Testing in Maintenance

author: Leo van der Aalst
based on the original white paper

TESTING IN MAINTENANCE

author: Leo van der Aalst
based on the original white paper

1 INTRODUCTION

Since the eighties, the use and development of information systems and technical infrastructures has grown more than ever. The dependency between business processes and automated information systems become increasingly stronger. Besides this, existing information systems often need to be adapted to new wishes. Since the beginning of the nineties awareness has grown that the costs during use and maintenance of information systems exceed the initial development costs by far.
The existing literature on testing and test methods provides sufficient grip on testing of new software applications. Practical experience has proved that it is difficult to apply this approach to testing of software during maintenance.
If we assume that the average life expectancy of an application is ten years [Tamai 1992], we can safely state that testing during maintenance happens more often than the testing of new software. Consequently, it is not surprising that Grady shows that more money is involved in testing during maintenance than during the development period of software applications [Grady 1987]. Therefore, it is all the more striking that little can be found on this subject in literature. With this document we attempt to fulfil the needs of many that deal with testing during maintenance on a regular basis. For this we employ the structured test methodology TMap® as a basis, and give the differences for testing during maintenance. The reader is assumed to have a basic knowledge of TMap®.

Testing during maintenance is not a new work area. On the contrary; for many people it is a daily routine. The advantage of this is that a lot of practical experience is available, and these experiences are incorporated in this document. Some examples are:

- Organizing a *kick-off session* with all relevant parties to obtain clarity on various subjects.
- Creating a *standard maintenance test plan* containing all reusable test aspects.
- Setting up, using and maintaining a *regression test set that can be adjusted in size*.

We would like to thank the people below for their cooperation and contribution to the creation of this document: Rob Baarda, Eric Begeer, Anna Blok, Martin Boomgaardt, Guy Holtus, Gina Koobs, Tim Koomen, Peter van Lint, Marc Valkier en Michiel Vroon.

2 SERVICE MANAGEMENT AND MAINTENANCE

Service management and maintenance are often bracketed together without considering different possible interpretations of those words. Before elaborating on testing during maintenance, this chapter will explain what is meant by service management and maintenance.
Furthermore a possible relation between various service management forms and an organisational classification is made, and the life expectancy of an information system is considered.

2.1 What is service management

In many organisations service management is applied according to the Mintzberg model [Delen 1992] on strategic, tactical and operational level.
On strategic level decisions are made that are of strategic importance to service management, for example: "service management is outsourced as much as possible".
On tactical level the tasks concerning making material, personal and financial resources available can be found.
On the operational level tasks that are essential for the daily execution of service management are defined, and therefore this level will be explained further.
The total service management of the IT on the operational level can be divided into three forms:

- Business information management
- Management of technical infrastructure
- Application management.

Business information management

Business information management contains all management and control tasks that are necessary for the daily use of information systems, the data infrastructure and changes of the specifications of this.

In daily practice business information management comprises:

- The acting as intermediary between users and service management of the IT.
- The assistance and education of users in relation to the use of the information systems.
- The assigning of authorisation.
- The management of application specific data.
- The content management of the database.
- Securing the correct use of the information system.
- The maintenance of manual procedures.
- The maintenance of functional specifications.
- The planning and execution of a (User) Acceptance test.

Management of technical infrastructure

Management of technical infrastructure includes all tasks that are necessary for the instalment of information systems and technical infrastructure, and to make and keep them operational.

Management of technical infrastructure comprises amongst other things:

- Making standard packages for personal use, like word processing and electronic calendar, available and maintain them.
- Making the information systems for the support of group work available and maintain them.
- A helpdesk function.
- The maintenance of technical infrastructure.
- The offering of processing capacity.
- The offering of storage capacity.
- The planning and execution of a Production Acceptance test.
- The planning and execution of unit-, integration and system tests on *technical* infrastructure.

Management of technical infrastructure is therefore mainly concerned with the technical platform, consisting of hardware with matching basic software, and making the information system that is built on it operational.

Application management

Application management includes, amongst other things:

- The management of the application library.
- The management of databases.
- The changing of application programs.
- The changing of databases.
- The planning and execution of the unit, integration and system tests on applications.

Even if there is no immediate reason to perform application maintenance, it is still necessary to maintain a full application management structure, including management and support functions, to guarantee continuity.

2.2 What is maintenance

A part of application management that occurs on operational level is called maintenance. Maintenance is the changing or expanding of an existing information system, and can be performed ad hoc or in a planned way.
Ad hoc maintenance is performed to fix defects that cannot be delayed, because they cause unacceptable damage in production. Corrective maintenance is the only form of ad hoc maintenance.
Planned maintenance comprises all other types of maintenance. These are performed in accordance with regular development processes per release, which usually start with an impact analysis. Some types of corrective maintenance can also be performed in a planned way. This concerns defects that do not need to be solved immediately, because they cause acceptable or no damage in production.

The figure below shows which five types of maintenance [Delen 1992] are distinguished.

	Ad hoc	Planned
Maintain	corrective	corrective
	🟦	preventative
	🟦	adaptive
Improve	🟦	perfective
Expand	🟦	functionality

🟦 No occurrences 🟪 Occurring most frequently

Corrective maintenance is the correcting of defects in components of the information system. Depending on the object of maintenance, this can vary from eliminating bugs from the application programs to setting hardware malfunctions right. In case of corrective maintenance the intended use and the exploitation of the IT remain unchanged.

Preventative maintenance is the rectification of components of the information system without immediate cause in a problem report. The purpose of preventative maintenance is: (a) preventing future problems, or (b) improving the maintainability.

Adaptive maintenance is the adjustment of one or more components of the information system due to changes in the environment of those components. Adaptive maintenance can be triggered by maintenance on another part (mostly in an underlying layer) of the information system; by changes in another information system with which an interface exists; or by changed rules concerning the business function supported by the information system.

Perfective maintenance is the adjustment of a component of the information system to the altered quality standards of the user.

Functional maintenance contains the expansion or change of the functionality of the information system.

2.3 Service management organisations

In this paragraph the three management forms are translated into three different organisational entities. Often three different types of organisations are distinguished, namely:

- The user organisation;
- The infrastructure organisation;
- The system development and maintenance organisation.

The service management types are attributed to these in the following way:

Business information management	→ User organisation.
Management of technical infrastructure	→ Infrastructure organisation.
Application management	→ System development and maintenance organisation.

The user organisation is the whole of people and means that deal with the business processes. Within such a user organisation the emphasis is on primary business functions in the first place, and the more supportive functions, like the management functions with regard to IT, in the second place. IT management functions in this context are business information management functions, which cannot be outsourced due to knowledge of the subject matter, responsibilities and knowledge of the user organisation that it involved. Usually a separate IT management organisation within the user organisation is needed because, as stated before, the user organisation cannot exclusively be engaged in business information management.

An infrastructure organisation is the whole of people and means that deal with making and keeping the information systems and technical infrastructure operational. These kind of organisational units can vary from an extensive infrastructure service centre to a small-scale infrastructure group at a department within the user organisation. Since the activities of an infrastructure organisation can be defined well, many organisations decide to outsource this line of business.

A maintenance organisation is the whole of people and means that deal with development and maintenance of applications. Within a maintenance organisation it is customary to have separate units for the benefit of application development and application maintenance. A difference between the two is that development is usually performed by a non-recurring project organisation, while the organisation for the maintenance is a permanent one. Like it was mentioned before for an infrastructure organisation, the outsourcing of activities of a maintenance organisation has increased as well.

Cooperation between the service management units

The business information management, application management and management of technical infrastructure could be performed at different locations within the same company and some maintenance could even be outsourced. It is therefore of great importance that the three service management units cooperate. When an information system is changed, close cooperation between these organisations, and, if necessary, with external parties to who work has been outsourced is required.
Two different change categories can be distinguished:

1. *Change management*
   This includes adaptive, perfective and functional maintenance. Within this maintenance types the use and/or exploitation of the information system changes and collective decision making by the management of all parties involved is required.
2. *Problem management*
   This includes corrective and preventative maintenance. In these the use and exploitation of the information system does not change. Therefore coordination on the operational level, like a help desk, is mostly sufficient here.

2.4 Life expectancy of an information system

Maintenance is costly. Therefore, it is important for an organisation to decide to which point maintenance is cost effective. After some time it is often cheaper to have a new information system built than to continue to maintain the existing one. Right after implementation a rise in maintenance can often be seen due to initial problems of the information system (corrective maintenance). During the operational period of the information system the number of real defects will hopefully decrease, but the other four maintenance types start to increase. Because of maintenance the structure of the program will usually start to deteriorate gradually and consequently the maintainability will decrease. Furthermore, maintenance, like development, is human work, so there is a chance that new defects are introduced. The deteriorating structure and the newly introduced defects give rise to so-called aging problems, because of which it will be

decided at a certain point that the information system in question should be replaced from an economic point of view.
Research performed by Tamai on 95 systems has shown that the average life expectancy of a system is 10 years [Tamai 1992]. A variation from 2 to 30 years was recorded. (See figure 1, life expectancy information systems).

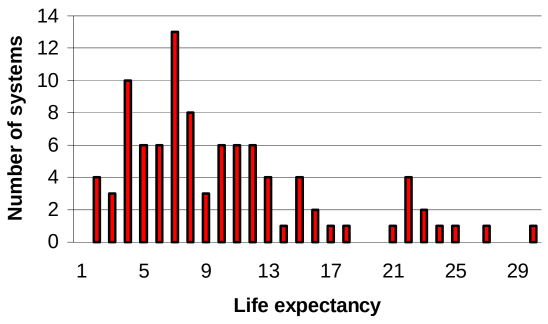

Figure 1, life expectancy information systems.

3 POINTS OF INTEREST FOR TESTING DURING MAINTENANCE

After the chapter explaining service management and maintenance and preceding the chapter describing testing during maintenance, this chapter covers some specific testing issues during maintenance. These are:

- Planned and unplanned maintenance.
- Development process for maintenance.
- Challenges for service management organisations.

In paragraph 3.4 references are made for solutions.

3.1 Planned and unplanned maintenance

Planned

90% of all maintenance on information systems is planned. Since an information system spends more time in maintenance than in new development, remember the earlier mentioned life expectancy, it will be tested during maintenance for a longer period, and it is therefore important to choose and apply a structured test approach.

Unplanned

The remaining 10% of maintenance is unplanned (ad hoc), mostly corrective maintenance. In these cases it is clear that solving a production problem has highest priority. In general, little or no time is available to perform a risk analysis, define a test strategy and even draw up test cases. It is however desirable to go into production with the changed software in as much a controlled fashion as possible. For the tester involved it is a challenge to give a quality judgement of the implemented change, even in non-planable, ad hoc maintenance.
It is not always necessary for corrective maintenance to be performed immediately. Sometimes solving a production problem consists of repairing the consequences. For example correcting data or helping users go on with their work. The structural correction of the problem will take place at a future moment and is planable.

3.2 Development process during maintenance

The development process for maintenance differs, although not substantially, from the one in new development. The differences mainly occur in the initiation of the process and during testing. Maintenance is usually initiated by a change request (for example changes in functionality, middleware or package) or a release plan, and at the same time the relative part of testing in maintenance is often larger than in development. Even though the process is similar to the one in new development it will influence the test strategy. After all, in case of new development the system is newly built and 'everything' is tested. In maintenance, however, changes are made to an existing system. Questions at this are, if it is sufficient to test just the changes, and how to deal with possibly already existing testware.
It becomes even more difficult if an uncontrollable situation threatens to arise because the deliveries of the maintenance team turn out to be partial solutions that are delivered spreaded over time.
Besides this the situation occurs that parallel to planned development and/or maintenance processes, ad-hoc changes are made to the system in production, and that these are not taken into account. This can cause, amongst other things, gaps or overlap in testing activities.

3.3 Challenges for the maintenance organisation

Contrary to new development, which is executed in a non-recurring project organisation, maintenance is mostly embedded in a maintenance organisation (usually in application management). A complicating factor in this is that besides this maintenance organisation, business information management and management of technical infrastructure organisations exist. After all, business information management 'supplies', amongst others, application management with the functional specifications, and is often responsible for the execution of a (user) acceptance test. Application management, in turn, is responsible for implementing the changes in the software, and the execution of a program and/or system test. Besides that, management of technical infrastructure is, for example, responsible for the test infrastructure and the execution of a production acceptance test. It is of great importance that these organisations cooperate in a clear way. An important aid in this is performing an impact analysis on every change request, from which should become clear what needs to be adjusted (infrastructure, specifications, applications and testware etc.) and which organisation needs to do what. Apart from this the organisations need to solve a number of (test) challenges together in the following areas:

Choices in test infrastructure

In test infrastructure choices need to be made concerning the availability of a permanent test environment containing a test data base with either up-to-date or outdated production data. The test environment that is necessary for mainly unplanned, ad-hoc maintenance also needs to be considered.
Furthermore it needs to be researched whether automated testing is an option. A challenge that comes with this is the availability of a correct, up-to-date test database. A possible conflict situation is that a test database with up-to-date (production) data may be needed for testing a change, while the automated testing requires a well-defined and stabile test database with known, but possibly outdated test data. To avoid a conflict situation, the data to be used in the test database, and the way it should be filled and/or kept up to date, needs to be considered thoroughly.

Testware management

Time is not always reserved for the management and maintenance of testware. On top of this, with ad hoc maintenance, solving the production problems appropriately gets more priority than bringing the testware up to date. In general, choices need to be made if, and if so how, the testware can be kept up to date. It is also desirable to have a regression test set available. Just like with testware in general, during maintenance questions arise like when, how and by whom the test set can be kept up to date.

Know-how is essential

It is not easy to write a test case that exactly tests the solved production problem or specific change made. It is not seldom that thorough knowledge of the system is needed for that, even more so because the system documentation is often absent or not up-to-date. Much knowledge is required to create exactly the initial situation in the test database that is necessary to execute the test case. Such know-how is not always sufficiently present in maintanance organisations. Often it is a path of growth that usually only starts at the moment a system is handed over to the line organisation by the project. If know-how is not sufficiently present in the maintanance organisation, thorough consideration has to be given to how to deal with this while writing the test cases.

Reduced motivation of testers
New development projects often have room for separate testers. During maintenance this room does not exist, and all tests have to be executed by developers or service managers. These often have an aversion to testing, because of the dull and repetitive character. This can lead to reduced motivation amongst the developers to start testing. It is important to take measures to prevent such a situation from occurring.
With this, solving production problems is given priority, appropriately, over test related work. This can however have negative influence on the quality and progress of the tests. Furthermore it is not unusual that service managers have little to no affinity with testing and will take any opportunity to do anything but spend time on testing.
It is important to recognize these symptoms and to take measures.

Scope of maintenance
Maintenance comes in all sizes. Ranging from small changes to complete redesign of the information system as it were. Extensive changes require a different test strategy than less extensive changes. Just like in new development projects it is often ‘forgotten’ to give sufficient attention to (testing of) non-functional quality attributes.

Maintenance influence on specifications and test cases
As described in paragraph 2.3 on service management organisations, a distinction is made between problem (or defect) management and change management. In case of problem management the specifications usually do not change, in contrast to change management. Complicating factors for the testing of defect fixes are that it is not always clear to the tester what exactly caused the defect and how to reproduce it. In case of change management it often happens that the change is not well documented. Due to the lack of an impact analysis it can be unclear how, and where the change was implemented. In both cases a certain route needs to be chosen to deduct test cases. In reality it has shown that it is difficult to choose exactly those test cases that 'put the finger on the sore spot'. It often turns out to be specialist work which requires specific know-how of the system and/or subject matter.
Besides this, specifically for problem management it goes that a defect should not only be repaired in the current production version, but in all future software versions as well. When drawing up a test strategy this has to be taken into consideration.

Improper test tasks
In maintenance situations testers are often confronted with improper, non-test tasks. This is mainly due to the fact that service managers need to perform test tasks as well as other service management tasks. In some cases a combination of tasks can be valid, but it is recommended that testers consider whether or not the following tasks are part of testing. The problem is not so much that one person, for example the business information manager has other tasks besides testing, but the danger is in the fact that somebody may have to perform tasks for which he or she is not suitable, or that management is faced with a huge amount of test hours and takes impropriate action to reduce these.
Examples of non-test tasks:

- Monitoring production trends. For example collecting statistics on system failure or the number of helpdesk calls per time unit.
- Analysis of the causes/consequences of disturbances, informing the end users about implemented changes and reporting project status to the business are all examples of tasks that are fitted to a business information manager, but probably not in the role of tester.
- When solving a production problem it may be that besides adjusting the software and/or developing a corrective program, actions towards employee/customer (like an apology letter) are also put out. The question is if a tester has anything to do with this.

3.4 Reference to solutions

The next chapter indicates what testing during maintenance situation should look like. The points of attention described in the former paragraphs are dealt with in various parts of the next chapter.

The table below describes per point of attention in which parts of chapter four the matching solution can be found.

Point of interest for testing		Solution(s)	
3.1 Planned maintenance		4.1	Specific process Standard maintenance test plan
		4.3	Permanent test environment
		4.4.1	Test strategy
3.1 Unplanned maintenance		4.1	Specific process Standard test plan for maintenance
		4.3	Test environment available on demand
		4.4.1	Test strategy
		4.4.4	Test quality improvement for ad-hoc maintenance
3.2 Development process during maintenance		4.4.1	Fixed and flexible test expenses Delivery by release Test strategy
3.3 Challenges for the maintenance organisation		4.2	Planning, priorities, structure
	Choices in test infrastructure	4.3	Permanent test environment Test environment available on demand Test database
	Testware management	4.4.3	Calibrated regression test set
		4.4.4	Test quality improvement in ad hoc maintenance
		4.4.5	Adjustment regression test set
	Know-how is essential	4.4.1	Kick-off
		4.4.2	Reproduce defect
		4.4.3	Specialist knowledge
	Reduced motivation of testers	4.2	Planning, priority, structure Automate, outsource, test organisation
	Scope of maintenance	4.4.1	Kick-off Test strategy
	Maintenance influence on specifications and test cases	4.4.1	Kick-off
		4.4.2	Test base by communication Reproduce defect
		4.4.3	Specialist knowledge
		4.4.5	Change regression test set
	Improper test tasks	4.1	Advice

4 TESTING DURING MAINTENANCE

In the previous chapter some issues concerning testing during maintenance were mentioned. In this chapter solutions, tips, hints and additional activities for these are suggested.
Since testing during maintenance is not always carried out within a project, test aspects could be mentioned that can be organised beyond the project organisation.
In the first few paragraphs we will focus on the organisation of:

- A standard test process;
- The test organisation;
- The test infrastructure.

After that the differences to the standard test process in the TMap® life-cycle model are explained [Aalst-2 2006]. The main focus in this is on planable maintenance types, because it is the most common form of maintenance. However, for ad hoc, mostly corrective, maintenance exceptions need to be made. Because of that special attention will be given to those exceptions when relevant.

4.1 Standard test process

Specific process

In general it can be assumed that the standard TMap® approach can be implemented completely. It is important to keep in mind with this that testing remains to be a discipline with its *own objectives and responsibilities*. If a mixture of specific service management tasks and test tasks is desirable, it has to be a conscious choice, because of the risk that the test tasks become less important than the service management tasks. To prevent testing from becoming an isolated activity, it is important that test activities are thought through well and put down in writing within the change procedure as well.

Advice

The ultimate goal of testing is to give an *advice with regard to the quality and risks*. In order to get to the advice some differences and similarities between planned and ad hoc maintenance can be indicated.
In case of planned maintenance testing mainly focuses on the quality of implemented changes or new functionalities.
In ad hoc maintenance testing mainly focuses on defect repairing and restoring of possible consequential loss.
Besides this, in both planned and ad-hoc maintenance it should be tested if any (new) defects are introduced. To do this sufficiently it is important to have a proper regression test set available. The value of such a set is higher, if it is composed based on an executed risk analysis. If, in addition, this set is set up modular and calibrated, it can even be tested specifically in an ad hoc maintenance situation by choosing the relevant tests, coverage and depth. In paragraph 4.4.3 "specification phase" an approach is described to be able to build such a calibrated regression set. More details on how to determine how thoroughly the tests should be can be found under 'test strategy' in paragraph 4.4.1 'phase planning and control'.

Standard maintenance test plan

It is advisable not to invent a new way of working again and again. Both the test process and reusable test aspects such as test specification techniques, required infrastructure, organisation, communication, procedures and regression test set can be worked out and written down once in a *standard maintenance plan*[1]. A (limited) test strategy needs to be included in this as well. By, for example, performing a risk analysis on a system, the

[1] A standard maintenance test plan can have several forms and names. Examples of these are: generic test agreement and generic master test plan

risky and less risky parts can be distinguished. A change on a risky system part requires more test effort than on less risky parts.
Since the impact of a good test strategy is of great influence on the quality of the test as a whole, this subject will be further explained in paragraph 4.4.1 'phase planning and control'.
In case of ad hoc corrective maintenance, solving the production problem has top priority. Even though this results in not taking all necessary steps of a structured test approach, it is vital just then to have a standard maintenance test plan available, since it describes which test activities are essential in an ad hoc situation and should always be performed. If, besides this, a calibrated regression test set is available that can be 'easily' adapted to the test strategy, it is possible to perform a high quality test, 'even' in an ad hoc corrective maintenance situation.

4.2 Test organisation

Planning, priorities and structure of service management organisations
In contrast to a project, during which management and control of activities often is very well organised (team leaders), it is a lot more complicated to achieve this in a maintenance organisation. In a maintenance organisation various groups can be performing maintenance on different systems or system parts at the same time. In those situations it is not unthinkable that the control of the line manager on the test activities is ineffective. That will be even more likely to happen if there are no separate testers present for maintenance. In that case it is advisable to appoint an employee, for example a system analyst, as group leader for each group, and make him/her responsible for the quality of the delivered products. This employee will supervise his/her colleagues when they perform test (related) activities.
Since testing during maintenance is mostly not executed within a project organisation, employees from the various service management organisations are seldom available to perform tests on a full time basis. As soon as a production problem occurs, test activities suffer the consequences. After all, solving the production problem is most important and apart from that the technical staff usually enjoys solving problems more than conducting boring and mostly repetitive test work. In general it could be said that the 'average' service manager has little affinity with testing. It would be recommended to discuss in advance with all parties involved how to act in such situations.
Draw up *an explicit planning* for example and lay down the *priorities* of the employees and their tasks, like preparing and executing the tests in 'tandems'. As soon as a production problem arises, this tandem can be temporarily dissolved. One of them can focus on solving the production problem, while the other keeps concentrating on testing.

Obviously the structure of the service management organisation needs to be taken into account. Not every organisation has structured these in accordance with the layout outlined in chapter 2 on 'service management and maintenance'.
In the standard maintenance test plan the test levels can be named, and who, at which department, is responsible for the interpretation and execution of these. In general the *user organisation* is responsible for the (user) acceptance test, the *infrastructure organisation* for the production acceptance test and the *maintenance organisation* for the unit, integration and system test. See figure 2, relation between service management forms and test levels.

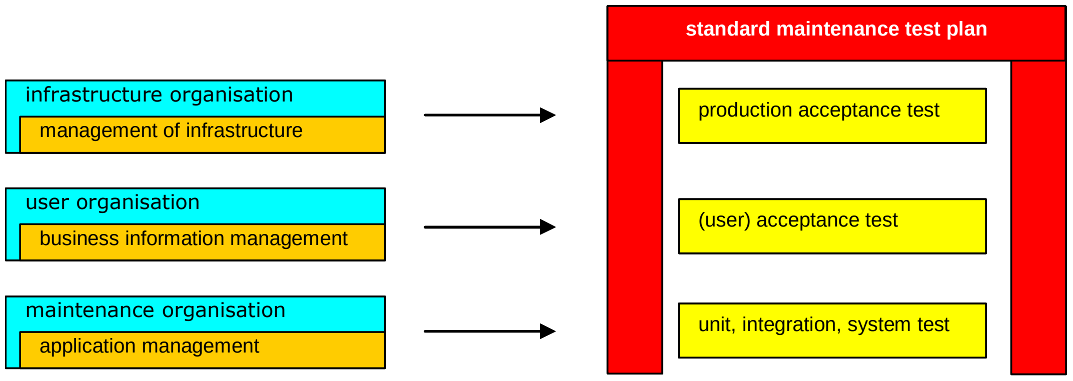
standard maintenance test plan
infrastructure organisation
management of infrastructure
user organisation
business information management
maintenance organisation
application management
production acceptance test
(user) acceptance test
unit, integration, system test

Figure 2, relation between service management forms and test types.

Test automation, outsourcing and separate test organisation

Several solutions are imaginable to minimise the test work that is boring and repetitive in the service manager's view. It can be done, for example by *automating* the regression tests or *outsourcing the test work*. Especially the latter solution is chosen more regularly by many organisations, possibly as a part of outsourcing all maintenance.
If outsourcing is (for now) too big a step, some service management organisations decide to set up a *separate test organisation* [Aalst-1 2010]. Possible considerations to make that decision are for example: size of the organisation, specific risks, impact of the problem or change, availability of qualified personnel, system availability, available knowledge on testing and subject matter, complexity of the systems and expected work load [TestNet 2005].

4.3 Test infrastructure

Permanent test environment

Anyone familiar with testing knows that the availability of a good test infrastructure is of the utmost importance for tests of high quality. It is advisable to set up a *permanent test environment* for systems that are frequently changed. In those situations it may also be advisable to investigate whether or not *automation of the regression test* is desirable. For systems that are less frequently changed, a test environment that is *available on demand* might be sufficient.
There can be more than one permanent and "on demand" test environment for that matter. For example a separate environment for the maintenance organisation and an acceptance test environment for the user organisation.

Test environment available on demand

Special attention should be given to the situation of ad hoc corrective maintenance. It frequently happens that 'patches' that have at best gone through a unit test are implemented in the production environment. It is often, however, possible and advisable to test this 'patch' in the acceptance test-, training- or fall-back environment.

Test database

A well known, recurring subject of discussion is whether or not the use of production data is possible, desirable, necessary or undesirable. To have as much as possible a true-to-life rendering of the production situation (especially important when end users will execute the tests), a widely used method is to copy the *production database*[2], after which it is adjusted and/or scaled down. The disadvantage of this method is that it can be difficult to discover the exact content, and that the database is filled with common data for approximately 95%. These disadvantages do not occur in a test database that is

[2] Privacy regulations have to be considered

built from ‘scratch’. In that case the exact composition of the test data is known, and besides e.g. 80% common data, 20% exceptions would be included. A complicating factor could be keeping this database up to date. Performing daily, weekly, monthly runs (etc.) could be considered for this. Which choice is best will vary from situation to situation.
The choice is more important even when testing is automated or will be in the near future. In that case the availability of a stable and known initial database is inevitable. This may create an area of tension with the alternative of testing with a recent, up-to-date database. This area of tension will be smaller if automated testing is or will be set up in as much a *data independent* way as possible.
Test lifecycle

4.3.1 Phase planning and control

Fixed and variable test costs

A common image problem testing has is that testing always takes too much time in the perception of management. This is even more so for testing during maintenance, because testing takes a relatively large share in the maintenance path. It is of importance to explain the reasons for this to the customer. It should be realised that the total test effort consists of *a fixed and a variable* part and consequently of fixed and variable test costs. Fixed can for example be the time needed to get the test environment up and running or the execution of a ‘standard’ regression test, and variable could be the preparation and testing of an implemented change. In case a small change is tested, the percentage fixed test time is high. As the size of the changes increase, the percentage of fixed test time decreases.
An example: While testing a change a 4 hours regression test is always performed. If the implementation of the change takes 8 hours in total, the percentage of fixed test time is 50% (4/8). If implementing one or more changes takes 40 hours, the percentage is lowered to 10%. See figure 3, relative share of fixed test time, for a graphical reproduction.
It is of importance to gain commitment from the customer on the test approach in advance. That way no discussions will arise afterwards about the relatively large test share during the maintenance path. Collecting evidence on the ratio between testing and remaining maintenance activities can provide the customer with clear insight. In general the share of all test activities (fixed and variable) ranges from 35% to 80% of all maintenance activities. It needs to be taken into account with this that an 80% test share can involve fewer absolute hours than a maintenance path with a test share of 35%. This mainly depends on the total size of the maintenance path.

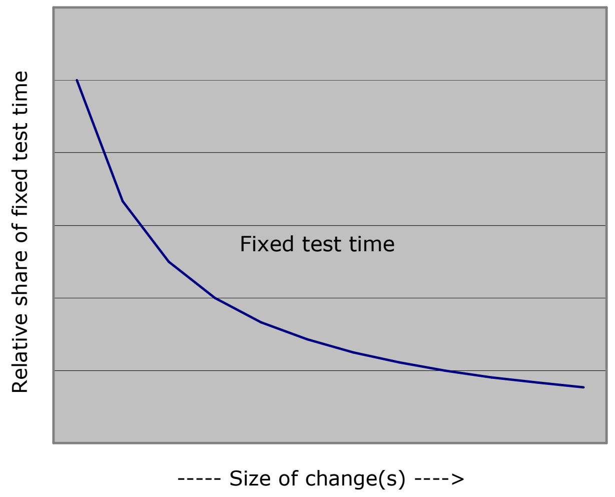

Figure 3, relative share fixed test time.

Anyway, in this phase a lot of time and quality can be gained if a *standard maintenance test plan* is present, and with that the next test phases can be passed through more quickly and efficiently. Actually, one of the key principles of the TMap® life cycle is removing the test activities from the critical path as much as possible, to ensure that only the test execution remains there.

Kick-off

When making an inventory of the test basis it sometimes happens that changes are not very well documented and in case of ad hoc maintenance that the actual cause for the change is not described. A well-tried method to gain clarity on this is to organise a *kick-off session* with all parties concerned (business information management and management of technical infrastructure, developers, users and testers). Defining the impact, signalling risks and defining and/or adapting the test strategy taking the non-functional quality attributes into account as well, are topics that fit very well into such a kick-off session. Keep the existing situation into mind while collecting non-functional quality attributes. Improving for example a performance requirement from three to one second is very hard to realise in a maintenance release, if this was not a prerequisite in the original design.
Specifically in case of ad hoc maintenance the kick-off session can be used to discuss ways to reproduce an error in a test situation.
The challenge in organising a kick-off session is getting the desired participants together at the same time, in view of the usually very limited timeframe for testing.

Delivery by release

Another phenomenon that occurs regularly in testing during maintenance is the scattered delivery of partial solutions to the testers, for example in forms of separate programs or system parts. This way, testers are forced to keep a detailed test administration to be able to examine which partial solutions have to be put together to be able to test a certain change. Using this method errors are more likely to occur and a higher number of retests may be needed. It is better to agree with the maintenance organisation in advance on the manner in which releases are delivered. A *delivery by release* is a good solution for this. This may also reduce the fixed test time since the regression test can be planned only once for the release as a whole.
Maintenance is not always performed in the line organisation. Considerations to range maintenance in a project are for example the amount of money, time and resources involved. Furthermore the number of disciplines/roles involved and the possible risks an

adjustment brings along are considered. By taking up maintenance in a project it is possible to work more according to plan, clear agreement on budgets can be reached and decisions will be taken on management level.

Test strategy

The standard maintenance test plan should describe how the test strategy in a maintenance situation should be defined. In the standard maintenance test plan risk levels can be appointed to system parts (for example based on the complexity and significance to the business of the system part). Next, a risk analysis for each change request is executed, based on the complexity of the change and the developer's experience. In case it is not clear in what parts the system has been changed, the new source code can be compared to the old to detect code changes.
Since there is a (test) difference between new development and maintenance, the steps to take within the risk analysis are influenced by it.
The most important difference between the test strategy for new development and maintenance is the *fault chance*. A number of changes are implemented on an existing system, mostly as a result of problem reports or change requests. These changes can be incorrectly implemented and need to be tested. With the adjustment there is also a minor chance that faults are introduced in the unchanged parts of the system, as a result of which the system deteriorates in quality. This phenomenon of quality deterioration is called regression and is the reason why the unchanged parts of the system are tested as well.
This distribution of faults during maintenance has as a result for the strategy development that the risk classifications of subsystems can differ from those of new development. A high-risk subsystem in new development may be unchanged in the maintenance release. Because the chance for regression is the only risk involved, a less thorough test can be performed. For that reason test strategy determination of a test level for maintenance can be modified by changing the concept of 'subsystem' to 'change' in the steps of the strategy determination. This type of strategy determination is called (test) impact analysis. Per change (an accepted request for change or a solved problem report) an inventory is done on which system parts were changed, which system parts may have been influenced by the chance and which quality characteristics are relevant.
There are various possibilities for testing each change, dependent on the risks:

- a limited test, focused on the change only;
- a complete (re)test of the function that is changed;
- the testing of the coherence of the function that is changed and the adjacent functions;
- to test the entire system.

The regression test of the system is also recognised as a whole. The regression test mostly focuses on the coherence between the changed and the unchanged parts of the system, since these are most likely to suffer from regression. If the test strategy for the new development is available, the levels of importance attributed to the subsystems play a role in the construction of the regression test. A regression test can be executed in part or in full, depending on the risks and the test effort required. The use of test tools is most effective in the execution of regression tests. The main advantage of automation of regression tests is that a full test can be executed each time with limited effort and it is not necessary to decide which parts of the regression test will, and which will not, be executed.
The choice to formulate the strategy either in terms of subsystems or change requests is affected by the number of change requests and the system part affected by the changes. The more changes and the larger the part of the system affected, the stronger the preference goes to determining the test strategy at the subsystem level, rather than basing it on change requests.
If it is decided to base the strategy and budget on change requests, the following steps are to be taken:

1. Collect changes (both accepted change requests and solved problem reports);

2. Determine the impact per change (analysis of subsystems affected);
3. Determine the damage at failure (high, medium, low), per change;
4. Determine the chance of failure (high, medium, low), per change;
5. Determine risk category (A, B, C) per change [Baarda 2005];
6. Group the changes by risk category and add the regression test;
7. Determine the size per change (and regression test).
8. Determine test effort per change, based on the size and risk category of the change (and regression test) involved.

An example of a "strategy matrix" following step 6:

	Risk category	**Size**	**Test effort**
Change request 1	A		
Change request 4	A		
Regression test whole system	A		
Solved defect rapport 2	B		
Change request 3	B		
Solved defect rapport 1	C		
Change request 2	C		

Size (design and development) and test effort in hours.

In the example above risk category A is assigned to the regression test. This means that executing a regression test is considered to be of great importance. In general, the total effort of a regression test is often much larger than the test effort required for detailed testing of the changes. This is caused by the fact that during maintenance usually just a very limited number of functions are changed and the regression test covers the entire system. Using the calibrated regression test set (see paragraph 4.4.3 specification phase), another strategy can be chosen, for example by choosing a regression test set with less coverage, less depth, etcetera.
The largest part of maintenance can be planned in advance and the strategy development described can be used just like that. For ad hoc maintenance it can be more difficult, when a production defect must be fixed and the system has to be up and running again as soon as possible. A formal strategy formulation often takes too much time. It is however possible to prepare a number of strategy blueprints in *advance*: if programme X fails and needs to be fixed, which items have to be tested? Only the changes or a (limited) regression test as well? Such scenarios support the best possible test in case of ad hoc maintenance.

4.3.2 Preparation phase

Test basis by communication

In this phase a testability review is carried out on the defect report or the change request. As mentioned before, it is not unusual that insufficient documentation is available for writing test specifications. The kick-off session can be used to clarify this, but it is preferable to take care of this at an earlier stage, for example by communicating in advance on what the testers expect to find in a change request. On the long term this may lead to a more structured approach of the documentation problem than discussing it during a kick-off session. Since it concerns maintenance on an existing system, it is often possible and desired to use the system itself as test base (reference).

Reproducing the defect

Next to testing the change requests it should also be tested if a problem is solved correctly. To be able to do that properly, it needs to be examined what the problem exactly was. In this phase the tester will have to try and *reproduce the defect*. In fact this is the only way, after implementing the solution ('patch'), to be able to determine during the test execution phase, if the defect is actually fixed. It is advisable to work on this together with the developer, since he is faced with the same challenge.

4.3.3 Specification phase

Subject matter expertise

It would have an optimal effect if the regression test set was transferred from the new development project to service management. Within the project this activity is planned and expertise on the subject matter is brought together. If this did not happen and the regression test set has to be put together in the maintenance situation, this often fails, due to available time, expertise on the subject matter and motivation. Delivery of a regression test set by the project should be in the acceptance criteria of the service management organisations.

While specifying the maintenance specific test cases at least the initial situation, the action and the expected result should be described. In the specification phase preferably 'only' the supplementary test cases are written, or the existing ones are changed, to be used to test the changes. A tricky point at this is that usually little time and insufficient expertise on the subject matter is present to specify test cases, and there is lack of knowledge of test specification techniques. It should therefore be recommended to collect as much knowledge of the change as possible and to secure the help of key users during the specification phase during the kick-off session. Besides specifying additional test cases, the impact of the changes on the regression test set has to be examined and this should be changed if necessary. In this situation you would want to have a calibrated regression test set ready as a starting point. During the kick-off session and/or the test strategy session it has been determined whether all or part of the test cases from this set should be carried out. How to set up and use a calibrated regression test set will be explained in the next paragraph.

Calibrated regression test set

A high quality regression test set is invaluable. Especially in the service management situation it is an archive of knowledge about the system and a source of ready to go test cases for regression tests.

In practice these test sets are often inadequate. Setting up a well-considered regression test set is easier said than done.

In this paragraph an approach is described to set up, use and control regression test sets based on the Test Cube principle, developed by Sogeti [Test cube 2005].

It describes a number of related principles that enables to:

- specify test cases and execute them based on priority;
- report quickly and adequately on the progress of test specification and/or execution;
- plan and budget test projects accurately;
- put together regression test sets variously and quickly;
- incorporate changes in the test base easily in the test set.

In concept, the Test Cube is not much more than a collection of complementary data that is recorded per test case: the test cases in the test set are 'ranked'. With these classifications, subsets of test cases can be selected by making various cross sections.

Examples of classifications are:

- application
- subsystem
- function

- product risk[3]
- risk category
- process (part)
- release
- requirement
- transaction
- weight category

The right selection of classifications and being able to correctly classify test cases is essential for the use of this concept.
In maintenance situations the classification according to weight category is essential. This classification indicates the 'weight' of a test case within the test set. This classification in particular provides the ability to perform a risk based regression test with flexible depth.
The use of these weight categories, for example by putting together regression sets, is described below (by classification into three categories):

- By selecting only test cases of category 1 for a subsystem, a small regression test set is created. This subset is used for an unchanged subsystem (or to perform an intake test on a new or radically changed subsystem).
- Test cases of category 2 (= including category 1) deliver a standard regression test set, for example for a subsystem that has been changed.
- Test cases of the third category (= including category 1 and 2) cover the entire subsystem and are used for new or radically changed subsystems.

The Test Cube concept does not require fixed test specification techniques. All existing techniques can be used to specify the test cases in the test set. Choosing a technique is done in the usual way: by defining a test strategy.
There are no requirements either for the level of detail. When the tests are expected to be performed by testers that lack expertise on the system, the test cases can be described in more detail.
The (Test Cube) concept has specific demands on only one aspect of the test cases. All test cases need to be independent. This is the so-called independence principle of the concept.

- No dependencies between test cases are allowed. When executing one test case is conditional for the execution of the second one, problems arise when test sets are chosen from the regression test set. This may cause all kind of test cases that have to be executed first, that are not part of the chosen test set.
- It has to be possible to execute individual test cases at the same time. Test cases that require exclusive use of the test environment hamper the execution of others. This hinders the planning of the timeframe required for the test.

While using this concept the size of the (regression) test set and its related activities within the test project are specifically measurable. By administrating some statuses for every test case, at any moment desired an up-to-date report on the progress can be given. If metrics are collected of past and current tests, a more accurate planning can be made for future tests or the continuation of current activities.

For a more detailed description of the concept, turn to the white paper on the Test Cube [TestKubus 2005].

[3] To determine and/or classify product risks it is recommendable to follow the operating procedure as described in the business driven test management approach [Baarda 2005]

4.3.4 Execution phase

The activities that need to be performed in this phase for testing during maintenance are no different from those during new development. In this phase the changed software and possible recovery programs are tested.

Test quality improvement in ad hoc maintenance
In case of ad hoc corrective maintenance the phase Execution seems to be the only phase, because all test activities are often executed at the same moment. This is, of course, due to defects that require instant solutions. Examples of this situation are a production run that dumps late at night, hundreds of users using a network that 'gets stuck', a mailing with incorrect addresses, etcetera. When solving these kinds of problems, different criteria and procedures are used. As mentioned before, it will not be possible to follow all prescribed steps of a structured test approach.
However, by selecting the relevant test cases that relate to the problem from the calibrated regression tests set, based on the already *present risk analysis* in the *standard maintenance test plan,* the possibility exists to (quickly) perform a limited regression test. That way, besides testing if the problem is in fact solved, it can also be tested, with limited effort, if no new defects have been introduced. For that matter, if the risk is limited, it can be decided to test afterwards (e.g. the next day).
With regard to ad hoc maintenance it is therefore possible to realize a quality improvement using a predefined test approach, where by it is desired to go through all test phases (however minimal they are). It is important to perform a proper risk analysis with regard to the information system beforehand and that the results are described in the standard maintenance test plan. Based on those results correct classifications and values have to be put into the calibrated regression test set as well.

4.3.5 Completion phase

Adjust the regression test set
Keeping the regression test set up to date is likely to be 'forgotten'. It is therefore advisable to 'standard' incorporate this activity in this test phase. While executing the test it may have happened that the system did not react in the way assumed in the test case. If that was an incorrect assumption, the test case has to be adjusted in accordance with the production situation. Furthermore, a decision has to be made if, and if so, which, new test cases need to be added to the regression test set. This can be done simply by using the changes and possible defect reports as basis. It has to be taken into account that the correct classifications used for the calibrated regression test set have to be added.
Besides test cases resulting from planned maintenance paths, they can result from ad hoc corrective maintenance as well. Although it is often stated in the latter case that it 'just' concerns the solving of a problem for which the functionality remains unchanged, and therefore does not have to be included in the regression test set, it can be useful to include them for the following reason: the problem was solved in one particular software version, but the change has to be implemented in the following software versions as well. It often happens that this does not happen for some reason, and therefore it is advisable to add a specific test case to the regression test set.
Finally it could be desirable to adjust the risk analysis in the standard maintenance test plan. This mainly applies to the chance of failure of the tested subsystems, which can be adjusted based on the number of detected defects.

5 REFERENCES

- [Aalst-1 2010]
  Aalst, L. van der, (2010), *Test Service Centre, a dedicated line organisation for testing*, www.tmap.net, Sogeti white paper

- [Aalst-2 2006]
  Aalst, L. van der, Broekman, B., Koomen, T., Vroon, M. (2006), *TMap Next, for result-driven testing*, 's-Hertogenbosch: Tutein Nolthenius Publishers, ISBN 90-72194-80-2

- [Baarda 2005]
  (editors) Rob Baarda, Tim Koomen, (2005) *TMap Test Topics*, (chapter 4), Tutein Nolthenius, Nederland, ISBN 90-72194-75-6

- [Delen 1992]
  Delen, G.P.A.J., Looijen, M., (1992), *Beheer van de informatievoorziening*, Rijswijk, Cap Gemini Publishing (in Dutch)

- [Grady 1987]
  Grady, R., (1987), *Software Metrics: Establishing a Company-Wide Program*, Prentice-Hall

- [Lint 2005]
  Lint, P. van., Kortman, M., (2005), *Sjabloon Generieke Testafspraak*, Rotterdam, versie 2.0, Sjabloon ontstaan uit samenwerking van Sogeti met een klant, (in Dutch)

- [Tamai 1992]
  Tamai, T., Torimitsu, Y., (1992*), Software Lifetime and its Evolution Process over Generation*, Proceedings of 1992 Conference on Software Maintenance

- [TestKubus 2005]
  Delprat, H., Jelgerhuis, R., Suiveer, M., Verweij, A., Wester, G., Winckelmann, P. von (2005), *De TestKubus, aanpak voor het structureren en beheren van testsets*, Diemen, versie 1.0, Sogeti white paper (in Dutch)

- [TestNet 2005]
  Thema-avond TestNet, (2005), *Testen in onderhoudssituatie*, 23 februari 2005 te NBC Nieuwegein (in Dutch)