Chimzim Ogbondah

Ben Othmane

SE 339

10 February 2021

Assignment 1 Reflection

This article looked at two different software development practices, one took a more structured architectural approach while the other developed through continuous. The Agile process focuses more on small continuous improvement which does not emphasize having a road map to how the software architecture should be structured because it assumes it emerges in the end. There are many variations to the Agile process but the most common reported by the article were SCRUM and XP. Chen and Babar conducted a survey of 102 participants to test the theory on whether or not software architecture emerges through the Agile process.

Chen and Babar initially asked yes or no questions on whether or not software architecture emerged from small continuous improvement, then asking if it also emerged when they expected it to. From there they opened it up for short answer responses to understand what caused the software architecture to emerge or why it did not. One factor that led to emergence of software architecture in the Agile process was the maturity of architectural knowledge of those work in the team. It was noted that as maturity of architecture rose so did the emergence of software architecture since those working on the team roughly knew where to start and the different paths that refactoring could take them. The age of software systems also plays a role in the emergence of architecture with new more contemporary software systems based on design principals producing it through refactoring. Another factor in producing software architecture is ASR (Architecturally Significant Requirements) which was more dependent on the nature of these requirements and projects. It can be seen that the projects where the requirements are needed in the designing stage led to more sound software architecture through small continuous improvement. Whom the team consisted of and size were also key to the emergence of software architecture through refactoring. A trend could be seen that smaller teams and those who had more experience with similar projects and technology produced useful code.

Satisfactory code does not emerge through small continuous improvement when the maturity of architectural knowledge is low since they do not have the experience to guide the code in a meaningful way through constant refactoring. Older software systems using older design principals are harder to produce meaningful software architecture since they tend to be more monolithic. The ASR of a project play a very big role in if software architecture can be produced. For example, in projects that require security these requirements shape the architecture and need to be addressed early. In projects like this it’s very hard to produce software architecture since refactoring later on in the project will not always fix the issues. It should also be noted that bigger teams and less experienced team members tend to not produce sound software architecture and mostly lead to wasted time. Bad communication often does not lead to software architecture through small continuous improvement because team members are not on the same page about ideas which later leads to failure in deployment.

In conclusion, software architecture produced through refactoring is very unlikely due to many factors needed for its success. It is very situational for it to be produced through refactoring because the communication and maturity of the team needs to be high, and the project requirements also need to align with the focus of this software development style. If the requirements are key to the design of the project, then it can be seen that refactoring typically does not lead to structurally sound software architecture.