Software Architecture Views

Lecture 06

BIL428 Software Architectures

Mustafa Sert
Asst. Prof.
msert@baskent.edu.tr

Department of Computer Engineering, Başkent University
Ankara 06810 TURKEY

Agenda..

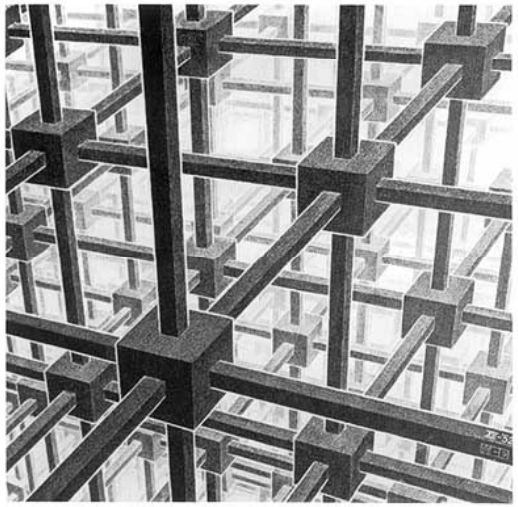
- Motivation for Multiple Views
- The 4+1 View Model
  - Logical view
  - Process view
  - Development view
  - Physical view
  - Scenarios
- The Iterative Process

Need for Multiple Views..

- Remember different stakeholders of a SW product

- Software architecture provides a common medium for communication among stakeholders

- Multiple stakeholders have multiple concerns, so that needed multiple architectural views
- A complex system can usually NOT represented with a single architectural view



Solution

- Using several concurrent *views* or *perspectives*, with different notations each one addressing one specific set for concerns

Architectural Views

- **View:**
  - A representation of a system from the perspective of one or more **concerns** which are held by one or more **stakeholders**
- **Viewpoint:**
  - A pattern or template from which to construct individual views
- **Concerns:**
  - Stakeholder's interest which pertain to the development, maintenance, operation or any other character of the system. Some examples:
    - Performance, reliability, functionality, adaptability, portability, maintainability, Cost, …
- View represents the system from a specific viewpoint
  - Complete and Consistent – Relative to that viewpoint

Kruchten's 4+1 View Model

- Example of viewpoint description
- 5 concurrent views addressing stakeholders/concerns
- Different view for different stakeholders
- Each view represents different concern(s)
- Each view is represented by different (modeling) elements

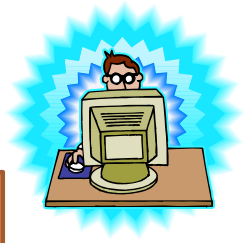
Paper published in IEEE Software 12 (6)
November 1995, pp. 42-50

Architectural Blueprints—The "4+1" View Model of Software Architecture

Philippe Kruchten
Rational Software Corp.

4+1 View Model of Architecture

End User,
Functionality



Programmers
& software
managers

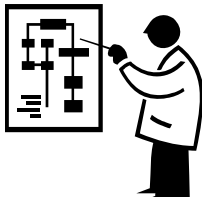
Logical view

Development
view

Scenarios

Process View

Physical View



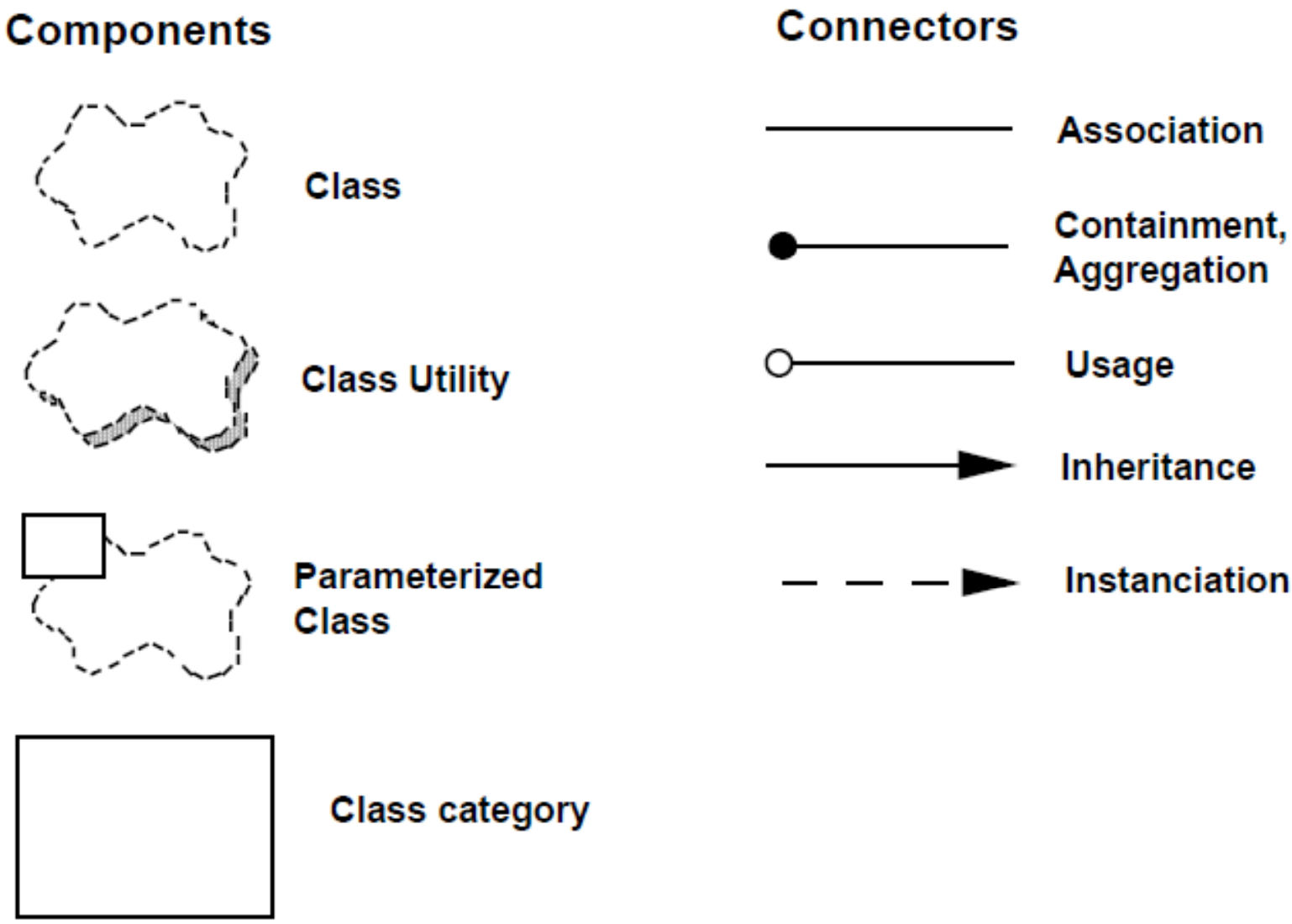
Integrators
Performance
Scalability

System Engineers
Topology
Communications

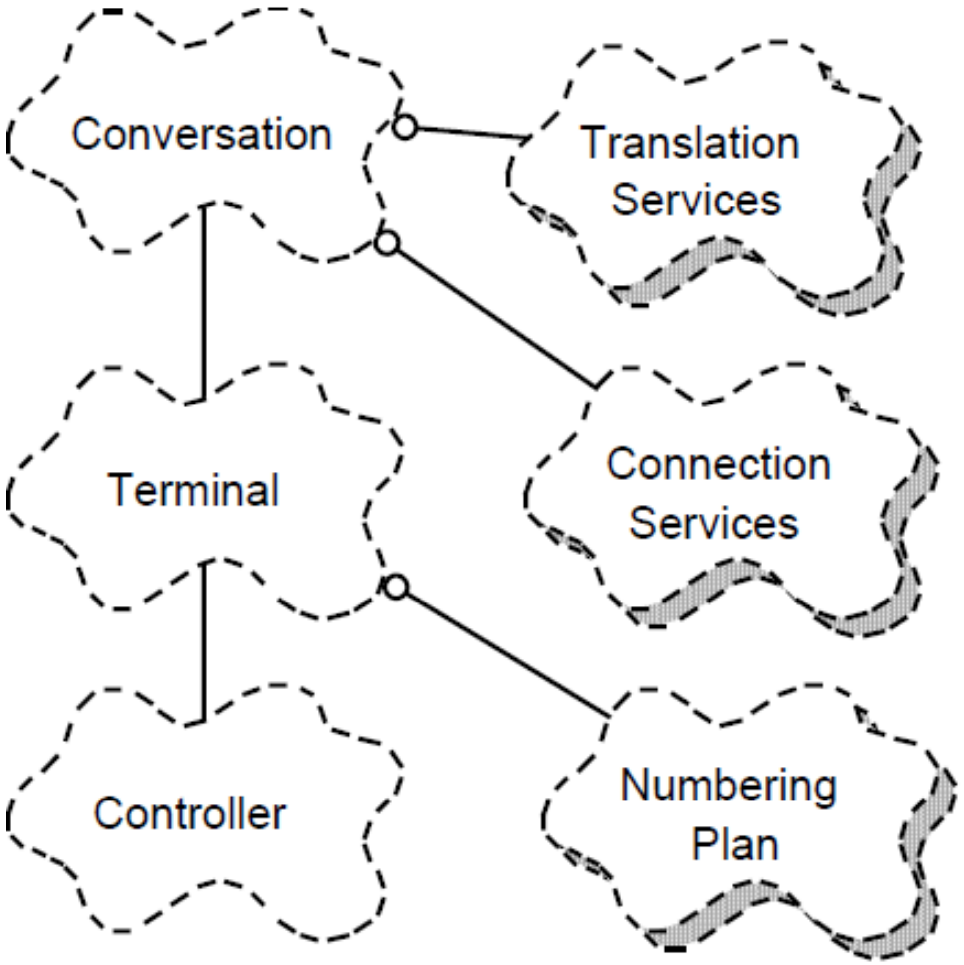
Logical Viewpoint

- **Concern**
  - Functional requirements
- **Stakeholder**
  - End-user
- **Components**
  - Class
- Set of key abstractions of the domain

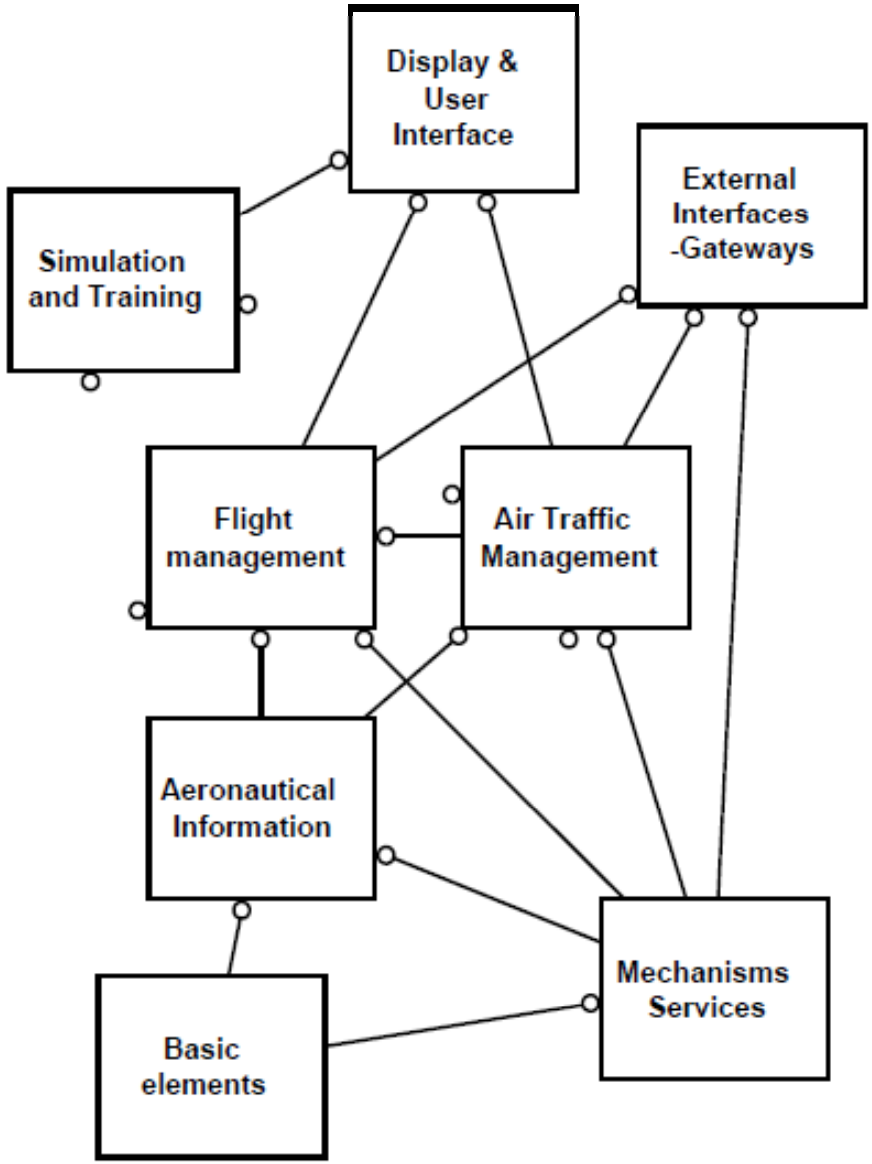
Logical Viewpoint – Notation



Logical View – Example



a. Logical blueprint for the Télic PABX .

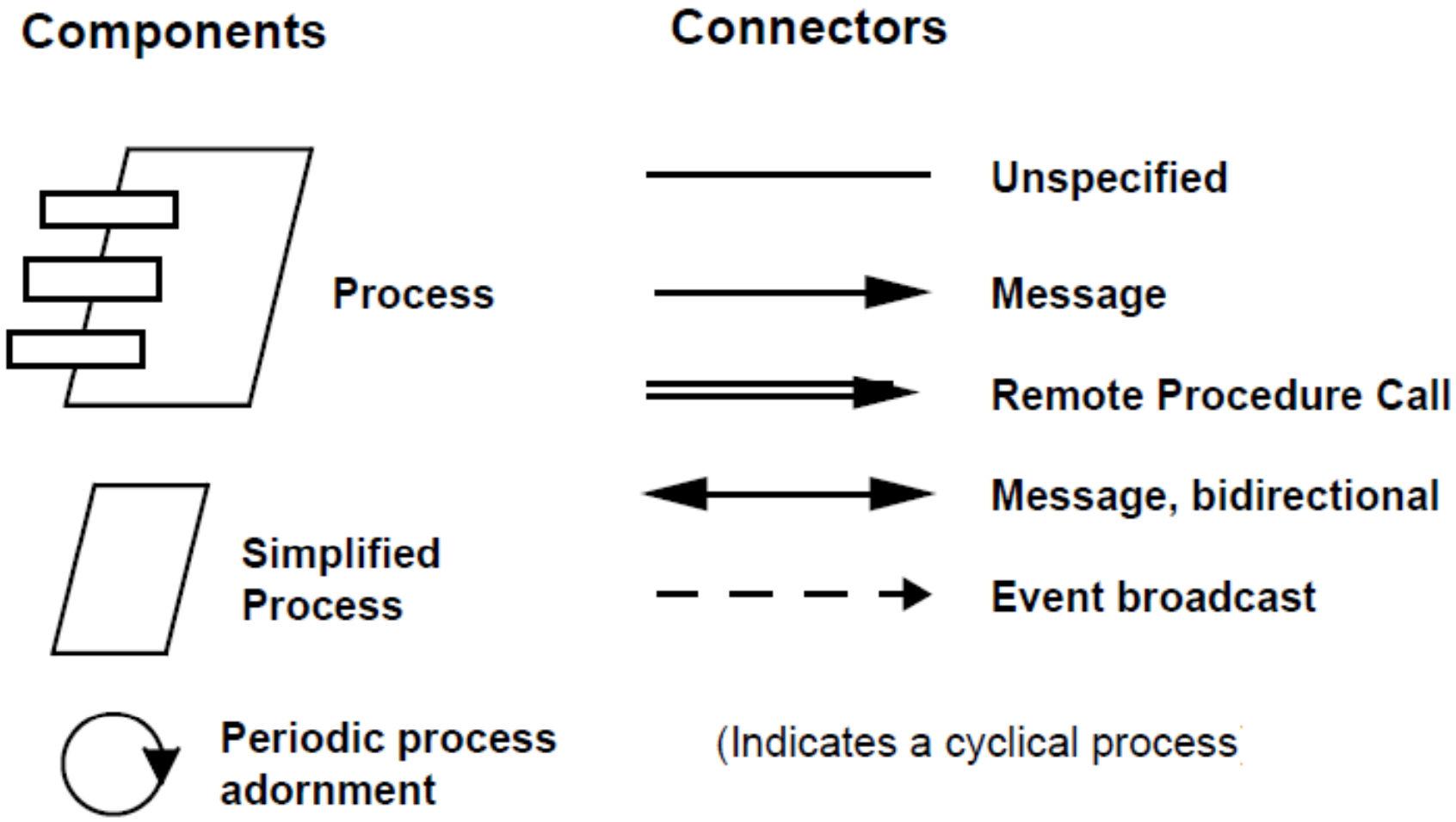


b. Blueprint for an Air Traffic Control System

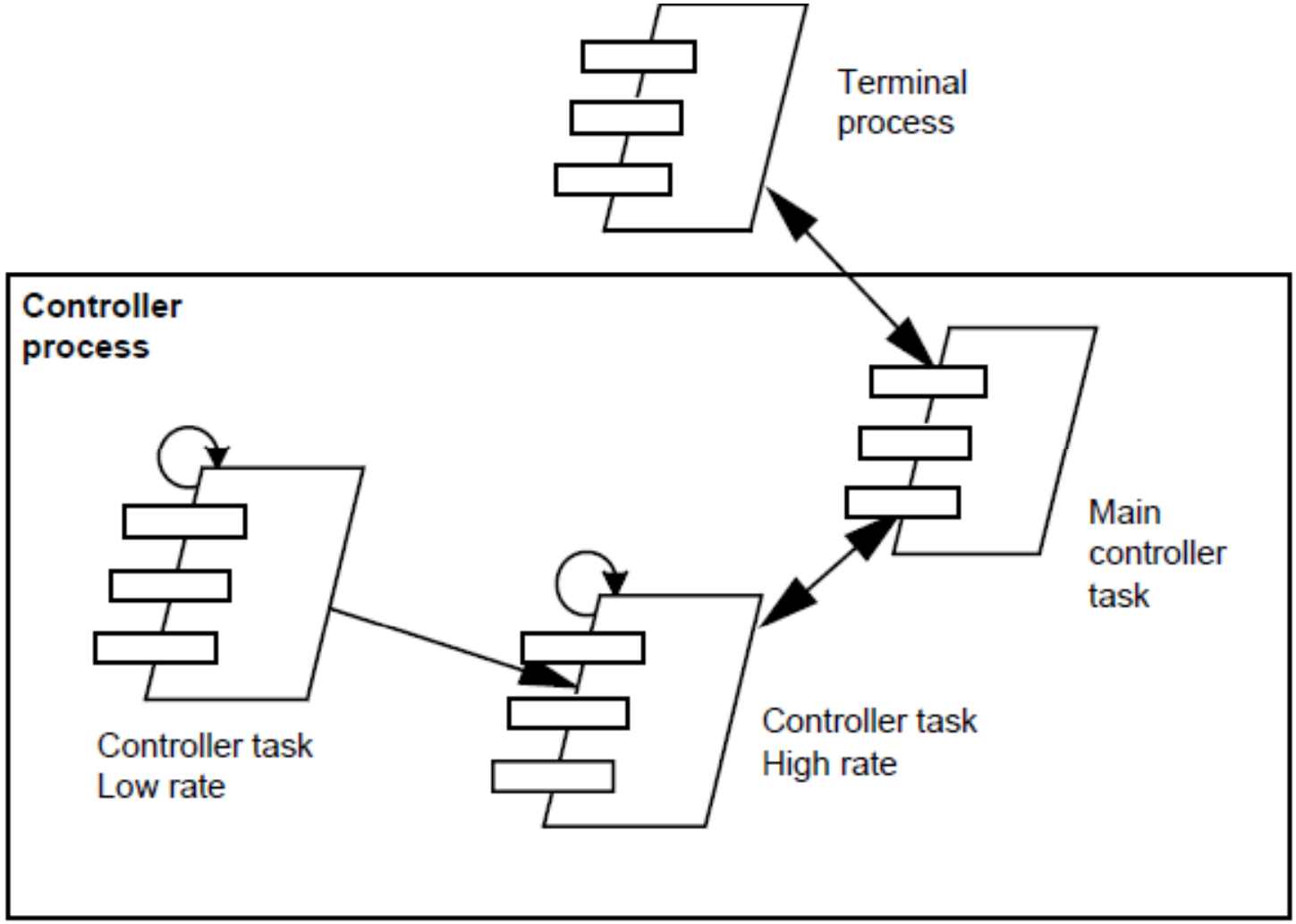
Process Viewpoint

- **Concerns**
  - Concurrency and synchronization
  - Considers non-functional requirements such as performance and availability
- **Stakeholder**
  - System Designer
  - Integrator
- **Components**
  - Tasks (process)
  - Task = Separate thread of control that can be scheduled individually
  - Major tasks represent architectural elements
  - Process = Grouping of tasks that form executable units

Process Viewpoint – Notation



Process View – Example



Process blueprint for the Télic PABX (partial)

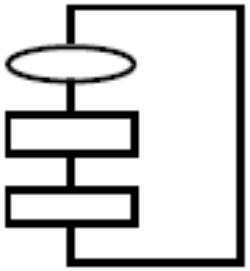
Development Viewpoint

- **Concern**
  - Organization of the actual software modules
- **Stakeholder**
  - Developer, (SW) manager
- **Components**
  - Modules, Subsystems

- Packaging into subsystems or layers
- Work breakdown structures

Development Viewpoint – Notation

Components



Module

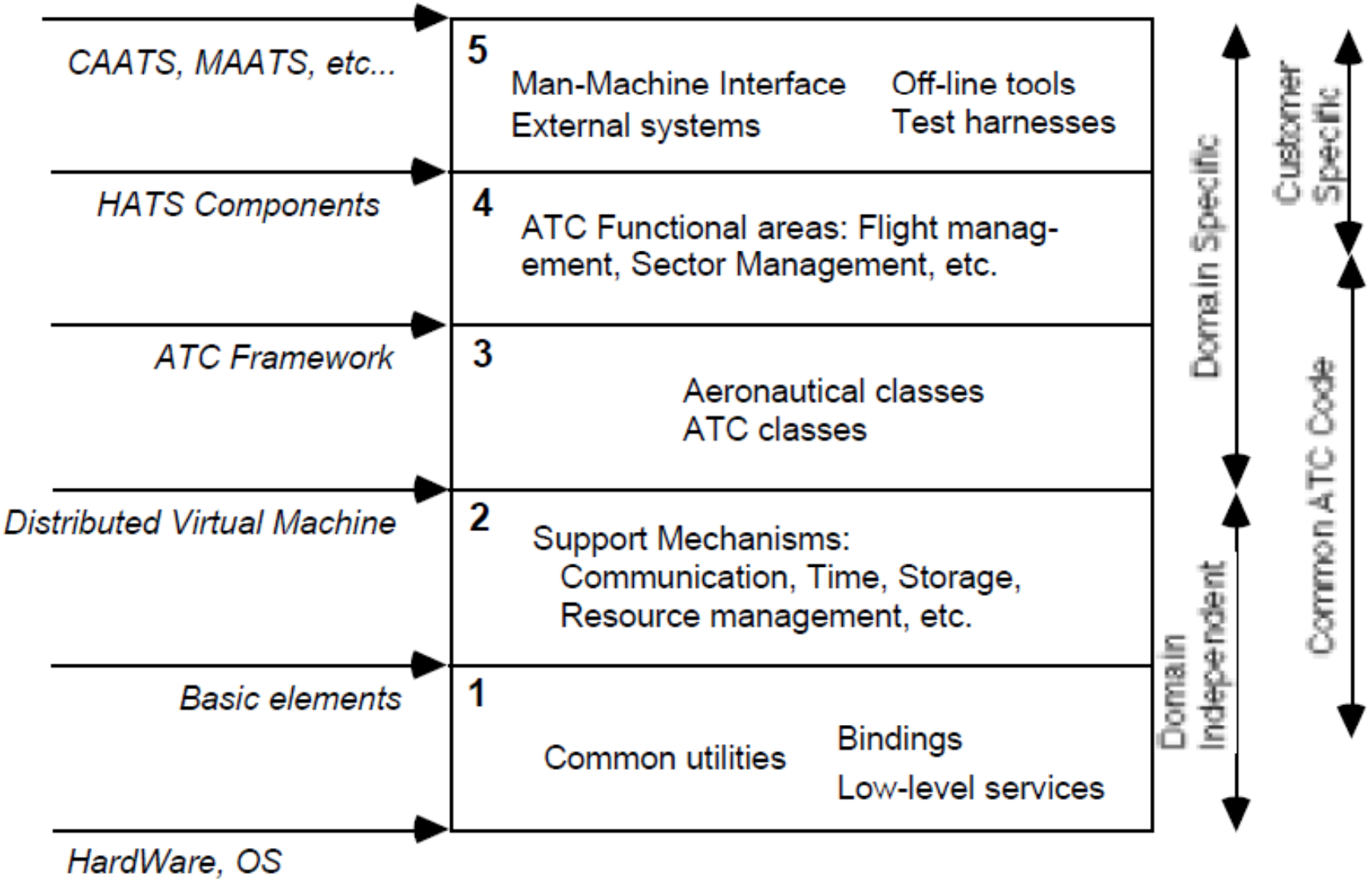
Subsystem

Layer

Connectors

Reference
Compilation dependency
(include, "with")

Development View – Example

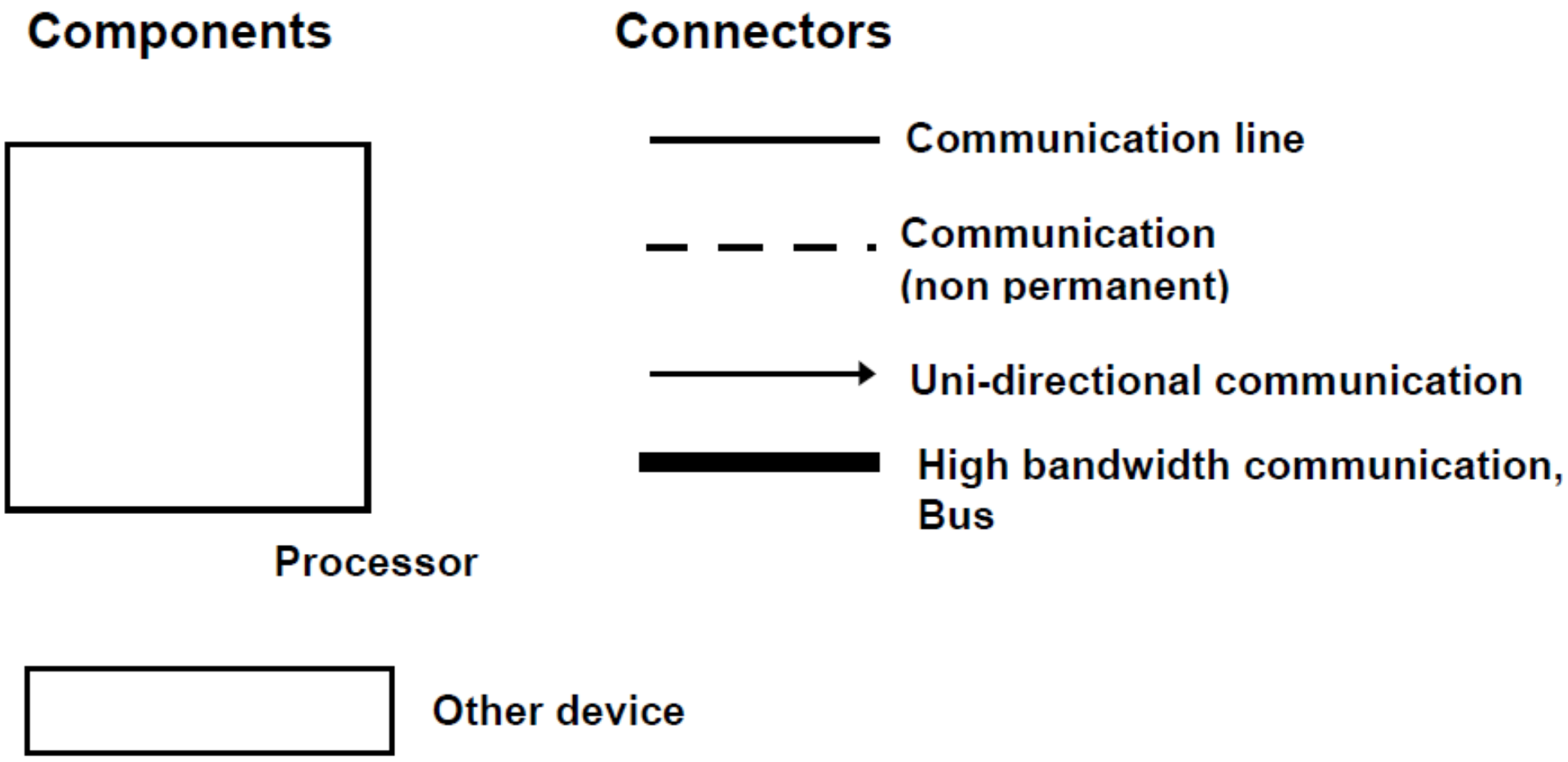


The 5 layers of Hughes Air Traffic Systems (HATS)

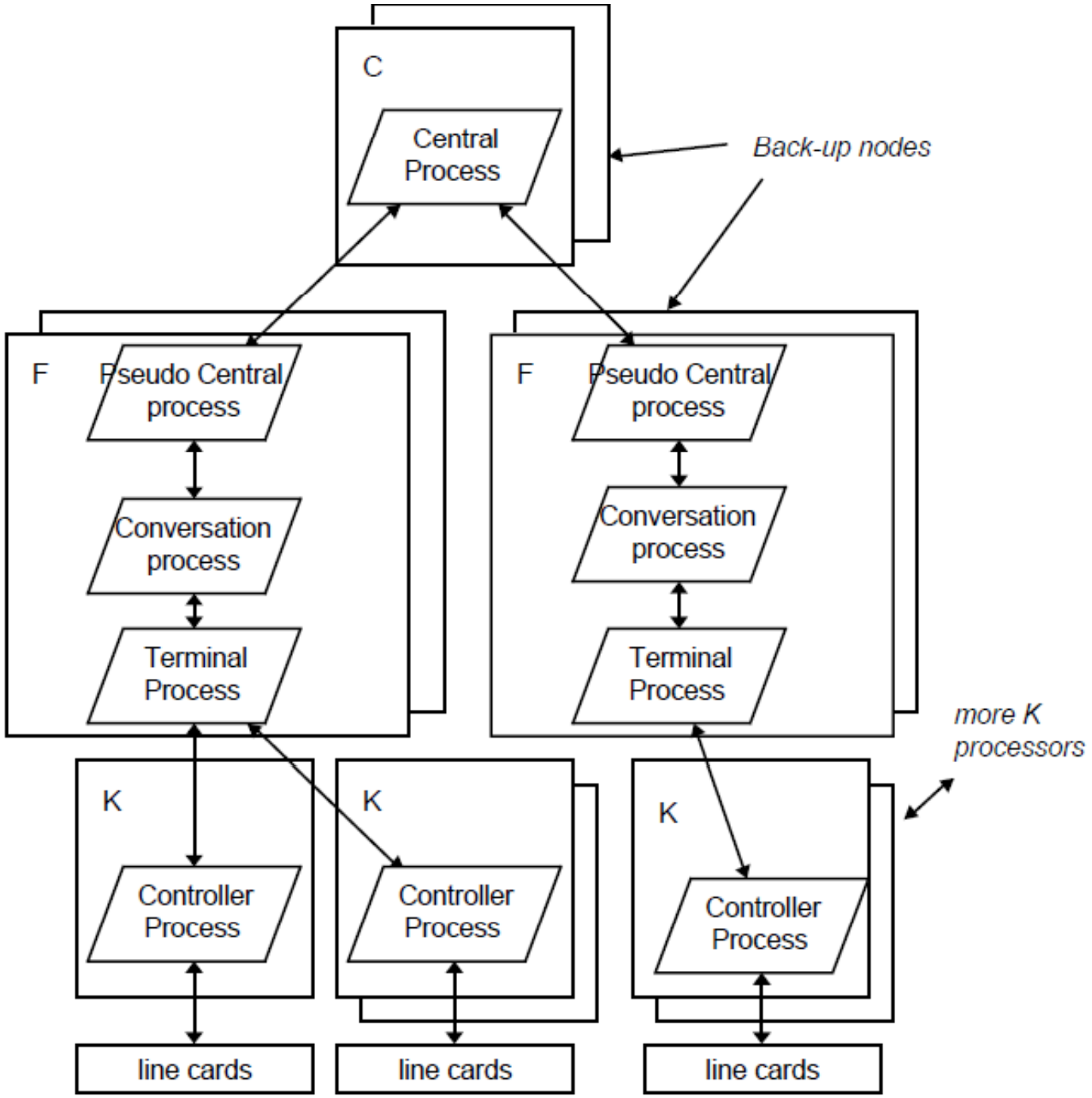
Physical Viewpoint

- Concerns
  - Scalability, performance, availability
- Stakeholders
  - System designer
- Components
  - Processing nodes

Physical Viewpoint – Notation



Physical View – Example



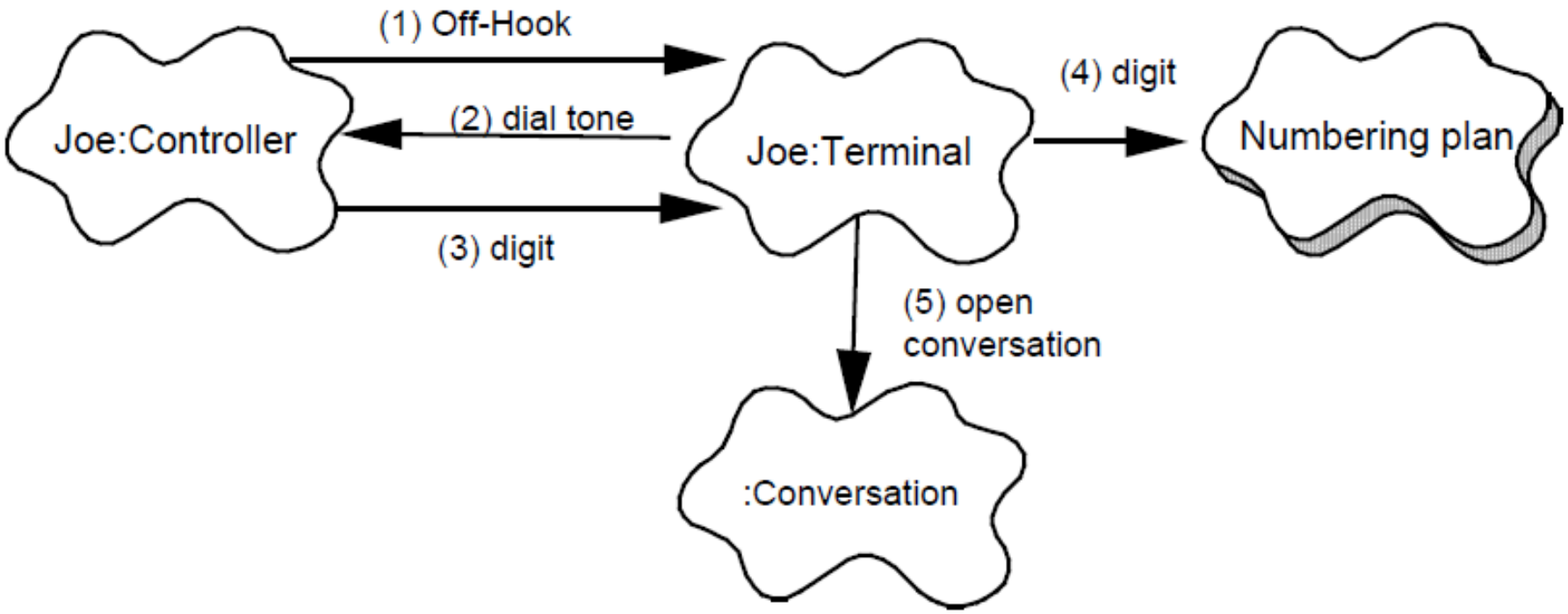
Physical blueprint for a larger PABX showing process allocation

Scenario Viewpoint

- Concerns
  - Understandability
- Stakeholders
  - End-user
- Components
  - Step, Scripts

- Link the various views via usage scenarios
- Driver for developing and evaluating architecture

Scenario View



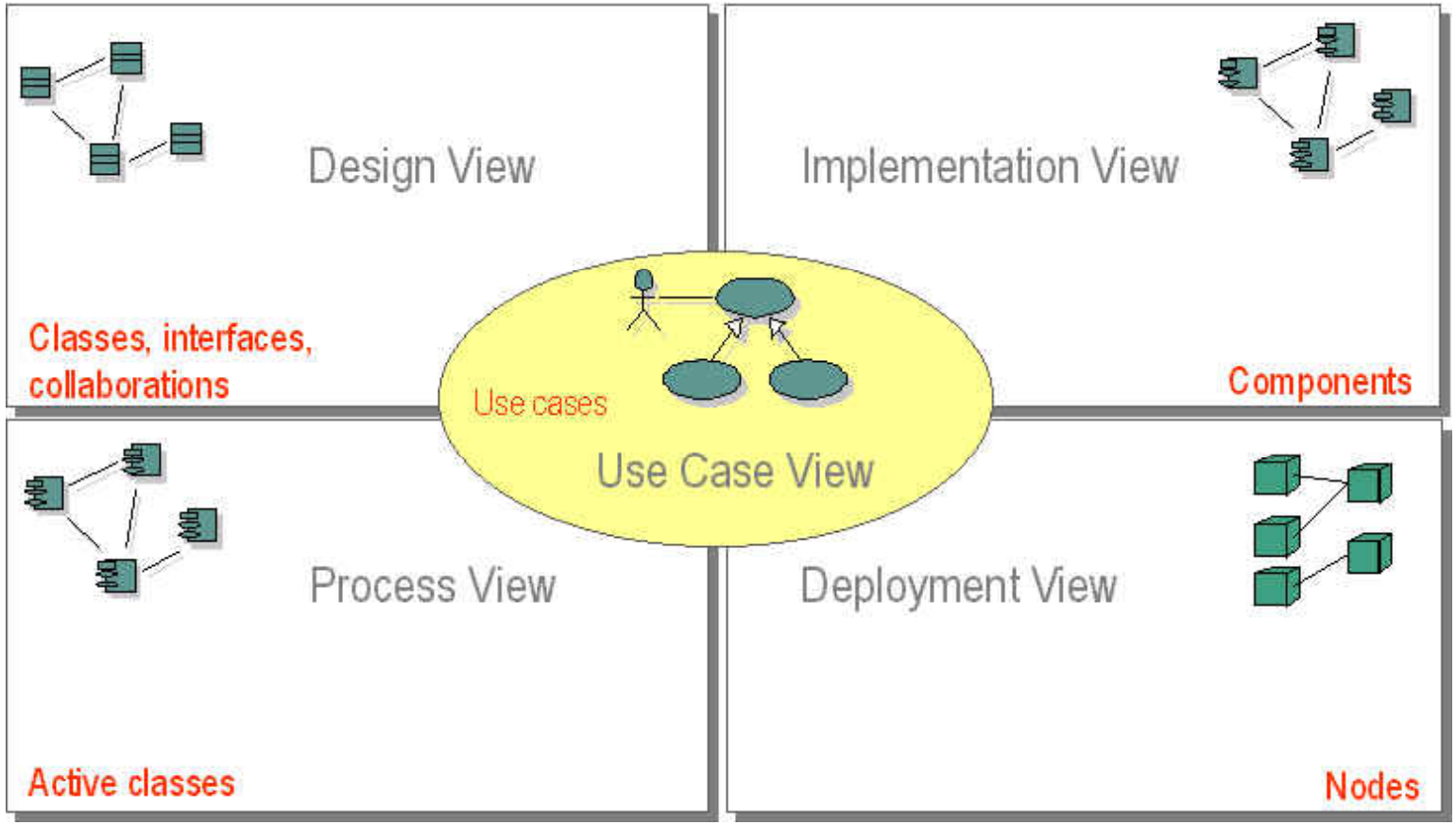
Embryo of a scenario for a local call—selection phase

Summary of "4+1" view model

View	*Logical*	*Process*	*Development*	*Physical*	*Scenarios*
Components	Class	Task	Module, Subsystem	Node	Step, Scripts
Connectors	association, inheritance, containment	Rendez-vous, Message, broadcast, RPC, etc.	compilation dependency, "with" clause, "include"	Communica-tion medium, LAN, WAN, bus, etc.	
Containers	Class category	Process	Subsystem (library)	Physical subsystem	Web
Stakeholders	End-user	System designer, integrator	Developer, manager	System designer	End-user, developer
Concerns	Functionality	Performance, availability, S/W fault-tolerance, integrity	Organization, reuse, portability, line-of-product	Scalability, performance,av ailability	Understand-ability
Tool support	Rose	UNAS/SALE DADS	Apex, SoDA	UNAS, Openview DADS	Rose

Summary of the "4+1" view model

Different Views in UML



Organization

Package, subsystem

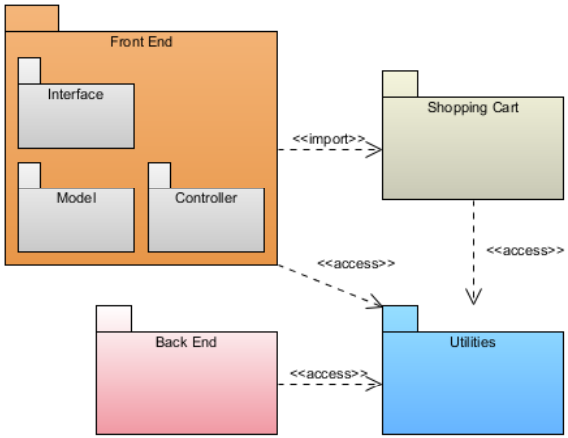
Dynamics

Interaction, state machine

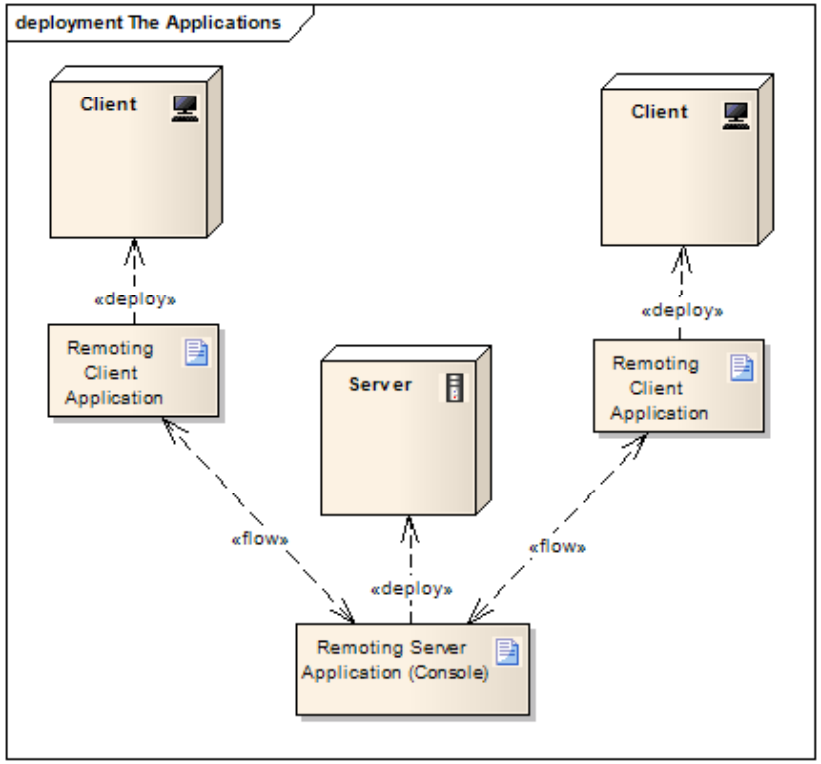
UML Views vs. Kruchten's Views

UML	Kruchten's 4+1 View
Design View	Logical View
Process View	Process View
Implementation View	Development View
Deployment View	Physical View
Use-Case View	Scenario View

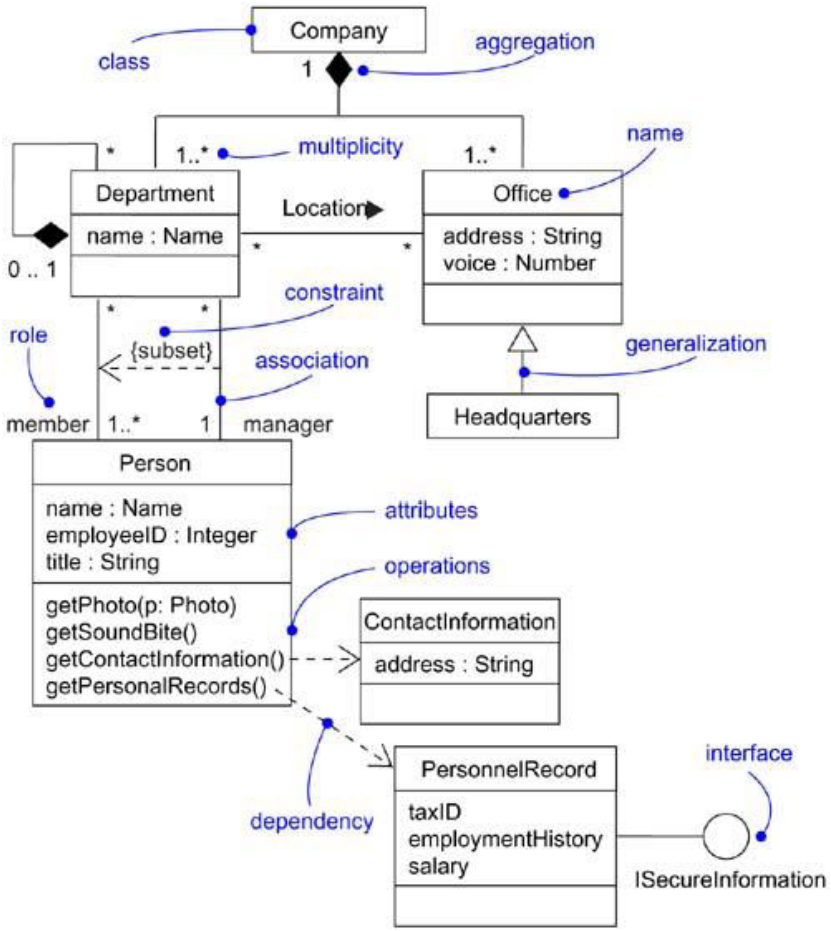
Examples of UML Views



Development View



Deployment View



Design View

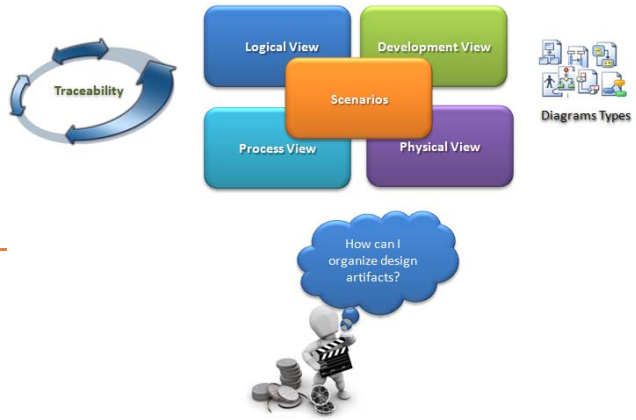
Views for Systems

- Number and type of views may differ per system
  - Single Processor: Drop deployment view
  - Single Process: Drop process view
  - Very small system: Drop implementation view
- Which views are relevant for the document?
  - Depends on who the stakeholders are ;)
  - And how they will use the documentation (concerns)

Summary (1/2)

- An architecture is addressed to one-or-more **stakeholders** to address their concerns
- An architecture is organized into one-or-more **views** of the system
- Each view addresses one-or-more **concerns** of the stakeholders
- Each view conforms to a particular **viewpoint**

Summary (2/2)



- **Logical**
  - Focus: Functionality of the system.
  - Contents: Class diagrams, Sequence diagrams, *Communication* diagrams, Layer diagrams.
- **Development** (Implementation)
  - Focus: Developer's perspective.
  - Contents: Component diagram, *Package* diagrams.
- **Process**
  - Focus: Runtime behaviour of the system, such as the system processes and communication, concurrency, performance and scalability.
  - Contents: Activity diagrams
- **Physical** (Deployment)
  - Focus: System Engineer's perspective, looking at the system topology, deployment and communication.
  - Contents: *Deployment* diagrams
- **Scenarios**
  - Focus: Use cases which not only describe the system scenarios and behaviour, but which play an important role in the illustration and validation of the overall architecture.
  - Contents: Use case diagrams