Software Development Lifecycle (SDLC) - Overall Project Measurement

March 2006 - Pragmatic Software Newsletters

Measuring the Software Development Lifecycle

Employing a solid software development lifecycle (SDLC) methodology can drastically increase your ability to deliver software projects on-time and on-budget. Once a solid SDLC methodology is in place, how do you know how efficient it is and how well it is performing? In the coming months, we will look at best practices for measuring the key indicators of the SDLC, and equip you with the tools to improve your SDLC. Below are the topics to be covered in the coming months:

1. **Defect and Test Case Measurement** - Defect and Test Case Measurement is a pre-production activity that allows teams to determine the quality of their software development, and indicates when the software is ready to be released to production. More...

2. **Project Task Measurement** - Project Task measurement allows your team to determine how well individual tasks were estimated, how well they were defined, and whether items are completed on-time and on-budget. More...

3. **Overall Project Measurement** - It is important to measure overall project success by determining if the project was estimated properly, risks were identified and mitigated, requirements were correctly identified and documented, and if the project was delivered on-time and on-budget. From this, we learn to provide better estimates, collect better requirements, and do better risk management. More...

4. **Support Ticket Measurement** - Support Ticket management is a post-production activity that allows teams to determine the quality of the software release, the quality of User Guides and other documentation, and provides insight as to how well the software was architected and implemented.

5. **Measuring Team Goals** - For technical teams to flourish, team goals must be established and measured. Constant evaluation of the goals, and progress towards them, is critical to ensuring that team goals contribute to departmental goals.

6. **Measuring Departmental Goals** - Establishing and measuring departmental goals allow your company to grow, allow your department to identify it's contribution to company growth and fosters and environment where team members thrive.

Overall Project Measurement

Overall Project Measurement allows you to determine if the project was estimated properly, risks were identified and mitigated, requirements were correctly identified and documented, and if the project was delivered on-time and on-budget. From this, we learn to provide better estimates, collect better requirements, and do better risk management. Below are some best practices for measuring overall project success:

1. **Review Estimate vs. Actual for the Project** - Prior to beginning a project, it is important to estimate each task that must be performed for the project. As the project progresses, team members should keep track of how much time it took to perform each task. It is also important to identify an allowable variance. For example, upon completion of the project, if

our actual hours/costs were within 5% of the estimate, we will consider it a success. Once the project is completed, run reports that show whether the project was successful (based on your variance allowance). How do you do this?

First, you must detail each of the requirements and develop a list of tasks for the delivery of each requirement (if you need help developing requirements, click here). With a good set of requirements, the project manager can work with the project team (programmers, testers, etc to develop a list of tasks that must be completed for each requirement, along with the estimated effort of each task. Once this is done, record your list of tasks, assign them to team members, and track their progress. As tasks are completed, record the number of actual hours it took to complete each task, as to allow you to determine the correctness your estimation process. Upon completion of the project (all tasks are 100% complete), run reports that show you how well you did. You can use this information on future projects, to create a buffer for improving the estimation process.

For example, let's assume we started a project for a new release of our Widgets product (**Widgets Release 4.1**). At the beginning of the project, we collected the requirements, identified and and estimated each task, and we decided that a 5% variance on the project was acceptable. As the project progressed, our team members logged time toward each task. Upon completion of the project, we analyzed the project results. See the **Basic and Advanced Approaches** below to see how the project turned out.

Basic Approach - If you do not have a software development lifecycle tool, a low-cost and simple approach to this is to create a spreadsheet that contains a list of your tasks and assignment information. As your team members work on items, each day you should make note of actual hours and costs thus far, and percentage complete. As items are finished, update the spreadsheet with Actual Hours, Actual Costs, and Completion Date. After all tasks are completed, analyze whether the project was successful, see attached spreadsheet to see how we did it.

Example: ProjectOverview.xls

Advanced Approach - A better approach is to utilize an SDLC tool that allows tracking of project tasks, assignment of the tasks to team members, tracking of hours and costs, and allows the team members to update their percentage complete. For ease of update, the software should be web based, so that it can be accessed from any location. To do this, you can use Software Planner, Microsoft Project or some other project management tool. The disadvantage of using windows-based tools (like Microsoft Project) is that they are not web based, so individual team members can not easily update their hours, costs and percentage complete; the project manager is forced to update that information. Software Planner (and other web based tools), empower the people doing the work to update this information, and has email alerts that alert the Project Manager as items are updated. Once the project is completed, run reports that analyze whether the project was successful, see attached reports so you can see how we did it.

Examples:
A) Project Tasks By Project Report - By reviewing this report, we can quickly see that our project went OK. The cost overruns were 5 hours of work, relating to $1,035 in costs. At the beginning of the project, we had agreed that if we were within 5% of our hours can and costs, we would consider the project a success. Based on reviewing this report, we were within 2% variance, which is OK. However, it would be good to drill down a little further and determine what tasks and what resources (people) contributed to the overage. See the next report to determine that.
*B) Project Tasks By Assignee Report - By reviewing this report, we can see that a couple of team members really excelled. Notice that **Jennie Jones**, **Mary Jones**, and **John Tester** actually finished their tasks under budget. However, **John Doe** and **Joe Millionaire** finished their tasks over budget. We may want to analyze their tasks further to determine why they had overages, and if we need to change our estimating techniques when assigning tasks to these individuals.*

2. **Post Mortem** - As projects are completed, you must perform a "post mortem" to identify the thir you did well, and the things you did poorly. Use this information to improve future projects. Ho you do this?

    1. **Plan Your Post Mortem Review** - Upon completion of a project, the Project Manager should conduct a "**Post Mortem**" review. This is where the Project Manager invites all the major players of the team (Analysts, Lead Programmers, Quality Assurance Leaders, Production Support Leaders, etc) to a meeting to review the successes and failures of the project.

    2. **Require Team Participation** - Ask the attendees to bring a list of 2 items that were done well during the project and 2 things that could be improved upon.

    3. **Hold the Post Mortem Review Meeting** - Go around the table and have each person to discuss the 4 items they brought to the meeting. Keep track of how many duplicate items you get from each team member. At the end of the round table discussion of items, you should have a count of the most popular items that were done well and the most agreed upon items that need improvement. Discuss the top 10 success items and the top 10 items that need improvement.

    4. **List Items Done Well and Things Needing Improvement** - Upon listing of the 10 success and improvement items, discuss specific things that can be done to avoid the items that need improvement upon the next release. If some items need more investigation, assign specific individuals to finding solutions.

    5. **Create a Post Mortem Report** - The best way to keep this information organized is to create a "Post Mortem" report, where you document your findings. Send the Post Morten report to all team members. Before team members embark on their next project, make sure they review the Post Mortem report from the prior project to gain insight from the prior project. We have created a template that you can use for the document, downloac it by clicking here.

Helpful Templates

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

- **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
- **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 20 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.htm. Steve's email is steve.miller@PragmaticSW.com.

Pragmatic Software Co., Inc.	Phone: 303.768.7480
383 Inverness Parkway	Fax: 303.768.7481
Suite 280	Web site: http://www.PragmaticSW.com
Englewood, CO 80112	E-mail: info@PragmaticSW.com