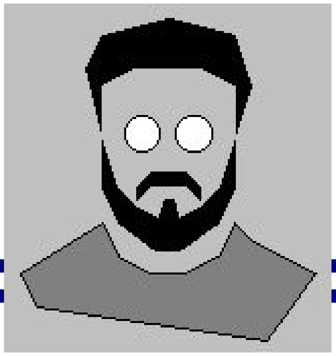
Software (Systems) Architecture Foundations

Lecture #1

Introduction

Alar Raabe

Alar Raabe



- Nearly 40 years in IT
  - held various roles from programmer to a software architect and to enterprise business architect
- 15 years in insurance and last 10 years in banking domain
  - developed model-driven technology for insurance applications product-line (incl. models, method/process, platform/framework and tools)
  - developing/implementing business architecture framework and methods for a banking group
- Interests
  - software engineering (tools and technologies)
  - software architectures
  - model-driven software development
  - industry reference models (e.g. IBM IAA, IFW, ...)
  - domain specific languages

Course Purpose

- Purpose – to provide understanding of
  - the core concepts in the discipline of software (systems) architecture
  - overview of different architecture styles
  - how software architecture affects quality attributes of the software systems
  - the value of software architecture and the architecture decisions
- Results
  - General understanding of the related concepts and techniques
  - Basic skills to
    - describe the architecture of software systems
    - evaluate the architecture of software systems
    - reason upon architectural decisions
    - organize the architecture work and governance

How we Work

- Weeks #09 ÷ #16
  - Monday 16:00-17:30 lecture (room EIK 221)
  - Tuesday 16:00-17:30 practice (room EIK 221)
- Practical work – happens in teams
  - Each team selects a software system, and
    - Designs architecture for it (studying the alternatives and selecting best)
    - Describes the architecture from selected viewpoints (to support stakeholders)
    - Analyses the architecture for selected quality attributes (ensuring that these will be met)
    - Presents their architecture with reasoning and analysis results (in written form)
- Evaluation
  - Presentation and defense of description and analysis of the architecture of the selected software system, done during practical work and off classes

Course Content

	Lecture	Practical Work
1	**Software (Systems) Architecture** what we call as architecture, how we speak of it and why architecture matters (that is why we need to design it)	distribution of subjects context of system stakeholders and their concerns
2	**Architecture Styles** what is architecture style, classification and analysis of main architecture styles, derived (complex) architecture styles, designing an architecture style (REST)	alternative system architectures
3	**Documenting Software Architectures** stakeholders, viewpoints and views, architecture decisions, architecture description languages, architecture in code	selecting viewpoints and views
4	**Evaluating Architectures** software quality attributes, evaluation of architectures (ATAM, CBAM, ...), cost and value of architecture decisions	important quality attributes selecting evaluation scenarios evaluating alternative architectures
5	**Larger Context** Systems-of-systems, architecture levels, enterprise architecture, enterprise architecture frameworks (TOGAF) (industry) reference architectures (BIAN)	selecting best architecture elaborating and documenting architecture
6	**From one System to Many** product-line architectures, model-driven development	supporting the systems family elaborating and documenting architecture
7	**Role of Architect** role of architect, architecture work and governance, architecture in the context of agile development	architecture function of an enterprise presentations of group-work
8	**(Boldly) to the Future** architecting for cloud, adaptive systems, AI systems & neural networks	presentations of group-work

Subjects for Practical Work

1. Cloud-ready product/agreement management system (i.e. ledger) for financial services industry
2. Software for mobile (smart-)phone
3. Automatic (intelligent) financial trading system
4. Fire control system for mobile (land or maritime) vehicle
5. Customer relationship management system with AI
6. Flight control (fly-by-wire) software for drone
7. Software for network of autonomous sensors (based of IoT)
8. Fault-tolerant and secure communication (chat) system
9. VR/AR role-playing or action game
10. Software for package delivery robot

- … or your own proposal ? ...

(Some) Sources

[1] Mary Shaw, David Garlan, **Software Architecture, Perspectives on an Emerging Discipline**, Prentice Hall,1996

[2] Len Bass, Paul Clements, Rick Kazman, **Software Architecture in Practice**, 2nd ed., Addison-Wesley, 2003

[3] Paul Clements, Felix Bachmann, Len Bass, David Garlan, James Ivers, Reed Little, Robert Nord, Judith Stafford, **Documenting Software Architectures, Views and Beyond**, Addison-Wesley, 2003

[4] PaulClements, Rick Kazman, Mark Klein, **Evaluating Software Architectures, Methods and Case Studies**, Addison-Wesley, 2001

[5] R. T. Fielding, **Architectural Styles and the Design of Network-based Software Architectures**, UCI, 2000

[6] ISO/IEC/IEEE 42010, **Systems and software engineering — Architecture description** (IEEE 1471)

[7] Open Group, **TOGAF** 9

[8] Open Group, **ArchiMate** 3

- … Google “Software Architecture” ...

Content

Software architecture is what software architects do

Kent Beck

- Intro to the Software (Systems) Architecture
  - What we call Architecture
  - Why we need to bother with Architecture
  - Design vs. Architecture
  - Early Views and Software Architecture
- Software Architecture (Description) related Concepts and Terminology (IEEE 1741 | ISO/IEC 42010)
- Other related Concepts
  - Abstraction
  - Complexity
  - Modularity
  - Model
- Conclusions

What we call Architecture

Architecture is about:

❖ Durability (*firmitas*)
❖ Utility (*utilitas*)
❖ Beauty (*venustas*)

Marcus Vitruvius Pollio
(Rome, 1st century BC)

- Merriam-Webster :: Architecture (n)
  - art or science of building
  - unifying or coherent form or structure
  - manner in which the components of the system are organized and integrated
- Wikipedia :: Systems Architecture
  - the conceptual model that defines the structure, behavior, and more views of a system
- Wikipedia :: Software Architecture
  - refers to the high level structures of a software system, the discipline of creating such structures, and the documentation of these structures

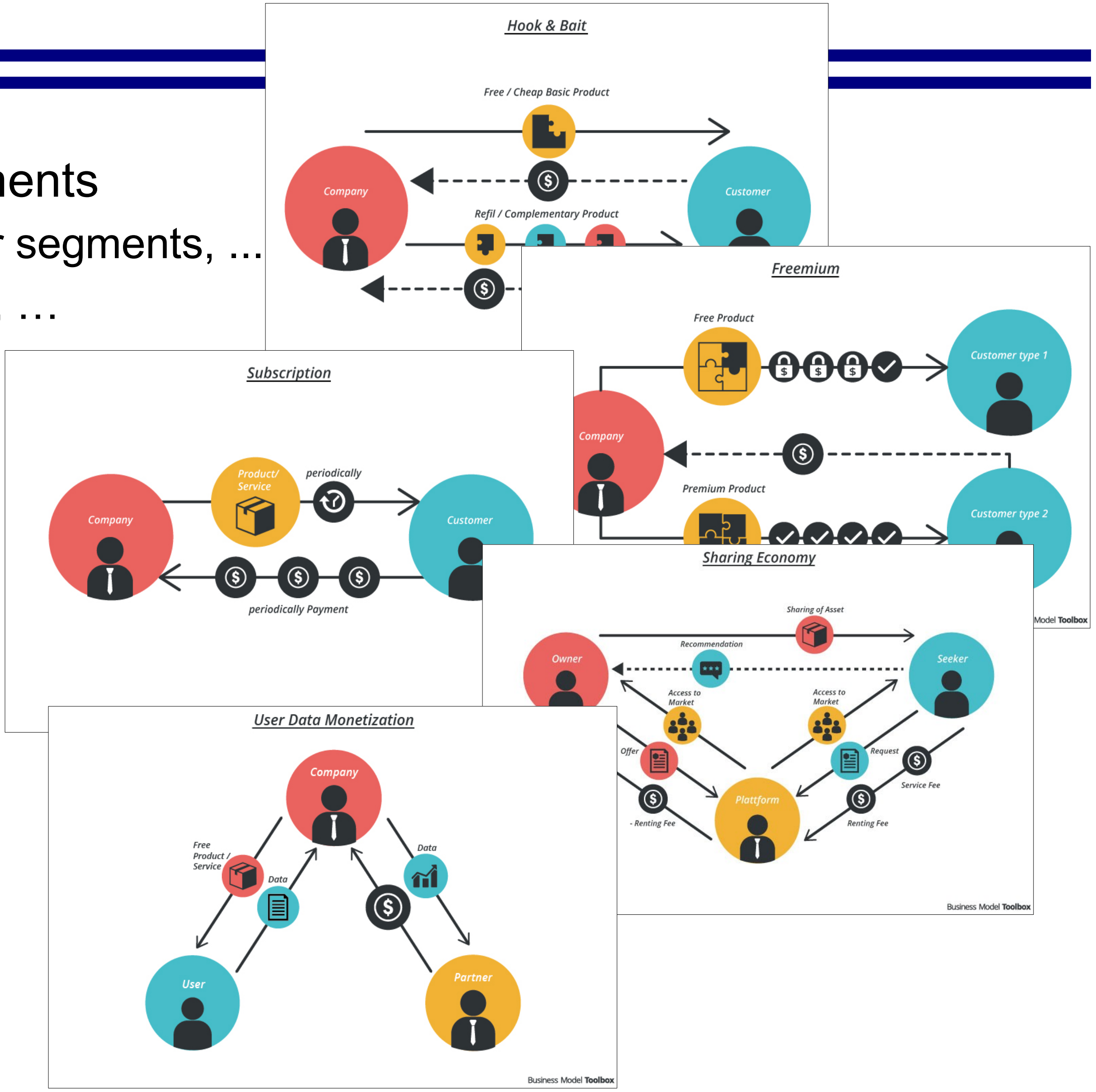
Building Architecture

- Parts or Components or Elements
  - Walls, roofs, …
  - Doors, windows, …
  - ...
- Connections or Relationships
  - On top of, inside of
  - ...
- Principles or Rules
  - Transfer weight to ground
  - Replace tension with compression
  - ...

Pictures © Wikipedia & Wikimedia Commons

Business Architecture

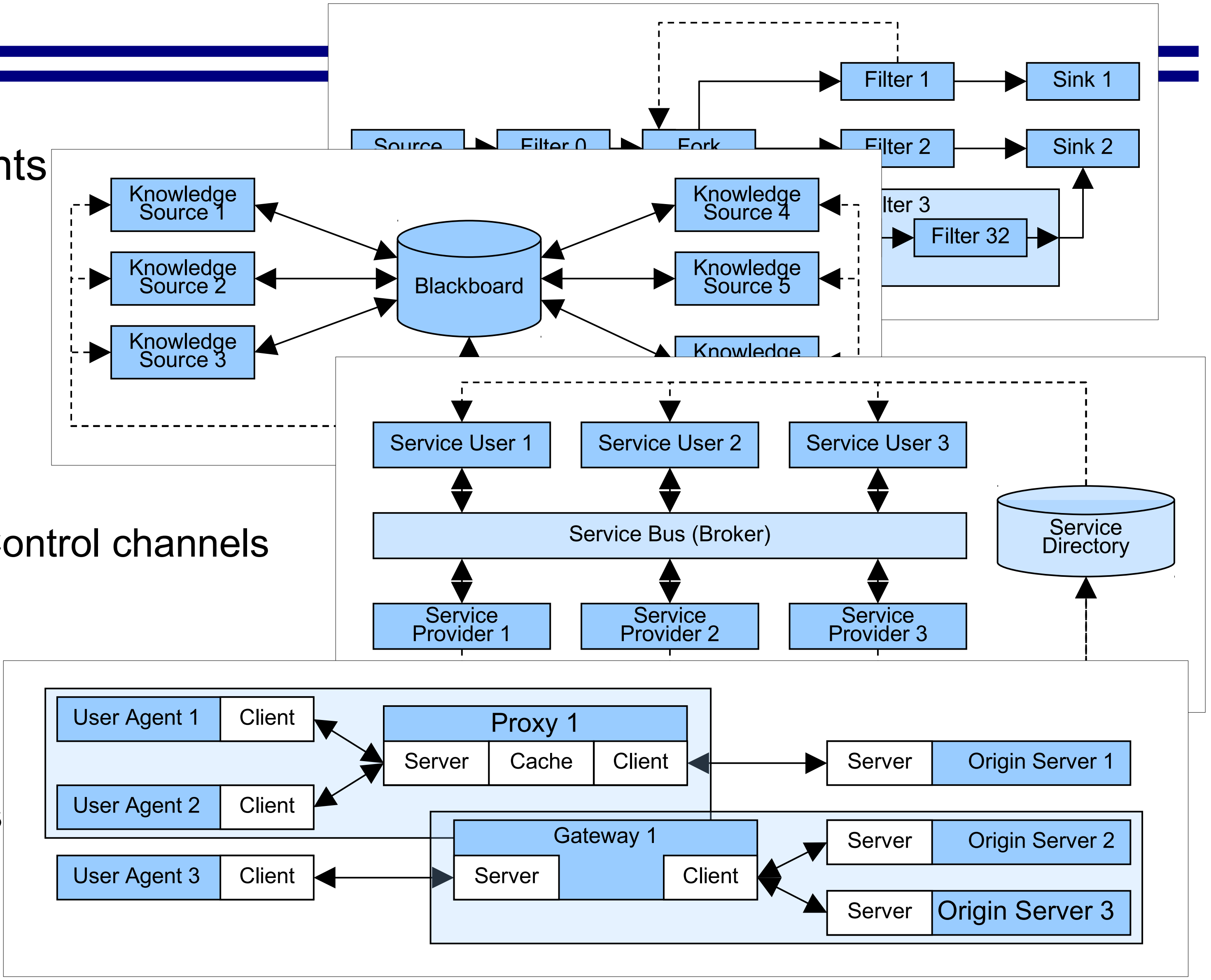
- Parts or Components or Elements
  - Value propositions, customer segments, …
  - Key activities, key resources, …
  - …
- Relationships
  - Channels, …
  - Customer relationships, supplier relationships, …
  - …
- Rules
  - Income should cover costs
  - …



Pictures © Business Model Toolbox

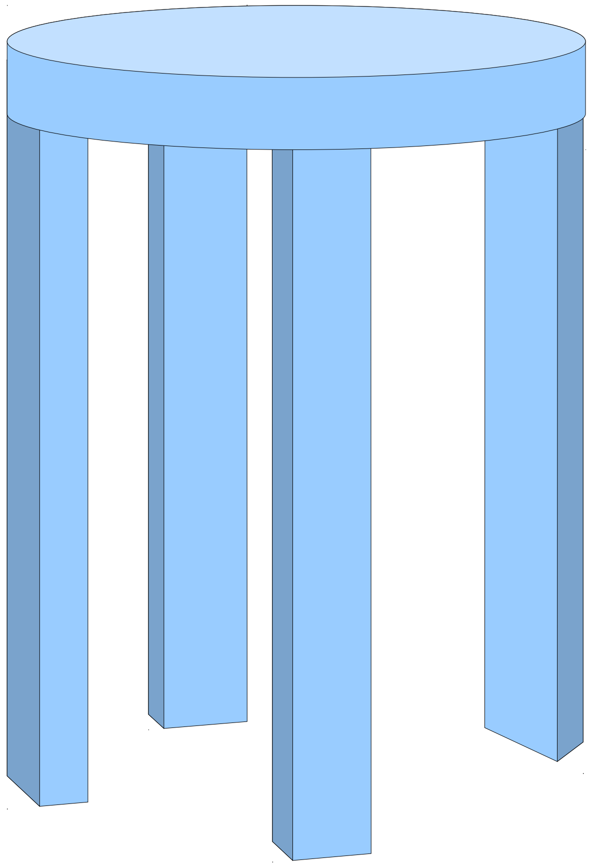
Software Architecture

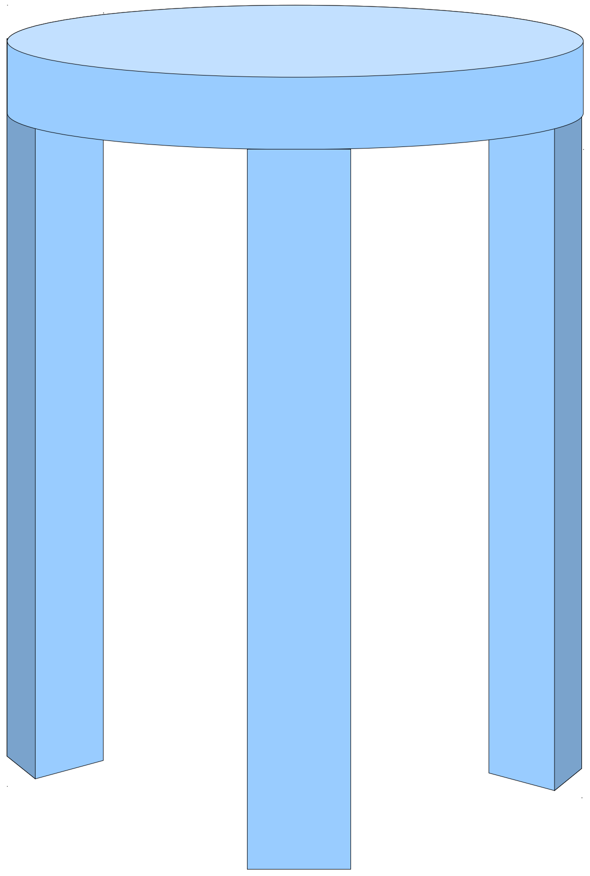
- Parts or Components or Elements
  - Processors
  - Data-stores
  - …
- Relationships
  - Data channels, Control channels
  - Interfaces
  - …
- Rules
  - Stateless servers
  - …



Two kinds of stools

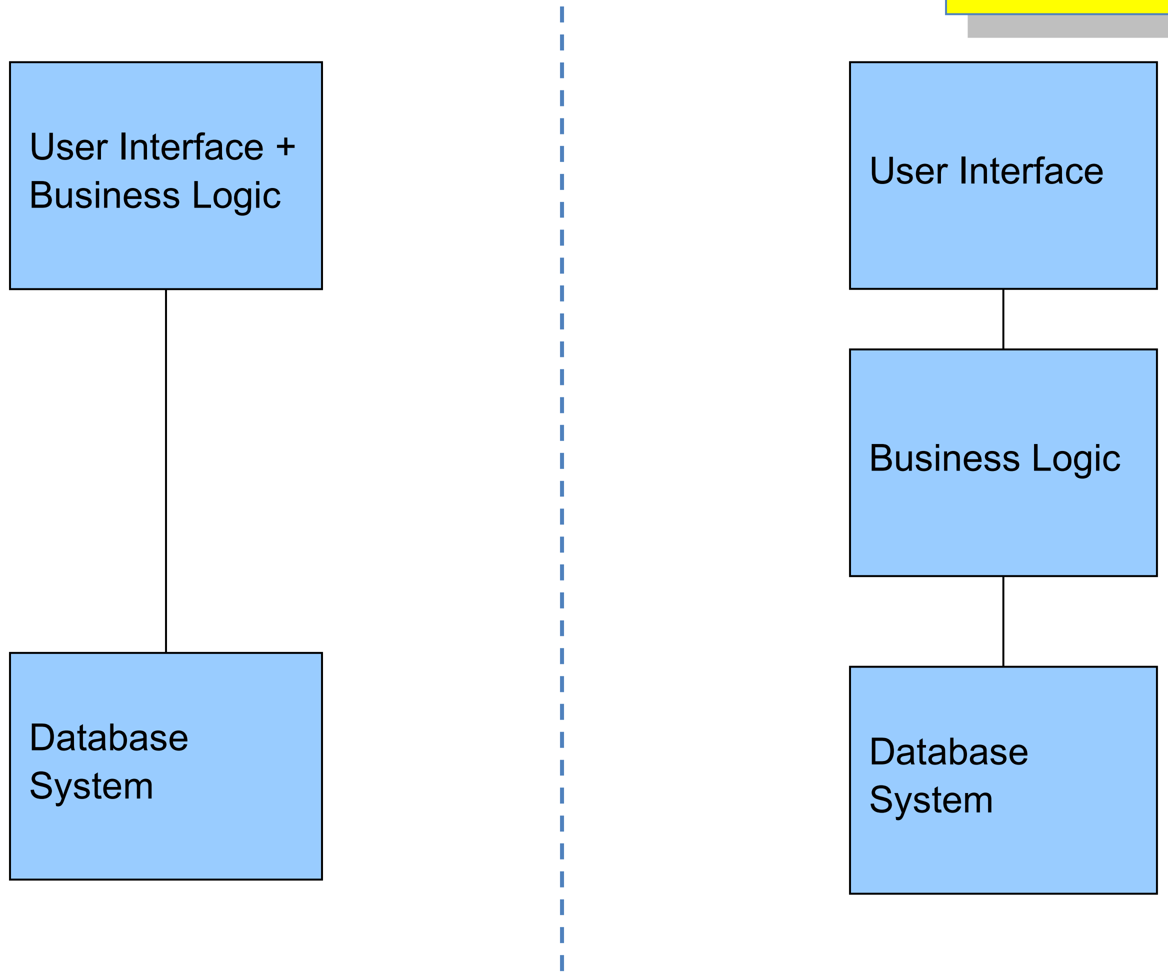
What's the difference ?
How many legs is good ?





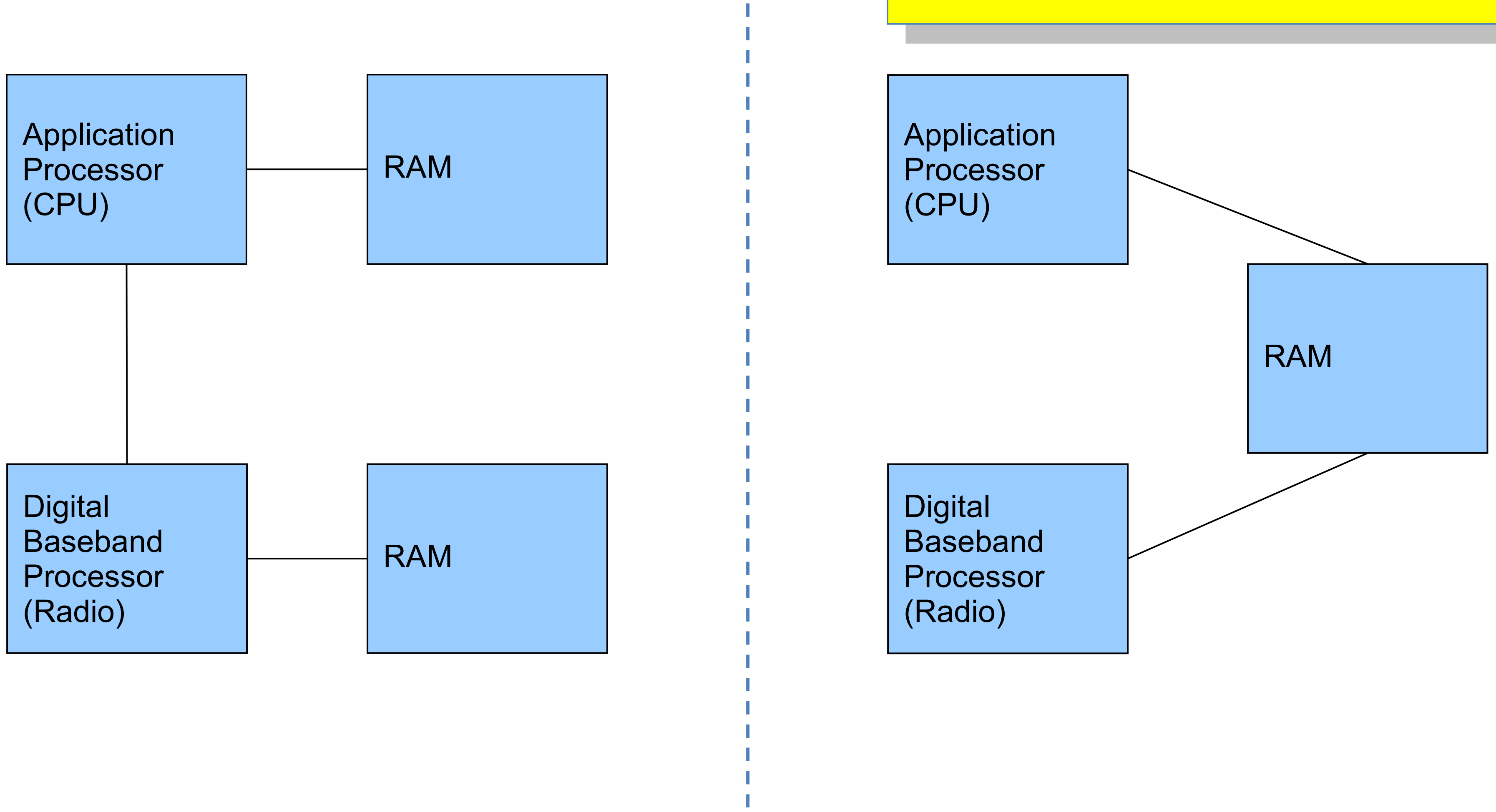
Two kinds of web applications

How many layers/tiers is good ?
What's the difference ?
During runtime or coding?



Two kinds of smart-phones

Do you know which one is yours ?
Do you care or should you care ?
Who cares which one is used ?



Application Processor (CPU)
RAM
Digital Baseband Processor (Radio)
RAM
Application Processor (CPU)
RAM
Digital Baseband Processor (Radio)

Emerging Architecture

LAGERFELD
Guy Laroche
Häagen-Dazs

Which one you would like to live in ?

Pictures © Wikipedia & Wikimedia Commons

Why we need to bother with Architecture ...

- Architecture is the **carrier of certain properties** of the system
  - **Same function, but different architecture → different properties / qualities**
- But **architecture** with desirable properties doesn't emerge itself, it **needs to be designed**
  - **Designing the architecture allows to reach certain, desirable, properties**
  - **Knowing the architecture allows to reason about some of the properties** of the system, and foresee those properties (without building and testing the actual system)

Design vs. Architecture

> All architecture is design but not all design is architecture
>
> Grady Booch

- Design = Plan
  - adaptation of means (*what we have*) to ends (*what we want*)
- Software Design can be viewed on many levels
  - design of higher levels is architecture for the lower levels
- G. Booch
  - architecture represents significant design decisions that shape a system, where *significant is measured by cost of change*
- A. H. Eden
  - Architectural decisions and specifications are
    - **Intensional**
      (*generic → applicable to many implementations*)
    - **Non-local**
      (*applicable to entire system*)

The Intension / Locality Thesis		
Non-Local	**Intensional**	**Architecture**
Local	Intensional	Design
Local	Extensional	Implementation

Early Views on Software Architecture

Reuse & Appearance
Structure Matters

- Ada Augusta, Countess of Lovelace (1842)
  - described the idea of a set of reusable instructions for Analytical Engine
- A. M. Turing & D. Wheeler (1946-50)
  - reuse of program code and modularization – (closed) subroutine
  - subroutine library (reusability, reliability, unit testing (testability), multiple versions with different non-functional qualities, …)
- K. E. Iverson, M. E. Conway & F. Brooks (1964-69)
  - architecture is a conceptual structure
  - system (architecture) design follows organization → Conway's Law
  - architecture is the complete and detailed specification of the user interface (!)
- E. W. Dijkstra, D. L. Parnas & M. A. Jackson (1972-76)
  - separation of concerns – isolation, encapsulation, modularization
  - program families can be described by a decision trees
  - ***structure influences non-functional 'qualities' of system***
  - ***structure of program is defined by domain structures***

Decomposition Criteria

Begin with a list of difficult design decisions or design decisions which are likely to change

D. L. Parnas (1971)

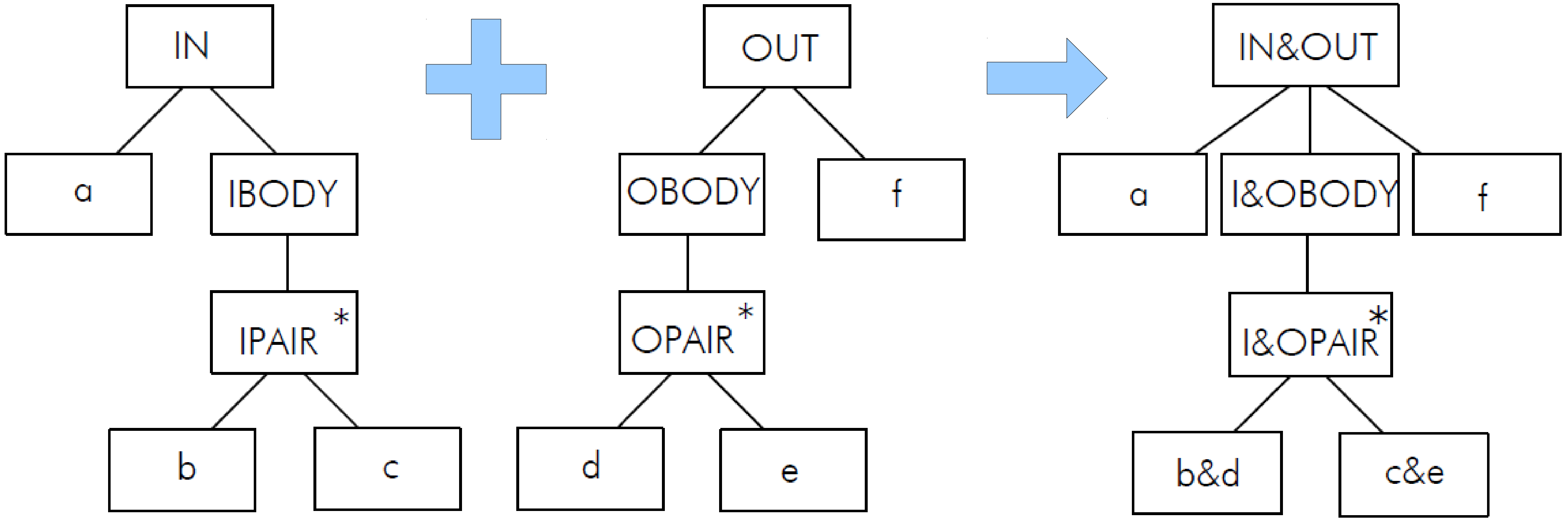
- Differences between the systems depend on how they are divided into modules
  - Systems can differ substantially even the runnable representation is same, because other representations are used for changing, documenting, understanding, etc.
  - Properties that vary depending on decomposition
    - Changeability
    - Independent development
    - Comprehensibility
    - Efficiency (of implementation)
- Criteria used to get systems with different properties
  - “major steps” (of algorithm)
  - “information hiding”
  - hierarchical structure
- Begin with a list of difficult design decisions or design decisions which are likely to change and design each module to hide such a decision from the others

Jackson Structured Programming (JSP) and System Development (JSD)

Domain Structures define Software Architecture

- Program structure should be dictated by the structure of its input and output data streams (Jackson 1975)

- Development must **begin by describing and modeling the real world**, not by specifying, describing or structuring the function which the system is to perform → a system developed according to the JSD method embodies an explicit simulation of the real world (Jackson 1981)



Pictures © Michael Jackson

4.6.18

Software Architecture as Discipline

elements + form/structure + rationale/principles
(*what*) (*how*) (*why*)

- D. E. Perry & A. L. Wolf (1992)
  - Software Architecture = { Elements, Form, Rationale }
    - a set of architectural (or, if you will, design) ***elements*** that have a particular form (of three different classes: processing, data, and connecting elements),
    - architectural ***form***, consisting of weighted properties and relationships, and
    - ***rationale*** for various choices made in defining an architecture
- D. Garlan & M. Shaw (1994)
  - a collection of computational components – or simply ***components*** – together with a description of the interactions between these components – the ***connectors***
- L. Bass, P. Clements, R. Kazman (1997)
  - the structure or structures of the system (comprise software ***components***, their externally visible properties, and the ***relationships*** among them)
- A. H. Eden, R. Kazman (2003)
  - strategic design decisions/statements – global design decisions/constraints

Agile and Software Architecture

Architecture is the important stuff – whatever that is

Ralph Johnson

- K. Beck (2000)
  - expressed in XP through system metaphor, which “helps everyone to understand basic elements and their relationships”
  - should be created by first iteration
- R. Johnson (2002)
  - a **shared understanding** of the system design of the expert developers working on the project (incl. how the system is divided into components and how the components interact through interfaces)
  - the **decisions** that you wish you could get right early in a project
- M. Fowler (2003)
  - a word we use when we want to talk about design but want to puff it up to make it sound important
  - things that people perceive as hard to change → if everything is simple to change there’s no architecture!

Software Architecture – a Set of Structures

CMU SEI Software Architecture

- The software architecture of a system is the set of structures **needed to reason about the system**, which comprise software elements, relations among them, and properties of both
  - **Architecture is a Set of Software Structures**
    - Software systems are composed of many structures, and no single structure holds claim to being the architecture
  - **Architecture is an Abstraction**
    - Architecture specifically omits certain information – selects certain details and suppresses others
  - **Every software system has a Software Architecture**
    - Even though every system has an architecture, it does not necessarily follow that the architecture is known to anyone
  - **Architecture includes Behavior**
    - The behavior of each element is part of the architecture insofar as that behavior can be used to reason about the system, and embodies how elements interact with each other
  - **Not all architectures are Good Architectures**
    - This raises the importance of architecture design, which is, and architecture evaluation

Three Kinds of Architecture Structures

CMU SEI Software Architecture

- **Module structures** – embody decisions as to how the system is to be structured as a set of code or data units that have to be constructed or procured
  - Represent a static way of considering the system (independent of how the resulting software manifests itself at runtime)
  - Modules are assigned areas of functional responsibility
- **Component-and-connector structures** – embody decisions as to how the system is to be structured as a set of elements that have runtime behavior (components) and interactions (connectors)
  - Components are the principal units of computation
  - Connectors, which are the communication vehicles among components
- **Allocation structures** – embody decisions as to how the system will relate to non-software structures in its environment
  - These structures show the relationship between the software elements and elements in one or more external environments in which the software is created and executed

Importance of Architecture

CMU SEI Software Architecture

- Architecture defines
  - the earliest and most fundamental (hardest-to-change) design decisions
  - a set of constraints on implementation
  - system's quality attributes
- Architecture
  - dictates the structure of an organization (that is building the system), or vice versa
  - reduces design and system complexity (by restricting design alternatives and channeling creativity)
  - focuses attention on the assembly of components, rather than simply on their creation
  - can be a transferable, reusable model that forms the heart of a family of systems (product line)
- Architecture description
  - allows to reason and predict of system's qualities
  - allows to reason about, and manage the change as the system evolves
  - allows to reason about cost and schedule – supports project management
  - enhances communication among stakeholders – provides common understanding of the system
  - can be the foundation for training a new team member

Software Architecture (in) Standards

architecture ≠ architecture description

- Open Group TOGAF 9 Enterprise Architecture Framework
  - a formal description of a system, or a detailed plan of the system at component level to guide its implementation
  - the structure of ***components***, their ***interrelationships***, and the ***principles and guidelines*** governing their design and evolution over time

- IEEE 1741 | ISO/IEC 42010 Systems and Software Engineering – Architecture Description
  - the fundamental conception of a system in its environment embodied in ***elements***, their ***relationships*** to each other and to the environment, and ***principles*** guiding system design and evolution

  - Architecture descriptions are for …
    - *Communicating* among the system's stakeholders
    - *Planning and Managing* system development and operations
    - *Evaluating and Comparing* systems architectures, and verifying system's implementation for compliance with its intended architecture

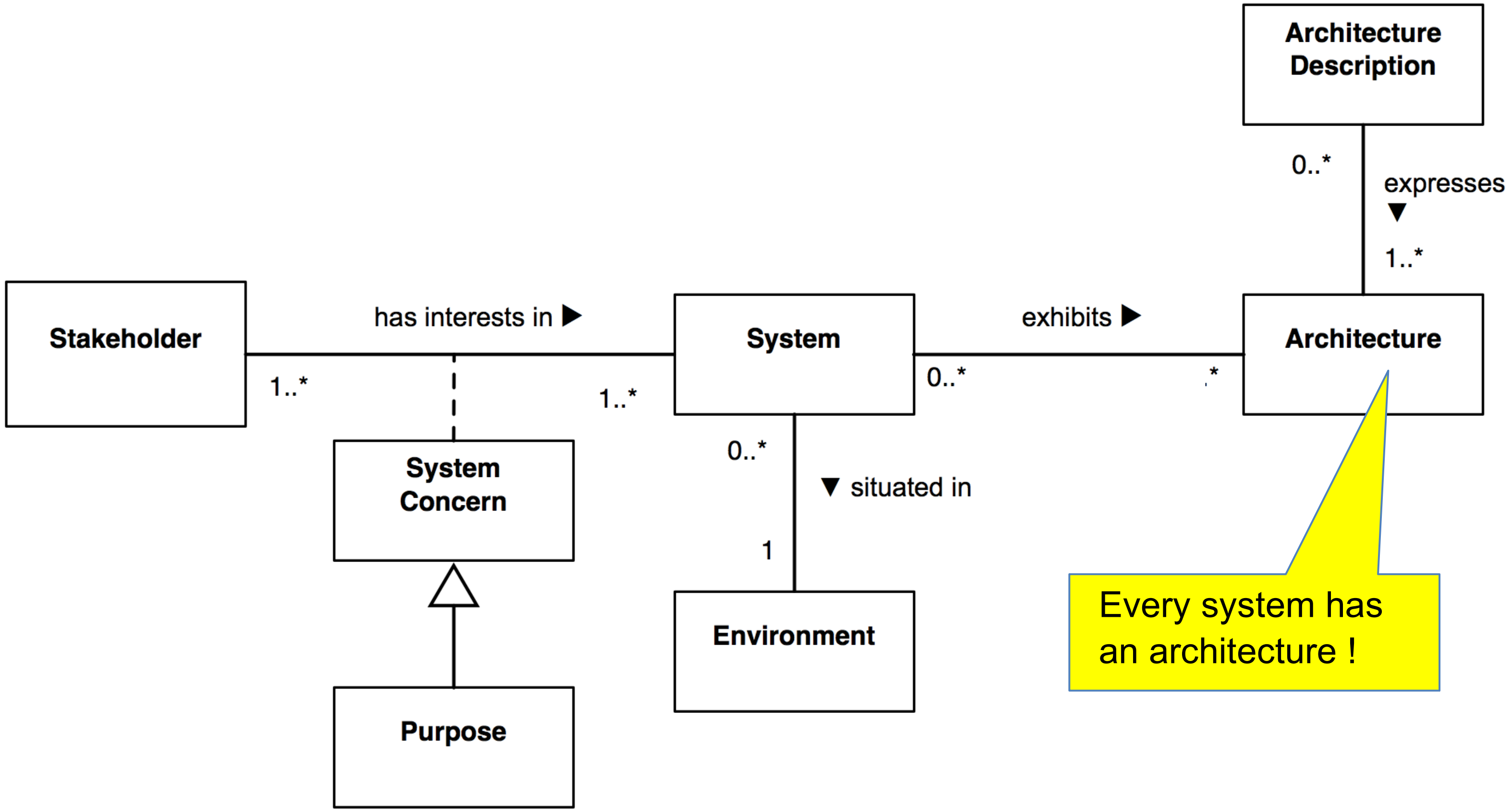
Content

> Software architecture is what software architects do
>
> Kent Beck

- Intro to the Software (Systems) Architecture
  - What we call Architecture
  - Why we need to bother with Architecture
  - Design vs. Architecture
  - Early Views and Software Architecture
- Software Architecture (Description) related Concepts and Terminology (IEEE 1741 | ISO/IEC 42010)
- Other related Concepts
  - Abstraction
  - Complexity
  - Modularity
  - Model
- Conclusions

System, Architecture and Architecture Description

ISO/IEC 42010



Every system has an architecture !

Pictures © ISO/IEC/IEE

System, Architecture and Environment

ISO/IEC 42010

- There are **Systems** and they are situated in their **Environment** (which could include other Systems)

- **System**
  - is used as a placeholder – it could refer to an enterprise, a system of systems, a product line, a service, a subsystem, or software
  - systems can be man-made or natural

- **Environment**
  - intended in the widest possible sense to include developmental, operational, technical, political, regulatory, and all other influences which can affect the architecture

- **Stakeholders** have interests in a System, called **Concerns** (one, very common of these, is considered as system's **Purpose**)

Examples of Stakeholders & Concerns

ISO/IEC 42010

- Stakeholders
  - users and operators
  - acquirers and owners
  - suppliers, developers, builders and maintainers

- Concerns
  - purposes of the system
  - suitability of the architecture for achieving the system’s purposes (and other stakeholders needs, goals and expectations)
  - feasibility of constructing the system
  - potential risks of the system to its stakeholders throughout its life cycle
  - maintainability, deployability, and evolvability of the system
  - compliance of system to design and other constraints

Architecture and Architecture Description

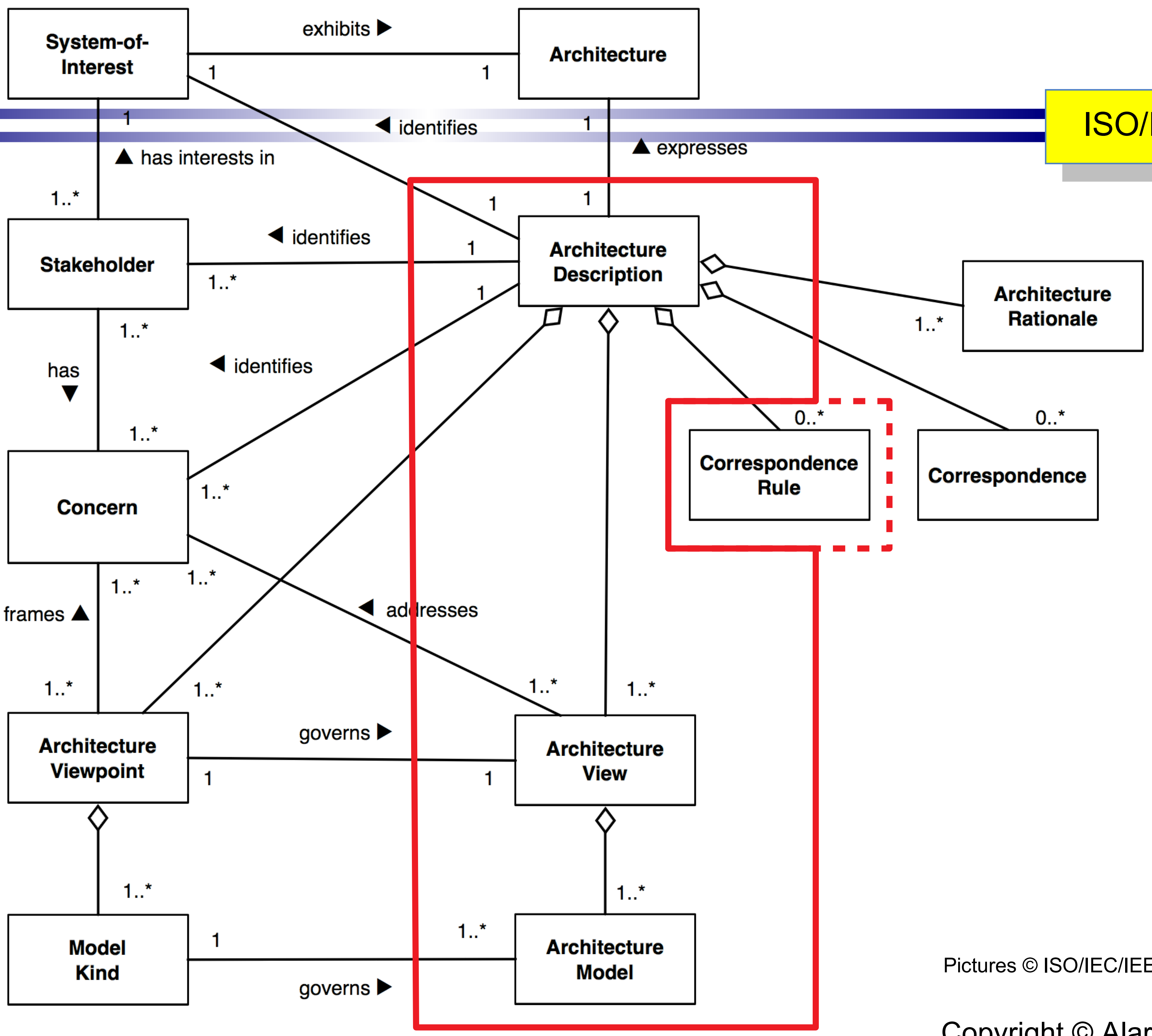
ISO/IEC 42010

- Systems have **Architectures** which are expressed by an **Architecture Description**

- **Architecture**
  - fundamental concepts or properties of a system in its environment embodied in its **elements**, **relationships**, and in the **principles** of its design and evolution
    - the architecture of X is what is fundamental to X (independent of what X is)
    - a system **can have an architecture even if** that architecture is **not written down**

- **Architecture Description**
  - a work product used to express the **Architecture** of some **System** Of Interest
  - it describes one possible **Architecture** for a **System** Of Interest
  - it may take the form of a document, a set of models, a model repository, or some other form
      - an artifact that expresses **Architecture** so that architects and other system stakeholders can understand, analyze and compare **Architecture**s, and can be used as “blueprint” for planning and construction

Architecture Description – a set of Views



ISO/IEC 42010

Pictures © ISO/IEC/IEE

Stakeholder, Architecture View & Model

ISO/IEC 42010

- **Stakeholder**
  - individuals, groups or organizations holding **Concern**s for the **System** of Interest
- **Architecture View**
  - expresses the **Architecture** of the **System** of Interest from the perspective of one or more **Stakeholders** to address specific **Concerns**, using the conventions established by its **Viewpoint**
  - consists of one or more **Architecture Models**
- **Architecture Model**
  - is constructed in accordance with the conventions established by its **Model Kind**, typically defined as part of its governing viewpoint
  - it provides a means for sharing details between views and for the use of multiple notations within a view

Architecture Viewpoint & Model Kind

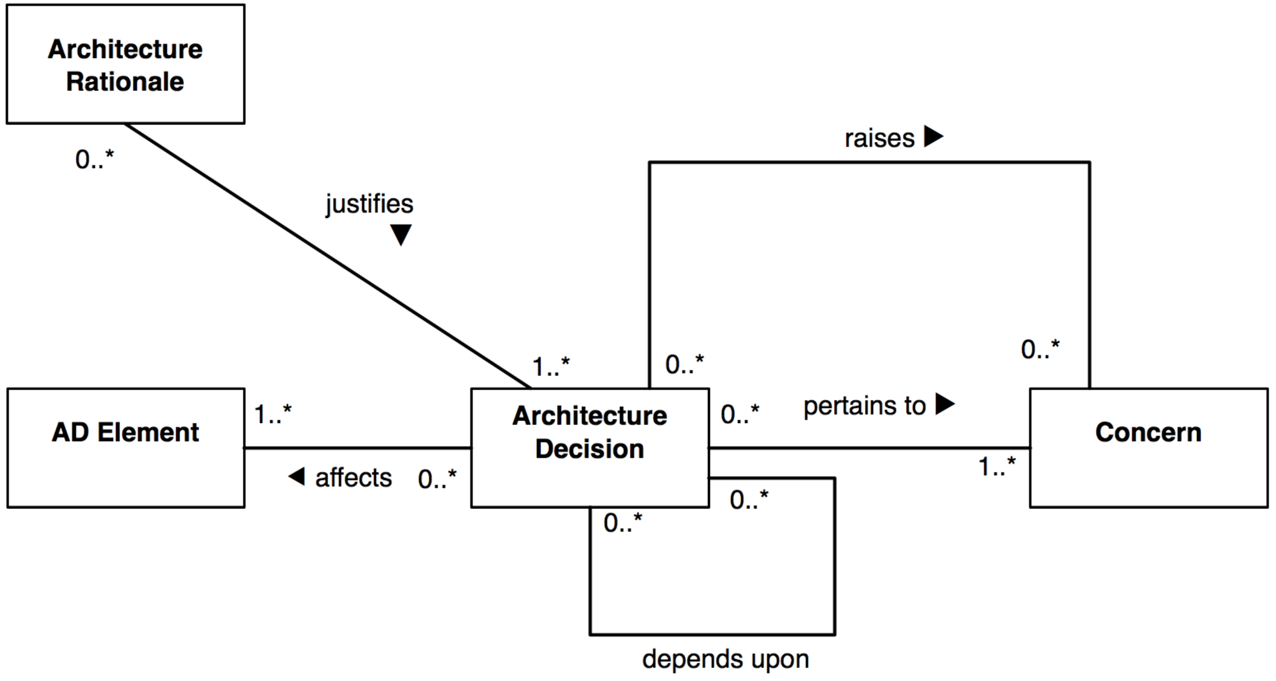
ISO/IEC 42010

- **Architecture Viewpoint**
  - a set of conventions for constructing, interpreting, using and analyzing one type of **Architecture View**
  - it includes **Model Kind**s, viewpoint languages and notations, modeling methods and analytic techniques to frame a specific set of **Concern**s (e.g. operational, systems, technical, logical, deployment, process, information)
- **Model Kind**
  - defines the conventions for one type of **Architecture Model**
- **Correspondence**
  - express a relation between the elements of **Architecture Description**

Architecture Decisions

ISO/IEC 42010

- An **Architecture Decision** affects elements of **Architecture Description** and pertains to one or more **Concerns**
  - by making an **Architecture Decision**, new **Concerns** may be raised
- **Architecture Rationale**
  - records the explanation, justification or reasoning about **Architecture Decisions** that have been made and architectural alternatives not chosen



Pictures © ISO/IEC/IEE

Architecture Decisions are Decisions ...

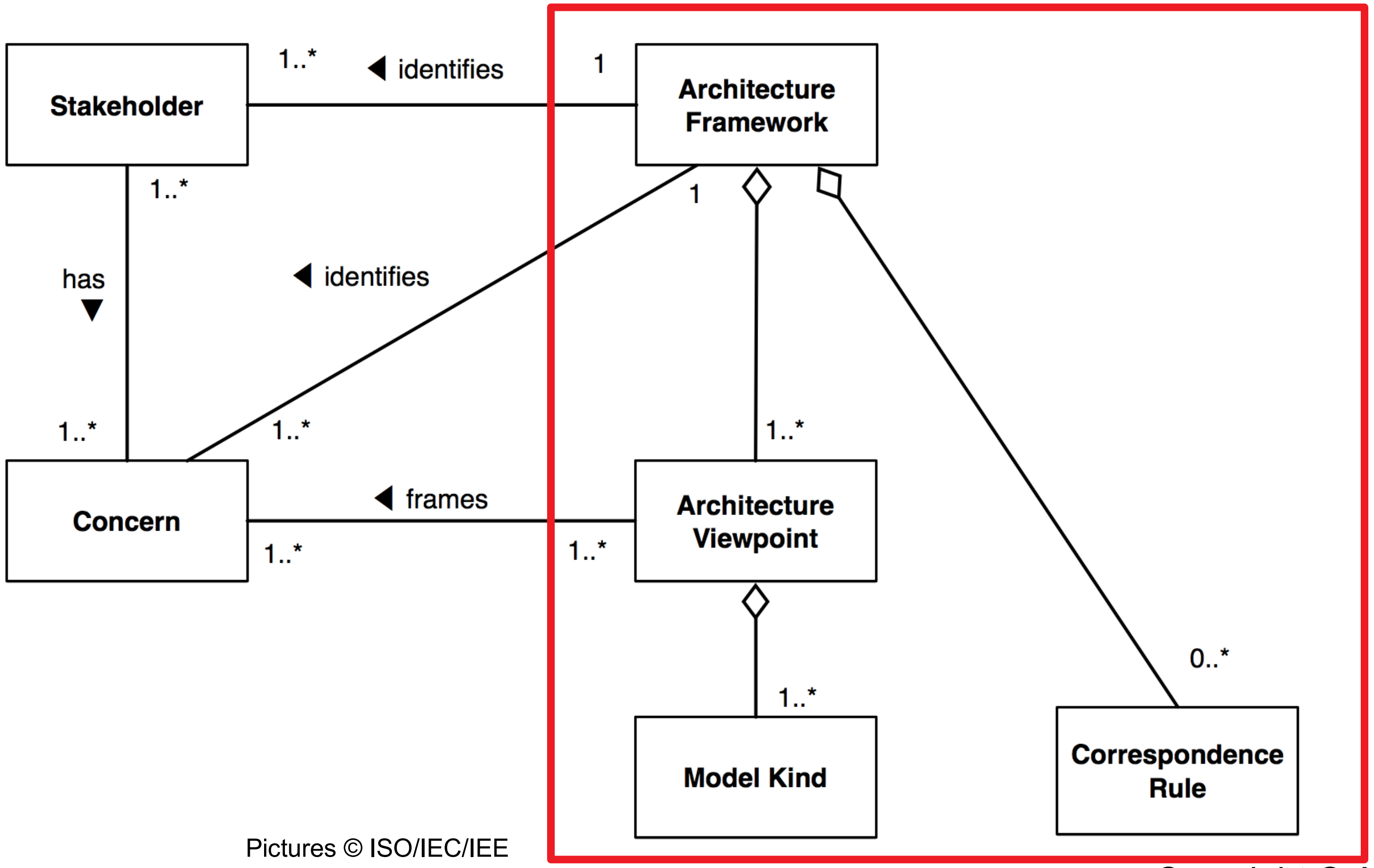
ISO/IEC 42010

- regarding **architecturally significant requirements**
- needing a **major investment** of effort or time to make, implement or enforce
- **affecting key stakeholders** or a number of stakeholders
- necessitating intricate or **non-obvious reasoning**
- that are highly **sensitive to changes**
- that could be **costly to change**
- forming a **base for project planning** and management (e.g. work breakdown structure creation, quality gate tracking)
- resulting in capital expenditures or indirect costs

Architecture Framework – a set of Viewpoints

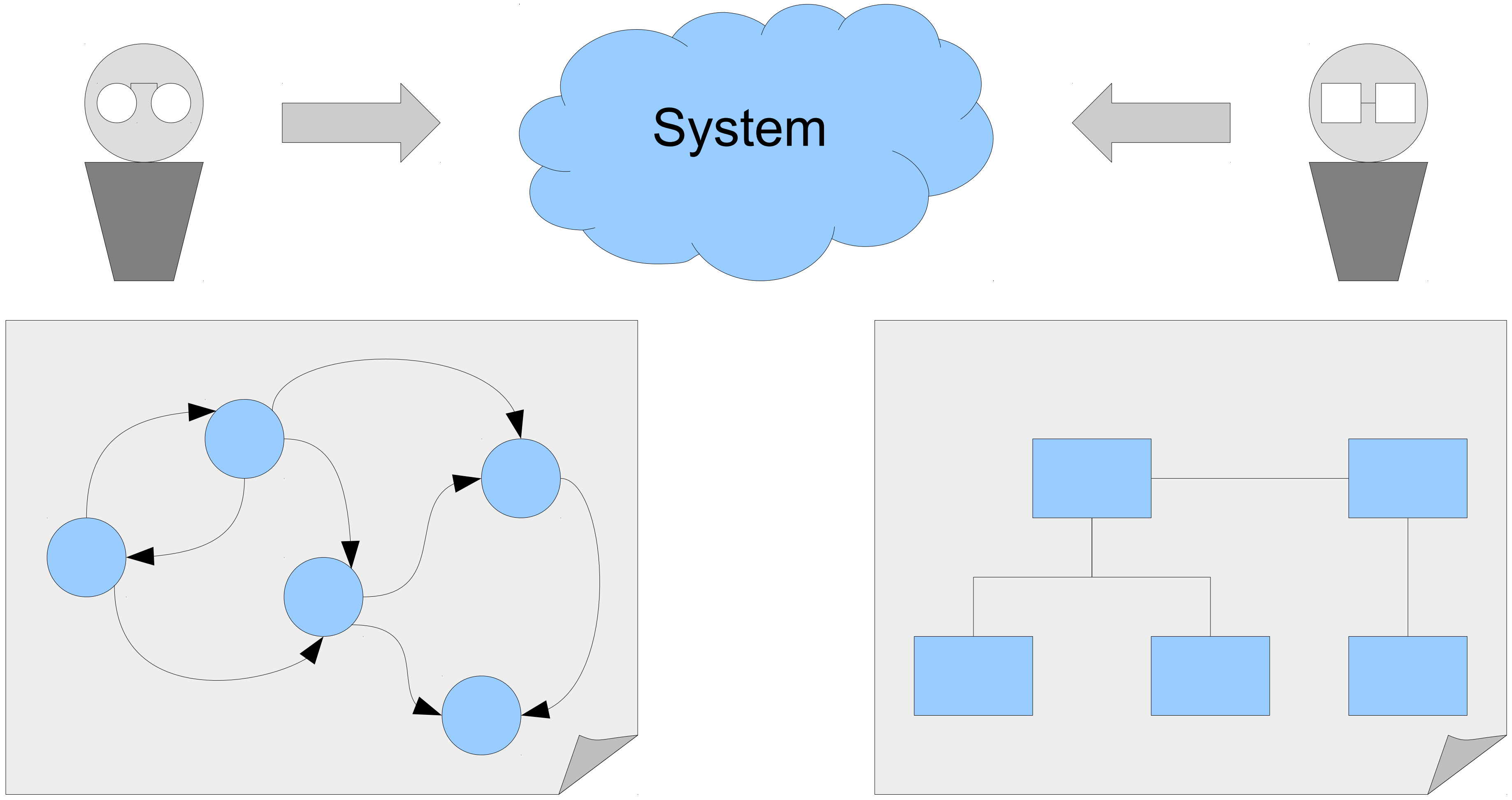
ISO/IEC 42010

- **Architecture Framework**
  - establishes a common practice for creating, interpreting, analyzing and using architecture descriptions within a particular domain of application or stakeholder community



Pictures © ISO/IEC/IEE

Architecture Frameworks – ways to Look at Systems



Example

Sensor Collection Service Architecture Description

ISO/IEC 42010

- Purpose (of the System)
  - Subscription-based service of providing access to a widely-distributed set of sensors
- Stakeholders
  - Users, developers, operators
- Architecture-related Concerns (by Stakeholders)
  - ROI (operators)
  - Timely delivery of sensor data (users)
  - Understanding of interactions between system elements (developers)

- Viewpoints (by Architecture-related Concerns)
  - Financial: cash-flow spreadsheet (ROI)
  - Operational: time-line diagram (timely delivery of sensor data)
  - System: system component diagram (understanding of interactions between system elements)
- View Consistency and Correspondence Rules
  - Each node in component diagram should appear at least once in time-line diagram

- Views (by Viewpoints)
  - Profit spreadsheet & profitability curve (cash-flow spreadsheet)
  - Time-line diagram (time-line diagram)
  - Data-flow diagram (system component diagram)

Content

Software architecture is what software architects do

Kent Beck

- Intro to the Software (Systems) Architecture
  - What we call Architecture
  - Why we need to bother with Architecture
  - Design vs. Architecture
  - Early Views and Software Architecture
- Software Architecture (Description) related Concepts and Terminology (IEEE 1741 | ISO/IEC 42010)
- Other related Concepts
  - Abstraction
  - Complexity
  - Modularity
  - Model
- Conclusions

Abstraction

> The essence of abstractions is preserving information that is relevant in a given context, and forgetting information that is irrelevant in that context.
>
> John V. Guttag

- Abstraction
  - process of generalization by reducing the information content of a concept or an observable phenomenon, selecting only the aspects which are relevant for a particular purpose
  - process of suppressing irrelevant details
- Abstraction is a technique for managing complexity
- In software development abstraction can apply to control or to data
  - Control abstraction involves the use of subroutines and control flow abstractions
  - Data abstraction allows handling pieces of data in meaningful ways
- The notion of an object in object-oriented programming can be viewed as a way to combine abstractions of data and code

Complexity

Software is not limited by physics, it is limited by imagination, by design, by organization – it is limited by properties of people, not by properties of the world.
“We have met the enemy, and he is us.”

R. Johnson

- Complexity
  - degree to which a system's design or implementation is difficult to understand and verify

    because of numerous components, numerous relationships among components and multiple ways that components can interact
- Complex systems are intrinsically difficult to model, and have distinct properties, such as non-linearity, emergence, spontaneous order, adaptation, feedback loops, ...
- Main source of complexity is the irreversibility of decisions (E. Zaninotto)
  - if you can easily change your decisions, it's less important to get them right
  - the consequence for evolutionary design is that designers need to think about how they can avoid irreversibility in their decisions
  - it's worth doing experiments to see how hard future changes can be, even if you don't actually make the real change now

Modularity

Modularity creates design options:
- taking more risk creates more value
- redundant efforts may be value-increasing
- designs becomes “tolerant of uncertainty”

Carliss Baldwin & Kim Clark

- Modularity
  - degree to which a system's components may be separated and recombined with minimal impact on other components
- Purpose of Modularization
  - to make complexity manageable
  - to enable parallel work
  - to accommodate future uncertainty
- Modularizing a system involves specifying
  - its architecture, that is, what its modules are
  - specifying its interfaces, i.e., how the modules interact
  - tests which establish that the modules will work together and how well each module performs its job

Some more Concepts related to Modularity

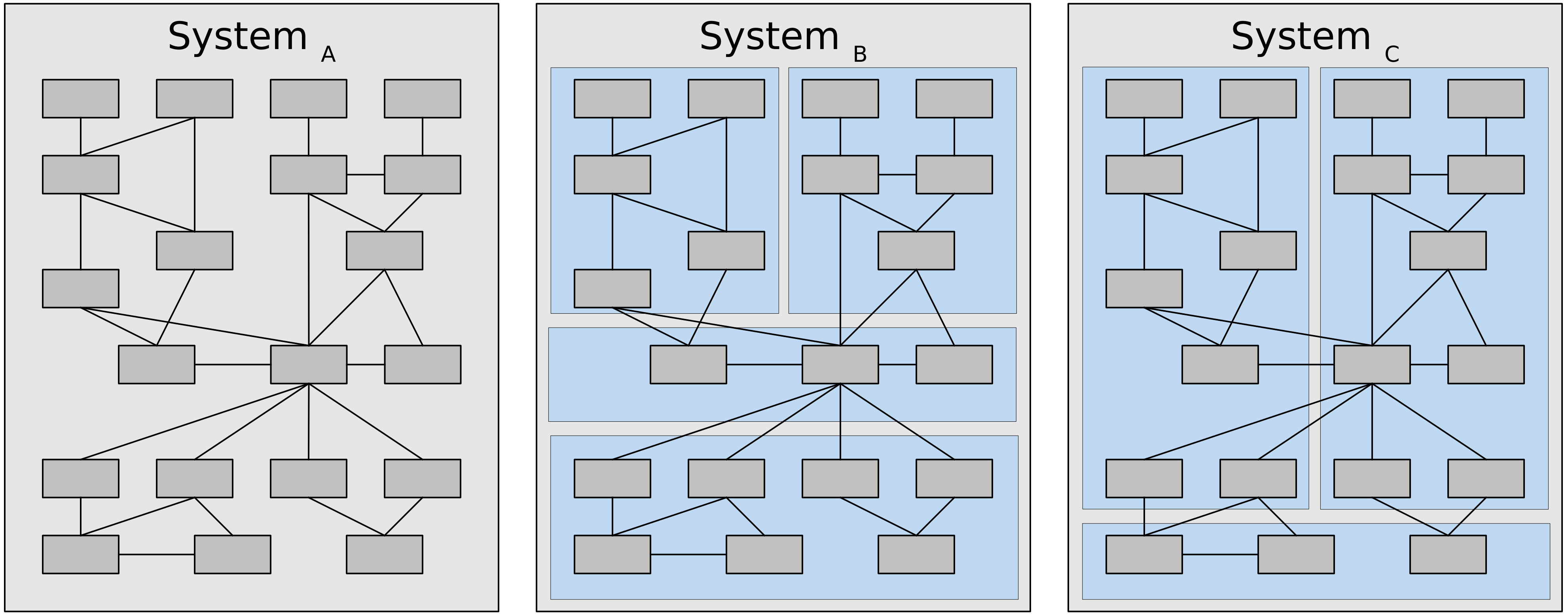
Strong cohesion and low coupling are the attributes of good design (produce a stable structure)

E. Yourdon, L. L. Constantine

- Encapsulation
  - isolating some parts of the system from the rest of the system
  - a module has an outside that is distinct from its inside (an external interface and an internal implementation)
- Coupling
  - the manner and degree of interdependence between modules
  - the strength of the relationships between modules
  - a measure of how closely connected two modules are
- Cohesion
  - the manner and degree to which the tasks performed by a single module are related to one another (the degree of functional relatedness of processing elements within a single module)
  - a measure of the strength of association of the elements within a module

Overall Complexity

- Using modules to balance between system and modules volume/complexity



Measuring Complexity

- Counting elements of source
  - LOC (count of all the code lines) & SLOC (number of executable code lines)
- Counting elements of architecture
  - Cyclomatic Complexity (T. J. McCabe, 1976) measuring the complexity of a structure (number of linearly independent paths through the structure)
  - OO Metrics (Chidamber & Kemerer) measuring the coupling, cohesion, depth of inheritance structures, etc.
  - Software Science (M. H. Halstead, 1977) measuring number of unique or distinct operators and operands and total usage of all the operators and operands to estimate the volume, intelligence content (complexity), etc.
  - Function Point (A. J. Albrecht, 1979, IFPUG) measuring data elements and queries to estimate functional size of system
- Calculating entropy
  - Entropy (E. B. Allen) calculating the information (average number of bits) needed to describe the interconnections
  - Source Code Entropy (O. Panchenko, S. H. Mueller, A. Zeier) calculating the information contained in source code

Model – a Tool for Thinking

architecture description is a model of architecture

- Model is
  - a representation that suppresses certain aspects of the modeled subject
  - a representation of a system of interest, from the perspective of a related set of concerns – a complete description of a system from a particular perspective
  - a formal theory (a body of knowledge) about modeled subject

- Model is

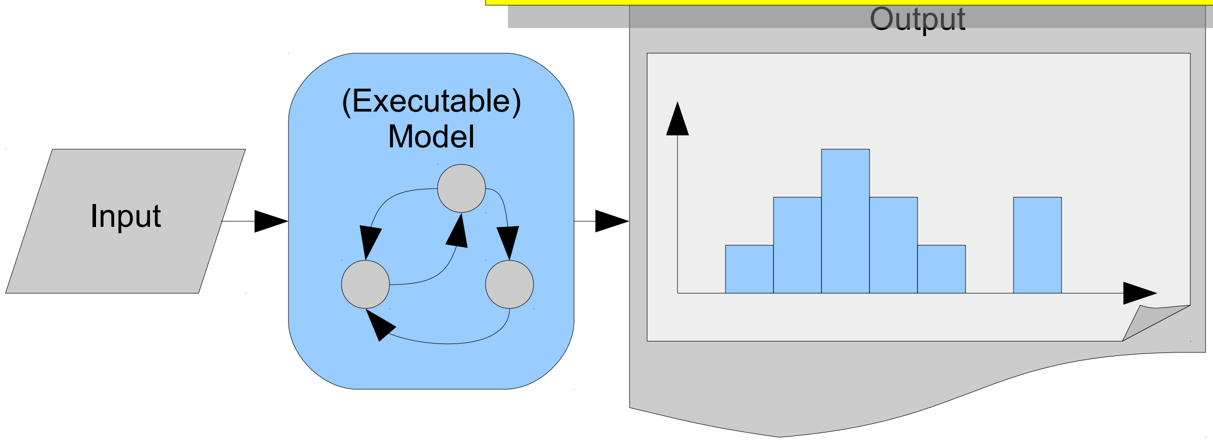
  anything that can be used to answer questions about the system

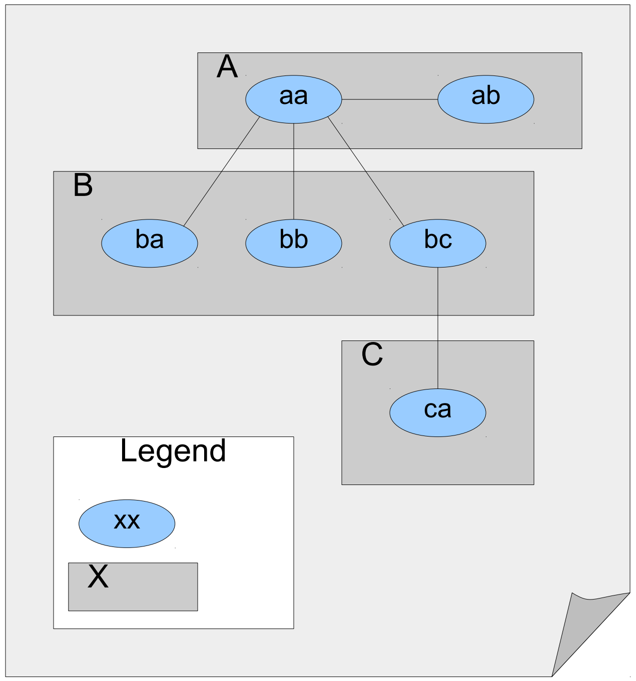
  - Marvin Minsky
    - to an observer B, an object M is a model of an object A to the extent that B can use M to answer questions that interest him about A
  - Doug Ross
    - M is a model of A with respect to question set Q if and only if M may be used to answer questions about A in Q within tolerance T

Examples of Models

Anything that can be used to answer questions about the system

ID	Group	Name	Parent	...
1	A	aa	-	ab
2	A	ab	-	aa
3	B	ba	aa	
4	B	bb	aa	
5	B	bc	aa	
6	C	ca	bc	





Picture © Rabobank & BIAN

Properties of a Good Model

Good model is easy to make, use and gives right answers

- Complete
  - in respect of particular property all questions can be answered with the model
  - model represents all the semantically correct statements about the subject
- Consistent
  - contains no conflicts/contradictions, i.e. is gives no conflicting answers
- Correct
  - relative to modeled subject – all answers that are produced by the model, are true for the subject
  - relative to modeling language – is not violating any rules and conventions
- Comprehensible
  - understandable for intended users (also machine readable)
  - there is correspondence between model elements and elements of subject (traceable)
- Confined – without unnecessary information

Content

Software architecture is what software architects do

Kent Beck

Conclusions

Architecture is the important stuff – whatever that is

Ralph Johnson

- Software Architecture is a
  - *fundamental conception* of a software system in its
  - ***environment*** embodied in
  - ***elements***, their
  - ***relationships*** to each other and to the environment, and
  - ***principles*** guiding software system design and evolution

- Software Architecture Description is a
  - collection of ***related (corresponding) models***, organized into cohesive groups of
  - *synthetic* (constructed) or *projective* (derived) ***views***, defined by ***viewpoints*** according to the related set of ***concerns*** *(defined in architecture framework)*

- Software Architecture Model is a
  - ***work product*** that can be used to answer questions about the software system

Conclusions

All architecture is design but not all design is architecture

Grady Booch

- Although every software system has an architecture, but **architecture** with desirable properties doesn't emerge itself, it **needs to be designed**
- Value of Software Systems Architecture
  - As a cause of certain properties of software systems, **designing architecture allows us to address concerns** and to achieve required and desirable properties of software systems
  - As fundamental conception of software system, architecture **allows us to reason** (i.e. answer questions) about the software system and its properties, and foresee those properties beforehand (without building and testing the actual system)
- Value of Software Systems Architecture Description
  - As a document, it provides guidance for constructing and evolving the software system, and allows us to **record and communicate our knowledge and decisions** about the software system architecture
  - As a model, it **allows us to reason** (answer questions) about the software system architecture

38. The architect concerns himself with the depth and not the surface, with the fruit and not the flower.

Lao Tsu (by Philippe Kruchten)

Thank You!

Questions

- What is architecture of a (software) system?
- Why architecture of (software) system is important?
- Does every (software) system has an architecture?
- Which part of the software architecture assures the durability (guards against the erosion)?
- What is a description of architecture of a (software) system? From what it consists of?
- What is the purpose of (software) system architecture description?
- What is a viewpoint and what is a view? How these differ?
- Who are stakeholders of a (software) system? Bring examples?
- What are system architecture related concerns? Bring examples?
- What is a (software) architecture framework? From what it consist of?
- Which kinds of (software) architecture structures there are?
- What is model? Is any picture a model?
- What is modularity? What it creates?
- What is abstraction?
- Does software architecture description has an architecture?

Literature

- Beginning
  - https://people.cs.clemson.edu/~mark/subroutines.html
  - https://www.fourmilab.ch/babbage/sketch.html
- Early writers
  - http://www.melconway.com/Home/Committees_Paper.html
  - http://worrydream.com/refs/Brooks-NoSilverBullet.pdf
  - http://repository.cmu.edu/cgi/viewcontent.cgi?article=2979&context=compsci
  - https://www.researchgate.net/publication/3188975_On_the_Design_and_Development_of_Program_Families
  - http://www.jacksonworkbench.co.uk/stevefergspages/jsp_and_jsd/index.html
- Software Architecture
  - http://users.ece.utexas.edu/~perry/work/papers/swa-sen.pdf
  - http://www.laputan.org/pub/sag/wolf-arch-foundations.pdf
  - https://www.sei.cmu.edu/library/assets/icse03-1.pdf

- http://www.iso-architecture.org/ieee-1471/cm/

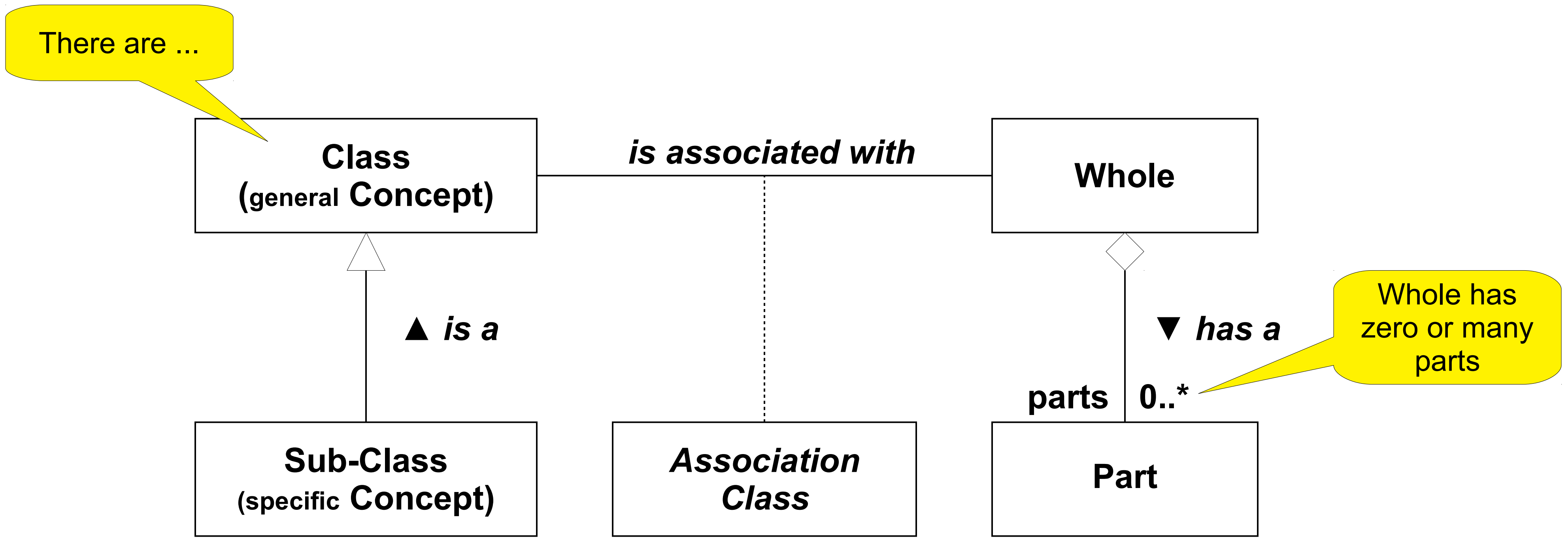
- … Google "software architecture" ...

Terms (Glossary)

ISO/IEC 42010

Term	Definition
architecture	fundamental conception of a system in its environment embodied in elements, their relationships to each other and to the environment, and principles guiding system design and evolution
architecture decision	choice made from among possible options that addresses one or more architecture-related concerns
architecture description	collection of work products used to describe an architecture
architecture model	work product from which architecture views are composed
architecture rationale	explanation or justification for an architecture decision
architecture view	work product representing a system from the perspective of architecture-related concerns
architecture viewpoint	work product establishing the conventions for the construction, interpretation and use of architecture views
architecture-related concern	area of interest in a system pertaining to developmental, technological, business, operational, organizational, political, regulatory, social, or other influences important to one or more of its stakeholders
environment	context determining the setting and circumstances of developmental, technological, business, operational, organizational, political, regulatory, social and any other influences upon a system
model correspondence	relation on two or more architecture models
stakeholder	individual, team, organization, or class thereof, having concerns with respect to a system
purpose	*{one of system concerns}*
system	*{a conceptual entity defined by its boundaries}*

Crash Course on UML


There are ...
Class
(general Concept)
is associated with
Whole
▲ is a
▼ has a
Whole has zero or many parts
parts
0..*
Sub-Class
(specific Concept)
Association Class
Part