EN
English

University of Oviedo

Universidad de Oviedo

School of Computer Science

Software Architecture
Basic definitions

SOFTWARE ARCHITECTURE

School of Computer Science

Course 2019/20

Jose E. Labra Gayo

Contents

What is a software Architecture?

Structure (or set of structures) of a system, which comprise:

- software elements
- relations among them
- properties of both.

High level structure of a software system

“Significant design decisions of a system”

If you have to change them $\Rightarrow$ High cost

Architecture design

Problem domain

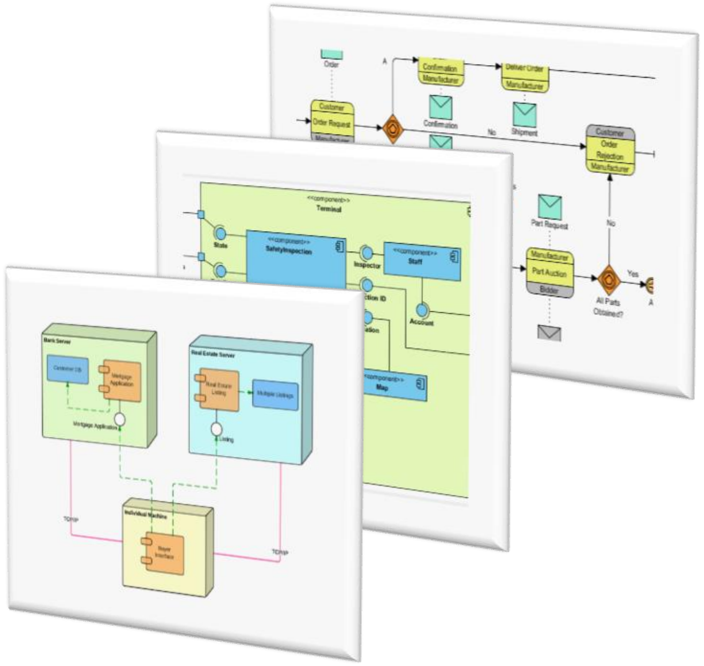
Solution domain

- Design Objectives
- Functional requirements
- Quality attributes
- Constraints
- Concerns

Architecture drivers (inputs)

Architect

Design activity



Design of the architecture (output)

Architeture drivers

- Inputs of the software architecture process
  - Design objectives
  - Functional requirements
  - Quality attributes
  - Constraints
  - Concerns

Design objectives

What are the business goals?

Why you are designing that software?

Some examples:

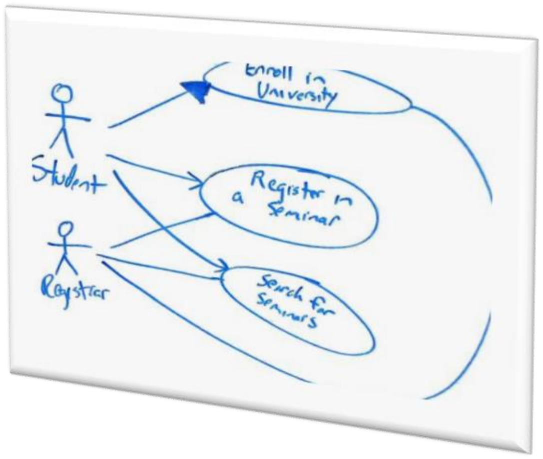
- **Pre-sales proposal**: rapid design of an initial solution in order to produce an estimate
- **Custom system** with established time and costs which may not evolve much once released
- **New increment** or release of a continuously evolving system

Functional requirements

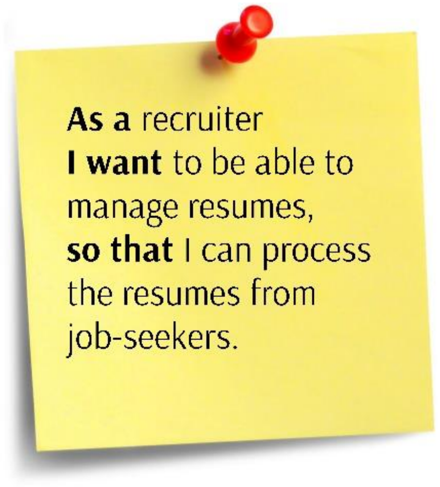
Functionality that supports the business goals

List of requirements as use cases or user stories

Use cases



User stories



As a recruiter
I want to be able to manage resumes,
so that I can process the resumes from job-seekers.

Quality attributes

Measurable features of interest to users/developers

- Also known as non-functional requirements
- Performance, availability, modifiability, testability,…
  - Also known as -ilities
- Can be specified with scenarios
  - Stimulus-response technique
    - *"If an internal failure occurs during normal operation, the system resumes operation in less than 30seconds, and no data is lost"*

- ISO 25010: list of some non-functional requirements
- List: https://en.wikipedia.org/wiki/List_of_system_quality_attributes

Quality attributes

- Quality attributes determine most architectural design decisions
  - If the only concern is functionality, a monolithic system would suffice
  - However, it is quite common to see:
    - Redundancy structures to increase **reliability**
    - Concurrency to increase **perfomance**
    - Layers for **modifiability**
    - …
- Quality attributes must be prioritized
  - By the client to consider system's success
  - By the architect to consider technical risk

Constraints

- Pre-specified design decisions
  - Very little software has total freedom
  - May be technical or organizational
  - May originate from the customer but also from the development organization
  - Usually limit the alternatives that can be considered for particular design decisions
  - Examples:
    - Frameworks, programming languages, DBMS,…
  - They can act as “friends”
    - Identifying them can avoid pointless disagreements

Concerns

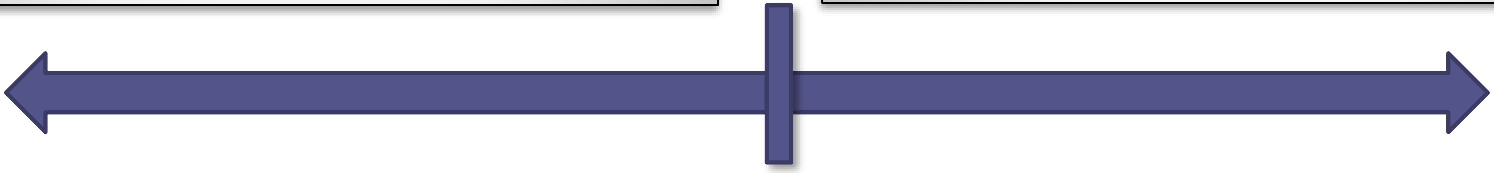
- Design decisions that should be made
  - Even if they are not stated explicitly
- Examples:
  - Input validation
  - Exception management and logging
  - Data migration and backup
  - Code styles…
  - …

Architecture as a trade-off

Between...

Creativity
Fun
Risk
Can offer new solutions
Can be unnecessary

Method
Efficient in familiar domains
Predictable result
Not always the best solution
Proven quality techniques



Architect

Types of systems

Greenfield systems in novel domains

- E.g. Google, WhatsApp,…
- Less well known domains, more innovative

Greenfield systems in mature domains

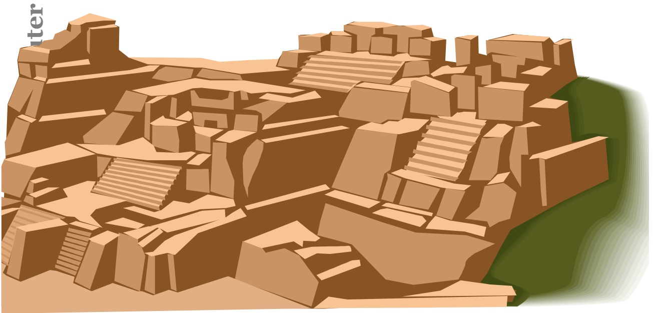
- E.g. "*traditional*" enterprise applications, standard mobile apps
- Well known domain, less innovative

Brownfield domains

- Changes to existing system

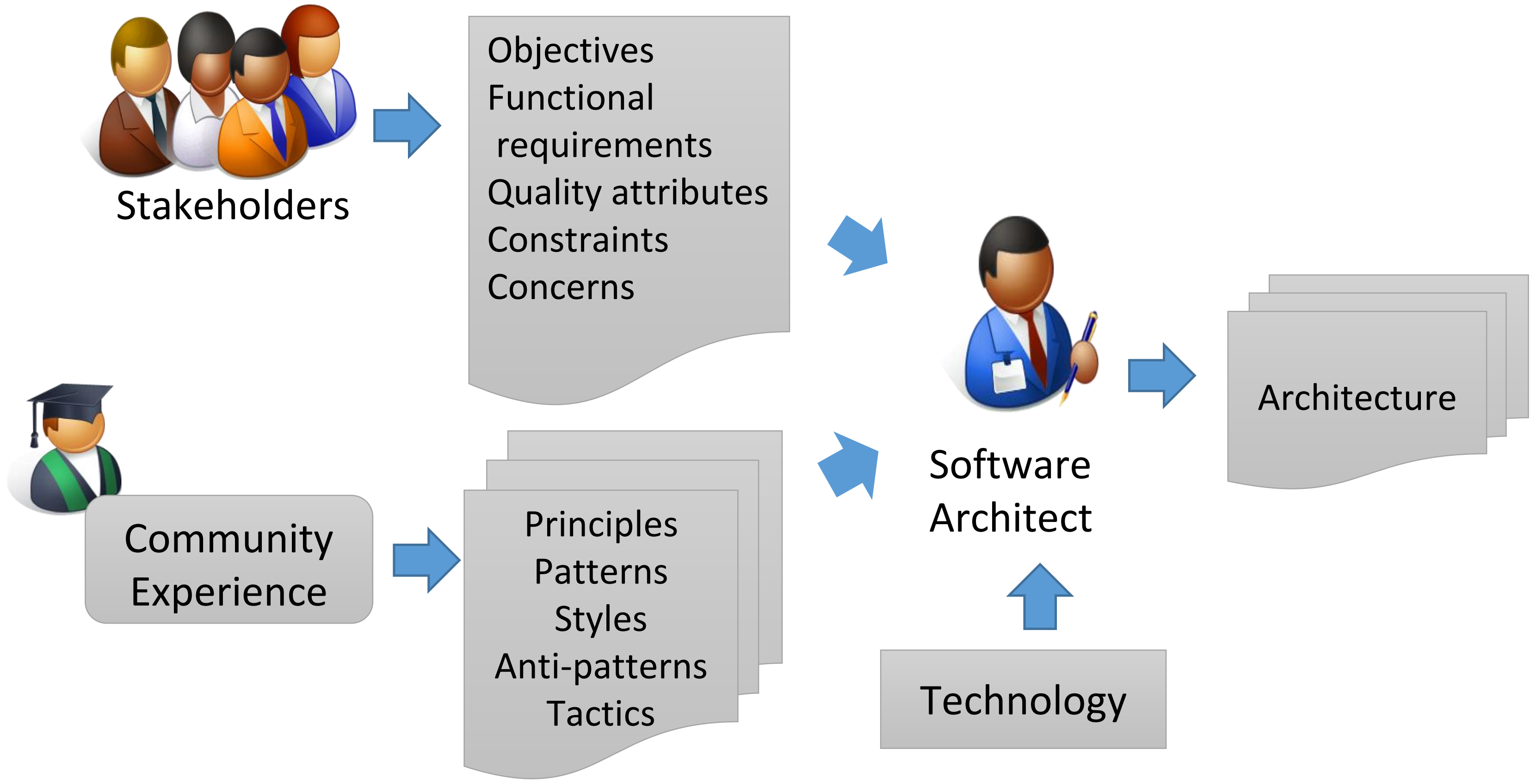
Software architect

- Discipline evolves
- Architect must be aware of
  - New development techniques
  - Styles and patterns
    - Best tool = experience (*no silver bullet*)
      - Self experience
      - Experience from community



Architect

Role of software architect


Stakeholders
Objectives
Functional
requirements
Quality attributes
Constraints
Concerns
Community
Experience
Principles
Patterns
Styles
Anti-patterns
Tactics
Software
Architect
Technology
Architecture