A case study of metric-based and scenario-driven black-box testing for SAP projects

Maya Daneva, Alain Abran**, Olga Ormandjieva***, Manar Abu Talib****

*University of Twente, **Uiversité du Québec à Montreal, ***Concordia University

m.daneva@utwente.nl, alain.abran@etsmtl.ca, ormandj@cse.concordia.ca, m_abutal@cse.concordia.ca

Abstract:

Enterprise Resource Planning (ERP) projects are perceived as mission-critical initiatives in many organizations. They are parts of business transformation programs and are instrumental in improving organizational performance. In ERP implementations, testing is an activity that is crucial in order to ensure that the functionality embedded in the solution matches the business users' requirements. However, little is known about how to make the testing process more predictable or how to increase its chances of success.

This paper makes a first attempt towards improving the quality of the testing process in ERP projects by using a metric-based test case selection approach. The paper reports on how this approach was adapted to an ERP package-specific project context, how it was applied in five settings in a mid-sized project and what was learnt about using it.

Keywords

Black-Box Scenario-based Software Testing, Enterprise Resource Planning, COSMIC-FFP

1 Introduction

ERP testing is the process by which configured ERP transactions are executed in a controlled fashion. Its objective is to validate whether or not the ERP solution behaves as defined in the business requirements document that has been signed off by business representatives in the early stage of an ERP implementation project. It can be an expensive and labor-intensive process, for both systems testers and business users. Industry trend observers [7,11,15] estimate that, for example, when following the Accelerated SAP methodology for rapid SAP implementation, it can take twenty testers four weeks to manually run a test cycle of an end-to-end business process (or a similar test) consisting of 100 transactions. With four testing cycles per year, organizations can expect to spend 12,800 hours manually testing, at a cost of $499,200 (based on average salaries of $60,000). Acknowledging that there is a relationship between the quality of the test process and the final quality of the solution being delivered, improvement of the ERP test process is essential for project success in any organization adopting ERP. The state of the art in ERP testing practice suggests that testers "break" the system that is being tested into

classes of test cases using guidelines and their experience. They then choose the most effective classes of test cases that have previously been found when uncovering defects in other systems. However, this traditional common-sense approach cannot entirely serve the needs of ERP project teams, as the client organization's testers often do not have any previous ERP testing experience or even knowledge of ERP testing in general.

In this paper, we take a first step towards systematically improving the ERP testing process by using a metric-based approach, where "metric" means the distance between two elements in a set of test cases, defined in terms of their similarity and dissimilarity. We draw on earlier experience [2] gained by three of the authors (Abu Talib, Ormandjieva and Abran) in metric-based and scenario-driven black-box testing as applied to COSMIC-FFP [1] functional size measurement projects using use-case modeling and UML [8] in the analysis phase. In [2], Abu Talib et al. investigated how a set of test cases and test plans can be built, partitioned and prioritized early in a software project for which the requirements are documented by means of the use-case scenarios and the corresponding UML sequence diagrams. Because the typical requirements engineering (RE) process for ERP delivers business requirements in terms of scenario models [4], we felt intuitively that the testing method proposed by Abu Talib et al. [2] looked like a good candidate to be considered for use in ERP testing. Our choice of this method is also motivated by earlier conclusions drawn by the authors [2] suggesting that, for a given testing strategy and budget constraints, the approach allows the quality of the testing processes to be maximized, and, as a consequence, our confidence in the quality of the tested product to be increased.

Below, we report on how we adapted this approach to ERP project settings and how we applied it to improve ERP testing. Aspects of the testing process were analyzed in the specific context of projects implementing the SAP R/3 package, a leading product in the ERP market.

The remainder of this paper is organized as follows: in section 2, we provide background on the state-of-the-art ERP testing process and its challenges, summarize the metric-based test selection method [2] that we propose for improving the current ERP testing practice and present our research approach; in section 3, we comment on an adaptation of the testing method [2] to SAP settings and the early findings that resulted from of its application are discussed; in section 4, our work is linked to existing publications in the area of both ERP and non-ERP testing; in section 5, two validation threats are evaluated; and, in a section 6, our early conclusions and future research priorities are presented.

2 Background

In this section, the notions of SAP testing and COMIC-FFP black-box testing are introduced.

2.1 SAP testing: process and issues

ERP vendors (e.g. SAP, PeopleSoft, Oracle) provide their client organizations with a package-specific standard testing process that is part of the package implementation methodology they suggest for the project. In SAP project settings, the testing process that a client company is likely to adopt is part of the Accelerated SAP methodology for rapid SAP implementation [10]. SAP testing, like the testing of any complex system, includes two distinct phases: integration, which is mostly concerned with finding defects in the ERP solution, and release testing, which is concerned with demonstrating whether or not the solution meets its business user requirements. The latter is known as "black-box" testing, where: (a) test cases are derived from requirements specifications; and (b) the behavior of the system is analyzed by its inputs and outputs. The primary goal of this process is to build up and increase the ERP adopter's confidence that the system, built by the consultants, meets its requirements. This involves showing that the system in fact embeds the specified functionality, and that it will exhibit a specified level of performance and not fail under normal conditions of use.

The SAP testing process deploys a requirements-based testing concept. The process is tightly linked with the SAP RE process and embodies the following principles:

- ERP RE postulates that the ERP requirements should be associated with the building elements of scenario process models [5], which ensures that the requirements are testable.
- ERP RE teams are prompted to document the requirements for their solutions in such a way that it is possible for testers to derive test cases and for business users to check that the requirements are indeed satisfied.
- Test cases are derived directly from the elementary, logical components of the SAP solution described in the business requirements specifications (which is the key deliverable from the RE process).
- Testing teams focus solely on end-user acceptance testing, with the assumption that the ERP vendor's internal software testing had already taken care of application bugs.

Typically, the SAP business requirements documents specify business scenarios represented in terms of Event-driven Process Chains (EPC) consisting of processes, events and logical connectors [4]. Thus, from the perspective of business users in a company, a process describes what an organizational unit does, and, from the perspective of the SAP R/3 System, a process means a physical transaction (which is a piece of code). For example, "Enter Material" is a user-

recognizable piece of business functionality and it also corresponds to transaction MM01 in the SAP Materials Management module. An example from a SAP scenario model is illustrated in Figure 1.

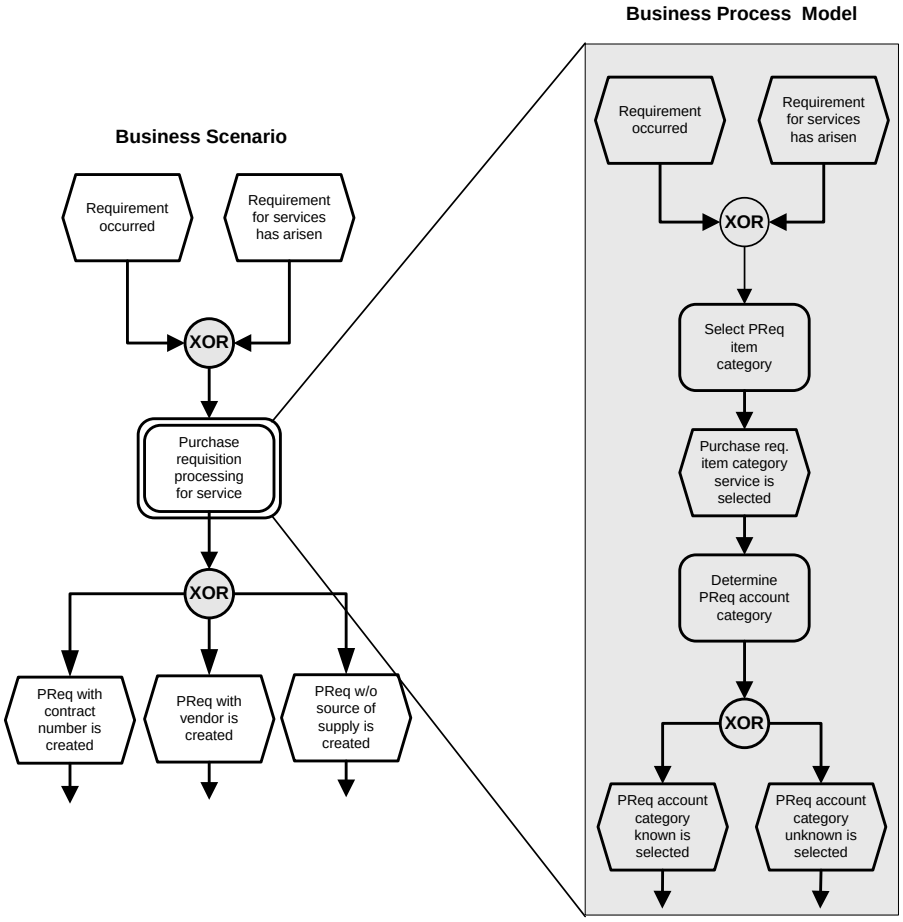

Figure 1: An example of a scenario process model

Like the SAP RE process, the SAP testing process relies exclusively on the concept of refining these business process scenarios into business processes, then mapping a process into process variants, and, finally, transforming each business function that makes up a process variant into a test case (Figure 2).

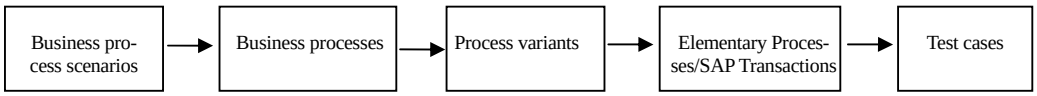

Figure 2: SAP Testing with eCATT

In other words, the test case models a complete business process, with all its prerequisites for a particular test.

The SAP testing process requires that a testing environment be set up separately from the configured solution and be in a ready-to-use state prior to testing. This environment is a replication of the productive system (including all its stored data), and serves to simulate customized developments and evaluate them against real data. Experiments [10] indicate that replicating the productive system is a time-consuming task requiring a great deal of storage and generating ongoing hardware costs. Moreover, the test manager will need to ensure that all the necessary data (master, transactional, test) are properly loaded as part of the assessment for the test-readiness review. To support clients in testing activities, SAP offers (as part of the SAP package) a suite of testing tools, the so-called SAP Test Workbench. These tools are used to create, analyze and monitor manual and automatic test cases, as well as tracking their progress and systematically archiving them. It allows clients to integrate cases and test scripts from non-SAP providers in test plans. It also ensures that the tests are reused each time the application is modified. The suite consists of: (i) Test Data Migration software, a data transfer tool enabling ERP adopters to extract representative application data from a productive system and standardize the set-up of non-productive systems; and (ii) the extended Computer-Aided Testing Tool (eCATT) for generating, recording, managing and executing test cases.

Despite the supporting SAP standards and tools, SAP testers receive very little guidance on: (a) what to do in order to test fragments of the SAP solution in a cost-effective way; and (b) how to prioritize test cases in the face of shrinking testing budgets. The quality of the decision-making processes related to these two questions depends heavily on the tester's judgment based on his/her personal testing experiences with SAP [10]. Moreover, an ERP market trend indicates that ERP solutions are becoming increasingly complex and are interconnecting with increasingly intricate coordination support systems across multiple operating environments. With the growing business demand for more cross-organizational cooperation and sharing of data and processes, coupled with the advent of newer, more productive integration concepts, like the Service Oriented Architecture, it is evident that more pieces of ERP functionality are being put together in very short periods of time. ERP users are expecting solutions to be tested to work the first time and in a 24/7 mode. They learned from earlier experiences [12] that ERP system outages are visible and costly in both the short and the long run. At the same time, ERP testing has expanded to the point that a rapidly growing testing services market has emerged. However, little or almost no evidence from real-life projects suggests that these developments in the ERP testing field make it easier for testers to confront the above two issues (labeled (a) and (b) above). This research targets the adaptation of the COSMIC-FFP back-box scenario-based testing method in-

troduced below for optimizing ERP testing quality while respecting budgetary constraints.

2.2 COSMIC-FFP-based black-box scenario-driven testing method

For the purpose of our case study research efforts, we have chosen to use COSMIC-FFP-based black-box scenario-driven testing [2]. The basic idea of this testing is to first apply the COSMIC-FFP functional size measurement method and count the COSMIC Full Function Points (FFP) of an application, the requirements of which are specified by scenarios. Then, as a by-product of the counting process, an exact inventory of the application usage scenarios and their elementary building blocks is carried out. In order to do this, we first map the scenarios to sequences of events, and, second, we divide them into equivalent classes of the input domain. Here, a domain refers to the input/output behavior space partitioned into sub-domains so that the behavior of software on the inputs/outputs of a sub-domain is similar. The scenarios within each equivalent class are prioritized for testing purposes based on the amount of information they carry to or from the system. The test cases are finally selected from all equivalence classes based on their priority level, which allows the highest level of coverage (and therefore testing quality) to be achieved within given budget constraints. This testing method includes the following steps: (i) generation of test cases through mapping the scenarios to a sequence of events in time (or data movements in COSMIC-FFP); (ii) partitioning test cases into equivalence classes by using a metric-based partitioning algorithm, where the partitioning criterion is the distance between the test cases based of their similarity and dissimilarity; (iii) prioritization of test cases based on functional complexity measurement in the COSMIC-FFP context; and (iv) selection of an optimal subset of test cases in terms of maximizing the test suite coverage within a given budget by using a budget-related algorithm.

Partitioning. The metric defined for the information distance between two test cases, for example t1 and t2, rests on the similarity/dissimilarity between their event sequences, and is calculated using the length of the longest common prefix (LCP) and the minimum amount of change necessary to convert test case t1 into test case t2. The distance between the two test cases, t1 and t2 (t2 is the test case that might be chosen next), is calculated as follows:

$$td(t1, t2) = similarity(t1, t2) * dissimilarity(t1, t2) * e^{-l}$$

where

$$l = (length(t1)/length(t2))$$

$$similarity(t1, t2) = 2^{(-\,length\,(LCP\,(t1, t2))}$$

LCP is the longest common subsequence of the two test cases starting from a triggering event. This metric is used to partition the sets of test cases into equivalent

classes, where an "equivalent" relation is defined in terms of the distance between the test cases, which has to be at most ε_{max}. The information distance is chosen because it is an absolute and objective quantification of the distance between individual objects.

Prioritization. The priority of the test cases for the scenarios is assigned based on the complexity of their functionality, which is defined as an amount of information handled by the events in one scenario [2]. Functional complexity characterizes the dynamic performance of the system seen as a sequence of events required to fulfill a certain piece of functionality of the system. The concepts of information theory are applied to measure the amount of work performed in a time slice by the system in terms of the amount of information in the event sequence. For the purposes of functional complexity measurement in the COSMIC-FFP context, different types of events to which the system must respond in some time interval were considered, as specified in the scenarios of usage of the software. Intuitively, the greater the variety and number of these events, the more complex the functionality. The probability of occurrence of the i-th most frequently occurring event is equal to the percentage of the total number of event occurrences it contributes, and is calculated as $p_i = f_i / NE$, where f_i is the number of occurrences of the *i*-th event and *NE* is the total number of events in the sequence. The classical entropy calculation quantifies the average amount of information contributed by each event. Therefore, the functional complexity (FC) in a time slice is defined as an average amount of information in the corresponding sequence of events, and is computed as follows:

$$FC = -\sum_{i=1}^{n} (f_i / NE) \log_2 (f_i / NE) \quad \ldots\ldots (3)$$

The functional complexity in a period of time with higher average information content should, on the whole, be more complex than another with lower average information content. That is, the FC measure is intended to order the uses of a system in a time period in relation to their functional complexity.

Optimal Test Set Selection. The authors in [2] assume that the total resource consumption is directly proportional to the number of the test cases selected. As a result, the total cost of the test set is calculated by multiplying the number of selected test cases by C, where C is a positive constant scalar denoting the cost of one test case. In order to select the optimal subset of test cases that will be characterized by the highest test coverage, the budget and the priority of the test cases are balanced as follows:

For all nonempty equivalence classes TSi

Step 1. Choose the highest-priority test case from the equivalence class *TSi*;

Step 2. Add the chosen test case to the *Optimal SubSet* and remove it from the equivalence class *TSi*;

Step 3. Increase the total testing cost in *C*;

Step 4. If (the total testing cost exceeds a given budget C*max)*

then

End the algorithm

End For

We decided to use this testing method because: (i) it rests on a model-driven RE process as the SAP project context does; (ii) its proof-of-concept has been demonstrated [2]; and (iii) it relies on an inventory of the elementary functional components of the system under development taken by a standardized functional size measurement technique, namely COSMIC-FFP [1].

3 Research method

The objective of our paper is to answer two research questions (RQ):

RQ1: How can the COSMIC-FFP testing method be adapted to the SAP testing process?

RQ2: Where in the SAP testing process can the newly introduced method create benefits for the testing team?

Our research method consists of two parts: first, we map the logical components of the testing strategy from the UML settings to the SAP settings, and then we carry out a case study to try out our approach to improve ERP test-case partitioning, selection and prioritization.

The unit of analysis in our case study is the testing strategy. Its application is observed in five SAP subproject settings [3, 4] which implemented five business scenarios based on the packaged SAP functionality. These are: External Service Procurement, Maintenance of a Service Master Record, Procurement of Stock Materials, Maintenance of a Materials Master Record and Procurement of Consumable Materials.

4 The case study

4.1 The SAP testing method

The adaptation of the approach in [2] to the SAP settings included:

(i) the replacement of use-case scenarios by SAP scenarios, which are the building blocks of the SAP requirements document,

(ii) the use of the International Function Point Users Group (IFPUG) method of Function Points Analysis [3] instead of the COSMIC FFP method (this was justified by the fact that FP counts already existed in the organization prior to carrying out this case study and that no one on board was familiar with the COSMIC FFP method), and

(iii) mapping the steps of the approach by Abu Talib et al. [2] to the steps being carried out using the standard SAP tools.

The results of this exercise are sets of rules for practitioners which help him/her translate the testing method of Abu Talib et al. [2] in the terms of SAP project artifacts. The adapted approach includes these steps:

First, the set of SAP scenarios that served as an input to FPA constitutes the set of test cases required to form the test set to be generated.

Second, that set of test cases is partitioned into equivalence classes.

Third, the test selection algorithm is run on the set of non-empty equivalence classes. The result is a set of selected test cases that ensures the best possible coverage [2].

However, as Abu Talib et al. [2] have already indicated, we cannot claim exhaustive fault coverage, since this algorithm maximizes test coverage within the limits of a given budget.

4.2 Application and findings

The adapted approach was applied to a case-study setting with five subprojects in a medium-sized project upgrading the SAP Materials Management module in a telecommunications company. The business process requirements for this project were documented in the form of event-driven process chains using the LiveModel tool [3], which assisted in automatically mapping the company-specific process diagrams to the transactions of the SAP software package. Using this tool, a user can navigate from a business scenario process, to an elementary process, to a transaction, to a screen, and ultimately to a test case. The use of the LiveModel tool ensured that the inventory of the transactions to be tested was carried out quickly. For each scenario, this tool made it possible to generate a mapping between the logical steps in the scenario and the unique identifiers of all the transactions that will be activated by a business user when executing these steps. Thus, visualization of a scenario using the LiveModel tool made it possible to obtain an

inventory of the relevant transactions and to clearly see the order of their execution. In other words, for each scenario, the LiveModel tool identified the elementary process components that served as inputs to our testing approach. For example, the External Services Procurement scenario involves seven elementary processes: (1) create purchase requisition for service; (2) determine possible service providers; (3) create purchase order for service; (4) monitor purchase order; (5) enter services actually performed; (6) accept services performed; and (7) verify invoices for services. Once these process components were identified, we applied the approach as described in [2]. This process was repeated with four more SAP scenarios. The results of the testing method's application to all five scenarios are described below.

Test case generation. The test selection domain V is the set of chosen SAP scenarios. The scenarios are mapped to sequences of elementary processes. The 25 elementary processes extracted from the SAP scenarios are listed in Table 1. The final set of generated test cases is STC (STC = {t1, t2, t3, t4, t5}) (see Table 2 for their descriptions).

Table 1: Elementary processes for the five SAP scenarios

Event: Name
e1: Create purchase requisition for service
e2: Determine possible service providers
e3: Create purchase order
e4: Monitor purchase order
e5: Enter services actually performed,
e6: Accept services performed
e7: Verify invoices for services
e8: Create request for materials master record change
e9: Display a materials master record
e10: Verify change request
e11: Notify requisitioner about failure
e12: Change a materials master record
e13: Send a master record update to requisitioner
e14: Create purchase requisition
e15: Create purchase order with a reference to a requisition
e16: Create purchase order without a reference to a requisition
e17: Create a requisition for a quotation
e18: Select a vendor
e19: Follow-up on purchase order
e20: Process goods receipt
e21: Verify invoice
e22: Request service master record change
e23: Display a service master record
e24:Verify service request
e25: Change a service master record

Table 2: Test case descriptions

Test case #	Test-case description	Event-driven process model from which a test case is obtained	Length of test case
t1	e1.e2.e3.e4.e5.e6.e7	External service procurement	7
t2	e8.e9.e10.e11.e12.e13	Materials master maintenance	6
t3	e14.e15.e16.e17.e18.e19.e20.e21	Procurement of stock materials	8
t4	e16.e18.e19.e20.e21	Procurement of consumable materials	5
t5	e22.e23.e24.e25	Service master maintenance	4

Test case partitioning algorithm. The metric-based test case partitioning algorithm is now applied to the set STC of generated test cases, which is {t1, t2, t3, t4, t5}. The test-distance calculation is illustrated in the first step of the partitioning algorithm. It is assumed that ε_{max} =1. The authors chose to start with the longest test case (length=8), which is test case t3; therefore, STC will become the set of {t1, t2, t4, t5} and the first element in the first equivalence class is t3 (TS1= {t3}). The distance is calculated between t3 and the rest of the test cases in STC, as shown in Table 3, so that the test cases with the minimal distance will be included in the same equivalence class TS1.

Table 3: Distance calculated between t3 and the remaining test cases

Ti	**similarity (*t3* , *ti*)**	**dissimilarity (*t3, ti*)**	**e^{-1}**	***td* (*t3, ti*)**
t1	$2^0 = 1$	8	$e^{(-8/7)} = 0.3189$	2.5512
t2	$2^0 = 1$	8	$e^{(-8/6)} = 0.2636$	2.1088
t4	$2^{-4} = 0.0625$	3	$e^{(-8/5)} = 0.2019$	0.0378
t5	$2^0 = 1$	8	$e^{(-8/4)} = 0.1353$	1.0824

As a result, t4 is minimally distant from t3, and both are in the same equivalence class, where TS1 = {t3, t4}, STC = {t1, t2, t5} and ε is set to 0.0378.

Application of the partitioning algorithm steps is continued until all the equivalence classes have been created: TS1 = {t3, t4}, TS2 = {t1}, TS3 = {t2} and TS4 = {t5}.

Prioritization of test cases. The test cases inside each equivalence class are prioritized according to their functional complexity. Greater and more diverse functionality of the system would lead to a larger portion of the system being involved in that usage, and, therefore, it will have more priority than other test cases since it could cover more failures. The functional complexity for each test case is shown in Table 4.

Table 4: The functional complexity values for the five test cases

TS1	
t3	$-\log_2 1/8$
t4	$-\log_2 1/5$
TS2	
t1	$-\log_2 1/7$
TS3	
t2	$-\log_2 1/6$
TS4	
t5	$-\log_2 1/4$

It is to be noted that, in TS1, the FC for t3 is greater than the FC for t4, and therefore t3 will be given a higher priority than t4. The new TS1 = {t3, t4}, is an ordered set. The sets TS2, TS3 and TS4 contain only one test case, and therefore the priority of that test case is considered to be the highest.

Test-selection algorithm. The last step in the authors' testing approach in [2] is based on the budget, since the testing process is so expensive. It is assumed that the total cost of the testing process is directly proportional to the number of test cases and does not exceed C_{max}. The average cost of a test case in the SAP subproject settings has been calculated based on the effort data presented in Table 5.

Table 5: Effort Data

Test case #	**Total person/hours in testing**	**EPC model from which a test case is obtained**	**Total cost**
1	40	External Service Procurement	$1730
2	25	Materials Master Maintenance	$ 504
3	37	Procurement of Stock Materials	$1483
4	26	Procurement of Consumable Materials	$ 992
5	14	Service Master Maintenance	$ 555

In Table 5, the total number of person/hours in the second column means the total number of hours spent by people acting in three roles: testers, business users and external consultants. The reported time includes: (i) the time they spent writing and correcting the test case; (ii) the time taken to run and review the results with a

business user; and (iii) the time taken to write a test acceptance report with the failed and successful test cases. For example, for test case 1, the total of 40 hours means that the tester logged 20 hours, the business user spent 10 hours and the consultant spent 10 hours working on this test case. Each of these three roles had an individual hourly rate: the tester's rate was $18, the business user's rate was $27 and the external consultant's rate was $110. To calculate the accrued test case cost generated by the tester, the business user and the consultant, we multiplied the number of hours each of them spent on the test case by each individual's hourly rate. We then calculated the total cost generated by each individual's role and obtained the dollar values in the right-hand columns.

We also found that the actual average test case cost C for the given five SAP scenarios was approximately $1,000.

Assume now that the budget permits three test cases only. The test-selection algorithm allows the testing process to be optimized through selection of the highest priority test cases from each equivalence class, TSi. Ultimately, the optimal set will contain the following test cases: {t3, t1, t2}. At that time, the total testing cost will not exceed the given budget C_{max} (in this case, $3,000, or three test cases).

Assessment of RQ1. Our early results gave the initial indication that partitioning of test cases and assigning priorities to each test case based on the Abu Talib et al. [2] algorithms fits with the SAP testing process, which provides an affirmative answer to research question *RQ1*. Our study suggests that this testing strategy can be adapted to package implementations with no substantial costs incurred by the project. We also identified a few prerequisites that significantly impacted the straightforward application of our testing approach:

First, we did not allow testing to span over several SAP system instances, each of which implements the same SAP-supported business process.

Second, the business process documentation was up to date and valid, so that testers were sure that they were testing the most recently acquired functionality.

Third, the transactions were automatically mapped to the business process scenarios using a tool. Thus, any subjective influence on the taking of our inventory of SAP functionality was eliminated.

However, to assess the extent to which our testing strategy improves the testing process within given budget constraints, we needed to evaluate the testing quality achieved with the original SAP testing process and the testing quality achieved after introducing the new test selection method into the SAP testing process. We then needed to compare the assessments of testing quality to be able to draw some early conclusion on the improved testing process.

4.3 Evaluation of testing quality

To address *RQ2*, we looked into the quality of the testing process. We defined its quality in terms of effectiveness and efficiency. A testing process is termed to be *effective* if the testing method being used uncovers as many defects as possible. The effectiveness of testing processes is measured in terms of coverage. Thus, in the context of SAP projects, coverage is defined as the percentage of elementary transactions (or series of screens) covered by the sets of test cases. Next, a testing process is termed *efficient* if the testing method finds the largest possible number of defects using the fewest possible tests. The efficiency of testing processes is measured in terms of the average number of transactions per test case.

To assess the effectiveness aspect of the testing process complemented with our method, we compared both sets of test cases (derived by the COSMIC-FFP method and by the SAP testing method) in terms of their percentages of the elementary processes covered. For instance, the effectiveness of the optimal set of test cases chosen by the test selection algorithm for the restricted budget is 70%, which in this case was achieved within the 60% of the cost of SAP-exhaustive testing. This clearly indicates an increase of more than 16% in the effectiveness of the optimized set of test cases.

To assess the efficiency aspect of the testing process, we compared the average numbers of elementary processes per test case for both the COSMIC-FFP and SAP testing methods. The efficiency of the test cases generated by the original SAP testing process (see Table 2) is 6; the efficiency of the optimal test selection applied on the SAP test cases (t1, t2 and t3) is 7, which also indicates an increase of more than 16%. This proves the increase in the efficiency of the optimized testing process.

The above results answer the research question *RQ2*, by demonstrating that the RE testing method for optimizing the generated set of test cases for a given testing strategy and budget constraints newly introduced into the SAP increases the quality of the testing process.

5 Evaluation of validity concerns

Following Yin [13], we considered two types of validity concerns regarding this case study. The threat to external validity means that the case study project may not be representative for all ERP projects. We decided on five subproject settings, which: (i) had up-to-date business requirement models in the form of scenarios; and (ii) are typical for the telecommunications companies in North America. We judge these settings to be typical because they seem common to all SAP-adopting organizations that are members of the American SAP Telecommunications Users Group (ASUG).

It is our experience that the method works best when the company has modeled its ERP-supported processes and mapped them to the set of configurable transactions in the package.

The threat to the internal validity of our study was seen as the risk that there might factors of which we are unaware and over which have no control, even though the testing strategy delivered meaningful results. To ensure internal validity, each process scenario was first taken and studied from the public SAP HELP library available on the Internet. Then, we reexamined the implementation of each process in the company-specific setting. We made sure that: (a) each business requirements document was created using the same RE process, standards and tools; and that (b) transactions were mapped to scenarios using the LiveModel tool. When adapting the Abu Talib et al. approach [2] to the SAP settings, the mappings between the UML concepts and the EPC modeling notation were borrowed from published literature sources [9].

6 Related work

Although ERP testing is recognized in the ERP literature as a critical issue and as a significant cost component of any ERP project, to the best of the authors' knowledge, there seem to be no published papers on this topic in the software engineering literature. We have searched in three commonly used databases of scientific literature sources, namely IEEE Xplore, the ACM Digital Library and Wiley InterScience, found that there is no publication on ERP testing that is accessible through these databases. We also found that K. Bassin et al. [3] are the only authors who have so far considered measurement to evaluate vendor-specific software based on test case execution results. This suggests that, by and large, the topic remains under-researched by the software engineering community.

A few papers have been published on the topic of testing non-ERP component-based software. Bassin et al. [3] discuss experiences of the way in which measurement has provided decision support in quantified terms with regard to assessing progress, analyzing test effectiveness and product stability, and calculating the degree of risk associated with each of these topics. Next, Yoon & Choi [14] propose a testing technique for component-based software built by component users who obtain components from component providers without having access to the source code of these components.

Our research work differs from related work in that we target the optimization of the testing process in the ERP environment prior to the actual implementation of the system. Our approach would allow achievement of better SAP RE testing results within a given budget. The approach is easy to implement as an enhancement to the current ERP RE testing process.

7 Conclusions

This paper explored the adaptation and the application of the metric-based scenario-driven black-box testing method by Abu Talib et al. [2] to a specific class of projects, namely ERP. We complemented the original requirements-based ERP approach to test-case derivation by finding partitions in the input and output data sets and by suggesting that testers carry out ERP transactions with values from these partitions. The reported study revealed that this method has the potential to complement traditional ERP testing approaches, such as the ones built in and assumed in the ERP packages. Our case study demonstrated an efficiency increase of approximately 16%. Moreover, it also revealed two issues:

- the business process documentation should be up to date and valid, so that testers are sure that they are testing the most recently acquired functionality;
- the transactions should be mapped to the scenario processes from the business requirements.

We recommend, however, a replicated follow-up case study be carried out to conduct a deeper investigation of the validity threats to our testing method. More case studies will also help promote the use of our method.

In addition to experimenting with the testing method presented in this paper, we plan, in the immediate future, to focus our research efforts on adapting and applying the COSMIC FFP functional sizing technique to a variety of ERP project contexts characterizing new ERP implementations, upgrades and cross-organizational alignments. This is motivated by one essential component of the testing strategy by Abu Talib et al. [2], namely the use of entropy, which relies on the application of the COSMIC method for FP counting. To this end, our current case-study settings involved solely the application of the IFPUG FPA standard. Our efforts in experimenting with the COSMIC method in ERP settings is considered to be part of a bigger research initiative, the COSMOS project [6], which is aimed at developing ERP functional size measurement and cost estimation models.

Acknowledgment

We would like to thank Tina James, Mac Kutty, Catherine Lee and Marry-Anne van Alpen for sharing their thoughts on aspects of this topic.

References

1. Abran, A., Desharnais, JM, Oligny, S., St-Pierre, D., Symons, C., Measurement Manual COSMIC Full Function Points 2.2 - The COSMIC Implementation Guide for ISO/IEC 19761, École de technologie supérieure, Université du Québec, Montréal, Canada, 2003, www.gelog.etsmtl.ca/cosmic-ffp.

2. Abu Talib, M., Ormandjieva, O., Abran, A., Khelifi A., Buglione, L. Scenario-based Black-Box Testing in COSMIC-FFP: a Case Study, ASQ Software Quality Professional Journal 8 (3), June 2006, pp. 23-33
3. Bassin, K., S. Biyan, P., Santhanam, Metrics to Evaluate Vendor-developed software based on test case execution results, IBM Systems Journal 41 (1), 2002, pp. 13-30.
4. Daneva M., Practical Reuse Measurement in ERP Requirements Engineering, Proc. of the 12th Intl Conf. on Advanced Information Systems Engineering (CaiSE'00), Stockholm, Sweden, June 5-9, 2000, LNCS, Springer Verlag, pp. 309-324.
5. Daneva, M., ERP Requirements Engineering Practice: Lessons Learnt, IEEE Software, 21(2), pp. 26-33.
6. Daneva, M., Status Report on Cross-organizational Functional Size Measurement and Cost Estimation, International Workshop on Software Measurement - IWSM 2006, Postdam, Germany, Shaker-Verlag, Nov. 2-3 2006.
7. Fiorenzi, B., J. Winkler, HR Testing Tips from SBC Communications, America's SAP User Group (ASUG) Conference, Miami Beach, May 20-23, 2001.
8. Larman, G., Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and the Unified Process (3d Edition). Prentice Hall, 2006.
9. Loos, P.; Allweyer, Th.: Object Orientation in Business Process Modeling through Applying Event Driven Process Chains (EPC) in UML, in: Kobryn, C.; Atkinson, C.; Milosevic, Z. (eds.), Enterprise Distributed Object Computing (2nd Int. Workshop EDOC'98, IEEE Comp Society Press, Piscataway 1998, pp 102-112.
10. Oberneidermaier, G., Geiss, M., Testing SAP R/3 Systems, Addison-Wesley, Paperback, Published October 2000
11. Radhakrishnan, R., Accelerated Testing for SAP Upgrades and Implementations, Mercury Inc., A tutorial at SAP TechED'06, Las Vegas, September 11-16, 2006.
12. Schwartz, M., Mastery in ERP Testing, Software Magazine, June '98, 18 (8).
13. Yin, R.: Case Study research: Design and Methods. Sage Publications, 2003.
14. Yoon H., Choi B., Effective test case selection for component customization and its application to Enterprise JavaBeans, Journal of Software Testing, Verification and Reliability, 2004, 14, pp. 45-70.
15. Zrimsek, B., SAP Implementation Project Staffing: Examining the Data, Gartner Group, March 7, 2005.