VU Research Portal

Scriptless Testing at the GUI Level in an Industrial Setting

Chahim, Hatim; Duran, Mehmet; Vos, Tanja E.J.; Aho, Pekka; Condori Fernandez, Nelly

published in

Research Challenges in Information Science
2020

DOI (link to publisher)

[10.1007/978-3-030-50316-1_16](https://doi.org/10.1007/978-3-030-50316-1_16)

document version

Publisher's PDF, also known as Version of record

document license

Article 25fa Dutch Copyright Act

[Link to publication in VU Research Portal](#)

citation for published version (APA)

Chahim, H., Duran, M., Vos, T. E. J., Aho, P., & Condori Fernandez, N. (2020). Scriptless Testing at the GUI Level in an Industrial Setting. In F. Dalpiaz, J. Zdravkovic, & P. Loucopoulos (Eds.), *Research Challenges in Information Science: 14th International Conference, RCIS 2020, Limassol, Cyprus, September 23–25, 2020, Proceedings* (pp. 267–284). (Lecture Notes in Business Information Processing; Vol. 385 LNBIP). Springer. https://doi.org/10.1007/978-3-030-50316-1_16

General rights

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal ?

Take down policy

If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

E-mail address:

vuresearchportal.ub@vu.nl

Scriptless Testing at the GUI Level in an Industrial Setting

Hatim Chahim¹, Mehmet Duran², Tanja E. J. Vos^{3,4}(✉), Pekka Aho³,
and Nelly Condori Fernandez⁵

¹ ProRail, Utrecht, The Netherlands

² Capgemini, Utrecht, The Netherlands

³ Open Universiteit, Heerlen, The Netherlands

⁴ Universidad Politecnica de Valencia, Valencia, Spain
tvos@dsic.upv.es

⁵ University of A Coruña/Vrije Universiteit, Amsterdam, The Netherlands

Abstract. TESTAR is a traversal-based and scriptless tool for test automation at the Graphical User Interface (GUI) level. It is different from existing test approaches because no test cases need to be defined before testing. Instead, the tests are generated during the execution, on-the-fly. This paper presents an empirical case study in a realistic industrial context where we compare TESTAR to a manual test approach of a web-based application in the rail sector. Both qualitative and quantitative research methods are used to investigate learnability, effectiveness, efficiency, and satisfaction. The results show that TESTAR was able to detect more faults and higher functional test coverage than the used manual test approach. As far as efficiency is concerned, the preparation time of both test approaches is identical, but TESTAR can realize test execution without the use of human resources. Finally, TESTAR turns out to be a learnable test approach. As a result of the study described in this paper, TESTAR technology was successfully transferred and the company will use both test approaches in a complementary way in the future.

Keywords: GUI test automation tools · TESTAR · Compare test approaches · Industrial case study · Technology transfer · Railway sector

1 Introduction

Testing software at the Graphical User Interface (GUI) level is an important part of realistic testing [1]. The GUI represents a central juncture to the Software Under Test (SUT) from where all the functionality is accessed. Consequently, testing at the GUI level means operating the application as a whole, i.e., the system's components are tested in conjunction. However, automated testing at the GUI level is difficult because GUIs are designed to be operated by humans. Moreover, they are inherently non-static interfaces, subject to constant change caused by functionality updates, usability enhancements, changing requirements or altered contexts. This makes it very hard to develop and maintain automated test scripts. And today, most companies still resort to time-consuming and expensive manual testing [2, 3].

The state of the practise tools for automated testing at the GUI level used in the industry are **script-based**. Capture and Replay (CR) tools [4–6], for example, enable the tester to record (or capture) the test scripts automating the sequences of actions (like clicks, keystrokes, drag and drop operations) that compose the use case. These scripts are then replayed on the UI to serve as regression tests for new product releases. However, since GUIs change all the time, recorded scripts break and need to be captured again or manually maintained. Consequently, the maintenance cost can get real high, and if the scripts are repaired instead of capturing the whole sequence again, the competence required can become an obstacle [7]. Due to this maintenance problem, companies return to manual regression testing which results in less testing being done and faults that still appear to the users.

Other script-based tools for automated testing at the GUI level are Visual GUI Testing (VGT) tools [8]. These tools take advantage of image processing algorithms to simulate the operations carried out manually by testers on the GUI. These visual testing approaches simplify the work of testers as other test automation tools. However, they also rely on the stability of the graphical appearance of the GUI. Although better than the CR approach, VGT tools also suffer from high scripts maintenance costs [9].

A different and **scriptless** approach to automate testing at the GUI level has been introduced with a tool called TEST Automation at the user inteRface level (TESTAR) tool [1]. TESTAR does not only automate test *execution*, it also automates the *design* of test cases. It does this without the need of test scripts, i.e., TESTAR is a scriptless test approach. And when there are no scripts, these do not need to be maintained. However, TESTAR has SUT-specific configuration that sometimes requires some maintenance effort.

The value of TESTAR in industrial context has been previously investigated and compared, to our knowledge, in four studies (discussed in Sect. 2). Such “which is better” case studies in realistic environments are powerful [10] even if there are a limited number of subjects involved. Although they cannot achieve the scientific rigor of formal experiments and they will never provide generalizable conclusions with statistical significance, the results can provide sufficient information to help companies judge if a specific technology being evaluated will benefit their organization. As Briand et al. [11] clearly, generalizability is overrated. We need more of this type of context-driven software engineering research to reach effective technology transfer of the academic results to the industry. According to Wieringa and Daneva [28] the variability of the context is reduced by decomposing a single case into elements with interactions, such as, for example experience of people involved in a project, defects present in a project, etc. Given that the generalizability might be enhanced due to that these elements may be recurrent across a large set of different cases, we conducted -another study but in a different context (i.e. problem domain, type of software).

The study in this paper has been carried out in the context of an industrial collaboration with a rail infrastructure company in the Netherlands and an IT services company which is global leader in information technology and outsourcing services. The rail infrastructure company is a Dutch company that has mission-critical software applications that ensure the daily train traffic control. Railway tracks are controlled by train traffic controllers using complex real-time systems. Any failure in one of these systems

may cause millions of passengers to be stranded in the stations and instantaneously the Railway Companies become breaking news in bad performance. This company has a real need to automate testing at the GUI level in their context. To this end, the rail company already tried to adopt a commercial Capture & Replay tool. However, this test automation tool was abandoned because of the high maintenance costs due to changes in the GUI. The test coverage reached by the commercial tool got lower and lower, while the company needs high coverage because of the mission critical application. Motivated by this real need, TESTAR was challenged to test a mission-critical web application called “Basis Gegevens Service” (BGS), which is in charge of carefully saving train and rail data. Given the complexity of this application and its IT landscape, the IT services company that developed BGS was committed to support the implementation and research described in this paper.

The structure of this paper is as follows. Section 2 describes related work. Then, Sect. 3 presents the methodology of the case study and provides further explanation of the context of this study. In Sect. 4, the results are presented. Section 5 reflects on the methodological issues, and Sect. 6 discusses the results and implications for practice and further research. Finally, the conclusions of this study are drawn.

2 Related Work

As indicated before, there are currently four case studies that have been conducted to evaluate TESTAR in different industrial contexts [12–14]. In these studies, mainly the efficiency and effectiveness of TESTAR were investigated. The results of these case studies showed that in some contexts, TESTAR could improve the fault-finding effectiveness of current test suites. For example, TESTAR detected 10 previously unknown faults that were classified as critical by the company Clavei¹ in [13]. In case of efficiency, we see that the preparation time of TESTAR is almost similar in all studies compared to the other test automation approach. In the study at SOFTEAM, learnability was also measured. Learnability seemed good to the subjects. However, during the hands-on learning process, they found out that developing test oracles was not well understood. The need to do Java programming caused some initial resistance, but this resistance disappeared after explaining the need of test oracles and turned into some enthusiasm to program them. Eventually, they were able to deliver a working version of TESTAR with test oracles far more sophisticated than the implicit oracles that are present in TESTAR by default.

Based on these case studies and contexts, no general conclusions are possible. More research is needed in different contexts. In this paper, we investigate TESTAR in another context in yet another “which is better” case study and compare it to the manual test approach used in an industrial context of the rail sector with real application and subjects. It would have been desirable to be able to compare TESTAR to another automation approach used in the company, but unfortunately this context did not allow for that [10, 15].

¹ www.clavei.es/.

3 Methodology

3.1 Case Study Design

Similar to the other studies [1, 12–14, 17], we use the methodological framework from the existing literature [16], this framework is based on well-known general case study design guidelines, e.g., by Runeson and Höst [15]. The following sections describe the design of our study.

3.2 Objectives

The research questions are formulated below:

- RQ1: How does TESTAR contribute to the *effectiveness* of testing when it is used in the described industrial context and compared to the current test approach?
- RQ2: What is the *learnability* of TESTAR for the test practitioners in the described industrial context?
- RQ3: How does TESTAR contribute to the *efficiency* of testing when it is used in the described industrial context and compared to the current test approach?
- RQ4: How *satisfied* are the testing practitioners with TESTAR when it is used in the described industrial context and compared to their satisfaction with the current test approach?

3.3 Objects of the Case Study

3.3.1 The Context: Rail Sector

This case study has been carried out in a rail infrastructure company within the Dutch rail sector. The company has approximately 4,000 employees and is responsible for controlling rail traffic all over the Netherlands. The core business of this company is controlling the traffic management, which is achieved through a chain of IT systems.

At the company, continuity of service is very important. Therefore, the applications should operate without faults. To assure a high level of quality, the web application BGS is manually tested in the current situation. This will give the company the ability to evaluate their own current test approach with the TESTAR tool.

3.3.2 The SUT

The BGS application records basic information about trains, which is required for the traffic controllers to control the traffic in the traffic support system. The application is developed by one of the biggest IT companies in Europe tailor-made for the railway company. The basic data can be read, modified, created, deleted, released, and withdrawn. The SUT is tested after each release. The web application is Java based (JEE6) and has 12,263 lines of code (LOC). This SUT has more LOC than in the previous case study of SOFTEAM.

The web application consists of three layers: the interface layer, the service layer, and the persistence layer. The interface layer contains the GUI and a Java Message

Service (JMS) receiver. It is set up with PrimeFaces² and Java Server Faces (JSF³). The service layer is the layer between the interface and the persistence layer. Finally, from the persistence layer, communication takes place with an Oracle database. Sending data to other applications takes place through JMS from the service layer. Tibco EMS⁴ is used as a JMS provider. Through the JMS receiver from the interface layer, the web application can receive data from other applications.

3.3.3 Manual Test Approach

The existing test approach used for the described SUT was manual and contained a test plan with a corresponding test suite of 100 test cases. After a change in the application, the SUT had to be re-tested and all the 100 test cases were executed as regression tests. The test cases were structured as follows: precondition, action, and expected results. To have an indication of the quality of the test suite, the company has defined a functional test coverage criterion to provide the business (customer) information about the coverage. The functional coverage is clearly preferred by the business instead of code coverage.

This functional test coverage is measured by logging the executions of the SUT. The web application BGS has 251 explicit functionalities. When executing any of these functionalities through the GUI level of the web application, this is logged. In this way, it is possible to calculate functional test coverage after executing the test suite. After execution of each test approach, the logs are inspected and the functional test coverage is determined. The test suite consisting of the abovementioned 100 test cases covered 73% of the defined functional test coverage.

The test suite is maintained in a quality management system. In this system, the test cases are grouped based on functionalities from the use case. As indicated, the test cases are performed manually. If a fault is found, it is registered in the issue tracking part of the quality management system.

Within the company, the testing process consists of preparing, specifying, executing, and evaluating. It took 43 h to prepare the test cases. Subsequently, manual testing of the test cases takes 6 h and the evaluation of the results another 2 h.

3.3.4 TESTAR: A GUI-Level Test Approach

TESTAR⁵ is an open source tool for traversal-based scriptless test automation at the GUI level. It carries out automated testing following the test cycle depicted in Fig. 1. The cycle consists of the following steps: (A) TESTAR starts the SUT. Thereafter it will (B) scan the GUI using the accessibility API of the operating system. The latest version of TESTAR also supports Selenium WebDriver, but during this case study, it was not available yet. This API provides information about all the widgets that are available in the state that the GUI is in. TESTAR uses this information to (C) derive a set of possible GUI actions that a user can execute in the current state s of the SUT. Subsequently, TESTAR randomly selects an action from this set (D) and executes it (E). This makes

² www.primefaces.org.

³ JSF is a Java-based component to build a GUI.

⁴ <https://www.tibco.com/products/tibco-enterprise-message-service>.

⁵ www.testar.org.

the SUT go into a new GUI state s' . This new state s' is evaluated with the available oracles (F). If no fault is found, again the set of possible GUI actions for the new state s' is derived, one action is selected randomly and executed. This continues until a fault has been found or until a stopping criterion is reached (G). In each state, all widgets are combined in a tree structure, the widget tree. Based on this widget tree, actions are derived, and one is selected. This way, the state of the GUI is continuously obtained and actions are selected based on the current implementation of the SUT [1, 19].

Fig. 1. The operational flow of TESTAR

The default configuration of TESTAR includes random selection of actions and implicit oracles for the detecting the violation of general-purpose system requirements, e.g., that the SUT should not crash, the SUT should not find itself in an unresponsive state (freeze), and the UI state should not contain any widget with suspicious titles, like *error*, *problem* or *exception*.

3.4 Subjects in the Case Study

In the case study, three subjects (test practitioners) were involved. These subjects were certified technical or functional testers working at case companies.

Within the companies, a technical tester is considered to have knowledge of programming (reading and writing code) and understands the operation of an application; moreover, he or she can view and interpret the logs of applications. A functional tester, on the other hand, tests the SUT through the GUI level and has limited knowledge on the technical implementation of an application.

The *technical tester* subject is identified as subject 1 and is graduated from a master's degree in computer science. This subject has 7 years of experience with software testing, of which 2 years in the rail sector. The functional tester subjects are identified as 2 and

3. Subject 2 has 12 years' experience with functional software testing and worked all these years in the rail sector. Subject 3 has four years of experience with functional software testing. Again, all four years the subject worked within the rail sector. The two subjects (2 and 3) with functional background have a bachelor degree in another field than computer science.

The SUT has been under development for 2 years. Subjects 2 and 3 have been involved from the start. All subjects have tested the SUT at least once.

3.5 Phases of the Case Study

The preparation consisted of 3 parts:

- **Deployment** (implementation) consisted of setting up the test environment such that the GUI widgets of the SUT were properly detected by TESTAR and actions on the GUI could be automated in a sandbox.
- **Training** consisted of reviewing the user manual and viewing an instructional video about TESTAR. Subsequently, three sessions of training were given. After the three sessions, the level of knowledge was measured using a questionnaire about TESTAR.
- **Hands-on learning** process makes the subjects work towards a configuration of TESTAR that was applicable to start testing the SUT. Basically, this means that action filtering and oracle definitions have to be put in place. The final working version was used to test the SUT and the results were evaluated with the exit criteria. This review took place with the trainer.

During the execution and evaluation phase, TESTAR was run, data was collected and both TESTAR and the manual test approach were compared. For TESTAR, the execution phase took a minimum of 24 h, as this has been shown in previous studies to reach a certain test coverage [12, 13]. The following sections discuss the data collected from the runs and the evaluation.

3.6 Variables and Metrics

3.6.1 Effectiveness

Effectiveness shows the ability of the approach to detect faults in the SUT [16]. Some studies evaluate test approaches by injecting known faults into the SUT [12] and calculating the (injected) fault detection rate. In this study, it was not allowed to inject faults into the SUT because this would represent a risk in the test environment for other projects, because there is a complex IT landscape and the applications are connected with each other. Hence we have used surrogate measures like coverage [1, 18, 21, 22] to measure the percentage of the total code of an application has been tested with the test execution. In our study, effectiveness was measured in terms of:

- Number of detected faults during testing with both test approaches.
- Impact or severity of the relevant faults if the SUT was released in the production environment. This impact consists of cosmetic (low), disturbing (middle), serious

(high), and production-blocking (top) faults. The case organization is using these categories to determine the severity.

- Functional test coverage of both test approaches. This was measured by dividing the number of tested functionalities by the total number of functionalities.

Moreover, three subjects are interviewed with semi-structured interviews [16] to complement our measurements with qualitative data.

3.6.2 Learnability

We measured the learning process using the level-based strategy method [1], as well as the time required to learn TESTAR [17]. In this way, we can investigate the learnability step by step in each level. The level-based strategy method consists of reaction, learning, and performance:

- Reaction: interviews and observations were used to evaluate how the subject reacted to the learning process. After the training, the level of knowledge was examined with a questionnaire.
- Learning: the hands-on learning process was recorded in a working diary where all activities and time needed for them are logged. Through the log and observations, knowledge growth was measured.
- Performance: the TESTAR version, delivered in the hands-on learning process, was checked through exit criteria. The exit criteria were conditions based on the hands-on learning process, and were determined as follows:
 - The protocol (Java code) does not provide errors after execution the compiler;
 - TESTAR run for more than 24 h;
 - TESTAR performed more than 50,000 actions.

3.6.3 Efficiency

Efficiency was measured by monitoring the time required for the test activities during the preparation, implementation, and evaluation phases. During the execution, all observations were also recorded in a working diary [1, 8, 12, 13, 17–19, 21–24].

3.6.4 Subjective Satisfaction

Subjective satisfaction is measured using reaction cards and semi-structured interviews. Semi-structured interviews were conducted with all three subjects. Questions and topics were related to ease of use, understandability of the user manual, positive/negative reactions, loyalty and overall satisfaction.

The *reaction cards* are a collection of words from which the subjects can choose (e.g., too technical, time-saving, stressful, efficient, high quality, etc.) that have been used successfully to measure subjective satisfaction [25]. The subjects had the opportunity to choose five keywords that they associated with the test approaches.

4 Results

4.1 RQ1: Effectiveness of Both Test Approaches

4.1.1 Functional Coverage

Functional test coverage was determined using the logs of the SUT. When executing a determined functionality, the SUT registers its status and detail in a log file. Table 1 shows that the functional test coverage of TESTAR is between 33% and 80%. The functional test coverage varies because it depends on the duration of the test, type of algorithm and number of actions has an impact on the functionality. The third run achieved the highest functional coverage, 80%, and all exit criteria were met. In comparison with TESTAR, the manual test approach only had a functional coverage of 73% (see Table 2). This is the first study and context where functional coverage or TESTAR out performs the manual approach.

Table 1. Run results of TESTAR

Run	1	2	3
Duration (hours)	18	33	71
Number of sequences	276	12	192
Action selection (type of algorithm)	Random	Random	Random
Functional coverage	33%	51%	80%
Number of actions	21.377	47.487	98.081
Number of faults	0	1	2
Reproducibility	–	No	Yes

Table 2. Effectiveness of both test approaches

Description	Manual test approach	All runs of TESTAR
Number of faults detected	0	4
Impact of the fault(s)	–	High
Reproducibility fault(s)	–	3 out of 4
Functional coverage	73%	80%
Required time for preparation	43	44
Execution time (hours)	6	71 (run 3)
Required time for evaluation/analysis	2	5
Number of test cases/number of test sequences	100	192

In runs 1 and 2, TESTAR ran for 18 and 33 h, respectively. These runs were experimental and intended for the training phase. In run 3, TESTAR ran for 71 h. Although

TESTAR needed time to achieve high functional coverage, no human resources were needed.

4.1.2 Faults Found and Severity

As shown in Table 2, TESTAR detected three faults with random action selection, three of which were reproducible. In contrast, the manual test approach did not detect any faults. The type of the reproducible faults that were found by TESTAR were:

- one *null pointer exception* due to clicking multiple times on a button;
- one *functional fault* while exporting to Excel to release the day plan;
- one *concurrent modification exception* when clicking two times on a button in a row without waiting.

All the three faults were unique and assigned high severity since they forced the end user to restore the application for further use. The criteria for assigning the severity is based on the impact of the fault on the business processes and end user. In this context, the end user must do actions, therefore the faults are classified as high (serious). The severity-level is defined this way in the context of the case organization and the subjects classified the faults based on the criteria similar to their daily practice.

TESTAR used random action selection in the first three runs. Random selection has no limitations and clicks anywhere on the GUI level. Run 3 showed that random selection is capable of achieving high functional coverage [26].

4.1.3 Qualitative Data

Based on the interviews, we find that the subjects feel that the manual test approach provides more structure and focus, since the effectiveness of TESTAR depends on the configurations and settings. However, they appreciate the fact that TESTAR can test automatically, and offers a variety in the coverage of functionalities. This allows for a higher functional test coverage to be achieved.

Finally, it appears that the preparation of both test approaches takes a similar amount of time, although TESTAR can test the implementation without the use of resources; see Fig. 2 for the qualitative comparison between both test approaches.

Summarizing, the effectiveness of TESTAR is higher compared to that of the manual test approach: more faults were found, and the functional test coverage was higher.

4.2 RQ2: Learnability of TESTAR

The second research question looks for answers related to the learnability of TESTAR for test practitioners in the described industrial context. Our findings are shown in each level: reaction, learning and performance.

4.2.1 Reaction – Learning TESTAR

Initially, the test approach and purpose of TESTAR was not entirely clear to the subjects. According to them, this information was not described in the user manual. Their first

Manual test approach	TESTAR
<p>continuous functional coverage.</p> <p>Provides focus because specific situations and boundaries can be tested.</p> <p>The ability to find faults depends on the test techniques.</p> <p>Based on the test cases, what has been accurately tested (and which paths) is traceable.</p> <p>It takes time and resources because it must be done manually.</p>	<p>coverage can be achieved.</p> <p>Provides variety in the tests, testing the functionalities that are not in the existing test suite.</p> <p>The ability to find faults depends on the configuration and settings of the tool.</p> <p>Based on the state graph, it is clear which widgets have been tested.</p> <p>It takes time to learn and configure the tool. After that it can be used</p>

Fig. 2. Qualitative comparison obtained from the interviews

experience made them somewhat sceptical about TESTAR. However, during the training sessions, while the subjects practiced using TESTAR, their scepticism disappeared and their enthusiasm grew. The subjects gained more of an understanding of the test approach of TESTAR. This is valuable feedback for the TESTAR team to enhance the manual and make people enthusiastic. Concerning the training and the time needed. The level of understanding that the participants had of TESTAR in order to be able to start testing proved sufficient after training. Altogether, learning TESTAR took about 10 h.

4.2.2 Learning – Hands on Learning Process

Although the subjects encountered challenges during the learning process, the trainer's support solved all the problems, enabling the subjects to effectively learn. The subjects configured TESTAR separately and created oracles (suspicious titles and crash) and filters (filtering exit and browser buttons). Later, they merged the TESTAR modifications and came to one modified version.

Subjects 2 and 3 worked together because they struggled to edit the protocol having less programming experience. Finally, during the hands-on learning process, the subjects' knowledge level grew. Particularly after solving the problems, they gained significantly more knowledge about TESTAR.

4.2.3 Performance of TESTAR

The first version ran for 18 h and was validated with the defined exit criteria. In the second run of TESTAR, the protocol and oracles worked well. To meet the exit criterion regarding test actions (more than 50,000) we needed to run TESTAR autonomously for longer than 33 h. Consequently, with 33 h, the exit criterion regarding the hours (more than 24 h) was also met.

In the third run, TESTAR ran autonomously for 71 h, and the exit criteria were also reached.

4.3 RQ3: Efficiency of Both Test Approaches

The third research question investigated TESTAR’s contribution to the efficiency when used in a real-life industrial context and compared to the current test approach.

Table 3 shows the duration of each of the phases in this study. For the manual test approach, this was data obtained from estimations of the subjects. For TESTAR, a working diary was kept which recorded the activities of the three subjects in each phase together with the time they spent on them.

Table 3. Time required for the phases

Manual test approach		TESTAR	
Phase	Hours	Phase	Hours
Preparation phase	43	Preparation phase	44
Detail intake	8	Implementation phase	2
Specify	35	Training phase - reaction	10
		Training phase - learning - hands-on	17
		Training phase - performance	13
Execution phase	6	Execution phase (run 3)	71
Evaluation phase	2	Evaluation phase (run 3)	5

The efficiency results show that the preparation time for the two test approaches is similar. During the preparation phase of TESTAR ten hours were spent on training to level up the knowledge of the subjects. This time is a one-time investment.

TESTAR autonomously spends 71 h testing while the manual test approach takes six hours to complete. However, as TESTAR needs no human resources to execute the tests, the test execution has no additional human costs.

The time spent during the evaluation phase of TESTAR was evidently higher than that of the manual test approach, since the latter did not find any faults. For each of the faults found with TESTAR, time was needed to reproduce them and find their type.

Summarizing, in terms of efficiency, the preparation time is similar for both approaches, but TESTAR can execute tests without the need for human resources. This means that no additional human costs are incurred, which contributes significantly to efficiency.

4.4 RQ4: Subjective Satisfaction

The last research question is about the satisfaction of the subjects. Our findings from the interviews are as follows.

4.4.1 The Interview Results

Based on the conducted interviews, we found the following.

Easy to Use. The subjects were satisfied with the logging facilities of TESTAR. In this context, the state graph was found to be highly powerful because it provided an overview of the executed test actions. On the other hand, the subjects lost a great deal of time analysing the logs. In this case, they would have appreciated a kind of dashboard to summarize the run results (number of faults, number of tests, duration and test coverage), which could save time in the analysis. The subjects reported that “regression testing with TESTAR could save time on maintenance of test cases compared to the manual test approach”. This is because after a change, they experienced the maintenance and adjustment of test cases as labour-intensive with the manual test approach. Also, the subjects found the flexibility of TESTAR to be good, because it is easily transferable in various test environments. Subject 1 was satisfied with the features of the approach, such as the oracles, filters and protocol. This was because it allows new functionalities to be added. Subjects 2 and 3 were unsatisfied with the protocol, because they were not technical enough to add sophisticated oracles with Java programming. They expected more of a functional layer to provide the possibility to easily add functionalities or oracles. All in all, the subjects found TESTAR to be a refreshing and time-saving tool.

Understandability of User Manual. The manual was found to be insufficient to understand TESTAR. Although the user manual was improved after the other studies, it still is not clear enough for the industrialists in this case study. The academics were urged again to put more effort into a clear manual. For example, the use and detection of logging faults is only explained to a limited extent. Also, the oracles and the needed Java programming again in this study seems to be a barrier. Although some of it might be solved with a better user manual section on that, the fact is still that in industry most testers do not have programming skills.

Positive/Negative Reactions. The subjects’ first impression of TESTAR was less positive, because the purpose of the approach was unclear to them and did not fit with their traditional ways of automated testing. Now that they were faced with a completely different test approach, they needed explanation and guidance. They were not familiar with the functionalities and possibilities of TESTAR. Their doubts further increased when they initially encountered the user manual. Because the way of working remained unclear. However, after the training, they developed a positive impression of the approach. This included the explanation during training and the run results that were booked. The subjects gained more confidence from the results that were achieved.

Loyalty. The subjects were asked if they would recommend the tool to the management and to colleagues. The first subject would definitely recommend the tool to management; the second subject wanted to wait to do so; and the third believed that TESTAR should have a chance to develop. Subject 1 recommended TESTAR probably because he had a technical background and because of that gave a higher satisfaction rating compared to the subjects with a functional background (subjects 2 and 3).

Overall Satisfaction. Both test approaches scored a 7 on satisfaction. The first subject gave TESTAR an 8, and the manual test approach a 6. This subject considered the manual test approach to be rigid, and TESTAR powerful. Whereas, the other two subjects found the manual test approach to be traceable and to provide structure, and therefore gave it

an 8. In contrast, they gave TESTAR a 7. They both indicated that TESTAR could score a higher satisfaction rating if it was more user-friendly.

4.4.2 Response Cards

Finally, the response cards showed that TESTAR and the manual test approach were generally associated with keywords that have a positive charge. The manual test approach also contained one negative point: “time-consuming in use”. In addition, the subjects found TESTAR too technical in use, because of the protocol which needs programming skills.

Overall, the test practitioners in the real-life setting were equally satisfied with both test approaches (see Fig. 3).

Manual test approach	TESTAR
Manual test approach	Efficient
Time consuming in use	Too technical (3)
Organizing	Fast
Easy to use (2)	Flexible (2)
Relevant (2)	Advanced (2)
Consistent	Time-saving
Predictable	Relevant
To be trusted	Useful (2)
Slowly	Customizable
Reliable	Exciting
Known	Fun
Useful	
Valuable	

Fig. 3. Results of the chosen reaction cards

5 Threats to Validity

Construct validity reflects to what extent our operational measures really represent what is investigated according to the research questions [28]. The learnability evaluation was based on a four-level strategy and we used logs for the learning level. Most of the collected data was based on subjects’ responses (interviews, post questionnaires, working diaries). However, in order to reduce possible misinterpretations of formulated questions and answers gathered, data analysed and interpreted by the second author was also validated by the subjects. Regarding effectiveness and efficiency, we used well-known defined metrics used also in the previous studies.

Internal validity is the extent to which a causal relationship between the variables is justified [28]. To mitigate this threat:

- The interviewees were informed about both the interview itself and the interview topic. As a result, they had enough time to prepare and were less quickly surprised by a question. This reduced the likelihood of unnecessary mistakes.

- The interviews took place in an area where no others were present. If a colleague or manager had been in the room, then the subjects would have been likely to give socially desirable answers. Furthermore, the anonymity of the subjects was ensured.
- The (semi-structured) interview questions concerning effectiveness and satisfaction were formulated to be open to obtain input from the subjects. We actively requested details of the activities and events that the subjects had experienced. Moreover, the critical incidence method was applied [30 p. 230]. Finally, the interview questions and results were reported to allow for replication.

External validity concerns the generalization of results [28]. As indicated before, this research is context-driven and hence not undertaken with the goal of generalizability. However, the more studies of this type we do, the more chance of eventual generalizability will be established through secondary studies.

To ensure the **reliability** of the research, all the procedures, questions, logs, were reviewed by another researcher.

6 Discussion, Conclusion and Implications

6.1 Discussion and Conclusion

In the current literature, four case studies have been reported on evaluating TESTAR in industry. In this case study, we did an evaluation with a Java-based web application (BGS). Also, the rail sector has not been studied in the previous case studies.

One case study in the literature (SOFTEAM) showed that the difference in preparation time between test approaches was approximately equal and execution of manual test approach took 1 h [12]. While TESTAR took 77 h without the use of resources. The results of our study are similar. TESTAR was used to test for 71 h, and the manual test approach for 6. For future work, it is interesting to compare in more companies to see whether the preparation time for both test approaches are approximately the same, but TESTAR can be executed without the use of human resources.

During the training, the subjects without a technical background needed more time to learn the oracles. It appears to be too difficult for functional testers with limited programming experience to learn to extend the protocol and develop oracles. Also, the case study in the literature showed that the subjects had some difficulties with programming oracles [12]. An important question that arises here is whether we can expect test engineers to have programming skills. It sounds almost logical, but in practice it turns out that not all testers have programming skills.

The main question of this research was as follows: How does the test approach of TESTAR compare to that of the manual test approach?

Looking at the results, the test approach of TESTAR was found to be more effective and efficient than that of the manual test approach. This was because TESTAR detected more faults than the manual test approach, and also showed higher functional test coverage. Thereby, the test execution can be realized without the use of human resources.

Regarding satisfaction, both test approaches were found to be identical. TESTAR needs improvements in user-friendliness, as it is too technical in use, and some points

could improve its usability. On the other hand, TESTAR was found to be learnable. It scored significantly higher on the variables of effectivity and efficiency compared to the manual test approach.

6.2 Theoretical and Practical Implications

A limitation of this type of research is that the results cannot be fully generalized, since a single case study was conducted. When the research is placed in a broader perspective, it appears that the results only apply to a Java-based web application. However, the study was very useful for technology transfer purposes: some remarks during the interviews indicate that TESTAR would not have been evaluated in so much depth if it would not have been backed up by our case study design. For follow-up research, it is advisable to repeat this research with multiple case studies and with different types of applications, so that the results can be generalized. Finally, having only a few real subjects available, this study took several weeks to complete and hence we overcame the problem of getting too much information too late.

In this study, TESTAR was compared to the manual test approach. In further research it is advised to involve a second GUI test automation tool. In doing so, the results of the case study could be explored in a broader perspective than only comparing it to the manual test approach.

The benefits of both test approaches can be extracted and combined during testing. TESTAR can be used to perform flexible and automated regression testing, while the manual test approach can be used to test a specific scenario. Based on this complementary use, cost and resources can be saved. Based on this case study the company will use TESTAR complementary with another test approach. In the future, the company is interested in follow-up research to determine if it can replace the current test approaches.

Acknowledgments. We would like to acknowledge the help of the case study companies They have been supporting our research, answering our questions, and validating our analyses. We would also like to thank the involved universities. This work was partially funded by the ITEA3 TESTOMAT project and the H2020 DECODER project.

References

1. Vos, T.E.J., et al.: TESTAR: tool support for test automation at the user interface level. *Int. J. Inf. Syst. Model. Des.* **6** (2015)
2. Kresse, A., Kruse, P.M.: Development and maintenance efforts testing graphical user interfaces: a comparison. In: *Proceedings of the 7th International Workshop on Automating Test Case Design, Selection, and Evaluation (A-TEST 2016)*, pp. 52–58 (2016)
3. Grechanik, M., Xie, Q., Fu, C.: Experimental assessment of manual versus tool-based maintenance of GUI-directed test scripts. In: *ICSM* (2009)
4. Nguyen, B.N., Robbins, B., Banerjee, I., Memon, A.: GUITAR: an innovative tool for automated testing of GUI-driven software. *Autom. Softw. Eng.* **21**(1), 65–105 (2013). <https://doi.org/10.1007/s10515-013-0128-9>

5. Garousi, V., et al.: Comparing automated visual GUI testing tools: an industrial case study. In: ACM SIGSOFT International Workshop on Automated Software Testing (A-TEST 2017), pp. 21–28 (2017)
6. Aho, P., et al.: Evolution of automated regression testing of software systems through the graphical user interface. In: International Conference on Advances in Computation, Communications and Services (2016)
7. Leotta, M., et al.: Capture-replay vs. programmable web testing: an empirical assessment during test case evolution. In: Conference on Reverse Engineering, pp. 272–281 (2013)
8. Alégroth, E., Nass, M., Olsson, H.H.: JAutomate: a tool for system- and acceptance-test automation. In: Proceedings - IEEE 6th International Conference on Software Testing, Verification and Validation. ICST 2013, pp. 439–446 (2013)
9. Alégroth, E., Feldt, R., Ryrholm, L.: Visual GUI Testing in Practice: Challenges, Problems and Limitations. *Empir. Softw. Eng. J.* **20**(3), 694–744 (2015). <https://doi.org/10.1007/s10664-013-9293-5>
10. Kitchenham, B., Pickard, L., Pfleeger, S.: Case studies for method and tool evaluation. *IEEE Softw.* **12**, 52–62 (1995)
11. Briand, L., et al.: The case for context-driven software engineering research: generalizability is overrated. *IEEE Softw.* **34**(5), 72–75 (2017)
12. Bauersfeld, S., et al.: Evaluating the TESTAR tool in an industrial case study. In: Proceedings of the 8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement - ESEM 2014, pp. 1–9. ACM Press, New York (2014)
13. Bauersfeld, S., de Rojas, A., Vos, T.E.J.: Evaluating rogue user testing in industry: an experience report. Universitat Politècnica de Valencia, Valencia 2014
14. Martínez, M., Esparcia, A.I., Rueda, U., Vos, T.E.J., Ortega, C.: Automated localisation testing in industry with test*. In: Wotawa, F., Nica, M., Kushik, N. (eds.) ICTSS 2016. LNCS, vol. 9976, pp. 241–248. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-47443-4_17
15. Runeson, P., Host, M.: Guidelines for conducting and reporting case study research in software engineering. *Empir. Softw. Eng.* **14**(2), 131–164 (2009). <https://doi.org/10.1007/s10664-008-9102-8>
16. Vos, T.E.J., et al.: A methodological framework for evaluating software testing techniques and tools. In: 2012 12th International Conference on Quality Software, pp. 230–239 (2012)
17. Condori-Fernández, N., et al.: Combinatorial testing in an industrial environment - analyzing the applicability of a tool. In: Proceedings - 2014 9th International Conference on the Quality of Information and Communications Technology, QUATIC 2014, pp. 210–215 (2014)
18. Borjesson, E., Feldt, R.: Automated system testing using visual GUI testing tools: a comparative study in industry. In: ICST 2012 Proceedings of the 2012 IEEE Fifth International Conference on Software Testing, Verification and Validation, pp. 350–359 (2012)
19. Nguyen, C.D., et al.: Evaluating the FITTEST automated testing tools: an industrial case study. In: 2013 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement, pp. 332–339 (2013)
20. Rueda, U., et al.: TESTAR -from academic prototype towards an industry-ready tool for automated testing at the User interface level (2014)
21. Imparato, G.: A combined technique of GUI ripping and input perturbation testing for Android apps. In: Proceedings - International Conference on Software Engineering, pp. 760–762 (2015)
22. Vos, T.E.J., et al.: Industrial scaled automated structural testing with the evolutionary testing tool. In: ICST 2010 – 3rd International Conference on Software Testing, Verification and Validation, pp. 175–184 (2010)
23. Bae, G., Rothermel, G., Bae, D.-H.: Comparing model-based and dynamic event-extraction based GUI testing techniques: An empirical study. *J. Syst. Softw.* **97**, 15–46 (2014)

24. Marchetto, A., Ricca, F., Tonella, P.: A case study-based comparison of web testing techniques applied to AJAX web applications. *Int. J. Softw. Tools Technol. Transf.* **10**, 477–492 (2008). <https://doi.org/10.1007/s10009-008-0086-x>
25. Benedek, J., Miner, T.: Measuring desirability new methods for evaluating desirability in a usability lab setting. Microsoft Corporation (2002)
26. Esparcia-Alcazar, A., et al.: Q-learning strategies for action selection in the TESTAR automated testing tool. In: *Proceedings of the 6th International Conference on Metaheuristics and Nature Inspired Computing. META* (2016)
27. Bohme, M., Paul, S.: Probabilistic analysis of the efficiency of automated software testing. *IEEE Trans. Softw. Eng.* **42**, 345–360 (2016)
28. Wieringa, R., Daneva, M.: Six strategies for generalizing software engineering theories. *Sci. Comput. Program.* **101**, 136–152 (2015). <https://doi.org/10.1016/j.scico.2014.11.013>. ISSN 0167-6423