The evolution of Java built systems

1. Summary

This paper summarizes the crucial role build systems play in the language of java, specifically ANT and Maven build systems. This paper describes two perspectives of ANT and Maven, a static perspective, where the complexity of the build system specification using software metrics adopted from the source code domain is examined and a dynamic perspective, where the complexity and overage of representative build runs are measured (McIntosh, Adams and Hassan, 2011).

It is also shown that that build system and source code size are highly correlated, with the source code restructuring, often requiring build systems restructuring, through the case studies of the build systems of size open source build projects. It is also shown that Java build systems evolve dynamically in terms of duration and recursive depth of the directory hierarchy (McIntosh, Adams and Hassan, 2011). It is apparent that build systems of complex entities and evolve both statically and dynamically in terms of size and complexity.

2. Findings and Benefits

Macintosh, Adams and Hassan finds that the Lehman's first twelve laws apply in the context of build systems where in this case study build system change is observed to be continuous. As an effect induced by Lehman's first law, build systems tend grow in complexity. Changes to a build system is not independent of the source code ans requires accompanied changes to the project's source code.

With the observations of ANT and Maven made through the six open source Java projects we can conclude that both static and dynamic size, and complexity of build systems show that the size of a project immediately correlates with the growth of the build system. With build systems like that of Eclipse's, the growth is related to its project's plugin count. Next, we can see that build systems that adopt a recursive or flat design does not switch to the other. Moreover, the build system's size or the BLOC is highly related to the Halstead complexity and is apparent is most of the open source projects. For those that adopted a Tomcat architecture, the management of third party libraries was a crucial faction in the evolution of their build system. Lastly, we can see that the most of the findings relate heavily towards previous observations made on the make-based systems with slightly different drivers of system evolution.

3. Thoughts

The evolution of build systems in Java have drastically improved and eased the development process of many projects as described by McIntosh, Adams and Hassan, 2011. ANT and Maven both bring a unique and powerful structure to a project and software projects specifically large software projects should adopt them in to their development process. With the understanding that build system's growth correlates to the project source size and code restructure often requires a build system restructure, I find that the benefits a build system such as ANT and Maven far out ways its resource intensity and projects teams should dedicate more resources to build maintenance and code restructure and adapt to this evolution.

References

1. McIntosh, S., Adams, B. and Hassan, A. (2011). The evolution of Java build systems. *Empirical Software Engineering*, 17(4-5), pp.578-608.