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 evidence 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 possibility to analyze multiple scenarios. We also show some drawbacks of this approach, such as the difficult to create models close to the reality and the lack of processes simulation and comparison support.

Keywords: process simulation, test design effort, model-based testing, process improvement assessment, technology adoption.

1 Introduction

In industry, we usually need empirical evidence of the benefits of a given solution in order to convince people to use it. For instance, Model-Based Testing (MBT) is a testing approach in which test cases are automatically generated from software specifications written in a more formal notation or structure that can be processed by a tool. In this way, MBT can significantly improve test coverage and reduce test design effort [7] [9]. However, although these improvements are expected to occur, they need to be quantified to verify if these benefits justify the effort to change the current process 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 availability of additional resources.

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 of pilot studies, we also can address the uncertainty and the different possible situations existing in a multi-site organization, achieving more representative results through the analysis of these multiple scenarios.

Currently, software testing is being considered so important that organizations can allocate teams exclusively for testing activities [12]. In this context, 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 its supporting tool, called TaRGeT [9].

We run the process simulation considering an industrial setting with multiple developing and testing sites distributed around the world. The simulation model was created using data analysis and expert opinion. Actually, these models can always be refined as data is acquired from new experiments, pilots and case studies. This paper also highlights interesting advantages of using process simulation, such as its reduced costs and the analysis of multiple scenarios. It also presents some drawbacks of this approach, such as the difficulty to create the simulation models.

2 Test Design Processes

In this section, we model two different processes used to design test cases. The first one is the process currently used to create test cases manually in a real company. The second process considers the use of TaRGeT [9], a tool developed to support a Model-Based Testing approach. Both test design processes are presented here using the notation of the process modeling tool used in this work [16].

2.1 Manual Test Design

This process is used to create test cases manually. This probably still the most common process used in industry due to the current lack of appropriate tool support for functional test 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 [13]. The output of this activity is the set of specified tests (test suite).

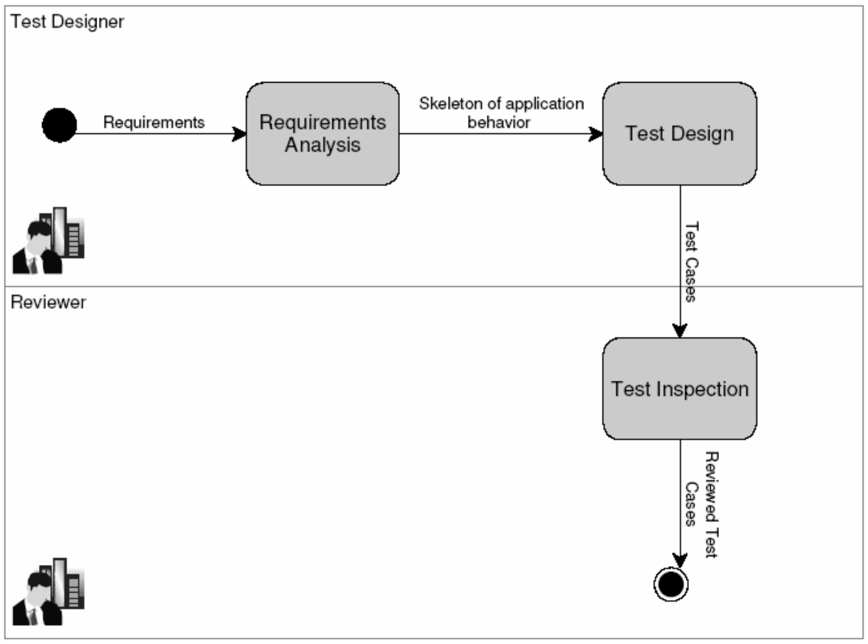


Fig. 1. Test design process for creating test cases manually.

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 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 reported problems. Finally, the reworked test cases are validated by one of the reviewers to confirm that the problems were really solved.

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 concentrate their efforts in modeling the application behavior using a specific notation instead of in the writing of the test specifications. In general, the notation to be used depends on the input required by the MBT supporting tool.

In this work, we consider the use of the Test and Requirements Generation Tool (TaRGeT), which is a MBT supporting tool for automatic test case generation from use case scenarios written in Natural Language (NL) [10]. This tool is a result of several research studies related to MBT [9]. In summary, TaRGeT 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 structured template that supports automatic processing.

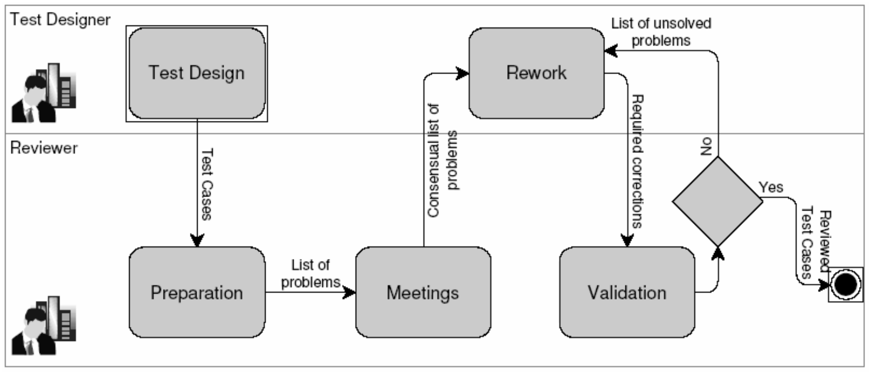


Fig. 2. Details of the test inspection activity.

The process required to use TaRGeT is presented in Figure 3. Similarly to the manual test design process, we have the roles of 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 Case 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 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). Although the use cases written by the testers can be based on use cases written by developers to save effort, we do not consider this possibility in this work because some development methodologies may not include the creation of use cases.

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

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. After that, the information written manually in the test specification must be inspected. We called this last activity by test validation to differentiate from the test inspection activity of the manual test design process.

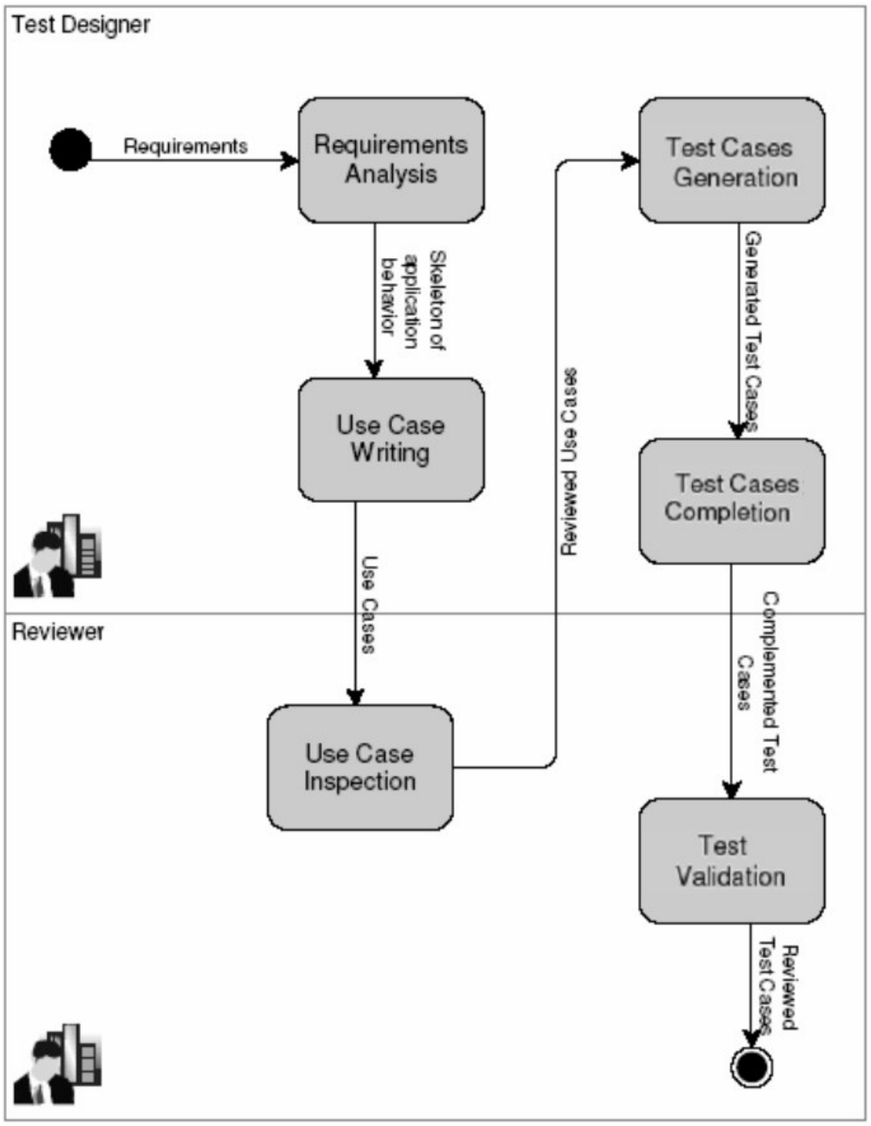
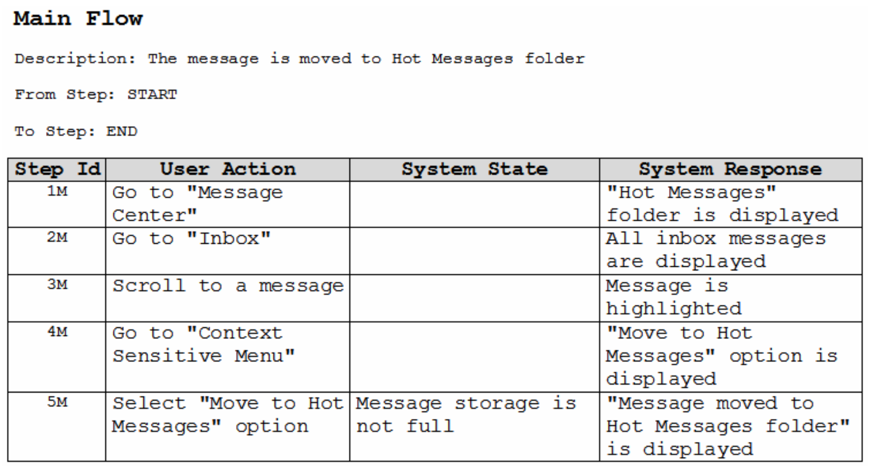


Fig. 3. Test design process for generating test cases automatically.



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

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

3 Assessment of Effort Reduction Using Processes Simulation

We want to use process simulation to assess the effort reduction achieved by using the automated test design process instead of using the manual test design process. For that, we created and run simulation models for these test design processes, as described next.

3.1 Simulation Models

For each process, we defined variables that characterize the artifacts used as inputs or outputs of the activities defined in the test design processes. Then, we modeled the effort spent in each activity based on the generated or updated artifacts (activity output). Table 1 lists the artifacts artifact used in the test design processes and describes the variables used to characterize them.

In addition, we are considering that 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 sample Equations 1 and 2.

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

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

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

In Equation 1, the number of requirements (*Req*) is a variable that has values assigned during the simulation to represent different situations, such as documents with small, medium or large number of requirements. These values are chosen according to a probability distribution, which basically describes the values and probabilities that the random event (e.g., size of the requirements) can take [14]. Then, the number of control flows created in the skeleton of the application behavior (SkF) is determined by Req and β_1, which represents the intrinsic variance and uncertainty of the (linear) relation between SkF and Req. We use the function round to ensure that only integer values are assigned to SkF.

Similarly, we see in Equation 2 that the number of test cases created manually (MTC) is determined by the number of control flows (SkF) and the intrinsic variance and uncertainty (β_2) of this other relation. To easy the reading of this paper, variables having probabilistic distribution (random values assigned during the simulation) appear in italic in the equations and texts.

The probabilistic distributions to use in the simulation model are defined using expert judgment and by the analysis of the available historical data. For instance, we analyzed the relation between the artifacts created in previous projects and then used normal, triangular and other probabilistic distributions that best fitted the data or the expert opinion. Regarding the automated test design approach, the only historical data available were extracted from studies carried out by the TaRGeT development team.

This lack of data is a characteristic of new proposed technologies. Since the details of these distributions (types and parameters) are specific for the analyzed industrial setting, we kept the privacy of this information without compromising the contributions of this paper.

As our goal is to assess the effort reduction by using the automated test design process, we need to model the effort required to perform the activities of both manual and automated test design processes. For that, we defined variables and equations related to the effort spent in the activities presented in Section 2.

Basically, we calculate the effort spent in each activity based on their output artifacts. For instance, 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*):

$$\text{RAEffort} = \text{SkF} * \mathit{TSkF}\ . \qquad \textbf{(3)}$$

The variable *TSkF* and some others used in the next equations represent the time required to generate output artifacts. Probabilistic distributions are assigned to these variables in order to model the uncertainty and variance related to the team productivity for the considered activity of the process.

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:

$$\text{TDEffort} = \text{MTC} * \mathit{TMTC}\ . \qquad \textbf{(4)}$$

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):

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

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

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

$$\text{RwEffort} = \text{RwMTC} * \mathit{TRwTC}\ . \qquad \textbf{(8)}$$

$$\text{ValidEffort} = \text{RwMTC} * \mathit{TValidTC}\ . \qquad \textbf{(9)}$$

Where:

- *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 done.
- *Reviewers* is the number of reviewers attending the inspection.

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) activities are calculated similarly to the Test Inspection effort for the manual test design process (Equation 5). For modeling the effort of the Test Case Generation activity (*TCGEffort*), we used a probabilistic distribution that represents the effort to use TaRGeT (create a project, import the use cases and generate the tests), which is practically independent of the size of the use cases.

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

$$\text{UCEffort} = \text{UCF} * \mathit{TUCF}\ . \qquad (\mathbf{10})$$

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*).

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

Finally, 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 (EffortReduction) of using the automated test design process:

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

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

$$\text{EffortReduction} = (\text{ManualProcessEffort} - \text{AutomatedProcessEffort}) / \text{ManualProcessEffort}\ . \qquad (\mathbf{14})$$

Due to some limitations of the tool used for modeling the process, it did not support the creation of the presented simulation model. Then, we created and run the simulation model in a general purpose simulation tool [15].

3.2 Simulation Results

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, which shows the effort reduction (gain) by using the automated process and the distribution of the effort spent in both process during the simulated situations.

In addition, we analyzed and compare the performance of the test design processes with respect to the effort distribution among their activities. Descriptive statistics and the data generated during the simulation were also a valuable source of information to analyze. Finally, we were able to change the probabilistic distribution of some variables to represent specific situations under investigation, such as during the adoption of the new technology (high productivity with the manual process and low productivity with the automated process).

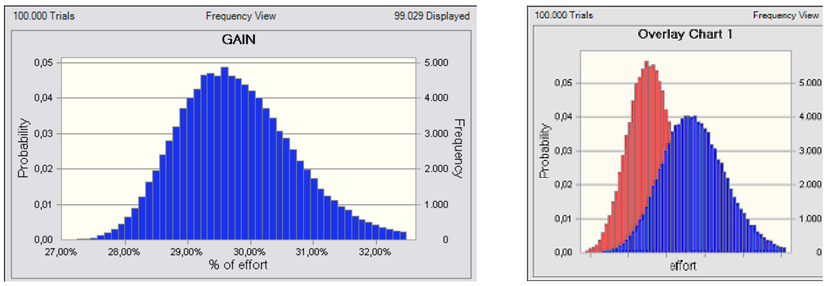


Fig. 5. Sample charts used to analyze the results of the simulation (with illustrative data).

4 Advantages and Drawbacks of This Approach

In this section, we summarize and generalize the advantages and drawbacks of using process simulation to compare two different processes. Regarding the advantages, we verified that:

- The process simulation usually has a low cost than pilot studies.
- Usually, we have historical data about the currently used process and this information can be used to create and validate its simulation model.
- The execution of new empirical studies (pilot projects, case studies, experiments, etc.) can generate data to incrementally calibrate the simulation models. Then, these models can be simulated again at a very low cost to provide more accurate results.
- We can perform the analysis of multiple scenarios, analyzing the application of new processes in specific situations.
- An analysis per process activity can highlight points in the process to improve.

Although all these advantages, some drawbacks of using process simulation to compare processes must be observed:

- Difficulty to create models close to reality. The validity of its results strongly depends on the correct simulation model construction. It may be a challenge to convince managers about the correctness of the model construction.
- New processes have only few data from expert opinion and initial experiments run by the developers of the new process (possibly biased information).
- The relationship between input and output artifacts of the process activities may be non-linear, making more difficult the modeling of these relations.
- The simulation models are valid only for the aspect under investigation, such as effort reduction, and not for others, such as the creation of effective tests.

- Process modeling and simulation tools may be too complex without providing the needed resources.

5 Related Work

In [6], the authors simulated a software development process to analyze different strategies, such as iterative development, pair programming and test automation. They defined parameters to represent the effort to perform the activities in the process (team productivity), the defect introduction rate, the probability of finding bugs, etc. They identified situations were the use of these strategies can provide benefits in terms of effort and bug detection. In our work, we study the automation of the test design process using MBT. The generated tests can then be automated or executed manually.

In [5] and [11], 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 simulation to assess the effort spent in two different test design processes.

Mutation testing [8] is a technique to simulate the introduction of software defects in order to evaluate the best set of tests cases able to reveal these defects (kill the mutants). While this paper only considers the effort reduction criteria to assess the test design approaches, Mutant testing can be used to assess these approaches considering the effectiveness of the created tests.

6 Conclusions

In this paper, we presented 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.

Acknowledgments. We would like to thank all anonymous reviewers who have helped us to improve this paper through their valuable comments. Also, the first author is a PhD candidate partially supported by Motorola, grant BCT-0021-1.03/05, through the Motorola Brazil Test Center Research Project. The second author is partially supported by CNPq, grant 306196/2004-2.

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. Ur, S., Yom-Tov, E., Wernick, P.: An Open Source Simulation Model of Software Testing, Hardware and Software, Verification and Testing. In: Ur, S.; Bin, E.; Wolfsthal, Y. (eds.) Hardware and Software, Verification and Testing 2006. LNCS, vol. 3875, pp. 124–137, Springer, Heidelberg (2006)
7. Alexander Pretschner. Model-based testing. In: 27th international conference on Software engineering, pp. 722–723. IEEE Press, St. Louis (2005)
8. Fabbri, S.C.P.F.; Maldonado, J.C.; Masiero, P.C.; Delamaro, M.E.: Mutation analysis testing for finite state machines. In: 5th International Symposium on Software Reliability Engineering, pp. 220–229, IEEE Press, Monterey (1994)
9. 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)
10. Schwitter, R: English as a formal specification language. In: 13th International Workshop on Database and Expert Systems Applications (DEXA02), pp. 228--232 (2002)
11. Shepperd, M., Kadoda, G.: Using Simulation to Evaluate Prediction Techniques. In: IEEE METRICS 2001. IEEE Press, Los Alamitos (2001)
12. Broekman, B., Notenboom, E.: Testing Embedded Software. Addison-Wesley (2002)
13. Jorgensen, P: Software Testing, A Craftsmans Approach. CRC Press (2002)
14. Maxwell, K: Applied Statistics for Software Managers. Prentice Hall (2002)
15. Crystal Ball, http://www.crystalball.com.
16. Metastorm Provision, http://www.metastorm.com.