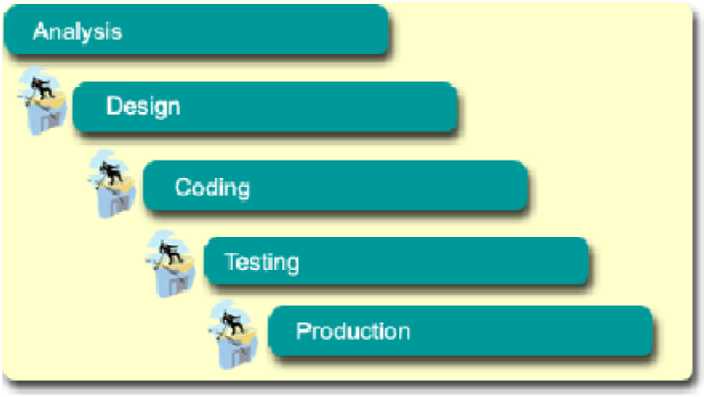
Best Practices for Software Projects - Iterative Development

August 2004 - Pragmatic Software Newsletters

For years, the primary methodology for delivering software projects was the "Waterfall Method". With the Waterfall method, all software requirements are gathered up front, designs are done for each requirement, and each feature is coded and tested before migrating to production. For projects that exceed one year of development and implementation, there are risks of the project being cancelled (according to the Standish Group, about 31%are cancelled before completion).

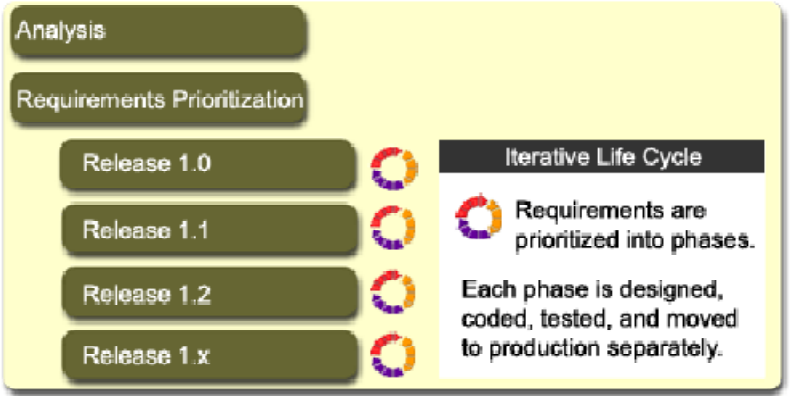


Weaknesses of the Waterfall Method:

- Since all requirements are gathered (and designs done) upfront, the software life cycle is normally a year or longer.
- Business rules change along with market conditions. Due to the long software life cycle, the software may no longer meet the needs of the company after being implemented.
- Testing is performed after all features are analyzed, designed and coded. If a major design flaw is found during the testing phase, it may need to be addressed in many functional areas, causing the time to fix the defect much longer due to the large number of requirements.

Iterative Development Model

An alternative approach is the Iterative Development Life Cycle (sometimes referred to as the Spiral Life Cycle).



With the Iterative Life Cycle, analysis is done just the same as with the Waterfall method. However, once

analysis is done, each requirement is prioritized as follows:

- **High** - These are mission critical requirements that absolutely have to be done in the first release.
- **Medium** - These are requirements that are important but can be worked around until implemented.
- **Low** - These are requirements that are nice-to-have but not critical to the operation of the software.

Once priorities have been established, the releases are planned. The first release (Release 1.0) will contain just the High priority items and should take about 1 to 3 months to deliver.

Below are the advantages of the Iterative Life Cycle:

- The Design phase goes much faster, as designs are only done on the items in the current release (Release 1.0 for example).
- Coding and Testing go much faster because there are less items to code and test. If major design flaws are found, re-work is much faster since the functional areas have been greatly reduced.
- The client gets into production in less than 3 months, allowing them to begin earning revenue or reducing expenses quicker with their product.
- If market conditions change for the client, changes can be incorporated in the next iterative release, allowing the software to be much more nimble.
- As the software is implemented, the client can make recommendations for the next iteration due to experiences learned in the past iteration.

Our experience has found that you should space iterations at least 2 – 3 months a part. If iterations are closer than that, you spend too much time on convergence and the project timeframe expands. During the coding phase, code reviews must be done weekly to ensure that the developers are delivering to specification and all source code is put under source control. Also, full installation routines are to be used for each iterative release as it would be done in production.

Using Online Tools

You can reduce time to production and minimize project risk by using tools to help you track the Iterative 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 and allows staging of deliverables into specific iterative releases.

Conclusion - Helpful Templates

As you can see, the Iterative Life Cycle can greatly improve your changes of delivering 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