Evaluating the TESTAR tool in an Industrial Case Study

Sebastian Bauersfeld
Universidad Politecnica de Valencia, Spain
sbauersfeld@pros.upv.es

Tanja E.J. Vos
Universidad Politecnica de Valencia, Spain
tvos@pros.upv.es

Nelly Condori-Fernandez
Vrije Universiteit van Amsterdam, The Netherlands
n.condori-fernandez@vu.nl

Alessandra Bagnato
Softeam, Paris, France
alessandra.bagnato@softeam.fr

Etienne Brosse
Softeam, Paris, France
etienne.brosse@softeam.fr

ABSTRACT

[Context] Automated test case design and execution at the GUI level of applications is not a fact in industrial practice. Tests are still mainly designed and executed manually. In previous work we have described TESTAR, a tool which allows to set-up fully automatic testing at the GUI level of applications to find severe faults such as crashes or non-responsiveness. [Method] This paper aims at the evaluation of TESTAR with an industrial case study. The case study was conducted at SOFTEAM, a French software company, while testing their Modelio SaaS system, a cloud-based system to manage virtual machines that run their popular graphical UML editor Modelio. [Goal] The goal of the study was to evaluate how the tool would perform within the context of SOFTEAM and on their software application. On the other hand, we were interested to see how easy or difficult it is to learn and implant our academic prototype within an industrial setting. [Results] The effectiveness and efficiency of the automated tests generated with TESTAR can definitely compete with that of the manual test suite. [Conclusions] The training materials as well as the user and installation manual of TESTAR need to be improved using the feedback received during the study. Finally, the need to program Java-code to create sophisticated oracles for testing created some initial problems and some resistance. However, it became clear that this could be solved by explaining the need for these oracles and compare them to the alternative of more expensive and complex human oracles. The need to raise consciousness that automated testing means programming solved most of the initial problems.

Categories and Subject Descriptors

D.2.8 [**Software Engineering**]: Quality—*software testing*

Keywords

Software Testing at the User Interface level, TESTAR

1. INTRODUCTION

Automated test case design and execution at the GUI level of applications is not a fact in industrial practice. Tests are still mainly designed and executed manually. In previous work we have presented an approach to automated testing at the GUI level [3] whose objective is to automatically generate and execute test cases based on a structure that is automatically derived from the GUI. Our tool is called TESTAR (Test Automation at the useR interface level) and it was evaluated in experimental conditions using different software applications like MS Word (running it 48 hours we detected 14 crash sequences [1]). Subsequently, TESTAR was also applied it to a mature industrial accounting software application that has been developed at a Spanish company for over 15 years and the results were similar in terms of fault effectiveness detection, the results are published here [2].

With the purpose of getting a better understanding about the applicability of the tool in an industrial environment, in this paper we report a case study, where real industrial subjects (and not the academics) apply the tool to their daily testing tasks. Consequently, besides effectiveness and efficiency of testing, this study also evaluated learnability and satisfaction of the tool in practice.

This case study reported on in this paper has been executed at the company SOFTEAM[2], a French software company. SOFTEAM develops Modelio SaaS[3], a cloud-based system to manage virtual machines that run their popular graphical UML editor Modelio.

The type of case study presented here can be powerful [7] since, although they cannot achieve the scientific rigor of formal experiments, their results can provide sufficient information to help other companies judge if the specific technology being evaluated will benefit their own organization [10, 6] and will boost technology transfer.

This paper is structures as follows. Section 2 described the context of SOFTEAM, the company where we executed the study. Section 3 describes the design of our study. Section 4 lists the collected data and Section 5 presents the analysis of the data related to the proposed research questions.Section 6 discusses the threats to validity, and Section 7, finally, concludes.

[1]Videos of these crashes are available at `http://www.youtube.com/watch?v=PBs9jF_pLCs`
[2]http://www.softeam.com/
[3]http://www.modeliosoft.com

2. THE CONTEXT: SOFTEAM

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 and testing team responsible for Modelio Saas, a rather new SOFTEAM product. Modelio SaaS is a web administration console written in PHP, which allows an administrator to connect to his account for managing modelling projects created with Modelio UML Modeling tool, another product from SOFTEAM.

One of the priorities of SOFTEAM is to maximize users-interaction coverage of their test suites with minimum costs. However, the current testing process has several limitations since test case design and execution is performed manually and resources for manual inspection of test cases are limited.

Learning to use and integrate TESTAR into SOFTEAM's current testing processes, could allow testers to reduce the time spent on manual testing. The downside of this potential optimization is the extra effort and uncertainty that comes with applying a new test approach. To decide if this extra effort is worth spending, a case study has been planned and carried out. The results will support the decision making about whether to adopt the TESTAR tool at SOFTEAM.

3. DESIGN OF THE CASE STUDY

3.1 Objective

The goal of the case study is to measure the learnability, the effectiveness, efficiency and subjective satisfaction when using TESTAR in the context of Modelio SaaS. We will concentrate on the following questions:

RQ1 How learnable is the TESTAR tool when it is used by testing practitioners of SOFTEAM?

RQ2 How does TESTAR contribute to the effectiveness and efficiency of testing when it is used in real industrial environments and compared to the current testing practices at SOFTEAM?

RQ3 How satisfied are SOFTEAM testers during the installation, configuration and application of the tool when applied in a real testing environment?

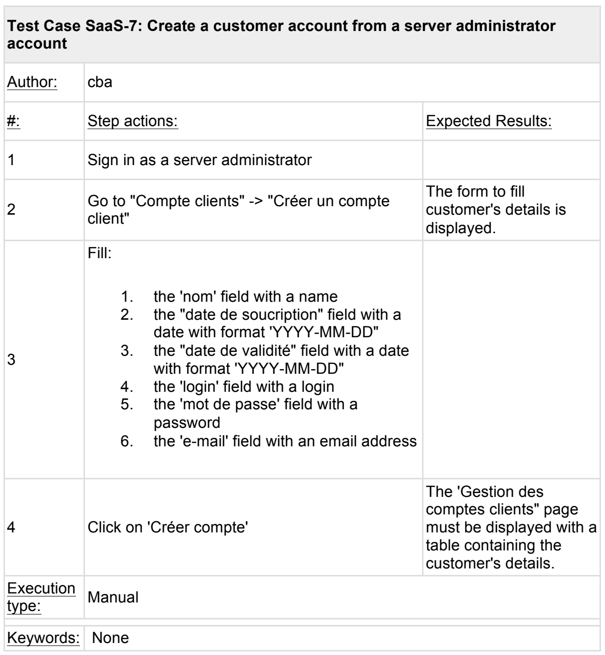
3.2 Objects of the study

3.2.1 The System Under Test (SUT)

The SUT selected for this study is the Modelio SaaS system developed at SOFTEAM. Modelio SaaS is a PHP web application, that allows for easy and transparent configuration of distributed environments. It can run in virtual environments on different cloud platforms, offers a large number of configuration options and hence poses various challenges to testing [1]. In this study we will focus on the web administration console, which allows server administrators to manage projects created with the Modelio modelling tool, and to specify user rights for working on these projects. The source code is composed of 50 PHP files with a total of 2141 lines of executable code.

3.2.2 TS_{Soft} – SOFTEAM's existing manual Test Suite

The existing test suite is a set of 51 manually crafted system test cases that SOFTEAM uses to manually perform regression testing of new releases. Each test case describes a sequence of user interactions with the graphical user interface as well as the expected results. Figure 1 shows an example of such a test case.



Test Case SaaS-7: Create a customer account from a server administrator account		
Author:	cba	
#:	Step actions:	Expected Results:
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.
Execution type:	Manual	
Keywords:	None	

Figure 1: Manual test case, used by Modelio SaaS testers for functional testing.

3.2.3 Injected Faults

In order to be able to study the effectiveness (i.e. fault finding capability) of TESTAR, SOFTEAM proposed to select a list of faults that have occurred in previous version of Modelio SaaS and which are considered important. These faults have been re-injected into the last version of Modelio SaaS that has been used during the study. Since all of these faults occurred during the development of Modelio SaaS, which makes them realistic candidates for a test with the TESTAR tool.

Table 1 shows the list of faults, their descriptions, identifiers and severity.

3.3 Cases or Treatments - What is studied?

3.3.1 Testing with TESTAR

TESTAR its basic test sequence generation algorithm comprises the following steps:

1. Obtain the GUI's state (i.e. the visible widgets and their properties like position, size, focus ...).

2. Derive a set of sensible actions (clicks, text input, mouse gestures, ...).

3. Select and execute an action.

4. Apply an oracle to check whether the state is valid. If it is invalid, stop sequence generation and save the suspicious sequence to a dedicated directory, for replay.

5. If the given amount of sequences has been generated, stop sequence generation, else go to step 1.

ID	Component	FileLocation	Description	Severity
1	Controller	AccountController.php line 102	When clicking on "no" for account deletion confirmation, the system nevertheless deletes the account	M
2	Controller	LoginController.php line 8	No login fields on login page	H
3	Controller	ProjectsController.php line 8	Empty page when accessing the project creation page	H
4	Controller	RamController.php line 31	Description is not added to the database when creating a component	L
5	Controller	RolesController.php line 48	Page not found error after editing a role	M
6	Model	DeploymentInstance.php line 10	"An error occurred" message when trying to view properties of a project	H
7	Model	Module.php line 21	"An error occurred" message when trying to add a module to a project	H
8	Model	Module.php line 34	"An error occurred" message when trying to upload a new module	M
9	Model	Project.php line 36	"An error occurred" message when trying to view managed project (need to be a project manager)	H
10	Model	ProjectModule.php line 29	"An error occurred" message when trying to view properties of a project	H
11	View	ComponentSelection line 19	Empty page when trying to add a component to a project	H
12	View	Modules.php line 18	Allow empty content for module	L
13	View	ModuleSelection.php line 30	Empty page when trying to add a module to a project	H
14	View	RoleSelection.php lines 27 to 30	"An error occurred" when trying to edit the role of a user of a project	H
15	View	Server.php line 42	The type of the server is missing	M
16	View	ServerSelection.php line 13	Empty form when trying to move a server	L
17	View	Users.php line 82	Editing is possible when accessing through view link and vice versa	L

Table 1: Injected Faults

TESTAR uses the operating system's Accessibility API to recognize GUI controls and their properties and enables programmatic interaction with them. It derives sets of possible actions for each state that the GUI is in and automatically selects and executes appropriate ones in order to drive the tests. In completely autonomous and unattended mode, the oracles can detect faulty behaviour when a system crashes or freezes. Besides these free oracles, the tester can easily specify some regular expressions that can detect patterns of suspicious titles in widgets that might pop up during the executed tests sequences. For more sophisticated and powerful oracles, the tester can program the Java protocol that is used to evaluate the outcomes of the tests.

3.3.2 Testing currently at Softeam

Modelio SaaS' testing and development team consists of 1 product director, 2 developers and 3 research engineers who all participate in the testing process. The testing practice at Softeam is to create test cases by relying on specified use cases. Each test case describes a sequence of the user interactions through the GUI as shown in Figure 1.

The test cases are managed with the *TestLink*[4] software and grouped as test suites according to the part of the system that they enable to test. All them are executed manually by a test engineer. If a failure occurs, the test engineer reports it to the *Mantis*[5] bug tracking system and assigns it to the developer in charge of the part affected by the failure. He also provides the Apache log file for the web UI as well as the Axis log file for the web services. Then, Mantis mails the developer in charge of examining/fixing the reported failure.

Softeam's testing process in projects other than Modelio SaaS is similar. A tester has access to the project specifications (most of the time a textual description).

3.4 Subjects - Who applies the techniques?

The subjects are two computer scientists that besides other responsibilities for Modelio SaaS are responsible for testing on the project. Subject one is a senior analyst (5 years), and trainee two is a software developer with 10 years of experience. Both have less than one year of experience in software testing and have previously modelled test cases using the OMG UML Testing Profile (UTP) and the Modelio implementation of the UML Testing Profile. In a previous study [8] they have obtained training in combinatorial testing. In addition, both testers also claim to be proficient in Java, the language used to develop and extend the TESTAR Tool.

3.5 The case study procedure

After TESTAR has been installed and a working testing environment has been set-up, the case study is divided into two phases (see Figure 2).

The Training Phase

During this phase, the subjects start to develop a working test environment for SOFTEAM's case study system. Challenges, difficulties and first impressions are gathered to evaluate how well the subjects understood the concepts of the technique and whether they are prepared to proceed to

[4]`http://sourceforge.net/projects/testlink/`
[5]`http://www.mantisbt.org/`

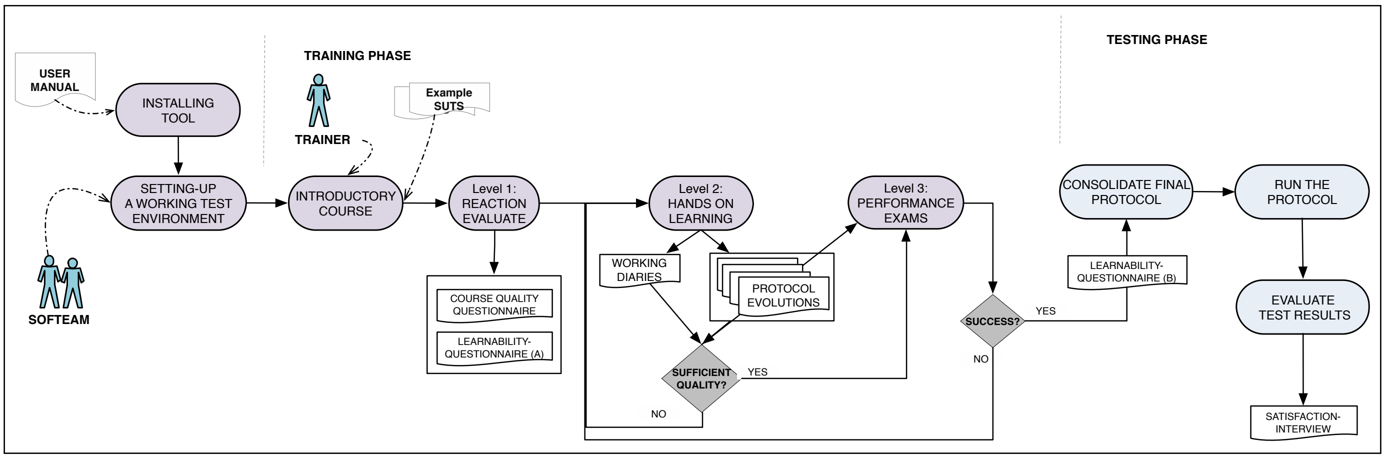

USER MANUAL
INSTALLING TOOL
SETTING-UP A WORKING TEST ENVIRONMENT
SOFTEAM
TRAINING PHASE
TRAINER
Example SUTS
INTRODUCTORY COURSE
Level 1: REACTION EVALUATE
COURSE QUALITY QUESTIONNAIRE
LEARNABILITY-QUESTIONNAIRE (A)
Level 2: HANDS ON LEARNING
WORKING DIARIES
PROTOCOL EVOLUTIONS
SUFFICIENT QUALITY?
YES
NO
Level 3: PERFORMANCE EXAMS
SUCCESS?
YES
NO
TESTING PHASE
CONSOLIDATE FINAL PROTOCOL
LEARNABILITY-QUESTIONNAIRE (B)
RUN THE PROTOCOL
EVALUATE TEST RESULTS
SATISFACTION-INTERVIEW

Figure 2: Case Study Procedure

the next phase. The following activities have been planned:

Presentational learning - The trainer gives an introductory course in which working examples are presented. Example SUTs are unrelated to the case study system so that the subjects get an insight into: How to setup TESTAR for a given SUT?, How to tell the tool which actions to execute?, How to program an effective test oracle for different types of faults?, How to define the stopping criteria?.

Autonomous hands-on learning (i.e. learning by doing) with online help from the trainer through Skype and/or email. The subjects will apply the learned techniques to setup a test environment for the selected SUT and write evolving versions of a protocol for the TESTAR tool. They will work together and produce one version of the TESTAR protocol. Each tester documents their progress in working diaries which contain information about: The activity that has been performed and the minutes spent on each activity; The questions and doubts that the tester had at the time he was doing this activity (so one can see if those were solved in later learning activities); Versions and evolutions of TESTAR protocols that are produced.

During the introductory course, audio-visual presentations (i.e. tool demos, slides) were used. For supporting the hands-on learning activities, the individual problem-solving method was used. The important issues considered were location and materials. The hands-on learning activities were carried out at SOFTEAM premises. Before the actual hands-on part, an introduction in terms of a course was given in-house at SOFTEAM. The training materials (e.g. slides, example files) were prepared by the trainer.

The Testing Phase

The subjects will refine and consolidate the last protocol made during the training phase work. This protocol will be used for testing, i.e. the protocol is run to test the SUT and the results are evaluated.

3.6 Measures

The *independent variables* of the study settings are: the TESTAR GUI Testing Tool; the complexity of the SOFT case study system (Modelio SaaS); level of experience of the SOFT testers who will perform the testing. The *dependent variables* are related to measuring the learnability, effectiveness, efficiency and subjective user satisfaction of the TESTAR tool. Next we present their respective defined metrics.

Measuring Learnability - Following [5], learnability can be understood and evaluated in two different ways: *Initial learning* allows users to reach a reasonable level of usage proficiency within a short time. But it does not account for the learning that occurs after such a level has been reached; *Extended learning*, in contrast to initial learning, considers a larger scope and long term of learning. It applies to the nature of performance change over time. In the presented study, we are interested in assessing extended learnability. For this purpose, the training program was designed in order to develop an individual level of knowledge on GUI testing and skills to use TESTAR.

In order to determine the effectiveness of the training program, feedback from the subjects on the training program as a whole was gathered in different ways. A levels-based strategy, similar to [8], for evaluating the learning processes was applied. Next we explain briefly each level that is used in this study (the numbers correspond to the levels mentioned in Figure 2) and the quantitative and qualitative measurement that were carried out:

1. *Reaction level*: is about how the learners perceive and react to the learning and performance process. This level is often measured with attitude questionnaires that are passed out after most training classes.

   In our study this is operationalized by means of a learnability-questionnaire (A) to capture first responses (impressions) on the learnability of the tool. Moreover, we will have a questionaire that concentrates on the perceived quality of the course. introductory course.

2. *Learning level*: is the extent to which learners improve knowledge, increase skill, and change attitudes as a result of participating in a learning process.

   In our study this is operationalized by means of self-reports of working diaries were collected to measure the learning outcomes; and the same learnability questionnaire (B) to capture more in-depth impressions after having used the tool during a longer time.

3. *Performance level*: involves testing the learner's ca-

pabilities to perform learned skills while on the job. These evaluations can be performed formally (testing) or informally (observation).

In our study this is operationalized by means of 1) using a measure adapted from [5] related to actual on-the-job performance, in this case evolution and sophistication of the developed artifacts (oracle, action definition, stopping criteria) over a certain time interval; and 2) conducting a performance exam.

Measuring Effectiveness was done during the testing phase. For test suites TS$_{Soft}$ and TS$_{Testar}$ we measured:

1. Number of failures observed by both test suites. The failures relate to the ones in Table 1 that were injected into the current version of Modelio SaaS.
2. Achieved code coverage (We measured the line coverage of the PHP code executed by both test suites. We took this as an indicator of how "thorough" the SUT has been executed during the testing process)

Measuring Efficiency was done during the testing phase. For both TS$_{Soft}$ and TS$_{Testar}$ and measured:

1. Time needed to design and develop the test suites. In the case of TESTAR we took the time that was necessary to develop the oracle, action definitions and stopping criteria.
2. Time needed to run TS$_{Soft}$ and TS$_{Testar}$.
3. Reproducibility of the faults detected.

Measuring Subjective Satisfaction is done after the testing phase has been completed and consists of:

1. Reaction cards session: each subject selects 5 cards that contain words with which they identify the tool (for the 118 words used see [4]).
2. Informal interview about satisfaction and perceived usefulness that is setup around the questions: *Would you recommend the tool to your peers or persuade your management to invest? If not why? If yes, what arguments would you use?*
3. Face questionnaires to obtain information about satisfaction through facial expressions. The informal interview from above will be taped and facial expression will be observed following the work in [4]. The purpose of the face questionnaire is to complement the satisfaction interview in order to determine whether their gestures harmonize with their given answers.

4. DATA COLLECTION

Data collection methods[6] included the administration of two questionnaires, test-based examination, working diaries, inspection of different TESTAR protocol artifacts (oracle, action, stopping), as well as video-taped interviews with the subjects.

Regarding to the working diaries, the trainees reported all the activities carried out over the hands-on learning period without a pre-established schedule. Table 2 shows the description data for these activities.

[6]All materials can be found here: `https://staq.dsic.upv.es/papers/softeam-TESTAR/index.html`

	Time reported (min)		
Activities	**S1**	**S2**	**In Pairs**
Oracle design + impl	1200	30	30
Action definition + impl	820	30	20
Stopping Criteria	30	0	10
Evaluating run results	240	20	30
Skype meeting with trainer	60	10	15
Total time	2350	90	105

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

Description	**Test Suite**	
	TS$_{Soft}$	**TS$_{Testar}$**
Faults discovered	14 + 1	10 + 1
Did not find IDs	1, 9, 12	1,4,8,12,14,15,16
Code coverage	86.63%	70.02%
Time spent on development	40h	36h
Run time	manual	automated
	1h 10m	77h 26m
Faults diagnosis and report	2h	3h 30m
Faults reproducible	100%	91.76%
Number of test cases	51	dynamic

Table 3: Comparison between tests

Figure 3 shows the quality of the different TESTARs setups, as rated by the trainer. The trainer rated each artifact of a version separately, i.e. oracle, action set and stopping criterion on a scale from 0 to 5 as if it was a student submitted assignment.

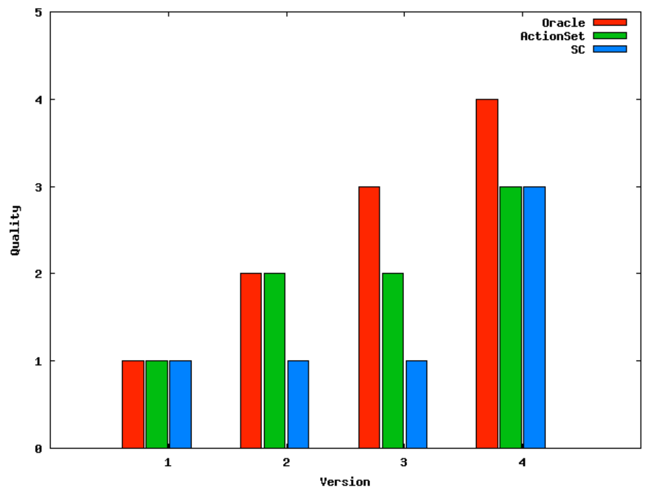


Figure 3: Evolution of artifact quality as rated by the trainer

Table 3 shows the descriptive values of bot test suites considered in this study: the existing manual test suite (TS_{Soft}) and the test suite generated by our tool (TS_{Testar}).

During the study we have used two questionnaires. The first is the questionnaire that evaluates the quality of the training course: its contents, the allocated time, and the provided materials. This questionaire contains one item in 5-points ordinal scale and six items in 5-points likert scale.

The learnability questionnaire is used to measure perceived learnability of the tool. The same questionnaire is applied at point A, after the course but before the hands-on learning, and at point B, after the hands-on learning. The questions have been taken from [9] where the authors are analyzing the learnability of CASE tools. They have been divided into 7 categories to separate different aspects of the tool. It consists of 18 items in 5-points likert scale.

5. ANALYSIS

RQ1: How learnable is the TESTAR tool when it is used by testing practitioners of SOFTEAM?

Empirical data was collected in order to analyze learnability at the three identified different levels.

Reaction (level 1) - Responses from two questionnaires about first impressions of the course (quality and learnability (A)) and another one applied after the test exam (learnability B) were analyzed. With respect to the course (at level 1), both respondents showed to be satisfied with the content of the course, and the time allocated for it. The practical examples during the course were perceived as very useful to understand the GUI testing concepts. Both subject S1 as S2 highlighted that it was very easy to get started and to learn how to first approach the use of the tool through the provided user manual, the testers were able to use the basic functionalities of tool right from the beginning and liked the friendliness and cleanness of the environment.

Learning (Level 2) - If we look at the self-reported activities during the hands-on process in Table 2 we see that subject 1 spend considerable more time than subject 2. This was due to unforeseen workload of S2 that in industrial environments cannot always be planned nor ignored. The role of S2 was reduced to that of revising the outcomes of the tests of S1 and being informed about the tool's features.

From the self-reported activities, and based on the opinion of the trainer, it could be deduced that the testers had a few problems with the definition of the TESTAR's action set. This set defines the TESTAR's behaviour and is crucial to its ability to explore the SUT and trigger crashes. Action definitions comprise the description of trivial behaviour such as clicks and text input, as well as more complicated drag and drop and mouse gestures.

With respect to the perceived learnability of the tool, we found that after one month of using the tool during the hands-on learning (see Table 2 for the time that was spend by each subject), their impressions on the training material had changed slightly. Both respondents found that the tool manuals would have to be extended with further explanations in particular on how to customize the tool by using its API methods, in particular on how to setup powerful oracles that detect errors in the SUT and how to setup powerful action sets that drive the SUT and allow to find problematic input sequences.

Moreover, it turned out that the concept of 'powerful' oracle was not totally understood after the course. First impressions were that the oracles were easy to set up (regular expressions) and quite powerful (since within a short period of time and without hardly any effort some of the injected faults were found). However, these are not what is considered a 'powerful' oracle because of the lack of 'power' to detect more sophisticated faults in the functionality of the applications. During the hands-on training it was realized that setting up more sophisticated oracles was not as easy as considered in the beginning, and programming skills and knowledge of the SUT were needed. The need to do Java programming to set-up tests caused some initial resistance towards the acceptance of the technique. However, by comparing them to the alternative of more expensive and complex human oracles and explaining the need to program these oracles in order to automate an effective testing process, consciousness was raised. Initial resistance was turned into quite some enthusiasm to program the oracles, such that the last versions even contain consistency checks of the database underlying the SUT.

Performance (Level 3) - In order to analyse the actual performance level of the subjects, the evolution of the artefacts generated during training and testing phases were studied. Throughout the course of the case study, the testers developed 4 different versions of the TESTAR's setup, with increasing complexity and power.

The first set-up offered a rather trivial oracle, which scraped the screen for critical strings such as "Error" and "Exception". The testers supplied these strings in the form of regular expressions. Obvious faults such as number 6 (see Table 1 for the list of injected faults) are detectable with this strategy. However, this heavily relies on visible and previously known error messages. More subtle faults, such as number 16 are not detectable this way.

The second oracle version made use of the web server's logging file which allowed to detect additional types of faults (e.g. errors caused by missing resource files, etc.).

Versions 3 and 4 also incorporated a consistency check of the database used by Modelio SaaS. Certain actions such as the creation of new users, access the database and could potentially result in erroneous entries. The more powerful database oracle in version 3, requires appropriate actions, that heavily stress the database. Thus, the simulated users should prefer to create / delete / update many records. Version 4 also defined a better test stopping criteria indicating when tests were considered enough.

Figure 3 shows the quality of the different TESTAR's setups, as rated by the trainer on a scale from 0 to 5. The perceived quality increases with each version and eventually reaches a sufficient level in the last one. Although, the trainer is not entirely satisfied with the quality of the testers' action definitions and stopping criteria, this coincides with the difficulties mentioned by the trainees. Overall, the graphic shows a clear increase in sophistication, indicating the ability of the testers to learn how to operate the tool and create more powerfull oracles.

RQ2: How does TESTAR contribute to the effectiveness and efficiency of testing when it is used in real industrial environments and compared to the current testing practices at SOFTEAM?

To answer the research questions regarding the efficiency and effectiveness of TESTAR, we collected data of the existing manual test suite (TS_{Soft}) and the test suite generated by the TESTAR tool (TS_{Testar}) (see Table 3). To obtain data for TS_{Testar} we used the last of the 4 versions of the setup for TESTAR created during the learning phase. However, the measure 'time spent on development' also includes the time necessary to develop the earlier versions in the development time, since these intermediate steps were necessary to build the final setup. To measure the variable values for TS_{Soft} we employed Softeam's current manual test suite for which the company has information about man hours dedicated to its development.

TS_{Soft} consists of a fixed set of 51 hand-crafted test cases, whereas TS_{Testar} does not comprise specific test cases, but rather generates them as needed. Softeam reported to have spent approximately 40 hours of development time on crafting the manual test cases, which roughly equals the 36 hours that their testers needed to setup TESTAR for the final test (including earlier setup versions).

The testers took about 3 hour to execute all manual test cases, identify the fault and report them. TESTAR simply ran automatically for about 77 hours. Of course they could have decided to perform a shorter run, but since the tool works completely automatic and ran over night, it did not cause any manual labour. The only thing that the testers had to do, in the mornings, consisted of consulting the logs for potential errors, report these. This took about 3,5 hours.

In terms of code coverage, the manual suite outperformed the automatically generated tests. However, the difference of approximately 16% is modest. Manual testing allows the tester to explore forms that might be locked by passwords or execute commands that require specific text input. A way to enable TESTAR to explore the GUI more thoroughly, would be to specify more complex action sets. We consider this as a plausible cause, as the trainer pointed out, that he was not entirely satisfied with the action definitions that the testers designed (see Figure 3).

Considering the amount of seeded faults that have been detected by both suites, the manual tests, unsurprisingly, outperformed those generated by the TESTAR tool. TS_{Soft} detected 14 of the seeded faults and the testers even found a previously unknown error. All of the erratic behaviors were reproducible without any problems. TS_{Testar}, on the other hand, detected 11 faults, including the previously unknown one. However, as expected, the tool had problems detecting certain kinds of faults, since it can be hard to define a strong oracle for those. Examples include errors similar to number 16 (Figure 1). Nevertheless, obvious faulty behaviour, which often occurs after introducing new features or even fixing previous bugs, can be detected fully automatic. However, if we look at the severity of the faults that were not found by TESTAR, we can see that 4 have severity Low, 2 have Medium and only one has High severity. On the other hand, the fault that was found by TESTAR and not by the manual test suite has high severity. So, given the low amount of manual labour involved in finding those, the TESTAR tool can be a useful addition to a manual suite and could significantly reduce manual testing time. One definite advantage, that TESTAR has over the manual suite is, that the setup can be replayed arbitrary amount of times, at virtually no cost, e.g. over night, after each new release. The longer the tool runs, the more likely it is to detect new errors. We think that the development of a powerful oracle setup pays of in the long term, since it can be reused and replayed automatically.

Finally, looking at the reproducibility of the faults, sometimes a test triggers a fault that is hard to reproduce through a subsequent run of the faulty sequence. Sometimes the environment is not in the same state as it was during the time the fault was revealed, or the fault is inherently indeterministic. The timing of the tool used for replay can have a major impact. Of the faults reported by the TESTAR tool, around 8% of the faults found were not reproducible. The others could be traced back to the injected faults.

RQ3: How satisfied are SOFTEAM testers during the installation, configuration and application of the tool when applied in a real testing environment?

A first source that we used to gain insight into the testers' mind were reaction cards as defined in [4]. We gave the testers a list of words and asked them to mark the ones, that they associate the most with the TESTAR tool. The words chosen by the two subjects had a positive connotation (such as "Fun", "Desirable", "Time-Saving", "Attractive", "Motivating", "Innovative", "Satisfying", "Usable", "Useful" and "Valuable") coinciding with their overall positive attitude towards the tool and the case study.

During the informal interview, when asked if they would recommend the tool to their peer colleagues: Subject 1 answers positively and would use the following arguments: the TESTAR tool is quite suitable for many types of applications; it can save time, especially in the context of simple and repetitive tests. This allows testers to concentrate on the difficult tests which are hard to automate. Also subject 2 is positive about the tool and wants to add the argument that it is very satisfying to see how easy it is to quickly set up basic crash tests.

On the negative side, both testers agree on the necessity to improve the tool's documentation: basically improvements related to action definitions and oracle design. Also some installation problems were mentioned.

When asked if they think they can persuade their management to invest in a tool like this, both subjects are a bit less confident. They argue that the benefits of the tool need to be studied during a longer period of time, especially maintenance of the test artefacts would need to be studied in order to make a strong business case and claim Return of Investment to convince the many people in the management layer. However, Subject 2 – being positive by nature – thinks that although in need of strong arguments, convincing management people is not impossible.

Finally, to cross-validate the testers claims, we video taped the testers while responding to the questions, and conducted a face questionnaire as described in [4]. The results of this analysis coincides with the findings from above and is summarized in the Appendix.

6. THREATS TO VALIDITY

Construct validity reflects to what extent our operational measures really represent what is investigated according to the research questions. In our case, although the learnability evaluation was based on a four-level strategy [9] that we have used before, some of the threats could not be fully mitigated, at least, for the two first levels (Reaction and Learning). This is because most of the collected data was based on trainee's responses. However, in order to reduce possible misinterpretations of formulated questions and answers gathered, data analyzed and interpreted by the second author was also validated by the respondents (trainees).

Internal validity is of concern when causal relations are examined. Although learning (level 2) and performance (level 3) criteria are conceptually related [9], this threat was not mitigated because environmental variables of the hands-on learning process could not be monitored. Only working diaries were self-reported by the trainees.

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. Statistical generalization is not possible from a single case study but the obtained results about the learnability of the TESTAR tool need to be evaluated further in different contexts. However, these results could be relevant for other companies like SOFTEAM, whose staff has experience in software testing, but is still very motivated to enhance its actual testing process. Regarding to the system under test (SUT), it was carefully selected by the trainees with the approbation of the rest of the research team (UPVLC) and management staff of SOFTEAM. So, the selected SUT is not only relevant from a technical perspective, but also from an organizational perspective, which facilitated to perform all the case study activities.

Reliability is concerned with to what extent the data and the analysis are dependent on the specific researchers. All the formulated questions were reviewed, in terms of clarity, by other three volunteer colleagues from UPVLC. A detailed protocol was also developed and all data collected was appropriately coded and reviewed by case subjects.

7. CONCLUSIONS

We have presented a case study for evaluating TESTAR [3] with real users and real tasks within a realistic environment of testing Modelio SaaS of the company SOFTEAM.

Although a case study with 2 subjects will never provide general conclusions with statistical significance, the obtained results can be generalized to other testers of Modelio SaaS in the testing environment of SOFTEAM [10, 6]. Moreover, the study was very useful for technology transfer purposes: some remarks during the informal interview indicate that the tool would not have been evaluated in so much depth if it would not have been backed up by our case study design. Also, having only two real subjects available, this study took a month to complete and hence we overcame the problem of getting too much information too late. Finally, we received valuable feedback on how to evolve the tool and its related documentation and course materials.

The following were the results of the case study:

1) The SOFTEAM subjects found it very easy to get started with the tool and to learn how to use the tool's default behaviour (i.e. free oracles and random actions) through the provided user manual, the testers were able to use the basic functionalities of tool right from the beginning and liked the friendliness and cleanness of the environment.

2) Programming more sophisticated oracles customizing the Java protocol raised some problems during the learning process of the SOFTEAM subjects. The problems were mainly related to the understanding of the role of oracles in automated testing. In the end, in pairs and with the guidance of the trainer, the subjects were capable to program the tool in such a way that it detected a fair amount of injected faults. This gives insight into the training material and the user manual that needs to be improved and concentrate more on giving examples and guidance on more sophisticated oracles. Also, we might need to research and develop a wizard that can customize the protocol without Java programming.

3) The effectiveness and efficiency of the automated tests generated with TESTAR can definitely compete with that of the manual tests of SOFTEAM. The subjects felt confident that if they would invest a bit more time in customizing the action selection and the oracles, the TESTAR tool would do as best or even better as their manual test suite w.r.t. coverage and fault finding capability. This could save them the manual execution of the test suite in the future.

4) The SOFTEAM subjects found the investment in learning the TESTAR tool and spending effort in writing Java code for powerful oracles worthwhile since they were sure this would pay off the ore often the tests are run in an automated way. They were satisfied with the experience and were animated to show their peer colleagues. To persuade management and invest some more in the tool (for example by doing follow-up studies to research how good the automated tests can get and how re-usable they are amongst versions of the SUT) was perceived as difficult. Nevertheless, enthusiasm to try was definitely detected.

In summary, despite criticism regarding the documentation and installation process of the tool, the testers' reactions and statements encountered during the interviews and the face questionnaire, indicate that they were satisfied with the testing experience. We came to a similar conclusion regarding the tool's learnability. Although, the trainer reported certain difficulties with the action set definition, the constant progress and increase of artefact quality during the case study, points to an ease of learnability. These items will be improved in future work to enhance the tool.

8. REFERENCES

[1] A. Bagnato, A. Sadovykh, E. Brosse, and T.E.J. Vos. The omg uml testing profile in use–an industrial case study for the future internet testing. In *Software Maintenance and Reengineering (CSMR), 2013 17th European Conference on*, pages 457–460, 2013.

[2] S. Bauersfeld, A. de Rojas, and T. E. J. Vos. Evaluating rogue user testing in industry: an experience report. In *Proceedings of 8th International Conference RCIS*. IEEE, 2014.

[3] S. Bauersfeld and T. E. J. Vos. Guitest: a java library for fully automated gui robustness testing. In *Proc of the 27th IEEE/ACM ASE 2012*, pages 330–333.

[4] J. Benedek and T. Miner. Measuring desirability: New methods for evaluating desirability in a usability lab setting. *Proceedings of Usability Professionals Association, Orlando, USA*, 2002.

[5] T. Grossman, G. Fitzmaurice, and R. Attar. A survey of software learnability: Metrics, methodologies and guidelines. In *SIGCHI Conference on Human Factors in Computing Systems*, pages 649–658. ACM, 2009.

[6] Warren Harrison. Editorial (N=1: an alternative for software engineering research). *Empirical Software Engineering*, 2(1):7–10, 1997.

[7] B. Kitchenham, L. Pickard, and S.L. Pfleeger. Case studies for method and tool evaluation. *Software, IEEE*, 12(4):52 –62, July 1995.

[8] P.M. Kruse, N. Condori-Fernandez, T.E.J. Vos, A. Bagnato, and E. Brosse. Combinatorial testing tool learnability in an industrial environment. In *ESEM 2013*, pages 304–312, Oct 2013.

[9] M. Senapathi. A framework for the evaluation of case tool learnability in educational environments. *Journal of Information Technology Education: Research*, 4(1):61–84, January 2005.

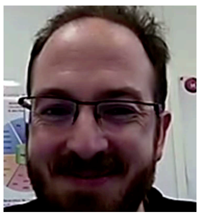
[10] A. Zendler, E. Horn, H. Schwartzel, and E. Plodereder. Demonstrating the usage of single-case designs in

experimental software engineering. *Information and Software Technology*, 43(12):681 – 691, 2001.

APPENDIX

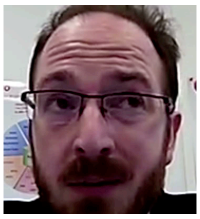
A. FACES QUESTIONNAIRE

Faces were rated with a scale from 1 to7 where 1 represented ”Not at all” and 7 represented ”Very much”.



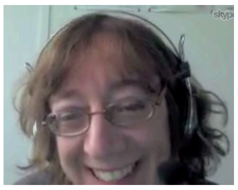
Would you recommend the tool to your colleagues?

1	2	3	4	5	6	7
				X		



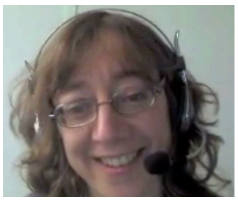
Could you pursuade your management to invest?

1	2	3	4	5	6	7
	X					



Would you recommend the tool to your colleagues?

1	2	3	4	5	6	7
					X	



Could you pursuade your management to invest?

1	2	3	4	5	6	7
		X				

Acknowledgements

This work was financed by the FITTEST (Future Internet Testing) project, ICT-2009.1.2, no 257574.