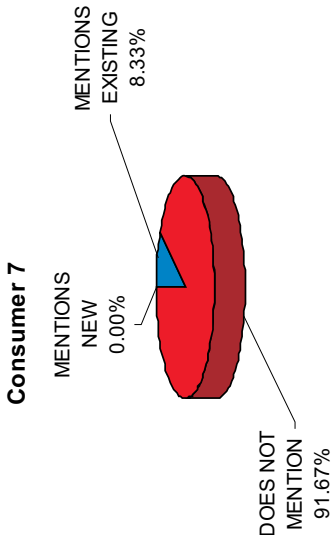


Figure D.22: Information supplied by respondents (4/8).

C7			
NEW	EXISTING	DOEST NOT MENTION	
	Element Inputs Development method	Quality attribute Rigour Phase Aspect Experience Knowledge Identifier Automation Cost Environment Support	Comprehensibility Maturity level Cost of application Adequacy criterion Test data cost Dependencies Repeatability Sources of info Coverage Effectiveness Type of defects N of generated cases Software type Sw architecture Prog language Size Reference projects Tools used Personnel Opinion Benefits Problems



C8			
NEW	EXISTING	DOEST NOT MENTION	
Rigour Comprehensibility Coverage	Quality attribute Experience Knowledge Identifier Environment Support Maturity level Cost of application Test data cost Effectiveness Software type	Sw architecture Reference projects	Phase Element Aspect Automation Cost Inputs Adequacy criterion Dependencies Repeatability Sources of info Type of defects N of generated cases Prog language Development method Size Tools used Personnel Opinion Benefits Problems

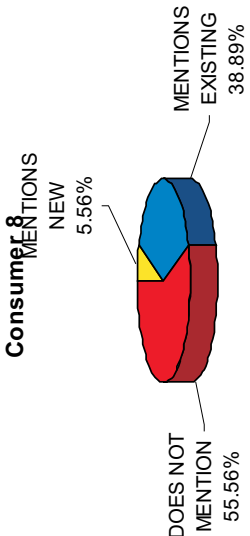
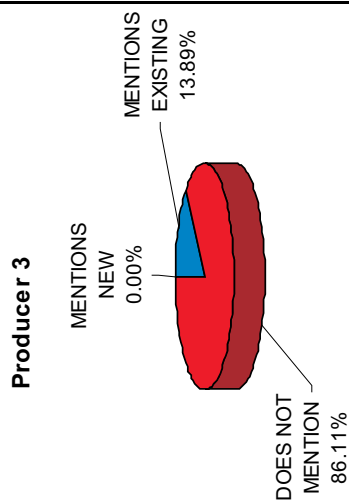


Figure D.23: Information supplied by respondents (5/8).

P3			
NEW	EXISTING	DOEST NOT MENTION	
	Knowledge Identifier Cost of application Inputs Coverage	Quality attribute Rigour Phase Element Aspect Experience Automation Cost Environment Support Comprehensibility	Prog language Development method Size Reference projects Tools used Personnel Opinion Benefits Problems



C9			
NEW	EXISTING	DOEST NOT MENTION	
	Phase Element Aspect Identifier Inputs Adequacy criterion	Quality attribute Rigour Experience Knowledge Automation Cost Environment Support Comprehensibility Maturity level Cost of application	Development method Size Reference projects Tools used Personnel Opinion Benefits Problems

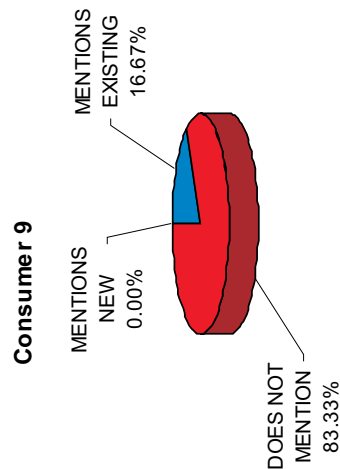
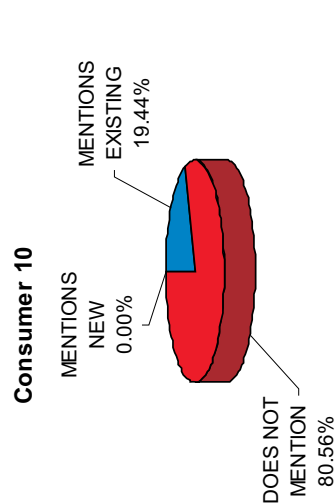
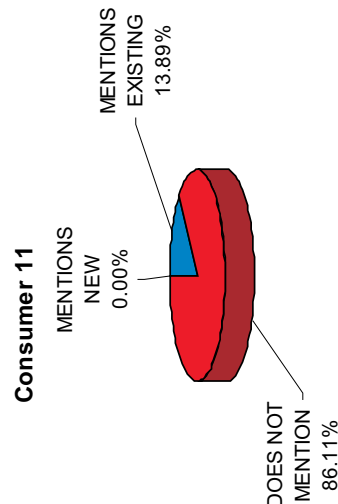


Figure D.24: Information supplied by respondents (6/8).



C10		
NEW	EXISTING	DOEST NOT MENTION
	Aspect Knowledge Identifier Comprehensibility Inputs Dependencies Sources of info	Quality attribute Rigour Phase Element Experience Automation Cost Environment Support Maturity level Cost of application Adequacy criterion Test data cost Repeatability Coverage Effectiveness Type of defects N of generated cases Software type Sw architecture Prog language Development method Size Reference projects Tools used Personnel Opinion Benefits Problems



C11		
NEW	EXISTING	DOEST NOT MENTION
	Identifier Environment Cost of application Inputs Prog language	Quality attribute Rigour Phase Element Aspect Experience Knowledge Automation Cost Support Comprehensibility Maturity level Adequacy criterion Test data cost Dependencies Repeatability Sources of info Coverage Effectiveness Type of defects N of generated cases Software type Sw architecture Development method Size Reference projects Tools used Personnel Opinion Benefits Problems

Figure D.25: Information supplied by respondents (7/8).



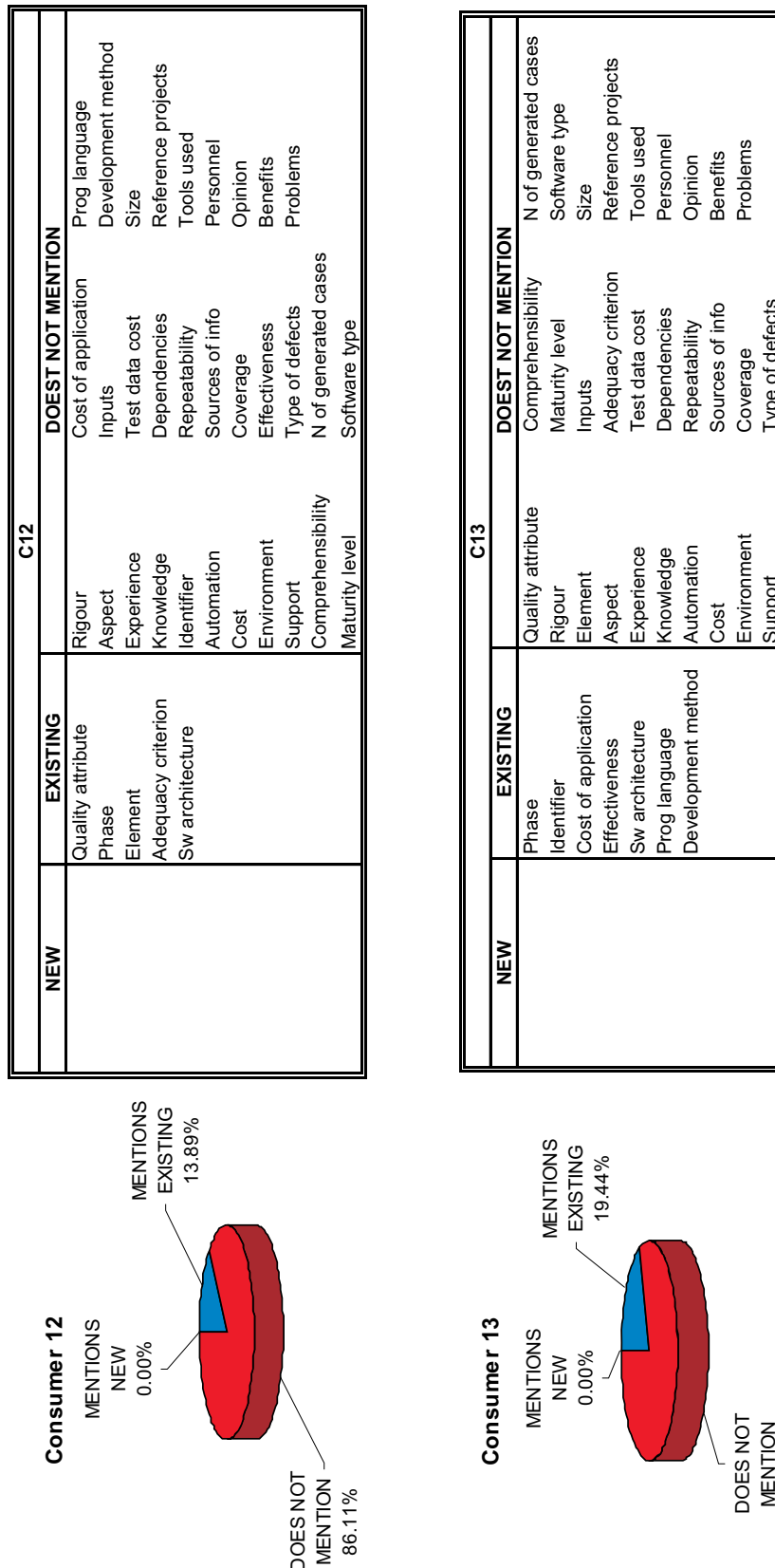


Figure D.26: Information supplied by respondents (8/8).



## Appendix E

# Additional Information on Expert Peer Review

This appendix contains information related to the expert peer review. Section E.1 shows the questionnaire sent to the experts, and Section E.2 shows the answers to the questionnaire given by the experts.

### E.1 Questionnaires used in Expert Peer Review

Figure E.1, Figure E.2, Figure E.3, Figure E.4 and Figure E.5 show the questionnaire used for the expert peer review.

### E.2 Answers Supplied by the Experts

Figure E.6, Figure E.7, Figure E.8 and Figure E.9 show the answers given by expert 1.

Figure E.10, Figure E.11 and Figure E.12 show the answers given by expert 2.

Figure E.13, Figure E.14 and Figure E.15 show the answers given by expert 3.

Figure E.16, Figure E.17 and Figure E.18 show the answers given by expert 4.

EXPERT FORM	
<b>I. PERSONAL DATA</b>	
Name:	_____
E-mail:	_____
Position:	_____
Company/Institution:	_____
Experience in software testing (brief paragraph):	_____
	_____
	_____
	_____
<b>II. GENERAL QUESTIONS</b>	
1. Do you think it would be useful for a company to have a repository containing information about software testing techniques in order to facilitate their selection?	
A. Yes	<input type="checkbox"/>
B. No	<input type="checkbox"/>
If not, why?	
_____	
_____	
_____	
_____	
2. Do you think that -in general- the proposed characterisation schema achieves this goal?	
A. Yes	<input type="checkbox"/>
B. No	<input type="checkbox"/>
If not, why?	
_____	
_____	
_____	
_____	
3. Do you think the proposed schema is easy to understand?	
A. Yes	<input type="checkbox"/>
B. No	<input type="checkbox"/>
If not, why?	
_____	
_____	
_____	
_____	
1	

Figure E.1: Questionnaire used for the expert peer review (1/5).

### III. REGARDING SCHEMA ATTRIBUTES

4. Do you think there are redundant attributes in the schema?

- A. \_\_\_\_\_ and \_\_\_\_\_ are the same  
 B. \_\_\_\_\_ and \_\_\_\_\_ are the same  
 C. \_\_\_\_\_ and \_\_\_\_\_ are the same  
 D. \_\_\_\_\_ and \_\_\_\_\_ are the same  
 E. \_\_\_\_\_ and \_\_\_\_\_ are the same

5. What three attributes do you think are the most and less relevant?

**Most relevant**

**Less relevant**

1. \_\_\_\_\_  
 2. \_\_\_\_\_  
 3. \_\_\_\_\_

1. \_\_\_\_\_  
 2. \_\_\_\_\_  
 3. \_\_\_\_\_

6. Would you delete attributes from the schema? Which one(s)? Why?

- A. Delete \_\_\_\_\_ from \_\_\_\_\_ because \_\_\_\_\_  
 B. Delete \_\_\_\_\_ from \_\_\_\_\_ because \_\_\_\_\_  
 C. Delete \_\_\_\_\_ from \_\_\_\_\_ because \_\_\_\_\_  
 D. Delete \_\_\_\_\_ from \_\_\_\_\_ because \_\_\_\_\_  
 E. Delete \_\_\_\_\_ from \_\_\_\_\_ because \_\_\_\_\_

7. Would you add attributes to any element in the schema? Which one(s)? Why?

- A. Add \_\_\_\_\_ to \_\_\_\_\_ because \_\_\_\_\_  
 B. Add \_\_\_\_\_ to \_\_\_\_\_ because \_\_\_\_\_  
 C. Add \_\_\_\_\_ to \_\_\_\_\_ because \_\_\_\_\_  
 D. Add \_\_\_\_\_ to \_\_\_\_\_ because \_\_\_\_\_  
 E. Add \_\_\_\_\_ to \_\_\_\_\_ because \_\_\_\_\_

8. Would you rename attributes in the schema? Which one(s)?

- A. Rename \_\_\_\_\_ as \_\_\_\_\_  
 B. Rename \_\_\_\_\_ as \_\_\_\_\_  
 C. Rename \_\_\_\_\_ as \_\_\_\_\_  
 D. Rename \_\_\_\_\_ as \_\_\_\_\_  
 E. Rename \_\_\_\_\_ as \_\_\_\_\_

Figure E.2: Questionnaire used for the expert peer review (2/5).

#### IV. REGARDING SCHEMA ELEMENTS

9. Do you think it makes sense grouping schema attributes in elements to which they refer?

A. Yes ☐

B. No ☐

If not, why?

---

---

---

---

10. Would you move attributes from one element to another?

A. Move \_\_\_\_\_ from \_\_\_\_\_ to \_\_\_\_\_

B. Move \_\_\_\_\_ from \_\_\_\_\_ to \_\_\_\_\_

C. Move \_\_\_\_\_ from \_\_\_\_\_ to \_\_\_\_\_

D. Move \_\_\_\_\_ from \_\_\_\_\_ to \_\_\_\_\_

E. Move \_\_\_\_\_ from \_\_\_\_\_ to \_\_\_\_\_

11. Would you delete elements from the schema? Which one(s)? Why?

A. Delete \_\_\_\_\_ because \_\_\_\_\_

B. Delete \_\_\_\_\_ because \_\_\_\_\_

C. Delete \_\_\_\_\_ because \_\_\_\_\_

D. Delete \_\_\_\_\_ because \_\_\_\_\_

E. Delete \_\_\_\_\_ because \_\_\_\_\_

12. Would you add elements to any level in the schema? Which one(s)? Why?

A. Add \_\_\_\_\_ because \_\_\_\_\_

B. Add \_\_\_\_\_ because \_\_\_\_\_

C. Add \_\_\_\_\_ because \_\_\_\_\_

D. Add \_\_\_\_\_ because \_\_\_\_\_

E. Add \_\_\_\_\_ because \_\_\_\_\_

13. Would you rename elements in the schema? Which one(s)?

A. Rename \_\_\_\_\_ as \_\_\_\_\_

B. Rename \_\_\_\_\_ as \_\_\_\_\_

C. Rename \_\_\_\_\_ as \_\_\_\_\_

D. Rename \_\_\_\_\_ as \_\_\_\_\_

E. Rename \_\_\_\_\_ as \_\_\_\_\_

Figure E.3: Questionnaire used for the expert peer review (3/5).

## V. REGARDING SCHEMA LEVELS

14. Do you think it makes sense grouping schema elements in levels?

A. Yes ☐

B. No ☐

If not, why?

---



---



---



---

15. Would you move elements from one level to another?

A. Move \_\_\_\_\_ from \_\_\_\_\_ to \_\_\_\_\_

B. Move \_\_\_\_\_ from \_\_\_\_\_ to \_\_\_\_\_

C. Move \_\_\_\_\_ from \_\_\_\_\_ to \_\_\_\_\_

D. Move \_\_\_\_\_ from \_\_\_\_\_ to \_\_\_\_\_

E. Move \_\_\_\_\_ from \_\_\_\_\_ to \_\_\_\_\_

16. Would you delete levels from the schema? Which one(s)? Why?

A. Delete \_\_\_\_\_ because \_\_\_\_\_

B. Delete \_\_\_\_\_ because \_\_\_\_\_

C. Delete \_\_\_\_\_ because \_\_\_\_\_

17. Would you add levels to the schema? Which one(s)? Why?

A. Add \_\_\_\_\_ because \_\_\_\_\_

B. Add \_\_\_\_\_ because \_\_\_\_\_

C. Add \_\_\_\_\_ because \_\_\_\_\_

D. Add \_\_\_\_\_ because \_\_\_\_\_

E. Add \_\_\_\_\_ because \_\_\_\_\_

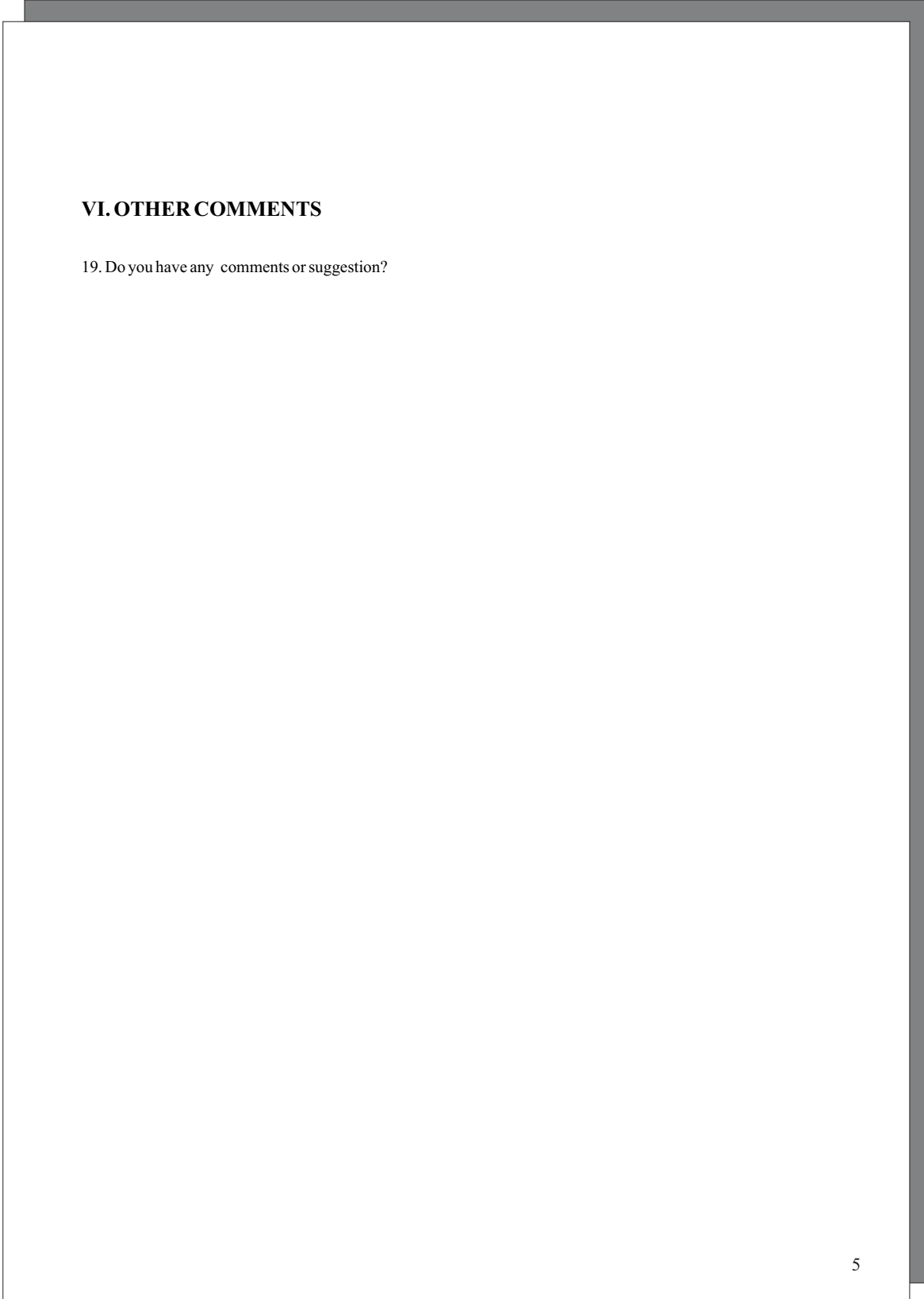
13. Would you rename levels in the schema? Which one(s)?

A. Rename \_\_\_\_\_ as \_\_\_\_\_

B. Rename \_\_\_\_\_ as \_\_\_\_\_

C. Rename \_\_\_\_\_ as \_\_\_\_\_

Figure E.4: Questionnaire used for the expert peer review (4/5).

The image shows a questionnaire form with a white background and a thin black border. At the top left, there is a bold section header. Below it, a question is listed. The rest of the page is left blank for input. In the bottom right corner, there is a small number.

**VI. OTHER COMMENTS**

19. Do you have any comments or suggestion?

5

Figure E.5: Questionnaire used for the expert peer review (5/5).



## EXPERT 1

### I. PERSONAL DATA

Position: Full Professor

Company/Institution: Instituto de Ciências Matemáticas de Computação (ICMC/USP)

Experience in software testing (brief paragraph): Expert

### II. GENERAL QUESTIONS

1. Do you think it would be useful for a company to have a repository containing information about software testing techniques in order to facilitate their selection?

A. Yes ☒

B. No ☐

2. Do you think that -in general- the proposed characterisation schema achieves this goal?

A. Yes ☒

B. No ☐

If not, why?

Yes, but it needs to be defined based on an accepted, common terminology. Maybe a glossary should be elaborated. This would avoid possible misunderstandings.

3. Do you think the proposed schema is easy to understand?

A. Yes ☒

B. No ☐

If not, why?

Yes, but it needs to be better characterized.

### III. REGARDING SCHEMA ATTRIBUTES

4. Do you think there are redundant attributes in the schema?

A. Maturity level and Repeatability are the same

5. What three attributes do you think are the most and less relevant?

#### Most relevant

1. Cost of application

2. Effectiveness

3. Type of defects

#### Less relevant

1. Tools used

2. Personnel

3.

Figure E.6: Answers given by expert 1 (1/4).

6. Would you delete attributes from the schema? Which one(s)? Why?

No

7. Would you add attributes to any element in the schema? Which one(s)? Why?

A. Add Support to other activities: debugging, maintenance, performance evaluation to Tools because testing information is relevant for debugging, maintenace and performance evaluation activities, as well as to other ones such as reliability estimation

B. Add Ease to integrate to Tools because a testing tool should provide mechanisms to facilitate its integration with other software development tools. Moreover, should also be easy to integrate to training and learning environments.

C. Add Complexity to Technique because this characteristic provides informaiton on how difficult is the application of the technique, e.e., it gives the worst case information

D. Add Lab package to Technique because this mechanism would provide facilities to replication of experiments and evaluation of the technique

E. Add Pilot study availability to Technique because this characteristic provides information about some aspects concerning the application of the technique and provides means for comparison with other techniques.

8. Would you rename attributes in the schema? Which one(s)?

A. Rename Sources of information as Sources of information and training material

#### IV. REGARDING SCHEMA ELEMENTS

9. Do you think it makes sense grouping schema attributes in elements to which they refer?

A. Yes ☒

B. No ☐

10. Would you move attributes from one element to another?

A. Move Cost of application from Technique to Results

11. Would you delete elements from the schema? Which one(s)? Why?

No

12. Would you add elements to any level in the schema? Which one(s)? Why?

A. Add Pilot project because this element would provide information about aspects of application of the technique, as well as means for comparison with other techniques

13. Would you rename elements in the schema? Which one(s)?

A. Rename Technique as Technique/criteria

B. Rename Results as Application results

Figure E.7: Answers given by expert 1 (2/4).

## V. REGARDING SCHEMA LEVELS

14. Do you think it makes sense grouping schema elements in levels?

A. Yes ☒

B. No ☐

15. Would you move elements from one level to another?

No.

16. Would you delete levels from the schema? Which one(s)? Why?

No.

17. Would you add levels to the schema? Which one(s)? Why?

No.

18. Would you rename levels in the schema? Which one(s)?

No.

## VI. OTHER COMMENTS

19. Do you have any comments or suggestion?

*To better understand our comments we provide our view of Software Testing techniques and criteria: "Software testing has as objective the identification of not-yet discovered errors. The success of the testing activity depends on the quality of a test set. There are a large number of criteria available to generate/evaluate a test set for a given program against a given specification. Testing criteria are classified into three techniques: functional, structural and error-based. In the functional technique, criteria and test requirements are established from the program specification; in the structural technique, the focus is on the details of a given implementation; in the error-based technique, criteria and test requirements are derived from the knowledge of typical errors which occur during the software development process. At the specification testing level, we have state-transition based testing, such as the W-Method. Care should be taken with the terminology used and the text should be coherent with the adopted one to avoid misunderstanding.*

*We use the term "testing techniques" to characterize the source of information used to derive the test requirements.. The term "testing criterion" defines a specific set of test requirements and exercises the program under different perspectives. Each technique comprises a set of testing criteria. Effectiveness, cost and strength are the three most meaningful bases for the comparison of testing criteria. Effectiveness is related to the fault detection ability of a criterion; cost indicates the effort to satisfy a criterion; and strength refers to the difficulty of a test case set T to satisfy a criterion C2 given that T already satisfies another criterion C1.*

*Based on the test requirements, a criterion may be used to generate test cases or to evaluate a given test case set. For instance, based on the test requirements, a tester is able to construct specific test cases for exercising the program under test, getting by construction an adequate test case set. Another relevant point concerns the adequacy analysis of the test cases with respect to the criteria. The goal is to exercise as extensively as possible the test requirements. The adequacy analysis provides a measure of how well the test requirements have been exercised. If the adequacy degree obtained is low, the tester may design additional test cases and require a new evaluation. This*

3

Figure E.8: Answers given by expert 1 (3/4).

process is repeated until an acceptable adequacy degree is attained. In this case, the testing criterion is being used as an adequacy criterion, providing a coverage measure. The adequacy of a test case set with respect to the All-Nodes criterion, for example, may be represented by a real value varying between 0 and 1. An adequacy of 1 represents that 100% of the test requirements for the criterion were met."

We understood the attributes Completeness and Correctness (Results) as part of the Adequacy degree. The underlying adequacy criterion provides the reference of completeness and correctness assessment.

We suggest replacing Execution Cost (Results) by Cost of application.

The "application domain" should be better characterized. The only characteristic that deals with this aspect is Software type (Object).

The description of some characteristics needs rewritten. For example:

\* Element: Software elements that can be tested using this technique: a function, a module, a subsystem, the whole system, the specification, etc.

\* Cost of application: an estimation of the effort that staff will need to apply the technique.

\* Support: Type of support the tool has. For example, hot line, documentation, training material.

Figure E.9: Answers given by expert 1 (4/4).

## EXPERT 2

### I. PERSONAL DATA

Position: System analyst

Company/Institution: NASA Goddard Space Flight Center/ Unisys

Experience in software testing (brief paragraph): practitioner - 10 years; researcher - 18 years (overlap)

### II. GENERAL QUESTIONS

1. Do you think it would be useful for a company to have a repository containing information about software testing techniques in order to facilitate their selection?

A. Yes ☒

B. No ☐

2. Do you think that -in general- the proposed characterisation schema achieves this goal?

A. Yes ☒

B. No ☐

3. Do you think the proposed schema is easy to understand?

A. Yes ☒

B. No ☐

If not, why?

Mostly, the schema is easy enough to understand after examining all of it. It took me two relatively quick passes to check for relationships and to determine where items I looked for are positioned. I think the tactical level is the least well-defined because at that level the consumer should have a good idea WHY that test should be used. For example, I would like the purpose of the test technique. This information is somewhat covered under operational: inputs, dependencies and under Results, type of defects found, but that's too many items for a manager to examine, too far down. The manager should know up front that test technique X test algorithms or tests for some other item. Keep the Technique description, but add a characteristic to Scope, perhaps called Purpose.

### III. REGARDING SCHEMA ATTRIBUTES

4. Do you think there are redundant attributes in the schema?

No.

Figure E.10: Answers given by expert 2 (1/3).

5. What three attributes do you think are the most and less relevant?

**Most relevant**

1. All under technique

**Less relevant**

1. Rigour- need some explanation

2. All are important - keep them

6. Would you delete attributes from the schema? Which one(s)? Why?

No.

7. Would you add attributes to any element in the schema? Which one(s)? Why?

A. Add Purpose to Scope because A test manager should see why the test technique may be useful, as soon as possible.

8. Would you rename attributes in the schema? Which one(s)?

No.

#### IV. REGARDING SCHEMA ELEMENTS

9. Do you think it makes sense grouping schema attributes in elements to which they refer?

A. Yes ☒

B. No ☐

10. Would you move attributes from one element to another?

No.

11. Would you delete elements from the schema? Which one(s)? Why?

No.

12. Would you add elements to any level in the schema? Which one(s)? Why?

No.

13. Would you rename elements in the schema? Which one(s)?

No.

#### V. REGARDING SCHEMA LEVELS

14. Do you think it makes sense grouping schema elements in levels?

A. Yes ☒

B. No ☐

Figure E.11: Answers given by expert 2 (2/3).

15. Would you move elements from one level to another?

No.

16. Would you delete levels from the schema? Which one(s)? Why?

No.

17. Would you add levels to the schema? Which one(s)? Why?

No.

18. Would you rename levels in the schema? Which one(s)?

A. Rename Use level as Historical

## VI. OTHER COMMENTS

19. Do you have any comments or suggestion?

I chose rigor as the least valuable. Usually the consumer determines how exhaustively he will use a test technique. A test technique itself does not usually determines rigor. I think it is EXTREMELY valuable for the consumer to know with what rigor an artifact should be tested. Perhaps I am misunderstanding how you are intending this attribute to be part of a test technique description.

I might prefer a different graphic, such as Tactical, followed by operational, followed by use. I know they fit nicely in the boxes as they are, but usage is the history of others' usage of the technique, so chronologically it follows. I suggest changing use to historical, or even use history. Initially, without seeing the elements, I expected attributes about how to use the technique.

Figure E.12: Answers given by expert 2 (3/3).

## EXPERT 3

### I. PERSONAL DATA

Position: Senior researcher

Company/Institution: IEI/CNR

Experience in software testing (brief paragraph): Researcher, 10 years

### II. GENERAL QUESTIONS

1. Do you think it would be useful for a company to have a repository containing information about software testing techniques in order to facilitate their selection?

A. Yes ☒

B. No ☐

2. Do you think that -in general- the proposed characterisation schema achieves this goal?

A. Yes ☐

B. No ☒

If not, why?

The present schema -in the attempt to be exhaustive- collects many attributes , and therefore it becomes very costly to read, use and maintain. To be continuously and effectively used, the repository should be maintained as essential as possible, and be designed to be easy to use for the testers, that will have little time and energy available. Not to completely give up with exhaustiveness, just an idea could be that the information reported be subdivided into two classes:

-ESSENTIAL, or PRIMARY attributes (e.g. OBJECT/programming language, or TOOLS/automation) - ADDITIONAL, or SECONDARY attributes (e.g. User satisfaction, or Understandability), whereby a user of the repository should necessarily fill each and every primary attribute field (only few), while the compilation of secondary attributes could remain optional. I am not implying with the examples that User's satisfaction is less important than the programming language; I only mean that without knowing the primary attributes a technique cannot be used, while other attributes are fine to know, but not essential to apply the technique. Several questions are very difficult to answer, if feasible at all: Test data cost, in Operational level: how can one easily introduce such an information?

In general, providing a spectrum of answers from which to pick one, where feasible, would make filling the repository easier and more immediate.

Moreover, I think that the schema could be improved with regard to avoiding redundancy and effectiveness in communication towards the repository user. In the next part of the questionnaire I will point at some characteristics that are very strictly related (the questionnaire says "the same": I know they are not exactly the same, nonetheless I think it is difficult to trace a clear boundary between them).

Figure E.13: Answers given by expert 3 (1/3).



3. Do you think the proposed schema is easy to understand?

A. Yes ☒

B. No ☐

If not, why?

*For two reasons: as I say above, there are too many attributes, and some of them are correlated (i.e., the distinction between them is not clear). In the current version of the schema I fear that data introduced by two different people could not be meaningfully compared.*

*Second reason, because the attributes are not identified at the same level of concreteness. Cost (money) of the tools is something very concrete, and could be introduced as a number; Development method is less concrete, some specification should be given to how identify it, but still decidable; Quality attribute that can be tested is very abstract.*

### III. REGARDING SCHEMA ATTRIBUTES

4. Do you think there are redundant attributes in the schema?

- A. Technique/Cost of application and Results/Execution cost are the same
- B. Objective/Quality attribute and Results/Type of defects are the same
- C. Scope/Aspect and Object/Software type are the same
- D. Agents/Experience and Agents/Knowledge are the same
- E. Tools/Environment and Object/Programming language are the same

5. What three attributes do you think are the most and less relevant?

#### Most relevant

1. Cost of application
2. Software architecture
3. Automation

#### Less relevant

1. Aspect
2. Adequacy degree
3. Correctness

6. Would you delete attributes from the schema? Which one(s)? Why?

- A. Delete Rigour from Objective because *it does not depend on the test technique, but on the test subject*
- B. Delete Adequacy degree from Results because *it is not known in most cases, and in some cases it may not be meaningful-for instance, in operational testing, which is the coverage, and who cares about knowing it?*
- C. Delete Correctness from Results because *it is not measurable (how can one know in general how many test cases should be deleted?), and confusing (maybe you mean precision, or avoidance of redundant test cases, but this is not correctness).*

7. Would you add attributes to any element in the schema? Which one(s)? Why?

*No.*

Figure E.14: Answers given by expert 3 (2/3).

8. Would you rename attributes in the schema? Which one(s)?  
No.

#### IV. REGARDING SCHEMA ELEMENTS

9. Do you think it makes sense grouping schema attributes in elements to which they refer?

A. Yes ☒

B. No ☐

10. Would you move attributes from one element to another?

A. Move Development method from Object to Objective

11. Would you delete elements from the schema? Which one(s)? Why?

No.

12. Would you add elements to any level in the schema? Which one(s)? Why?

No.

13. Would you rename elements in the schema? Which one(s)?

No.

#### V. REGARDING SCHEMA LEVELS

14. Do you think it makes sense grouping schema elements in levels?

A. Yes ☒

B. No ☐

15. Would you move elements from one level to another?

No.

16. Would you delete levels from the schema? Which one(s)? Why?

No.

17. Would you add levels to the schema? Which one(s)? Why?

No.

18. Would you rename levels in the schema? Which one(s)?

A. Rename Use level as Historical level

#### VI. OTHER COMMENTS

19. Do you have any comments or suggestion?

As a general guideline try to avoid to require data or information that is not generally or easily available.

3

Figure E.15: Answers given by expert 3 (3/3).

## EXPERT 4

### I. PERSONAL DATA

Position: Full professor

Company/Institution: Portland State University

Experience in software testing (brief paragraph): Five years in industry (long ago, 1970), 30 years in testing research and teaching.

### II. GENERAL QUESTIONS

1. Do you think it would be useful for a company to have a repository containing information about software testing techniques in order to facilitate their selection?

Yes, it would be useful if it could be done, but I doubt if a useful one can be constructed today. Part of the trouble is the information base is too large, and so any repository is bound to be a distorted simplification of complex technical issues (or, at the other extreme, it would be too messy to use). But the worst difficulty is that the information about the techniques is not available in any form that can be collected, summarized, or compared reasonably. Frankly, we don't know very much about them, and most of what we know is based on wishful thinking.

Doing such a study is the kind of thing the SEI has done very well in the past. They have produced a number of quite scholarly works (esp. has Gary Ford; now who trained him in this work?). But these studies have not been used, since they attempt the impossible: to make sense of a chaotic, developing discipline.

2. Do you think that -in general- the proposed characterisation schema achieves this goal?

(Does not answer).

3. Do you think the proposed schema is easy to understand?

Some parts are easy to understand, others are hard. The former don't say much, and the latter often fail to say anything. Examples from Tactical level, illustrating answer 1 above:

Easy: Scope: Element: A simple enough idea, but who is to answer for even the simplest techniques? "Unit testing" is applied at unit (routine) level, but that isn't exactly helpful information. There will be people who think it applies to subsystems, and others who think not, etc.

Hard: Objective (both): There are no answers in the literature to these questions. People argue about them, without any real basis for their opinions.

The plan for such a characterization is so far from realizable that a detailed critique would be impossible to give.

Figure E.16: Answers given by expert 4 (1/3).

### III. REGARDING SCHEMA ATTRIBUTES

4. Do you think there are redundant attributes in the schema?

*(Does not answer).*

5. What three attributes do you think are the most and less relevant?

*(Does not answer).*

6. Would you delete attributes from the schema? Which one(s)? Why?

*(Does not answer).*

7. Would you add attributes to any element in the schema? Which one(s)? Why?

*(Does not answer).*

8. Would you rename attributes in the schema? Which one(s)?

*(Does not answer).*

### IV. REGARDING SCHEMA ELEMENTS

9. Do you think it makes sense grouping schema attributes in elements to which they refer?

*(Does not answer).*

10. Would you move attributes from one element to another?

*(Does not answer).*

11. Would you delete elements from the schema? Which one(s)? Why?

*(Does not answer).*

12. Would you add elements to any level in the schema? Which one(s)? Why?

*(Does not answer).*

13. Would you rename elements in the schema? Which one(s)?

*(Does not answer).*

### V. REGARDING SCHEMA LEVELS

14. Do you think it makes sense grouping schema elements in levels?

*(Does not answer).*

2

Figure E.17: Answers given by expert 4 (2/3).

15. Would you move elements from one level to another?

(Does not answer).

16. Would you delete levels from the schema? Which one(s)? Why?

(Does not answer).

17. Would you add levels to the schema? Which one(s)? Why?

(Does not answer).

18. Would you rename levels in the schema? Which one(s)?

(Does not answer).

## VI. OTHER COMMENTS

19. Do you have any comments or suggestion?

If anything like this is to be of any use, the scope has to be cut drastically. To say a few things about a few techniques and situations, things that can be agreed upon and defended, is far more useful than to say almost nothing (and that controversialia) about everything. So the major work in the study would be to identify those few things and techniques. It would also be valuable (and a by-product of finding the few) to give some evidence why the vast majority of topics are not acceptable.

Figure E.18: Answers given by expert 4 (3/3).



# Appendix F

## Schema Instantiation

This appendix contains the repository obtained when instantiating the following testing techniques:

- Boundary value analysis. Shown in Table F.1.
- Random testing. Shown in Table F.2.
- Sentence coverage. Shown in Table F.3.
- Decision coverage. Shown in Table F.4.
- Path coverage. Shown in Table F.5.
- Threads coverage. Shown in Table F.6.
- All-possible-rendezvous. Shown in Table F.7.
- All-c-uses. Shown in Table F.8.
- All-p-uses. Shown in Table F.9.
- All-uses. Shown in Table F.10.
- All-du-paths. Shown in Table F.11.
- Mutation. Shown in Table F.12.
- Selective mutation. Shown in Table F.13.

LEVEL	ELEMENT	ATTRIBUTE	VALUE
Tactical	Objective	Purpose	Find defects
		Defect type	Control
		Effectiveness	Finds 55% of defects
	Scope	Element	Any
		Aspect	Any
Operational	Agents	Knowledge	None
		Experience	None
	Tools	Identifier	--
		Automation	--
		Cost	--
		Environment	--
		Support	--
	Technique	Comprehensibility	High
		Cost of Application	Low
		Inputs	Code specification
		Test Data Cost	Low
		Dependencies	When applied with black-box the effectiveness may rise to 75%
		Repeatability	No
		Sources of Information	Beizer, Sommerville
		Adequacy criterion	Functional: Boundary Value Analysis
	Test Cases	Completeness	--
		Precision	-- --
		Number of generated cases	Depends on the complexity of the input domain
	Object	Software type	Any
		Software architecture	Any
		Programming language	Any
		Development method	Any
		Size	Any
Historical	Project	Reference projects	--
		Tools used	--
		Personnel	--
	Satisfaction	Opinion	--
		Benefits	--
		Problems	--

Table F.1: Boundary value analysis technique



LEVEL	ELEMENT	ATTRIBUTE	VALUE
Tactical	Objective	Purpose	Find defects
		Defect type	Any
		Effectiveness	Prob. detecting a fault=42%
	Scope	Element	Units (functions), complete systems
		Aspect	Any
Operational	Agents	Knowledge	None
		Experience	Errors people usually make
	Tools	Identifier	--
		Automation	--
		Cost	--
		Environment	--
		Support	--
	Technique	Comprehensibility	High
		Cost of Application	Low
		Inputs	Code specification
		Test Data Cost	Low
		Dependencies	Might (and should) be completed with other technique
		Repeatability	No
		Sources of Information	Beizer, Myers, Sommerville, Pfleeger
		Adequacy criterion	Random testing
	Test Cases	Completeness	
		Precision	
		Number of generated cases	As many as wanted
	Object	Software type	Any
		Software architecture	Any
		Programming language	Any
		Development method	Any
		Size	Any
Historical	Project	Reference projects	--
		Tools used	--
		Personnel	--
	Satisfaction	Opinion	It is fine for complementing other techniques, or for acceptance testing.
		Benefits	It is very easy to apply
		Problems	- Although the mean effectiveness is high, its variance is also high - Maximum benefit is get with people with experience

Table F.2: Random testing technique

LEVEL	ELEMENT	ATTRIBUTE	VALUE
Tactical	Objective	Purpose	Find defects
		Defect type	Control
		Effectiveness	Prob. detecting a fault: 32%
	Scope	Element	Units
		Aspect	Any
Operational	Agents	Knowledge	None
		Experience	None
	Tools	Identifier	LOGISCOPE
		Automation	Obtain paths
		Cost	Between € 3.000 and € 6.000
		Environment	Windows; Any; Ada, C/C++
		Support	24 Hot-line
	Technique	Comprehensibility	High
		Cost of Application	Low
		Inputs	Source code
		Test Data Cost	High (less with tools)
		Dependencies	Might be completed with techniques that find processing errors.
		Repeatability	No
		Sources of Information	Sommerville
		Adequacy criterion	Control flow: Sentence coverage
	Test Cases	Completeness	
		Precision	
		Number of generated cases	Rises exponentially with the number of decisions in the code
	Object	Software type	Any
		Software architecture	Any
		Programming language	Any
		Development method	Any
		Size	Any
Historical	Project	Reference projects	--
		Tools used	Static analysers
		Personnel	--
	Satisfaction	Opinion	It is acceptable, but should be complemented with other techniques
		Benefits	It is easy to apply
		Problems	When used with real time and concurrent systems, the use of the dynamic analyser should be avoided, because it instruments the code and might change timing constraints.

Table F.3: Sentence coverage technique

LEVEL	ELEMENT	ATTRIBUTE	VALUE
Tactical	Objective	Purpose	Find defects
		Defect type	Control
		Effectiveness	Prob. detecting a fault: 48%
	Scope	Element	Units
		Aspect	Any
Operational	Agents	Knowledge	Flow graphs (when tool is not used)
		Experience	None
	Tools	Identifier	LOGISCOPE
		Automation	Obtain paths
		Cost	Between € 3.000 and € 6.000
		Environment	Windows; Any; Ada, C/C++
		Support	24 Hot-line
	Technique	Comprehensibility	High
		Cost of Application	Low
		Inputs	Source code
		Test Data Cost	High (less with tools)
		Dependencies	Should be completed with techniques that find processing errors.
		Repeatability	No
		Sources of Information	Sommerville
		Adequacy criterion	Control flow: decision coverage
	Test Cases	Completeness	
		Precision	
		Number of generated cases	Rises exponentially with the number of decisions in the code
	Object	Software type	Any
		Software architecture	Any
		Programming language	Any
		Development method	Any
		Size	Medium
Historical	Project	Reference projects	--
		Tools used	--
		Personnel	--
	Satisfaction	Opinion	It is okay, but should be completed with others
		Benefits	It is easy to apply
		Problems	When used with real time and concurrent systems, the use of the dynamic analyser should be avoided, because it instruments the code and might change timing constraints.

Table F.4: Decision coverage technique

LEVEL	ELEMENT	ATTRIBUTE	VALUE
Tactical	Objective	Purpose	Find defects
		Defect type	Control
		Effectiveness	Prob. detecting a fault: 66%
	Scope	Element	Units
		Aspect	Any
Operational	Agents	Knowledge	Flow graphs (if tool is not used)
		Experience	None
	Tools	Identifier	LOGISCOPE
		Automation	Obtain paths
		Cost	Between € 3.000 and € 6.000
		Environment	Windows; Any; Ada, C/C++
		Support	24 Hot-line
	Technique	Comprehensibility	High
		Cost of Application	Low
		Inputs	Source code
		Test Data Cost	High (less with tools)
		Dependencies	Should be completed with techniques that find processing errors.
		Repeatability	No
		Sources of Information	Sommerville
		Adequacy criterion	Control flow: path coverage
	Test Cases	Completeness	
		Precision	
		Number of generated cases	Rises exponentially with the number of decisions in the code
	Object	Software type	Any
		Software architecture	Any
		Programming language	Any
		Development method	Any
		Size	Medium
Historical	Project	Reference projects	--
		Tools used	--
		Personnel	--
	Satisfaction	Opinion	It is okay, but should be completed with others
		Benefits	It is easy to apply
		Problems	When used with real time and concurrent systems, the use of the dynamic analyser should be avoided, because it instruments the code and might change timing constraints.

Table F.5: Path coverage technique

LEVEL	ELEMENT	ATTRIBUTE	VALUE
Tactical	Objective	Purpose	Find defects
		Defect type	Control
		Effectiveness	--
	Scope	Element	Units
		Aspect	Any
Operational	Agents	Knowledge	--
		Experience	None
	Tools	Identifier	--
		Automation	--
		Cost	--
		Environment	--
		Support	--
	Technique	Comprehensibility	High
		Cost of Application	Low
		Inputs	Source code
		Test Data Cost	High
		Dependencies	Should be completed with black-box techniques
		Repeatability	No
		Sources of Information	Pressman
		Adequacy criterion	Control flow: thread coverage
	Test Cases	Completeness	--
		Precision	--
		Number of generated cases	Rises exponentially with the number of decisions in the code
	Object	Software type	Any
		Software architecture	Object Oriented
		Programming language	Any, but Object Oriented
		Development method	Any
		Size	Medium
Historical	Project	Reference projects	--
		Tools used	--
		Personnel	--
	Satisfaction	Opinion	--
		Benefits	--
		Problems	--

Table F.6: Threads coverage technique

LEVEL	ELEMENT	ATTRIBUTE	VALUE
Tactical	Objective	Purpose	Find defects
		Defect type	Any
		Effectiveness	Prob. detecting a fault: 66%
	Scope	Element	Any
		Aspect	Any
Operational	Agents	Knowledge	Reachability graphs
		Experience	None
	Tools	Identifier	--
		Automation	--
		Cost	--
		Environment	--
		Support	--
	Technique	Comprehensibility	Medium
		Cost of Application	Low
		Inputs	Source code
		Test Data Cost	High
		Dependencies	Might be completed with any other black box technique
		Repeatability	No
		Sources of Information	ISSTA'96
		Adequacy criterion	Data flow: all-possible rendezvous
	Test Cases	Completeness	--
		Precision	--
		Number of generated cases	High
	Object	Software type	Any
		Software architecture	Concurrent
		Programming language	Ada
		Development method	Ada
		Size	Medium
Historical	Project	Reference projects	--
		Tools used	--
		Personnel	--
	Satisfaction	Opinion	Works better for concurrent software than generic techniques
		Benefits	It found many defects that generic techniques did not find
		Problems	It is not easy to apply

Table F.7: All-possible-rendezvous technique

LEVEL	ELEMENT	ATTRIBUTE	VALUE
Tactical	Objective	Purpose	Find defects
		Defect type	Processing and control
		Effectiveness	Prob. detecting a fault: less than 60%
	Scope	Element	Units (modules and algorithms)
		Aspect	Any
Operational	Agents	Knowledge	None
		Experience	None
	Tools	Identifier	ASSET
		Automation	Data flow graph
		Cost	Symbolic. Academic tool
		Environment	UNIX; Any; Ada, C/C++
		Support	By tool developers
	Technique	Comprehensibility	Low (tool use)
		Cost of Application	Low
		Inputs	Source code
		Test Data Cost	Medium/high
		Dependencies	None
		Repeatability	No
		Sources of Information	Beizer
		Adequacy criterion	Data flow: all-c-uses
	Test Cases	Completeness	--
		Precision	--
		Number of generated cases	Theoretical limit: $(d^2+4d+3)/4$ ( $d=n^\circ$ binary decisions) Practical: 13% of the theoretical
	Object	Software type	Any
		Software architecture	Any
		Programming language	Estructured, OO, real time and concurrent
		Development method	Any
		Size	Medium
Historical	Project	Reference projects	--
		Tools used	--
		Personnel	--
	Satisfaction	Opinion	It is more easy to apply than it seems
		Benefits	It finds a lot of defects and it is easy to use
		Problems	Should not be use without a tool

Table F.8: All-c-uses technique

LEVEL	ELEMENT	ATTRIBUTE	VALUE
Tactical	Objective	Purpose	Find defects
		Defect type	Processing and control
		Effectiveness	Prob. detecting a fault 70% (aprox.) Cubre a all-c-uses
	Scope	Element	Units (modules and algorithms)
		Aspect	Any
Operational	Agents	Knowledge	None
		Experience	None
	Tools	Identifier	ASSET
		Automation	Data flow graph
		Cost	Symbolic. Academic tool.
		Environment	UNIX; Any; Ada, C/C++
		Support	By tool developers
	Technique	Comprehensibility	--
		Cost of Application	Low
		Inputs	Source code
		Test Data Cost	Medium/high
		Dependencies	--
		Repeatability	No
		Sources of Information	Beizer
		Adequacy criterion	Data flow: all-p-uses
	Test Cases	Completeness	--
		Precision	--
		Number of generated cases	Theoretical limit: $(d^2+4d+3)/4$ ( $d=n^o$ binary decisions) Practical: 23% of the theoretical
	Object	Software type	Any
		Software architecture	Any
		Programming language	Estructured, OO, real time and concurrent
		Development method	Any
		Size	Medium
Historical	Project	Reference projects	--
		Tools used	--
		Personnel	--
	Satisfaction	Opinion	It is easier to use than it seems
		Benefits	It finds a lot of defects and it is easy to use
		Problems	Should not be used without a tool

Table F.9: All-p-uses technique



LEVEL	ELEMENT	ATTRIBUTE	VALUE
Tactical	Objective	Purpose	Find defects
		Defect type	Processing and control
		Effectiveness	Prob detecting a fault: 70% Cubre all-c-uses
	Scope	Element	Units (modules and algorithms)
		Aspect	Any
Operational	Agents	Knowledge	None
		Experience	None
	Tools	Identifier	ASSET
		Automation	Data flow graph
		Cost	Symbolic. Academic tool.
		Environment	UNIX; Any; Ada, C/C++
		Support	By tool developers
	Technique	Comprehensibility	--
		Cost of Application	Low
		Inputs	Source code
		Test Data Cost	Medium/high
		Dependencies	--
		Repeatability	No
		Sources of Information	Beizer
		Adequacy criterion	Data flow: all-uses
	Test Cases	Completeness	--
		Precision	--
		Number of generated cases	Theoretical limit: $(d^2+4d+3)/4$ ( $d=n^\circ$ binary decisions) Practical: 24% of the theoretical
	Object	Software type	Any
		Software architecture	Any
		Programming language	Structured, OO, real time and concurrent
		Development method	Any
		Size	Medium
Historical	Project	Reference projects	--
		Tools used	--
		Personnel	--
	Satisfaction	Opinion	It is easier to use than it seems
		Benefits	It finds a lot of defects and it is easy to use
		Problems	Should not be used without a tool

Table F.10: All-uses technique

LEVEL	ELEMENT	ATTRIBUTE	VALUE
Tactical	Objective	Purpose	Find defects
		Defect type	Processing and control
	Alcance Scope	Effectiveness	Units (modules and algorithms)
		Element	Any
Operational	Agents	Aspect	None
		Knowledge	None
	Herramientas Tools	Experience	ASSET
		Identifier	Data flow graph
		Automation	Symbolic. Academic tool.
		Cost	UNIX; Any; Ada, C/C++
		Environment	Sujeto a los desarrolladores de la herramienta
	Técnica Technique	Support	--
		Comprehensibility	Medium
		Cost of Application	Source code
		Inputs	Medio/alto
		Test Data Cost	--
		Dependencies	No
		Repeatability	Beizer
		Sources of Information	Flujo de datos: all-du-paths
	Casos de prueba Test Cases	Adequacy criterion	--
		Completeness	--
		Precision	Theoretical: $2^d$ ( $d=n^\circ$ binary decisions) Practical: 0,4% of the theoretical
	Objeto Object	Number of generated cases	Any
		Software type	Any
		Software architecture	Structured, OO, real time and concurrent
		Programming language	Any
		Development method	Medium
Historical	Project	Size	--
		Reference projects	--
		Tools used	--
	Experiencia Satisfaction	Personnel	It is easier to use than it seems
		Opinion	It finds a lot of defects and it is easy to use
		Benefits	Should not be used without a tool

Table F.11: All-du-paths technique

LEVEL	ELEMENT	ATTRIBUTE	VALUE
Tactical	Objective	Purpose	Find defects
		Defect type	Any
		Effectiveness	Detects approx.. 72% of the faults
	Scope	Element	Units (modules and algorithms)
		Aspect	Any
Operational	Agents	Knowledge	None
		Experience	None
	Tools	Identifier	Mothra
		Automation	Generates mutants automatically
		Cost	Free. Academic tool.
		Environment	Windows/UNIX; Any; Pascal, C
		Support	By tool developers
	Technique	Comprehensibility	High
		Cost of Application	Low
		Inputs	Source code
		Test Data Cost	Medium/high
		Dependencies	--
		Repeatability	Yes
		Sources of Information	ACM's papers
		Adequacy criterion	Mutation
	Test Cases	Completeness	--
		Precision	--
		Number of generated cases	$A+b*n+c*n^2$ , with $n = n^0$ of lines of code
	Object	Software type	Any
		Software architecture	Any
		Programming language	Structured, OO, real time and concurrent
		Development method	Any
		Size	Medium
Historical	Project	Reference projects	--
		Tools used	--
		Personnel	--
	Satisfaction	Opinion	It is easier to use than it seems
		Benefits	It finds a lot of defects and it is easy to use
		Problems	Should not be used without a tool

Table F.12: Mutation technique

LEVEL	ELEMENT	ATTRIBUTE	VALUE
Tactical	Objective	Purpose	Find defects
		Defect type	Any
		Effectiveness	More than 99% of mutation
	Scope	Element	Units (modules and algorithms)
		Aspect	Any
Operational	Agents	Knowledge	None
		Experience	None
	Tools	Identifier	Mothra
		Automation	Generates mutants automatically
		Cost	Free. Academic tool.
		Environment	Windows/UNIX; Any; Pascal, C
		Support	By tool developers
	Technique	Comprehensibility	High
		Cost of Application	Low
		Inputs	Source code
		Test Data Cost	Medium/high
		Dependencies	--
		Repeatability	Yes
		Sources of Information	ACM's papers
		Adequacy criterion	Selective Mutation
	Test Cases	Completeness	--
		Precision	
		Number of generated cases	24% of std. mutation for 2-selectiva 42% of std. mutation for 4-selectiva 60% of std. mutation for 6-selectiva
	Object	Software type	Any
		Software architecture	Any
		Programming language	Structured, OO, real time and concurrent
		Development method	Any
		Size	Medium
Historical	Project	Reference projects	--
		Tools used	--
		Personnel	--
	Satisfaction	Opinion	It is easier to use than it seems
		Benefits	It finds a lot of defects and it is easy to use
		Problems	Should not be used without a tool

Table F.13: Selective mutation technique

## Appendix G

# Additional Information on the Experiment

This appendix contains additional information about the experiment. Section G.1 presents the description (contexts) of the projects used in the experiment. Section G.2 presents the forms used in the experiment. Finally, Section G.3 shows the the validity proofs of the ANOVAS (normal sample with constant variance) calculated during the data analysis.

### G.1 Project Contexts

Four projects have been used during the experimental evaluation: a system which manages the parking spaces in a garage, a loan arranger for the concession of loans in a bank, a system to manage movie renting in a video club and a system for monitoring the water level of a pump. Next, each project context is presented.

#### G.1.1 Parking Garage Project Context

The characteristics of the environment where the development of this application will take place are:

- The system to be developed is concurrent. This exercise intends the evaluation of its correctness.
- The development team is composed of experienced people in the development of this sort of systems.
- The testing team has experience in testing this sort of systems (concurrents).
- The source code is available during the testing stage.
- The system will be codified in a concurrent programming language.
- In this moment, the company does not own tools. The testing process is totally manual.

- The total time available for testing is two months, which for this system is enough (this is, there are not heavy time constraints).
- The system should go on to the market with as less defects as possible.
- The company is not interested in buying any testing tool.

### **G.1.2 Loan Arranger Project Context**

The characteristics of the environment in which the development of this application will take place are the following:

- Although the development of the application has been negotiated with the company Fanny Mae, the company developing the system intends selling it to other companies in the same area.
- The development team is experienced.
- The testing team is novice. This means that they are only familiar with control-flow, structural and functional techniques.
- The system will be developed using a structured methodology.
- The company owns a dynamic analyser which will ease the application of white-box techniques, but the tool is not familiar to the testing team.
- The available time for testing the software is tight (not enough).
- The size of the software system is not very big. Programmers are used to code big and complex functions.
- The testing process should find as many defects as possible. However, the software will be delivered to Fanny Mae for beta testing.

### **G.1.3 Video Project Context**

The characteristics of the environment in which the development of this application will take place are the following:

- The testing to be performed for this system aims at defect finding.
- The development team is composed mostly by people with no working experience. This means that they will have problems mainly with the first stages of the development (requirements and design), which are the most difficult.
- The testing team is experienced in testing this sort of systems (management).
- The source code will be available during testing.

- The development will be performed according to the OO paradigm, and the programming language to be used will be C++.
- The company does not own tools to ease the testing phase, although the video company is going to pay for a testing tool during the development of this product.
- The total time available for testing is 2 months, which is enough for a system like this.
- The size of the software being developed is small.
- The software should go on to the market with as less defects as possible.

#### **G.1.4 WLMS Project Context**

The characteristics of the environment in which the development of this system will take place are the following:

- The testing objective is evaluating the correctness of the system; this is, find faults.
- The system to be developed is a real time one. Its correctness is essential, as well as meeting its time constraints.
- In this moment, the company does not have money to spend in tools, but owns a dynamic analyser which is widely used by the testing team.
- The available time for testing is 3 months. This time should be enough to test a system like this.
- The source code will be available during testing.
- The testing team is experienced, but not in testing this sort of systems (real time).
- The development team is experienced with the development of this sort of systems.
- The programming language will be Ada, and a specific methodology for real time systems will be used for development.
- The company has a problem, and is that it is level 1 CMM. This means that its process is not mature and they devote more time to coding than to analysis and design. This implies that a lot of effort is spent in testing, but the system may have a lot of faults.

## **G.2 Forms Used**

Next, the forms used during the experiment are presented:

- *Form E0*. Used before the experiment starts. It is shown in Figure G.1 and Figure G.2.
- *Form E1*. Used during selection with books. It is shown in Figure G.3 and Figure G.4.
- *Form E2*. Used during selection with schema. It is shown in Figure G.5 and Figure G.6.
- *Form E3*. Used during selection with both books and schema. It is shown in Figure G.7.
- *Form E4*. Used during selection with both books and schema. It is shown in Figure G.8.
- *Form E5*. Used during selection with both books and schema. It is shown in Figure G.9.
- *Form E6*. Used during selection with both books and schema. It is shown in Figure G.10.
- *Form E7*. Used during selection with both books and schema. It is shown in Figure G.11.
- *Form E8*. Used during selection with schema. It is shown in Figure G.12.
- *Form E9*. Used at the end of the experiment by the group who made both selections with books. It is shown in Figure G.13.

### G.3 ANOVA Validity Testing

In order to check the validity of the results of an ANOVA test, the sample under analysis should met two restrictions:

1. The sample corresponds to a normal population.
2. The sample has constant variance (it is homocedastic).

In order to check these restrictions, a study of the residuals obtained with the ANOVA has to be done.

1. *Normality test*. It is done by using a P-P graph of the residuals. The deviation of the figure plotted by the residuals against the normal line shows whether or not the normality constraint is met.



## Form E0 Selection Exercise/General

**Name:** \_\_\_\_\_

**Group:** \_\_\_\_\_ **Date:** \_\_\_\_\_

**Objective:** Know subject background and experience.

### Generic Questions

1. Do you have experience in the development of software systems? (it is considered as experience to have worked full or part time in a company or in a department in the university).

Yes ☐

No ☐

If yes, what is your experience in each position?

Position	Years	Months
Project manager		
Developer		
Analist		
Member of the testing team		
Others:		

2. Are you familiar with software testing? (Mark the scale)

Value	0	1	2	3
Familiarity level	Never tested	Small exercises	Class assignements	Real development

3. What is your experience in the specification of test cases for software testing?

Value	0	1	2	3
Experience	Never specified	Small exercises	Class assignements	Real development

Number of years/months since you started: \_\_\_\_\_.

Figure G.1: Form E0 (1/2).

4. What is your absolute experience in running test cases?

Value	0	1	2	3
Experience	Never run	Small exercises	Class assignments	Real development

Number of years/months since you started: \_\_\_\_\_.

5. What kind of experience do you have in software testing? Explain briefly in a paragraph theoretical and practical knowledge. Use this question to extend the answers to questions 2-4.

6. What heuristic(s) or technique(s) do you use when selecting the set of test cases to test a program?

7. What heuristic(s) do you use when selecting the software testing technique(s) to develop the set of test cases for a program?

8. What information do you think is relevant when deciding the testing technique(s) to develop the set of test cases for a program? (about the project, the problem to be solved, etc.). Try to be as specific as possible when referring to the information you would like to know when doing the selection.

9. What kind of problems do you think you will have to face when selecting the testing technique(s) to use in a project?

10. How long do you think it should take you the process of selecting the software testing techniques for a project? Which variables do you think will affect this time?

11. What do you think you will learn with this exercise?(Mark in the scale)

Value	0	1	2	3
Usefulness	Useless knowledge	Seldom used knowledge	Often used knowledge	Essential knowledge

Justify your answer

Figure G.2: Form E0 (2/2).

## Form E1 - Selection Exercise/Execution

**Name:** \_\_\_\_\_

**Group:** \_\_\_\_\_ **Date:** \_\_\_\_\_

**Objective:** Measure times and conclusions.

### Material

Check that you have all forms needed for this part of the work. You should have:

- A requirements document, with name: \_\_\_\_\_
- A project context document.
- Forms numbered: E1, E3,, E4, E5, E6 y E7.

### Exercise

The steps to be followed are:

1. Write down the time when you start the exercise: \_\_\_\_: \_\_\_\_ [hour:minutes]
2. To do this exercise remember that you have to:
  - (a) (Optional, discount this time). Take a look at the available techniques.
  - (b) Read the project documentation.
  - (c) Examine one by one the available testing techniques.
  - (d) Decide what are the technique(s) you would use for the project..
3. Write down the time when you start examining the project documentation: \_\_\_\_: \_\_\_\_ [hour:minutes]
4. Write down the time when you finish examining the project documentation: \_\_\_\_: \_\_\_\_ [hour:minutes]
5. How long has it taken you to examine the project documentation? (do not take into account possible interruptions) \_\_\_\_\_ minutes
6. Write down the time when you start the selection: \_\_\_\_: \_\_\_\_ [hour:minutes]
7. Write down the time when you finish the selection : \_\_\_\_: \_\_\_\_ [hour:minutes]
8. How long has it taken you to perform the selection? (do not take into account possible interruptions) \_\_\_\_\_ minutes

Figure G.3: Form E1 (1/2).

9. Write down the time when you finish the exercise: \_\_\_\_\_: \_\_\_\_\_ [hour:minutes]
10. How long has it taken you to do the exercise? (do not take into account possible interruptions)  
\_\_\_\_\_ minutes

### Conclusions

The aim of the conclusions is to study the evolution of the subject's opinion about testing techniques selection problem when doing this exercise.

1. What have you learned with this exercise?
  - (a) Regarding the type of software to be tested.
  - (b) Regarding the characteristics of the project environment.
  - (c) Others.
2. ¿Has your perception of the selection problem changed regarding the type of information needed for the selection, difficulty and problems encountered?
  - (a) Regarding the type of software to be tested.
  - (b) Regarding the characteristics of the project environment.
  - (c) Others.
3. If you had to do this exercise again, would you do things differently? what? why?
  - (a) Regarding the type of software to be tested.
  - (b) Regarding the characteristics of the project environment.
  - (c) Others.

Figure G.4: Form E1 (2/2).

## Form E2 - Selection Exercise/Execution

**Name:** \_\_\_\_\_

**Group:** \_\_\_\_\_ **Date:** \_\_\_\_\_

**Objective:** Measure times and conclusions.

### Material

Check that you have all forms needed for this part of the work. You should have:

- A requirements document, with name: \_\_\_\_\_
- A project context document.
- Forms numbered: E2, E3, E4, E5, E6, E7 y E8.

### Exercise

The steps to be followed are:

1. Write down the time when you start the exercise: \_\_\_\_: \_\_\_\_ [hour:minutes]
2. To do this exercise remember that you have to:
  - (a) (Optional, discount this time). Take a look at the available techniques.
  - (b) Read the project documentation.
  - (c) Examine one by one the available testing techniques.
  - (d) Decide what are the technique(s) you would use for the project..
3. Write down the time when you start examining the project documentation: \_\_\_\_: \_\_\_\_ [hour:minutes]
4. Write down the time when you finish examining the project documentation: \_\_\_\_: \_\_\_\_ [hour:minutes]
5. How long has it taken you to examine the project documentation? (do not take into account possible interruptions) \_\_\_\_\_ minutes
6. Write down the time when you start the selection: \_\_\_\_: \_\_\_\_ [hour:minutes]
7. Write down the time when you finish the selection : \_\_\_\_: \_\_\_\_ [hour:minutes]
8. How long has it taken you to perform the selection? (do not take into account possible interruptions) \_\_\_\_\_ minutes

1

Figure G.5: Form E2 (1/2).

9. Write down the time when you finish the exercise: \_\_\_\_\_: \_\_\_\_\_ [hour:minutes]
10. How long has it taken you to do the exercise? (do not take into account possible interruptions)  
\_\_\_\_\_ minutes

### Conclusions

The aim of this section is to know the opinion the subject has about the characterisation schema.

1. Do you think it achieves the goal of easing the selection? Why?
2. Do you think the names in the schema could be improved?
3. In general, do you think the schema could be improved in any way? How?
4. If you had the opportunity of using the schema in your work, would you do it?
5. What process have you followed to use the schema? In case it has not been the one recommended, please, explain it

### Conclusions

The aim of the conclusions is to study the evolution of the subject's opinion about testing techniques selection problem when doing this exercise.

1. What have you learned with this exercise?
  - (a) Regarding the type of software to be tested.
  - (b) Regarding the characteristics of the project environment.
  - (c) Others.
2. ¿Has your perception of the selection problem changed regarding the type of information needed for the selection, difficulty and problems encountered?
  - (a) Regarding the type of software to be tested.
  - (b) Regarding the characteristics of the project environment.
  - (c) Others.
3. If you had to do this exercise again, would you do things differently? what? why?
  - (a) Regarding the type of software to be tested.
  - (b) Regarding the characteristics of the project environment.
  - (c) Others.

Figure G.6: Form E2 (2/2).

*Sira Vegas*

**Form E4 - Selection Exercise/Information not Found**

**Objective:** Reflect the information you have missed about a technique during selection

**Name:** \_\_\_\_\_ **page** \_\_\_\_\_ **of** \_\_\_\_\_

**Document Name:** \_\_\_\_\_

Technique	Information	Comments

Figure G.8: Form E4.



Sira Vegas



*Sira Vegas*



## Form E9 - Selection Exercise/Schema

**Name:** \_\_\_\_\_

**Group:** \_\_\_\_\_ **Date:** \_\_\_\_\_

**Objective:** Check the potential the subject sees to the schema.

### About the Schema

The aim of this section is know the opinion the subject has about the characterisation schema.

1. Do you think it facilitates selection? Why?
2. Do you think it is easy to understand?
3. Do you think the names in the schema could be improved?
4. If you had to use it, do you think it would be easy to use? Do you think it could be improved to make it easier to use? How?
5. As you have seen, the development of the schema implies the investment of additional time in instantiating it. Taking this into account, if you had the opportunity to use it in your work, would you do it?

### Conclusions

The aim of the conclusions is to study the evolution of the subject's opinion about testing techniques selection problem when doing this exercise.

1. What have you learned with the characterisation schema?
2. When knowing the schema, has your perception of the selection problem changed regarding the type of information needed for the selection, difficulty and problems encountered?
3. If you had to do the previous exercises again, but using the schema, would you do things different? what? why?

Figure G.13: Form E9.

2. *Homocedasticity test.* It is done by using a dispersion graph, representing the x-axis the value of the predicted value for the response variable and the y-axis the value of the residuals. If the figure plotted by the residuals is similar to a funnel ( $>$  or  $<$ ), the homocedasticity restriction will not be met.

### G.3.1 Schema Efficiency

Figure G.14, Figure G.15 and Figure G.16 show the ANOVA validity tests for the response variables related to the effectiveness of the schema.

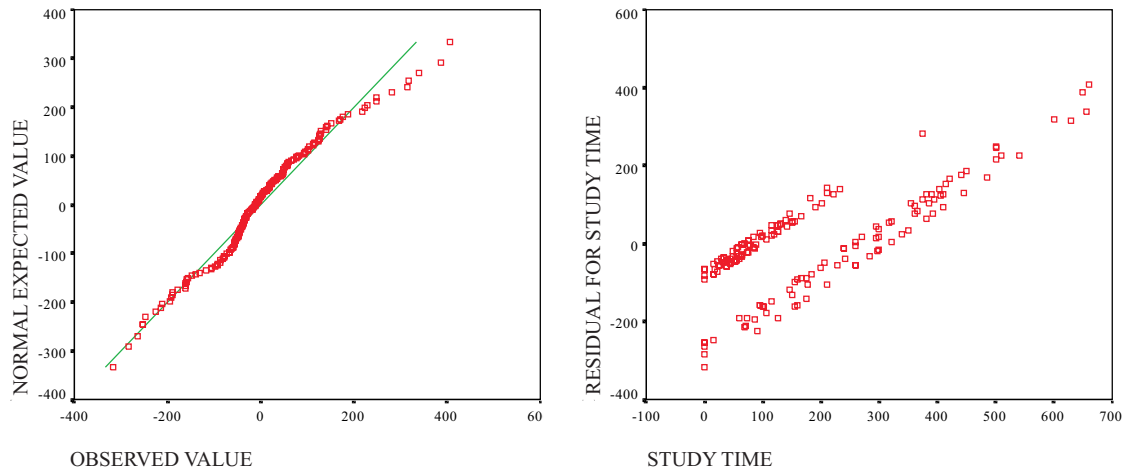


Figure G.14: Validity of the ANOVA results for study time.

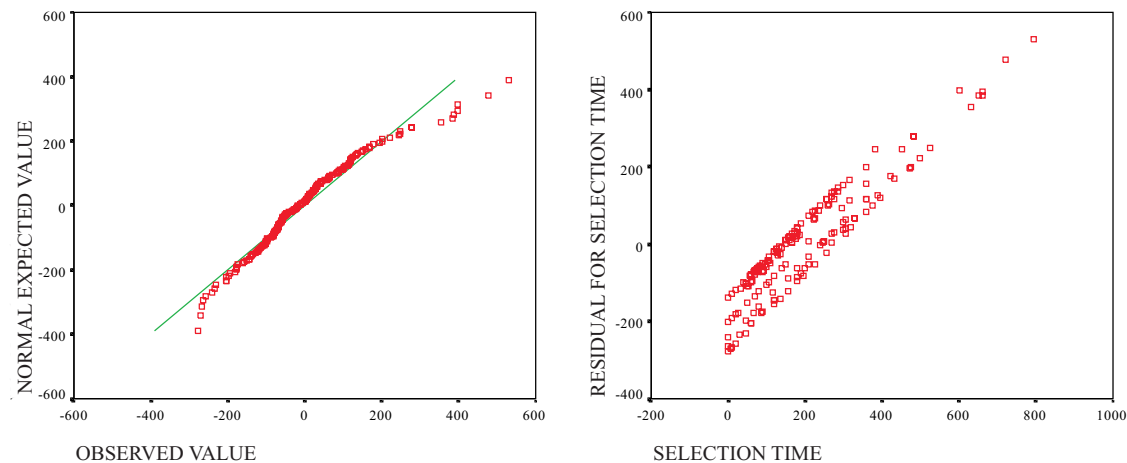


Figure G.15: Validity of the ANOVA results for selection time.

### G.3.2 Schema Usability

Figure G.17, Figure G.18, Figure G.19 and Figure G.20 show the ANOVA validity tests for the response variables related to the usability of the schema.

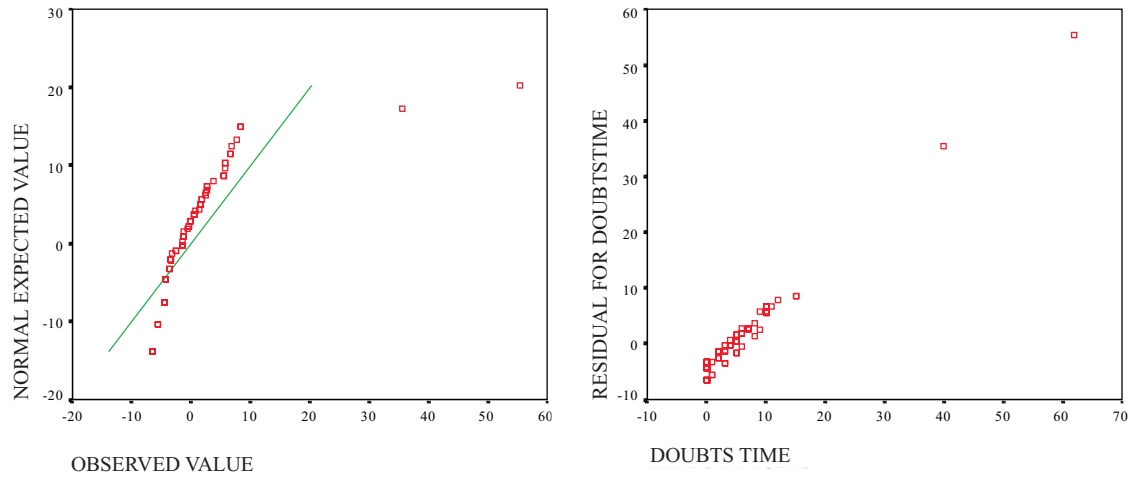


Figure G.16: Validity of the ANOVA results for the time spent consulting doubts about the schema.

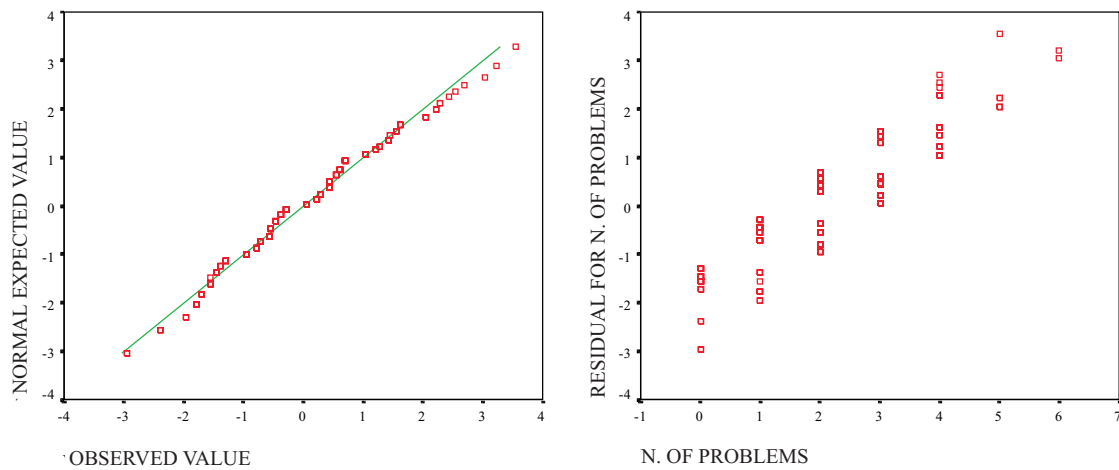


Figure G.17: Validity of the ANOVA results for number of problems encountered.

### G.3.3 Schema Completeness

Figure G.21, Figure G.22, Figure G.23 and Figure G.24 show the ANOVA validity tests for the response variables related to the completeness of the schema.

### G.3.4 Schema Effectiveness

Figure G.25, Figure G.26 and Figure G.27 show the ANOVA validity tests for the response variables related to the effectiveness of the schema.

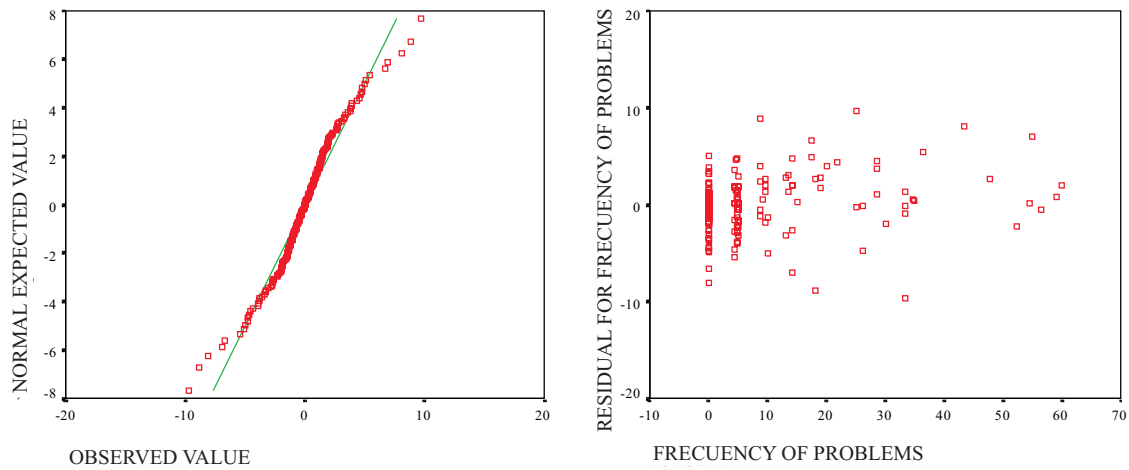


Figure G.18: Validity of the ANOVA results for frequency of problems encountered.

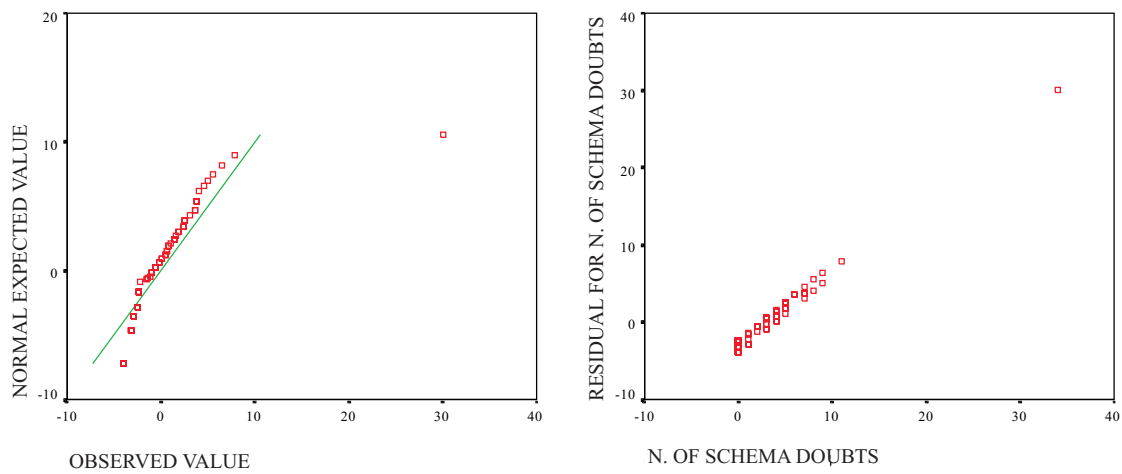


Figure G.19: Validity of the ANOVA results for the number of doubts about the schema.

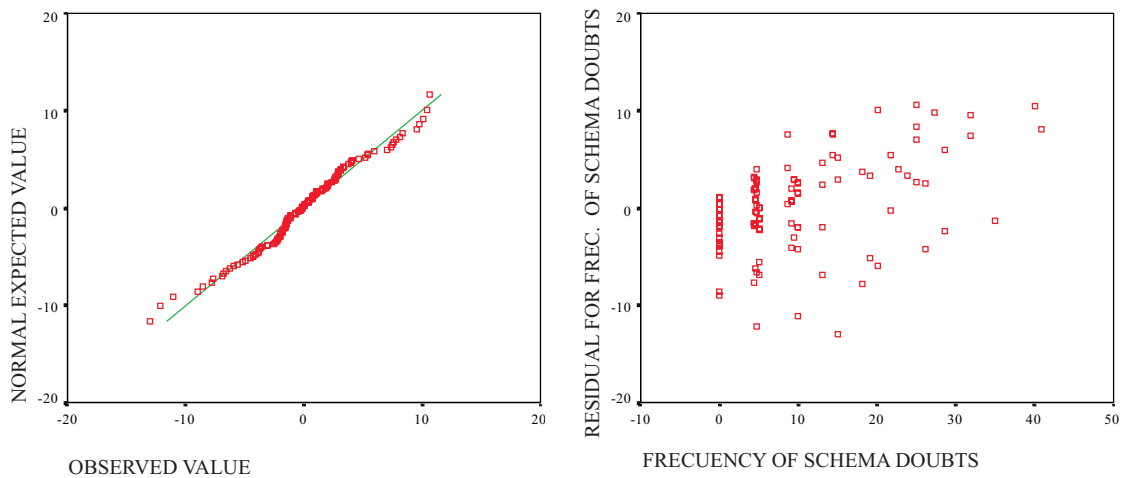


Figure G.20: Validity of the ANOVA results for the sort of doubts about the schema.



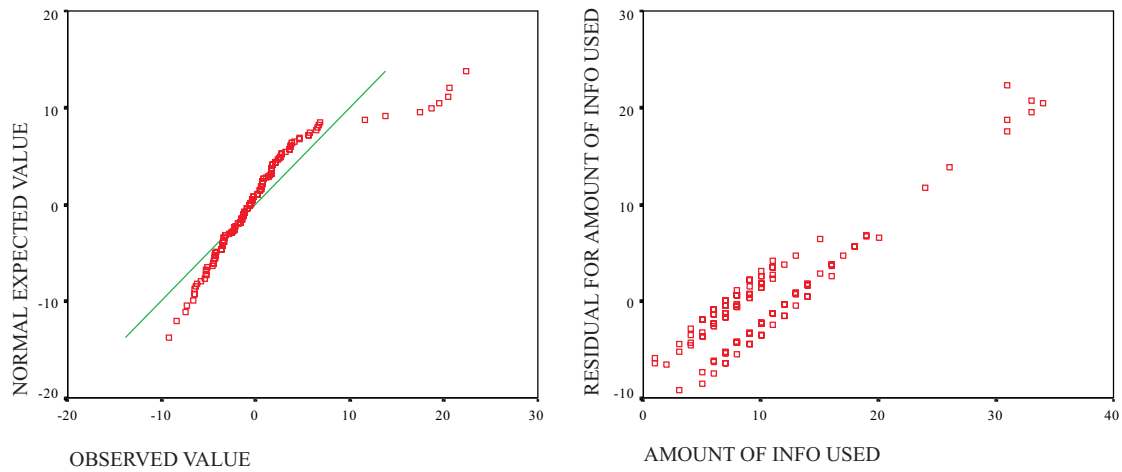


Figure G.21: Validity of the ANOVA results for the amount of information used in selection.

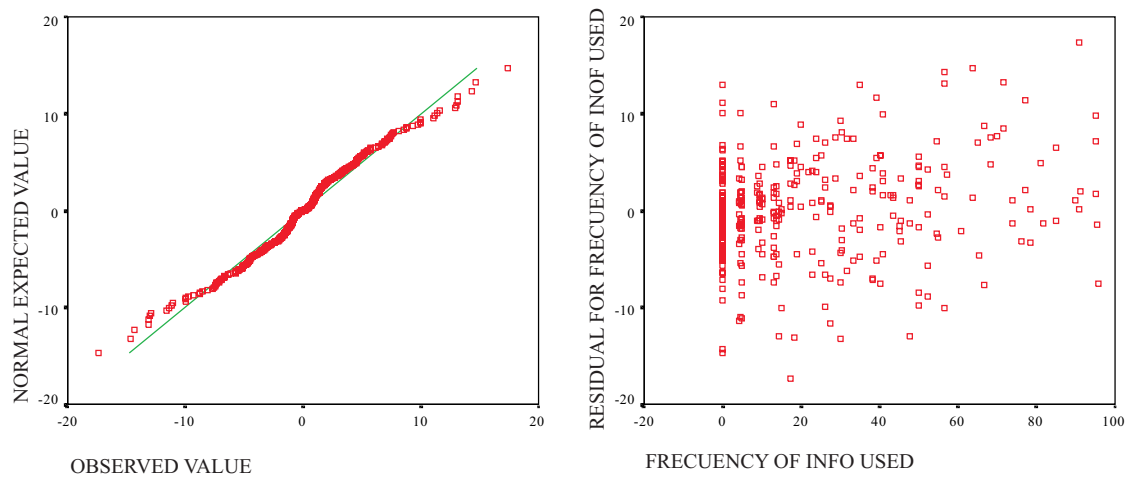


Figure G.22: Validity of the ANOVA results for the frequency of use of information.

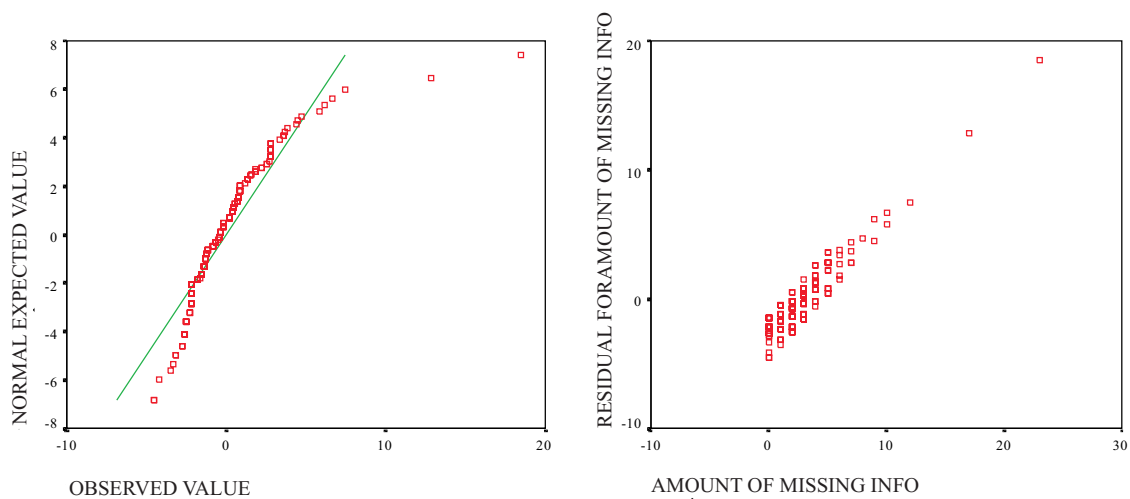


Figure G.23: Validity of the ANOVA results for the amount of missing information.

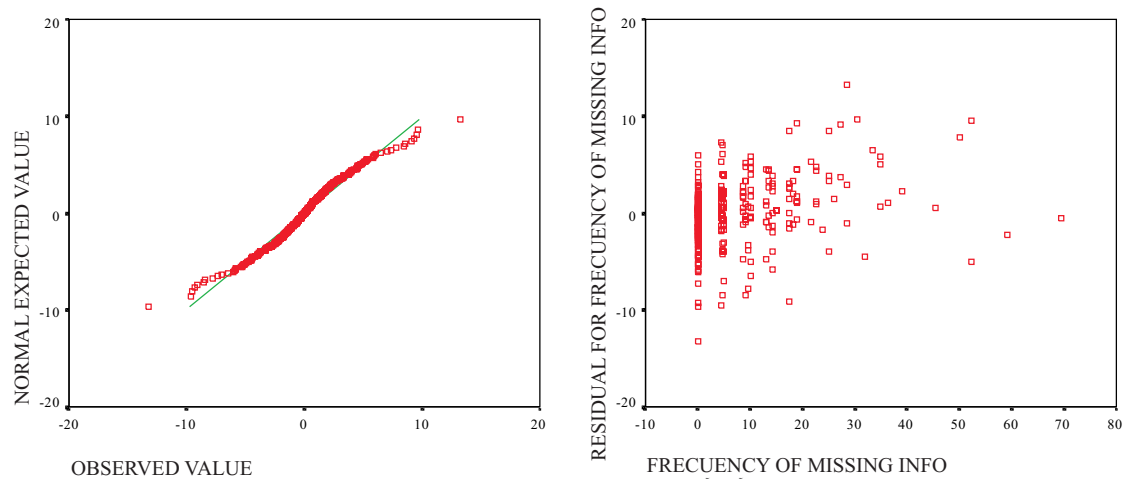


Figure G.24: Validity of the ANOVA results for the frequency of appearance of missing information.

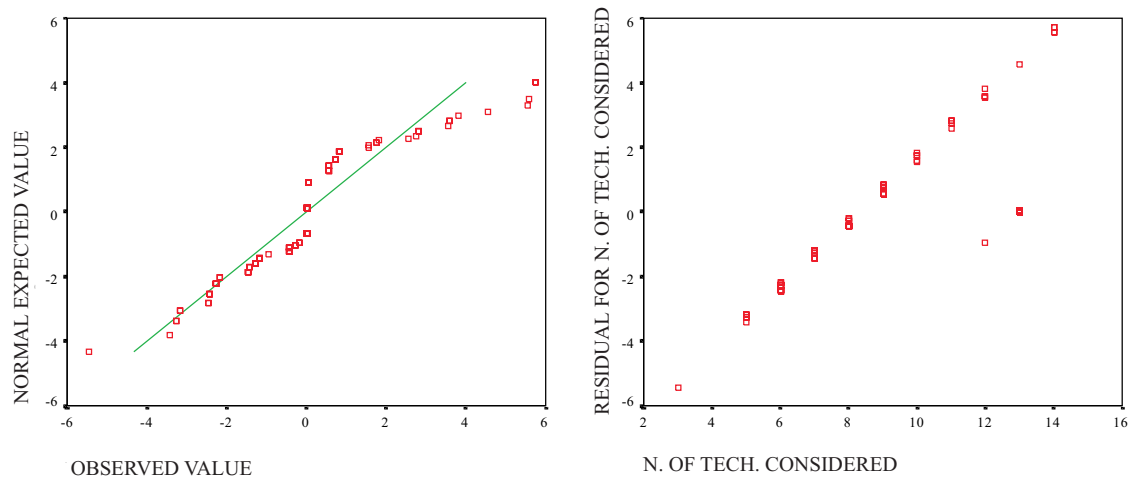


Figure G.25: Validity of the ANOVA results for number of techniques considered.

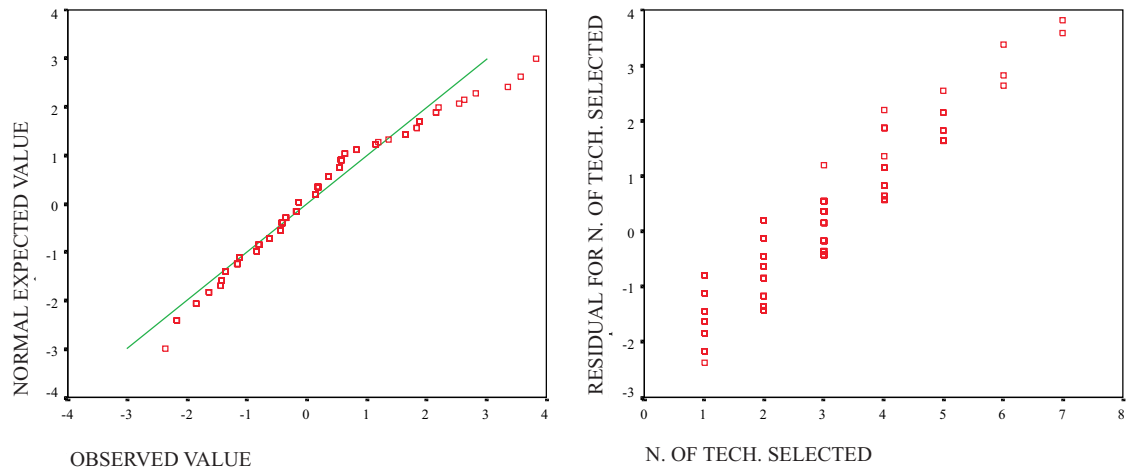


Figure G.26: Validity of the ANOVA results for the number of techniques selected.

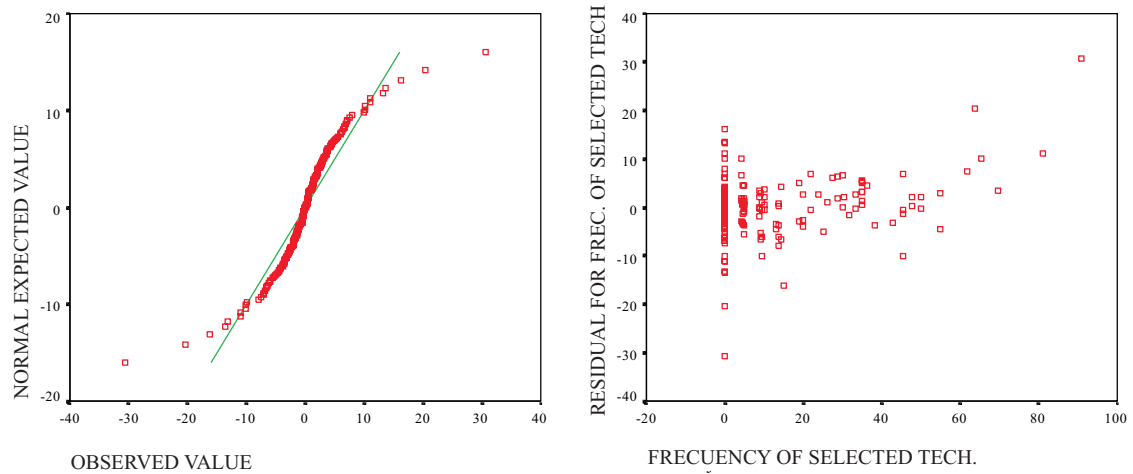


Figure G.27: Validity of the ANOVA results for the frequency of selection of each technique.

