Managing Workflow for Project Management

For teams managing software development, it is crucial to manage the workflow around the project management process so that everyone understands how a task moves from definition to completion Below are some tips for defining the workflow for software project management.

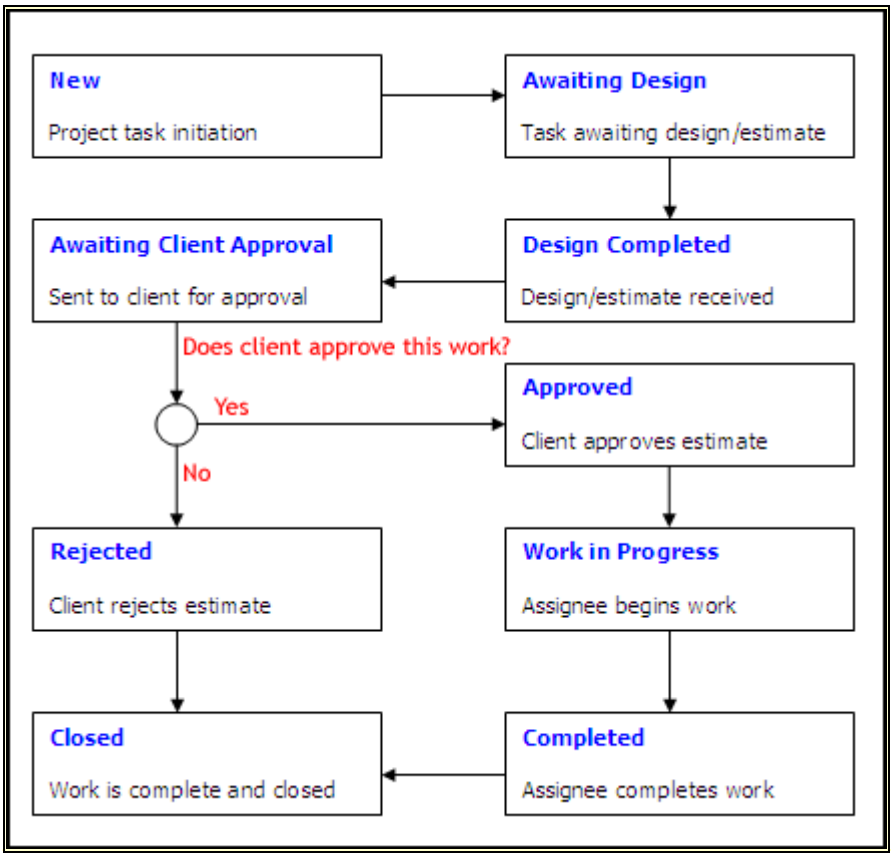
1. **Define the Workflow Statuses** - When tracking project tasks, it is important to define the workflow. Workflow is normally tracked via the "status". Let's create a simple workflow for a software development team, where the project manager creates the task and tracks the status through completion of the task. Below are some possible sets of statuses (workflow) for this process.

Workflow Statuses:

New
Awaiting Design
Design Completed
Awaiting Client Approval
Approved
Rejected
Work in Progress
Completed
Closed

(continued on next page…)

2. **Flowchart the Workflow** - Flowcharting the workflow allows team members to understand the process in full. We created the flowchart below using Microsoft Word.



3. **Advanced Workflow** - In our example above, we used simple workflow. However, if your team uses software to manage project tasks, you should be able to implement more robust workflow. For example, the software should allow you to define "state transitions". This identifies how a status can transition from one status to another. In our example, above, you may want to setup these transitions:

» **New** - Can only transition to Awaiting Design
» **Awaiting Design** - Can only transition to Design Completed
» **Design Completed** - Can only transition to Awaiting Client Approval
» **Awaiting Client Approval** - Can only transition to Approved or Rejected
» **Rejected** - Can only transition to Closed
» **Approved** - Can only transition to Work in Progress
» **Work in Progress** - Can only transition to Completed
» **Completed** - Can only transition to Closed
» **Closed** - No transitions allowed

Likewise, the software should also allow you to define what fields (or items) you wish to make required upon different states. In the example above, if the

task is changed to **Rejected**, we may want to require that the project manager enter the **reason for rejection.** Robust project tracking software will allow you to define the field attributes for each state transition. **Software Planner** (http://www.SoftwarePlanner.com) does this nicely, you can see how this is handled from **Software Planner** by viewing this movie:

http://www.pragmaticsw.com/GuidedTours/Default.asp?FileName=Workflow

4. **Recording Project Results** - 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:

   http://www.pragmaticsw.com/Newsletters/newsletter_2002_06.htm

   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.

5. **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: http://www.pragmaticsw.com/newsletters/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 (http://www.softwareplanner.com), Microsoft Project (http://www.microsoftproject.com)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.

6. **Examples:**
   A) Project Tasks By Project Report (http://www.pragmaticsw.com/newsletters/SP_ProjectTasksByProject.pdf) - 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 (http://www.pragmaticsw.com/newsletters/SP_ProjectTasksByAssignee.pdf) - 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.*

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.
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