An Experience on Applying Learning Mechanisms for Teaching Inspection and Software Testing

Conference Paper *in* Software Engineering Education Conference, Proceedings · May 2008

DOI: 10.1109/CSEET.2008.12 · Source: IEEE Xplore

CITATIONS
11

READS
78

3 authors:

Ellen Barbosa
University of São Paulo
208 PUBLICATIONS **1,737** CITATIONS
SEE PROFILE

Simone Do Rocio Senger de Souza
University of São Paulo
119 PUBLICATIONS **1,059** CITATIONS
SEE PROFILE

José Carlos Maldonado
University of São Paulo
382 PUBLICATIONS **4,884** CITATIONS
SEE PROFILE

An Experience on Applying Learning Mechanisms for Teaching Inspection and Software Testing

Ellen Francine Barbosa
Simone do Rocio Senger de Souza
José Carlos Maldonado
University of São Paulo (ICMC/USP), Computer Systems Department
São Carlos/SP, Brazil 13560-970
{*francine, srocio, jcmaldon*}*@icmc.sc.usp.br*

Abstract

Educational modules, concise units of study capable of integrating theoretical/practical content and supporting tools, are relevant mechanisms to improve learning processes. In this paper we briefly discuss the establishment of mechanisms to ease the development of educational modules – a Standard Process for Developing Educational Modules and an Integrated Modeling Approach for structuring their learning content. The proposed mechanisms have been investigated in the development of the ITonCode module – an educational module for teaching inspection and testing techniques. Aiming at evaluating the module we have replicated an extended version of the Basili & Selby experiment, originally used for comparing V&V techniques, now considering the educational context. The obtained results were mainly analyzed in terms of the student's uniformity in detecting existent faults, giving us very preliminar evidences on the learning effectiveness provided by the module produced.

1: Introduction

Several initiatives on using new computing technologies have been investigated in order to facilitate learning processes. Educational modules, which consist of concise units of study delivered to students by using technologies and computational resources [1, 2], is one of these initiatives. The development and application of educational modules should consider intrinsic characteristics of knowledge, such as its dynamic and evolutionary aspect. Also, there is a need for adaptability and reusability – educational modules should be seen as independent units of study, subject to be adaptable and reusable in different educational and training scenarios, according to the learner's profile, instructor's preferences, learning goals and course length, among others [2].

In a previous work, we have investigated the definition of a Standard Process for Developing Educational Modules [2]. As part of this standard, we have also proposed an integrated modeling approach for developing learning content, named *IMA-CID* (Integrated Modeling Approach – Conceptual, Instructional and Didactic) [1]. The aim in the long-term is to provide a well-defined set of guidelines and supporting mechanisms to ease the cooperative work to create, adapt, reuse and evolve the underlying processes and products, taking also into account the impact on the learning process. We also intend to provide a context for "open learning materials", which could facilitate the cooperation and use in different institutions and learning environments and effectively support new learning approaches.

All the learning initiatives, at the very end, have to be shown to be efficient and effective. Efficient in the sense that we need better process to develop use, reuse, evolve and distribute

the learning materials and environments. Effective in the sense that the learners master the involved knowledge and supporting technologies. In this perspective, we need to understand how to evaluate and provide evidences of the benefits and improvements of them, as well as their possible disadvantages. In this matter, we consider two basic research questions: (1)"*Students' attitude toward accepting non-traditional educational module is more positive than toward accepting traditional one*?"; and (2)"*If subjects were given training using non-traditionally-produced educational module would behave more uniformally, in the sense of fault detection rate, than if they were given training using traditionally-produced module*?".

Concerning the first question, in a previous study we have applied a *IMA-CID*-based testing module in a three-hour short-course for a group of about 60 CS undergraduate students. We focused on theoretical aspects, providing an introductory perspective on software testing. We investigated the students' attitude toward: (1) content, regarding the concepts, additional information, examples and exercises used in the module; (2) usability, in terms of the interface of the module; (3) navigational aspects; and (4) general aspects about the module. The results obtained provide some evidences on the practical use of the standard process and of the *IMA-CID* and its modeling mechanisms as a support to the development of effective educational modules.

The above-mentioned evidences motivated us to address the second question, carried out in the context of experimental software engineering, more specifically in the scope of lab packages for evaluating V&V techniques (inspection and software testing). We have replicated an extended version of the Basili & Selby experiment [3, 8], originally used for comparing V&V techniques, now considering the educational context.

Aiming to be more efficient we have reengineered the lab package training material on V&V techniques applying the standard process and the *IMA-CID* approach. The result was the *ITonCode* module – an educational module for teaching inspection and testing. The learning effectiveness can be evaluated by the students' ability and uniformity on: (1) detecting existent faults; (2) generating test cases; and (3) covering the test requirements.

Although in this experiment we have collected data for all the above metrics, in this paper we focus on the benefits of the educational module produced by analyzing the student's uniformity in detecting existent faults, in the sense of measuring the percentage of faults each subject identified applying the involved techniques.

The remainder of this paper is organized as follows. In Section 2, we provide a brief overview of educational modules. We also summarize the main aspects of the standard process we have proposed and describe the main aspects of *IMA-CID*. The experimental study we have conducted in order to validate the proposed mechanisms and the *ITonCode* module is discussed in Section 3. The results obtained are also analyzed. Finally, in Section 4 we summarize our contributions and discuss the perspectives for further work.

2: Developing Educational Modules

Educational modules are concise units of study, composed by theoretical content, practical content, and supporting learning environments [1, 2]:

- Theoretical content: books, papers, web information, slides, class annotations, audio, video, and so on.
- Practical content: instructional activities and evaluations, as well as their resulting artifacts (e.g., executable programs, experiments, collaborative discussions). Specific

tools in the subject domain and the related results can also be seen as practical content. Theoretical and practical content constitute the learning materials.

- Learning environments: infrastructure for delivering the learning materials.

The development and application of educational modules should consider intrinsic characteristics of knowledge, such as its dynamic and evolutionary aspect. Also, there is a need for adaptability and reusability – modules should be seen as independent units of study, subject to be adaptable and reusable in different scenarios, according to the learner's profile, instructor's preferences, learning goals and course length, among others [2].

2.1: A Standard Process for Developing Educational Modules

Similar to software, we understand that educational modules also require the establishment of systematic development processes in order to produce reliable, evolvable and quality products. The Standard Process for Educational Modules we have proposed is based on the International Standard ISO/IEC 12207, tailored to the context of educational modules by including aspects of content modeling, practices from instructional design, and issues of distributed and cooperative work. The proposed standard categorizes the defined processes into three groups. The primary processes deal with the main activities and tasks performed during the life cycle of an educational module. The supporting processes support other processes and contribute to the success and quality of the development project. The organizational processes are employed to establish, implement and improve an underlying structure made up of associated life cycle processes and personnel.

Aspects of specialization/instantiation have also been explored in order to apply the standard process into specific projects, for different knowledge domains. In the same line as *CMMI* (*Capability Maturity Model Integration*) for software development, a maturity model for educational modules – *CMMI/Educational* – has been proposed as a mechanism to support the specialization of the standard process in different maturity levels. The *ITonCode* module has been developed as an instantiation of such process. More detailed information on the standard process and on the *CMMI/Educational* can be found in [2].

2.2: IMA-CID: An Integrated Approach for Modeling Learning Content

Content modeling plays a fundamental role in the development process of educational modules. How the content is structured impacts on the reusability, evolvability and adaptability of the module. Despite its relevance, there are few approaches for modeling educational content, which considers different perspectives that are suitable for a given learning context, but inadequate for others. Motivated by this scenario, we have proposed *IMA-CID* (*Integrated Modeling Approach – Conceptual, Instructional and Didactic*) [1] – an integrated approach for modeling learning content, composed by a set of models, each one considering specific aspects of the development of learning content (Figure 1).

The conceptual model corresponds to a high-level description of the domain, representing the main concepts and the relationships among them. The instructional model characterizes what kind of additional information (e.g., facts, principles, procedures, examples, and exercises) can be used to develop learning materials. The didactic model characterizes the prerequisites and sequences of presentation among conceptual and instructional elements.

As part of content modeling, we have also introduced the idea of open specifications, which provide support for the definition of dynamic contexts of learning. Depending on

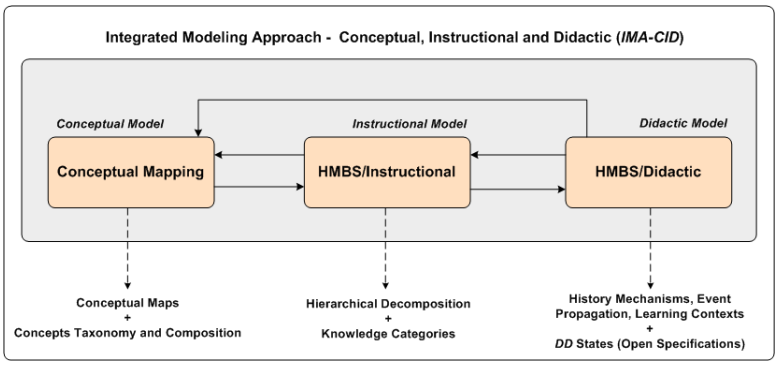

Figure 1. The *IMA-CID* Approach [1].

aspects such as audience, learning goals and course length, distinct ways for presenting and navigating through the same content can be required. An open specification allows representing all sequences of presentation in the same didactic model. So, from a single model, several versions of the same content can be generated according to different pedagogical aspects. Moreover, when an educational module is implemented based on an open specification (open implementation), its navigation paths can be defined by the user, in "execution time", based on the learner's understanding and feedback, for instance. Issues of content modeling were addressed as part of the Standard Process for Developing Educational Modules, particularly in the design activity of the development process.

3: Evaluating the Effectiveness of *ITonCode*: an Experimental Study

As already mentioned, the standard process and the *IMA-CID* approach have been applied in the development of the *ITonCode* educational module [1, 2]. The testing tools *PROTEUM* [5] and *PokeTool* [4] compose the *ITonCode* module, acting as supporting mechanisms for conducting the proposed instructional activities. The module has been developed according to the characteristics of open specification/implementation.

In order to address the second question discussed in Section 1, we have replicated an extended version of the Basili & Selby experiment [3, 8], used for comparing V&V techniques, into the educational context. The following hypotheses have been formulated:

- *H0*: There is no difference in the fault detection rates uniformity of subjects given training using non-traditionally-produced module as compared to subjects given training using traditionally-produced module.
- *Ha*: The fault detection rates uniformity of subjects given training using non-traditionally-produced module are higher compared to subjects given training using traditionally-produced module.

We have applied *ITonCode* in two one-semester undergraduate courses at ICMC/USP: *Exp*1 and *Exp*2 correspond to the experiments applied to SCE-702 (Software Testing and Inspection) and SCE-221 (Verification, Validation and Software Testing) courses, respectively. The main goal of both courses is to explore the fundamentals of V&V. The training was given in expositive classes, exploring the theoretical aspects of inspection and testing activities and related supporting tools. At the end of each class, practical exercises were

proposed. $Exp1$ and $Exp2$ involved 36 (9 teams) and 52 (13 teams) students, respectively. For the sake of space, in this paper we present only the data obtained from $Exp1$.

For developing the experimental study we used a lab package created for replication in experiments involving V&V techniques [3, 8]. From this lab package we extracted information with respect to the selected programs, existent faults, forms for data collection, and procedures for conducting the study.

Regarding the inspection activity, we used the code reading technique [9]. In the case of software testing, we used the following criteria: (1) from functional technique – equivalence classes partitioning [9]; (2) from structural technique – all-nodes [9], all-edges [9] and all-potential-uses [7]; and (3) from error-based technique – mutation analysis [6]. We also considered the idea of incremental testing – strategy in which the positive aspects of different techniques are combined in an evolutive testing process. So, in $Exp1$ were applied, in separate, code reading (inspection technique – $T1$) and mutation analysis (error-based technique – $T2$). The third technique refers to the incremental testing ($T3$) which, in this case, was the combination of functional and structural criteria, applied in this order: equivalence classes partitioning, all-nodes, all-edges and all-potential-uses.

Similarly to the experiment of Basili & Selby [3], the three V&V techniques were combined with three different programs, all of them having faults. *Cmdline* analyzes a command line for syntactic and partially for semantic correctness (268 LOC and 9 faults). *Nametbl* reads commands from a file and processes them in order to test functions which implement a symbol table for a certain computer language (270 LOC and 8 faults). *Ntree* reads commands from a file and processes them in order to test functions which implement a tree in which each node can have any number of child nodes (244 LOC and 8 faults).

Table 1 presents the definition of $Exp1$ relating teams, techniques and programs. It also illustrates the total number of faults each team have detected for each technique; this data will be analyzed in Subsection 3.1. All teams have only been trained by using the non-traditionally-produced module (*ITonCode*). In the previous studies, the subjects have only been trained by using the traditional-produced-module. Each team have applied each technique only once, always considering a different program. The idea was to assure that all techniques would be applied to all programs.

Table 1. Total Number of Valid Detected Faults for Techniques $T1$, $T2$ and $T3$.

Teams	*ntree* (8 faults)			*cmdline* (9 faults)			*nametbl* (8 faults)		
	$T1$	$T2$	$T3$	$T1$	$T2$	$T3$	$T1$	$T2$	$T3$
$G1$	-	1 (12.5)	-	-	-	5 (55.5)	3 (37.5)	-	-
$G2$	-	3 (37.5)	-	-	-	3 (33.3)	2 (25.0)	-	-
$G3$	-	5 (62.5)	-	-	-	9 (100.0)	7 (87.5)	-	-
$G4$	-	-	8 (100.0)	5 (55.5)	-	-	-	4 (50.0)	-
$G5$	-	-	0 (0.0)	3 (33.3)	-	-	-	5 (62.5)	-
$G6$	-	-	1 (12.5)	2 (22.2)	-	-	-	4 (50.0)	-
$G7$	1 (12.5)	-	-	-	4 (44.4)	-	-	-	5 (62.5)
$G8$	1 (12.5)	-	-	-	3 (33.3)	-	-	-	5 (62.5)
$G9$	5 (62.5)	-	-	-	3 (33.3)	-	-	-	3 (37.5)

3.1: Results and Data Analysis

Table 2 shows the average number of test cases generated by the teams in order to satisfy the test requirements of $T2$ and $T3$ (code reading application does not involve the generation of test cases). The test cases were manually generated based on the programs'

specification and on the test requirements of each technique. For instance, consider the program *cmdline*. To satisfy $T2$ (mutation analysis), 217.7 test cases were generated, in average, by the three teams responsible for its application.

Table 2. Average Number of Generated Test Cases by Techniques $T2$ and $T3$.

Experiment	*Technique*	*cmdline*	*ntree*	*nametbl*
*Exp*1	Mutation Analysis – $T2$	217.7	23.0	18.3
	Incremental – $T3$	38.0	14.7	17.0

Figure 2 shows the coverage obtained by the teams, in average, with respect to the application of the generated test cases against $T2$ and $T3$. The highest coverages that could be obtained for each program and each technique are also presented. For instance, consider the program *cmdline*. The highest coverage for the techniques $T2$ and $T3$ would be 100.0% and 83.3%, respectively. The teams obtained, in average, a coverage of 97.0% for $T2$ and 83.2% for $T3$. Analyzing the table, we noticed that the teams have reached adequate coverages and, in most of the cases, near to the highest possible values. Besides that, the teams have also identified equivalent mutants for $T2$ and non-executable associations for $T3$. These activities have to be manually performed, being considered difficult, specially for "testing beginners".

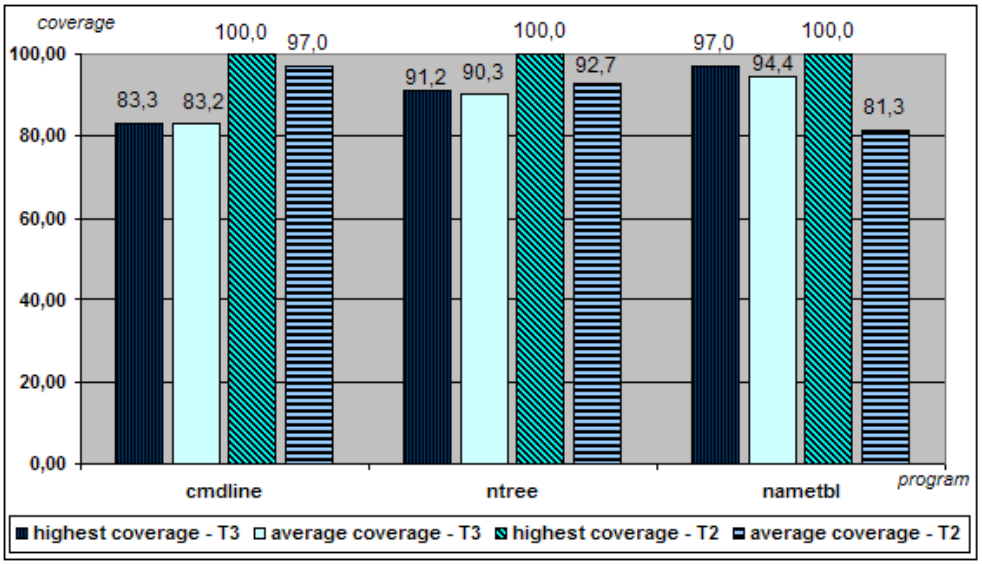

Figure 2. Coverage Analysis for Techniques $T2$ and $T3$.

Data on the total number of faults each team have detected and which of them were valid or false-positive based on the existent faults have also been collected. Figure 3 shows the average number of valid detected faults by the teams with respect to the existent faults in each program, considering techniques $T1$, $T2$ and $T3$. For instance, for *cmdline*, from the 9 existent faults, the teams responsible for applying $T1$, $T2$ and $T3$ identified, in average, 3.33, 3.33 and 5.33 faults, respectively. Notice that these values refer to the application of each technique in separate. Analyzing the techniques together, we observed that all the 9 existent faults were identified by the teams. The same analysis was performed to the other

programs; we noticed that most of the existent faults in the programs were detected when considering the application of all techniques together.

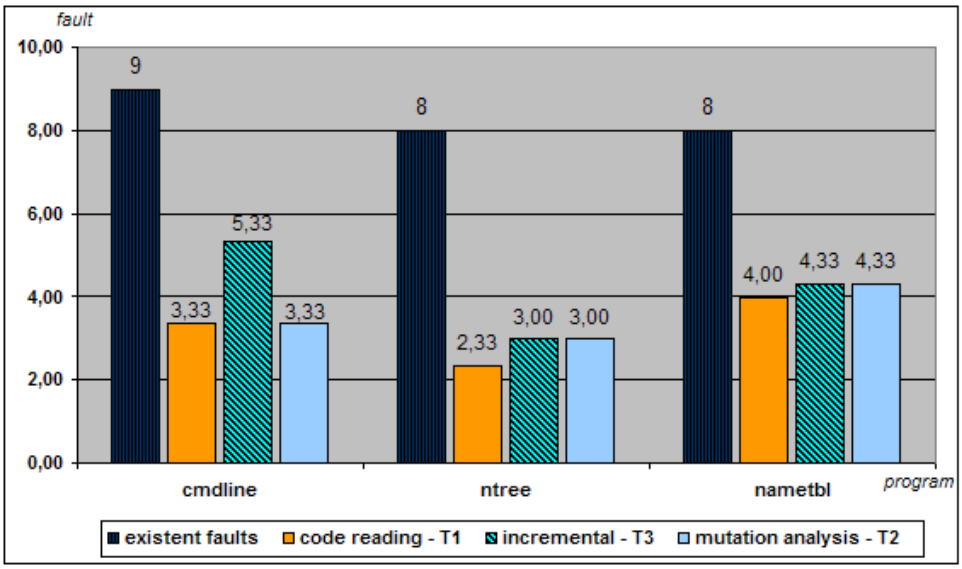

Figure 3. Average Number of Valid Detected Faults.

The data regarding the fault detection rates uniformity has been organized per team and per technique. Table 1 illustrates the total number of faults each team detected for techniques $T1$, $T2$ and $T3$. We have also analyzed the false-positives but they are not reported in this paper.

Any team with a fault detection rate over a pre-established threshold will be considered to have reached a satisfactory performance. We consider the behavior between two groups uniform if both overcome this pre-established threshold. In this study we consider a threshold of 30%; i.e., discovering over 30% of total faults.

In the previous experiments using traditionally-produced module it has been observed that for all the techniques, but for code reading, the behavior of the subjects was not uniform; e.g., only 50% of the subjects reached the threshold. From Table 1, we notice that for all the techniques, the behavior of the subjects was uniform: 55.5%, 88.8% and 77.7% of the subjects have reached the threshold for $T1$, $T2$ and $T3$, respectively. Actually, the student's uniformity in detecting faults seems to be better when using *ITonCode* module.

The obtained results provided us some very preliminar evidences that hypothesis $H0$ can be refuted, meaning that the fault detection rates uniformity of subjects given training using non-traditionally-produced module (in our case, *ITonCode*) are higher compared to subjects given training using traditionally-produced module.

Based on these results, we intend now to replicate $Exp1$ in order to compare the learning effectiveness in both scenarios: (1) training V&V techniques using traditionally-produced module; and (2) training V&V techniques using non-traditionally-produced module. The replication of $Exp1$ has been planned for the next term.

4: Conclusions and Further Work

In this paper some supporting mechanisms – a standard process and an integrated modeling approach – for developing educational modules were briefly discussed and their application was illustrated by the development of the *ITonCode* module – an educational module for teaching inspection and testing techniques.

In order to evaluate *ITonCode* and the mechanisms we proposed, we replicated an extended version of the Basili & Selby experiment [3, 8], originally used for comparing V&V techniques, now considering the educational context. *ITonCode* was applied to the students as the V&V training material for conducting the experiment. The obtained results, mainly analyzed in terms of the student's uniformity in detecting existent faults, giving us some very preliminar evidences on the learning effectiveness provided by the module.

We highlight that experiment $Exp1$ has some threats that might have an impact on the validity of the results. For instance, in the previous experiments the subjects were individuals and the time given to the training was very concentrate (around 6 hours). On the other hand, in $Exp1$ the subjects have been considered as teams of individuals and the training has been given in one-semester course (the students have more time to "mature" the V&V techniques).

Based on these results we are motivated to conduct more systematic and controlled experiments to validate our ideas. Such experiments have already been planned for the next term, involving inspection and testing courses, offered to graduate and undergraduate students at ICMC/USP as well as to professionals from local industries. Furthermore, both students and instructors' attitudes toward the module should be evaluated.

Acknowledgments

This research is supported by grants from FAPESP, CNPq and CAPES (Brazilian funding agencies).

References

[1] E.F. Barbosa and J.C. Maldonado. An integrated content modeling approach for educational modules. In *IFIP 19th World Computer Congress – International Conference on Education for the 21st Century*, pages 17–26, Santiago, Chile, August 2006.

[2] E.F. Barbosa and J.C. Maldonado. Towards the establishment of a standard process for developing educational modules. In *36th Annual Frontiers in Education Conference (FIE 2006)*, San Diego, CA, October 2006. CD-ROM.

[3] V. R. Basili and R. W. Selby. Comparing the effectiveness of software testing strategies. *IEEE Transactions on Software Engineering*, SE-13(12):1278–1296, 1987.

[4] M. L. Chaim. PokeTool – Uma ferramenta para suporte ao teste estrutural de programas baseado em análise de fluxo de dados. Master's thesis, DCA/FEEC/UNICAMP, Campinas, SP, April 1991. (In Portuguese).

[5] M. E. Delamaro, J. C. Maldonado, and A. P. Mathur. Interface mutation: An approach for integration testing. *IEEE Transactions on Software Engineering*, 27(3):228–247, March 2001.

[6] R. A. DeMillo, R. J. Lipton, and F. G. Sayward. Hints on test data selection: Help for the practicing programmer. *IEEE Computer*, 11(4):34–43, April 1978.

[7] J. C. Maldonado. *Critérios Potenciais Usos: Uma Contribuição ao Teste Estrutural de Software*. PhD thesis, DCA/FEEC/UNICAMP, Campinas, SP, July 1991. (In Portuguese).

[8] J. C. Maldonado, S. C. P. F. Fabbri, M. Mendonça, E. Dória, L. A. F. Martimiano, and J. Carver. Comparing code reading and testing criteria. In *International Symposium on Empirical Software Engineering (ISESE'06)*, pages 42–44, Rio de Janeiro, RJ, 2006.

[9] G. J. Myers, Corey Sandler, Tom Badgett, and Todd M. Thomas. *The Art of Software Testing*. John Wiley & Sons, 2nd. edition, 2004.