Test Coverage and Traceability

July 2007 - Pragmatic Software Newsletters

Upon embarking on a new software development project, it is important to ensure that the software will be thoroughly testing during the Quality Assurance phase. But how do you know if you have adequate test cases for each requirement in the new software?

Traceability

Traceability is a term that refers to the ability to review all the test cases you have defined for each requirement specification. Traceability also includes the ability to review any defects that have been associated with failed test cases. By reviewing traceability, you can ensure that each requirement has an adequate number (and level) of testing, and you can determine which requirements spawned the most defects during the testing process.

Requirements Review

Quality starts with requirements definition. Upon defining a requirements specification, it is important to have a team review of the requirement specification. The owner of the requirement specification should distribute the requirement specification to all team members for review (this includes the project manager, analyst that created the specification, the programmer(s) that will perform the coding, and the tester(s) that is responsible for testing the new requirement). Prior to the meeting, each person should have already reviewed the requirement in detail and should come prepared with comments for the review. Below are things that should be reviewed:

1. **Completeness** - Is the requirement defined in enough detail to code and test against?
2. **Accuracy** - Is the requirement accurately defined and logical? Are there missing elements?
3. **Testable** - Is the requirement testable (can a complete set of test cases be written for it)?

Test Case Review

Once the requirement specification has been reviewed and approved by the reviewers, the test team should begin developing test cases. As each test case is created, it should be identified as belonging to the requirement specification. This ensures that you can quickly determine exactly what test cases belong to each specification, which provides traceability. Once all test cases have been defined for the requirement specification, the tester should distribute the test cases to all team members for review (the same people that attended the requirements review). Like the requirements review, the reviewers should review the test cases in detail and should come prepared with comments for the review. Below are the things that should be reviewed:

1. **Positive Test Cases** - Positive test cases are designed to test the feature per

the design. It does not try to break the feature, it is simply test cases that demonstrate how a user might use the software in a normal environment. The review should ensure that there are an adequate number of test cases for this.

2. **Negative Test Cases** - Negative test cases are designed to test the feature in ways that the developer may not have envisioned. This includes bounds and logical tests. For example, try testing invalid dates, entering alphabetic information in numeric fields, entering numeric information outside of the bounds that it was designed for (e.g. try entering 101% in a percentage field that should only allow up to 100%), and entering field values that are larger than the field size (e.g. try entering 101 characters in a field that is designed for 100 characters). The review should ensure that there are an adequate number of test cases for this.
3. **Performance** - Performance test cases ensure that the code will not become unusable with large amounts of data. For example, import 50,000 items and record the timings. Compare those timings to when you only have 50 items. For most applications, acceptable response time is anything under 5 seconds, while good response time is anything under 2 seconds. This may vary depending on the application and your own performance guidelines. The review should ensure that there are an adequate number of test cases for this.
4. **Security Testing** - If the feature is a secured feature, there should be security test cases to ensure that the correct rights are granted before specific actions can occur. The review should ensure that there are an adequate number of test cases for this.
5. **Regression Testing** - If the feature will become a base part of your product, identify specific test cases from this test suite that would be a good candidate for Regression Testing in future releases. The review should ensure that there are an adequate number of test cases for this.

Defect Linkage

Once testing commences, it is important to identify the defects that were created based on the specific test case that failed. Good defect management solutions will allow you to automatically generate a defect when a test case fails and provide a linkage between the failed test case and the automatically generated defect. Assuming your test cases are linked back to the original requirement specification, you can then link the defects by requirement. This allows you to determine which requirements spawn the most defects.

Traceability Report

Good defect management solutions will provide a report that shows you the traceability mentioned above. If yours does not, you can put this together manually, but it does require a bit of work and must be updated frequently. Here is an example of a traceability report:

http://www.pragmaticsw.com/newsletters/TraceabilityReport.pdf (*when you click this link, zoom to 150% to see it better*).

Helpful Templates

Below are some helpful templates to aid you in developing software solutions on-time and on-budget:

- **Pragmatic Agile Development (PAD) Overview** - http://www.PragmaticSW.com/PADOverviewPresentation.pdf
- **PAD Road Map** - http://www.PragmaticSW.com/Pragmatic/Templates/RoadMap_Template.pdf
- **PAD Best Practices Excerpt -** http://www.PragmaticSW.com/PADBestPracticesExcerpt.pdf
- **Additional PAD Information** - http://www.pragmaticsw.com/PADOverview.pdf
- **Project Management Guidelines -** http://www.PragmaticSW.com/Pragmatic/Templates/ProjectMgtGuidelines.rtf
- **Functional Specifications -** http://www.PragmaticSW.com/Pragmatic/Templates/FunctionalSpec.rtf
- **Architectural Overview -** http://www.PragmaticSW.com/Pragmatic/Templates/ArchitectureOverview.rtf
- **Detailed Design** - http://www.PragmaticSW.com/Pragmatic/Templates/DetailedDesign.rtf
- **Strategic Planning Document** - http://www.PragmaticSW.com/Pragmatic/Templates/StrategicPlanning.rtf
- **Test Design** - http://www.PragmaticSW.com/Pragmatic/Templates/TestDesign.rtf
- **Risk Assessment** - http://www.PragmaticSW.com/Pragmatic/Templates/Risk%20Assessment.rtf
- **Weekly Status** - http://www.PragmaticSW.com/Pragmatic/Templates/WeeklyStatusRpt.rtf
- **User Acceptance Test Release Report** - http://www.PragmaticSW.com/Pragmatic/Templates/UATRelease.rtf
- **Post Mortem Report** - http://www.PragmaticSW.com/Pragmatic/Templates/PostMortem.rtf
- **All Templates** - http://www.PragmaticSW.com/Templates.htm
- **Prior Newsletters** - http://www.PragmaticSW.com/Newsletters.htm
- **Software Planner** - http://www.SoftwarePlanner.com/SoftwarePlannerPro.asp
- **Defect Tracker** - http://www.DefectTracker.com
- **Remoteus (Remote Desktop Sharing)** - http://www.PragmaticSW.com/Remoteus.asp

About the Author

Steve Miller is the President of **Pragmatic Software** (http://www.PragmaticSW.com). With over 21 years of experience, Steve has extensive knowledge in project management, software architecture and test design. Steve publishes a monthly newsletter for companies that design and develop software. You can read other newsletters at http://www.PragmaticSW.com/Newsletters.asp. Steve's email is steve.miller@PragmaticSW.com.

Pragmatic Software Co., Inc.
383 Inverness Parkway
Suite 280
Englewood, CO 80112

Phone: 303.768.7480
Fax: 303.768.7481
Web site:
http://www.PragmaticSW.com
E-mail: info@PragmaticSW.com