**An Approach to Software Development**

Mark Doan

Department of Computer Science, Winona State University

CS 410: Software Engineering

Professor Chi-Cheng Lin

January 16, 2024

Abstract

The following paper briefly describes a personal approach to software development from the knowledge and perspective gathered and understood by Mark Doan, a computer science student at the University of Winona. The entire process consists of three major steps recognized as specification, implementation, and release. Specification gives structure to ideas, allowing a referenceable plan throughout development. When a proper guide is complete, work can begin in the implementation step. This includes all practical work done to make the software tangible through computer hardware. When a desirable build of software is complete, actions are taken in the release step to share the software to all relevant parties. All three major steps are required to create and share useful software.

An Approach to Software Development

Like creating great products, creating great software requires organization, forethought, and effort. Efficiently building software that meets the needs of those requiring virtual tools requires much more than the ability to code. Computers are quickly evolving to be faster and more efficient than ever so the potential and turnout for software is only ever increasing. To keep up with such a market, structure for development is required. The development of software consists of many stages recognized as specification, implementation, and a release to form and distribute useful software. However, to achieve desirable outcomes the process must start with formally addressing what is desired.

Creating software is taking an idea of a service or product and offering it a place in the real world. This can become a strenuous process that requires time that is hard to quantify, effort from multiple parties, and often funding whether that be an exchange of legal tender or personal time. Thus, it is crucial to begin the construction of software with a plan that allows one to minimize miscommunication and confusion so that the entire process becomes efficient as possible. This time spent can be recognized as the specification step. Here, many questions are presented such as what the purpose of the software is, who is the target audience, where will this software exist, and how can this software exist. Because this step has the potential to save time and effort, understanding the importance of each question is necessary and will further be covered.

The software in this stage is mostly in the form of an idea and must take on an identity so that all effort is not spent superfluously. Asking for the purpose of the software is the most important question as it allows for all subsequent actions to uphold the ultimate purpose. If this is not clear, time may be wasted, and the product may become drastically different from the original intent. The following questions such as who the software is for and where the software will exist are supporting questions that further define its identity. Knowing who the target audience is helps aim focus toward desired features and knowing where the software will be used, workstations or mobile, will allow for teams to account for which environments to develop for. Additionally, big projects involving multiple people usually require compensation for their time so allocation and distribution of funds must be planned to sustain development from start to finish. Once these questions are resolved, then practical development can take place.

The development process can split into two broad stages, building and testing. Now that the software has an identity, work can be sectioned and developed. This may be defined using customary practice language such as “front-end” and “back-end” where one is focused on the part of the software users will interact with and another is focused on the innerworkings of the program. These sections can be further subdivided, and tasks handed over to individuals. In this way, all the parts or modules will combine to form one product or software. Because a module may rely on another module, work on all parts of the project may not start at the same time and may be specified through a timeline in the specification step. Once the software is assembled, all parts being fully developed in their respective manner, a first test-run or trial may begin to expose bugs and errors that may have not been accounted for in the specification step or went unnoticed during development. Between these two stages, building and testing, focus may now shift back and forth to refine the software so that is ready for practical use.

Once all tests meet defined standards, an initial release of the software will take place conventionally defined as “1.0.” Before this happens, teams may opt to bridge development and release by releasing a beta version of the software. This, however optional, may be beneficial as it allows for “free” testing of the software before it is formally released and allows for feedback from the target audience, those who will be most impacted by the software’s efficacy. After the software’s initial release, its shelf-life may be sustained through patches and major updates that are influenced by time spent using the software and current trends that require new features. This is the tail end of software development, and it is now the product or service that had once existed through thought.

The ability to code is only one important part of many parts that are required to develop useful software. Because outcomes are hard to control from constructive work alone, it is best that forethought is given to the entire process to create structure throughout development. Giving the thoughts and ideas an identity, constructing from that identity, and supporting the release of software are all crucial to providing useful software. Time and resources spent creating software are maximized with structure. Whether it is a tool or entertainment platform, all forms of software can benefit from developmental structure.