Antipattern: Over-Engineering

Author(s): Andreas, Jacek and Christian

Systems have an unnecessary complex architecture. A simpler approach would be sufficient to fulfill the business requirements.

Architects may tend to choose cutting edge technologies, frameworks and concepts for new projects. This might lead to the following problems:

- The developers have to spend more time and mental capacity on understanding the used concepts and technologies. Therefore, development might take longer.
- Logically dependent code might get scattered over several files or modules. Additional framework specific code might reduce readability. This makes code harder to reason about, more fragile and in the result harder to maintain.

Due to the focus on technology, business logic could be implemented later and/or with bugs. See Domain Allergy

What are some examples?

- Splitting a checkout system into too many services
- Generic Product model for 12 insurance products

Why does this happen?

Many developers have a natural curiosity for new concepts, technologies and frameworks. They like to try these out without considering, if they fit the problem domain. Furthermore, some developers prematurely optimize software for:

- flexibility: e.g. too generic software Misapplied Genericity
- extendability: e.g. building a framework instead of a single solution
- scalability: e.g. optimizing for high load without current demand

Other developers have a need for change. They don't want to use the same approach over and over again. They want to gain practical experience with new technologies.

How can we avoid getting into the situation in the first place?

- Keep solutions as simple as possible (KISS)
- Optimize only if required
- only implement for current requirements not for possible future requirements
- audit the requirements to address the actual problems of the business
- use innovation tokens

Sometimes problems don't need to be solved with software. Instead, processes in the business could be changed.

What are suggestions to get out of the situation if we ended up in it?

Refactor or rewrite the application.