AdaTEST, Cantata

and the

SEI Capability Maturity

Model (CMM)

Executive Summary

This paper describes how AdaTEST and Cantata can be used within the CMM framework, and details how they can be used to assist an organisation's transition from one CMM level to another. The conclusion is drawn that AdaTEST and Cantata are most effective in aiding the transition from Level 2 (Repeatable) to Level 3 (Defined), specifically by providing direct support for the software unit testing activity within the Software Product Engineering KPA. The contributions of AdaTEST and Cantata in moving from Levels 3 to 4, and then 4 to 5 are also described.

AdaTEST and Cantata are 'low-level' software testing tools, for Ada and C/C++ respectively. They are produced and marketed by IPL, which is an independent software house founded in 1979, based in Bath, England. IPL was accredited to ISO9001 in 1988, and gained TickIT accreditation in 1991.

1. Introduction

The Capability Maturity Model produced by the Software Engineering Institute of Carnegie-Mellon University, Pittsburgh, USA, offers the means to assess an organisation's ability to produce good quality software. The CMM can be used by an external assessor to determine whether the subject organisation is suitable for undertaking software development sub-contracts. Alternatively, it can be applied by the organisation itself to help discover its software process weaknesses and thereby identify areas for improvement.

The CMM posits five capability 'levels' ranging from One ('Initial') to Five ('Optimizing'). These describe organisations ranging in ability from those with no guaranteed success in producing software (though might achieve success because of talented individuals), to those where the software process is not only well-defined and effective but which have also institutionalised an ongoing improvement program. To qualify for a given level an organisation must demonstrate minimum competence in a number of Key Performance Areas (KPAs), which contain detailed accounts of various software activities.

AdaTEST and Cantata are software testing tools, and the purpose of this paper is to cross-reference their functionality with the requirements of the KPA's at the different CMM levels (not counting Level 1, which has no KPAs). If applied sensibly this guide will illustrate how an organisation can use AdaTEST and Cantata to improve their overall capability from one CMM level to another.

The paper consists of a short description of AdaTEST and Cantata, and a detailed analysis of the various CMM KPA's (only those where the tools are considered to help). The particular contributions of AdaTEST and Cantata in meeting the KPA requirements are supplied in italics. Finally, a conclusion provides a summary of the analysis, spelling out the main value of the tools.

The CMM version referred to in this paper is 1.1.

2. A General Description of AdaTEST and Cantata

AdaTEST and Cantata can be described as software testing tools specifically aimed at the 'low-levels' of the testing process. That is to say they would most typically be used at the module/unit/component testing stage, and then during 'early integration' for task/program testing. The tools provide nearly identical functional for specific programming languages: AdaTEST for Ada, and Cantata for C and C++.

The facilities and features of AdaTEST and Cantata can be summarised as follows:

- **Dynamic Testing:** demonstrates that the Software Under Test (SUT) performs according to its specification. Test scripts (test drivers) can either be written longhand in the native language (Ada, C or C++) using the tools' test harness library or generated from user-supplied test case data. Either way, the user is responsible for setting up defined test cases, which specify known inputs to the SUT and predicting the expected outputs. Using the script generation approach, stubs are automatically generated for external calls, and all global data is automatically checked for accidental corruption (i.e. unpredicted updates). Test runs generate complete result files. Tests can be run in both 'host' and 'target' environments.

- **Test Coverage Analysis:** provides proof that the source code has been executed (or covered) by the test cases. Many different types of coverage are supported by AdaTEST and Cantata ranging from the simplest (Statement coverage) to the most detailed (e.g. Condition, Call-Pair, and Data Value coverage). Coverage information can be output in different forms ranging from a simple numerical report to a detailed statistical breakdown. Coverage can also be supplied in graphical (Flowgraph) format.

- **Static Analysis:** AdaTEST and Cantata offer access to a large number of static 'metrics'. These would typically be used as input to code or post-implementation reviews. The metrics cover such aspects as code construct usage, size, complexity, and structuredness of the code. Reports can be generated in a wide variety of different formats to suit different purposes.

AdaTEST and Cantata have themselves both been produced to software quality standards approved for ISO9000-3. AdaTEST has additionally been audited as a tool suitable for use on safety-critical projects, as defined by the RTCA DO-178B standard.

The software testing 'philosophy' which AdaTEST and Cantata are intended to support can be summarised in a few short phrases:

- Systems built from **thoroughly tested software components** will be more reliable than those which are not.

- Software testing needs to be a **repeatable** and **automated** activity, though software test design tends to remain a 'creative' activity, best carried out by the software designer.

- Software testing should be done in a manner such that **test inputs can be reviewed** and **test outputs are readable hard-copy**. Test scripts should be treated as a (version-managed) **reusable resource**.

3. CMM Analysis

The following pages offer a breakdown of the CMM starting at Level 2, and focusing on the Key Process Areas (KPAs) where there are implications for the need for low-level software testing. Using KPA 'Activities' as the main cues, the specific details of these requirements are spelt out. Following these statements there is a short description in italics of the role of AdaTEST/Cantata in assisting satisfying these needs.

LEVEL 2. Repeatable

KPA Software Project Planning

Activity 5 states decisions have been taken regarding the development process to be used and hence may have specified the inclusion of low-level testing; unit and integration testing may have been specified as "pre-defined stages of manageable size"

Activity 7. If low-level testing is to be carried out, there will need to be procedures and standards to cover this activity.

Activity 14. If low-level testing is to be carried out, plans for the provision of tools will be required.

Low-level testing is an activity generally recognised to be a worthwhile software engineering step in any process model. The incorporation of fully-tested components into the finished product is one of the best ways of promoting that product s chances of being reliable. However, many projects will shy away from incorporating formal unit testing into the project plan on the grounds of productivity i.e. they will claim it takes "too long" or is "too expensive".

AdaTEST and Cantata, through their productivity enhancing features, remove this excuse. Through the coverage analysis features of the tools it becomes possible to set objective standards for the low-level testing process.

KPA Software Project Tracking

Activities 8 & 9. If low-level testing is to be carried out, provision must be made to monitor its completion, using the agreed standards. Test failures will need to be tracked to closure.

AdaTEST and Cantata are readily configurable to generate hard-copy evidence of conformance to test coverage standards. In the most accessible form, they will simply cause a test to fail until it meets the required standard.

KPA Software Project Subcontract Management

Activity 1. The documented procedure agreed with the sub-contractor may specify that low-level testing is to be carried out, to agreed procedures and standards.

Activity 3. The software products to be delivered may include copies of the low-level tests as well as the code. The acceptance criteria (e.g. coverage analysis) may have to be shown to have been met.

Activity 10. Software QA staff, on both sides, may need to monitor software testing steps.

AdaTEST and Cantata are the ideal tools to agree between a main and a sub-contractor to ensure a no-argument acceptance criteria for the hand over of tested code. The hard-copy evidence of test results is available to show that the code units perform as specified, while the coverage results can prove that testing was completed to the standard required. Finally, if size, and complexity of code units form part of the sub-contract standards, these can also be subjected to objective scrutiny using the AdaTEST and Cantata static analysis facilities. The test scripts, as ASCII files, can easily be version-managed and included in the project deliverables.

KPA Software Quality Assurance

Activity 3. If the software development plan has specified that low-level testing will be carried out, then standards and procedures for these steps must be defined.

Activities 4 & 5. The software products which result from the engineering activities need to be monitored for conformance to the agreed standards. Deviations must be tracked to closure.

AdaTEST and Cantata provide a convenient medium for specifying low-level testing standards in terms of both the quality and the quantity of testing. The emphasis on the generation by the developers of readable (hence auditable) test input scripts, and the generation by the tools of readable output offer a unique opportunity to move into a world of properly managed software testing. The auditable, pass/fail nature of the test results should mean the avoidance of disputes as whether a given unit has completed testing or not.

LEVEL 3. Defined

KPA Organisation Process Focus

Activities 1 & 2 suggest that the organisation's standard process should be reviewed periodically and improved in the light of findings. One area specifically named (Ability 1) is Software Testing.

Activity 5 states that successful methods, procedures and tools should be actively transferred through an organisation.

Many organisations have found that software testing is an area of weakness in their process. Use of AdaTEST/Cantata (and the disciplines that go with their successful use) helps strengthen the development process in this area. Use of the tools usually spreads from project to project.

KPA Organisation Process Definition

Activities 1 & 2 suggest that the organisation's standard process should be documented and that standards for its use should be created. Software Testing is a specifically mentioned discipline.

Activity 3 refers to some well-known software life cycle models. Most of these have a location for testing (though surprisingly the V-Model is not listed).

Use of AdaTEST and Cantata form a very convenient focus for successful procedures and standards in the area of low-level software testing once an organisation has committed to this in its process.

KPA Training Program

Activities 1 & 2 suggest that the training should be provided to meet the organisation's software engineering needs. Software Testing should be among the skills supported.

IPL runs training courses covering the activity of Unit Test Planning , as well as specific courses relating to the use of AdaTEST and Cantata.

KPA Integrated Software Management

Activities 1 through 4 state that a project's process and plan be derived from the organisation's standard through a set of documented procedures. If the standard process includes a role for low-level software testing can a project be justified in leaving it out?

If a project has elected to do unit testing it will be advised to look at tool support. If the project uses Ada, C or C++, then AdaTEST and Cantata were specifically designed for that role.

KPA Software Product Engineering

Activities 1 through 3 detail that a thorough software engineering process must exist and include testable requirements and verifiable design.

Activity 4 is explicit on the need for unit testing of code.

Activity 5 details some of the various approaches to different levels of software testing, including the unit and integration levels. It mentions white-box and black-box testing approaches, and specifies various coverage types (statement, path, branch, etc.) which can be used as testing standards. The need for tools is mentioned in Ability 1, and the need for training in test planning and testing techniques is detailed in Ability 2.

AdaTEST and Cantata are tools designed specifically for unit and integration testing activities. They permit various approaches to testing including black-box and white-box. The coverage analysis facilities of AdaTEST and Cantata support all the main coverage types. IPL offer training in unit test planning, techniques and use of the tools.

(Activity 5) Integration testing should not begin until all relevant code has passed its unit tests.

A clear pass criterion is needed to determine when a code unit has passed its test. The coverage analysis checks of AdaTEST and Cantata provide this in very convenient form.

(Activity 5) The necessity for regression testing at each level whenever changes are introduced is made clear.

With AdaTEST and Cantata one of the aims is to produce repeatable tests, which means that regression unit test runs are very easy to set up. A clean statement of pass or fail can reached with very little effort.

(Activity 5) The requirement for reviews of test plans and test cases is stated, as is the need to manage and control them.

Test plans should be reviewed to ensure they are satisfactory and then the test cases can be implemented in the form of AdaTEST or Cantata test scripts. The tools offer every opportunity to create readable (hence reviewable) test cases. The scripts are stored as text files, making them easy to version and configuration manage.

Activity 10 states that traceability should be maintained between requirements and software tests.

If necessary requirements can be traced to individual AdaTEST/Cantata test cases.

KPA Peer Reviews

Activity 2 suggests that peer reviews should be carried out on code units and their associated tests.

AdaTEST and Cantata can provide static reports on code units to aid the peer reviews of code. The test scripts should be readable, to ensure that they too are reviewable.

LEVEL 4. Managed

KPA Quantitative Process Management

Activity 4 states that process data will be collected. Examples of this process data include test coverage and software defect information.

Activity 7 mentions that process areas likely to be ripe for improvement include 'quality-oriented activities such as peer reviews and testing'.

Information on software defects and test coverage can be obtained very directly from appropriate use of AdaTEST/Cantata. Peer reviews and testing are both improved through use of AdaTEST/Cantata, for reasons already discussed.

KPA Software Quality Management

Activities 3 & 4 state that testing is one of the ways of getting measurements on software product quality.

An appropriately controlled testing phase, using AdaTEST and Cantata, is an excellent way of finding out how effectively software design and implementation are being carried out. Frequent rework, with corresponding need to retest, are indicators that the design and code stages possibly need more attention.

LEVEL 5. Optimizing

KPA Defect Prevention

Activity 5 states that defect prevention data will be documented.

Testing is of course one of the prime methods of detecting defects in software. It is most effective when test runs generate hard-copy evidence, complete with diagnostics on test failures. Test result trends can typically be analysed back to expose defects in design for testability, weak coding practices, or inadequate peer reviews.

AdaTEST and Cantata will automatically generate hard-copy evidence on every test run. Whether a project chooses to log each test run is an option open to the project but the data are there for collection.

4.   Conclusion

It should be clear from the preceding analysis that the single biggest contribution of AdaTEST and Cantata in the CMM is in helping to meet the requirements of the Software Product Engineering KPA in Level 3. This KPA is quite specific about the need to include software unit testing as a planned, controlled and managed activity within an organisation's standard software engineering process. The implication is that any organisation which is not currently doing unit testing will need to start doing so if it is to progress from Level 2 to Level 3. Since AdaTEST and Cantata are equally specific in being tools to support the activity of unit testing, there would seem to be a good fit.

At Level 2 unit testing can be regarded as an option. If it is included then provision must be made for it in the form of standards, procedures and software QA activities. AdaTEST and Cantata can then help with these. Equally, the tools can provide a useful means of monitoring sub-contract software development activities.

At Level 3, in addition to the Software Product Engineering KPA contributions already mentioned, AdaTEST and Cantata can provide support for a variety of other KPA activities.

Moving from Level 3 to 4, and then from Level 4 to 5, the main contribution of the tools is in the generation of objective software quality data which can be used to improve the overall process.