Software Architecture

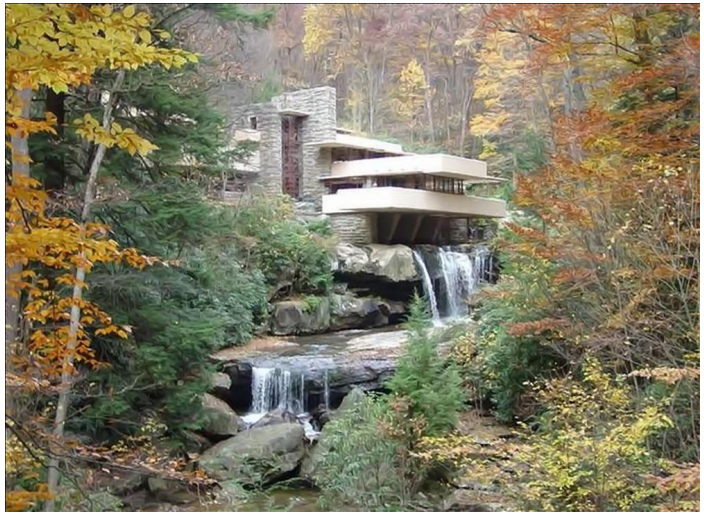
Paolo Ciancarini

Agenda

- Software Architecture: definitions
- The standard IEEE 1471 and its successors
- Architectural frameworks
- Architectural assets

What is the role of architecture?

Form vs function:
how many types of houses…



The role of architect of houses

- Architects focus on people's need
- They design buildings and interiors according to the desires of their customers
- In software terms this would be roughly equivalent to «designers of the user experience»

What is Software Architecture?

- Software Architecture is like any other architecture: it is about the function - or purpose - and the form - or structure
- It is about the concerns of the stakeholders
- It is about the gap between a current system's design and its future design
- It is about the evaluation of the properties which govern the design and the evolution of a system

Software architecture

- Software architecture is what is hard to change
- Software architecture is about decisions so critical that if they are wrong they will kill the project or at least will be very expensive to fix

What is the goal of Software Architecture?

- To understand the fundamental elements of a system that contribute to its utility, cost, effort to build, and risk to use within its environment
- In some cases, the fundamental elements are physical or structural components of the system and their relationships
- Sometimes, the fundamental elements are functional or logical elements
- In other cases, what is fundamental to the understanding of a system are its overarching principles or patterns

Form and function of software

- What is the function of a software system?
  - The function of a software system is its mission, as described by some use cases in some scenario and as verified by some tests which refer to the requirements
- What is the form of a software system?
  - The form of a software system is a model of the system as described by some structural or behavioral views and as validated by some stakeholder

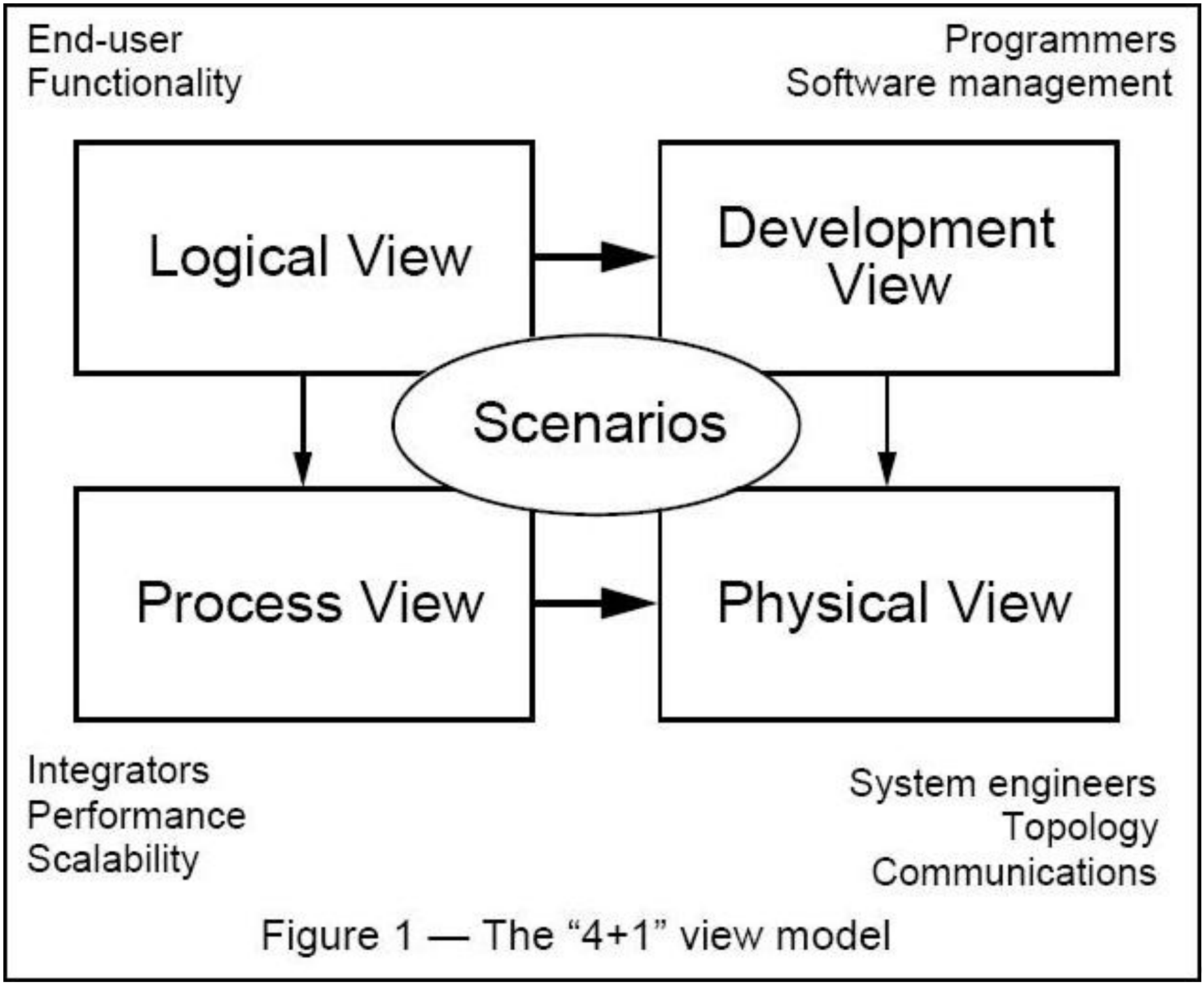
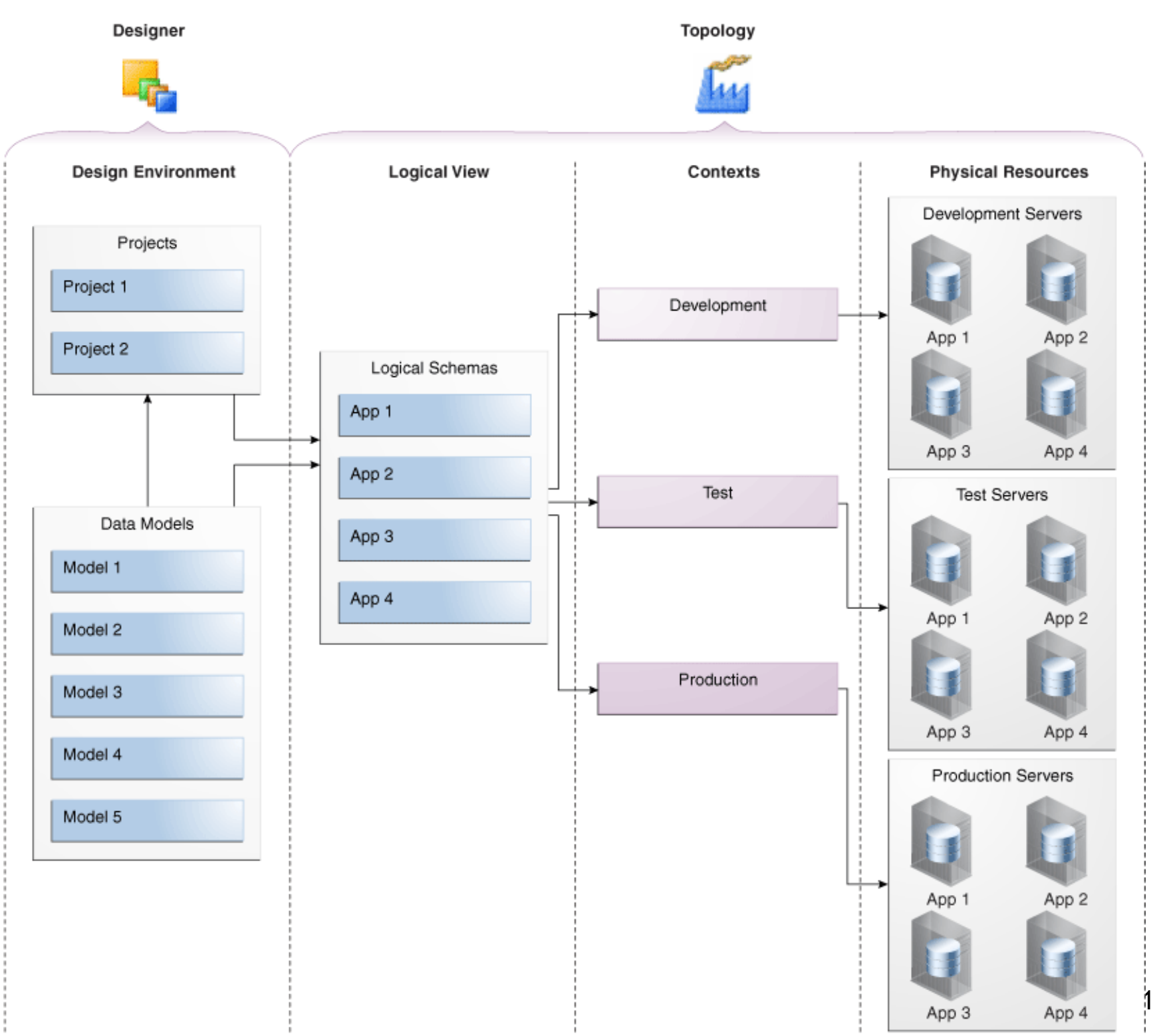


Figure 1 — The “4+1” view model

Views

- **Logical view** Addresses the static design model
- **Process view** Addresses the design's dynamic view
- **Physical view** Addresses how the software components are mapped onto the hardware infrastructure
- **Development view** Represents the static organization of the software components in the development time environment


Designer
Topology
Design Environment
Logical View
Contexts
Physical Resources
Projects
Project 1
Project 2
Data Models
Model 1
Model 2
Model 3
Model 4
Model 5
Logical Schemas
App 1
App 2
App 3
App 4
Development
Test
Production
Development Servers
App 1
App 2
App 3
App 4
Test Servers
App 1
App 2
App 3
App 4
Production Servers
App 1
App 2
App 3
App 4

What is software architecture?

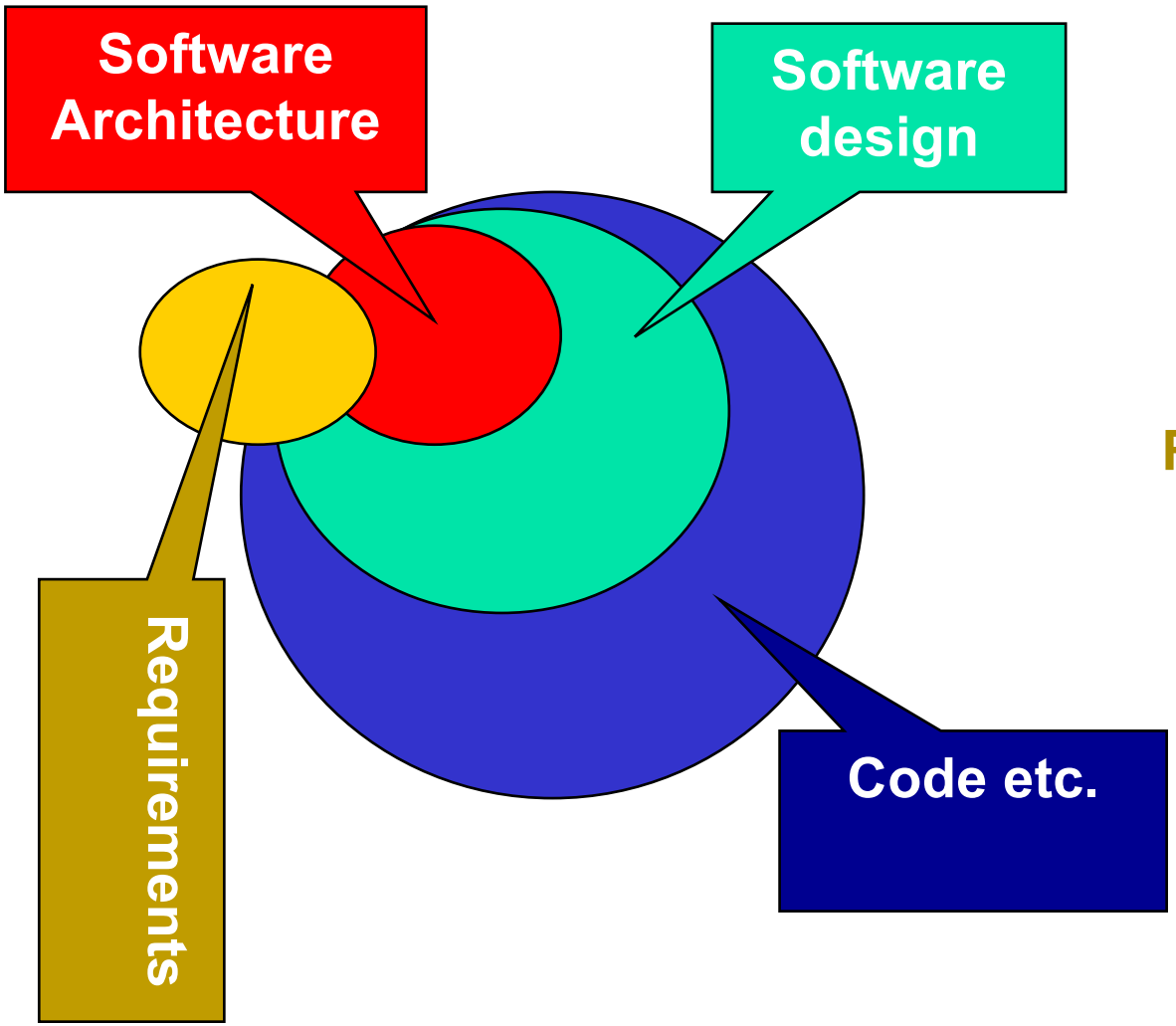
The software architecture represents the "significant decisions", where significance is measured by cost of change

Grady Booch

Software architecture is the set of design decisions which, if made incorrectly, may cause your project to be cancelled

Eoin Woods

Architecture = design decisions



Decisions:
Design decisions
Architectural decisions
Requirements constraints

decision: a choice that is binding in the final product

Hoare on software design

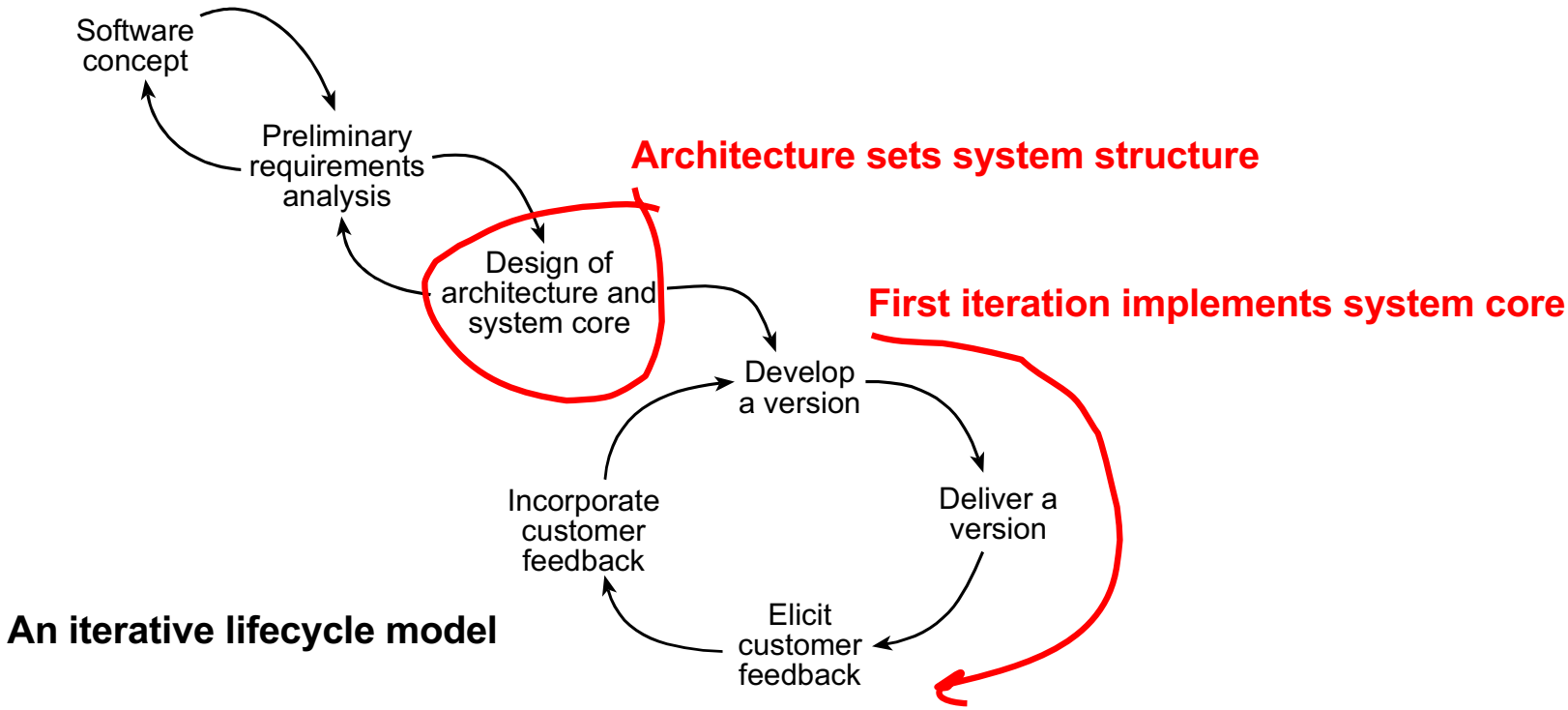
There are two ways of constructing a software design:

- *One way is to make it so simple that there are obviously no deficiencies,*
- *and the other way is to make it so complicated that there are no obvious deficiencies.*

Architecting software

- Architectural design has no stopping rule:
  there is no criterion that tells when the architecture is finished
- Solutions to architectural problems are not correct or wrong:
  usually they are good or bad; solving one problem may very well result in an entirely different problem elsewhere in the system
- Architectural design involves trade-offs, such as those between *speed* and *robustness*: as a consequence, there is a number of acceptable solutions, rather than one best solution
- Architectural design problems do not have a well-defined set of potential solutions: software architects cannot exploit a set of ready-made solutions, but have to apply knowledge, practices, and creativity to arrive at a satisfactory solution

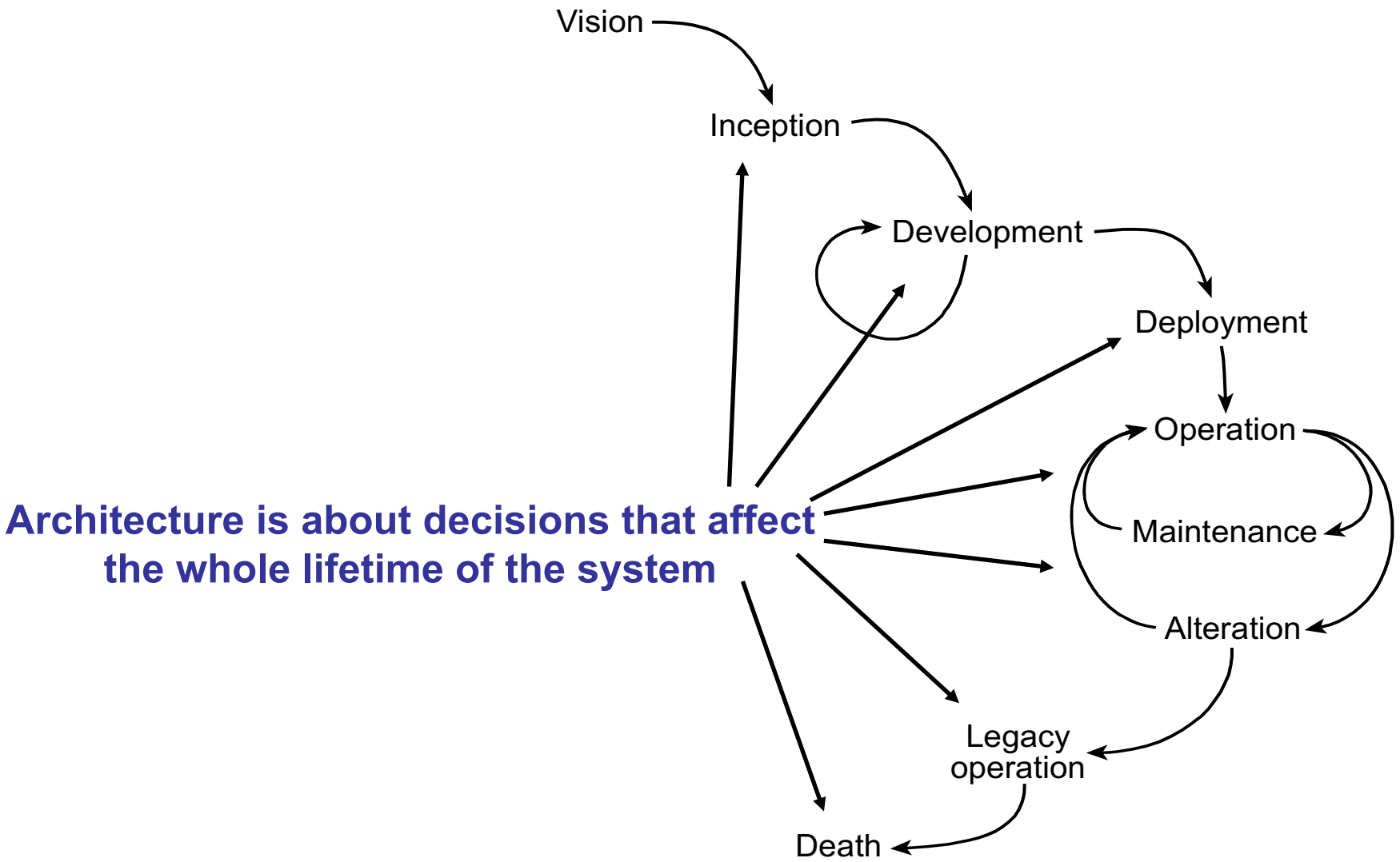
Role of architecture



An iterative lifecycle model

Architecture plays a vital role in establishing the structure of the system, early in the development lifecycle

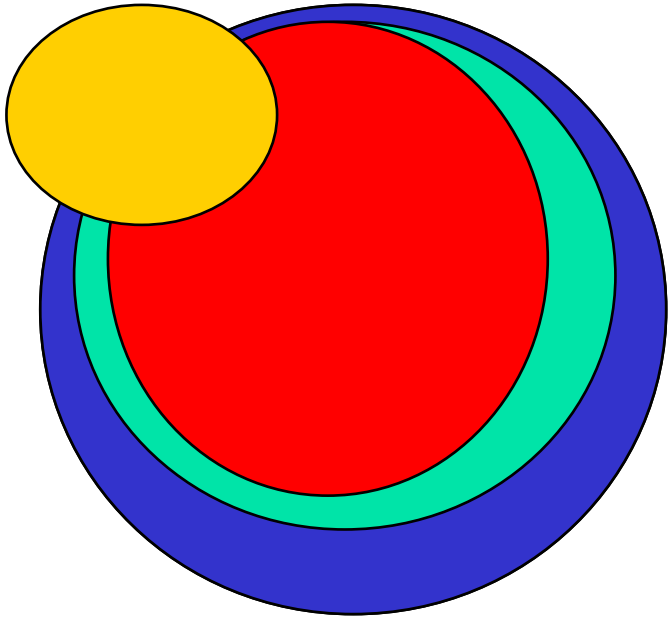
Role of architecture



Significant decisions

- The software architecture represents the "significant decisions", where significance is measured by cost of change
  - Example1: if the system is an individual application, any decision that could be made by implementors should be deferred to them and not appear as part of the architecture
  - Example2: if the scope of the architecture is a family of applications (a product line), then any decision that relates only to a single application should be deferred to the application architecture and not be part of the application family architecture

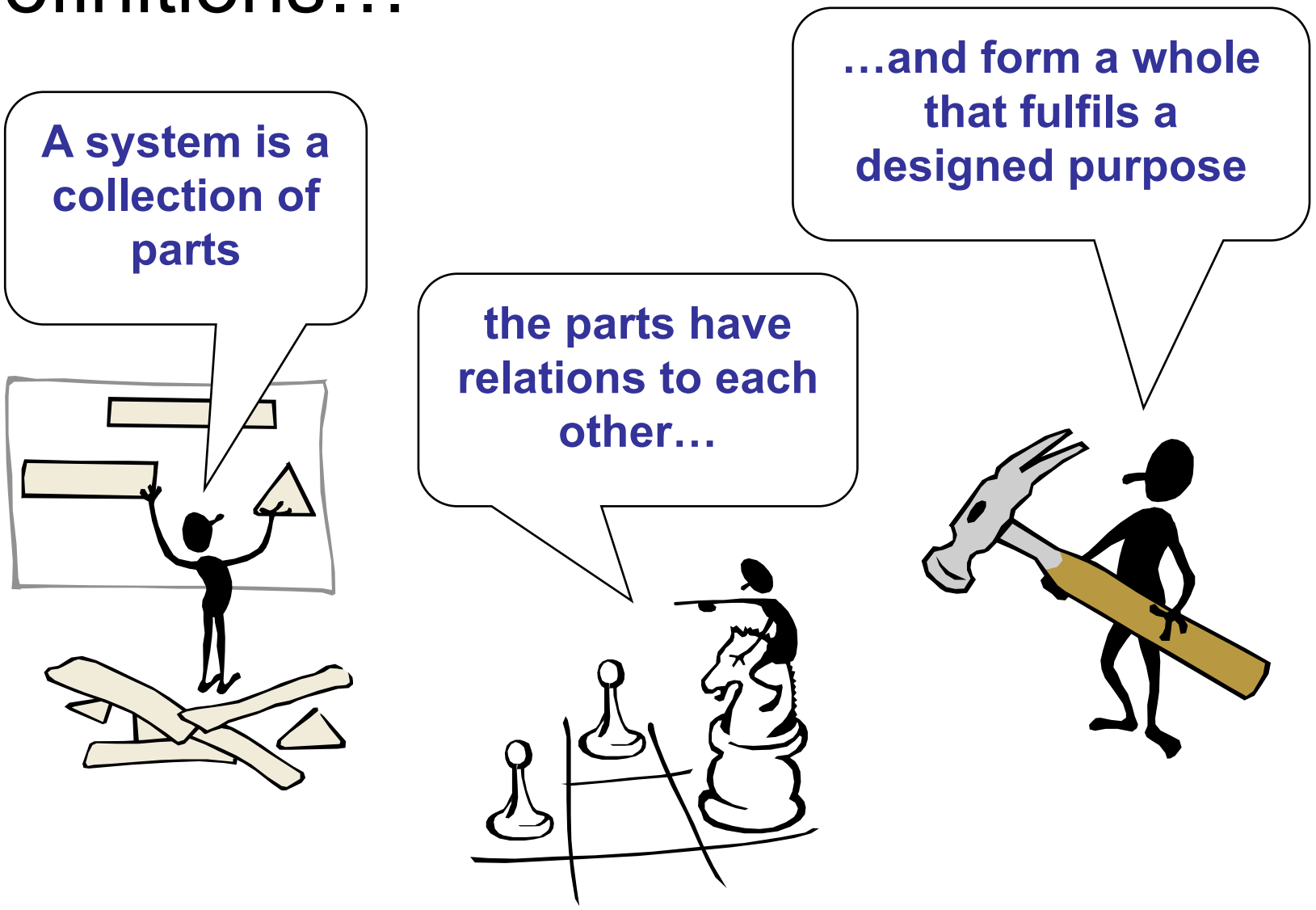
Architecture = Design? No!



"Do not diluite the meaning of the term architecture by applying it to everything in sight."

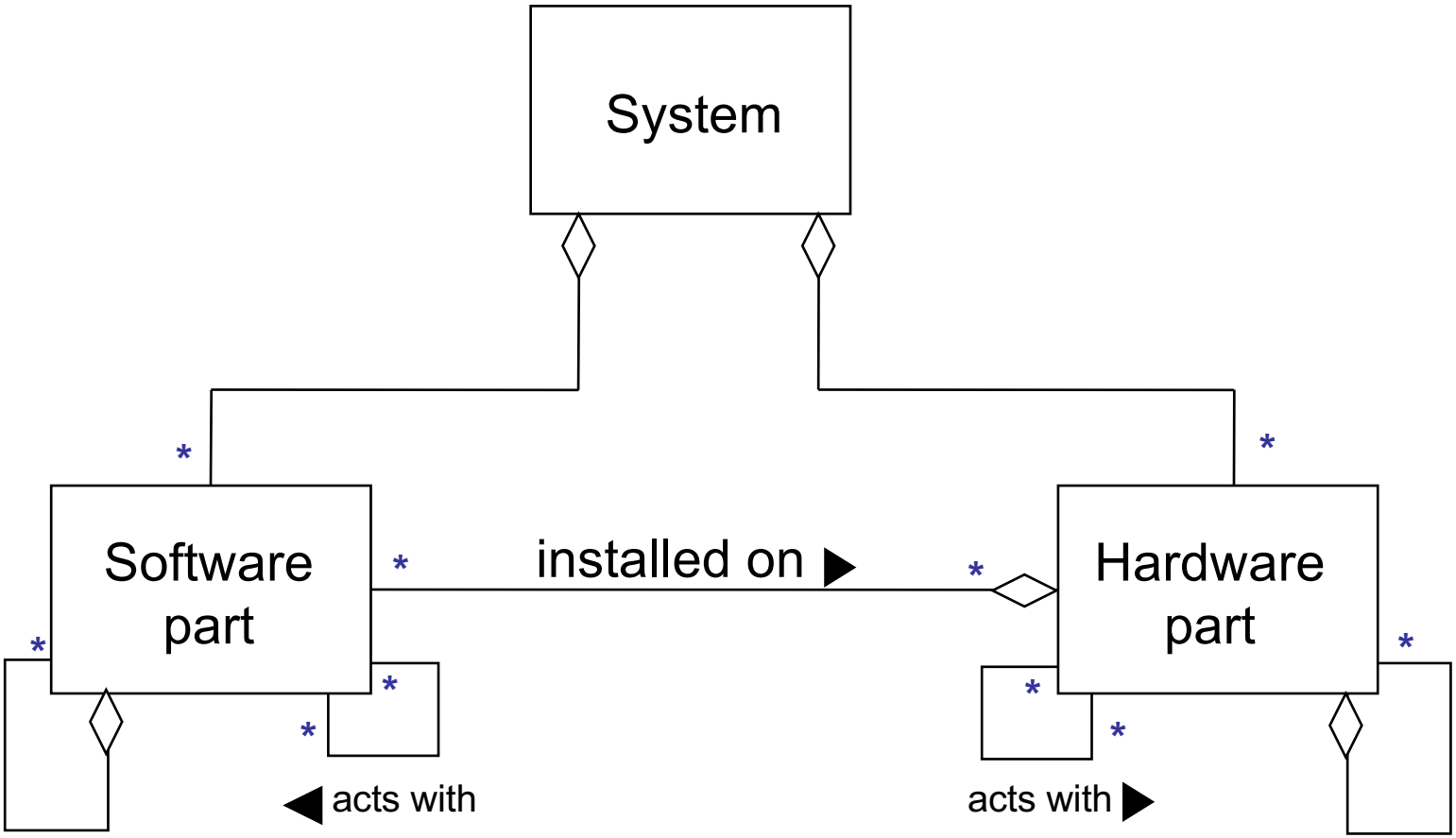
Mary Shaw

Definitions…



System

(actually, “software-intensive system”)



Most systems include also:

- Natural elements
- Humans

Definition: system [IEEE 2000]

- A system is a collection of components organized to accomplish a specific function or set of functions
- A system exists to fulfill one or more missions in its environment

- The term “system” encompasses individual applications, systems in the traditional sense, subsystems, systems of systems, product lines, product families, enterprises, and other entities of interest

Two main families of systems

- Information systems, necessary for enterprise business (eg. banks, e-commerce, egovernment)
- Embedded software systems, necessary for engineering domains (eg. automotive, aerospace)

- Information systems' architectures are usually related to an enterprise organization, meaning that human roles and actions are included in the description of the system
- Embedded systems' architectures are usually related to a system architecture (people are "outside" the system)

Abstraction Granularity:	Key Design Concerns:
Enterprise Architecture	• Business Processes and Models • Business Data • Organizational Structure and Relationships • Enterprise Stakeholders • IT Infrastructure
System Architecture	• Identification of System Context • Partitioning (hardware/infrastructure focus) • Identification of Software Requirements • Overall Systemic Functional Requirements • Systemic Integration and Testing
Software Architecture	• Identification of Crosscutting Design Concerns (quality attributes) • Software Functional Requirements • Partitioning of Software Application(s) • Software and Systemic Integration and Testing
Detailed Software Design	• Language Features • Algorithmic Efficiencies • Data Structure Design • Software Application Testing • Implementation of Functionality

Definition: Software architecture

- The software architecture of a system is the **structure** of the system, which comprises its **software elements**, the **externally visible properties** of those elements, and the **relationships** between them
- The term also refers to the **documentation** of a system's software architecture
- Documenting a software architecture facilitates the communication between stakeholders, documents the early decisions about high-level design, and allows the reuse of its components across projects

Short history

- The basics of software architecture were introduced by Dijkstra (1968) and Parnas (1970). They studied:
  - The key role of the structure of a software system, and
  - How tricky the definition of the *right structure* for a given system is
- A great deal of research has been conducted on software architectures starting from the ’90s, mainly focusing on:
  - How to solve recurrent architectural problems (*patterns*)
  - Which architectural forms are more common (*styles*)
  - Defining languages to describe software architectures (ADL)
  - How to document the architecture of a system (*views*)
- … claimed to give rise to the definition of a discipline
  - See: Shaw and Garlan *Software Architecture: Perspectives on an Emerging Discipline*, 1996

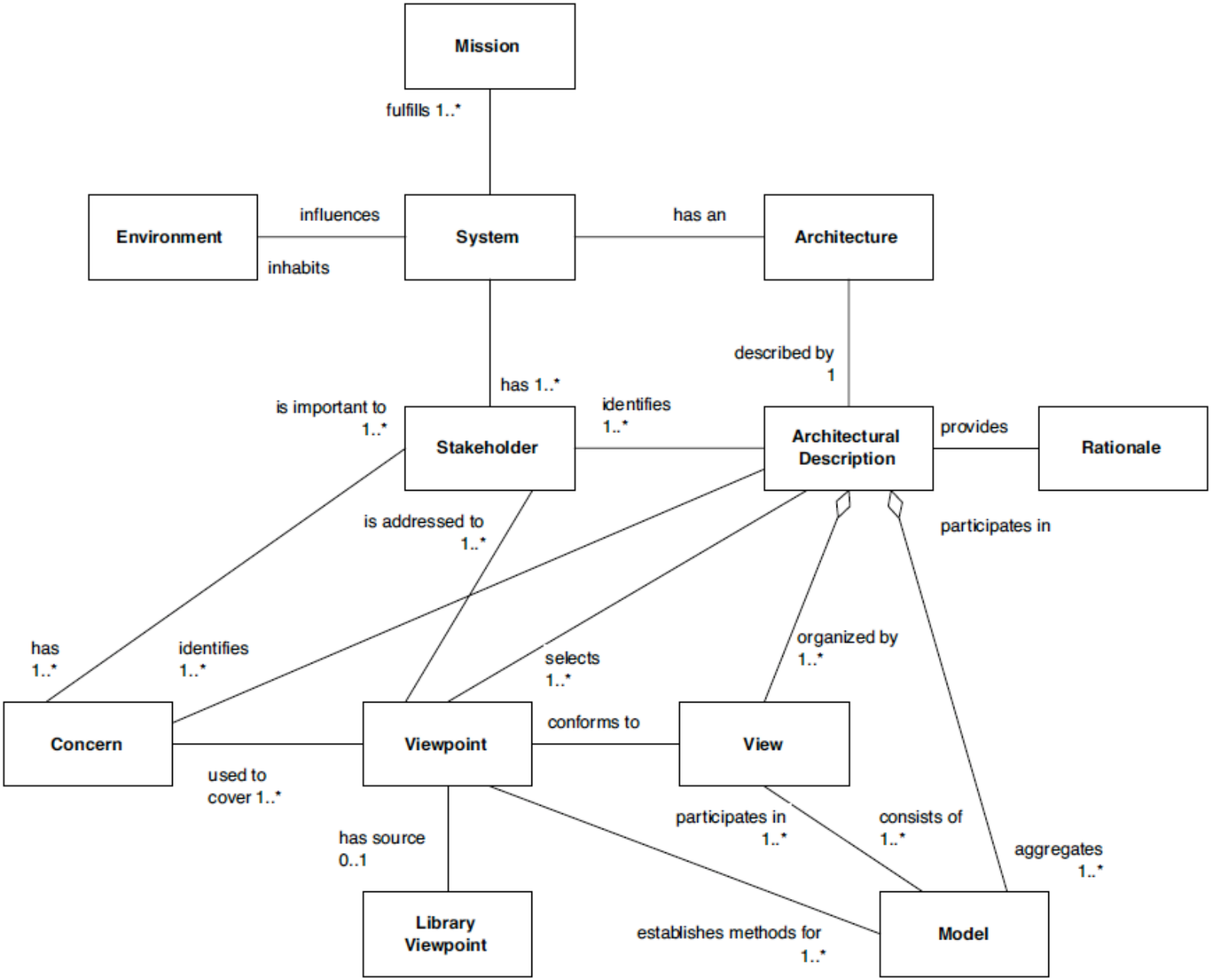
Standard: IEEE 1471

- The main standard in the field of software architectures is the **ANSI/IEEE 1471-2000: Recommended Practice for Architecture Description of Software-Intensive Systems**
  - better known as IEEE 1471 standard
- It is a “recommended practice”
  - the “weakest” type of IEEE standards, whose adoption and interpretation are the responsibility of the using organization
- It has been adopted by ISO/IEC JTC1/SC7 as ISO/IEC 42010:2007, in 2007
- most recent version is ISO/IEC/IEEE 42010:2011

Definition: IEEE 1471

- Intuitively, the **architecture of a software system** represents its internal structure, in terms of the single components of which it is made up and of the interactions among them
- According to the standard, the architecture of a software system is its *basic organization*, in terms of its *components*, of the *interactions* among components and with the external environments, as well as of the basic, driving, *principles* for system design and evolution

IEEE1471 Conceptual Model for Sw Architectures



Scope of the standard

The standard covers software-intensive systems in which software development and/or integration are dominant issues in terms of cost and resources

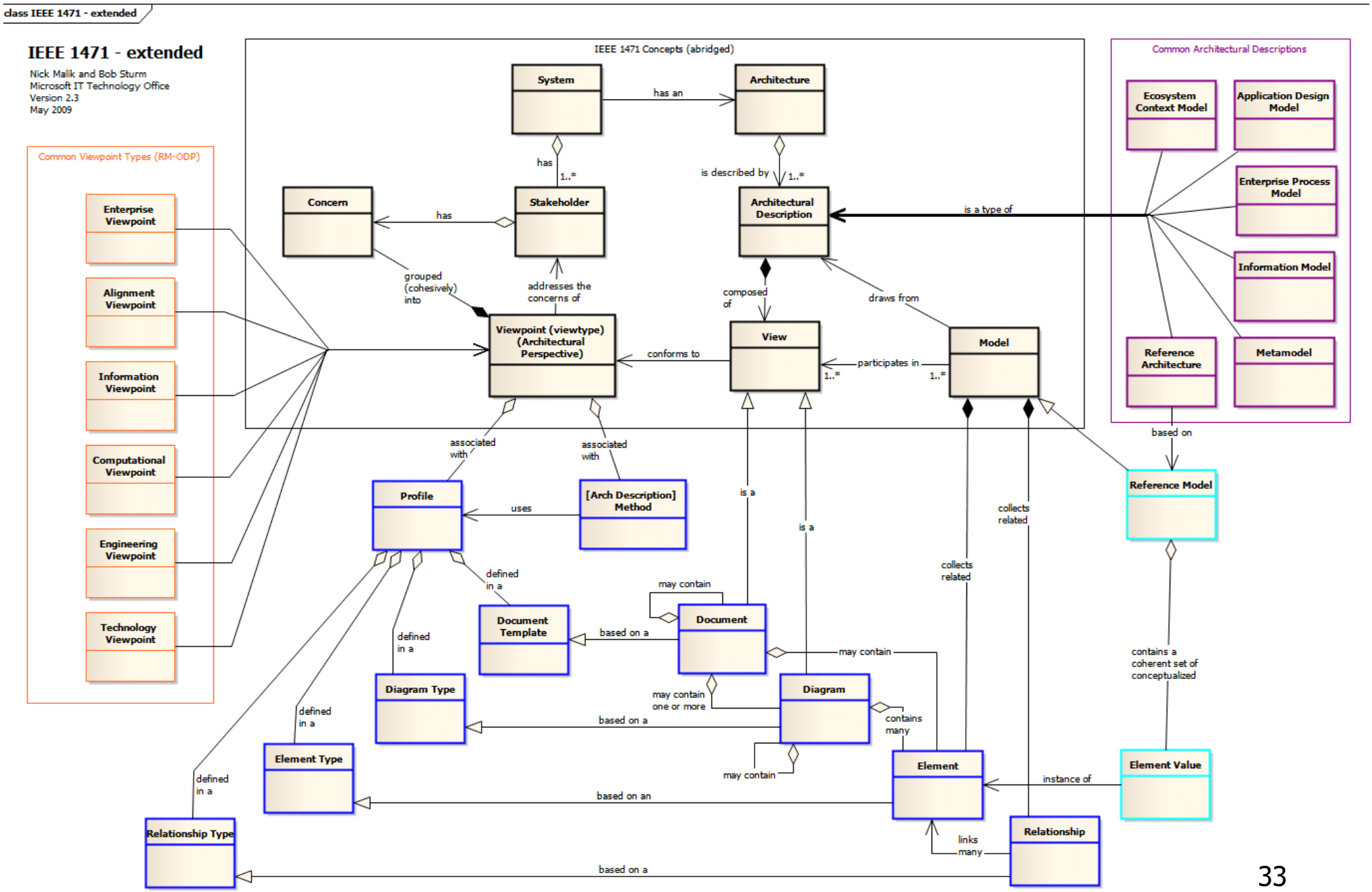
Software-intensive systems include: individual software applications, information systems, embedded systems, software product lines and product families, and systems-of-systems

Consequences of IEEE 1471

IEEE 1471's contributions:

- It provides the definitions and a conceptual model for the *description of a software architecture*
- It states that an architecture goal is to respond to specific *stakeholders*' concerns about the software system being described
- It asserts that architecture descriptions are inherently **multi-view**, as no single view can capture all stakeholders' concerns about an architecture
- It separates the notion of a view from a *viewpoint*, which identifies the set of **concerns** and the representations/modeling techniques, etc. used to describe the architecture to address those concerns
- It establishes that a *conforming* architecture description has a 1-to-1 correspondence between its viewpoints and its views
- It can be extended to become a company specific architectural model

Extending 1471



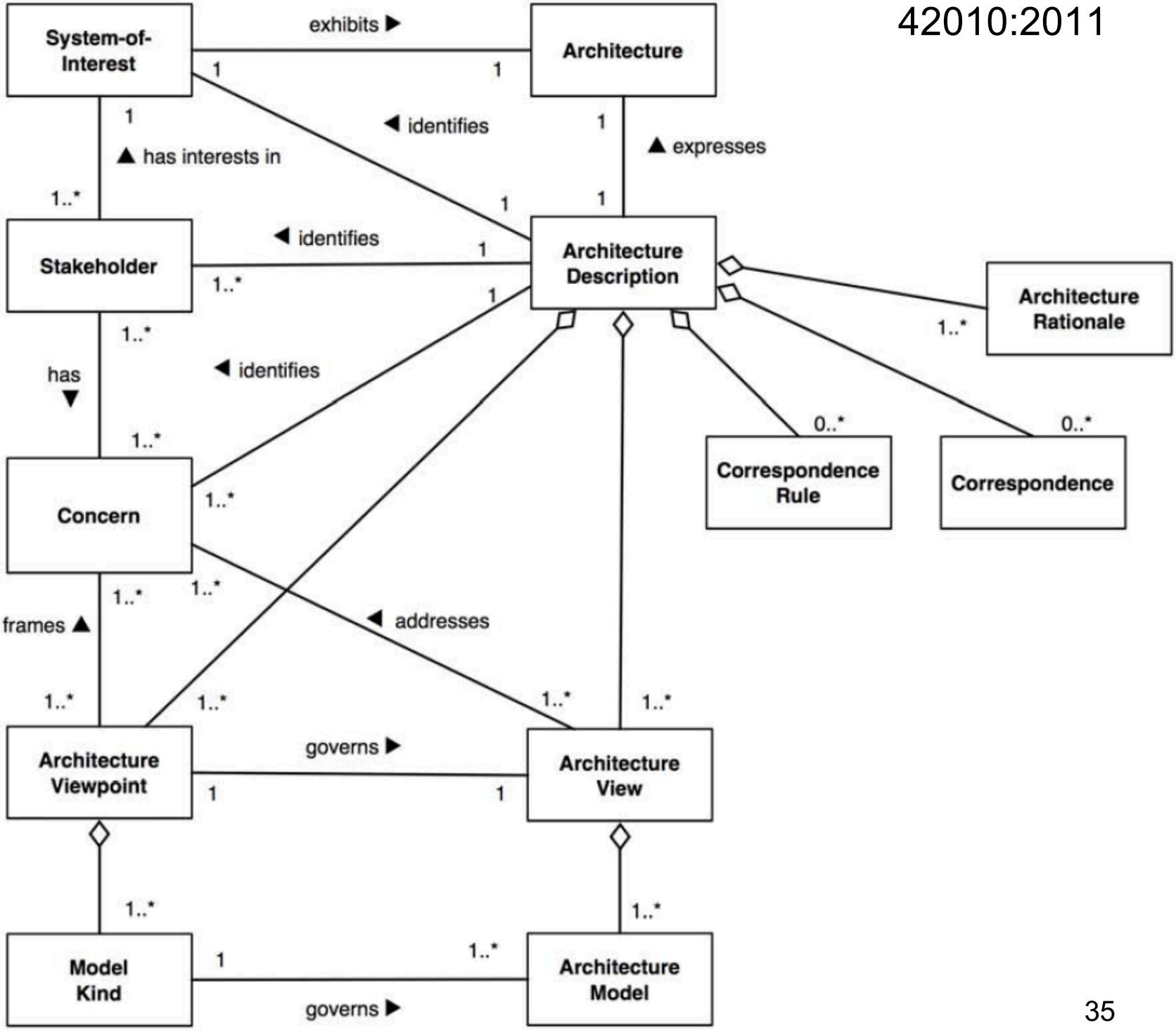
Evolution of the standard

- 2000: IEEE 1471
- 2001: ANSI
- 2007: ISO/IEC 42010
- 2011: ISO/IEC/IEEE 42010:2011 Systems and software engineering -- Architecture description

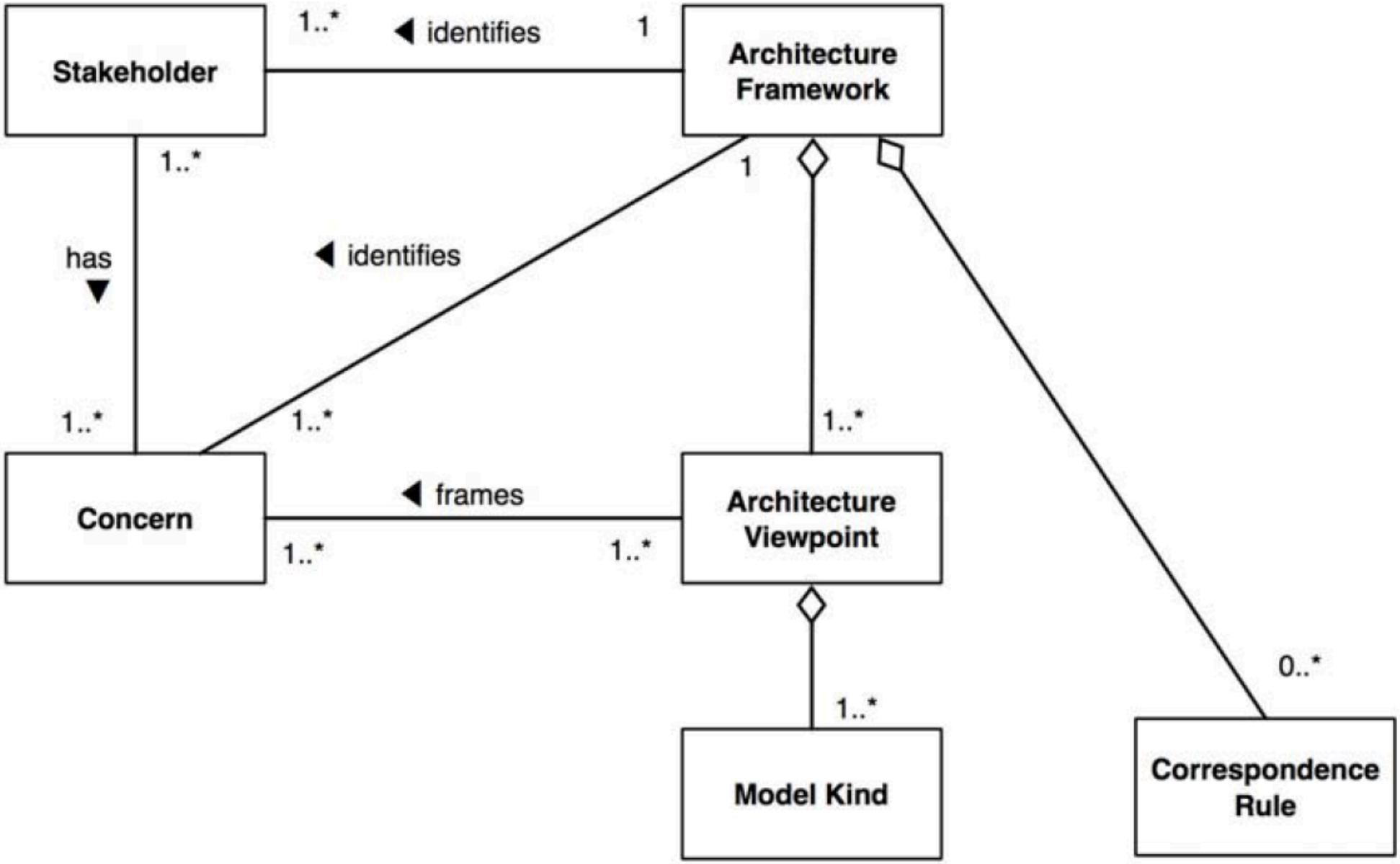
- New version: architecture frameworks & conformance

- www.iso-architecture.org/ieee-1471/

42010:2011


System-of-Interest
exhibits
Architecture
identifies
has interests in
expresses
Stakeholder
identifies
Architecture Description
Architecture Rationale
has
identifies
Concern
Correspondence Rule
Correspondence
addresses
frames
governs
Architecture Viewpoint
Architecture View
Model Kind
governs
Architecture Model

Architecture frameworks



Architecture framework

- An architecture framework includes one or more concerns, one or more stakeholders having those concerns, any correspondence rules
- These are architecture frameworks recognized by the standard: Krutchen 4+1, Siemens' 4 views method, Zachman framework, UKMODAF, TOGAF, RM-ODP, GERAM

Defining a software architecture

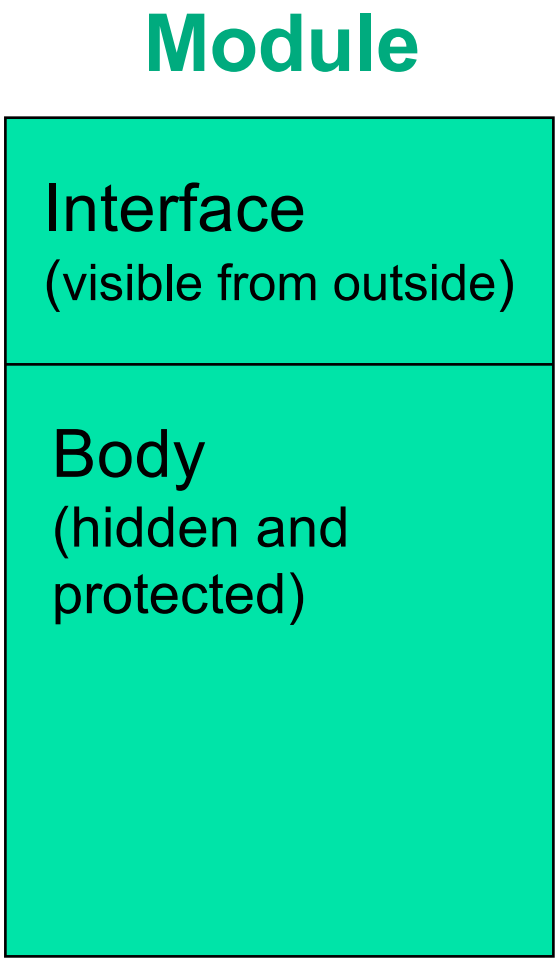
- The definition of the architecture is an important step of system design (i.e., *architectural* design)
- Its aim is the structural decomposition of a system into sub-systems:
  - *Divide et impera* approach: to develop single sub-entities is simpler than to develop an entire system
  - It allows to perform several development activities at the same time (different entities can be developed in parallel)
  - It fosters the modifiability, the reusability, and the portability of a system

Defining the architecture

- The definition of the criteria for the identification of sub-entities is a preliminary step:
  - A typical criterion is the functional one, i.e., a sub-entity may be identified by mapping software functionalities to its parts:
    - E.g., user interface, database access, security management…
- **High Cohesion** and **Low Coupling** among components are key guidelines
  - Each sub-system has to contain cohesive components (e.g., modules providing services which are strictly related one to each other)
  - Minimize interactions among subsystems, that should result scarcely coupled

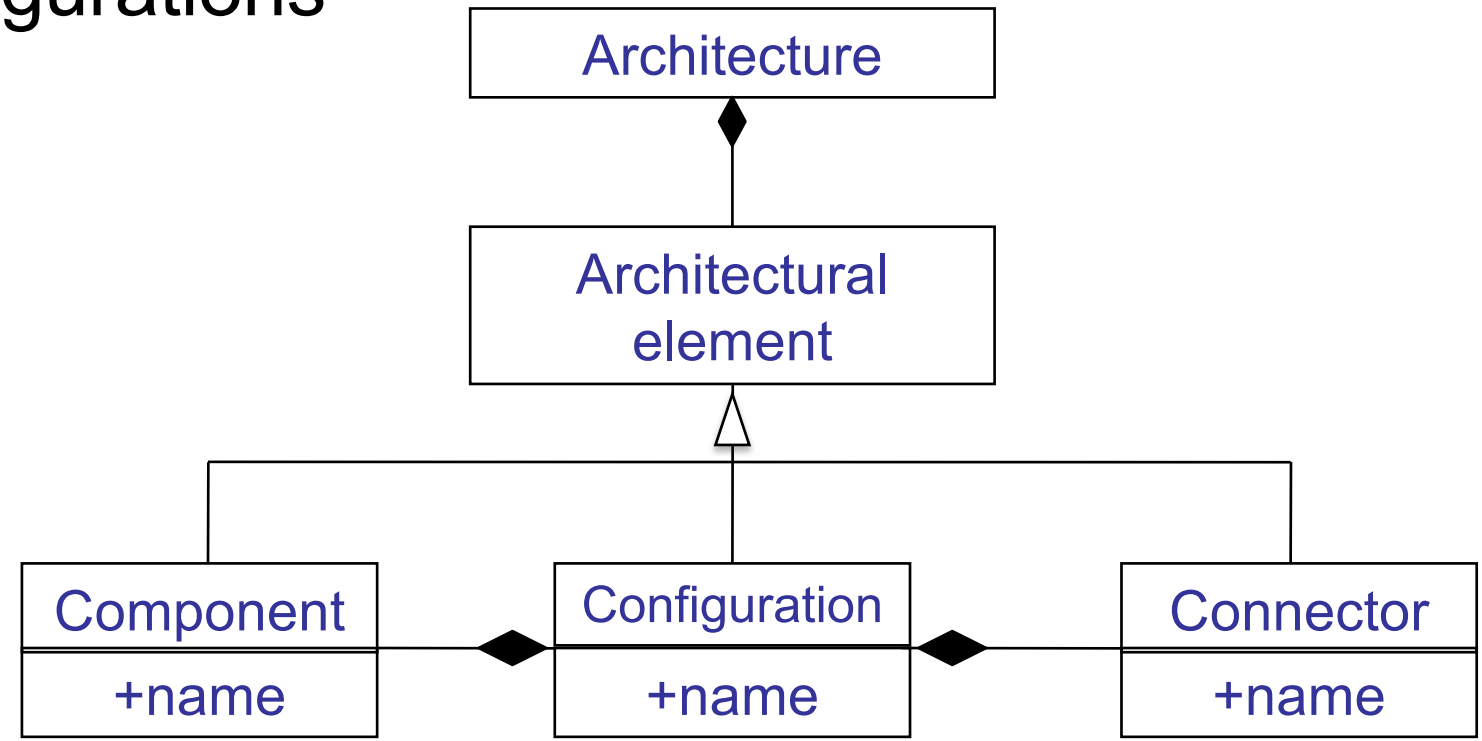
Software module

- A **module** is a piece of software that:
  - Provides an abstraction (on data, on function)
  - Clearly separates the interface from the body
  - Usually makes sense at compilation time rather than at run time
- The **interface** specifies “what” the module is (the abstraction provided) and “how” it can be used
  - The interface is the *contract* between a user and the provider of the abstraction
- The **body** describes “how” the abstraction is implemented



Main elements of a sw architecture

- Components
- Connectors
- Configurations



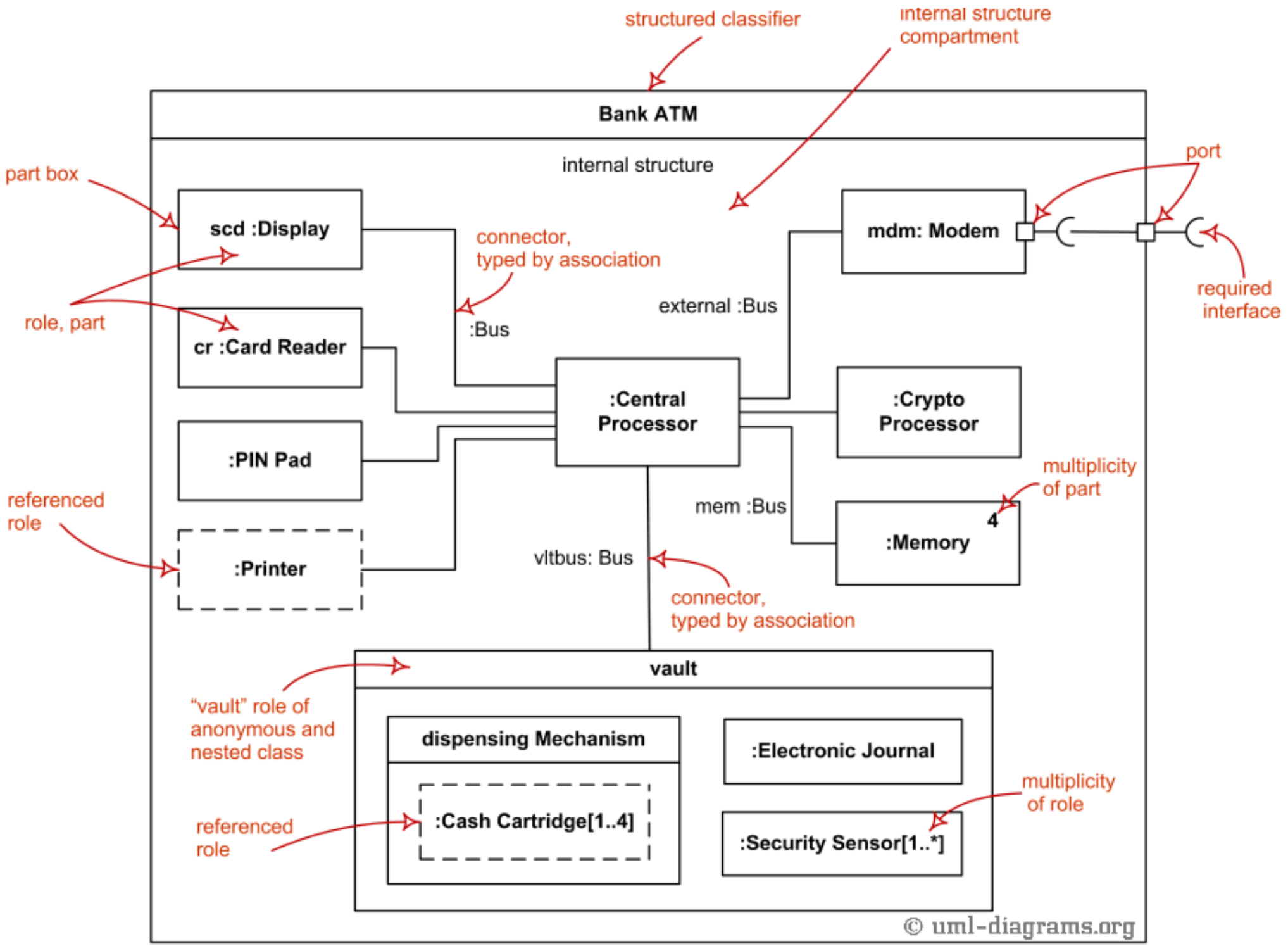
Component

- A software component is a **unit of composition** with contractually specified interfaces (including some *ports*) and explicit *dependencies*
- Example: a web server
- A software component can be deployed independently and is subject to composition by third parties
- A *component model* defines rules (standards) for naming, meta data, behavior specification, implementation, interoperability, customization, composition, and deployment of components
- Example: the CORBA component model

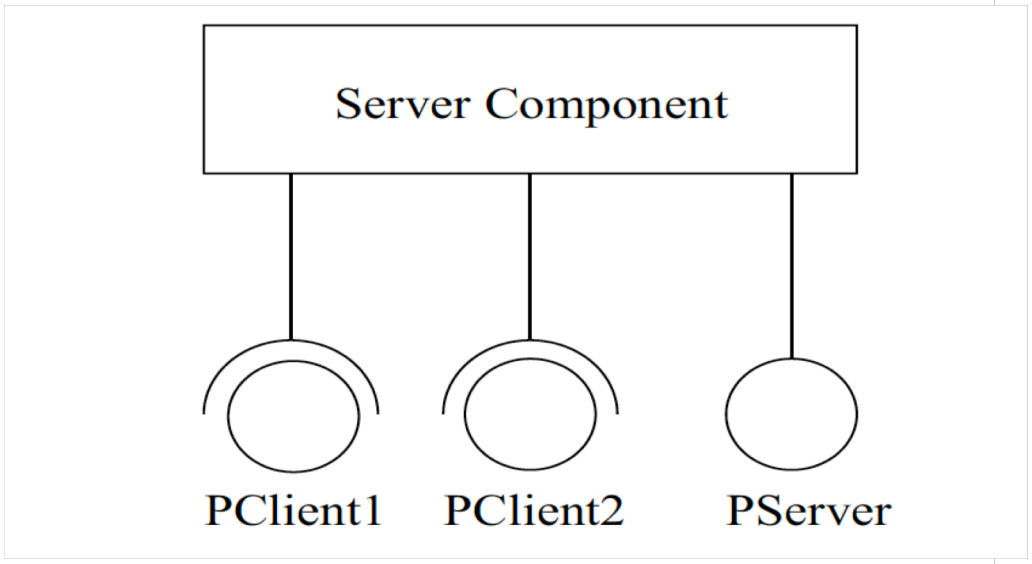
Connector

- A software connector specifies the mechanisms by which components transfer control or data
- Examples: procedure call, protocol, buffer, channel, stream

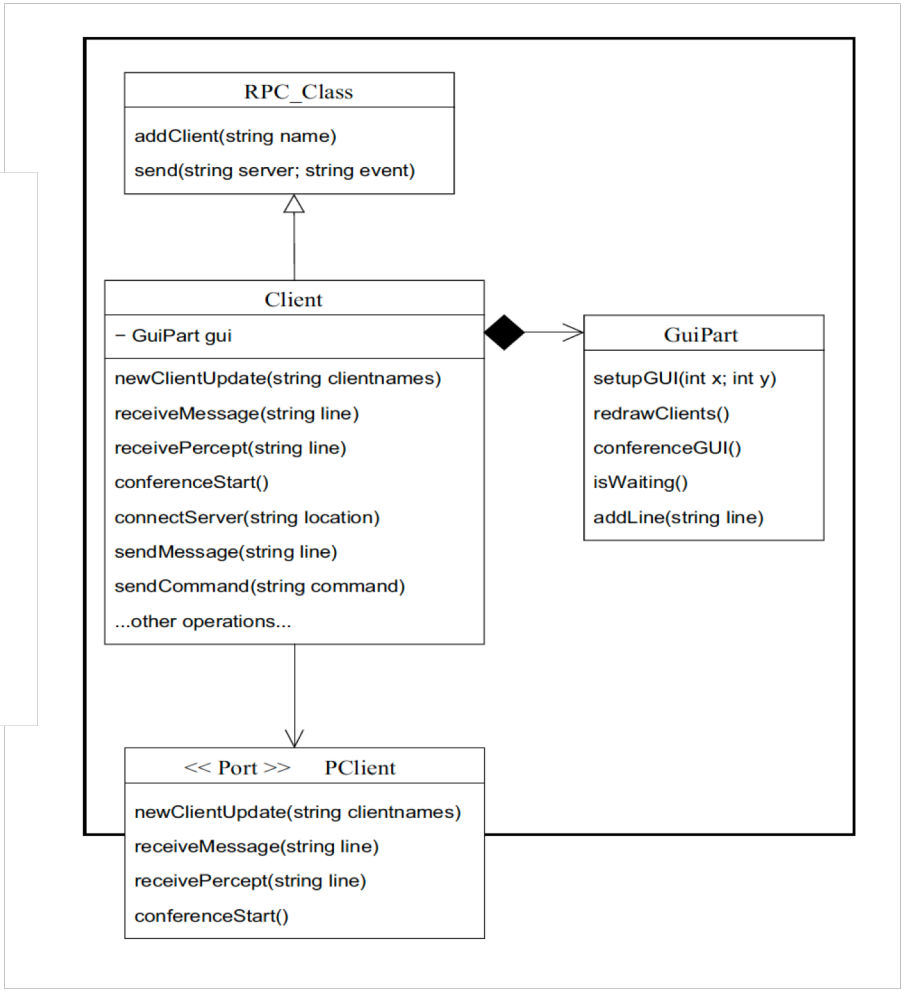
- A connector defines the possible interactions among components: the interfaces of connectors are called **roles**, which are attached to ports of components


structured classifier
internal structure compartment
Bank ATM
internal structure
part box
port
scd :Display
mdm: Modem
connector, typed by association
role, part
external :Bus
required interface
:Bus
cr :Card Reader
:Central Processor
:Crypto Processor
:PIN Pad
multiplicity of part
referenced role
mem :Bus
4
:Memory
:Printer
vltbus: Bus
connector, typed by association
vault
"vault" role of anonymous and nested class
dispensing Mechanism
:Electronic Journal
multiplicity of role
referenced role
:Cash Cartridge[1..4]
:Security Sensor[1..*]
© uml-diagrams.org

UML component model



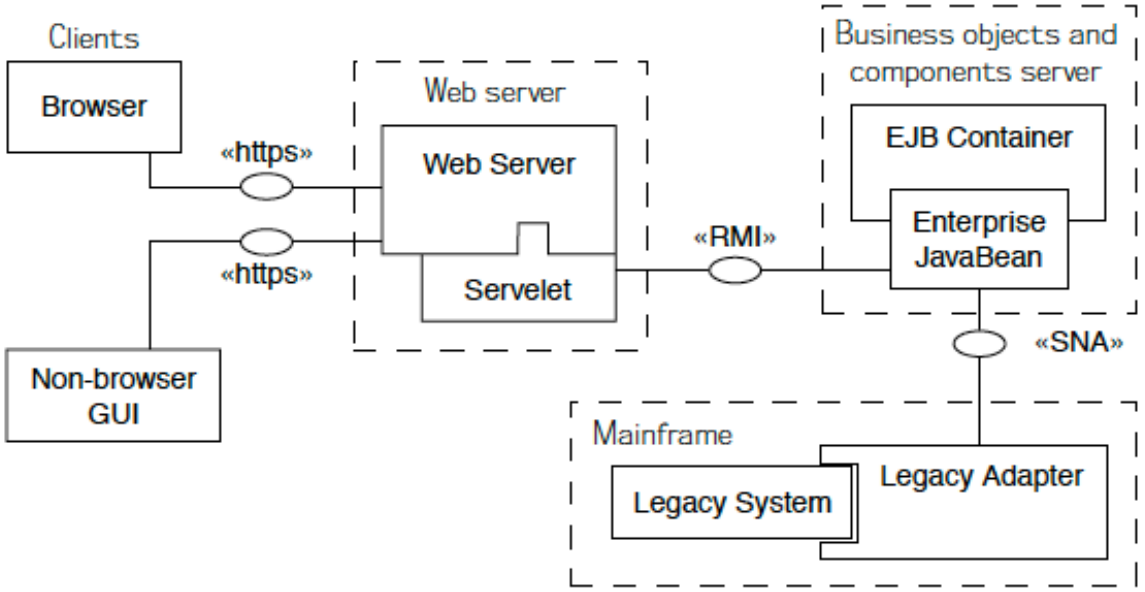
Black box view



White box view

Architectural configuration

- An architectural configuration is a connected graph of components and connectors that describes the structure of a software architecture
- Example:

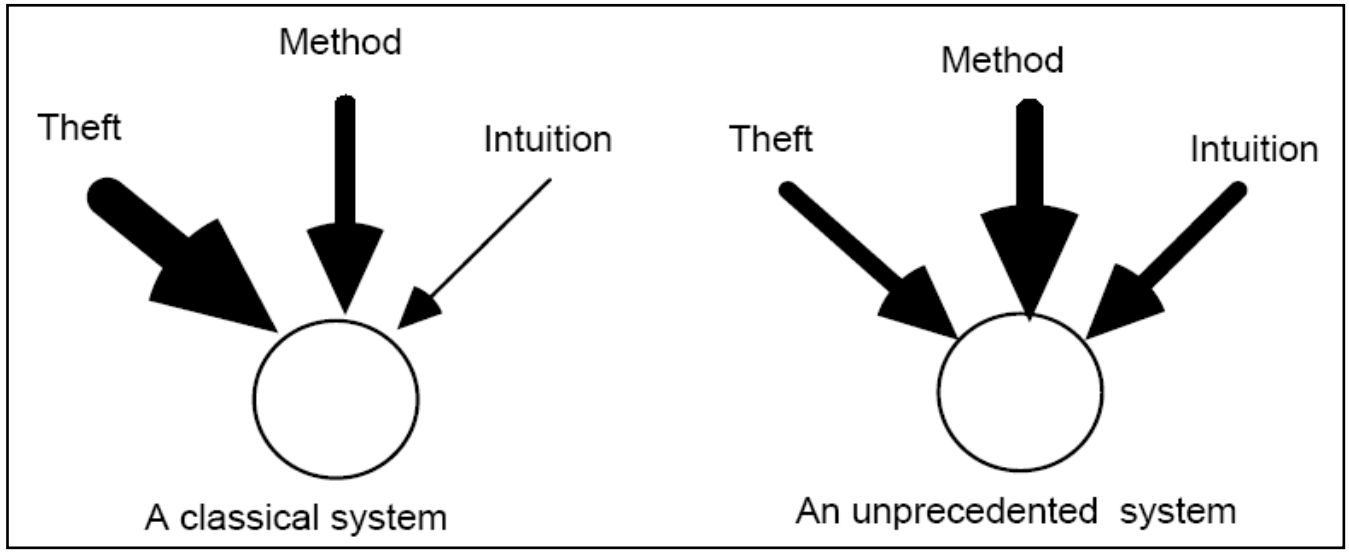


Key architectural design principles

- Simplicity
  - Modularity
  - Low coupling
  - Separation of concerns
- Abstraction
  - Postponing decisions
  - Encapsulation
  - Information hiding
  - Clear interface
  - High cohesion

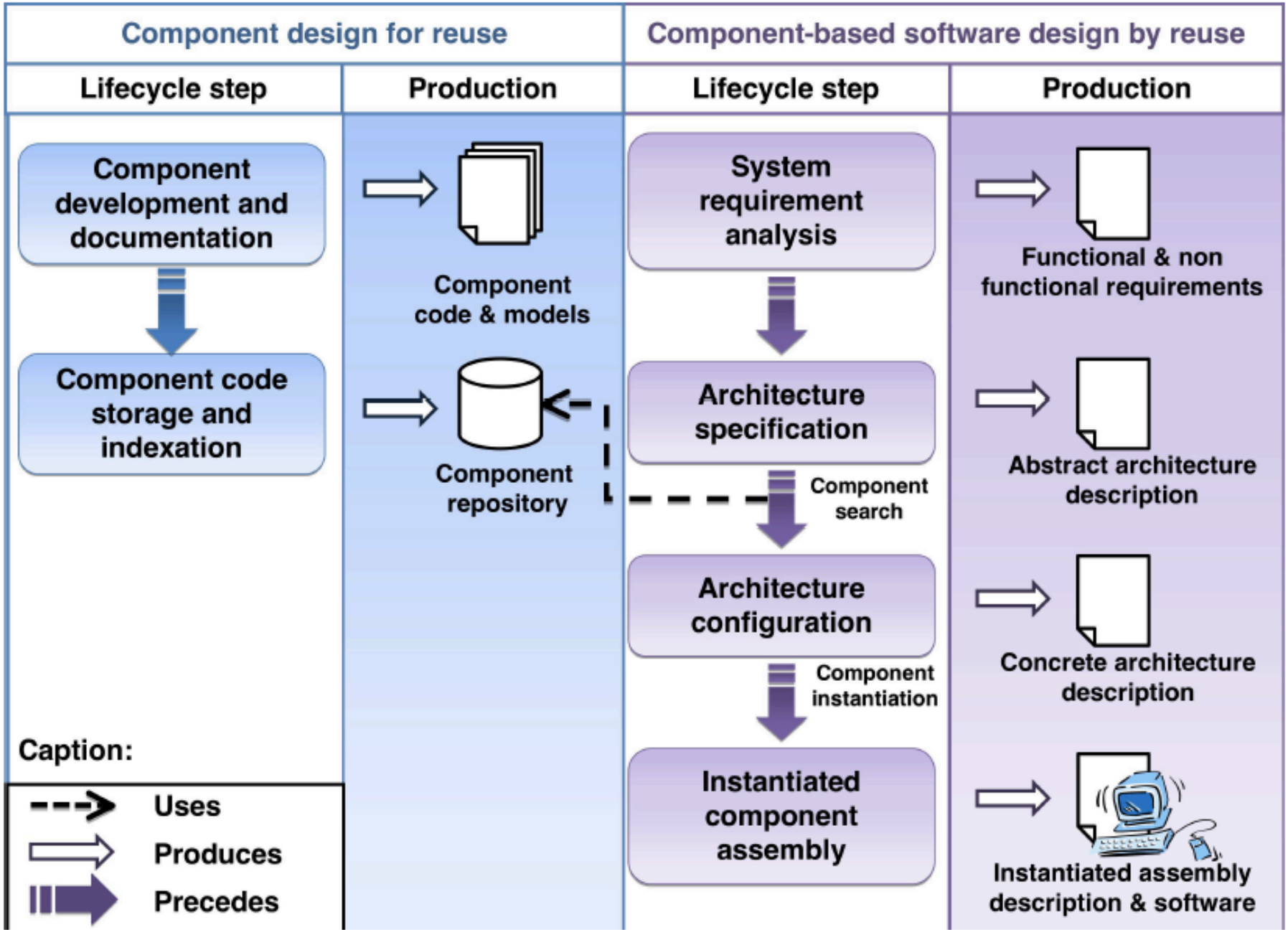
Sources of Architecture

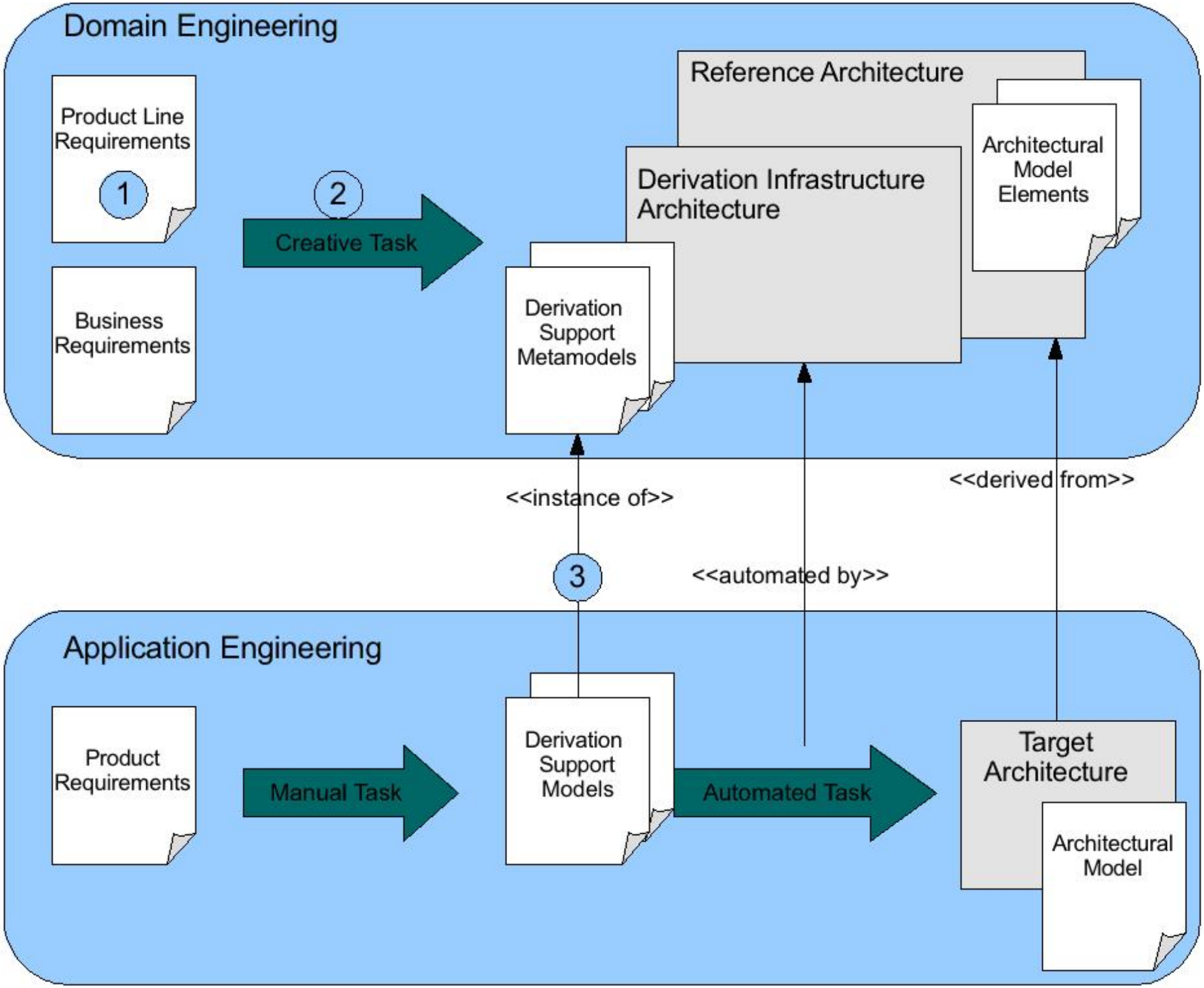
- Theft
  - From a previous system or from technical literature
- Method
  - An approach to deriving the architecture from the requirements
- Intuition
  - The experience of the architect



From "*Mommy, Where Do Software Architectures Come From?*", Philippe Kruchten
1st International Workshop on Architectures for Software Systems, Seattle, 1995

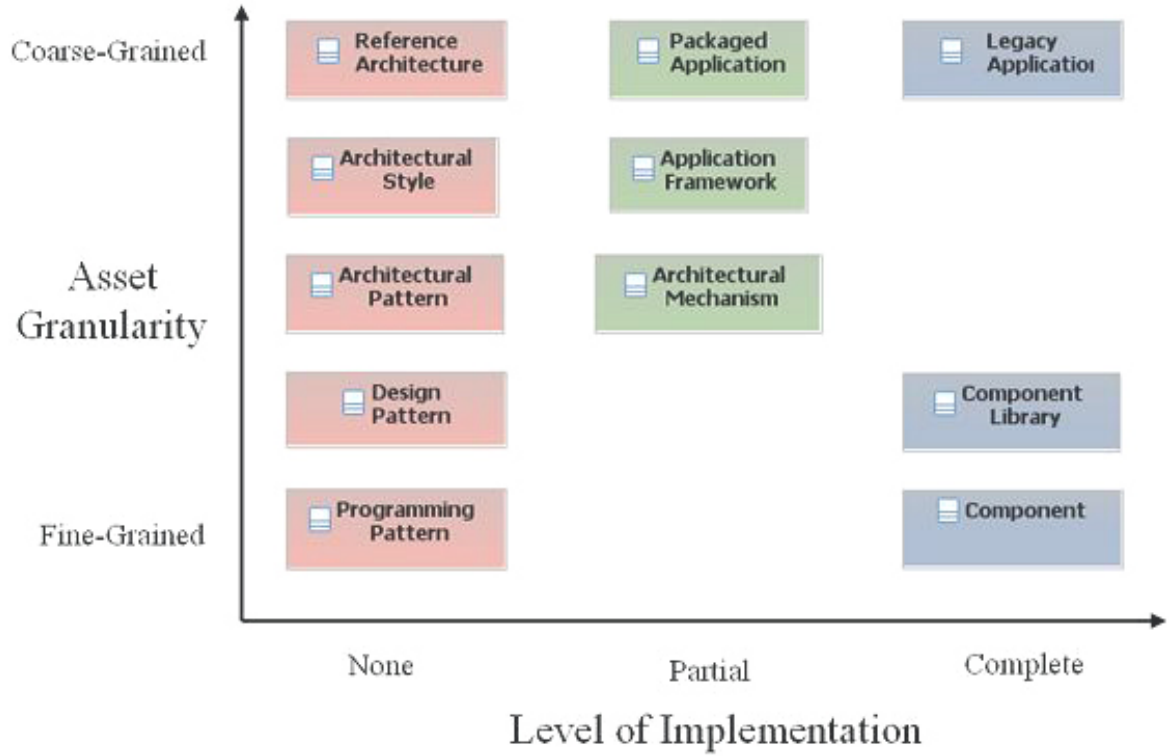
Architecture goal: reuse of components



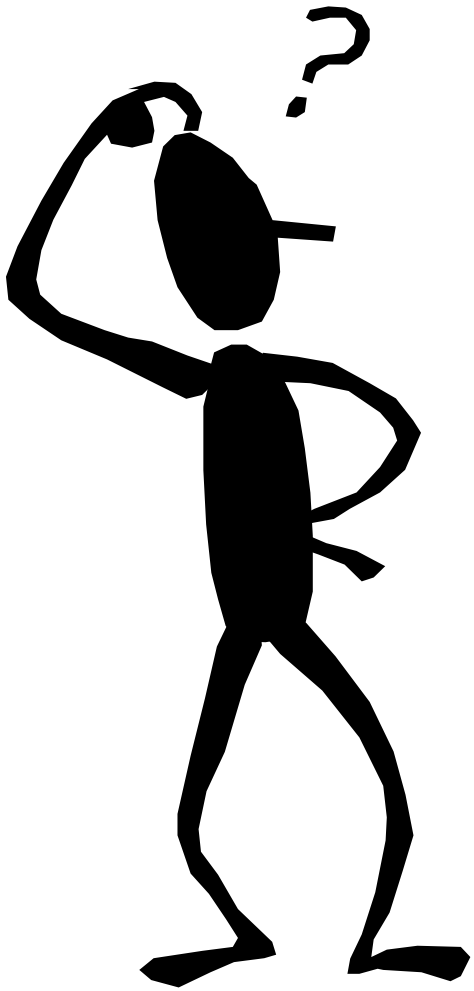

Domain Engineering
Product Line Requirements
1
Business Requirements
2
Creative Task
Reference Architecture
Derivation Infrastructure Architecture
Architectural Model Elements
Derivation Support Metamodels
<<instance of>>
<<derived from>>
3
<<automated by>>
Application Engineering
Product Requirements
Manual Task
Derivation Support Models
Automated Task
Target Architecture
Architectural Model

Architectural assets

Most elements of a software architecture are derived from a previous system of the same kind, another system with similar characteristics, or an architecture found in technical literature



What Types of Architectural Asset are there?



Reference Architecture	Design Pattern
Legacy Application	Architectural Mechanism
Pattern Language	Packaged Application
Development Method	Reference Model
Architectural Decision	Programming Pattern
Pattern	Component Library
Component	Architectural Pattern
Architectural Style	Application Framework

Pattern

- *[A pattern is] a common solution to a common problem in a given context.* [UML User Guide]
- Pattern types
  - Architectural Patterns
    - Distribution patterns
    - Security Patterns
    - …
  - Design Patterns
  - Programming Patterns
  - Requirements Patterns
  - Testing Patterns
  - Project Management Patterns
  - Process Patterns
  - Organizational Patterns
  - …

Architectural Pattern

- *An architectural pattern expresses a fundamental structural organization schema for software systems. It provides a set of predefined subsystems, specifies their responsibilities, and includes rules and guidelines for organizing the relationships between them.* [Buschmann]
- Example:

Pattern: Layers

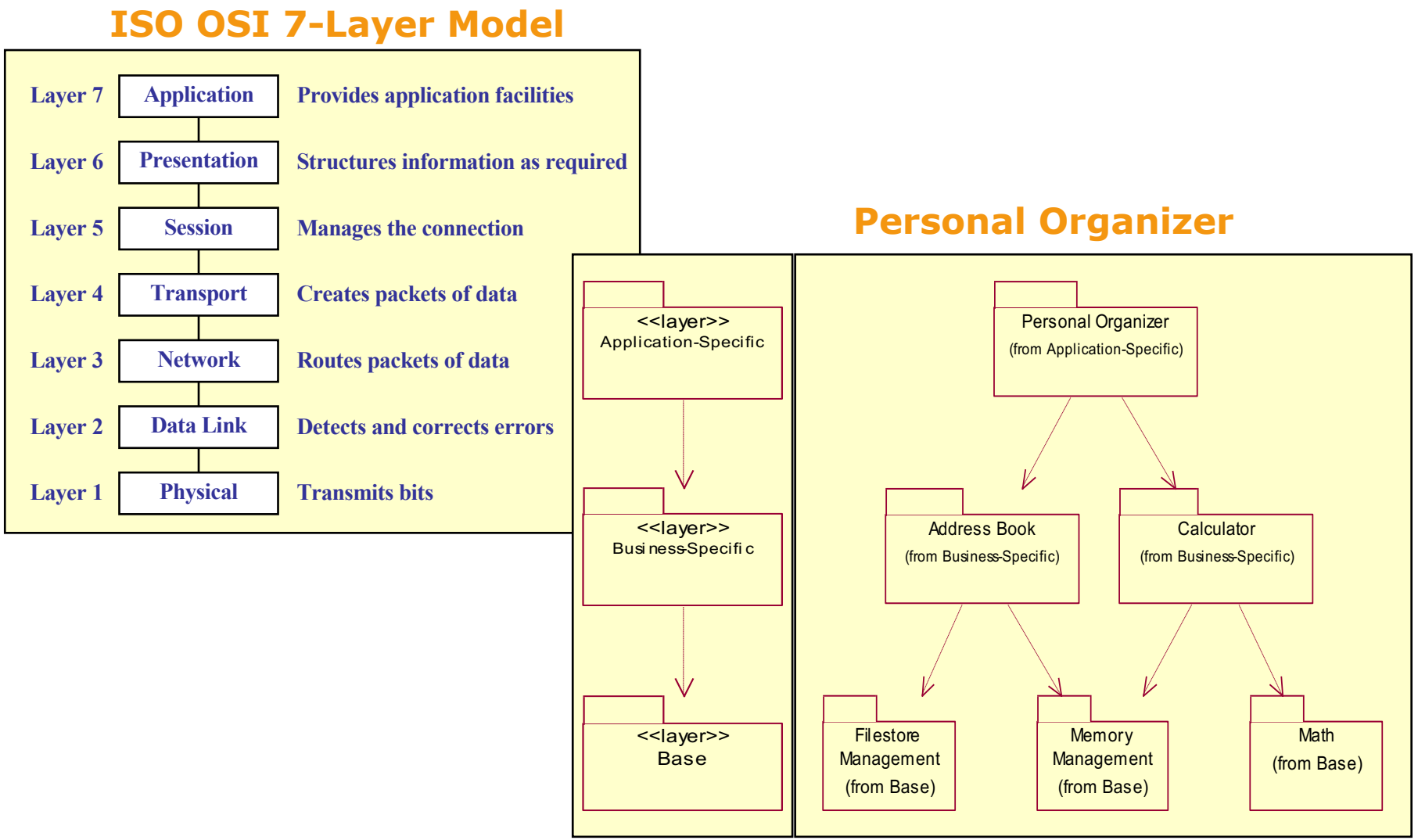
Context
A system that requires decomposition

Problem
High-level elements rely on lower-level elements and the following forces must be balanced:

- Interfaces should be stable
- Parts of the system should be exchangeable
- Source code changes should not ripple through the system

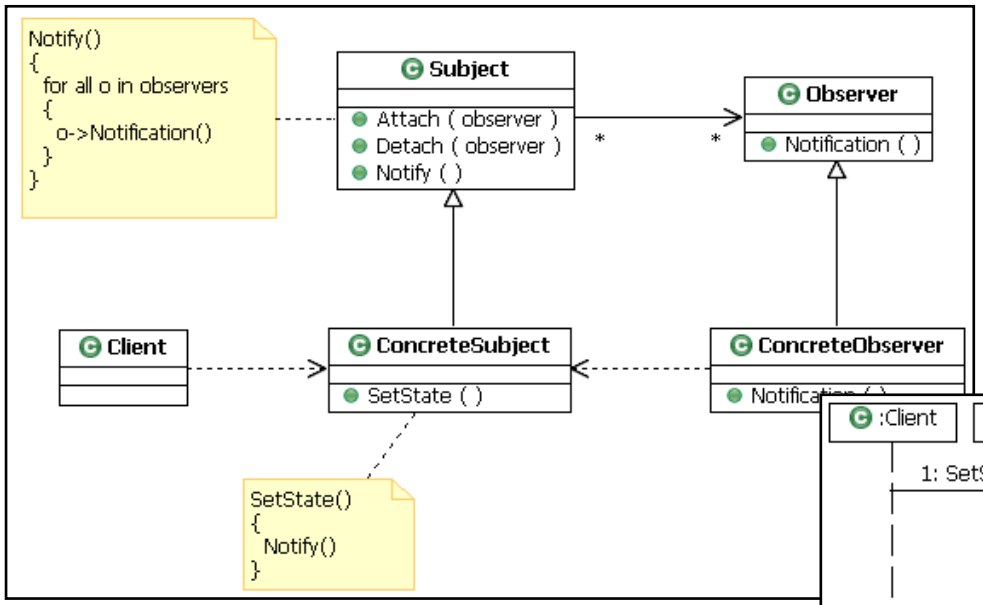
Solution
Structure the system into layers

Architectural pattern – Layers

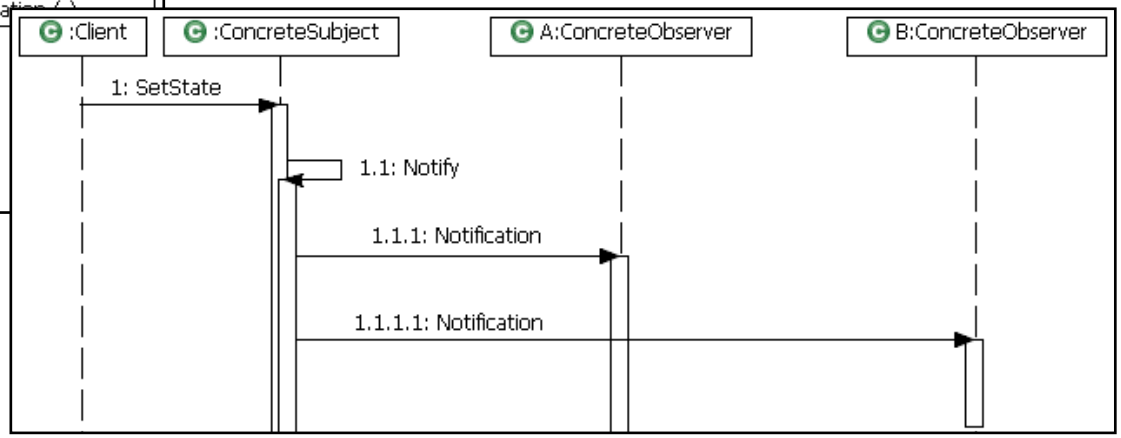


Design Pattern

- *A design pattern provides a scheme for refining the subsystems or components of a software system, or the relationships between them. It describes a commonly-recurring structure of communicating components that solves a general design problem within a particular context.* [Gamma]



Observer Pattern



Programming Pattern

- *An idiom is a low-level pattern specific to a programming language. An idiom describes how to implement particular aspects of components or the relationships between them using the features of the given language.* [Buschmann]

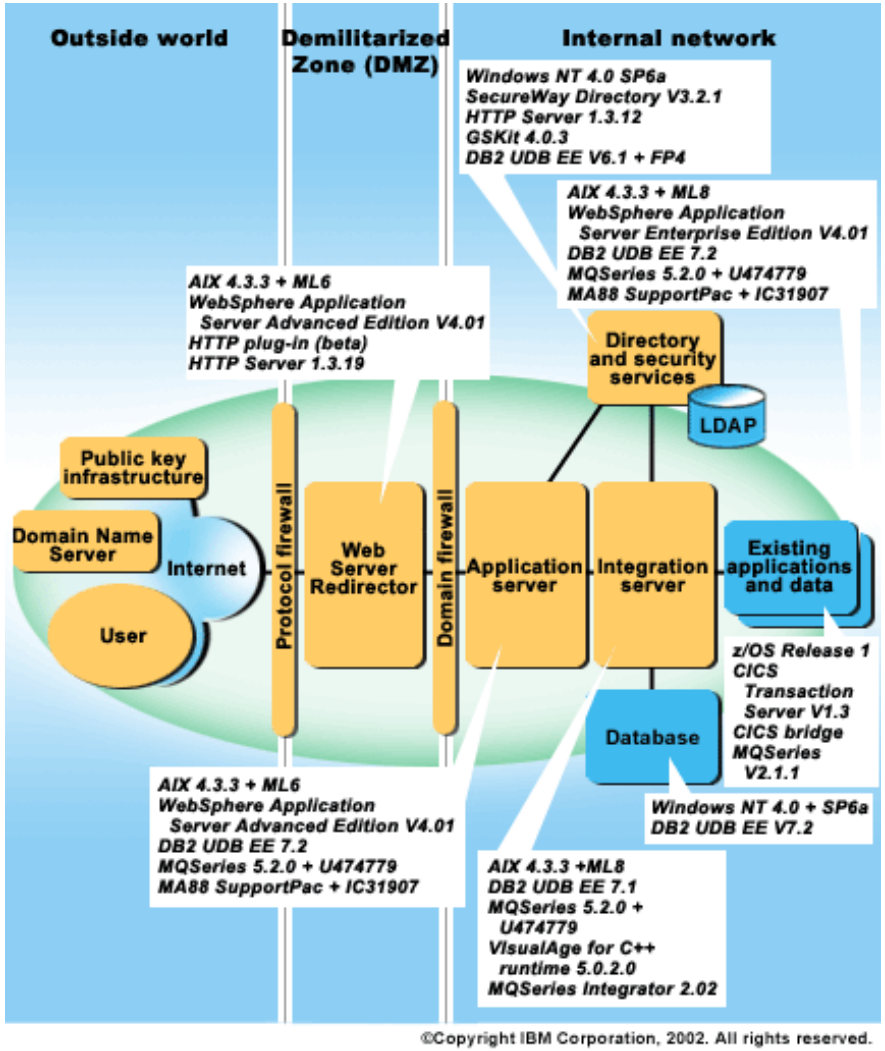
```
// Swap the values of 2 variables
temp = a;
a = b;
b = temp;
```

Architectural Style

- *[An architectural style] defines a family of systems in terms of a pattern of structural organization. More specifically, an architectural style defines a vocabulary of components and connector types, and a set of constraints on how they can be combined. [Shaw]*
- Client-server
  - Supports the physical separation of client-side processing (such as a browser) and server-side processing (such as an application server that accesses a database)
- Event-based
  - Promotes a publish-subscribe way of working, applied strategically across large areas of the architecture
- Pipes-and-filters
  - A series of filters that provide data transformation, and pipes that connect the filters. Examples include compilers, signal processing

Pattern Language

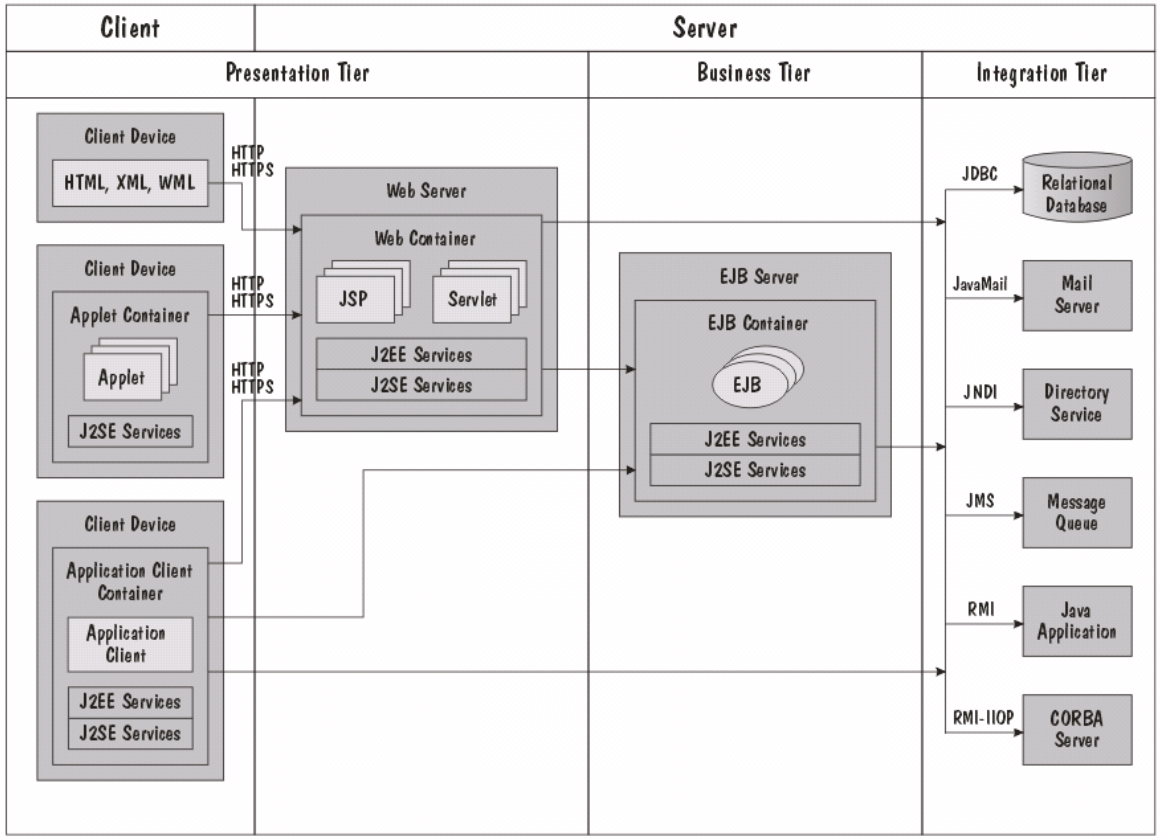
- *A pattern language defines a collection of patterns and the rules to combine them. Pattern languages are often used to describe a family of systems*
- IBM Patterns for e-Business
  - A set of architectural patterns that describe various web-based applications
  - Includes a pattern selection process that drives:
    - Selection of a business, integration or composite pattern
    - Selection of application patterns
    - Selection of runtime patterns
    - Identification of product mappings



*See http://www.ibm.com/developerworks/patterns

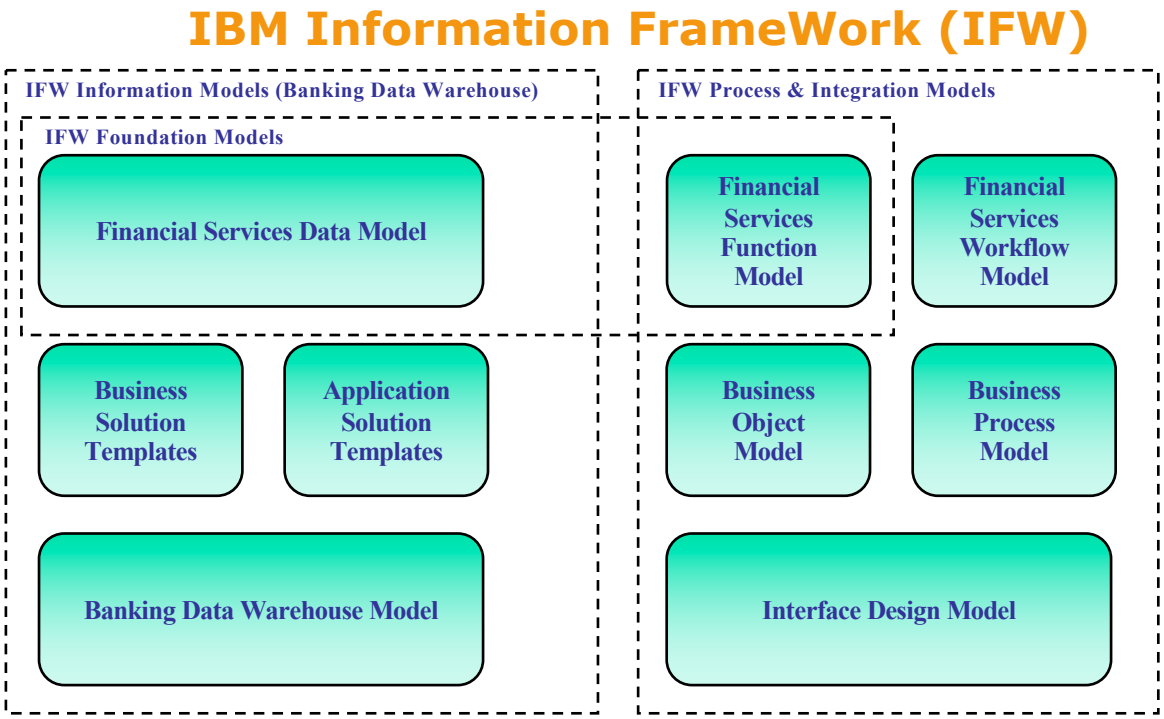
Reference Architecture

- *A reference architecture is an architecture representation of a particular domain of interest. It typically includes many different architectural patterns, applied in different areas of its structure*
- Examples include JEE and .NET

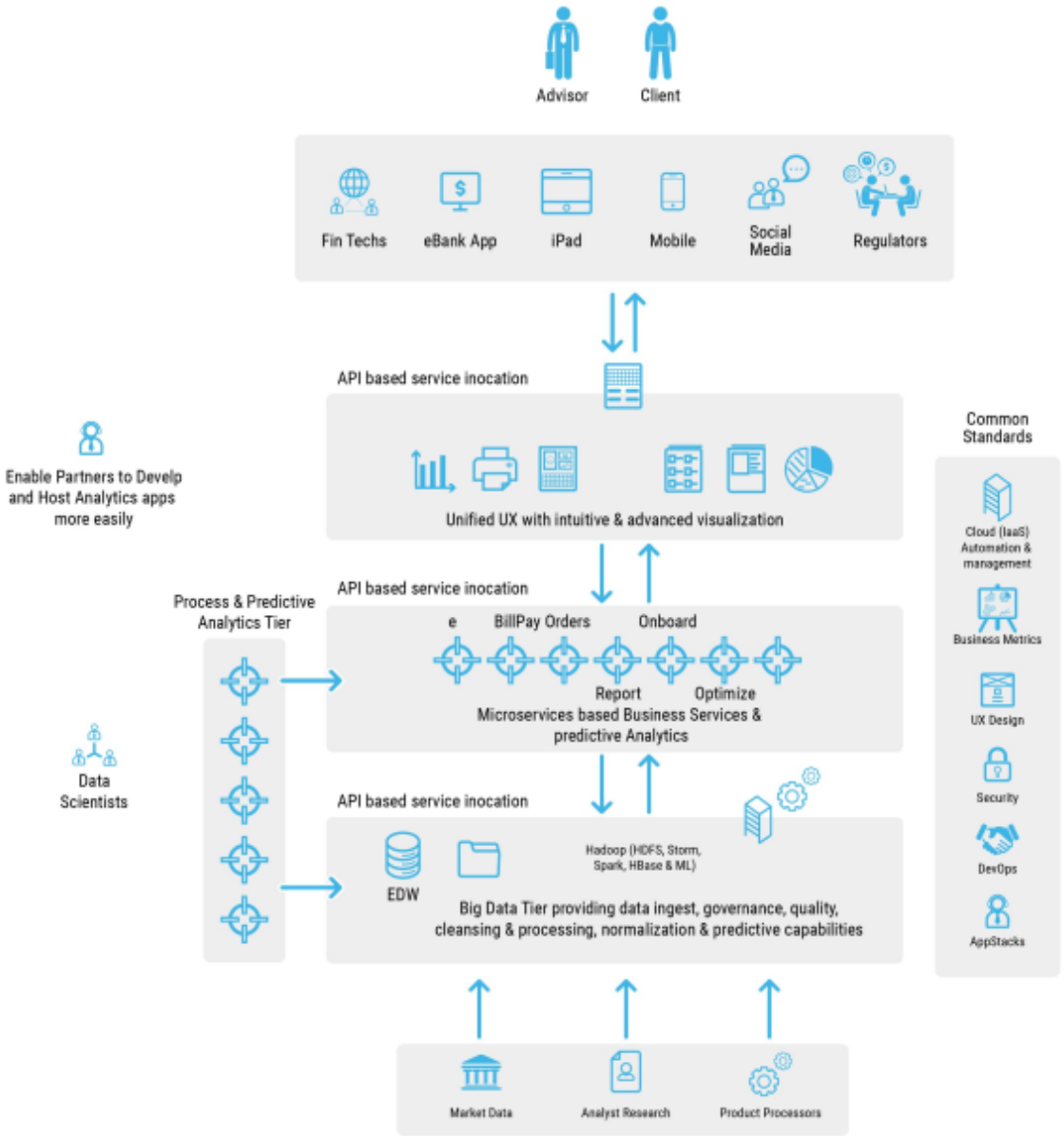


Reference Model

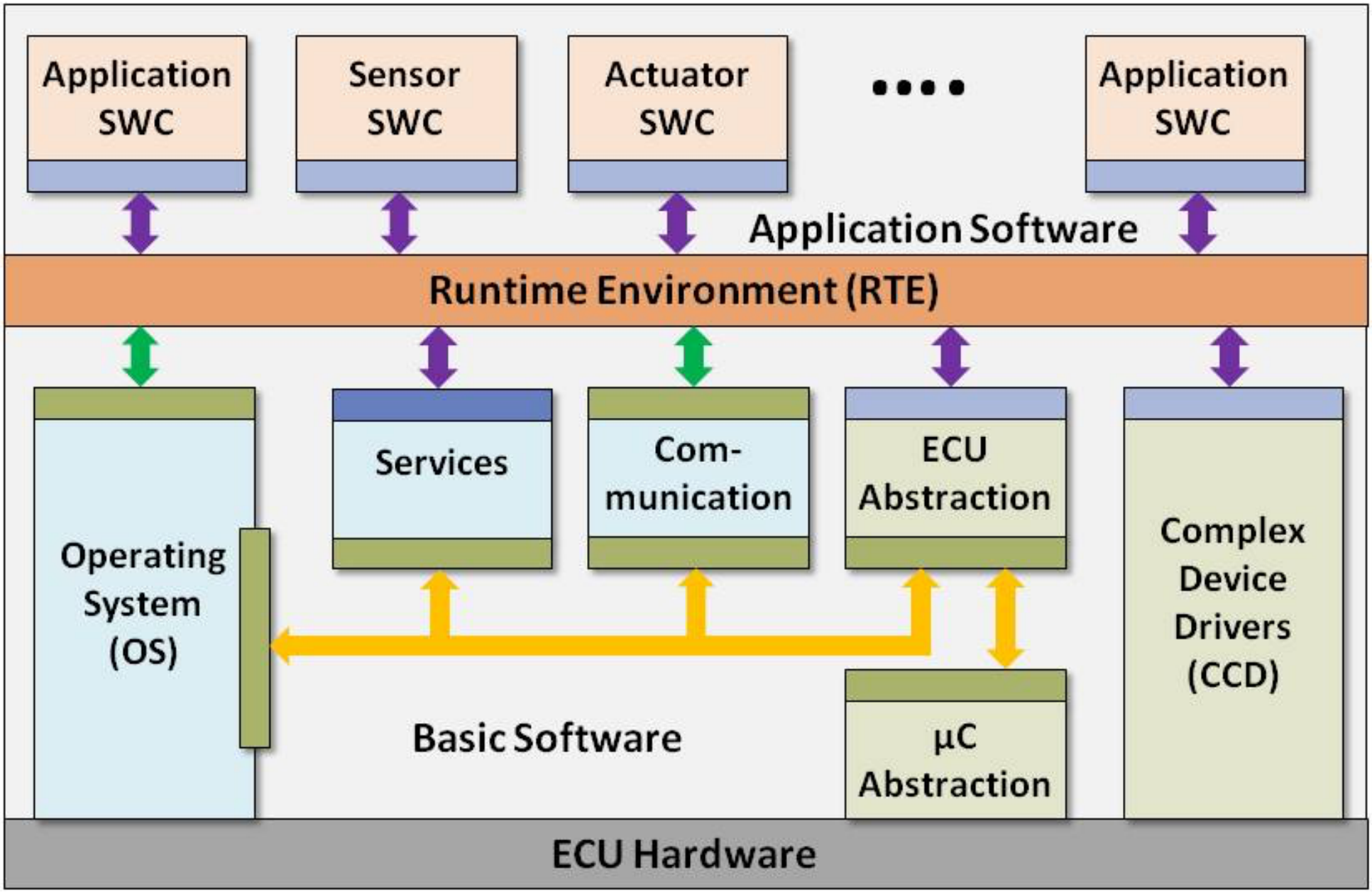
- *A reference model is an abstract representation of entities, their relationships and behavior, in a given domain of interest, and which typically forms the conceptual basis for the development of more concrete elements*
- Examples: a business model, an information model, or a glossary of terms



Strawman Reference Architecture for Open Bank Standard

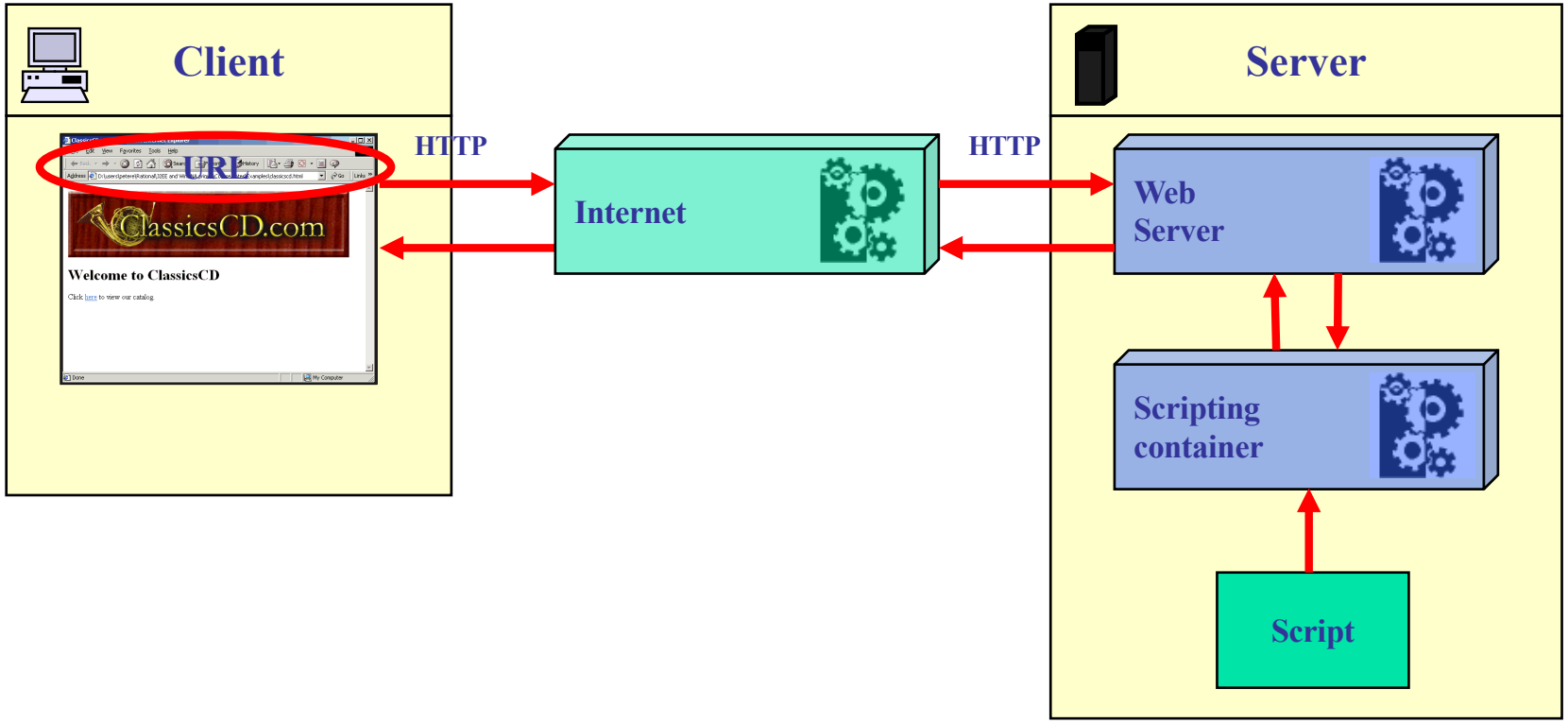


Example: automotive sw architecture (AUtOSAR)



Application Framework

- *An application framework represents the partial implementation of a specific area of an application*
- Most widely-known frameworks support user interfaces
  - Java Server Pages
  - ASP.NET



Architectural Mechanism

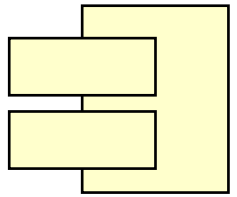
- *Architectural mechanisms represent common concrete solutions to frequently encountered problems. They may be patterns of structure, patterns of behavior, or both.* [RUP]
- Often described as
  - “the mechanism for achieving X”
  - “this element is underpinned by mechanism Y”
- Examples
  - Persistency mechanism
  - Error logging mechanism
  - Communication mechanism
  - Shopping cart

Packaged application

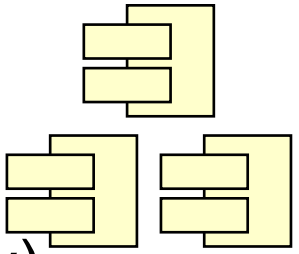
- *A packaged application is a Commercial-Off-The-Shelf (COTS) product* available in the market that can be bought "as is"
- *These applications are usually large grained, that is they provide a significant amount of capability (and reuse)*
- Examples
  - Customer Relationship Management (CRM) application (e.g. Siebel)
  - Enterprise Resource Planning (ERP) application (e.g. SAP)
- The amount of required custom development is minimal
- Primary focus is on configuring the application

Component & component library

- Component examples
  - GUI widget (such as a table)
  - Service



- Component library examples
  - Class libraries (e.g. Java class library)
  - Procedure libraries

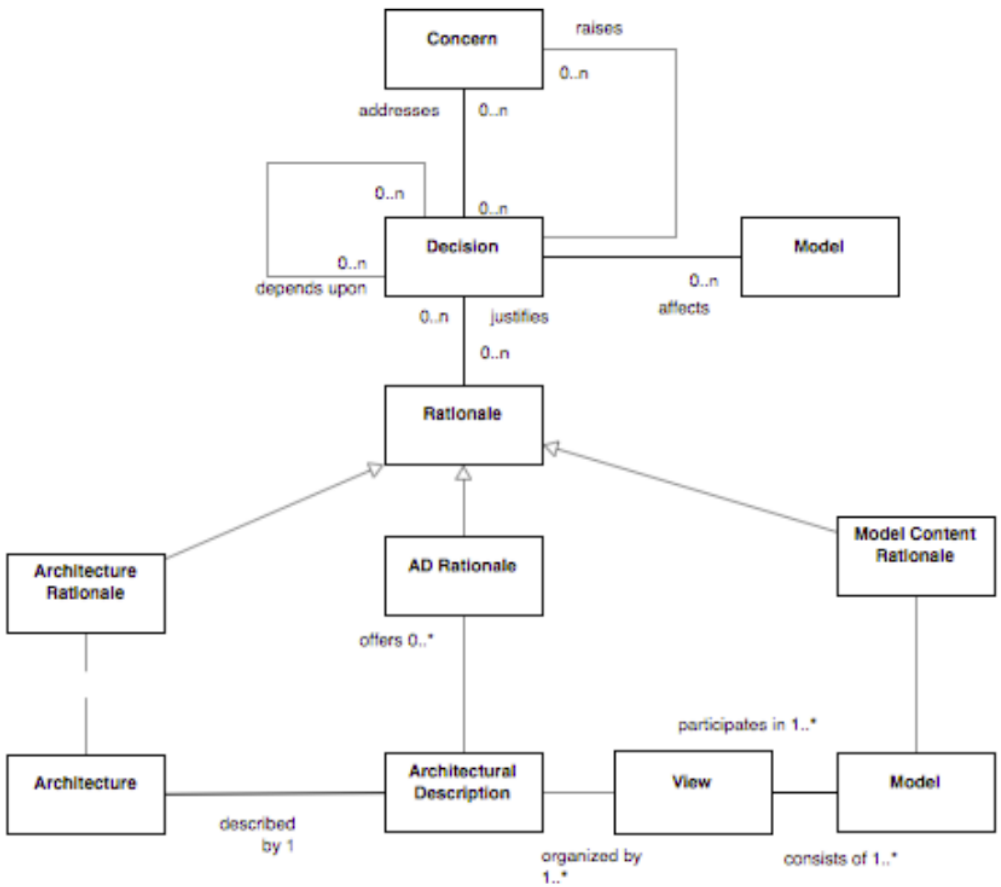


Legacy Application

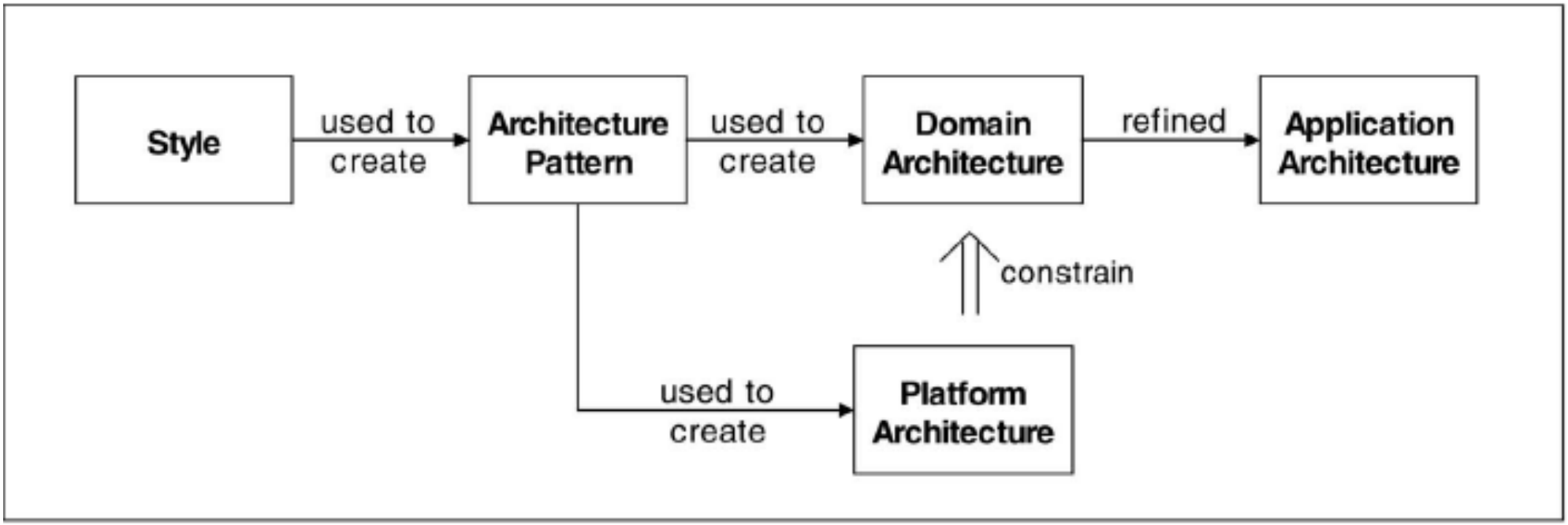
- *A legacy application is a system that continues to be used because the owning organization cannot replace or redesign it*
- Usually no way to have past (regression) tests
- Tends to be a focus on integration rather than new development
- Often results in a focus on enterprise application integration (EAI)

Architectural Decision

- *[Architectural decisions are] conscious design decisions concerning a software system as a whole, or one or more of its core components. These decisions determine the non-functional characteristics and quality factors of the system. [Zimmermann]*
- Decision rationale may come from experience, method or some other asset



A map of architectural concepts



Summary: what is a software architecture?

An architecture

- defines the form of software
- defines the function of software
- focuses on significant elements
- reuses and conforms to an architectural style
- balances the needs of its stakeholders
- is influenced by its environment
- influences its environment
- influences its development team

Quotation

Whether something is part of the architecture is entirely based on
whether the developers think it is important.
Architecture is a social construct
because it doesn't just depend on the software,
but on what part of the software
is considered important by group consensus.

Martin Fowler

Self questions

- What is a software architecture?
- What is an architectural concern?
- What is an architectural view?

- What is a component?
- What is a connector?

- What is an architectural framework?
- What is an architectural asset?

Suggested readings

- `www.iso-architecture.org/ieee-1471/`
- Perry & Wolf, Foundations for the study of software architecture, *ACM Sw Eng Notes*, 17:4(40-52), 1992
- Cervantes & Kazman, *Designing Software Architectures*, AddisonWesley, 2016

References

- **Shaw** & **Garlan** *Software Architecture: Perspectives on an Emerging Discipline*, Prentice Hall, 1996
- **Bass, Clemens** & **Kazman**, *Software Architecture in Practice*, 3rd ed, Addison Wesley, 2013
  `softarchpract.tar.hu` 2nd ed
- **Clements**, *Documenting Software Architectures: Views and Beyond*, The SEI Series in Software Engineering, 2010
- **Rozanski & Woods**, *Software Systems Architecture*, Addison-Wesley, 2012
- **Garland**, *Large-Scale Software Architecture: A Practical Guide using UML*, Addison-Wesley, 2005
- **Mistrik**, *Relating System Quality and Software Architecture*, MorganKaufman 2014

Interesting sites

- www.sei.cmu.edu/architecture
- www.softwarearchitectureportal.org
- www.viewpoints-and-perspectives.info
- msdn.microsoft.com/en-us/library/ee658093.aspx
- www.bredemeyer.com/definiti.htm
- www.booch.com/architecture
- www.ivencia.com/index.html?/softwarearchitect/
- stal.blogspot.com
- softwarearchitecturezen.blogspot.com

Questions?