Session 09
Documenting Architecture Views

BITS Pilani
Pilani|Dubai|Goa|Hyderabad

Harvinder S Jabbal
SSZG653 Software Architectures

innovate achieve lead

BITS Pilani

Pilani|Dubai|Goa|Hyderabad

Architecture Views

Views

- Representation of a coherent set of architectural elements , as written by and read by system stakeholders.

Documenting

- “Documenting an architecture is a matter of documenting the relevant **views** and then adding a documentation that applies to more than one view.”

Solution to a Problem

- Problem
  - Architecture documents do not address the concerns of all stakeholders .
  - Deferent Stakeholders : end-user, system engineers, developers and project managers.
  - Architecture documents contained complex diagrams some times they are hard to be represented on the documentation.
- Solution
  - Using different notations for several **Views** each one addressing one specific set for concerns.

4+1 Model -
Philippe Kruchten, Rational Software Corp.

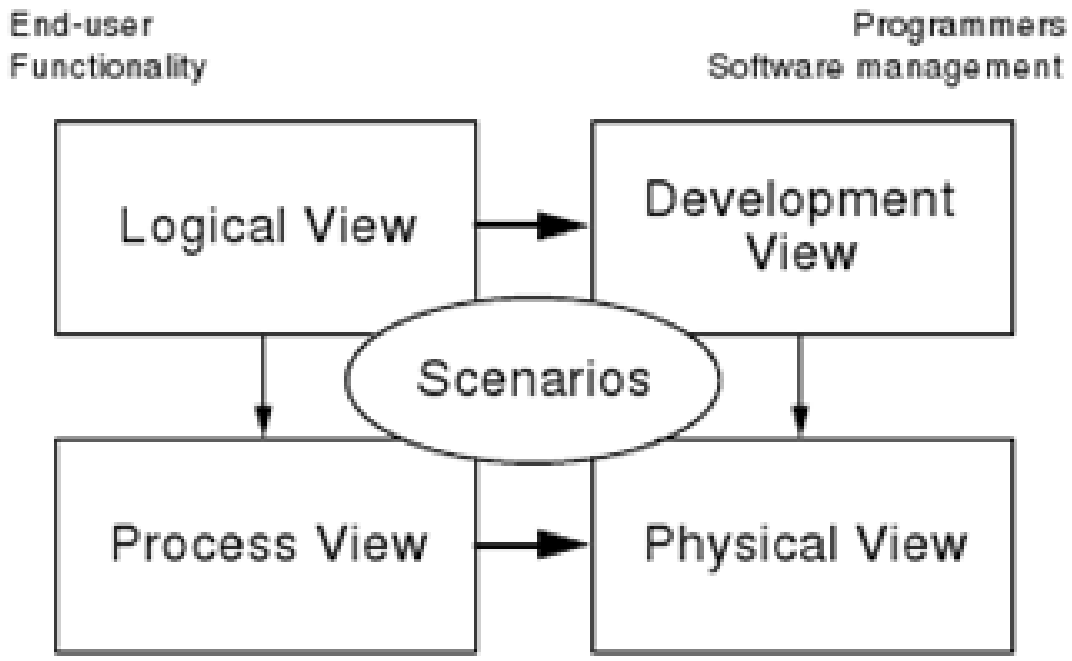


Figure 1 — The "4+1" view model

Logical View

- **The logical view**, which is the object model of the design (when an object-oriented design method is used)

Viewer: End-user

considers: Functional requirements- What are the services must be provided by the system to the users.

Notation: The Booch notation .

Tool: Rational Rose

Notation for Logical View-

Philippe Kruchten, Rational Software Corp.

Notation for the logical view

The notation for the logical view is derived from the Booch notation[4]. It is considerably simplified to take into account only the items that are architecturally significant. In particular, the numerous adornments are not very useful at this level of design. We use Rational Rose® to support the logical architecture design.

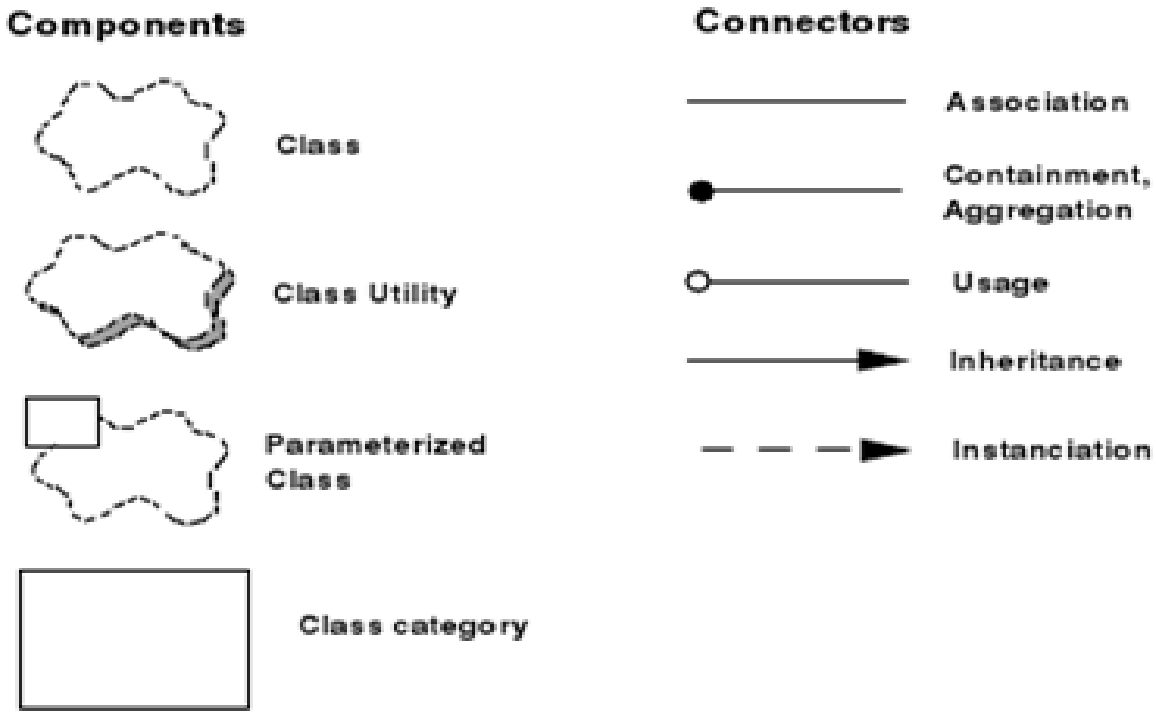
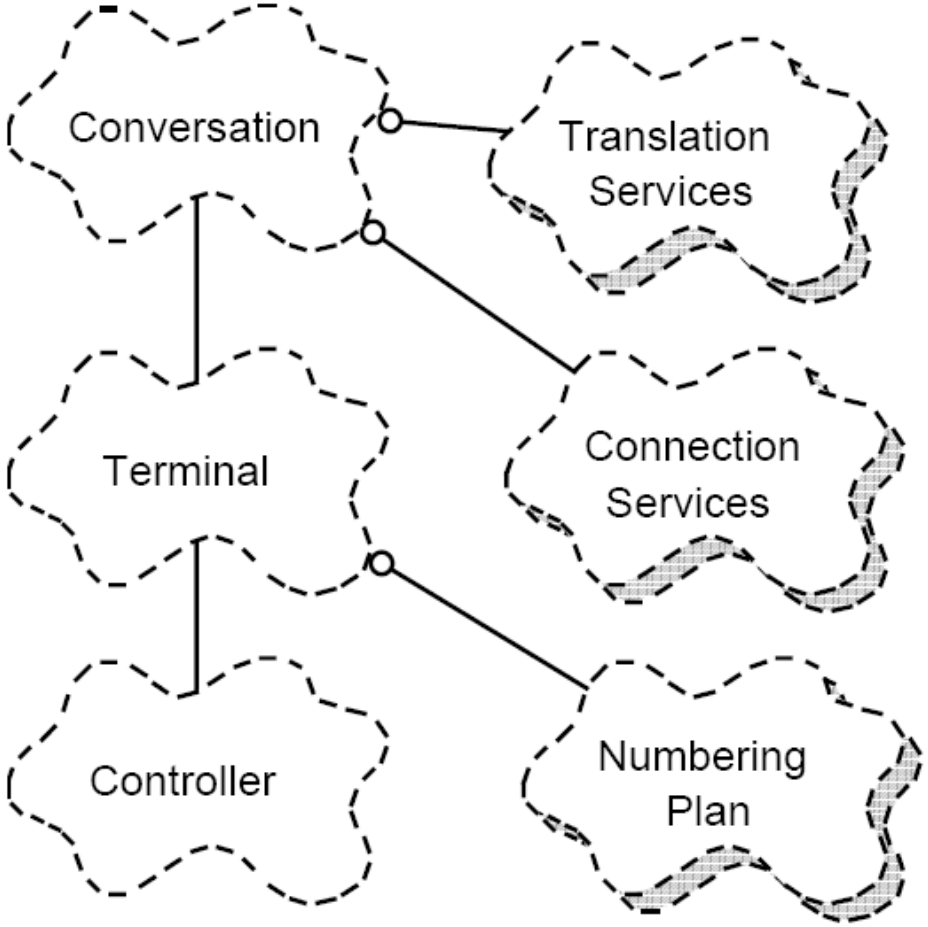


Figure 2 — Notation for the logical blueprint

Logical view Example PABX-

Philippe Kruchten, Rational Software Corp.



Process View

The process view, which captures the concurrency and synchronization aspects of the design**(The process decomposition).**

viewer: Integrators

considers: Non - functional requirements (scalability, concurrency, and performance)

style: Garlan and Shaw ‘s Architecture styles.

Process (cont.)

Uses multiple levels of abstractions.

A process is a grouping of tasks that form an executable unit:

- Major Tasks: Architecture relevant tasks.
- Minor or helper Tasks: (Buffering)

Notation-

Philippe Kruchten, Rational Software Corp.

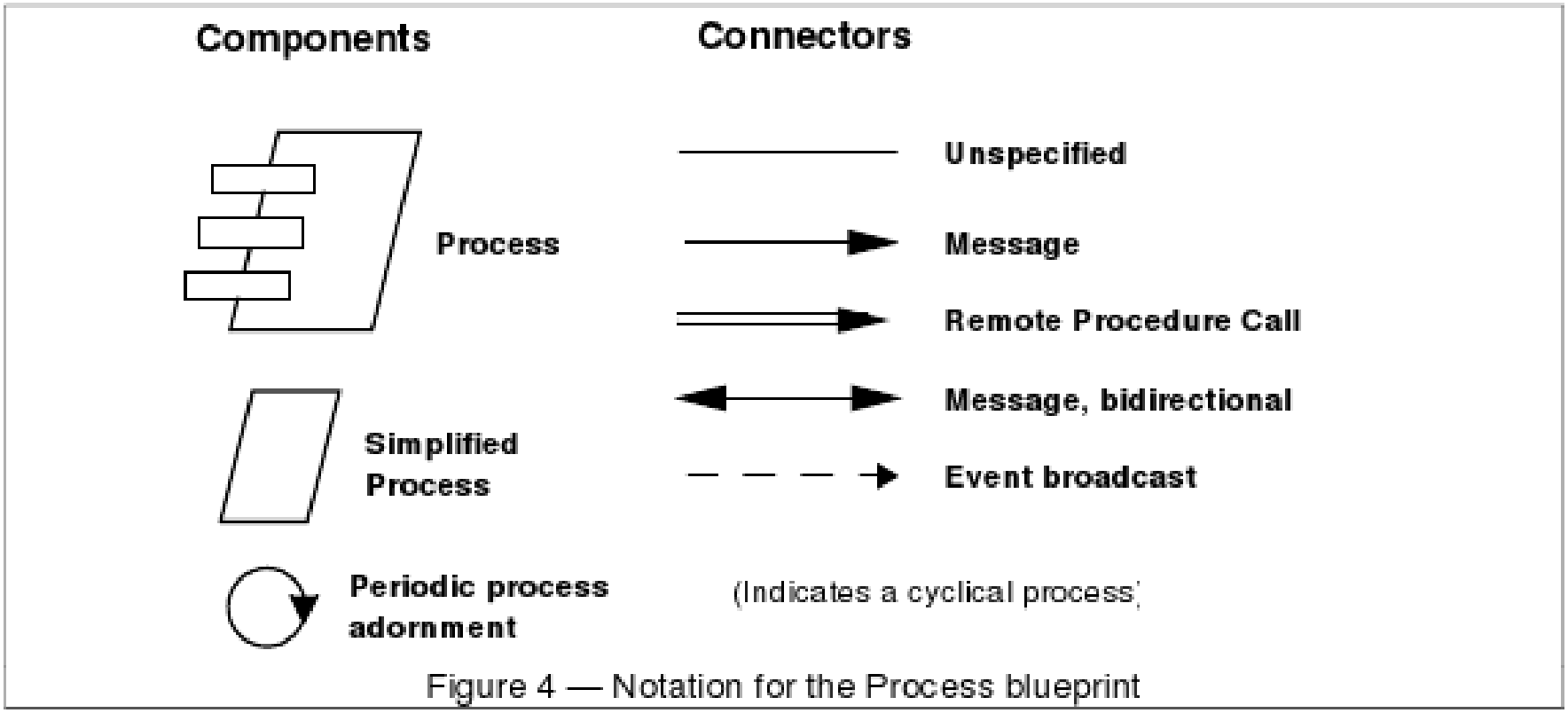
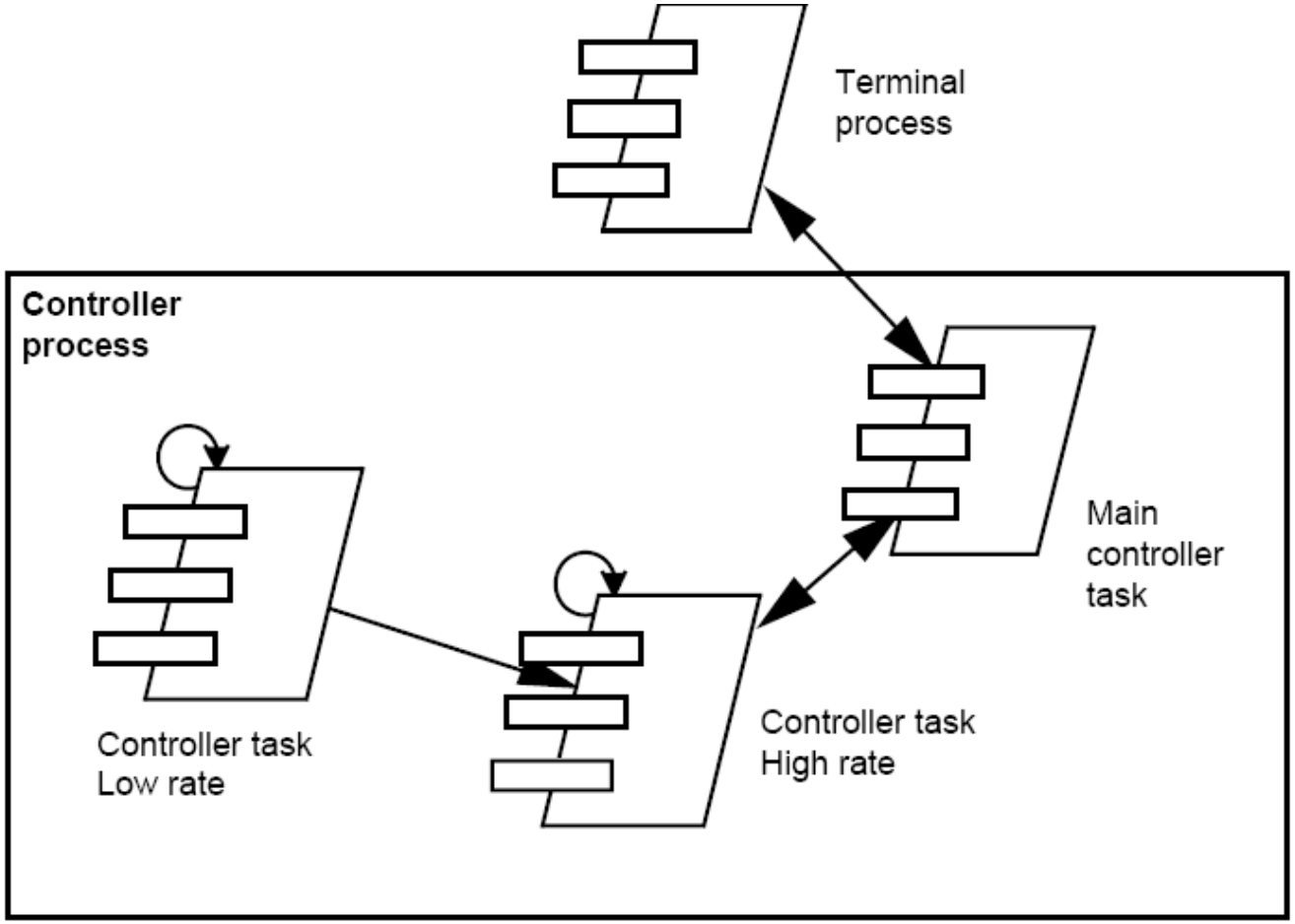


Figure 4 — Notation for the Process blueprint

Process View example PABX (partial)-

Philippe Kruchten, Rational Software Corp.



Development View

The *development* view, which describes the **static** organization of the software in its development environment.

Viewer: Programmers and Software Managers

considers: software module organization. (Hierarchy of layers, software management, reuse, constraints of tools).

Notation: the Booch notation.

Style: layered style

Notation for Development blueprint-

Philippe Kruchten, Rational Software Corp.

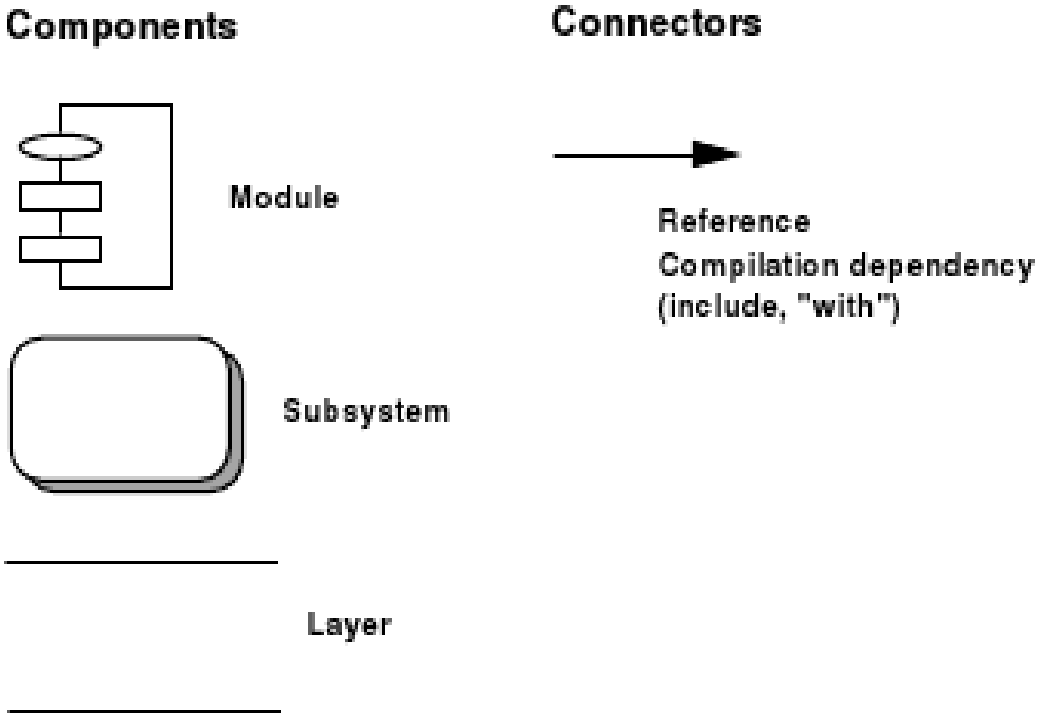


Figure 5 — Notation for the Development blueprint

Development View – Layered Style

Philippe Kruchten, Rational Software Corp.

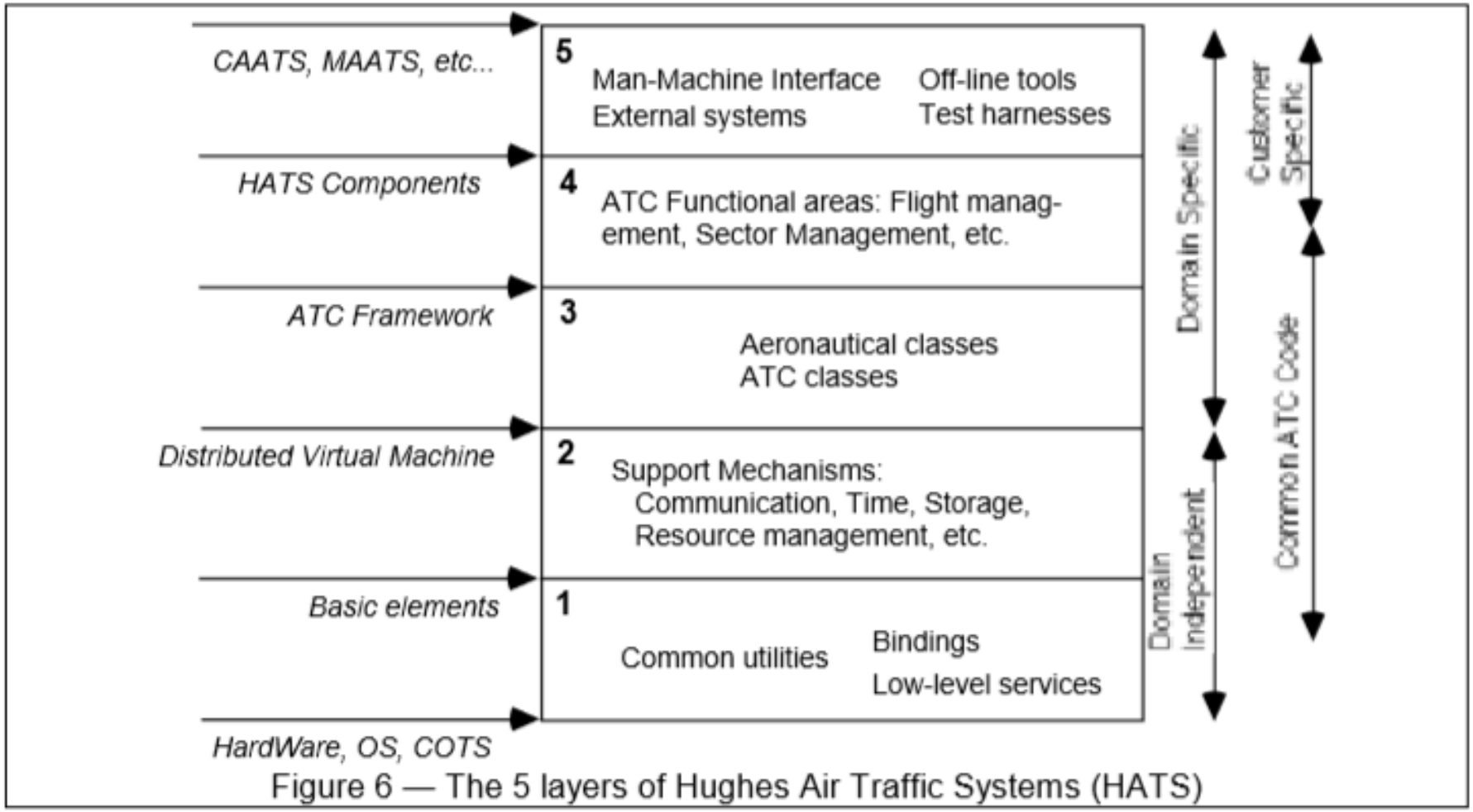


Figure 6 — The 5 layers of Hughes Air Traffic Systems (HATS)

Physical View

the physical view, which describes the mapping(s) of the software onto the hardware and reflects its distributed aspect.

Viewer: System Engineers

Considers: Non-functional requirement (reliability, availability and performance). regarding to underlying hardware.

There may be two architecture:

- Test and development
- deployment

Notation for Physical view-

Philippe Kruchten, Rational Software Corp.

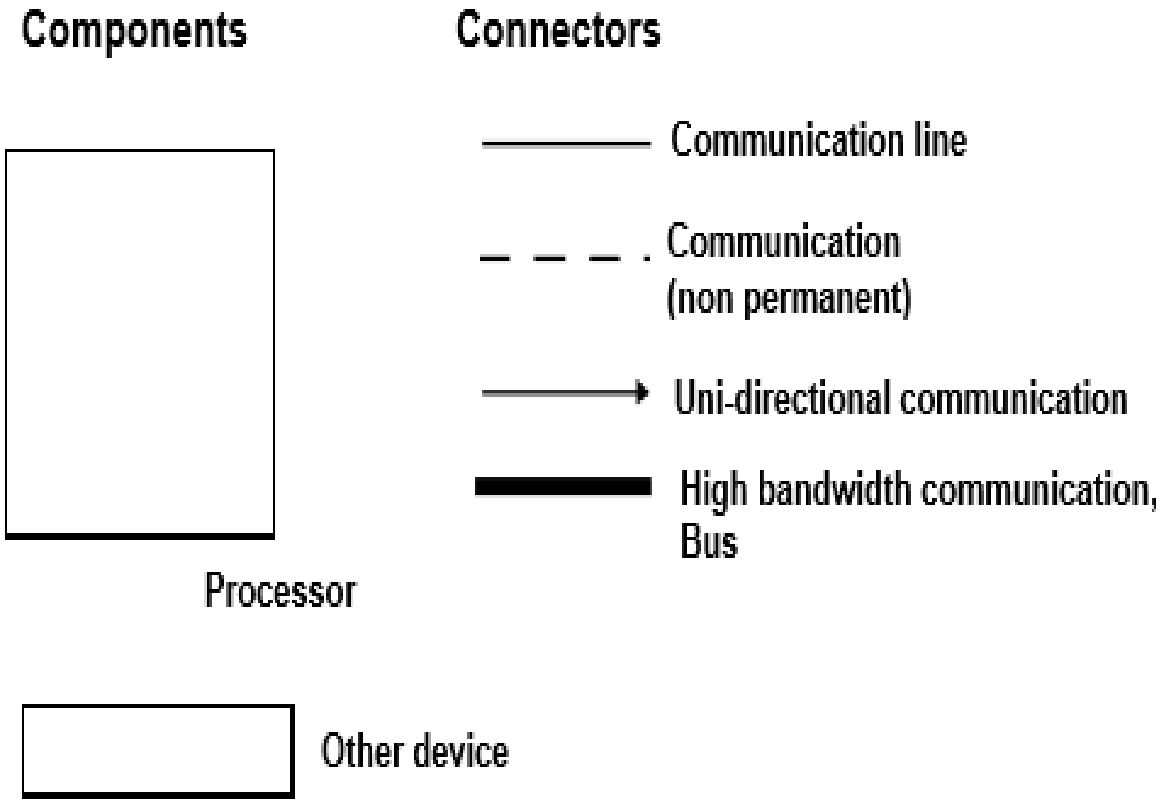


Figure 7 — Notation for the Physical blueprint

Physical blueprint PABX-

Philippe Kruchten, Rational Software Corp.

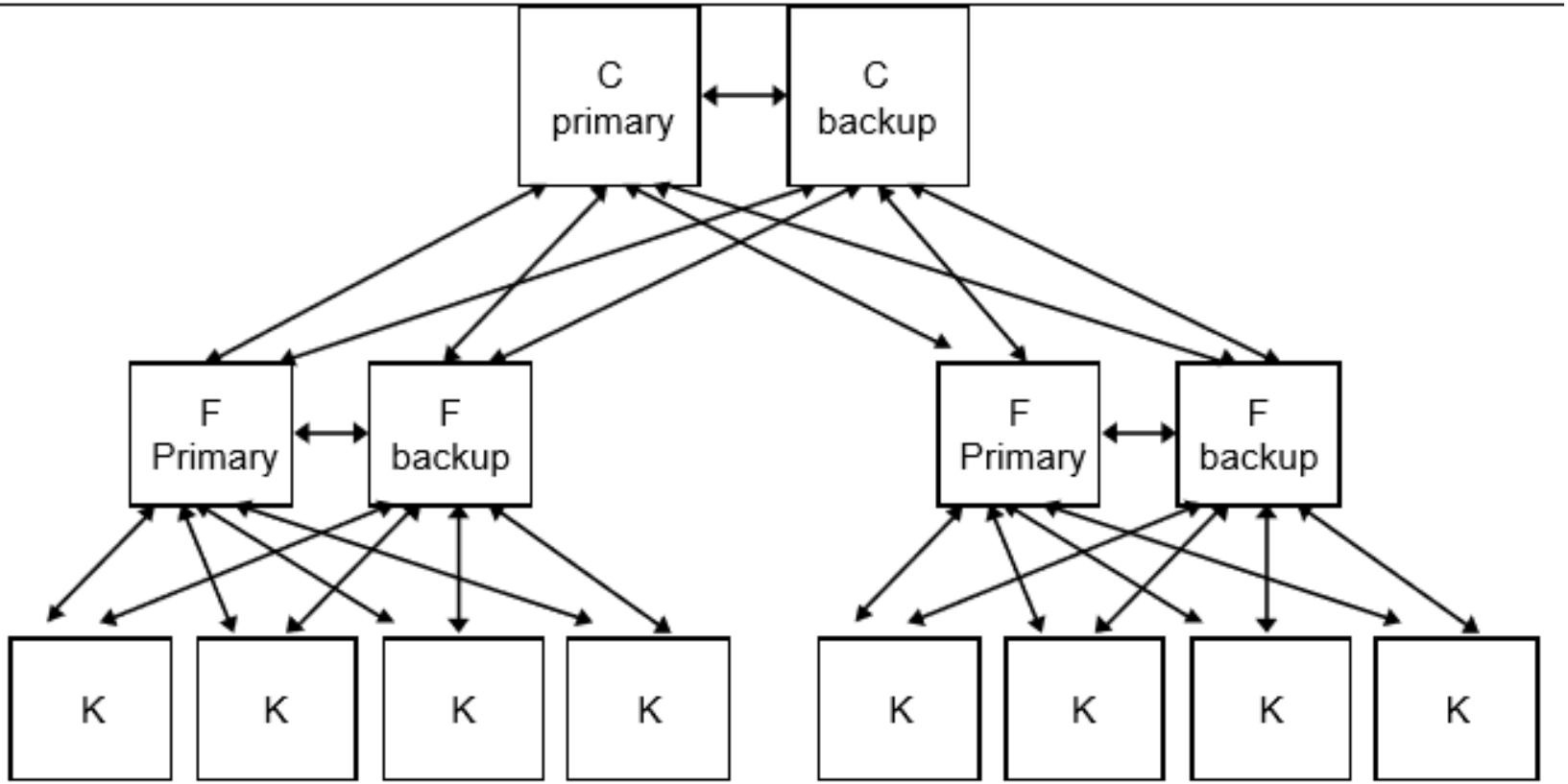
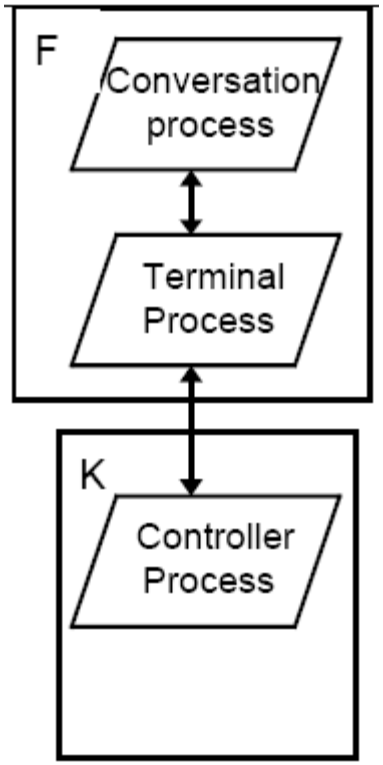


Figure 8 — Physical blueprint for the PABX

Physical view example-

Philippe Kruchten, Rational Software Corp.



A small PABX physical architecture with process allocation.

Physical view example-

Philippe Kruchten, Rational Software Corp.

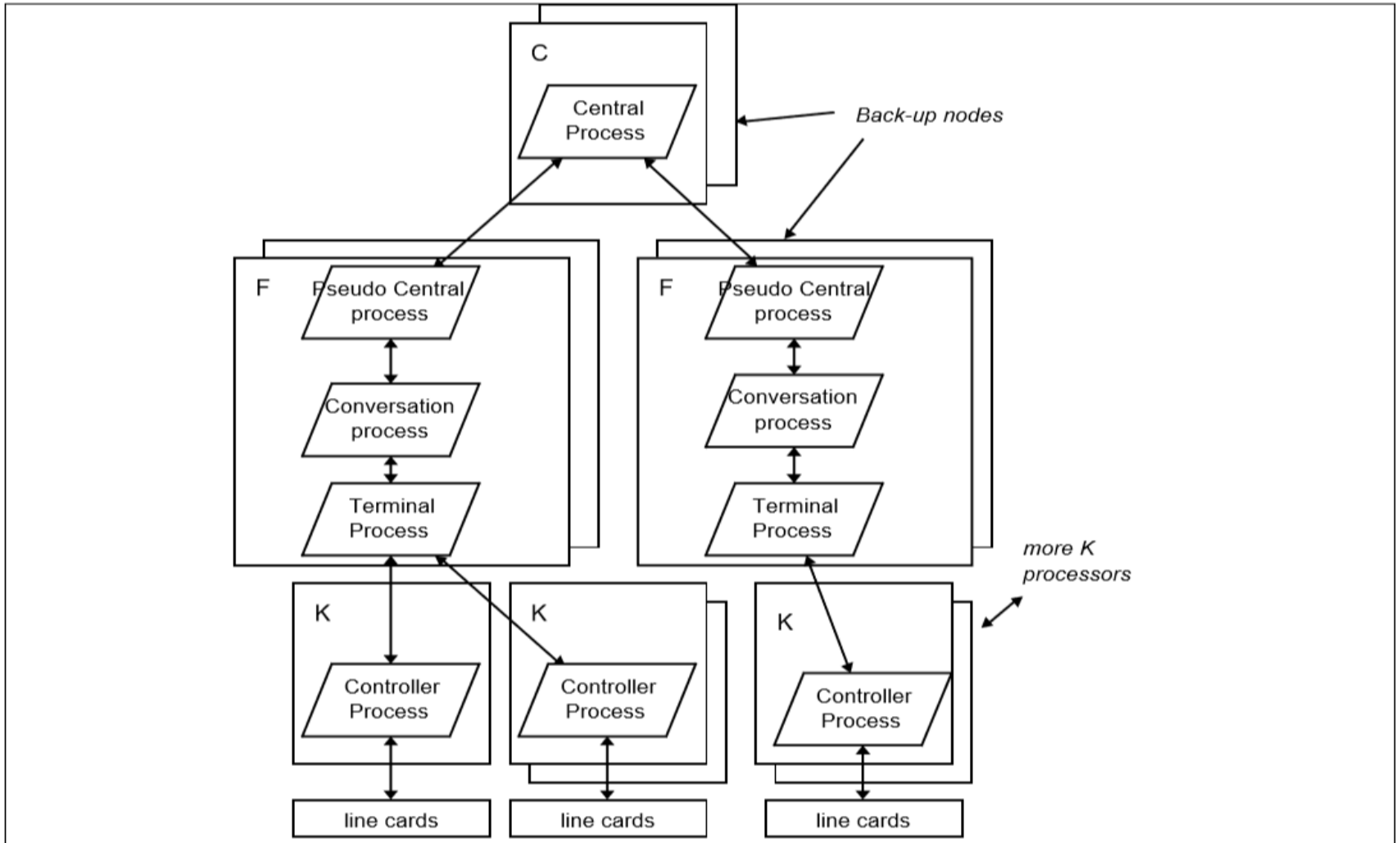


Figure 10 — Physical blueprint for a larger PABX showing process allocation

Scenarios

(Putting all “4 views” together)

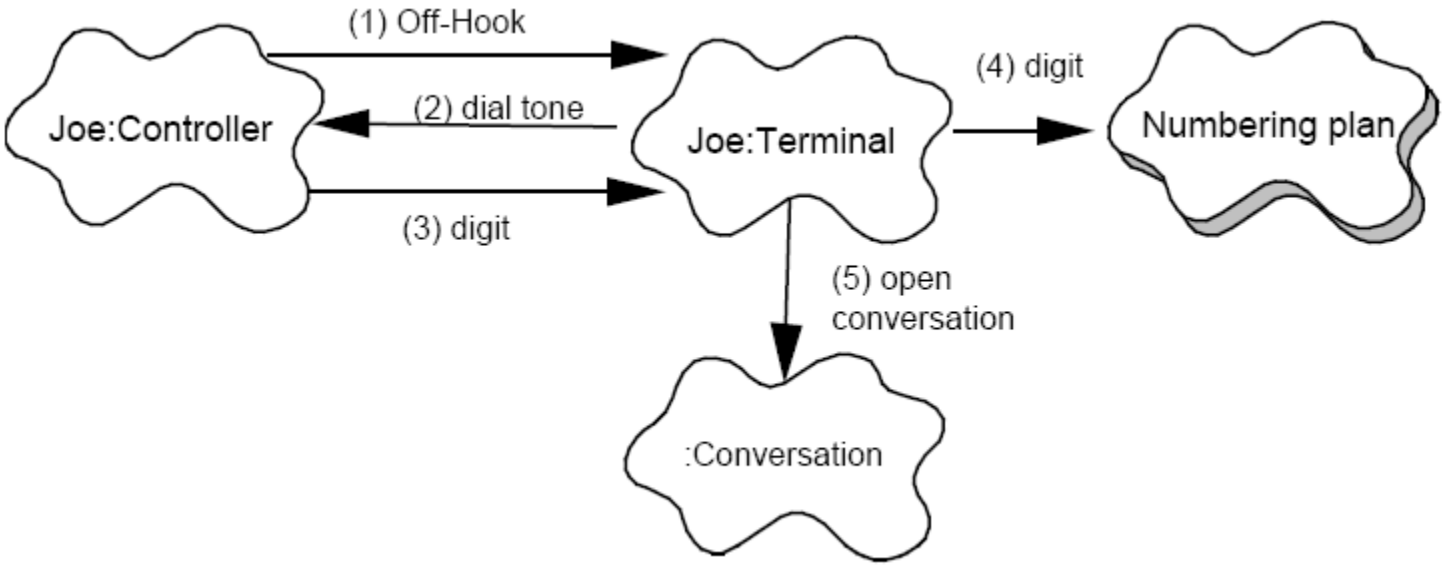
Viewer: All users and Evaluators.

Considers: System consistency and validity

Notation: Similar to logical view

Scenario example-

Philippe Kruchten, Rational Software Corp.



Scenario for a Local call – selection phase

Correspondence between the views

The **views** are interconnected.

Start with Logical view and Move to Development / Process view and then finally go to Physical view.

From logical to Process view

Two strategies :

- Inside-out: starting from Logical structure
- Outside-in: starting from physical structure

From logical to Process view

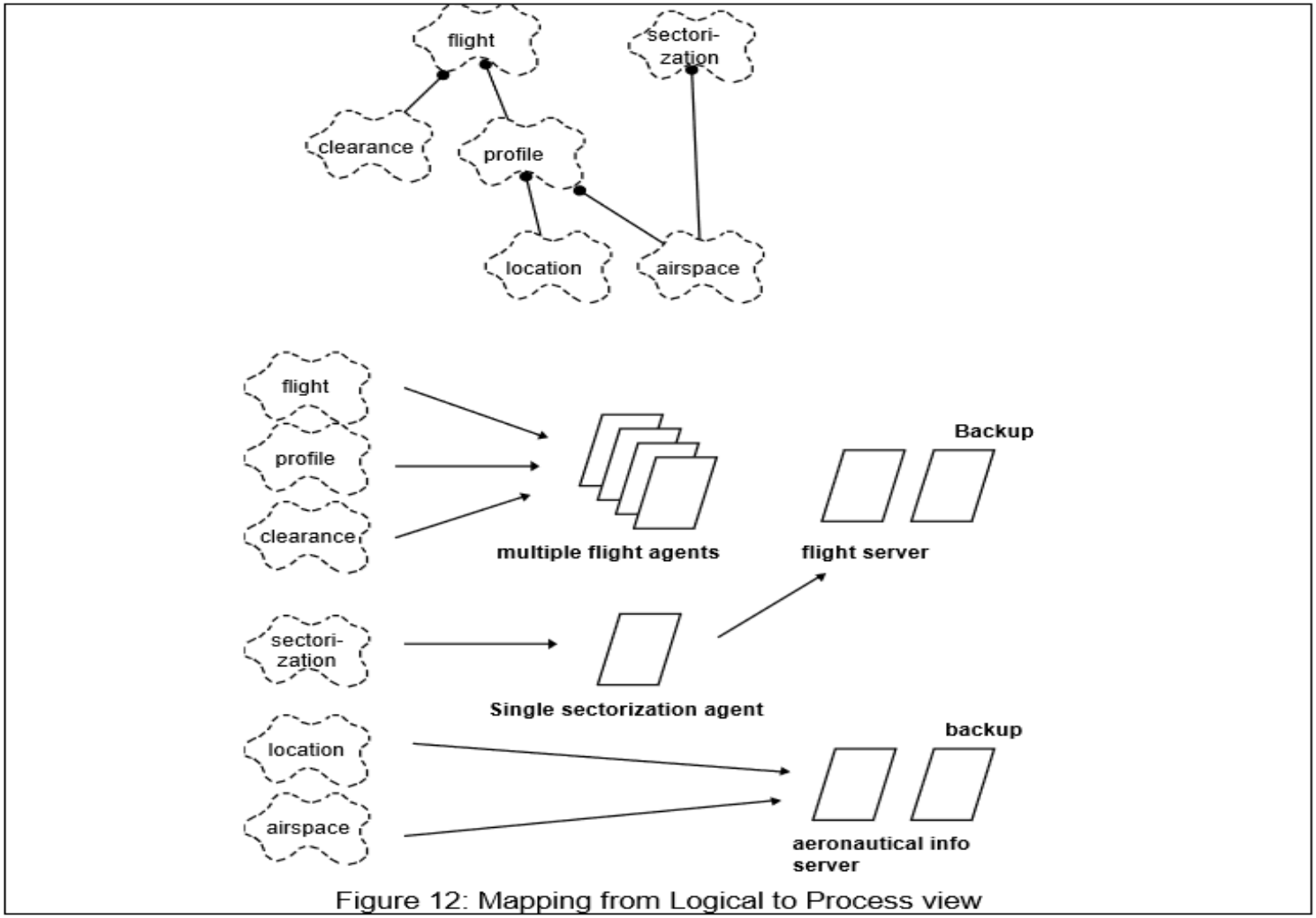


Figure 12: Mapping from Logical to Process view

From Logical to development

They are very close, but the larger the project, the greater the distance between these views.

Grouping to subsystems depending on:

- The team organization.
- The class categories which includes the packages.
- The Line of codes.

Iterative process

Not all architectures need all views.

A scenario-driven approach to develop the system is used to handle the iterative.

Documenting the architecture:

- **Software architecture document**: follows closely “4+1” views.
- **Software design guidelines:** it captured the most important design decisions that must be respected to **maintain** the architectural **integrity**.

Software Design Document

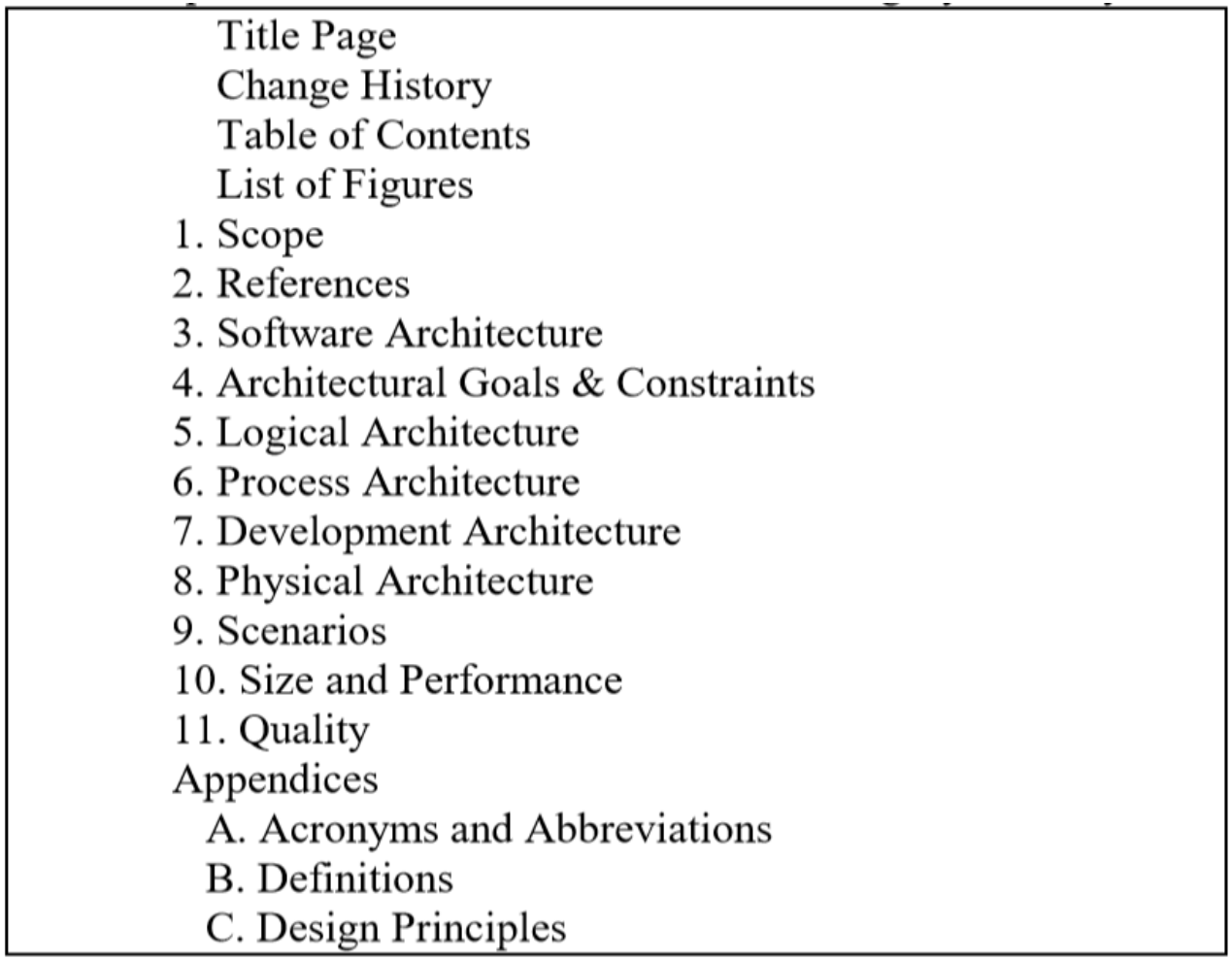


Figure 13 — Outline of a Software Architecture Document

Annotation:

- “4+1 views” methodology successfully used in the industry
  - Air Traffic Control
  - Telecom
- This paper missing the tools to integrate these views which lead to an inconsistency problem.
- The inconsistency problem is more tangible in the maintenance of the architecture.

Summary

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

Table 1 — Summary of the "4+1" view model