Using Process Simulation to Assess the Test Design Effort Reduction of a Model-Based Testing Approach

Eduardo Aranha and Paulo Borba

Informatics Center of the Federal University of Pernambuco, PO Box 7851, Recife, PE, Brazil
{ehsa, phmb}@cin.ufpe.br

Abstract. Several researches are being performed to address current software development problems in industry. However, quantifying the benefits of using these solutions in the practice is also a challenge. Usually, pilot studies are run to get evidences about these benefits. Nevertheless, it may be difficult to run these studies due to the required changes in the development process and the lack of available resources. In this work, we address the problem of assessing the test design effort reduction provided by TaRGeT, a tool that supports a Model-Based Testing (MBT) approach. We used process simulation to provide evidences of this effort reduction in a multi-site industry. For that, we modeled, simulated and compared the use of the current and the model-based test design processes. We identified interesting advantages of using process simulation, such as its reduced costs and the achievement of more representative results. We also show some drawbacks of this approach, such as the difficult to create models close to the reality.

Keywords: process simulation, test design effort, process improvement assessment, technology adoption.

1 Introduction

Global software development is becoming more common in industry due to its advantages, such as time-zone effectiveness and the reduced cost of others countries. However, this distributed software development also has drawbacks. For instance, it is more difficult to manage people, since there are cultural barriers and other obstacles to deal with [2].

Although several researches are being performed in order to find solutions to current software development problems, starting using these solutions in the practice is also a challenge, mainly when considering distributed development environments. For example, how to get evidences to convince managers that a given solution will result in cost reduction and product quality improvement in a multi-site organization with so many different environments, products and teams characteristics?

In industry, we usually need empirical evidences of the benefits of a given solution in order to convince people to use it. For instance, Model-Based Testing (MBT) may significantly improve test coverage and reduce the test design effort [6] [7]. However,

although these improvements are expected to occur, they need to be quantified in order to verify if these benefits pays the effort to change the current process in order to adopt MBT.

In general, pilot studies are run to evaluate these solutions and to get evidences about their benefits. Nevertheless, pilot studies are usually simples and they are not representative when considering all different environments of a global software development. Also, even pilot studies are difficult to run, since they usually require changes in the development process and the available of resources, resulting in additional costs and other factors that may impact the performance of the organization.

An alternative way to evaluate new solutions and to support their adoption is the use of process simulation [3]. With this technique, we can model processes as currently implemented and as planned for future implementation. Then, we run these models to get useful insights, predictions and empirical evidences for questions related to the benefits and drawbacks of each modeled process. Despite of the reduced costs of using process simulation instead pilot studies, we also can address the uncertainty and the different behaviors existing in a multi-site organization, achieving more representative results.

This work shows the use of process simulation to provide evidences of the effort reduction provided by the use of a new test design process and supporting tool. Currently, software testing is being considered so important that organizations can allocate teams exclusively for testing activities [10]. We analyzed this use of process simulation in an industrial setting with multiple developing and testing sites distributed around the world.

The simulation model was created using expert opinion and data analysis. Actually, these models can be refined as the data is acquired through new experiments, pilots and case studies. We identified interesting advantages of using process simulation, such as its reduced costs and the achievement of more representative results, since we can simulate several different situations.

2 Test Design Processes

In this section, we model two different processes used to design test cases. The first one is the current process used to create test cases manually. The second process consider the use of tools that support Model-Based Testing, in which test cases are generated from the software specification written in a more formal notation or structure [7].

2.1 Manual Test Design

This process is used to create test cases manually, with no support for automatic test case generation. We modeled this process as shown in Figure 1. There are two types of roles in this process, the test designers and the reviewers.

The first activity of the test designers is the requirements analysis. Basically, the test designers read the requirements and any other source of information that helps to

describe the behavior of the application to be tested. The output of this activity is a skeleton of the application behavior, which summarizes how to navigate in the application, what are the alternative flows in case of errors, etc.

With the skeleton of the application behavior, the test designer is able to start the next activity, which goal is to write the test cases. These specifications are commonly written in natural language and they usually describe the test precondition, procedure (list of test steps with inputs and expected outputs) and post-condition [11]. The output of this activity is the set of specified tests (test suite).

Once the test cases are written, they need to be inspected in order to ensure their correctness, the conformity with writing standards and the quality of the text. This activity is detailed in Figure 2. First, two or more reviewers are responsible to read the test specifications and take notes about any identified problem. This activity is called inspection preparation.

After that, the identified problems are discussed with the test designers in one or more meetings. The goal of these meetings is to confirm all the identified problems. The test designers are then responsible to rework the test specifications in order to correct all the identified problems. Finally, the reworked test cases are validated by one of the reviewers to confirm that all problems were really solved.

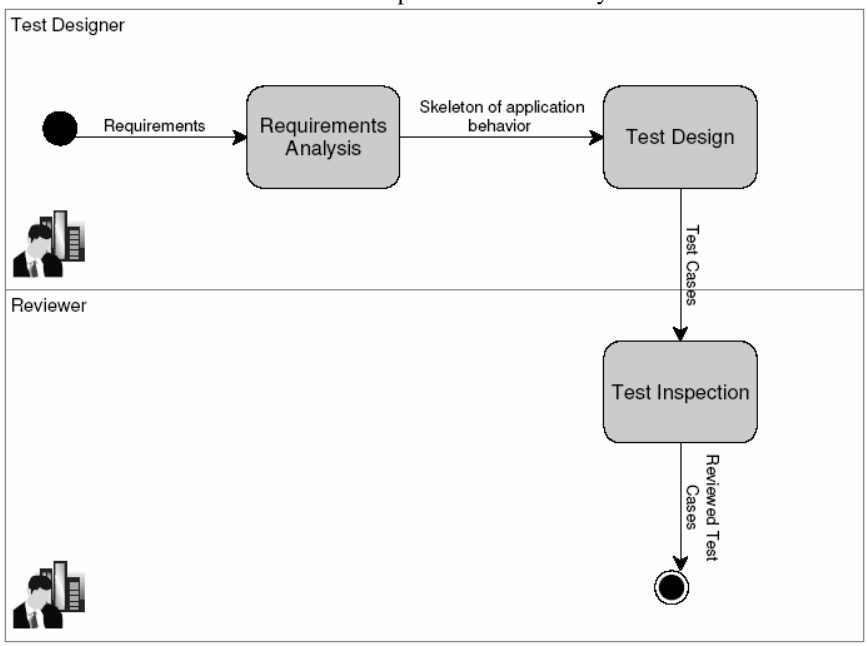

Fig. 1. Test design process for creating test cases manually.

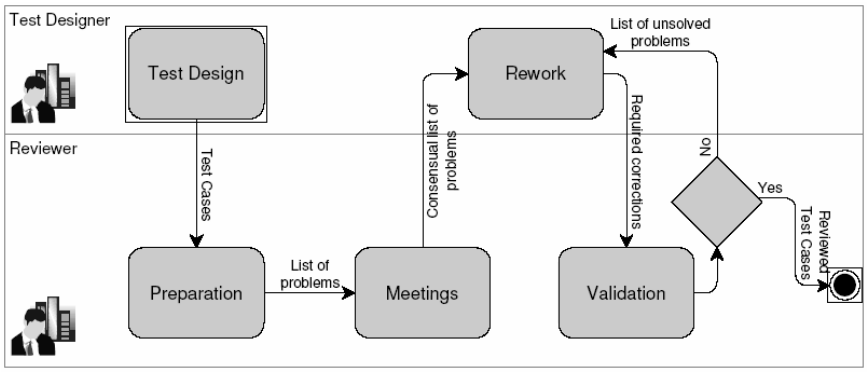

Fig. 2. Details of the test inspection activity.

2.2 Automated Test Design

Model-Based Testing (MBT) is a technique used to generate test cases from the application requirements. Using this approach, test designers concentrates their efforts in modeling the application behavior in a specific notation instead of in the writing of the test specifications. The notation to be used depends on the input required by the supporting tools.

In this work, we consider the use of the Test and Requirements Generation Tool (TaRGeT). TaRGeT is a tool for automatic test case generation from use case scenarios written in Natural Language (NL) [8]. It automates a systematic approach for dealing with use cases scenarios and test artifacts in an integrated way. The possible scenarios are specified as use cases using NL and a template that supports automatic processing.

The process required to use TaRGeT is presented in Figure 3. Similarly to the manual test design process, we have test designers and reviewers and the first activity is also the requirement analysis, performed in the same way as described in Section 2.1 for the manual test design process.

Then, we have the use cases writing activity. Basically, the test designers describe the application behavior writing use case flows, as shown in Figure 4. In this example of a messaging application, the main flow of the use case describes a scenario in which the user selects his favorite message and moves it to the hot message folder.

The alternative flow describes a scenario in which the message is not moved to the hot message folder because the message storage is full. The flows are described through steps, identified by a Step Id, and composed by a User Action, the respective System Response and System State (the necessary conditions to occur the system response).

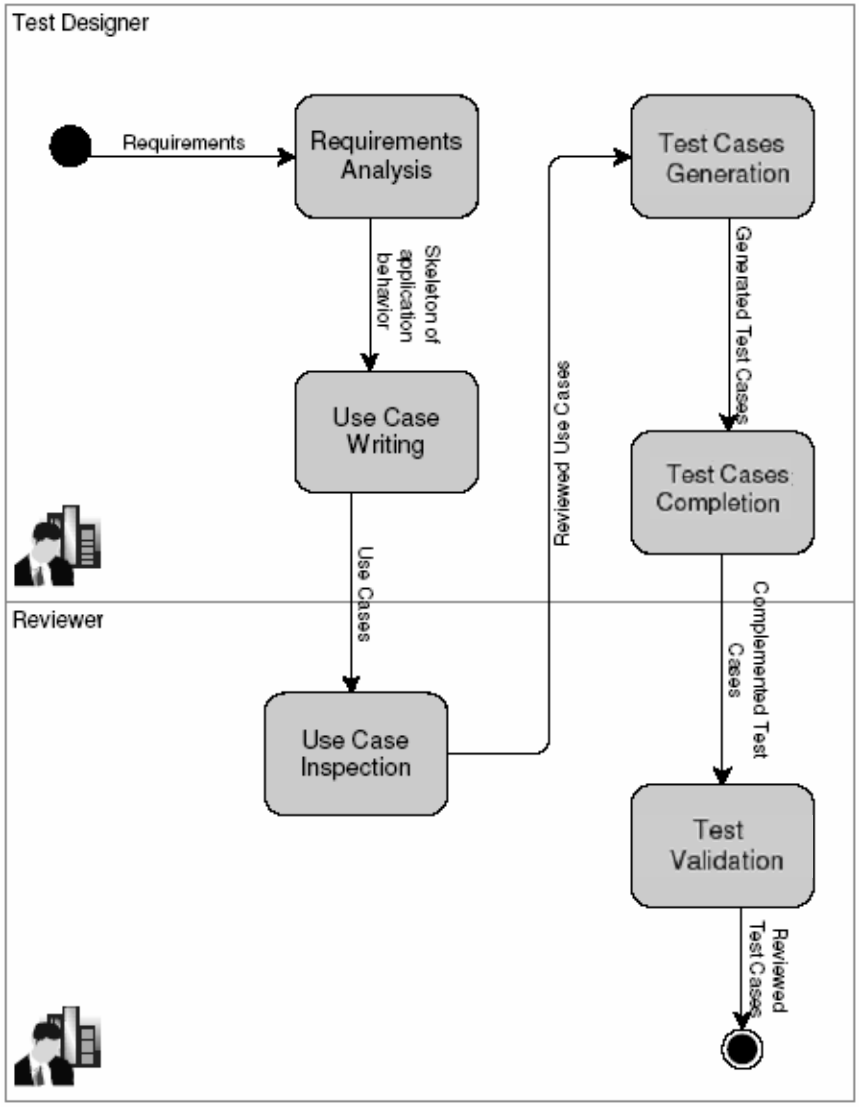

Fig. 3. Test design process for creating test cases using TaRGeT tool.

The use cases written by the test designers are then inspected by two or more reviewers. The activities that compose this inspection process are very similar to the ones described in Section 2.1 and, for this reason, it is not described here.

Main Flow

Description: The message is moved to Hot Messages folder

From Step: START

To Step: END

Step Id	User Action	System State	System Response
1M	Go to "Message Center"		"Hot Messages" folder is displayed
2M	Go to "Inbox"		All inbox messages are displayed
3M	Scroll to a message		Message is highlighted
4M	Go to "Context Sensitive Menu"		"Move to Hot Messages" option is displayed
5M	Select "Move to Hot Messages" option	Message storage is not full	"Message moved to Hot Messages folder" is displayed

Alternative Flows

Description: The message is not moved to Hot Messages folder because message storage is full

From Step: 5M

To Step: END

Step Id	User Action	System State	System Response
1A	Select "Move to Hot Messages" option	Message storage is full	"Memory required" dialog is displayed
2A	Confirm memory information dialog		Message content is displayed

Fig. 4. Use case flows used to describe the application behavior.

Based on the reviewed use cases, the test designers use TaRGeT to automatically generate the test cases. Due to some technical limitations, it is not possible to generate all the information required to execute the tests. For this reason, the test designers have to complement the generated test specifications with some additional information, such as some test preconditions or setup information. The information written manual must then be inspected. We called this last activity by test validation to differentiate from the test inspection activity of the manual test design process.

3. Assessment of Effort Reduction Using Processes Simulation

We want to use process simulation in order to assess the effort reduction that can be achieved using the automated test design process instead of using the manual test design process. For that, we need to create simulation models for these test design processes. For each process, we defined variables that characterize the artifacts used as inputs or outputs in the activities of the test design processes. Then, we modeled the effort spent in each activity based on the manipulated artifacts. Table 1 lists the

artifacts and describes the variables used to characterize each artifact used in the test design processes.

Table 1. Variables used to characterize the artifacts used in the test design processes.

Artifact	Variable	Description	Determined by
Requirements	Req	Number of requirements defined in the document.	
Skeleton of application behavior	SkF	Number of control flows identified in the skeleton.	Req
Use cases	UCF	Number of control flows written in the use cases.	SkF
	RwUCF	Number of control flows written in the use cases that required rework after inspection.	UCF
Test cases	MTC	Number of test cases created manually.	SkF
	RwMTC	Number of manually created test cases that required rework after inspection.	MTC
	GTC	Number of test cases generated automatically.	UCF
	CompGTC	Number of generated test cases that required a manual completion.	GTC
	RwGTC	Number of generated test cases that required rework after validation.	CompGTC

In addition, we are considering that the all the output artifacts of a process activity can be characterized only by analyzing the input artifacts (see column "Determined by" of Table 1). These relations are modeled as shown by the following sample equations.

$$\text{SkF} = \text{round}(\beta_1 * Req) \; . \tag{1}$$

$$\text{MTC} = \text{round}(\beta_2 * \text{SkF}) \; . \tag{2}$$

In Equation 1, the number of requirements (Req) is a variable that has values assigned during the simulation according to a given probability distribution, which basically describes the values and probabilities that a random event can take [12]. In this paper, variables having probabilistic distributions appear in italic in the equations.

The number of control flows identified in the skeleton (SkF) has then its value determined by Req and the number of test cases generated created manually (MTC) is determined by SkF (see equation 2). The variables β_1 and β_2 also have probabilistic distribution to represent the variance and uncertainty of the represented relations. We used the function round when the variables of the left side of the equation represent an integer value.

The used probabilistic distributions were defined using expert judgment and by the analysis of the available historical data. We analyzed the relation between the artifacts of previous projects and identified normal, triangular and other probabilistic

distributions that fitted the data or the expert opinion. Since the details of these distributions (types and parameters) are specific for the analyzed industrial setting and they do not compromise the contributions of this paper, we kept the privacy of this information.

As our goal is to assess the effort reduction by using the automated test design process, we need to analyze the effort required to perform each activity of both manual and automated test design processes. For that, we defined variables and equations related to effort that are considered in the simulation.

We calculate the effort spent in each activity based on the generated output artifacts. The Effort required to perform the Requirements Analysis activity (RAEffort) is given by multiplying the number of flows in the skeleton of the application behavior (SkF) by the average time required to write each flow (TSkF):

$$\text{RAEffort} = \text{SkF} * \mathit{TSkF} \ . \quad (\mathbf{3})$$

The variable TSkF and others used in the next equations model the uncertainty and variance related to the team productivity.

For calculating the effort spent in the Test Design activity of the manual process (TDEffort), we multiply the number of test cases created manually (MTC) by the average time required to write each test case (TMTC) manually:

$$\text{TDEffort} = \text{MTC} * \mathit{TMTC} \ . \quad (\mathbf{4})$$

The effort spent in the Test Inspection (TIEffort) is the sum of the efforts spent in preparation (PrepEffort), Meetings (MeetEffort), Rework (RwEffort) and Validation (ValidEffort):

$$\text{TIEffort} = \text{PrepEffort} + \text{MeetEffort} + \text{RwEffort} + \text{ValidEffort} \ . \quad (\mathbf{5})$$

$$\text{PrepEffort} = \text{MTC} * \mathit{TPrepTC} * \mathit{Reviewers} \ . \quad (\mathbf{6})$$

$$\text{MeetEffort} = \text{RwMTC} * \mathit{TMeetTC} * \mathit{Reviewers} \ . \quad (\mathbf{7})$$

$$\text{RwEffort} = \text{RwMTC} * \mathit{TRwTC} \ . \quad (\mathbf{8})$$

$$\text{ValidEffort} = \text{RwMTC} * \mathit{TValidTC} \ . \quad (\mathbf{9})$$

In equations 6 to 9, TPrepTC is the average time for reading a test case for the inspection, TMeetTC is the average time to discuss a test case in a meeting, TRwTC is the average time to rework a test case, TValidTC is the average time to verify if the corrections were made and Reviewers is the number of reviewers attending the inspection. We are considering that only one reviewer is responsible to validate the corrections of the test cases.

For the automated test design process, the effort spent in requirement analysis is considered the same as in the manual process. Also, the effort spent in the Use Case Inspection (UCInspEffort) and Test Validation (TVEffort) are calculated similarly to the Test Inspection effort for the manual test design process. For the Test Case Generation effort, we used the variable TCGEffort having a probabilistic distribution that represents the effort to use TaRGeT, which is practically independent of the size of its input (use cases).

For the Use Case Writing activity, we calculate the spent effort (UCEffort) multiplying the number of use case flows (UCF) by the average time required to write each of these flows (TUCF):

$$\text{UCEffort} = \text{UCF} * \textit{TUCF} \ . \quad (\mathbf{10})$$

For calculating the time spent in the Generated Test Cases Completion (CompGTCEffort), we consider the average time spent to analyze each generated test case (TAnalyzeGTC) and the average time spent to complement each test case with missing information (TCompGTC).

$$\text{CompGTCEffort} = \text{GTC} * \text{TAnalyzeGTC} + \text{CompGTC} * \text{TCompGTC} \ . \quad (\mathbf{11})$$

To support the analysis of the effort reduction provided by the automated test design process, we defined equations to calculate the effort spent in each test design process, as well the percentage gain of using the automated test design process:

$$\text{ManualProcessEffort} = \text{RAEffort} + \text{TDEffort} + \text{TIEffort} \ . \quad (\mathbf{12})$$

$$\text{AutomatedProcessEffort} = \text{RAEffort} + \text{UCEffort} + \text{UCInspEffort} + \textit{TCGEffort} + \text{CompGTCEffort} + \text{TVEffort} \ . \quad (\mathbf{13})$$

$$\text{Gain} = (\text{ManualProcessEffort} - \text{AutomatedProcessEffort}) / \text{ManualProcessEffort} \quad (\mathbf{14})$$

After running the simulation, we were able to perform several analyses. First, we analyzed the effort reduction by using the automated test design process instead the manual process and the differences of the effort distributions of both processes, as the sample graphs presented in Figure 5.

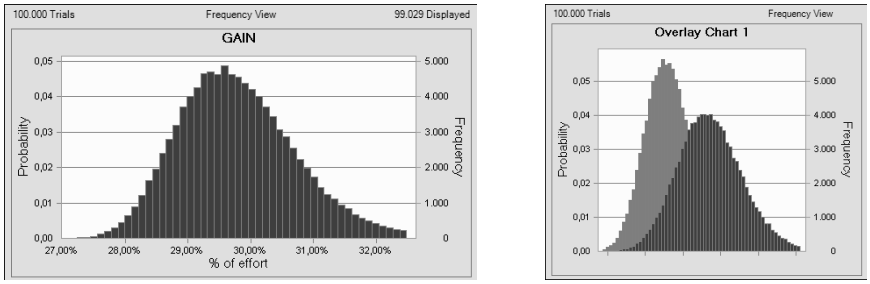

Fig. 5. Charts used to analyze the results of the simulation.

In addition, we analyzed and compare the performance of the test design processes with respect to the effort distribution among their activities.

4. Discussion

Several lessons were learned during the modeling and simulation of the manual and the automated test design processes. For the currently used process, we were able to get historical data and use it to support the definition of probabilistic distributions and equations. Some of the used data were stored in databases, such as the time spent in each activity. Others had to be manually calculated, such as the number of requirements per application. The equations were defined by experts and then calibrated and validated using the historical data.

Considering the automated test design process, we had few historical data from initial experiments and case studies using TaRGeT [7]. Therefore, it is more difficult to justify and to validate the probabilistic distributions and equations defined in the process model. We also could not consider only the opinion of experts from the TaRGeT development team in order to avoid bias in the results.

However, each new case study or experiment of TaRGeT can be used to validate or to adjust the process model. After adjusting the model, we had to repeat the simulation, which requires a low effort, since we only need to set the new parameters of the model, such as new probabilistic distributions.

We used two different tools for modeling and simulating the process due to the lack of support of both of them. In fact, even using these two different tools, it was hard to identify the values of the processes variables for the best and worst cases found by the simulation. Finally, the credibility and representativeness of the results depends on convincing managers about the correctness of the model construction.

5. Related Work

In this section we present related works that used simulation to compare different software strategies, techniques and metrics. In [5] and [9], the authors compared different prediction techniques using simulation. Basically, their works simulate data to create datasets with different characteristics. They compared the predictions techniques with these simulated datasets and identified that the best technique can be defined only for particular contexts. In our work, we create a larger number of different situations through the simulation of the artifacts used in two different test design processes. We also modeled and simulated the effort spent in each of their activities.

Another related work used simulation to compare two specification level software size metrics, the Albrecht ’s Function Point and the DeMarco’s Function Bang [4]. The authors generated randomized dataflow diagrams representing different artifacts to be measured, allowing sufficient sampling sizes to make statistical analysis of data. In our work, we used simulation to generated and analyze artifacts with different characteristics. However, we also generated different teams productivity to evaluate the effort spent during the execution of the processes.

6. Conclusions

In this paper, we showed the use of process simulation to address the problem of assessing the test design effort reduction provided by a Model-Based Testing (MBT) approach. We modeled both manual and automated test design processes. For the automated process, we modeled the process supported by the Test and Requirements Generation Tool (TaRGeT).

We used process simulation to provide evidences of this effort reduction in a multi-site industry. For that, we created a simulation model that characterized the artifacts produced and manipulated during the processes. Then, we modeled the effort spent in each processes activity based on these artifacts. The modeling of each individual process activity make easy the model construction and validation process. All these models were created using expert opinion and data analysis. Actually, these models can be refined as the data is acquired through new experiments, pilots and case studies.

We identified interesting advantages of using process simulation, such as its reduced costs and the achievement of more representative results, since we can simulate several different situations. We also show some drawbacks of this approach, such as the difficult to create models close to the reality and the lack of support of the process simulation tools.

We also believe that similar works can be done to provide evidences about the benefits of solutions for other software development problems. After this study, we believe that process simulation is an interesting alternative to evaluate the impact of using new technologies that require changes in the processes.

References

1. Angelis, L., Stamelos, I.: A Simulation Tool for Efficient Analogy Based Cost Estimation. Empirical Software Engineering. 5(1), 35--68 (2000)
2. Ebert, C., De Neve, P.: Surviving Global Software Development. IEEE Software. 18(2), 62--69 (2001)
3. Kellner, M.I., Madachy, R.J., Raffo D.M.: Software process simulation modeling: Why? What? How?. Journal of Systems and Software, 46(2), 91--105 (1999)
4. Rask, R., Laamanen, P., Lyytinen, K.: Simulation and Comparison of Albrecht's Function Point and Demarco's Function Bang Metrics in a CASE Environment. IEEE Transactions on Software Engineering. 19(7), 661--671 (1993)
5. Shepperd, M., Kadoda, G.: Comparing Software Prediction Techniques Using Simulation. IEEE Transactions on Software Engineering. 27(11), 1014--1022 (2001)
6. Alexander Pretschner. Model-based testing. In ICSE '05: Proceedings of the 27th international conference on Software engineering, pp. 722--723. IEEE Press, St. Louis (2005)
7. Nogueira, S., Cartaxo, E., Torres, D., Aranha, E., Marques, R.: Model based test generation: A case study. In: 1st Brazilian Workshop on Systematic and Automated Software Testing, Recife (2007)
8. Schwitter, R: English as a formal specification language. In: 13th International Workshop on Database and Expert Systems Applications (DEXA02), pp. 228--232 (2002)

9. Shepperd, M., Kadoda, G.: Using Simulation to Evaluate Prediction Techniques. In: IEEE METRICS 2001. IEEE Press, Los Alamitos (2001)
10. Broekman, B., Notenboom, E.: Testing Embedded Software. Addison-Wesley (2002)
11. Jorgensen, P: Software Testing, A Craftsmans Approach. CRC Press (2002)
12. Maxwell, K: Applied Statistics for Software Managers. Prentice Hall (2002)