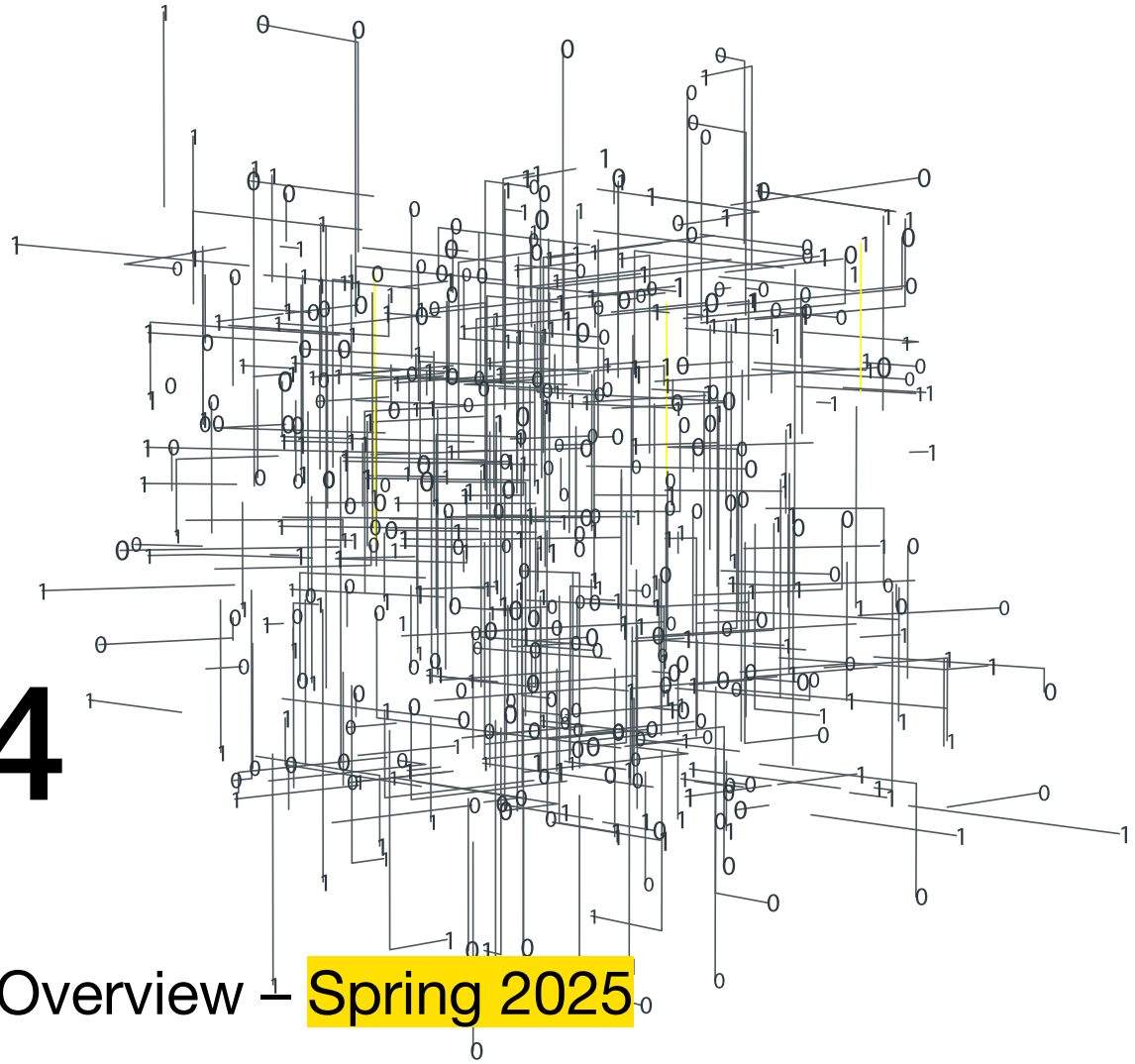
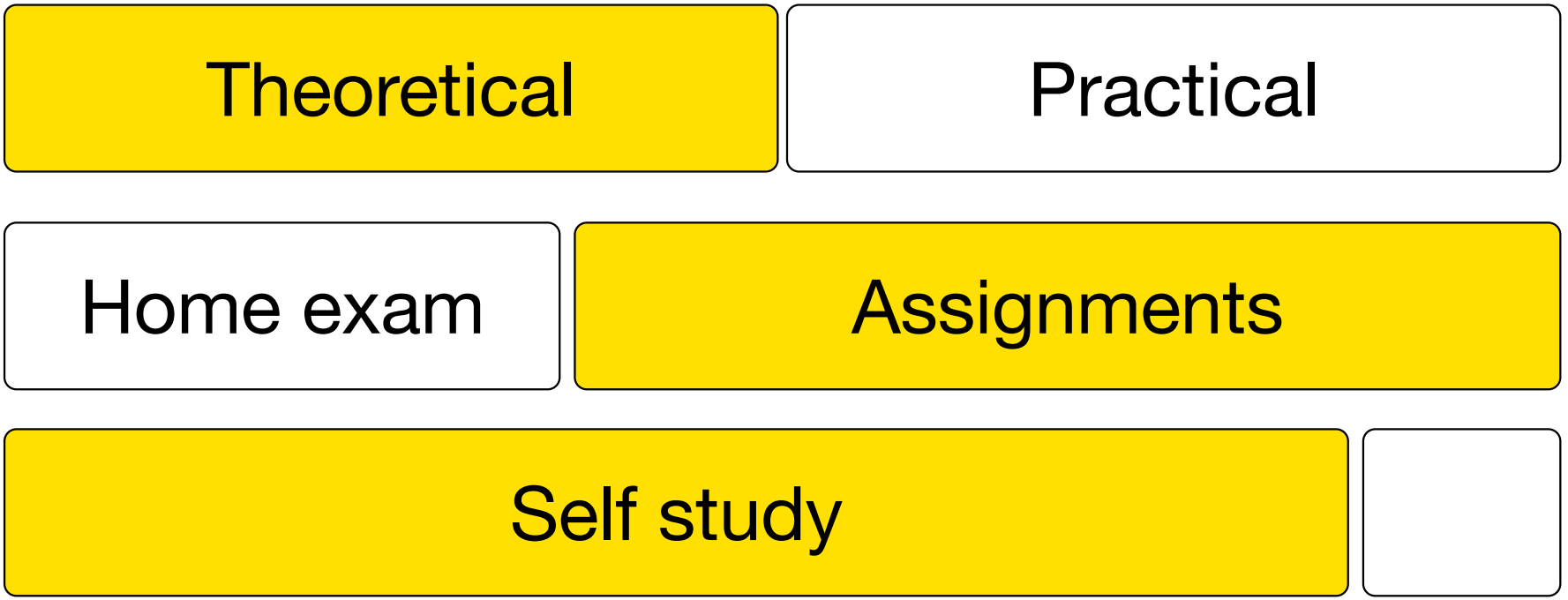


2DV604

Course Setup and Overview – Spring 2025

Linnæus University

Course Setup

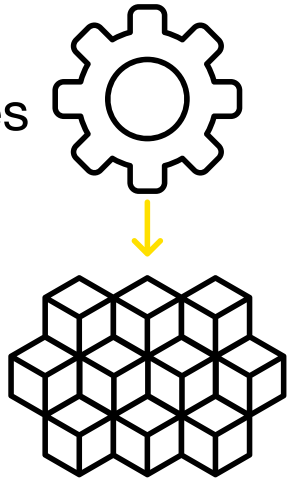


Objectives

- explain and apply software architecture concepts,
- describe software architecture design and evaluation methods
- perform basic software architecture design and evaluation
- explain and apply advanced software architecture design principles.
- describe and apply software architectures documentation concepts an d strategies
- explain the connection between software architecture and software qu ality
- describe how software architectures may assist in software reuse

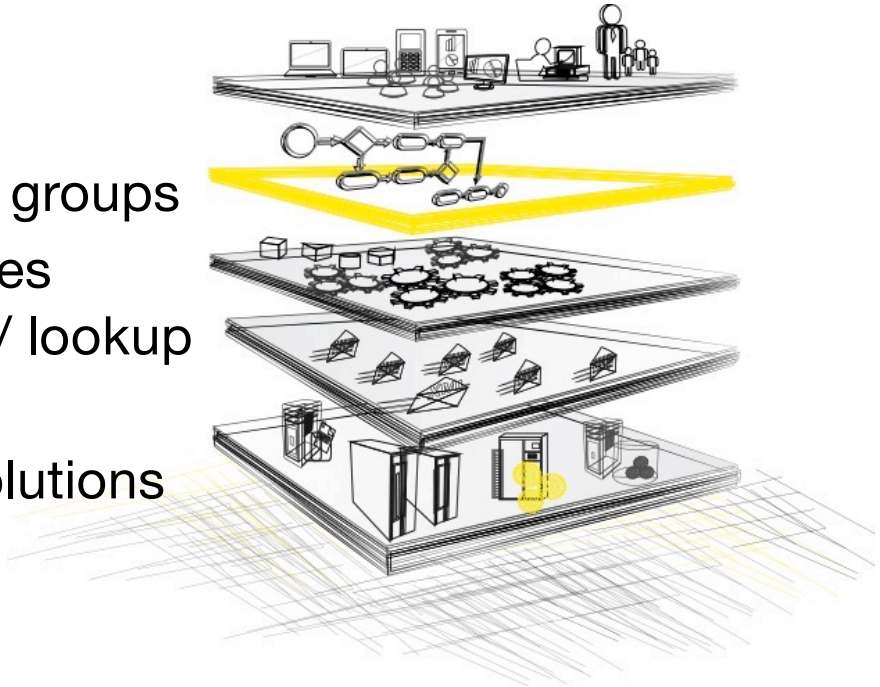
Content

- introduction to software design and software architectures
- introduction to software architecture concepts
- overview of architecture description techniques and architectural view s
- architectural styles and patterns
- software product-lines – concepts and its architectures
- software architecture design and evaluation



2025 instance

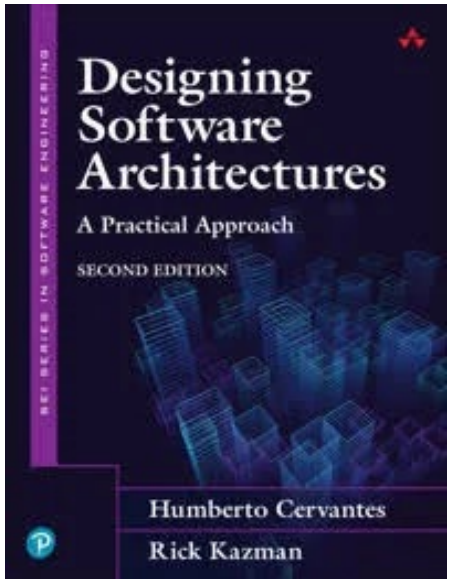
- Lectures
  and time on tutoring/discussions in smaller groups
- Detailed Reading Instructions & Study guides
  encourage reading and learning not search/ lookup
- Case studies and exercises
  exemplify key points, key problems, and solutions
- Tutoring sessions
  Q&A, examples, discussion



Examination

- Assignments
- Written exam

New book!



- Humberto Cervantes och Rick Kazman,
- Designing Software Architectures: A Practical Approach
- Most recent edition (2nd). Addison-Wesley Professional.

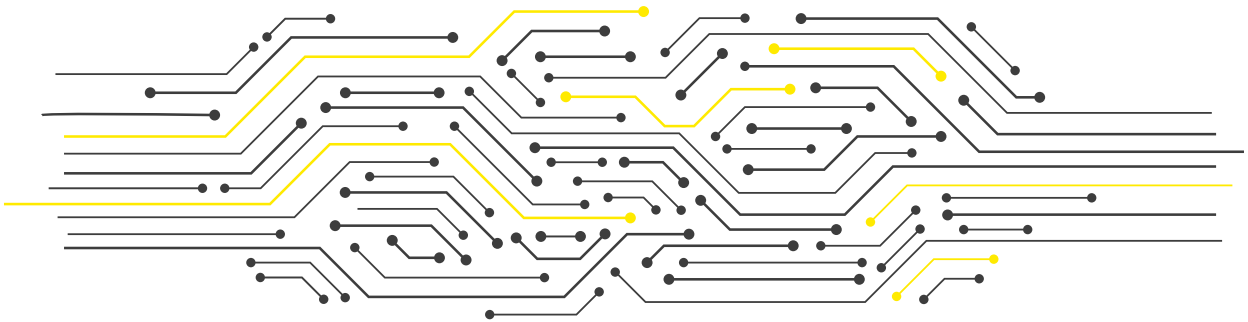
Tutoring & Q/A

Format

- 2-hours every week

Questions and Discussions

- Exercises
- Study questions
- Assignments/Project



Examination

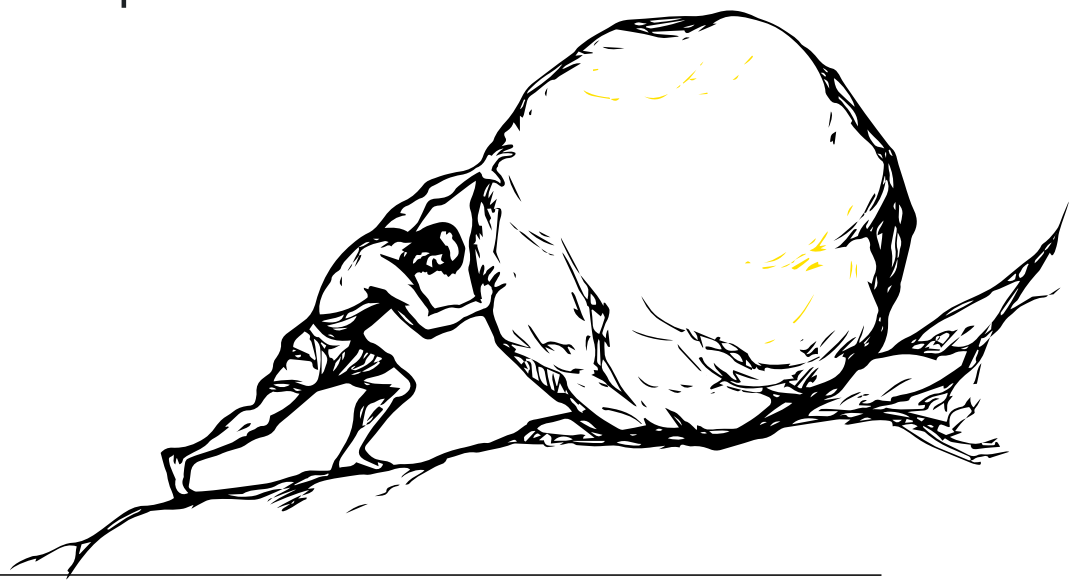
Practical

- Assignment 1 – Functional Decomposition
- Assignment 2 – Architecture Requirements and Design
- Assignment 3 – Design with tactics and patterns

Written examination

March 24, 14 – 19

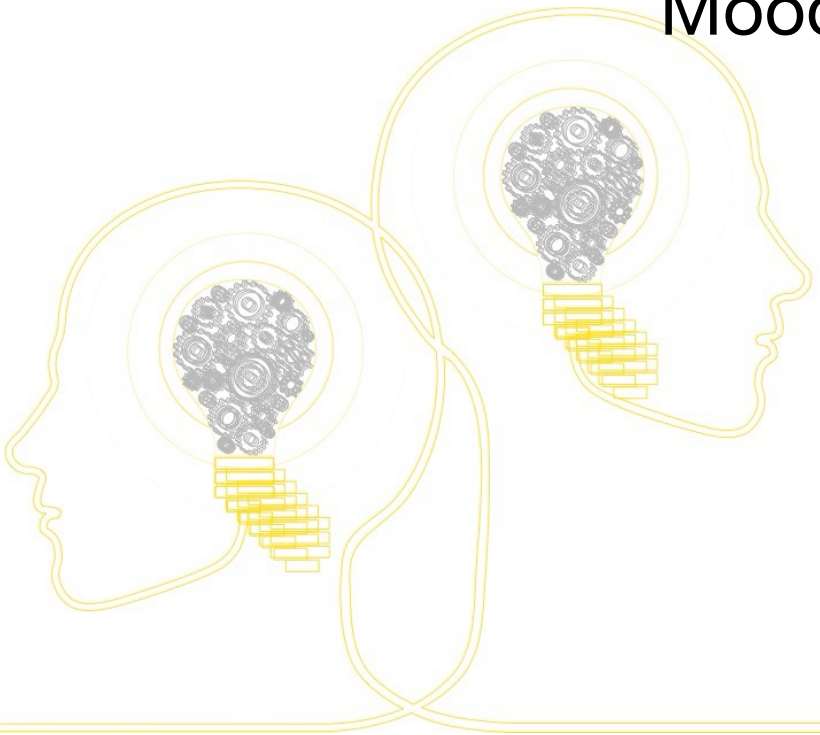
Retake, May 9, 8 – 13



Communication

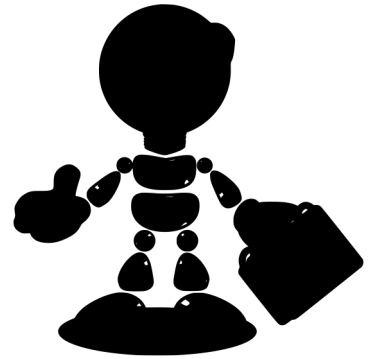
Slack channel - coursepress.slack.com **2dv604-2025**

Moodle – forums, submission/feedback



slack

AI Statement



Assistive AI is ok – e.g., language checkers

GenAI

Since genAI creates content, its use must align with specific ethical, academic, and procedural guidelines to ensure transparency, proper attribution, and learning integrity.

We assess your knowledge and skills for the course's learning outcomes.

We regulate when and how you may use genAI. In this class, the use of GenAI is governed by three levels **Forbidden**, **Cited use**, and **Explorative use**

Teaching Statement

The course is 7,5 HEC, corresponding to at least 20 hours per week. We have scheduled 30 hours for lectures and around 20 hours for tutoring. This course contains a minimum of 150 individual study hours.

We expect that you prepare for the lectures and tutoring sessions, read the recommended parts of the course literature, attend the lectures, complete the quizzes, and work on the hand-ins and exercises.

Use the *reading instructions* and *study questions* to prepare and use the opportunity to ask questions and discuss during lectures. Don’t make the mistake and start with the assignments unprepared.

If you do your part, we promise to provide answers and help you understand and learn the theoretical and practical parts the course covers.

Why Software Architecture?

Arguments with examples.

Linnæus University

Architecture Decisions

Examples

Decisions that need to be joint

How do we store information?

How is the security of the system managed?

Architecture Decisions

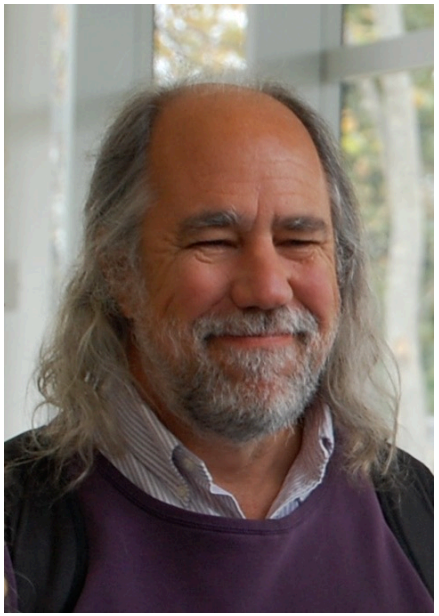
Examples

How do we store information?

What are the alternatives?

Which option suits our needs?

How are we to use it?



All architecture is design but not all design is architecture; architecture represents the set of significant design decisions *that shape the form and the function of a system where significant is measured by the cost of change!*

Grady Booch 2023 on software architecture.

Grady Booch, IBM Research, UML principal creator

Design Decisions

Developing software is about making (the right) decisions!

Collect and Understand the requirements!

Design and Implement a system that meets the requirements!

Adding to the complexity – size matters!

System size
The number and relative importance of a decision
The team size
Share decisions within a team

Design Decisions

Checklist

1. Impact on business value and risk?
2. Key stakeholder concern?
3. Unusual quality-of-service requirement (at least one order of magnitude more advanced than previous ones)?
4. About external dependencies that are uncontrollable, unpredictable or unreliable?
5. Cross-cutting, system-wide impact?
6. First-of-a-kind character (novelty for team)?
7. Caused bad experience and trouble in the past?

Different design decisions

Examples

What communication strategy should we have in a distributed system

What data type should we use for the customer's first name?

Design Decisions – Examples

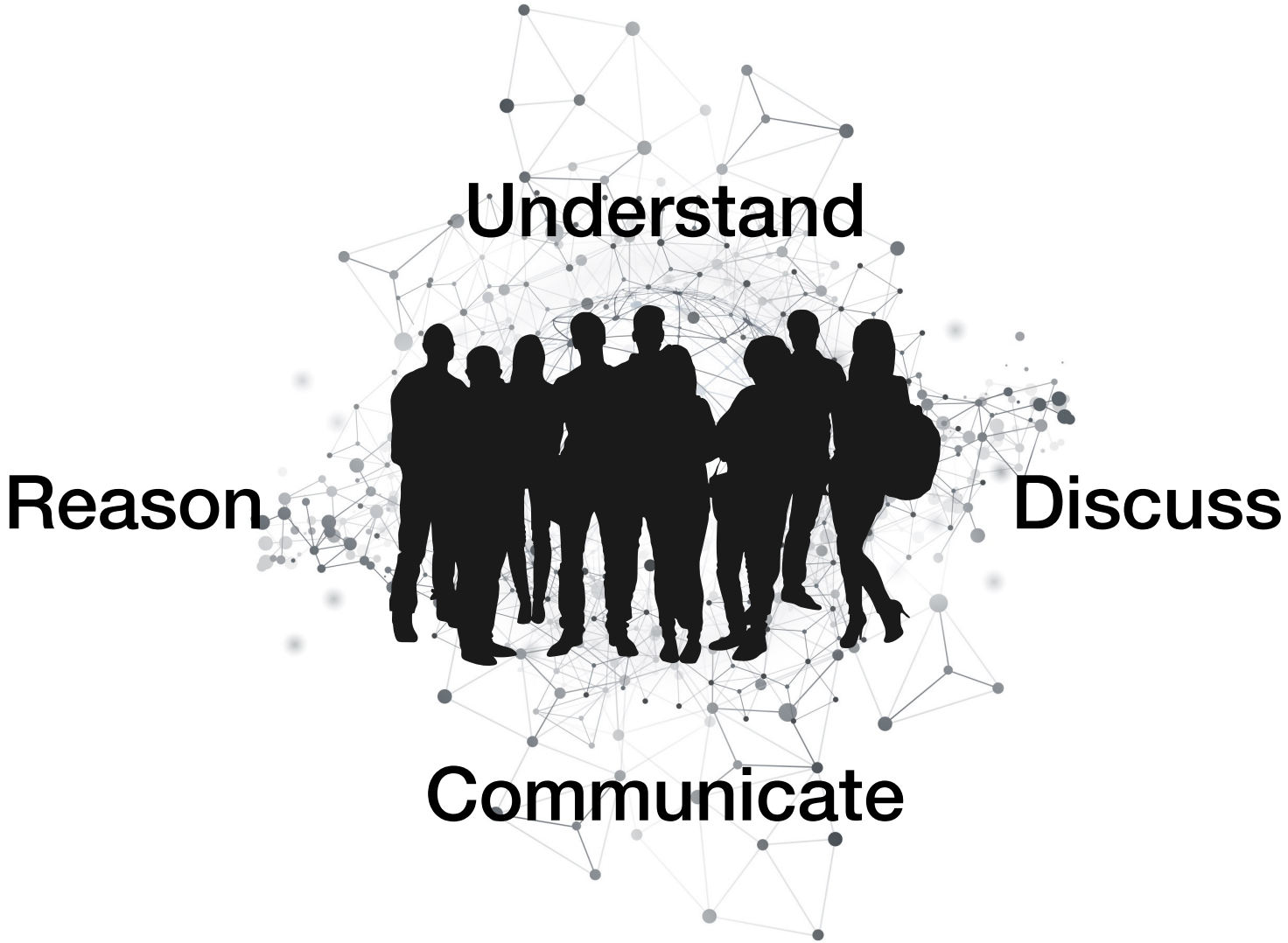
Architecturally significant

Communication strategy in a distributed system

Identification and Authentification mechanism for security

Not architecturally significant (with exceptions of course)

Choice of data structure or type
Class internal decisions
Choice of algorithm, for example, password encryption


Understand
Reason
Discuss
Communicate

Software Architecture

An abstraction that enables discussions and reasoning among architects (designers) developers about architecturally significant decisions

Example

Enable discussion

Complexity

Complexity & Decomposition

A system consists of subsystems, which contain subsystems, which can be further decomposed in subsystems …

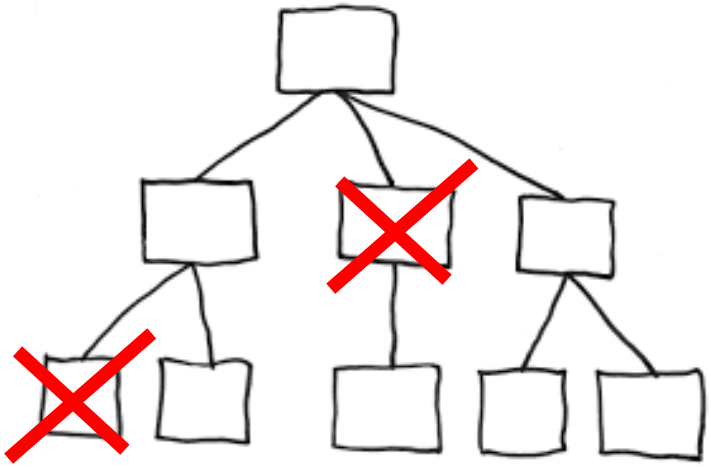
Architecture is and hierarchical abstraction

This is true for all systems, biological as well as artificial

So problem solving is all about

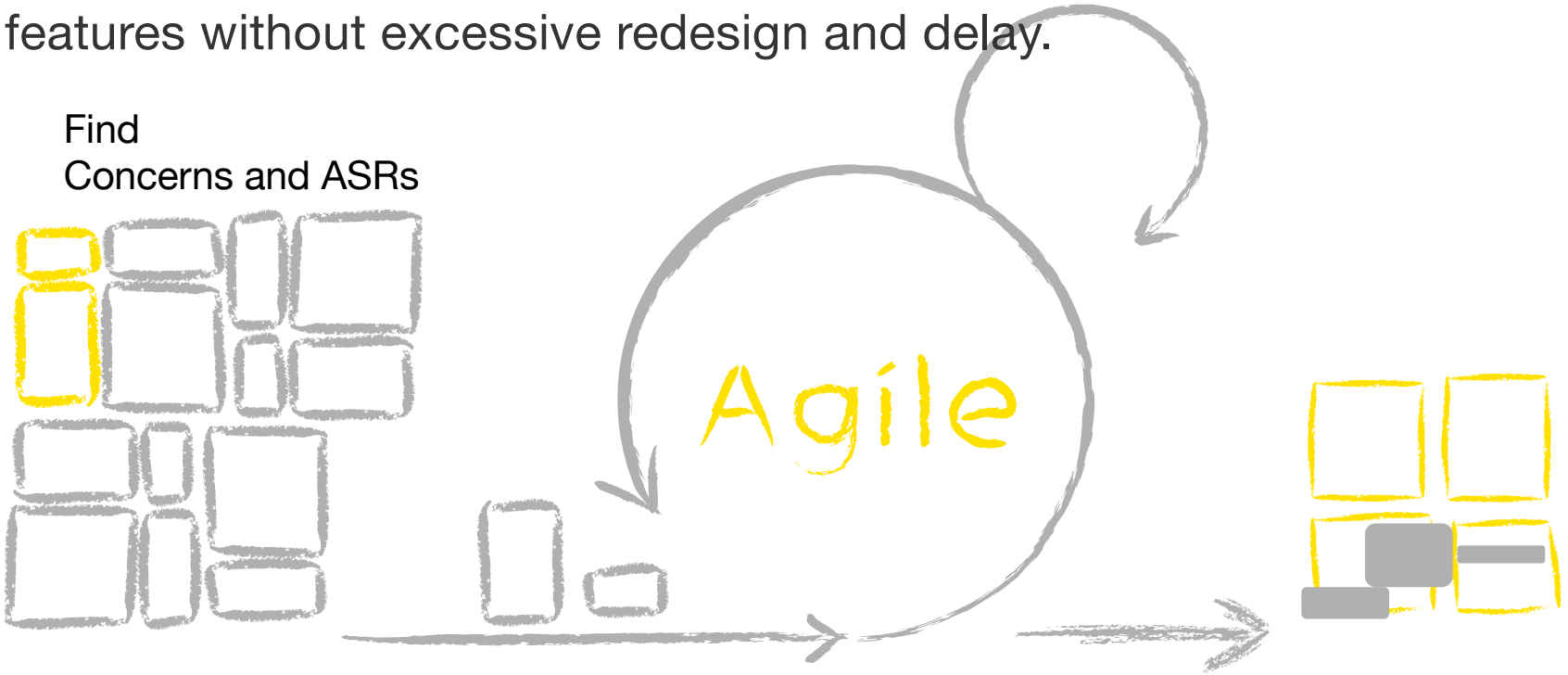
1. Finding the best “pieces”
2. Join them together

Will the parts fit function together?



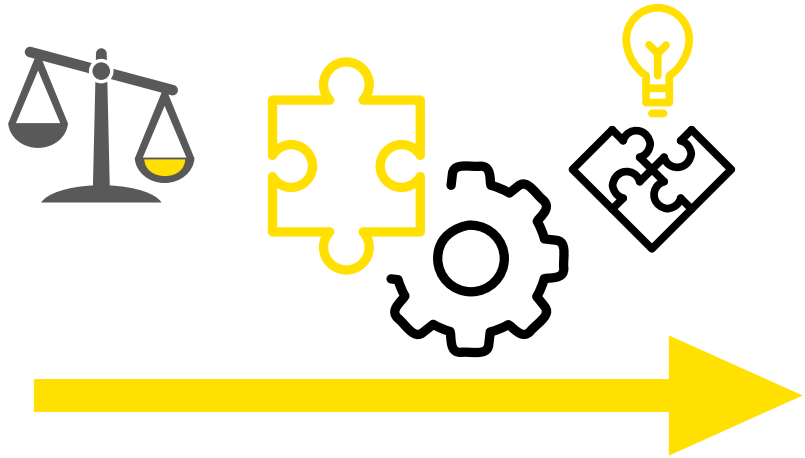
Iterative & Incremental?

Challening to work iteratively and incrementally with the existing code, components, and technical infrastructure needed to implement near-term features without excessive redesign and delay.



Motivation

Architecture Design



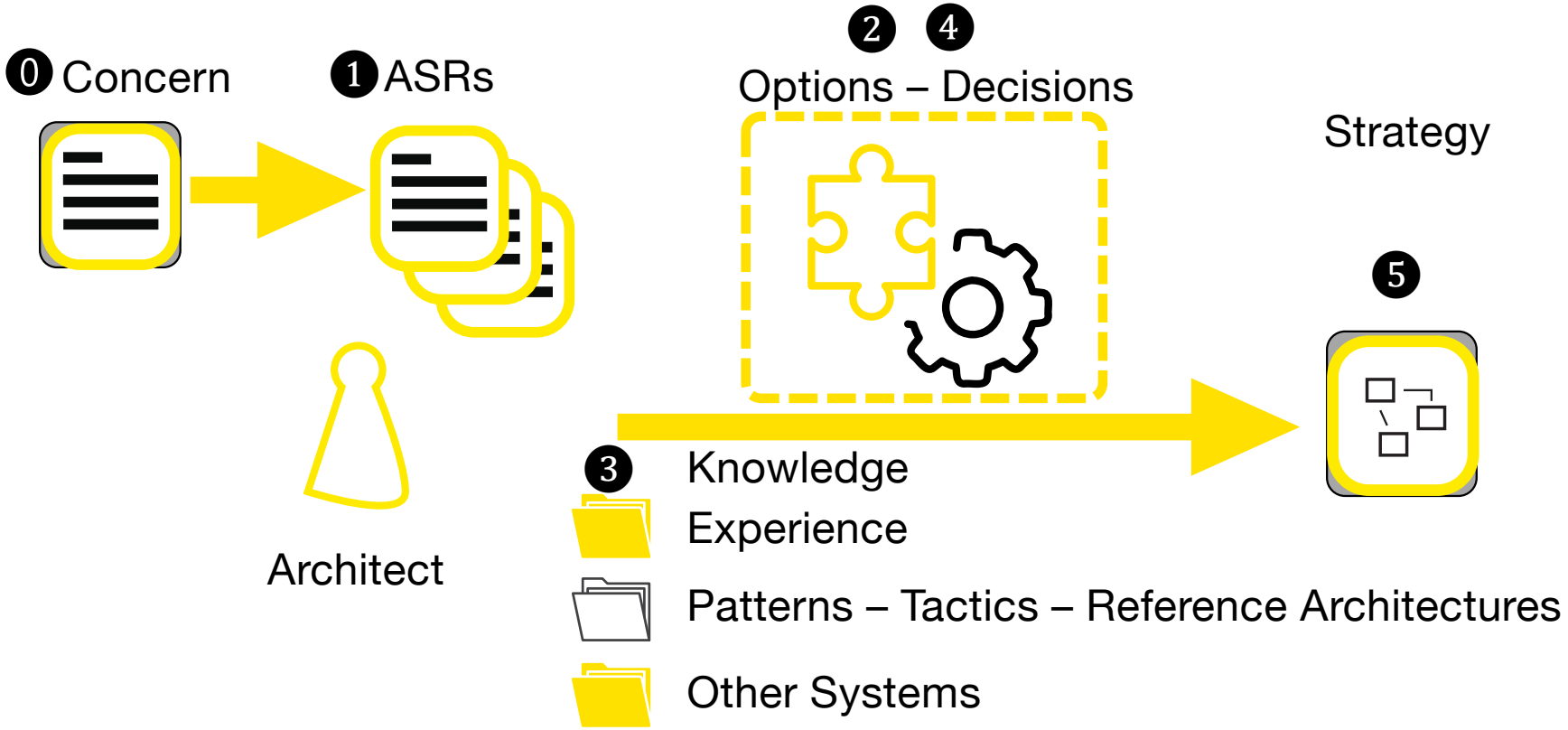
The Role of Reasoning and Integrity

Design integrity

Maintain the "value" of the previous design decisions when new ones are m

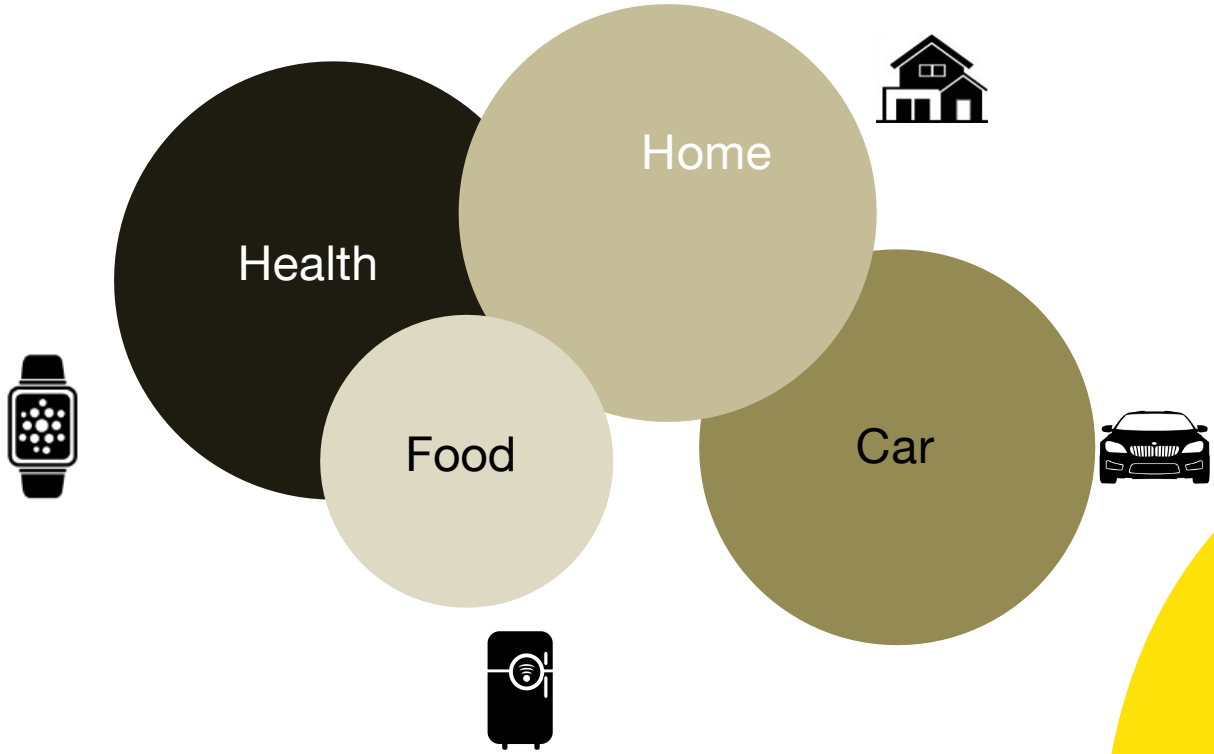
cmp. Regression testing

Architecture Design



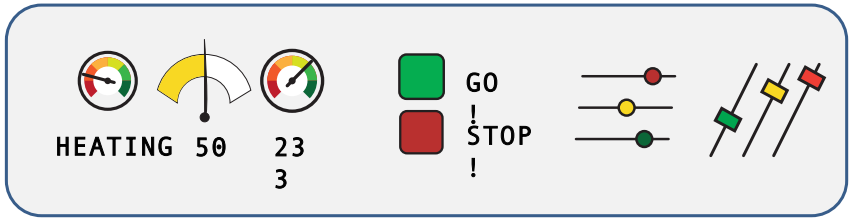
Examples adopted from the openHAB case study

A world of services
Smart homes

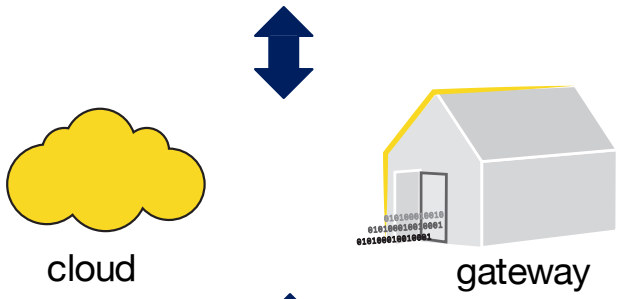


Smart Home Infrastructure

services and applications



infrastructure



things

Main Technical Challenges

provide interoperable APIs and data models for infrastructure operators, service and device providers

provide a platform for end-users supporting seamless device integration, rule-based automation and configurable UI

Architecture Design

0 Concern

interoperability

automation

configurability

security

performance

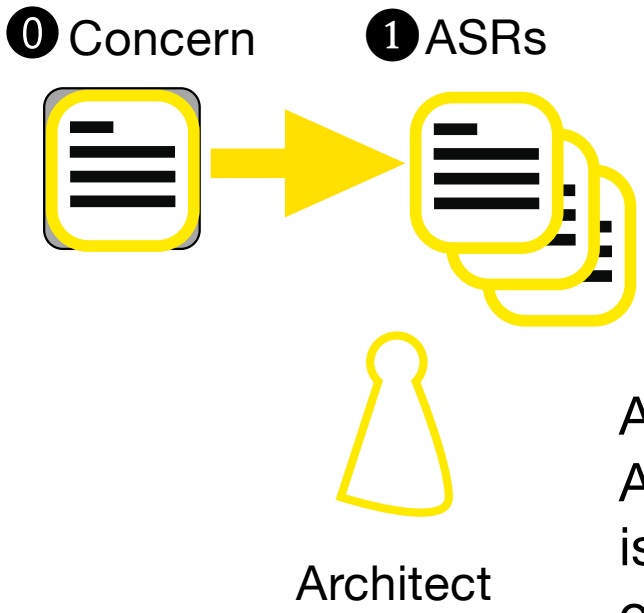
0 provide interoperable APIs and data models for infrastructure operators, service and device providers

0 provide a platform for end-users supporting seamless device integration, rule-based automation and configurable UI

Architect

Concern
A category of requirements that is important to one or more stakeholders.

Architecture Design



interoperability – asr#1, asr#2 , …

automation – asr#1, asr#2 , …

configurability – asr#1, asr#2 , …

security – asr#1, asr#2 , …

performance - – asr#1, asr#2 , …

An ASR

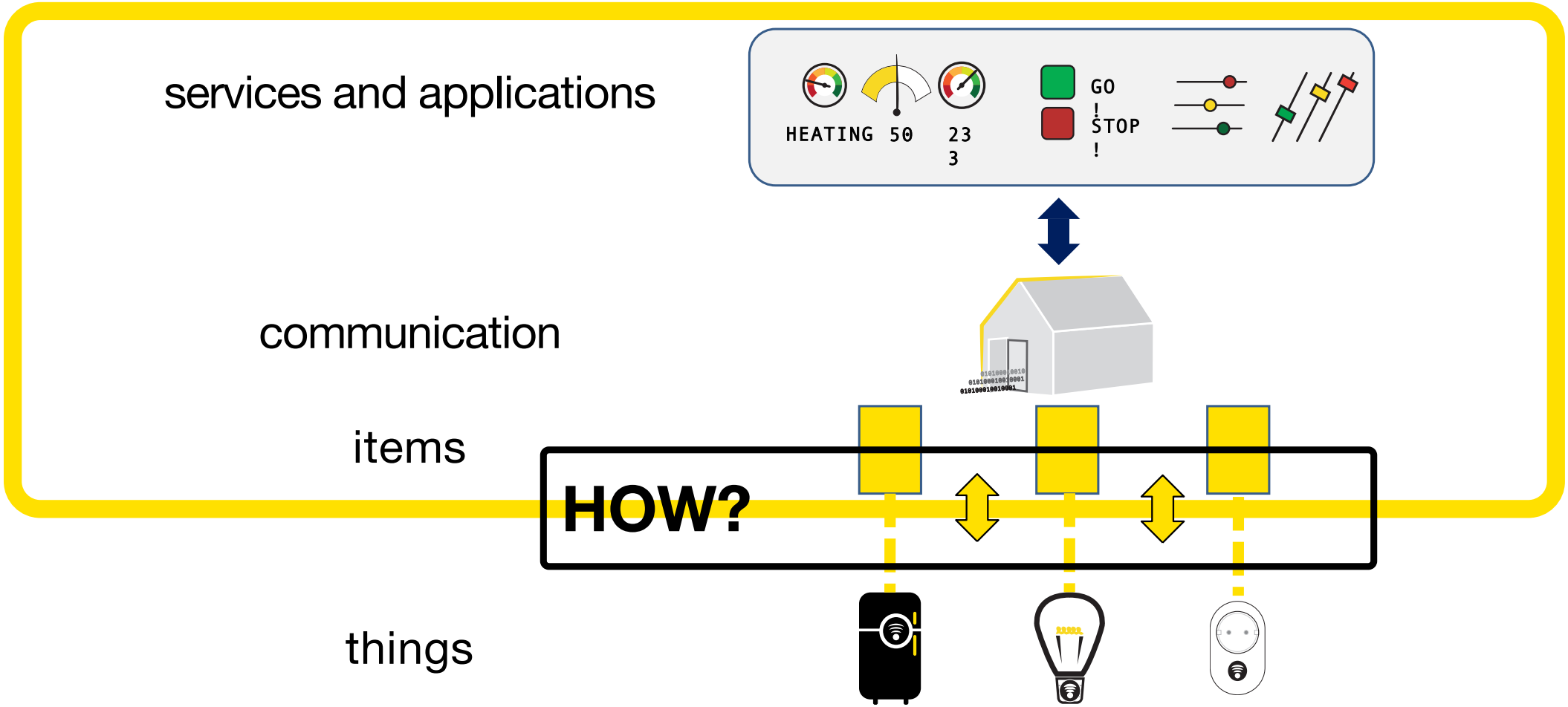
An architecturally significant requirement (ASR) is a requirement that will have a profound effect on the architecture — that is, the architecture might well be dramatically different in the absence of such a requirement.

Architecture Decisions

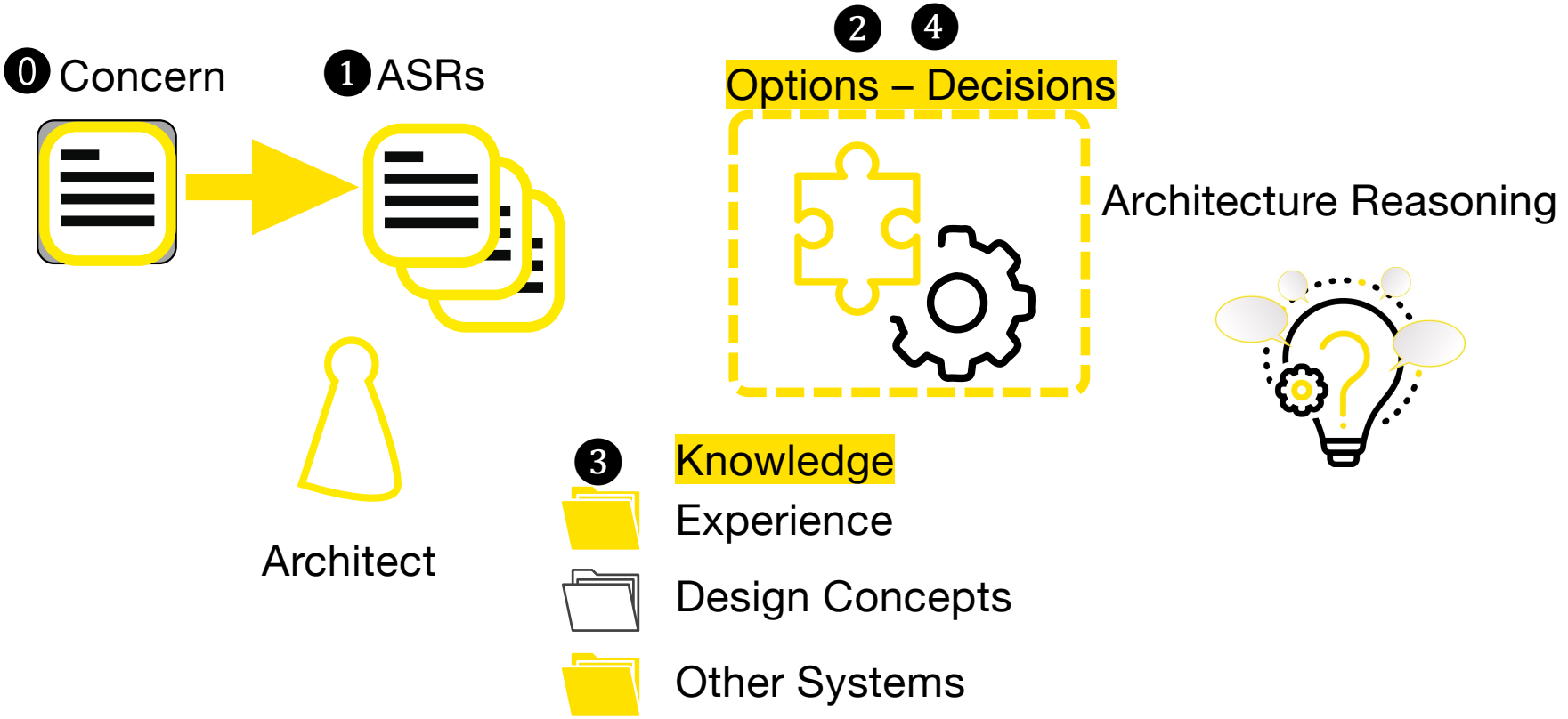
A decision that has a global impact on the system under design!

Linnæus University

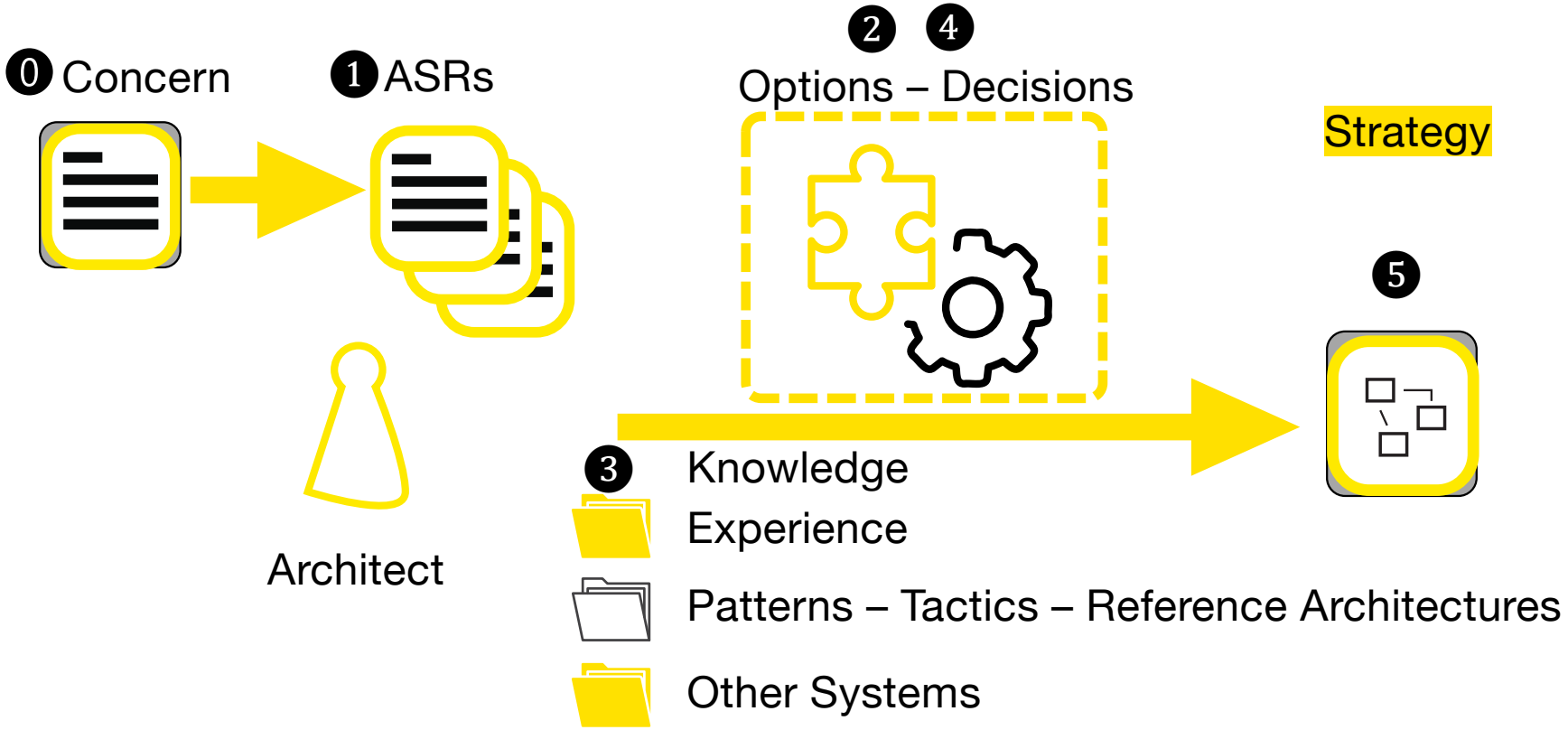
Interoperability and Configurability



Architecture Design



Architecture Design



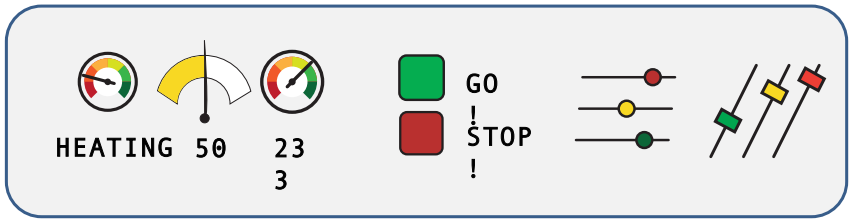
openHAB strategy (static)

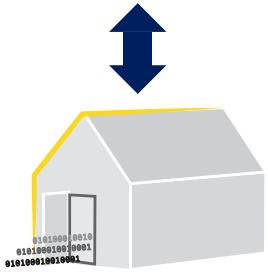
services and applications

communication

items

things



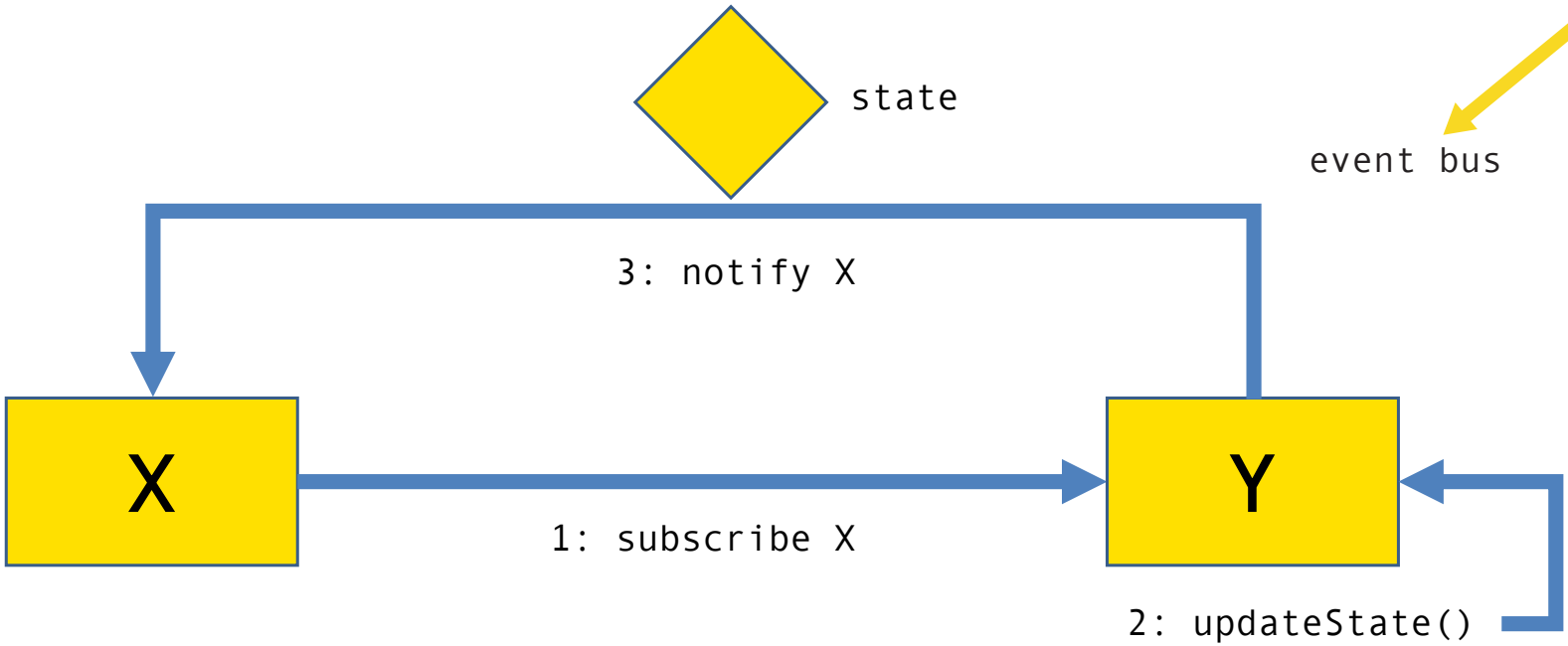


event bus

things – communication

the EventBus – a publish – subscribe implementation

asynchronous communication



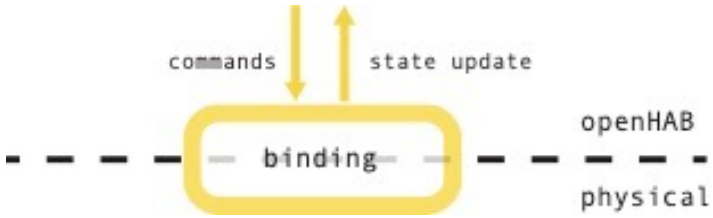
things – communication

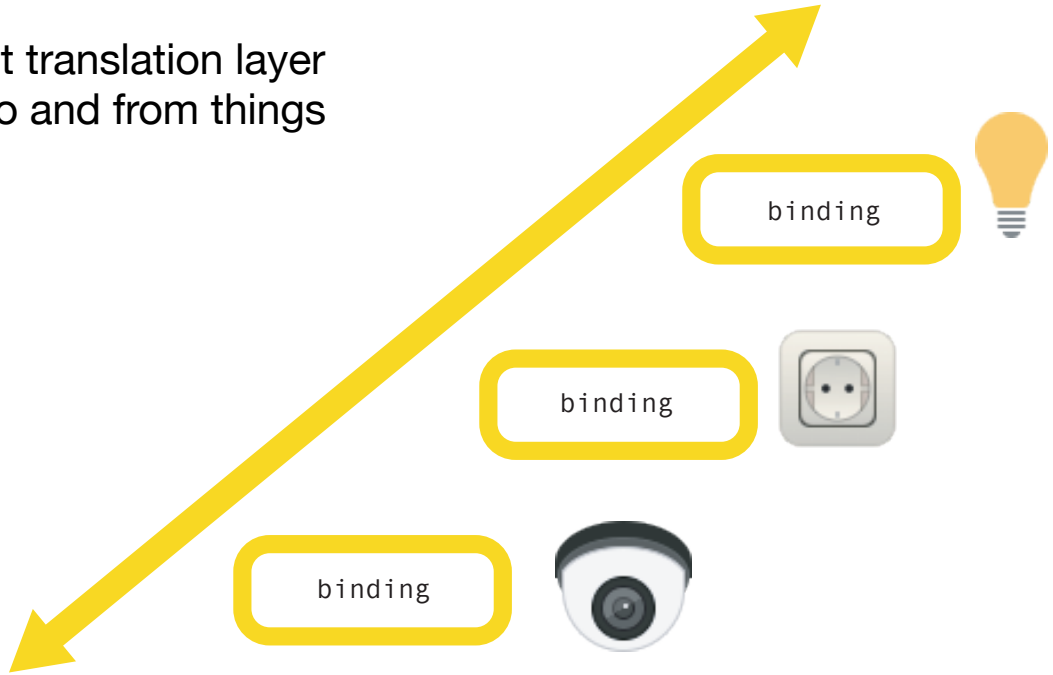
Strategy

a binding is an extension to the Eclipse SmartHome runtime
integrates an external system such as

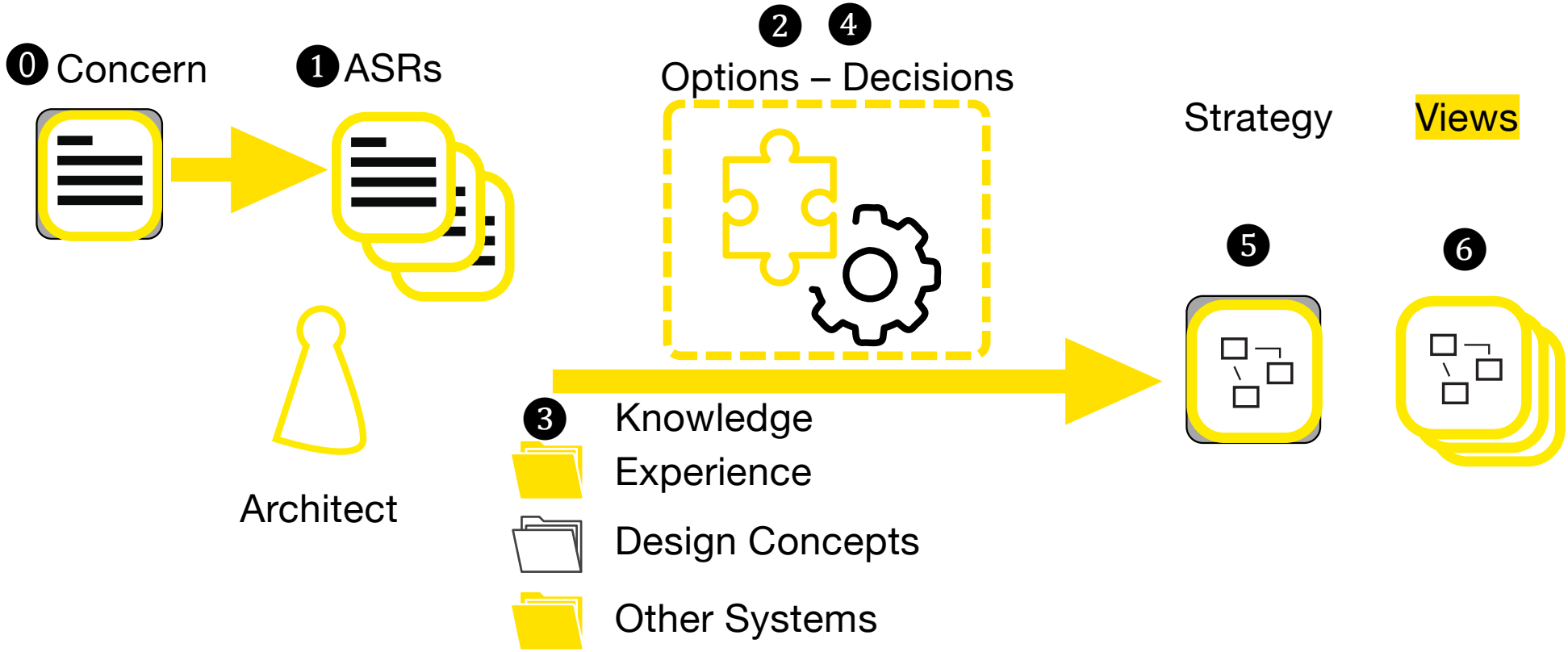
- a service,
- a protocol or
- a single device.

the OpenHAB bindings constitute an intelligent translation layer
it translates and routes Event Bus messages to and from things



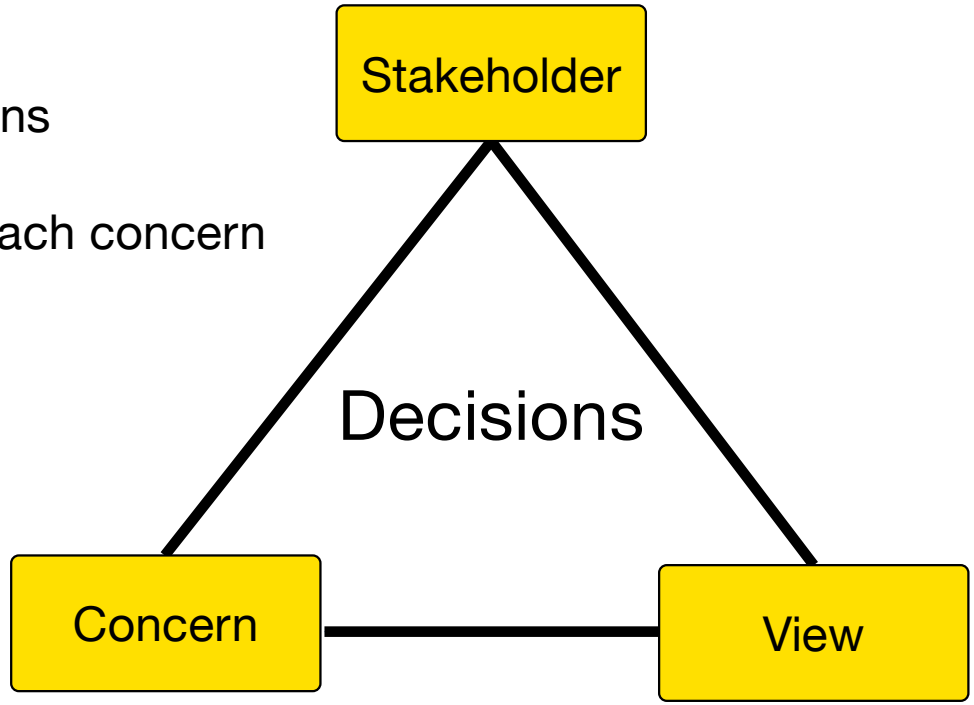


Architecture Design



Software Architecture 101

1. Identify stakeholders and their concerns
2. Specify requirements
3. Develop an architecture strategy for each concern
4. Document everything in a view



2DV604 Software Architecture

Lnu.se