Selecting the right Tests at the right Time

A Guideline for Test Prioritization

Version 1.1

Feedback of any kind is welcome! Contact us via Twitter @TestomatProject or using this **feedback** form: https://goo.gl/J5wnjm
Please take part in our **survey** on test prioritization! https://goo.gl/PCSpU6

CONTENT

Why Prioritize Your Tests Systematically?

The ideal situation for test is that you could test everything, but this is often not feasible for real-world systems.
Prioritization of what you are testing can mean:

- to create only tests for aspects of your software that are important to you
- to reorder an existing test suite to increase its efficiency during test execution
- to choose when to run which tests
- to select and run only those test cases of an existing test suite that are important right now
- to permanently remove test cases that are obsolescent
- to select manual or semi-manual tests for full automation

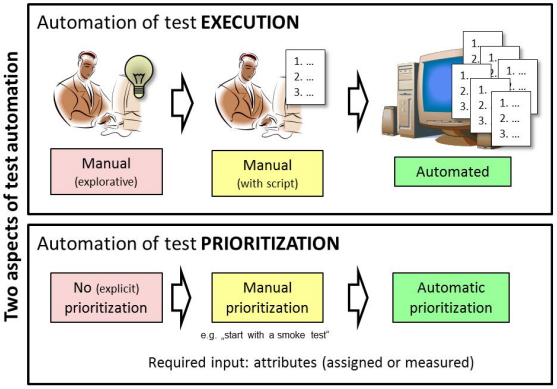
which allows you to...

- focus on what is important for you and your user or customer
- save time, effort, and cost
- save on hardware utilization and test environment
- save on energy and power

- create a test suite that gives you the best benefits
- state the reasons for your TC selections, required by standards, e.g. ISO/IEC 26262
- get an earlier feedback regarding faults in case of very large test suites

It is simply smart to utilize your test suite as efficiently and effective as possible.

The methods of test prioritization can be used and adapted to achieve various goals. By using test prioritization methods like test case selection, removal, or scheduling, you will be able to improve the efficiency of your testing process by reducing testing effort, time and cost. In the case of test case reordering, a prioritized and reordered test suite allows you to increase the rate of fault detection in an early phase of testing without compromising on test coverage. This is especially interesting for large test suites, where an early feedback regarding faults is often beneficial.



When to Prioritize Your Tests - in your Development & Test Process

Prioritization is both done in the long term - for a software systems life-cycle, but mostly done every day. This means that methods of test prioritization can be applied every time you run your tests manually or during your automated regression testing. Going one step backwards, it's also a good idea to think about prioritization when you are in the process of creating tests - to keep the size of your test suite and your testing costs at a minimum from the beginning. Generally speaking, the earlier you start to plan for test prioritization in your testing process, the easier it will be to actually implement and use prioritization methods. For example, if you want to integrate prioritization methods in your process of automated regression testing, it might be necessary to augment your test suite with metrics or attributes to utilize certain prioritization methods. Changing an

existing test suite to allow for efficient prioritization can be a lot more work than having your test model optimized for prioritization from the beginning.

So far, we only talked about when to start implementing test prioritization. Another important topic is when to actually use your implemented test prioritization methods. This depends on various factors, for one the cost of utilizing it. Another factor is the actual size of the test suite. Generally speaking again, using a prioritization method will always be beneficial as soon as a full execution of an existing test suite will either be too expensive regarding cost or time. For example, if a full execution of a test suite takes several days and an early feedback regarding faults is necessary, it is beneficial to only execute the important test cases which are determined by the prioritization. Alternatively, if the test suite is prioritized using adequate methods and metrics to increase the fault detection rate in the early testing process, the entire test suite can still be executed - but most of the faults will be reported in the first few hours of test execution (assuming that an efficient prioritization method has been chosen).

Prioritization Versus Development Methodologies

For software development different standard approaches exist in order to

achieve a certain quality. Depending on the project and its boundary conditions the waterfall approach, the Agile/Scrum approach, or the DevOps approach might be suiting. Regardless the chosen approach a prioritization might be useful.

Different Types of Prioritizations

In the beginning, you are generally faced with the prioritization issue of what quality attributes to test for. It is often our first priority to test the functionality or features that our customer demands. This early testing phase has its own requirements regarding the test process which are often different to those of later regression testing. Depending on where you are in your testing process (or which level of automation you have already reached), we can differentiate between three different types (or phases) of prioritization:

1. Initial test prioritization
2. Manual prioritization for regression testing
3. Automated prioritization for variable CI/CD testing

Each type has two major steps: The selection and the ordering of execution.

We define this first prioritization step as type 1:

Type 1: Initial Test Prioritization

The general question here is: What do I actually have to test? Meaning, we want to know what aspects we have to create test cases for; also called “defining the test scope”. This question can most of the time only be answered generally, because it really depends on what kind of software you are developing and what your customer requirements are.

There can be variety of factors that influence your decision - but in most of the cases, it should at least satisfy the stakeholders.

Example of stakeholders are:

- Customers
- Users of systems
- Company strategy
- Economic situation

Most other aspects can be summed up as TEST CRITERIA:

- Requirements/User Stories (if it was important to describe in a requirement - it is important to test that it fulfills the requirement)
- Non-functional aspects (e.g. using quality attributes from the ISO/IEC std 25010)

- Necessity of certification and standardization adherence (there could be some necessity test that must be executed)

We could also assume we already have a test suite existing. If it is a manual set of test cases, we will be forced to select carefully among which of these tests we have time to test again, in addition to the re-test necessary for tests that revealed faults. Such an existing test suite can be prioritized using the same prioritization metrics that are used for prioritization type 3 (see chapter: “Type 3: Prioritization for your variable CI/CD testing”). However, the goal of testing (increasing fault detection rate, decreasing risk etc.) might be a different one.

We define the following prioritization as type 2:

Type 2: Manual Prioritization For Regression Testing?

If you have automated your testing process - this choice would appear trivial: You simply test everything again. When you work e.g. agile with a continuous build and test manner, and have automated most of your tests, you are likely to encounter these problems quickly:

1. The time to run all tests is too long
2. The number of tests is too high (for the time given)

3. It is a waste of energy and resources to rerun every test
4. That currently run tests are not related or important for the moment

Then you need to select among your automated suite of tests to choose in which order you want to test them.

Select TC to Regression Test?

As mentioned, this assumes that you are working manually, otherwise you would go to Type 3.

Various prioritization criteria may be applied to the regression test suite. Test cases can be prioritized in terms of:

- random
- optimal
- total or additional statement coverage
- total or additional branch coverage
- failure rates
- fault exposing potential

of the test cases. Test case prioritization can help with detecting high risk faults earlier in the testing life cycle, improving the detection of regression errors related to code changes, enhancing the coverage of the code and making a system reliable. Test case prioritization thus helps in reducing the effort and therefore time and cost of testing [7].

Type 2 prioritization also covers the prioritization process when deciding which test cases are automated first. This first selection can be done by choosing test cases that:

- "cover" as many different aspect of the system under test as possible (code, model, system architecture, modules, functionality, requirements, non-functional aspects, etc.)
- need re-testing (that found a fault in the earlier version) and still has to be confirmed as "fault-free"[1] (note that this option does not necessarily mean you should automate for it)
- you think has a high probability of finding a fault (e.g error guessing)

We define the last step of prioritization as type 3:

Type 3: Automated Prioritization For Variable CI/CD Testing

Changing the software to correct faults or add new functionality can cause the existing functionality to regress,

[1] It's important to note that in software testing, tests can be used to show the presence of bugs, but never show their absence!

introducing new faults. To avoid such kind of defects, software can be retested after modification which is known as regression testing. Regression testing is known to be one of the most expensive parts of software maintenance and therefore, it is necessary to prioritize test cases in test suites. Several techniques are available for prioritizing the test cases to accelerate the rate of fault detection in regression testing. Some existing approaches rely on requirement coverage. These approaches consider prioritization as an unordered, independent and one-time model. They do not take into account the performance of test cases [6].

When choosing a regression test selection, a goal oriented approach is generally a good idea. Here, a test selection for regression testing can be chosen according to one or more of the following goals:

- Increase fault detection rate
- Decrease risk of faults
- Decrease testing time
- Decrease cost of testing
- Feature-based prioritization focused on testing of current development
- A mix of those described by an optimization problem: maximize test coverage for a limited available testing time

An automated prioritization of your regression testing could be based on one or several of the following prioritization metrics:

- Coverage (additional) based on
  - Configuration
  - Test Environment
  - System architecture, structure
  - Customer prioritization
  - Code Coverage in case of white box testing [1]
  - Model Coverage in case of black box testing
  - N-F Quality Attributes
- Time for execution of TC [8]
- Type of TC (Level, or specific quality attribute)
- Resources required/Availability of
- Test Environment (context) needed/Availability of
- History of TC
  - Latest 1- x testresults (pass/fail) [4]
  - How long ago a TC was last executed
  - When a TC was introduced (age)
- Code Churn (based on latest changes of the code)
  - Dependability analysis
  - Static Analysis
  - Build information
- Dependency in group, If verdict/results reacts “together” in

a group - only select one of group of “similar” TC

- Code complexity or model complexity
- Requirements
  - Requirement changes [2]
  - Requirement complexity [2]
- Risk-based metrics [3]
- Error-Probability-based metrics
  - Based on bayesian networks [5]
- Other constraints - e.g. goal to fulfill

Example of such selections criteria for different ordering or group’s are:

- Time/Duration to execute the Test suite overall (or a specific TC)
- “group”, e.g. Smoke tests, Functional tests, Non-functional tests

How Do You Implement Test Prioritization?

What do you do now?

There is a significant body of research around test prioritization. However, knowledge about its importance and concrete techniques and tools has not spread to all industries. Inside TESTOMAT project we are identifying techniques to apply prioritization to the testing process. Future publications will

outline lessons learned and additional concrete advice to help in industrial applications.

Please follow the TESTOMAT Project to get to know more about the next steps of test automation.

References:

Related Work

[1] S. Elbaum, A. Malishevsky and G. Rothermel, "Test Case Prioritization: A Family of Empirical Studies," in *IEEE Transactions on Software Engineering, Vol. 28, No. 2*, 2002.

[2] H. Srikanth, L. Williams and J. Osborne, "System Test Case Prioritization of New and Regression Test Cases," in *2005 International Symposium on Empirical Software Engineering*, Raleigh, NC, 2005.

[3] H. Stahlbaum, A. Metzger and K. Pohl, "An Automated Technique for Risk-based Test Case Generation and Prioritization," in *AST '08 Proceedings of the 3rd international workshop on Automation of software test*, Leipzig, 2008.

[4] C. T. Lin, C. D. Chen, C. S. Tsai and G. M. Kapfhammer, "History-based Test Case Prioritization with Software Version Awareness," in *18th International Conference on Engineering of Complex Computer Systems*, Singapore, 2013.

[5] S. Mirarab , L. Tahvildari , A prioritization approach for software test cases based on Bayesian networks, Fundam. Approaches Softw. Eng. Springer Berlin Heidelberg 4422 (2007) 276–290 .

[6] A. Khalilian, M.A. Azgomi, Y. Fazlalizadeh, "An improved method for test case prioritization by incorporating historical test case data," *Science of Computer Programming Vol. 78, (2012), pp.93–116.*

[7] A. Ansari, A. Khan, A. Khan, K. Mukadam, "Optimized Regression Test using Test Case Prioritization," *Procedia Computer Science, Vol. 79 (2016), pp. 152 – 160.*

[8] Kristen R. Walcott, Mary Lou Soffa, Gregory M. Kapfhammer, and Robert S. Roos. *Timeaware test suite prioritization. In Proceedings of the 2006 International Symposium on Software Testing and Analysis, ISSTA '06, pages 1--12, New York, NY, USA, 2006. ACM.*

Further Reading

[1] B. Jian, W.K. Chan, "Input-based adaptive randomized test case prioritization: A local beam search approach," *Journal of Systems and Software, 105, (2015), pp. 91-106.*

[2] S. Sharma, P. Gera, "Test Case Prioritization in Regression Testing using Various Metrics," *International Journal of Latest Trends in Engineering and Technology, Vol. 4 Issue 2, 2014.*

[3]R. Pradeepa, K. VimalaDevi, "Effectiveness of Testcase Prioritization using APFD Metric: Survey," *International Conference on Research Trends in Computer Technologies (ICRTCT - 2013).*

Abbreviations & Terminology

CI = Continuous Integration
TC = Test Case or Test Cases
SUT = System Under Test
N-F = Non-Functional (or Extra-functional) test (based on quality attributes listed other than Functional Suitability in ISO/IEC Standard 25010)

This booklet was produced by a research collaboration between the following partners:

Acknowledgements:
This booklet is produced by
EUREKA ITEA3 TESTOMAT PROJECT
The Next Level of Test Automation
Find out about us on the web:
https://www.testomatproject.eu/

Follow us on Twitter
@Testomatproject

The Testomat Project is sponsored by:

Disclaimer: The content of this booklet is true to the best of our current knowledge. The authors, publishers, participating partners of the project as well as the funding agencies disclaim any liability in connection to use of this information.

Since this document is only a first draft, we'd love to hear feedback from you! Contact us via Twitter or this feedback form: https://goo.gl/J5wnjm