

Pragmatic Agile Development - The Lifecycle Phases

April 2007 - Pragmatic Software Newsletters

Last month, our newsletter explored the differences between Agile and the Waterfall methodology for software development. It explained that Agile is a very nimble software development process that allows new requirements to enter the software lifecycle as iterations proceed while the Waterfall methodology is a more traditional approach where all requirements are gathered and designed up front and strict controls govern the life cycle and allow better control of milestones. This month, we discuss a blended approach between Agile and Waterfall, called Pragmatic Agile Development (PAD).

Last month's newsletter: [Exploring Agile Development](#)

Pragmatic Agile Development (PAD) - A Blended Approach

Pragmatic Agile Development (PAD) is designed to encompass all of the measurement and oversight of the Waterfall methodology (similar to approaches used by PMI / CMMI) with the nimbleness of Agile development, providing a methodology that yields the best of both worlds.

With PAD, software is delivered in release iterations, similar to Agile, with a typical release to production in about a 3 month period. Unlike Agile, PAD does not compile complete builds weekly; instead, it favors a complete coding cycle before entering quality assurance, as this reduces convergence and unnecessary testing while the product is still being developed.

Like Agile, PAD is very nimble, allowing new requirements to enter the release anytime before quality assurance begins. Features are prioritized so that high priority items are worked on first, and if new high priority customer requests are submitted, they can be accommodated in the release, if they do not pose risk to meeting the on-time delivery of the product.

PAD Phases

PAD is similar to most development methodologies, containing the same basic lifecycle phases. The difference is that each phase is defined for a specific period of time. Your team can decide on the durations of each phase up front. Below is an example of phase durations we have used successfully in the past:

- **Analysis Phase** - 10 working days
- **Coding and Test Preparation** - 30 working days
- **Code Merge** - 3 working days
- **Quality Assurance** - 20 working days
- **Documentation / Production Preparation** - 5 working days
- **Production Release / Fixes** - 3 working days
- **Post Mortem** - 2 working days

Notice that specific time is built into the software lifecycle for **Code Merge** and **Documentation / Production Preparation**. Many teams do not account for these in their project plans and their projects are shipped late due to this. Also, notice that time is built in for **Production Fixes**. No matter how much quality assurance your team performs, issues will be found when you move to production and your team should be at-the-ready to quickly identify and fix those issues. Last, notice that specific time is allocated for **Post Mortem**. A well documented post mortem provides critical feedback towards things done right and things that can be improved upon in the next release and can be used by the next release team to minimize mistakes.

Learning More About PAD

Our upcoming newsletters will drill into the details of utilizing PAD to consistently manage your software lifecycle. Next month, we will discuss the PAD Road Map.

The PAD Road Map defines the schedule (which is a locked duration), the requirements (in priority order), customer facing defects that will be fixed with the software, and the documented features that will be released with the iteration. The PAD Road Map also contains specific measurements to

ensure that each phase of the lifecycle is completed on-time and on-budget.

If you wish to learn more about PAD, here are some helpful links:

- [PAD Overview Presentation](#)
- [PAD Best Practices Excerpt](#)
- [Additional PAD Information](#)

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/SoftwarePlannerPro.asp>
- **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 21 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.asp>. 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