Evaluating Combinatorial Testing in an Industrial Environment: a case study

Tanja E.J. Vos
Universidad Politécnica de Valencia
Camino de vera s/n
Valencia, Spain
tvos@pros.upv.es

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

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

***Abstract*—Case studies for evaluating tools in software engineering are powerfull. 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 excuted 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.**

I. 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 [1] 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 [1], [2], [3]. 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], [5], [6]. In this paper we present such a case study for evaluating the CTE [7], a Classification Tree Editor, within a realistic environment of the company Sulake[1], the maker of a very popular virtual community, Habbo hotel[2]. Several studies can be found that evaluate combinatorial testing techniques in a controlled experimental environment (e.g. [8], [9], [10], [11]). 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.

[1]http://www.sulake.com/
[2]http://www.habbo.com

II. Case Study Design

A. Context

The case study is executed at Sulake[1], a company that develops social entertainment games and communities whose main product is Habbo Hotel[2]. 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.

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.

B. Objective - What to achieve?

Sulake wants to evaluate the combinatorial testing tool CTE [7] 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 [12], [13] to describe the goal of the case study we will get the following:

Analyze	the CTE tool
For the purpose of	evaluating effectiveness, efficiency and subjective satisfaction
With respect to	Sulake's current combinatorial testing practice
From the viewpoint	of the testing practicioner
In the context of	Sulake developing and testing departments

Consequently, focussing on the three measures of usability defined by ISO 9241-11: 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?

C. Cases or Treatments - What is studied?

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) The Classification Tree Editor: The CTE implements and supports the test suite design technique called the Classification Tree Method (CTM) [14]. 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.

CTE comes with an included help file. It explains all parts of the program and features tutorial and first steps. From the webpage[3] some product specific details as well as, release notes, FAQ, and Tips and Tricks can be found.

D. Subjects - Who apply the techniques/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 II-C1. The tester had no previous knowledge of the CTE and holds a Master in Computer Science.

E. Objects - What are the pilot projects?

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

For comparison with the test suite generated by the CTE (from now on called TS_{cte}), the already existing test suite that Sulake has used for testing these builds (from now on denoted by TS_{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_{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.

F. 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 towards measuring effectiveness, efficiency and subjective user satisfaction and are listed below.

[3] www.cte-xl-professional.com

1) Measuring effectiveness
   a) Number of test cases designed or generated.
   b) Number of invalid test cases generated.
   c) Number of repeated test cases generated.
   d) Number of failures observed.
   e) Number of faults found.
   f) Type and cause of the faults that were found.
   g) Feauture coverage reached.
   h) Allpairs coverage reached.
2) Measuring efficiency
   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.
   e) Time needed to execute the test suite.
   f) Time needed to identify fault types and causes for each observed failure.
3) Measuring subjective satisfaction
   a) **S**ystem **U**sability **S**core (SUS) questionnaire [15] consisting of 10 questions with 5 Likert-scale and a total score.
   b) 5 reactions (through reaction cards).
   c) Informal interviews about satisfaction and perceived usefulness.

The SUS questionnaire was used because it is known to give reliable results [16]. Since SUS for 1 individual does not give meaningful results, we complement it with reaction cards and informal (taped) interviews [17], [18].

G. 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_{cte} by selecting the 42 highest ranking tests for comparison against existing test suite TS_{sulake} that has 42 tests.
   - Automate the test cases from the new test suite TS_{cte}. Measurements for this task include time to automate TS_{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) Fill the SUS questionnaire.
6) 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?
7) Reaction cards session (see [17] for the words used).

H. Threats to Validity

The following confounding factors have been identified and taken into account when answering the research questions: (1) The evaluation was only done by one person, there are no more resources in the company to allocate to this evauation. However, this is the person that is normally engaged in testing Habbo and applying the approach described in Section II-C1. Moreover, his opinion is valued within the company as to take decisions about whether or not to adopt a tool. If the tool would be adopted this person would be working with it and so he is not under too much pressure that his decision about the tool would affect many collegues and or management. (2) Only two builds were used for measuring the variables. (3) 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.

III. Results

The following variables were measured during the study:

Variable	TS_{sulake}	TS_{cte}
1a	42	68 (only 42 with highest priority were chosen for automation).
1b	0	0
1c	0	26
1d	0	12
1e	0	2
1f	N/A	1 critical fault (browser hang) 1 minor fault (broken UI element)
1g	100%	40%
1h	N/A	80%
2a	N/A	116 min
2b	N/A	95 min (60 for creating the tree, 2 for generating, 33 for removing duplicates)
2c	N/A	74 min
2d	N/A	357 min
2e	114 min	183 min (both builds)
2f	0	116 min
3a	N/A	50
3b	N/A	Comprehensive, Dated, Old, Sterile, Unattractive
3c	N/A	http://youtu.be/dm3wD6n5yDM

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 II-B will be based.

A. 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 shown by measured variables 1e, the CTE found 2 faults that the existing test suite did not reveal. One fault was a critical one. 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) 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).

B. 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 and installing the CTE is medium sized (2a and 2c) and can be justified for Sulake as they only need to invest this effort once. Designing and generating test cases for creating TS_{cte} (2b) suffered from duplicates 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_{cte} within the test automation environment (2e) takes nearly one hour more than executing TS_{sulake}. This fact is not surprising since pairwise coverage of TS_{cte} is higher than TS_{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_{sulake} takes less than 1 hour while TS_{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 possible high number of builds a day.

C. Research question ***RQ3***

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

Implementing TS_{cte} took approximately 6 hours (i.e. one working day) (see variable 2d). 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.

D. Research question ***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?

Qualitative feedback was neutral towards usage of the CTE, which comes across from all variables measuring subjective satisfaction (including SUS although it was only performed by one person). The rationale for the reaction cards were:

- Comprehensive - Seems to have everything I can think of needing, the logical conditions for my own conditions/constraint rules were good;
- Sterile - looks sterile when opening. It gets the job done but could look nicer;
- Unattractive - is slowish and makes me want to find faster ways of doing this which makes it a bit unusable;
- Old - built on top of eclipse makes it look old, like something I used while studying during 90s;
- Dated: Old fashioned by looks and usability.

Subjective satisfaction measurements indicate that tool itself is perceived to meet most expectations for such tool, although additional case studies comparing similiar tools would be appreciated. Most of the constructive feedback is related to appearance of the tool.

IV. Conclusions

We have presented a "which is better" [1] case study for evaluating the CTE [7] within a realistic environment of the company Sulake[1], the maker Habbo Hotel[2]. Although a case study will never provide conclusions with statistical significance, enough information has been compiled to make an informed decision. 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:

- a combinatorial testing 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.

V. Acknowledgments

This work was funded by the EU project FITTEST (ICT257574, 2010-2013).

REFERENCES

[1] B. Kitchenham, L. Pickard, and S. Pfleeger, "Case studies for method and tool evaluation," *Software, IEEE*, vol. 12, no. 4, pp. 52 –62, Jul. 1995.

[2] B. A. Kitchenham, S. L. Pfleeger, L. M. Pickard, P. W. Jones, D. C. Hoaglin, K. E. Emam, and J. Rosenberg, "Preliminary guidelines for empirical research in software engineering," *IEEE Trans. Softw. Eng.*, vol. 28, no. 8, pp. 721–734, 2002.

[3] P. Runeson and M. Höst, "Guidelines for conducting and reporting case study research in software eng," *Empirical Softw. Engg.*, vol. 14, no. 2, pp. 131–164, 2009.

[4] N. Fenton, S. Pfleeger, and R. Glass, "Science and substance: a challenge to software engineers," *Software, IEEE*, vol. 11, no. 4, pp. 86 –95, Jul. 1994.

[5] N. Juristo, A. Moreno, and S. Vegas, "Reviewing 25 years of testing technique experiments," *Empirical Softw. Engg.*, vol. 9, no. 1-2, pp. 7–44, 2004.

[6] S. Hesari, H. Mashayekhi, and R. Ramsin, "Towards a general framework for evaluating software development methodologies," in *Proc of 34th IEEE COMPSAC*, 2010, pp. 208–217.

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

[8] D. R. Kuhn, D. R. Wallace, and A. M. Gallo, "Software fault interactions and implications for software testing," *IEEE Transactions on Software Engineering*, vol. 30, pp. 418–421, 2004.

[9] D. R. Wallace and D. R. Kuhn, "Failure modes in medical device software: an analysis of 15 years of recall data," *International Journal of Reliability Quality and Safety Engineering*, vol. 8, pp. 351–372, 2001.

[10] D. R. Kuhn and M. J. Reilly, "An investigation of the applicability of design of experiments to software testing," in *Proceedings of the 27th Annual NASA Goddard/IEEE*, 2002, pp. 91–95.

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

[12] V. R. Basili, F. Shull, and F. Lanubile, "Building knowledge through families of experiments," *IEEE Trans. Softw. Eng.*, vol. 25, no. 4, pp. 456–473, 1999.

[13] C. Lott and H. Rombach, "Repeatable software engineering experiments for comparing defect-detection techniques," *Empirical Software Engineering*, vol. 1, pp. 241–277, 1996.

[14] M. Grochtmann and K. Grimm, "Classification trees for partition testing," *Softw. Test., Verif. Reliab.*, vol. 3, no. 2, pp. 63–82, 1993.

[15] J. Brooke, "SUS: a 'quick and dirty' usability scale," in *Usability Evaluation in Industry*, 1996.

[16] T. Tullis and J. N. Stetson, "A comparison of questionnaires for assessing website usability." in *Proc of the Usability Professionals Association Conf*, 2004.

[17] J. Benedek and T. Miner, "Measuring desirability: New methods for measuring desirability," in *Proc of the Usability Professionals Association Conf*, 2002.

[18] D. Williams, G. Kelly, and L. Anderson, "Msn 9: new user-centered desirability methods produce compelling visual design," in *Extended abstracts on Human factors in computing systems*, ser. CHI '04, 2004, pp. 959–974.