ManualTEST: Improving Collection of Manual Test Execution Data in Empirical Studies

Eduardo Aranha, Paulo Borba

[1]Informatics Center Federal University of Pernambuco (UFPE)
Av. Professor Luís Freire s/n - Cidade Universitária – 50.740-540 – Recife – PE – Brazil

{ehsa,phmb}@cin.ufpe.br

Abstract. *Manual test execution can represent a significant part of the software testing effort. For this reason, new technologies are being proposed and evaluated through empirical studies to help the planning of test execution teams and to reduce the cost of manual test execution. The validity of these studies strongly depends on the accuracy of the collected data, which can be affected by several problems during the execution of the studies. In this paper, we present ManualTEST, a tool developed to improve the accuracy of the collected data and provide information to locate and identify sources of problems occurred during the execution of these studies. The benefits of ManualTEST were confirmed by several studies run in two test sites of a major mobile phone manufacturer.*

Keywords. Manual testing, data collection, execution time, outlier detection.

1. Introduction

Software testing is an important activity that usually requires a significant effort. In addition, testing is being considered so important that an organization can allocate teams exclusively for testing activities in order to achieve unbiased test results [Broekman and Notenboom 2002]. When regarding product lines of embedded systems, test managers usually have to deal with high cost to port automated tests for different platforms (operating system, hardware version, etc.) and limited automation technology for testing some types of features (sounds, videos, pictures, etc.).

In this context, manual test execution can represent a significant part of the testing effort. This is even more evident when considering regression tests, that is, when you have to execute tests to check if the whole application (or system) still working after some change. For this reason, new technologies are being investigated to help the planning of test execution teams and to reduce the cost of the manual test execution. Examples of research in this area are the development of test prioritization techniques, development of test execution effort estimation models and the investigation of manual test execution cost drivers. Nevertheless, all proposed solutions must be properly evaluated to justify their adoption by industry.

A common way to evaluate new technologies is the running of empirical studies [Shull et al. 2008], such as case studies and controlled experiments. Despite of the control of confounding factors and the use of statistical analyses, the validity of these studies can strongly depends on the accuracy of the collected data. The importance of having accurate data collection mechanisms in empirical studies is discussed in several studies

in software engineering, such as [Selby 2007], [Kitchenham et al. 2002], [Seaman 2008] and [Singer et al. 2008].

Regarding manual test execution activities, several problems can occur during the test execution. For instance, phone calls and test environment changes (e.g., network bandwidth) can affect tester performance and possibly change the results. Hence, the data collection instrument should avoid these types of problems or, at least, provide information to detect them.

In this paper, we present ManualTEST (Manual Test Execution assiSTant), a tool developed to automate part of the data collection procedures of empirical studies involving manual test execution. With ManualTEST, we can improve the accuracy of the collected data and provide information to locate and identify sources of problems occurred during the execution of these studies. ManualTEST was successfully used in several studies run in two real test sites that provide testing services for a major mobile phone manufacturer.

2. Manual Test Execution

In empirical studies involving manual test execution, it is necessary to have at least one or more of the following components: tester, product under tested, test specification and test environment. In a manual test execution, a tester reads and executes each step of a test specification using a product under test and a test environment.

The test specifications are usually described in terms of pre-conditions (initial conditions to start the test), procedure (list of test steps with inputs and expected outputs) and post-conditions [Jorgensen 2002]. In addition, they can contain more information, such as a test description, test objectives, the tested requirements and a software and hardware setup procedure. These specifications are commonly written in natural language, as shown in Figure 1.

The data collected during manual test execution depend on the objective of the study, but the most common collected data are:

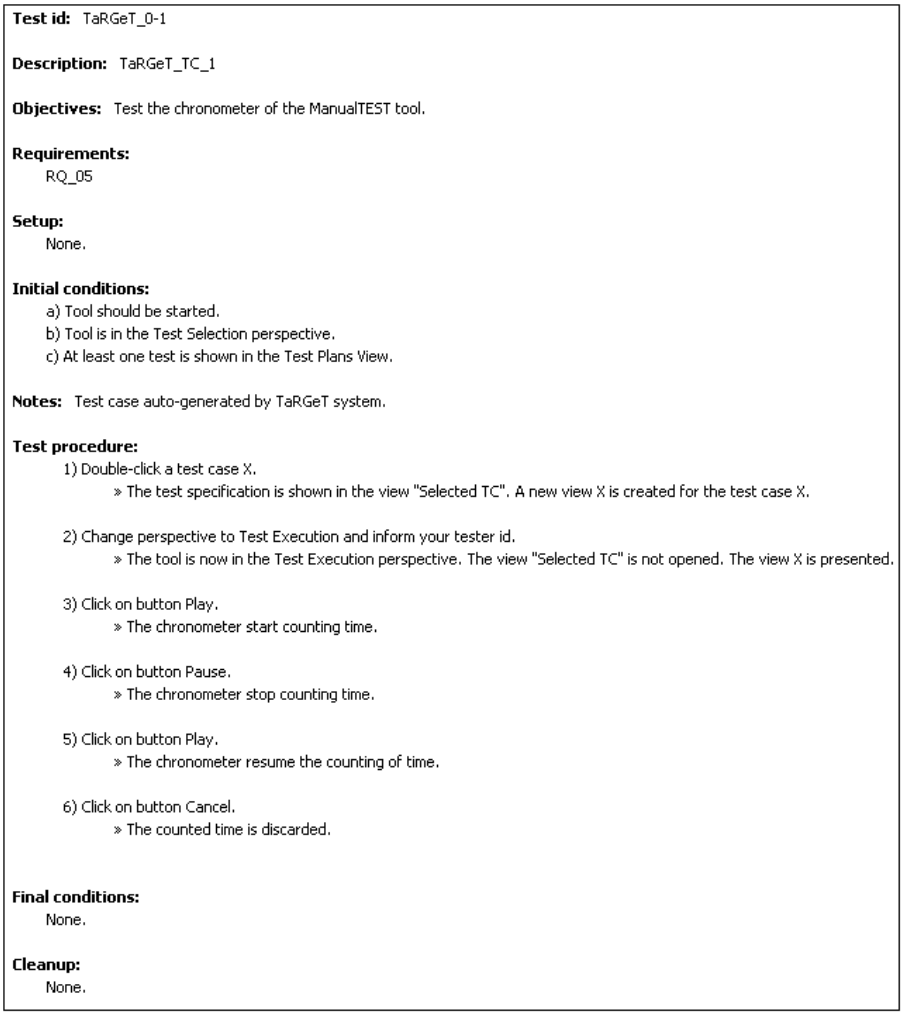
- Test id.
- Tester id.
- Date and time in which the test execution was started.
- Test execution time, which is the time spent to execute the test.
- Test result.

In general, test result indicates if the test was executed successfully (passed), if a bug was detected (failed) or if the tester could not execute the test (blocked) due to some constraint, such as lack of resource, test not applicable to the product under test, etc.

During the manual test execution in an empirical study, some problems can occur and threat the validity of the data analysis:

P1. While executing the tests, testers have to start counting test execution time just before the beginning of the test execution activity and to stop just after finishing it. The tester can forget to start or stop the chronometer, including between test executions, reducing the accuracy of the collected data.

P2. When running studies in industrial settings, participants may be interrupted by important phone calls or by other emergencies [Arisholm et al. 2002]. The time spent during these interruptions should not be taken into account.



Test id: TaRGeT_0-1

Description: TaRGeT_TC_1

Objectives: Test the chronometer of the ManualTEST tool.

Requirements:
RQ_05

Setup:
None.

Initial conditions:
a) Tool should be started.
b) Tool is in the Test Selection perspective.
c) At least one test is shown in the Test Plans View.

Notes: Test case auto-generated by TaRGeT system.

Test procedure:
1) Double-click a test case X.
» The test specification is shown in the view "Selected TC". A new view X is created for the test case X.

2) Change perspective to Test Execution and inform your tester id.
» The tool is now in the Test Execution perspective. The view "Selected TC" is not opened. The view X is presented.

3) Click on button Play.
» The chronometer start counting time.

4) Click on button Pause.
» The chronometer stop counting time.

5) Click on button Play.
» The chronometer resume the counting of time.

6) Click on button Cancel.
» The counted time is discarded.

Final conditions:
None.

Cleanup:
None.

Figure 1. Sample test specification written in natural language.

P3. These interruptions should also be recorded to verify if they impacted the study [Arisholm et al. 2002]. In our context, these interruptions can impact the total test execution time and the tester performance.

P4. From previous experiences [Aranha and Borba 2007], we observed that the effect of some confounding factors can be hard to detect, such as changes in network bandwidth or in other dynamic environmental conditions. These problems are usually hard to avoid when considering studies in industrial settings, even when running supervised experiments.

P5. For some studies, such as test prioritization based on data reuse [Lima et al. 2008], it is important to distinguish between test procedure time, test setup time, etc., making difficulty the data collection.

To solve these problems, we developed and used a tool to collect manual test execution data, as described in the next sections.

3. Manual Test Execution assiSTant (ManualTEST)

This section presents the main functionalities of ManualTEST, a tool developed to automate the collection of manual test execution data, avoiding the problems previously

described. The tool was developed using the Eclipse Rich Client Platform (RCP) [McAffer and Lemieux 2005] and have two different perspective, one to select the tests to be executed and another one to support the test execution activity. Next, we present the details of these two perspectives and information about the data collected by the tool.

3.1. Test Selection

To automatically read test specifications, ManualTEST consider that these specifications are stored in spreadsheets files. In addition, the tool can be customized to read spreadsheets in different formats. Once a spreadsheet file is opened, the tests are listed to the user, as shown in Figure 2. Then, the tester must double-click to open the tests that she is going to execute.

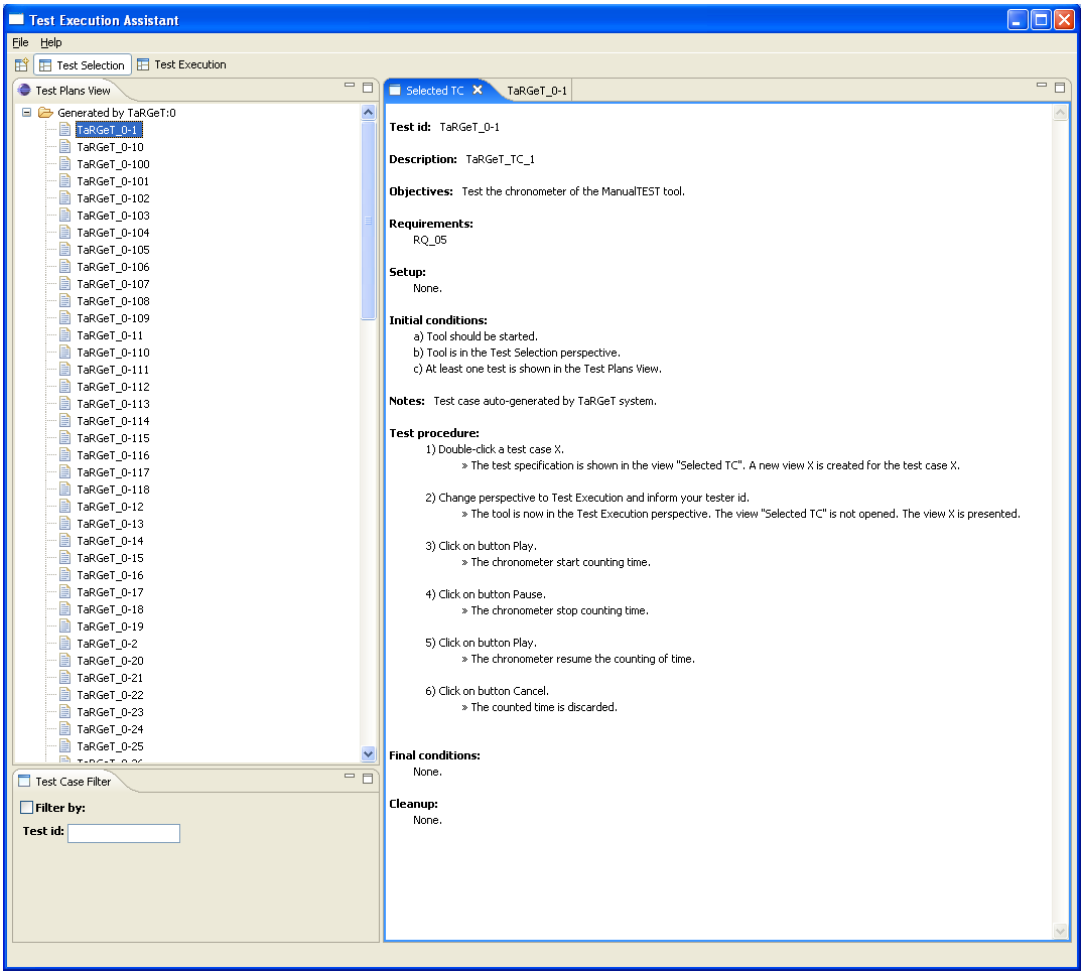


Figure 2. Test selection perspective of ManualTEST.

3.2. Test Execution

When the tester selects the Test Execution perspective, only the selected tests are presented. A Test Execution Controller view is also presented, as shown in Figure 3. In this view, the play and the pause buttons are presented to the tester to control the counting of time. This integrated functionality make easy to the tester to start, pause and resume the test execution time (problem P1 and P2).

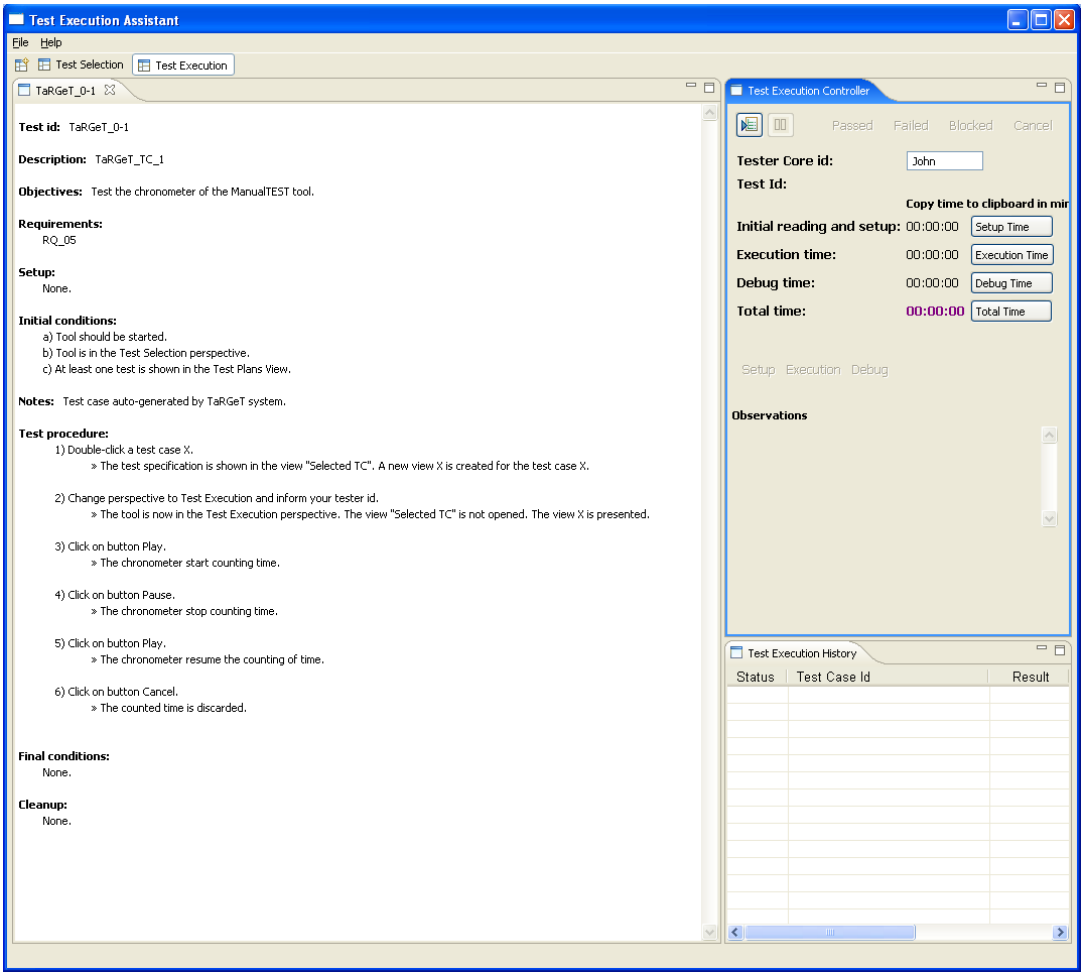


Figure 3. Test execution perspective of ManualTEST.

When the tester pushes the play button, the chronometer is activated and the first line of the test specification is highlighted. Using keys ↑ and ↓ from the keyboard, the tester executes the highlighted test step and go to the next one or go back to the previous one, as presented in Figure 4. This functionality will ensure that the tester will read and execute each test step. Also, the time spent in each step is properly recorded (problem P5). For pausing the chronometer, the tester should push the pause button and push the play button later to resume the time counting (problem P2).

ManualTEST counts the time based on three stages: Setup, Execution and Debug. The Setup stage is relative to the execution of steps required to build the test preconditions. The Execution stage is relative to the execution of the test procedure (execution of test steps and verification of expected results). Finally, the Debug stage is used for recording the time spent by the tester for confirming that a test failed. For instance, the tester may have to execute the test again to confirm the test failure or s/he may have to read the specification to confirm that the problem is a defect in the application and not a problem in the test specification. This approach is interesting when the study analyzes only one of these stages (problem P5). The total execution time is also presented by the tool.

By default, the initial test execution stage considered by the tool is Setup. To change the stage, participants of the study can use the Setup, Execution or Debug buttons provided in the Test Execution Controller view. However, the tool automatically detects when the cursor is on the "Test procedure:" line and automatically consider the change from test setup to procedure execution, improving data collection accuracy. The test setup time, procedure execution time, debug time and total test execution time are presented to the tester in the Test Execution Controller view.

The field observations can be used to describe any problem occurred during the execution of the test. To finish the execution of a test case, the tester should press in button Passed, Failed or Blocked to indicate the correct test result. The Cancel button will discard the data collected for the current test execution.

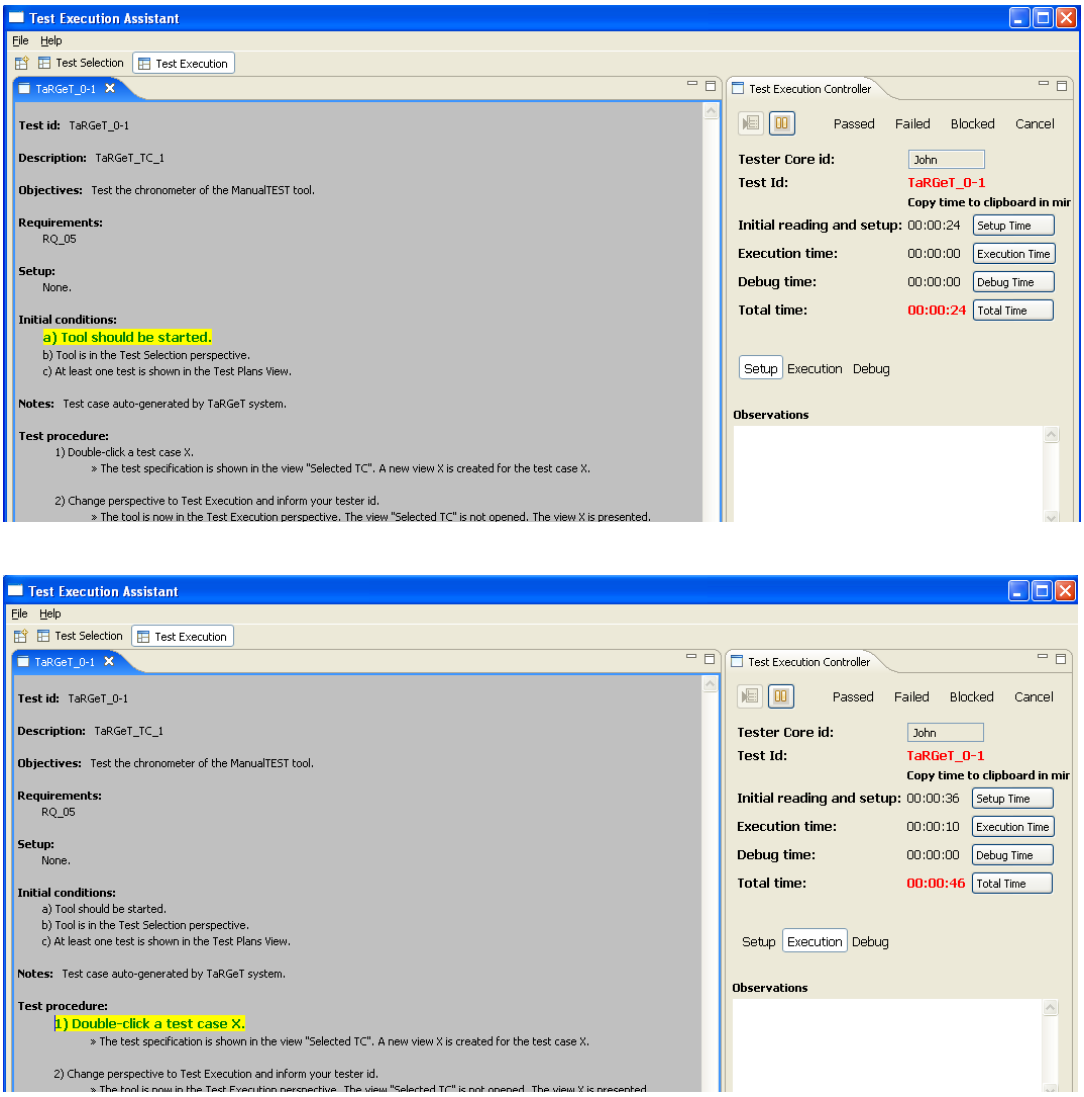


Figure 4. ManualTEST at different moments: test step under execution is highlighted and time is automatically counted as setup time or procedure execution time.

3.3. Collected Data

As test execution is finished, ManualTEST stores the test result, execution times and other related information in two different spreadsheet files. The first one is presented in Figure 5, which is basically the general information recorded for each executed test.

	A	B	C	D	E	F	G	H	I	J
1	Date	Time	Tester	Test	Result	Setup	Execution	Debug	Total time	
2	10/6/2008	19:47:44	John	TaRGeT_0-1	PASSED	00:00:29	00:00:51	00:00:00	00:01:20	

Figure 5. Test result for a single test case.

The second spreadsheet file contains detailed information about the test execution. As presented in Figure 6, ManualTEST stores the time spent to execute each step of the test specification. This characteristic is very important to investigate the presence of aberrant values in outlier analyses (problems 3 and 4) and to identify sources of variations on test execution time.

K	L
Step	Time
Description: TaRGeT_TC_1	00:00:03
Objectives: Test the chronometer of the ManualTEST tool.	00:00:00
Description: TaRGeT_TC_1	00:00:00
Objectives: Test the chronometer of the ManualTEST tool.	00:00:04
a) Tool should be started.	00:00:06
b) Tool is in the Test Selection perspective.	00:00:06
c) At least one test is shown in the Test Plans View.	00:00:09
Notes: Test case auto-generated by TaRGeT system.	00:00:01
Test procedure:	00:00:00
1) Double-click a test case X.	00:00:04
» The test specification is shown in the view "Selected TC". A new view X is created for the test case X.	00:00:04
2) Change perspective to Test Execution and inform your tester id.	00:00:04
» The tool is now in the Test Execution perspective. The view "Selected TC" is not opened. The view X is presented.	00:00:06
3) Click on button Play.	00:00:03
» The chronometer start counting time.	00:00:06
4) Click on button Pause.	00:00:04
» The chronometer stop counting time.	00:00:03
5) Click on button Play.	00:00:04
» The chronometer resume the counting of time.	00:00:03
6) Click on button Cancel.	00:00:03
» The counted time is discarded.	00:00:07

Figure 6. Detailed test result includes time spent in each test step.

4. Advantages and Current Limitations of ManualTEST

We used ManualTEST in several studies carried out by researchers from a research group in testing [Torres et al. 2007]. These studies involved manual test execution and some of them are overviewed next.

- A case study and a controlled experiment designed to compare the time reduction for manual test setup provided by different test case prioritization techniques.
- A case study to compare functional tests generated manually or automatically by a model-based testing tool [Nogueira et al. 2007] with respect to their manual test execution time.
- Controlled experiments designed to analyze the relationship of different test size measures with manual test execution time.
- Controlled experiments designed to investigate the effect of cost drivers for manual test execution.
- Case studies to verify if code instrumentation has impact on manual test execution time when considering mobile phone applications.

These studies were run in industrial settings and most of them were carried out by researchers different from the authors of the tool. The researchers were asked to write the benefits and limitations of ManualTEST observed during the execution of their empirical studies. We received several feedbacks showing the advantages and limitations of ManualTEST. These feedbacks are summarized next.

- Automated test execution time collection with the click of few buttons.
- The integration of the chronometer and test specifications increased productivity and reduced the occurrence of problems (forgetting to start or pause the time, etc.). Also, this integration make easy to the researcher to monitor more than one tester at the same time.
- Detailed log files included all information required for data analysis. Due to the high level of detail, the effect of some confounding factors were detected and outliers were treated properly.
- With the detailed and automated data collection, it was also possible to study the impact of each test step on total execution time, including an analysis of the main sources of variation.
- The format of the log files was not easy to be understood by other researchers.
- There are possible improvements to implement and minor defects to correct in the tool, improving its usability.
- The available version of ManualTEST did not guide the tester through the correct sequence of tests to be executed, requiring a careful attention of testers and researchers. The implementation of this functionality can improve the benefits of the tool.
- After some interruptions, some testers forgot to resume the chronometer (push the the play button), requiring the intervention of the researcher that was monitoring the test execution. The tool can be improved by having warning messages blinking while the chronometer still paused.
- To write into the field Observations, the time should be paused by the tester. It would be interesting if the tool could do that automatically.

As we can see, ManualTEST can be used to avoid several problems during the execution of empirical studies related to manual test execution. However, its benefits depend on the correct use by the testers. For instance, the tester may still forget to push the play and pause buttons. As suggested in the feedbacks, some improvements in the tool should be done to avoid the reported problems.

5. Related Work

Several testing tools are available in the marketing. For manual test authoring and execution, there is a tool that can be found in [IBM Rational Software 2008]. This commercial tool provides functionalities to execute and mark some steps of the test (comparisons and other verifications) as they are executed, as well as the storage of test results and test execution time. For collecting data in empirical studies, our tool presents better benefits due to characteristics such as the more detailed data collection and the highlighting of the test step being executed.

Some other researchers also have developed data collection tools for supporting their empirical studies. In [Karahasanovic et al. 2001], the authors reported a tool under

development for collecting data during software engineering experiments. They were not interested in execution time, but in data concerning subjects, interactions between subjects and technology and changes in engineering artefacts. In [Karahasanoviæet al. 2005], the authors proposed an unobtrusive method of collecting feedback from subjects during an experiment. They developed a tool to collect the feedbacks from experimental subjects and they identified several benefits of using the tool during the execution of four experiments, such as its use for validating the data obtained from other sources, checking process conformance and identifying problems with the experiments.

In [Arisholm et al. 2002], the authors developed a web-based support environment for planning and running software engineering experiments. One of its functionality is the collection of the experiment results, such as the answers of web questionnaires. The authors reported common interruptions (phone calls, lunch break, etc.) that occurred during the experiments run in real environments. In their approach, the subjects had to report the nature and time span of the interruptions. In our approach, the subject has to pause and resume the chronometer.

6. Conclusions and Future Work

This paper presented the main functionalities of ManualTEST, a tool developed for improving the collection of manual test execution data. This tool not only helps to collect data accurately, but it also provides information in different levels of detail, supporting the identification of problems in the collected data and analyses of outliers, sources of variability, etc. These benefits were observed during the execution of several case studies and controlled experiments in two real test sites. We intend to evolve the tool in order to overcome the current limitations and include the suggestions received during the empirical studies.

Despite of the use of ManualTEST for empirical studies, the tool can also be used to support the manual test execution activities. Its use can help, for instance, to have more accurate historical data in industrial settings. Although our tool is structured for experiments related to manual test execution, we believe that it can be extended (or similar ones can be developed) to support data collection for other types of manual activities that can have problems similar to those reported here. Finally, the benefits observed during the use of ManualTEST justified the cost to develop the tool, which took approximatelly two months of work (partial time) of one experienced developer.

7. Acknowledgments

We would like to thank all anonymous reviewers for their valuable comments that were used to improve this paper. The first author is partially supported by Motorola, grant BCT-0021-1.03/05, through the Motorola Brazil Test Center Research Project. The second author is partially supported by CNPq, grant 306196/2004-2. We would like to thank Lucas Lima, Juliano Iyoda and the other researchers from the BTC Research Project for their feedbacks about using ManualTEST in their empirical studies.

References

Aranha, E. and Borba, P. (2007). An estimation model for test execution effort. In *1st Intl. Symp. on Empirical Software Engineering and Measurement (ESEM 2007)*, pages 107–116, Madrid, Spain.

Arisholm, E., Sj, D. I. K., Carelius, G. J., and Lindsj, Y. (2002). A web-based support environment for software engineering experiments. *Nordic J. of Computing*, 9(3):231–247.

Broekman, B. and Notenboom, E. (2002). *Testing Embedded Software*. Addison-Wesley.

IBM Rational Software (2008). Rational Manual Tester. http://www.ibm.com/software/awdtools/tester/manual.

Jorgensen, P. (2002). *Software Testing, A Craftsmans Approach*. CRC Press, second edition.

Karahasanoviæ, A., Anda, B., Arisholm, E., Hove, S. E., J, M., Sj, D. I., and Welland, R. (2005). Collecting feedback during software engineering experiments. *Empirical Software Engineering*, 10(2):113–147.

Karahasanovic, A., Sjøberg, D. I. K., and Jørgensen, M. (2001). Data collection in software engineering experiments. In , editor, *Managing Information Technology in a Global Economy, Information Resources Management Association International Conference IRMA 2001, Software Engineering Track*, pages 1027–1028, Toronto, Ontario Canada. Idea Group Publishing.

Kitchenham, B. A., Pfleeger, S. L., Pickard, L. M., Jones, P. W., Hoaglin, D. C., Emam, K. E., and Rosenberg, J. (2002). Preliminary guidelines for empirical research in software engineering. *IEEE Trans. Softw. Eng.*, 28(8):721–734.

Lima, L., Iyoda, J., and Sampaio, A. (2008). A permutation technique for test case prioritization in a black-box environment. In *2nd Brazilian Workshop on Systematic and Automated Software Testing*, Campinas-SP, Brazil. To appear.

McAffer, J. and Lemieux, J.-M. (2005). *Eclipse Rich Client Platform: Designing, Coding, and Packaging Java(TM) Applications*. Addison-Wesley Professional.

Nogueira, S., Cartaxo, E., Torres, D., Aranha, E., and Marques, R. (2007). Model based test generation: A case study. In *1st Brazilian Workshop on Systematic and Automated Software Testing (SAST 2007)*.

Seaman, C. B. (2008). *Guide to Advanced Empirical Software Engineering*, chapter Qualitative Methods, pages 35–62. Springer London.

Selby, R. W. (2007). Data collection, analysis, and sharing strategies for enabling software measurement and model building. In *Empirical Software Engineering Issues. Critical Assessment and Future Directions, International Workshop*, volume 4336 of *Lecture Notes in Computer Science*, pages 70–76, Dagstuhl Castle, Germany. Springer Berlin / Heidelberg.

Shull, F., Singer, J., and Sjϕberg, D. I., editors (2008). *Guide to Advanced Empirical Software Engineering*. Springer London.

Singer, J., Sim, S. E., and Lethbridge, T. C. (2008). *Guide to Advanced Empirical Software Engineering*, chapter Software Engineering Data Collection for Field Studies, pages 35–62. Springer London.

Torres, D., Nogueira, S., Cartaxo, E., Aranha, E., Borba, P., Barros, F., Machado, P., Sampaio, A., and Mota, A. (2007). Brazil test center research group. In *1st Brazilian Workshop on Systematic and Automated Software Testing (SAST 2007)*.