10. A Process for Identifying Relevant Information for a Repository: A Case Study for Testing Techniques

S. Vegas[1], N. Juristo[1], V.R. Basili[2]

[1] Facultad de Informática. Universidad Politécnica de Madrid. Campus de Montegancedo. 28660 Boadilla del Monte, Madrid, Spain.

[2] Department of Computer Science. University of Maryland. College Park, Maryland 20742, USA

Abstract: One major issue in managing software engineering knowledge is the construction of information repositories for software development artefacts (techniques, products, processes, tools, etc.). But how does one package each artefact so that the package contains the appropriate information to understand and use the artefact? What is the appropriate characterisation schema? This chapter proposes an empirical and iterative process to identify the information that should be used to characterise a software engineering artefact, using both theoretical knowledge, practical experience, and expert opinion to generate a schema. The ultimate goal is to improve the schema and the package contents based upon it experience in their application. The proposed process has been applied to define a characterisation schema for testing techniques. There are nowadays numerous testing techniques available for generating test cases. However, many of them are never used, while a few are used over and over again. Testers have little (if any) information about the available techniques, their usefulness and, generally, how suited they are to the project at hand. This lack of information means less tuned decisions on which testing techniques to use. This chapter also shows this characterisation schema, discussing the information it contains and why it has been included in the schema.

Keywords: Knowledge management, experience packaging, software testing, testing techniques.

1 Introduction

The goal of knowledge management (KM) is to take advantage of an organisation's intellectual capital [15]. When applied to software development, this discipline deals with knowledge related to the whole range of software engineering *artefacts* (techniques, products, processes, methods, etc.).

To make the best possible use of organisational knowledge, this knowledge must be created, captured, distributed and applied [15]. Information organisation,

also known as packaging, is a key activity within this process. It is so critical that a poor information structure has led to the failure of many KM initiatives [11]. If the available information is well structured, knowledge will be more widely and better disseminated and applied, as people will be interested in and tend to consult well structured information and will be clearer about when to use it. The knowledge generation and capturing activities will also be more effective, as the format of this knowledge will be defined beforehand, specifying which items of knowledge need to be gathered.

One possible means of recording and giving access to the knowledge of an organisation is experience bases [4]. Experience bases are composed of experience packages. SE experience packages usually contain knowledge on how to use given artefacts. This knowledge must be associated with information for deciding when and where a given artefact will be useful. Experience packages are described by instantiating characterisation schemas. The information reflected by the characterisation schema is vital for effectively identifying which artefacts are useful in a given situation. But experience packages should be as compact as possible, meaning that characterisation schemas should contain the least possible information, that is, they should include the minimum set of relevant information. Nevertheless, it is not easy to find out which information these characterisation schemas should include. On the one hand, the information reflected by a characterisation schema is totally dependent on the artefact it characterises, which means that when characterising a new artefact, we cannot benefit from the fact that other artefacts have already been characterised. On the other, the theoretical foundation of the artefact in question may not be mature enough to be of assistance in deciding which information the characterisation schema should include; if we do not know the parameters that may have an influence on the behaviour of an artefact, it will be more difficult to develop a characterisation schema for it than if these parameters were known.

Here, we propose a process for identifying what information a characterisation schema should include for the purpose of building an experience base. The proposed process is empirical and iterative. It is empirical because it is not based purely on how the person who is designing the schema sees the artefact to be characterised, but also takes into account the view of potential experience base users and artefact builders. It is iterative because it begins with a preliminary schema that is refined as different views are incorporated.

The proposed process has been applied to define a characterisation schema for testing techniques. Besides the generation process, we also show the resulting characterisation schema for testing techniques, discussing the information it contains and why it has been included in the schema.

The chapter has been organised as follows. Section 2 presents a series of approaches described in the literature for developing characterisation schemas for a range of software artefacts. Section 3 discusses the proposed process for developing characterisation schemas. Section 4 is an application of the process presented in section 3 for a particular artefact: software testing techniques. Section 5 presents the evaluation of the proposed process and, finally, section 6 provides some conclusions.

2 Related Work

Although the activities of which KM is composed are clear, it is not so clear which methods should be applied within each of these activities. Indeed, while it is generally accepted that the acquired knowledge needs to be packaged [2], [15], and several proposals have been made [1], [19], [18], no one has formalised or standardised what these knowledge packages should be like, not to mention how they should be built.

Nonetheless, the use of characterisation schemas in SE as an aid for selecting different artefacts is not new. In the field of software reuse, where there is a repository of coded software modules ready for use, there is already an emerging need for characterisation schemas. In the case of reuse, characterisation schemas summarise the characteristics of the module and then, by inspecting these characteristics, a decision can be made on which module is or which modules are best suited. The characteristics encompass both the module attributes, and its application conditions and the characteristics of the operating environment. Apart from the reuse field, other areas of SE, like software architectures or software technology selection, also use characterisation schemas.

Below, we examine a series of characterisation schema proposals described in the literature, as we have not found any formalised proposal of how to develop such a schema within KM. For each proposal, we discuss the artefact it aims to characterise, the characterisation proposal, the process followed for characterisation and the information proposed for inclusion.

Prieto-Díaz [14] was the first researcher to realise the benefits of using characterisation schemas for classifying reusable artefacts. In [14], he presents a characterisation schema for reusable software modules to aid the identification and later retrieval of such modules (stored in a repository) and find the components that are less costly, in effort terms, to adapt to the current project. The schema was constructed by means of discrimination or examination 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 of the object (which represents *what*) and (2) the environment (which represents *where*).

Based on the idea that anything related to development, and not just software products, is reusable [3]. Basili and Rombach [3] present a characterisation metaschema for any software development element: products, processes, techniques, etc. Owing to the generality of this metaschema, it needs 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 schema designers, based on a reuse model, which is gradually refined 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 (the object), characteristics of the relationships between the artefact and other artefacts (interface) or environment, and characteristics of the environment in which the artefact can be used (the context or problem).

In [10], Henninger proposes 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 process followed for creating the schema is not fully explained, from which we infer that it is developed from the reflections of the schema designer. 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 the environment associated with the object. Accordingly, one can start from any of the three aspects to arrive at any of the other two.

Bass *et al.* [5] provide in a catalogue of architectural design styles, which means that the schema is already completely instantiated. The catalogue has been designed following a process of discrimination by studying and classifying numerous designs. This means that the different designs were observed and, on this basis, the authors deduced which characteristics differentiate one style from another. The catalogue contemplates not only the characteristics proper to the styles (the object), but also characteristics of the application requirements (the problem) and characteristics of the environment in which the design is to be implemented (the context), which can place restrictions on the developer when using the style.

In [7], Birk proposes a characterisation metaschema for characterising software technologies. This work is based on the fact that methods, techniques and tools are not universally applicable, and the goal is to improve the selection of technologies for use in a software project. The process followed to design the schema is not made explicit, and it is, therefore, assumed to be the result of the reflection of the schema designer. This metaschema focuses primarily on reflecting the application domain (the context) and the problem for which the technology is suited.

Similarly, von Wangenheim proposes a metaschema for characterising software engineering experiences in [19]. The author recommends asking experts on the artefact to design the schema. Therefore, the author does not discuss the information that the metaschema should contain.

Maiden and Rugg present, in [12], a schema for characterising requirements acquisition methods to improve method selection 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 produce 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.

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

- There is no proposal that sufficiently formalises the process to be followed for defining or building experience packages for a knowledge base. This process

must be defined so that other people attempting to build a knowledge base can follow it.

- The schemas are usually designed either by discriminating existing elements, asking experts (which are at least justified) or, at worst, on the basis of the personal opinions of the schema designers and are 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 the three desirable aspects: object, environment and problem. However, although they propose storing information based on developers' experiences in using the elements, they do not have an aspect that asks developers for their personal (subjective) opinions about the elements.

The process proposed here intends to overcome these problems.

3 Proposed Process for Discovering Relevant Information

Having detected the pitfalls of current characterisation schema construction processes, we propose a means of determining relevant information about any particular artefact type for inclusion in an experience repository. Sections 3.1 to 3.5 justify each stage of the proposed characterisation schema construction process. This process can be divided into two parts: schema generation and schema testing.

- *Schema Generation.* Schema generation has been divided into four different stages. They explicitly state each source of information used to formulate the schema, and each stage aims to gather different information types. The generation stages are: development of a theoretical schema, development of an empirical schema, synthesis of perspectives and expert peer review.
- *Schema Testing.* Schema testing or start up involves having two different population groups examine the schema and assess two different facets: population and use.

Fig. 1 shows the resulting process for developing the characterisation schema.

3.1 Know the Artefact: Development of a Theoretical Schema

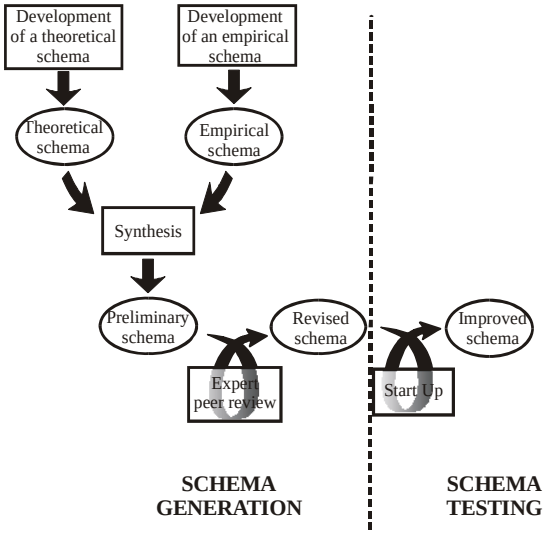
As discussed in section 2, there are two usual ways of developing characterisation schemas:

- Starting from the set of artefacts for characterisation (or as complete as possible a subset of these artefacts, if this is not feasible), analyse the similarities and

differences between the different artefacts to build a schema that contains the parameters that reflect the differences.

- On the basis of the knowledge that the people who are building the characterisation schema have of the artefact type, reflect the most prominent features of

Fig. 1. Proposed characterisation schema development process.



this artefact type that are likely to vary from one artefact to another.

Therefore, the construction of a schema is guided by deductive reasoning concerning available artefacts and what relevant characteristics they all have in common. Here, we propose to use a combination of the two strategies, aiming primarily to develop a first draft of the characterisation schema to serve as a starting-point that will be added to and improved in later iterations. A secondary goal of this stage is to familiarise the people developing the schema as much as possible with the artefacts they are trying to characterise. This is why this step is done first. This stage is, therefore, a sort of introduction to the development of what will be the final characterisation schema.

A strategy of decomposition is followed to build this theoretical schema. First, the high level information the schema should contain is identified. Then, this information is refined until an adequate level of granularity is reached.

3.2 Incorporation of Diverse Viewpoints: Development of an Empirical Schema

Our aim is to facilitate or improve the process of artefact selection in experience bases and thus contribute to the construction of higher quality software systems.

The proposed process can be considered successful if the resulting characterisation schema is used; that is, the schema should *be workable*, which means that the process must be aimed at promoting (and even guaranteeing) its use. This focus on schema use is what made us decide to get people related to the artefact area involved.

During characterisation schema design, the main decision relates to 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 be:

- Useful for consumers when selecting the artefacts for their project situation.
- Possible for producers to fill in the information asked for in the schema.

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

Therefore, the question is *What information does the consumer need to select an artefact from the experience base?* One possibility is to think about what one believes consumers would like to know when deciding on which artefact or artefacts 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 by a theory (and no such theory exists today for most SE artefacts) or is incomplete with respect to the items required to make the selection, the fitness of the resulting characterisation schema, or even its validity, could be questioned. By this reasoning, the schema generated would possibly be of little use and it would take longer to reach a satisfactory solution.

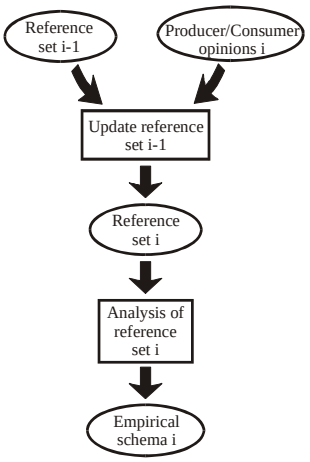
We need to be pragmatic and have the resulting schema used (in fact, this is the only way of improving artefact selection). So, in the absence of a theory that confirms why some information facilitates or is necessary for selection and other information is not, the schema should reflect the opinion of consumers and producers (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 schema designer are used as a basis upon which the opinions of the participants take shape;
2. A study is carried out to see if the theoretical and empirical opinions converge, i.e. if there is sufficient common ground between the theoretical and empirical knowledge about the subject to generate an experience base for the artefact type. 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 generate an experience base for this kind of artefacts.

The empirical schema is developed incrementally. A set of opinions (questions or information) about the information required to completely select/define an artefact is gathered for each consumer/producer surveyed. The sets of questions/information obtained are analysed incrementally. This means that the producers/consumers are gradually incorporated, making it possible to cover the total set of possible producers/consumers according to their characteristics. The process is, therefore, inductive, producing a schema containing the characteristics desired by producers and consumers.

To be more precise, the iteration for running the analysis is 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 opinions or reformulating others to make them more generic or more specific (never by deletion). Fig. 2 shows the activities to be performed to get the i-th empirical schema.

Fig. 2. Activities to get the i-th empirical schema.



One interesting point is that the characteristics of the participants should be known, as it is important to be acquainted with what type of producers/consumers are represented in the schema. Another point (not as important as accounting for all producer/consumer types) is the number of people that have to participate in this stage. The number is not essential, as Glaser and Strauss [9] 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 criterion 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 zero for at least the last 25% of subjects. Therefore, what we are examining at this stage is the evolution and change of the characterisation schema as new producers/consumers are incorporated.

3.3 Synthesis of Perspectives: Theory and Practice

As we now have two independent sets of information about the object to be characterised, they have to be merged. Accordingly, a synthesis stage will be required in which the theoretical and empirical schemas are united to produce a schema that contains the information from both.

In this stage, the two characterisation schemas created earlier (the theoretical and the empirical schemas) are taken and synthesised into a single characterisation schema to provide a single view of the information that is relevant for selection. Rules should be defined to guide this process and ensure that the schemas are synthesised in an orderly manner and no information is lost. Depending on the environment in which the schema is to operate, the synthesis rules could vary from the collection of all the information that appears in the two schemas to the selection of given types of information if performance or the amount of information handled for selection are critical factors. However, if there is no restriction on the amount of information the preliminary schema should contain, the recommended heuristic is that all information appearing in either the empirical or theoretical schema should appear in the preliminary schema. This can be translated into:

- Any information that appears in at least one schema will be directly entered in the preliminary schema.
- If there is similar information or some information is more generic or more specific than others, study the best way of adding it to the schema to assure that no information is lost during synthesis and there is no redundancy.

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

3.4 Expert Peer Review

The schema obtained after the synthesis of the theoretical and empirical schemas reflects the viewpoint of the schema designer, consumers and producers concerning the selection problem. However, 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). It would, therefore, appear to be a good idea to get someone else to inspect and give an opinion on the schema. Also, according to the principles of some sciences (for example, medicine), it is advisable to get a second opinion about a complex problem. Therefore, a series of experts in the area the artefact belongs to, should be asked to give their verdict on the preliminary schema prior to start up. The goal of this expert peer review is to correct possible schema defects caused by the way

in which it was derived. The typical 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 that particular artefact. However, the schema designer 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 is gathered indiscriminately. It may contain errors involuntarily introduced by the schema designer 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 designer.

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 and almost final schema.

The ideal number of experts for an expert peer review is as many as possible, and no less than three, so that discrepancies among experts can be handled. However, it is not easy to find experts, and therefore any number would be acceptable.

3.5 Start Up

Owing to the risk involved in deploying the characterisation schema, a preliminary evaluation must be run in order to detect possible improvements. The best way of examining product validity is to put it into operation and observe how well it fits in with development: what problems users come up against and how the product could be improved to make it useful for developers. For this purpose, once the preliminary schema has been built, it will be first instantiated for a range of artefacts, and, then, potential users of the repository (producers, consumers and librarians) will be asked to use it under several circumstances. The use of the schema will provide feedback to the schema designer, which can be used to improve it.

As mentioned before, the start-up stage consists of two parts: first, a mini repository is populated with representative artefacts from the whole population; later, this repository is used by people under different circumstances. A refined version of the schema is created on the basis of the results of the data analysis.

1. *Repository Population.* The aim of this part of the start-up stage is to examine basic schema characteristics, namely, its feasibility and flexibility from the producer viewpoint.
   For this purpose, the characterisation schema will be instantiated over again to study these aspects. The ideal situation is to have the future producers, consumers and librarians instantiate the characterisation schema for the different artefacts. However, if this is not possible, the people who created the

schema are perfectly qualified to do this job. They can act as librarians, getting the necessary information from books, papers and past projects.

2. *Repository Use*. This part of the start up involves running the repository populated during repository population. The primary aim of this part of the start-up stage is to observe the feasibility and completeness of and user satisfaction with the schema from the consumer viewpoint.
   This second part of the start-up stage is again carried out on the preliminary schema. Here, a number of subjects will act as consumers and use the schema to select artefacts. Both quantitative and qualitative data is collected during this stage, which, after analysis, will be used to again modify and improve the schema. Again, it would be desirable to have real consumers perform a pre-test of the schema. If no real consumers are available, however, other types of developers could be used (students for example).

4 Case Study: Developing a Characterisation Schema for Software Testing Techniques

The process described in section 3 has been applied to build a characterisation schema for testing techniques. The construction of this schema is described step by step throughout this section as an example for readers who are interested in applying the process for characterising any SE artefact in order to build an experience base.

4.1 Development of a Theoretical Schema

As discussed in section 3.1, the schema was developed by gradually refining the information that it is to contain. In this case, the relevant information for selecting testing techniques (schema attributes) have been grouped around the elements that are involved in software testing, which are then organised around the levels of which the testing process is composed.

4.1.1 Schema Levels

The software system testing process can be divided into the following stages:

1. Selection of the quality attributes that are to be tested, as well as the expected values for each attribute, when they are to be tested, the metrics to be used for the evaluation, and the parts of the system that will be affected by each test.
2. For each of the attributes identified in the previous stage, the tests identified above should be performed, which means: generate and execute the test cases, and evaluate the results obtained, always considering the environment where the test is to take place.

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 stage is necessary because not all software systems are the same, and a decision must therefore be made on which is the best way to evaluate each system. As regards stage 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 organisation, nor the team members, 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 optimum testing techniques selection. More formally, we have named these types of information as tactical and operational information and they correspond to two different levels.

The information contained in the *Tactical Level* is related to the initial or tactical planning that will be followed to run the tests, and reflects information related to the use to which the generated test cases will be put.

As is the case with 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, whether it 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, we should 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 it will not be useful in other circumstances.

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 technique application conditions 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.

4.1.2 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. There would appear to be two parameters:

1. The *purpose* or *objective* of the test, which defines the software attribute that is to be evaluated and how rigorously this is to be done.
   The set of cases generated when applying a 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 represent attacks or unlikely situations rather than the routine use of the system. 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 the purpose of the test will reflect: 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.

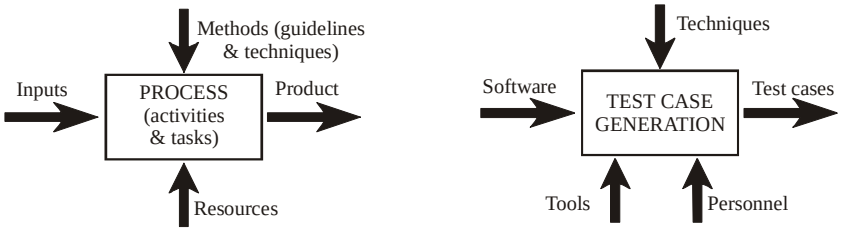
2. The *scope* of the test, which can be defined by saying what part of the software system is to be tested, when the test is to be run and the components of the software system that are affected by the test.
   Depending on which test is run, it will affect different parts of the software, ranging from 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 system development has been organised, the test will take place at one time or another within the process. We should also specify the part of the functionality offered by the system that needs to be tested. The scope, then, refers to the part of the system involved in the test.

4.1.3 Operational Level

As mentioned earlier, 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 Fig. 3 (a). As shown in Fig. 3 (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 Fig. 3 (b).

Fig. 3. Representation of the software process.



(a) (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.

Therefore, according to Fig. 3 (b), it can be said that the information that the operational level of the characterisation schema should contain has to refer to:

- 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.
- 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.
- 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.
- 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 the key for deciding on its use.
- 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.

4.1.4 Attributes of the Theoretical Schema

Table 1 shows the composition of the theoretical schema.

Table 1. Theoretical schema.

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

4.2 Development of an Empirical Schema

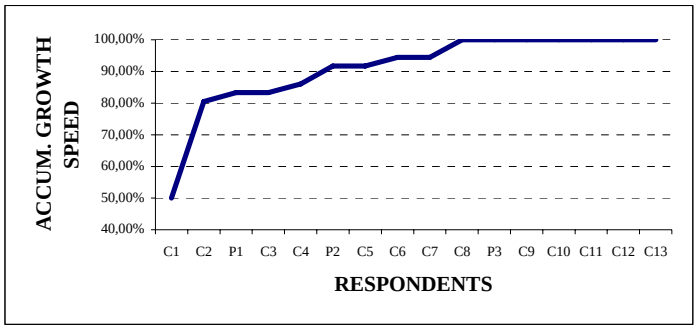
The tasks to be carried out to get the empirical schema include sending out two different questionnaires to respondents: a questionnaire that asks the consumer what information (s)he believes to be relevant for selection purposes, and another that asks the producer what information (s)he believes to be necessary to define a testing technique. The responses are 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 section 3.5. That is, the first version of the empirical schema was

generated with the information received from the first respondent and the schema version was updated as the information from successive respondents was analysed.

When working with the empirical schemas, we tried to use the levels and elements of the theoretical schema as far as possible, because the respondents only supplied attributes.

An important issue we had to deal with during this stage was the stability analysis of the empirical schema. This analysis was performed in order to find out when to stop gathering information. Fig. 4 shows the accumulated growth speed of the empirical schema. The x-axis shows the different people surveyed ordered according to time (C stands for consumer and P stands for producer) and the y-axis shows the size of the empirical schema as a percentage of its final size. It can be seen that the empirical schema reaches 50% of its final size with the first respondent. This figure increases to 80% with the second respondent, and the schema reaches its final size with the tenth respondent. This means that the last six respondents did not add any new information to the empirical schema and, therefore, the empirical schema can be considered as stable at this point.

Fig. 4. Schema accumulated growth speed.



Another 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: from a range of fields, with varying experience and of different nationality. 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. Also, respondents are asked for information starting with the ones that were most likely to give us more useful one.

Table 2 shows the contents of the empirical schema. Note that the empirical schema provides us with some information that did not appear in the theoretical schema, since practitioners care about practical issues that are very often

overlooked by theoreticians. The main differences of the empirical schema from the theoretical schema are:

- *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 questions of which this new level is composed refer to the personal experiences of people who have used the technique. This level contains two elements:
  - *Project*. The information covered in this element refers to the respondents' interest 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.

Table 2. Empirical schema.

LEVEL	ELEMENT	ATTRIBUTE
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
		Adequacy criterion
		Test data cost
		Dependencies
		Repeatability
		Sources of information
	Results	Coverage
		Effectiveness
		Type of defects
		Number of generated cases
	Object	Software type
		Software architecture
		Programming language
		Development method
		Size
Use	Project	Reference projects
		Tools used

		Personnel
	Satisfaction	Opinion
		Benefits
		Problems

  - *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.

- *Tools element*. The information covered in the tools element refers to the characteristics of the tools that can be used when applying the technique.

However, the inclusion of too much information can also lead to difficulties. Experts will play an essential role during peer review in dealing with this matter.

4.3 Synthesis

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 software projects. These are: theory, represented by the schema designer, and practice, represented by testing technique producers and consumers. 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: all information appearing in either the theoretical or empirical schema will appear in the synthesised schema. In no case has the possibility of removing information from the characterisation schema been considered at this stage. The reason is that the fact that the schema designer 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.

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

- *Equality*. Two attributes are considered equal if they bear the same name and belong to the same element and level.

- *Similarity*. Two attributes are considered 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 original two 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 have been synthesised into what will be termed hereinafter the preliminary schema and is shown in Table 3. Table 3 also shows the source of the attributes of the preliminary characterisation schema. Columns 1 to 3 show the schema itself (levels, elements and attributes). The next two columns indicate whether the information represented by an attribute is present in either of the two schemas: theoretical and empirical. Accordingly, the original composition of the two schemas can be traced back from Table 3.

Table 3. Preliminary schema.

LEVEL	ELEMENT	ATTRIBUTE	THEOR. SCHEMA	EMPIR. SCHEMA
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		■
		Adequacy criterion	■	■
		Test data cost		■
		Dependencies	■	■
		Repeatability	■	■
		Sources of information	■	■
	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		

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, except for two attributes, the empirical and the preliminary schema are practically identical. In other words, 58% of the attributes of the preliminary schema are common to the two original schemas, 5% are supplied by
the theoretical schema, and 37% by the empirical schema. This is an interesting point that 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 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 likely due to the fact that not enough people were interviewed or that the set of possible respondents was not satisfactorily covered.

4.3 Expert Peer Review

Taking into account that the experts use open-ended questionnaires, in which their 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 schema designer checks whether the opinions are contradictory or coincident and, finally, makes a decision on whether or not to accept the suggestion, and, when accepted, how it can be included. The decision on whether or not to accept the experts' suggestions is made according to a series of rules, which are now presented. Table 4 shows the results of this stage.

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. 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 expert's versus the schema designer's opinion. It is sometimes impossible to reconcile the two viewpoints, and it was decided that the opinion of the schema designer should take precedence. One such case is the suggestion to replace the attribute *cost of application* (technique) by *complexity*, as the schema designer is of the opinion that a technique can be easy and still take a long time to use. It is contradictory to make modifications in which the schema designer does not believe or about which (s)he is not sure.
4. 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.

The changes the four experts involved in the expert peer review made to the preliminary schema can be briefly 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*). This was done because the experts pointed out dependencies or redundancies with respect to other attributes.
- The *correctness* attribute of the *operational* level was replaced by another named *precision*.
- Two attributes were 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, as the experts noted that it was missing and justified its need.
- The *results* element was renamed as *test cases*.

- The *use* level was renamed as *historical* level.

4.4 Start Up

The reviewed characterisation schema has been put into practice according to the process described in Section 3, for a university environment, using final-year (sixth grade) students as consumers. The results are presented below.

4.4.1 Repository Population

The first thing to do before starting to populate the repository is to decide which techniques will be used 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 cover the variation between techniques of different families, and a number of techniques within each family, which cover the variation between techniques of the same family. Additionally, we resolved to choose well-known techniques, as this gives a better understanding of how the schema is instantiated.

Table 4. Final schema (1/2).

LEVEL	ELEMENT	ATTRIBUTE	DESCRIPTION
Tactical	Objective	Purpose	Type of evaluation and quality attribute to be tested in the system
		Defect type	Defect types detected in the system
		Effectiveness	Percentage of defects detected by the technique out of the total number of defects detected
	Scope	Element	Elements of the system on which the test acts
		Aspect	Functionality of the system to be tested
Operational	Agents	Knowledge	Knowledge required to be able to apply the technique
		Experience	Experience 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
		Cost of application	How much effort it takes 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
		Sources of information	Where to find information about the technique

Table 4 (cont.). Final schema (2/2).

LEVEL	ELEMENT	ATTRIBUTE	DESCRIPTION
Operational	Test cases	Completeness	Coverage provided by the set of cases
		Precision	How many repeated test cases the technique generates
		Number of generated cases	Number of cases generated per software size unit
	Object	Software type	Type of software that can be tested using the technique
		Software architecture	Development paradigm to which it is linked
		Programming language	Programming language with which it can be used
		Development method	Development method or life cycle to which it is linked
		Size	Size that the software should have to be able to use the technique
Historical	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
	Satisfaction	Opinion	General opinion about the technique after having used it
		Benefits	Benefits of using the technique
		Problems	Problems with using the technique

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.

The authors of this chapter were responsible for instantiating the above-mentioned techniques. Table 5 shows the results of instantiating the chosen technique for feasibility purposes: *decision coverage*.

The findings of the *schema feasibility* check were:

- There is information that is difficult to find, especially information related to reference projects. This is due to the fact that companies do not like to see their confidential data published. 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. Also, companies have to get used to doing *post-mortem* analyses of projects to weigh up the results of using the techniques.
- There were also two schema attributes (*precision* and *completeness*), whose value was not found anywhere. This casts doubts upon the advisability of these two attributes appearing in the schema. However, they are found in both the theoretical and empirical schemas and the experts did not consider them unsuitable. This appears to be relevant information that is not available in the literature on testing techniques. So, it is an omission of the testing literature, not of the schema, as this information is considered relevant from all viewpoints (note that there are not many attributes in the schema of which this can be said), but is, however, not easy to locate.
- 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. The studies carried out on testing techniques should 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 has to do with what developers would like to know and what can be easily collected [8]. 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.

Table 5. Decision coverage technique.

LEVEL	ELEMENT	ATTRIBUTE	VALUE
Tactical	Objective	Purpose	Find defects
		Defect type	Control
		Effectiveness	48%
	Scope	Element	Units
		Aspect	Any
Operational	Agents	Knowledge	Flow graphs
		Experience	None
	Tools	Identifier	LOGISCOPE
		Automation	Obtain paths
		Cost	€3,000-6,000
		Environment	Windows; C/C++
		Support	24 hour hot-line
	Technique	Comprehensibility	High
		Cost of application	Low
		Inputs	Source code
		Adequacy criterion	Control flow
		Test data cost	Medium
		Dependencies	Supplemented with techniques that find processing errors.
		Repeatability	No
		Sources of information	Sommerville
	Test cases	Completeness	--
		Precision	--
		Number of generated cases	Exponential # decisions
	Object	Software type	Any
		Software architecture	Any
		Programming language	Any
		Development method	Any
		Size	Medium
Historical	Project	Reference projects	--
		Tools used	--
		Personnel	--
	Satisfaction	Opinion	OK, but should be complemented with others
		Benefits	It is easy to apply
		Problems	Dynamic analyser should be avoided when used with real time and concurrent systems due to code instrumentation

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 can be very useful as an aid for looking for information on testing techniques. This includes information that is at present very disperse and information that is not now disseminated, like the opinion of other people who have used the technique.

As regards *schema flexibility*, it was possible to satisfactorily instantiate all the testing techniques that were originally selected. This means that we were able to instantiate the schema for 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.

4.4.2 Repository Use

Repository use aims to assess schema feasibility, completeness and user satisfaction from the consumer viewpoint.

The following project was used to check schema feasibility.

> 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 it has also 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.

When illustrating how the problem is solved, the process defined is also shown:

- *Determine bounded variables* (attributes of the schema whose value is determined by the software project and cannot be changed). 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 6 shows the associated variables for the example.

- *Pre-select an initial set of techniques.* Given the associated variables in Table 6, their value was compared with those of the technique contained in the repository. The techniques that will be selected are: *boundary value analysis*, *random*, *path coverage*, *all-possible-rendezvous*, *all-c-uses*, *all-p-uses*, *all-uses*, *all-du-paths*, *standard mutation* and *selective mutation*. The techniques *sentence coverage* and *decision coverage* will be rejected because their effectiveness is low, and the technique *threads coverage* will be discarded because it is for object-oriented software.

Table 6. Bounded variables.

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

- *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, the technique *path coverage* states that when used with concurrent and real-time systems, a dynamic analyser cannot be used as a tool. Additionally, the techniques *all-c-uses*, *all-p-uses*, *all-uses*, *all-du-paths*, *standard mutation* and *selective mutation* 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 (*boundary value analysis* and *random*), 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 for *schema feasibility* is, therefore, that it is possible to make at least one selection using the characterisation schema.

The study of *schema completeness* 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.

To assess *satisfaction* with the schema, the subjects are asked by means of open questions to subjectively summarise their perceptions of the selection process. These questions are related to: the advantages and disadvantages the subjects have seen with the schema, whether they would use it in their work if available, the

improvements they would make to the schema, what they liked and did not like about the schema, whether their view of the selection problem has changed after using the schema, what have they learned, and the suitability of the names in 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. All the subjects would be prepared to use the schema, provided they do not have to instantiate it. 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). 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.

5 Process Evaluation

Additionally, we wanted to check whether the process followed output a suitable schema and whether repository use really improves selection. For this purpose, we ran an experiment with the repository built, as described in section 4.5.1, with 87 students. For details about the experiment, see [17]. The experiment compared characterisation schema use with books used for selecting testing techniques [6], [13] and [16]. The findings are reported below.

As regards *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 has been 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.

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.

With respect to *schema completeness*, it has been observed that the schema contains more useful information for selection purposes than books. Books focus on explaining how a technique works rather than when to use it.

As regards *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 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 are well known: 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 said 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.

From all this, it can be concluded that the use of characterisation schemas improves selection and also that the proposed process helps in the construction of characterisation schemas, since it defines a systematic way of identifying relevant information.

6 Conclusions

Throughout this chapter, we have presented a process for developing characterisation schemas. As discussed in section 2, the generation of characterisation schemas is one of the most important activities for creating an experience base. We also found that no process has yet been defined for their development.

The proposed process has been applied to a particular artefact type: software testing techniques. The existence of a large group of testing techniques, the lack of pragmatic information about these techniques and the lack of a theoretical foundation makes them a paradigmatic example of the difficulties involved in building experience bases.

Thanks to the practical application of the proposed process, we have been able to demonstrate, first, the adequacy of the characterisation schema output by following the process and, second, the soundness of the process.

We operated a mini repository containing thirteen testing techniques to test the adequacy of the resulting schema. By setting up and using the repository, we were able to detect some of the possible schema defects (in this case, none).

Additionally, we ran an experiment to check the soundness of the proposed process, which compared the use of the mini repository developed from the schema with the use of testing books. From this experiment, we were able to find that the schema generated with the process proposed here contains more complete information than testing books, is easier to use, is more efficient and leads to better selections than books. Thanks to this experiment, we were also able to confirm the generic hypothesis that artefact selection improves with the use of characterisation schemas.

Going back to the more generic problem of using characterisation schemas in software engineering, it is important to note that the areas that can benefit most from these conceptual tools are the ones in which:

- There is a wide variety of elements to be characterised.
- There is knowledge to be stored.

While the first bullet represents an essential issue (at the moment there would be no point in developing a characterisation schema for selecting development paradigms, since there are only two: structured and OO), the second one represents an issue that can be somehow overcome by having researchers perform more research into the issues that are relevant for the characterisation schema (for example, inspections where there is not much knowledge). However, some knowledge must always be available about the element that is to be characterised.

References

[1] Althoff, K. D., Birk, A., Hartkopf, S., Müller W., Nick M., Surmann, D. and Tautz, C. . Systematic population, utilization and maintenance of a repository for comprehensive reuse. In Learning Software Organizations, Methodology and Applications, 11th International Conference on Software Engineering and Knowledge Engineering (SEKE'99). 2000. Springer-Verlag: Berlin. Pages. 25-50.

[2] Basili, V. R., Lindvall M. Costa, P. Implementing the experience factory concepts as a set of experience bases. In *Proceedings of the 13th International Conference on Software Engineering and Knowledge Engineering.* Buenos Aires, Argentina. 13-15 June 2001.

[3] Basili, V. R., Rombach, H. D. Support for comprehensive reuse. *Software Engineering Journal*, 303-316, September 1991.

[4] Basili, V. R., Rombach, H. D., Caldiera, G. The *Experience Factory*. Encyclopedia of Software Engineering – 2Volume Set, pages 469-476 John Wiley & Sons, Inc. 1994.

[5] Bass, L., Clements, P., Kazman, R. Bass, K. *Software Architecture in Practice.* SEI Series in Software Engineering. Addison-Wesley, January 1998.

[6] Beizer, B.. *Software Testing Techniques.* International Thomson Computer Press, Second Edition, 1990

[7] Birk, A. Modelling the application domains of software engineering technologies. *Proceedings of the Twelfth International Conference on Automated Software Engineering (ASE).* Lake Tahoe, California, November 1997.

[8] Fenton, N, Krause, P. Neil, M. Software Measurement: uncertainty and causal modeling. *IEEE Software.* 2002. 19(4): p. 116-122.

[9] Glaser, B., Strauss, A.. *The Discovery of Grounded Theory: Strategies for Qualitative Research.* Aldine de Gruyter, 1967.

[10] Henninger, S. Accelerating the successful reuse of problem solving knowledge through the domain lifecycle. *Proceedings of the Fourth International Conference on Software Reuse*, 124-133, Orlando, Florida, April 1996.

[11] Komi-Sirvio, S., Mäntyniemi, A., Seppänen, V. Toward a practical solution for capturing knowledge for software projects. *IEEE Software*, 60-62, May/June 2002.

[12] Maiden, N., Rugg, G. ACRE: Selecting methods for requirements acquisition. *Software Engineering Journal.* 11(3):183-192,1996.

[13] Pfleeger, S. L. *Software Engineering: Theory and Practice*, Mc-Graw Hill. 1999

[14] Prieto-Díaz, R. *Software Reusability*, Vol 1, Chapter 4. Classification of Reusable Modules, 99-123. Addison-Wesley, 1989.

[15] Rus, I., Lindvall, M. Knowledge management in software engineering. *IEEE Software*, 26-38, May/June 2002.

[16] Sommerville, I. *Software Engineering, 5th edition*. Pearson Education. 1998.

[17] Vegas, S. *A Characterisation Schema for Selecting Software Testing Techniques.* PhD Thesis. Facultad de Informática. Universidad Politécnica de Madrid. February 2002.

[18] von Wangenheim, C. G.. REMEX- A Case-Based approach for reusing software measurement experienceware. In *Case-Based Reasoning Research and Development (LNAI 1650).* K.D. Althoff, R. Bergmann andL.K. Branting, Editors. 1999. pp 173-187.

[19] von Wangenheim, C. G., Althoff, K. D., Barcia, R. M. Goal-oriented and similarity-based retrieval of software engineering experienceware. In *Learning Software*

Organizations: Methodology and Applications (LNCS, 1756). G. Ruhe, F. Bomarius, editors. 2000. Springer-Verlag: Berlin. Pp 118-141.

Author Biography

Dr. Sira Vegas is assistant professor with the Computer Science at the Universidad Politécnica de Madrid in Spain. She had a summer student grant at the European Centre for Nuclear Research (Geneva) in 1995. In 1997, she worked at GMV (Madrid) in the ENVISAT project for the European Space Agency. She has been a regular visiting scholar at the University of Maryland from 1998 to 2000. Sira has a BS and PhD in computer science from the Universidad Politécnica de Madrid. She is a member of IEEE Computer Society and ACM.

Dr. Natalia Juristo is full professor with the Computer Science at the Universidad Politécnica de Madrid in Spain. She is the Head of the Politecnica Master of Software Engineering degree program. Natalia has worked at the European Centre for Nuclear Research (Geneva), and at the European Space Agency (Rome). In 1992 she was Resident Affiliate at the Software Engineering Institute (Pittsburgh) on a NATO Fellowship. Natalia has a BS and PhD in computer science from the Universidad Politécnica de Madrid. She has served as Member of the Editorial Board of the IEEE Software Magazine from 1997 to 2001. She is a senior member of IEEE Computer Society and member of ACM, AAAS and NYAS.

Dr. Victor Basili is Professor of Computer Science at the University of Maryland, College Park, the Executive Director of the Fraunhofer Center Maryland, and one of the founders and principals in the Software Engineering Laboratory (SEL) at NASA/GSFC. He works on measuring, evaluating, and improving the software development process and product. Dr. Basili is the recipient of the 2000 Outstanding Research Award from ACM SIGSOFT, has authored over 160 journal and refereed conference papers, and has served as Editor-in-Chief of the IEEE TSE. He is co-editor-in-chief of the International Journal of Empirical Software Engineering, and is an IEEE and ACM Fellow.