Architectural Blueprint –

“The 4+1 View Model of Software Architecture”

Philippe Kruchten

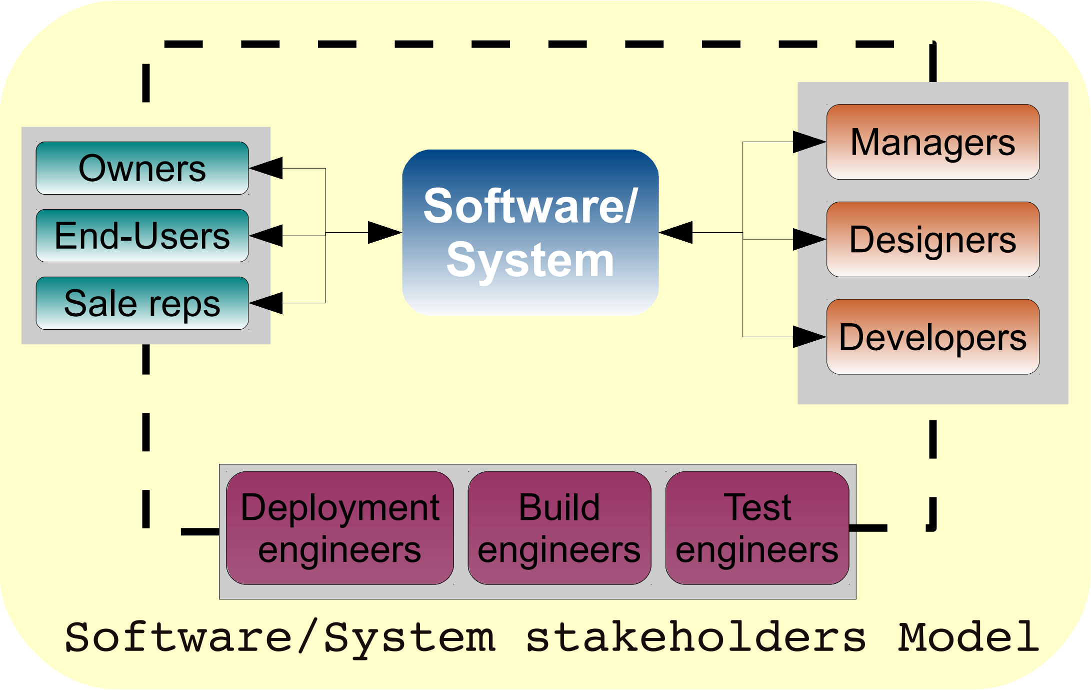
UNIVERSITY OF
Waterloo

Model

What is a model?

- simplified abstract representation
- information exchange
- standardization
- principals (involved)
- communication
  - channel
  - flow

I am a Model



Software/System stakeholders Model

Software Architectural Model

Definition

- *"a model to represent the architecture of a software system"*
  - which architecture (reference | conceptual | concrete)?

Desired Attributes

Addresses & captures

- concerns of various stakeholders
  - stakeholders
    - end-users, developers, system engineers, project management
    - testers, support teams
- requirements
  - functional
  - non-functional
    - performance, availability, concurrency, distribution, fault tolerance
    - security, testing, usability, configuration management, evolution, monitoring

Desired Attributes

An abstraction

- represents the high level view

Is robust

- adaptable
- scalable
- iterative

Meaningful & maintainable

- has to be a live document
  - changes with the system

Types

Box & Line model

Architectural definition languages

View based models

- Zachman Framework (1987)
- Three schema approach (1977)
- **4+1 view model (1995)**
- RM-ODP
- DoDAF

4+1 View Model

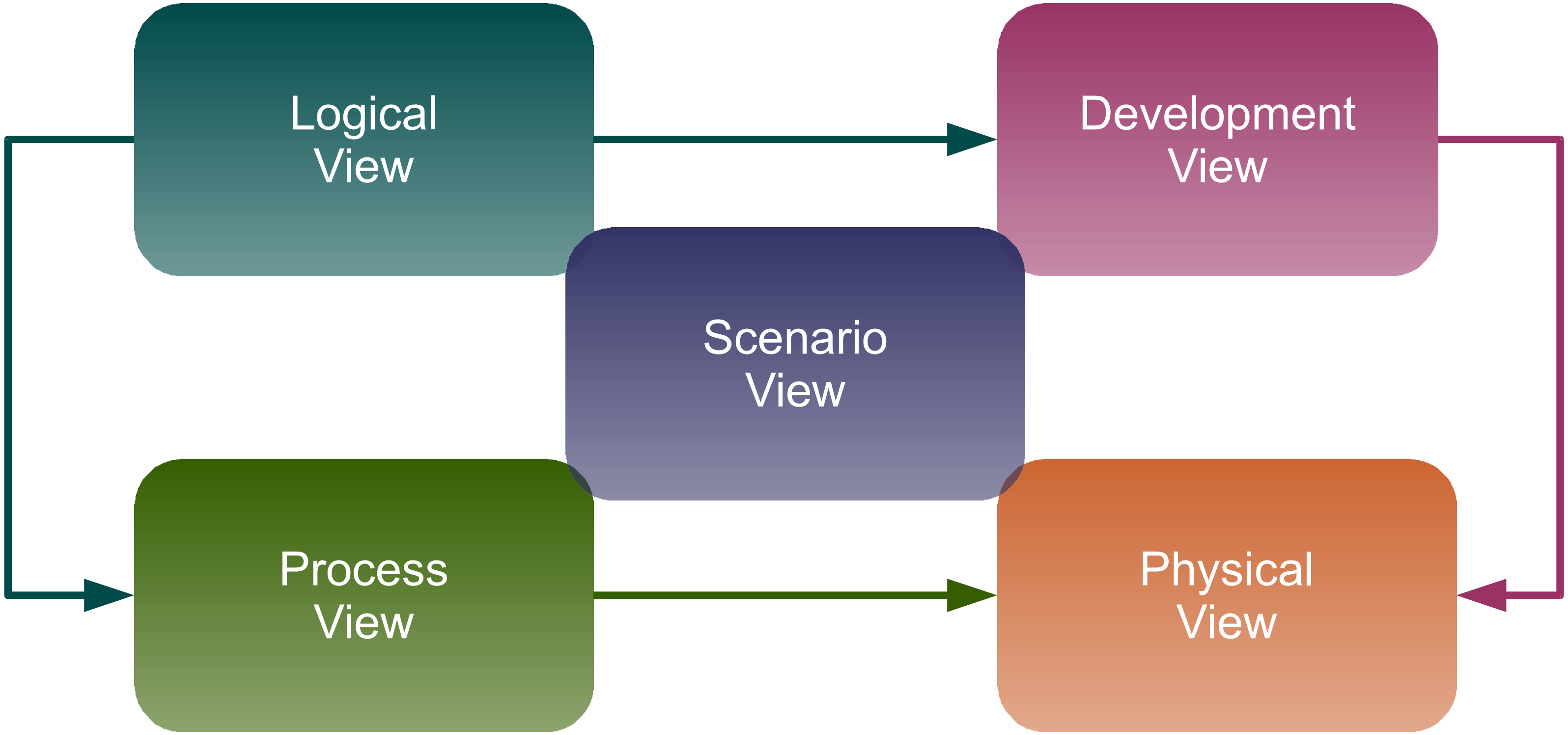
Model

- composed of 5 views
  - a single view is not enough

View

- is catered for a set of corresponding stakeholders
  - addresses the concerns of its stakeholders
- view elements
  - components, connectors, notation
- generic representation

4+1 View Model



Logical View

Intent

- is the object model of the design
- is generally the starting point
- addresses functional requirements
  - decomposition into “architectural entities”

Style

- abstract entities

Stakeholders

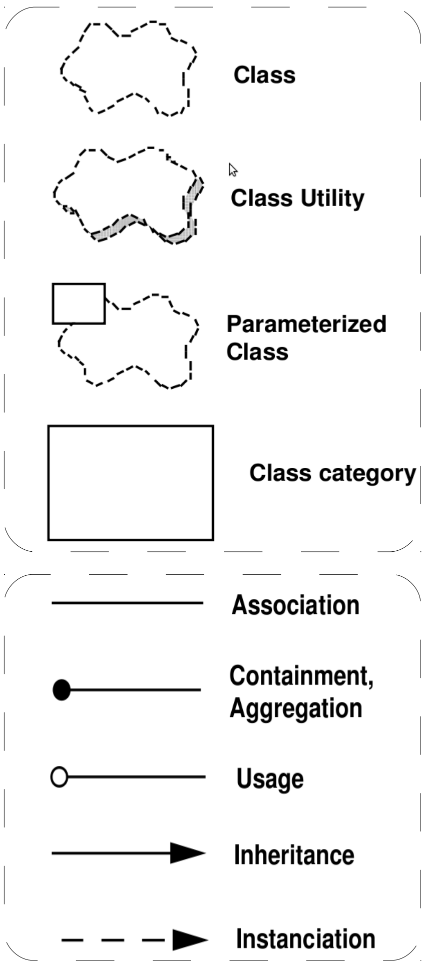
- end-users, architects, designers

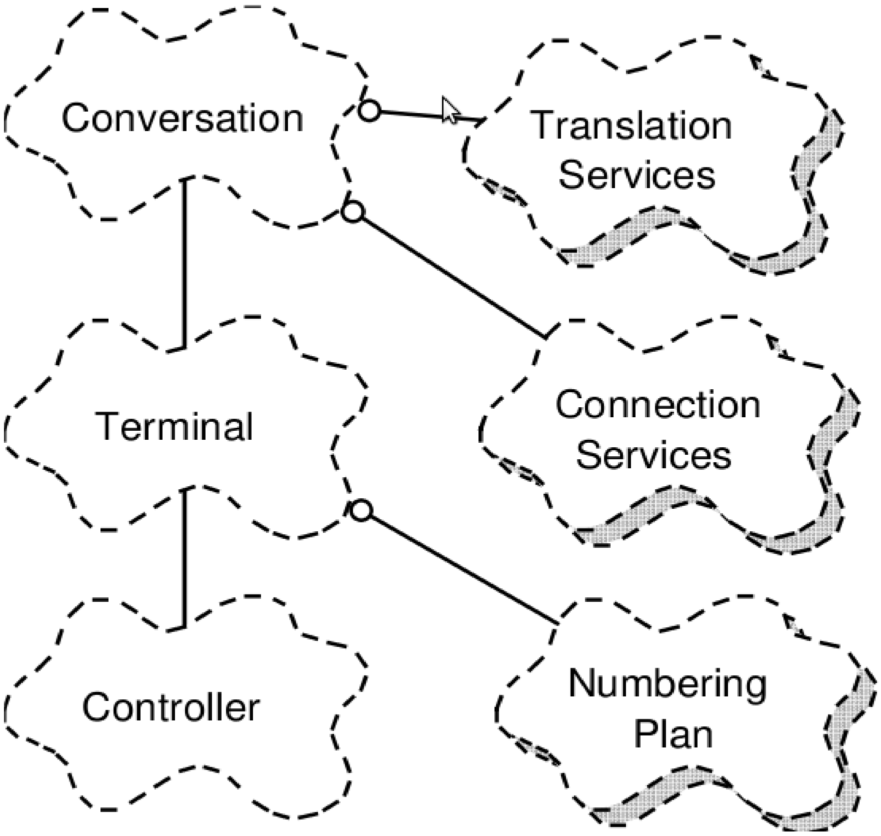
Logical View

View representation

- OOA (object oriented analysis)
  - entities are analysis classes
  - application of OOA principles
    - abstraction, encapsulation. inheritance
    - association (aggregation, composition)
  - class diagrams, state diagrams
- data centric analysis
  - entity relationship (ER) diagrams

Logical View





Logical View

Design guidelines

- a single object model across the system
- avoid premature specialization
  - entities
  - mechanisms (per site or per processor)

Process View

Intent

- handles the non-functional requirements
- provides an abstraction of architectural processes
  - process
    - **process**: grouping into executable units
    - **hierarchy**: major & minor tasks
    - **types**: atomic & distributed
  - process communication
    - messaging (synchronous, asynchronous, RPC, broadcast)
    - shared memory

Process View

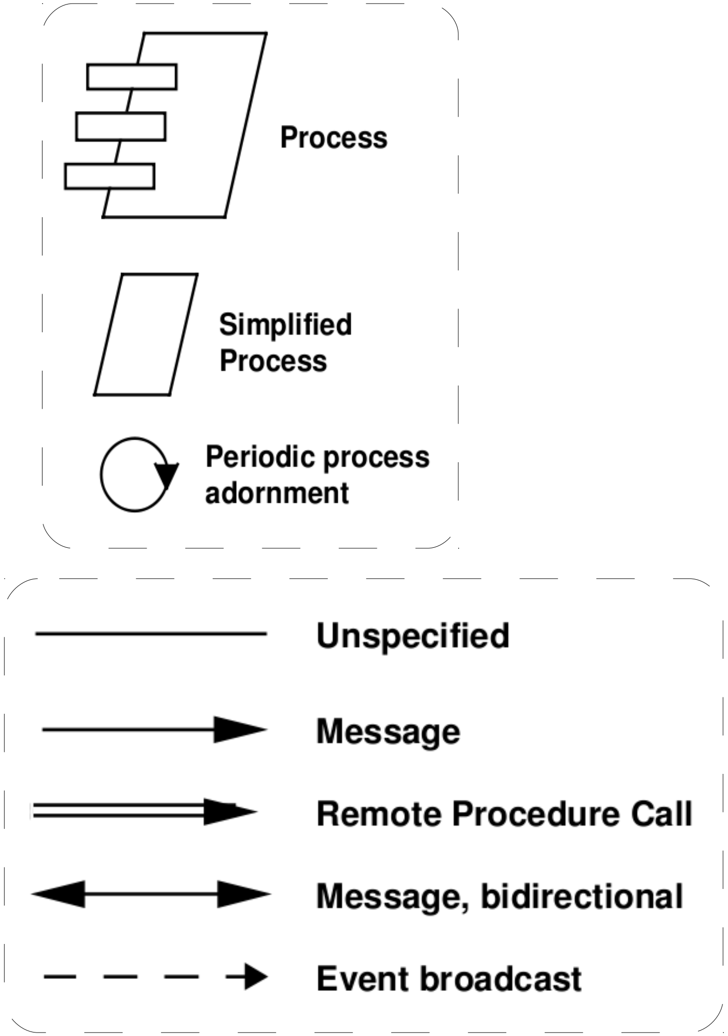
Style

- several styles are applicable
  - pipes & filters
  - layered
    - client / server

Stakeholders

- integrators, architects

Process View



Process

Simplified Process

Periodic process adornment

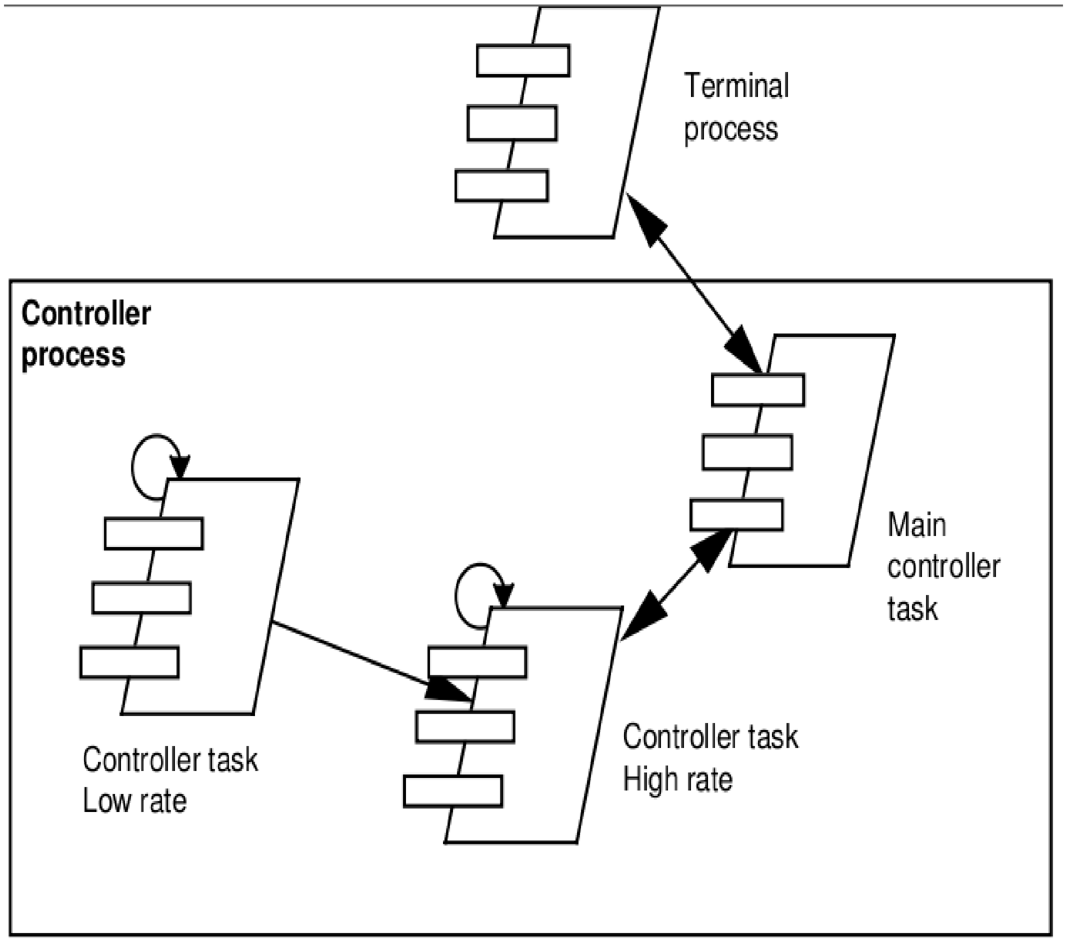
Unspecified

Message

Remote Procedure Call

Message, bidirectional

Event broadcast



Terminal process

Controller process

Main controller task

Controller task Low rate

Controller task High rate

Development View

Intent

- software/system decomposition into software modules
- software modules
  - name space, packages, libraries, subsystems
  - modules are scoped for small (development) teams

Driven by internal requirements

- management, requirement allocation, cost evaluation, progress monitoring
- reuse, commonality, programming language and development environment

Development View

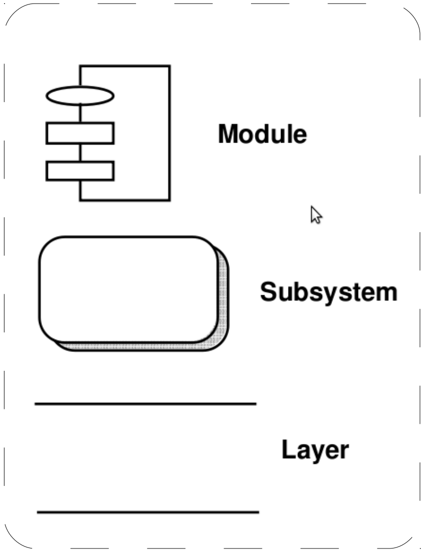
Style

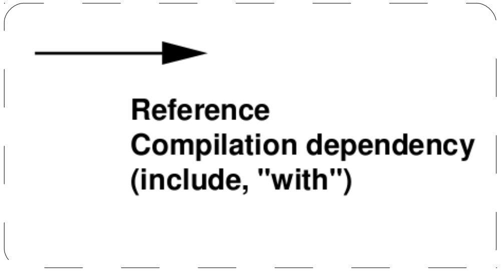
- layered style
  - each layer with **well defined** interface
  - subsystem dependencies on other subsystems
    - in the same layer or lower
  - each layer provides a development abstraction (**responsibility**)

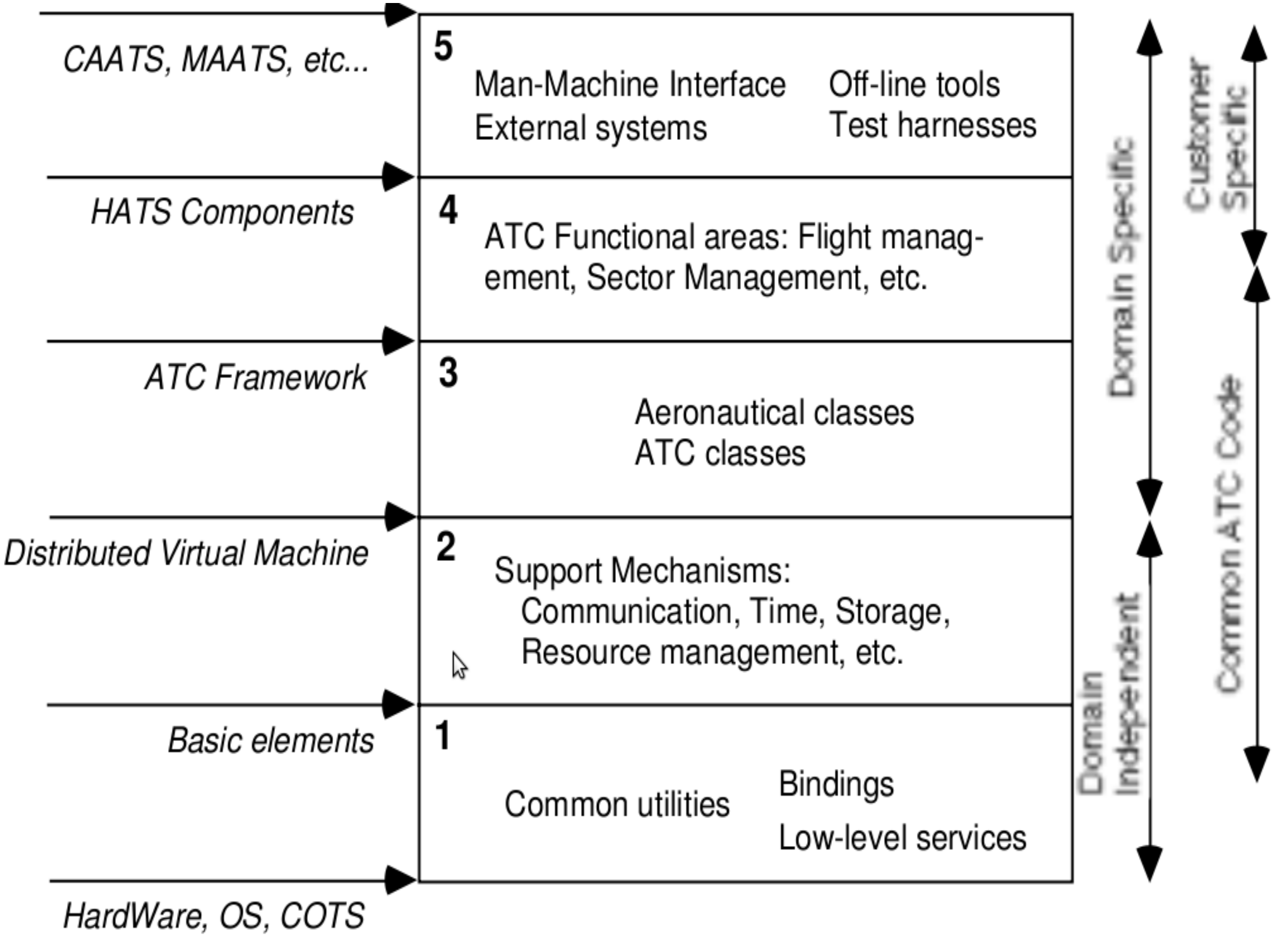
Stakeholders

- managers, architects, designers, developers, testers

Development View







5 Man-Machine Interface, External systems; Off-line tools, Test harnesses

4 ATC Functional areas: Flight management, Sector Management, etc.

3 Aeronautical classes, ATC classes

2 Support Mechanisms: Communication, Time, Storage, Resource management, etc.

1 Common utilities; Bindings, Low-level services

Physical View

Intent

- physical manifestation of process view
  - processes are mapped to processing nodes

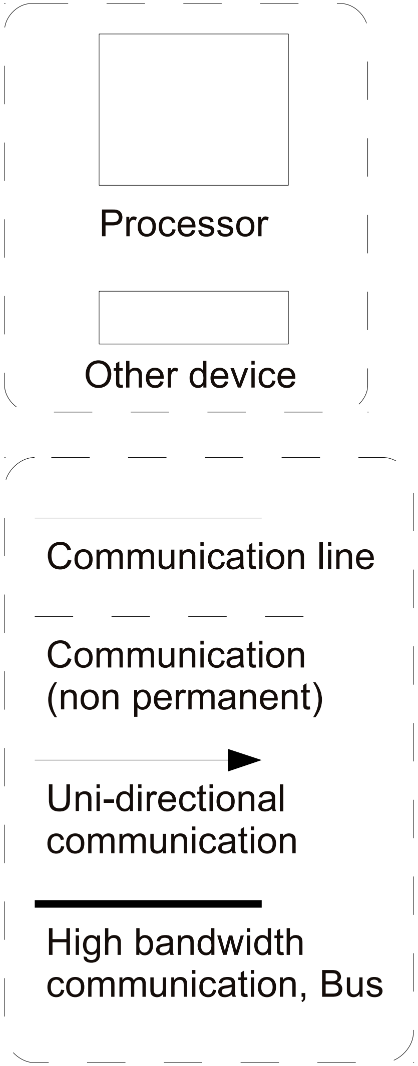
Concerns

- installation, configuration, deployment & delivery, networking, messaging protocols

Stakeholders

- system engineers, installers, architects, operators

Physical View



Processor

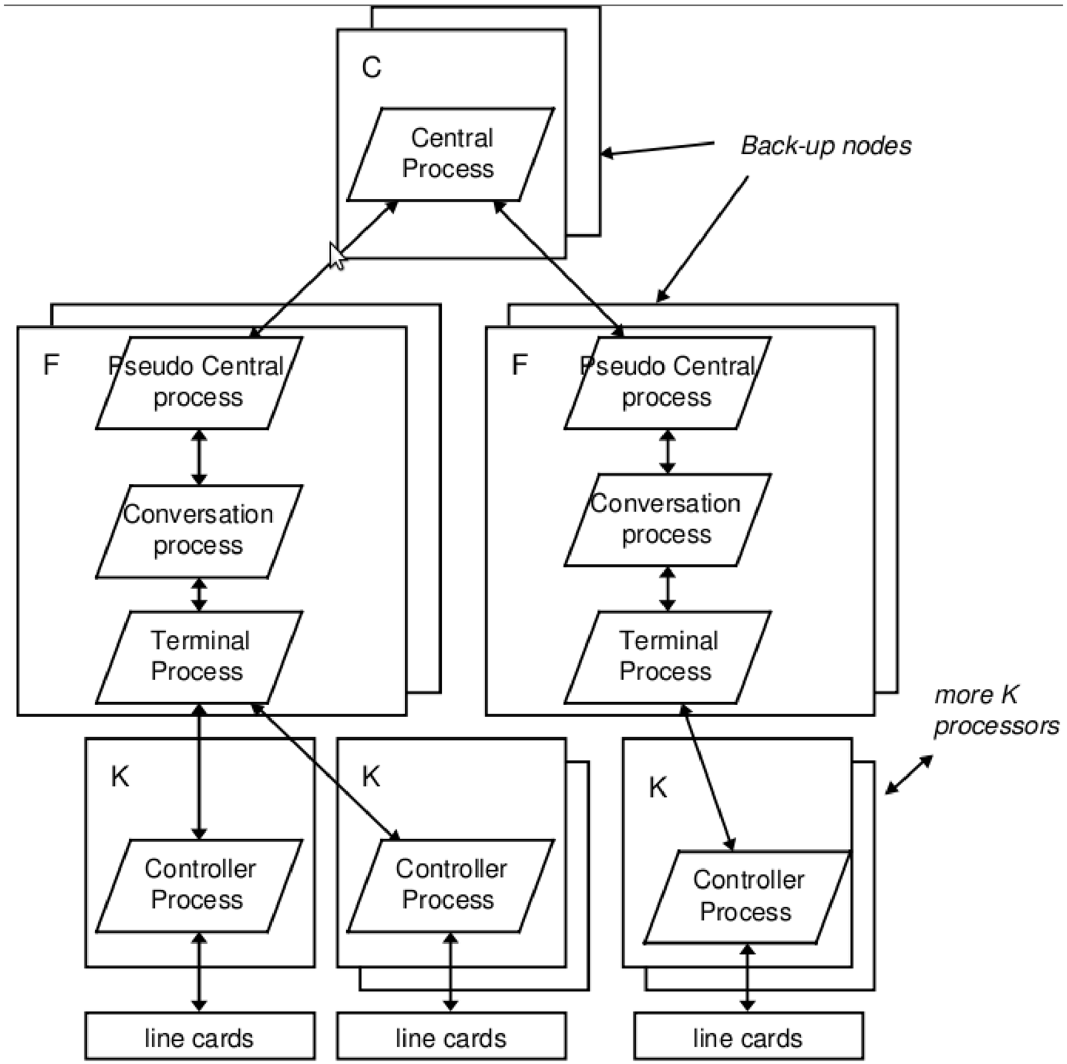
Other device

Communication line

Communication (non permanent)

Uni-directional communication

High bandwidth communication, Bus



Physical View

Design guidelines

- mapping to be flexible
- minimal impact on source code

Scenario View

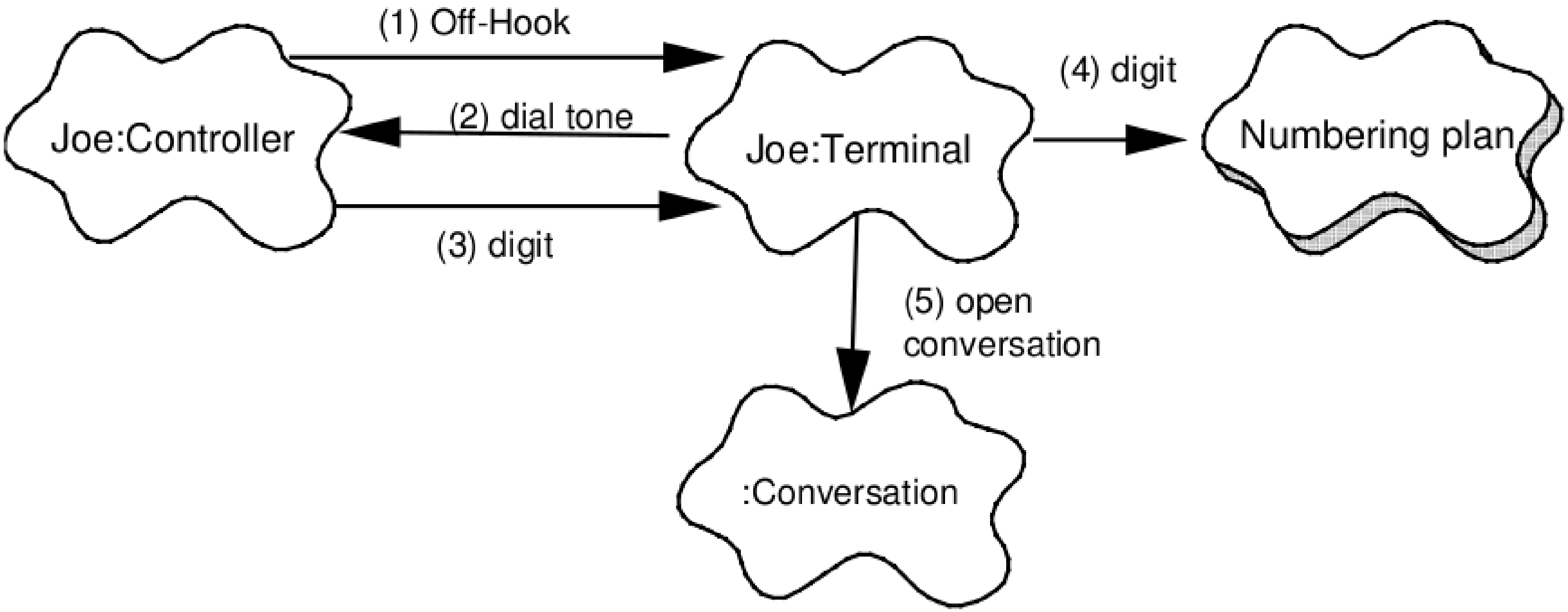
Intent

- one rule to rule them all
- capture system functionality in scenarios
  - interaction of objects & processes
  - driven by important scenarios
- provides architecture validation

Stakeholders

- all stakeholders from the other views

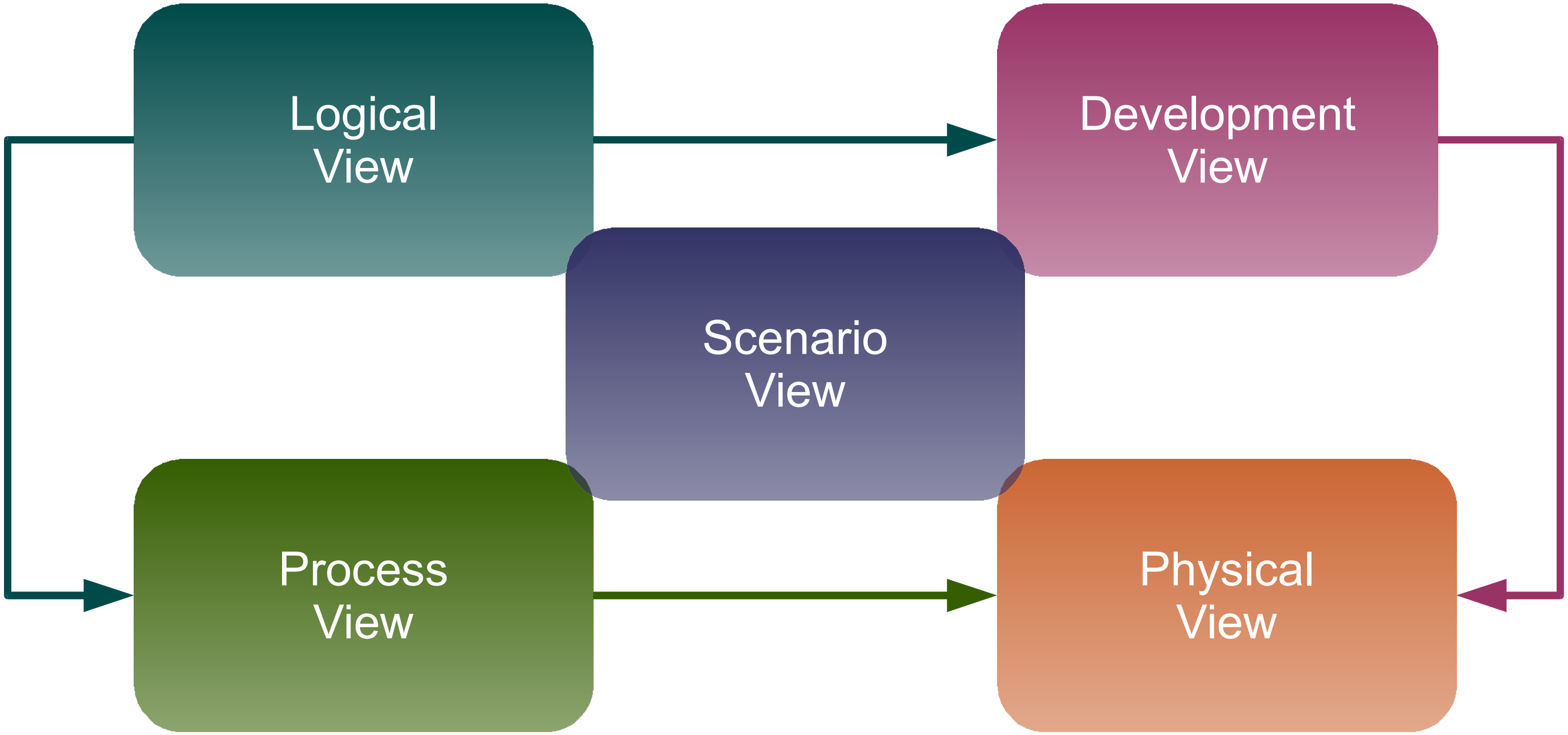
Scenario View



Components from the logical view

Connectors from the process view

View Mappings



Logical to Process View

Objects are mapped to processes

- considerations
  - autonomy
  - persistence
  - subordination
  - distribution

Strategy

- inside-out: identify processes for objects
- outside-in: identify processes (based on system requests) and then allocate objects to these processes

Logical to Development View

Architectural component decomposition

- architectural entities are broken down into design components
  - packages, modules
  - classes
- mapping is governed by development concerns
- distance between logical and design view
  - an indication of the size of the system

Process to Physical View

Processes assignment to hardware

- major and minor tasks are assigned to physical machines
- various configurations
  - development
  - testing
  - deployment

Iterative Process

Start with a model

In each iteration the architecture is

- prototyped
- tested: under load if possible
- measured & analyzed
- refined
  - add more scenarios
  - detect abstractions and optimizations
- each iteration should takes us a step closer to a stable architecture

Discussion

Lacks some fundamental views

- security, user interface, testing
- upgrade, disaster recovery

Are the views ever complete

Change in architectural style

- data centric to OO architecture

Meet Kruchten

Phillipe Kruchten

- professor of software engineering at the University of British Columbia
- professional software engineer with 30+ years of experience

Major Work

- RUP
- Canadian automated air traffic control system – lead designer