Integrating Verification and Validation Techniques Knowledge into Software Engineering Environments

Ana Candida C. Natali, Ana Regina C. da Rocha, Guilherme H. Travassos, and Paula G. Mian

COPPE / UFRJ – Computer Science Department
Caixa Postal: 68511 - CEP 21945-970 - Rio de Janeiro – RJ, Brasil
{anatali, darocha, ght, pgmian}@cos.ufrj.br

Abstract. Accomplishing software verification and validation (V&V) activities is not a simple task. It involves a great number of techniques to choose and there is no sufficient organized information to support the selection regarding the V&V technique to be used. This paper describes an ongoing research work concerned with the definition of an approach to plan verification and validation processes. Its objective is to define how V&V activities can be supported throughout software development processes accomplished by a Software Engineering Environment (SEE). Besides V&V knowledge integrated into the SEE, this approach will also organize some practical recommendations generated by experimental studies regarding V&V techniques to support their use.

1 Introduction

The accomplishing of software verification and validation (V&V) activities is not simple. For many software development projects, half of the planed schedule is spent on software verification and validation activities [1]. Additionally, in most cases the software developers make use of technologies for which we have not enough evidence to confirm their suitability, limits, qualities, costs, and inherent risks [2].

A variety of methods and techniques for software V&V do exist. Several of them aim at detecting software defects, but there is no evidence about what defects types are better found by which technique.

Despite all the important works regarding software V&V, there is no enough grouped information about V&V methods and techniques that could allow software developers to properly decide which techniques they should use in a given context. For instance, Anderson [3] focuses on exploring the state of practice of the verification and validation process in Swedish software development organizations. Berling [4] presents an industrial case study of software verification and validation activities. Several works regarding software verification and validation techniques are described as follow:

- Study about software inspection [5];

- Study about software tests, comparing different test techniques [6];
- Study comparing different software inspections techniques [7];
- Studies characterizing V&V activities [3, 4];
- Study characterizing the use of software inspection [8].

All these works search a better understanding of V&V activities. However, they do not define direct ways to support these activities. Therefore, there is a need to help developers planning such activities for software development processes.

When we consider the software verification and validation processes, we should take into account the international standard ISO/IEC 12207[9] and the related process areas in the CMMI [10]. The purpose of ISO/IEC 12207 is to establish a common structure for defining software processes. This standard describes the V&V processes as support processes. CMMI was developed to allow the evaluation of the organizational processes, determining their capability and the maturity of the organization according to its software processes. This model defines V&V as two engineering process area belonging to maturity level 3 (if we consider the staged representation). Despite the existence of this standard and CMMI, the definition of the software processes is an activity requiring experience and it needs software engineering knowledge. Besides, ISO 12207 and CMMI describe V&V activities and practices using different denominations and detail levels, which make hard their combined use for the software process definition. We argue that knowledge management can easy their use.

So, to be able to explore knowledge regarding V&V techniques and the requirements for software processes definition presented in ISO/IEC 12207 standard and CMMI model, the developers need some sort of knowledge management support. However, there is no correlated work defining knowledge management facilities concerned with V&V activities and software processes. Some examples concerned with experimental software engineering knowledge can be represented by CeBASE (http://www.cebase.org), a center whose goal is the collection and dissemination of empirically-based software engineering knowledge, and ESERNET (http://www.esernet.org) repository, containing empirical results about the software engineering technologies effectiveness in certain organizational contexts. Despite the fact that these knowledge bases have information applicable to V&V activities, such knowledge is not described in a direct way to provide practical guidelines of its use and also, the knowledge is not integrated into the software development process, making its use workload higher.

Considering these scenarios, this work proposes an approach that intends to directly integrate V&V knowledge and knowledge management activities (like search, dissemination, use support) into a software engineering environment (SEE). Software processes definition and planning, and, the choice of the suitable V&V techniques to a specific software product should be supported by the knowledge management (KM) of a body of practical recommendations built through systematic reviews of experimental studies.

This paper is organized in five sections. The first section comprises this motivation. In section 2 we present the approach based on knowledge to support V&V activities into a SEE. In section 3 we discuss how to collect knowledge regarding V&V techniques. In section 4 we points out future work. Finally, section 5 presents final considerations.

2 Knowledge to Support V&V into a SEE

The proposed approach to support software verification and validation rely on: (1) process definition, which is the starting point to support V&V activities and tasks; (2) a software engineering environment, which aims at providing support to the whole software process, including the verification and validation processes; and, (3) experimental knowledge, which can indicate factual recommendations that can be applied to accomplish software development activities.

In the following subsections we describe the importance of which item in our approach.

2.1 Software engineering environments

An important way of getting benefits of productivity and quality in software engineering is by the use of software engineering environments. SEEs can be defined as integrated collections of tools that facilitate software engineering activities across the software lifecycle [11], providing mechanisms to integrate people in a software development organization with the software processes and with the supporting technology. A SEE provides support for the construction, management, quality assurance and maintenance of a software product, and comprises a repository that stores all information related to the software project throughout its life-cycle and a set of tools that supports technical, support and management activities.

In this work context, it has already been built a SEE, to which the support for software verification and validation will be provided. This specific SEE is categorized as an Enterprise-Oriented Software Development Environment. Enterprise-oriented SE [12] supports the activity of Software Engineering, making possible to manage knowledge that can be useful to software developers and managers when accomplishing software projects in an organization. The enterprise oriented SEE have the following goals: (i) to provide software developers with all relevant knowledge for software development held by the company, and, (ii) to support organizational learning about software development, maintenance, management and support processes.

The creation of an enterprise oriented SEE involves the definition of the software processes and the identification of the methods and tools to be provided for a specific organization. This task is accomplished by means of a meta-environment already implemented [12]. Although all these benefits, software verification and validation

are not yet properly supported in the enterprise oriented SEEs. There is no software verification and validation processes formally and completely defined in SEE and also, specific CASE tool support for the V&Vactivities is not available yet. That is, software verification and validation activities are defined and planed in the SEE based on guidelines for peer-reviews and templates for documents (peer review reports, test plans and test reports) lacking a more effective support based on experimental knowledge about the adequacy of the V&V techniques and CASE tool support.

It is necessary to formally define the V&V processes and to identify needs for tools supporting it.

2.2 Software process definition

Software process definition is not a simple task but it is a fundamental requirement to guarantee the quality of software products and to allow the definition and construction of tools. Nevertheless, the effectiveness of such processes depends on their adequacy to the characteristics of the organization, the product to be developed, and the project. In an organization, various processes can coexist to support projects with different characteristics. To guarantee the desirable discipline it is important to determine the fundamental activities that should be present in any defined process Eman [13] defines this group of fundamental elements as the standard process, or in other words, the basic process which guides the establishment of a common process in the organization. In this way, a standard process defines a single structure to be followed by all the teams involved in a software project [14] independently from the characteristics of the software product to be developed. Consequently, the definition of a standard process establishes a common structure to be used by the organization in its software projects, as it institutes the basis for the definition of all specific processes.

In the Enterprise Oriented SEE there are facilities to software process definition, specialization to a specific paradigm or software type and to process instantiation for specific projects. However, there is no standard software verification process neither standard software validation process already defined to be specialized and instantiated. These standard V&V processes need to be defined.

Therefore, we have defined these two standard processes: a verification process [15] and a validation process, both based on ISO/IEC 12207 [9] standard and CMMI [10] model. With the definition of these processes, the V&V activities were described, specifying its purpose, pre-conditions, sub activities, responsibilities and work products. The effective practice of V&V tasks requires its integration with the software development and maintenance processes. So, V&V activities are positioned into software development process.

However, the definition of a software process is not enough. As several activities of V&V processes are knowledge-intensive, only the process definition does not

provide necessary support for its accomplishment. In this context, knowledge and experience can be very useful.

2.3 Knowledge to Support V&V Activities

Software development is knowledge-intensive. Several knowledge representations and transformations are required along the software development process. Planning the software verification and validation activities involve several decisions regarding techniques and methods to select.

Vegas [16] stresses the importance of proper selection, claiming that different software systems have different aspects or characteristics to be verified or validated depending on how and for what purpose they were developed. On this basis, the selection of an unsuitable technique or method can lead to an inappropriate procedure which will bring with it an inaccurate (if not erroneous) evaluation of the software aspect being tested.

Instead of being based in mere intuition, good decisions regarding software technology need to be based on factual knowledge. A body of knowledge can indicate which are the available technologies to choose, their strengths and limitations, and under which conditions such technologies work best. Experimentation is an important way to produce this common body of knowledge regarding software technologies.

When developers use experimental knowledge they can be benefited from mature recommendations that can be applied to predict software development results. Therefore, it is necessary to organize knowledge and evidences generated by experimental studies and support their use.

However, although several experimental studies have been accomplished, some issues avoid the use of their results in practice. One problem is where developers can find useful recommendations: studies are dispersed, distributed across different sources of information. Besides, obtaining knowledge of a technology is more time demanding. It requires the running of multiples studies under different contexts, to abstract specific results into an objective body of knowledge to be used by developers.

This work stands out the need for a knowledge management infrastructure to gather, describe, store and support the use of experimental knowledge in software development organizations. Specifically, knowledge will be regarding V&V techniques.

Our goal is to directly integrate facilities for experimental knowledge management into a SEE used by an organization. It will allow a more effective support to developers when planning V&V tasks throughout the software process.

3 Capturing V&V Techniques Knowledge

We decided to capture relevant knowledge to V&V tasks and directly integrated this knowledge into SEE. In our context, SEE being used has already facilities for management the knowledge related to software processes. There are mechanisms [17] defined to support the acquisition, filtering and packing of knowledge valuable to organization, such as domain knowledge (domain theory), business knowledge (best practices, knowledge about clients and technology), past experiences knowledge (lessons learned, common problems), and organization members' knowledge acquired during process execution.

So, an infrastructure to store and retrieval knowledge is already available in SEE and it can be used to store knowledge captured during the accomplishment of V&V activities.

However the capture of relevant experimental knowledge regarding V&V tasks still has to be done, since it is not well understood which techniques can be applied in each V&V activity. And even when software developer knows which technique to apply in a specific situation sometimes he/she doesn't t know *how to use* this technique.

Our approach is to collect experimental knowledge to describe techniques and its evidence of use, indicating practical recommendations through techniques attributes, such as its purpose, benefits, limitations, experience required to use and others.

These recommendations have the purpose of helping developers in the decision of which V&V technique select to perform a particularly activity and how to use the selected technique. To do this, firstly we need to revise literature to identify which techniques can be used in each V&V activity. Secondly, we have to collect information about the technique and, then describe this knowledge appropriately to facilitate its use.

The first major step is deciding where to collect information about V&V techniques. Practical recommendations about V&V techniques should be based on experimental knowledge generated in studies, or, on evidence of its use related in experimental studies. The great number of published studies in this area implies that it may be possible to combine results to generate practical recommendations based on experimental knowledge or evidence of use. These studies and experiments are stored in public Web-based repositories like CeBASE and ESERNET, or described in research registers (journal, technical reports, conference proceedings).

Second major step is defining how to collect knowledge, and our approach is basing this collect on systematic reviews. A systematic literature review is a means of identifying, evaluating and interpreting all available research relevant to a particular research question, or topic area, or phenomenon of interest. Systematic reviews require considerably more effort than traditional reviews. Their major advantage is

that they provide information about the effects of some phenomenon across a wide range of settings and empirical methods. If studies give consistent results, systematic reviews provide evidence that the phenomenon is robust and transferable. If the studies give inconsistent results, sources of variation can be studied [18]. Kitchenham [18] propose a guideline for systematic reviews appropriate for software engineering researchers, including three main phases: Planning the Review, Conducting the Review and Reporting the Review.

The third important step is how to describe collected information in a practical manner. A characterization schema should be defined to transform information obtained through systematic reviews in practical recommendations. Vegas [16] defined a characterization schema to software testing techniques. Important attributes were included on this schema such as [16]: experience required to use the technique, how experimental or well validated the technique is, type of defects the technique helps to discover, type of software to which the technique can be applied, earlier projects in which the technique as been use and the personnel who work with it, benefits and problems of its use, and so on.

Certainty, when characterizing other types of V&V techniques, such as inspection techniques, there are common and different attributes. However, complete study should be realized to determine the set of attributes that better describe a technique, according to its type, providing support to the V&V activity that it is related.

This knowledge organized on practical recommendations will be stored in a repository, as shown in Figure 1, which will be directly integrated into the software engineering environment used by an organization.

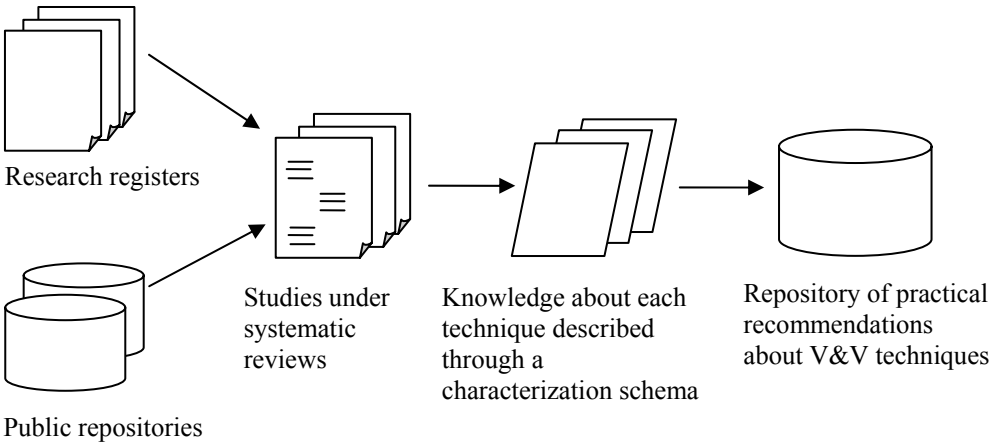

Figure 1 – Systematic reviews resulting in practical recommendations about a V&V technique's use.

4 Future Works

Enterprise oriented SEE has implemented mechanisms to knowledge management, including acquisition, filtering, packaging, and publishing resulting package knowledge in a community of practice repository through a Web-based system [17].

However, when compared to traditional KM infrastructures, the approach we propose will deal with new challenges since knowledge to be managed will be also experimental knowledge. According to [19], a traditional knowledge process involves the following steps: creation, capture, retrieval, access and use of knowledge. Services correspondents to these activities shall be arranged around an organizational memory, providing useful information to users. However, since our approach to support V&V activities will be supported by experimental knowledge, some new challenges to attend these traditional steps must be observed:

- Knowledge Creation and Capture: Knowledge needs to be created or imported. An important challenge regarding creation and capture of experimental knowledge is how to generate knowledge and evidences through many individual studies. Difficulties to combine results from multiples studies into a reliable body of knowledge are discussed in [20, 21]. Another important challenge is how to support an organization to create experimental knowledge, that is, to perform its own software engineering experiments, and aggregate its results into organizational memory.
- Knowledge Retrieval and Access: These steps satisfy the searches and queries for knowledge. An important challenge is how to describe experimental knowledge items to support retrieval and future accesses. A characterization schema of knowledge items was developed by Vegas [16]. Its proposed schema describes knowledge regarding testing techniques. However, extensions should be provided to support the representation of different types of verification and validation techniques.
- Knowledge Use: The developer will not only recall knowledge items, but will process them for further use. A challenge such as integration with the user's task plays a crucial role for the effective use of experimental knowledge. We have to associate such knowledge to the software process or even embed it into CASE tools to support the execution of software activities.
- Knowledge Maintenance and Evolution Facilities: A challenge of these steps is to define how organizational memory should aggregate results of new experimental studies and, also, how to evaluate its repository to decide what knowledge item is obsolete or which one had never been used.

6 Conclusions

This paper describes a research which aims at supporting the definition, specialization and instantiation of the V&V processes and the choice of the suitable techniques for the activities to be performed. This support is based on knowledge management of a

body of practical recommendations, built through systematic reviews of experimental studies. Our approach intends to directly integrate V&V knowledge of experimental studies into a software engineering environment.

Since the 2003, enterprise oriented SEEs are in use in 27 small and medium-size Brazilian software companies [22]. For each software company, standard processes are defined and the already available CASE tools are used to support some specific activities especially those related to CMMI maturity level 2. The results of the current use of such SEE encourage more research and the implementation of other tools.

In this scenario, the current set of users already makes possible an evaluation of the approach to support V&V activities. We intend to survey companies to characterize its actual V&V efforts. Then, we intend to evaluate the support provided by SEE with V&V processes defined. And finally, we will evaluate the benefits obtained with the use of SEEs, with V&V processes implemented with knowledge support.

Acknowledgement

The authors would like to thank CAPES and CNPq for the financial support granted. We also acknowledge the ESE (Experimental Software Engineering) group at COPPE/UFRJ (www.cos.ufrj.br/~ese) and the TABA group (www.cos.ufrj.br/~taba) for their contributions to this research.

References

1. Brooks, F. P., The Mythical Man-Month (Anniversary ed.), Addison-Wesley Publishing Company, 1995.
2. Kitchenham, B. A., Dyba, T., Jorgensen, M., Evidence-based Software Engineering, 26th International Conference on Software Engineering (ICSE 2004), Scotland.
3. Andersson, C., 2003, "Exploring the Software Verification and Validation Process with Focus on Efficient Fault Detection", Licentiate Thesis, Lund Institute of Technology (LTH), Lund University, Sweden.
4. Berling, T., 2003, "Increasing Product Quality by Verification and Validation Improvements in an Industrial Setting", Doctoral Thesis, Lund University, Sweden.
5. Aurum, A., Petersson, H., Wohlin, C., 2002, "State-of-the-Art: Software Inspections after 25 Years", Software Testing, Verification and Reliability, 12(3):133-154.
6. Juristo N., Moreno A. M., Vegas S., "A Survey on Testing Technique Empirical Studies: How Limited is Our Knowledge", Proceedings of the 1st International Symposium on Empirical Software Engineering, pp. 161-172, 2002.

7. Thelin, T., 2002, “Empirical Evaluations of Usage-Based Reading and Fault Content Estimation for Software Inspections”, Doctoral Thesis, Lund University, Sweden.
8. Laitenberger, O., Vegas, S., Ciolkowoski, M., 2002, “The State of the Practice of Review and Inspection Technologies in Germany”, Tech Report Number: ViSEK/011/E.
9. ISO/IEC 12207, 1995, Information Technology – Software Life Cycle Processes.
10. SEI, 2002, “Capability Maturity Model Integration, Version 1.1 Staged Representation”. URL: http://www.sei.cmu.edu.
11. Oliveira, K.M., Zlot, F., Rocha, A.R, Travassos, G.H., Silva, C.G.M.; Menezes, Silva C., Domain Oriented Software Development Environment, Journal of Systems and Software, v. 72, n. 2, p. 145-161, 2004.
12. Villela, K., Oliveira, K., Santos, G., Rocha, A. R., Travassos, G. H., Cordis-FBC: an Enterprise Oriented Software Development Environment. In: Workshop Learning Software Organization, Luzern, Switzerland, 2003.
13. Emam, K. E., Drouin, J. N., Melo, W., 1998, SPICE – The Theory and Practice of Software Process Improvement and Capability Determination, IEEE Computer Society, Edwards Brothers Inc., USA.
14. Maidantchik, C., Rocha, A. R. C., Xexeo, G. B., 1999, "Software Process Standardization for Distributed Working Groups". In: Proceedings of the 4 th IEEE International Software Engineering Standards Symposium, Curitiba, Paraná, Brasil, Maio.
15. Barreto, A. S., Rocha, A. R. , Apoio ao Processo de Verificação em Ambientes de Desenvolvimento de Software Orientados a Organização, WTDQS 2004, Brasília, 2004.
16. Vegas, S., Identifying the Relevant Information for Software Testing Technique Selection. ACM/IEEE International Symposium on Empirical Software Engineering (ISESE'04), Aug. 2004, CA, USA.
17. Montoni, M., Miranda, R., Rocha, A. R., Travassos, G. H. Knowledge Acquisition and Communities of Practice: an Approach to Convert Individual Knowledge into Multi-Organizational Knowledge. VI International Workshop on Learning Software Organizations (LSO 2004), Banff, Canada, June, 2004.
18. Kitchenham, B., Procedures for Performing Systematic Reviews, Technical Report, TR/SE-0401.
19. Staab, R. Studer, H. P. Schurr and Sure, Y. “Knowledge Processes and Ontologies”, IEEE Intelligent Systems, January/February, Vol. 16, No. 1, 2001.
20. Shull, F., Carver, J., Travassos, G. H., Maldonado, J. C., Conradi, R., and Basili, V. R.,"Replicated Studies: Building a Body of Knowledge about Software Reading Techniques.," in *Lecture Notes on Empirical Software Engineering,* N. Juristo and A. Moreno (eds.) ,Ed. River Edge, NJ, USA: World Scientific Publishing, October 2003, pp. 39-84.
21. Wohlin, C., Petersson, H., and Aurum, A., “Combining Data from Reading Experiments in Software Inspections: A Feasibility Study”, Lecture Notes on Empirical Software Engineering, edited by N. Juristo and A. Moreno, pp. 85-132, World Scientific, 2003.

22. Santos, G., Villela, K., Shnaider, L., Rocha, A. R., Travassos, G. H. Building ontology Based Tools for a Software Development Environment. VI International Workshop on Learning Software Organizations (LSO 2004), Banff, Canada, June, 2004.