Skills Paper

CSC 424 – Software Engineering

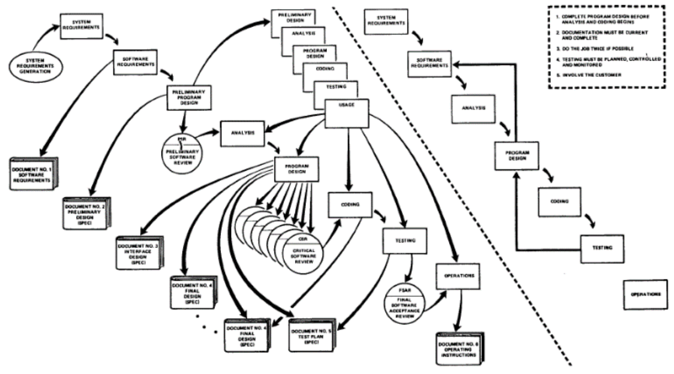
James Setliff

April 17, 2021

As future software developers, there are many aspects of the business side of development that are lacking in a traditional Computer Science education. Programmers come from all backgrounds and disciplines and as such will be heading into various career fields after graduation. One easy way to categorize experience and knowledge bases is using skill sets. When it comes to skill sets, most people generally speak the same language, and it makes identifying and quantifying a candidate's experience easier. At the same time, skill sets make identifying jobs and positions of interest easier. One important skill set we can use to determine the type of working environment we will head into as prospective developers is the type of software development methodology a company uses. As someone that has spent the last two years at NASA working with development teams, I have a unique view of some methodologies that I would like to expound upon.

It is important to note, first, that just like software development itself, the methodologies employed are also constantly evolving. Though most of our formal education is done one project at a time, and often one piece of a code at a time, the professional development environment differs substantially. It is not unusual for a single development team to have multiple projects in-flight at the same time, with several developers interweaving to produce large bodies of code and interconnected systems. Second, a business is almost always interested in delivery, and as such invest heavily in finding a methodology that gives them the highest possible productivity at the lowest possible cost. There are dozens of available methodologies to chose from, with each claiming to be more productive than the last, and often changes from one methodology to another will be implemented to try and extract that little extra bit of performance. So, what can a new software developer expect to encounter at their first job?

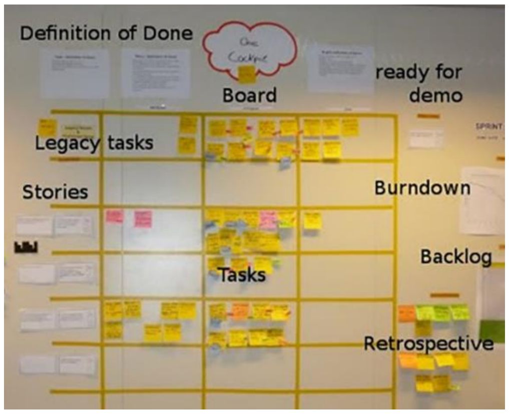
The first documented software development methodology is referred to as the waterfall method. Originally authored by Winston Royce in 1970, the waterfall methodology borrowed heavily from other industries and presented a top-down, plan then execute approach to the software development lifecycle. In his article, "Manage the Development of Large Software Systems" [1], Royce outlined seven steps for the process including, in order, System Requirements, Software Requirements, Analysis, Program Design, Coding, Testing, and Operations. As the only game in town with regards to software development, the U.S. Department of Defense instituted the standard in their Defense System Software Development standard DOD-STD-2167 in June 1985. [2]


SYSTEM REQUIREMENTS GENERATION
SYSTEM REQUIREMENTS
SOFTWARE REQUIREMENTS
PRELIMINARY PROGRAM DESIGN
PRELIMINARY DESIGN
ANALYSIS
PROGRAM DESIGN
CODING
TESTING
USAGE
PDR
PRELIMINARY SOFTWARE REVIEW
ANALYSIS
PROGRAM DESIGN
CODING
TESTING
OPERATIONS
CDR
CRITICAL SOFTWARE REVIEW
FSAR
FINAL SOFTWARE ACCEPTANCE REVIEW
DOCUMENT NO. 1 SOFTWARE REQUIREMENTS
DOCUMENT NO. 2 PRELIMINARY DESIGN (SPEC)
DOCUMENT NO. 3 INTERFACE DESIGN (SPEC)
DOCUMENT NO. 4 FINAL DESIGN (SPEC)
DOCUMENT NO. 4 FINAL DESIGN (SPEC)
DOCUMENT NO. 5 TEST PLAN (SPEC)
DOCUMENT NO. 6 OPERATING INSTRUCTIONS
SYSTEM REQUIREMENTS
SOFTWARE REQUIREMENTS
ANALYSIS
PROGRAM DESIGN
CODING
TESTING
OPERATIONS
1. COMPLETE PROGRAM DESIGN BEFORE ANALYSIS AND CODING BEGINS
2. DOCUMENTATION MUST BE CURRENT AND COMPLETE
3. DO THE JOB TWICE IF POSSIBLE
4. TESTING MUST BE PLANNED, CONTROLLED AND MONITORED
5. INVOLVE THE CUSTOMER

Waterfall relies heavily on project planners and managers doing much of the initial work upfront, and the need for accurate identification of requirements is necessary to the eventual timely success of the project. Waterfall makes sense to most people and is the method that many people would logically choose to complete a project. If you were to build a fence, you would want to lay out the area, take measurements, draw up plans, and purchase materials before you started construction. As such, waterfall is still widely used, and many software shops will see waterfall concepts creep in even if they claim to use another methodology. In my own personal experience, this has been the hardest hurdle to overcome when interacting with new customers to identify business needs. People like to present their grand plan and then move on, expecting it to be completed perfectly when they eventually get the product. There are many issues with this methodology, and this expectation of perfection without clearly defined requirements is chief amongst them. Once presented with the completed product, clients will often want some redevelopment, which delays deployment of the software and increases overall costs.

In response to the perceived failings of waterfall, Agile has seen an explosion in popularity and is now the go-to standard methodology for software development. Though alternative methods of development were prevalent, the term Agile was a result of a meeting in Snowbird, Utah in 2001. “They didn’t agree upon a lot of things, but there were a few things that they were able to agree upon, and that ended up becoming the Manifesto for Agile Software Development. The two main things the Agile Manifesto did was to provide a set of value statements that form the foundation for Agile software development and to coin the term Agile software development itself. In the months afterward, the authors expanded on the ideas of the Agile Manifesto with the 12 Principles Behind the Agile Manifesto.” [3] In the same way Winston Royce had documented his seven-step process, Agile has a set of Agile software development values; individuals and interactions over process and tools, working software over comprehensive documentation, customer collaboration over contract negotiation, and responding to change over following a plan. Additionally, there are twelve Agile development principles from this manifesto that help further detail and outline what successful Agile development looks like.

Though not free from criticism itself, the emergence and growth of more flexible development methodologies shows that there are needs for improvement in certain areas at least of the waterfall cycle. Agile breaks development apart into smaller increments, which helps to reduce the amount of upfront planning and foresight that is necessary to get started on a project. The main goal of each of these smaller cycles is to have a deliverable at the end, something often referred to as the minimum viable product, a working piece of the larger whole, that testers and the customer can get their hands on and provide feedback in a far timelier manner than would be possible under waterfall. Agile also puts a larger focus on developer driven planning and time-management, with some frameworks putting the assignment of work to the development team itself. The sticky note board, popularized by Scrum, is an often thought of representation of this type of team driven task assignment, but is present in other frameworks like the Kanban task board with its swim lanes. However, Agile is not without its own criticisms, and at the top of the list of those complaints are that Agile is simply making up for bad project management.



The Project Management Institute defines Project Management as “the application of knowledge, skills, tools, and techniques to project activities to meet the project requirements,” and goes on to define the five project management processes as Initiating, Planning, Execution, Monitoring and Controlling, and Closing. [4] The similarities to this project management discipline and our software development methodologies are immediately apparent. Project management is, simply put, achieving goals within given constraints. In general, a project manager must be mindful of all the pitfalls associated with a given project and needs to be mindful of the best way to achieve those goals with the team they have. Often cited in project management circles are the 4 P’s: plan, process, people, and power. Ultimately it is the job of project management to define the plan, apply the process, manage the people, and connect with the power, all in service of getting things done.

So why write a paper about Waterfall and Agile and then switch to generic project management at the end? Why not write about how Scrum is better than Kanban or how Lean offers so much more than LeSS? Probably because I think DevOps is the best right? After all, I did name my Senior software project after it! Unfortunately, none of those are true. I switched to the generic topic of Project Management because I believe that is, in general, where the problem lies and software development methodologies are a good way to handle process, but the dozens of frameworks are, in my eye’s, just cash-ins on a fad.

I have worked as a Scrum Master since the beginning of the year at the NASA Shared Services Center, and as I mentioned with how waterfall can creep into your process, the truth about software development is that one size does not fit all. If you have the benefit of amazing businesspeople and talented planners, waterfall can and will be a successful methodology. If your development team directly supports IT infrastructure, then DevOps will be an excellent choice of framework. The simple answer, to me, is that every software shop is deferent and will be successful in deferent ways, often changing over time. Though I am a “Scrum Master” in title, we use concepts from several frameworks including everything I have mentioned in this paper and others. I am always mindful of ability of the customer to define requirements, the maturity of the developers on a team to make determinations on how and when we work stories, the bandwidth of the team before we commit to something, how familiar the test team is with a certain piece of our platform, and on what the end-user expects to see when the release is over. Each of the development teams I work with has different strengths and weaknesses and those change over time, and there is no single framework that provides the answers to all these questions every time.

Ultimately, I believe that the success of a development process is down to the people involved in the process. I give an edge to Agile as a process over Waterfall because it allows things to shift and change, if necessary, over time instead of beating your head against the same wall, but I believe the frameworks are a commercialization of common sense. Like staying healthy, I do not need Richard Simmons to come Jazzercise with me every time I want to work out, but Jazzercising did get thousands of people off the couch, so I would not outright condemn it. In the end, a talented and invested project manager that is proactive about adjusting to meet a team’s unique strengths and overcome their weaknesses is worth a hundred fancy frameworks.

[1] W. W. Royce, "Managing the Development of Large Software Systems," August 1970. [Online]. Available: http://www-scf.usc.edu/~csci201/lectures/Lecture11/royce1970.pdf.

[2] Department of Defense, "Military Standard - Defense System Software Development," 4 June 1985. [Online]. Available: https://www.product-lifecycle-management.com/download/DOD-STD-2167A.pdf.

[3] Editor, "Agile 101," Agile Alliance, [Online]. Available: https://www.agilealliance.org/agile101/. [Accessed 17 April 2021].

[4] Editor, "What is Project Management?," Project Management Institute, Inc., [Online]. Available: https://www.pmi.org/about/learn-about-pmi/what-is-project-management. [Accessed 17 April 2021].