Establishing a Mutation Testing Educational Module based on *IMA-CID*

Ellen Francine Barbosa
José Carlos Maldonado
Department of Computer Systems – ICMC/USP
São Carlos (SP), Brazil
{francine, jcmaldon}@icmc.sc.usp.br

Abstract

Software testing is one of the most relevant activities regarding system development but, at the same time, it is a difficult topic to learn or teach without the appropriate supporting mechanisms. A learning process of software testing should involve the cooperation of theoretical and experimental knowledge with related tools in order to facilitate the apprenticeship of specific testing theories and skills. In previous works we investigated the establishment of supporting mechanisms for developing educational modules. The idea was contribute to the improvement of learning processes in general by producing quality educational products, capable of motivating the learners and effectively contribute to their knowledge construction process. In this paper we illustrate the application of our ideas in the context of software testing, more specifically by the development of an educational module for mutation testing. We consider mutation testing since it is a sound and powerful approach, but its use is still limited, mainly due to its high cost of application and, also, due to the need of an adequate learning process.

1. Introduction

Software testing is one of the most important activities to guarantee the quality and the reliability of the software under development. There are a large number of criteria available to evaluate a test set for a given program against a given specification. These criteria systematize the testing activity and may constitute a coverage measure [7]. A tester may use one or more criteria to assess the adequacy of a test set for a program and, if it is the case, enhance the test set by constructing additional test cases needed to satisfy the selected criteria.

Although a lot of work related to software testing has been carried out – definition and evaluation of testing criteria, development of supporting tools, conduction of theoretical and experimental studies –, there is a gap between the state of the practice and the state of the art in the area. Concepts and tools mastered in academia and in advanced research centers are not naturally transferred or at least tried out in real software production. Case of, for instance, mutation testing, introduced in the 70's by DeMillo et al. [7]. Although sound and powerful approach, its application in industry is still limited.

Two main reasons can explain the difficulties on applying mutation testing. The first one, which is technical, is related to the fact that mutation testing is computationally expensive, mainly due to the high number of mutants created and the effort to determine the equivalent ones. Its high cost of application, however, has motivated the proposition of several alternative mutation criteria for its application [3, 13, 15, 20, 17].

The second reason, which is more general, refers to the lack of an adequate learning process of software testing, capable of promoting both the improvement of personnel formation as well as the technology transfer between academia and industry. This envisioned learning process should involve the cooperation of theoretical and empirical knowledge with related testing tools aiming at providing appropriate support, to learners and instructors, for the apprenticeship of specific testing theories and skills.

Several initiatives on using new computing technologies have been investigated in order to facilitate the learning process in general [19, 10, 11, 4]. 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. Our goal is to explore the development and use of educational modules through the establishment of a learning process of software testing.

In a previous work, we investigated the definition of a Standard Process for Developing Educational Modules [2]. As part of this standard, we also proposed an integrated modeling approach for developing educational 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. At the very end, we 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.

We have applied and validated the feasibility of our ideas in the context of software testing. In this paper we focus on the practical application of *IMA-CID* by the development of an educational module for the mutation testing knowledge domain. Particularly, the availability of learning facilities, allied to the development of testing tools and to the conduction of experimental studies, would promote better dissemination conditions that may lead to practical evaluation and application of mutation testing.

The remainder of this paper is organized as follows. In Section 2, we provide a brief overview on educational modules, describing its components. We also summarize the standard process we proposed and describe the main characteristics of *IMA-CID*. The application of *IMA-CID* to the development of a mutation testing educational module is illustrated in Section 3. Results from a very preliminary evaluation on the effectiveness of mutation testing module are presented in Section 4. In Section 5 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 and practical content which can be delivered to learners by using technological and computational resources [1, 2]. Figure 1 illustrates an educational module with respect to its main components.

For theoretical content, instructors use books, papers, web information, slides, class annotations, audio, video, and so on. Practical content is the instructional activities and evaluations, as well as their resulting artifacts (e.g., executable programs, experiments, collaborative discussions). Specific tools related to the subject knowledge domain and the results obtained from their application can also be seen as practical content. In the case of mutation testing, for instance, *Proteum* [6] is an example of automated tool which can be integrated as part of the educational module in order to enable the application of basic concepts in realistic situations, fostering training situations and promoting exchange of technology between industry and academia.

Theoretical and practical content are integrated in terms of learning materials. Learning environments constitute the required infrastructure for delivering the educational module. For instance, presentation tools, such as *WebCT* [11], can be used to support delivery of learning materials. Collaborative tools, such as *CoWeb* [9], support collaborative

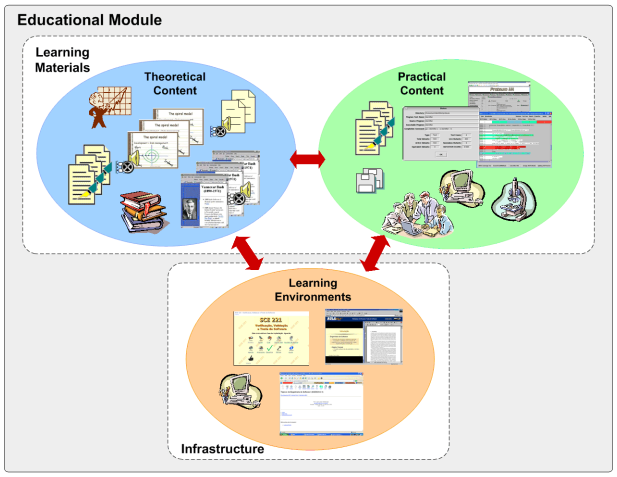


Figure 1. Components of an Educational Module

work and augment communication and discussion among instructors and learners. Capture tools, such as *e-Class* [4], provide ways to transform the content of a traditional lecture into browsable, searchable and extensible digital media that serves both short- and long-term educational goals.

The development and application of educational modules should consider intrinsic characteristics of knowledge, such as its dynamic and evolutionary aspect. In testing domain, for instance, practical activities involving the conduction of theoretical and experimental studies can result in new knowledge of testing techniques and criteria, which should be incorporated to the previously defined content. 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 previous works we investigated the establishment of supporting mechanisms for the development of educational modules: a standard process [2] and a content modeling approach [1]. Next, we provide a brief overview on them.

2.1. A Standard Process for Developing Educational Modules

Similar to software, 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 [2] is based on the International Standard ISO/IEC 12207, tailored to the context of educational modules by including aspects of content modeling [14], practices from instructional design [8], and issues of distributed and cooperative work [12]. Basically, a set of processes that can be employed to acquire, supply, develop, deliver, operate, and maintain educational modules is established. Three categories of processes are defined: (1) primary processes; (2) supporting processes; and (3) organizational processes.

The standard process is responsible for the establishment of a unique development structure to be adopted and followed by the entire organization [2]. However, changes in organizational procedures, educational paradigms and principles, learning requirements, development methods and strategies, as well as the size and complexity of the projects, among other aspects, impact the way an educational module is produced. To be used into particular projects, the processes should be defined case by case, taking into account the specific features of each project.

Process specialization and instantiation have also been explored in order to apply the standard process into specific learning environments and organizations. The definition of a process for developing a given educational module should consider its adequacy to: (1) the involved technologies, supporting mechanisms and economical resources; (2) the domain of the educational application; (3) the characteristics of the module; (4) the maturity level of the development team; and (5) the characteristics of the organization. As a result, processes into different levels of abstraction can be defined. More detailed information can be found in [2].

2.2. IMA-CID: An Integrated Approach for Modeling Educational Content

Content modeling plays a fundamental role into the development process of educational modules [14]. The way 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, each of them dealing with different modeling perspectives that can be suitable for a given learning scenario and inadequate for others. Motivated by this scenario, we proposed *IMA-CID* (*Integrated Modeling Approach – Conceptual, Instructional and Didactic*) [1] – an integrated approach for modeling educational content. *IMA-CID* is composed by a set of models – conceptual, instructional and didactic –, each one considering specific aspects of the development of educational content (Figure 2).

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. *IMA-CID* and its related models were adopted in the instantiated version of the standard process used for developing the mutation testing educational module. The main steps of *IMA-CID* application in the context of mutation testing are illustrated next.

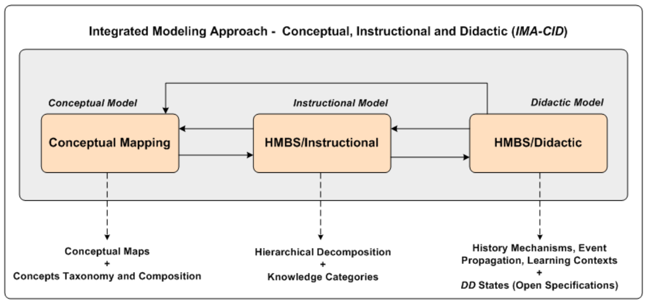

Integrated Modeling Approach - Conceptual, Instructional and Didactic (IMA-CID)
Conceptual Model
Instructional Model
Didactic Model
Conceptual Mapping
HMBS/Instructional
HMBS/Didactic
Conceptual Maps + Concepts Taxonomy and Composition
Hierarchical Decomposition + Knowledge Categories
History Mechanisms, Event Propagation, Learning Contexts + DD States (Open Specifications)

Figure 2. The *IMA-CID* Approach

3. Applying *IMA-CID* to the Development of a Mutation Testing Educational Module

The standard process and the *IMA-CID* approach were applied in the development of a software testing educational module [1, 2] . The produced module – *Software Testing: Theory and Practice*[1] – addresses a set of relevant topics of software testing, such as quality assurance, basic testing concepts, testing steps, test documentation, testing phases, testing techniques, testing tools, theoretical and experimental studies, testing perspectives, among others.

In this work we focus on the *IMA-CID* models developed for a particular subject of the testing techniques topic – the mutation testing [7]. Figure 3 provides a brief overview on the main components of the mutation testing module and illustrates the integration among them. Specific information on mutation (concepts, facts, principles, procedures, examples and exercises) was identified, modeled and implemented as a set of slides, integrated to HTML pages, text documents, learning environments and testing tools.

3.1. Conceptual Model

The *Conceptual Model* established by *IMA-CID* consists in a high-level description of the knowledge domain, representing its main concepts and the relationships among them. The relationships can be divided into two classes. Structural relationships are useful to set up taxonomies among concepts and make inferences about the knowledge domain, representing a generic category of relationships that can be applied to any kind of knowledge domain. Relations such as *type-of* and *part-of* are examples of structural relationships. On the other hand, domain-specific relationships are user-defined and have their meaning associated to a particular subject, carrying their own semantics. That is, they represent specific relations, whose interpretation depends on the domain being modeled.

To construct the conceptual model we focused on the conceptual mapping ideas [16]. This technique has been

[1]The module is in Portuguese.

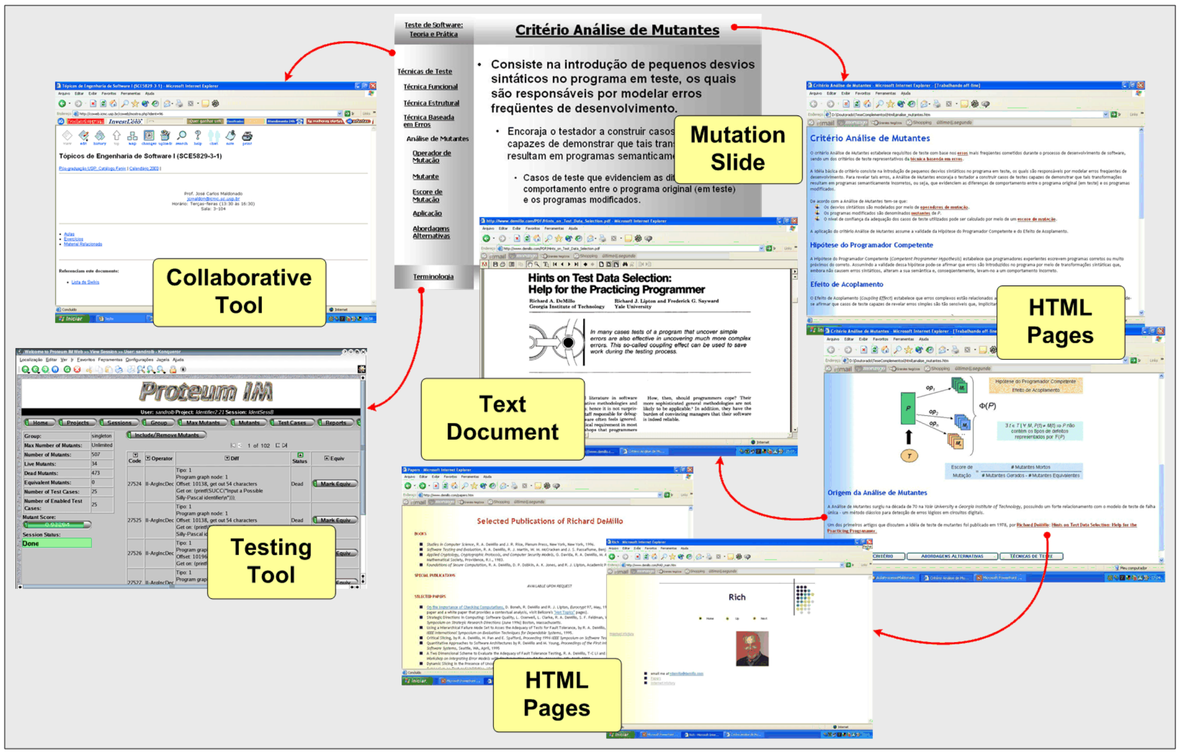

Mutation Slide
Collaborative Tool
Testing Tool
Text Document
HTML Pages
HTML Pages
Critério Análise de Mutantes
Proteum IM
Hints on Test Data Selection: Help for the Practicing Programmer

Figure 3. Mutation Testing Educational Module

adopted by the majority of existent modeling approaches for educational content. We also included some additional notations to the rules for creating conceptual maps aiming at representing the relationships of concepts taxonomy (*type-of*) and concepts composition (*part-of*).

Figure 4 shows the conceptual model for mutation analysis [7]. From the model, we can infer: (1) mutation analysis is one of the testing criteria of the error-based technique; (2) mutation analysis assumes the principles of competent programmer hypothesis and coupling effect; (3) mutation analysis uses information about typical errors, which are modeled by mutant operators in order to generate the mutant programs; (4) mutation analysis has a mutation score associated with; (5) mutants characterize syntactic transformations of programs and can be classified as error-revealing, dead or equivalent ones; (6) syntactic transformation of programs can result on equivalence of programs, characterizing an equivalent mutant; (7) randomly mutation, selective mutation and constrained mutation are variants of mutation analysis; and (8) constrained mutation can be used in the establishment of sufficient mutant operators sets, which represent subsets of mutant operators. It is important to highlight the subjective aspect of the conceptual modeling activity. In fact, different conceptual models can be developed for the same knowledge, depending on the domain expert and/or designer responsible for its construction.

3.2. Instructional Model

Based on the conceptual model, the main concepts of the knowledge domain can be specified. Besides them, information items and instructional elements should also be considered as part of the knowledge domain. In the *Instructional Model* we are interested in defining such additional information, relating them to the concepts we have previously identified. Notice we are not interested in how the information will be associated, but in what kind of information we can use to develop more significant and motivating educational content. Several theories and techniques can be referred to support the establishment of information items. As previously discussed, we specify as information items four types of elements: (1) concepts; (2) facts; (3) procedures; and (4) principles. For the sake of illustration, consider the following examples as some of the relevant information to be addressed in a mutation testing module:

- Concepts: definitions of mutant (dead, equivalent, error-revealing), mutant operator, mutation score, and so on;
- Facts: when mutation testing was proposed and by whom;
- Principles: competent programmer hypothesis and coupling effect; and
- Procedures: basic steps for applying mutation testing.

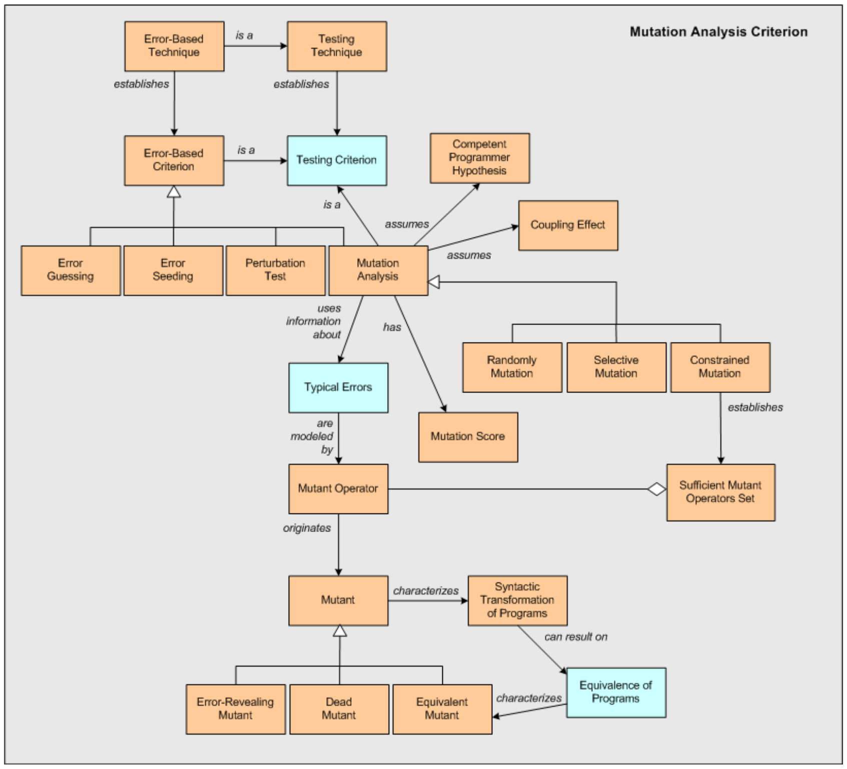


Figure 4. Conceptual Model for Mutation Analysis

Regarding the instructional elements, they can be classified into three categories: (1) explanatory; (2) exploratory; and (3) evaluative. The explanatory elements, such as examples, hints, and suggestions of study, address the complementary information used for explaining a given topic. They can play a number of different roles depending on their purpose. An example, for instance, can be associated according two distinct perspectives – to motivate the study of the topic, or to illustrate its use. The exploratory elements allow the learner to navigate through the domain, practising the concepts and the other relevant information. Guided exercises, simulations and hands-on assignments are representative of this category. The evaluative elements allow to assess the learner's proficiency on the domain. Diagnostic, formative and somative evaluations, in terms of subjective and/or objective questions, are examples of evaluative elements. In the context of mutation testing, for instance, examples could be excerpts of mutants generated by specific operators, while exercises and evaluations could be proposed tasks involving the application of mutation testing through automated tools.

As a support to construct the instructional model, we adopted the HMBS (Hypertext Model Based on Statecharts) model [18]. In short, HMBS uses the structure and execution semantics of statecharts to specify both the structural organization and the browsing semantics of hyper-documents. At the instructional level, we just focused on the mechanisms for hierarchical decomposition HMBS provides, complementing the idea of hierarchical organization already explored in the conceptual model. Actually, the instructional model keeps the hierarchical structure of statecharts, letting the behavioral aspects to be addressed in the next level, the didactic one. Notice that the hierarchy deals with the depth of the knowledge/material to be presented. In order to make HMBS suitable for modeling the instructional aspects, it was extended for representing different *knowledge categories*. So, the model allow to represent: concepts, information items (facts, principles and procedures), and instructional elements (explanatory, exploratory and evaluative). The extended version of HMBS is named *HMBS/Instructional*.

Figure 5 shows the instructional model, constructed according to the *HMBS/Instructional*, for the mutation analysis criterion. Some slides illustrating how the representation of different knowledge categories are implemented into the module are also presented. At first, we identify all relevant information that can be associated to the concepts previously represented, i.e., the information items. Consider, for instance, the `MutationAnalysis` state. In addition to the concept itself (`MA:concept:text`), some related facts (`MA:fact:text`) and principles (`CompetentProgrammer:principle:text` and `CouplingEffect:principle:text`) are specified. Regarding the `Application` state, a procedure (i.e, the basic steps for applying mutation analysis) is specified considering both textual (`ApplicationMA:procedure:text`) and graphical (`ApplicationMA:procedure:figure`) representations. The first and second slides of Figure 5 illustrate this categorization. Similarly, other concepts and more detailed information on mutation analysis are modeled into separated states: `MutantOperator`, `MutantGeneral`, `MutationScore`, `ApproachesGeneral`, and so on.

Next, we determine the instructional elements. In the case of the mutation testing educational module, only explanatory and exploratory elements were used. As explanatory elements consider, for instance, the states `Mutant:example:figure`, `DeadMutant:example:figure`, `EquivalentMutant:example:figure` and `ErrorRevealingMutant:example:figure`. These states represent examples related to the concepts of a mutant program (`Mut:concept:text`), a dead mutant (`DeadMutant:concept:text`), an equivalent mutant (`EquivalentMutant:concept:text`) and an error-revealing mutant (`ErrorRevealingMutant: concept:text`), respectively. Similarly, these examples are related to another explanatory element (`IdentifierImplementation: complementary:figure`), which provides complementary information for them.

In terms of exploratory elements, consider the exercise represented by the state `ApplicationMA:exercise:text`. Basically, it consists in the application of the mutation analysis criterion for testing a given program (in our module, the factorial one). Explanatory elements, included in order to provide some help for solving the exercise, are modeled by the states `FactorialImplementation:complementary: figure` and `FactorialHintMA:complementary: figure`. Some required tools for doing the exercise are modeled too. The `Coweb:tool` state represents a collaborative learning environment (*CoWeb* [9]), used as a discussion space among learners and instructors. The `ProteumIM:tool` state corresponds to a specific testing tool (*Proteum* [6]), used for applying the mutation analysis criterion. The third slide of Figure 5 illustrates the proposed exercise regarding mutation analysis application.

3.3. Didactic Model

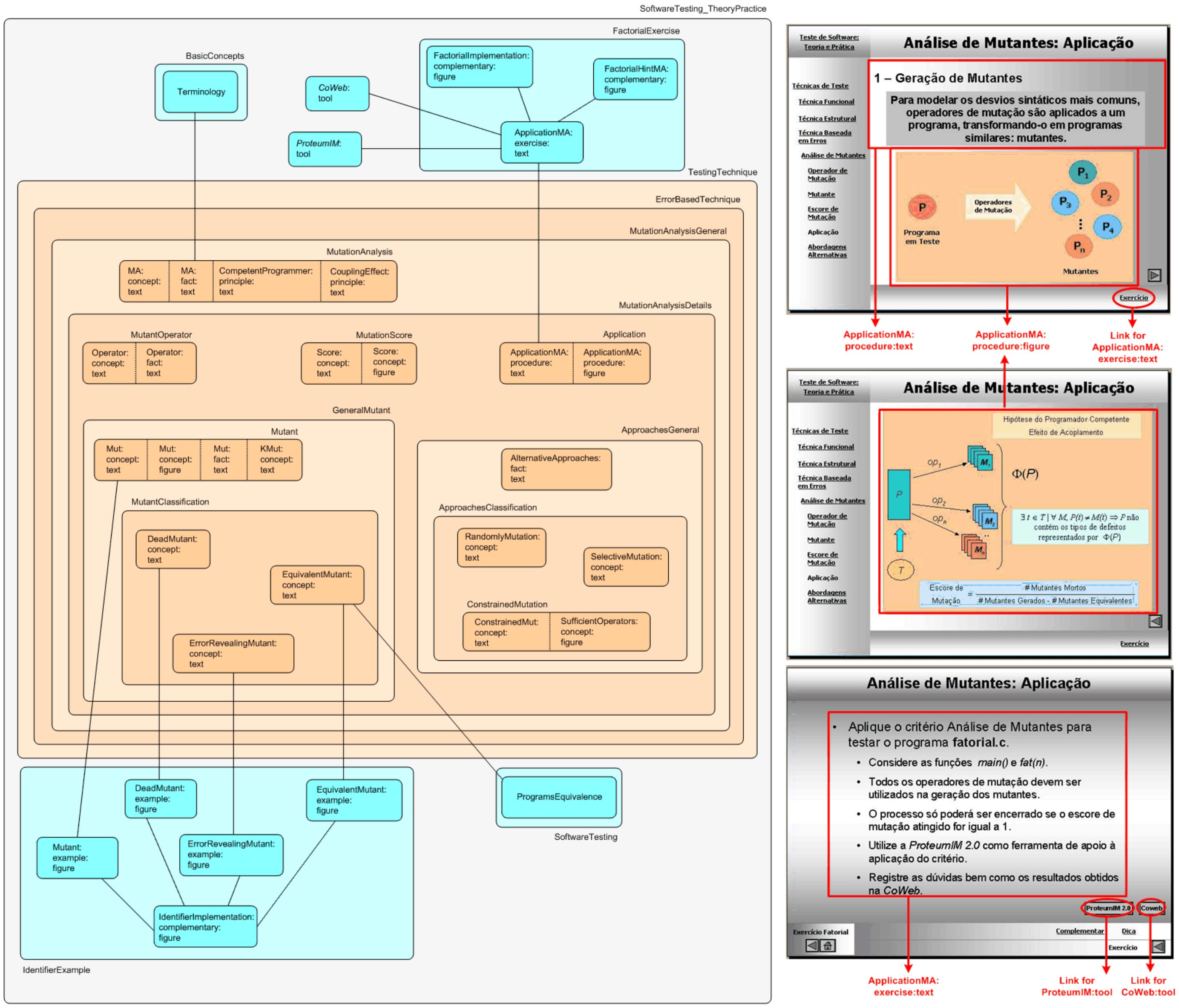
The *Didactic Model* established by *IMA-CID* is responsible for defining prerequisites and sequences of presentation among the objects characterized in the conceptual and instructional models. Moreover, the specification of behavioral aspects can be explored by means of a didactic model. The model can be used to illustrate the way the didactic space is modified while being navigated by the user, i.e, which information become active/deactive when a given path is traversed. Also, didactic models are useful to represent dynamic contexts of learning, where the elements of the content are determined according to specific parameters, defined in terms of the characteristics of the course, learners and instructors.

Since HMBS deals with relevant aspects under the didactic perspective (history mechanisms, event propagation and learning contexts definition), it was also adopted for the didactic model. In fact, we keep both the hierarchical structure and the behavioral aspects provided by statecharts. Additionally, by using HMBS we can validate the educational content through the analysis of the subjacent statechart properties [18]. As an extension to HMBS at the didactic level, we introduced the idea of *open specifications*, providing support for the definition of dynamic contexts of learning. Depending on 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 to represent all sequences of presentation in the same didactic model. 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” – the user is able to dynamically decide which topics of the module should be navigated and in which sequence, for instance, based on the feedback of the learners.

Aiming at representing open specifications, we extended HMBS with the notion of *DD* (Dynamically Defined) *states*. In short, all OR substates of a *DD* state (OR_{DD}) are totally connected to each other, i.e., from any substate of a *DD* state *X*, we can reach all other substates of *X*. For the sake of legibility, transitions and events are implicitly represented. We also established an hierarchy of *DD*-superstates – leaving a *DD* state *X* can active the OR_{DD} states from the hierarchy of *DD*-superstates of *X*. Both the notion of DD states as well as the hierarchy of *DD*-superstates help us to the establishment of open specifications in the sense they allow to represent all sequences of presentation in the same didactic model. The extended version of HMBS to support *DD* states (and open specifications) is named *HMBS/Didactic*.

Figure 6 illustrates part of the didactic model[2], constructed according to the *HMBS/Didactic*, for the mutation analysis criterion. It corresponds to an *open specification*, in which all possible sequences of presentation among the modeled objects are represented. Consider, for instance, the

[2]For the sake of space, explanatory and exploratory were not considered.


SoftwareTesting_TheoryPractice
FactorialExercise
BasicConcepts
Terminology
CoWeb: tool
FactorialImplementation: complementary: figure
FactorialHintMA: complementary: figure
ProteumIM: tool
ApplicationMA: exercise: text
TestingTechnique
ErrorBasedTechnique
MutationAnalysisGeneral
MutationAnalysis
MA: concept: text
MA: fact: text
CompetentProgrammer: principle: text
CouplingEffect: principle: text
MutationAnalysisDetails
MutantOperator
Operator: concept: text
Operator: fact: text
MutationScore
Score: concept: text
Score: concept: figure
Application
ApplicationMA: procedure: text
ApplicationMA: procedure: figure
GeneralMutant
Mutant
Mut: concept: text
Mut: concept: figure
Mut: fact: text
KMut: concept: text
ApproachesGeneral
AlternativeApproaches: fact: text
MutantClassification
DeadMutant: concept: text
EquivalentMutant: concept: text
ErrorRevealingMutant: concept: text
ApproachesClassification
RandomlyMutation: concept: text
SelectiveMutation: concept: text
ConstrainedMutation
ConstrainedMut: concept: text
SufficientOperators: concept: figure
ProgramsEquivalence
SoftwareTesting
DeadMutant: example: figure
EquivalentMutant: example: figure
Mutant: example: figure
ErrorRevealingMutant: example: figure
IdentifierImplementation: complementary: figure
IdentifierExample
Teste de Software: Teoria e Prática
Técnicas de Teste
Técnica Funcional
Técnica Estrutural
Técnica Baseada em Erros
Análise de Mutantes
Operador de Mutação
Mutante
Escore de Mutação
Aplicação
Abordagens Alternativas
Exercício
Programa em Teste
Operadores de Mutação
Mutantes
ApplicationMA: procedure:text
ApplicationMA: procedure:figure
Link for ApplicationMA: exercise:text
Hipótese do Programador Competente
Efeito de Acoplamento
$\Phi(P)$
$\exists t \in T \mid \forall M, P(t) \neq M(t) \Rightarrow P$ não contém os tipos de defeitos representados por $\Phi(P)$
Escore de Mutação = # Mutantes Mortos / # Mutantes Gerados - # Mutantes Equivalentes
Exercício Fatorial
ProteumIM 2.0
Coweb
Complementar
Dica
ApplicationMA: exercise:text
Link for ProteumIM:tool
Link for CoWeb:tool

Análise de Mutantes: Aplicação

1 – Geração de Mutantes

Para modelar os desvios sintáticos mais comuns, operadores de mutação são aplicados a um programa, transformando-o em programas similares: mutantes.

Análise de Mutantes: Aplicação

Análise de Mutantes: Aplicação

- Aplique o critério Análise de Mutantes para testar o programa **fatorial.c**.
  - Considere as funções *main()* e *fat(n)*.
  - Todos os operadores de mutação devem ser utilizados na geração dos mutantes.
  - O processo só poderá ser encerrado se o escore de mutação atingido for igual a 1.
  - Utilize a *ProteumIM 2.0* como ferramenta de apoio à aplicação do critério.
  - Registre as dúvidas bem como os resultados obtidos na *CoWeb*.

Figure 5. Instructional Model / Slides for Mutation Analysis

MutationAnalysisDetails state. By exploring the notion of DD states, the MutationAnalysisDetails substates (OR_{DD} states) – MutantOperator, MutantGeneral, MutationScore, Application and ApproachesGeneral – are all connected to each other by implicit transitions, which are responsible for establishing the navigation paths among them. So, from MutantOperator we can get to the states MutantGeneral, MutationScore, Application and ApproachesGeneral (and vice versa). Similarly, consider the Mutant state. From Mutant we are able to get to MutantClassification (and vice versa). Actually, both states are substates of MutantGeneral (DD state) and, in this sense, they are connected to each other by means of implicit transitions.

We can also explore the idea of an hierarchy of DD-superstates. For instance, consider the sequence (MutantGeneral, MutationAnalysisDetails, MutationAnalysisGeneral, ErrorBasedTechnique, TestingTechnique, SoftwareTesting_TheoryPractice) as the hierarchy of DD-superstates of the Mutant state. According to this hierarchy, from Mutant we can reach all OR_{DD} states of MutationAnalysisDetails. To define the full set of states we can reach from Mutant, the same analysis should be carried out for all states of the hierarchy

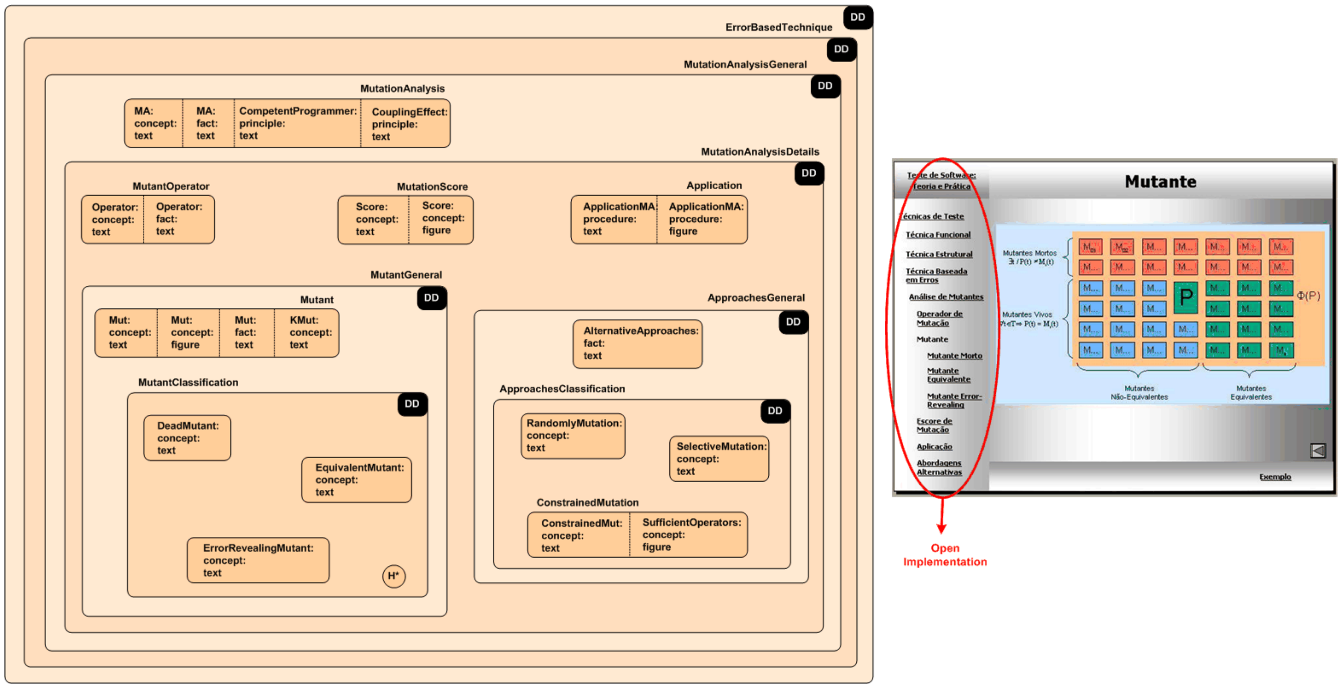


Figure 6. Didactic Model / Slides for Mutation Analysis: Open Specification

of *DD*-superstates of `MutantGeneral`. Notice that we cannot get to the states `AlternativeApproaches` and `ApproachesClassification` from the `Mutant` state. Indeed, `ApproachesGeneral` does not pertain to the hierarchy of *DD*-superstates of `Mutant`.

Additionally to the definition of an *open specification*, definitions of a *partially open specification* and of a *close specification* were also established in scope of the didactic model. Basically, in a partially open specification, while some sequences of presentation can be established in "execution time", others are previously defined by the domain expert and/or the instructor during the development of the module. Indeed, instead of having just implicit transitions, the idea is to make some of them be explicitly represented in the didactic model. On the other hand, in a close specification all sequences are predefined, that is, just a fixed sequence of presentation is available in the module. In this case, the transitions would be explicitly represented.

The sequences of presentation derived from partially open specifications and from close specifications represent subsets of the total set of sequences established by an open specification. As highlighted before, a didactic model defined in terms of an open specification can be seen as the basis from which all sequences of presentation are derived. So, by using the didactic model illustrated in Figure 6, several implementations of the same content about mutation analysis can be obtained. Such characteristic is essential to generate differentiated content, whose topics, depth and sequences of presentation are established according to some particular aspects (e.g., course length, pedagogical goals, instructor's preference, learner's profile).

Figure 7 illustrates part of the didactic model for mutation analysis constructed by using the idea of a *close specification*. Notice that all sequences of presentation are predefined. In fact, none of the states represented in the model was specified as a DD-state. Also, all possible transitions between states were explicitly represented. Again, consider the `Mutant` state. Now, from `Mutant` we are able to get only two states: `DeadMutant` and `MutationScore`.

The decision on which kind of specification to use should be based on the users (learners and instructors) as well as on the expected characteristics of the module. For instance, one strength of open specifications is the flexibility to navigate the material according to the feedback and questions of the audience. On the other side, the instructor has to make sure to achieve the objectives of the lessons in order to keep the learners localized. Indeed, while for less experienced instructors a close specification (and implementation) seems to be the better choice, for the most experienced ones, an open specification would be an adequate alternative too.

4. Evaluating the Educational Module

To provide a preliminary evaluation on the effectiveness of the mutation testing module presented herein, it was applied as part of a three-hour short-course on software testing

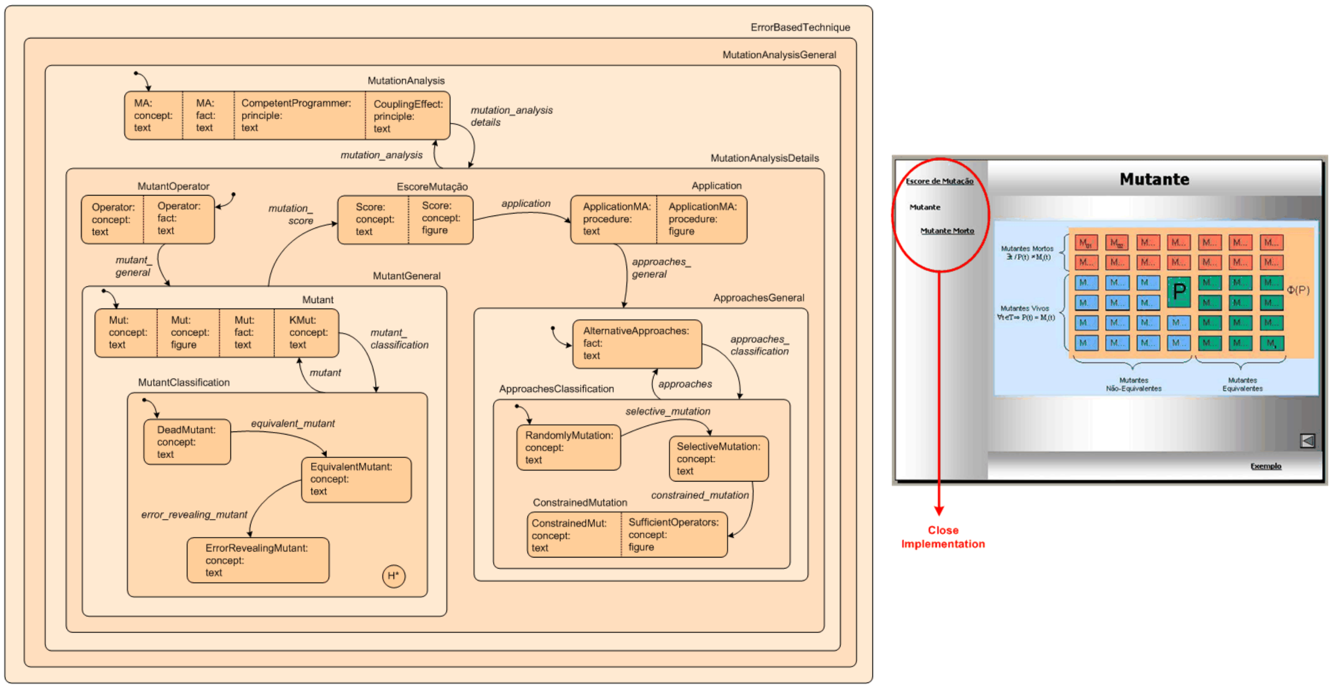


Figure 7. Didactic Model / Slides for Mutation Analysis: Close Specification

for a group of about 60 undergraduate students with previous knowledge of software engineering. Specifically, we focused on theoretical aspects of mutation testing, providing an introductory perspective on this subject. Practical aspects were illustrated but, due to time constraints, there was no direct participation by the audience on using any of the materials.

The effects of our approach were informally evaluated by applying a voluntary survey to the students after they had finished the course. The survey covered 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. In general, we could observe a positive attitude toward the flexibility provided by the produced module. Furthermore, instructor's responses were also observed by his comments after using the module. The possibility of having defined the sequences of navigation through the module during the "execution time", based on the learner's understanding and feedback, was a significant point highlighted.

The results obtained provide preliminary evidence on the practical use of the standard process and the *IMA-CID* approach as supporting mechanisms to the development of effective educational modules. However, it is important to conduct more systematic and controlled experiments to validate our ideas. Such experiments have already been planned for the next term, involving different courses on software testing, 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.

5. Conclusions and Further Work

In this paper some mechanisms for developing educational modules were discussed and the application of an integrated modeling approach (*IMA-CID*) was illustrated by the development of an educational module for the mutation testing domain. Our main goal was to motivate the use of systematic processes and supporting mechanisms for creating well-designed, highly flexible and configurable educational modules, particularly in the context of mutation testing. Such module would provide, among others: (1) transferability of mutation testing to different institutions and learning environments; and (2) effectively supporting traditional learning approaches to facilitate the apprenticeship of specific mutation testing theories and skills.

As further work, we intend to focus on the development of educational modules to be applied in non-traditional environments, motivating the transition from lecture-based to active learning. The input of students in the very early stages of the module development, similarly to the participative approach in software development, is another point to be further investigated. In particular, the availability of distance learning facilities, such as open materials, and testing tools would promote better learning and dissemination conditions that may lead to practical evaluation and application of mutation testing.

Additionally, we are motivated to keep evolving and evaluating the mechanisms we have proposed in future offerings of testing courses. We are now working on the development of an educational module for the integrated teaching of testing and programming foundations in introductory CS courses [5]. The educational module for integrating these knowledge domains has been designed and implemented according to the standard process and the *IMA-CID* models. Since the standard process can be applied to different domains, we are also interested in using it to develop and evaluate educational modules for other areas and broader projects.

Another perspective is to explore the development of learning objects [10, 19] under the context of educational modules. The idea is to apply the proposed modeling mechanisms to structure, store and retrieve the internal components of these objects. Further studies have also been planned in order to investigate the use of conceptual models on the development of domain ontologies and vice versa.

Acknowledgements

The authors would like to thank the Brazilian funding agencies – FAPESP, CAPES and CNPq – for their support to this research.

References

[1] E. Barbosa and J. 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, Aug. 2006.

[2] E. Barbosa and J. Maldonado. Towards the establishment of a standard process for developing educational modules. In *36th Annual Frontiers in Education Conference (FIE 2006)*, San Diego, CA, Oct. 2006. Accepted, to appear.

[3] E. F. Barbosa, J. C. Maldonado, and A. M. R. Vincenzi. Towards the determination of sufficient mutant operators for C. *Software Testing, Verification and Reliability*, 11(2):113–136, June 2001.

[4] J. A. Brotherton and G. D. Abowd. Lessons learned from eClass: Assessing automated capture and access in the classroom. *ACM Transactions on Computer-Human Interaction*, 11(2):121–155, June 2004.

[5] C. K. D. Corte, E. F. Barbosa, and J. C. Maldonado. Integrated teaching of programming foundations and software testing. In *IFIP 19th World Computer Congress – International Conference on Education for the 21st Century (Poster Session)*, page 425, Santiago, Chile, 2006.

[6] 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, Mar. 2001.

[7] 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, Apr. 1978.

[8] W. Dick, L. Carey, and J. O. Carey. *The Systematic Design of Instruction*. Longman, 5th edition, 2001.

[9] A. Dieberger and M. Guzdial. *CoWeb – Experiences with collaborative web spaces*. Springer-Verlag, 1st edition, 2003. C. Lueg and D. Fisher (eds.).

[10] S. Downes. Learning objects: Resources for distance education worldwide. *International Review of Research in Open and Distance Learning*, 2(1), July 2001.

[11] M. W. Goldberg, S. Salari, and P. Swoboda. World Wide Web - Course Tool: An environment for building WWW-based courses. *Computer Networks and ISDN Systems*, 28(7–11):1219 – 1231, May 1996.

[12] C. L. L. Maidantchik and A. R. Rocha. Managing a worldwide software process. In *Workshop on Global Software Development – International Conference on Software Engineering (ICSE 2002)*, Orlando, FL, May 2002.

[13] J. C. Maldonado, E. F. Barbosa, A. M. R. Vincenzi, and M. E. Delamaro. Evaluating N-selective mutation for C programs: Unit and integration testing. In *Mutation 2000 Symposium*, pages 22–33, San Jose, CA, Oct. 2000. Kluwer Academic Publishers.

[14] J. I. Mayorga, M. F. Verdejo, M. Rodríguez-Artacho, and M. Y. Calero. Domain modelling to support educational web-based authoring. In *TET 99 Congress*, Norway, June 1999.

[15] E. Mresa and L. Bottaci. Efficiency of mutation operators and selective mutation strategies: an empirical study. *The Journal of Software Testing, Verification and Reliability*, 9(4):205–232, Dec. 1999.

[16] J. D. Novak. Concept mapping: A useful tool for science education. *Journal of Research in Science Teaching*, 27:937–949, 1990.

[17] A. J. Offutt, A. Lee, G. Rothermel, R. H. Untch, and C. Zapf. An experimental determination of sufficient mutant operators. *ACM Transactions on Software Engineering Methodology*, 5(2):99–118, Apr. 1996.

[18] M. A. S. Turine, M. C. F. Oliveira, and P. C. Masiero. Designing structured hypertext with HMBS. In *VIII International ACM Hypertext Conference (Hypertext 97)*, pages 241–256, Southampton, UK, Apr. 1997.

[19] D. A. Wiley. Connecting learning objects to instructional design theory: A definition, a metaphor, and a taxonomy. In *The Instructional Use of Learning Objects*, Bloomington, IN, 2001. Association for Educational Communications and Technology.

[20] W. Wong, J. Maldonado, and M. Delamaro. Reducing the cost of regression test by using selective mutation. In *8th CITS – International Conference on Software Technology*, pages 11–13, Curitiba, PR, June 1997.