Porantim: An Approach to Support the Combination and Selection of Model-Based Testing Techniques

Arilo Claudio Dias Neto Guilherme Horta Travassos
Federal University of Rio de Janeiro
Systems Engineering and Computer Science Program – COPPE/UFRJ
P.O. Box 68511 – 21941-972 Rio de Janeiro, RJ, Brazil
{acdn,ght}@cos.ufrj.br

Abstract

The technical literature regarding Model-based Testing (MBT) has several techniques with different characteristics and goals available to be applied in software projects. Besides the lack of information regarding these techniques, they could be applied together in a software project aiming at improving the testing coverage. However, this decision needs to be carefully analyzed to avoid loss of resources in a software project. Based on this scenario, this paper proposes an approach with the purpose of supporting the unique or combined selection of MBT techniques for a given software project considering two aspects: the adequacy level between MBT techniques and the software project characteristics and impact of more than one MBT technique in some testing process variables. At the end, preliminary results of an experimental evaluation are presented.

1. Introduction

The selection of software technologies[1] to use in a given software project is a research topic that started to be addressed in 1991 when a proposal for all software elements types (process, products, techniques, etc.) was published by Basili and Rombach [1]. Since then, there has been consensus that the selection of software technologies is a complex task that may directly influence process effectiveness and product quality. There is currently a wide variety of technologies available to support software system development and the main reasons why software engineers do not make good choices are: information available on the technologies distributed across different sources (e.g.: books, papers, repositories), lack of guidelines to support the selection, and poor scientific knowledge regarding the technologies and their use in previous software projects [2].

For some development process task, the choice of a software technology needs to be unique, that is, only one software technology is enough as the combination of technologies would not introduce better results for the software development process (for instance, the selection of the modeling tool). This situation may make the software technologies selection easier. However, for other tasks the combination of two or more software technologies may introduce an improvement to the effectiveness process and consequently the final product quality. An example of this situation is the selection of testing techniques to evaluate a software system, where more than one testing technique may be applied resulting in a higher testing coverage. However, it is necessary a feasibility analysis regarding how to apply more than one technique as besides the improvement of testing coverage, there is a risk concerned with the improvement of testing effort and cost that would render the testing activity unfeasible [3].

There are some approaches in literature that support the selection of software technologies, including testing techniques [2][4][5][6]. However, they are focused on the exclusive selection of only one software technology, that is, they do not provide information and features to support the combined choice of more than 1 software technology.

Contextualizing this scenario for the Model-Based Testing (MBT) field, we observed some additional challenges to support the selection of MBT techniques for a given software project. MBT provides a method for the automatic generation of test cases using models extracted from software artifacts [7]. This category of testing technique has additional and specific characteristics in comparison to general testing techniques (non model-based), such as the dependency of a formal model to support test case generation. Moreover, it introduces several observed benefits concerned with the testing process for a software organization, such as: shorter schedules; lower cost and effort; better quality; enhanced communication among developers and testers; early exposure of ambiguities in specification and design; capability to automatically

[1] In this paper, a software technology may be a technique, method, process, or tool that could be used to support software development.

978-1-4244-3711-5/09/$25.00 © 2009 IEEE

generate many non-repetitive and useful tests; test harness to automatically run generated tests; easing of the updating of test suites for changed requirements, and capability to evaluate regression test suites [8]. Dias-Neto *et al*. [9] report a systematic review analyzing MBT techniques published in the technical literature where we could observe that almost all MBT techniques have not been experimentally evaluated and there is not enough knowledge regarding their performance, effectiveness, and complexity for the use in the industry. Thus, the selection of a MBT technique for a software project is a hard task and inadequate choices may render invalid, or even not feasible, the tests for a software project.

Based on this context, this paper presents an approach, called *Porantim*, aimed at supporting the selection of MBT techniques for software projects. In the proposed selection approach, the MBT techniques are firstly analyzed individually according to their adequacy for a software project, crossing information regarding the software project requirements and MBT techniques characteristics. In a second step, more than one MBT techniques may be selected for a software project, and so *Porantim* provides an analysis regarding the impact of the MBT techniques combination in some testing process variables, such as their coverage considering the software project characteristics and modeling effort.

This paper is organized as follows: section 2 discusses the problem of software technologies selection, contextualizing this problem for Model-based Testing field. Section 3 presents *Porantim*, an approach to support the selection of MBT techniques for software projects. Section 4 describes the evaluation of *Porantim* by a characterization and an experimental study. Finally, section 5 reports the conclusions and future works.

2. Selection of MBT Techniques

2.1. Selection of Software Technologies

The selection of technologies for a software project has been reported since 1991 and since then a lot of approaches have been developed for different activities in the software development process. The support provided by these approaches ranges from the generic characterization of software technologies aimed at their reuse in software projects ([1] and [4]) to approaches to support the selection of software technologies for a specific activity, such as Requirements Elicitation [5], Software Testing [2], or Model-Based Testing [6].

The main challenge concerned with the selection problem is to understand and select which software technology better fits a specific task, in a specific software project, in a software organization. Thus, several aspects may influence this decision such as: project team expertise and background, project schedule, effort and cost required/introduced to use it, amongst others.

We can observe several aspects, in different fields, that make the selection of software technologies hard for a given software project, such as information available on the technologies distributed across different sources, lack of guidelines to support the selection, and poor scientific knowledge (evidence) regarding the technologies and their use in previous software projects [2].

When we contextualize this problem in the MBT field, new aspects are introduced, as we discussed in the next section.

2.2. Selecting Model-Based Testing Techniques

When we look specifically at the Model-Based Testing field we observe that while current research has been producing interesting results regarding the development of new techniques and infrastructures to support MBT, there is still a lack of scientific knowledge regarding already-developed MBT techniques, making the transferring of these technologies harder from academic to industrial environments. According to Dias-Neto *et al*. [9], some factors contribute to this scenario:

- High number of MBT techniques available in technical literature.
- Lack of a body of knowledge or repository with information on MBT techniques.
- Lack of scientific knowledge (evidence) on the use of MBT in different software projects.

These aspects make the selection of just one MBT technique subset for a given software project difficult, as one does not have enough information to be sure (or minimize the risks) on the decision. Currently, this task is performed based on the convenience (that is, which MBT technique we already know, independently of its adequacy for the software project) of the test team in a software project [2].

Some works have tried to create approaches to support the selection of testing techniques for a software project. For instance, in [2], a description can be found of an approach aiming at to support the selection of testing techniques based on a characterization schema. The attributes of the testing techniques are stored in a repository, and for each software project a catalogue of techniques is generated through schema definitions. The positive aspects of this proposal are to make knowledge available and to facilitate the choice of general purpose testing techniques for software projects. However, the generality of the attributes makes their customization hard for specific characteristics of model-based testing techniques, which could mean a risk for a software

project when using this type of approach to select testing techniques. In [10], an experimental study describing differences and similarities amongst testing techniques can be found. Despite its purpose not being to support the selection of testing techniques, test practitioners may use this knowledge with this purpose.

The next section will present an approach to support the selection of MBT techniques for a software project. Its main characteristics are: (1) analysis of the adequacy of MBT techniques according to software project requirements, suggesting the best-suited and (2) analysis of the impact of MBT techniques on some testing process variables (e.g.: their coverage considering the software project characteristics and modeling effort) when more than 1 MBT techniques is selected for a software project.

3. *PORANTIM*: An Approach for the Selection of MBT Techniques

The proposed approach, called *Porantim*[2], represents an evolution of the characterization schema of testing techniques proposed by Vegas and Basili [2]. The characterization attributes for the testing technique were updated with specific MBT characteristics and a selection process was introduced to indicate which MBT techniques would be best-suited, and to analyze their impact on a software project. It is based on two elements: a body of knowledge on MBT techniques that works as a repository of techniques that may be used in a given software project and a process to support the selection of MBT techniques.

The body of knowledge on MBT techniques is not the focus of this paper. However, it works as a repository, and it has been developed based on the results of two experimental studies. In order to define its **structure**, a survey was performed with specialists (researchers and practitioners) in MBT approaches. Detailed information regarding its planning, execution, and results, including the final set of characterization attributes of MBT techniques and their respective relevance level during the selection of MBT techniques for a software project was published in [11]. In order to define the **content** of the body of knowledge, a systematic review was performed to identify and characterize MBT approaches published in technical literature. The protocol used in this Systematic Review and its results was published in [9]. Thus, the body of knowledge on MBT approaches has been defined and may be found by contacting the authors. The next step performed was to define the process for the selection of MBT approaches.

3.1. MBT Techniques Selection Process

In *Porantim*, the selection of MBT techniques is guided by a process as shown in Figure 1. It consists of 5 activities as shown below:

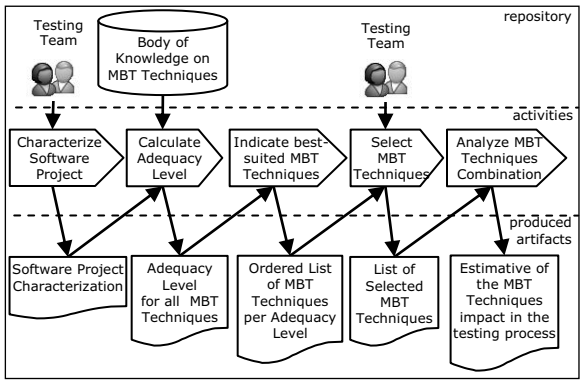

Figure 1. MBT Techniques Selection Process

- **Characterize Software Project:** the testing team needs to fill a questionnaire about the software project characteristics where MBT techniques will be applied. In total, 16 attributes regarding the software project should be filled in such as: execution platform, programming language, models provided by the development process, testing level required, software quality characteristics to be evaluated, amongst others.
- **Calculate Adequacy Level:** internally, *Porantim* will calculate an adequacy level between the software project characterized in the previous step for each MBT technique included in the repository. The adequacy level is a numeric value indicating how close the MBT technique characteristics are from the software project characteristics. In the next section this step will be detailed.
- **Indicate best-suited MBT Techniques:** after calculating the adequacy level, *Porantim* presents a small MBT techniques subset (the number may be defined by the testing team) showing the adequacy level and characteristics of the best-suited MBT techniques.
- **Selecting MBT Techniques:** the testing team needs to select a subset of MBT techniques suggested in the previous step to be used in the software process.
- **Analyze MBT Techniques Combination:** after the selection task, *Porantim* analyzes the impact of the selected MBT techniques combination on some variables of the testing process, such as software project coverage, modelling effort, and human resources required to use the MBT techniques. In the next section this step will be detailed.

[2] *Porantim* is piece of wood used by the *Sateré-Mawé* (Amazon Indians) that has several attributes: it is their bible, works as a crystal ball preventing things from happening, and solving internal conflicts, and is used in the *Wat'amã* ritual, a challenge set to select tribe warriors.

The next sections will describe how *Porantim* supports the calculus of the adequacy level between a software project and MBT techniques and analyzes the impact of a combination of MBT techniques.

3.2. Analyzing the MBT Techniques' Adequacy

During the MBT techniques selection process by *Porantim*, an adequacy level between the software project and MBT techniques characteristics is calculated. For that, we are using the mathematical concept of *Euclidian distance* [12]. This mechanism explores the evaluation of conceptual distance from the comparison of elements in a multidimensional vector space mathematically represented by the norm of the difference between two vectors *v1* and *v2*. In *Porantim*, the software projects (v1) and MBT techniques (v2) are represented by a set of characterization attributes as listed in Table 1. These attributes have a specific weight (importance level) obtained by a survey performed with MBT specialists [11]. These characteristics are translated into values using some pre-defined rules[3] (*if [software project attribute = MBT technique attribute] → MBT technique attribute is transformed into 1*attribute's weight; else it is 0*), and these values are represented in vectors. The distance between the two vectors indicates the adequacy level of a MBT technique for a given software project. The closer they are, the more adequate a MBT technique is for a software project.

Table 1. Software Projects Attributes

Characterization Attribute	Weight [11]	Transformation rule
Behavioural/ Structural Model	0.8348	(SW Project = MBT technique) → 0.8348 (SW Project ≠ MBT technique) → 0
Development paradigm	0.5585	(SW Project = MBT technique) → 0.5585 (SW Project ≠ MBT technique) → 0
Modelling Technology	0.5254	(SW Project = MBT technique) → 0.5254 (SW Project ≠ MBT technique) → 0
Programming Language	0.5585	(SW Project = MBT technique) → 0.5585 (SW Project ≠ MBT technique) → 0
Software Execution platform	0.5585	(SW Project = MBT technique) → 0.5585 (SW Project ≠ MBT technique) → 0
Software quality characteristic desired	0.6830	(SW Project = MBT technique) → 0.683 (SW Project ≠ MBT technique) → 0
Supporting Tools	0.6374	(SW Project = MBT technique) → 0.6374 (SW Project ≠ MBT technique) → 0
Testing Level(s) required	0.6970	(SW Project = MBT technique) → 0.697 (SW Project ≠ MBT technique) → 0
Type of failures to be revealed	0.6970	(SW Project = MBT technique) → 0.697 (SW Project ≠ MBT technique) → 0
Type of Testing Technique required	0.6989	(SW Project = MBT technique) → 0.6989 (SW Project ≠ MBT technique) → 0

The formulas used to calculate the distance and adequacy level between the software project and MBT techniques characteristics are provided in Figure 2.

[3] The software project attribute is considered the requirement to be attended to. It is automatically changed into *1*attribute's weight*.

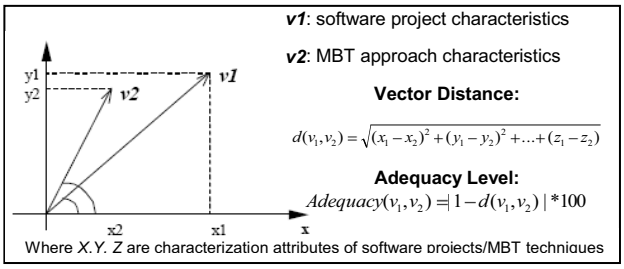

Figure 2. Formula for vector distance calculus

Table 2 presents an example of Software Project and two MBT techniques characterization. We will use these data to simulate the adequacy level calculus.

Table 2. Example of Characterization

Characterization Attributes	Software Project	Technique 1	Technique 2
Behavioural/Structural Model provided	Use Case, Activity, Statechart, Classes Diagram	Collaboration Diagram	Use Case, Activity, Classes Diagram
Development paradigm	Object oriented	Object oriented	Object oriented
Modelling Technology	UML	UML	UML
Programming Language	Java	Java	Any
Software Execution platform	Information System	Any	Information System
Software quality characteristics desired	Functionality Efficiency	Functionality Efficiency	Functionality Efficiency Security
Supporting Tools	No tool available	Rational Rose plug-in	TDE
Testing Level(s) required	System testing	System Testing	System Testing
Type of failures to be revealed	Navigational Error, Interface Error, Data type wrong	Navigational Error, Data type wrong	Navigational Error, Interface Error, Data type wrong
Type of Testing Technique required	Functional	Structural	Functional

By applying the transformation rules, weights and formula, we can obtain the adequacy level between the software project and MBT techniques. Table 3 shows the calculus performed to calculate the adequacy level between the software project and the two MBT techniques characterized in Table 2.

Table 3. Example of Adequacy Level calculus

Characterization Attributes	SW Project	Technique 1	Technique 2
Behavioral/Structural Model	0.8348	0	0.6261
Development paradigm	0.5585	0.5585	0.5585
Modelling Technology	0.5254	0.5254	0.5254
Programming Language	0.5585	0.5585	0.5585
Software Execution platform	0.5585	0.5585	0.5585
SW quality characteristic desired	0.6830	0.6830	0.6830
Supporting Tools	0.6374	0.6374	0.6374
Testing Level(s) required	0.6970	0.6970	0.6970
Type of failures to be revealed	0.6970	0.46	0.6970
Type of Testing Technique required	0.6989	0	0.6989
Adequacy Level		52.47%	87.21%

Porantim also suggests the using of a radar (spider) chart to support the individual analysis of adequacy for each MBT technique, as visualized in the Figure 3. In this chart, each axis represents a characterization attribute. The resulting radar chart graphically shows the coverage of a MBT technique in comparison with

the characteristics of a software project. In the Figure 3, the dark color indicates the software project characteristics are covered by the analyzed MBT technique. Therefore, the MBT technique 2 is more adequate than MBT technique 1 (as the dark area is higher for Technique 2).

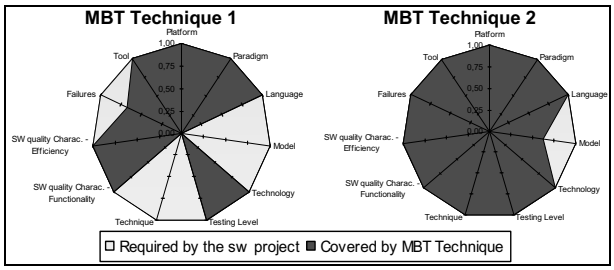

Figure 3. Adequacy Level by a Radar chart

The next step to be performed is the selection of MBT techniques by the testing team. As described in the section 3.1, more than one MBT technique may be selected, what may cause a negative or positive impact for the software testing process. In the next section, *Porantim*'s support for this task will be described.

3.3. Analyzing MBT Techniques Combination

During the selection task, more than one MBT technique may be selected, which may result in different needs for a software project. Thus, the next activity (after the selection of MBT techniques) of the *Porantim*'s selection process presented in Figure 1 corresponds to an analysis on the impact of the selected MBT techniques in some testing process variables:

- **Software Project coverage:** indicator about which software project requirements the selected MBT techniques attend, such as testing level, software quality characteristics, and types of failures to be revealed.
- **Modelling Effort:** indicator about which models required by the MBT technique(s) are already available in the software development software process and which ones need to be built to allow the use of the selected MBT techniques.
- **Human Resources:** indicator about the testing team members who have the best background to use the selected MBT techniques.

Thus, in order to support the analysis of these variables, we defined some mathematical formulas using metrics obtained during the software project and MBT techniques characterization. Each variable is analyzed considering a different formula to obtain the respective indicator. They all use the concept of the *Jaccard Coefficient*[4] [13] and they were defined considering our experience in applying testing techniques in real software projects, as described in the next sections.

These formulas need to be implemented in a computerized infrastructure to support the automatic calculus of these variables.

3.3.1. Software Project Coverage Indicator. This indicator analyzes whether the selected MBT techniques are attending/covering three aspects required by the software project: testing level, software quality characteristics, and types of failures to be revealed.

- **Testing Level Analysis:**
  - ✓ Software project requires the testing level provided by the selected MBT techniques: this is the ideal scenario and therefore there is no negative impact on the software project.
  - ✓ Software project requires a testing level, but it is not provided by the selected MBT techniques: it is not the desired scenario (something required but not provided) and therefore there is a negative impact on the software project.
  - ✓ Software project does not require a testing level, but it is provided by the selected MBT techniques: it is not the desired scenario (we would be inserting an additional effort to test a software level different from the desired ones) and therefore there is a negative impact on the software project.

Thus, the formula used to calculate the testing level indicator is:

$$TLevel = 1 - \left(\frac{\#TLevel(proj - techs) + \#TLevel(techs - proj)}{\#TLevel(proj \cup techs)}\right)$$

Where:

- *#TLevel(proj – techs)* is the number of testing levels required by the software project, but not attended by the selected MBT techniques.
- *#TLevel(techs – proj)* is the number of testing levels provided by the selected MBT techniques, but not required by the software project.
- *#TLevel(proj ∪ techs)* is the number of different testing levels required by the software project OR provided by the selected MBT techniques.

- **Software Quality Characteristics:**
  - ✓ Software project requires the software quality characteristics provided by the MBT technique: this is the ideal scenario and therefore there is no negative impact for the software project.
  - ✓ Software project does not require the software quality characteristic, but it is provided by the MBT technique: this is an additional aspect that does not have a negative impact on the software project.

[4] *Jaccard Coefficient* (1901) compares the number of similar elements and the total number of elements in a set (called *similarity* coefficient).

✓ If the software project requires the software quality characteristic but it is not provided by the MBT technique: it is not the desired scenario (something required but not provided) and therefore there is a negative impact for the software project.

Thus, the formula used to calculate the software quality characteristic indicator is:

$$SWQuality = 1 - \left(\frac{\#SWQ(proj) - \#SWQ(proj \cap techs)}{\#SWQ(proj)}\right)$$

Where:

- *#SWQ(proj)* is the number of software quality characteristics required by the software project.
- *#SWQ(proj ∩ techs)* is the number of different software quality characteristics required by the software project AND provided by the selected MBT techniques.

- **Types of Failures to be revealed:**

✓ Software project requires the detection of a failure type provided by the MBT technique: it is the ideal scenario and therefore there is no negative impact on the software project

✓ Software project does not require the detection of a failure type, but it is provided by the MBT technique: it is an additional aspect that does not have a negative impact on the software project.

✓ Software project requires the detection of a failure type, but it is not provided by the MBT technique: it is not the desired scenario (something required but not provided) and therefore there is a negative impact on the software project.

Thus, the formula used to calculate the types of failure indicator is similar to the previous one:

$$Failure = 1 - \left(\frac{\#Failure(proj) - \#Failure(proj \cap techs)}{\#Failure(proj)}\right)$$

Where:

- *#Failure(proj)* is the number of types of failures required by the software project.
- *#Failure(proj ∩ techs)* is the number of different types of failures required by the software project AND provided by the selected MBT techniques.

In order to obtain the software project coverage indicator we calculate the average amongst the three values previously obtained according to the formula:

$$SWP\_Coverage = \frac{TLevel + SWQuality + Failure}{3}$$

The obtained value is a number between 0 and 100% that may be represented by a speedometer chart considering 4 different ranges: [100-75%]: High; [75-50%]: Medium; [50-25%]: Low; [25-0%]: Very Low. An example of the calculus of this indicator is given in Table 4 and Figure 4.

Table 4. Software Project Coverage Indicator

Attribute	Software Project	MBT Technique 1	MBT Technique 2	Calculus
Testing Level	System and Integration Testing	Unit Testing	System and Integration Testing	$1-(\frac{0+1}{3}) = 0.66$
Software Quality Characs	Functionality, Security, Performance, Usability	Functionality, Performance, Usability	Functionality	$1-(\frac{4-3}{4}) = 0.75$
Type of Failures	Type 1, Type 2, Type 3	Type 2 Type 4	Type 1	$1-(\frac{3-2}{3}) = 0.66$
Software Project Coverage	$\frac{0.66+0.75+0.66}{3} = 0.69 \rightarrow 69\%$			

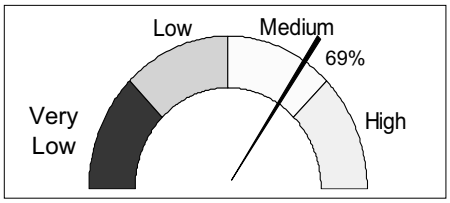

Figure 4. Speedometer Chart for Software Project Coverage Indicator

3.3.2. Indicator of Modeling Effort. This indicator analyzes an estimative of the effort to prepare the models used by the selected MBT techniques for testing according to the models provided by the software development process. Five different scenarios are considered in Figure 5. We used an ordinal scale (from 0 to 4) to define the weight referent to the modelling effort in each scenario:

a) If the model is built throughout the software development process but is not required by the selected MBT techniques: there is no modeling effort to adapt it, then these models are not considered during the analysis.

b) If the model is built throughout the software development process and it is required by only 1 selected MBT technique: there is a minimum modeling effort to adapt this model for the MBT technique, then we consider a weight 1 to perform this task.

c) If the model is built throughout the software development process and it is required by more than 1 selected MBT technique: there is a small modeling effort (higher than the previous case) to adapt this model for the MBT techniques, then we consider a weight 2 to perform this task.

d) If the model is not built throughout the software development process but is required by only 1 selected MBT technique: there is a significant modeling effort (higher than the 2 previous cases) to build and to adapt this model for the selected MBT technique, then we consider a weight 3 to perform this task.

e) Finally, if the model is not built throughout the

software development process, but is required by more than 1 selected MBT technique: there is a high modeling effort (higher than the 3 previous cases) to build and to adapt this model for the selected MBT techniques (each one may require a different adaptation), then we consider a weight 4 to perform this task.

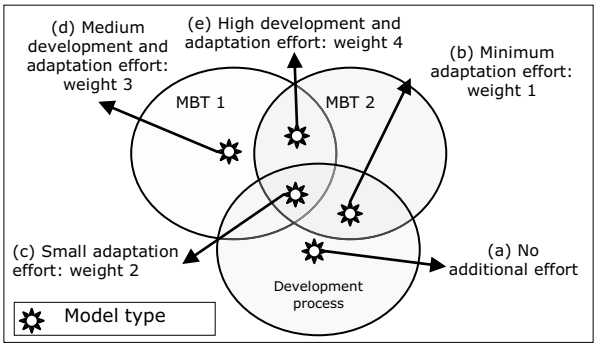

Figure 5. Modeling effort analysis

Thus, by also using the *Jaccard Coefficient*, the formula used to calculate the modeling effort indicator corresponds the total of models in each category (B to E) described above (multiplied by their respective weight) divided by the total number of models.

The modeling effort formula is presented in Figure 6, where #Mod[cat] is the number of models used by a MBT technique or provided in the software project in one of the categories described previously (A, B, C, D, or E).

$$Effort = \frac{(\#Mod[b]*1)+(\#Mod[c]*2)+(\#Mod[d]*3)+(\#Mod[e]*4)}{\#Mod[a+b+c+d+e]*4}$$

Figure 6. Modelling Effort Indicator Formula

The value obtained is also a number between 0 and 100% that may be also represented by a speedometer chart considering 4 different ranges, as presented in Figure 4. Considering the scenario in Figure 5, the modeling effort indicator is:

$$Effort = \frac{(1*1)+(1*2)+(1*3)+(1*4)}{5*4} = \frac{10}{20} = 0.5 \rightarrow 50\%$$

3.3.3. Human Resources Indicator. This indicator analyzes the human resources in the testing team in a software organization as required to use the selected MBT techniques. This analysis indicates which skills set required to use the selected MBT techniques (we will call the variable *MBT*) are being provided by the selected testing team (we will call the variable *Team*). The formula used is:

$$Human\ \text{Re}\ sources = \frac{|Team \cap MBT|}{|MBT|}$$

The value obtained is a number between 0 and 100% that may be also represented by a speedometer chart considering 4 different ranges, as presented in Figure 4. An example of the calculus of this indicator is presented in Figure 7.

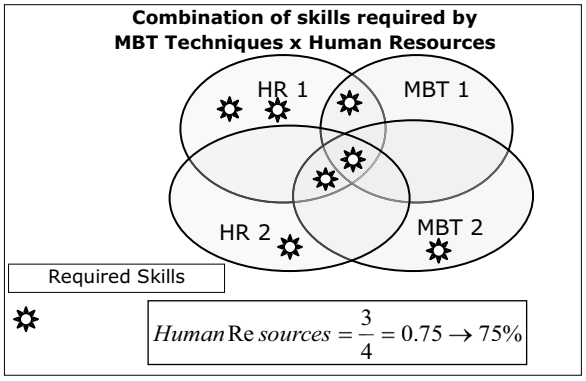

Figure 7. Human Resources Indicator Example

Thus, we defined the activities forming the *Porantim*'s MBT techniques selection process. In the next section, the *Porantim*'s evaluation by a characterization and preliminary results of an experimental study will be discussed.

4. *Porantim* Evaluation

The *Porantim*'s evaluation will be described in two ways: First, a characterization against other similar approaches, discussing the main differences among them. Secondly, the preliminary results of an experimental study executed to observe the *Porantim*'s efficiency when compared to another testing technique selection approach.

4.1. *Porantim* Characterization

During the development of *Porantim*, we observed some important characteristics to be analyzed in an approach with the purpose of to support the selection of software technologies. These characteristics are shown in Table 5 for three different approaches: Birk [4], Vegas & Basili [2], and *Porantim* [6]:

- **Scope:** which software development task the approach may be applied to.
- **Based on Measurement:** if the approach uses measurement or subjective information as mechanism to support the selection of MBT techniques.
- **Based on Adequacy:** if the approach considers some adequacy level between MBT techniques and the software project during the MBT techniques selection.
- **Based on Impact:** the approach considers the

possible impact of selected MBT techniques on the testing process or not.

- **Suggestion Way:** if the approach suggests an order for the MBT techniques based on some criteria during the selection or just lists the MBT techniques as a catalogue.
- **Selection Way:** if the selection of MBT techniques may be combined or needs to be individual.
- **Supporting Tool:** the existence of a support tool.

Table 5. Selection Approaches Characterization

Approach	Birk	Vegas & Basili	*Porantim*
Scope	Software Technologies	Testing Techniques	MBT Techniques
Based on Measurement	Yes	Yes	Yes
Based on Adequacy	Yes	Yes	Yes
Based on Impact	No	No	Yes
Suggestion Way	Catalogue of techniques	Catalogue of techniques	Ordered list by adequacy level
Selection Way	Individual	Individual	Combined
Support Tool	Not Found	Yes	Yes

4.2. Experimental Evaluation

The goal of this experimental study was to analyze two different approaches for the selection of MBT techniques (*Porantim* [presented in this paper] and the Characterization Schema [2]) as regards completeness, effectiveness, efficiency, usability, and satisfaction from the standpoint of software engineering students, in the context of software projects.

This study was executed between Nov 27th and Dec 11th, 2008 with 21 Master and PhD students with poor experience in the selection of MBT techniques. We are currently analyzing all obtained data. As of this writing, we have already completed the analysis of the statistical data regarding efficiency (selection time) and effectiveness (percentage of correct choices). These results will be presented in this section.

Subjects were divided an allocated randomly into four groups. Each one made two selections for two different software projects using different selection approaches according to Table 6. In total, we used 4 different software projects and 13 MBT techniques. Thus, each subject received a different combination of selection approach and software project.

Table 6. Strategy Allocation as per groups

	Group1	Group 2	Group 3	Group 4
Selection 1	*Porantim* Complete	Schema	Schema	*Porantim* Incomplete
Selection 2	*Porantim* Complete	*Porantim* Incomplete	Schema	Schema

- **Schema:** selection approach proposed by Vegas & Basili [2].
- ***Porantim* incomplete:** just the characterization of MBT techniques, without to show the adequacy level between software projects and the MBT techniques.
- ***Porantim* complete:** characterization + radar chart and adequacy level indicator for all MBT techniques considering the software projects.

4.2.1. Efficiency (Selection Time) Analysis. Preliminary results on the selection time are shown in Table 7. After analyzing the data obtained we can see that subjects using *Porantim* (complete or incomplete) spent on average less time to analyze and select MBT techniques than when using the Schema.

Table 7. Efficiency (Selection Time) Analysis

Selection Approaches	Selection Time		
	MEAN	VAR	STD
Schema	61.40	694.1474	26.3467
Porantim incomplete	56.30	349.3444	18.6908
Porantim complete	53.08	92.4470	9.6149

Moreover, the standard deviation with *Porantim* is also smaller that when using the Schema, that is, the results are more homogenous, which indicates that possibly the reason for this less time for MBT techniques selection would be the selection approach.

Besides that, we need to still analyze the data using, for example, ANOVA (Analysis of Variance) to evaluate the significance of the different factors (selection approaches and software projects) in the results.

4.2.2. Effectiveness (% of Adequate Choices) Analysis. We have also analyzed the results for the number of correct choices performed by the subjects. For each software project, 2 MBT techniques were previously classified as *Ideal* (the right choices), three as *Adequate* (could be used with low effort), two as *Feasible* (could be used with high effort) and five as *Not Adequate* (the wrong choices).

Table 8 shows this analysis where we can see that the *Porantim*'s effectiveness (in the two versions) is significantly higher than Schema's effectiveness (70% and 62.5% of ideal choices against 47%). These results indicate that the use of *Porantim* contributes for the more adequate choice of MBT techniques for different types of software projects. Moreover, we can observe the number of not adequate choices using *Porantim* is significantly lower than using the Schema (0% and 6.25 against 21.05%).

These numbers suggest that the used approach has a significant impact in the selection of MBT techniques. While the two (complete and incomplete) versions of *Porantim* guide us usually to adequate or ideal choices (82.6% and 81.25%, respectively), using the Schema this number is lower (71.05%), that is, the confidence that the selection approach will guide us to an adequate or ideal choice looks like lower.

Table 8. Effectiveness Analysis

Selection Approach	Not Adequate	Feasible	Adequate	Ideal
Schema	21.05%	7.90%	23.68%	47.37%
Porantim incomplete	6.25%	12.50%	18.75%	62.50%
Porantim complete	0%	17.40%	13.04%	69.56%

As said previously for efficiency, we also need to analyze if there is a statistical difference between the three selection approaches using ANOVA. Moreover, we need to analyze the other aspects evaluated in this experimental study (completeness, usability and satisfaction). The data were already collected, but its analysis has not being finished.

5. Conclusions and Future Work

This paper introduced *Porantim*, an approach aimed at supporting the selection and combination of MBT techniques for a software project. The main characteristics of *Porantim* are the existence of a body of knowledge regarding MBT techniques developed from the results of two experimental studies and a supporting process providing information on the adequacy of MBT techniques and an impact analysis of their combination for a given software project.

This work is still in progress. *Porantim* has been developed following a scientific methodology based on experimental studies. Thus, the next steps in this work are concerned with the evaluation of the proposed selection approach, including:

- Finishing the analysis of the experimental study for all evaluated aspects (completeness, efficiency, effectiveness, usability, and satisfaction) as described in section 4.2;
- Evaluate experimentally *Porantim* to analyze the combination of MBT techniques for a software project (the formulas described in this paper), since the first experimental study focuses just on the adequacy level analysis between software project and MBT techniques;
- Continuing the development of the computerized infrastructure to support the *Porantim*'s features;
- Evaluating experimentally the computerized infrastructure in the software industry;

Acknowledgments

The authors would like to thank CNPq, FAPERJ, FAPEAM, and SCR for their financial support. We also would like to thank Sira Vegas and Natalia Juristo for your supporting, making available the plan and instruments used in the experimental study.

References

[1] Basili, V. R., Rombach, H. D. (1991), "Support for comprehensive reuse". Software Engineering Journal 6(5): September, 303-316.

[2] Vegas, S.; Basili, V. (2005), "A Characterization Schema for Software Testing Techniques", Empirical Software Engineering, v.10 n.4, p.437-466, October.

[3] Menzies, T., Owen, D., Cukic, B. (2002), "Saturation Effects in Testing of Formal Models". In: *13th international Symposium on Software Reliability Engineering (Issre'02),* Washington, DC, 15.

[4] Birk, A. (1997), "Modelling the application domains of software engineering technologies", In: International Conference on Automated Software Engineering (ASE). Lake Tahoe, CA, November

[5] Maiden, N. A. M.; Rugg, G. (1996), "ACRE: Selecting methods for requirements acquisition", Software Engineering Journal 11(3): 183-192.

[6] Dias-Neto, A.C.; Travassos, G.H. (2008), "Supporting the Selection of Model-based Testing Approaches for Software Projects", In: Workshop on the Automation of Software Test 2008, Leipzig, Germany, May.

[7] Dalal, S.; Jain, A.; Karunanithi, N.; Leaton, J. M.; Lott, C. M.; Patton, G. C.; Horowitz, B. M. (1999), "Model-based testing in practice", In: ICSE, May, pp. 285-294.

[8] El-Far, I. K.; Whittaker, J. A. (2001) "Model-Based Software Testing". Encyclopedia of Software Engineering (edited by J. J. Marciniak). Wiley.

[9] Dias-Neto, A.C.; Subramanyan, R.; Vieira, M.; Travassos, G.H.; Forrest, S. (2008), "Improving Evidence about Software Technologies: A Look at Model-Based Testing", IEEE Software, Vol. 25, Issues 3, pp 10-13, May.

[10] Juristo, N.; Moreno, A.M.; Vegas, S. (2004), "Reviewing 25 years of testing technique experiments". Empirical Software Engineering: An International Journal, 9(1), p. 7-44, March.

[11] Dias-Neto, A.C., Travassos, G.H. (2008), "Surveying on Model Based Testing Approaches Characterization Attributes", In: International Symposium on Empirical SW Engineering and Measurement, Oct, Kaiserslautern.

[12] Gower, J.C. (1985), "Properties of Euclidean and non-Euclidean distance matrices", Linear Algebra and its Applications, Vol. 67, pp. 81-97.

[13] Jaccard, Paul (1901), "Étude comparative de la distribution florale dans une portion des Alpes et des Jura", Bulletin del la Société Vaudoise des Sciences Naturelles 37, 547-579.