ATAC

ITEA 2

INFORMATION TECHNOLOGY FOR EUROPEAN ADVANCEMENT

About selecting open source model-based testing tools

Project number:	ITEA 2 10037
Edited by:	Matti Vuori, TUT
Date:	2014/03/18
Document version no.:	*1.0*

Table of Contents

APPENDICES

- Test automation system quality model mind map
- Test automation system review checklist
- MBT tool characteristics checklist

1. Introduction

The state of model-based testing technology and the understanding about model-based testing (MBT) has reached such a level that many companies are beginning to seriously consider starting to use model-based testing.

For many, the next step is a practical one – how to select the tool that is the best one for them, among all the ones that are available?

At the same time, the processes for selection have changed a lot. In the ”previous world” selection of tools was often based on the vendors’ marketing pitches and presentations, leading to “corporation choosing”. But today, when most of the tools are open source and freely available to evaluate, the process is more flexible, more a bottom-up process. That gives the choosing more potential to really meet the needs of testing, but still there are many opportunities for making wrong choices.

This report will discuss some of the issues to consider and provides a simple checklist to aid in the evaluation of the tools.

2. MBT tool selection process in a nutshell

There is no generic MBT testing and there are no generic tools. The first thing to do is to understand the purpose of the tool: in which kind of testing it should be used. This leads to the understanding the basic requirements for the tool, and also to the understanding of usability requirements, because the tools need to fit the people who are planned to use it.

This already leads to a possibility to find a selection of tool candidate that can be assessed further.

A traditional systematic selection process will now continue with a rough level assessment of the tools from a broad perspective, taking into consideration issues such as the maturity and development of the tool – does it have a solid roadmap ahead or is it perhaps at the end of its lifespan, the support available for the tool, its reputation amongst the tester community etc.

This assessment will lead to dropping some candidates and should highlight some of the tools as very potential ones, which can be obtained (read: downloaded and installed) and tested in practice.

The testing can be done at two levels: first just trying out the tool to get impression of it, and later, for a very strong candidate, even a proper test – duplicating an existing test or trying it in a real project. During that, there should be technical and other problems and that gives a possibility to test the support the tool has – if not commercial support or support from its developers, at least community support.

After that it should be clear which tool can be chosen.

3. Details and notes of the selection issues

3.1. Do others use it?

First thing to do in the evaluation of a tool is googling about it. Are there reports of usage of the tool or is there silence? One does not want to select a tool that nobody else is using, unless there are good reasons to do so.

3.2. Is the tool developing?

The open source world is full of tools. Some are one-off projects by a developer or a research project. Those are to be avoided. Look for tools that are actively developed right now. Signs for that include a web site or preferably a public source code and binary distribution repository that shows continuous activity.

3.3. Support and community issues

Support is essential. A tool should have support forums that the users, vendors and tool developers use. Questions can be asked on the forums but do they get answered? It is not uncommon that users post a question about a problem but it never gets answered. That tells about an unresponsive vendor and a passive user community, both of which are very bad signs. Sometimes the questions point out a bug in a tool, but again, the most important thing is, so soon will the bug get fixed? Many tools have a public bug tracker and that will tell a lot about how the tool is developed – are there serious bugs, how they are handled, how soon they are fixed – if ever.

3.4. Is your environment supported?

This is obvious, but some tools only support Windows or Linux as a working environment. Some use virtual machines to do the actual work. You don't want to impose a new environment for testers.

3.5. A quick smoke test

Nowadays it is expected that tools should be easy to take into use. It is not acceptable that hours should be spent on installing a tool and crafting all sorts of configuration files and such. Instead, a tools should be as easy to take into use as an office application. Just install it – preferably just by dropping it to a directory – and the tool will imply how it is used. A first test should be under implementation and running very rapidly. Of course, adaptation to a SUT is often more difficult, by its nature, but testing of a local application should be possible "immediately".

3.6. Testing the system

The test system should be evaluated by using it in a real and real-sized testing tasks. As they say, "devil is in the details" and in test automation it definitely is so. One needs to assure that the hard testing tasks can be handled with the system and the largest things to model in the future can be handled easily.

3.7. Likeability

It is important that the testers like the tool. Ideally, some will like it so much that they will fall in love with it and want to become virtuoso gurus of it, being able to use it in optimal ways, to do tool integration and some tailoring too.

3.8. Simple is good

During previous decades, complexity was culturally appreciated in processes and tools alike. If something was not complex, it was obviously not good. But things change. Now it is understood that simple is good. People understand simple things, they are easy to use and maintain. Note that for example a simple interface must not mean lack of features, but good design in packaging the features.

3.9. Need for tailoring and integratability

Sometimes it is thought that open source tools can be tailored for own use, but in practice, people in companies have more pressing things to do than tailor tools. It is best to be able to use a tool as it is. That makes deploying updates much easier. So, the key thing here is configurability and the quality of the tools interfaces. Sometimes the tailoring is about integrating tools, but if the existing interfaces are based on industry standards (at least de-facto-standards, such as support for tools used in JUnit based tool chains), no modifications are needed.

Integratability is indeed important in serious industrial testing. Areas for that include test management, test drivers (using for example Robot Framework to act as a driver), measurements and logs (in long duration test mapping of instrumented measurements into test activity) and reporting (automatic pushing of results to reporting systems, radiators).

4. MBT tool assessment checklists

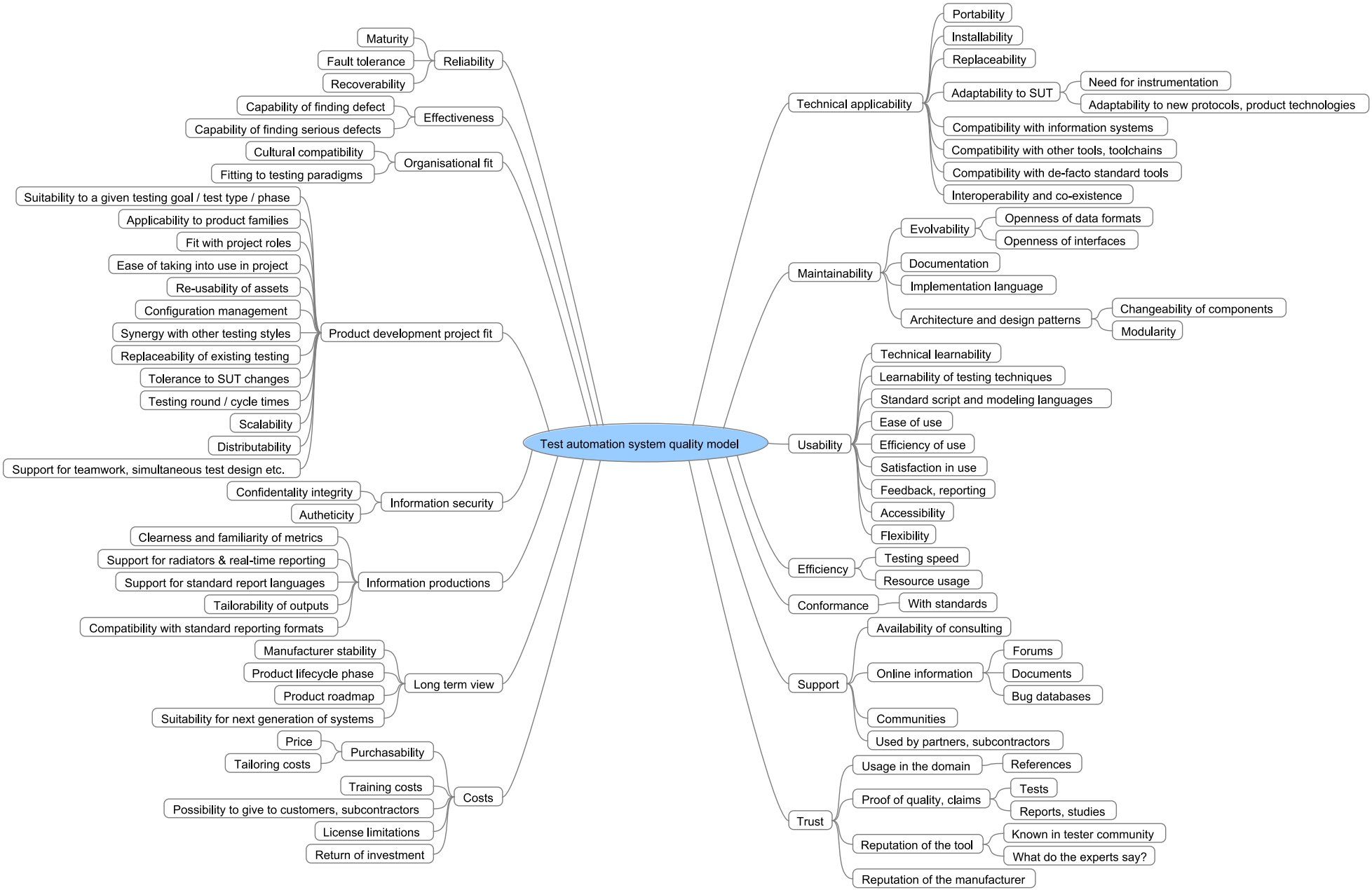
Note: Of course, both checklists only contain a small subset of potential issues in order to stay compact and usable. For a serious use in a company, the lists should be assessed and edited – essential things added and non-essential things removed.

There are two checklists as attachments of this report. The first one is “Test automation system review checklist” that is designed to support the overall assessment of the proposed test system where the MBT tool is planned to be used. After all, MBT based automation is just automation, it just uses different kind of tools.

The other checklist “MBT tool characteristics checklist” is for assessing the MBT characteristics of the tool and how it can overcome the many obstacles of MBT in real life.

.

APPENDIX 1: Test automation system quality model mind map



Matti Vuori, 2014-01-10

Test automation system review checklist

This is a general checklist for use in reviewing a test automation system's applicability in an organisation. The list should be tailored for any particular use.

Target of review	
Reviewers	
Date	

1. Test effectiveness

Criteria	Status	Notes
Test techniques are able reveal defects effectively.		
Negative test cases are easy to implement and the system gives good support for negative test cases (handling of exceptions for example).		
Automatic test generation supports different algorithms and strategies for diversity (especially for safety-critical domains).		
Other		

2. Technological suitability

Criteria	Status	Notes
Technological compatibility with system under test is sufficient.		
The limitation of testing (API's, configurations or bypassing things during testing) has been assessed.		
Other		

3. Test system as part of testing infrastructure

Criteria	Status	Notes
Compatibility with overall tool chains.		
Compatibility with test management systems and reporting system.		
The system supports well the test phases and levels it is intended to.		
Other		

4. Effectiveness in use

Criteria	Status	Notes
Test design is fast and easy.		
Test design can use as domain-specific terms as possible.		
Test design is done in as abstract level as appropriate.		
Test designs, cases and data can be reused easily.		
Test system is fast to adapt to a new project.		
Testers can learn to use the system rapidly.		
Licensing or cost issues do not restrict its use.		
The system is secure, if used in distributed manner or in a public network or cloud.		
Other		

5. Efficiency and reliability

Criteria	Status	Notes
Test runs are fast and do not cause a bottleneck to development workflows.		
Test system is reliable enough to be used in long test runs.		
Test system can recover from problems during a test run and be restarted from where it was stopped.		
Other		

6. Maintainability and evolvability

Criteria	Status	Notes
Test system is easy to enhance and repair.		
Test system is easy to adapt to new product technologies (UI technology, protocols etc.).		
The system is based on standards and/or open or very common specifications (source, protocols, data formats).		
Other		

7. Validity

Criteria	Status	Notes
Everybody understands the possibilities and limits of the system.		
The system can be used with confidence in most demanding projects.		
Test system conforms to relevant domain standards.		
Correct application of the system can be audited or verified.		
Other		

8. Support

Criteria	Status	Notes
There is knowledgeable and fast support available for the system.		
There is a user community that can aid in problem situations.		
Other		

9. Other issues

Criteria	Status	Notes
Something else that is relevant in this context?		

Matti Vuori, 2014-01-10

MBT tool characteristics checklist

This is a general checklist for use in reviewing a MBT tool's characteristics. Note: generic test automation characteristics are not included in this list. See "Test automation system review checklist" for those.

Target of review	
Reviewers	
Date	

1. Technical suitability

Criteria	Status	Notes
Can the tested functionality be modelled with the tool?		
Does the tool easily allow good negative testing, including variation in data?		
Does the tool support sufficient test generation algorithms?		
Do the tools support online and offline testing?		
Can the tools integrate with other testing tools used in test management, as test drivers or in reporting?		
Other		

2. Working environment

Criteria	Status	Notes
Does the tool support your main operating system used in testers' workstations?		
Does it support your favourite virtual machines?		
Can it be used with your standard IDE?		
Other		

3. Ease of modelling

Criteria	Status	Notes
Does the tool use a modelling style that is “natural” to testers?		
Is modelling simple and fast to do with the tool?		
Is the tool suitable for the smallest ad-hoc tests that you want to run *now*?		
Are models understandable by others than the modeller?		
Are models fast enough to maintain that they can be used in rapid agile development?		
Other		

4. Management of models and scaling

Criteria	Status	Notes
Is it possible to split the models so that individual models will not become too large for maintenance?		
Is parametrisation of the models or other techniques possible to make the models work for a similar SUTs without remodelling or copying?		
Do the models support version management and concurrent editing by more than one tester?		
Can models be edited with external tools? (Such as a bulk change of some terms.)		
Is adaptation handled in such a way that adapter code can be managed separately from the models?		
Other		

5. Flexibility

Criteria	Status	Notes
Is it possible to create the models with various tools, from various sources?		
Do the models use a standard logical and file format to support that?		
Is there a system of tags or equivalent to mark model elements for customized handling or logging?		
Other		

6. Validation of the models

Criteria	Status	Notes
Are there tools that make it possible to validate the model by simulation or other means?		
Do the visualization tools produce output that is easy to understand and to review?		
Other		

7. Reporting

Criteria	Status	Notes
Does the system provide clear reports of the findings that as much as possible aid in understanding and tracing of the bugs?		
Does the system support business-level metrics that give value to test management and quality management?		
Does the system have mechanisms (tagging. file format etc…) that give options for generation of customised re-ports?		
Other		

8. Debugging support

Criteria	Status	Notes
Are there sufficient features for debugging the model?		
Does the system produce logs that aid in understanding what it does during a run and what happens in each step?		
Other		

9. Multi-platform

Criteria	Status	Notes
Is the MBT system target system independent and can be used for example if the target platform changes?		
Other		

10. Deployability

Criteria	Status	Notes
Is the MBT system easy to deploy to new environments, new computers, to other parties?		
Other		

11. Reliability

Criteria	Status	Notes
Is the tool reliable enough for serious long duration testing?		
Other		

12. Likeability

Criteria	Status	Notes
Do testers like the tool?		
Would they want to use it – in preference to other tools?		
Other		

13. Other issues

Criteria	Status	Notes
Something else that is relevant in this context?		