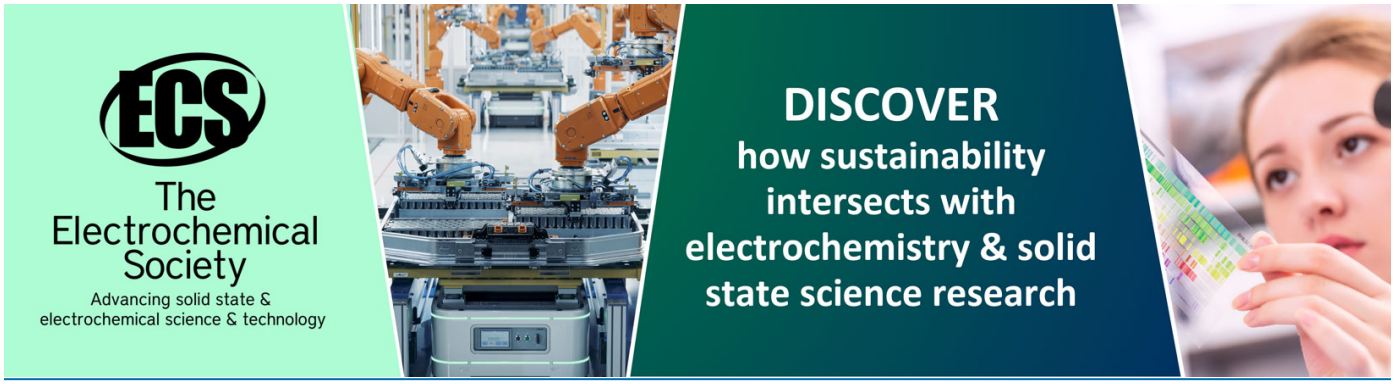
Inspection technique for the validation of software development processes



IOP Conf. Series: Materials Science and Engineering **10** (2010) 012008 doi:10.1088/1757-899X/10/1/012008

Inspection technique for the validation of software development processes

Bruno Ferreira[1] **and Gray F Moita**[2,3]
[1] Instituto Federal de Educação, Ciência e Tecnologia – Campus Formiga (IFMG), Rua São Luiz 5, São Luiz, Formiga, MG, CEP: 35.570-000, Brazil
[2] Centro Federal de Educação Tecnológica de Minas Gerais – CEFET-MG, Av. Amazonas 7675, Nova Gameleira, Belo Horizonte, MG, CEP: 30.510-000, Brazil

E-mail: bruno.ferreira@ifmg.edu.br[1], gray@dppg.cefetmg.br[2]

Abstract. Although, there are several software development processes with different characteristics in the literature, such processes are not enough to encompass the numerous fields of software applications. The practices of development differ among the developers and, sometimes, it is necessary to improve, customize or join processes already known to guarantee one needs. However, a process can only guarantee its correctness and efficiency after being thoroughly tested and validated. In the current work, a process validation model is proposed, based on the concepts of Verification and Validation (V&V), making use of the Software Inspection Technique to validate software development processes. It is presented a case study that was carried out as a first attempt of analysing the proposed model.

1. Introduction

It is well known by the whole Software Engineering community that for accompanying the creation and maintenance of more and more complex systems is necessary a Software Development Process, which enables the development of systems of high quality and reliability. This takes place in order to meet the needs of a society eager for automation of tasks, increase of control and efficiency in specific proceedings and possibility of anticipation of problems.

In spite of the advantages in the use of several development processes proposed in the literature, the introduction and application of these processes reveal to be a complex task, highly dependent on the environment in which they are inserted. As a consequence of this, a considerable number of processes has been created, improved or simply been optimized to attend the necessities of the stakeholders of the projects.

In addition to above, it is also relevant to mention the importance of a customized process to the developments in different areas, for example, computational mechanics. Needless to say that the computational mechanics community lacks a simple and straightforward - but yet reliable - process to improve the management of the everyday software development. The current work can further help the industry environment, facilitating the creation of tools and procedures to improve the techniques and\evaluation of software development process.

[3] To whom any correspondence should be addressed.

To avoid problems while creating or perfecting a process, Humphrey [1] proposes steps that must be followed to minimize possible problems, among them the Validation of the proposed process. The Validation of the Process is an important step and it can answer to a natural preoccupation of the management of projects – the guarantee that the development process guides the participants in a correct and an efficient manner during the creation of the software. Excessive bureaucracy or an ambiguous direction can disturb, instead of orientating, the development in the product creation cycle.

Considering this context, the present work presents a technique to validate a software development process, based on concepts, directives and techniques of Software Verification and Validation, minimizing the subjectivity of creation of processes. For this technique, some characteristics that are important and interesting for the validation activity are defined. Hence, the procedure must be: (1) Generic and Adaptable – so that the technique can be applied in the evaluation of different artifacts of the development process; (2) Simple – so that it is not necessary the execution of prepared activities for its application, the demand of specific types of knowledge or the allocation of great quantities of resources; (3) Extensible – so that it is easily modifiable and allows to be applied by other development processes validation techniques.

The next section introduces a short analysis on the works found in the literature on Processes Validation. The Section 3 makes an introduction to the necessary concepts and directives of Verification and Validation to the current research work. In the Section 4, a proposed validation technique is introduced, and, in the Section 5, the obtained results in the experiment of this technique are pointed out. The final considerations are discussed in the Section 6.

2. Correlated works

Validating a development process is not a simple task. One of the problems is that the validation is conceptually complex, because it refers to a wide scale of questions, very often subjective. Another problem is the absence of directives, techniques or standards, consolidated and documented, to help in the validation processes. Most of the few published researches in this area are complex and often dependents upon the context in which they were applied. Besides, the documentation was not found proving of the applicability of these validation techniques in a relevant quantity of real environments. Also, there is the absence of a technique that includes the concern to be adaptable, simple and extensible.

References [2] and [3] adopt, as the start for the processes validation, the comparison between model and practice, however in the different ways: in [3], a formalism based on the Graph Theory is used, in such an approach that the model and the practice are represented and compared; on the other hand, the method adopted in [2] to measure the differences between the model and the practice is the String Edits, where the number of insertions, exclusions and symbolic substitutions necessary (tokens) to turn a string (sequence of events of the process/practice) into other one is analyzed. For more details on String Edits please refer to [4].

The proposal of comparing the formal model of the process and its execution, as advocated by [2] and [3], appears able to produce interesting results to validate a process. Nevertheless, the use of concepts of Graph Theory or String Edits may impair the validation of agile development processes, which assume the flexibility and the easiness of use.

New processes, as the PESC proposed by [5], are created with the intention of providing quality in the development of software without the application of great quantity of human or financial resources. If the technique of validation of the process does not follow this intention, it may weaken its use. In this perspective, this work proposes a new approach to compare the formal model with the real implementation of the process, adopting concepts and techniques of verification and validation of software, to simplify and give flexibility to this comparison.

3. Software Verification and Validation

Software are broadly used to resolve problems and to take decisions. The users start to believe and to work in the results presented by them. For this, the software must be built paying much attention to the

specifications of the project and the final product must serve to the real requirements of a user. The Verification and the Validation, or simply V&V, have, respectively, these goals [6].

The V&V can be connected with the development process in several manners. The literature treats this form of relationship like paradigm. Each paradigm dictates which activities, and at which moment, they will be applied. The specification of the documents or products of entry and the presentation of the results are also defined. Two paradigms widely used are presented in [7] and [8]. They show points in common and some dissonance. This fact is common amongst several paradigms and can be seen in [8] and [9]. But, apart from the differences, all the paradigms demand that a team of professionals drives the activities of verification and validation.

Nevertheless, in spite of the differences between the paradigms, the V&V uses common techniques to carry out the activities of tests. The Table 1 shows taxonomy presented in [10], which categorizes the techniques of V&V in six perspectives. The level of mathematical formalism of each category increases from very informal in the first line of classification of the table, to very formal in the final line, implying also in the increase of the complexity.

Table 1. Taxonomy of V&V Techniques (Source: [10]).

Validation, Verification, and Testing Techniques	
Classification	Techniques
Informal	Audit, Desk Checking, Face Validation, Inspections, Reviews, Turing Test , Walkthroughs
Static	Consistency Checking, Data Flow Analysis, Graph-Based Analysis, Semantic Analysis, Structural Analysis, Syntax Analysis
Dynamic	Black-Box Testing, Bottom-Up Testing, Debugging, Execution Monitoring, Execution Profiling, Execution Tracing, Field Testing, Graphical Comparisons, Predictive Validation, Regression Testing, Sensitivity Analysis, Statistical Techniques, Stress Testing, Submodel Testing, Symbolic Debugging, Top-Down Testing, Visualization, White-Box Testing
Symbolic	Cause-Effect Graphing, Partition Analysis, Path Analysis, Symbolic Execution
Constraint	Assertion Checking, Boundary Analysis, Inductive Assertions
Formal	Induction, Inference, Lamda Calculus, Logical Deduction, Predicate Calculus, Predicate Transformation, Proof of Correctness

According to [10], the informal techniques of V&V are among the most used. They are called informal because the used tools and methods that rely strongly in the reasoning and in the human subjectivity without mathematician strict formalism. The "informal" label does not imply the any lack of structure or formal directives for the use of the techniques.

The static techniques are applied in the source code of the systems without demanding the execution of the same. The compiler of the language used in the development of the software is an example of static tool. Conversely, the dynamic techniques require the execution of the system in order to evaluate of its behavior.

The symbolic techniques also evaluate the system dynamic behavior, but they supply symbols of entry, producing expressions with the result of the transformation of these symbols, during the software run. The V&V restrictions technicians are employed to evaluate the exactness of the model using checking affirmation, analysis of limit, and inductive affirmations.

The last one, the V&V formal techniques are based on the formal mathematical proofs of exactness. According to [10], if attainable, this type of proof is the most efficient and accurate. But, these techniques, and mainly the techniques based on statistical tests, need a very big number of points of data. So, even when the hypotheses are observable and satisfied, the statistical tests are considered

significant only if the volume of data is large enough. And that, in some cases, is impracticable from the financial point of view. More details on the techniques of V&V are presented in [11], [8] and [12]. [6] present basic directives to apply V&V techniques to present satisfactory results.

4. Inspection validation processes technique

As it was exposed in the Section 3, there are several techniques that can be applied together or separately. So, it is necessary to select the most adapted for the processes validation.

According to [10], the formal techniques need mathematical data and it is the most efficient way of evaluating software. However, in a software process, mathematical data are not produced. There are in the literature works that carry out measurements of the process, but only to present possible improvements in the processes. However, the validation is carried out in the creation of the process or in an initial traineeship of its use and, in these phases, statistical data do not exist. Besides, the formal techniques are considered complex and the great worry is to select a simple technique, since a validation process must not have a degree of complexity bigger than the process (itself) that will be validated. The Dynamic, Symbolic and Restrictions techniques were, then, discarded, because they need a product (software) to be used. Here, the validation is something conceptual, due to the inherent nature of the development process.

The informal techniques are the most used. They are simple and can be applied during the whole cycle of development [10]. Following a directive defended by [6] and [8], of which the software validation must be applied during the whole cycle of development, the informal techniques consider this premise.

Among the informal techniques listed in the Section 3, the Inspection of Software stands out, due to the fact that it is a technique vastly used in several products, during the software cycle of life. The inspection is used in researches to validate documents such as tests cases and software architectural models, supplying knowledge and practical examples of use of the validation in conceptual documents [11]. More details on Software Inspection can be seen in [13], [14] and [15].

As mentioned by [11], in the software inspection, the product is analyzed against predetermined criteria of entry or against the specifications which were used to build the product. Applying this technique to the processes validation, one has the documentation of the process as predetermined criterion, in other words, the reference model, which will be compared with all the produced products. These products (artifacts) are considered the entries, so that the comparisons are done.

The technique of inspection proposed in the current paper, called VProcInsp (Validation of Processes by Inspection) consists of the adaptation of the technique of [15] that is used for the validation of software. The proposal of the technique follows the steps described by [16] and defines the type of the artifact of software processes to be evaluated; the form as the team of professionals will be defined to carry out the inspection; the technique used to evaluation and the following process by execution of the VProcInsp.

4.1. Definition of the artifacts

A product of software normally involves the understanding of the problem, the modeling of the problem, the identification of possible alternatives of solution for this problem, the analysis of these alternatives, the decision taking on which solutions they will be used in the construction of the software, the software itself and its documentation. But, as the VProcInsp proposes the comparison with the reference model, in other words, with the documentation of the process and with the artifacts produced in the execution of such a process, this formal model becomes an important product for the validation of process. Besides, the documentation of the process, the information of the software (instant messages, control of versions, compilers, among others), used daily by team of development, are also considered artifacts.

Through the analysis of logs and information of the control versions software, it is possible to guarantee that the management of configuration is being performed and the standard of the source code is being followed. With the instant message logs, it is possible to discover if technical meetings

and contacts with the client are being done. Thus, the VProcInsp "converts" the term "artifacts of software" into "artifacts of development processes".

4.2. Definition of the roles

The VProcInsp is constituted of four roles with well defined characteristics of acting. The difference between the model of inspection proposed by [15] and the current proposition is the creation of the Analyst's role. The roles are defined as following:

- Moderator – independent, individual and fair person who coordinates the planning, the preparation, the conduct of inspection and the meetings. During the validation, he secures that the inspection is efficiently driven
- Author – person or a team responsible for the formal model of the software development process. The responsibility of the author is to explain and to introduce the documentation of the process, besides explaining the objectives, peculiarities and expectations on the validation.
- Inspector – person responsible by the identifying, analyzing and comparing of the formal model of the process, with the process executed in the practice. The inspector does not have the function to find mistakes in the products, but so to analyze if what was proposed by the process is being done.
- Analyst – person or group of people, involved in the software development project, which they are conducting of the process under validation.

4.3. Technique of the Inspection used

The main characteristic of an inspection approach used here is the formalism and how it helps the inspector to find the faults. And, in order to find different techniques are utilized. [17] present the techniques Ad-hoc, Checklist and Reading Techniques.

The Reading Techniques are applied in source code or in specific artifacts and not in development processes. The *Ad-hoc* techniques do not orientate and are highly dependent on the inspectors. Hence, the VProcInsp validate processes, using as guidelines, questions defined in a checklist of evaluation. The idea is not to raise a fixed, already definite checklist, but to show a template with directives to allow for the moderator to devise the checklist in accordance with the necessities imposed by each development process. It is expected that the use of checklist as a technique of inspection will make possible to compare the formal or reference model with the real implementation of the process.

4.4. Process used during the inspection

The inspection is used to compare products and the flow of the process actually executed with the products and the flow of the formal model. For that, the model of [15] was altered and the stage of Detection was substituted by the stage of Analysis and Comparison.

Another change was made in the Planning stage. The idea is that this stage should be executed by the moderator and by the author (or authors) of the process. This joint participation allows for an author to do a presentation of the process, indicating points that must be reached, type of environment proposed for use (commercial, academic), expectations and aspirations. Figure 1 shows the model proposed by the VProcInsp. The description of this model is given below.

In the Planning, the first step is to identify a professional who will play the role of the moderator of the inspection. The Planning is similar to a Project Management, because in this activity the team, terms, costs, among other appropriate functions to the management, can be defined. The most important activity of the Planning is the configuration of the checklist for the validation inspection. To configure the checklist, the Moderator possesses the documentation or formal (reference) model of the process under validation and the support of the process author. At the end of this phase, a planning document is prepared.

In the Analysis and Comparison phase, the inspectors individually analyze the artifacts and compare them with the formal model of the process. They are guided by the checklists, identifying discrepancy between two sets of documents.

It is important to emphasize that most of the times the inspectors will have to investigate all the information produced during the development process. As mentioned previously, the term artifact in the VProcInsp is wider and involves, for example, logs and information produced by the software used during the development process. During the implementation of this phase, the inspector must register and classify the discrepancies in a report.

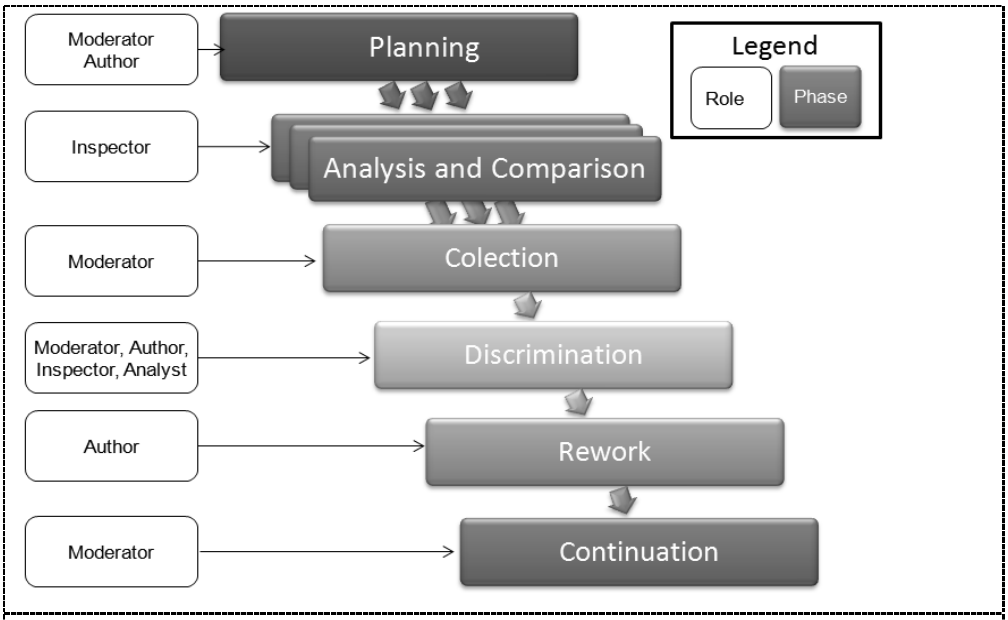


Figure 1. Model proposed by VProcInsp (Adapted from [15]).

In the stage of Collection, the Moderator of the inspection has access to all the lists of discrepancies through the Reports of Discrepancies produced by the Inspectors in the activity of Analysis and Comparison. The Moderator will then be able to select discrepancies of these lists and to discard or to classify them as replication, if there is more than one discrepancy representing the same defect. When a discrepancy is discarded, it is withdrawn of the next activities of the VProcInsp.

The Moderator, the Author and the Inspectors have participation in the phase of Discrimination. During this phase, the discrepancies are treated like topics of discussion. Each participant can add his comments relative to each one of the discrepancies, which became available as a topic of discussion, while the Moderator and the Author do not decide if the item really represents a discrepancy or not.

It is important stand out that the solutions for the discrepancies are not discussed in the Discrimination. This is a task of the author of the process in the phase of Rework, where, the author corrects the discrepancies detected during the inspection.

The Continuation consists in reviewing again the material corrected by the authors for the Moderator, who carries out an analysis of the inspection as a whole and re-evaluates the quality of the inspected product. He has the freedom to decide if a new inspection will be necessary.

5. Presentation of the results

In addition to developing the method of validation, it is important to evaluate if it considers the originally defined objectives. For this to be possible, several questions must be answered until it is possible to affirm that the VProcInsp really contains to his objectives. Among the possible questions, it is important to answer, initially, whether it is viable to apply the proposed approach in the processes validation and whether it is efficient.

For that, a case study was planned, taking into account the idea of introducing the VProcInsp in an industrial environment, which already uses a validated development process. If this is not the case, it would be questionable to apply a technique that is being evaluated at a process of development that

also needs to pass an evaluation. It is important to emphasize that to apply an immature method in a real environment with the intention of obtaining reliable results can be very dangerous.

The primary objective of the case study was to identify and to solve possible problems observed during the experiment, so that the proposed method, currently under investigation, would become more established and its results were not influenced by human variations or experimental mistakes. Thus, the purpose is to assess if the support supplied by the VProcInsp for the activity of validation of development processes is suitable and results in actual benefits.

The experiment was carried out in the TOTVS S.A. Corporation that, according to the 19th Information Technology Research carried out by the Getúlio Vargas Foundation, is the leader of the Brazilian market in packets ERP and is established in countries such as Mexico, Argentina and Portugal. The study was performed exclusively on the RM System, that has its roots in the Belo Horizonte, Brazil. The enterprise was selected because it uses a validated process that controls, with efficiency, the whole of the development systems department, managing more two hundred participants than are directly involved with the products. Besides to be two level certified by the CMMI.

The case study followed all the phases and activities, and produced all the documents proposed by the VProcInsp. In the tests, four employees played the roles specified by the VProcInsp.

The analysis phase contemplated ten software requirements. As a result, the inspectors could apply the checklist in the process, comparing the formal (reference) model with the implementation using ten requirements chosen randomly. The objective was to analyze if the requirements followed the activities or tasks proposed by the formal model. It is important to emphasize here that the amount of requirements analyzed would not be sufficient to completely validate a development process. But, once the objective of the case study is not to validate the actual process, but only to analyze the applicability and efficiency of the VProcInsp, the number of requirements was considered appropriate.

At the end of the case study, two discrepancies were discussed and confirmed. The enterprise's Engineering Software team is presently studying the viability of tuning the process based upon the obtained results. However, the initial intention was to evaluate the proposed technique, and, following the experiments, some improvements in the proposed documents were done and a questionnaire to evaluate the technique was answered by the participants.

The easiness of inspecting the process through checklists was pointed as advantageous, because, according to the answers, lets the inspection automatic and simple. In reality, the participants described the checklists as guidelines, in other words, the inspectors are guided by the checklists. It also was emphasized that the stages proposed by the VProcInsp provide good organization in the validation process.

Negative points, like the difficulty to handle the several not-computerized documents, were reported. Data could be easily modified or even deleted. The absence of an online feedback mechanism and the lack of the users control over the information and documents were also mentioned as disadvantages.

6. Conclusions

With the increase of the demand and complexity of the computational systems, the software development processes are considered fundamental to get a quality product. Nowadays, big or small enterprises and scientific institutions take advantage of the benefits acquired by the application of these processes. Nonetheless, several development processes are customized and others proposed to meet the requirements of each software factory, such as, in the present context, RM Systems and PESC. From this, the need of validating the processes to guarantee that they are readily usable, without unnecessary bureaucracies, and avoiding a more methodical orientation emerges.

During the literature review, some of the techniques for process or software validation were considered. With the characterization of these techniques, it was possible to identify its benefits and limitations, and a different validation technique was proposed. The new proposal tries to minimize the raised limitations and the lack of a generic and well documented validation processes methodology.

Therefore, it is possible to conclude what:

(a) The comparison of the formal (reference) model with the implemented process can be a good strategy to validate a process. Activities not existent in the formal model, but frequently performed, or proposed – and never used – steps can be easily perceived and show evidences that something is not correctly being followed.

(b) The use of concepts of V&V and, mainly, of the Software Inspection technique makes the validation processes systematic, so that the participants know exactly what they must do in each established stage, without compromising the simplicity.

In spite of showing signs that to compare the formal model with the process to be evaluated can be a good strategy to validate processes, several questions should still be addressed. This is due to the fact that to validate a product, process or phenomenon is conceptually complex and refer to many subjective questions.

(a) Even a process that cannot be properly followed might produce satisfactory results, or else, the practice and the process can be coincident, but the final product might not correspond to the expectations. Consequently, there is the necessity of focusing on other variables in the process of validation, for example, the satisfaction of the software developers, users and management.

(b) The efficiency can be another interesting point of analysis, where resources and time spent during the process can be analyzed and checked if it is compatible with the expected performance.

At the end, it is worth to emphasize that this is an ongoing investigation and some limitations must still be dealt with. The next steps include the application of the technique to the overall stages of a proven process, including a larger number of products and requisites. Also, it is important to apply the proposed technique in other enterprises, with different processes, so that the results can be considered definitive.

7. References

[1] Humphrey W S, 1995 *A Discipline for Software Engineering* (Reading, MA: Addison-Wesley)

[2] Cook J E and Wolf A L 1999 Software validation: quantitatively measuring the correspondence of a process to a model *ACM Trans. on Software Eng. and Methodology* **8** (2) 147–76

[3] Moor A and Delugach H 2006 Software process validation: comparing process and practice models *Contributions to ICCS 2005* Kassel, Germany 533–40

[4] Kruskal J B 1983 An overview of sequence comparison. In time warps, string edits, and macromolecules: the theory and practice of sequence comparison *Proceedings of SIAM 1983 Society for Industrial and Applied Mathematics* **25** (2) 201–37

[5] Pereira Jr M 2007 *The Conception of a Specific Development Process for Scientific Software* Master´s Dissertation. Centro Federal de Educação Tecnológica de Minas Gerais, CEFET-MG, Brazil (in Portuguese)

[6] Oberkampf W L and Trucano T G 2007 Verification and validation benchmarks *Nuclear Eng. Design, Validation and Uncertainty Estimation Department* Sandia National Laboratories, USA **238** 716–43

[7] NAS - System Engineering Manual – Version 3.1 2006 Source: http://www.faa.gov/about/office_org/headquarters_offices/ato/service_units/operations/sysengsaf/seman/SEM3.1/Section%204.14%20v3.pdf. Accessed in 30/03/2008

[8] Sargent R G 2001 Some approaches and paradigms for verification and validation simulations model *Proc. of the 2001 Winter Sim. Conf.* 50–59

[9] Knepell P L and Arangno D C 1993 Simulation validation: a confidence assessment methodology *IEEE Comp. Society Sim.*

[10] Balci O 1995 Principles and Techniques of Simulation Validation, Verification, and Testing. *Procs. of the 27 th Winter Simulation Conference* Arlington, USA 147–54

[11] Perry W 1995 *Effective Methods for Software Testing*. (John Wiley & Sons)

[12] Ferreira B and Moita G F 2008 The evaluation of different validation techniques for software development process. *Procs. of the 8th World Congress on Computational Mechanics WCCM8* Venice, Italy
[13] Sommerville I 2007 *Software Engineering*. 8th ed (Addison Wesley)
[14] Pressman R S 2006 *Software Engineering*. 6th ed (McGraw-Hill)
[15] Sauer C, Jeffery D R, Land L and Yetton P 2000 The effectiveness of software development technical review: a behaviorally motivated program of research *IEEE Trans. Soft. Engrg.* **26** 1–14
[16] Laitenberger O and Atkinson C 1998 Generalized perspective based inspection to handle object oriented development artifacts *Procs. of ICSE 99* 494–03
[17] Barcelos R F and Travassos G H 2006 Evaluation approaches for software architectural documents: a systematic review *Procs. of Ideas 2006 - 9o Workshop Iberoamericano de Ingenieria de Requisitos y Ambientes de Software* La Plata, Argentina 433–46

Acknowledgments

The authors would like to acknowledge the Brazilian sponsors FAPEMIG, CAPES and CNPq for their financial support.