THE FIVE RESPONSIBILITIES OF ARCHITECTS

Eltjo R. Poort

Position paper - SATURN 2019 ‘Architecture Island’ Workshop

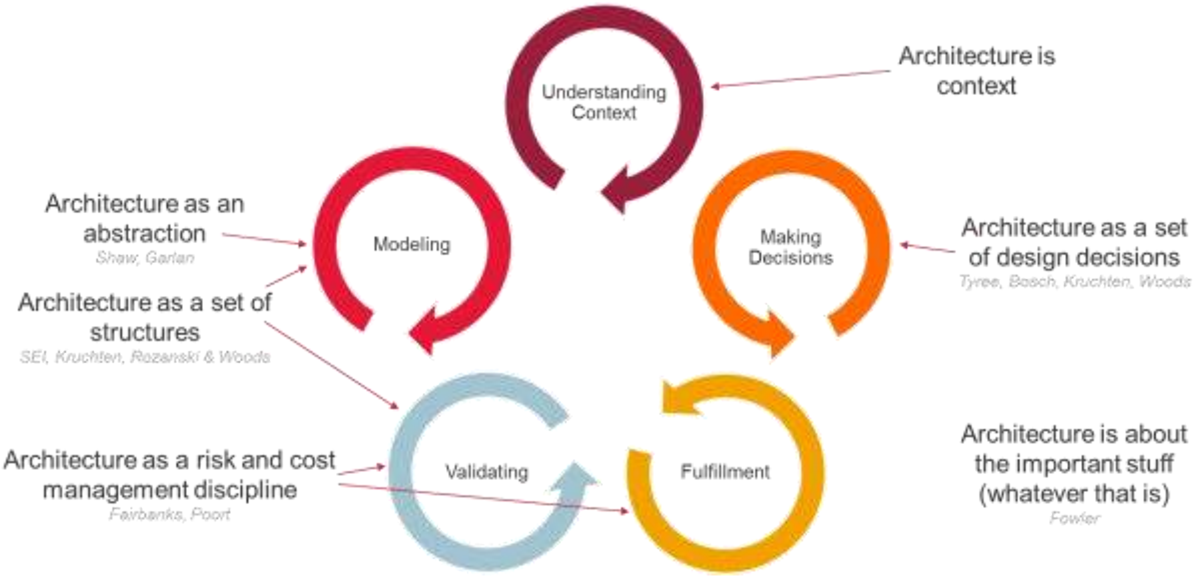
THE NATURE OF ARCHITECTURE

The perception of what ‘Architecture’ should mean in the context of software engineering has gone through a number of changes since it was first used in that context.

The original 1990s perception views architecture as a set of structures that represent an abstraction of a system being delivered – needed to deal with the growing complexity of typical software systems. In that view, the main responsibilities of the architect are to create (visual) models of the system, and to use those models to validate the architecture.

In the early 2000s, a second perception emerged, with a new responsibility focus: architects needed to make important decisions in order to create the right models of their solutions[1] (‘right’ meaning that they fulfill their stakeholders’ needs). If ‘abstraction’ and ‘structures’ describe *what* the architect creates, the decision making refers to *how* they create.

Around 2010, partly under pressure of the agile movement’s focus on business value, a third perception was proposed: the *why* was added to the *what* and the *how* of architecture. This view shed light on the goal of architecture: to improve organizations’ control over risk and cost[2] – not only during design, but extending the architects’ responsibility to the fulfillment of the solution.


Understanding Context
Architecture is context
Architecture as an abstraction
Shaw, Garlan
Modeling
Making Decisions
Architecture as a set of design decisions
Tyree, Bosch, Kruchten, Woods
Architecture as a set of structures
SEI, Kruchten, Rozanski & Woods
Validating
Fulfillment
Architecture as a risk and cost management discipline
Fairbanks, Poort
Architecture is about the important stuff (whatever that is)
Fowler

[1] I prefer the word ‘solution’ as the object of the architects’ work to ‘system’, since architects nowadays not only design systems (and systems of systems): more and more they tend to design transformational solutions that do not create new systems, but mainly change the behavior and organization of existing systems of systems.

[2] Poort, E. R., & van Vliet, H. (2012). RCDA: Architecting as a Risk- and Cost Management Discipline. Journal of Systems and Software, 1995-2013.

The list of architecture responsibilities would not be complete without the one prerequisite without which the architect would be unable to fulfill any responsibility: understanding the context of the solution, meaning the stakeholders, their needs, and the environment in which the solution is to be delivered.

So we end up with five responsibilities of architects (or of the 'architecture function' of organizations): understanding context, making decisions, modeling, validating and fulfillment of the solution. These five responsibilities also map very nicely to Philippe Kruchten's "Things architects actually do"[3], to Eoin Woods' "Architectural focus areas"[4] and to the RCDA practices we have been applying for years.

DEPENDENCIES

It is important to note the many dependencies between the five responsibilities identified above. Just to mention a few:

- modelling and decision making without understanding context will lead to wrong models and decisions
- modelling actually implies decision making (about decompositions, relationships, etc.)
- if there are no models and no decisions, there is nothing to validate
- fulfillment of unvalidated decisions and models leads to trouble

So fulfilling the five responsibilities is not enough: they should be fulfilled in a coherent way.

GOOD ARCHITECTURE

Since architecture is context[5], there's no such thing as *good architecture* in an absolute sense: the best one can hope for is an architecture that *fits* the stakeholder needs in its context. In my experience, the best fitting architectures result from paying proper attention to *all five* responsibilities mentioned above. This is not easy; due to mostly cultural pressures, dogmas and misconceptions, many organizations ignore some of the responsibilities, resulting in a flawed architecture function. Two extreme examples are the Waterfall Wasteland and the Agile Outback caricatures described below.

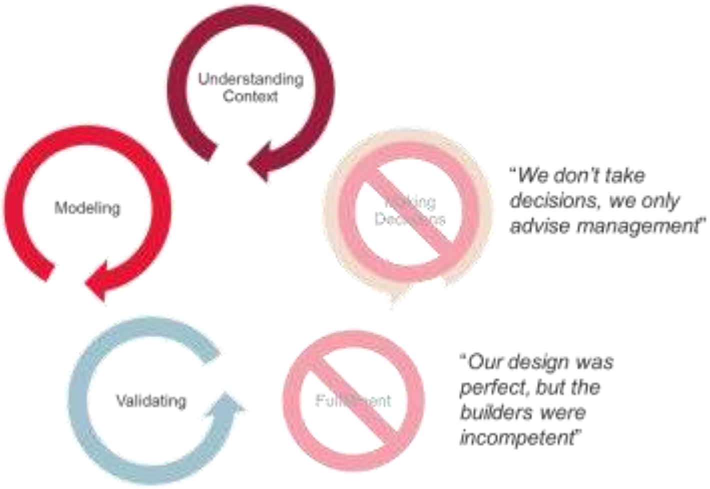
Paying *proper* attention to all five responsibilities, however, does not mean always paying *equal* attention: depending on the context, modeling may indeed require more attention than decision making, and validation may be more critical in some situations than in others.

[3] Ph. Kruchten, "What do software architects really do?" Journal of Systems & Software, vol. 81, pp. 2413-2416, 2008 doi: 10.1016/j.jss.2008.08.025.

[4] Woods, E, & Bashroush, R. (2019). How Software Architects Focus Their Attention. Journal of Systems and Software, submitted.

[5] https://eltjopoort.nl/blog/2018/01/31/architecture-is-context/

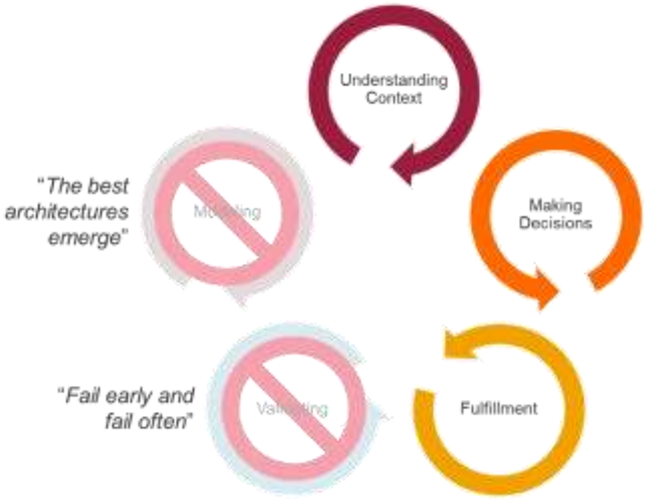
The Waterfall Wasteland


Understanding Context
Modeling
Making Decisions
"We don't take decisions, we only advise management"
Validating
Fulfillment
"Our design was perfect, but the builders were incompetent"

In the Waterfall Wasteland, the architects live in an ivory tower. They ignore the decision making and fulfillment responsibilities, which they consider to be someone else's. They have a very clear job description: to create perfect models and validate them against stakeholder needs. If the resulting solution is unsuccessful, it's obviously not their fault. The idea that they would be responsible for decisions or share responsibility for successful delivery is abhorrent to them: it would mean that their success would depend on the capability of others, and on their ability to cooperate…

The Agile Outback

In the Agile Outback, teams usually don't have architects (although they might use euphemisms like 'pathfinder' or 'master builder'). Modeling is studiously avoided, since "The best architectures…emerge from self-organizing teams"[6], and doing modeling might offend the agile gods and disturb the 'magical' emergence process. Similarly, validating designs is considered a waste of time: failing early and often is a much quicker way to learn and improve.



[6] http://agilemanifesto.org

ESSENTIAL ARCHITECTURE SKILLS

The five responsibilities of architects are a good basis for deriving the essential knowledge and skills of the architect:

- For **understanding context**, we need *analytic* skills to identify the environment and stakeholders of our solution, *communication* and *social* skills to understand stakeholder needs and *knowledge management* skills to cultivate and share that understanding.
- For **making decisions**, we need *decision making* skills like *making trade-offs* and *prioritizing*. We also need extensive knowledge of the relevant *architectural tactics and strategies* in our (business and technology) domains.
- For **modeling**, we need *creative* skills for *visualizing* our design, and knowledge of relevant *modeling languages and techniques* like *decomposition* and *composition.*
- For **validating**, we need *analytic* and *trade-off* skills (again), and knowledge about techniques for *risk management*, *cost estimation* and making *business cases*.
- To fulfill our **fulfillment** responsibility, we need *leadership* skills like *communication*, *convincing*, *listening* and *anticipation*. But we also need to know a bit about the *economy* of delivering software-intensive systems and the relevant *software lifecycles*.

Looking at this list, it is no surprise that good overall architects are very rare. It might be a good idea to spread these responsibilities over a number of people, and create an *architecture function* in an organization: a (possibly virtual) team made up of members with complementing skills and knowledge.

CONCLUSION

Architecture has five responsibilities: understanding context, making decisions, modelling, validating and fulfillment. Organizations need to pay proper attention to each of these responsibilities; what is 'proper' depends on the context. Dogmatically denying some of the responsibilities will lead teams astray into the waterfall wasteland or the agile outback. The breadth of knowledge and skills needed to fulfill these responsibilities is substantial, and hard to find in one individual; it almost always takes a team to produce good architecture.