Software Development Best Practices

by Darian Rashid

Agile and Design for Lean Six Sigma

Introduction

Since the first programmable computers, software developers and managers have struggled with fundamental problems in executing a project from inception to deployment. After nearly 65 years, most of the same breakdowns still remain:

- Requirements are often:
    - Poorly Defined
    - Ambiguous
    - Incomplete
    - Incorrectly Interpreted by the Architect/Developer
    - Changed when Development is Well Underway

- Schedules are too tight:
    - Estimates are Rarely Accurate
    - Committed to 6-12 Months Before the Release
    - Deadlines are Often Missed
    - Becomes “Normal” for Developers to Work Nights and Weekends

- Never enough adequate testing:
    - Developers Prefer Not to or Don’t know How to Adequately Unit Test
    - Too Many Bugs Slip into Delivered Content
    - Software is Often Released Because Time is up, Not Because It Is at the Right
    - Level of Quality/Maturity
    - Usually too few resources in any department (development, testers, etc.).

Waterfall and Six Sigma

At the highest level, the most commonly used software development lifecycle methodology, called the Waterfall Model, can be summarized as the following sequential series of steps:

1. Requirements and Specification
2. Design
3. Construction (Coding)
4. Integration
5. Testing
6. Deployment
7. Maintenance

From this high-level view, the Waterfall Model looks like an ideal process where the output of one step is the input into the next, finally resulting in the deployed software. Because of this process view, it would seem that Lean Six Sigma, more specifically DMAIC, could help solve the issues outlined above and help improve the Waterfall process to deliver higher quality software on time.

©2011 Air Academy Associates. All rights reserved.

Software Development Best Practices

by Darian Rashid

Shortcomings of the Waterfall Model

Many have argued that DMAIC by itself can and has been used to solve deep problems within software development, including:

- Analyzing the requirements elicitation and VOC methods to fix holes in obtaining requirements for less ambiguity.
- Analyzing data to discover and fix root causes for changing requirements.
- Using historical data to make better forecasts to increase accuracy on release plans 6 - 12 months out.
- Understanding causes of phase containment breaches to have less bugs (defects) leaving each step of Waterfall, especially in deployed software.

On the surface, it seems that these are perfect problems for Black Belts to take on and solve and indeed this has been the thinking for the last 15 years. However, with all the effort poured into software development improvement, why hasn't DMAIC yet delivered the breakthrough results in the software development organization that it has in every other industry and organization? The answer is two-fold.

First, the fundamental goal of DMAIC is to standardize the output of a process, to reduce variation and make the final result as consistent and as repeatable as possible. However, software development is not a repeatable process and can never have the same output. Software development creates a unique product (the output) every single time. As an analogy, DMAIC aims to create perfectly consistent cakes, but software development aims to create a new recipe each time.

Second, the problem isn't as much DMAIC applied to Waterfall, but the Waterfall method itself. DMAIC cannot solve issues that are beyond the capabilities of the process itself and thus cannot make Waterfall perform past its own barriers. For example,

- Waterfall works by freezing the requirements up-front and leaves little to no room for evolution of requirements due to the changing market or the customers' changing business problems. Any hope of getting legitimately new or changed requirements has to go through a cumbersome and time-consuming change-control process, usually designed to make customers or business owners think twice before invoking it.

- Waterfall relies on up-front scheduling where schedules are fixed before assumptions on architecturally risky issues are verified. Any later falsification of key assumptions just means more "scrambling" to achieve the goal as the deadlines are usually set in stone by that point

- Waterfall relies on "batch processing," requiring the output of each step to be completed before the next step may begin. If any step is delayed, all of the latter steps will be impacted. Either the deadline is slipped or the remaining work has to be done in the shortened time frame leading to severely reduced quality.

©2011 Air Academy Associates. All rights reserved.

- Waterfall has a separate construction (development) and testing phase, each of which focus on the entire scope of the deliverable. Developing that much complexity at the same time leads to large amounts of bugs due to several issues, the least of which is the erroneous idea that many developers share that this phase is to write code and check it in and bug-fixing will be done in the next phase (testing). This idea leads to severely harmful practices of checking-in code that has severe defects and in some cases, barely compiles just so that task is off the developer's plate.

The Waterfall Model's origin is often cited to be from an article published in 1970 by W. W. Royce where he discusses an "iterative" approach to software development using a series of linear processes described as the seven steps given above and in fact doesn't even use the word Waterfall. Unfortunately, many people only read the first half of the article and decided to use it as they understood it. Ironically, the whole point of Royce's article more than 37 years ago is that Waterfall is not the best way.

Lean Software Development

Lean is a process improvement and management philosophy which focuses on maximizing the value to the customer. One of the ways it does this is to remove the waste in all of its forms from the process and organization. Lean is based on the same queuing theory principles that CPU and other microchip architects use to optimize parallel pipelining to maximize speed but structured in a way that any process improvement practitioner can understand and apply.

Lean identifies many forms of waste that directly apply to the Waterfall method, including:

1. Handoffs

There is at least one handoff between any two phases of Waterfall but the bulk of the handoffs and waste occur when going upstream and reworking requirements, architecture and code from later phases.

2. Work in Progress (Inventory)

In software, any work that has been started but not deployed to the customer has the potential of being cancelled or unneeded, resulting in lost work as well as lost opportunities since the same resources could have been working on revenue-generating features. To minimize inventory, it is essential to get features out to the customer as soon as possible in small batches. This concept of reducing our batch size (as contrasted with batching the number of features to a full release) and having multiple small deployments has incredible positive effects:

- The customers see something sooner than 6-12 months and may even choose to use it after every sub-release. Customers may well see software released as often as every one week to one month.

©2011 Air Academy Associates. All rights reserved.

- The customers may choose to revise what they see, even if the feature delivered was 100% to specification. It is often incredibly difficult for customers to envision the right solution when eliciting requirements, but it is very easy for them to see what it is not when looking at working software. Because we are releasing small chunks but releasing often, the customer is free to change or add new requirements between each sub-release, putting them in the driver's seat. This has the added benefit of pushing the tradeoffs between the right features and the impact on deadline up to them where it belongs instead of the development organization, which is usually forced to make it.

- Because of the reduced number of features per sub-release, developers face tasks where complexity is greatly reduced as compared to the Waterfall development phase and can focus their efforts on writing code where quality is built-in the first time.

3. Rework of defects

Anytime a defect is found and has to be reworked, it is considered a defect. Lean considers the time, cost and effort spent in reworking defects a form of waste. In software development, anytime we have a bug in the testing phase that should have been caught in the development phase, there is a lot of wasted time and energy spent on tracking it down and fixing it. Using test-driven development in conjunction with smaller deliverables, quality is built in and we have significantly less bugs created, which saves all the effort of finding and fixing them.

4. Traceability:

Because of the reduced number of deliverables for each of the smaller set of deliverables, user-acceptance test cases may be defined for each requirement. This practice eliminates any need for code-to-requirements traceability as long as the test cases are traceable to the requirement. Mature organizations that have practiced lean methods for years eventually do away with requirements in the traditional sense and use the test cases as their only requirements.

The Agile software development framework is the evolution of lean applied to the software development lifecycle. It not only does not shy away from changing and evolving software, but in-fact embraces the changing nature of requirements, architecture and code. It attempts to solve the most common problems outlined above by small releases, called iterations, which typically last anywhere between one week and one month. Smaller release cycles give us less complexity in the code, leading to significantly less bugs as well as a structure that is conducive to accepting changing requirements, if need be. Note that in many companies, a release doesn't necessarily mean deployment to the customer, but is simply a part of the final release that is defined, coded, tested and ready to be deployed when the full release plan is fulfilled.

The Agile approach embodies the best practices in software development that have worked for over a decade in large and small companies alike. It does not give incremental benefits within traditional methods but results in breakthrough benefits in requirements elicitation and management, code quality and release management.

©2011 Air Academy Associates. All rights reserved.

Software Development Best Practices

by Darian Rashid

Design for Lean Six Sigma (DfLSS) for Software

The ultimate evolution of best practices in software development is taking the Lean philosophies and integrating Six Sigma tools and practices where they belong. Lean, through Agile, gives us a new way to think about software. Simultaneously, Six Sigma gives us tools for:

- Tracking and analyzing metrics to ensure accuracy in schedules through use of forecasting techniques in combination with confidence intervals.

- Discovering and eliminating (or mitigating) risks and failure modes in the requirements and/or architecture using a failure modes and effects analysis (FMEA).

- Use of design of experiments (DOE) techniques to discover new and efficient ways of testing (i.e., Combinatorial Testing (CT), High Throughput Testing or Pairwise Testing).

Conclusion

By understanding Agile software development along with basic Six Sigma tools, developers and managers will have a full arsenal of best practices, methodologies and tools available to put quality first while meeting scheduled deadlines.
Air Academy's DfLSS for Software is the only methodology in the industry that combines the best of Lean and Six Sigma. In addition, Air Academy is the only organization who has experts in the field of Agile development and deployment who are also practicing Six Sigma Master Black Belts. They bring the perfect blend of both methodologies to give the developers and managers just the right level of each tool to make every hour of training and coaching value-added.

What is Scrum?

Scrum is an agile framework for managing complex and scalable projects Unlike other types of Agile practices (i.e., XP, etc.) which are more principle-driven, Scrum dictates specific roles, responsibilities, procedures and practices. It is perfect for large companies who have multiple, distributed projects. According to the Agile Alliance(1):

"In a recent survey of Scrum Alliance members, respondents were asked about their companies' satisfaction with the Scrum process. Seventy-five percent of those who responded report that Scrum is meeting or exceeding their needs. According to survey results, training contributes to that success."
"Scrum is no longer isolated to one team within a company, either. More than 75 percent of respondents report that Scrum had expanded to the group, division, and company-wide levels. Within organizations that employ Certified ScrumMasters (CSMs), 53 percent report that 50-100 percent of their development work is accomplished using the Scrum process. Seventy-eight percent of those surveyed report that utilization of Scrum is expanding. Scrum is also increasingly used for maintenance work, with 47 percent reporting that they currently use Scrum for maintenance, and 27 percent reporting a plan to use Scrum for maintenance work in the near future."

©2011 Air Academy Associates. All rights reserved.

“Scrum usage isn’t the only thing expanding. The number of Certified ScrumMasters increases every year, and each year more CSMs are certified than the year before. Eighty-eight CSMs were trained in 2003. In the first six months of 2007, the number of new CSMs reached a staggering 4,681. Respondents report a 92.5 percent satisfaction rate with the Scrum certification training they received. More than 80 percent (82.5%) of Scrum Alliance members would definitely recommend certification to others. Only 4 percent of those surveyed indicate they could not recommend Scrum certification at all. In addition, nearly 40 percent of Scrum Alliance members plan to seek further certification.”

(1) sourced from www.agilealliance.org

©2011 Air Academy Associates. All rights reserved.