Architecture of Distributed Systems

Intro Arch. 2019-2020

Original : J.J. Lukkien
Revision: R.H. Mak

Technische Universiteit
Eindhoven
University of Technology

Where innovation starts

Goal of the lecture

Students should understand:

- the notion of an architecture as a high-level description of a system from various viewpoints, its purpose and its role within the system development cycle.
- the formalization of this for the domain of software intensive systems and the motivation behind this.
  - as explained in the ISO/IEC/IEEE 42010 standard

Students should know:

- some common viewpoints, the principle concerns they address and typical models they use.

Material

International standard ISO/IEC/IEEE 42010

- Systems and software engineering – Architecture description ISO/IEC/IEEE STD 42010-2011, 35 pages

Software Systems Architecture: Working with Stakeholders Using Viewpoints and Perspectives, 2ed, *Rozanski and Woods*, Barnes and Noble, 2011

- http://www.viewpoints-and-perspectives.info/
- the stakeholders, views and perspectives tabs on the website cover the material of the slides

Some questions

- **What *is***
  - (a) software architecture?
  - (a) system architecture?
- **Why do we need them?**
- **What do I *make* when I say I make**
  - a design?
  - an architecture?
  - what is the *tangible* result of my work?
- **What is the *quality* of an architecture?**
  - how to discriminate between good and bad ones?
  - is the tangible outcome according to accepted rules?
- **Can I see that a system has been built according to a certain architecture?**
  - Is there a description that serves as a form of documentation or prescription for realization?

Architecture vs Design (from presentation by L. Fehskens)

Architecture:

- about the *entire entity* in its environmental context;
- a different architecture implies a *different mission*
- defines *a class* of acceptable solutions
- role of the architect is mostly to make *correct inferences*
- is based on the (business) context (concerns of stakeholders)
- is technology independent

Design:

- about *components and subsystems* of the entity
- different designs may address the *same mission*
- defines *a single* specific solution
- role of the designer is mostly to make *correct decisions*
- conforms to an architecture.

Architecture: why bother?

- **Do I need an architecture when I build**
  - a fence?, a dog shed?, a program to compute the first 10000 primes?
- **However, it is different for tasks that require**
  - a global understanding of a complicated system
    - including its role in and interaction with its environment
  - analysis of design alternatives, and of general system properties
    - *before such system is built*
    - *or without having access to it*
  - communication between team members and with customers
  - communication (documentation) for handing off parts of the work
  - documentation for later reference
    - in the maintenance phase of the lifecycle
    - to guide the evolution of the system
  - decomposition and synthesis of parts
- **Larger technical systems are simply too complex and long-lasting for one person**

Purpose of the architecture

- **Communication / Understanding**
  - To define global meaning and scope of the system
  - With stakeholders to see whether the right system is made
  - With (teams) of developers to guide the construction
  - With users / customers to provide insight how to use the system
- **Analysis**
  - In general to answer questions about the system
  - To evaluate candidate architectures on an abstract level
    - w.r.t. all kinds of quality attributes
  - To make design decisions that have significant impact on the cost and performance of the system
- **Construction (Synthesis)**
  - Floorplan / blueprint for the overall structure
    - Complete, but at a high level of abstraction
  - Basis for detailed design by developers
    - Indication of used technologies / of-the-shelf components

Questions about a system

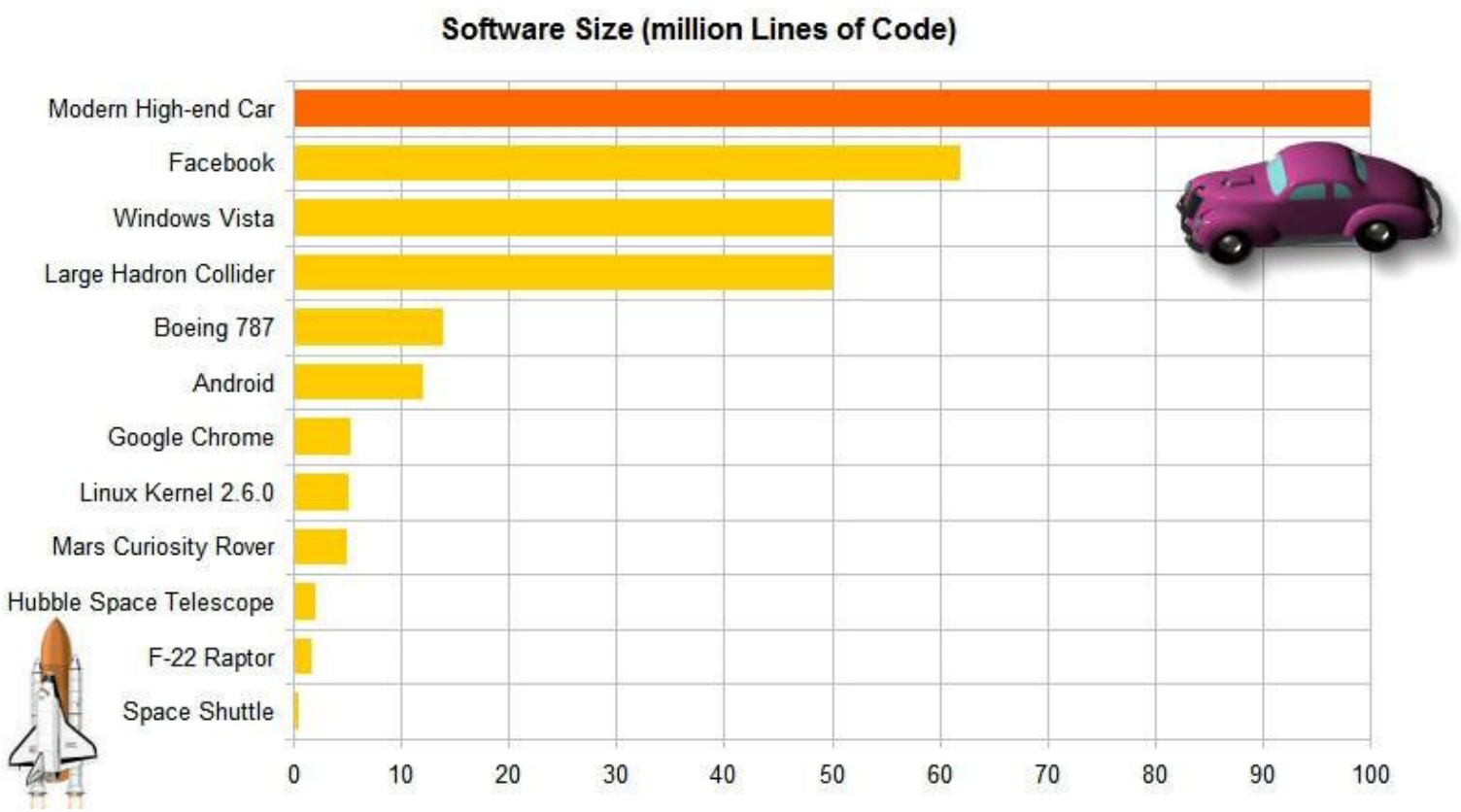
Will it not kill the patient?

https://www.usa.philips.com/healthcare/solutions/interventional-xray/allura

Will it bring me to Paris within 3 hours?

https://en.wikipedia.org/wiki/Fyra

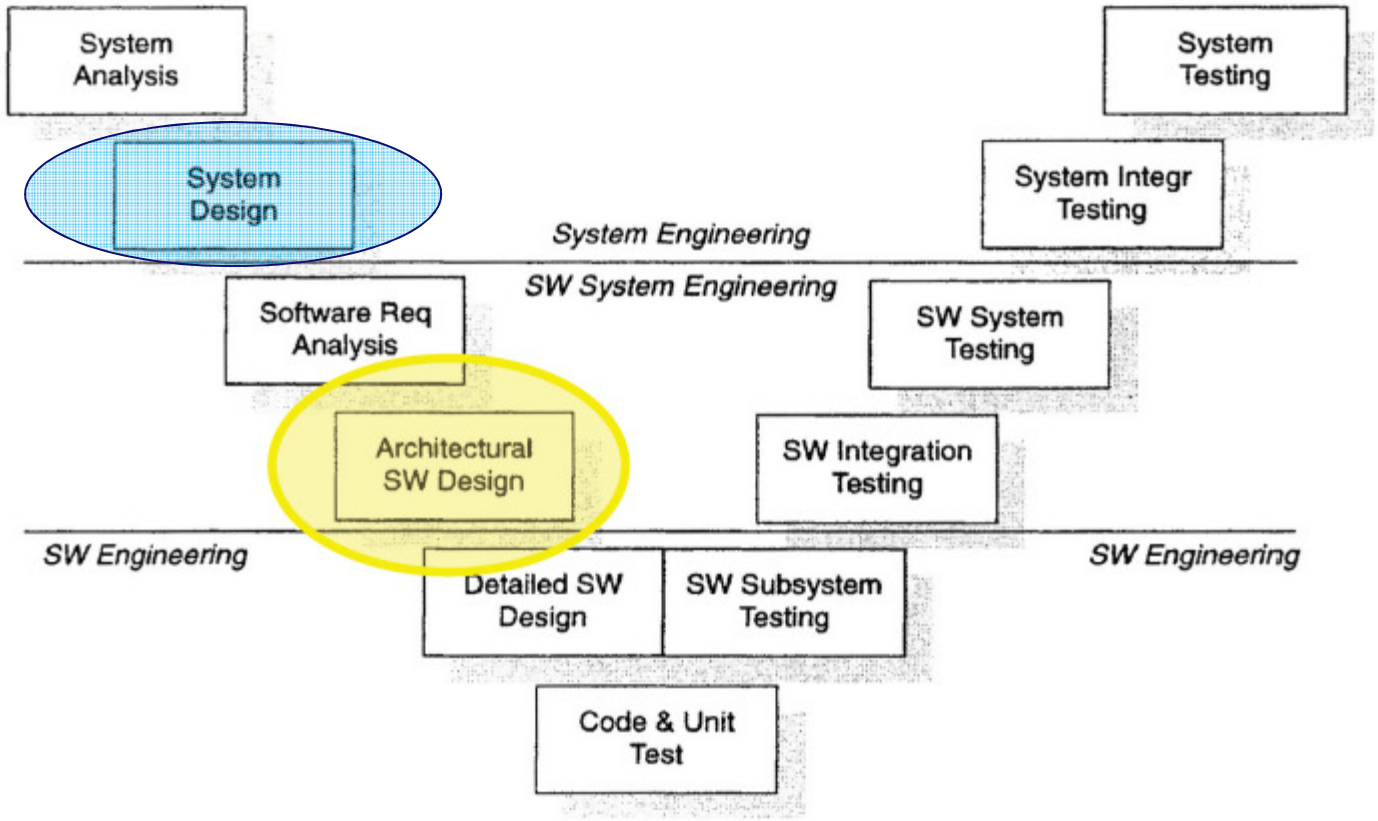
(Embedded) Software Complexity



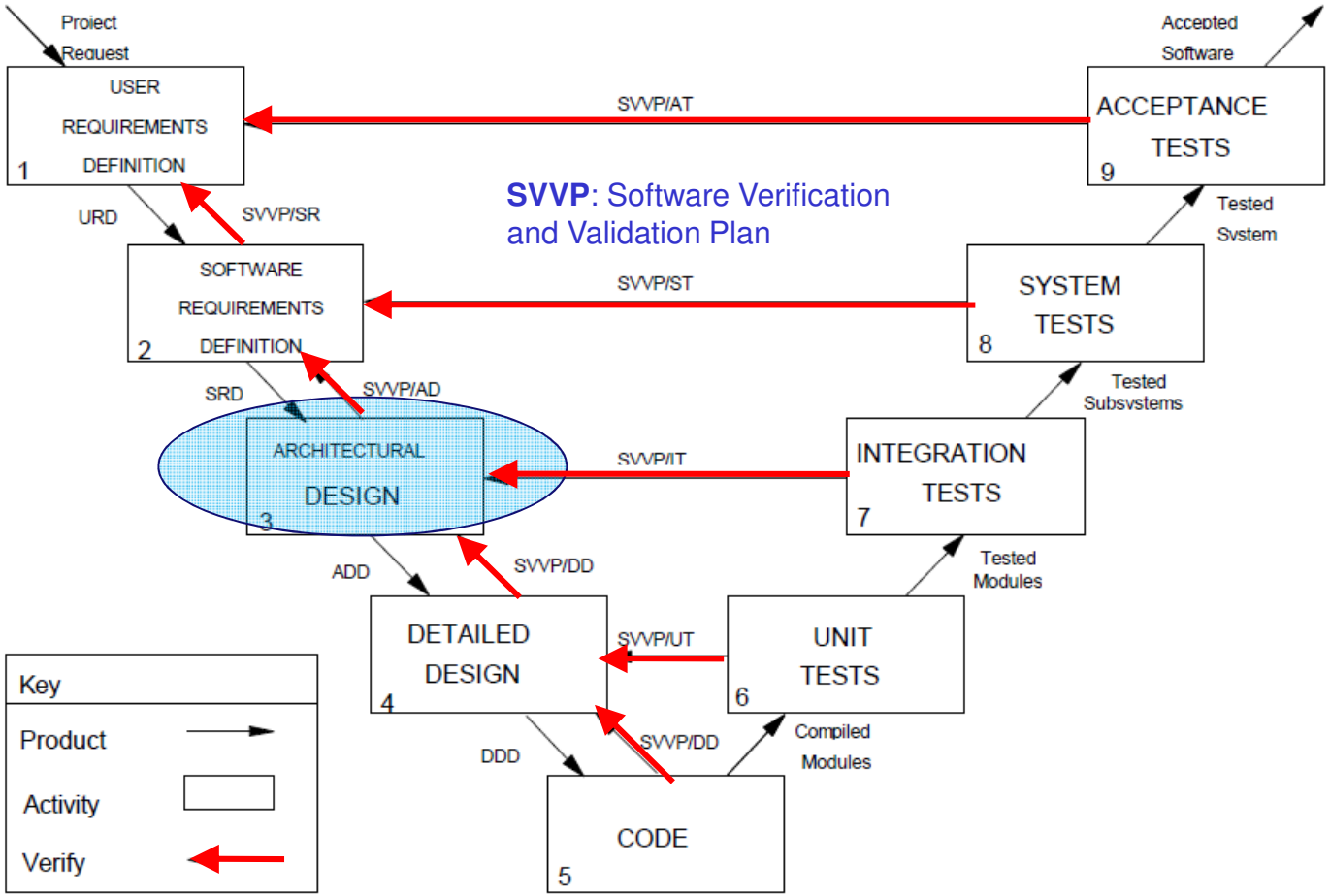
Technische Universiteit Eindhoven University of Technology

Architecture: place in system development cycle

From: M.J. Christensen, R.H. Thayer. *The Project Manager's Guide to Software Engineering's Best Practices*. Wiley, 2002



Software Architecture: place in software development cycle



Esa standard: life cycle verification approach

Design *process*

- **Translate a *design problem* into (a *model/blueprint* of) a solution**
- **The process is *iterative* (perhaps *recursive*)**
  - level *n+1* solves a set of smaller design problems than level *n*
    - smaller problems are subject to the same approach
  - models are increasingly more detailed until the problems are no longer of a structural nature
    - hence, 'detailed design' is of a different nature than architecture design
  - during this process *design decisions* are taken
    - choosing among options
    - challenge: *document* these decisions and their rationales (whose concern?)
- **Building blocks per level are different**
  - needs a good understanding per level
    - and the mutual impact of lower levels on higher levels
  - may lead to *specialization* in the development process
    - ....a building architect usually does not design the plumbing details
    - ....a software architect is not concerned with implementation of simple data structures
  - note that building block and connector properties determine system functionality

Design *process* – four elements

- **(Domain) analysis**
  - increase knowledge, make models
    - use cases, based on stakeholder viewpoints
  - feedback to stakeholders:
    - validation of requirements ("Do we solve the right problem?")
- **Apply strategies**
  - hierarchical decomposition:
    - top-down (factorization): *specify* advanced building blocks (decompose functional specification, and derive extra-functional properties for the parts)
    - bottom up: *design and compose* advanced building blocks
  - apply patterns, styles, tactics
    - pattern, style: *coherent set of design decisions*
  - generate alternatives
- **Synthesis**
  - evaluate and choose alternatives, combine partial solutions
- **Verification**
  - is the system according to specification? ("Did we solve the problem right?")

Architecture (42010 definition)

The architecture of the system is

- the set of *fundamental concepts* or properties of a system in its *environment* embodied in its *elements, relationships*, and in the *principles of its design and evolution*

1. Architectures are created solely for the benefit of stakeholders
   - A *good* architecture is one that successfully addresses the concerns of its stakeholders and, when those concerns are in conflict, balances them in a way that is acceptable to the stakeholders
2. Every system has an architecture, but it may be implicit.
3. An architectural description makes an architecture explicit

Architecture description

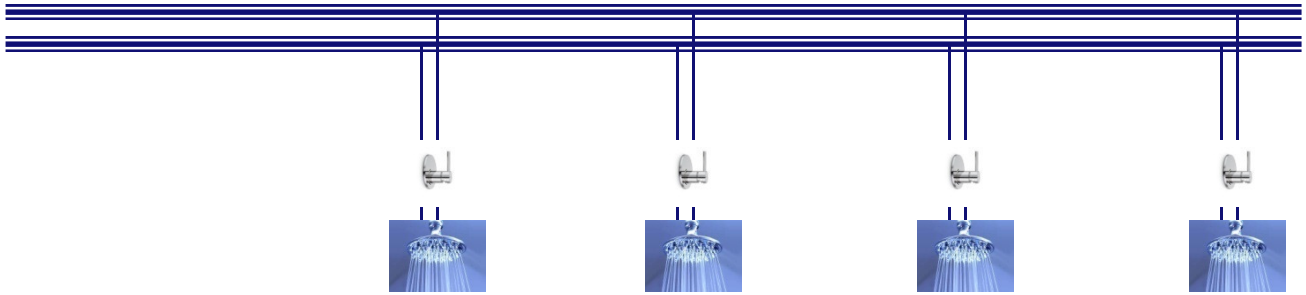
An architecture is described by a collection of models/ blueprints

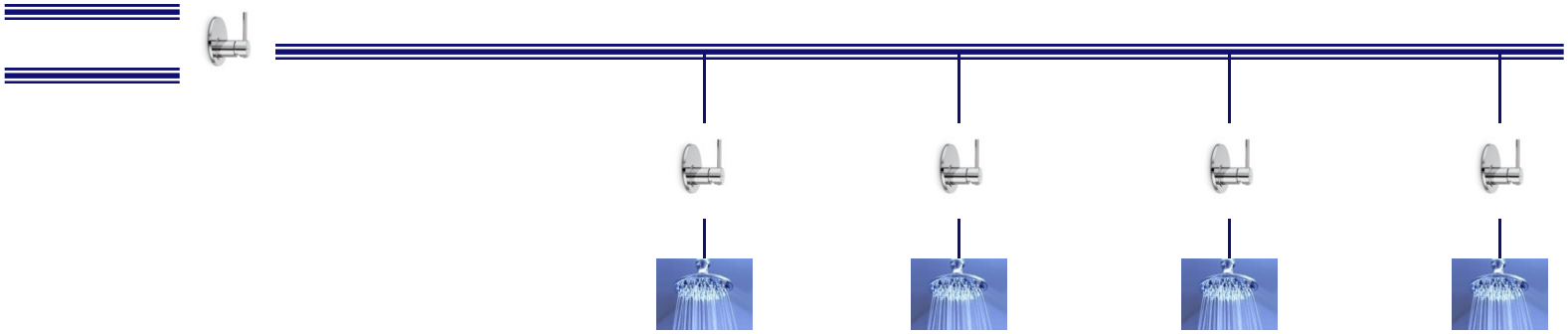
- the models are organized into *views*
- the architecture description can be examined at varying levels of abstraction and different *viewpoints*
- the first (top-most) set of blueprints is special, and is also referred to as ‘the architecture’ (or better: ‘the architecture description’ AD):
  - it needs to make the transition to the real world (technical and operational environment, use cases)
  - it presents the system at a high level (the highest) of abstraction

as such it is used for *understanding*, *analysis, communication, construction, documentation .... answering questions*

- e.g. evaluation of utility, cost and risk

Two models for a shower control system





History of Architecture Description

- **1995: “4+1” paper by Kruchten**
  - Makes the usage of multiple views widely accepted.
  - Advocates 4 specific views, plus scenarios to address system behavior.
- **2000: IEEE Standard 1471**
  - Presents a formal conceptual model for architectural descriptions (ADs) that standardizes terminology.
  - Distinguishes between viewpoints and views
    - Kruchten’s views are in fact viewpoints!!!
- **2011: New ISO/IEC IEEE standard 42010**
  - Extension of 1471.
  - Introduces the notion of *architecture frameworks* and their support by means of *architectural description languages*

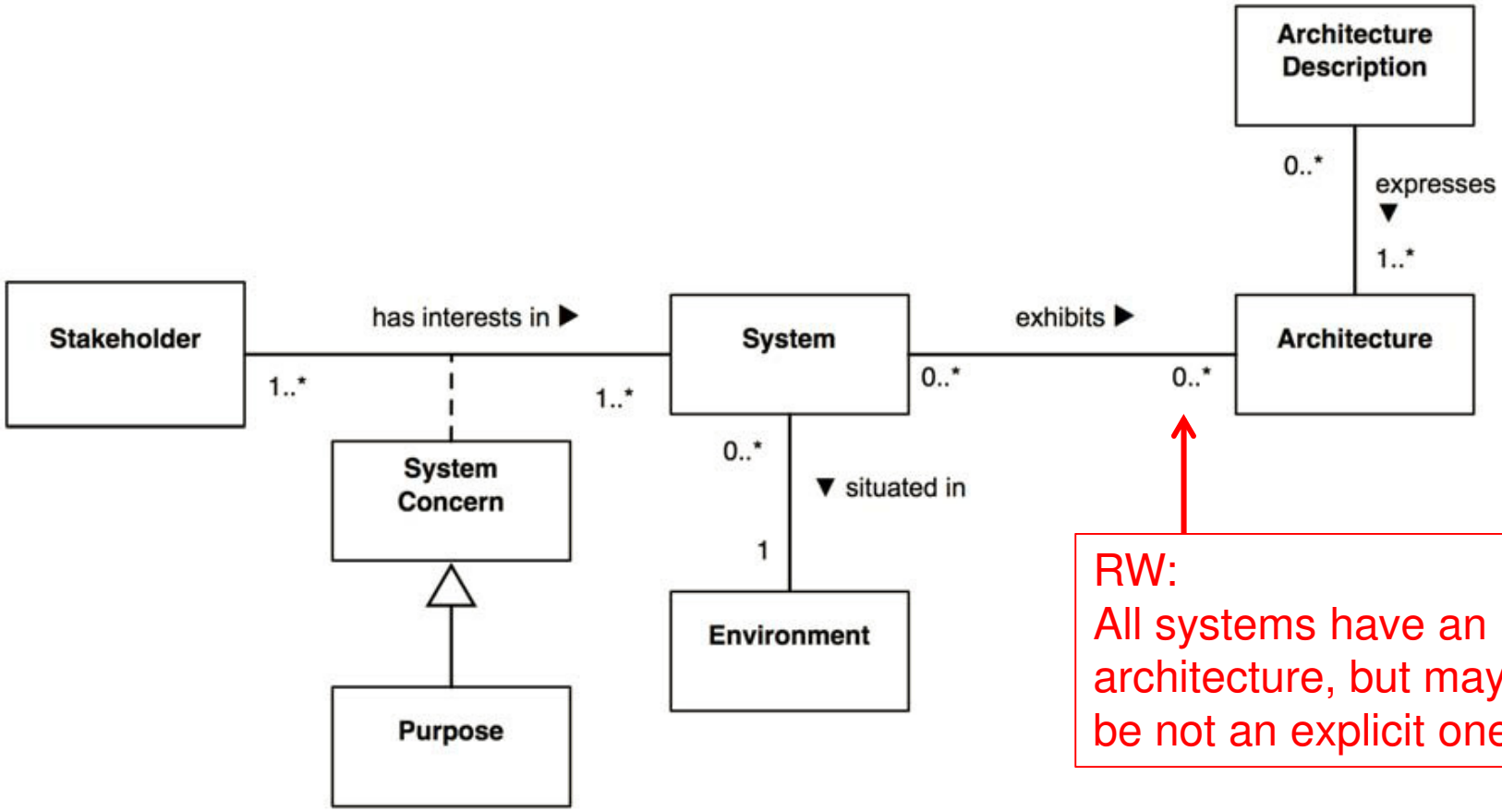
ISO/IEC/IEEE 42010 standard

- **Revision/update of IEEE 1471**
- **Provides a core ontology for the description of architectures.**
- **Specifies the organization of these descriptions using**
  - Architecture viewpoints, Architecture frameworks and Architecture description languages
- **Defines and motivates concepts and presents their relationships in a set of (meta-)models**
  - Architecture context model
  - Architecture description model
  - Architecture decision and rationale model
  - Architecture framework model.
  - Architecture description language model

ISO/IEC/IEEE 42010: Concepts

- **Most concepts have already been defined in the IEEE 1471 standard (see previous slide) and their definition remains largely the same. The most important ones are:**
  - Architecture, architecture description,
  - System, environment
  - Viewpoint, view,
  - Architecture model
  - Stakeholder, concern
- **New concepts are**
  - Architecture rationale
  - Architecture framework
  - Architecture description language

Architecture context model



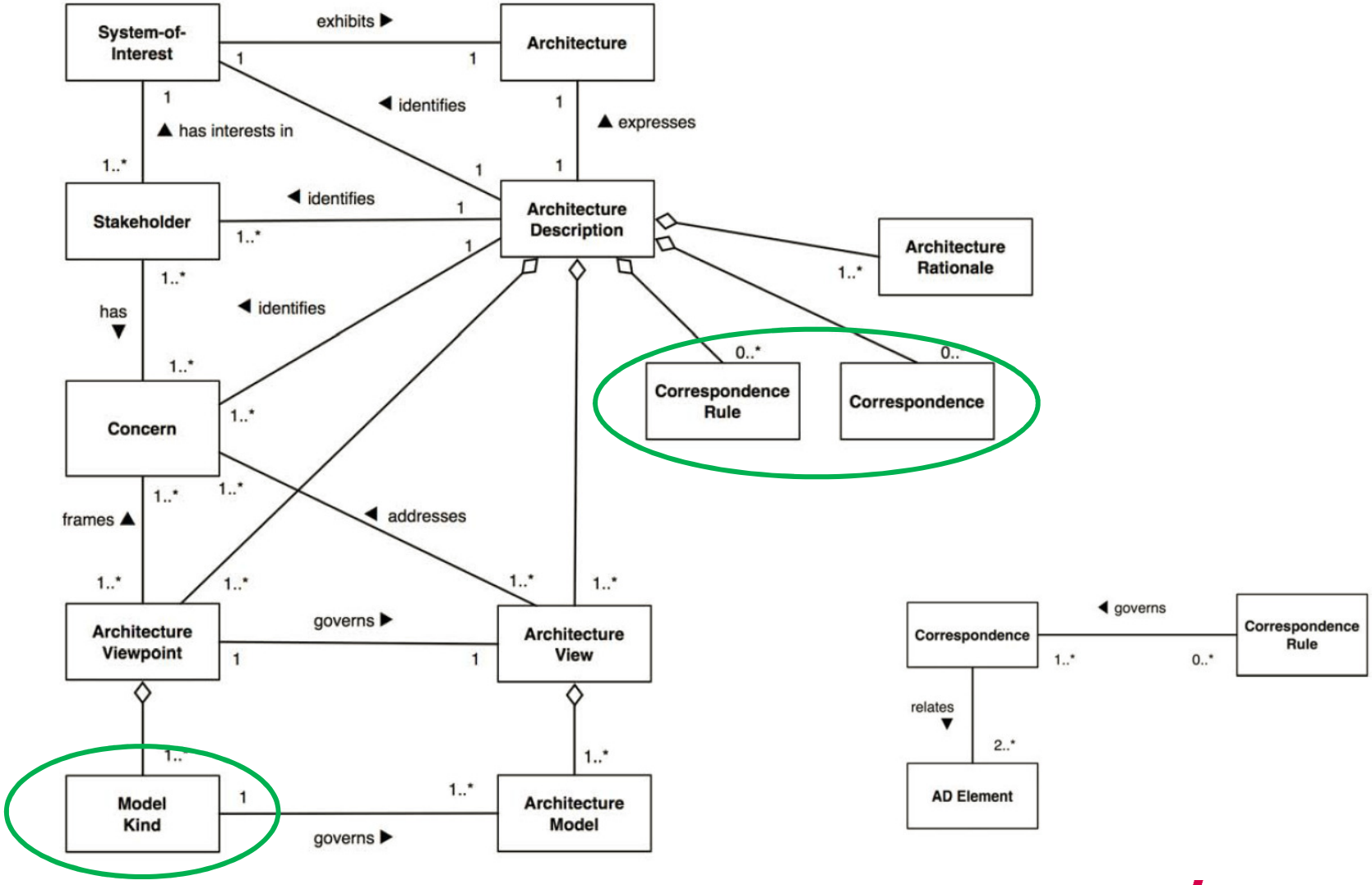
RW:
All systems have an architecture, but may be not an explicit one.

Quiz

- What is the system?
- What is its purpose?
- What is, *or are important aspects of*, its environment?
- What about the other entities from the context model?

Taken from:
https://upload.wikimedia.org/wikipedia/commons/c/c8/De_Rat_molen_IJlst_25.JPG

Architecture description model



Stakeholders (taken from Rozanski-Woods)

- **Definition**
  - A person, group, or entity with an interest in or concern about the realization of the architecture.
- **Criteria**
  - Stakeholders should be:
    informed, committed, authorized, representative.
- **Responsibilities**
  - Communicate concerns (theirs or those of the group they represent) to the architect.
  - Review the architectural description (AD).
  - Decide on the acceptance of the architecture and other issues about the system.

Beware. It are the roles of stakeholders that are important not their identities. The same person can hold multiple roles.

Stakeholder role classification (from RW)

Class	Description
Acquirers	Oversee the procurement of the system
Assessors	Oversee conformance to standards and legal regulations
Communicators	Explain the system via documentation and training
Developers	Construct and deploy the system from its specifications
Maintainers	Manage the evolution of the system once operational
Suppliers	Provide hardware/software platform on which the system will run
Support staff	Assist users to make use of the running system
System administrators	Run the system once deployed
Testers	Check whether the system meets its specifications both functional and extra-functional
Users	Define the system's functionality and use it once running

Models and views

- **Model: abstract, simplified (partial) representation of a system**
  - leaving out details irrelevant to a given set of criteria (“concerns”)
  - while preserving the properties of interest with respect to those concerns
  - can be used to answer a set of questions about the system
  - a single model is often insufficient to describe a complex system

- **View: a representation of one or more structural aspects of an architecture that illustrates how the architecture addresses some concerns of some stakeholders**
  - a collection of models
  - formal or informal, graphic, text
  - the view yields the means to address the concerns in the viewpoint, typically, via particular models
  - the view *conforms* to the viewpoint, i.e., it is described according to the conventions laid-down in the model kinds of the viewpoint and represents what you ‘see’ from that viewpoint

Viewpoints (42010 definition)

A *viewpoint* is a

- collection of patterns, templates, and conventions for constructing one type of view. It defines the stakeholders whose concerns are reflected in the viewpoint and the guidelines and principles, and template models for constructing its views

A *viewpoint* is characterized by

- a set of *concerns* to be addressed
- a set of *stakeholders* interested in how they are addressed
- one or more *model kinds*
- the *conventions*: concepts, notations, rules, patterns, styles and semantics to be invoked in creating, interpreting and using models of each kind
- *correspondence rules* linking the models together.

Viewpoints libraries

Viewpoints are generic, hence they can be collected in viewpoint catalogs (or libraries) and reused.

Examples:

- **Kruchten 4 + 1 views**
  - **Beware Kruchten's views are viewpoints in 42010 parlance**
- **Rozanski-Woods**
- **Many-many more, see:**
  - **ISO/IEC 42010 Viewpoints Repository [42]**

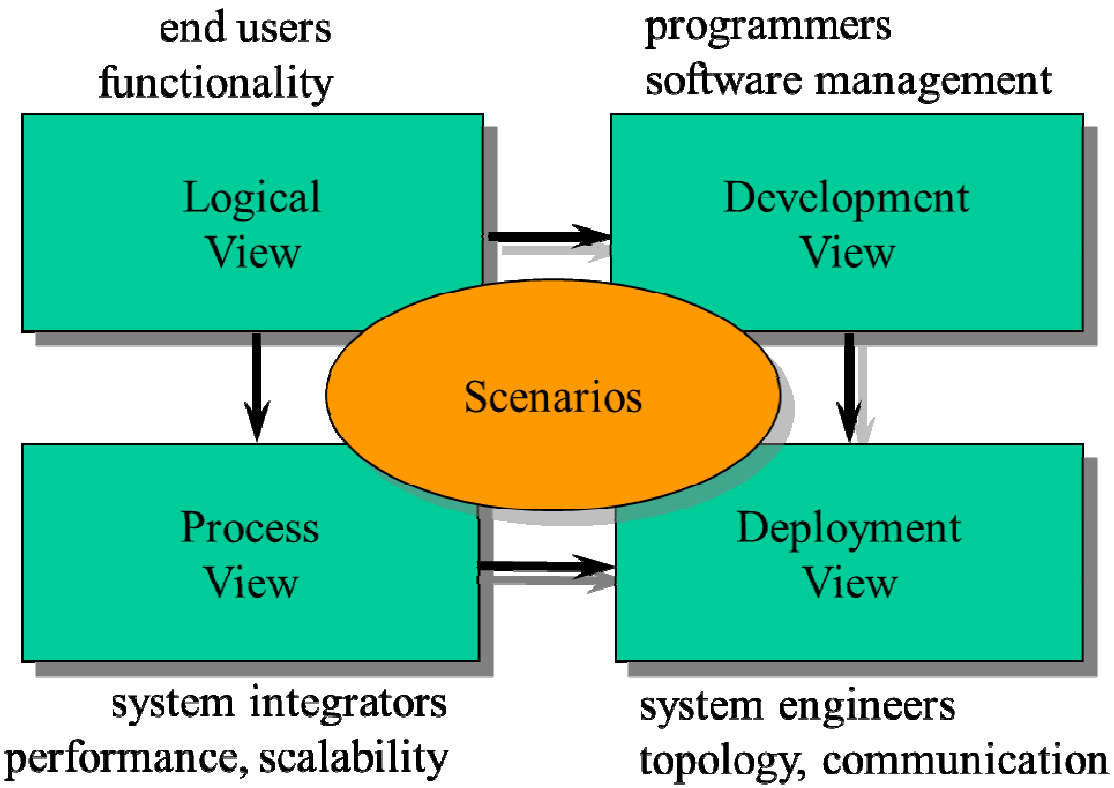
Kruchten's views and stakeholders

Article: Kruchten, Architectural Blueprints—The "4+1" View Model of Software Architecture, IEEE Software 12(6), Nov. 1995, pp 42—50.

Stakeholder	Concerns	View
User	Using the system; associated qualities • Externally visible structure and functionality	Logical
Programmer	Implementing/modifying the system • Decomposition into subsystems • Organization into files, components and modules	Development
System Integrator	Performance aspects, interaction between parts • Units of deployment (programs, components) and concurrency (processes, threads)	Process
System Engineer	Installing/deploying/realizing the system • Computers, networks, infrastructure, protocols, distribution, mapping of software to hardware	Deployment Physical
	Sets of interactions with or within the systems, integrating (the models in) the views and providing behavioral models within the views	Scenarios

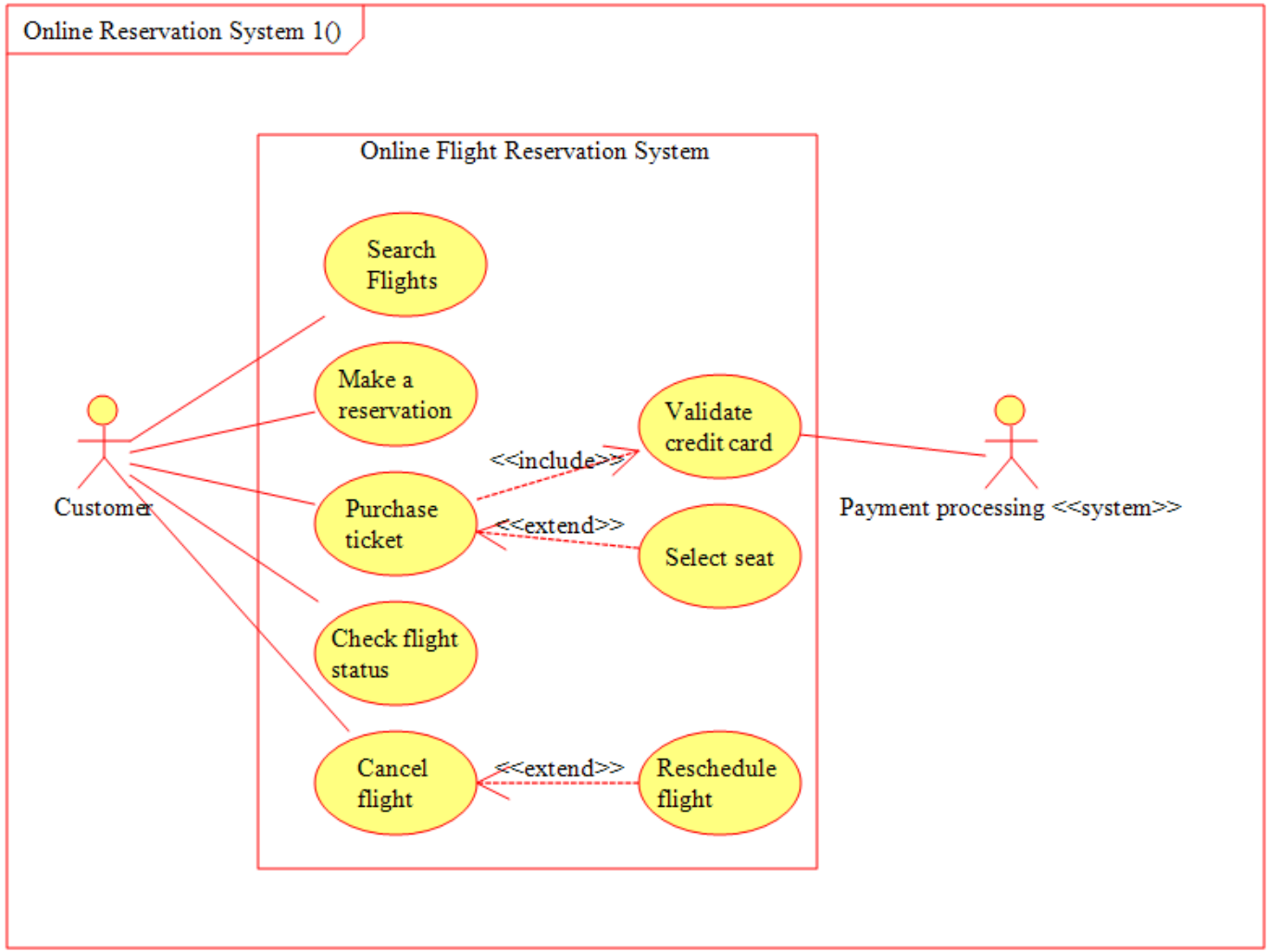
Kruchten views (viewpoints)

A number of standard (library) views



4+1 View Model [Kruchten 95]

Logical view + Scenario



Taken from:
https://tjhconsulting.wordpress.com/2012/12/20/uml-part-1-use-case-diagrams/

RW: Viewpoint catalog

- Context viewpoint
- Functional viewpoint
  - (analogous to Kruchten's Logical view)
- Information viewpoint
- Concurrency viewpoint
  - (analogous to Kruchten's Process view)
- Development viewpoint
- Deployment viewpoint
- Operational viewpoint

See also http://www.viewpoints-and-perspectives.info/home/viewpoints/

RW: viewpoint library (1 of 4)

The context viewpoint

- describes the relationships, dependencies, and interactions between the system and its environment (people, systems, external entities).
- is relevant to all stakeholders, in particular, acquirers, users and developers.
- addresses concerns like: system scope and responsibilities; identity and nature of external services; data used; identity, nature and characteristics of external interfaces; impact of the system on its environment; overall completeness, consistency and coherence.

RW: viewpoint library (2 of 4)

The functional viewpoint

- describes the system's runtime functional elements and their responsibilities, interfaces and primary interactions.
- is relevant to all stakeholders
- addresses concerns like: functional capabilities, external interfaces, internal structure and design philosophy.

The information viewpoint

- describes the way the architecture stores, manipulates and distributes information.
- is primarily relevant for users, acquirers, developers and maintainers
- addresses concerns about information structure, content and flow; data ownership, volume, validness, lifetime, and accessibility; transaction management; recovery; regulations.

RW: viewpoint library (3 of 4)

The concurrency viewpoint

- describes the concurrency units of the system, their functionality, and the required coordination
- is relevant to developers, testers and some administrators
- addresses concerns like: task structure, inter process communication, state management, synchronization and reentrance, process creation and destruction

The development viewpoint

- describes the architecture that supports the development process
- is relevant to software developers and testers
- addresses their concerns about module organization, common processing, standardizations of design and testing, code organization and instrumentation.

RW: viewpoint library (4 of 4)

The deployment viewpoint

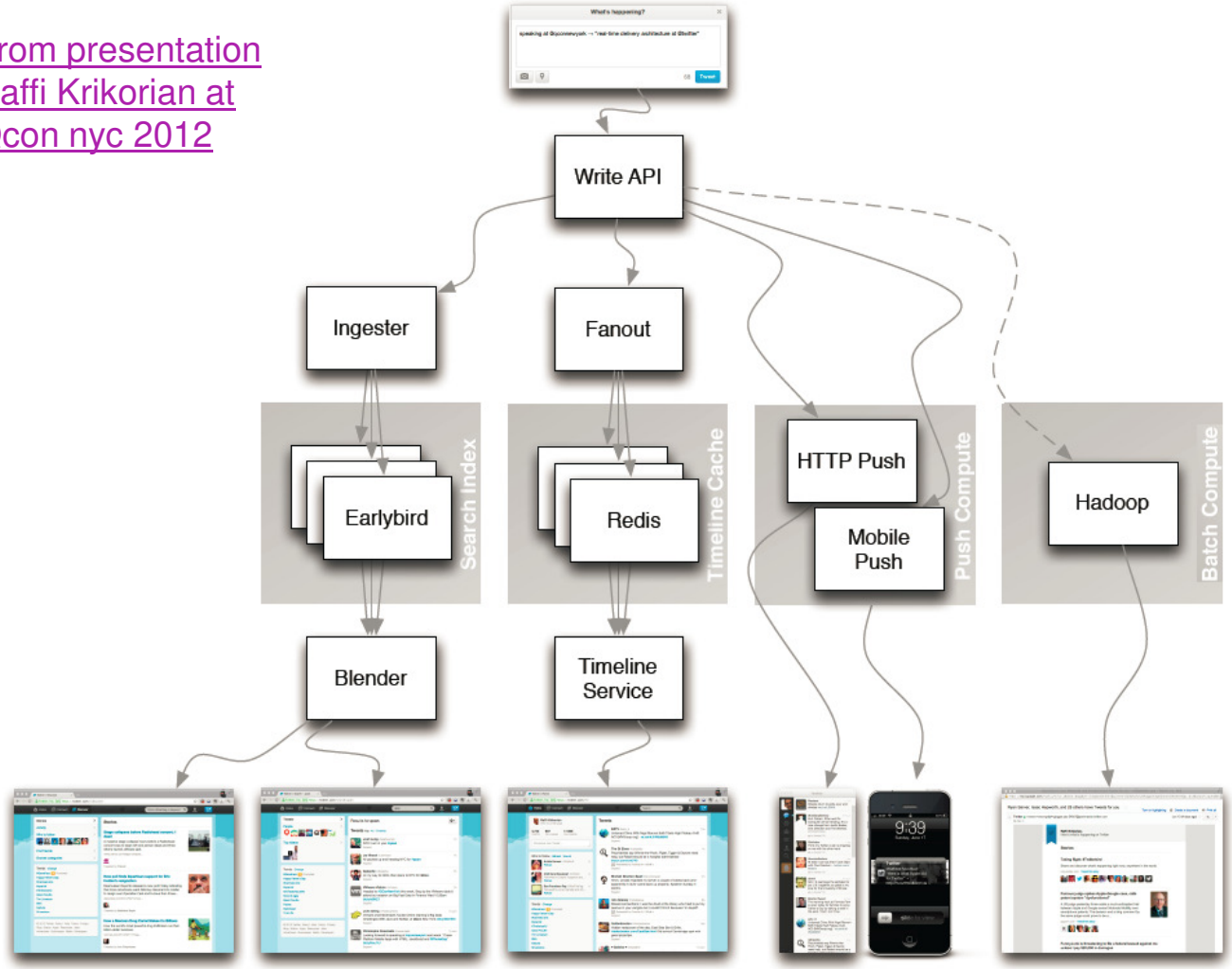
- describes the environment into which the system will be deployed, including dependencies the system has on its run-time environment
- is relevant for system administrators, developers, testers, communicators and assessors and addresses their concerns about
- hardware (processing elements, storage elements, and network), third-party software, and technology compatibility.

The operational viewpoint

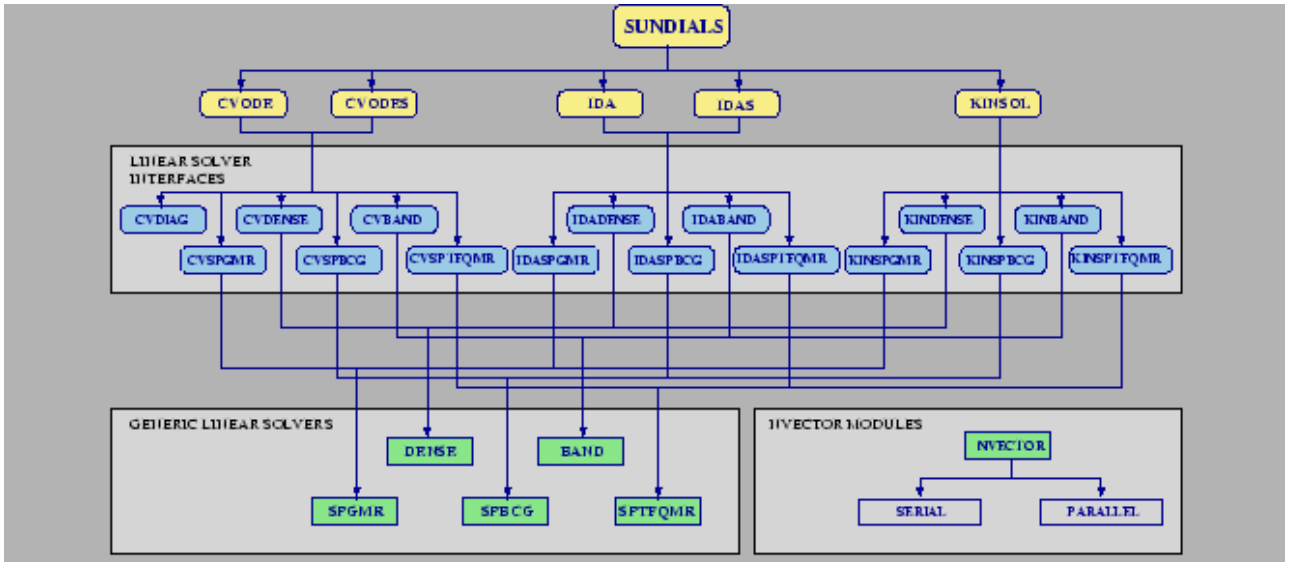
- describes how the system will be operated, administered, and supported when running in its production environment
- is relevant to system administrators, developers, testers, communicator, and assessors, and addresses their concerns about
- installation and upgrade, operational monitoring and control, configuration management, resource management.

Process (K), logical (K) information (RW) viewpoint

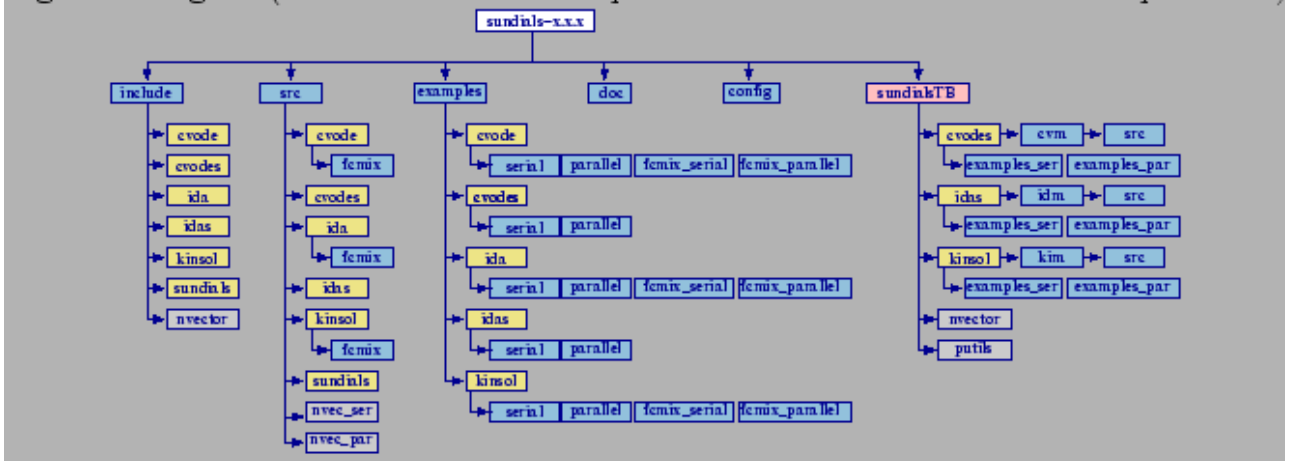
From presentation Raffi Krikorian at Qcon nyc 2012



Development (K + RW) viewpoint



High-level diagram (note that none of the Lapack-based linear solver modules are represented.)



Directory structure of the source tree

Taken from:

http://computation.llnl.gov/casc/sundials/documentation/idas_guide/

Process (K), Concurrency(RW); Deployment viewpoint (K + RW)

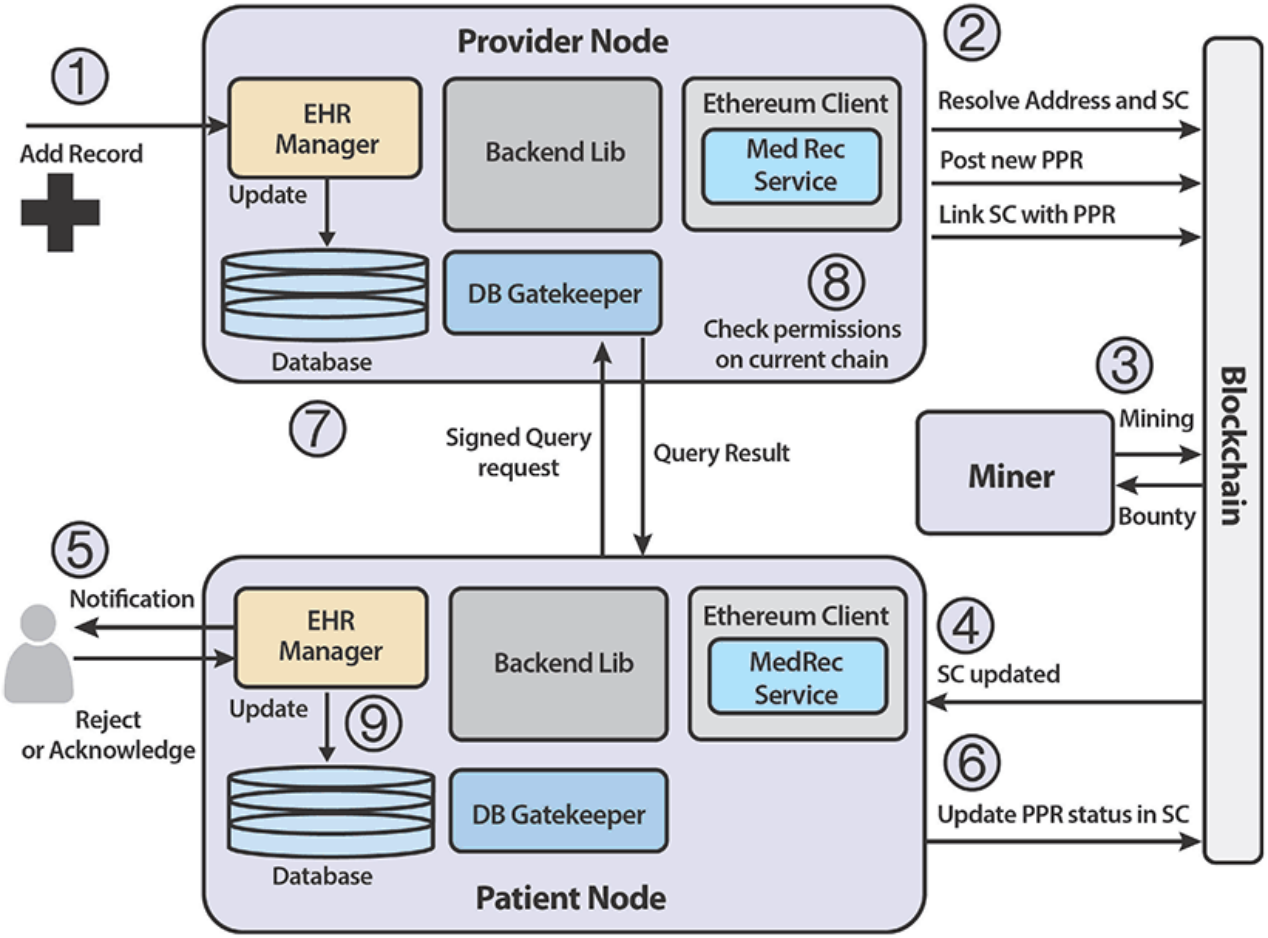
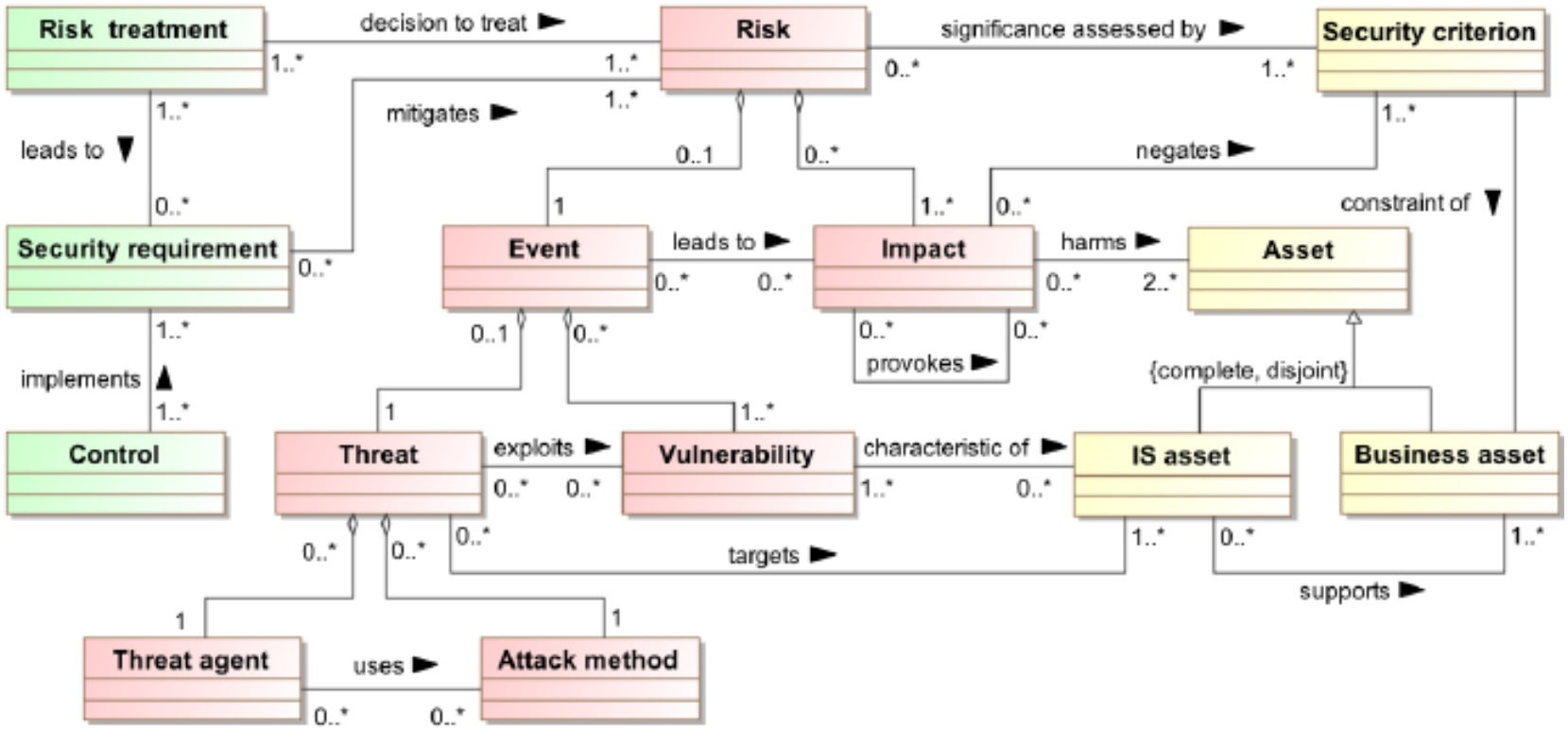


Figure 7. MedRec's Custom Blockchain Node Design (Eckblaw, Azaria, Hamalka, & Lippman, 2016)

Taken from:
https://www.freedassociates.com/insights/point-of-view/innovative-blockchain-uses-in-health-care/

Logical, functional, information, development viewpoint?

- Domain model: difficult to classify



Taken from: Towards Definition of Secure Business Processes

Extra-functional properties (EFRs)

- **Extra-functional concerns are addressed through the architecture**
  - Each stakeholder can come up with an extra-functional concern
    - e.g. security or performance
  - A very important general concern is *to limit dependencies*
- **These concerns often specify *emergent system properties (cross-cutting concerns)***
  - they arise from the collaboration of system components
  - e.g. security, performance
- **In the architecture, all these (conflicting) concerns are balanced**
  - the real challenge of architecting, not getting the functionality right
- **It is highly desirable that the architecture can be used to see if these concerns are met**
  - metrics, based on the architecture
  - transparency towards the final system realization(s)
- **Commonly also referred to as non-functional properties (NFRs)**

Quality Attributes (a.k.a. 'ilities')

Accessibility, Understandability, Usability, Generality, Operability, Simplicity, Mobility, Nomadicity, Portability, Accuracy, Efficiency, Footprint, Responsiveness, Scalability, Schedulability, Timeliness, CPU utilization, Latency, Throughput, Concurrency, Flexibility, Changeability, Evolvability, Extensibility, Modifiability, Tailorability, Upgradeability, Expandability, Consistency, Adaptability, Openness, Composability, Interoperability, Integrability, Accountability, Completeness, Conciseness, Correctness, Testability, Traceability, Coherence, Analyzability, Modularity, Reusability, Configurability, Distributeability, Availability, Confidentiality, Integrity, Maintainability, Reliability, Safety, Security, Affordability, Serviceabilility, …

Not all independent, can be organized into hierarchies

see e.g.: http://www.thomasalspaugh.org/pub/fnd/ility.html

ISO/IEC 25010

Classification of “ilities” according to the ISO/IEC 25010:2011 international standard on system and software quality requirements and evaluation

See https://iso25000.com/index.php/en/iso-25000-standards/iso-25010

RW: Perspectives

- **An (*architectural) perspective* is a collection of architectural activities, tactics and guidelines that are used to ensure that a system exhibits a particular set of related quality properties that require considerations across a number of system's architectural views**
  - i.e., address cross-cutting concerns
- **Perspectives are for QAs, what viewpoints are for views**
- **RW-perspective catalog (a selection)**
  - Security, Performance and scalability, Availability and resilience, Evolution, ...
- **See also: http://www.viewpoints-and-perspectives.info/home/perspectives**

ISO/IEC/IEEE 42010: New concepts

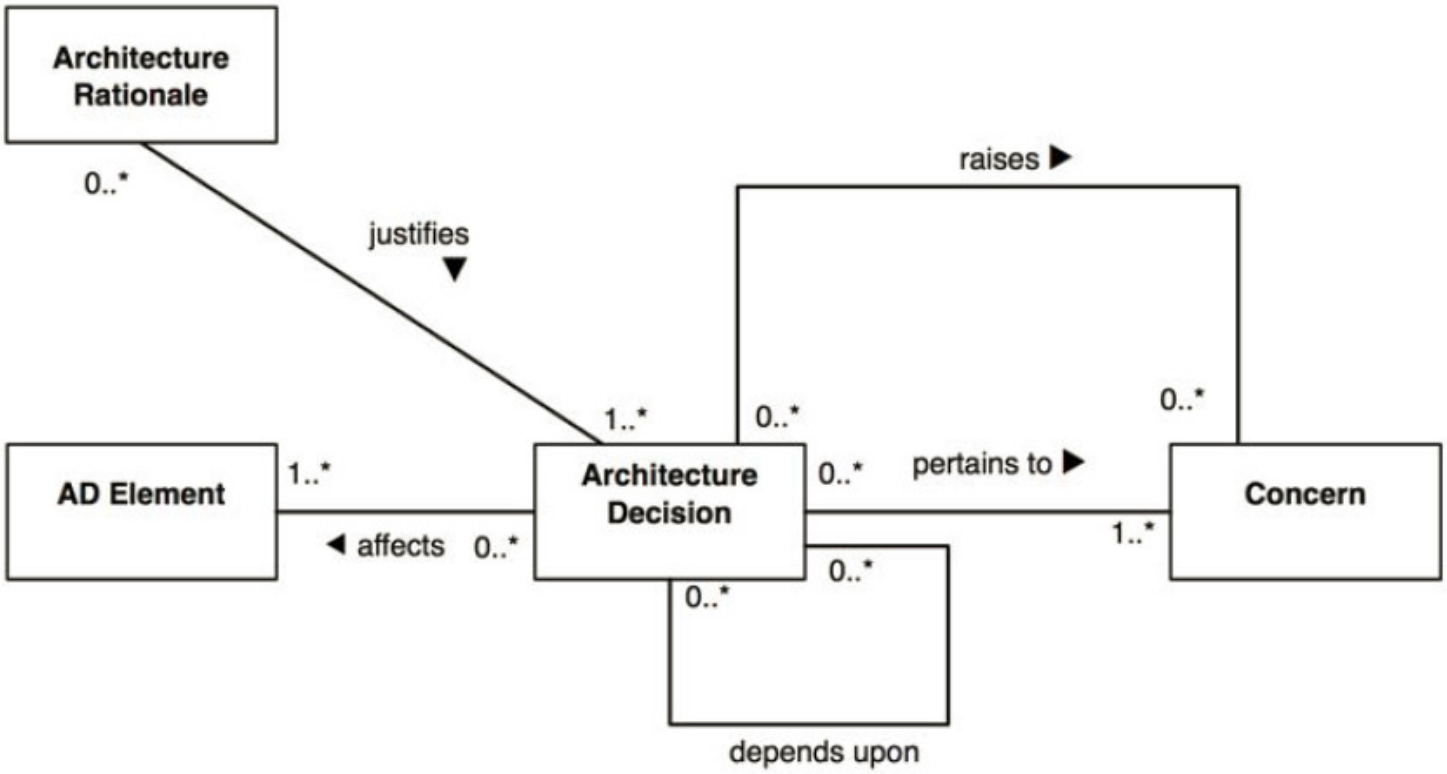
Architecture rationale **records explanation, justification or reasoning about architecture decisions that have been made**

- the basis for a decision
- alternatives and trade-offs
- consequences of a decision
- citations to sources of additional information

Decisions pertain to system concerns; however, there is often no simple mapping between the two. A decision can affect the architecture in several ways:

- requiring the existence of AD elements;
- changing the properties of AD elements;
- triggering trade-off analyses in which some AD elements, including other decisions and concerns, are revised;
- raising new concerns.

Architecture decision and rationale model



Architectural Reasoning-diagram

- Specifies a format for documenting design decisions
- Establishes the rationale for architectural entities (AEs)
  - visible in the top layer
  - necessary for the proper execution of scenarios
  - traceable through a chain of decisions to requirements of particular stakeholders
- Documents alternatives
  - also rejected ones

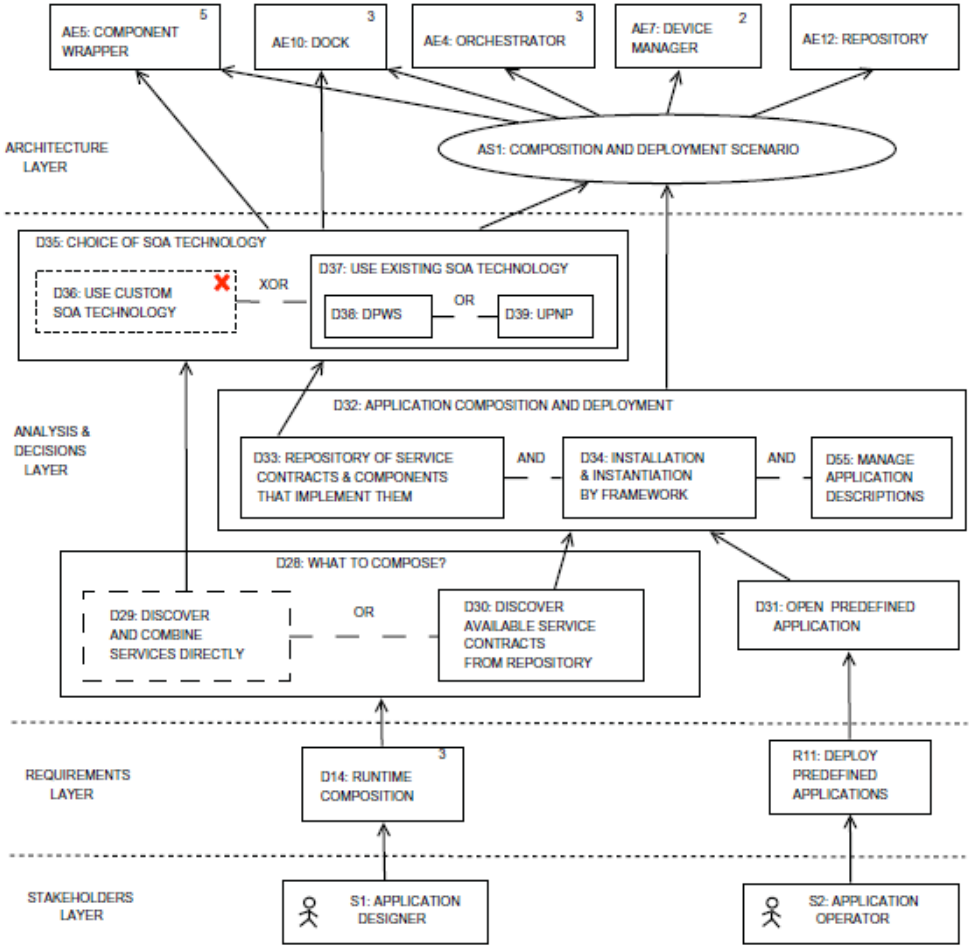
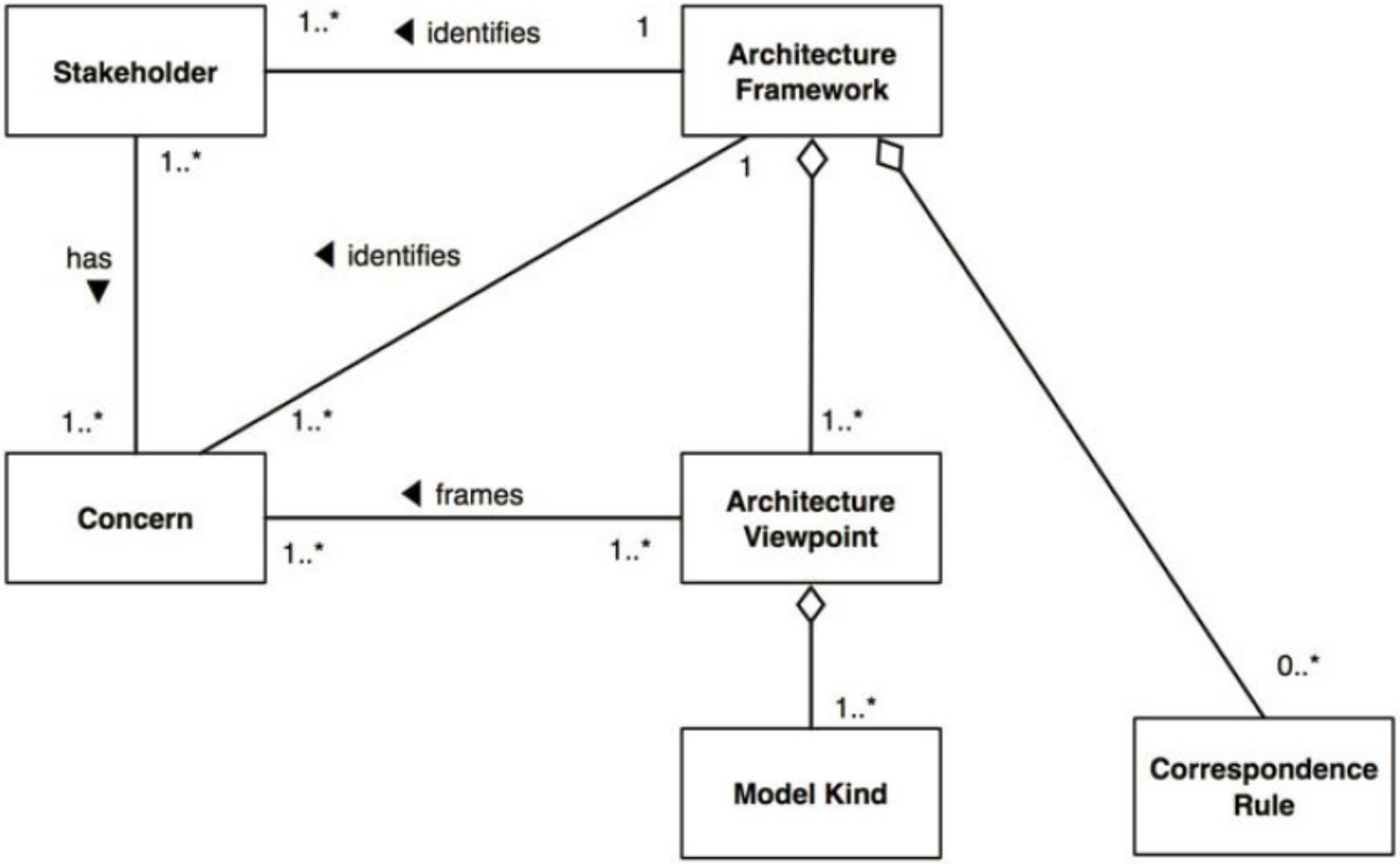


Figure 9: Decisions about usage of SOA technology

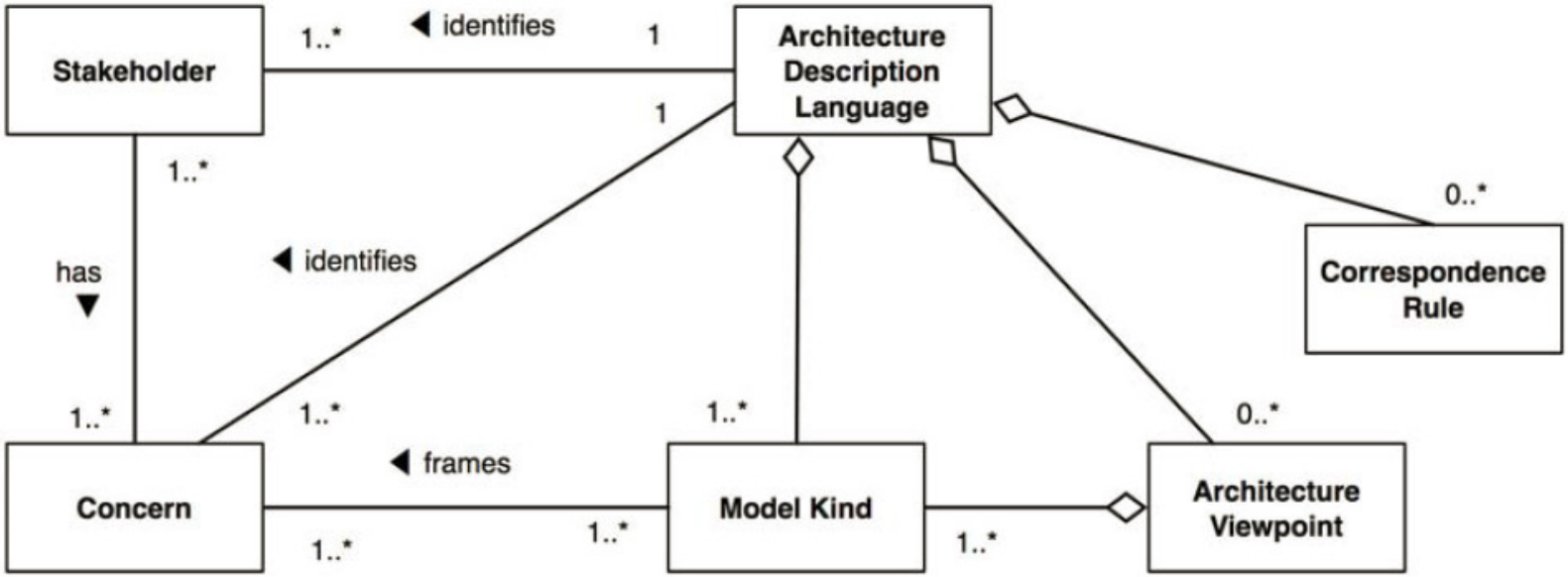
Architecture framework model



ISO/IEC/IEEE 42010: New concepts

- **An *architecture description language* (ADL) is any form of expression for use in architecture descriptions.**
- **An ADL provides one or more model kinds as a means to frame some concerns for its audience of stakeholders. An ADL can be narrowly focused, defining a single model kind, or widely focused to provide several model kinds, optionally organized into viewpoints. Often an ADL is supported by automated tools to aid the creation, use and analysis of its models.**
- **Examples: UML, SysML, AADL**

Architecture description language model



- Makes explicit which concerns are framed by what model kind
- Model kinds may have associated analysis methods and tools

UML views and diagrams

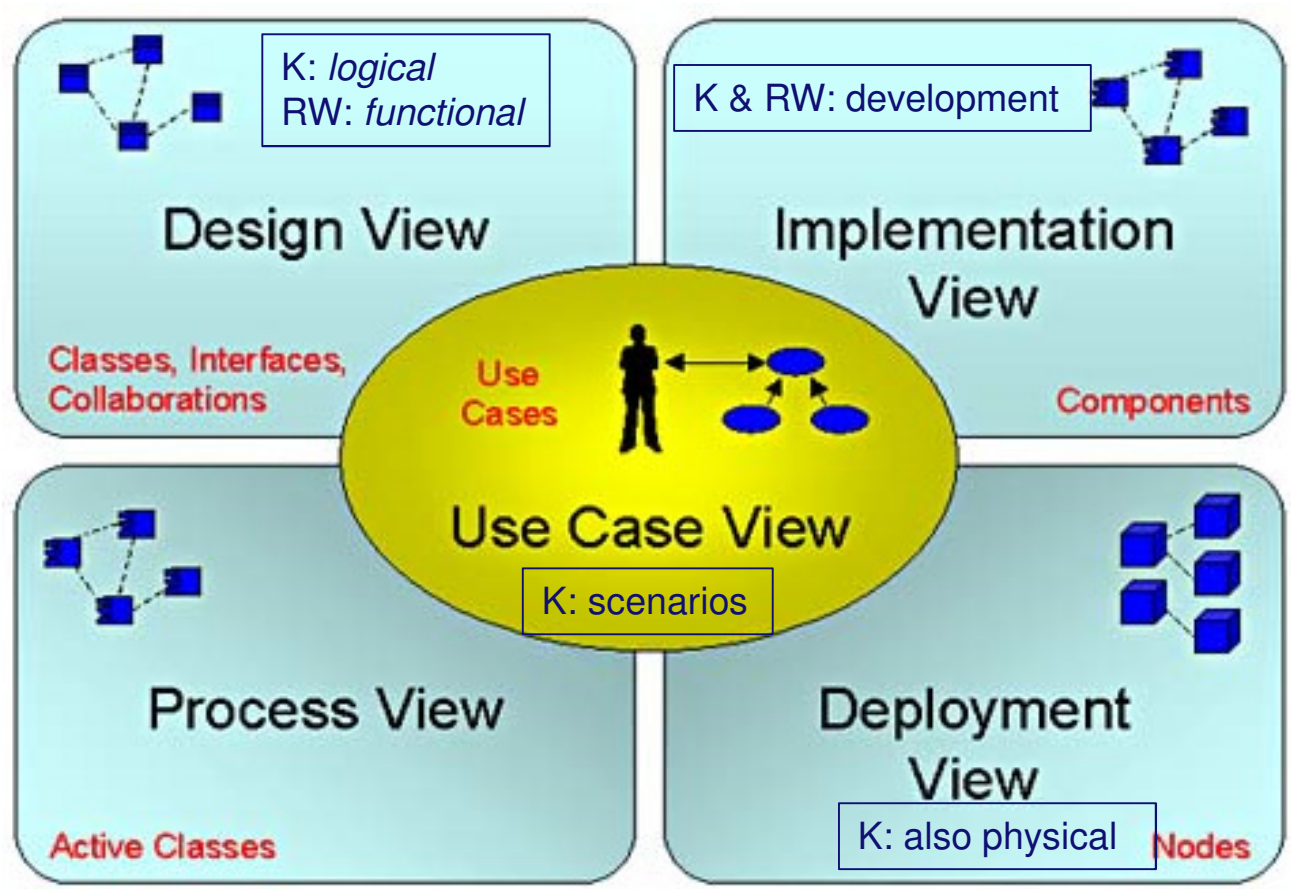


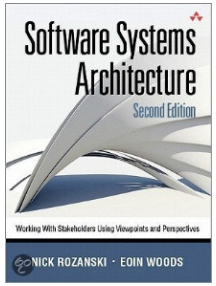
Diagram type
≈
Model kinds

- Class diagram
- Object diagram
- Component diagram
- Deployment diagram
- Use case diagram
- Sequence diagram
- Collaboration diagram
- State machine
- Activity diagram

Correspondence with K & RW only roughly correct

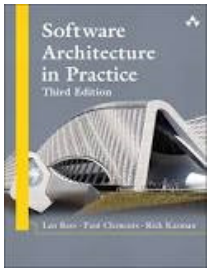
Literature

- Software Systems Architecture: Working with Stakeholders Using Viewpoints and Perspectives, 2ed, *Rozanski and Woods*, Barnes and Noble, 2011

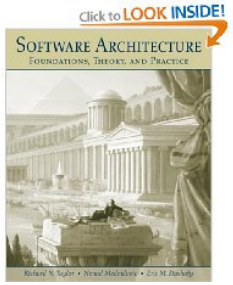


- http://www.viewpoints-and-perspectives.info/

- Software Architecture in Practice, Third Edition, L. Bass, P. Clements, R. Kazman, SEI Series in Software Engineering, Addison-Wesley, 2013



- Software Architecture, Foundations, Theory, and Practice, R.N. Taylor, N. Medvidovic, E.M. Dashofy, Wiley & Sons, 2009

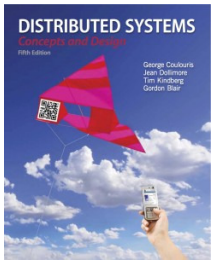


Literature

- **ISO/IEC/IEEE standard 42010-2011,**

Systems and software engineering – Architecture description ISO/IEC/IEEE STD 42010-2011

- **Distributed Systems: Concepts and Design, 5ed, Coulouris, Dollimore, Kindberg and Blair, Addison Wesley, 2011.**



- **Distributed Systems, 3rd ed., version 01, Maarten van Steen, Andrew Tanenbaum, 2017**