Chapter 7: Rapid Development Practices

During development of major projects, coders must work efficiently at a high productivity rate to satisfy the requirements of the clients, executives, and company as a whole. Often, however, programmers are pressured to finish the project as soon as possible, which leads to security flaws, bugs, and an ultimately unfinished product. Initially, programmers took a waterfall approach that involved heavy load very early on in development. This approach actually reduces productivity a lot of the time, as programmers get tired out and neglect the testing methodology of their project. Agile methodology was used to replace the waterfall model as an alternative that produces results faster. This methodology was intended to reduce the stress and “Analysis paralysis” from too much information at one time. Using pseudo-agile methodology, the developers would spend more time testing and working out bugs, often causing them to stop working fast. Management typically does not like a cessation of work, but ensuring the code is stable early on is important in preventing troubleshooting later and may actually cause the project to be developed faster. The programmers should be focused on increasing productivity in interaction between people, computers, and people interacting with computers (and vise versa). The book lists a few ways to increase productivity, including redefining the task, increasing efficiency (of a given resource or in interactions between resources), increasing resources, and increasing effort. Developers can also implement iterations to their project, which acts as an experiment with a given outcome. Each iteration will change slightly, and each will be worked out until a certifiably better result comes out of the process. This will take more time, but is helpful in an efficient development strategy. Developers can also optimize these tasks to be even more efficient.

Chapter 8: API Design

In the book, two ways of solving problems are specified: start with a theory and work to prove it (“Big Picture”), and form a theory based in the facts laid out (“Detailed”). REST and API design are intended to please both these types of people and is adaptable to any developer’s style. REST is often attributed to ease-of-use and provides many preconfigured tools for a developer to use to their advantage, but REST was not specifically designed for this purpose. In a practical API design, some aspects conflict with the REST standards. Mainly, this method uses JSON, which means the API is at a disadvantage when it comes to linkability with media types. Because these implementations are not self-describing, these designs require proper documentation for their assets. This can be inefficient and time consuming, compared to REST, which will practically create the documentation itself. REST is mainly used in theory, while a practical API is used in practice. A developer can use the REST standards to develop their API to make the later stages of development easier. Opposed to JSON, which does not have a universal standard for linking media, REST does. JSON is still one of the more popular frameworks for development, however. This is due in part to its compatibility with JavaScript and the fact that it just works well as sites get more advanced with the use of JavaScript programming. RESTful APIs can distribute the processing power to different clients to reduce load on the main webserver. This can cause the developer’s project to run efficiently and meet the user’s expectations. More and more developers are switching to RESTful API opposed to JSON because of its ease of use, spiking productivity in the process. They can also be used to handle and efficiently use data on the client application side, which results in a more seamless user experience.