EN
English

Universidad de Oviedo

School of Computer Science

Software Architecture
Basic definitions

SOFTWARE ARCHITECTURE

Course 2018/2019

Jose E. Labra Gayo

Contents

What is a software Architecture?

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

High level structure of a software system

- "Main design decisions of a system"
  - If you have to change them $\Rightarrow$ High cost

Architecture design

Problem domain

Design Objectives

Functional requirements

Quality attributes

Constraints

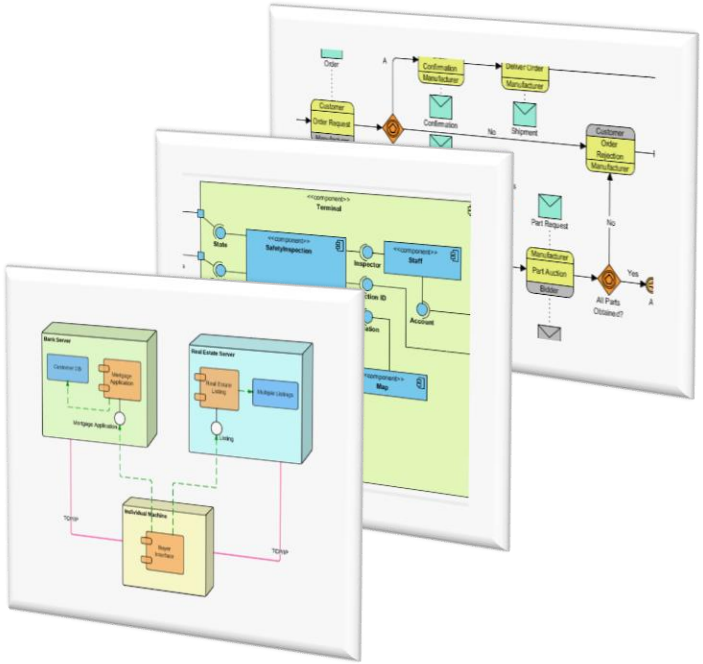
Concerns

Drivers (inputs)

Architect

Design activity

Solution domain



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?

Examples:

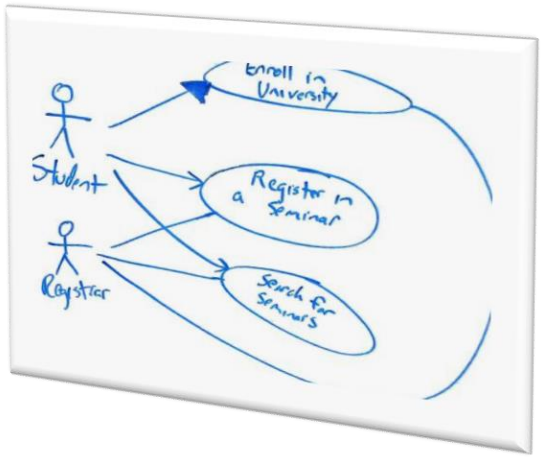
- **Pre-sales proposal**: rapid design of an initial solution in order to produce an estimate
- **Custom system** with established time and consts which may not evolve much once released
- **New increment** or release of a continuously evolving system

Functional requirements

Functionality that supports the business goals

List of requirements as use cases or user stories

Use cases



User stories

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 for reliability
    - Concurrency structures for 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 actually be your “friends”

Concerns

Design decisions that should be made whether or not they are stated explicitly as part of the goals or requirements

Examples:

- Input validation
- Exception management and logging
- Data migration and backup
- Code styles…
- …

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

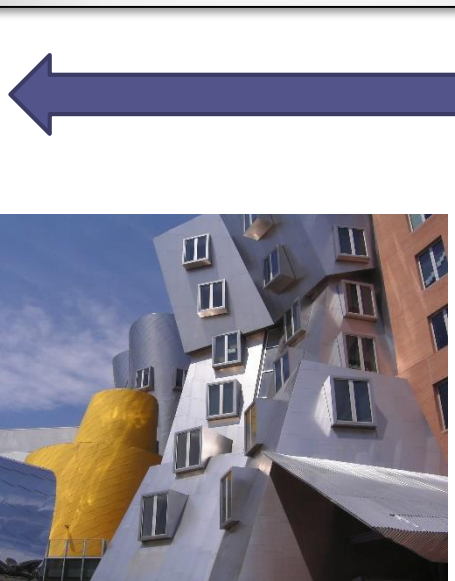
Brownfield domains

- Changes to existing system

Architecture as a trade-off

Between...

Creativity	Method
Fun	Efficient in familiar domains
Risk	Predictable result
Can offer new solutions	Not always the best solution
Can be unnecessary	Proven quality techniques



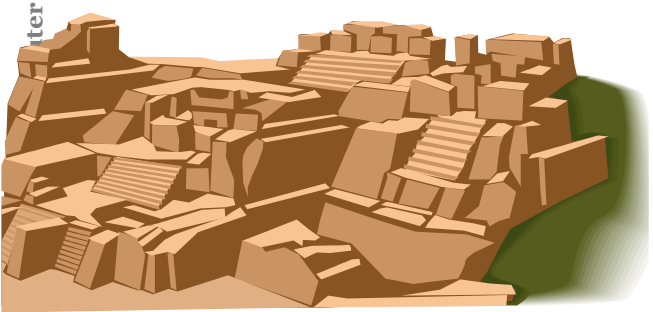
Architect

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

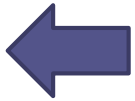
Software Architect

Technology

Architecture

Software architecture documents

Several possibilities

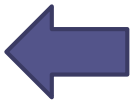
- Arc42 templates 
- C4 model
- . . .

Arc42: https://arc42.org/

Structure to document software systems

Goal: Clear, simple and effective

Templates

- Word (docx)
- Asciidoc 
- Markdown
- LaTeX
- ReStructuredText
- Confluence
- . . .

Arc42 overview

1.- Introduction and goals
2.- Constraints
3.- Context & scope
4.- Solution strategy
5.- Building block view
6.- Runtime view
7.- Deployment view
8.- Crosscutting concepts
9.- Architectural decisions
10.- Quality requirements
11.- Risks and technical debt
12.-Glossary

Picture source: https://commons.wikimedia.org/wiki/File:Ficherosclasicoscatalogo.JPG

1 - Introduction and goals

Short description of:

- Requirements
- Main quality goals
- Stakeholders

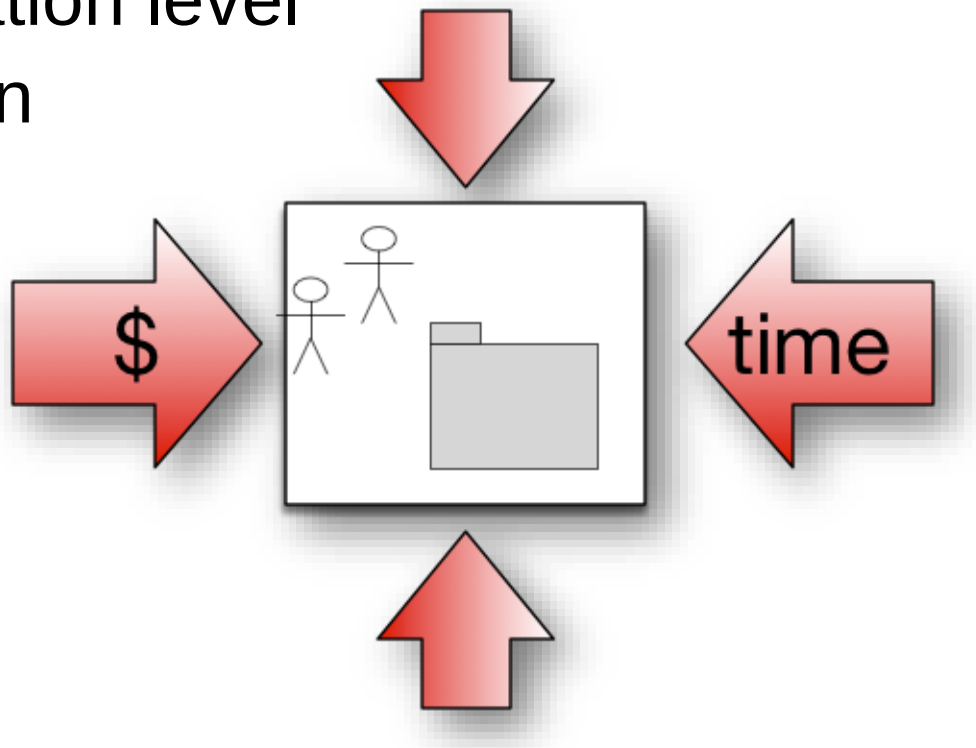
Picture source: https://arc42.org/overview/

2 - Constraints

Anything that constrains teams in design and implementation decisions

Sometimes at organization level

Decisions already taken



Picture source: https://arc42.org/overview/

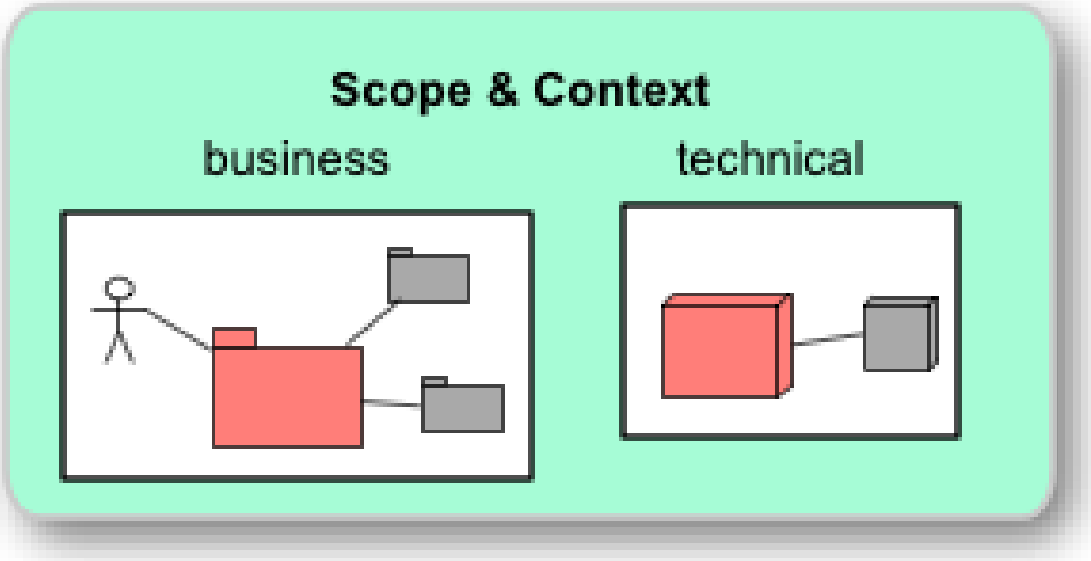
3 - Context and scope

Delimits the system from external partners

- Neighbouring users and systems

Specifies the external interfaces

Business and technical perspective



Picture source: https://arc42.org/overview/

4 - Solution strategy

Summary of fundamental decisions and strategies

Can include:

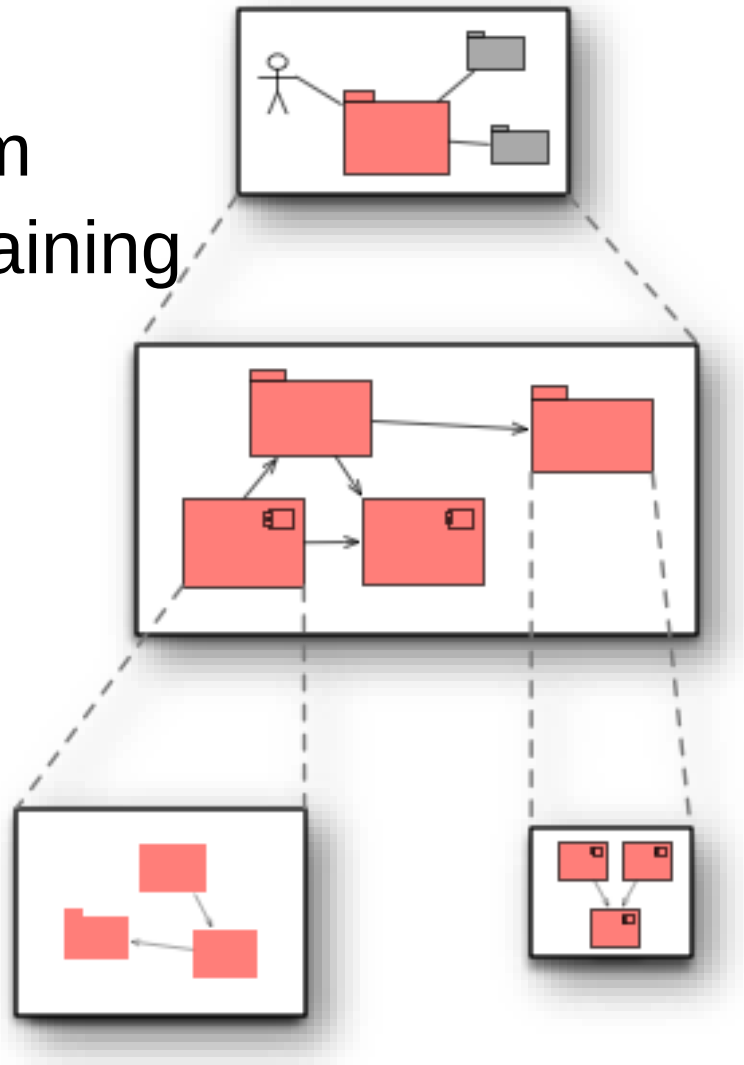
- Technology
- Top-level decomposition
- Approaches to achieve top quality goals
- Relevant organizational decisions.

Picture source: https://commons.wikimedia.org/wiki/File:Light_Bulb_or_Idea_Flat_Icon_Vector.svg

5 - Bulding block view

Static decomposition of system

Hierarchy of white boxes containing black boxes



Source: https://arc42.org/overview/

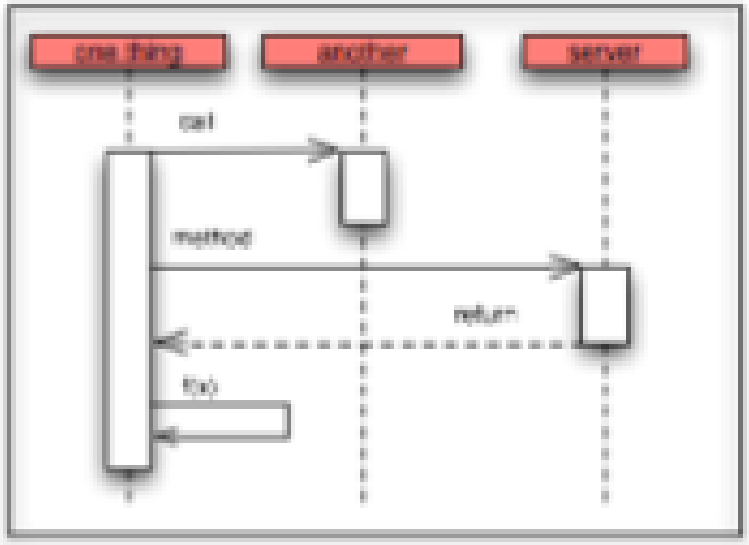
6 - Runtime view

Behavior of building blocks as scenarios

Important use cases or features

Interactions at critical external interfaces

- Error and exception behavior.

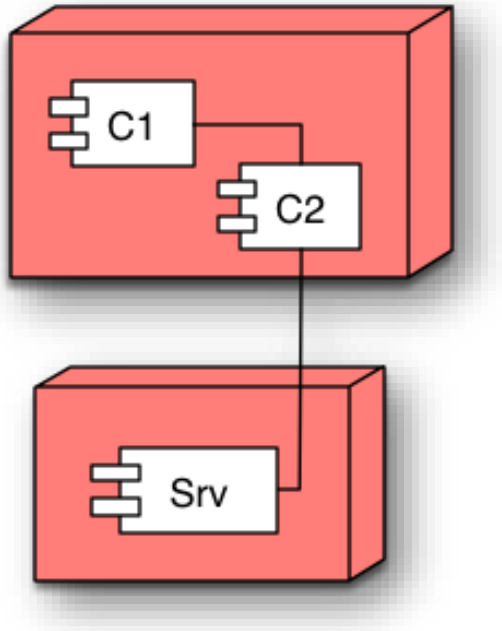


Source: https://arc42.org/overview/

7 - Deployment view

Technical infrastructure with environments, computers, processors, topologies.

Mapping of (software) building blocks to infrastructure elements.



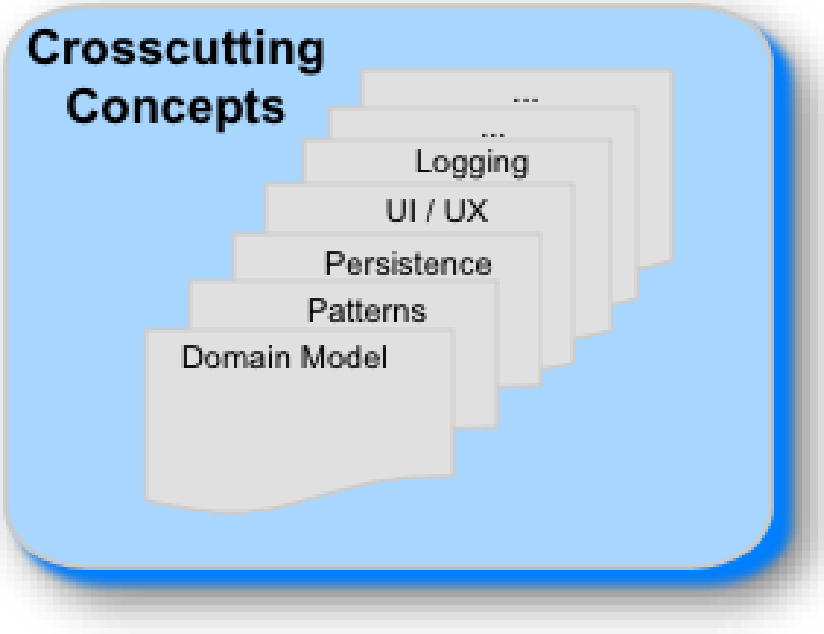
Source: https://arc42.org/overview/

8 - Crosscutting concepts

Approaches relevant in multiple parts of system

Topics like:

- Domain model
- Architecture pattern and styles
- Specific rules

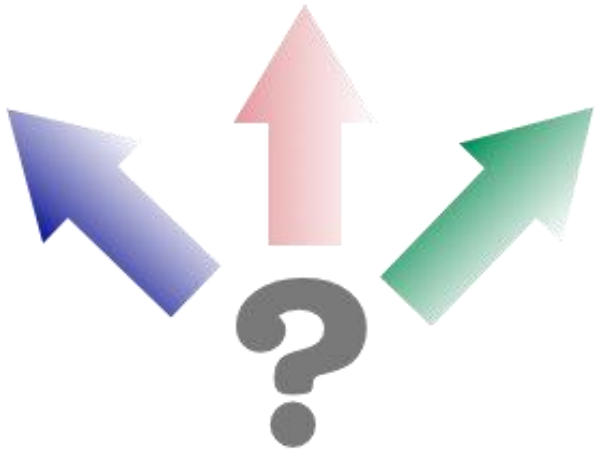


Source: https://arc42.org/overview/

9 Architectural decisions

Important, expensive, critical, large scale or risky architecture decisions

Includes rationales for the decisions

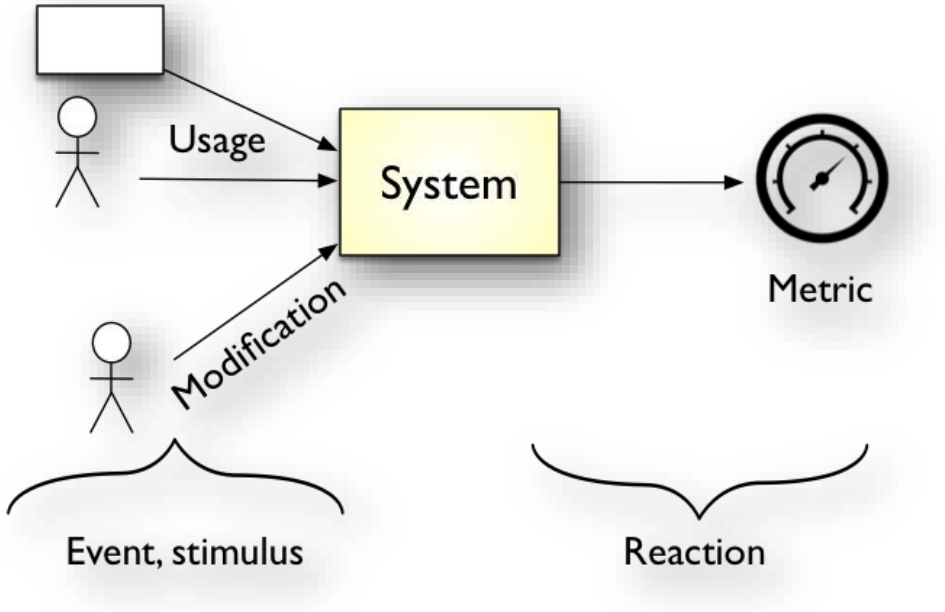


Source: https://arc42.org/overview/

10 - Quality requirements

Quality requirements as scenarios

- Quality tree to provide high-level overview
- The most important quality goals should have been described in section 1 (quality goals)



Source: https://arc42.org/overview/

11 - Risks and technical debt

Known technical risks or technical debt

- What potential problems exist?
- What does the development team feel miserable about?

Source: https://arc42.org/overview/

12 - Glossary

Important domain and technical terms

- Terms used by stakeholders when discussing the system
- Translation reference in multi-language environments

Term	Definition
aarkward	....
churizzo	.....
domoklesian	....
growidarian	.....
klicktilazation	....

Source: https://arc42.org/overview/