Published in ZIO's Blog: Architectural Decisions, (Micro-)Services and More

Doc SoC Follow

May 7, 2020 · 4 min read · Listen

Y-Statements

A light template for architectural decision capturing

Architectural Decisions (ADs) answer "why?" questions about design and justify why an option is chosen. As a consulting IT architect at IBM, I followed their ARC-100 template for AD capturing on my projects, usually trimmed down quite a bit (following the first law of method adoption: "if in doubt, leave it out"). As a researcher, I proposed an even richer meta model later, optimized for decision reuse (decisions *required*, not decisions *made*, that is, or AD issues as opposed to outcomes). During this effort, I had to learn not to overdo it: the cost for maintaining full-fledged decision models, possibly shared between projects, became rather high ("feeding the beast").

My next two projects had senior managers and tech leads as sponsors, who did not want to spend too much of their precious time on reviewing lengthy text documents. One of the catchers of my work challenged me to fit each decision on one presentation slide (including rationale!). Giving people what they want is not necessarily a bad idea, especially if these people sponsor your project. So I rethought my approach to AD capturing and sharing and tried to focus on the bare essentials of a decision. The result are *(WH)Y-statements*, presented at SATURN 2012, used on a large enterprise transformation and IoT cloud project and published in an IEEE Software/InfoQ article.

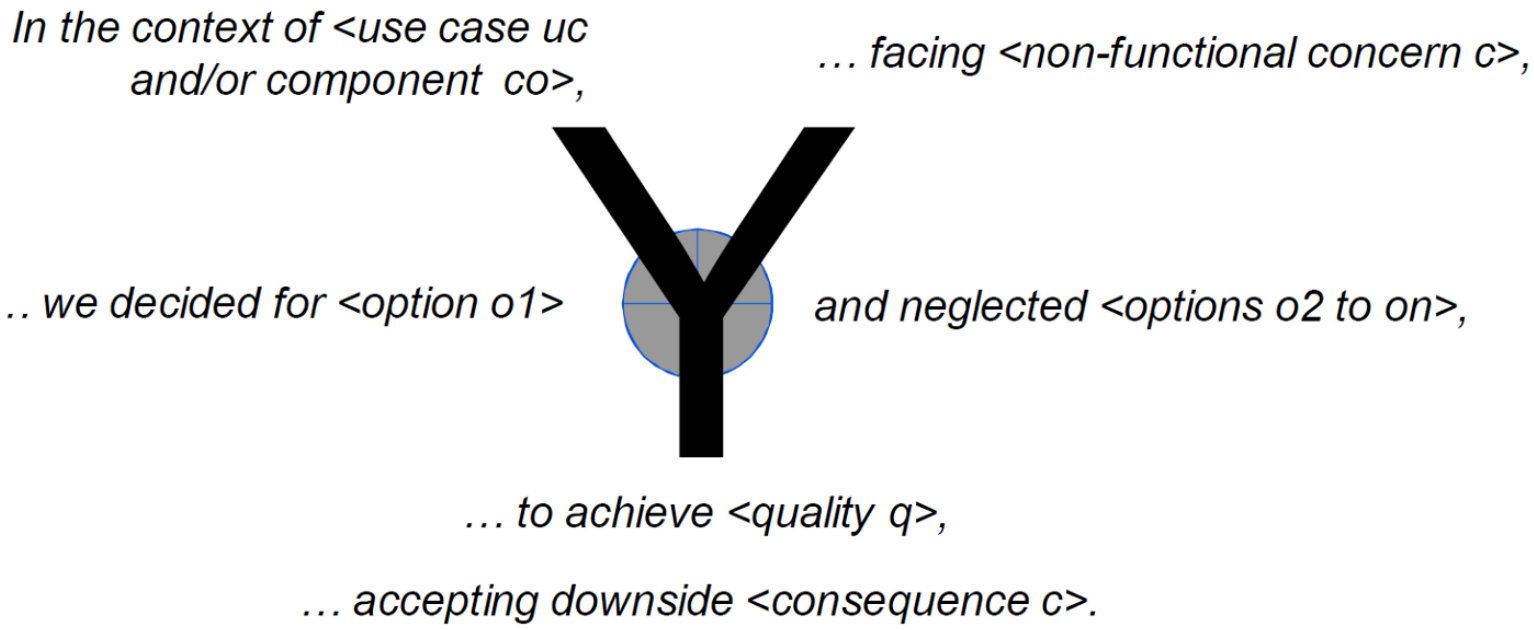
Here is an example:

In the context of the Web shop service,
facing the need to keep user session data consistent and current across shop instances,

we decided for the Database Session State pattern
and against Client Session State or Server Session State

1. *context*: functional requirement (story, . component,
2. *facing*: non-functional requirement, for instance a desired quality,
3. *we decided*: decision outcome (arguably the most important part),
4. *and neglected* alternatives not chosen (not to be forgotten!),
5. *to achieve*: benefits, the full or partial satisfaction of requirement(s),
6. *accepting that*: drawbacks and other consequences, for instance impact on other properties/context and effort/cost (both short term and long term).

The six sections of the resulting AD records form one (rather long) sentence. They can be visualized with the letter “Y”, pronounced just like the English word “why”, which explains why I decided to call them Y-statements:



This six-part structure is is much leaner than my previous attempts. George Fairbanks provided inspiration for it in his Architecture Haikus; decision outcome and its rationale are part of his one-napkin approach to architecture documentation (“x is a priority, so we chose y and accepted downside z”).

It is perfectly fine to modify the template; for instance, an extra half sentence starting with “because” can supply additional justifications gained during the decision making. If the single sentences get rather long, they can be split.

Tools: (M)ADR

has become a favorite of mine, for instance for pattern writing (all Microservice API Patterns are written in it); there is some tool support and additional templates in Markdown Architecture Decision Records (MADR).

Deep Dive: "Architectural Decisions (ADs) — The Making Of"

There is a longer version of this post also looks into the long history of architecture decisions (from industry to academia and back), compares Y-statements with other templates (such as the IBM one and Michael Nygard's ADRs), and gives examples of good AD justifications. It can be found here.

Adoption

I use Y-statements in teaching and consulting regularly. Others have picked them up too. The Cards for Analyzing and Reflecting on Doomed Software (card42) initiative by innoQ architects features them as the third of many very nice cards. The decision story approach in this thesis starts from them. And Herberto Graca seems to use them in the summary section of his ADR template, see his blog post "Documenting Software Architecture". See this post on LinkedIn for a few more examples.

I hope you find the Y-template useful too. If not, please tell me (wh)y :-)

Olaf

Next Up

I propose a *Definition of Done for AD Making* here (short version) and here (full proposal).