Best Practices for Software Projects - Change Control

June 2004 - Pragmatic Software Newsletters

Best Practices for Software Projects - Change Control

When developing and enhancing software, a well laid plan has a documented set of features, detailed designs, and estimates that allow the project manager to quickly determine if the project is on track to completion. As development proceeds, it is not uncommon for business or market conditions to change, thereby changing the needs of a software project that is currently in development.

Change control is the process of managing changes as to ensure that decisions are not made hastily and that the decision to add an additional feature is in the best interest of the project. If change control is missing from a project, new features will be introduced at random, jeopardizing the delivery date and quality of the software being developed.

To manage change, it is best to have a **Change Control Board**. The Change Control Board normally consists of the project manager, client manager, development (programming) manager, quality assurance manager, user documentation manager, and support manager. As new changes are requested, the Change Control Board analyzes the risks and rewards of the proposed change. They analyze how the change affects the overall schedule, costs, and feature set. Below are some guidelines for the Change Control Board:

1. **OK to say "No"** - If the project team has done a good job of collecting requirements and managing the software life cycle, many features have already been reviewed and prioritized before coding begins. If the new requirement is not worth the time of analyzing it, then it is not worth the time to implement it, therefore, it should be rejected immediately. Normally, a good Change Control Board will say "No" more times that it says "Yes", as to ensure that only critical changes are implemented.
2. **Changes should be Bundled** - A large number of small changes, when done independently, can greatly affect the project timeline because each one affects many areas of the system (analysis, coding, testing, user documentation, support, etc). To gain economies of scale and effective use of personnel, it is wise to document a set of changes at one time and include them into a bundled release. By doing this, you eliminate a lot of the convergence needed with implementing each feature separately.
3. **Eliminate Bureaucracy** - Some Change Control Boards are tainted by individuals that just like to say "No" for the sake being in charge. This can bring ill-will to a project, when it seems as if the Change Control Board is not making decisions that are in the best interest of the project. To eliminate this problem, educate all members of the Change Control Board before the project begins. Ensure they understand that there will be frivolous changes that will be submitted as well as legitimate changes necessary for the product to be marketable and to meet the needs of the business. So as each new change is suggested, have a risk/rewards discussion that documents the business reasons for the change and the risk associated with the change. Documenting and publishing your findings for each change request provides understanding and buy-in from members that requested the change.
4. **Document Changes** - Document each change by first assigning the change request a unique tracking number. Then have your team do an analysis of the risk/rewards for the change and document that. It is wise to use an on-line change management solution that allows you to track the changes and manage the workflow of it (reviewing, approving and rejecting changes). There are many solutions for aiding you in this process. For example, Software Planner (http://www.softwareplanner.com) has a functional specifications feature that tracks change requests and allow you to manager the associated workflow.

As you have seen, changes can be easily managed with a disciplined approach to project management. Below are some helpful templates to aid you in managing the entire software life cycle:

- **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

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.
1745 Shea Center Drive
Suite 400
Highlands Ranch, CO 80129

Phone: 720.344.4846
Fax: 720.344.4847
Web site: http://www.PragmaticSW.com
E-mail: info@PragmaticSW.com