

IPL Testing Tools in the ISO9001/BS5750/TickIT/ISO9000-3 Software Quality Process

Executive Summary

ISO9001, EN29001, and BS5750 Part 1 are equivalent standards relating to quality in industry in general. Their purpose is to establish some ground rules whereby purchasers of a product or service can have some confidence in the quality of what they buy. Suppliers are required to develop and follow standards covering all stages of the production process. Purchasers are similarly given guidance on what to look for when choosing a supplier

TickIT, a program originally sponsored by the UK Department of Trade and Industry (DTI), aims to establish guidelines for the application of the above standards to the software industry. Software is considered in its widest definition, from user-applications such as accountancy packages and washing machine controllers, to system software such as database packages and operating systems. TickIT is now known as ISO 9000-3.

Cantata++ (for C and C++) and AdaTEST (for Ada) are software testing and verification packages developed by IPL primarily for application at the unit and integration levels of testing. They combine facilities for dynamic testing, test coverage analysis and static analysis in a single, easy to use toolset.

IPL is an independent software house founded in 1979 and based in Bath. IPL was accredited to ISO9001 in 1988, and gained TickIT accreditation in 1991. Both Cantata++ and AdaTEST have been produced to these standards.

Copyright

This document is the copyright of IPL Information Processing Ltd. It may not be copied or distributed in any form, in whole or in part, without the prior written consent of IPL.

*IPL
Eveleigh House
Grove Street
Bath
BA1 5LR
UK
Phone: +44 (0) 1225 475000
Fax: +44 (0) 1225 444400
email: ipl@iplbath.com*



Certificate Number FM1589

*Last Update: 30/05/2002 11:49
File: TICKIT.DOC*

REQUIREMENTS OF THE STANDARDS

The standards require that quality is ensured at all stages and levels of the production process. The aim is to provide a purchaser with "**confidence that... quality is being built into the product, as well as ensuring that the right product is being built**" (TickIT Guide, page 3.20).

While the standards obviously cover general aspects such as the need for a quality policy, a quality management system (QMS), design control, etc, a significant component is validation, verification and testing. For quality of software to be "built-in" this requires testing to be carried out at all stages of the development process.

No particular life-cycle model is taken as the norm for the TickIT standard, but the 'V' model, illustrated over the page, serves usefully to describe what is meant. The Supplier's section of the TickIT Guide considers three useful stages where validation, verification and testing activities can be applied:

Code and Unit Test (Section 4.5.7)

Integration and System Test (Section 4.5.8)

Acceptance Test (Section 4.5.9)

What now follows is a walkthrough of some of the main issues in the relevant sections of the standards, with a commentary on how Cantata++ and AdaTEST might be used to help.

1. **The development plan should identify the verification procedures to be carried out at each phase of the development process.** The issues at stake here revolve around a decision on at which levels testing should be applied, and what types of test. It is a convenient point to consider whether or not unit testing is to be carried out, and if so should it be done using 'isolation' tests, or are 'bottom-up' tests acceptable?

Cantata++ and AdaTEST are very effective in providing for tests at the unit and integration levels. Tests can be based round 'isolated' units using the sophisticated stubbing mechanism provided by the tools, or alternatively around integrated (presumably pre-tested) units.

2. **Test plans should be reviewed, and the test completion (pass/fail) criteria should be established.** The essence of real testing, as opposed to unstructured fault-finding, is planning. The best time for test plans to be created and reviewed is when specifications are being created. Thus, a software specification ought not just say what it should do and how it does it, but also **how it is to be tested.**

Cantata++ and AdaTEST focus the test designers' minds on organising dynamic tests around 'test cases', in each of which initial conditions are established, code is called, and the results are automatically checked. The test results are generated automatically given the pass/fail criteria, and an 'Overall Test Result' is delivered.

3. **Test results should be recorded and reviewed.** It is a fact that is often overlooked that the best tests are those where the full set of results as well as a summary are available for inspection. In this way, independent verification effort can be applied, and dependence on the test team's personal word that, "it has been tested", is considerably reduced. Moreover, it is suggested in the standards that **purchasers could usefully involve themselves in the inspection of test results at levels prior to Acceptance Tests.**

Cantata++ and AdaTEST provide fully documented test output, on a test case by test case basis. All 'failures' are highlighted at the point in the test where they occur, together with suitable diagnostics. The test review teams can choose between summary results and full results when they are auditing tests.

4. **Coding and testing standards should be set and monitored.** Coding standards define the allowed and preferred use of the source language, while testing standards mandate both the form of the test and the required test coverage. Testing standards also define the required method of testing.

Cantata++ and AdaTEST lend themselves to the verification of the way the source language is used. They achieve this through the static analysis of source code files, and the production of metrics relating to the use of code and complexity. The results can be either reviewed manually or, better, built into automatic checks to be carried out as part of the testing process.

Cantata++ and AdaTEST are ideal for checking that test coverage standards are being met. A variety of coverage measurements can be taken during dynamic test runs, and the results are available immediately. These can then be built into the pass/fail criteria for the test as a whole.

5. **During the maintenance phase there will be a need to repeat tests.** Following every modification to the code there should follow 'regression tests', which involve repeat testing at all levels where the affected code might have any kind of impact. Frequently the effort of doing this is a major disincentive to the work being done.

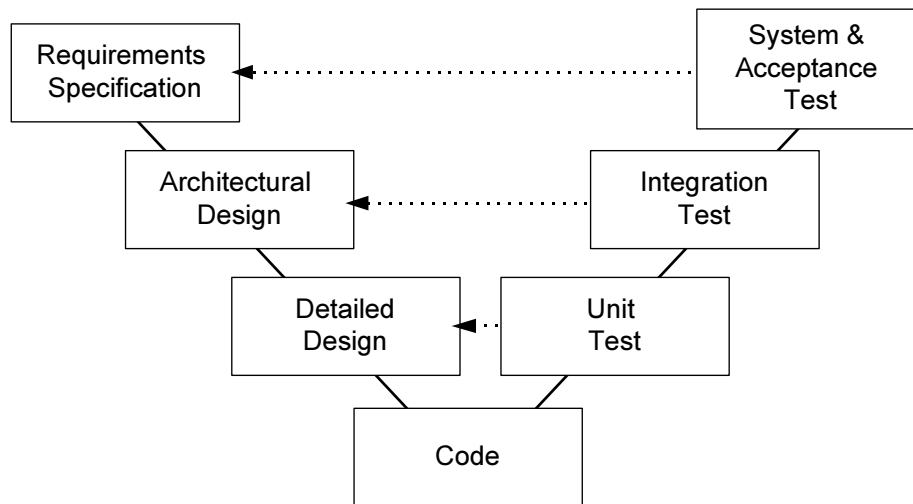
Cantata++ and AdaTEST tests are readily repeatable, and can be batch run overnight with automatic reporting of any discrepancies.

6. **The tools used for testing and verification must be reviewed.** The standards strongly encourage the active formulation of a policy on all tools used in the development process including testing tools. Furthermore, they suggest that "the standards have to be traceable to a recognised national or international standard". In general there is considerable and justifiable emphasis on all contributors (suppliers, sub-contractors, materials, tools, etc) to be at a comparable level of quality.

Cantata++ and AdaTEST were both produced to the ISO9001 and TickIT standards. Their continued maintenance and development will, likewise, follow these standards.

THE SOFTWARE LIFECYCLE - 'V' MODEL

All software standards encourage the modelling of the software development process as a series of phases or levels. Different quality activities are applicable at each of these different stages. While no particular model is perfect, the 'V' model, shown here, is a useful way to illustrate the development process.



The process starts at the top left corner with a specification of the system requirements. From this, successively lower levels of specification and design take place and are reviewed. As the final stage of this process, code is produced (bottom of the diagram).

The system is then progressively integrated by testing individual components and building them into higher level objects, which are in turn tested. The key element of the diagram here are the horizontal dotted lines, indicating the dynamic testing of components against their specification. Thus, testing of software components is envisaged as a planned activity, rather than an ad hoc process.

The final step is of course Acceptance Testing. Purchasers will clearly want to be involved in this, but it is reasonable of them to expect that all previous levels of testing have been carried out, and that the results of these tests are available for inspection. The solution for suppliers is to make life easy for themselves by **building quality into the product**, by thorough reviewable testing at all stages.