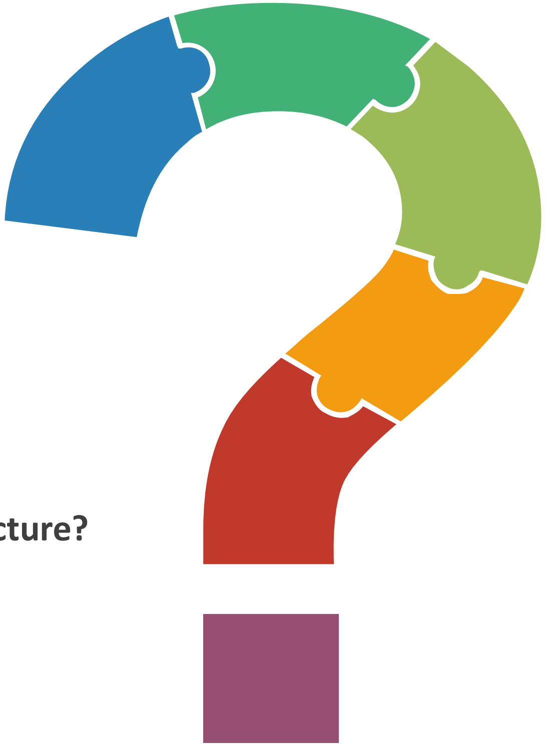
An introduction to architectural modeling

Software Engineering

Jose María Álvarez Rodríguez

First questions

- What is an architecture?
- Any example of architecture?
- What is a software architecture?



- Who affects a software architecture?
- What is the role of a software architect?
- How a software architecture is described and documented?

Notion of architecture (Oxford dictionary)

1. The art or practice of designing and constructing buildings.
2. The style in which a building is designed and constructed, especially with regard to a specific period, place, or culture.
3. The complex or carefully designed structure of something.

Domains:

- Civil architecture
- Military architecture
- Hydraulic architecture
- Naval architecture
- Sacred architecture
- …

Software Engineering-Systems and Software architecture

Architecture and architects

What is the role of an architect?

Design an infrastructure

- Needs
- Legal framework
- Security, economical, etc. restrictions
- Art, Science and Engineering

Stakeholders

- Person, group or entity
- User, client or affected by

Holistic/complete view

- All elements
- All relationships
- All interactions

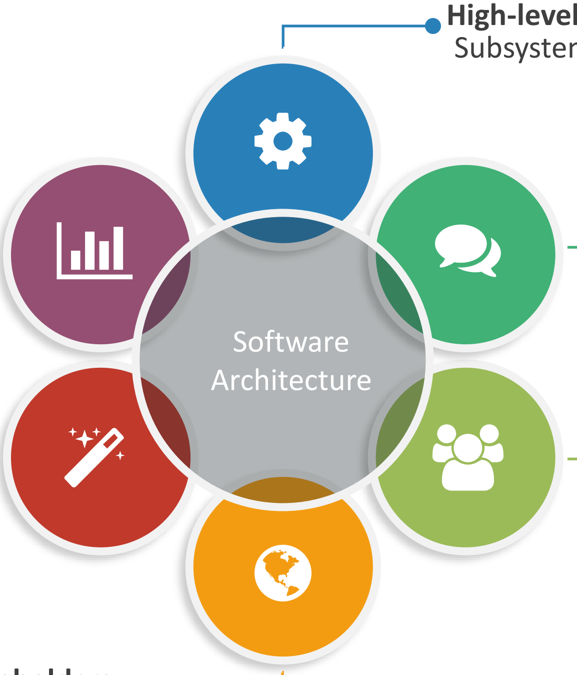
Use cases

- Exploitation
- Enjoy
- …

Definition of software architecture

*"<system> **fundamental concepts** or **properties** of a system in **its environment** embodied in its **elements**, **relationships**, and in the **principles** of its **design** and **evolution**."*

Source: ISO/IEC/IEEE 42010:2011 Systems and software engineering —Architecture description



High-level view of an infrastructure
Subsystems and components

Relationships
Structural (static)
Interactions (Dynamic)

Provision
Products and services to a domain

Stakeholders
Meet needs

Use cases
Reach goals

Restrictions
Quality, legal, economical, etc..

Complexity Management

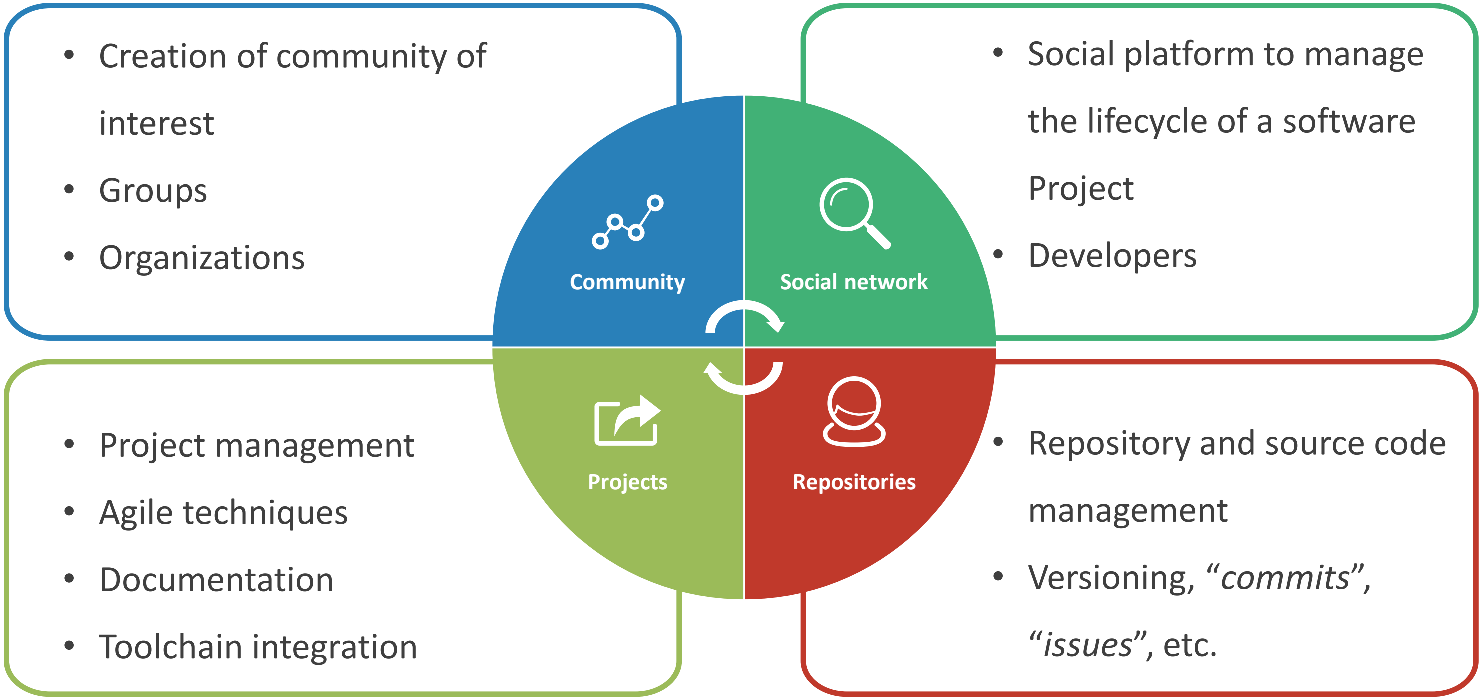
A complex software system: Smart cars

Who are the stakeholders?

Which are their goals?

How goals are reached?

On-going example: Github



- Creation of community of interest
- Groups
- Organizations

- Social platform to manage the lifecycle of a software Project
- Developers

- Project management
- Agile techniques
- Documentation
- Toolchain integration

- Repository and source code management
- Versioning, “*commits*”, “*issues*”, etc.

Learn more: https://github.com/

A software architecture as a System of Systems

System of Systems (SoS)

A system of systems (SoS) brings together a set of systems for a task that none of the systems can accomplish on its own. Each constituent system keeps its own management, goals, and resources while coordinating within the SoS and adapting to meet SoS goals.

Fuente: ISO 15288:2015

Data management
Representation, storage and access.

Social network
Relationships

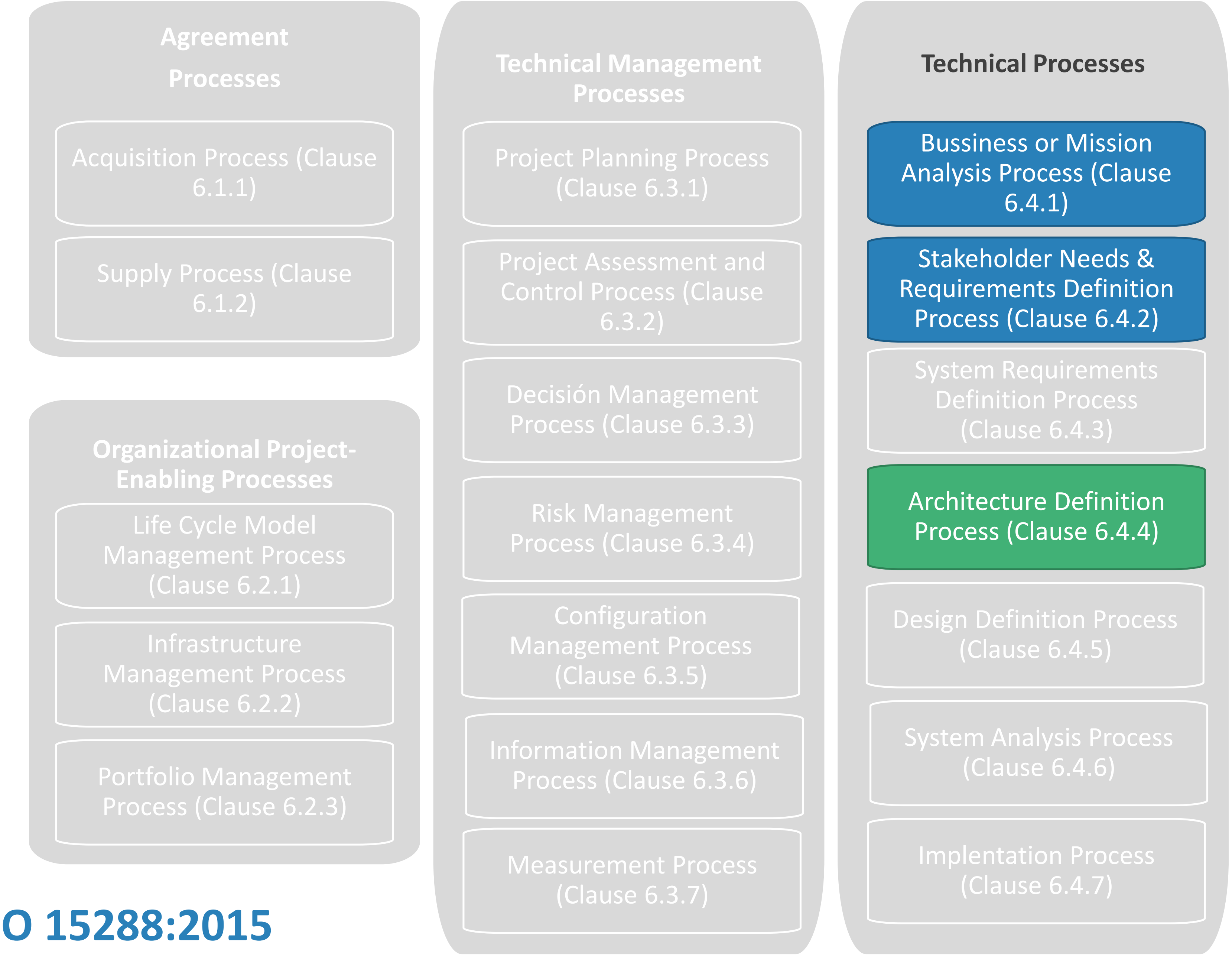
Repository management
Git protocl

Software distribution
Release management

Authentication
User management

Graphical user interface
Depending on functionality and end-user

System Life Cycle Processes



Agreement Processes

- Acquisition Process (Clause 6.1.1)
- Supply Process (Clause 6.1.2)

Organizational Project-Enabling Processes

- Life Cycle Model Management Process (Clause 6.2.1)
- Infrastructure Management Process (Clause 6.2.2)
- Portfolio Management Process (Clause 6.2.3)

Technical Management Processes

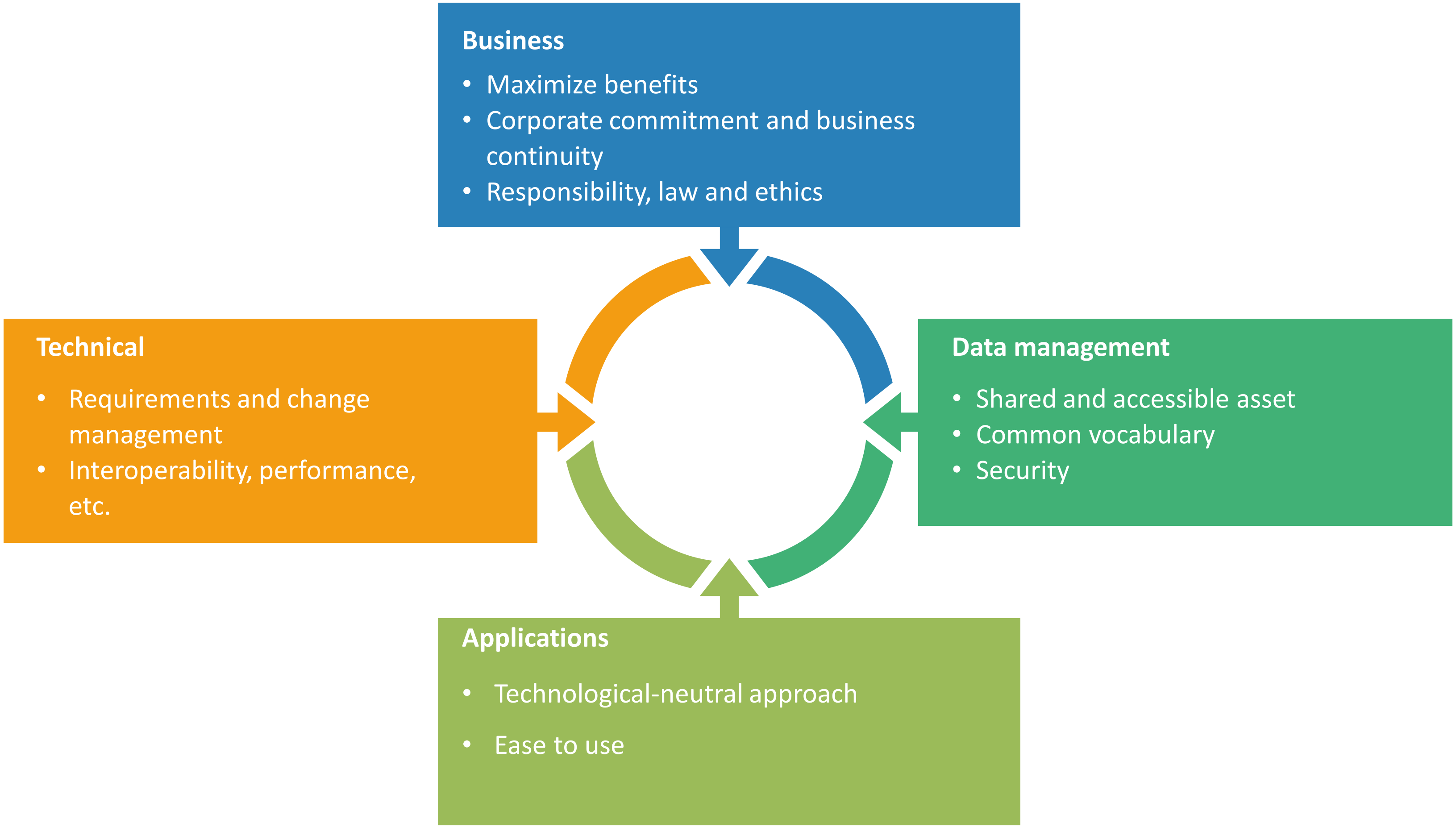
- Project Planning Process (Clause 6.3.1)
- Project Assessment and Control Process (Clause 6.3.2)
- Decisión Management Process (Clause 6.3.3)
- Risk Management Process (Clause 6.3.4)
- Configuration Management Process (Clause 6.3.5)
- Information Management Process (Clause 6.3.6)
- Measurement Process (Clause 6.3.7)

Technical Processes

- Bussiness or Mission Analysis Process (Clause 6.4.1)
- Stakeholder Needs & Requirements Definition Process (Clause 6.4.2)
- System Requirements Definition Process (Clause 6.4.3)
- Architecture Definition Process (Clause 6.4.4)
- Design Definition Process (Clause 6.4.5)
- System Analysis Process (Clause 6.4.6)
- Implentation Process (Clause 6.4.7)

ISO 15288:2015

Principles of a software architecture



Business

- Maximize benefits
- Corporate commitment and business continuity
- Responsibility, law and ethics

Technical

- Requirements and change management
- Interoperability, performance, etc.

Data management

- Shared and accessible asset
- Common vocabulary
- Security

Applications

- Technological-neutral approach
- Ease to use

Definitions

Software Architecture

It is the structure or structures of the system, which comprise software elements, the externally visible properties of those elements, and the relationships among them.

Static structure

It defines its internal design-time elements and their arrangement

Dynamic structure

It defines its runtime elements and their interactions.

Quality property

It is an externally visible, nonfunctional property of a system such as performance, security, or scalability.

Architectural element

It is a fundamental piece from which a system can be considered to be constructed.

Architectural description

It is a set of products that documents an architecture in a way its stakeholders can understand and demonstrates that the architecture has met their concerns

Any software system has an architecture even if it is unknown or not documented...

See original definition in [1].

Description of a software architecture

Stakeholder

A stakeholder in a software architecture is a person, group, or entity with an interest in or concerns about the realization of the architecture.

Concern

A concern about an architecture is a requirement, an objective, an intention, or an aspiration a stakeholder has for that architecture.

View

A view is a representation of one or more structural aspects of an architecture that illustrates how the architecture addresses one or more concerns held by one or more of its stakeholders

Viewpoint

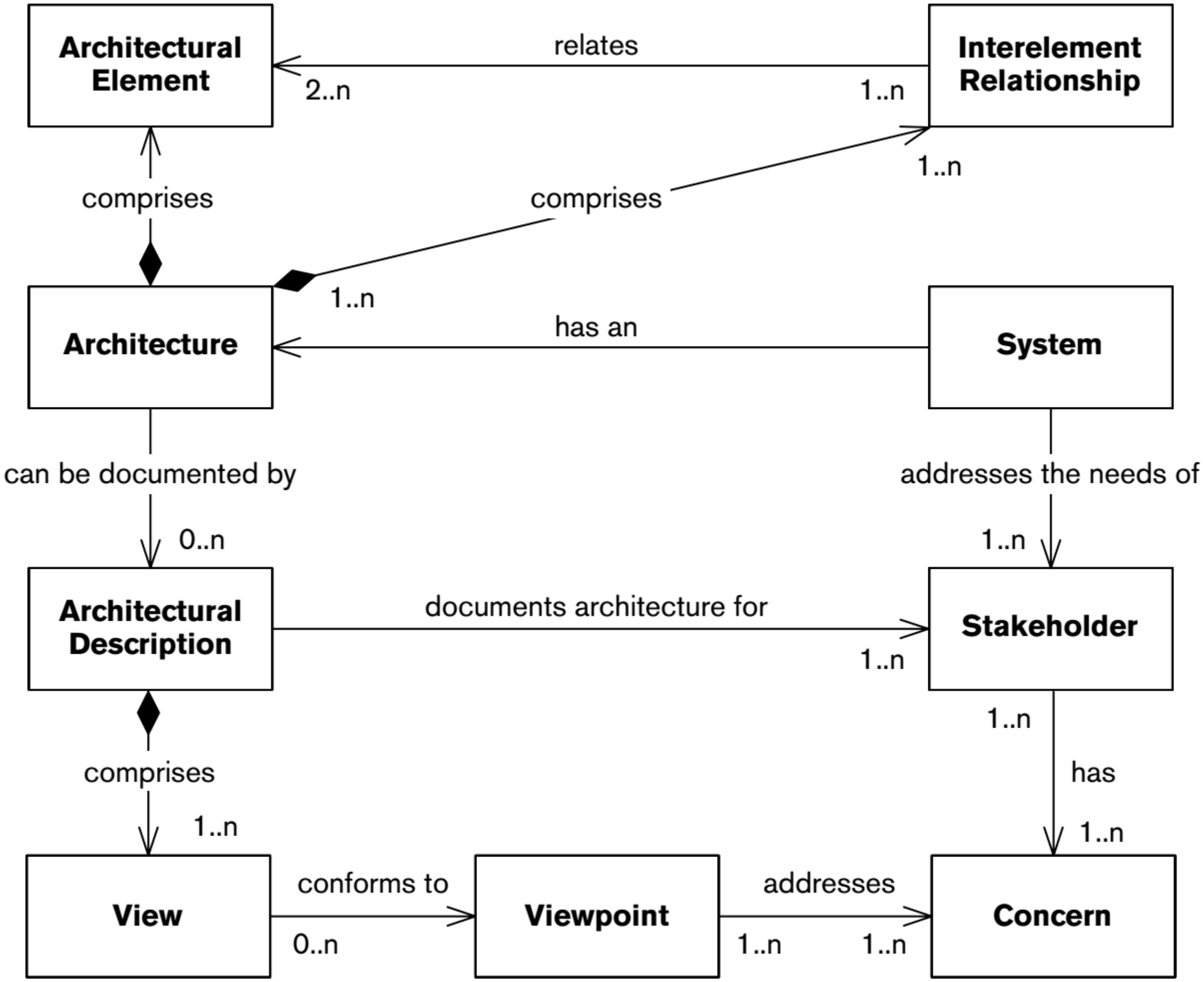
A viewpoint is a collection of patterns, templates, and conventions for constructing one type of view.

Model

An abstract representation of a process, activity o physical reality.

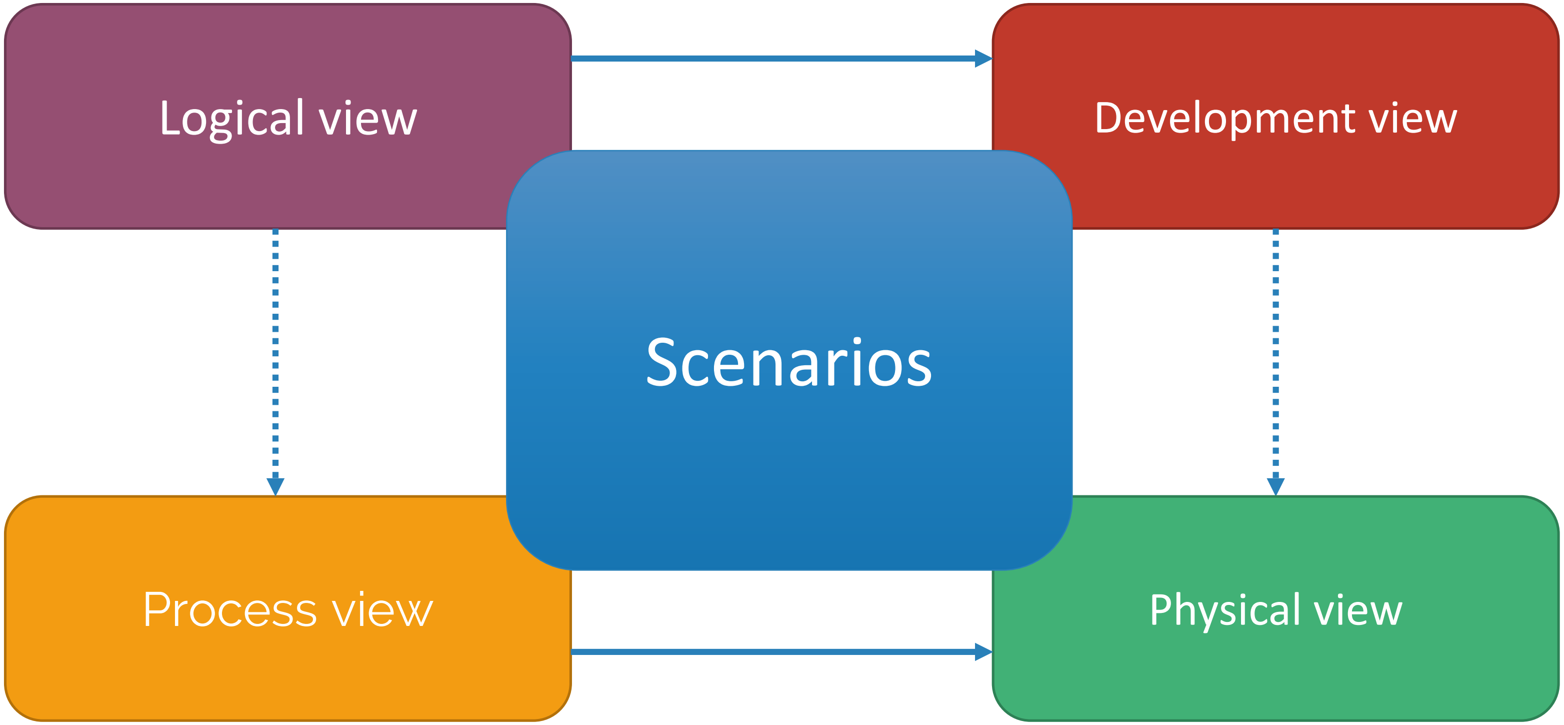
See original definitions in [1].

Description of a software architecture



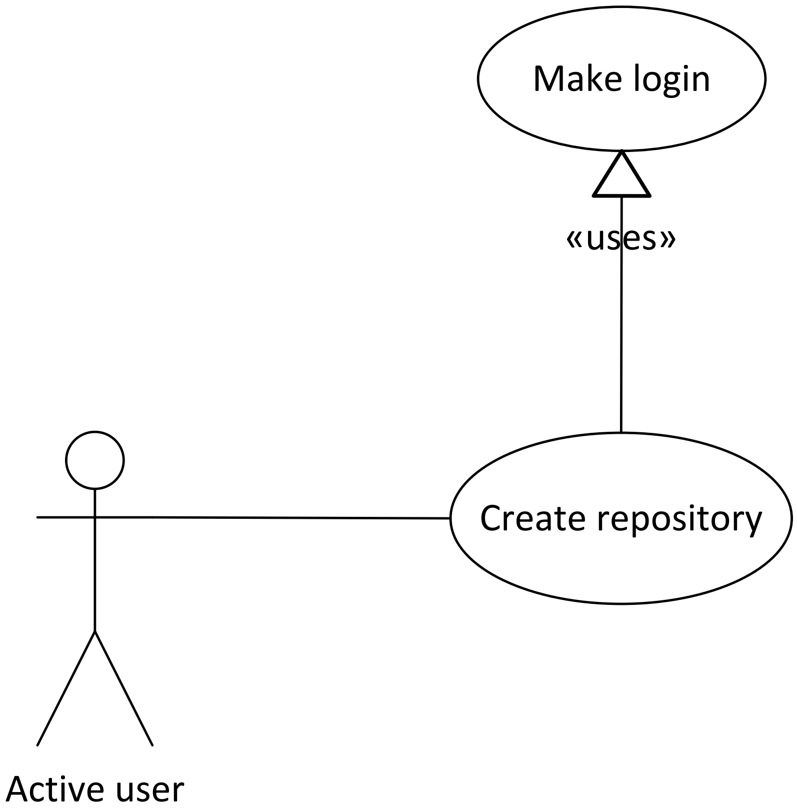
See original model in [1].

Documenting a software architecture: the 4+1 view model by Krutchen [2]

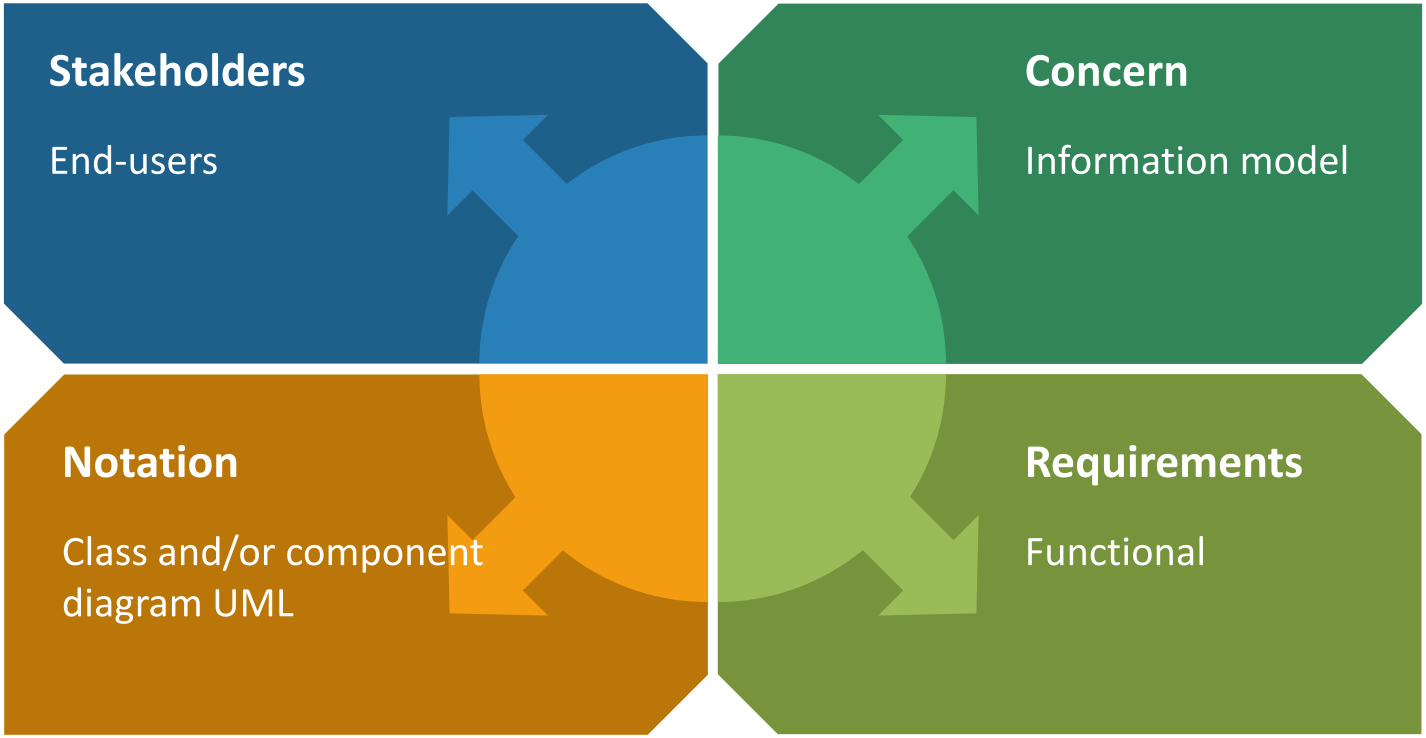


See original document in [2].

On-going example: Github scenarios



The 4+1 view model by Krutchen: logical view



Stakeholders

End-users

Concern

Information model

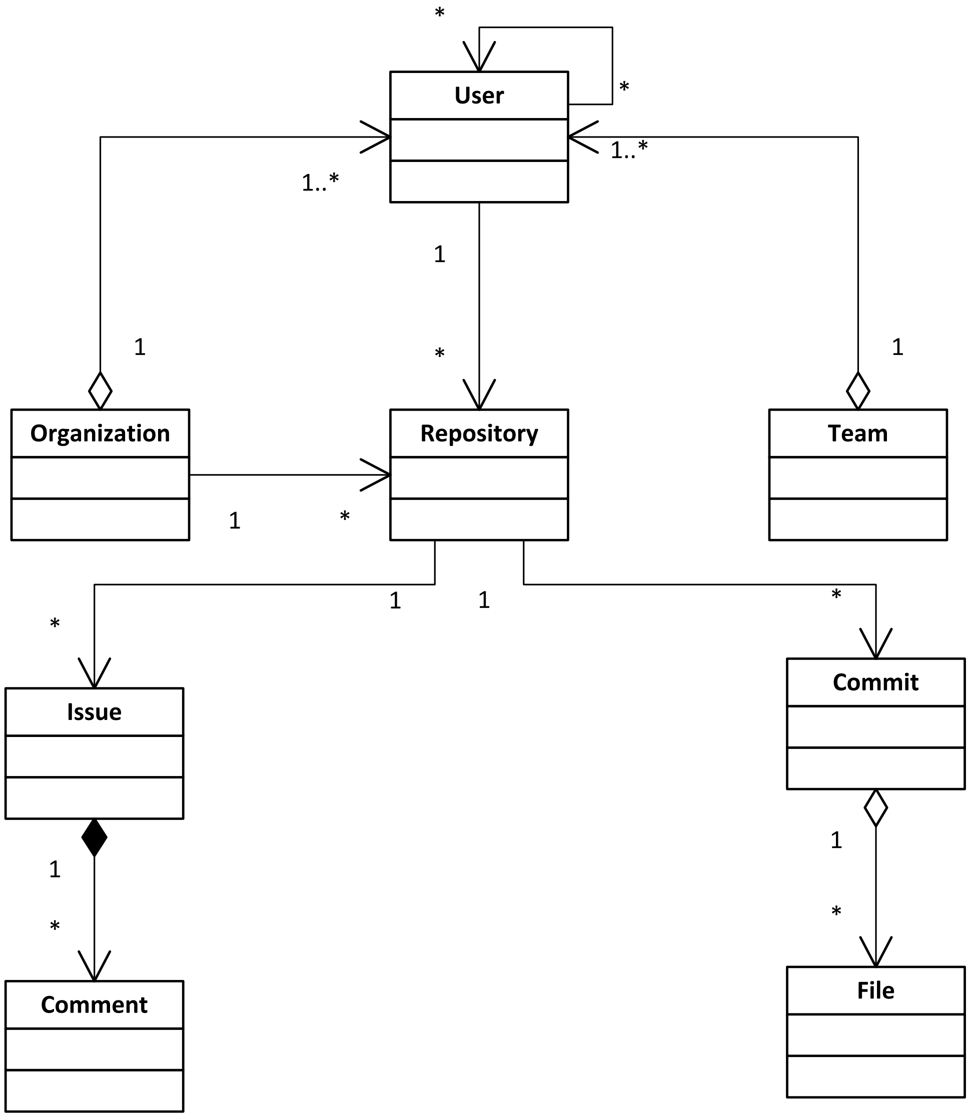
Notation

Class and/or component diagram UML

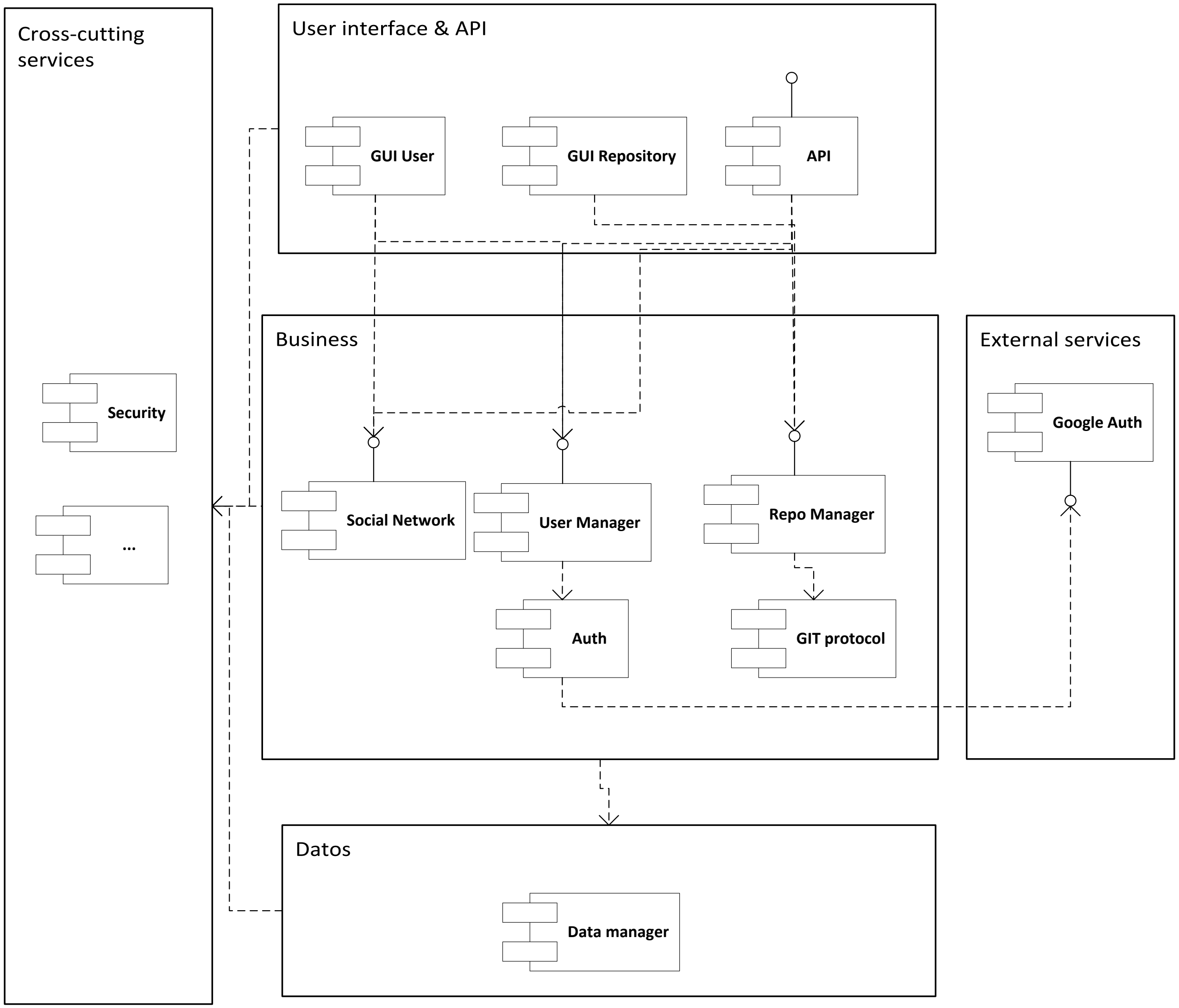
Requirements

Functional

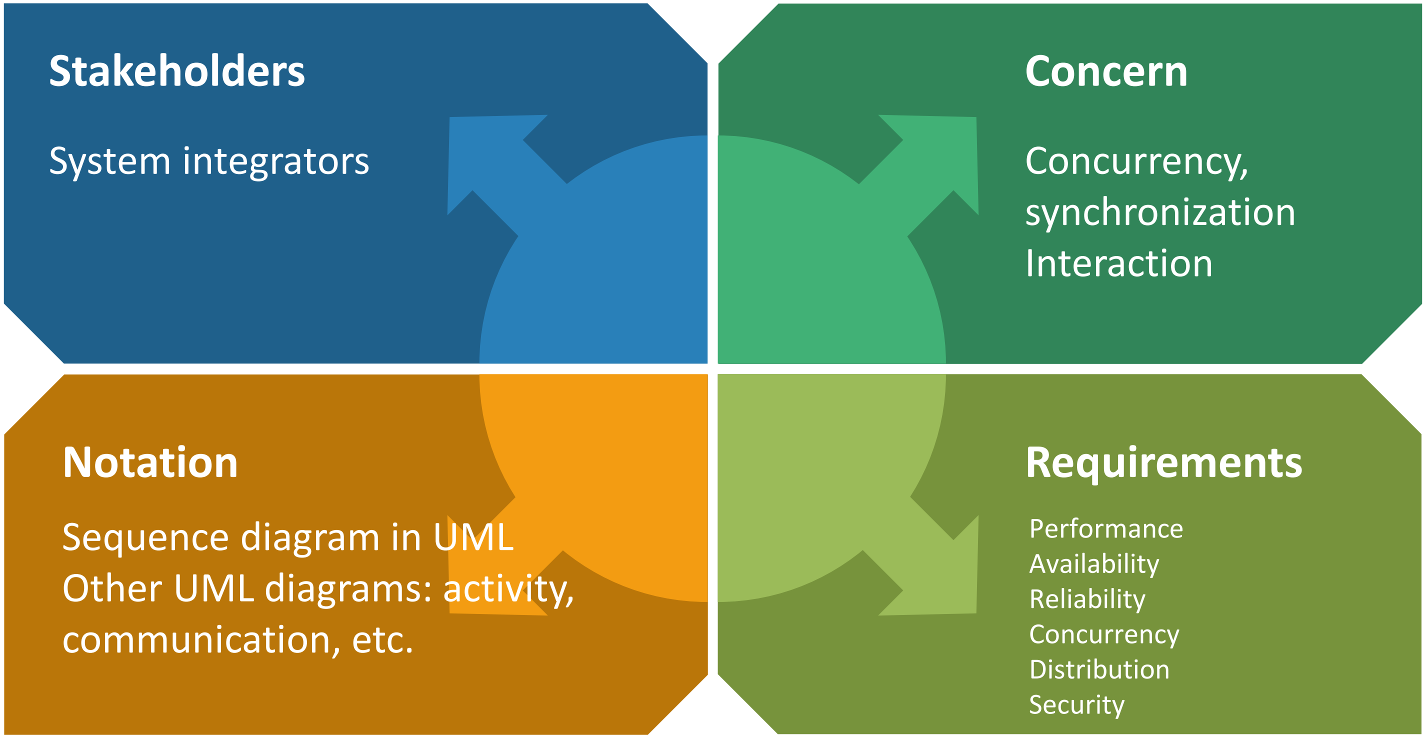
On-going example: a class diagram for Github



On-going example: a component diagram for Github



The 4+1 view model by Krutchen: process view



Stakeholders

System integrators

Concern

Concurrency,
synchronization
Interaction

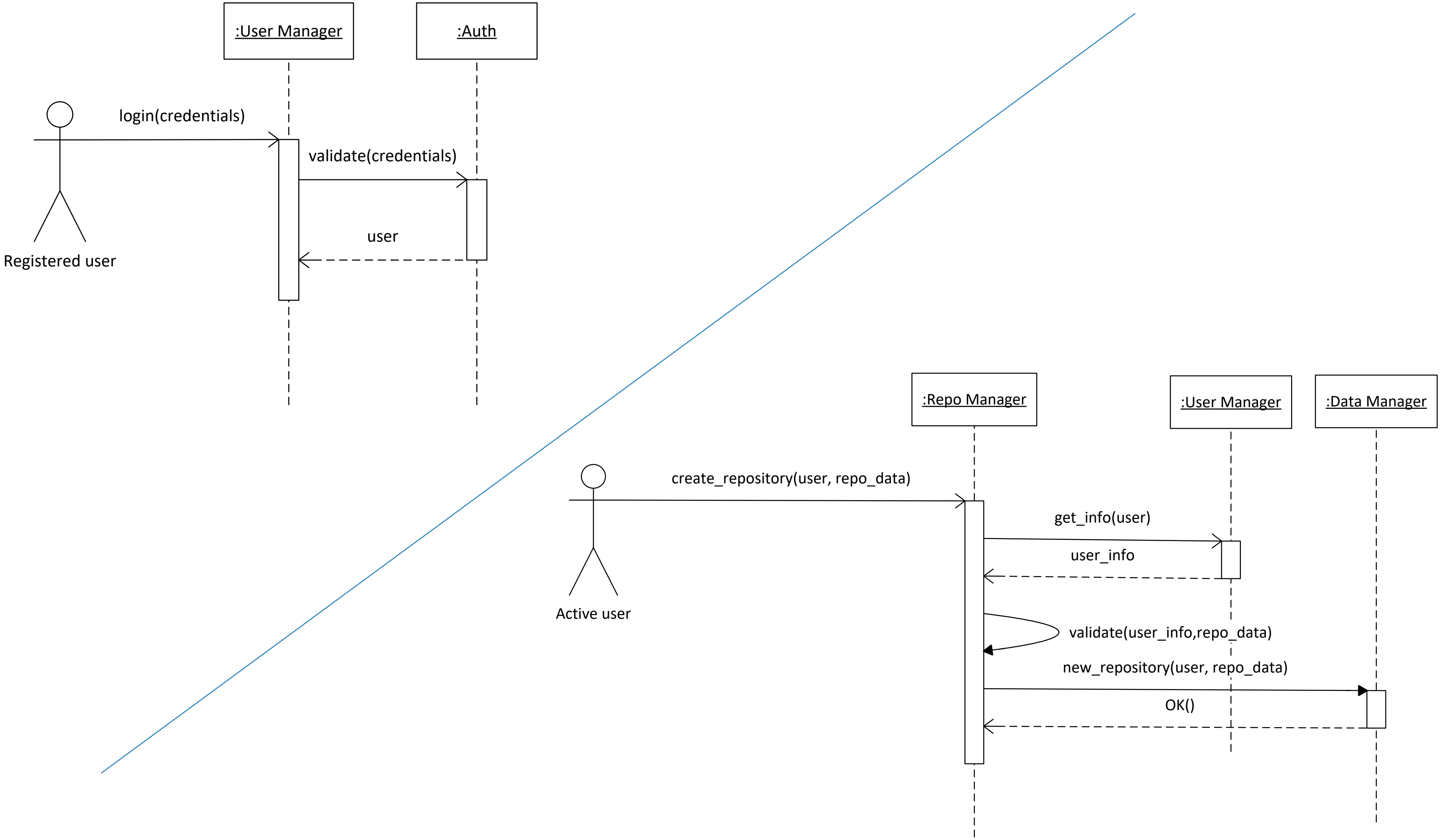
Notation

Sequence diagram in UML
Other UML diagrams: activity,
communication, etc.

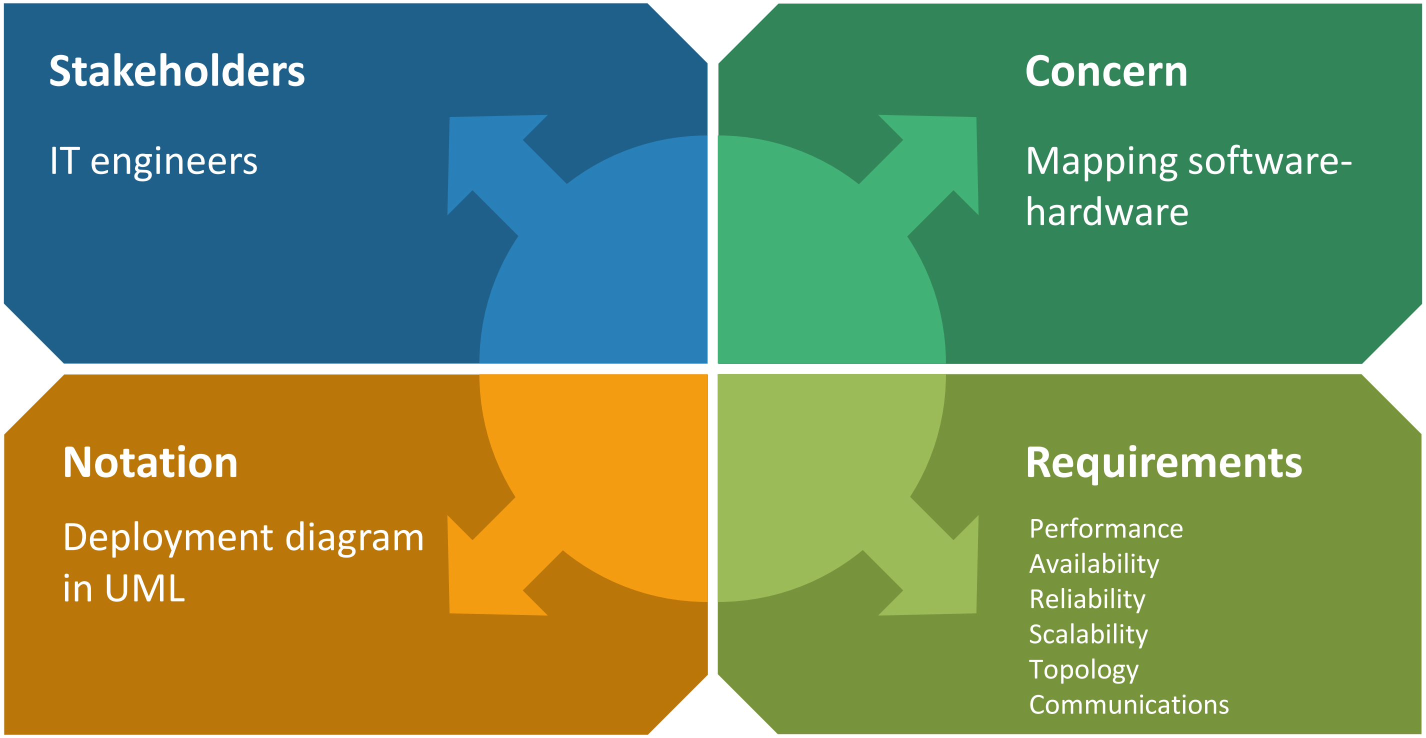
Requirements

- Performance
- Availability
- Reliability
- Concurrency
- Distribution
- Security

On-going example: a sequence diagram for Github



The 4+1 view model by Krutchen: physical view



Stakeholders

IT engineers

Concern

Mapping software-hardware

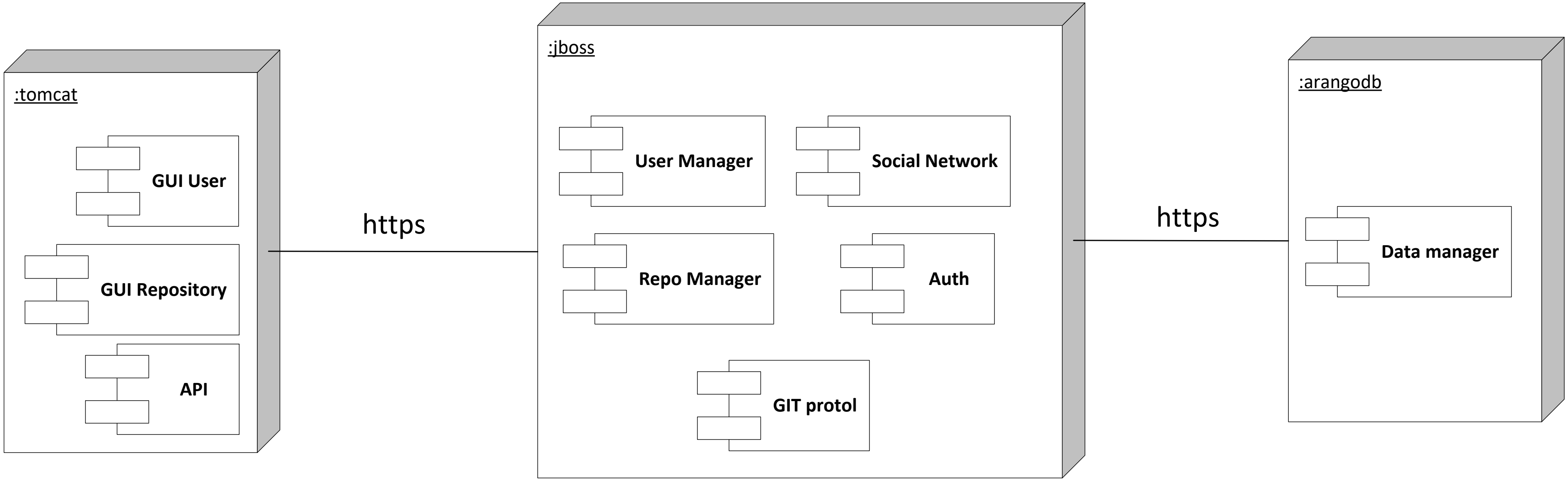
Notation

Deployment diagram in UML

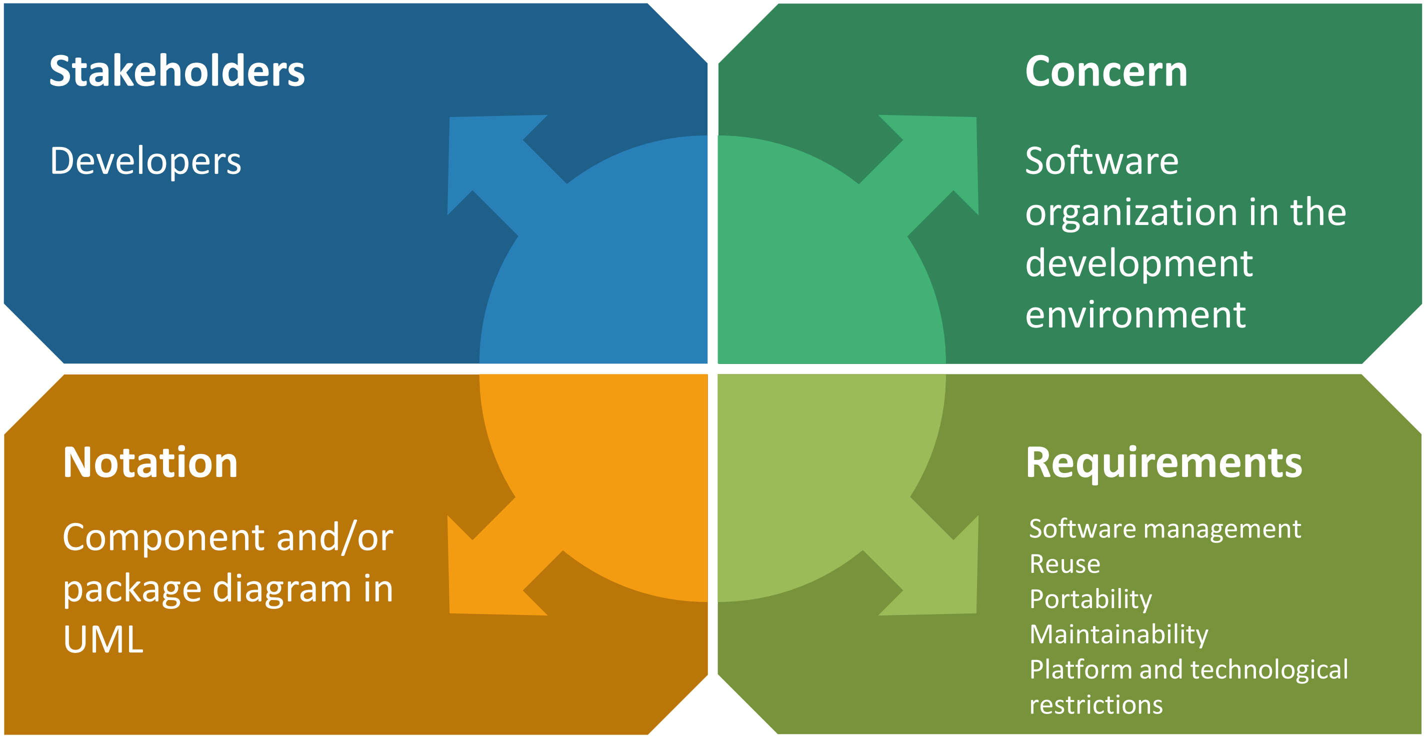
Requirements

Performance
Availability
Reliability
Scalability
Topology
Communications

On-going example: a deployment diagram for Github



The 4+1 view model by Krutchen: development view



Stakeholders

Developers

Concern

Software organization in the development environment

Notation

Component and/or package diagram in UML

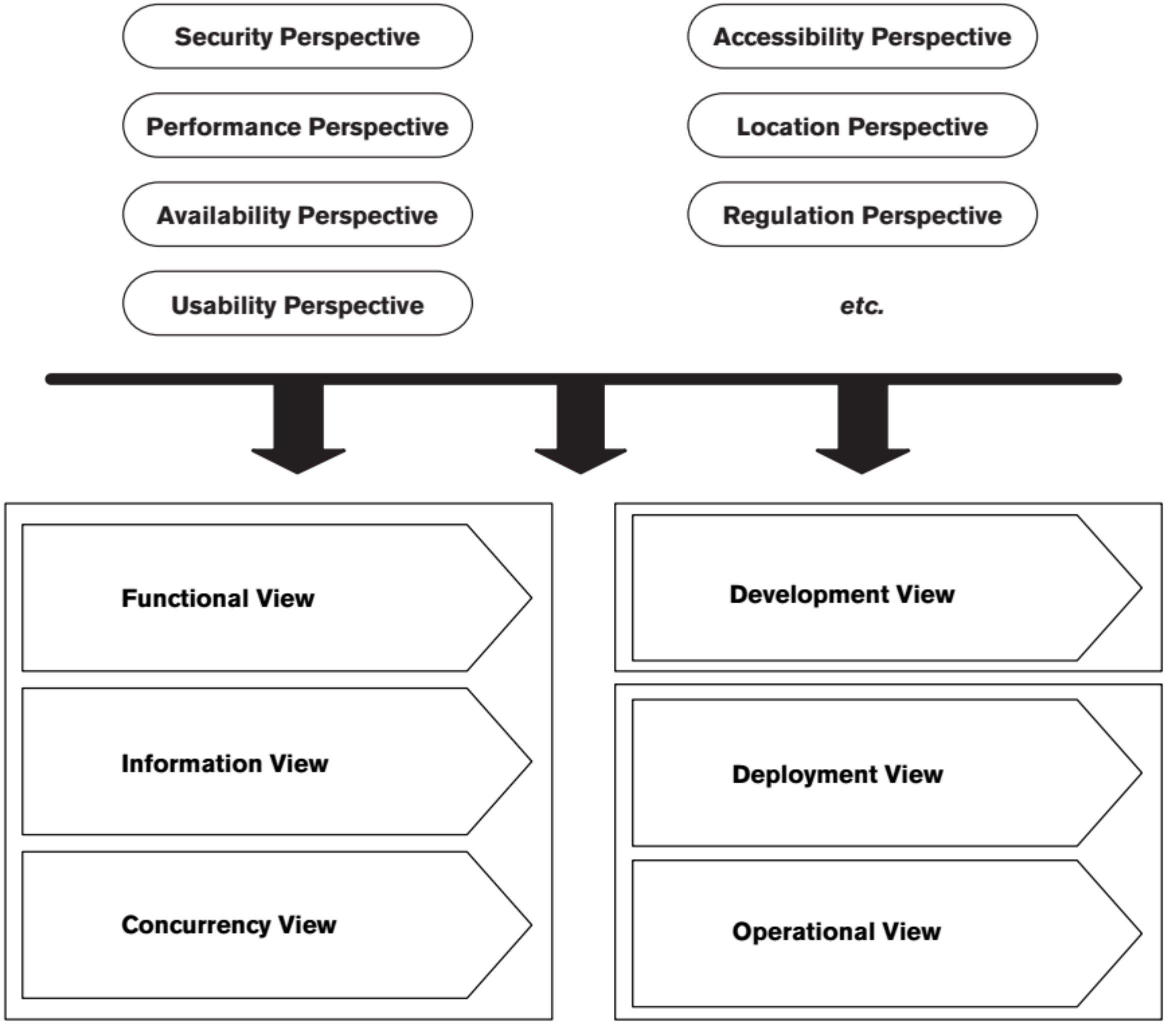
Requirements

Software management
Reuse
Portability
Maintainability
Platform and technological restrictions

The 4+1 view model by Krutchen: summary

	Logical (conceptual)	Process (runtime)	Development (implementation)	Physical (deployment)
Stakholders	Information model	Concurrency, synchronization Interaction	Software organization in the development environment	Mapping software-hardware
Concern	End-users	System integrators	Developers	IT engineers
Requirements	Functional	Performance Availability Reliability Concurrency Distribution Security	Software management Reuse Portability Maintainability Platform and technological restrictions	Performance Availability Reliability Scalability Topology Communications
Notation	Class and/or component diagram in UML	Sequence diagram in UML	Component and/or package diagram in UML	Deployment diagram in UML

Other view models: Rozanski & Woods [1]



Main risks of a multiple view model

Wrong views

- It is necessary to make a decision of which view are more adequate
- Dependencies among the different system concerns
- In general, a good architecture depends on experience and skills

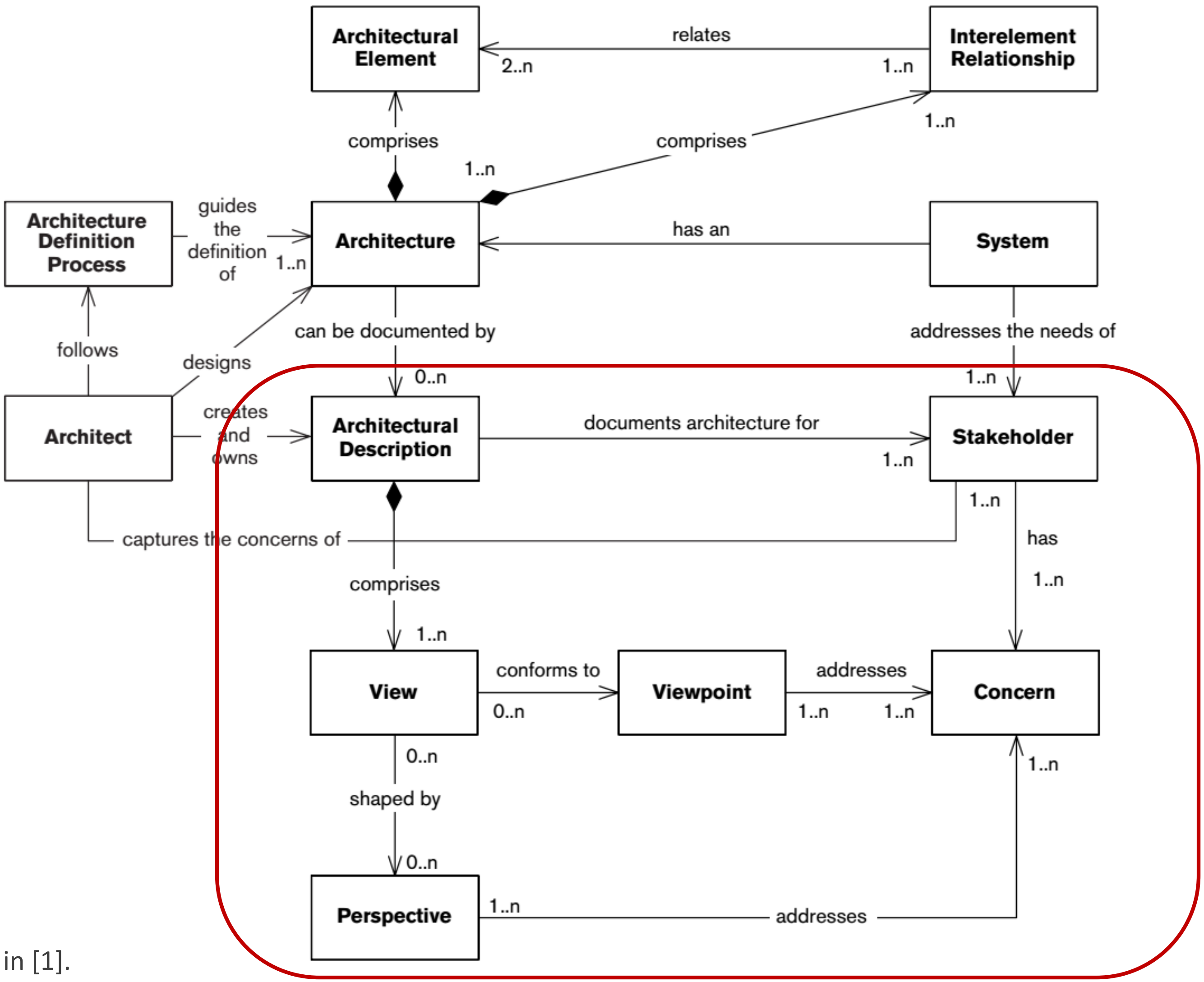
Fragmentation

- Kruchten 4-1 and Rozanski-Woods 6
- A view → creation and maintenance costs (consistency)
- Join view for the shake of simplicity

Inconsistency

- All views must be consistent, they represent the same system
- Maintenance costs to keep consistency
    - An scenario → process → components → classes
- Establish the proper dependencies to calculate the impact of a change.

Description of a software architecture



See original description in [1].

Some questions to think about…

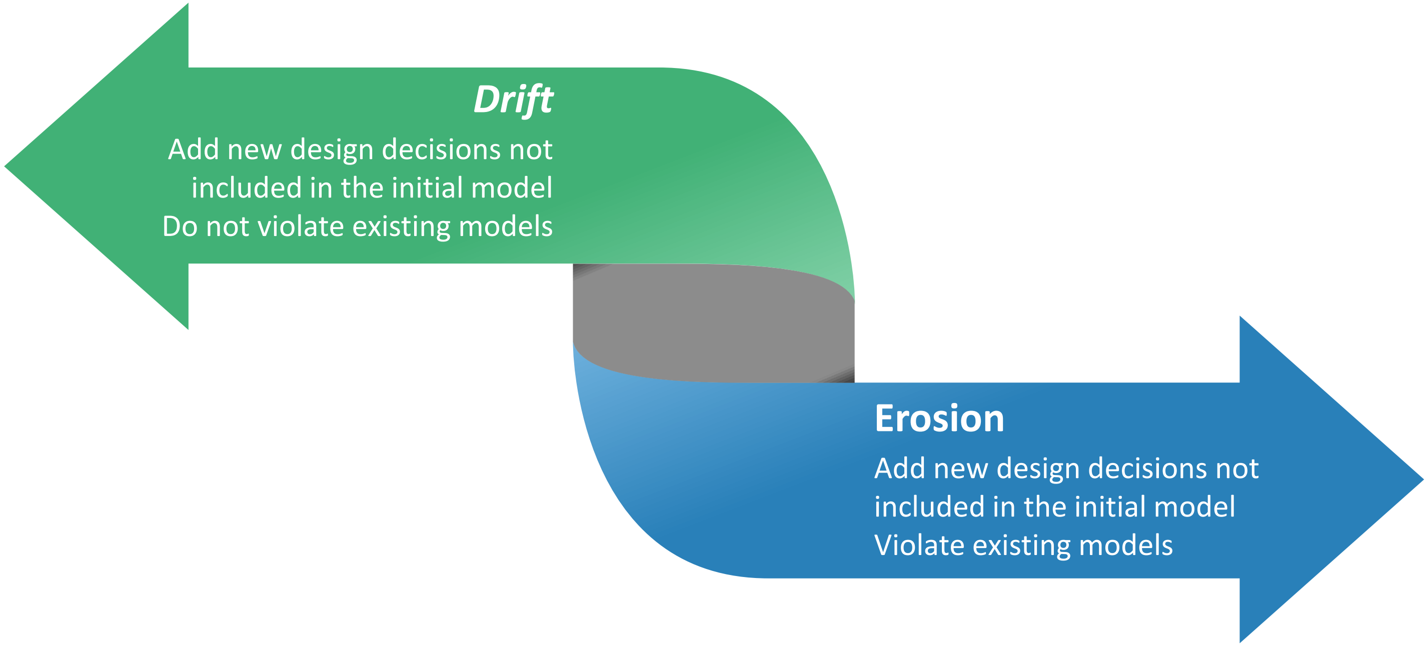
Why architecture and architecture descriptions?

How can I build relationships between different views?

What is the relationship with forward engineering?

What is the relationship with reverse engineering?

Known-problems in a software architecture



Drift

Add new design decisions not included in the initial model
Do not violate existing models

Erosion

Add new design decisions not included in the initial model
Violate existing models

→ Software architecture re-definition

A good enough software architecture…

1. It shall meet the needs of the stakeholders.

2. It shall communicate the relevant concerns to the proper stakeholders.

3. All functional and non-functional requirements cannot be specified in just one iteration.

A good enough software architecture…

Cohesion

A measure of the strength of association of the elements within a module. (ISO/IEC/IEEE 24765)

-Intra-module functionality

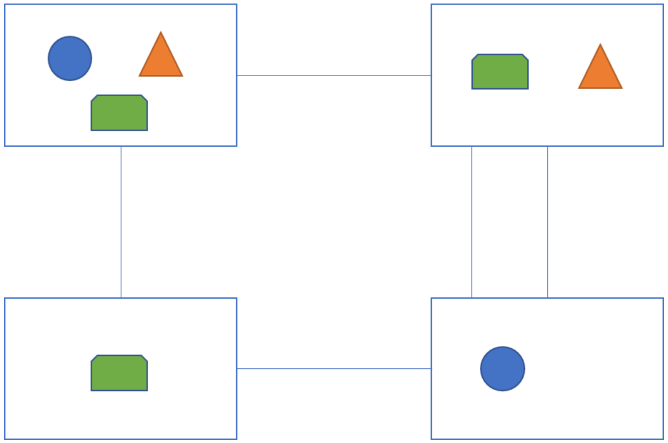
Preferred value: **high**

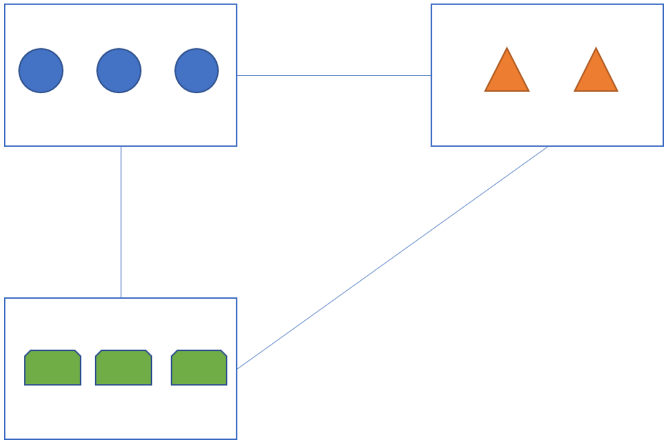
Coupling

A measure of the interdependence among modules in a computer program. (ISO/IEC/IEEE 24765)

-Inter-module dependencies

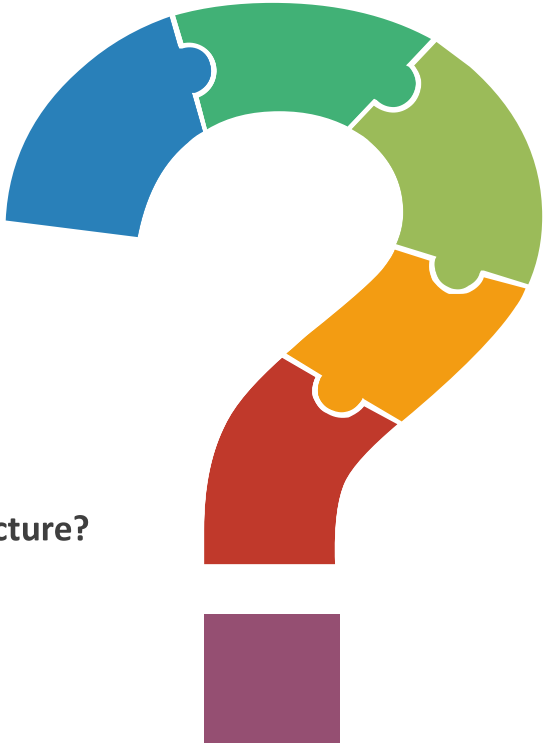
Preferred value: **low**





First questions: any answer?

- What is an architecture?
- Any example of architecture?
- What is a software architecture?



- Who affects a software architect?
- What is the role of a software architect?
- How a software architecture is described and documented?

Task

Objective

- Make a description of an existing software system
- E.g.: Twitter, Instagram, Linkedin, etc.

Method

- Work in groups
- 2-3 people
- 20 minutes

Outcome

- Use case, class and component diagrams

Evaluation

- Presentation and discussion
- 2 groups (volunteers?)
- 10 minutes

1-minute quizz questions (Aulaglobal)

Q1:A definition of “software architecture”…

a) Architecture is defined by the recommended practice as the fundamental organization of a system, embodied in its components, their relationships to each other and the environment, and the principles governing its design and evolution.
b) Architecture is defined by the recommended practice as the functional organization of a system.
c) Architecture is defined by the recommended practice as the fundamental organization of a system.
d) None of the answers is correct.

Q2: **In the 4+1 model, the development view…**

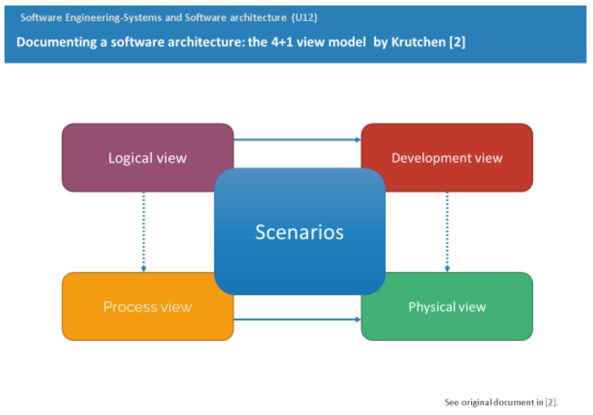
a) Represents the system from the developer perspective with a class diagram.
b) Represents the system from the developer perspective with a component or package diagram.
c) Represents the interactions between system components through a component diagram.
d) a) and b) are correct.

Q3: Which is an issue of having multiple views of an system?

a) Fragmentation
b) Inconsistencies
c) Too much views
d) All of them are problems.

Software Engineering-Systems and Software architecture

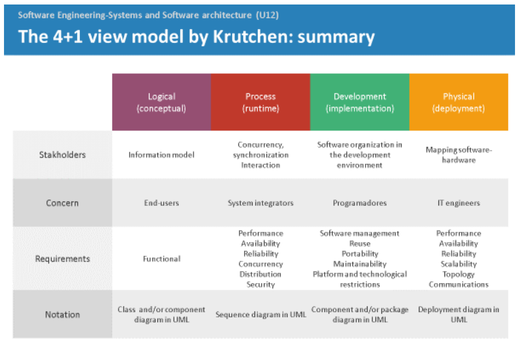
Summary



Software Engineering-Systems and Software architecture (U12)

Documenting a software architecture: the 4+1 view model by Krutchen [2]

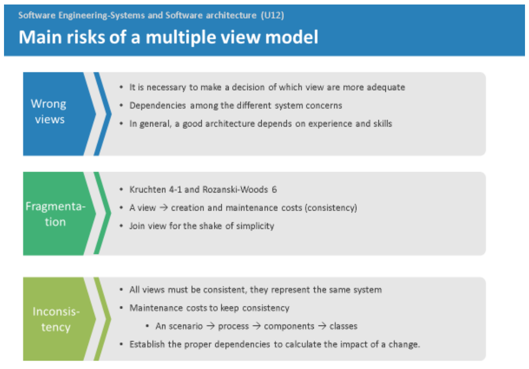
See original document in [2].



Software Engineering-Systems and Software architecture (U12)

The 4+1 view model by Krutchen: summary

	Logical (conceptual)	Process (runtime)	Development (implementation)	Physical (deployment)
Stakholders	Information model	Concurrency, synchronization Interaction	Software organization in the development environment	Mapping software-hardware
Concern	End-users	System integrators	Programadores	IT engineers
Requirements	Functional	Performance Availability Reliability Concurrency Distribution Security	Software management Reuse Portability Maintainability Platform and technological restrictions	Performance Availability Reliability Scalability Topology Communications
Notation	Class and/or component diagram in UML	Sequence diagram in UML	Component and/or package diagram in UML	Deployment diagram in UML



Software Engineering-Systems and Software architecture (U12)

Main risks of a multiple view model

Wrong views

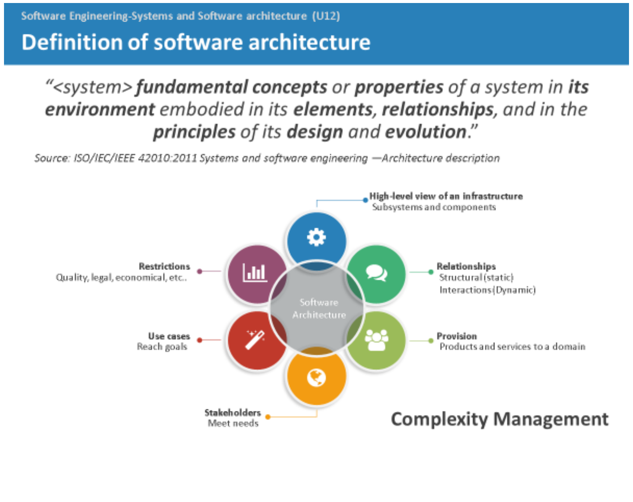
- It is necessary to make a decision of which view are more adequate
- Dependencies among the different system concerns
- In general, a good architecture depends on experience and skills

Fragmentation

- Kruchten 4-1 and Rozanski-Woods 6
- A view → creation and maintenance costs (consistency)
- Join view for the shake of simplicity

Inconsistency

- All views must be consistent, they represent the same system
- Maintenance costs to keep consistency
  - An scenario → process → components → classes
- Establish the proper dependencies to calculate the impact of a change.



Software Engineering-Systems and Software architecture (U12)

Definition of software architecture

"<system> ***fundamental concepts*** *or* ***properties*** *of a system in* ***its environment*** *embodied in its* ***elements, relationships****, and in the* ***principles*** *of its* ***design*** *and* ***evolution***."

Source: ISO/IEC/IEEE 42010:2011 Systems and software engineering —Architecture description

Complexity Management

References and specific resources

[1] N. Rozanski and E. Woods, *Software systems architecture: working with stakeholders using viewpoints and perspectives*, 2nd ed. Upper Saddle River, NJ: Addison-Wesley, 2012.

[2] P. B. Kruchten, “The 4+1 View Model of architecture,” *IEEE Softw.*, vol. 12, no. 6, pp. 42–50, Nov. 1995.

-Section “6.1 Architectural design decisions” from “Software Engineering 10th Edition”, I. Sommerville, 2016.

-Section “6.2 Architectural views” from “Software Engineering 10th Edition”, I. Sommerville, 2016.