Software Engineering Tools

17

Abstract

This chapter discusses various tools to support the various software engineering activities. The focus is first to define the process and then to find tools to support the process. Tools to support project management are discussed as well as tools to support requirements engineering, configuration management, design and development activities and software testing.

Keywords

Microsoft project · Cocomo · Planview enterprise · IBM Rational DOORS · Rational software modeler · LDRA testbed · Integrated development environment · Sparx Enterprise Architect · HP Quality Center

17.1 Introduction

The goal of this chapter is to give a flavour of a selection of tools[1] that can support the performance of the various software engineering activities. Tools for project management, requirements management, configuration management, design and development, testing and so on are considered. The approach is generally to choose tools to support the process, rather than choosing a process to support the tool.[2]

Mature organizations will employ a structured approach to the introduction of new tools. First, the requirements for a new tool are specified, and the options to satisfy the requirements are considered. These may include developing a tool

[1]The list of tools discussed in this chapter is intended to give a flavour of what tools are available, and the inclusion of a particular tool is not intended as a recommendation of that tool. Similarly, the omission of a particular tool should not be interpreted as disapproval of that tool.

[2]That is, the process normally comes first then the tool rather than the other way around.

G. O'Regan, *Concise Guide to Software Engineering*, Undergraduate Topics in Computer Science, DOI 10.1007/978-3-319-57750-0_17

Table 17.1 Tool evaluation table

	Tool 1	Tool 2	…	Tool *k*
Requirement 1	8	7		9
Requirement 2	4	6		8
…				
…				
Requirement *n*	3	6		8
Total	35	38	…	45

internally, outsourcing the development of a tool to a third-party supplier or purchasing an off-the-shelf solution from a vendor.

The sample tool evaluation process below (Table 17.1) lists all of the requirements vertically that the test tool is to satisfy, and the candidate tools that are to be evaluated and rated against each requirement are listed horizontally. Various rating schemes may be employed, and a simple numeric mechanism is employed for the example below. The tool evaluation criteria are used to rate the effectiveness of each candidate tool and to indicate the extent to which the tool satisfies the defined requirements. The chosen tool in this example is Tool *k* as it is the most highly rated of the evaluated tools.

Several candidate tools will be identified and considered prior to selection, and each candidate tool will be evaluated to determine the extent to which it satisfies the specified requirements. An informed decision is then made, and the proposed tool will be piloted prior to its deployment. The pilot provides feedback on its suitability, and the feedback will be considered prior to a decision on full deployment, and whether any customization is required prior to roll-out.

Finally, the users are trained on the tool, and the tool is rolled out throughout the organization. Support is provided for a period post-deployment. First, we consider a selection of tools for project management.

17.2 Tools for Project Management

There are several tools to support the various project management activities such as estimation and cost prediction, planning and scheduling, monitoring risks and issues, and managing a portfolio of projects. These include tools like Microsoft Project, which is a powerful project planning and scheduling tool that is widely used in industry. Small projects may employ a simpler tool such as Microsoft Excel for their project scheduling activities.

The Constructive Cost Model (Cocomo) is a cost prediction model developed by Boehm [1], and it is used to estimate effort, schedule and cost for small and medium projects. It is based on an effort estimation equation that calculates the software development effort in person-months from the estimated project size. The effort estimation calculation is based on the estimate of a project's size in thousands of

source lines of code (SLOC[3]). The accuracy of the tool is limited, as there is a great deal of variation among teams due to differences in the expertise and experience of the personnel in the project team.

There are several commercial variants of the tool including the COCOMO Basic, Intermediate and Advanced Models. The Intermediate Model includes several cost drivers to model the project environment, and each cost driver is rated. There are over fifteen cost drivers used, and these include product complexity, reliability and experience of personnel as well as programming language experience. The COCOMO parameters need to be calibrated to reflect the actual project development environment. The effort equation used in COCOMO is given by:

$$\text{Effort} = 2.94 * \text{EAF} * (\text{KSLOC})^{\text{E}} \tag{17.1}$$

In this equation, EAF refers to the effort adjustment factor that is derived from the cost drivers, and E is the exponent that is derived from the five scale drivers.[4] The Costar tool is a commercial tool that implements the COCOMO Model, and it may be used on small or large projects. It needs to be calibrated to reflect the particular software engineering environment, and this will enable more accurate estimates to be produced.

Microsoft Project (Fig. 2.2) is a project management tool that is used for planning, scheduling and charting project information. It enables a realistic project schedule to be created, and the schedule is updated regularly during the project to reflect the actual progress made, and the project is re-planned as appropriate. We discussed project management in Chap. 2.

A project is defined as a series of steps or tasks to achieve a specific goal. The amount of time that it takes to complete a task is termed its duration, and tasks are performed in a sequence determined by the nature of the project. Resources such as people and equipment are required to perform a task. A project will typically consist of several phases such as planning and requirements, design, implementation, testing and closing the project.

The project schedule (Fig. 2.2) shows the tasks and activities to be carried out during the project, the effort and duration of each task and activity, the percentage complete of each task and the resources needed to carry out the various tasks. The schedule shows how the project will be delivered within the key project parameters such as time, cost and functionality without compromising quality in any way.

The project manager is responsible for managing the schedule and will take corrective action when project performance deviates from expectations. The project schedule will be updated regularly to reflect actual progress made, and the project re-planned appropriately.

[3]SLOC includes delivered source lines of code created by project staff (excluding automated code generated and also code comments).

[4]The five scale drivers are factors contributing to duration and cost and they determine the exponent used in the Effort equation. Examples include team cohesion and process maturity.

The project manager may employ tools for recording and managing risks and issues, and this may be as simple as using an excel spreadsheet. The project manager may maintain a lessons learned log to record the lessons learned during a project, and these will be analysed towards the end of a project and the lessons learned report prepared. The project reporting may be done with a tool or with a standard Microsoft Word report.

Project portfolio management (PPM) is concerned with managing a portfolio of projects, and it allows the organization to choose the mix and sequencing of its projects in order to yield the greatest business benefit to the organization.

PPM tools analyse the project's total expected cost, the resources required, the schedule, the benefits that will be realized as well as interdependencies with other projects in the portfolio. This allows project investment decisions to be made methodically to deliver the greatest benefit to the organization. This approach moves away from the normal once off analysis of an individual project proposal, to the analysis of a portfolio of projects. PPM tools aim to manage the continuous flow of projects from concept all the way to completion.

There are several commercial portfolio management tools available from various vendors. These include Clarity PPM from Computer Associates, Change Point from Compuware, RPM from IBM Rational, PPM Center from HP and Planview Enterprise from Planview. We limit our discussion in this section to the Planview Enterprise tool.

Planview Enterprise Portfolio Management allows organizations to manage projects and resources across the enterprise and to align their initiatives for maximum business benefit. It provides visibility into and control of project portfolios and allows the organization to prioritize and manage its projects and resources. This allows it to make better investment decisions and to balance its business strategy against its available resources. Planview helps an organization to optimize its business through eight key capabilities (Table 17.2):

Planview allows key project performance indicators to be closely tracked, and these include dashboard views of variances of cost, effort and schedule, which are used for analysis and reporting (Fig. 17.1).

Planview includes Process Builder (Fig. 17.2), which allows modelling and management of enterprise-wide processes. It provides tracking, control and audit capabilities in key process areas such as requirements management and product development, as well as satisfying key regulatory requirements.

The organization may define and model its processes in Process Builder, and this includes process adoption, compliance and continuous improvement. The functionality includes as follows:

- Process design.
- Process automation.
- Process measurement.
- Process auditing.

Table 17.2 Key capabilities of planview enterprise

Capability	Description
Strategic planning	Define mission, objectives and strategies Allocate funding/staffing for chosen strategy Automate and manage strategic process
Investment analysis	Devise strategic long-term plans Identify key criteria to evaluate initiatives Optimize strategic and project investments to maximize business benefit
Capacity management	Balance resources with business demands Ensure capacity supports business strategy Align top-down and bottom-up planning Forecast resource capacity
Demand management	Request work and check status Review lifecycles
Project management	Scope, schedule and execution of work Track/report time worked against projects Track and manage risks and issues Track/display performance and trend analysis
Financial management	Collaborate to better forecast cost Monitor spending
Resource management	Balance portfolios/assign people efficiently Improve forecasting Keep staff productive
Change management	Determine impact of change on schedule/cost Effectively manage change

Fig. 17.1 Dashboard views in planview enterprise

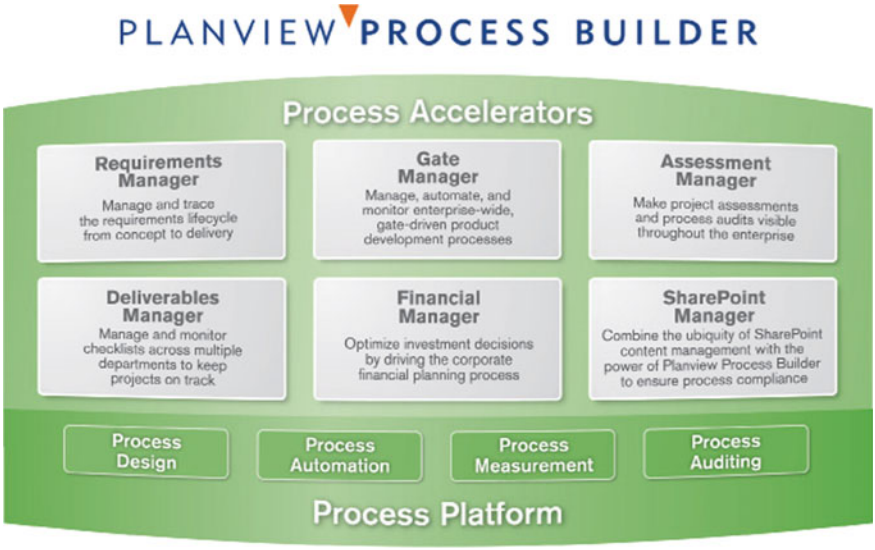

Fig. 17.2 Planview process builder

Next, we will consider tools to support requirements development and management.

17.3 Tools for Requirements

There are several tools available to assist organizations in carrying out requirements development and management. These tools assist in eliciting requirements from the stakeholders, modelling requirements, verifying and validating the requirements, managing the requirements throughout the lifecycle, and providing traceability of the requirements to the design and test cases. The following is a small selection of some of the tools that are available (Table 17.3):

DOORS® (Dynamic Object-Oriented Requirements System) is a requirements management tool developed by IBM Rational. It allows the stakeholders to actively participate in the requirements process and aims to optimize requirements communication, collaboration and verification. High-quality requirements help the organization in reducing costs[5] and in meeting their business objectives.

[5]A good requirements process will enable high-quality requirements to be consistently produced, and the cost of poor quality is reduced as wastage and rework are minimized. The requirements are the foundation of the system, and if they are incorrect then the delivered system will not be fit for purpose.

Table 17.3 Tools for requirements development and management

Tool	Description
DOORS (IBM/Rational)	This is a requirements management tool developed by Telelogic (which is now part of IBM/Rational)
RequisitePro (IBM/Rational)	This is a requirements management and use case management tool developed by IBM/Rational
Enterprise Architect (Sparx Systems)	This is a UML analysis and design tool that covers requirements gathering, analysis and design, and testing and maintenance. It was developed by Sparx Systems and integrates requirements management with the other software development activities
CORE (Vitech)	This is a requirements tool developed by Vitech, which may be used for modelling and simulation
Integrity (MKS)	This tool was developed by MKS and enables organizations to capture and validate software requirements, and to link them to downstream development and testing activities

The tool can capture, link, trace, analyse and manage changes to the requirements. It enhances communication and collaboration to ensure that the project conforms to the customer requirements, as well as compliance to regulations and standards.

Requirements are documented in a way that is easy to interpret and navigate. It is easy to locate information within the database, and the user requirements are recorded in a document style showing each individual requirement. It provides views of the list with assigned identifiers and also an Explorer-like navigation tree.

The tool employs links to support traceability of the requirements, and these are traversed with a simple click of the mouse to the corresponding object. The links are easy to create by dragging and dropping; for example, a new link from the user requirements to the system requirements is created in this way. The tool provides dynamic reporting on traceability, and filters may be employed to ensure that traceability is complete. Traceability is essential in demonstrating that the requirements have been implemented and tested.

The management of change is an important part of the requirements process. The DOORS tool supports changes to requirements and allows an impact analysis of the proposed changes to be performed. It allows changes that could impact other requirements or design items and test cases to be tagged. The DOORS® tool (Fig. 17.3) provides:

- A comprehensive requirements management environment.
- Web browser access to the requirements database.
- Manages changes to requirements.
- Scalable solution for managing project scope and cost.
- Traceability to design items, Test Plans and test cases.
- Active engagement from stakeholders.
- Integrates with other IBM Rational tools.

There are several other IBM Rational tools that may be integrated with DOORS®. These include the IBM Rational System Architect, Requirements Composer, Rhapsody and Quality Manager.

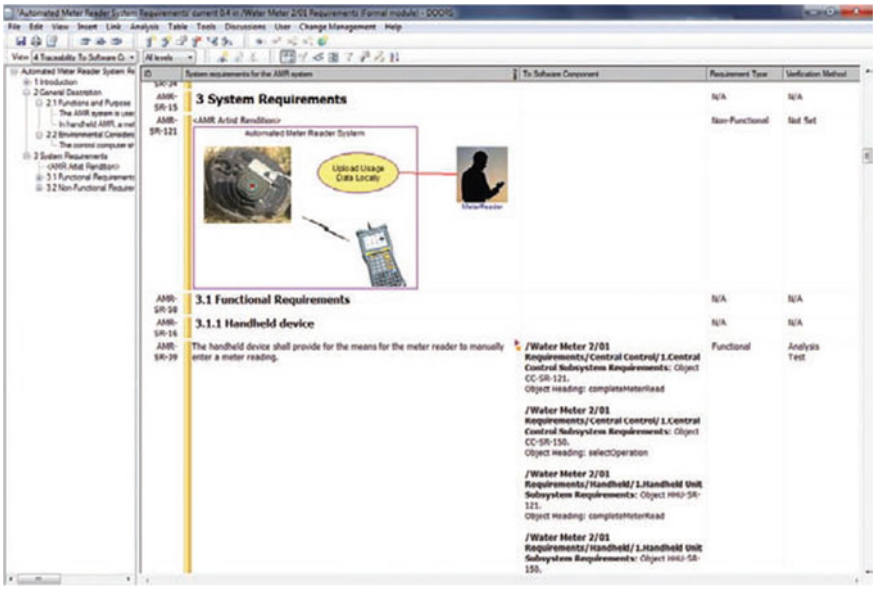

Fig. 17.3 IBM Rational Doors tool

IBM Rational RequisitePro is a requirements management tool that allows requirements to be documented with familiar document-based methods, and it provides capabilities such as requirements traceability and impact analysis. Requirements are managed throughout the lifecycle, and changes to the requirements controlled.

The Core product suite was developed by Vitech, and it has functionality for requirements management, modelling and simulation, and verification and validation. It supports UML activity and sequence diagrams, which are used to describe the desired behaviour and flow of control, as well as allowing analysis to be carried out. The tool provides:

- Comprehensive end-to-end system traceability.
- Change impact analysis.
- Multiple modelling notations with integrated graphical views.
- System simulation based on behavioural models.
- Generation of documentation from the database.

The Integrity tool was developed by MKS, and it enables organizations to capture and validate software requirements. It enables them to link the requirements to downstream development and testing activities, and to manage changes to the requirements. Next, we will consider tools to support software design and development.

17.4 Tools for Design and Development

Table 17.4 describes various tools to support software design and development activities. The software design includes the high-level architecture of the system, as well as the lower level design and algorithms.

Table 17.4 Tools for software design

Tool	Description
Microsoft Visio	This tool is used to create many types of drawings such as flowcharts, work flow diagrams and network diagrams
IBM Rational Software Modeler	This is a UML-based visual modelling and software design tool
IBM Rational Rhapsody	This modelling environment tool is based on UML and provides a visual development environment for software engineers. It uses graphical models and generates code in C, C++ and Java
IBM Rational Software Architect	This modelling and development tool uses UML for designing architecture for C++ and Java applications
Enterprise Architect (Sparx Systems)	This UML analysis and design tool is used for modelling systems with traceability from requirements to design and testing. It supports code generation

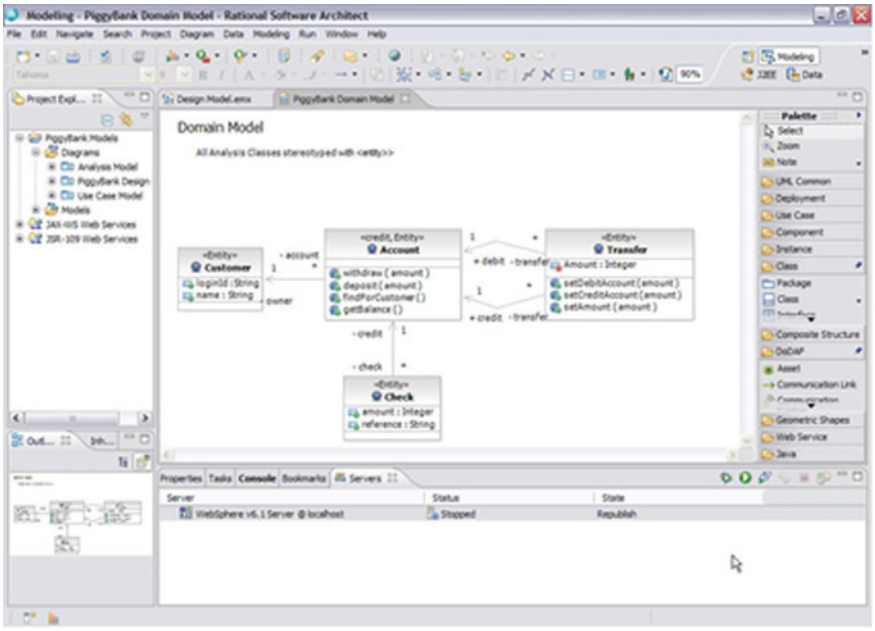

Fig. 17.4 IBM Rational Software Modeler

IBM Rational Software Modeler® (RSM) is a UML-based visual modelling and design tool (Fig. 17.4). It promotes communication and collaboration during design and development and allows information about development projects to be specified and communicated from several perspectives. It is used for model-driven development and aligns the business needs with the product.

It gives the organization control over the evolving architecture and provides an integrated analysis and design platform. Abstract UML specifications may be built with traceability and impact analysis shown.

It has an intuitive user interface and a diagram editor to create expressive and interactive diagrams. The tool may be integrated with other IBM Rational tools such as Clearcase, Clearquest and RequisitePro.

IBM Rational Rhapsody® is a visual development environment used in real-time or embedded systems. It helps teams collaborate to understand and elaborate requirements, abstract complexity using modelling languages such as UML, validate functionality early in development and automate code generation to speed up the development process.

Sparx Enterprise Architect (Fig. 17.5) is a UML analysis and design tool used for modelling business and IT systems. It was developed by the Australian company, Sparx Systems, and it covers the full product development lifecycle, including business modelling, requirements management, software design, code generation and testing. It supports automated document generation, code generation and reverse engineering of source code. Its reverse engineering feature allows a visual representation of the software application to be provided.

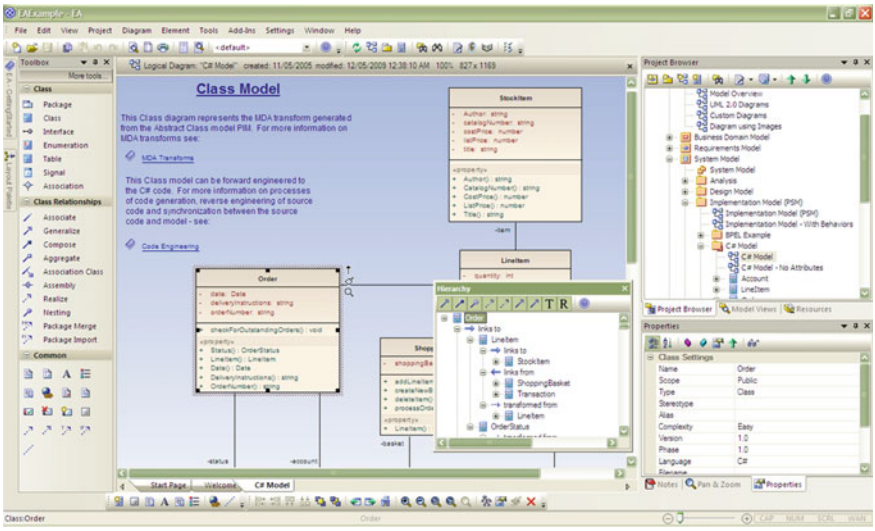

Fig. 17.5 Sparx Enterprise Architect

It is a multi-user graphical tool with built-in reporting and documentation. It can model, manage and trace requirements to the design, test cases and deployment, and it can trace the implementation of the system requirements to model elements. It can search and report on requirements and perform an impact analysis on proposed changes to the requirements.

The tool allows deployments scripts to be built, debugged and tested and executed from within its development environment. UML and modelling are integrated into the development process, and debugging capabilities are provided. This includes runtime examination of the executing code for several programming languages, and NUnit and JUnit test classes (used as part of test-driven development) may be generated and integrated directly into the test process.

An integrated development environment (IDE) is a software application that provides comprehensive support facilities to software developers. It includes specialized text editors, a compiler, build automation and debugging capabilities. The features of an IDE are described in Table 17.5 below:

IDEs help to improve programmer productivity. They are usually dedicated to a specific programming language, although there are some multi-language tools such as Eclipse and Microsoft Visual Studio. There are many IDEs for languages such as Pascal, C, C++ and Java. The next section is concerned with tools to support configuration management.

Table 17.5 Integrated development environment

Item	Description
Source code editor	This is a specialized text editor (e.g. Microsoft Visual Studio) designed for editing the source code. It includes features to speed up the input of source code, including syntax checking of the code while the programmer types
Compiler or interpreter	A compiler is a computer program that translates the high-level programming language source code into object code to produce the executable code. A compiler carries out lexical analysis, parsing and code generation An interpreter is a program that executes instructions written in a programming language. It may involve the direction execution of the code, translation of the code into an intermediate representation and immediate direct execution, or execution of stored precompiled code made by a compiler which is part of the Interpreter System
Build automation tools	Build automation involves scripting to automate the build process. This includes tasks such as compiling the source code, linking the object code and building the executable software, performing automated tests and reporting results, reporting the build status and generating release notes
Debugger	A debugger is a software application that is used to debug and test other software programs. Debuggers offer step-by-step execution of the code, or execution to break points in the code. Examples include IBM Rational Purify and Microsoft Visual Studio Debugger

17.5 Tools for Configuration Management and Change Control

Configuration management is concerned with identifying the work products that are subject to change control, and controlling changes to them. It involves creating and releasing baselines, maintaining their integrity, recording and reporting the status of the configuration items and change requests, and verifying the correctness and completeness of the configuration items with configuration audits.

Visual Source Safe (VSS) is a version control management system for source code and binary files. It was developed by the Microsoft Corporation and is used mainly by small software development organizations. It allows multiple users to place their source code and work products under version control management. It is fairly easy to use and may be integrated with the Microsoft Visual Studio tool. Microsoft plans to replace VSS with its Visual Studio Team System tool.

Polytron Version Control System (PVCS) is a version control system for software code and binary files. It was developed by Serena Software Inc. and is suitable for use by large or small teams. It allows multiple users to place their source code and project deliverables under version control management, and it allows files to be checked in and checked out, baselines to be controlled, rollback of code and tracking of check-ins. It includes functionality for branching, merging and labelling. It includes the PV Tracker tool for tracking defects, and the PV Builder tool for performing builds and releases.

The PV Tracker tool automates the capture and communication of issues and change requests. This is done throughout the software development lifecycle for project teams, and the tool allows the developers to link the affected source code files with issues and changes. It allows managers to determine and report on team progress, and to prioritize tasks. PV Builder maintains an audit trail of the files included in the build as well as their versions.

IBM Rational Clearcase and Clearquest are popular configuration management tools with a rich feature set. Clearcase allows software code and other software deliverables to be placed under version control management, and it may be employed in large or medium projects. It can handle a large number of files, and it supports standard configuration management tasks such as checking in and checking out of the software assets as well as labelling and branching. Objects are stored in repositories called VOBs.

Clearquest may be linked to Clearcase and to other IBM Rational tools. It allows the defects in a project to be tracked, and it allows the versions of source code modules that were changed to be linked to a defect number in Clearquest.

17.6 Tools for Code Analysis and Code Inspections

Static code analysis is the analysis of software code without the actual execution of the code. It is usually performed with automated tools, and the analysis performed depends on the sophistication of the tools. Some tools may analyse individual

statements or declarations, whereas others may analyse the whole source code. The objective of the analysis is to highlight potential coding errors early in the development lifecycle.

The LDRA Tools automatically determine the complexity of the source code and provide metrics that give an indication of the maintainability of the code. A useful feature of LDRA is that it gives a visual picture of system complexity, and it has a re-factoring tool to assist with its reduction. It generates code assessment reports listing all of the files examined and providing metrics of the clarity, maintainability and testability of the code. Other LDRA tools may be used for code coverage analysis (Fig. 17.6).

Compliance to coding standards is important in producing readable code and in preventing error-prone coding styles. There are several tools available to check conformance to coding standards including the LDRA TBvision tool, which has reporting capabilities to show code quality as well as fault detection and avoidance measures. It provides intuitive functionality to view the results in various graphs and reports.

Some static code analysis tools (e.g. tools for formal methods) aim to prove properties about a particular program. This may include reasoning about program correctness or that of a program meeting its specification. These tools often provide support for assertions, and a precondition is the assertion placed before the code fragment, and this predicate is true before execution of the code. The post-condition is the assertion placed after the code fragment, and this predicate is true after the execution of the code.

	Percentage	Success Limit
Productdatabase.cpp		
Combined Coverage Run	Failed	
Statement Coverage	99	100
Branch/Decision Coverage	94	100
Modified Condition / Decision Coverage	75	100
main		
ProductDatabase		
Combined Coverage Run	Passed	
Statement Coverage	100	100
Branch/Decision Coverage	100	100
Modified Condition / Decision Coverage	100	100
resetCountedProducts		
Combined Coverage Run	Passed	
Statement Coverage	100	100
Branch/Decision Coverage	100	100
Modified Condition / Decision Coverage	100	100
countProduct		
Combined Coverage Run	Failed	
Statement Coverage	97	100
Branch/Decision Coverage	86	100
Modified Condition / Decision Coverage	50	100

Fig. 17.6 LDRA code coverage analysis report

There are several open-source tools available for static code analysis, and these include the RATS tools which provide multi-language support for C, C++, Perl and PHP, and the PMD tool for Java. There are several commercial tools available, and these include the LDRA Testbed tool which provides support for C, C++ and Java. The fortify tool helps developers to identify security vulnerabilities in C, C++ and Java, and the Parasoft tool helps developers to identify coding issues that lead to security, reliability, performance and maintainability issues later.

17.7 Tools for Testing

Testing plays a key role in verifying that the software system satisfies the requirements and is fit for purpose. There are various tools to support testing such as test management tools, defect tracking tools, regression test automation tools and performance tools. The tools considered in this section include as follows:

- Test Director (HP Quality Center).
- Winrunner.
- Load Runner.

Test Director (now called HP Quality Center) is a web-based test management tool developed by HP Mercury.[6] It provides a consistent repeatable process for gathering requirements, planning and scheduling tests, analysing results and managing defects. It consists of four modules namely:

- Requirements.
- Test Plan.
- Test Lab.
- Defect management.

The Requirements module supports requirements management and traceability of the test cases to the requirements. The Test Plan module supports the creation and update of test cases. The Test Lab module supports execution of the test cases defined in the Test Plan module. The Defect Management module supports the logging of defects, and these defects can be linked back to the test cases that failed.

HP Quality Center supports a high level of collaboration and communication between the stakeholders. It allows the business analysts to define the application requirements and testing objectives. The test managers and testers may then design Test Plans, test cases and automated scripts. The testers then run the manual and

[6]Mercury is now part of HP.

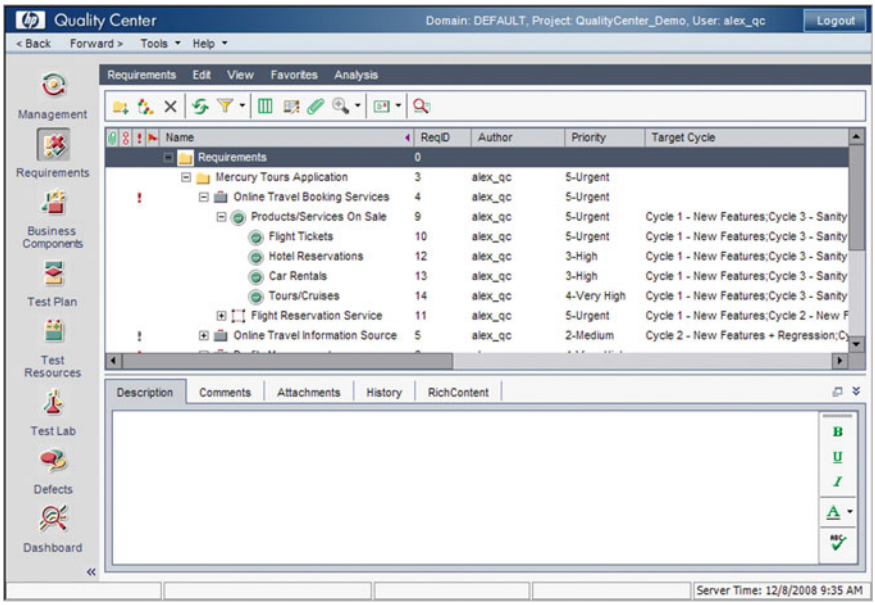

Fig. 17.7 HP Quality Center

automated tests, report results and log the defects. The developers review and correct the logged defects. Project and test managers can create status reports and manage test resources. Test and product managers decide objectively whether the application is ready to be released.

The HP Quality Center™ tool (Fig. 17.7) standardizes and manages the entire test and quality process and is a web-based system for automated software quality management and testing. It employs dashboard technology to give visibility into the process.

Mercury developed the Winrunner tool that automatically captures, verifies and replays user interactions. It is mainly used to automate regression testing, which improves productivity and allows defects to be identified in a timely manner. This provides confidence that enhancements to the software have had no negative impact on the integrity of the system. The Winrunner tool has been replaced by HP Unified Functional Testing Software, which includes HP Quick Test Professional and HP Service Test.

Mercury developed the LoadRunner performance testing tool, which allows a software application to be tested with thousands of concurrent users to determine its performance under heavy loads. It allows the scalability of the software system to be determined, and whether it can support future predicted growth.

17.8 Review Questions

1. Why are tools used in software engineering?
2. How should a tool be selected?
3. What is the relationship between the process and the tool?
4. What tools would you recommend for project management?
5. Describe how you would go about selecting a tool for requirements development.
6. Describe various tools that are available for design and development.
7. What tools would you recommend for testing?
8. What tools would you recommend for configuration management?

17.9 Summary

The objective of this chapter was to give a flavour of various tools available to support the organization in engineering software. These included tools for project management, configuration management, design and development, test management. The tools are chosen to support the process.

The project management tools included a discussion of the COCOMO Cost Model, which may be employed to estimate the cost and effort for a project, and the Microsoft Project tool, which is used extensively by project managers to schedule and track their projects. The Planview Portfolio Management Tool was also discussed, and this tool allows an organization to manage a portfolio of projects.

The tools to support requirements development and management included IBM Rational DOORS, RequisitePro and CORE. The DOORS tool allows all stakeholders to actively participate in the requirements process and aims to optimize requirements communication, collaboration and verification.

The tools to support design and development included the IBM Rational Software Modeler tool, the Sparx Enterprise Architect tool and Integrated Developer Environments to support software developers. The Rational Software Modeler® (RSM) is a UML-based visual modelling and design tool. Enterprise Architect is a UML analysis and design tool and provides traceability from requirements to design, testing and deployment. The tools discussed to support configuration management included PVCS and Clearcase.

The tools to support testing included Quality Center™, Winrunner and LoadRunner. HP Quality Center™ standardizes and manages the entire test process. It has modules for requirements management, test planning, Test Lab and defect management.

Tool selection is done in a controlled manner. First, the organization needs to determine its requirements for the tool. Various candidate tools are evaluated, and a decision on the proposed tool is made. Next, the tool is piloted to ensure that it meets the needs of the organization, and feedback from the pilot may lead to changes or customizations of the tool. Finally, the end-users are trained on the use of the tool, and it is rolled out throughout the organization.

Reference

1. B. Boehm, *Software Engineering Economics* (Prentice Hall, New Jersey, 1981)