# Introduction
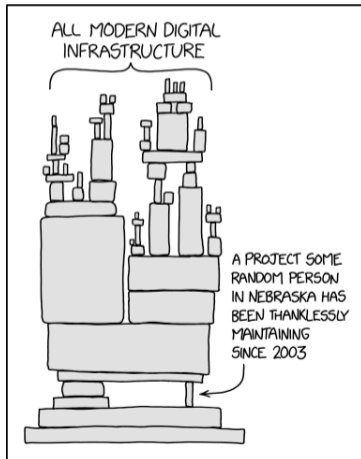
## Software Architecture

Richard Thomas

February 23, 2026
University of Queensland

# What is *Software Architecture*?

*Software Architecture* is design.

*Software Architecture* is design.

*Design* is *not* software architecture.

*So...*

*Software Architecture* is hard to define.

# Let's hear from an expert

*Okay so...*

> **Definition 0.** Software Architecture
>
> The important stuff; whatever that is.

# What do *you* want from this course?

*Maybe...*

> **Definition 0.** 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.

---

[1]Yes, "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

What are *Quality Attributes*?

# What are *Quality Attributes*?

Non-functional requirements for the success of software.

---

Hint: CSSE3002 / CSSE3012

# 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** Software is *accessible* 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** Software is *accessible* by end users, either at any time or on any platform, or both.

**Scalability** 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** Software is *accessible* by end users, either at any time or on any platform, or both.

**Scalability** 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** Software is *accessible* by end users, either at any time or on any platform, or both.

**Scalability** 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.

**Testibility** Software is designed so that *automated tests* can be easily deployed.

Software cannot meet all quality attributes.

*"Solution"*

Software architects prioritise the important attributes.

*"Solution"*

# Software architects prioritise the important attributes.

---

on a software-by-software basis

**Definition 0.** The First Law of Software Architecture
*[Richards and Ford, 2020]*

Everything in software architecture is a trade-off.

**Definition 0.** 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 0.** 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.