MONTHLY NEWSLETTER

PRAGMATIC S·O·F·T·W·A·R·E

January 2008 - Pragmatic Software Newsletters

6 Tips for Quickening Software Releases

Many of us have experienced projects that drag on much longer than expected and cost more than planned. Most times, this is caused either from inadequate planning (requirement collection and design) or from an inordinate number of defects found during the testing cycle. A major ingredient to reducing development life cycle time is to eliminate defects before they happen. By reducing the number of defects that are found during your quality assurance testing cycle, your team can greatly reduce the time it takes to implement your software project. Below are some tips to aid in reducing software defects:

Tip 1 - Deliver Releases in Iterations

When collecting requirements, it is best to identify the priority of each feature for a release. By prioritizing features as High/Medium/Low, your team can include only the high priorities in the first iteration of the software, then can move to the medium and lower priority items in future iterations. An iteration is a set of features that are bundled into a release that could be launched to production, if desired. Agile scrum developers refer to these as sprints. By implementing features in iterations, it allows you to begin using the software earlier, flesh out requirements that did not pan out as well as thought, and to discover any technology glitches that may have been found at a later date.

Tip 2 - Conduct Requirement Walkthroughs

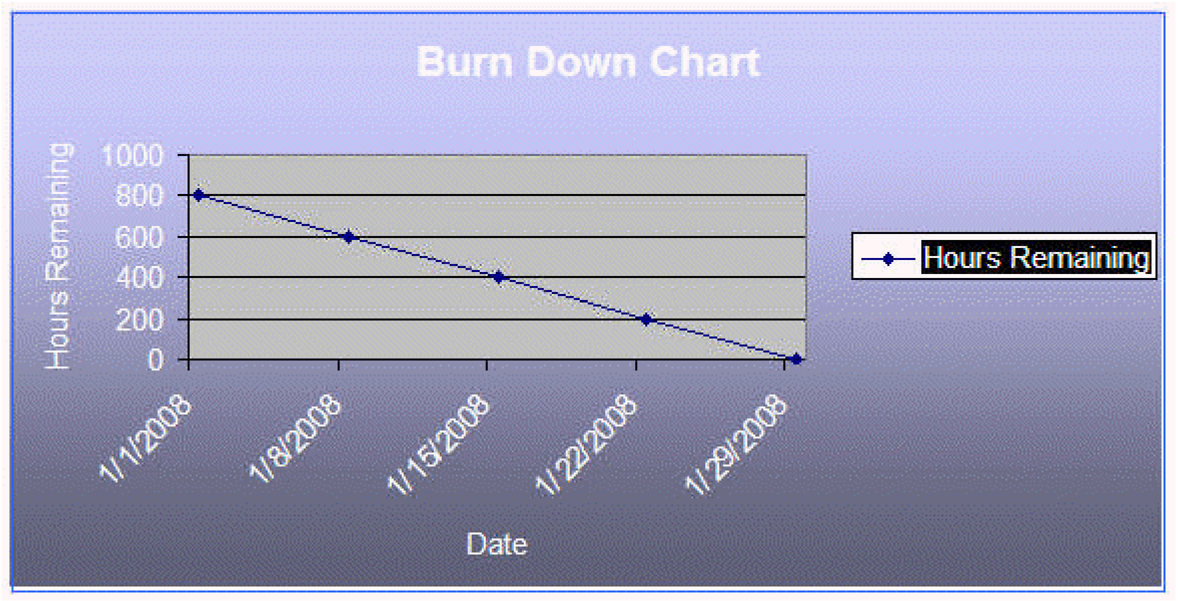
The best time to stop defects is before coding begins. As the project manager or requirements manager begins collecting the requirements for the software, they should hold meetings with two or more developers to ensure that the requirements are not missing information or are not flawed from a technical perspective. These meetings can bring to surface easier ways to accomplish the requirement and can save countless hours in development if done properly. As a rule of thumb, the requirements should be fully reviewed by the developers before the requirements are signed off.

Tip 3 - Distribute Test Cases to Programmers Before Coding Begins

Once a requirement is fully defined and approved, your Quality Assurance team should create a full suite of test cases for the requirement. Once this is done (and before coding begins), publish those test cases to the programmer that is developing the code. Ensure that your project manager adds a task for the programmer to run each test case prior to delivering it to the test team for testing. This approach may add a few hours (or a few days) to the programmer's task list but will pay dividends by reducing the number of defects found and improving the quality of the release. By following this approach, you can expect 70% - 80% less defects and can reduce the quality assurance time by 75% or more.

Tip 4 - Review Burn Down Statistics

Burn down statistics are simply a way of analyzing how much effort is remaining in a release. For example, if you begin a project that has 5 full time resources working 40 hours a week, you can assume that 200 hours of work per week will be accomplished. If you are planning a release that will take 4 weeks, your team of 5 should be able to deliver 800 hours of work. Each day, each team member should record the number of hours worked against tasks, and the number of hours remaining to complete their tasks. By recording the number of hours remaining, you can determine if you will finish on time or not (this also helps validate your estimates). Each day, you can compare how many hours are remaining versus how many hours should be remaining. If you have more hours remaining that hours that should be remaining, you can help your team by adding additional resources, reducing feature set, or by asking for overtime work. As your project begins getting closer toward delivery, your burn down (hours remaining) should decline, below is an example:



Tip 5 - Conduct Code Reviews
Once coding begins, each programmer should be encouraged to conduct weekly code reviews with their peers. The meeting is relatively informal, where the programmer distributes source code listings to a couple of his/her peers. The peers should inspect the code for logic errors, reusability and conformance to requirements. This process should take no more than an hour and if done properly, will prevent many defects that could arise later in testing.

Every few weeks (or before a minor release), the chief architect or technical team leader should do a formal inspection of their team's code. This review is a little more formal, where the leader reviews the source code listings for logic errors, reusability, adherence to requirements, integration with other areas of the system, and documentation. Using a checklist will ensure that all areas of the code are inspected. This process should take no more than a couple of hours for each programmer and should provide specific feedback and ideas for making the code work per the design.

As inspections are held, someone (referred to as a scribe) should attend the meetings and make detailed notes about each item that is found. Once the meeting is over, the scribe will send the notes to each team member, ensuring that all items are addressed. The scribe can be one of the other programmers, an administrative assistant, or anyone on the team. The defects found should be logged using your defect tracking system and should note what phase of the life cycle the defect was found.

Tip 6 - Collect Metrics
Collect statistics that show how many defects (along with severity and priority) are found in the different stages of the life cycle. The statistics will normally show over time that when more defects are resolved earlier in the life cycle, the length of the project decreases and the quality increases.

Helpful Templates

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

Project Management Templates

- **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
- **Architectural Overview** - http://www.PragmaticSW.com/Pragmatic/Templates/ArchitectureOverview.rtf
- **Risk Assessment** - http://www.PragmaticSW.com/Pragmatic/Templates/Risk%20Assessment.rtf
- **Post Mortem Report** - http://www.PragmaticSW.com/Pragmatic/Templates/PostMortem.rtf

Quality Assurance Templates

- **Functional Specifications** - http://www.PragmaticSW.com/Pragmatic/Templates/FunctionalSpec.rtf
- **Detailed Design** - http://www.PragmaticSW.com/Pragmatic/Templates/DetailedDesign.rtf
- **Test Design** - http://www.PragmaticSW.com/Pragmatic/Templates/TestDesign.rtf
- **Weekly Status** - http://www.PragmaticSW.com/Pragmatic/Templates/WeeklyStatusRpt.rtf
- **User Acceptance Test Release Report** - http://www.PragmaticSW.com/Pragmatic/Templates/UATRelease.rtf

Other Helpful Links

- **Prior Newsletters** - http://www.PragmaticSW.com/Newsletters.asp
- **Software Planner** - http://www.SoftwarePlanner.com
- **Defect Tracker** - http://www.DefectTracker.com

About the Author

Steve Miller is the President of Pragmatic Software (http://www.PragmaticSW.com). With over 23 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.