



White Paper

Requirements-Driven Testing

Jeremy Dick

Version 3

22 October 2004

This document contains proprietary information that belongs to Telelogic AB. Using any of the information contained herein or copying or imaging all or part of this document by any means is strictly forbidden without express written consent of Telelogic AB.

Telelogic[®], TAU[®] and DOORS[®] are registered trademarks, SYNERGY[™] is a trademark of Telelogic AB – www.telelogic.com

Table of Contents

Introduction	4
What is requirements management?	4
What is testing?	4
Some principles of requirements-driven testing.....	5
Plan tests early	5
Conduct tests early	5
Relate tests to requirements	6
Relate defects to requirements	6
Measure progress against requirements	7
The W-Model	7
Roles in Development	14
Role: Manager.....	14
Role: Analyst	14
Role: QA/Tester.....	15
Role: Developer	15
Integrating Requirements Management, Test Management and Defect Management.....	16
The DOORS - TestDirector Integration	17
Analyst.....	18
QA/Tester	18
Manager	19
Integrations Administrator	19
Overall benefits:	19
The SYNERGY - TestDirector Integration	20
Conclusion	21
About Telelogic.....	22

Overview

This paper outlines the benefits of *requirements-driven testing*, combining best practice in requirements management and test management from a process perspective.

The principles that should guide such process integration are reviewed, and the W-Model introduced as an evolution of the classic V-Model. In this model, the concept of the *qualification plan* is used to bridge the disciplines.

Various roles and the kinds of activities they undertake in the process are identified and described.

Finally, integrations between the requirements management tool, DOORS; the test management tool, TestDirector; and the defect tracking tool, SYNERGY, are described as an example of how requirements-driven testing can be supported.

Introduction

What is requirements management?

A requirement is an assertion that must be satisfied by a product or a process.

There are many kinds of requirements, and many words used to describe them. For example, each of the following may be viewed as a requirement: aim, aspiration, condition, contract, constraint, goal, mandate, need, objective, obligation, regulation, requisite, rule, specification, target, and many others. We choose to use the word *requirement* as a synonym for all of these concepts, and we use it to refer to an individual statement, clause or item rather than for a whole requirements document.

Requirements management is the set of disciplines and activities concerned with the capture, formulation, organization, versioning, publishing, tracing, analysis and change of requirements.

Many people mistakenly believe that requirements management is something that takes place during the definitional stages of a project, and is then finished. In fact, requirements exist in one form or another at every stage of development. In particular, they play a vital role in the various stages of testing, even towards the end of development.

Three key disciplines in requirements management are pertinent to this paper:

- *How requirements are expressed.* Not only should requirements be concise and unambiguous, but they should also be testable. This usually means making sure that requirements are properly quantified, and that test criteria can be clearly identified. This test criteria provide essential information used during testing.
- *How requirements are classified.* Not all requirements are born equal. Some are far more significant or important than others. For effective management of development, each statement of requirement has to be classified, in whatever way is most appropriate to the application, so that engineering and managerial judgments can be made as to the best allocation of resources.
- *How requirements are traced.* Creating traceability is a vital part of requirements management. Being able to trace requirements up and down through the stages of development, and to trace tests backwards and forwards to their associated requirements, allows progress to be reported in various ways, and the impact of change to be analyzed.

What is testing?

Testing is any activity aimed at revealing defects.

This is deliberately a very broad definition of testing. It covers all sorts of early activities such as requirements reviews, design inspections, structured walkthroughs, analysis from modeling, as well the traditional unit, system and acceptance testing stages.

For many people, this definition is too broad, because it departs from the traditional positioning of testing firmly on the right-hand side of the V-model. But the definition admits that requirements and designs themselves can be “tested” (reviewed, validated, verified, qualified, or evaluated) before anything has actually been built or coded.

In keeping with the breadth of this definition, we have selected the word *qualification* to cover testing, validation, verification and evaluation. Thus “test criteria” become “qualification criteria”, and “test plans” become “qualification plans”.

A defect is any departure from a requirement.

Defects are recognized, ideally, with reference to a requirement. There are requirements directly on the product (originating from customers) and requirements on the development process (standards and procedures, etc.). The latter encapsulate best practice, in turn designed to reduce the likelihood of introducing and propagating defects.

Some principles of requirements-driven testing

There are some principles of testing that should be respected. They represent, at a high level, elements of best practice in requirements-driven testing.

Plan tests early

Plan tests for each requirement as the requirement is written.

Considering testing at the same time as capturing requirements improves the way the requirement is expressed by encouraging consideration of how the requirement is quantified. For every requirement, the question should be asked:

How will we know if this requirement will be, or has been, satisfied?

Note the use of both future and past tenses in this question; early tests, such as design inspections, check that when the product is built according to the design, it *will* meet requirements; later tests check that requirements *have been* satisfied by what has been built.

The above question should lead to a *qualification* strategy for each requirement. Because of the broad range of qualification activities available, every requirement will give rise to multiple tests covering all stages of development. For instance, an early opportunity for qualification is to check the design against a requirement. Are the necessary elements present in the design to ensure that the requirement will be met? In addition, there will be one or more system tests to ensure that what is finally built really does meet the requirement. The set of selected tests form the qualification strategy.

The question should also lead to the identification of *qualification criteria* for every test. These formalize what is considered a successful outcome for each test. Qualification criteria vary with the nature of the requirement and the planned test.

Conduct tests early

Perform tests as early as possible in the development process.

“A stitch in time saves nine.” Defects propagate themselves from stage to stage, and the cost of correcting them geometrically increases through the development process. Because of this, development cost and schedule can be dramatically reduced by identifying and correcting defects as early as possible, and thus minimizing expensive rework in later stages.

The qualification plan for each requirement should therefore consider the earliest possible means of detecting defects. Examples of early testing are reviews, inspections and walkthroughs.

Relate tests to requirements

Trace tests back to the requirements they are designed to check.

Tracing is a means of documenting the relationships between artifacts during development. The most common form of tracing is the *satisfaction* relationship between layers of requirement. However, the *qualification* relationship between requirements and the tests that show that they have been met is also important to capture.

Tracing enables two kinds of analysis to be carried out:

- *Coverage analysis* is used to ensure, for example, that every requirement has at least one qualification action planned and eventually executed. It can also be used to ensure that every qualification action has an associated requirement, and, therefore, provides benefit.
- *Impact analysis* is used to determine the qualification actions that may need to be revisited when requirements change. Before accepting a change to a requirement, the cost of redefining and re-executing its associated qualification actions is taken into account. Similarly, when a test fails, the impact of the failure can be viewed in terms of the requirements.

Impact analysis uses a combination of the *satisfaction* and *qualification* relationships to determine potential rework. For instance, if a component test fails, it not only affects the associated components requirements (through the *qualification* relationship), but also the sub-system requirements that those component requirements are supposed to satisfy (through the *satisfaction* relationship), and, in turn, the satisfied system and customer requirements.

Testers understand well that testing at every level is necessary. For instance, a sub-system requirement cannot be fully tested by just testing its components; the sub-system (the integration of those components) still needs to be tested. This aligns with the concept of *emergent properties* in requirements management: a system is more than the sum of its parts, and some behavior arises from the interaction of components.

Relate defects to requirements

Trace defects back to the requirements that they show are not satisfied.

When a test reveals unexpected behavior, or that test criteria are not met, defects are raised. There are three possible causes of defects:

- There is an error in the definition of the test.
- There is an error in the expression of the requirement or its qualification criteria.
- There is a genuine defect in the product.

In every case, a defect is identified by tracing it to the requirements that are shown not to be satisfied. This is a good discipline, because a defect is defined as a departure from requirements. Since each test may be traced to multiple requirements, analysis of each defect is needed to determine precisely which requirements are affected.

Measure progress against requirements

Set targets and measure progress of testing in terms of those requirements that are shown to be satisfied or not.

One of the most difficult judgments to make during development is deciding when enough testing has been done. Limited resources are available, and the test manager needs to know where to apply effort most effectively. When testing is isolated from requirements, information about the relative importance of different aspects of the system is not available to inform the test manager, making it hard to allocate resources and difficult to assess the impact of failure.

With the appropriate traceability in place, the relative importance of each test, and the success and failure of tests, can be related to the requirements affected, at whatever desired level — customer requirements, system requirements, etc. Targets for completed testing can be set in terms of requirements, the test manager knows where to apply resources, and reports can be generated that show the progress of testing in requirements terms.

The W-Model

As we consider how best to implement these principles, a process and data model emerges called the W-Model, an evolution of the V-Model. The classic V-Model is illustrated in Figure 1, in which the rounded boxes represent activities through the life-cycle. (This model should not be interpreted as a “Waterfall” model, since all these activities may take place in parallel, with constant feedback.)

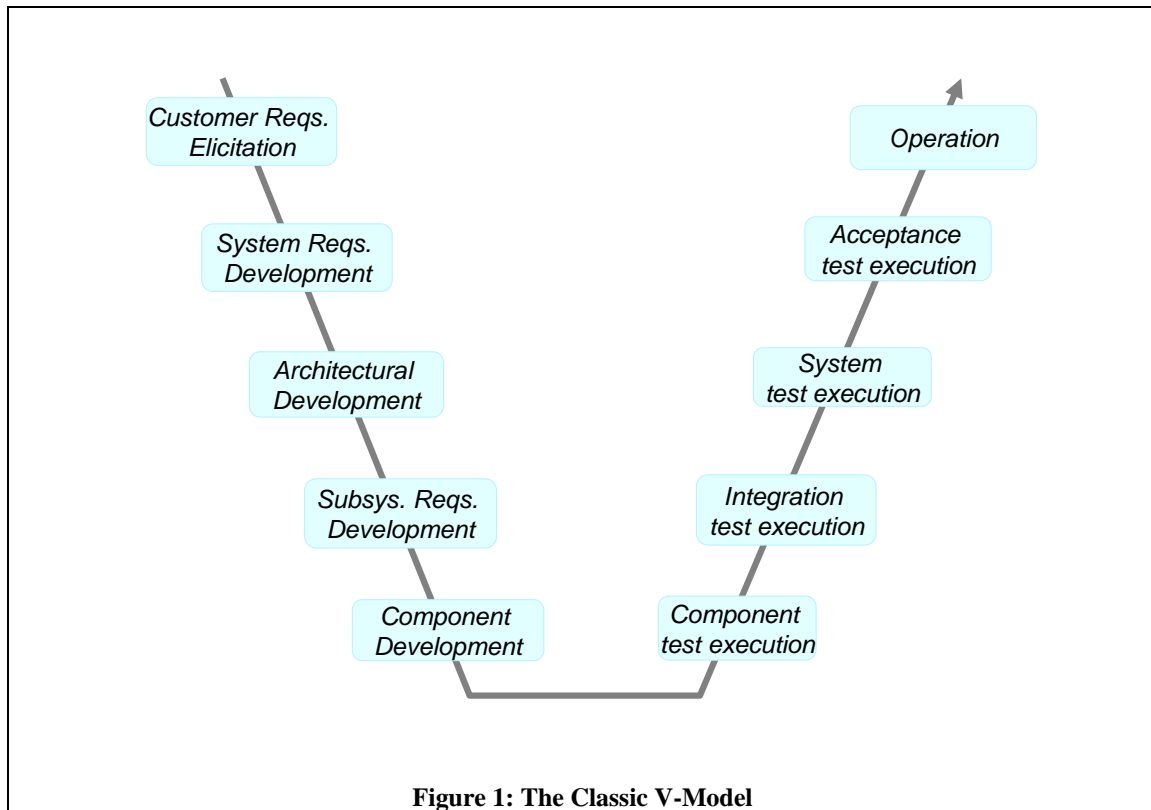
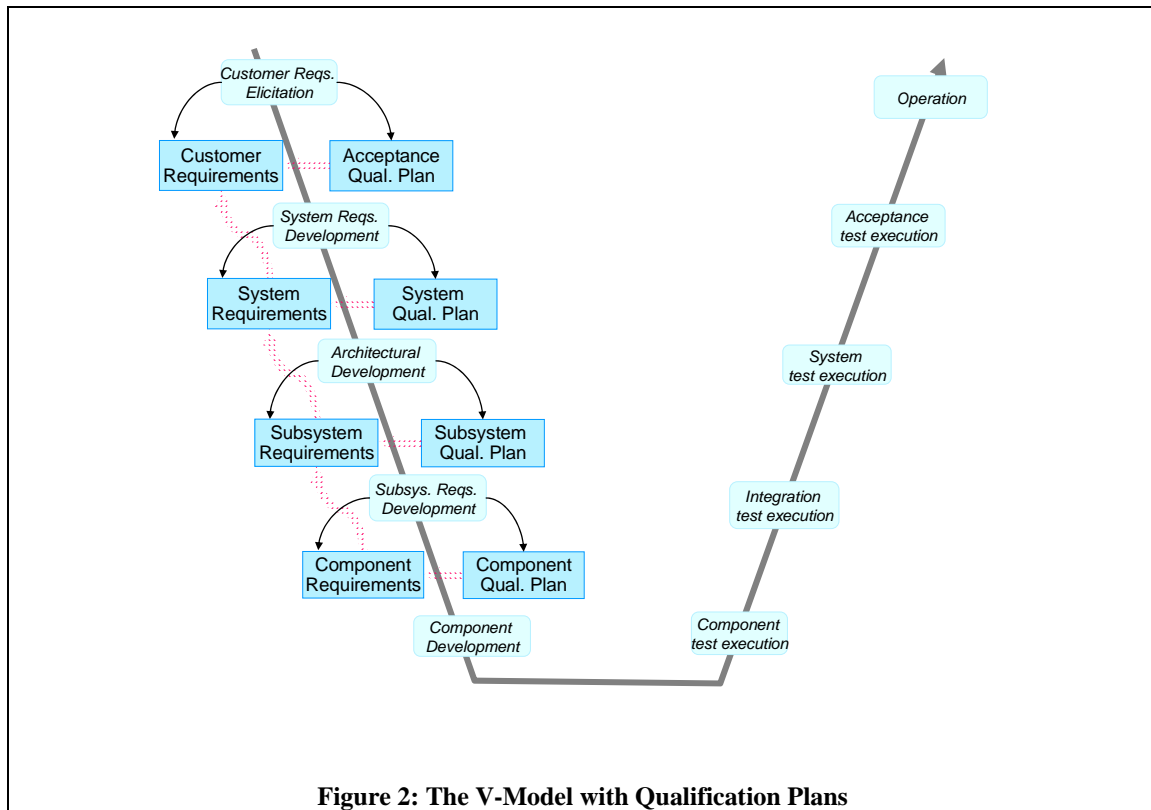


Figure 2 adds the data produced by the activities, represented by rectangles, and the tracing between the data, represented by thick arrows. This model places qualification plans parallel to the requirements on left-hand axis of the V-Model. The qualification plans contain just the planned qualification actions and the qualification criteria. The actual results of the actions are collected at the time the actions are executed at the appropriate stage of development.

Tracing is used to document two relationships:

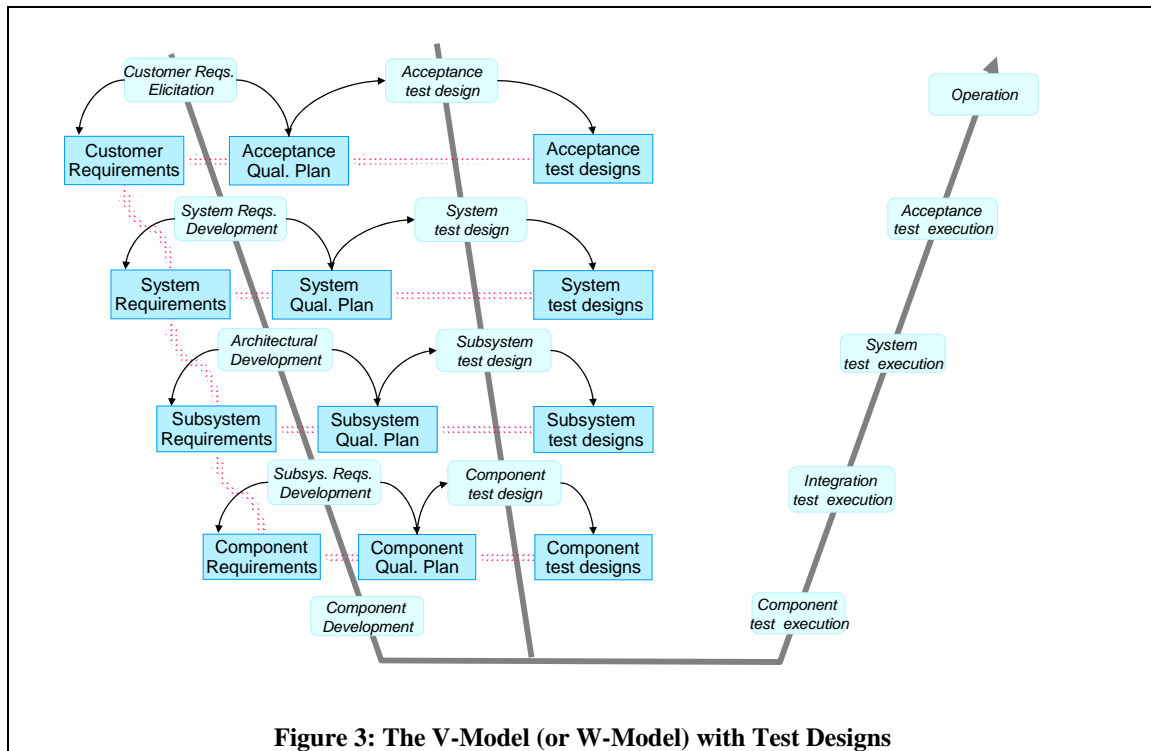
- *Satisfaction* between layers of requirements.
- *Qualification* between planned qualification actions and requirements.

Each qualification action — review, inspection, unit test, integration test, etc. — identified in the qualification plans can be scheduled to be performed at the appropriate stage of development. Together they form a “Qualification Schedule” for the whole project.



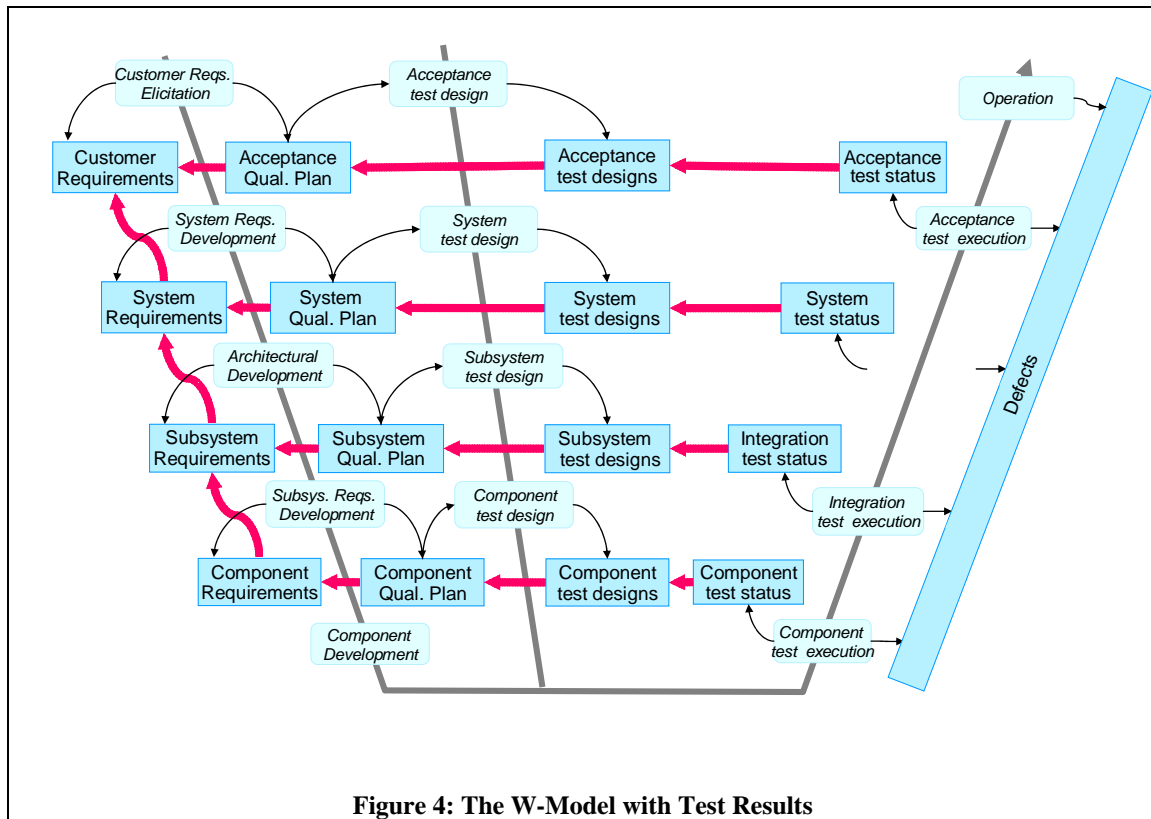
At the time that tests are planned, there is not enough information to design the exact details of the test. So Figure 3 shows test designing as separate activities at each level. These activities lie on a parallel axis to the development axis. Due to its underlying shape, this is sometimes referred to as the W-Model.

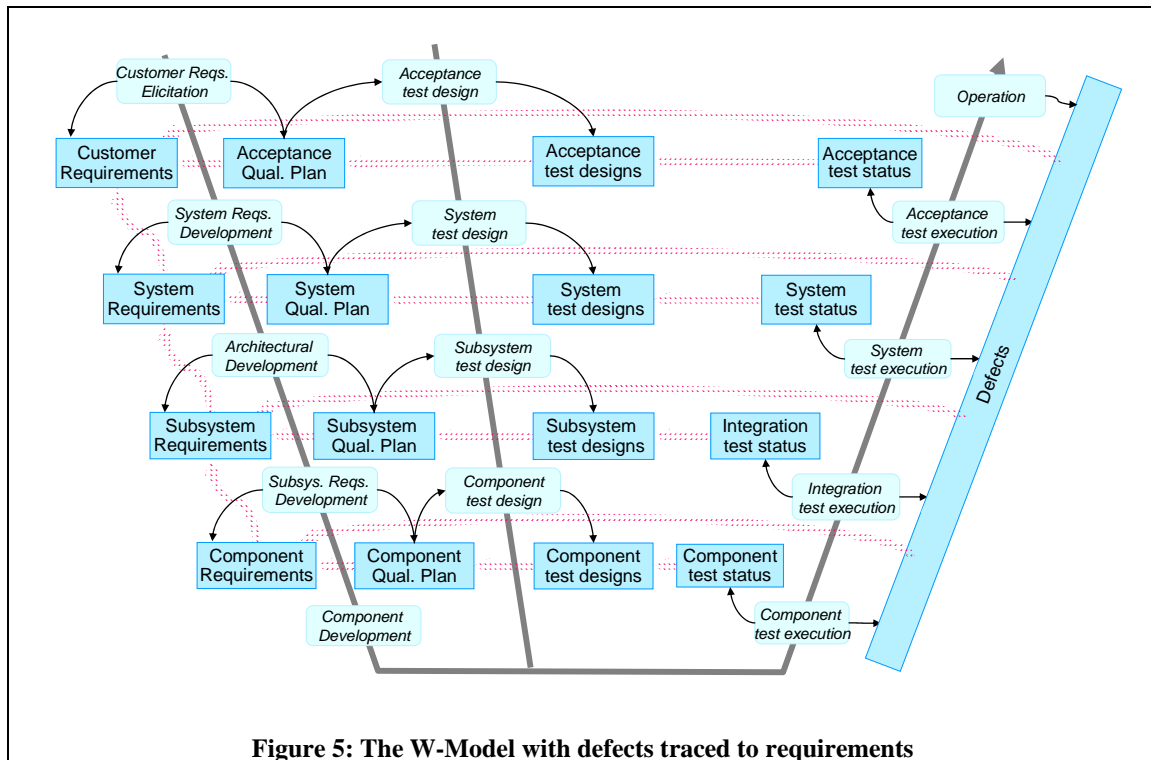
Test design uses the qualification plans to devise specific tests, which are traced back to the plans. Several tests may be designed for each plan, or a single test may cover several planned tests.



We can now add to the model the data that results from executing the tests. We are interested in test status and in the defects that are raised from tests, as illustrated in Figure 4.

All defects are placed in a single repository, since there is no neat assignment of defects to layers of development. The test execution activities include the analysis of defects to determine whether there is a genuine product defect, and which requirements are concerned. The result of this analysis is documented by tracing the defects to the affected requirements, as illustrated in Figure 5.





This approach accommodates the three principles outlined above as follows:

- *Plan tests early.* By writing the qualification plans in parallel with the requirements, consideration is given to testing as the requirement is written.
- *Conduct tests early.* The mapping of qualification plans onto the Qualification Schedule encourages the consideration of the earliest possible qualifications and tests that can be performed on each requirement.
- *Trace test and identified defects back to the requirements.* The tracing model allows this relationship to be maintained and analyzed.

Note that, to determine whether an individual requirement has been fully realized, both the satisfaction and qualification relationships are used to gather the complete set of test results.

Finally, testing tends to be highly iterative. Testing is performed, defects are found and then corrected, and tests are repeated. Figure 6 represents multiple test runs against the same criteria. Through the sequence of test runs, the test results evolve and progress is determined.

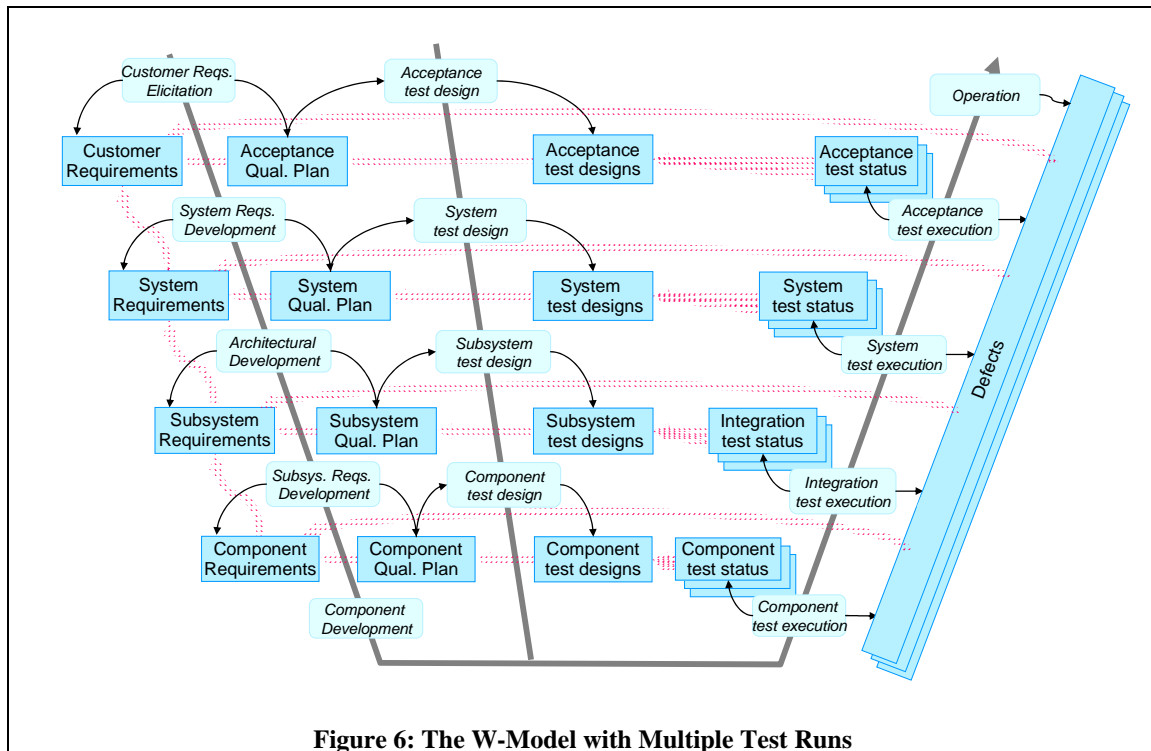


Figure 6: The W-Model with Multiple Test Runs

Roles in Development

For the purposes of this paper, we distinguish 4 roles:

- Manager
- Analyst
- QA/Tester
- Developer

Role: Manager

The Manager encompasses managers of analyst teams, managers of test teams, managers of development teams and overall project management. These managers form the major part of System Review Boards (SRB), sometimes otherwise known as Change Control Boards (CCB). The SRB makes the ultimate decision on whether changes are implemented and determines when a system or product version is ready for release. With a Requirements-Driven Testing approach, test managers influence the SRB in the confidence that they have an efficient test process working against the requirements; development management influence the SRB based on the latest test information against their development efforts; and analyst management influence the SRB with the full business impact of making a release based on requirements met or not.

The Manager oversees development by

- setting release targets,
- gathering information to measure progress against those targets,
- assessing the impact of test failure,
- prioritizing defects and development effort, and
- recording and taking responsibility for key release decisions.

Role: Analyst

The Analyst is often identified (but not exclusively) by job titles such ‘requirements analyst’, ‘business analyst,’ ‘systems analyst,’ ‘requirements engineer’ or ‘systems engineer.’

The Analyst is responsible for

- ensuring that requirements are expressed and classified properly,
- defining qualification criteria against every requirement,
- submitting defects as a result of requirements review activities,
- performing coverage analysis to ensure that every requirement has tests,

- performing impact analysis across the lifecycle for proposed requirements changes,
- analyzing defects and traces them to affected requirements, and
- monitoring the qualification status of each requirement.

Role: QA/Tester

The QA/Tester is a team member of the ‘Quality Assurance’ or ‘Testing’ team, identified by job titles like ‘Quality Engineer,’ ‘QA Engineer,’ or ‘Test Engineer.’

The QA/Tester is responsible for

- designing tests against requirements and qualification criteria,
- participating in requirements reviews to ensure requirements are testable,
- establishing traceability between requirements and tests,
- allocating tests to test runs,
- executing test runs,
- recording test status, and
- submitting defects when tests fail.

Role: Developer

The developer is someone who participates in the design and implementation of the system or product. This is a broad definition and covers everything from a ‘Design Authority’ or ‘Technical Architect’ to a ‘Software Engineer’ or ‘Software Developer.’

The developer role carries out development by

- working to complete implementation tasks,
- working to complete defect resolution tasks,
- working to complete tasks arising from changed requirements, and
- submitting defects found during unit testing.

Integrating Requirements Management, Test Management and Defect Management

The W-Model motivates a natural way of dividing data and roles between the two areas of requirements management and test management, as portrayed in Figure 7. The Analyst and QA/Tester roles, with their respective disciplines of requirements management and testing, overlap on the qualification plans, which are created as the requirements are written, and contain the criteria defining the tests.

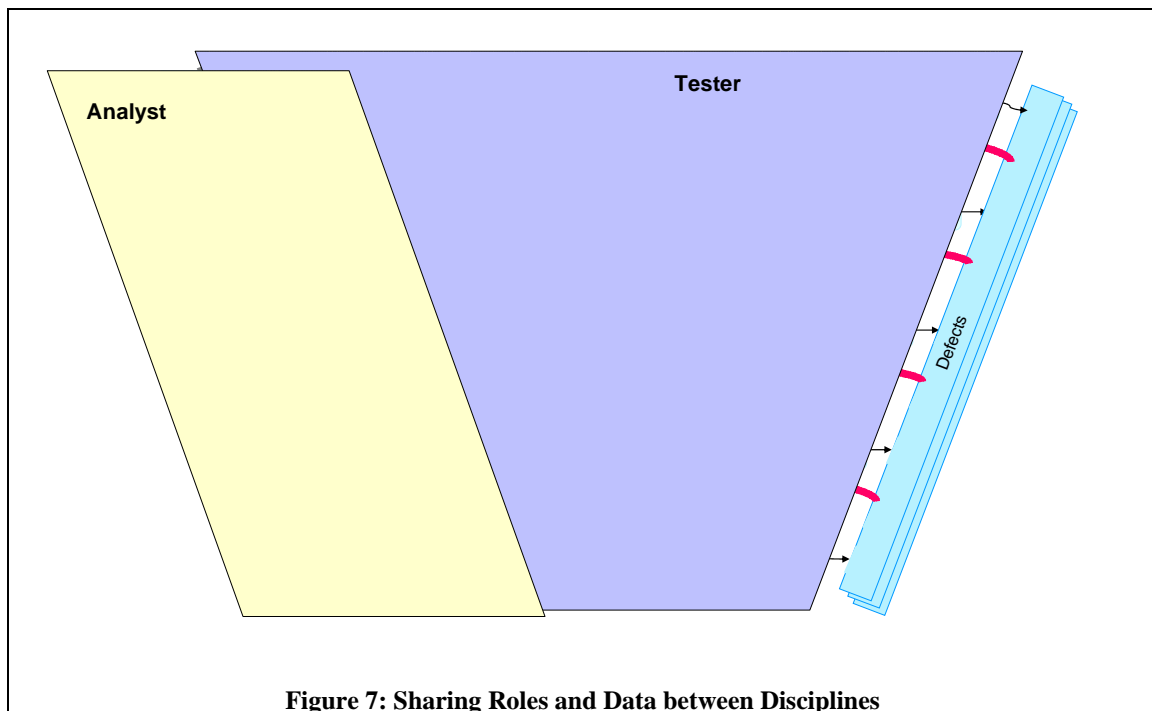


Figure 7: Sharing Roles and Data between Disciplines

The DOORS - TestDirector Integration

The DOORS - TestDirector Integration organizes the bi-directional transfer of requirements and test information between Telelogic DOORS and Mercury TestDirector enabling Testers to work against defined requirements and Analysts to monitor overall test coverage to requirements. The integration works in both an “Online” mode where information is seen as live and also a “Batch” mode where data synchronization tasks are needed.

The DOORS – TestDirector integration provides a unique role-based approach to systems integration. The roles addressed include the Analyst, the QA/Tester, the manager and the overall tools administrator.

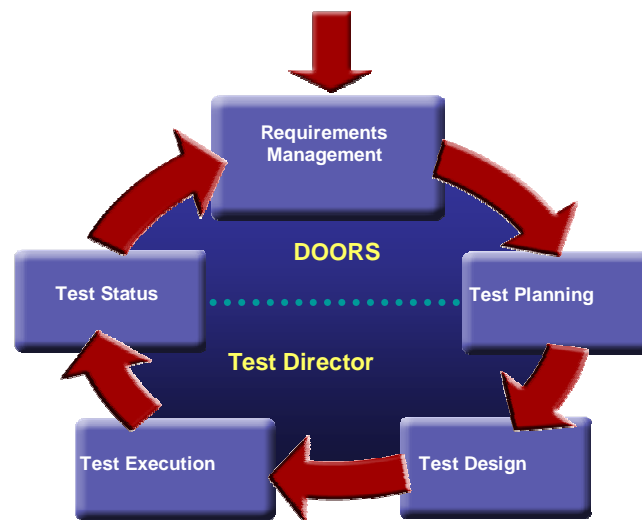
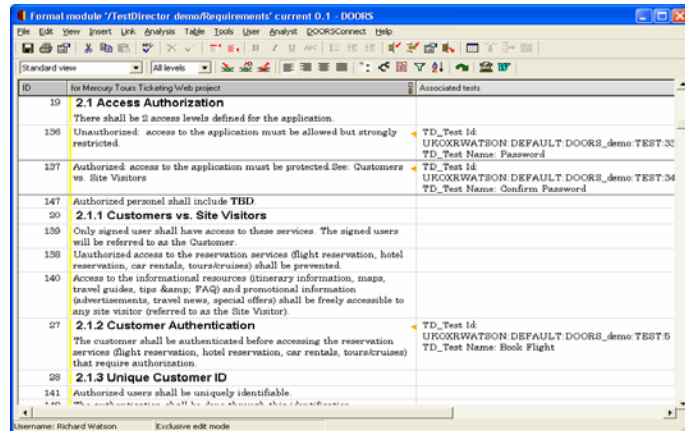


Figure 8: The requirements-driven testing process

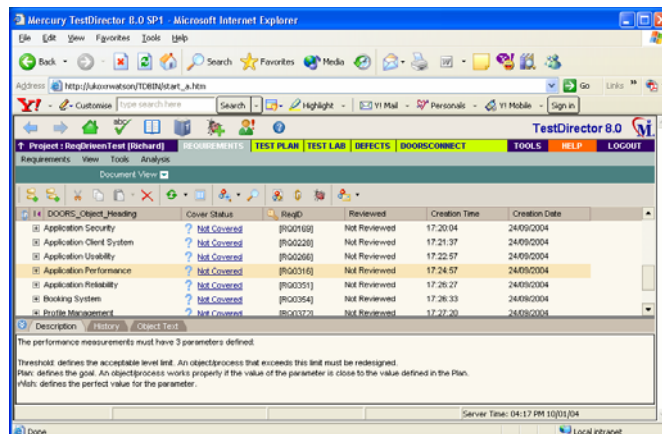
Analyst

The analyst is principally interested in Requirements Management and would like to do most of his work inside of DOORS. The DOORS – TestDirector integration enables the Analyst to produce requirements for the test team – the qualification criteria. Through the integration the analyst is able to monitor that test planning has aimed to test all of the original requirements and can interrogate this plan to review that the Test Plan does indeed meet the qualification criteria. Once testing begins, the Analyst can monitor the requirements for associated defects that have been raised against tests that have failed. The analyst can now finally identify full business impact for defect resolution against the original customer demand.



QA/Tester

The QA/Tester is principally interested in Quality Assurance and testing; most of the QA/Testers work is performed within TestDirector. Receiving qualification criteria from the Analyst, the QA/Tester sees this inside the Requirements tab within TestDirector and works against it as his test requirements. Requirements can be reviewed from within TestDirector and feedback recorded directly into TestDirector fields to be automatically fed back to the Analyst. As Test Planning is performed, the QA/Tester creates an audit trail from test information to requirements. Any comments made by the Analyst on test data from within DOORS will be reflected into the central TestDirector database. The QA/Tester produces test cases against the test plan and then moves into test execution. As defects are raised, the root cause can be quickly identified by following traceability back to either the test case, the development (through the SYNERGY integration – see the later section in this paper for more details) or the original requirement through the DOORS integration.

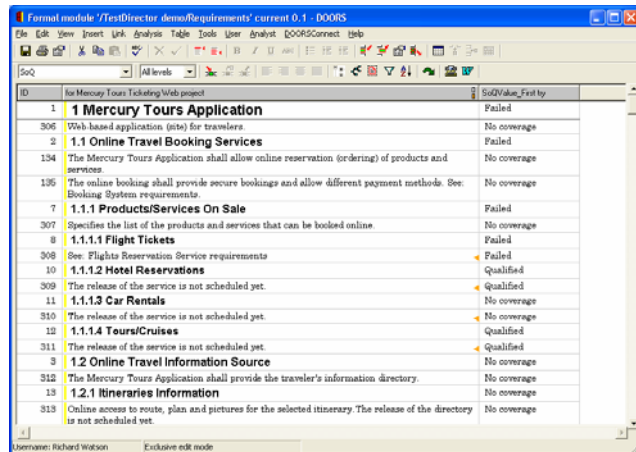


Manager

The manager is principally interested in taking part in Change Control Boards or System Review Boards.

The manager comes from all disciplines and therefore uses whichever tool he is most familiar with, TestDirector, DOORS, SYNERGY, etc. Through the DOORS – TestDirector integration, the manager is able to monitor overall requirement status through a number of “Statement of Quality” (SoQ) reports. The SoQ reports

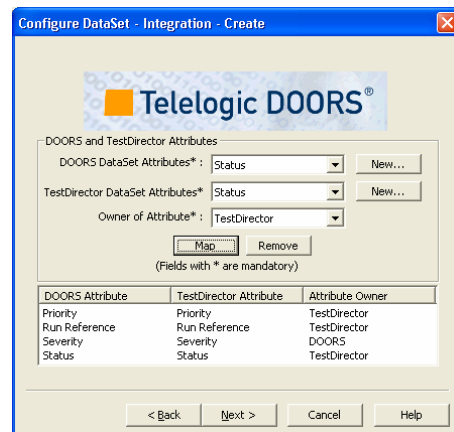
provide a hierarchical set of information regarding the overall test status of a requirement under development. The manager can attend a Systems Review Board with the confidence that he is fully aware of the business impact of releasing a system or not, based on the original user needs.



ID	Requirement	SoQValue_Failty
1	1 Mercury Tours Application	Failed
306	Web-based application (site) for travelers	No coverage
2	1.1 Online Travel Booking Services	Failed
134	The Mercury Tours Application shall allow online reservation (ordering) of products and services.	No coverage
135	The online booking shall provide secure bookings and allow different payment methods. See: Booking System requirements.	No coverage
7	1.1.1 Products/Services On Sale	Failed
307	Specify the list of the products and services that can be booked online.	No coverage
8	1.1.1.1 Flight Tickets	Failed
308	See: Flights Reservation Service requirements	Failed
10	1.1.1.2 Hotel Reservations	Qualified
309	The release of the service is not scheduled yet.	Qualified
11	1.1.1.3 Car Rentals	No coverage
310	The release of the service is not scheduled yet.	No coverage
12	1.1.1.4 Tours/Cruises	Qualified
311	The release of the service is not scheduled yet.	Qualified
3	1.2 Online Travel Information Source	No coverage
312	The Mercury Tours Application shall provide the traveler's information directory.	No coverage
13	1.2.1 Itineraries Information	No coverage
313	Online access to route, plan and pictures for the selected itinerary. The release of the directory is not scheduled yet.	No coverage

Integrations Administrator

The integration administrator is one of a minority in an organization. Principally the administrator aims at automating as much of an organization's lifecycle as possible while limiting the adverse business impact of tool introduction within a company. The integration administrator defines the mapping between DOORS information and TestDirector. The administrator defines which type of information should be shared between DOORS and TestDirector (Requirements, Tests, Defects, etc.) and also any attribute mapping between the data (business criticality for the Analyst might be seen as priority to the QA/Tester).



Configure DataSet - Integration - Create

Telelogic DOORS®

DOORS and TestDirector Attributes

DOORS DataSet Attributes*: Status

TestDirector DataSet Attributes*: Status

Owner of Attribute*: TestDirector

(Fields with * are mandatory)

DOORS Attribute	TestDirector Attribute	Attribute Owner
Priority	Priority	TestDirector
Run Reference	Run Reference	TestDirector
Severity	Severity	DOORS
Status	Status	TestDirector

Overall benefits:

- The Manager now finally understands the full business impact of releasing a system,
- The Analyst now has a focus on providing testable requirements and can be involved in prioritization of defects dependant on business impact, and
- The QA/Tester can test against a defined set of requirements rather than simply test what was built.

The SYNERGY - TestDirector Integration

The SYNERGY-TestDirector Integration (for integrated defect management) organizes the transfer of defect data between Mercury TestDirector and Telelogic SYNERGY/Change, enabling Testers and Developers to manage project defects using the most suitable tool for their needs.

For example, if the development team uses SYNERGY/Change and the quality assurance team uses TestDirector, the SYNERGY - TestDirector Integration enables these teams to integrate projects so that they are both working with the same defect data. The quality assurance team can use one tool to handle its testing process, while the development team can use its preferred change management tool to deal with software development.

The SYNERGY - TestDirector Integration employs multiple transfer options to satisfy varying user needs enabling users to create defects that do not yet exist in the receiving tool, or update existing defects. The SYNERGY - TestDirector Integration is initiated by creating a synchronization link in the SYNERGY Lifecycle Manager. When creating a synchronization link, options are set to define which data is to be synchronized between a ChangeRequest entity in a SYNERGY database and a defect entity in a TestDirector project. After creating a synchronization link, work continues on defects stored within TestDirector or in SYNERGY/Change, as required. The Synchronization can also be run as a service. The service synchronizes links automatically, according to predetermined time intervals.

This integration supports flexible and customizable team-specific lifecycles that are already used by development and testing teams. It supports lifecycle hiding by hiding some workflow states used by one team from the other.

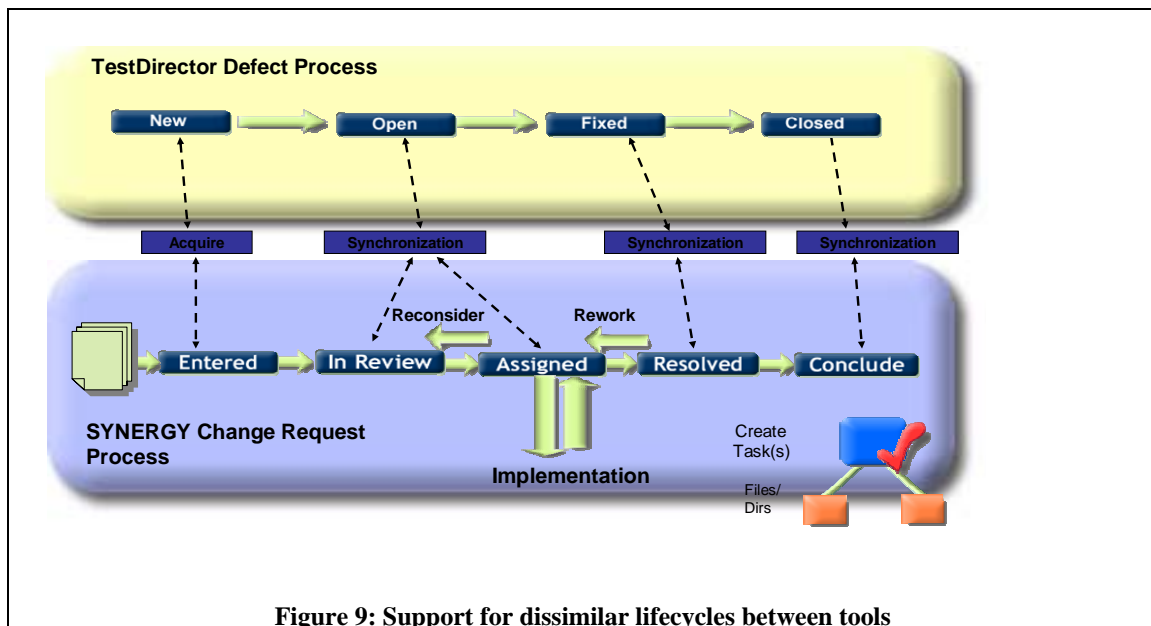


Figure 9: Support for dissimilar lifecycles between tools

Conclusion

This paper has discussed the integration of Requirements Management, Test Management and Defect Management into a Requirements-Driven Testing process.

We have started by stating the principles, or key practices, of testing in relation to requirements. We then extended the classic “V” -model to include testing processes and information, and portrayed it as the “W”-model. This model shows the relationships between the various kinds of data that should be managed in the integrated approach.

Next, we have described various key roles, and shown how these map onto the process. Finally, we have shown how this approach has guided the way in which best-in-class solutions for requirements management, test management and change/configuration management - Telelogic’s DOORS, Mercury’s TestDirector and Telelogic’s SYNERGY - have been integrated to support this broader process.

Today, product, systems and software development organizations cannot be satisfied with making individuals more productive and implementing point solutions. To achieve a sustainable competitive advantage, the analysts, developers and testers must be in sync to enable the development organization to achieve its full potential. Requirements-driven testing, with support from integrated tools, offers a viable solution to a key aspect of an Automated Lifecycle Management (ALM) approach to help organizations meet the challenges of meeting ever rising customer expectations, reducing time-to-market and increasing product, system or software quality.

About Telelogic

Telelogic® is the leading global provider of solutions for advanced systems and software development.

Telelogic's best-in-class software tools automate and support best practices throughout the development lifecycle, so nothing is lost, overlooked, or misunderstood. By optimizing each phase of development, Telelogic tools improve visibility and collaboration, enabling customers to deliver better software and systems faster.

Telelogic's requirements-driven solutions span a project's entire lifecycle, from initial analysis and planning through development, testing and tracking. They include:

- Telelogic DOORS® - an integrated family of tools that empowers an organization through a structured requirements management process, enabling more effective communication and collaboration.
- Telelogic TAU® - an integrated family of analysis, modeling, design and testing tools that provides a unique visual development environment for complex, demanding systems and software.
- Telelogic SYNERGY™ - an integrated tool family for change and configuration management, which provides lifecycle control for a company's digital assets.

Telelogic's Professional Services Team offers a combination of training, consulting and support that makes the introduction and ongoing use of our tools as smooth as possible, while maximizing your return on investment and empowering you to become extremely competitive.

Headquartered in Malmö, Sweden, with U.S. headquarters in Irvine, California, Telelogic has operations in 17 countries worldwide. Customers include Alcatel, BAE SYSTEMS, BMW, Boeing, DaimlerChrysler, Deutsche Bank, Ericsson, General Motors, Lockheed Martin, Motorola, NEC, Nokia, Philips, Siemens, Thales and Vodafone. For more information, please visit www.telelogic.com.