Best Practices for Software Projects - Risk Management

July 2004 - Pragmatic Software Newsletters

To deliver software on-time and on-budget, successful project managers understand that software development is complex, and that unexpected things will happen during the project life cycle. There are 2 types of risks that may affect your project during it's duration:

- **Risks you know about** - There are many risks that you know about, that you can mitigate. For example, let's assume that you have assembled a team to work on the project and one of the stellar team members has already scheduled a 3 week vacation just before testing is scheduled, which you agreed to allow. The successful project manager will identify this risk and provide some contingency plans to control the risk.

- **Risks you don't know about** - There are also risks that you don't know about, so a general risk assessment must be done to build time into your schedule for these types of risks. For example, your development server may crash 2 weeks into development and it may take you 3 days to get it up and running again.

The key to managing risks is to build contingency plans for risk and to build enough time into your project schedule to mitigate risks that you do not know about. Below are a list of the **5 most common scheduling risks** in a software development project:

1. **Scope and feature creep** - Here is an example: Let's say the client agrees to a requirement for a Logon page. The requirement specifies that the client will enter their userid/password, it will be validated and will allow entry upon successful validation. Simple enough. Then in a meeting just before coding is commencing, the client says to your project manager "I was working with another system last week and they send the client a report each day that shows how many people log in each day. Since you have that information already anyway, I'm sure it will only take a couple of minutes to automate a report for me that does this." Although this sounds simple to the client, it requires many different things to happen. First, the project manager has to amend the requirement document. Then the programmer has understand the new requirement. The testing team must build test scenarios for this. The documentation team must now include this report in the documentation. The user acceptance team must plan to test this. So as you can see, a simple request can add days of additional project time, increasing risk.

2. **Gold Plating** - Similar to scope and feature creep, programmers can also incur risk by making the feature more robust than is necessary. For example, the specification for the Logon page contained a screen shot that showed very few graphics, it was just a simple logon process. However, the programmer decides that it would be really cool to add a FLASH based movie on the page that fades in the names of all the programmers and a documentary on security. This new movie (while cool in the programmer's eyes), takes 4 hours of additional work, put their follow-on tasks are n jeopardy because they are now behind schedule.

3. **Substandard Quality** - The opposite of Gold Plating is substandard quality. In the gold plating example, the programmer got behind schedule and desperately needed to catch up. To catch up, the programmer decided to quickly code the next feature and not spend the time testing the feature as they should have. Once the feature went to the testing team, a lot of bugs were found, causing the testing / fix cycle to extend far beyond what was originally expected.

4. **Unrealistic Project Schedules** - Many new team members fall into this trap. Project members (project managers, developers, testers, etc), all get pressure from customers and management to complete things in a certain time frame, within a certain budget. When the timeframes are unrealistic based on the feature set dictated, some unseasoned team members will bow to the

pressure and create their estimates based on what they think their managers want to hear, knowing that the estimates are not feasible. They would rather delay the pain until later, when the schedule spirals out of control.

5. **Poor Designs** - Many developers and architects rush the design stage in favor of getting the design behind them so the "real" work can begin. A solid design can save hundreds of programming hours. A design that is reusable, allows changes to made quickly and lessens testing. So the design stage should not be rushed.

Calculating Risk

The key to successful risk management is to identify all risks you know about and build time in for risks you do not know about. The mechanics of doing this is to simply list the risks, and figure out the loss (in hours) you expect would happen if the risk occurs, and then list the probability of the risk happening. Then you can factor that risk into your project schedule by multiplying the loss by the probability.

For example, let's assume that you have decided to use a new outsourcing company to do some of the coding for your project. Since you have not worked with them before, you surmise that there is a 50% probability that you could incur 40 hours of additional development because of the new resource. You would build 20 additional hours (40 hours * 50% probability) into your project plan to mitigate the risk.

Here is a template for calculating risks: Risk Management Template
Here is an example of using Software Planner to manage risks: Manage Risks with Software Planner

Using Online Tools

You can minimize project risk by using tools to help you track the entire software life cycle. There are many solutions for aiding you in this process. For example, Software Planner (http://www.SoftwarePlanner.com) has features that allow you to track all phases of the life cycle from start to finish.

Software Planner also has a unique feature called the **List Manager**, where you can define a list of information to track. In this case, you can create a Risk Management list and track your risks online. To see how that works, see this movie.

Conclusion - Helpful Templates

As you can see, risk management is key to ensure that your projects are delivered on-time and on-budget. 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

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