CSCC01 Reading Summary 2: “The evolution of Java Build systems”

Sameed Sohani (1003153115)

The paper is about the role of build systems in software development and maintenance and how a build system evolves and changes with the project. It includes a general background of build system concepts, discussing the two major layers known as the configuration and construction layer of the build system and references the make build system which is the standard system for C and C++ projects, then introduces two specific build systems, ANT and Maven as the focus of the paper is on Java.

The paper states that “[build systems] simplify the lives of developers, who constantly need to re-build testable artifacts after completing a code modification.” (McIntosh, Adams, &Hassan, 2012). However, “they have been largely ignored by software evolution researchers.” (McIntosh, Adams, &Hassan, 2012). From this, the paper poses two integral research questions to understand how a build system evolves alongside a project. These questions are: “Do the size and complexity of source code and build system evolve together?” (McIntosh, Adams, &Hassan, 2012), and “Does the perceived build-time complexity evolve?” (McIntosh, Adams, &Hassan, 2012). These questions are answered with case studies of some open-source projects. It then addresses that this paper is an extension of an earlier work to provide an empirical study of evolution of Maven build systems.

After showing case studies of both ANT and Maven build systems in projects, the paper concludes that “software build systems are complex entities in and of themselves. They evolve both statically and dynamically in terms of size and complexity… our case study indicates that build systems change continuously… Changes to the build system often need to be accompanies

by changes to the project’s source code.” (McIntosh, Adams, &Hassan, 2012). They also find several observations as a result of the case study, those being patterns of growth overtime that correlate the size of the project source code with the static and dynamic size as well as the complexity of a build system, the correlation between the project’s plugin count and the exponential growth of Eclipse’s build system, that a build system does not switch designs, the high correlation between the size of the build system and its Halstead complexity, that management of third-party libraries is a crucial factor in the evolution of a build system and that major project events such as restructuring or releases correspond with large changes in target coverage. These observations and conclusions answer the formerly posed research questions that they sought to answer.

What I found interesting about the paper was the case studies and how they show several correlated factors through different measures and values to obtain several statistics through several analyses. The statistics and graphs were nice to visually see the correlated factors and where they draw their conclusions from. Along with their explanations of the graphs and statistics, the paper outlines its findings well. I can’t relate to well to the complexity of a build system evolving over time as I have not used build systems much over my projects and haven’t had a large-scale project with a build system, however it gets me excited to be doing a larger project with a build system in this course and knowing how it is evolving alongside my source code.

References

McIntosh, S., Adams, B., & Hassan, A.E. (2012). The evolution of Java build systems.