An Overview of Software Design

Parinya Ekparinya

Parinya.Ek@kmitl.ac.th

What is **“Software Design”** ?

Definition in the 70s

Nowadays, it is called "UX and UI Design"

During an interview with David E. Liddle in 1978, he said:

"Software design is the act of determining the user's experience with a piece of software. It has nothing to do with how the code works inside, or how big or small the code is. The designer's task is to specify completely and unambiguously the user's whole experience.... The most important thing to design properly is the user's conceptual model. Everything else should be subordinated to making that model clear, obvious, and substantial. That is almost exactly the opposite of how most software is designed."

Definition in the 90s

The Association for Software Design (ASD) membership brochure offers a definition:

"Software design sits at the crossroads of all the computer disciplines: hardware and software engineering, programming, human factors research, ergonomics. It is the study of the intersection of human, machine, and the various interfaces—physical, sensory, psychological—that connect them."

Not quite right… This is now "HCI"

Definition in 21st Century

Stated in "*A science of design for software-intensive systems*" by Peter A. Freeman and David Hart in 2004:

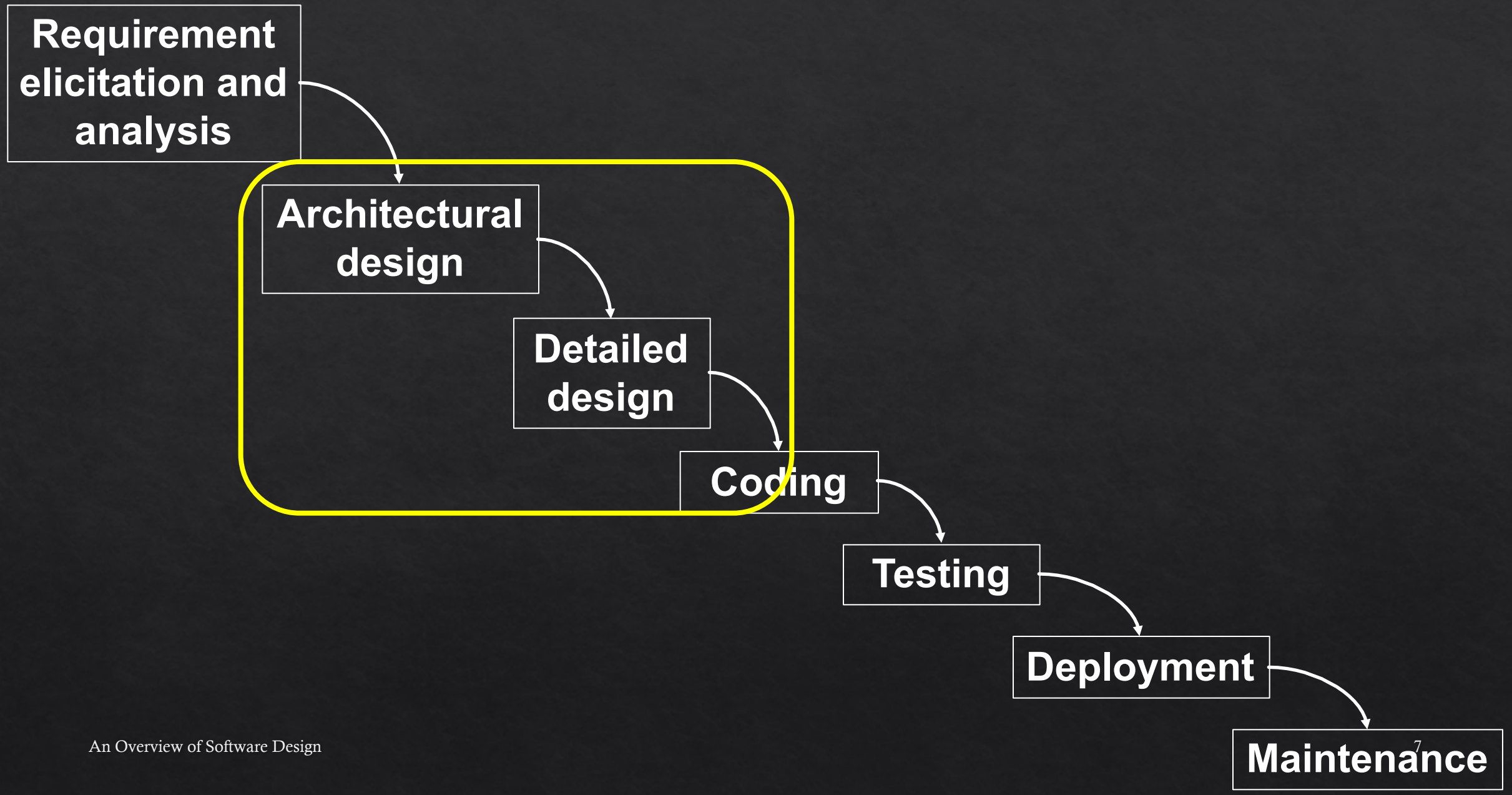
"Design encompasses all the activities involved in conceptualizing, framing, implementing, commissioning, and ultimately modifying complex systems, not just the activity following requirements specification and before programming, as it might be translated from a stylized software engineering process."

This sounds better but there are 2 aspects!!

- All the activities involved in conceptualizing, framing, implementing, commissioning, and ultimately modifying complex systems

- The activity following requirements specification and before programming

The subject partly covers these!!


Requirement elicitation and analysis
Architectural design
Detailed design
Coding
Testing
Deployment
Maintenance

Architecture and Detailed Design

- Software design can be partitioned into:
  - Architectural design
  - Detailed design
- Some might even suggest 3 design levels:
  - Architectural design
  - High-level design
  - Detailed design
- In practice, it is difficult to differentiate between these levels.

What is **“Software Architecture”**?

What comes to your mind when you hear the word

“ARCHITECTURE”?


Library
Entrance

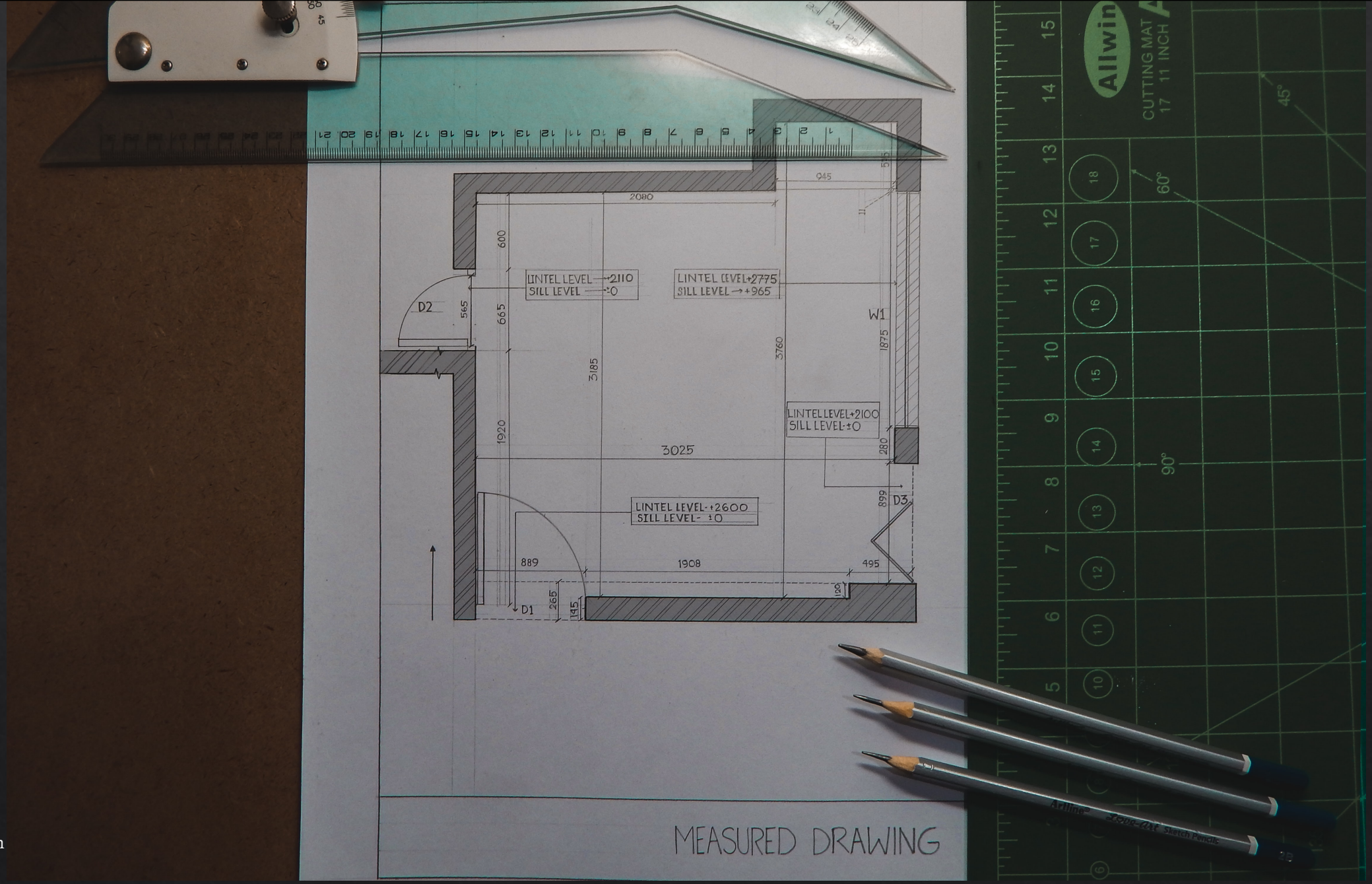


Photo by SOHAM BANERJEE on Unsplash

Definition of Software Architecture

One of the most popular definitions is from the Software Engineering Institute (Clements et al., 2010):

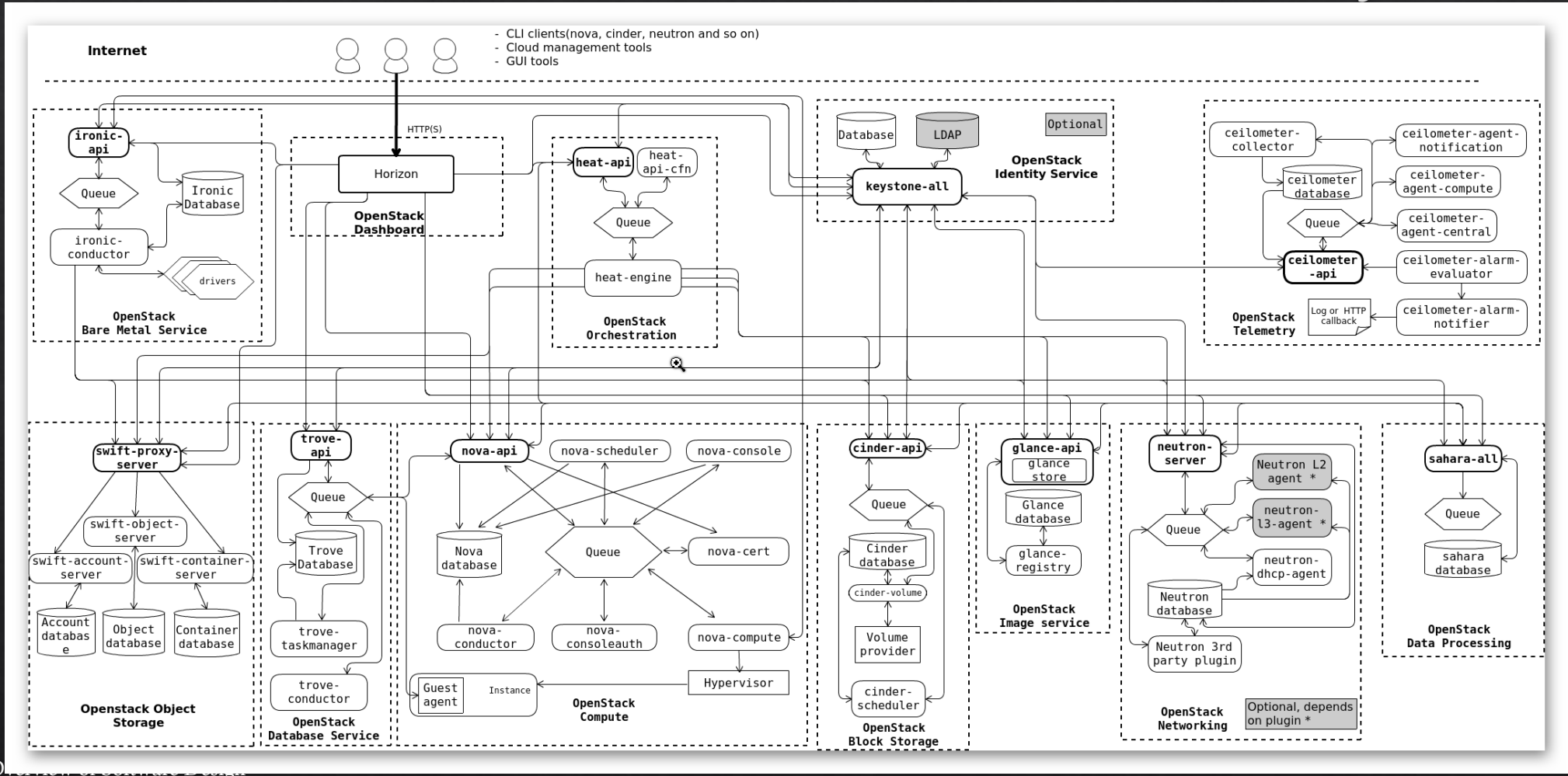
"The software architecture of a system is the set of structures needed to reason about the system, which comprise software elements, relations among them, and properties of both."

Why is (software) architecture important?

- Architecture acts as the skeleton of a system
- Architecture influences quality attributes
- Architecture is (mostly) orthogonal to functionality
- Architecture constrains systems

Fairbanks, G. (2010). *Just enough software architecture: a risk-driven approach*. Marshall & Brainerd.

Architecture acts as the skeleton of a system



https://docs.openstack.org/install-guide/_images/openstack-arch-kilo-logical-v1.png

Architecture influences quality attributes

- Given that a proxy performs both access control and data masking, please consider the followings:
  - Caching for performance improvement
  - Replacement of access control
  - Scaling up a data masker

Architecture is (mostly) orthogonal to functionality

When the architecture is poorly matched to the functionality, however, developers will struggle against it.

Architecture constrains systems

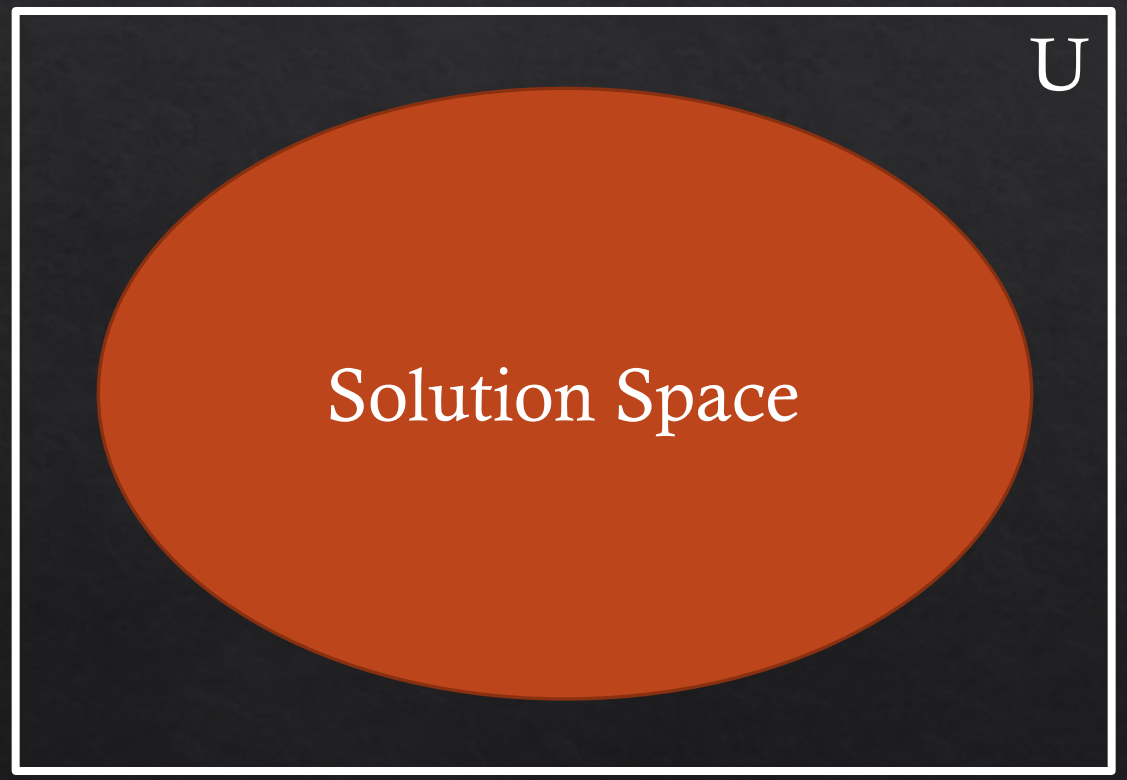
- A train is constrained by its tracks!!
- Procedural programming
  - Edgar Dijkstra: Go To Statement Considered Harmful
    https://homepages.cwi.nl/~storm/teaching/reader/Dijkstra68.pdf
  - For example: Pascal, C
- Functional Programming
  - Functions are treated as first-class citizens
  - Immutable Data Structures — Once created their values never change!
  - For example: LISP, Erlang, Haskell

When is (software) architecture important?

- Small solution space
- High failure risk เสี่ยงต่อความล้มเหลวสูง
- Difficult quality attributes (to be discussed in later weeks…) วัดประสิทธิภาพยาก
- New domain
- Product lines (common factors)

Fairbanks, G. (2010). *Just enough software architecture: a risk-driven approach*. Marshall & Brainerd.

Small solution space: (Venn diagram illustration)



High failure risk

Photo by Mathias P.R. Reding on Unsplash

Photo by Alex Azabache on Unsplash

New domain

- Meaning it is new to you. It does not need to be an innovation!!
- For instance, when you need to develop your first app on Android,
  you may need to pay more attention to its architecture
  even though you have already created a dozen web applications before.
- In other words, your knowledge and experience are valuable!!

Product lines (common factors)

- Some set of products share a common architecture.
- The shared architecture may easily support some kinds of product variations, but difficultly support others.
- For instance, a cross-platform application might be easily upgraded on Linux, but its Windows based counterpart is almost impossible to change.

What is **“Detailed Design”**?

What comes to your mind when you hear the word

“DETAIL”?

Photo by Sven Mieke on Unsplash

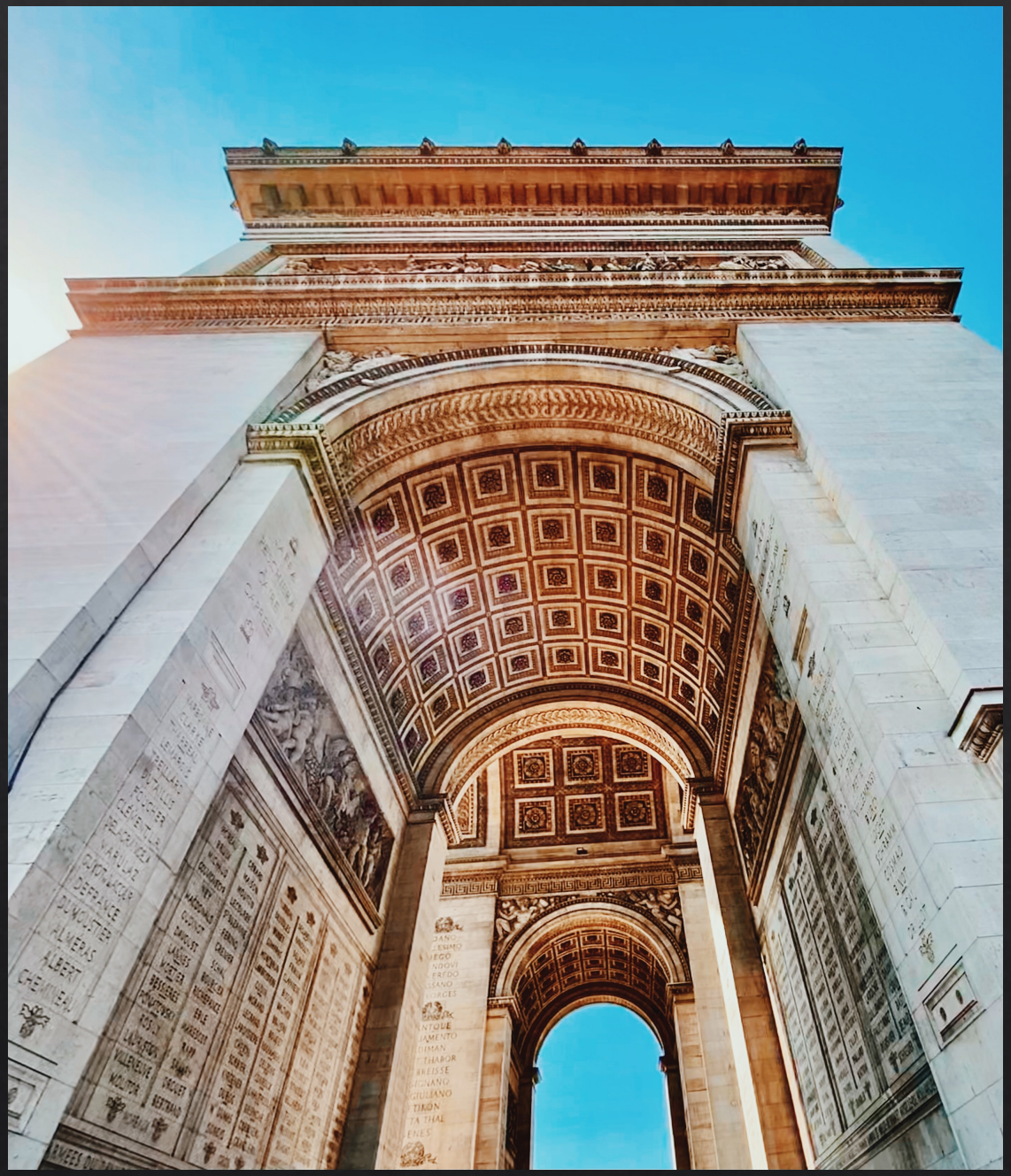


Photo by Andor on Unsplash

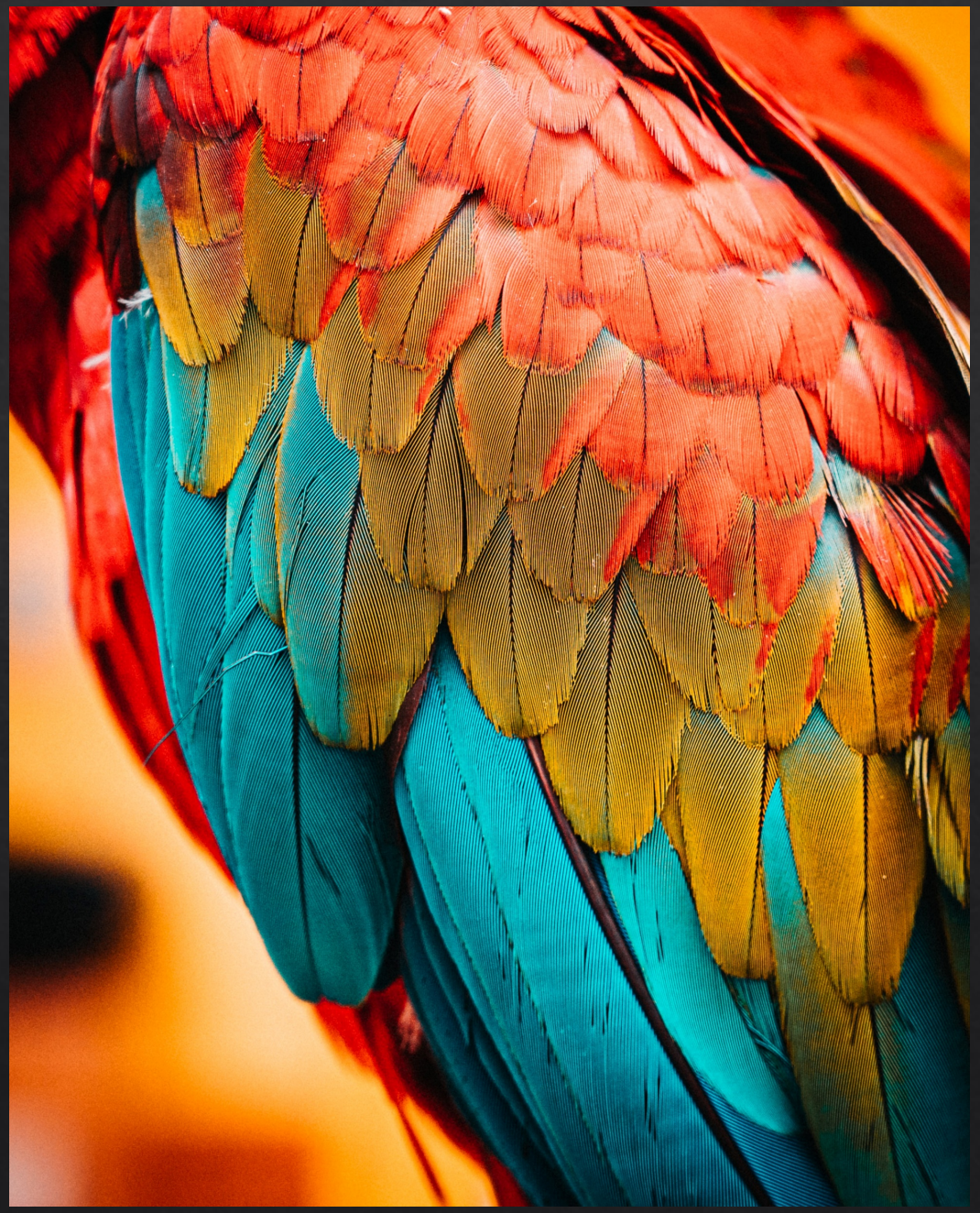


Photo by Jan Kopřiva on Unsplash

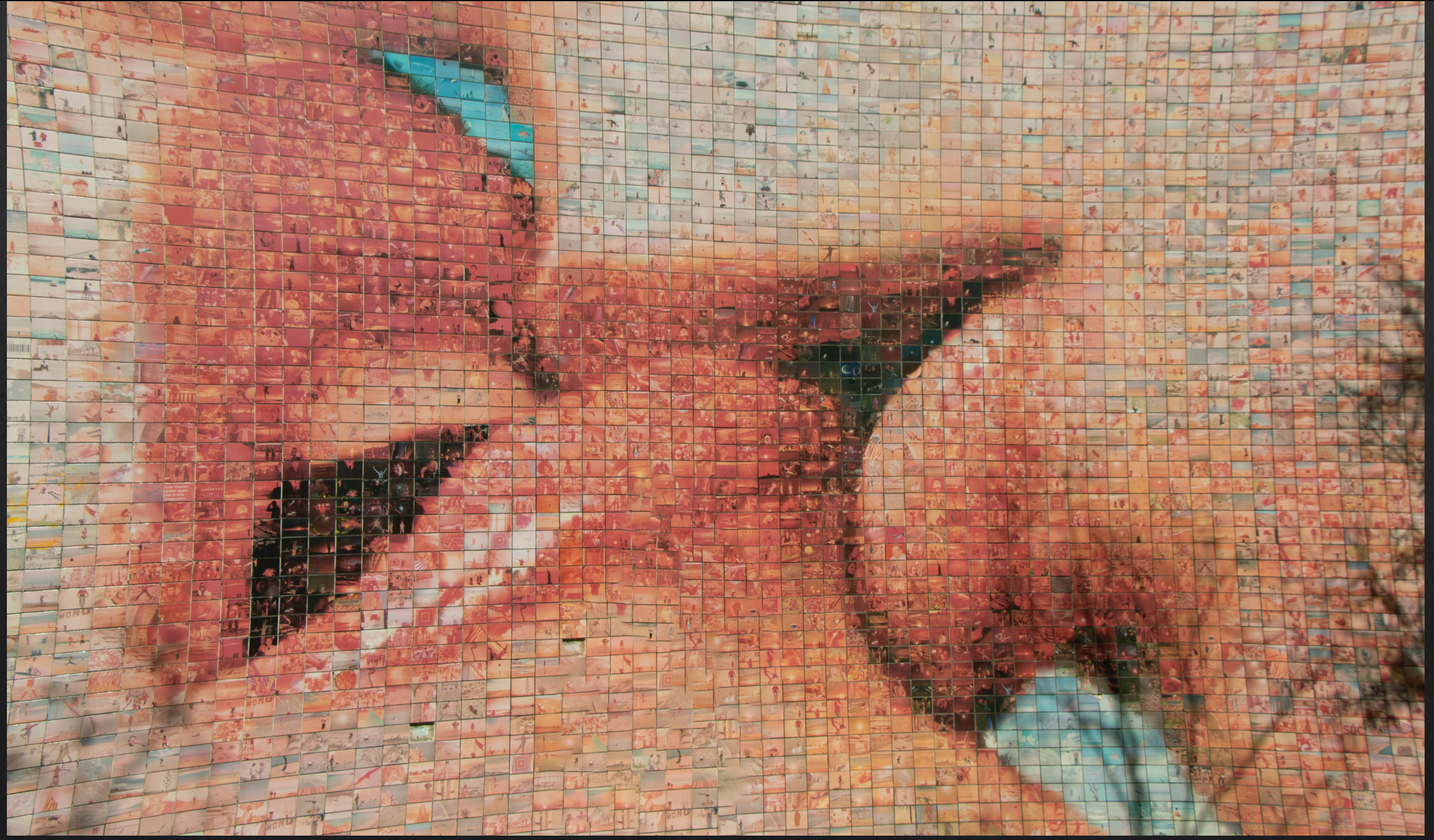


Photo by Nik Nikolla on Unsplash

What is detailed design?

Basically, it covers the rest of design decisions.

- Development tools
  - Programming languages
  - Programming frameworks and libraries
- Components and their organization
  - Data structures
  - Classes
  - Methods

We cannot cover them all !!

- Interaction between components, e.g., communication protocols
- User interfaces
- Etc.

This is what we focus on!!

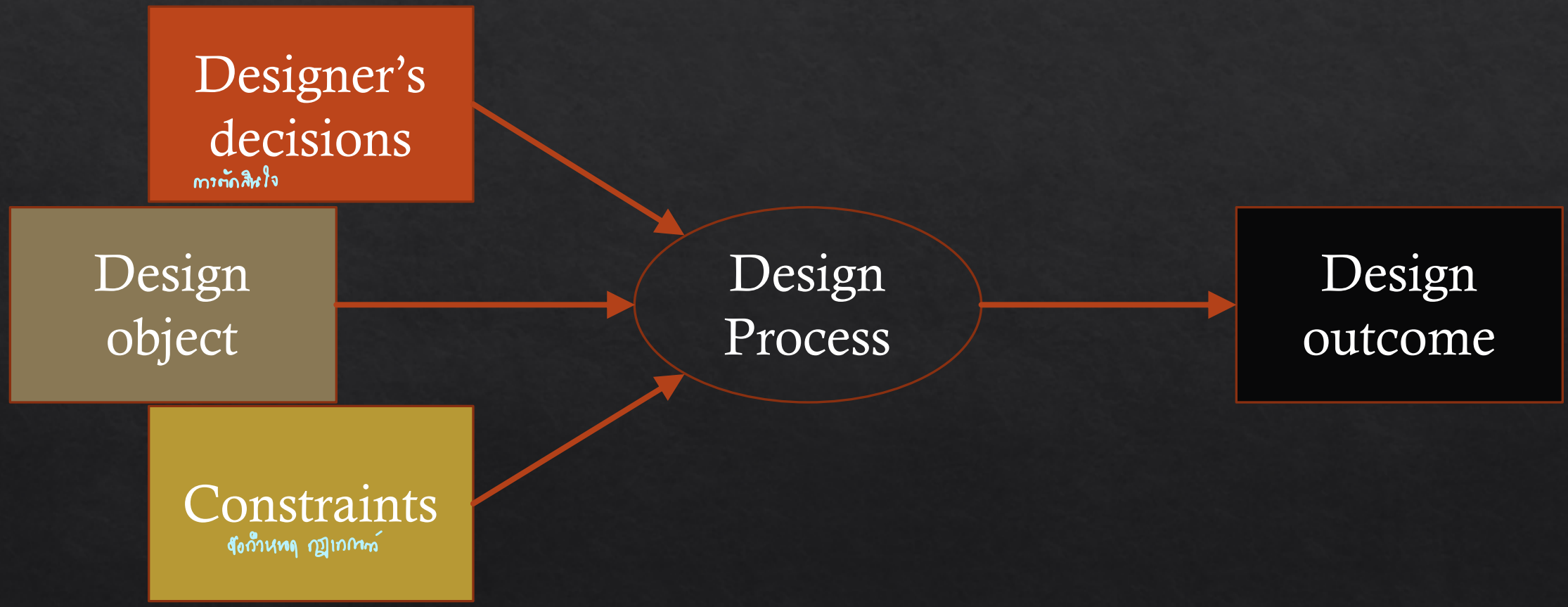
Design of components and their organization

- The design of components directly affects quality attributes, such as:
  - To tolerate change
  - Easy to understand
  - To be used in many software systems

- Some design principles to avoid common issues:
  - KISS – Keep It Simple Stupid
  - GRASP
  - SOLID principles

Design Process (in general !!)

Design Process



Development process variations

1. Is there up-front design?
2. What is the nature of the design
   - planned/evolutionary
   - redesign allowed
3. How is work prioritized across iterations?

Process	Up-front design	Nature of design	Prioritization of work
Waterfall	Big Design Up Front	Planned design No redesign	Open
Incremental	Optional	Planned design & redesign allowed, Evolutionary design	Often feature-centric
Agile	None	Evolutionary design	Highest customer value first

Is **software architecture and design** **less important** than it used to be years ago? If so, why?

Summary

- A variety of informal definition of software design
- Architecture
  - What & Why
- Detailed design
  - What & Why (partly)
  - “How” will be discussed in a few weeks later…
- Design Process
  - Variations