

Software Verification and Validation Tool Comparison and Selection Criteria

Drumm, Paul M.

Table of Contents

Table of Contents	1
1 Software Verification and Validation Defined	1
1.1 Verification.....	1
1.2 Validation	1
2 Tool Qualities for a Tester’s Workbench.....	1
2.1 Ease of Use.....	2
2.2 Power.....	2
2.3 Robustness.....	2
2.4 Functionality.....	2
2.5 Ease of Insertion.....	2
2.6 Quality of Support.....	2
2.7 Cost of the Tool.....	2
2.8 Fit with Organizational Policies and Goals.....	2
3 Testing Maturity Model (TMM) phases and recommended tools and capabilities at each phase	3
3.1 TMM Level 1 – Initial.....	3
3.2 TMM Level 2 – Phase Definition	3
3.3 TMM Level 3 – Integration.....	3
3.4 TMM Level 4 – Measurement and Management.....	4
3.5 TMM Level 5 - Optimization, Defect Prevention, and Quality Control.....	4
4 Brief overview of waterfall steps and possible testing activities	5
4.1 System Requirements	5
4.2 Software Requirements	5
4.3 Analysis.....	6
4.4 Design.....	6
4.5 Coding	6
4.6 Testing.....	7
4.7 Operations	7
5 Analysis of Microsoft Visual Studio Team Suite with Team Foundation Server	7
5.1 URL.....	7
5.2 Primary Features	8
5.3 Facilitation of V&V	8
5.4 Reasoning for Selecting Tool.....	9
5.5 Waterfall Phase Usage	12
5.5.1 System Requirements.....	12
5.5.2 Software Requirements.....	12
5.5.3 Analysis.....	12
5.5.4 Design	13
5.5.5 Coding.....	13
5.5.6 Testing.....	13
5.5.7 Operations.....	14
5.6 Tool Benefit Summary	14
6 Analysis of Mercury Quality Center (MQC) 9.2.....	14
6.1 URL.....	15

6.2	Primary Features	15
6.3	Facilitation of V&V	17
6.4	Reasoning for Selecting Tool.....	18
6.5	Waterfall Phase Usage	20
6.5.1	System Requirements.....	20
6.5.2	Software Requirements.....	20
6.5.3	Analysis.....	20
6.5.4	Design	21
6.5.5	Coding.....	21
6.5.6	Testing.....	21
6.5.7	Operations	21
6.6	Tool Benefit Summary	21
7	Analysis of Imagix 4D	22
7.1	URL.....	22
7.2	Primary Features	22
7.3	Facilitation of V&V	23
7.4	Reasoning for Selecting Tool.....	23
7.5	Waterfall Phase Usage	25
7.5.1	System Requirements.....	25
7.5.2	Software Requirements.....	25
7.5.3	Analysis.....	26
7.5.4	Design	26
7.5.5	Coding.....	26
7.5.6	Testing.....	26
7.5.7	Operations	26
7.6	Tool Benefit Summary	26
	References.....	27
	Appendix A: Glossary.....	28
	Appendix B: Software V&V Company Listing.....	28
	Appendix D: Software V&V Tool Classification Listing and Examples.....	29

1 Software Verification and Validation Defined

Software Verification and Validation (V & V) have various definitions depending on the source and the year of the reference. The definitions used in this paper are from current CMMI references.

1.1 Verification

Software verification in the CMMI sense is defined as “The Purpose of verification (VER) is to ensure that selected work products meet their specified requirements” [Chrisis03] p. 575

There are three main specific goals associated the CMMI Verification Process Area [Ahern04]

- Prepare for Verification – Select the work products, Establish and maintain the verification environment, procedures, and criteria for the selected work products.
- Perform Peer Reviews on Selected Work Products – Prepare and conduct peer reviews, identify issues and analyze data from peer reviews on selected work products.
- Verify Selected Work Products Against Their Requirements– Perform verification and analyze the results of the activities.

1.2 Validation

Software validation in the CMMI sense is defined as “The Purpose of validation (VAL) is to demonstrate that a product or product component fulfills its intended use when placed in its intended environment” [Chrisis03] p. 563

There are two main specific goals associated the CMMI Validation Process Area

- Prepare for Validation – Select work products to be validated and methods for validation. Establish and maintain those procedures and criteria.
- Perform Validation – Perform validation and analyze the results.

2 Tool Qualities for a Tester’s Workbench

According to Burnstein [Burnstein03], there are 8 criteria you should use for evaluating your tool for inclusion into your testing workbench which I summarize below. A good software V&V tool should meet these criteria.

2.1 *Ease of Use*

This characterizes the usability of the product from the testers and organizations perspective.

2.2 *Power*

This characterizes the number of features and the ability of it to do everything the user wants.

2.3 *Robustness*

This characterizes how the tool handles internal errors and how it recovers.

2.4 *Functionality*

This characterizes how the tool meets the particular area it was procured for such as final test or static code checking. It also may show if new functionality can be easily integrated into the system.

2.5 *Ease of Insertion*

This characterizes the ability to fit into an organizations test tool arsenal and how long it takes the organization to start using it in a constructive fashion.

2.6 *Quality of Support*

This characterizes the overall tool vendor's atmosphere when dealing with tool issues. This may cover helpdesk, maintenance, upgrades and documentation.

2.7 *Cost of the Tool*

This characterizes not only how much the tool costs upfront but also ongoing maintenance, initial and ongoing training costs, insertion costs, and other organizational costs.

2.8 *Fit with Organizational Policies and Goals*

This characterizes how well the tool fits in with the existing institutionalized policies and goals.

3 Testing Maturity Model (TMM) phases and recommended tools and capabilities at each phase

At each juncture of the TMM, certain tools should be used. When making tool selection for software V&V the phase or maturity your organization is at should be considered along with the quality of the tool such as power and functionality. Even if an organization does not use TMM, the listing below shows an increased level of complexity and advancement in tool capability as you progress towards higher levels of V&V in your processes. This toolset listing should be compared with any software V&V tool you are considering to see if the necessary features are included when a wide-ranging software tool suite (as opposed to a single tool) is being evaluated. Taken from Burnstein [Burnstein03].

3.1 TMM Level 1 – Initial

Recommended Tool Set

- Interactive debuggers -
- Configuration Building Tools – e.g. a make utility
- LOC counters

3.2 TMM Level 2 – Phase Definition

Phase Definition Goals: Develop testing and debugging goals, Initiate a test planning process, Institutionalize basic testing techniques and methods

Recommended Tool Set

- Project Planners and test planners
- Run-Time error checkers – bounds checkers, memory testers, leak testers
- Test Prep support tools – for development of white and black box testing.
 - Black box – cause and effect graph or equivalence classes
 - White box- control flow analyzers that generate control flow graphs, data flow analyzers that support data flow graphs.
 - Helps identify branches, basis paths, and variable usages. This helps developers to design test cases.
- Coverage analyzers – These assist with white box testing. Tells how much coverage of the code path was given and if the coverage goal was met.
- Cross-Reference tools – They trace occurrences of items as they appear in different software artifacts, i.e. where does a specific variable occur in all the code

3.3 TMM Level 3 – Integration

Integration Goals: Establish a software test organization, Establish a technical training program, Integrate testing into the software life cycle, Control and monitor the testing process.

Improve software quality, control and monitor testing, improve tester/developer productivity, integrate testing throughout the life cycle, give visibility to the testing organization, illustrate the benefits of having both a dedicated test org and a technical training program

Recommended Tool Set

- Requirement Recorders (use case recorders)
- Requirements Verifiers
- Requirements-to-Test Tracers
- Capture-Replay Tools – automating the execution of tests and capturing all input and output
- Comparators – compare actual test outputs to expected
- Defect Trackers

3.4 TMM Level 4 – Measurement and Management

Measurement and Management Goals: Establish an organization wide review program, Establish a test measurement program, Software quality evaluation

Recommended Tool Set

- Code checkers - static analyzers
- Auditors – code standard/formatting violations
- Code Comprehension Tools
- Test Harness Generators
- Performance Testing Tools
- Network Analyzers
- Simulators and Emulators
- Web Testing Tools
- Test management tools

3.5 TMM Level 5 - Optimization, Defect Prevention, and Quality Control

Goals: Test process optimization, Quality Control, Application of process data for defect prevention

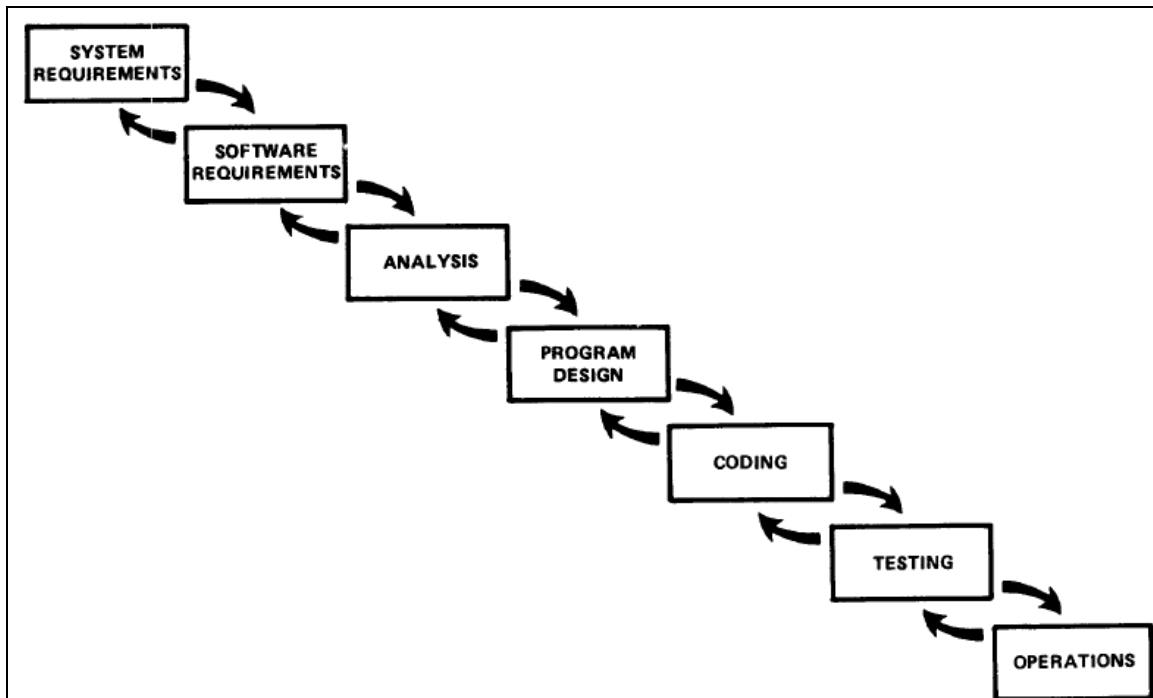
Recommended Tool Set

- PAL support tools
- Advanced Test Scripting tools

- Assertion checkers
- Advanced Test Data Generators
- Advanced Test Management Systems
- Usability Measurement Tools

4 Brief overview of waterfall steps and possible testing activities

The Waterfall Model as originally presented by Dr. Winston Royce [Royce]



4.1 *System Requirements*

Elicitation and Gathering of system requirements from the customer

Possible V&V activities

- Record system requirements in source control
- Make test cases and plans that map to the system requirements
- Review work products

4.2 *Software Requirements*

Elicitation and Gathering of software requirements from the customer

Possible V&V activities

- Record software requirements in source control
- Make test cases and plans that map to the software requirements
- Review all work products

4.3 Analysis

Performing analysis on the requirements for validity and generating a specification document. This also helps to see if it is feasible to build the software product.

Possible V&V activities

- Validating that the requirements map to the vision and scope to fulfill the intended purpose of the software product(s)
- Verifying all requirements are mapped to the specification document
- Verifying all requirements map to project plans
- Review all work products

4.4 Design

Performing high-level design and architecture of the product

Possible V&V activities

- Using UML for the design and mapping the components back to the requirements.
- Design test cases for interface testing
- Design stubs for test cases
- Map component/module design to requirements
- Review all work products

4.5 Coding

Writing the code per the design specifications

Possible V&V activities

- Write unit tests that map to code modules
- Map code modules to functional requirements
- Run static code testing to verify conformance to policies and standards
- Put code base under configuration control
- Review all work products

4.6 Testing

Testing the software product to verify and validate the software product

Possible V&V activities

- Design black box and white box tests
- Run black box and white box tests
- Ensure all test results are input into a test database repository.
- Verify the test environment is similar to the intended environment the component or product will actually operate in.
- Validate the component under test will perform in its intended environment.
- Verify the component/system/product under test fulfills the requirements.
- Map requirements to tests to verify all requirements have been accounted for in test cases.
- Design and map Equivalence Classes and Boundary Values to modules for dynamic execution testing
- Run flow analysis to verify all code paths have been tested.
- Review all work products

4.7 Operations

Deployment, Support, and ongoing Software Maintenance of the software product

Possible V&V activities

- Verify deployment actions map to deployment requirements
- Verify test deployment environment is similar to intended environment
- Validate intended deployment environment characteristics
- Verify support requirements map to support environment
- Review all work products

5 Analysis of Microsoft Visual Studio Team Suite with Team Foundation Server

5.1 URL

<http://msdn2.microsoft.com/en-us/teamssystem/default.aspx>

5.2 Primary Features

- Dynamic Code Analyzer - Code Profiler – Profiles and collects data on threads and processes while the system is running
- Static Code Analyzer - Prefast static analysis tool and FxCop add annotations and rules to find more defects during static analysis.
- Load Testing - simulate production loads and diagnose performance issues
- Manual Testing – Allows you to run tests manually
- Test Case Management – Manage your tests conveniently directly from the IDE
- Database Unit Testing – Generate unit tests for stored procedures and other parameter based objects
- Data Generation – Auto generate test data into your database for testing
- Schema Compare – Compare metadata schemas between database objects
- Data Compare – Compare data between databases
- Database Schema Deployment Tools – Deploy objects to the targeted deployment environment
- Load Testing – Generate load and collect performance data for stressing objects such as databases and web applications
- Manual Testing - test tasks are to be completed by a test engineer as opposed to an automated script
- Test Case Management
- Ordered test: ordered test executes a set of tests in an specific order determined by the tester
- Web Testing - Automated record playback functionality to create repeatable web tests including Web Services, HTTP and XML
- Unit Testing – Automatic creation of unit testing and parameters from Visual Studio IDE
- Insertion of test data in database
- Extensibility with Generic testing and customized Test-Type Add ins
- Built-in Agile and CMMI methodologies
- Code Coverage – Automated Analysis of code coverage during testing
- Team collaboration – real time collaboration of team communications
- Configuration Management – manage source code, deployment, change history
- Project Management – Integrating the Team Suite with the Team Foundation Server allows for real time project management.
- Team Repository
- Reporting System – Includes Reporting Services for instantaneous reporting of team data such as project status, test status, and code base.

5.3 Facilitation of V&V

Most of what the tool offers makes V&V easier to accomplish. The Team Foundation Server (TFS) lets you group and assign roles to team members comparable to what one would find in a CMMI based organization. That is your foundation for determining who is going to be responsible for establishing V&V and the process that controls it. The mapping of requirements to code testing can be facilitated by the use of creating work items within Team Foundation server and then mapping the work items to your test projects. The test projects may have many kinds of test applicable to the code modules created to satisfy the requirements. Work item satisfaction or progress can be tracked via the project management feature of TFS.

Peer reviews can be facilitated by posting your work in the Team SharePoint Portal site and having others review it. This can be tracked via a work item in TFS.

Work Items in TFS are a major staple of requirements, activity, defect, and other items you need tracked throughout a project. The reporting of work items can focus the project managers ability on how verification is proceeding by looking at all the successful activities related to verification.

The various testing capabilities such as static analysis, dynamic analysis, automated unit test generators, and others, make it easier to help test various parts of the system from class methods (unit test), integration test, and system testing.

Validation of the software with this tool is mostly accomplished through the load and performance testing that lets you simulate the actual load environment the software may be used in. Various browsers can also be tested for in the automated web page testing which facilitates validation of the software further.

All the automation and built in features of the tool makes V&V much easier for the test, requirements, and software engineers.

5.4 Reasoning for Selecting Tool

Visual Studio Team Suite when used in conjunction with the Team Foundation Server provides a wide coverage of the toolset recommended in the various phases of TMM and fits the criteria for tools in the tester's workbench. This was also one of the top tool suites of the year in Software Test and Performance magazine [STP06].

- Ease of Use – Microsoft has made the tool very easy to use. The GUI, its' menu system, and the online help makes it a real easy tool to use.

- Power – When the team suite is combined with the foundation server, it makes a very powerful and capable tool that meets the needs of most shops throughout the software development life cycle.
- Robustness – The tool will tell you when it has a problem and will close. Like other tools, you may or may not be able to save your work. The tool has a high mean time to failure (MTTF) which makes it a robust product.
- Functionality – The tool has most functions you need for most V&V activities. The only real lack is enforcement of workflow activity and “true” mapping of requirements to functionality.
- Ease of Insertion – The majority of the testing tools and framework are geared only towards languages that can plug into Microsoft’s Visual Studio Platform. There is even an ADA# plug-in available. There are also various converters to go from one V&V platform to another. For example, there is a Clear Quest converter that lets you import your Clear Quest data into the TFS as work items.
Tool installation was straight forward designed for even the most junior of developers or support staff.
- Quality of Support – Microsoft offers support on various models. They offer everything from a premier support for production work stoppage issues to developer issues.
- Cost of the Tool – Cost of the toolset is comparable to other V&V platforms. A single user license is ~\$5000.
- Fit with Organizational Goals and Policies – The platform is a Microsoft based platform and only makes sense if you are a Microsoft shop or your development environment fits into the Visual Studio framework. The CMMI and Agile MSF templates make it easy for an organization to base the profiles using one of the two templates.

The following table shows a comparison of the tools needed for a tester’s workbench versus the features available with this tool.

TMM Level 1	Availability
• Interactive debuggers -	✓
• Configuration Building Tools – e.g. a make utility	✓
• LOC counters	✓
TMM Level 2	
• Project Planners and test planners	✓
• Run-Time error checkers – bounds checkers, memory testers, leak testers	✓
• Test Prep support tools – for development of white and black box testing.	

<ul style="list-style-type: none"> ▪ Black box – cause and effect graph or equivalence classes 	
<ul style="list-style-type: none"> ▪ White box- control flow analyzers that generate control flow graphs, data flow analyzers that support data flow graphs. 	
<ul style="list-style-type: none"> ▪ Helps identify branches, basis paths, and variable usages. This helps developers to design test cases. 	
<ul style="list-style-type: none"> • Coverage analyzers – These assist with white box testing. Tells how much coverage of the code path was given and if the coverage goal was met. 	✓
<ul style="list-style-type: none"> • Cross-Reference tools – They trace occurrences of items as they appear in different software artifacts, i.e. where does a specific variable occur in all the code 	✓
TMM Level 3	
<ul style="list-style-type: none"> • Requirement Recorders (use case recorders) 	
<ul style="list-style-type: none"> • Requirements Verifiers 	
<ul style="list-style-type: none"> • Requirements-to-Test Tracers 	
<ul style="list-style-type: none"> • Capture-Replay Tools – automating the execution of tests and capturing all input and output 	✓
<ul style="list-style-type: none"> • Comparators – compare actual test outputs to expected 	✓
<ul style="list-style-type: none"> • Defect Trackers 	✓
TMM Level 4	
<ul style="list-style-type: none"> • Code checkers - static analyzers 	✓
<ul style="list-style-type: none"> • Auditors – code standard/formatting violations 	✓
<ul style="list-style-type: none"> • Code Comprehension Tools 	
<ul style="list-style-type: none"> • Test Harness Generators 	✓
<ul style="list-style-type: none"> • Performance Testing Tools 	✓
<ul style="list-style-type: none"> • Network Analyzers 	✓
<ul style="list-style-type: none"> • Simulators and Emulators 	✓
<ul style="list-style-type: none"> • Web Testing Tools 	✓
<ul style="list-style-type: none"> • Test management tools 	✓
TMM Level 5	

• PAL (Process Asset Library) support tools	✓
• Advanced Test Scripting tools	✓
• Assertion checkers	
• Advanced Test Data Generators	✓
• Advanced Test Management Systems	

5.5 Waterfall Phase Usage

Note: VER = VERification, VAL = VALidation

5.5.1 System Requirements

Gathering of system requirements from the customer

Put the requirements into work items in TFS for tracking

5.5.2 Software Requirements

Gathering of software requirements from the customer

Put the requirements into work items in TFS for tracking

5.5.3 Analysis

Performing analysis on the requirements for validity and generating a specification document. This also helps to see if it is feasible to build the software product.

A feature of the Visual Studio add-in from Microsoft is the Domain Specific Language add-in that gives you the ability to use a pseudo-UML like designer and generate use cases. - VER

Use Microsoft Word to generate the vision and scope document and put in the TFS SharePoint Portal

Use Microsoft Word to generate the software requirements specification and put in the TFS SharePoint Portal

Use Microsoft Word to generate the software development plan and put in the TFS SharePoint Portal

5.5.4 Design

Performing high-level design and architecture of the product from the requirements in the TFS SharePoint Portal

Use the class designer to map out the architecture based on the requirements and use cases. - VER

Break down the modules further in to subclasses and methods

Map the work items back into the design - VER

Generate the initial requirements/functional test cases in a new test project - VER

Generate interface tests based on the class design using the test project - VER

Generate a code template the limits the developers ability to add objects that are not part of the code template policy into the system

5.5.5 Coding

Writing the code per the design specifications

Use the static code analyzer to verify that coding meets coding standards - VER

Use the unit test generator to verify the modules can accommodate legal, illegal, and boundary value parameters. VER

Use work items to select work products for peer review - VER

Use the TFS SharePoint Portal and work items to hold peer reviews - VER

Use the work items to validate the requirements are mapped to the code modules - VER

Check code into source control.

Use the MSBUILD tool to automate build per configuration environment - VAL

5.5.6 Testing

Testing the software product to verify and validate the software product

Verify the requirements functional tests projects are complete and map-VER

Write/rewrite more tests in the test projects to map to requirements - VER

Collect test case projects and put into configuration control - VER

Automate web page and web service testing - VER

Verify the system functionality using the requirements in the work items- VER

Report on test failures via the reporting services in the TFS SharePoint Portal

Perform Regression Testing via automated playback - VER
Perform Ordered testing of various test cases - VER

Use the Load Testing tool to Validate and simulate the environment in which the product will actually be used - VAL
Use the requirements document in the TFS SharePoint Portal to validate the configuration environment to be used at the customer's site. - VAL

5.5.7 Operations

Deployment, Support, and ongoing Software Maintenance of the software product

Verify configuration environment from requirements and test project documents.-VER

Automate current builds and future builds using the MSBUILD tool and deploy to update web site using a CAB project that is taken out of configuration control.-VER

5.6 Tool Benefit Summary

Microsoft's tool offering almost covers the whole SDLC. There are gaps in its presentation as compared to other tools that have a higher degree of workflow integration between requirements and other phases of the SDLC. The project management feature is not as good as I would like. However, Microsoft has supplemented it with having the ability to integrate with Microsoft Project server through an add-on link. The MS Project server does cost more money but the link between the TFS and MS Project server is free. This tool is mostly geared toward Microsoft shops as far as V&V is concerned but can accommodate other compilers as long as that manufacturer makes a plug in for Visual Studio. Microsoft has a C++ compiler that is not .Net that is available for use. For small shops of around 25 people, the cost is around \$12,000 per month over 3 years for a complete package consisting of Operating system, Team Suite, Upgrade Insurance, Support, online tool training, and several other premium services. Microsoft also makes available an Enterprise Agreement where you "True-Up" for all the licenses you have acquired or released over the last year for all products in your organization which does make it convenient to add in other Microsoft product offerings without having to deal with the licensing costs immediately.

6 Analysis of Mercury Quality Center (MQC) 9.2

6.1 URL

<http://www.mercury.com/us/products/quality-center/>

6.2 Primary Features

Mercury Quality Center is a suite consisting of 6 products.

- Mercury Quality Center Dashboard
- Mercury TestDirector
 - HP Service Test Management module
 - Release Management
 - Requirements Management
 - Test Plan and Defects Management
 - Test Lab and Defects Management
- Mercury QuickTest Professional
- Mercury WinRunner
- Mercury Business Process Testing
- Mercury Service Test

A graphical representation is of how they relate is shown below.

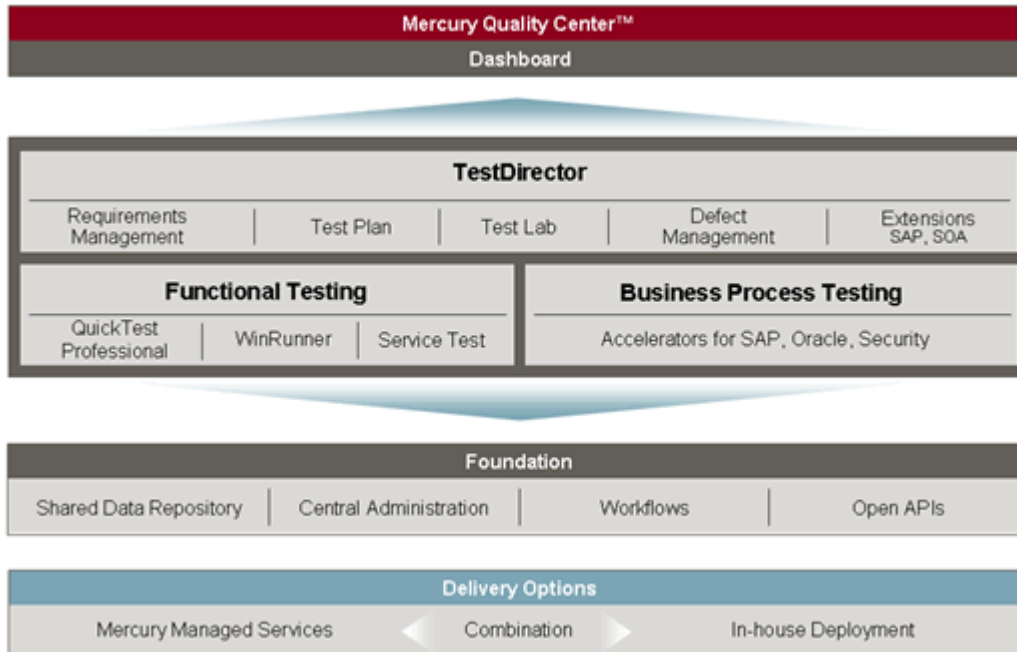


Figure 1. from <http://www.mercury.com/us/products/quality-center/>

Some of the key features are

MQC Dashboard

- View Key Performance Indicators (KPIs) that are relevant to your project. These may be related to defects (e.g. Defects by Status), requirements (e.g. Requirements by Status), or tests (e.g. Tests Execution by Status) from MQC Test Director
- Drill-down through KPIs to look at problems.
- Stoplight graphs for visual queuing on problem issues.
- Customize KPIs based on data you want to see
- Use for trend analysis and for historical test/defect/requirements management

MQC Test Director

- Has many features you need during the SDLC - requirements management, planning tests, building tests, scheduling and executing tests, defect management, and test project status analysis
- All interaction can be done through a single web interface
- Test case management can be automated or manual
- Can schedule test cases to run automatically and store data for later viewing.
- SOA services and environments can be tested
- Increase your Process Asset Library by tracking quality assets and progress across releases and test phases in a single repository
- Auto-generate functional test cases and traceability to the requirements
- Define test plans and import from MS Word or Excel
- Real time reporting on test defects

MQC Quick Test Professional

- Defect documentation for developers
- Regression test applications and their environments
- Replicates defect testing via automated test recording
- Supports a wide variety of systems such as J2EE and .NET
- Interoperability with WinRunner so all functional tests can be called from either software module
- Integrates into the HP Business Process Testing Framework to combine documentation and testing
- Codeless creation and modification of test scripts
- Covers functional and regression testing

MQC WinRunner

- Capture, Verify, and Replay user interactions
- Emulates user interaction via automated recording
- Verifies expected values from statements and tests
- Function Generator
- Virtual Object Wizard for testing almost any object

- Test exception handling and reporting

Mercury Business Process Testing

- Allows non developers to create and run tests
- Test case design
- Automatically update test cases and test plans
- Automatically generate test plan documentation
- Manage change risk for SAP
- Detect application problems early in the SDLC
- Automated and manual functional testing software for test design, test creation, test maintenance, test execution and test data management
- Enables User Acceptance Test (UAT) to deploy automation

Mercury Service Test

- Automated functional, regression, and stress testing of services involved in SOA
- Runs in different environments - J2EE, AXIS, and .NET
- Validation of environments using asynchronous calls and stubs
- Auditing of all calls and recording of tests and parameters
- Ability to simulate services in order to develop test cases early

6.3 Facilitation of V&V

The majority of the functionality is found in the Test Director module which is the base Mercury Quality Center (MQC) program.

MQC does a good job of allowing the analyst to manage requirements over the life-cycle of the product with very high visibility of traceability to test cases and vice versa. This visibility is coupled with reports and graphs that allow a manager or test engineer to immediately tell where the requirements or testing process is at any given time.

The test plan and test lab allows essentially any user to design a test plan and automate the scheduling of tests and cross reference the tests back to requirements.

The automated documentation allows acceptance tests to be reviewed and printed for any type of sign-off.

Quick Test Pro and WinRunner allows even a business analyst or other non-developer types to create web based or windowing forms based functional and regression test by the automated recording and assignment of parameters for any perceived use input.

The Dashboard allows Test Engineers, Architects, Project Managers, and other stakeholders to evaluate the SDLC at a glance given the stoplight graph model and the ability to drill down and focus on key problem areas that might be occurring among the various custom and standard KPIs selected.

With the advent of today's proliferation of SOA, the Mercury Service Test make verification and validation of almost any web service easy for any test engineer.

6.4 Reasoning for Selecting Tool

Mercury Quality Center provides a good coverage of the toolset recommended in the various phases of TMM and fits the criteria for tools in the testers' workbench and was the winner of last year's award for best software testing suite by Software Test and Performance Magazine [STP2006].

- Ease of Use – Easier to use than most tools and has a more intuitive interface than any other tool I have seen. You can open it up and start working on requirements or test cases almost immediately.
- Power – This tool has almost everything except for the tools you would expect to find with an integrated development environment and detailed project management tools.
- Robustness – HP purchased Mercury last year which is a tribute to the product itself. It is on version 9 which makes it a time tested product. I have yet to see an error be thrown by this tool.
- Functionality – The tool has something for everyone. It has breadth and depth to most features especially the requirements and test. Quick Test Pro and Test Directory by themselves have a lot of functionality by themselves with the requirements and test management.
- Ease of Insertion – Will run on a variety of platforms and environments and is easily inserted into almost any shop since it is a browser based system. The installation is quite effortless but does take a while.
- Quality of Support – HP/Compaq has a high reputation for support and any issues can easily be moved up to a higher level of support if needed.
- Cost of the Tool – N/A
- Fit with Organizational Goals and Policies – Mercury Quality Center should be a good fit with any TMM level 3 and above, CMM/CMMI level 3 and above organization. Like any tool you must have a plan to implement it.

The following table shows a comparison of the tools needed for a tester's workbench versus the features available with this tool.

TMM Level 1	Available
<ul style="list-style-type: none"> • Interactive debuggers - 	
<ul style="list-style-type: none"> • Configuration Building Tools – e.g. a make utility 	
<ul style="list-style-type: none"> • LOC counters 	
TMM Level 2	
<ul style="list-style-type: none"> • Project Planners and test planners 	✓
<ul style="list-style-type: none"> • Run-Time error checkers – bounds checkers, memory testers, leak testers 	
<ul style="list-style-type: none"> • Test Prep support tools – for development of white and black box testing. <ul style="list-style-type: none"> ▪ Black box – cause and effect graph or equivalence classes 	
<ul style="list-style-type: none"> ▪ White box- control flow analyzers that generate control flow graphs, data flow analyzers that support data flow graphs. 	
<ul style="list-style-type: none"> ▪ Helps identify branches, basis paths, and variable usages. This helps developers to design test cases. 	
<ul style="list-style-type: none"> • Coverage analyzers – These assist with white box testing. Tells how much coverage of the code path was given and if the coverage goal was met. 	
<ul style="list-style-type: none"> • Cross-Reference tools – They trace occurrences of items as they appear in different software artifacts, i.e. where does a specific variable occur in all the code 	
TMM Level 3	
<ul style="list-style-type: none"> • Requirement Recorders (use case recorders) 	
<ul style="list-style-type: none"> • Requirements Verifiers 	
<ul style="list-style-type: none"> • Requirements-to-Test Tracers 	✓
<ul style="list-style-type: none"> • Capture-Replay Tools – automating the execution of tests and capturing all input and output 	✓
<ul style="list-style-type: none"> • Comparators – compare actual test 	✓

outputs to expected	
• Defect Trackers	✓
TMM Level 4	
• Code checkers - static analyzers	
• Auditors – code standard/formatting violations	
• Code Comprehension Tools	
• Test Harness Generators	✓ With SOA tester
• Performance Testing Tools	
• Network Analyzers	
• Simulators and Emulators	
• Web Testing Tools	✓
• Test management tools	✓
TMM Level 5	✓
• PAL support tools	✓
• Advanced Test Scripting tools	✓
• Assertion checkers	
• Advanced Test Data Generators	✓
• Advanced Test Management Systems	✓

6.5 Waterfall Phase Usage

Note: VER = VERification, VAL = VALidation

6.5.1 System Requirements

Gathering of system requirements from the customer

Enter requirements for multiple projects into the system and assign them to analysts and testers. - VER

6.5.2 Software Requirements

Gathering of software requirements from the customer

Enter requirements for multiple projects into the system and assign them to analysts and testers. – VER

6.5.3 Analysis

Performing analysis on the requirements for validity and generating a specification document. This also helps to see if it is feasible to build the software product.

Generate Requirements Document to discuss with customer-VER
Graph requirements according to various criteria and display status

6.5.4 Design

Generate Test Cases from functional design modules - VER

6.5.5 Coding

N/A

6.5.6 Testing

Generate Test Plans - VER
Generate Test Scripts - VER
Automate tests and schedule tests -VER
Capture End User Usage to generate acceptance tests – VAL
Generate Test Data to simulate intended environment – VAL
Generate Defect Tracking Graph to verify Stop-Test – VER
Match Test Cases to Test/Software requirements – VER
Run Regression tests when code base changes – VER
Update test cases and manage change history of tests and test cases – VER
Increase your test repository for future testing project estimates

6.5.7 Operations

N/A

6.6 Tool Benefit Summary

Mercury Quality Center is a testing centric SQA tool. It has a wide variety of test and test project management tools to run your functional QA tests on most any product including SOA services. With an easy-to-use web interface, it makes alignment with any IT environment an acceptable task to bring into your shop. The QuickTest and WinRunner modules make short work of automating testing for non-developers and junior test engineers.

The drawbacks are that it is testing centric and not much for generating white box or unit tests. There is no source control or integration with a total project management package.

7 Analysis of Imagix 4D

7.1 URL

<http://www.imagix.com>

7.2 Primary Features

Imagix 4D is a code analysis tool that provides a valuable group of software metrics to insure quality of your product, and document and diagram the product.

Flow Charts and Control Flow Analysis

- Control Flow Analysis – More than just call graph generation it shows you the sequence of calls and under what conditions the calls are made. If you have mission critical or a high level of security in mind then this is the feature that will give you that extra edge
- Flow Charts – Shows the complete flow of events weaving through your program
- Function Pointer Support – shows how and when your functions are called through pointers

UML Diagrams and Other High Level Abstractions

- UML Class Diagrams – Generates UML diagrams of your code
- UML File Diagrams – makes file based UML so you can see what/where files are called
- Built-In Abstractions – shows member to member relations and does grouping

Data Collection and Visualization

- Graph Window Display – viewing of code via a series of view such as file or variable or class
- Integrated Source Code Browsing – browse your source code with color coded attributes
- Associated browsing tools – class, file, use

Automated Software Documentation

- Auto generation of documentation – Generates all your code documentation

7.3 Facilitation of V&V

The tool provides testers with internal knowledge of the program to design better tests and more test cases quickly.

The tool provides information on branching to help testers design more complete test coverage based on the code paths found.

The tool gives more information on conditional branching which will provide test engineers data to make boundary value partitioning testing decisions and allow them have more complete data set when testing.

The automatic code documentation generation makes it easier to produce a final code or acceptance documentation package.

UML generation for classes provides a clearer understanding of involved classes in your software. The UML generation for files can help with deployment and make issues to ascertain all file are present in your system.

7.4 Reasoning for Selecting Tool

I chose Informix 4D because it provides coverage of a specific area of the tester's workbench that other products on the market do not cover. Although its use is focused, it may provide a high ROI in the hands of a capable test engineer when designing test cases.

- Ease of Use – Although almost anyone can run it, the tool is geared toward an end user with specific knowledge of c++/c programming.
- Power – It does what it claims. There are not too many other products out on the market that do the same thing.
- Robustness – It is at version 5.4.5 and has been around for at least 5 years. I have not received any errors when using this tool but have only tried it once.
- Functionality –It is a focused product with specific functionality in mind. The control flow analysis and flow charts allow you to focus on what your code is doing.
- Ease of Insertion –This is designed to look at C/C++ code. If that is what your shop supports then it will fit right in
- Quality of Support – n/a/
- Cost of the Tool – Unavailable

- Fit with Organizational Goals and Policies –If you have a level 2 TMM organization or above this tool will help you meet your testing goals and policies by giving you insight into the depths of your code.

The following table shows a comparison of the tools needed for a tester’s workbench versus the features available with this tool.

TMM Level 1	Available
• Interactive debuggers -	
• Configuration Building Tools – e.g. a make utility	
• LOC counters	✓
TMM Level 2	
• Project Planners and test planners	
• Run-Time error checkers – bounds checkers, memory testers, leak testers	
• Test Prep support tools – for development of white and black box testing.	✓
▪ Black box – cause and effect graph or equivalence classes	✓
▪ White box- control flow analyzers that generate control flow graphs, data flow analyzers that support data flow graphs.	✓
▪ Helps identify branches, basis paths, and variable usages. This helps developers to design test cases.	✓
• Coverage analyzers – These assist with white box testing. Tells how much coverage of the code path was given and if the coverage goal was met.	
• Cross-Reference tools – They trace occurrences of items as they appear in different software artifacts, i.e. where does a specific variable occur in all the code	✓
TMM Level 3	
• Requirement Recorders (use case	

recorders)	
• Requirements Verifiers	
• Requirements-to-Test Tracers	
• Capture-Replay Tools – automating the execution of tests and capturing all input and output	
• Comparators – compare actual test outputs to expected	
• Defect Trackers	
TMM Level 4	
• Code checkers - static analyzers	✓
• Auditors – code standard/formatting violations	✓
• Code Comprehension Tools	✓
• Test Harness Generators	
• Performance Testing Tools	
• Network Analyzers	
• Simulators and Emulators	
• Web Testing Tools	
• Test management tools	
TMM Level 5	
• PAL support tools	
• Advanced Test Scripting tools	
• Assertion checkers	
• Advanced Test Data Generators	
• Advanced Test Management Systems	

7.5 Waterfall Phase Usage

Note: VER = VERification, VAL = VALidation

7.5.1 System Requirements

N/A

7.5.2 Software Requirements

N/A

7.5.3 Analysis

N/A

7.5.4 Design

During design, this tool can help facilitate reuse of existing code libraries by documentation of the code base if no documentation existed before.

The flow analysis may also show design defects upon design of high level classes and module.

7.5.5 Coding

N/A

7.5.6 Testing

Testing the software product to verify and validate the software product

In the testing phase the control flow will help you verify your software by giving test engineers a better grasp on how to design tests, when to stop-test, what types of tests to run, and the various boundaries and boundary values to use when testing.

When hunting down defects, having a graph to trace through the code steps will reduce rework time since the developers can pinpoint a problem much faster.

7.5.7 Operations

Deployment, Support, and ongoing Software Maintenance of the software product

The UML generation and documentation generation based on code will assure you of what files need to be included in a make and also provide a formal (with some additional formatting) documentation package that you can provide to your customer.

7.6 Tool Benefit Summary

Overall I think that Imagix 4D is necessary tool for any shop that deals with large or highly complex software. The flow charts and control flow analysis

capabilities give you a great deal of insight into your program not only for test planning, defect prevention, and complexity analysis, but also for documentation of your product.

References

[STP06] – Software Test and Performance Magazine, December 2006

[Chrisis03] Chrisis, M.B.; Konrad, M.; Shrum. S. *CMMI: Guidelines from Process Integration and Product Improvement*, Addison-Wesley, 2003

[Burnstein03] Burnstein, I. *Practical Software Testing*, Spring, 2003

[Royce] Royce, Winston *MANAGING THE DEVELOPMENT OF LARGE SOFTWARE SYSTEMS*, IEEE WESCON, 1970

[Microsoft] <http://msdn2.microsoft.com/en-us/teamsystem/default.aspx>

[Mercury] <http://www.mercury.com/us/products/quality-center/>

Appendix A: Glossary

Term	Definition
Software Verification	Ensuring that selected software work products meet their specified requirements
Software Validation	Demonstration that a software product or software product component fulfills its intended use when placed in its intended environment

Appendix B: Software V&V Company Listing

Company	Tool(s)	Website
Agitar	-Agitator	www.agitar.com
Atlassian Software Systems	-JIRA	www.atlassian.com
AutomatedQA	-AQtime -TestComplete	www.automatedqa.com
Axosoft		www.axosoft.com
Borland	-Silk Central Test Manager -Gauntlet -Silk Test -Silk Performer	www.borland.com/silk
Cenqua	-Clover	www.cenqua.com
Compuware	-QA Center Enterprise Edition -Test Partner -QA Director -Reconcile -Track Record	www.compuware.com
Coverity	-Prevent SQS	www.coverity.com
dynaTrace	-dynaTrace Diagnostics	www.dynatrace.com
Empirix	-e-Test Suite	www.empirix.com
Gomez	Monitoring Services	www.gomez.com
Heroix	-Longitude	www.heroix.com
IBM	-Functional Tester Plus -Purify Plus	www.ibm.com/rational

	-ClearQuest	
ICS	-Qt Testing -Motif Testing	www.ics.com
Instantiations	WindowTester Pro	www.instantiations.com
Imagix	-Imagix 4D	www.imagix.com/
Klocwork	-Klockwork K7	www.klocwork.com
Mercury - HP	-Mercury Quality Center -Mercury TestDirector -Mercury QuickTest Professional -Mercury WinRunner -Mercury Business Process Testing -Mercury Service Test	www.mercury.com/
Microsoft	-Visual Studio Team Edition for Software Testers -Visual SourceSafe -Project Server	www.microsoft.com/teamsystem
MindReef	-SoapScope product line	www.mindreef.com
Parasoft	- Jtest	www.parasoft.com
Perforce	-Perforce	www.perforce.com
Radview	-WebLoad	www.radview.com
Seapine Software	-TestTrack Studio	www.seapine.com

Appendix D: Software V&V Tool Classification Listing and Examples

[STP06]

Classification - Example	Definition
.NET Test/Performance Solution <i>Microsoft Visual Studio Team Edition Software Tester</i> <i>Mercury LoadRunner</i> <i>NUnit</i>	Tools and platforms for .NET developers and test/QA professionals, including ASP.NET
Data Test/Performance Solution <i>Mercury LoadRunner</i> <i>Compuware Vantage</i> <i>Embarcadero Performance Center</i>	Tools for testing, validating and tuning databases and data sets

<p>Defect/Issue Management Solution</p> <p><i>Mercury TestDirector for Quality Center</i></p> <p><i>Microsoft Visual Studio Team Edition for Software Testers</i></p> <p><i>Mozilla Organization's Bugzilla</i></p>	<p>Tools and platform for reporting, assigning and tracking bug reports and change requests</p>
<p>Embedded/Mobile Test/Performance Solution</p> <p><i>Mercury QuickTest Professional</i></p> <p><i>IBM Rational Test RealTime</i></p> <p><i>Coverity</i></p>	<p>Tools and platforms for evaluating the quality of embedded, wireless mobile applications and systems</p>
<p>Functional Test Solution</p> <p><i>Mercury QuickTest Professional</i></p> <p><i>Empirix e-Test Suite</i></p> <p><i>IBM Rational Functional Tester</i></p>	<p>Tools and platforms for evaluating applications and components, including unit, requirements and user-interface testing</p>
<p>Integrated Test/Performance Suite</p> <p><i>Mercury Quality Center</i></p> <p><i>Microsoft Visual Studio Team Edition for Software Testers</i></p> <p><i>Compuware QACenter Enterprise Edition</i></p>	<p>A full Test/QA solution that spans multiple test categories</p>
<p>Java Test/Performance Solution</p> <p><i>JUnit</i></p> <p><i>Mercury LoadRunner</i></p> <p><i>Eclipse Test & Performance Tools Platform</i></p>	<p>Tools and platforms for Java and Java 2 Enterprise Edition developers and test/QA professionals</p>
<p>Load/Performance Test Solution</p> <p><i>Mercury LoadRunner</i></p> <p><i>Microsoft Visual Studio Team Edition for Software Testers</i></p> <p><i>IBM Rational Performance Tester</i></p>	<p>Tools and platforms for determining the performance of Web, client and server applications before or after deployment</p>

<p>SCM/Build Management Solution</p> <p><i>Apache Ant</i></p> <p><i>IBM Rational BuildForge</i></p> <p><i>Catalyst Openmake</i></p>	<p>Tools and platforms for maintaining asset repositories and managing the build and deployment process</p>
<p>Security Test Solution</p> <p><i>Compuware DevPartner Security Checker</i></p> <p><i>Empirix e-Test Suite</i></p> <p><i>Watchfire AppScan</i></p>	<p>Tools and platforms that would be used to identify potential vulnerabilities of applications before or after deployment</p>
<p>SOA/Web Services Test Solution</p> <p><i>Compuware DevPartner Studio</i></p> <p><i>Parasoft SOAtest</i></p> <p><i>Empirix e-Test Suite</i></p>	<p>Tools and platforms for evaluating a Web services stack, including SOAP, XML, WSDL, UDDI and WS-* implementations</p>
<p>Static/Dynamic Code Analysis Solution</p> <p><i>IBM Rational PurifyPlus</i></p> <p><i>Eclipse Test & Performance Tools Platform</i></p> <p><i>Parasoft Jtest</i></p>	<p>Tools and platforms for evaluating the quality of source code and compiled binaries</p>
<p>Test Automation Solution</p> <p><i>Mercury QuickTest Professional</i></p> <p><i>Mercury WinRunner</i></p> <p><i>Borland SilkCentral Test Manager</i></p>	<p>Tools and platforms designed to create, maintain and deploy test-automation systems</p>
<p>Test/QA Management Solution</p> <p><i>Mercury TestDirector for Quality Center</i></p> <p><i>Borland SilkCentral Test Manager</i></p> <p><i>Compuware QADirector</i></p>	<p>Tools and platforms for managing the test process, including test cases, beta programs and reporting</p>