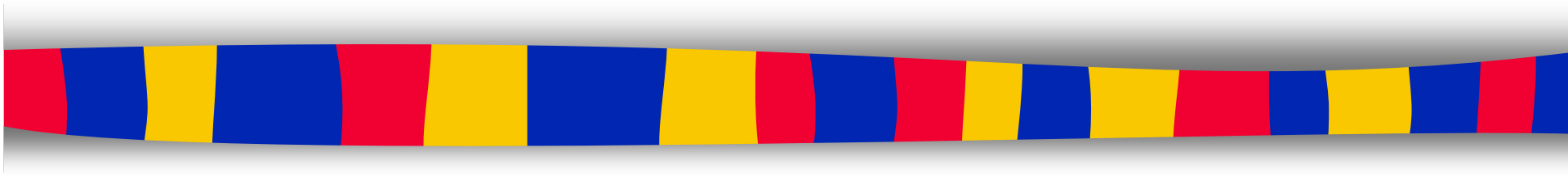
EECS 4314

Advanced Software Engineering



Topic 09:

4+1 Views

Zhen Ming (Jack) Jiang

Relevant Readings

- Philippe Kruchten. The 4+1 View Model of architecture. IEEE Software. 1995

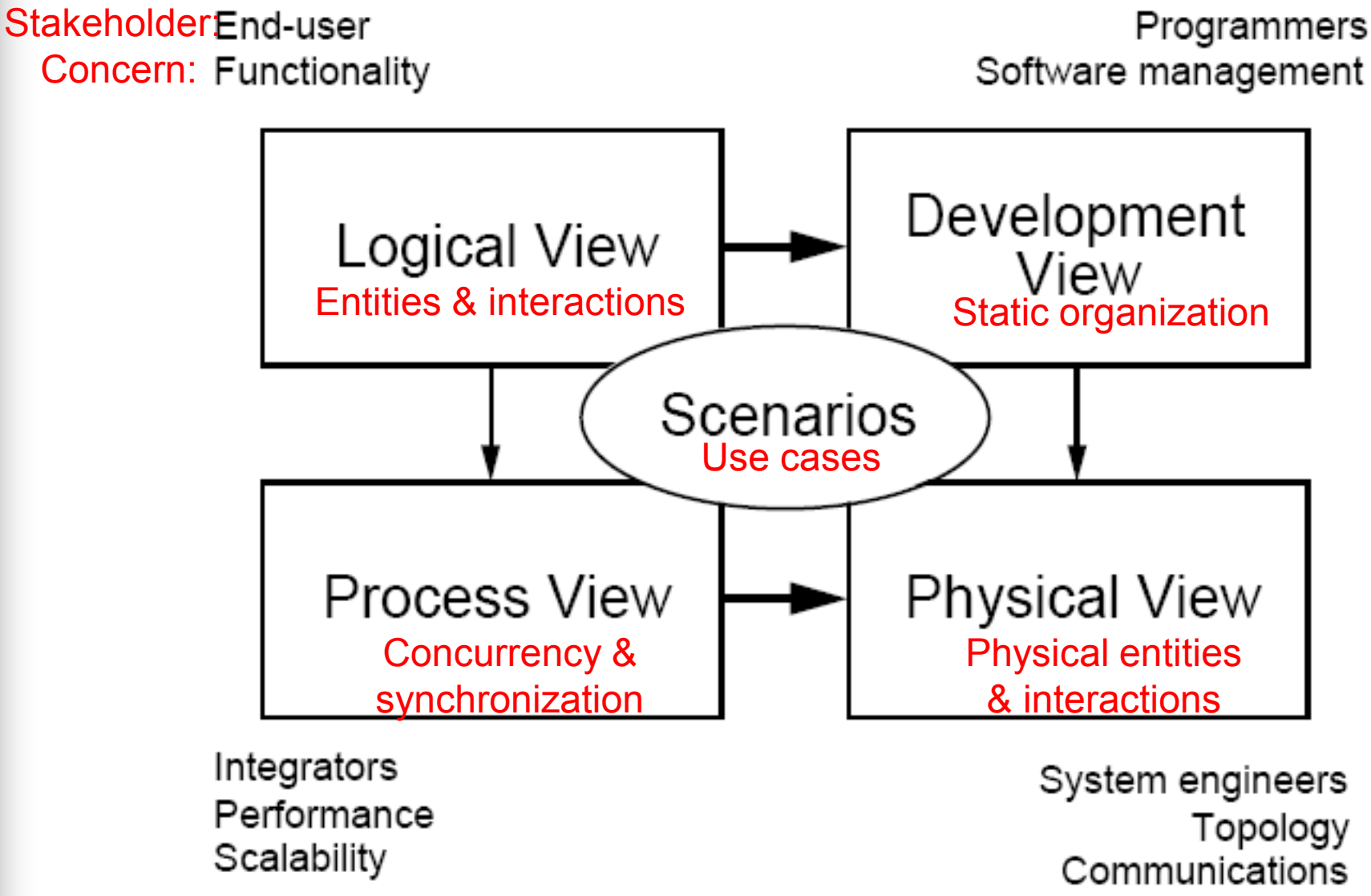
The Problem

- Ambiguities in Boxes-and-Arrows Diagrams
  - Boxes can be programs, chunks of source code, physical computers, logical groupings of functionalities, …
  - Arrows can be data flow, control flow, or both.
- Architecture documents over-emphasize one aspect of software development or
- Architecture documents do not address the concerns of all stakeholders
  - Many stakeholders (e.g., customers, developers, project managers, etc.), who care about different aspects of the system
    - Cannot provide one representation to satisfy all stakeholders
    - Stakeholders want to interact with parts that are most important to them

Architecture Views

- Various parts of the architecture have to be modeled using different approaches
- **View:** is a set of design decisions related to a common concern (or set of concerns)
- **Concern:** is an aspect of the system that a stakeholder cares about

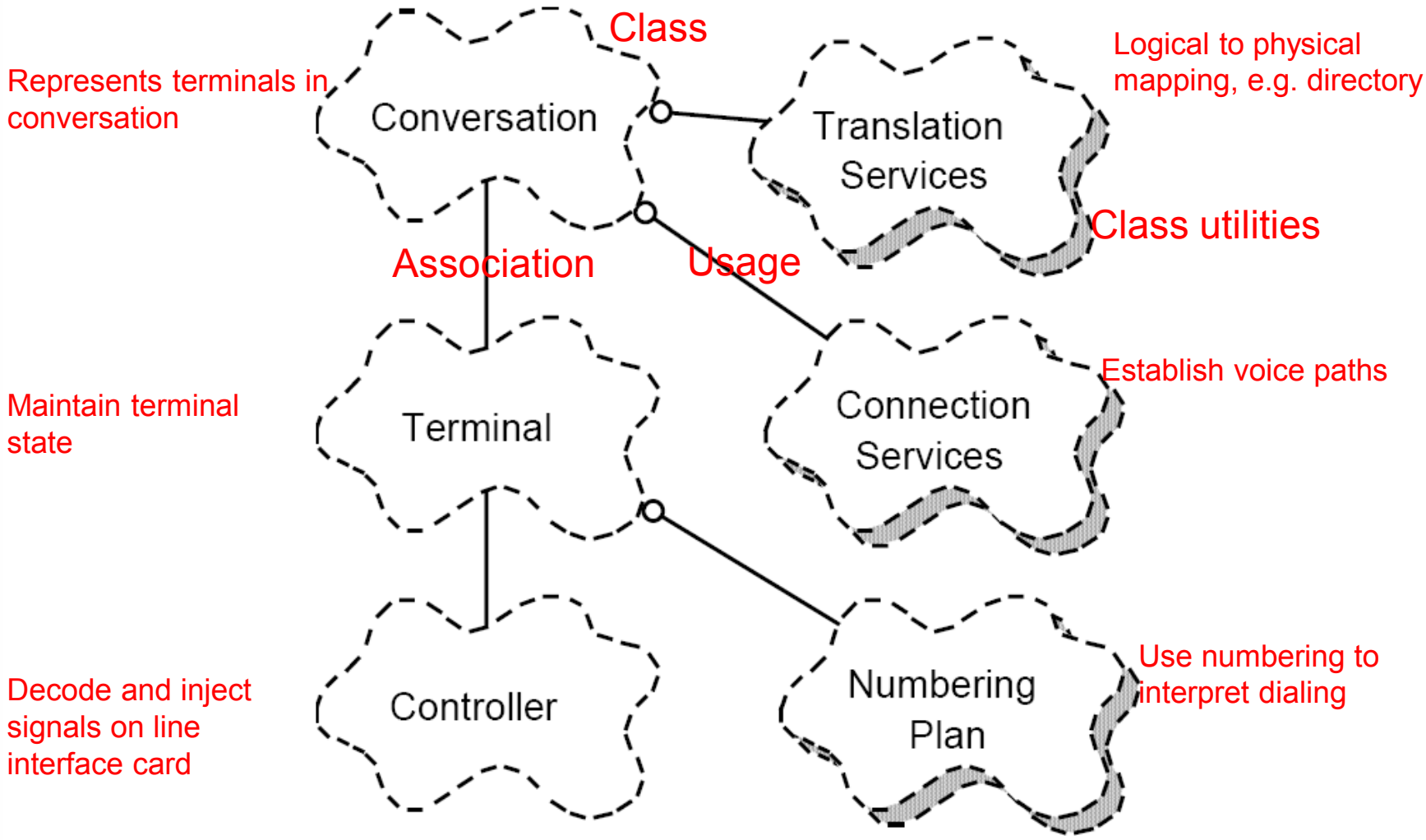
Architectural Views



The Logical View

- Captures the logical (often software) entities in a system and their interconnections
  - **Components**: Classes
  - **Connectors**: Associations, containment, inheritance
  - **Stakeholders**: End-users
  - **Concerns**: Functionality (i.e., functional requirements)
- Why do we need logical views?
  - It provides decompositions used for
    - functional analysis and,
    - to identify common elements in the system

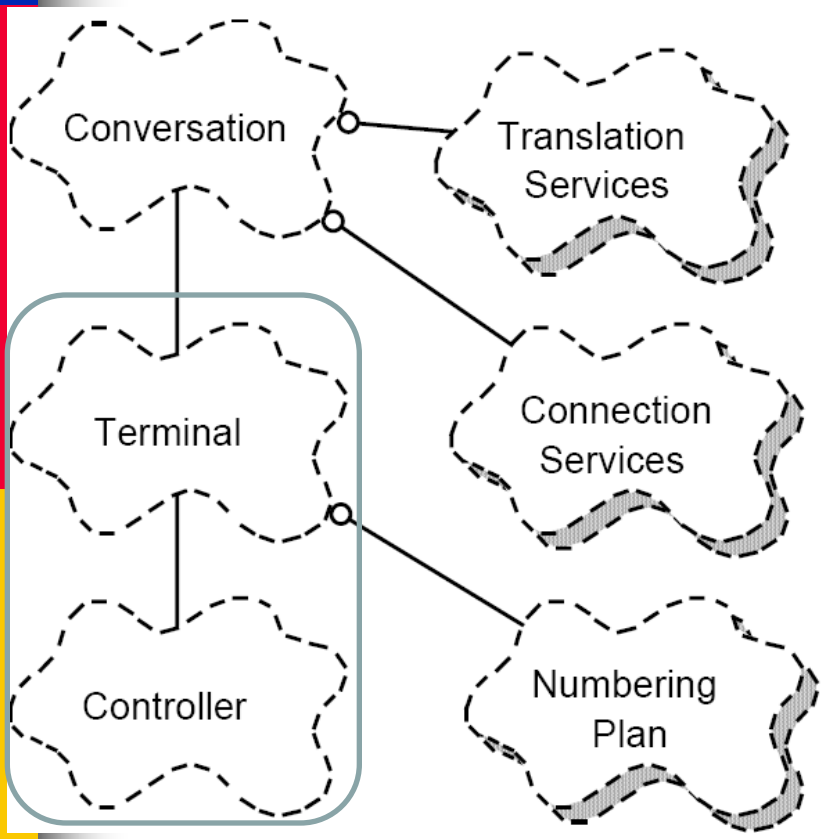
Logical View Example

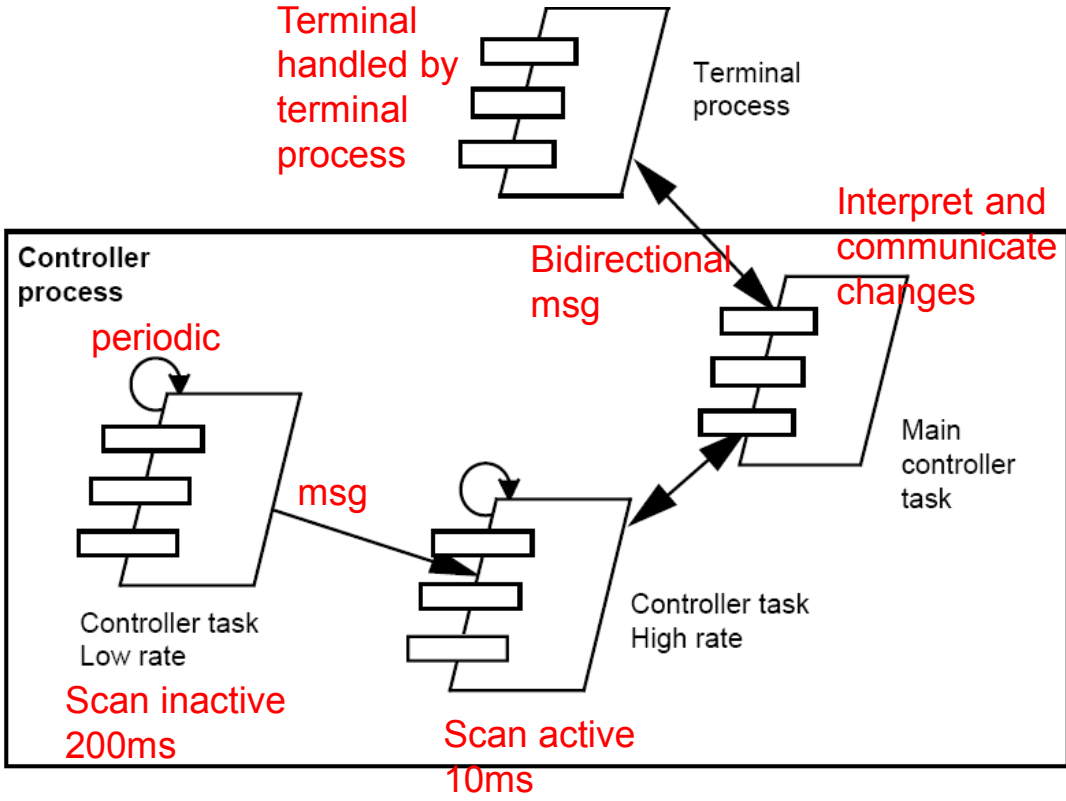


The Process View

- Captures the concurrency and synchronization aspects of a design
  - **Components**: Tasks/threads, processes
  - **Connectors**: Messages, RPC
  - **Stakeholders**: Integrators
  - **Concerns**: Performance, availability, fault tolerance
- Defines a grouping of *tasks* that form an executable unit
  - Major tasks are uniquely addressable
  - Minor tasks are helper tasks (e.g., buffering)
- Processes
  - Can be replicated for load distribution or improved availability
  - Flow of messages and process loads can be estimated and used to gauge performance

Process View



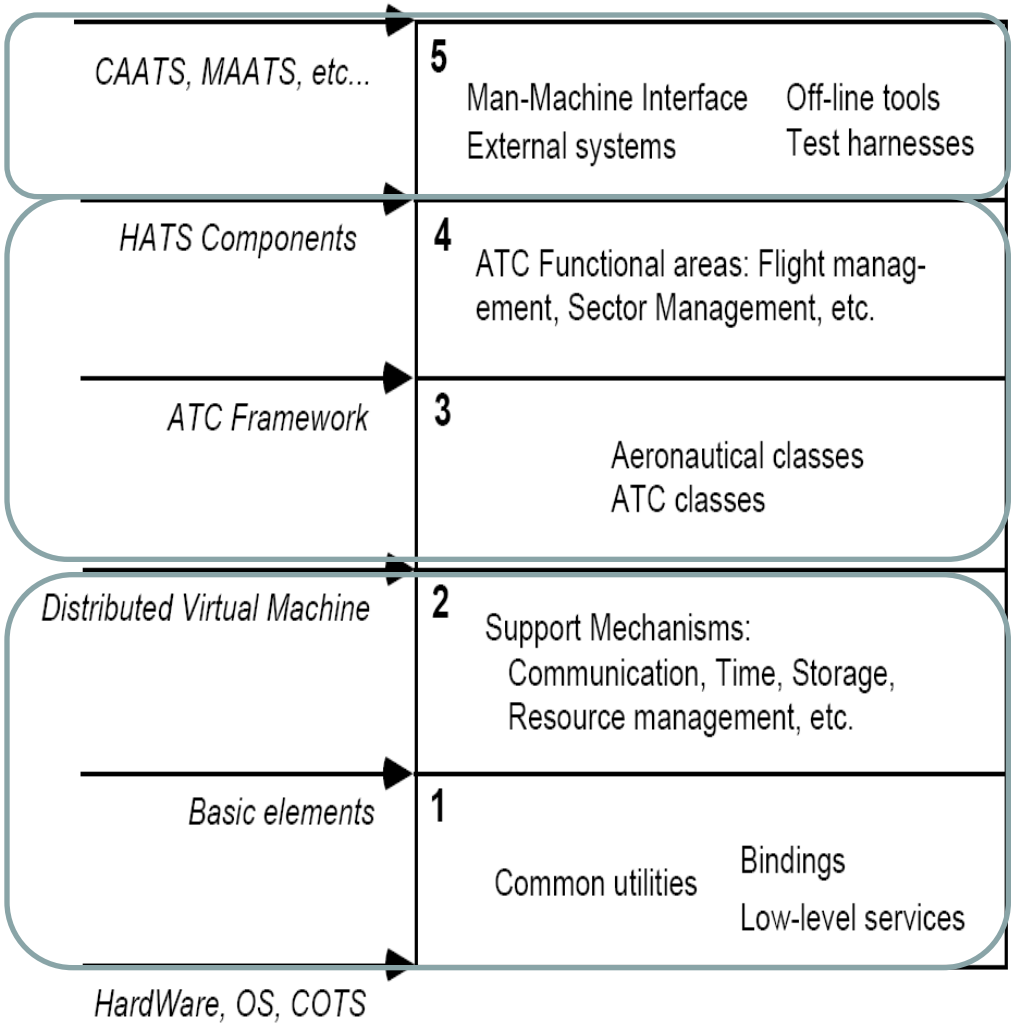


The Development View

- Describes the static organization of the software in its development environment
  - **Components**: Module/Subsystem
  - **Connectors**: Dependency (e.g. include)
  - **Stakeholders**: Developers
  - **Concerns**: Organization, reuse
- Why do we need development view?
  - Takes into account **internal requirements** related to ease of development, software management, reuse
  - Serves as basis for **work allocation**

Development View

72 subsystems



Customer specific

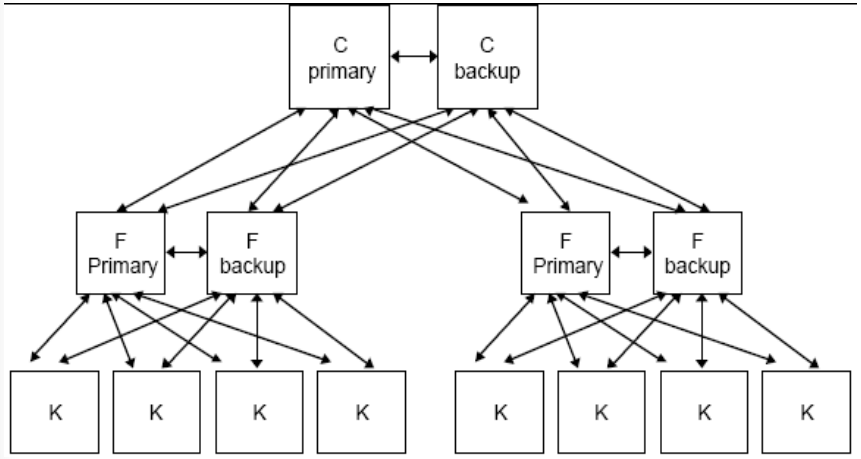
Domain specific

Domain independent

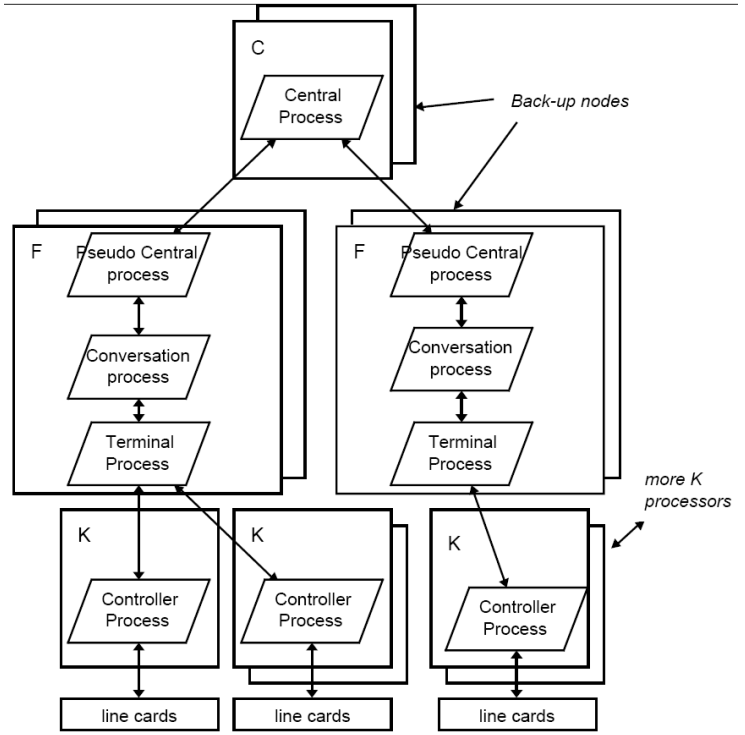
The Physical View

- Captures the physical (often HW) entities in a system and their interconnections
  - **Components**: Nodes
  - **Connectors**: Network (LAN, WAN)
  - **Stakeholders**: System designer
  - **Concerns**: Nonfunctional requirements (e.g. Scalability, performance, availability)

Physical View



C,F, and K are different types of computers



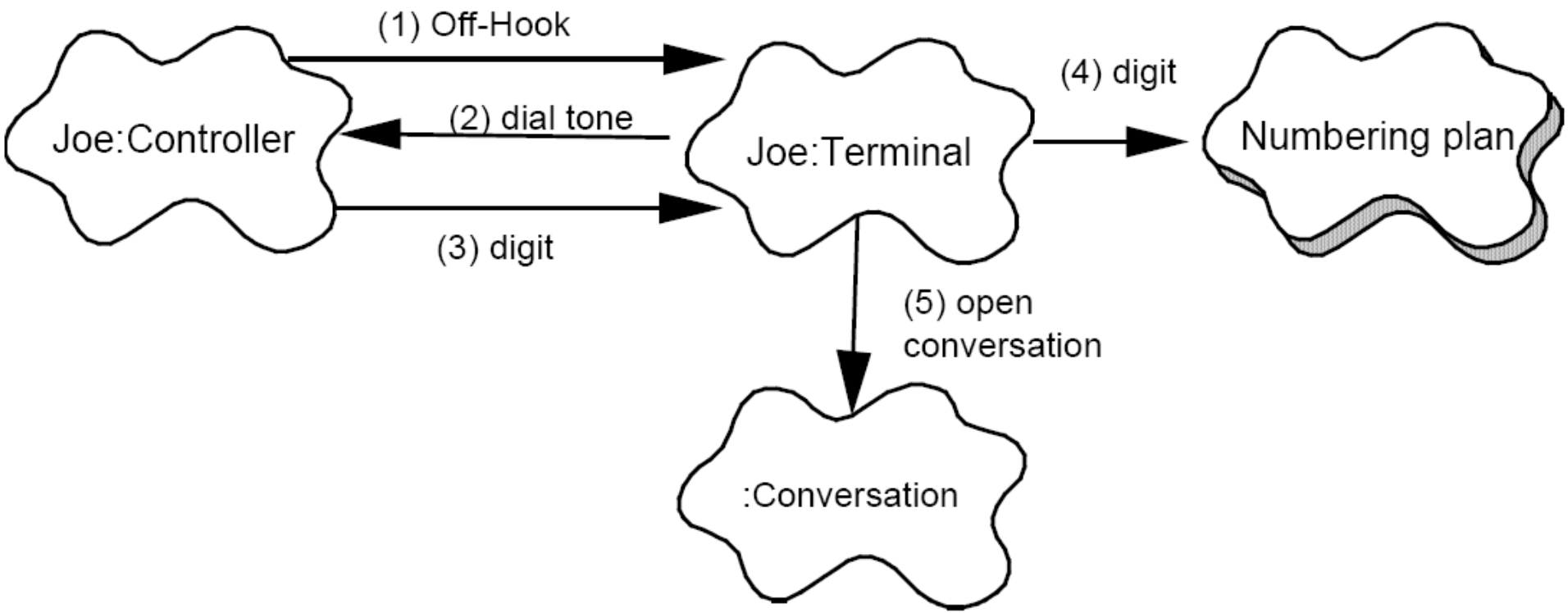
The Different Views

- **+1:** Use cases and scenarios to illustrate these views

Scenarios “+1” view

- Shows how the four views work together seamlessly
- Redundant with other views, hence “+1”
  - Drives the **discovery** of architectural elements during architecture design
  - **Validates** and illustrates role after architecture design in complete
- Representation is similar to logical view

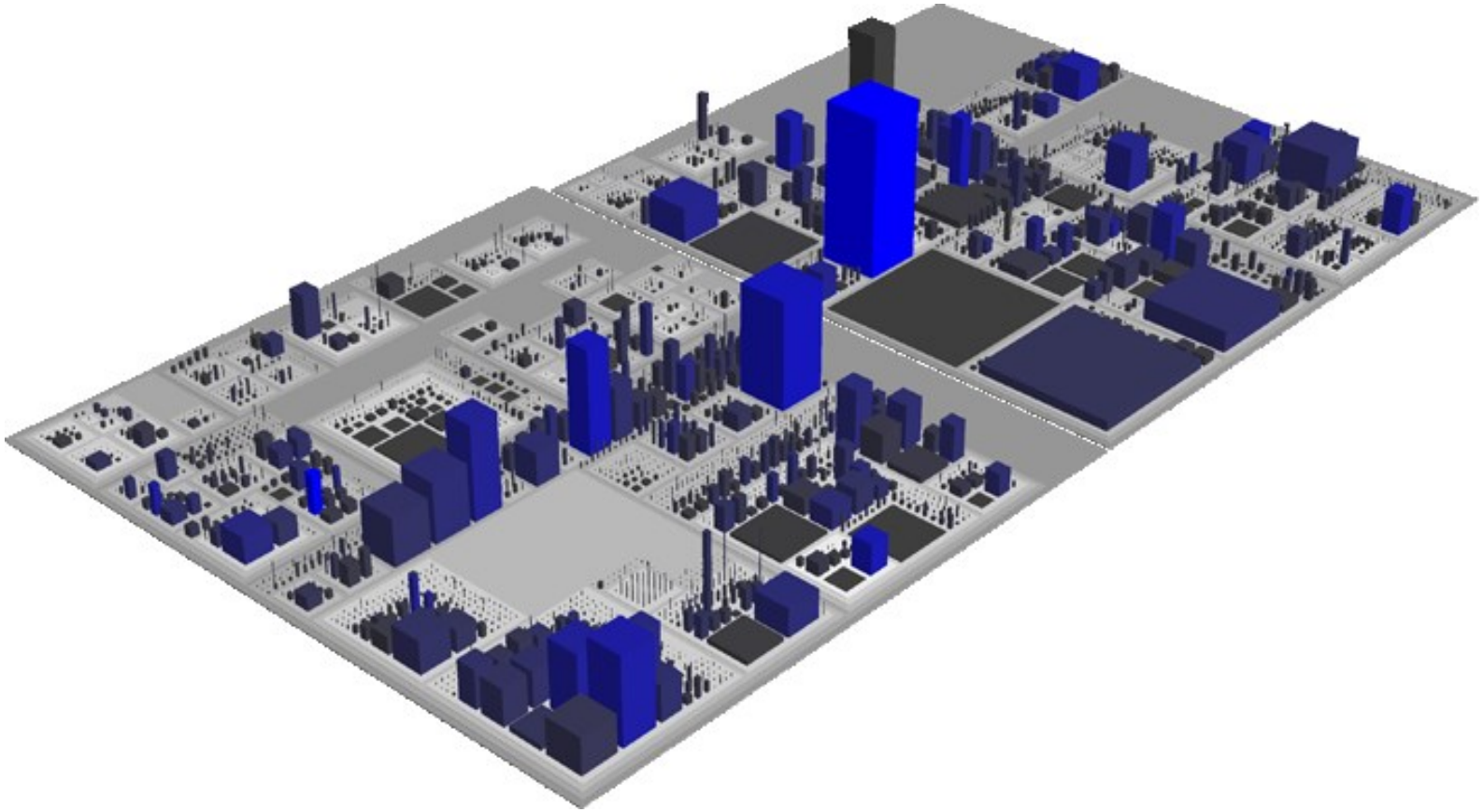
Scenarios “+1” View Example



Summary

- Different views address different concerns
- Not all views are necessary
- Lots of efforts needed to maintain these concurrent views, especially as the software system evolves

Other Visualizations for Software Architecture - Code City



http://wettel.github.io/codecity.html