Systems Architecture

A PRACTICAL APPROACH

Approach

This is DIFFERENT!

We will not treat requirements as separate from the technology/ implementation

We will weave the concepts and importance of requirements into the Architectural process

We will, actually, start with Architecture and Architecture concepts – then relate requirements back to what we learn in Architecture.

The goal is to have a more ‘real world’ approach to Architecture and how requirements relate to Architecture

The goal is to ensure you understand the foundational elements of what makes software work, so you can then actually create software that DOES work (well).

The desired approach is to have hands on activities that give you a more visceral feel for how architecture shapes what Software Engineering is about

Yes, you will have to ‘write’ … but you will also have to ‘create’, and learn

So you think you know software?

But do you understand what you are doing?

Do you understand WHY your code works?

Do you understand HOW it works?

Do you understand all the pieces that come together to make it work?

Do you understand how changes in one area affect other areas?

Do you understand how the architecture and the software affect the people who use it?

It’s all architecture

Until you understand the architecture, you can’t really build powerful applications with your code.

And what is an architect, anyway?

Architect:

[A] person who plans, designs and oversees the construction of [applications/ systems]

What skills does the [software] architect need?

- Computers & OS, Networks, Software
- People, Processes, Business, Law …
- Communications, Requirements, Analysis, …

Yeah, that's a lot!

And we will cover many of these topics [not all in depth … you have a lifetime for that!]

What is Software Architecture

The software architecture of a system is the set of structures needed to reason about the system. These structures comprise software elements, relations among them and the properties of both.

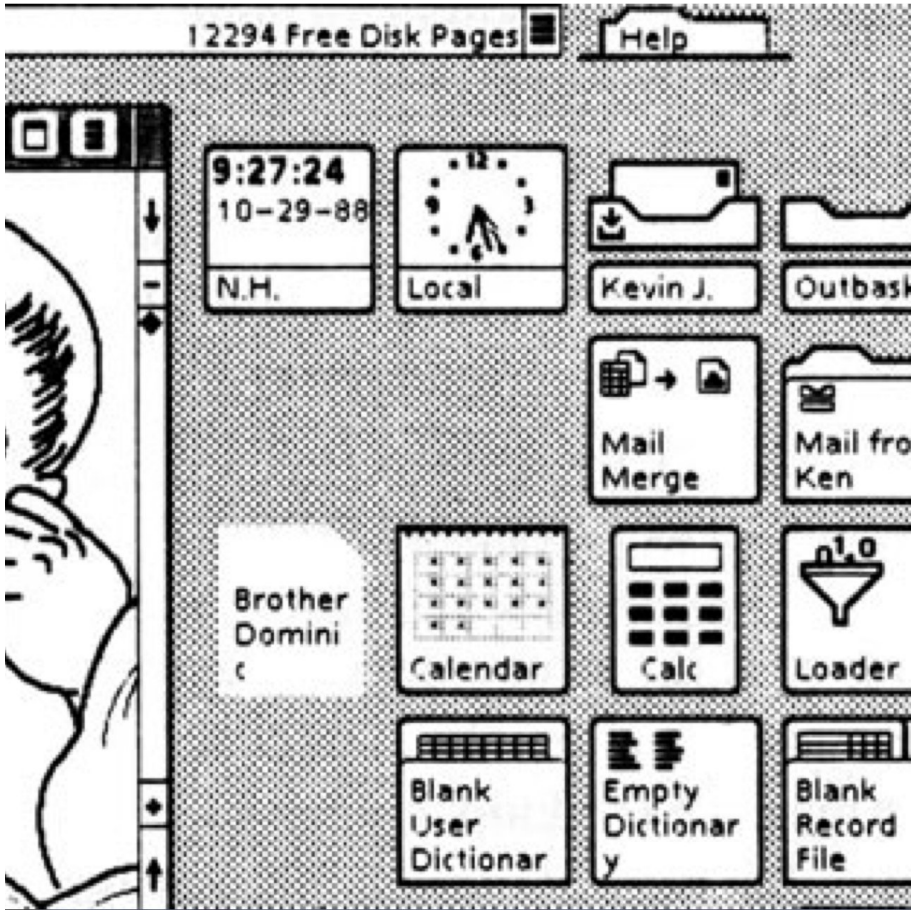
Observations:

- Every software system is constructed to satisfy an organization's business goals
- Architecture is a bridge between business goals and the implementation of a system
- Every software system has a software architecture
- Every software system affects users and the environment around it

Architecture is an Abstraction

- An architecture selects certain details and suppresses others
- Architecture is concerned with the public facing portion of the system
  - APIs
  - Public Classes
- Abstraction is essential to taming the complexity of an architecture
- Metaphors can provide good tools for abstracting away complexity
  - Desktop Metaphor – Alto, Viewpoint, MacOS, Windows, Gnome

Example



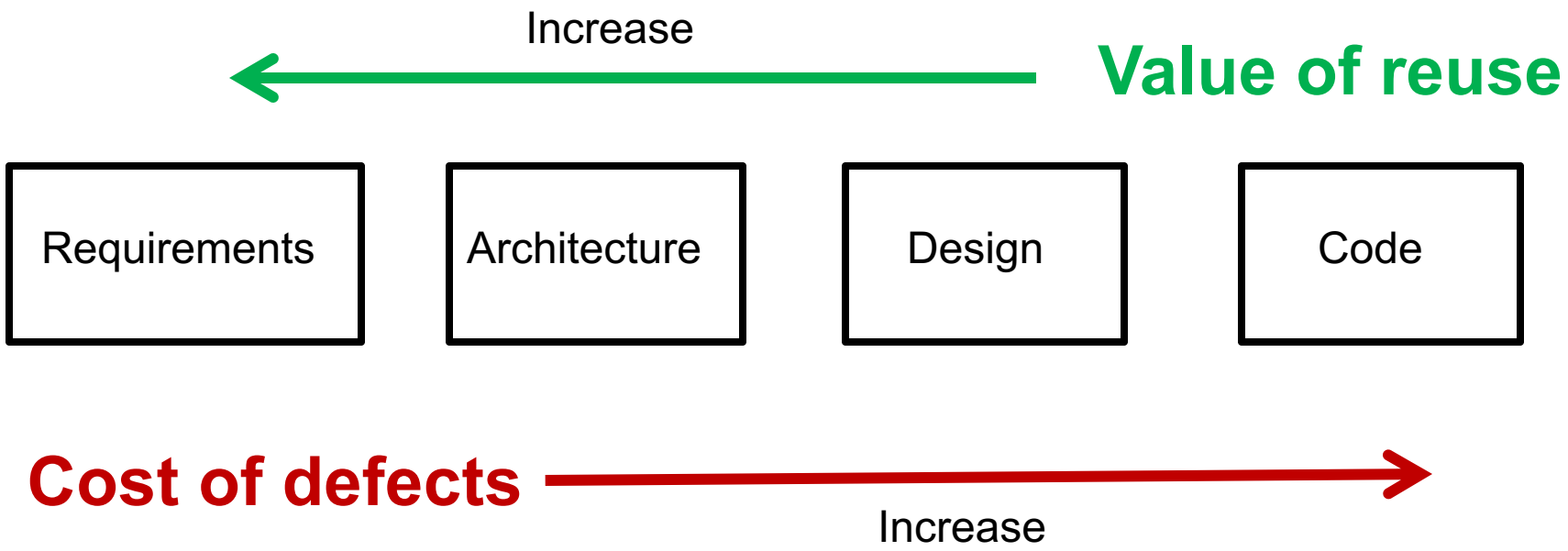
You probably don't recognize this version of a desktop metaphor, but

- You know what each of these icons does, or at least can make a good guess
- Good software architecture metaphors work the same way

Patterns are an area in software architecture that is rich in metaphors:

- Pipe & Filter
- Blackboard
- Etc.

The Benefit of Good Architecture



What Makes a “Good” Architecture?

Product (Structural) Guidelines

Well defined functional modules using information hiding, and separation of concerns

- **Encapsulates changeable aspects** to facilitate change
- Enables **independent concurrent development** by separate teams

Quality attributes achieved using well-known architectural patterns and tactics specific to each attribute

Limited dependencies on a particular technology or commercial product or tool

What Makes a “Good” Architecture?

A well thought-out architecture must consider these important principles:

- Build to change instead of build to last
- Understand the end user needs and the domain before designing components
- Identify sub-systems in your product and consider layers and components to abstract them and identify the key interfaces
- Use an incremental and iterative approach to designing the architecture
- Learn from history, document your decisions and identify and mitigate key risks
- Do not under-invest in architecture

https://www.infovista.com/blog/the-importance-of-software-architecture

Architecture is a set of Software Structures

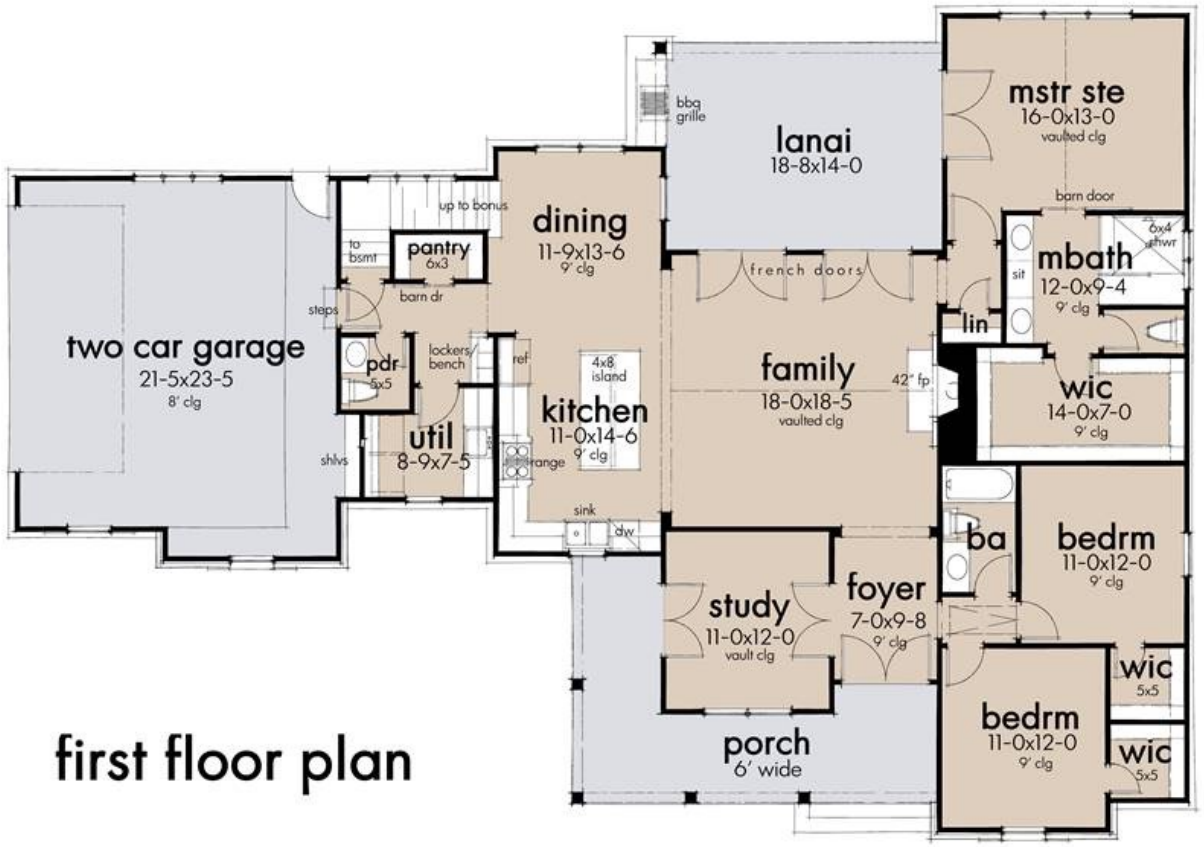
No single structure defines the architecture

There are three broad categories of structures:

- Module Structures – the static design of the system
- Component-and-Connector Structures – describe the dynamic aspects of the system
- Allocation Structures – how the software maps into non-software entities, think about how a system is built.

We will discuss these in detail throughout this course

Architecture of a house



When you build a house:

You lay out things like:

Foundation structure

Framework (rooms, doors)

Wiring, plumbing

Then builders follow the instructions to put up blocks, timber, walls etc.

How does the architect know what to tell the builders?

They ask the client (Requirements!)

Similar with Software

https://www.thehousedesigners.com/plan/mill-creek-farm-7844/

Architecture is about Behavior

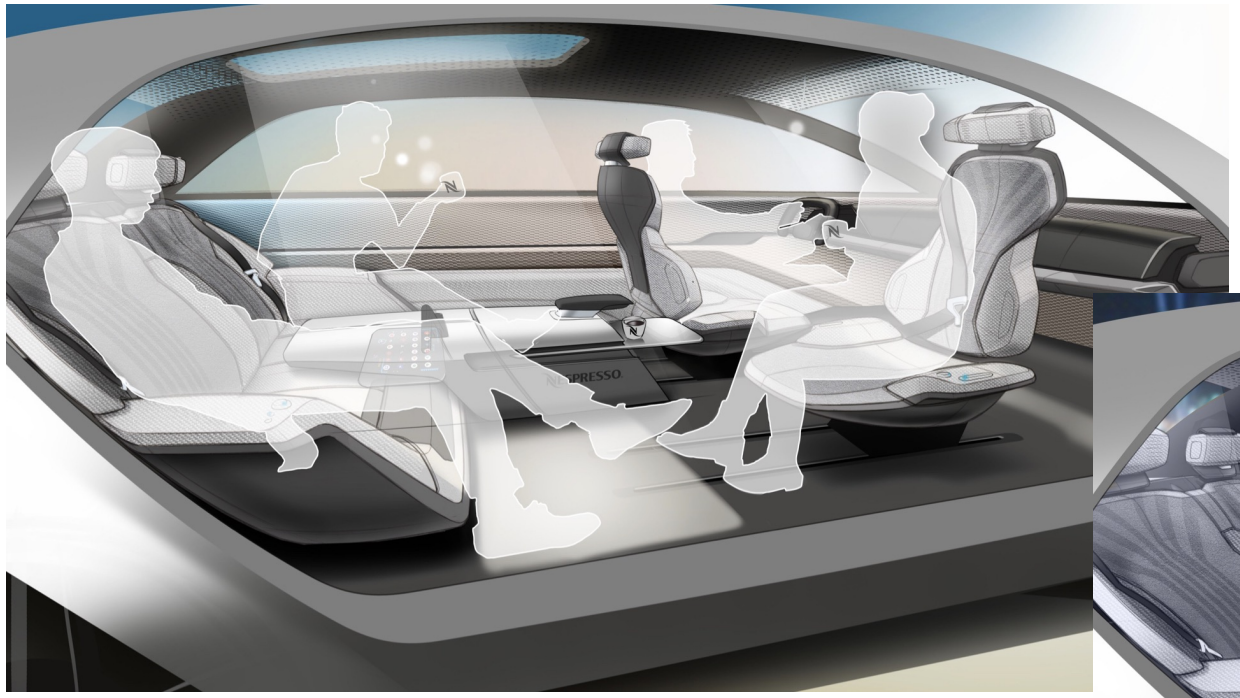
Software structures provide the physical building blocks of the system.

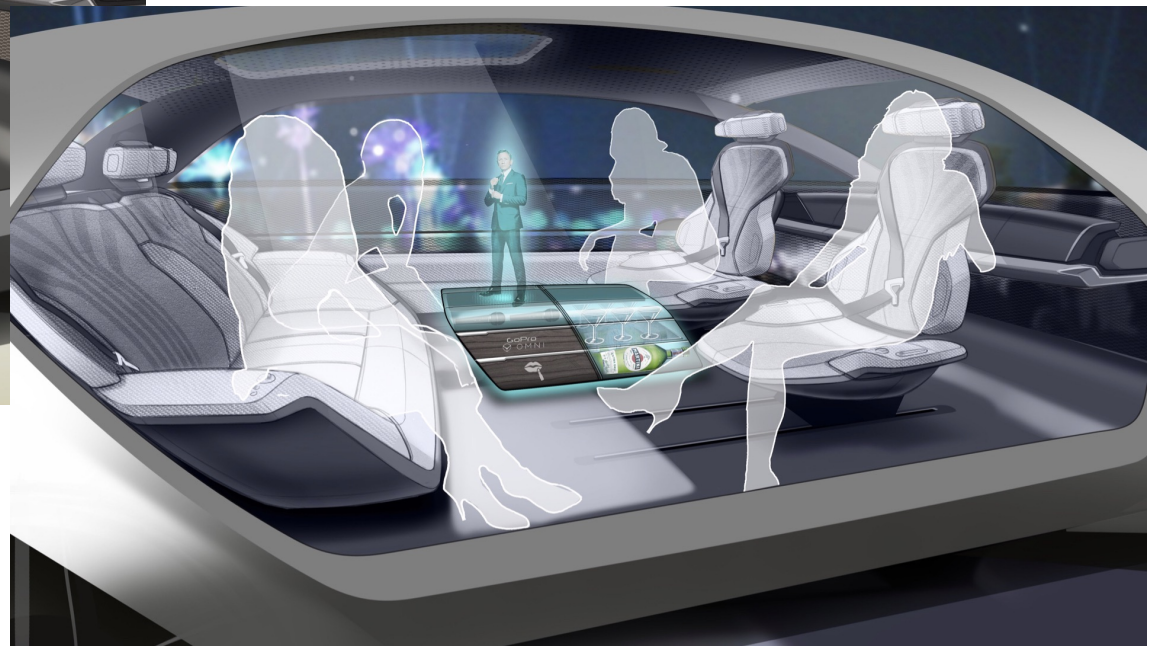
But, to fully capture a system we need to describe how it is supposed to perform

- Who/what does it interact with?
- How does it behave in response to specific requests?
- What does it respond when something goes wrong?

Understanding, capturing, documenting and ensuring that these ideas are carried forward into the implementation is **Software Architecture**.

Concept Car – understanding behavior





Designing an autonomous car interior for endless possibilities

For Grammer AG, a leading interior component supplier, we defined a vision on how semi-autonomous car interiors might look for the next generation of cars with Level 4 Autonomy.

https://www.granstudio.com/grammer

What type of requirements matter to Architecture?

Does the architect need every detail?

No, only things that affect the architecture

Examples: # of stories; # rooms; position of house relative to sun; local regulations/ ordnances; geographic location (hot/ code); type of heating; open spaces desires

Examples of don’t care: Paint color; trim types; kitchen appliances; …

Types of requirements:

User requirements

Workflow requirements

Environment and constraints

… but not every detail

ASRs

In houses, architects need to understand the physics and science that are behind the decisions and impacts of their decisions (load, stability, safety, aesthetics, …)

In software you need the same thing. Understand what is behind all the possibilities for arch. choices and decisions and understand the implications with each choice.

There are Architecturally important needs, and then ‘other’

- ASR = Architecturally Significant Requirements
- PS: Sometimes ‘other’ comes back as architecturally important

Understanding basics

We will build from the ground up – just like a house

In software, what makes an Application? What are the main pieces?

- Computer
- Connectors (networks)
- Components
- And then your own software

To be able to architect software, you must first be literate in the basics

- Before you can write a novel, you must learn English, grammar, etc.
- We will give you a (brief) foundation in computer literacy before architecture

To be able to architect, you must be able to analyze and measure!

Gall's law

1975 observation known as Gall's Law (named for pediatrician and systems design theorist John Gall)

- "A complex system that works is invariably found to have evolved from a simple system that worked,"
- "A complex system designed from scratch never works and cannot be patched up to make it work. You have to start over with a working simple system."

No 'big-bang theory'. Architecture (like everything else in software) is inherently iterative …

John Gall (author) - Wikipedia

Discussion Assignment

Software Architecture In Practice (4th ed.)

- How does an architecture serve as a basic for analysis? For decision making … etc.