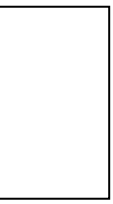
Analyzing the Applicability of a Combinatorial Testing Tool in an Industrial Environment

Nelly Condori-Fernández
Tanja Vos
Peter M. Kruse
Etienne Brosse
Alessandra Bagnato

Technical Report UU-CS-2014-008
May 2014

Department of Information and Computing Sciences
Utrecht University, Utrecht, The Netherlands
www.cs.uu.nl

ISSN: 0924-3275



Department of Information and Computing Sciences
Utrecht University
P.O. Box 80.089
3508 TB Utrecht
The Netherlands

Analyzing the Applicability of a Combinatorial Testing Tool in an Industrial Environment

Nelly Condori-Fernández, Tanja Vos
ProS Research Center,
Universitat Politècnica de València
Camino vera S/N, Valencia
Spain
{nelly, tvos}@pros.upv.es

Peter M. Kruse
Berner & Mattner Systemtechnik GmbH
Gutenbergstr. 15
Berlin, Germany
peter.kruse@berner-mattner.com

Etienne Brosse, Alessandra Bagnato
SOFTEAM R&D Department
8 Parc Ariane Guyancourt, France
{alessandra.bagnato, etienne.brosse}@softeam.fr

ABSTRACT
— Numerous combinatorial testing tools are available for generating test cases. However, many of them are never used in practice. One of the reasons is the lack of empirical studies that involve human subjects applying testing techniques. This paper aims to investigate the applicability of a combinatorial testing tool in the company SOFTEAM. A case study is designed and conducted within the development team responsible for a new product. The participants consist of 3 practitioners from the company. The applicability of the tool has been examined in terms of efficiency, effectiveness and learning effort.

Categories and Subject Descriptors
D.2.5 [**Testing and Debugging**]: Testing tools. H.3.4 [**Systems and Software**]: Performance evaluation (efficiency and effectiveness)

General Terms
Measurement, Performance, Human Factors, Verification.

Keywords
combinatorial testing, classification tree method, industrial case study, effectiveness, efficiency, learning effort.

1. INTRODUCTION
Although laboratory experiments have shown that Combinatorial Testing (CT) can be a very efficient and effective strategy for testing software systems [1], industrial take-up is still low [2]. To improve this take-up, we need to execute more industrial case studies that evaluate combinatorial techniques and tools within real industrial environments, with real people and real systems [3],[4]. We need them as a vehicle for technology transfer, as well as to obtain general guidelines on the applicability of different testing techniques in different settings, and understand the current needs of industry to plan future research directions.

Many papers report on the experience of applying CT tools to various types of applications [4]. However, although these works concentrate on testing industrial systems (e.g. [5]), researchers mostly carry out the studies and focus more on effectiveness of combinatorial testing. The empirical research, we present here, has been done with real subjects (testers) in a real environment, where aspects like cost-effectiveness, capacity and organizational culture are very important for assessing the applicability of a CT tool in an industrial setting.

More specifically, we aim to analyze the combinatorial testing tool CTE XL Professional [6] to assess its applicability to a selected System under Test (SUT) and its contribution to current practice within the French company SOFTEAM (www.softeam.fr). We focus on measuring the efficiency, effectiveness, and learning effort needed to use CTE XL Professional.

The paper is structured as follows: Section II provides an overview of the combinatorial testing tool used in this case study. Section III describes planning and design of our case study research method. Section IV discusses the results. Section V summarizes the threats and limitations of our case study, and Section VI presents conclusions and further work.

2. COMBINATORIAL TESTING WITH THE CTE XL PROFESSIONAL

The Classification Tree Editor XL Professional (CTE) implements the Classification Tree Method [7] by offering a graphical editor [6] and different strategies for automated test case generation.

Applying the classification tree method involves two steps—(1) designing the classification tree and (2) generating test cases.

(1) First, all aspects of interests and their disjoint values are identified. Aspects of interests, also known as parameters, are called classifications; their corresponding parameter values are called classes. Any system under test can be described by a set of classifications, holding input and output parameters. Each classification can have any number of disjoint classes, describing the occurrence of the parameter. All classifications together form the classification tree. Besides the modeling of a classification tree, the CTE allows the tester to define state diagrams where classes are interpreted as states and the state diagram models the possible transitions between them.

(2) Having composed the classification tree and possibly state models, the CTE allows for two ways of generating test cases.

The first way is based on combinatorial testing techniques: test cases are defined by combining classes of different classifications. For each classification, a significant representative (class) is selected. The most common coverage criteria are 2-way or 3-way testing that is fulfilled if all possible pairs/triplets of values are covered by at least one test case in the result test set.

The second approach for automated test generation is based on generating test sequences. This is possible if the combinatorial tree is augmented with state diagrams defining the possible transitions between the different states identified in the tree. Coverage criteria that the tester can choose from for test case generation are: State or Transition Coverage (in the resulting test sequence, each state or Transition is used at least once), State or Transition Pair Coverage (using each possible pair of states or Transitions at least once).

3. CASE STUDY DESIGN

This case study has been designed according to [8] and [12].

3.1 Objective

The main goal of this study is to analyze the CTE to assess its applicability to a selected SUT and how it compares to current practice within SOFTEAM. We aim to answer the following general question:

RQ1. How much effort would be required to learn the CTE for the current testing practitioners at SOFTEAM?

RQ2. How does the CTE contribute to the effectiveness of testing using it in a real testing environment of SOFTEAM and compared to the current testing practices used at SOFTEAM?

RQ3. How does the CTE contribute to the efficiency of testing using it in a real testing environment of SOFTEAM and compared to the current testing practices used at SOFTEAM?

3.2 Empirical context

Our research has been conducted as an industrial case study at SOFTEAM, which was planned for a period of six months (January 2013 to June 2013).

SOFTEAM is a private software vendor and engineering company with about 700 employees located in Paris, France. This case study has been executed within the development team responsible for Modelio Saas, a rather new SOFTEAM product. The team is composed of 1 project manager, 2 software developers and 3 software analysts.

The subjects with whom this study was conducted consisted of 3 members of this team. Subject S1, is an analyst with 5 years of experience. Subject S2, is a software developer with 10 years of experience. Subject S3 is the project manager with 8 years of experience. Both S1 and S2 have less than one year of experience in software testing. Both had previously modeled test cases using the OMG UML Testing Profile (UTP).

The SUT selected by SOFTEAM to serve as a pilot project for this study is the Modelio SaaS system, a prototype system developed at SOFTEAM. Modelio SaaS is a web application written in PHP that allows for the easy configuration of distributed environments. It runs in virtualized environments on different cloud platforms presenting a high number of configurations and hence presents various challenges to testing [9]. We focus on the Web administration console, which allows administrators to manage projects created with the Modelio modeling tool [10], and to specify allowed users for working on projects. The source code is composed of 50 PHP files with a total of 2141 lines of executable code.

The existing test suite that was used for comparison with current practice (TSSOFT) is the current set of 47 manually designed system tests cases currently used for testing new releases of the Modelio SaaS system

3.3 Treatments

The combinatorial testing tool CTE [6] will be compared to the current test design practices implanted at SOFTEAM.

Currently at SOFTEAM, Modelio SaaS test cases are designed manually. The process is based on a series of specified use-cases to support exploratory testing. As indicated before, the objective of test design is to maximize use-case coverage. Each test case describes a sequence of user interactions with the graphical user interface. An example of a test case is in Figure 1.

Test cases are managed with TestLink and are grouped according to the part of the system that they test. Their execution is also done manually by a designated analyst or developer. Line coverage of the executed test suites is measured by a SOFTEAM script that uses Xdebug for gathering the coverage data.

If a failure occurs, the test engineer reports it in the Mantis bug tracking system and assigns it to the developer in charge of the part affected by the failure. Test engineer provides as much information as possible for example relevant Apache or Axis log files. Then, Mantis mails the developer in charge of examining/fixing the reported failure.

Test Case SaaS-7: Create a customer account from a server administrator account		
Auteur:	cba	
#:	Actions de pas:	Résultats attendus:
1	Sign in as a server administrator	
2	Go to "Compte clients" -> "Créer un compte client"	The form to fill customer's details is displayed.
3	Fill: 1. the 'nom' field with a name 2. the "date de soucription" field with a date with format 'YYYY-MM-DD" 3. the "date de validité" field with a date with format 'YYYY-MM-DD" 4. the 'login' field with a login 5. the 'mot de passe' field with a password 6. the 'e-mail' field with an email address	
4	Click on 'Créer compte'	The 'Gestion des comptes clients" page must be displayed with a table containing the customer's details.
Type d'exécution:	Manuel	
Mots-clés:	Aucun	

Figure 1. An example test case for Modelio SaaS

3.4 Procedure

The case study was planned in two phases:

Training phase. A training program was designed in order to develop an individual level of knowledge on combinatorial testing and skills to use the combinatorial testing tool. In this training program two staff members from SOFTEAM (subjects S1 and S2) were involved. By answering some specific questions, their competence level on testing was determined. This information

was helpful to structure an initial training that started with an introductory course on combinatorial testing, which took 4 hours.

Then, the CTE tool was installed and set-up using manuals and online assistance provided by the trainer from B&M. After that, the trainees carried out hands-on learning sessions using CTE including classification tree creation, automatic generation and prioritization of abstract test cases. These hands-on learning sessions took approximately 1 month (from 4 January to 6 February 2013). Working diaries were maintained by the two practitioners and an exam was also conducted to evaluate the competence level on combinatorial testing and CTE.

Testing phase. Next, as is shown in Figure 2, both practitioners (subjects) started designing classification trees for the Modelio SaaS case without any further support from the trainer, and generating abstract test cases. Both consolidation of classification trees and generation of abstract test cases, were performed in ten iterations including manually inspection of resulting test cases by the practitioners.

Once the resulting abstract test cases were finalized, concretization of the test suite TS_{CTE}, execution and evaluation were carried out by only one of them (S2).

After the testing phase informal interviews were conducted with all three subjects.

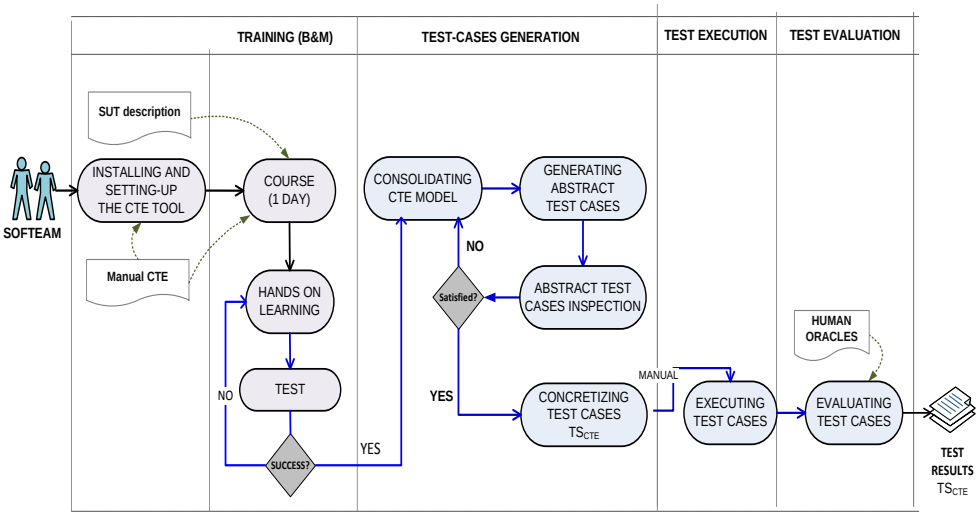


Figure 2. Activities conducted by SOFTEAM practitioners along the case study.

3.5 Data collected

The selected measures to answer our research questions are described in this section.

Learning effort is measured by the time needed for achieving each CTE learning objective. This total time is calculated by selecting relevant learning activities, which are self-reported in working diaries, and adding their respective times expressed in minutes.

The CTE learning objectives covered the following issues: classification trees elements, test elements, abstract test generation, and dependency rules. In order to evaluate the learned competency of the SOFTEAM trainees, a final exam is also formulated. The exam consisted of 29 questions organized in four parts: classification trees, abstract test generation, dependency rules, and test elements.

Effectiveness was measured in terms of fault detection capability and coverage of the test suites compared (i.e. TS_{SOFT} versus TS_{CTE}):

- Number of failures observed

- Number of faults found

- Type and cause of found faults

- Line coverage (using SOFTEAMs script and XDebug)

Efficiency is measured in time testers spend on different activities. For TS_{CTE}, we have measured the following (in hours per tester):

- Time to set up the testing infrastructure (install, configure, develop test drivers, etc.).

- Time to create the Classification Tree

- Time to generate the abstract test suite

- Time to concretize abstract test cases

Moreover, for TS_{SOFT} as well as TS_{CTE} we have measured:

- Time to execute the test cases

- Time to detect faults related to found failures

- Time to identify the fault type and cause for each observed failure (i.e. time to isolate).

4. RESULTS ANALYSIS

A total of 13 consolidated classification trees were constructed to generate TS_{CTE}: five modeled classifications of input parameters and hence pairwise coverage was used as the criterion to generate abstract test cases; the remaining eight trees modeled classifications of user actions together with state models

identifying the starting state and the possible transitions between them, the state coverage criterion was used to generate test sequences for them. The table below presents descriptive measures specific for the classification trees used to generate TS_{CTE}.

Table 1. Descriptive measures of the classification trees used to generate TS_{CTE}

	Item	Value
clasifications of input parameters	Number of classifications	35
	Number of classes	70
	Number of test cases in TS_{CTE}	17
	2-way coverage	57%
	3-way coverage	29%
clasifications of states	Number of classifications	20
	Number of classes	73
	Number of test cases in TS_{CTE}	8
	Number of test steps in TS_{CTE}	67
	State coverage	100%

Next we discuss the results regarding our research questions.

4.1 RQ1: Effort required to learn the CTE

In order to answer our first question about the effort required to learn a combinatorial testing technique supported by the CTE tool, we first analyzed the working diaries maintained during the hands-on learning activities. Through these working diaries, the practitioners reported all activities carried out over that period without a pre-established schedule. Most of the activities were performed individually, but some of them were also performed in pairs (e.g. consolidation of CTE models).

Table 2 shows the time needed for learning activities: the first practitioner (S1) consumed more time than the second practitioner (i.e. more time in Skype meetings with the trainer for solving doubts). Both practitioners spend most time in studying and analyzing the CTE-Material (e.g. CTE manual, course slides), mainly due to the fact that they found CTE manuals very hard to understand. For this reason, they complemented their CTE learning with other activities like internal discussions (120 minutes) and several Skype sessions with the trainer. During this learning process, they created several versions of Classification Trees (4 versions made by S1 and 3 by S2).

Table 2. Self-reported activities during the hands-on learning process

	Time reported in minutes		
Activities	**S1**	**S2**	**In pairs**
CTE-Material analysis	370	360	
Creation CTE tree	180	160	
CTE-models reproduction	55	35	120
Internal discussion meeting			120
Skype meeting with trainer	120	80	90
Total time	725	635	330

Asked about the three most difficult issues considered while learning the CTE, both trainees agreed with "how to create a good tree" as the most difficult. This, however, is not really related to the CTE, but with the implemented modeling method [7]. Other difficulties mentioned were related to defining rules for test case generation. Besides having to choose between conventional combinatorial or sequence-based test generations, there are also several options to choose from for each one of them which were not always clear.

Reviewing the exam results both trainees successfully demonstrated to achieve the first two learning objective related to the classification tree and test elements. However, they demonstrated more difficulties for the other two objectives related to dependency rules and abstract test generation. Figure 3 shows the results obtained of this exam.

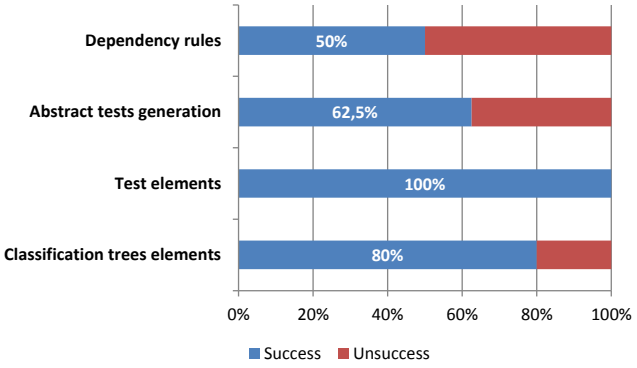


Figure 3. Distribution of success percentage by objective learning

22 out of 29 questions were successfully answered for both trainees. The questions QA1, Q4, Q6 of Part 1 (classification trees), Q1, Q3, and Q4 of Part 3 (abstract test generation), and Q10 of the part 4 (dependency rules) were unsuccessful. A last feedback by the trainer for these questions was given.

4.2 RQ2: Contribution of the CTE to the effectiveness of SOFTEAM testing practices

The size of the final test suites that were compared can be found in Table 3. Although the number of test cases of TS_{CTE} represents almost half of the TS_{SOFT} test cases, the number of HTTP requests is about 20% higher for TS_{CTE} than TS_{SOFT}. This is due to the structure of TS_{CTE} test cases, which are composed of more steps and more events.

Regarding the effectiveness measure in terms of fault detection, the SOFTEAM tester observed 7 failures and found 3 faults using TS_{CTE} and nothing with TS_{SOFT}, (evidently because the test suite has already been used for regression testing purposes on the considered version of Modelio SaaS).

Table 3. Effectiveness Measures of both test suites

Descriptive Metrics	Value TS_{SOFT}	Value TS_{CTE}
Number of test cases	47	25 (17+8)
Number of HTTP requests	123	146
Effectiveness Metrics		
Number of failures observed	0	7
Number of faults found	0	3
Line coverage	85,75%	86,64%

From these 3 identified faults (Table 4), one was rated to be of high severity because it was related to the checking of the login name that failed in two different occasions: (1) permitting the creation of two different client accounts with the same login name; (2) permitting entering a user account with an invalid login name.

Table 4. List of faults detected using TS_{CTE}

ID	Fault type	Fault description	# Failures	Severity
F1	Incorrect data	Inputs of type "Date" are not well validated	4	Minor
F2	Incorrect data	Inputs of type "Email" are not well validated	1	Minor
F3	Incorrect data	The login name is not checked during an account creation	2	Major

Even though TS_{CTE} found faults that TS_{SOFT} did not find, line coverage for TS_{CTE} was only slightly higher than TS_{SOFT}. This made it clear to the SOFTEAM testers that this basic coverage measure did not give them a reliable estimate of the quality of their testing activities and they should consider more sophisticated coverage criteria.

Regarding the combinatorial coverage for TS_{CTE} a percentage of 57% of 2-way (or pairwise) coverage was obtained. Considering 3-way (or three-wise), we still have 29%. This was considered very motivating by the SOFTEAM testers in the sense that they wanted to continue with the CTE approach to see if higher combinatorial coverage could be obtained for possibly finding more faults.

4.3 RQ3: Contribution of the CTE to the efficiency of SOFTEAM testing practices

The time for the activities related to use the CTE can be found in Figure 4.

Time needed for ***setting-up*** the testing tool included 10 minutes (0,17 hour) for CTE installation, and 50 minutes (0,83 hour) to upgrade the tool with a new license. Although the total time for both activities was higher than expected, it was acceptable for SOFTEAM since it only has to be done once.

The time reflected for ***creating classification trees*** (29,33 hours) is the time accumulated for each activity carried out in 10 iterations. This iterative strategy, was supported by the inspection of different versions of abstract test cases generated.

The time needed for generating the last abstract test suite with the CTE was around only 1 minute of processor time and is done automatically by the tool.

Fetching the 25 automatically generated test cases from the CTE, understanding them and manually make them concrete was around 16 hours which comes down to approximately half an hour per test case. Subsequently, an additional 8 hours was spent to input the test cases in TestLink.

Except for creating the CTE tree, all the activities and the time needed for them were acceptable for SOFTEAM, since also it was estimated that similar time was spend on making TS_{SOFT}. Creating the CTE tree, however, was considered rather high. Subject S3 (project manager) indicated that, although for the Modelio SaaS project the CTE would be adopted, more case studies with the same subjects (that are now familiar with the CTE and Classification Trees technique) are needed to be able to conclude more about the applicability for the whole company.

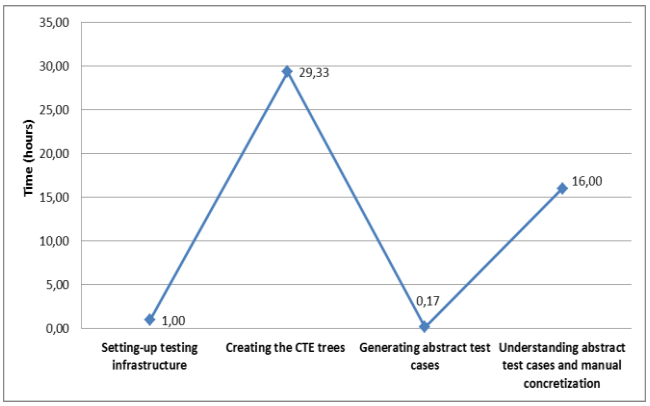


Figure 4. Time needed for setting up, designing and generating a test suite with CTE

Executing the TSCTE, took only 10 minutes more than executing TSSOFT. This was to be expected from the amount of HTTP requests reported in Table 5.

Table 5. Efficiency Measures of both test suites

Descriptive Metrics	Value TS_{SOFT}	Value TS_{CTE}
Time needed to execute the test cases	60 min	70 min
Time needed to identify the fault type and cause for each observed failure.	--	15 min

5. THE THREATS VALIDITY

This section discusses some of these threats addressed in [12].

Construct validity. With respect to the efficiency and learning effort, we could not fully mitigate the threat caused by self-reported working times (e.g. by means of working diaries). Accuracy of these measures could have been affected by other (e.g. social psychological) factors. However, using other complementary measures for learning effort (i.e. test-exam and post questionnaires) helps to triangulate the observations. In order to reduce possible misinterpretations of formulated questions and answers gathered, data analyzed and interpreted by the authors was also validated by the respondents (SOFTEAM practitioners). Moreover, an online post-questionnaire was designed to evaluate the effectiveness of our training program, and the perceived learnability of the CTE (more details in [11]).

Severity of the faults was determined based on its criticality for the system. As criticality levels used in SOFTEAM include some subjective values (i.e. minor and major), the construct validity is affected.

Internal validity. The quality of the classification trees could have been affected by the level of modelling experience . Although a training program was duly implemented this threat could be only reduced.

Existing documentation (e.g. requirements) was used without any improvement by the practitioners for building classification trees. This was because the company mainly was interested in comparing the quality of test cases that were obtained from this case study, with its own test cases obtained within its testing process. The textual description of concrete test cases for both suites could be understood differently, and it could have affected in the faults detection; more even when each test suite was generated by different testers.

External validity is concerned with to what extent it is possible to generalize the findings, and to what extent the findings are of interest to other people outside the investigated case. Generalization is not possible from a single case study. The obtained results about the applicability of CTE need to be evaluated with more SUTs. However, these results could be interesting for other companies like SOFTEAM, whose staff is still very motivated to enhance its actual testing process. Regarding the SUT, it was carefully selected by the testers with the approbation of management staff of SOFTEAM, and the rest of the research team. So, the selected SUT is not only relevant from a technical, but also from an organizational perspective, which facilitated to perform all case study activities.

6. CONCLUSIONS AND FUTURE WORK

6.1 Summary

The main outcomes of the presented study are: (1) with the test suite designed with the CTE, the testers were able to find faults that the traditional test suites did not find, one of them a severe fault; (2) the company realized that the current coverage metrics used for evaluating the quality of test suites needs to be changed to a more sophisticated one; (3) SOFTEAM's motivation to do more case studies with the CTE is high, mainly to see if the improved skills for modeling classification tree can make test case generation more efficient for the whole company.

6.2 Lessons Learned

In general, we can say that SOFTEAM's acceptance of the CTE has improved thanks to the training activities conducted during the case study, which hence served as a successful technology transfer exercise. The hands-on learning activities were carried out at SOFTEAM premises as a further positive factor. The team at SOFTEAM felt comfortable using the tool thanks to the approach followed.

For the SOFTEAM testers, the possibility of rapidly generating abstract test cases automatically with the CTE was an approach allowing them to inspect the effect of changes immediately. The hands-on learning activity was perceived a good learning method to reach "good enough" classification trees. Consequently, this resulted in a fine-tuned final test suite.

We might consider more trainers' feedback on modeled trees or generated abstract test cases after the inspection phase performed within the company to better highlight any other potential errors. The case study protocol does not only need to be updated continuously, it is also important to verify that changes incorporated were appropriately understood by the involved participants.

6.3 Future work

We plan to conduct more case studies that evaluate combinatorial testing tools within real contexts. This will enable us to understand the difficulties that industry has with taking up combinatorial testing tools, the real needs from industry. Moreover, when sufficient empirical evidence is available, the results might be aggregated in order to build empirical knowledge and obtain some applicability guidelines on when to use which technique in what situation.

Finally, since learning is a fundamental part of this type of case studies, we need to also work on a general platform or infrastructure facilitating the monitoring of learning activities carried out in real environments. These are currently missing.

7. ACKNOWLEDGMENTS

This work is partly supported by EU grant ICT-257574 (FITTEST).

8. REFERENCES

[1] Burr K., Young W.. Combinatorial test techniques: Table-based automation, test generation and code coverage. Proc. of the Intl. Conf. on Software Testing Analysis & Review. Citeseer, 1998.

[2] Kuhn R., Kacker R., Lei Y., Hunter J.. Combinatorial Software Testing, IEEE Computer, vol. 42, no. 8., pp 94-96. August 2009

[3] Briand L., Labiche Y.. Empirical Studies of Software Testing Techniques: Challenges, Practical Strategies, and Future Research. In WERST Proceedings/ACM SIGSOFT. 2004, Vol 29, No 5, pp 1-3.

[4] Nie C., Leung H.. 2011. A survey of combinatorial testing. ACM Computing Surveys (CSUR), v.43 n.2, pages 1-29, January 2011

[5] Shikh Gholamhossein L., Bourazjany M. N., Yu Lei, Kacker R.N., Kuhn D.R.. Applying Combinatorial Testing to the Siemens Suite. 2nd Intl Workshop on Combinatorial Testing, Luxembourg, IEEE, March. 2013.

[6] Lehmann E., Wegener J. Test case design by means of the CTE XL. Proceedings of the 8th European International Conference on Software Testing, Analysis & Review (EuroSTAR 2000), Kopenhagen, Denmark. Citeseer, 2000.

[7] Grochtmann M., Grimm K.. Classification trees for partition testing. Softw. Test., Verif. Reliab., 3(2):63-82, 1993.

[8] Vos T., Marín B., Escalona M. J., Marchetto A.: A Methodological Framework for Evaluating Software Testing Techniques and Tools; 12th International Conference on Quality Software, Shaanxi, China, August 27-29, 2012. pp 230-239

[9] Bagnato A., Sadovykh A., Brosse E., Vos T.; (2013). The OMG UML Testing Profile in Use-An Industrial Case Study for the Future Internet Testing, CSMR 2013 17th European Conference on Software Maintenance and Reingineering, March 5-8 2013. pp.457,460.

[10] Modelio.org. Available: http://www,modelio.org/.

[11] Kruse P., Condori-Fernandez N., Vos T., Bagnato A., Brosse E. Combinatorial Testing Tool Learnability in an Industrial Environment. 2013 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement, ESEM '13. Baltimore, USA. October 10-11 2013.(Accepted for publication).

[12] Runeson P., Höst M. Guidelines for conducting and reporting case study research in software engineering. Empirical Software Engineering Journal. April 2009, Volume 14, Issue 2, pp 131-164.