

Introduction

Software Architecture

Richard Thomas

February 19, 2024

University of Queensland



Question

What is *Software Architecture*?

Software Architecture is design.

Design is not software architecture.

But...

Software Architecture is hard to define.

Let's hear from an expert



<https://www.youtube.com/watch?v=DngAZyWMGR0>

Okay so...

Definition 1. Software Architecture

The important stuff; whatever that is.

Question

What do *you* want from this course?

Maybe...

Definition 2. Software Architecture: The Course

A set of tools, processes, and design patterns which enable me to deliver high quality software.

High Quality Software?¹

Functional Requirements – Functional features to be delivered.

Constraints – Real world constraints on development.

Principles – Ideas adopted to encourage design consistency.

Quality Attributes – Quality of service & cross-cutting concerns.

¹Yes, “high quality” is intentionally vague.

Functional Requirements

- Architecture must enable delivery of functionality.
- Support interaction model.
 - A mobile dating app may be difficult to deliver using *Pipe and Filter*.
- Don't over architect.
 - A mobile dating app doesn't need a six-layer *PCBMER* architecture.

Constraints

- Externally determined restrictions
- Time and budget
- Technology
 - Interoperability with existing systems
 - Deployment platform
 - Vendor relationships
- People
- Organisation
 - Strategic or tactical system?
 - Politics may limit choices

Principles

- Standards developers are expected to follow
 - Avoid unintentionally breaking the architecture
- e.g. Architectural structure
 - Layering strategy
 - Location of business logic
 - Stateless components

Question

What are *Quality Attributes*?

Question

What are *Quality Attributes*?

Answer

Non-functional requirements for the success of software.

Quality Attributes: Examples

Modularity Components of the software are separated into *discrete modules*.

Quality Attributes: Examples

Modularity Components of the software are separated into *discrete modules*.

Availability The software is *available to access* by end users, either at any time or on any platform, or both.

Quality Attributes: Examples

- Modularity** Components of the software are separated into *discrete modules*.
- Availability** The software is *available to access* by end users, either at any time or on any platform, or both.
- Scalability** The software can handle peaks of high demand by *taking advantage of available computing resources*.

Quality Attributes: Examples

- Modularity** Components of the software are separated into *discrete modules*.
- Availability** The software is *available to access* by end users, either at any time or on any platform, or both.
- Scalability** The software can handle peaks of high demand by *taking advantage of available computing resources*.
- Extensibility** Features or extensions can be *easily added* to the base software.

Quality Attributes: Examples

Modularity Components of the software are separated into *discrete modules*.

Availability The software is *available to access* by end users, either at any time or on any platform, or both.

Scalability The software can handle peaks of high demand by *taking advantage of available computing resources*.

Extensibility Features or extensions can be *easily added* to the base software.

Testability The software is designed so that *automated tests* can be easily deployed.

Problem

Software cannot meet all quality attributes.

“Solution”

Software architects prioritise the important attributes.

“Solution”

Software architects prioritise the important attributes.

Definition 3. The First Law of Software Architecture

[Richards and Ford, 2020]

Everything in software architecture is a trade-off.

Definition 4. Wicked Architecture [Galster and Angelov, 2016]

There are often *no clear problem descriptions*, *no clear solutions*, good or bad solutions, *no clear rules* when to “stop” architecting and mostly team rather than individual work.

Definition 5. Wicked Architecture [Galster and Angelov, 2016]

There are often *no clear problem descriptions*, *no clear solutions*, good or bad solutions, *no clear rules* when to “stop” architecting and mostly team rather than individual work.

Don't expect “clean” solutions.

Why now?

Architecture is more important today thanks to *expectations* and *infrastructure*.

Big design up front is dumb.

Doing no design up front is even dumber.

- *Dave Thomas*

References

[Galster and Angelov, 2016] Galster, M. and Angelov, S. (2016).

What makes teaching software architecture difficult?

In *Proceedings of the 38th International Conference on Software Engineering Companion*, ICSE '16, pages 356–359. Association for Computing Machinery.

[Richards and Ford, 2020] Richards, M. and Ford, N. (2020).

Fundamentals of Software Architecture: An Engineering Approach.

O'Reilly Media, Inc.