COM S 339 Assignment 1

Alec Meyer

February 9, 2021

Emergence of Architecture Through Refactoring

Agile is one of the most common software development practices and is used widely throughout the tech industry. Agile is also what we are talking about when we discuss satisfactory architecture being built from refactoring. This essay will discuss the downfalls of producing software through constant refactoring and failing a project's needs.

Satisfactory architecture emerging from refactoring is a considerable debate in the software architecture community. The idea is that a project can be developed without traditional architecture through refactoring and editing code. Agile development depends on this claim. This ideology allows engineers to disregard the future of the project's architecture based on refactoring any future issues.

Satisfactory architectures can also be shown to not emerge from refactoring. "Failure [architecture did not emerge] is also more prevalent in cases of large pre-existing products which either had no discernable architecture to begin with or had their architecture erode away over years of maintenance and haphazard addition of new features by persons who didn't understand the pre-existing architecture or ignored it for one reason or another." This quote from the paper is one example of how satisfactory architecture does not arise from refactoring. A point brought up multiple times through the article is that there must be good communication in a team when constant refactoring is happening. This is because a lack of communication with continuous refactoring will cause conflicts between sections of the project in question.

One way to look at AGILE strategies is that it is a development pattern whose organization and structure are based on its lack of a framework This makes it unlikely for an architecture to be made through refactoring. Refactoring doesn't allow structure since it is a strategy based on in-the-moment development. Thapparambil stated, "no agile methods discuss Architecture in any length." A factor that can warp the architecture of a project is having an ineffective safe net. A safe net is set in place so that if refactoring goes wrong it will not crash an entire system.

There are a few project-related factors mentioned in the paper, some of them are: the rate of change, size, type, the maturity of AK, system age. Each of these factors has its pros and cons when creating satisfactory architecture. The factors mentioned use refactoring as their primary way of development.

The rate of change is the speed at which refactoring occurs. Participants stated that a high rate of change would result in minor nuanced refactoring, which does not benefit a satisfactory architecture. On the other hand, low refactoring rates will muddy the primary goal of the project, causing non-satisfactory architecture.

Participants stated how projects too small and too large would suffer from non-satisfactory architecture. Projects with multiple teams can have communication issues, while projects with few people are hard to develop a strict architecture. The type of project has a similar result where a project too small will suffer from a lack of structure, and a project too large could become overcomplicated.

The maturity of architectural knowledge is another prominent proponent of satisfactory architecture. A team will be more likely to succeed if they have previous architectures to reference. System age is similar to maturity where an older system will not have the experience needed to create a successful architecture.

Since there is no set architecture in a refactoring-based development, recent changes are needed to be communicated. “They [the team] were used to work heads down in their cubicles for months without speaking to anybody. After that time they simply deposit hundreds of pages of useless diagrams and felt good about it. [...] the sense of responsibility that comes with Agile is not there” (Page 201). This quote sums up how necessary good culture and communication in a team is for developing satisfactory architecture through refactoring.