

# Necessities and Tactics of Tool-Based Software Testing

Mingtao Shi

FOM Fachhochschule für Oekonomie & Management

University of Applied Science

Bismarckstr. 107, 10625 Berlin, Germany

Tel: 49-171-2881-169 E-mail: Consulting\_Shi@yahoo.de

## Abstract

Today's Small and Medium-sized Enterprises are confronted with the IT-reality, the resource-reality and document-reality. Introduction of software systems and thus performing software testing have long become an indispensable necessity for a firm's survival. A number of suggestions relevant to the practice are advanced by this paper, targeting at conducting successful software testing. A minimal set of business information must exist. Operational software testing should be supported by a software-based testing tool that provides a number of essential technical features. Seamless communication among participating departmental areas is another key success factor.

**Keywords:** IT-reality, resource-reality, document-reality, minimal set of business information, testing tool, communication

## 1. Introductory Remarks

The literature has discussed the process of *software development* extensively. Authors have modelled a number of mainstream methodologies used by practitioners, such as Waterfall Model, V-Model, Incremental Model, Evolutionary Models, component-based development, aspects oriented development (see e.g.: Pressman, 2009; Rupp, 2007; Wiegers, 2003). It can be clearly recognised from the literature that no matter which model is considered, software development essentially consists of the following sequential key elements: requirement elicitation and analysis; design analysis; coding; testing analysis and testing execution; delivery and maintenance; progressive updates and upgrades. Business firms usually package these development activities in a project management environment.

Thus, software testing is an indispensable issue in the development cycle. The quality of testing activities matters importantly to the overall quality of the software system produced. Software testing methodology must generally be independent upon programming languages and independent upon industrial context where the software is applied. The existing body of literature has treated the strategic issues of software testing sufficiently. The major arguments are that software testing should optimally be conducted in a *hierarchical manner* including unit test, component test, integration test, system test, user acceptance test, and methodologically integrate *mathematical-logical models* which enable the design of effective and economical test cases (see e.g.: Bath & McKay, 2010; Ammann & Offutt, 2008; Patton, 2006; Myers, 2004; Burnstein, 2002).

Testing models and techniques advanced in the literature are undoubtedly of high value, but are not always completely and perfectly applicable in the practice. This is due to a set of business conditions, especially those representing the mundane reality in Small and Medium-sized Enterprises (*SMEs*). It is worth discussing the reality of these businesses, because they form the basis of many economies. In Germany, for example, *SMEs* are the major contributors of aggregative GDP. While looking at the software testing, strategies and particularly tactics used must recognise and be in alignment with the conditions existing in the reality.

This paper first examines these real-life challenges briefly before it turns to investigate the technical features required for tools that support and administrate software testing in business practice. Finally, this contribution also succinctly discusses the organisational "features" necessary for ensuring the positive testing results. Notably, technical and organisational (hard and soft) features are the two significant success factors in software testing.

## 2. Business Reality

*SMEs* today are confronted with the "*IT-reality*". Software deployment has become ubiquitous in almost all industries. Most *SMEs* are primarily interested in applying software and only few software houses are "making" the software. Gradually, industrial applications based upon core systems that can be individualised by parameterisation and customisation have emerged over the past decades. Industrial firms are not producing core, the core system is rather delivered by specialised software houses such as SAP. Industrial firms are more

concerned with using (testing) the functionalities and features made available by the core system to build their respective value activities.

SMEs today are further confronted with the “*resource-reality*”. Time, budget and human resources are always in shortage. Without considering the human resources, the industrial experience of R&D projects shows that at least the doubled amount of originally planned time and doubled volume of originally planned financial resources are required for the arrival at the originally planned target. To be sure, the time and budget pressure of a software implementation project is equally high. Human resources are also usually in shortage. Testing professionals equipped with systematic background knowledge and practical experiences are rare and extremely expensive. Without much exaggeration, such personnel can only be found in large firms or specialised software houses. Testing teams in SMEs are frequently not immediately familiar with the business and do not in possession of vast testing knowhow in theory and practice. Generally speaking, smaller firms are, in terms of technical competencies, usually not capable of comprehending, constructing or changing the source codes in the core system. In other words, SMEs are heavily dependent upon operational and managerial software systems without possessing the in-depth technical knowhow of these systems.

SMEs today are also confronted with the “*document-reality*”. For firms not specialised in the software development, fine-grained specification documentation is most probably non-existent. One the one hand, it is virtually impossible, especially for SMEs to allocate significant funds and human resources (system analyst, software engineer) to external or internal software specification activities as new businesses are added to the firm’s profit portfolio or existing product lines are upgraded. One the other hand, as the experience has shown, the mentality of the top management in many SMEs is that detailed business-related and IT-related software specifications are less important for the system development: “Functional managers can always tell the developers what to programme.” Lack of systematic written information and structured documentation is a general difficulty for software implementation projects which will continue to exist. Experiences of the author during various testing processes has repeatedly revealed the lack of and holes in requirement documentation and thus the existence of the document-reality in most SMEs.

These reality realms do have profound implications for the testing practice in SMEs. To put it simple: Software testing is always required in firms because of the IT-reality; Full-fledged testing as described in the literature is hardly possible because of the resource-reality; Software testing cannot solely rely on the existing documentation because of the document-reality. The author argues therefore that testing tactics leading to visible success may be formulated as: Software testing should enlist the *support of software-based testing tools* and conducted in an organisational architecture that advocates and supports *mutual communication* of different expertise areas.

### 3. Tool-based Testing

Given the complexity of today’s businesses and their related software systems, software testing without tool support would be unimaginable. However, testing teams should be aware of the prerequisite of using testing tools and of the major aspects of what the tool should perform, in order to select and purchase the most effective tool for the firm-specific situation.

Although comprehensive documented business information is rarely available in most SMEs, a minimal set of essential business information must exist for testers and tool-based testing. This minimal set of business information is the prerequisite for using the testing tool and should at least cover a number of informational areas. Firm’s operational business *processes* must be mapped, encompassing the description of functional roles (who), process steps (what) and detailed activities of the respective process steps (how). If allocated resources allow, system-related steps and activities should be examined in greater detail, including system inputs, outputs and printouts. Firm’s *products* must also be mapped, at least all important functionalities and features, and all salient internal and external business interfaces. Aggregative *reports* related to finance and accounting, regulation and law compliance, administration and management certainly belong to the minimal set of written business information that need to be at the disposal of a firm. Furthermore, *user rights* and *parameters* to be applied in the system should be reviewed and determined precisely. Despite the resource-reality firms are forced to invest in the production of these documents, without which the testing team is unable to design and carry out meaningful tests. Notably, the document-reality still exists even if the minimal set of business information is available, because the scope and depth of these documents are unlikely to be in line with the documentation requirements of software engineering.

Based upon the prerequisite, testing activities can commence. The minimal set of business information forms the basis for designing the test cases. Important strategic issues related to test case design are debated and agreed upon widely in the literature. However there are a number of important tactical issues, of which the testing team

should take care. These tactics are simultaneously the major considerations for choosing testing tools. There are virtually no testing tools that can replace the human brain and design the *test cases* automatically. Test case designer must be familiar with the business and the testing strategies, and input the test cases into the testing tool manually. The tool should preferentially be able to present the *linkage between designed test cases and business information* provided. Business information related to user rights provides the basis for testing the *interrelatedness among users/roles, screens made available to these roles and the functional permission accessible for the roles on the screens*. One test case describes one single business function or one report unit. To realise this function/test case, a number of *test steps* are necessary and to be input into the tool. The test step delineates and describes which action is to be taken (in interaction with the system) and how exactly the system should react on this action. If numbers and figures are treated at a certain test step, *designed input and expected output values* are to be input into the testing tool. *Alternative paths, invalid cases and boundary values* may exist at a certain test step and should also be tested. These “exceptions” usually result from the previously defined system parameters related to process or product. A whole business process or product can be tested if the designer can *connect the individual test cases to “reconstruct” the process or product* in the tool.

Subsequently, the tester utilises the work products of the test designer to execute the tests in the current system release and register the *test results* in the testing tool. The tests should be performed for both individual and connected test cases. Therefore, *test executions and results are to be associated with test cases*. Additionally, the testing tool should also be used to register *irregular events and phenomena not immediately related to test cases* but observed during the testing in the system. The information of the test results contributes importantly to bug fixes, release update and system upgrade. It is worth mentioning that script-based automatic testing is an issue in software testing, but most SMEs lack technical expertise to produce useable results in this area.

Additionally, there are some administrative aspects that the testing tool should integrate. Test cases have different *priorities*; test case execution and debugging tasks must be assigned to *responsible persons* and the *time* of these activities be *estimated*. The tester should be able to *upload* and archive screenshots or other additional *textual or graphical information* to the testing tool for communication purposes. The in-depth discussion of these administrative aspects is however beyond the scope of this paper.

A number of test tools and test management tools exist currently, performing the features mentioned above, such as Selenium and Spira.

#### 4. Organisational tactics

Software testing requires *seamless communication* among several departmental participants. The *testing team* must design and carry out the tests. The *programming team* must react on the feedbacks of the testing team and perform debugging activities. The *commercial or operational team* must enrich, change and explain the existing business information to the testing and programming teams as such requirements or inquiries arise. The *project manager* must monitor and evaluate the overall progress of software testing, in order to make necessary adjustments in questions of time, budget and resources.

The detected bugs, failures and malfunctions must be described and documented in adequate scope by understandable written information in the tool, so that the programming team is able to understand and subsequently embark on these irregularities. Communication between the testing and programming units thus contributes importantly to the incremental system amendments during the testing. The commercial team needs to be aware of the fact that members of the testing team may raise questions related to operational businesses (processes, products and parameters) from time to time, in order to design useful test cases. Members of the commercial team can use this opportunity to discover the insufficiency in the documentation and make improvements correspondingly on the one hand, and to train the testing team for better business understanding on the other hand. The improvements in the business documentation in turn necessitate the interaction between the commercial and programming teams. Change requests or even new features may become the results of testing activities. Changes after the system has already been constructed are usually expensive undertakings; however such situation is hardly completely avoidable in the practice.

The means of communication can be telephonic and personal or textual and graphical. Most importantly, the results of the communication should be in written form and cumulatively recorded in the testing tool. The testing tool is the platform and the medium of the communication. Effective and economical communication is only possible, if the above mentioned technical features are readily available in the testing tool and the teams are willing to collaborate smoothly and to tolerate each other's occasional mistakes.

## 5. Conclusive Remarks

Software testing in operational practice has been dependent upon a set of business conditions. Software testing is ubiquitous in today's business environment because of the IT-reality. Testing techniques and processes described in the literature is not completely feasible in the practice because of the resource-reality. IT-related specification and documentation are either non-existent or too thin in most SMEs because of the document-reality.

This paper argues that in order to encounter these real-life challenges, firms conducting software testing need to put a minimal set of business information in place, enlist the support of a software-based testing tool and enable seamless communication among participating departmental areas.

The minimal set of business information should include the firm's business processes, products and parameters at an at least moderately detailed level. The testing tool to be selected should be able to record test cases and input/output values, show the linkage between the designed test cases and related business requirements, allow test cases to be connected to the firm's processes and products, record the results of the execution of test cases and register irregular events and phenomena not immediately related to test cases. Seamless communication among testing, commercial and programming teams also contributes greatly to the success. Software testing can become much more effective, if the functional participants can supplement their respective expertise through pleasant knowledge exchange and thereby checking and improving the system incrementally.

## References

- Ammann, P., & Offutt, J. (2008). *Introduction to Software Testing*. New York, NY: Cambridge University Press.
- Bath, G. & McKay, J. (2010) *Praxiswissen Softwaretest - Test Analyst und Technical Test Analyst: Aus- und Weiterbildung zum Certified Tester - Advanced Level nach ISTQB-Standard*. Heidelberg: dpunkt Verlag.
- Burnstein, I. (2002) *Practical Software Testing*. New York, NY: Berlin/Heidelberg: Springer Verlag.
- Myers, G. J. (2004). *The Art of Software Testing*. (2<sup>nd</sup> ed.). Hoboken, NJ: John Wiley & Sons, Inc.
- Pressman, R. S. (2009). *Software engineering: A practitioner's approach*. (7<sup>th</sup> ed.). New York, NY: McGraw Hill.
- Patton, R. (2006). *Software Testing*. (2<sup>nd</sup> ed.). Indianapolis, Indiana: Sams Publishing.
- Rupp, C. (2007). *Requirements-Engineering und Management: Professionelle, iterative Anforderungsanalyse für die Praxis*. (4th ed.). München/Wien: Carl Hanser Verlag.
- Wiegers, K. E. (2003). *Software requirements*. (2nd ed.). Redmond, Washington: Microsoft Press.