Evaluating Applicability of Combinatorial Testing in an Industrial Environment: A Case Study

Elisa Puoskari
Sulake
Porkkalankatu 1
Helsinki, Finland
elisa.puoskari@sulake.com

Tanja E. J. Vos, Nelly Condori-Fernandez
Universidad Politécnica de Valencia
Camino de vera s/n
Valencia, Spain
{nelly, tvos}@pros.upv.es

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

ABSTRACT
Case studies for evaluating tools in software engineering are powerful. Although they cannot achieve the scientific rigour of formal experiments, the results can provide sufficient information to help companies judge if a specific technology being evaluated will benefit their organization. This paper reports on a case study done for evaluating a combinatorial testing tool in a realistic industrial environment with real objects and subjects. The case study has been executed at Sulake, a company that develops social entertainment games and whose main product is Habbo Hotel, a social network community in the shape of an online Hotel that is visited by millions of teenagers every week all around the world. This paper describes the experimental design of the case study together with the results and decisions that Sulake has taken about the further study, adoption and implantation of these type of tools.

Categories and Subject Descriptors
H.4 [Information Systems Applications]: Miscellaneous;
D.2.8 [Software Engineering]

General Terms
Quality; Empirical study

Keywords
Software Testing; Combinatorial Testing; Test Automation;
Industrial Case Study

1. INTRODUCTION
Suppose a company has read or heard about a new technique or tool and is considering to use it on their projects. In order to find out whether is works for this specific company, a "which is better" type of case study [9] should be performed. These case studies are powerful since, although they cannot achieve the scientific rigour of formal experiments, the results of a case study can provide sufficient information to help other companies judge if the specific technology being evaluated will benefit their own organization [9, 10, 15]. In order to assess tools, evaluative case study research must involve realistic systems and realistic subjects, and not toy-programs and students as is the case in most current work [4, 8, 7]. In this paper we present such a case study for evaluating the CTE [13], a Classification Tree Editor, within a realistic environment of the company Sulake1, the maker of a very popular virtual community, Habbo hotel2. Several studies can be found that evaluate combinatorial testing techniques in a controlled experimental environment (e.g. [12, 17, 11, 3]). To our knowledge no published work exists describing a case study done in a realistic environment with real objects and subjects to evaluate a combinatorial testing tool. Consequently, the contribution of this paper is twofold. On the one hand, it describes the design and results of the performed case study. One the other hand, the study in this paper serves as an example that other companies can follow when encountering the need to evaluate a combinatorial testing tool.

The remainder of the paper is organized as follows: Section 2 presents the case study design framework, and Section 4 presents the results and Section 5 concludes.

2. CASE STUDY DESIGN
2.1 Context
The case study is executed at Sulake1, a company that develops social games and communities whose main product is Habbo Hotel2. Habbo is the world’s largest virtual community for teenagers. Localized Habbo communities are visited by millions of teenagers every week all around the world. Habbo Hotel can be accessed via the local Habbo sites, and through Facebook, where all 11 of Habbo Hotel’s language versions are also available. Through Facebook Connect, Habbo users around the world can easily find their Facebook friends in the virtual world and share their in-world experiences. Some quick Habbo facts (from 2011): 11 language versions; Customers in over 150 countries; Registered users: 218.000.000; Unique visitors: more than 11.000.000 per month; 90% of the users between the age of 13-18.

1http://www.sulake.com/
2http://www.habbo.com
Habbo is built on highly scalable client-server technology. The ActionScript 3 client communicates with a Java server cluster to support tens of thousands of concurrent, interactive users. The server cluster is backed by a MySQL database and intensive caching solutions to guarantee performance under heavy load. The game client is implemented with AS3 and takes care of room rendering, user to user interactions, navigation, purchasing UIs etc. Client performance is challenging as it has to handle up to 50 moving, dancing, talking characters with disco balls, fireworks and user defined game logic running real time in AS3. The game server is a standalone JavaSE with game logic fully implemented in server side. The server handles 10K+ messages / second and Habbo runs as highly distributed system with dedicated role-based clusters, the largest instances having over 100 JVMs. Habbo website and tools are built on Java/Struts2 with AJAX, CSS etc and run on Tomcats.

2.2 Objective - What to Achieve?

Sulake wants to evaluate the combinatorial testing tool CTE [13] to see if it can help improve (or "is better than") their current combinatorial testing practices and would be a technology worthwhile adopting and how. Using the Goal / Question / Measure (GQM) template [1, 14] to describe the goal of the case study we will get the following:

<table>
<thead>
<tr>
<th>Analyze</th>
<th>the CTE tool</th>
</tr>
</thead>
<tbody>
<tr>
<td>For the purpose of</td>
<td>evaluating applicability</td>
</tr>
<tr>
<td>With respect to</td>
<td>the effectiveness and efficiency of Sulake's current combinatorial testing practice</td>
</tr>
<tr>
<td>From the viewpoint</td>
<td>of the testing practitioner</td>
</tr>
<tr>
<td>In the context of</td>
<td>Sulake developing and testing departments</td>
</tr>
</tbody>
</table>

Consequently, following [16] and focussing on the three measures of applicability defined: efficiency, effectiveness, and subjective satisfaction. The research questions for each case study correspond to:

RQ1 Compared to the current test suites used for testing, can the test cases generated by the CTE contribute to the effectiveness of combinatorial testing?

RQ2 How much effort would be required to introduce the CTE into the testing processes currently implanted?

RQ3 How much effort would be required to add the generated test cases into the exiting testing infrastructure?

RQ4 How satisfied are Sulake testing practitioners during the learning, installing, configuring and usage of CTE when it is used in their real testing environments?

2.3 Cases or Treatments - What is Studied?

2.3.1 Current Testing Practice at Sulake

Combinatorial testing for a system such a Habbo is a challenging task, since there exists a wide variety of operating systems and browsers (and their different versions) used by players. Currently, at Sulake, testing new features is planned using high level feature charters to support exploratory testing and automated regression tests are designed for most critical use cases identified during exploratory testing. Teams make sure that the developed features have automated regression tests. Besides feature coverage, test engineers (and their teams) are provided user information that contains for example % of users using each browser, operating system and flash player version. This information is used to take combinatorial aspects into account and design the tests in such a way that user variables that cover most users' setups are tested. For example, when a test is being designed that needs more than one browser (e.g. a friend request from user A to B), this user information is used to make sure that two different user set-up (one for A and one for B) are configured in such a way that most popular user configurations are tested.

The test suites evolve as product is developed and new test cases are added constantly. When tests are added, a browser is selected based on the knowledge of the most used browsers at the time of adding the test.

2.3.2 The Classification Tree Editor

The CTE implements and supports the test suite design technique called the Classification Tree Method (CTM) [5, 13]. Basically the method consist of classifying the combinatorial aspects that are relevant for the test and then decomposing it into classes. The classes found can again be classified in different ways. This creates a tree made of classes and classifications - a Classification Tree (CT). By means of the CT, the CTE then generates test cases by combining the individual classes of the various classifications. In this way we obtain a test suite that uses every class pair from disjunctive classifications at least once in a test case (i.e. Pairwise combination). Since, large trees can generate many test cases, the CTE offers the possibility to prioritize classes in a tree and hence the generated test cases will have an associates prioritization.

There are two editions of the tool, CTE XL Professional (commercial software) and CTE XL (freeware). For this study the commercial variant has been used. Both editions of the CTE come with an included help file. It explains all parts of the program and features tutorial and first steps. From the webpage some product specific details as well as, release notes, FAQ, and Tips and Tricks can be found.

2.4 Subjects - Who Has Applied the Tools?

The subject in this case study was a 37-year old senior tester from Sulake with 6 years of software development experience, 8 years of experience with testing of which 1 year of Habbo and the approach described in Section 2.3.1. The tester had no previous knowledge of the CTE and holds a Master in Computer Science.

2.5 Objects - What Are the Pilot Projects?

The objects used in the studies were 2 nightly builds of Habbo Hotel2. As indicated before, Habbo has been in production at Sulake since 2000 and is as such a very typical project of the company.

3www.cte-xl-professional.com
For comparison with the test suite generated by the CTE (from now on called TS\textsubscript{cte}), the already existing test suite that Sulake has used for testing these builds (from now on denoted by TS\textsubscript{sulake}) will be used. This existing test suite has been implemented incrementally over the past years and continues to evolve as the product develops. Consequently, it is not possible to compare the time necessary for creating the existing test suites with the time needed to create TS\textsubscript{cte}.

Sulake indicated that no faults could be injected into the system. So, effectiveness of both test suites will be compared on real faults present in the identified builds.

### 2.6 Variables - Which Data to Collect?

The independent variables of the study setting are: The CTE combinatorial testing tool; the complexity of the Industrial systems; the level of experience of testers of Sulake that will do the testing. The dependent variables are related those for measuring the applicability of the CTE in terms of effectiveness, efficiency and subjective user satisfaction. Next we present their respective defined metrics:

1. Comparing effectiveness between TS\textsubscript{sulake} and TS\textsubscript{cte}:
   (a) Number of test cases designed or generated.
   (b) Number of invalid test cases generated.
   (c) Number of duplicated test cases generated.
   (d) Number of failures\textsuperscript{5} observed.
   (e) Number of faults found.
   (f) Type and cause of the faults that were found.
   (g) Feature coverage reached.

2. Comparing efficiency between TS\textsubscript{sulake} and TS\textsubscript{cte}:
   (a) Time needed to execute the test suite.
   (b) Time needed to identify fault types and causes for each observed failure.

3. Measuring the effort needed to use the CTE:
   (a) Time needed to learn the testing method CTE.
   (b) Time needed to design or generate the test suite with the CTE.
   (c) Time needed to set up the testing infrastructure specific to CTE (install, configure, develop test drivers, etc.).
   (d) Time needed to automate the test suite generated by the CTE within the Sulake environment.

4. Measuring subjective satisfaction
   (a) Select 5 reaction cards (see [2] for the words used on the reaction cards).
   (b) Informal interview about satisfaction and perceived usefulness.

\textsuperscript{5}Note, we will use the terminology that is consistent with IEEE Standard Glossary of Software Engineering Terminology (Std 729-1983), the IEEE Standard Classification for Software Anomalies (Std 1044-2009) and the IFIP (International Federation for Information Processing): a Fault is the incorrect code that results from a human mistake. A Failure is the incorrect behaviour of the software that the user can observe.

### 2.7 Protocol - How to Execute the Study?

The following steps were taken when executing the study:

1. Fill profiling demographic questionnaire.
2. CTE short course over Skype by a CTE instructor.
3. Set up, install and configure CTE with help of manuals.
4. Do the following testing tasks and collect the data.
   - Make a CTE tree as instructed so that it contains browser usage % as weight for browsers and business criticality % as weight for test cases.
   - Generate the test cases with the CTE using prioritized all pairs as coverage type.
   - Create a new test suite TS\textsubscript{cte} by selecting the 42 highest ranking tests for comparison against existing test suite TS\textsubscript{sulake} that has 42 tests.
   - Automate the test cases from the new test suite TS\textsubscript{cte}. Measurements for this task include time to automate TS\textsubscript{cte} and possible problems in interpreting the resulting test suite from CTE tool.
   - Execute the new test suite 2 times for different builds and collect data.
5. Informal satisfaction and perceived usefulness interviews, that consist of answering the following two questions while recording the interview:
   (a) Would you recommend this tool to other colleagues? If not why? If yes what arguments would you use?
   (b) Do you think you can persuade your management to invest in a tool like this? If not why? If yes what arguments would you use?
6. Reaction cards session.

### 2.8 Threats to Validity

#### Internal validity. This aspect of validity is of concern when causal relations are examined. In our case study the following threat was identified: Personal motivation and beliefs may shape the response of the involved person. Although the selected subject is the most representative for the case study, since this is the person that is normally engaged in testing Habbo, he may be under pressure because his decision about the tool would affect many colleagues and or management.

#### External validity. This aspect of 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. As mentioned in Section II.C, Combinatorial testing for a system such a Habbo is a challenging task, since there exists a wide variety of operating systems and browsers (and their different versions) used by players. We think that the conclusions we draw will hold for other companies that develop similar systems like Habbo.
Table 1: Descriptive measures for test suites.

<table>
<thead>
<tr>
<th>Variable</th>
<th>TS\textsubscript{Sulake}</th>
<th>TS\textsubscript{cte}</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of test cases</td>
<td>42</td>
<td>68</td>
</tr>
<tr>
<td>Number of invalid test cases</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Number of duplicated test cases</td>
<td>0</td>
<td>26</td>
</tr>
</tbody>
</table>

Table 2: Effectiveness measures for both test suites.

<table>
<thead>
<tr>
<th>Variable</th>
<th>TS\textsubscript{Sulake}</th>
<th>TS\textsubscript{cte}</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of failures observed</td>
<td>0</td>
<td>12</td>
</tr>
<tr>
<td>Number of faults found</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>Feature coverage reached</td>
<td>100%</td>
<td>40%</td>
</tr>
</tbody>
</table>

Construct validity. This aspect of validity reflect to what extent the operational measures that are studied really represent what the researcher have in mind and what is investigated according to the research questions. There was no way to measure and compare time spent creating existing test suite as it has been developed and modified over the years together with the product. For this reason, the evaluation was based on measures related only to failures and faults detected. Moreover, only two builds were used for measuring the variables.

3. RESULTS AND DISCUSSION

Since we only have one value from each testing method, no analysis techniques are available and the table from above just presents the measured data on which the answers to the research question from Section 2.2 will be discussed.

3.1 Research Question RQ1

Compared to the current test suites used for testing, can the test cases generated by the CTE contribute to the effectiveness of combinatorial testing?

As can be read from Tables 1 and 2, the CTE found 2 faults that the existing test suite did not reveal considering the same amount of test cases. One fault was a critical one (browser hang) and the other was a minor fault (broken UI element) Considering the fact that this is a realistic system, Sulake concluded that, although the CTE seems a bit more effective, there current combinatorial testing practices are in good shape considering the high allpairs coverage percentage of the CTE test suite. Moreover, feature coverage (variable 1g, see Table 2) for the test suite generated by the CTE was significantly lower, implying that the CTE can only be seen as a complementary tool to the existing practices (which is not strange since CTE is a combinatorial testing tool and not a functional one).

3.2 Research Question RQ2

How much effort would be required to introduce the CTE into the testing processes currently implanted?

The effort and time for learning the CTE is 116 minutes, this was considered medium sized and can be justified for Sulake as they only need to invest this effort once.

3.3 Research Question RQ3

How much effort would be required to add the generated test cases into the existing testing infrastructure?

The time and effort for the activities related to the use of the CTE can be found in Figure 1. Time needed for setting up the testing infrastructure and installing the CTE was also medium sized, but again was good for Sulake since this is also something you do only once.

Designing (i.e. creating a CTE model) and generating abstract test cases for creating TS\textsubscript{cte} suffered from duplicates (see Table 1) that needed to be removed manually. However, the total effort can be accepted within Sulake since faults were found that were not found with the existing test suite and effort for duplicate removal wasn’t significant.

Executing TS\textsubscript{cte} within the test automation environment (variable 2a) takes nearly one hour more than executing TS\textsubscript{Sulake} (see Figure 2). This fact is not surprising since pairwise coverage of TS\textsubscript{cte} will be higher than TS\textsubscript{Sulake} and hence there is more switching between browsers and users logging in and out which evidently takes time. However, close to one hour of extra testing time to test software to be deployed to live in continuous integration is not acceptable for Sulake in light of the other effectiveness and efficiency variable values. TS\textsubscript{Sulake} takes less than 1 hour while TS\textsubscript{cte} takes close to 2. Although 1 hour of automated testing that found a critical fault might seem worthwhile, 1 extra hour of testing for each build means possibility of less builds during working day. That can be seen as impediment for efficient continuous deployment process. Consequently, Sulake concluded that the test cases generated by the CTE could not be used for the automated testing activities during the day, but instead could be better added to the nightly builds. This way, advantage is being taken of the fault-finding capabilities without disturbing the high number of builds a day.

3.3 Research Question RQ3

Implementing TS\textsubscript{cte} took approximately 6 hours (i.e. one working day) (see Figure 1). For automated regression testing (during nightly builds) and the fact that higher pairwise coverage might be obtained (and hence a higher chance for finding more combinatorial faults) this effort might be justified as standard procedure in testing at Sulake.
3.4 Research Question \textbf{RQ4}

\textit{How satisfied are Sulake testing practitioners during the learning, installing, configuring and usage of CTE when it is used in their real testing environments?}

Subjective opinions about the CTE that were mentioned during the informal interview\(^7\) (see appendix A for the analysis following the work from \cite{2}) are:

- The generation of duplicated test cases that generated unnecessary manual effort was not appreciated at all. However, the subject identified that this duplicated test cases problem might not be inherent to the tool but could possibly be solved with better CTE models and for this more experience with the tool and its underlying models is needed.

- The CTE trees that can grow rather big and hence suffer from bad readability which could become a problem.

- The technical support and manuals available during the case study were very good and quick.

- The appearance of the tool could be improved, it looks a bit outdated.

The subject concludes that the tool meets most expectations and that yes he would recommend it to someone with a good technical background and to management. However, considering the more negative issues, additional case studies comparing similar tools would be appreciated.

The rationale for the selected reaction cards were:

- \textbf{Comprehensive} - Seems to have everything I can think of needing, the logical conditions for my own conditions/constraint rules were good;

- \textbf{Sterile} - looks sterile when opening. It gets the job done but could look nicer;

- \textbf{Unattractive} - is slowish and makes me want to find faster ways of doing this which makes it a bit unusable;

- \textbf{Old} - built on top of eclipse makes it look old, like something I used while studying during 90s;

- \textbf{Dated}: Old fashioned by looks and usability.

4. CONCLUSIONS

We have presented a "which is better" \cite{9} case study for evaluating the combinatorial testing tool CTE \cite{13} with a real user and real tasks within a realistic environment of the company Sulake\(^1\), the maker Habbo Hotel\(^2\). Although a single-subject case study will never provide general conclusions with statistical significance, the obtained results can be generalized to other testers of Habbo in the testing environment of Sulake \cite{18, 6}. Moreover, the study was very useful for technology transfer purposes: some remarks during the informal interview indicate that the CTE would not have been evaluated in so much depth if it would not have been backed up by our case study design. Finally, having only one real subject available, this study took several weeks to complete and hence we overcame the problem of getting too much information too late.

Considering that Sulake's objective was to see "which is better" (current practice versus CTE tool) the following were the results of the case study:

- the CTE tool can improve the effectiveness of current test suites designed at Sulake, but only if used complementary to the current practice.

- the test suites generated by a combinatorial tool can only be executed during the nightly builds, because otherwise they will be an impediment for efficient continuous deployment process.

- before making a final decision upon which combinatorial tool to purchase, some more case studies with other tools will be done to see if besides the effective and efficiency considerations, there exists a tool that is more attractive is its usage.

5. ACKNOWLEDGMENTS

We would like to thank Vesa Ekholm for being our "N=1". This work was financed by the FITTEST project (contract number 257574).

6. REFERENCES

\begin{thebibliography}{9}
\end{thebibliography}
Appendix A. Faces Questionnaire

Faces were rated with a scale from 1 to 7 where 1 represented "Not at all like this" and 7 represented "Very much like this".

Duplicate test cases (time interval: 1.00-1.05).

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Readability of CTE trees (interval time: 1.07-1.25).

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Technical support and CTE manuals (interval time: 1.35-1.40).

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Appearance CTE tool (time interval: 1.07-1.25)

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>