Economic Perspectives in Test Automation: Balancing Automated and Manual Testing with Opportunity Cost

Rudolf Ramler and Klaus Wolfmaier
Software Competence Center Hagenberg GmbH
Hauptstrasse 99
A-4232 Hagenberg, Austria
+43 7236 3343-872

{rudolf.ramler | klaus.wolfmaier}@scch.at

ABSTRACT

Testing is a major cost factor in software development. Test automation has been proposed as one solution to reduce these costs. Test automation tools promise to increase the number of tests they run and the frequency at which they run them. So why not automate every test? In this paper we discuss the question "When should a test be automated?" and the trade-off between automated and manual testing. We reveal problems in the overly simplistic cost models commonly used to make decisions about automating testing. We introduce an alternative model based on opportunity cost and present influencing factors on the decision of whether or not to invest in test automation. Our aim is to stimulate discussion about these factors as well as their influence on the benefits and costs of automated testing in order to support researchers and practitioners reflecting on proposed automation approaches.

Categories and Subject Descriptors

D.2.9 [**Software Engineering**]: Testing and Debugging. D.2.9 [**Software Engineering**]: Management - *Software quality assurance (SQA)*.

General Terms

Management, Economics, Verification.

Keywords

Manual testing, automated testing, testing economics, benefits and costs of automation.

1. INTRODUCTION

Quality assurance measures, such as testing, are a major cost factor in software development projects. Studies have shown that testing accounts for 50 percent and more of total project costs. At the same time, growing international competition, shrinking schedules and budgets, and a fast-paced development process all put additional pressure on software testing.

Test automation has been proposed as one solution to reduce testing costs [9]. Furthermore, test tool vendors advertise: "Automating test script execution enables firms to increase the number of tests they run and the frequency at which they run them by many orders of magnitude. So why not automate every test?"

However, practitioners frequently report disastrous failures in the attempt to reduce costs by automating software tests, particularly at the level of system testing. Similarly, the authors have encountered several cases of failed endeavors of test automation when providing consulting and support to industrial projects. Our experience is supported by narrative evidence of practitioners and researchers on test automation (e.g., [18], [19]).

In most cases, reasons for failed projects include a gross underestimation of the effort required to develop and maintain automated tests. Managers and engineers are often surprised about the substantial investment of money and time necessary to automate tests in order to realize the promised benefits. As [12] states, "Like all testing activities, behind the decision to automate some tests is a cost and benefit analysis. If you get the analysis wrong, you'll allocate your resources inappropriately." So for any significant project, investments to test automation have to be analyzed primarily from an economic perspective, taking into account costs and benefits as well as the potential return on investment.

The objective of this paper is to discuss the question "When should a test be automated?" and the trade-off between automated and manual testing. Therefore, this paper is structured as follows: In Section 2 we summarize a common, simplistic approach and the involved problems of modeling the decision of automating testing. In Section 3 we describe the concept of opportunity cost in the context of test automation. An alternative model applying opportunity costs to balance automated and manual testing is presented in Section 4. Section 5 elaborates on additional influencing factors for test automation not yet considered in the proposed model. Finally, Section 6 discusses the applicability of the model and sums up the work presented in this paper.

2. A CRITIQUE ON OVERLY SIMPLISITC COST MODELS FOR AUTOMATED SOFTWARE TESTING

Accurate estimates of the return on investment of test automation require the analysis of costs and benefits involved. However, since the benefits of test automation are particularly hard to quantify, many estimates conducted in industrial projects are limited to considerations of cost only. In many cases the investigated costs include: the costs for the testing tool or framework, the labor costs associated with automating the tests, and the labor costs associated with maintaining the automated tests. These costs can be divided into fixed and variable costs. Fixed costs are the upfront costs involved in test automation. Variable costs increase with the number of automated test executions.

In [7], a case study originally published by Linz and Daigl [16] is presented, which details the costs for test automation as follows:

V := Expenditure for test specification and implementation

D := Expenditure for single test execution

Accordingly, the costs for a single automated test (A_a) can be calculated as:

$A_a := V_a + n * D_a$

whereby V_a is the expenditure for specifying and automating the test case, D_a is the expenditure for executing the test case one time, and n is the number of automated test executions.

Following this model, in order to calculate the break-even point for test automation, the cost for manual test execution of a single test case (A_m) is calculated similarly as

$A_m := V_m + n * D_m$

whereby V_m is the expenditure for specifying the test case, D_m is the expenditure for executing the test case and n is the number of manual test executions.

The break-even point for test automation can then be calculated by comparing the cost for automated testing (A_a) to the cost of manual testing (A_m) as:

$E(n) := A_a / A_m = (V_a + n * D_a) / (V_m + n * D_m)$

According to this model, the benefit of test automation seems clear: “From an economic standpoint, it makes sense to automate a given test only when the cost of automation is less than the cost of manually executing the test the same number of times that the automated test script would be executed over its lifetime.” [24]

Figure 1 depicts this interrelation. The x-axis shows the number of test runs, while the y-axis shows the cost incurred in testing. The two curves illustrate how the costs increase with every test run. While the curve for manual testing costs is steeply rising, automated test execution costs increase only moderately. However, automated testing requires a much higher initial investment than manual test execution does. According to this model, the break-even point for test automation is reached at the intersection point of the two curves.

This “universal formula” for test automation costs has been frequently cited in software testing literature (e.g. [7], [8], [15]) and studies (e.g., [24], [16]) to argue in favor for test automation.

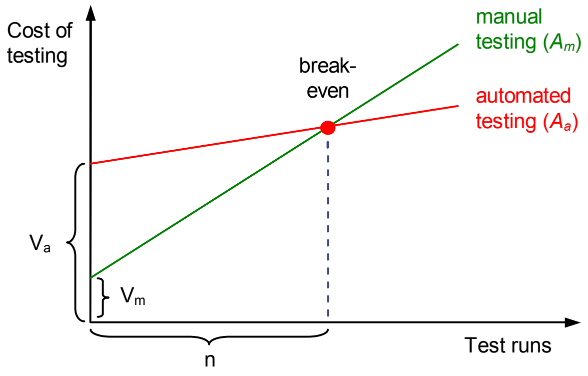

Figure 1: Break-even point for automated testing

Depending on the author, the quoted number of test runs required to reach the break-even point varies from 2 to 20.

The logic of this formula is appealing and – in a narrow context – correct. “As a simple approximation of costs, these formulas are fair enough. They capture the common observation that automated testing typically has higher upfront costs while providing reduced execution costs.” [12] For estimating the investment in test automation, however, it is flawed for the following reasons:

- *Only costs are analyzed* – The underlying model compares the costs incurred in testing but excludes the benefits. Costs and benefits are both required for an accurate analysis, especially when the analyzed alternatives have different outcomes. This is true for test automation, since manual testing and automated testing follow different approaches and pursue different objectives (e.g., exploring new functionality versus regression testing of existing functionality).
- *Manual testing and automated testing are incomparable* – Bach [2] argues that “… hand testing and automated testing are really two different processes, rather than two different ways to execute the same process. Their dynamics are different, and the bugs they tend to reveal are different. Therefore, direct comparison of them in terms of dollar cost or number of bugs found is meaningless.”
- *All test cases and test executions are considered equally important* – Boehm criticizes in his agenda on value-based software engineering [4]: “Much of current software engineering practice and research is done in a value-neutral setting, in which every requirement, use case, object, test case, and defect is equally important.” In a real-world project, however, different test cases and different test executions have different priorities based on their probability to detect a defect and on the impact which a potential defect has on the system under test.
- *Project context is not considered* – The decision about whether or not to automate testing is restricted to a single, independent test case. Nevertheless, the investment decision has to be made in context of the particular project situation, which involves the total set of test cases planned and the budget and resources allocated for testing.
- *Additional cost factors are missing* – A vast number of additional factors influencing costs and benefits are not con-

sidered in the overly simplistic model [11]. Examples are costs of abandoning automated tests after changes in the functionality, costs incurred by the increased risk of false positives, or total cost of ownership of testing tools including training and adapting workflows.

3. OPPORTUNITY COST IN TEST AUTOMATION

In this section we present a fictitious example to illustrate the problems listed in the previous section. Please note that this example simplifies a complex model to highlight and clarify some basic ideas. We discuss and expand the model in sections 4 and 5 where we add further details and propose influencing factors typically found in real-world projects.

The example describes a small system under test. The effort for running a test manually is assumed to be 0.25 hours on average. For the sake of simplicity, we assume no initial costs for specification and preparation. Automating a test should cost 1 hour on average, including the expenditures for adapting and maintaining the automated tests upon changes. Therefore, in our example, running a test automatically can be done without any further effort once it has been automated. According to the model presented in the previous section, the break-even point for a single test is reached when the test case has been run four times (cp. Figure 2).

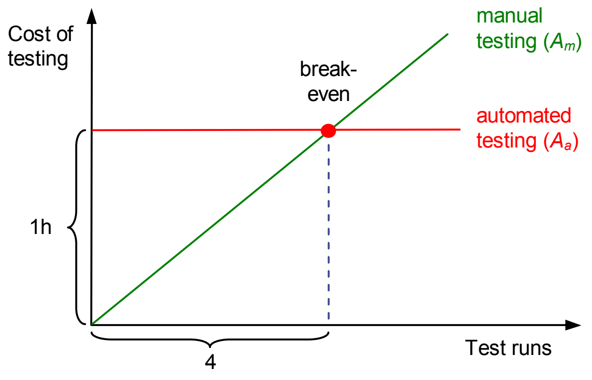

Figure 2: Break-even point for a single test case

Furthermore, for our example let us assume that 100 test cases are required to accomplish 100 percent requirements coverage. Thus it takes 25 hours of manual testing or 100 hours of automated testing to achieve full coverage. Comparing these figures, the time necessary to automate all test cases is sufficient to execute all test cases four times manually.

If we further assume that the project follows an iterative (e.g. [14]) or agile (e.g. [10]) development approach, we may have to test several consecutive releases. To keep the example simple, we assume that there are 8 releases to be tested and each release requires the same test cases to be run. Consequently, completely testing all 8 releases requires 200 hours of manual testing (8 complete test runs of 25 hours each) or 100 hours to automate the tests (and running these tests "infinitely" often without additional costs).

Taken from the authors' experience, the average time budget available for testing in many industrial projects is typically far less – about 75 percent at most – than the initially estimated test effort. In our example we therefore assume a budget of 75 hours for testing. Of course, one could argue that a complete test is not possible under these limitations. Yet many real-world projects have to cope with similar restrictions; fierce time-to-market constraints, strict deadlines, and a limited budget are some of the typical reasons. These projects can only survive the challenge of producing tested quality products by combining and balancing automated and manual testing.

Testing in the example project with a budget of 75 hours would neither allow to completely test all releases manually nor to completely automate all test cases. A trade-off between completely testing only some of the releases and automating only a part of the test cases is required. In economics, this trade-off is known as the "production possibilities frontier" [17]. Figure 3 shows the combinations of automated and manual test cases that testing can possibly accomplish, given the available budget and the choice between automated and manual testing. Any combination on or inside the frontier is possible. Points outside the frontier are not feasible because of the restricted budget. Efficient testing will choose only points on rather than inside the production possibilities frontier to make best use of the scarce budget available.

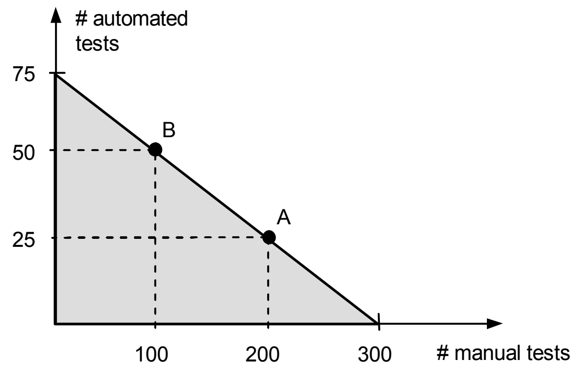

Figure 3: Production possibilities frontier for an exemplary test budget of 75 hours

The production possibilities frontier shows the trade-off that testing faces. Once the efficient points on the frontier have been reached, the only way of getting more automated test cases is to reduce manual testing. Consequently Marick [18] raises the following question: "If I automate this test, what manual tests will I lose?" When moving from point A to point B, for instance, more test cases are automated but at the expense of fewer manual test executions. In this sense, the production possibilities frontier shows the opportunity cost of test automation as measured in terms of manual test executions. In order to move from point A to point B, 100 manual test executions have to be abandoned. In other words, automating one test case incurs opportunity costs of 4 manual test executions.

4. A COST MODEL BASED ON OPPORTUNITY COST

Building on the example from the previous section, we propose an alternative cost model drawing from linear optimization. The model uses the concept of opportunity cost to balance automated and manual testing. The opportunity cost incurred in automating a test case is estimated on basis of the lost benefit of not being able to run alternative manual test cases. Hence, in contrast to the simplified model presented in Section 2, which focuses on a single test case, our model takes all potential test cases of a project into consideration. Henceforth, it optimizes the investment in automated testing in a given project context by maximizing the benefit of testing rather than by minimizing the costs of testing.

4.1 Fixed Budget

First of all, the restriction of a fixed budget has to be introduced to our model. This restriction corresponds to the production possibilities frontier described in the previous section.

R1: $n_a * V_a + n_m * D_m \leq B$

n_a := *number of automated test cases*
n_m := *number of manual test executions*
V_a := *expenditure for test automation*
D_m := *expenditure for a manual test execution*
B := *fixed budget*

Note that this restriction does not include any fixed expenditures (e.g., test case design and preparation) manual testing. Furthermore, with the intention of keeping the model simple, we assume that the effort for running an automated test case is zero or negligibly low for the present. This and other influence factors (e.g., the effort for maintaining and adapting automated tests) will be discussed in the next section.

This simplification, however, reveals an important difference between automated and manual testing. While in automated testing the costs are mainly influenced by the number of test cases (n_a), manual testing costs are determined by the number of test executions (n_m). Thus, in manual testing, it does not make a difference whether we execute the same test twice or whether we run two different tests. This is consistent with manual testing in practice – each manual test execution usually runs a variation of the same test case [2].

4.2 Benefits and Objectives of Automated and Manual Testing

Second, in order to compare two alternatives based on opportunity costs, we have to valuate the benefit of each alternative, i.e., automated test case or manual test execution. The benefit of executing a test case is usually determined by the information this test case provides. The typical information is the indication of a defect. Still, there are additional information objectives for a test case (e.g., to assess the conformance to the specification). All information objectives are relevant to support informed decision-making and risk mitigation. A comprehensive discussion about what factors constitute a good test case is given in [13].

In our model we simply assume that the benefit of a test case is equivalent to its potential to contribute to risk mitigation, whereby automated and manual testing may address different types of risk. One common distinction is based on the finding that automated testing best addresses the regression risk, i.e. defects in modified but previously working functionality are quickly detected, while manual testing is suitable for exploring new ways in how to break functionality. Accordingly, the objectives of automated and manual testing are usually different. Therefore we estimate the contribution to risk mitigation for both automated testing (R_a) and manual testing (R_m) separately.

We suggest approximating the contribution to risk mitigation by calculating the risk exposure [3] of the tested requirement or system component. In general the risk exposure is defined as the probability of loss times the size of loss and it varies between requirements and system components. For our general model, we suppose that the contribution to risk mitigation for n test cases is calculated by the function $R_a(n)$ or $R_m(n)$ respectively. Note: Since the contribution to risk mitigation varies between test cases, ordering the test cases according to their contribution makes sure that the most beneficial test cases are run first [23].

Although a number n equal to the total number of test cases would be preferable, the restricted budget does not allow for complete testing. However, test and project planning have to set a minimum level of testing – sometimes referred to as "quality bar" – to make sure the information provided by testing is a representative source for risk mitigation and decision-making. We consider this minimum testing requirements for automated and manual testing based on n_a and n_m as additional restrictions in the optimization model.

R2: $n_a \geq min_a$

R3: $n_m \geq min_m$

min_a := *minimum number of automated test cases*
min_m := *minimum number of manual test executions*

This approach is in line with requirements- and risk-based testing (see e.g. [22], [1]) and is supported by commercial test management tools [6]. A simple technique to estimate the risk exposure based on the Pareto principle is described in [20] and will be used in the following example.

4.3 Maximizing the Benefit

Third, to maximize the overall benefit yielded by testing, the following target function has to be added to the model.

T: $R_a(n_a) + R_m(n_m) \rightarrow max$

Maximizing the target function ensures that the combination of automated and manual testing will result in an optimal point on the production possibilities frontier defined by restriction *R1*. Thus, it makes sure the available budget is entirely and optimally utilized.

4.4 Example

To illustrate our approach we extend the example used in Section 3. For this example the restriction *R1* is defined as follows.

R1: $n_a * 1 + n_m * 0.25 \leq 75$

To estimate benefit of automated testing based on the risk exposure of the tested object, we refer to the findings published by Boehm and Basili [5]: "Studies from different environments over many years have shown, with amazing consistency, that between 60 and 90 percent of the defects arise from 20 percent of the modules, with a median of about 80 percent. With equal consistency, nearly all defects cluster in about half the modules produced." Accordingly we categorize and prioritize the test cases into 20 percent highly beneficial, 30 percent medium beneficial, and 50 percent low beneficial and model following alternative restrictions to be used in alternative scenarios.

R2.1: $n_a \geq 20$

R2.2: $n_a \geq 50$

To estimate the benefit of manual testing we propose, for this example, to maximize the test coverage. Thus, we assume an evenly distributed risk exposure over all test cases, but we calculate the benefit of manual testing based on the number of completely tested releases. Accordingly we categorize and prioritize the test executions into one and two or more completely tested releases. We model following alternative restrictions for alternative scenarios.

R3.1: $n_m \geq 100$

R3.2: $n_m \geq 200$

Based on this example we illustrate three possible scenarios in balancing automated and manual testing. Figures 4a, 4b and 4c depict the example scenarios graphically.

- **Scenario A** – The testing objectives in this scenario are, on the one hand, to test at least one release completely and, on the other hand, to test the most critical 50 percent of the system for all releases. These objectives correspond to the restrictions *R3.1* and R2.2 in our example model. As shown in Figure 4a the optimal solution is point *S1* (n_a = 50, n_m = 100) on the production possibilities frontier defined by *R1*. Thus, the 50 test cases referring to the most critical 50 percent of the system should be automated and all test cases should be run manually once.
  Note: In this example we assume that – as discussed in Section 2 – running the same test automatically and manually may detect different types of failures. Therefore, an automated test case should be executed manually too.
- **Scenario B** – The testing objectives in this scenario are, on the one hand, to test at least one release completely and, on the other hand, to test the most critical 20 percent of the system for all releases. These objectives correspond to the restrictions *R3.1* and *R2.1* in our example model. As shown in Figure 4b any point within the shaded area fulfills these restrictions. The target function, however, will make sure that the optimal solution will be a point between *S1* (n_a = 50, n_m = 100) and *S2* (n_a = 20, n_m = 220) on the production possibilities frontier defined by *R1*.
  Note: While all points on *R1* between the *S1* and *S2* satisfy the objectives of this scenario, the point representing the optimal solution depends on the definition of the contribution to risk mitigation of automated and manual testing, $R_a(n_a)$ and $R_m(n_m)$. While the optimization model may be solved mathematically once these functions have been defined, the effort of doing so has to be weighted against the pragmatic approach of approximating the optimal solution by stepwise refining the restrictions to narrow the solution space.
- **Scenario C** – The testing objectives in this scenario are, on the one hand, to test at least two releases completely and, on the other hand, to test the most critical 50 percent of the system for all releases. These objectives correspond to the restrictions *R3.2* and *R2.2* in our example model. As shown in Figure 4c a solution that satisfies both restrictions cannot be found. This scenario depicts an unsatisfactory situation which falls short of meeting the testing objectives and, as a result, the quality target.
  Note: A situation like this is a strong indicator for an inappropriate test budget, and renegotiation of the budget and objectives of testing is required before an investment in automated and manual testing is made.

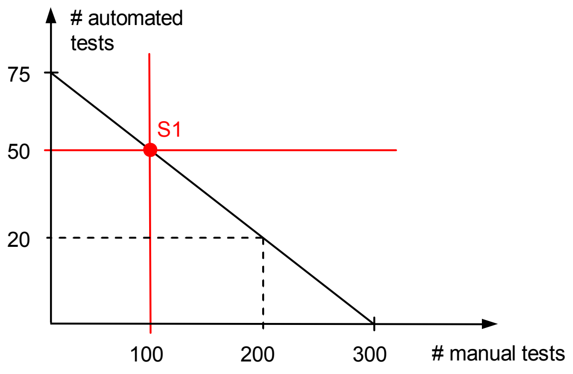

Figure 4a: Example scenario A

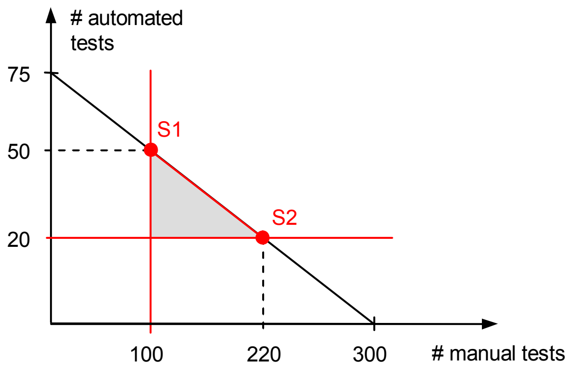

Figure 4b: Example scenario B

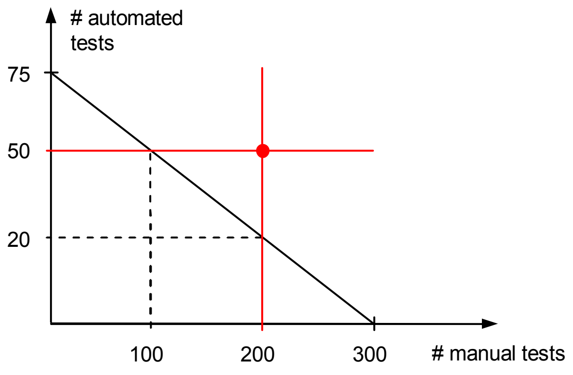

Figure 4c: Example scenario C

5. ADDITIONAL INFLUENCE FACTORS

The model presented in the previous section has been simplified to facilitate comprehension and discussion. Thus, in order to support decision-making in realistic real-world scenarios, the model has to be extended by including additional influencing factors. In this section we describe typical factors distilled from literature and the authors' experience in test automation.

- *Tests that rely on automation* – The production possibilities frontier shown in Figure 3 usually is not linear but bowed outward. The curve flattens when most of the budget is used to either do automated or manual testing, because some test cases are more amenable to be automated and others are best run manually. Consider for example performance and stress testing with 100 concurrent transactions accessing the same data. These types of testing require automation as it is impossible (or only at an unreasonable expense) to simulate such a scenario by hand. In contrast, manual testing is preferable to assess "intangible" requirements such as the aesthetic appearance and attractiveness of a user interface or Web site.
- *Productivity changes over time* – The production possibilities frontier shows the trade-off between automated and manual testing at a given time, but this trade-off can change as productivity changes over time. This is especially the case in projects introducing test automation. The authors have observed cases where the productivity of automating a test case had approximately doubled after a reusable test library containing basic routines had been developed and the engineers had advanced on the learning-curve.
- *Growing test effort in iterative development* – Iterative development approaches do not produce complete systems with every release. The first releases are typically used to construct the basic building blocks and the core functionality of the system. Less valuable features are deferred to later releases. Therefore, in the beginning, the effort for testing may be low and narrowly focused. As the system grows the effort for testing increases and has to be expanded to a wider range of requirements and system components.
- *Maintenance costs for automated tests* – Iterative development not only adds new functionality with every release but also changes existing functionality. Kaner [12] points out that "testers invest in regression automation with the expectation that the investment will pay off in due time. Sadly, they often find that the tests stop working sooner than expected. The tests are out of sync with the product. They demand repair. They're no longer helping finding bugs. Testers reasonably respond by updating the tests. But this can be harder than expected. Many automators have found themselves spending too much time diagnosing test failures and repairing decayed tests."
- *Early/late return on investment* – "An immediate impact of many automation projects is to delay testing and delay bug finding. This concern is also an argument for using upfront automation techniques that let you prepare automated tests before the software is ready for testing. For example, a data-driven strategy might let you define test inputs early. In this case, test automation might help you accelerate testing and bug finding." [12]
- *Defect detection capability* – The capability of a test to reveal a defect if there actually is a defect may vary. Not every test case has the same power to reveal a defect since not all defects will result in a failure or an observable behavior of the tested system, and not all test cases are able to indicate the occurrence of a failure [25]. This is particularly true for automated testing relying on partial oracles, for instance random "monkey" testing, driving the system under test with random input provoking major system failures such as crashes.

6. DISCUSSION AND CONCLUSION

In this paper we discussed cost models to support decision-making in the trade-off between automated and manual testing. We summarized typical problems and shortcomings of overly simplistic cost models for automated testing frequently found in literature and commonly applied in practice:

- Only costs are evaluated and benefits are ignored
- Incomparable aspects of manual testing and automated testing are compared
- All test cases and test executions are considered equally important
- The project context, especially the available budget for testing, is not taken into account
- Additional cost factors are missing in the analysis

We address these problems by introducing an alternative model using opportunity cost. The concept of opportunity cost allows us to include the benefit and, thus, to make the analysis more rational. Instead of minimizing the overall costs of testing, the benefits of testing are maximized. We suggest determining the benefit of a test case by estimating its contribution to risk mitigation for the tested requirement or system component. This approach is in line with requirements-based testing and risk-based testing [21]. Building on the potential benefit of a test case, which we consider to be different for automated or manual testing, allows a realistic comparison between these two alternatives. Furthermore, it avoids considering all test cases and test executions equally important. By representing the available budget for testing as production possibilities frontier according to economic theory, we include the specific project context and the resulting restrictions in our model.

We simplified the complex model to highlight and clarify some basic ideas and to facilitate comprehension and discussion. For the sake of simplicity, some typical factors influencing the trade-off between automated and manual testing have been omitted from the model as well as from the example in Section 4. Therefore the model, as described in Section 4, has to be extended to represent real-world scenarios more realistically. We have described influencing factors commonly found in literature or which have emerged in practice in Section 5 for further consideration.

On the one hand, adding further influencing factors to the model increases the usefulness for decision-making. On the other hand, however, additional influencing factors increase the model's complexity and limit the applicability in practice due to the high effort required for collecting the data and for solving the model. Identifying and incorporating the most important influencing factors is therefore crucial. Thus, future plans include identifying and adding further factors influencing decision-making to make the model more realistic. As a first step, the authors appreciate the opportunity to discuss the model and further influencing factors in the workshop. Our aim is to compare the underlying assumptions of the model to the automation approaches presented by the workshop participants to derive and discuss appropriate influencing factors.

At the same time, the model's preciseness and completeness has to be evaluated from a value-based perspective. With every additional influencing factor, adjustment parameter and restriction introduced to the model, its complexity increases and its practical value decreases. The balance between keeping the model simple but not too simple to draw valid conclusions has to be found. So far the model has proven a highly valuable tool to foster discussion about the opportunities and pitfalls of automated testing, to draw the attention on the importance of balancing automated and manual testing, and to support test planning by quickly sketching and evaluating alternative scenarios. In these cases the model was not at all intended to be implemented and solved.

7. ACKNOWLEDGEMENTS

The authors gratefully acknowledge support by the Austrian Government, the State of Upper Austria, and the Johannes Kepler University Linz in the framework of the K*plus* Competence Center Program.

8. REFERENCES

[1] Amland, S., *Risk Based Testing and Metrics.* 5th Int. Software Testing Analysis & Review Conference (EuroSTAR'99), Barcelona, Spain, Nov. 1999

[2] Bach, J., *Test Automation Snake Oil,* 14th International Conference and Exposition on Testing Computer Software, Washington, DC, 1999

[3] Boehm B.: *Software Risk Management: Principles and Practices*. IEEE Software, pp. 32-41, Jan. 1991

[4] Boehm, B., *Value-Based Software Engineering: Overview and Agenda*. In: Biffl S. et al.: Value-Based Software Engineering. Springer, 2005

[5] Boehm, B., Basili, V.R., *Software Defect Reduction Top 10 List*, IEEE Computer, pp. 135-137, January 2001

[6] Compuware QADirector, Advanced risk-based test management. Compuware 2006 (www.compuware.com/products/qacenter/qadirector.htm)

[7] Dustin, E. et. al., *Automated Software Testing*, Addison-Wesley, 1999

[8] Fewster, M., Graham, D., *Software Test Automation: Effective Use of Text Execution Tool*, Addison-Wesley, 1999

[9] Harrold, M.J.: *Testing: A Roadmap*. In: The Future of Software Engineering, ed by Finkelstein, A., 22th International Conference on Software Engineering, Limerick, Ireland, June 2000

[10] Highsmith, J., *Agile Software Development Ecosystems,* Addison-Wesley, 2002

[11] Hoffman, D., *Cost Benefits Analysis of Test Automation*, Software Testing Analysis & Review Conference (STAR East). Orlando, FL, May 1999

[12] Kaner, C., Bach, J., and Pettichord, B., *Lessons Learned in Software Testing*, Wiley, 2002

[13] Kaner, C., *What is a Good Test Case*, Software Testing Analysis & Review Conference (STAR East). May 2003

[14] Kruchten, P., *The Rational Unified Process: An Introduction, 3rd Ed.*, Addison-Wesley, 2003

[15] Link, J., *Unit Testing in Java: How Tests Drive the Code,* Morgan Kaufmann, 2003

[16] Linz, T., Daigl, M., *GUI Testing Made Painless.* Implementation and results of the ESSI Project Number 24306, 1998. In: Dustin, et. al., Automated Software Testing, Addison-Wesley, 1999, pp. 52

[17] Mankiw G.W., *Principles of Economics, 2nd Ed*. Hartcourt College Publishers, 1998

[18] Marick, B., *When Should a Test Be Automated*, Software Testing Analysis & Review Conference (STAR East). Orlando, FL, May 1999

[19] Persson, C., Yilmaztürk, N., *Establishment of Automated Regression Testing at ABB: Industiral Experience Report on Avoiding the Pitfalls*, 19th IEEE International Conference on Automated Software Engineering (ASE'04), Linz, Austria, 2004

[20] Ramler R., Weippl E., Winterer M., Schwinger W., and Altmann J., *A Quality-Driven Approach to Web-Testing.* 2nd Conf. on Web Engineering (ICWE'02), Santa Fe, Argentina, Sept. 2002

[21] Ramler R., Biffl S., Grünbacher P., *Value-based Management of Software Testing*. In: Biffl S. et al.: Value-Based Software Engineering. Springer, 2005

[22] Redmill, F., *Exploring risk-based testing and its implications.* Software Testing, Verification and Reliability. 14, 3-15 (2004)

[23] Rothermel, G., Elbaum, S., *Putting Your Best Tests Forward*. IEEE Software, 20(5), pp. 74-77, Sept/Oct 2003

[24] Schwaber, C., Gilpin, M., *Evaluating Automated Functional Testing Tools*, Forrester Research, February 2005

[25] Voas, J.M., Morell, L.J., and Miller, K.W., *Predicting where faults can hide from testing.* IEEE Software, 8(2), pp. 41-48, March 1991