Software Engineering Reflection

Colin Knebl

Maryville University

Software Engineering Reflections

Historically, software engineering has not been able to effectively respond to new requirements very quickly. Software was developed with a very iterative approach, with little to no feedback from stakeholders throughout the development process. This type of approach would often lead to software that did not fully meet the customer’s needs, be over budget, and be delivered late. To help solve this problem a new approach has been introduced called agile. This paper will outline the history of software engineering, the new agile movement, and my personal experiences with the difficulties of software engineering and how those difficulties were mitigated.

In the early days of software engineering, programming and computing were separate tasks (Wirth, 2008). Programmers would write a program, hand that program over to a dispatcher, who would queue their program for processing; the programmer would have to wait hours if not days for their programs to be run and the results to be generated (Wirth, 2008). In today’s world of computing, where software engineers can write and run programs within minutes all from a single machine, this process seems unimaginable.

In 1957, the “first widely known language” was developed by IBM – this language was called Fortran (Wirth, 2008). Fortran gave engineers the ability to solve more complex problems than they were previously capable of solving. However, as the level of complexity increased so did the need for better languages and tools. Many languages have been developed over the years to combat the ever-increasing complexity our programs are attempting to solve.

With the rise in complexity of the programs themselves, the complexity of the design and development also increased. Software projects became increasingly difficult to manage, and as a result many were over budget and not delivered on time. This became known as the “software crisis”.

The software crisis gave way to new tools, ideas, and languages that sought to mitigate the problems that software engineers and the industry were experiencing. Structured programming was one ideas born of the software crisis which is defined as “a subset of procedural programming that enforces a logical structure on the program being written to make it more efficient and easier to understand and modify” (Rouse, 2005). Languages such as Ada, which was developed by the Department of Defense, was created that enforced the ideas put forth from structured programming.

Another methodology that was offered was object-oriented programming. Margaret Rouse (2019) defines object-oriented programming as “a programming language model in which programs are organized around data, or objects, rather than functions and logic. An object can be defined as a data field that has unique attributes and behavior.” Although many of the ideas offered during the software crisis were short lived, object-oriented programming has not only remained to this day, but it is one of the most popular development methodologies.

Now that we have looked at some of the ideas, tools, and languages that came from the software crisis, let’s turn our attention to the idea of agile. “Agile is the ability to create and respond to change. It is a way of dealing with, and ultimately succeeding in, an uncertain and turbulent environment.” (Agile 101, n.d.). In the world of software engineering, there are several different frameworks that can be used to implement the values and principles outlined by the Agile Manifesto; scrum and extreme programming are two of these frameworks.

Agile is not a new concept, but it is relatively new to the field of software engineering. Since its adoption, teams that follow the agile methodology report that their software, on average, is delivered in far less time than traditional waterfall methodologies and more completely meets the customer’s requirements – among other benefits reported. Agile has been used in many other industries including manufacturing and construction. My father is an engineer in the vehicle manufacturing world, and his team follows a methodology similar to agile known as Six Sigma.

I have been working professionally as a front-end software developer for a little over a year. The team I have been working on follows a flavor of agile development; we hold daily scrums and we regularly meet with the different product stakeholders to include business analysts, designers, developers on other teams, and product owners and managers. I think my team is fairly capable of changing our course based on feedback if necessary, but it is not always easy. If feedback is given too late, or a new requirement is uncovered after development work has begun, it can make the pivot difficult, if not impossible without throwing out all or a portion of the code that had already been written. Although this does save the team from creating a product or feature that does not meet requirements, it can be demoralizing for the team.

The main takeaway I have learned so far in the difficulties to software engineering is to over communicate. Misunderstood requirements or misinterpreted communication can be detrimental to any project. It is important to seek clarification even if I feel like I understand the requirements.

Software engineering has come a long way since the 1960s, and continues to improve. The languages, tools, and methodologies such as object-oriented programming and agile have pushed the industry to deliver higher quality software and have given us a way forward.

References

Agile 101. (n.d.). Retrieved from <https://www.agilealliance.org/agile101/>

Rouse, M (2019). Object-oriented programming (OOP). Retrieved from <https://searchapparchitecture.techtarget.com/definition/object-oriented-programming-OOP>

Rouse, M. (2005). Structured Programming (Modular Programming). Retrieved from https://searchsoftwarequality.techtarget.com/definition/structured-programming-modular-programming

Wirth, N. (2008). *A Brief History of Software Engineering* (Vol. 08). IEEE Computer Society.