Software Architecture in Practice

Architectural description

Overview

Definition of Software Architecture

Introduction to the POS case

Motivation

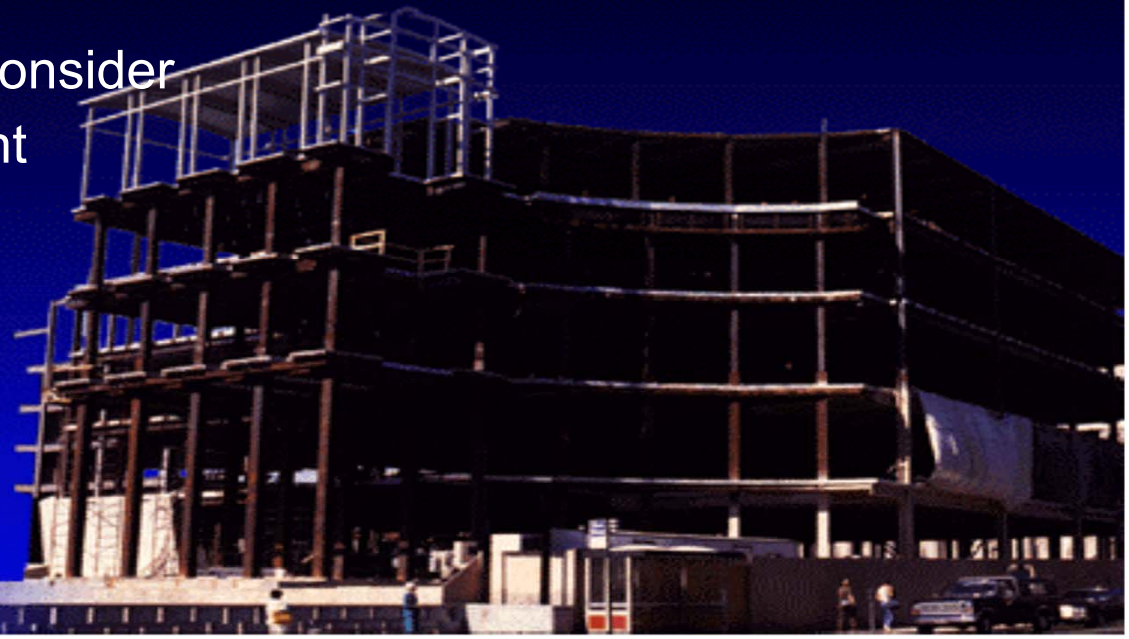
Architectural description

- Formal description languages
- Using UML

What Is Software Architecture?

Software architecture is the structure or structures of the system, which comprise software elements, the externally visible properties of these elements, and the relationships among them.

The exact structures to consider and the ways to represent them vary according to engineering goals.



Implications of this Definition – 1

A software architecture is an abstraction of a system.

- Architecture defines elements and how they interact.
- Architecture suppresses purely local information about elements; private details are not architectural.

Externally-visible properties of elements are assumptions that one elements can make about another:

- provided services, required services, performance characteristics, fault handling, resource usage

Implications of this Definition – 2

Every system *has* an architecture.

- Every system is composed of elements and there are relationships among them.
- In the simplest case, a system is composed of a single elements, related only to itself.

Just having an architecture is different from having an architecture that is known to everyone:

- The architecture versus specification of the architecture

If you don't explicitly develop an architecture, you will get one anyway – *and you might not like what you get.*

Implications of this Definition – 3

This means that box-and-line drawings alone are *not* architectures; but they are just a starting point.

- You might *imagine* the behavior of a box labeled "database" or "executive" -- but that's all
- You need to add specifications and properties.

Systems have many structures (views).

- No single structure can be *the* architecture.
- The set of candidate structures is not fixed or prescribed: choose whatever is useful for analysis, communication, or understanding.

A Simple Case

NextGen Point-Of-Sales (POS) System

- Record sales and handle payments
  - Typically used in retail stores
- Hardware
  - Terminal
  - Barcode scanner
- Interfaces with external systems
  - Inventory
  - Accounting
  - …

From [Larman, 2001]

- See also note on architecture description using UML, [Christensen et al., 2004]

Why Focus on Description?

- Architecture as a means for communication among stakeholders
  - Need suitable (stakeholder-dependent) representations
  - Architect -> developer
    - Needs precise understanding of design choices
    - Architecture as “blueprint” for development
  - Architect -> customer
    - Precision needs to be balanced with ability to understand
      - Box-and-line vs. formal
- Architecture as basis for design and evaluation
  - Precise semantics of description beneficial to analyse non-trivial properties
  - Support analytical evaluation
    - questioning techniques
    - automated tools

Architecture Description Languages (ADLs)

Languages for describing software architectures

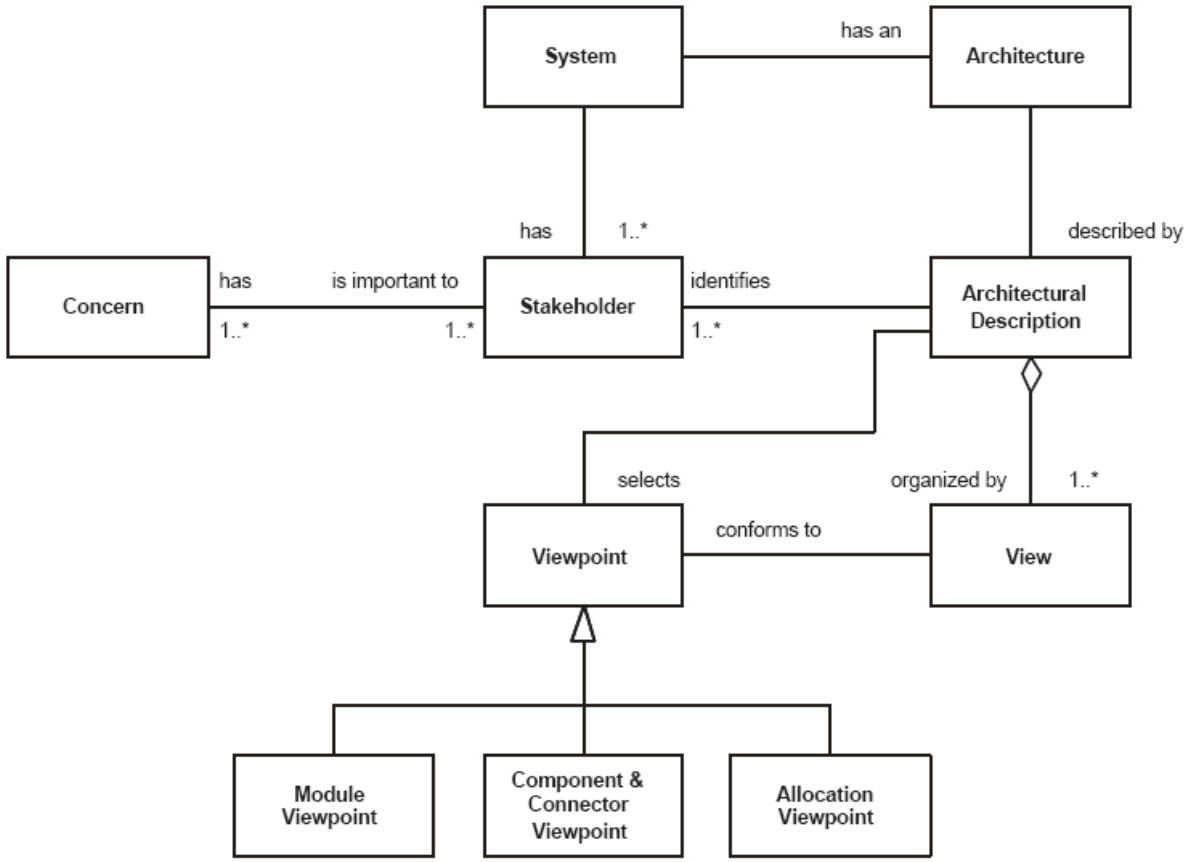
- Broad categories
  - Box-and-line drawings
    - Not really an ADL
  - Formal descriptions
  - Multiple view-based descriptions

This course

- Unified Modeling Language (subset) as example of multiple-view descriptions
  - Core, [Christensen et al., 2004]
- Formal description languages
  - Briefly…

An Ontology of Architectural Descriptions

Here: ontology = model of concepts



Developed from [IEEE 1471, 2000]

Elements of an Architectural Description

Architectural views (N)

- What is the software architecture?
  - Multiple viewpoints, here
    - Module viewpoint
    - Component & Connector viewpoint
    - Allocation viewpoint

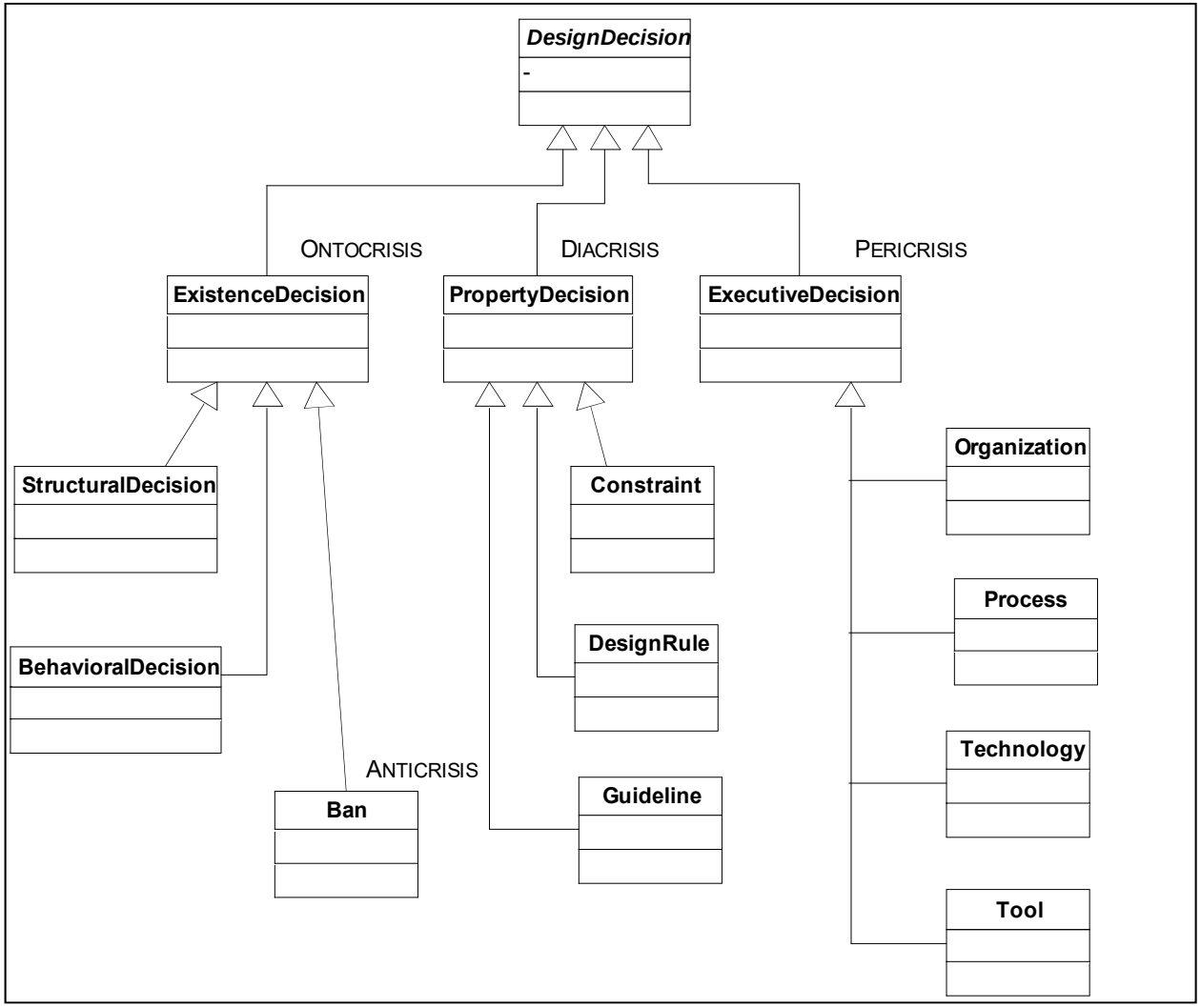
Architectural requirements (+1)

- Why is the software architecture the way it is?
  - Scenario-based requirements
    - E.g., paths through significant use cases
  - Quality-attribute-based requirements
    - Primary concerns (e.g., critical quality requirements)
    - Quality attribute specifications
  - Design decisions

Architectural Requirements: POS Scenarios

Process Sale: A customer arrives at a checkout with items to purchase. The cashier uses the POS system to record each purchased item. The system presents a running total and line-item details. The customer enters payment information, which the system validates and records. The system updates inventory. The customer receives a receipt from the system and then leaves with the item

Types of Design Decisions [Kruchten, 2004]



Architectural Requirements: POS Qualities

Architectural drivers

- Availability
  - The system shall be highly available since the effectiveness of sales depends on its availability
- Portability
  - The system shall be portable to a range of different platforms to support a product line of POS systems
- Usability
  - The system shall be usable by clerks with a minimum of training and with a high degree of efficiency

This is not operational!

- More later on quality attribute scenarios…

Architectural Views

How is the functionality of the system mapped to runtime components and their interaction?

- Component & connector viewpoint/structure

How is the functionality of the system to be mapped into implementation?

- Module viewpoint/structure

How are software elements mapped onto environmental structures?

- Allocation viewpoint/structure

Component & Connector Viewpoint

The software architecture of a computing system is the structures of the system, which comprise software elements, the externally visible properties of those elements, and the relationships among them [Bass et al., 2003]

Elements
- Components
  - Functional behaviour
  - What part of the system is doing what?

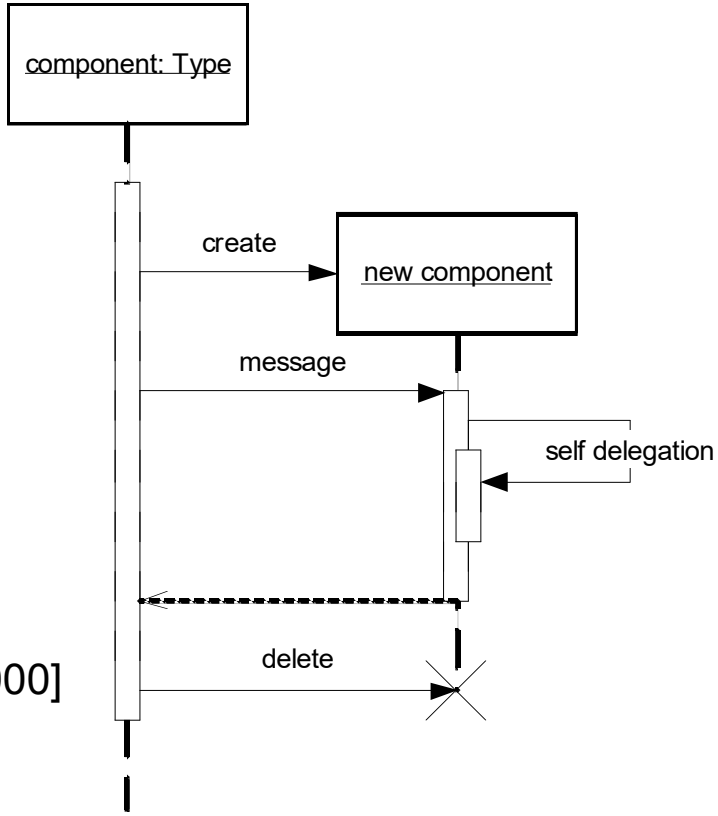
Relations
- Connectors
  - Control and communication aspects
  - Define protocols for control and data exchange
    - Incoming and outgoing operations
    - Mandates ordering of operations
    - Define roles for attached components

Mapping to UML
- Object diagrams, interaction diagrams
- Components = active objects
- Connectors = links + annotations, messages

+ textual description of responsibilities

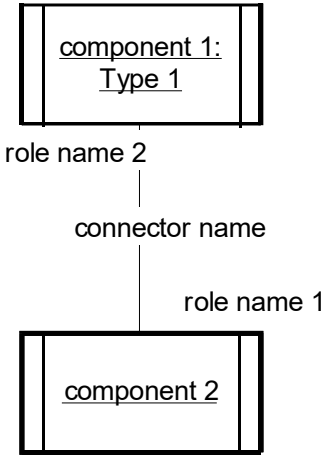
Basic C&C Elements

Sequence Diagrams

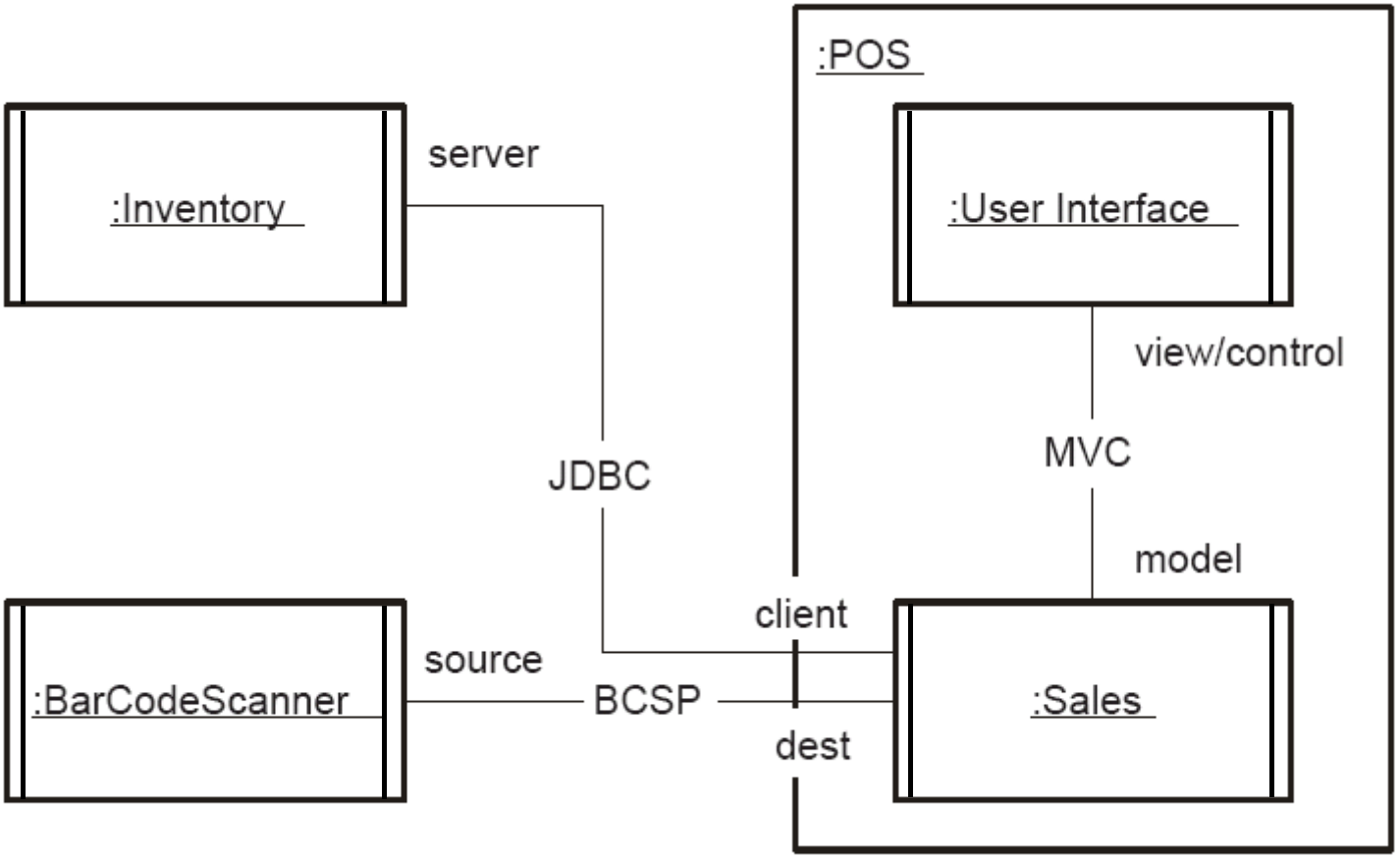


[Fowler, 2000]

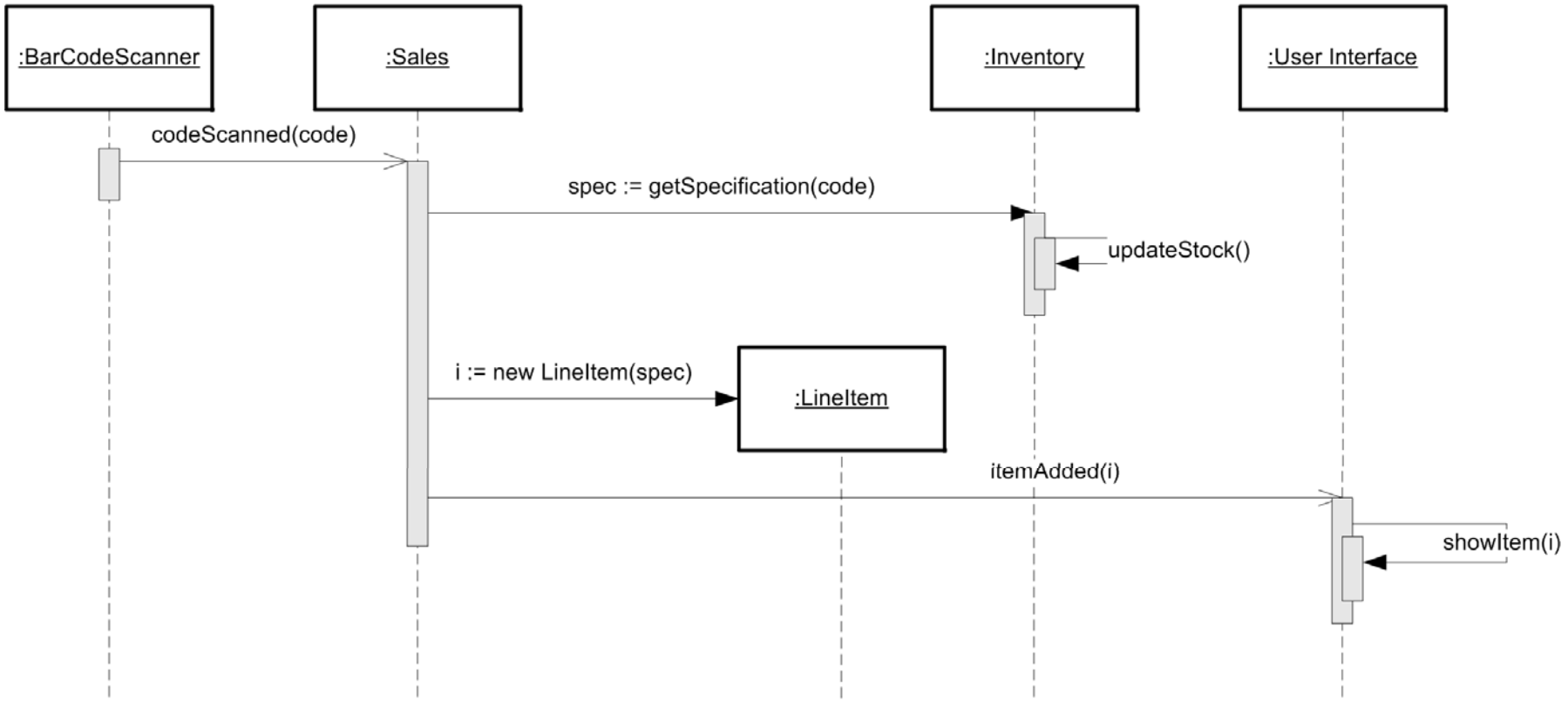
Active Objects



POS Example: C&C View



POS Example: C&C View (2)



Module Viewpoint

Elements

- Classes, packages, interfaces

Relations

- Associations, generalizations, realizations, dependencies

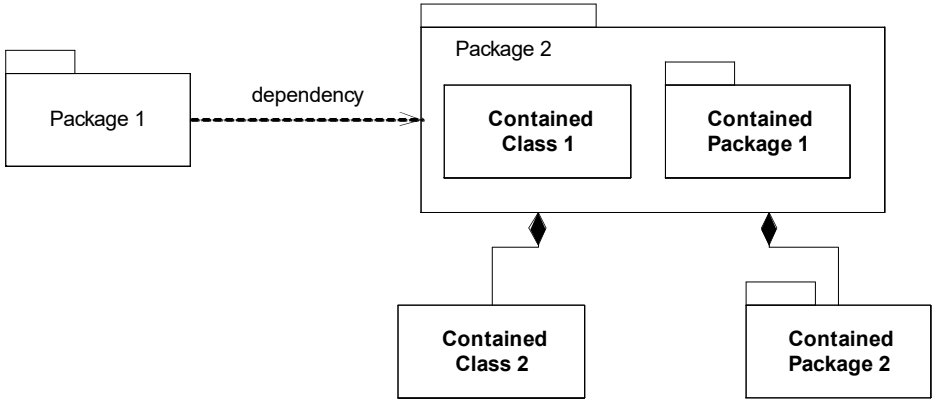
Mapping to UML

- Class diagrams…

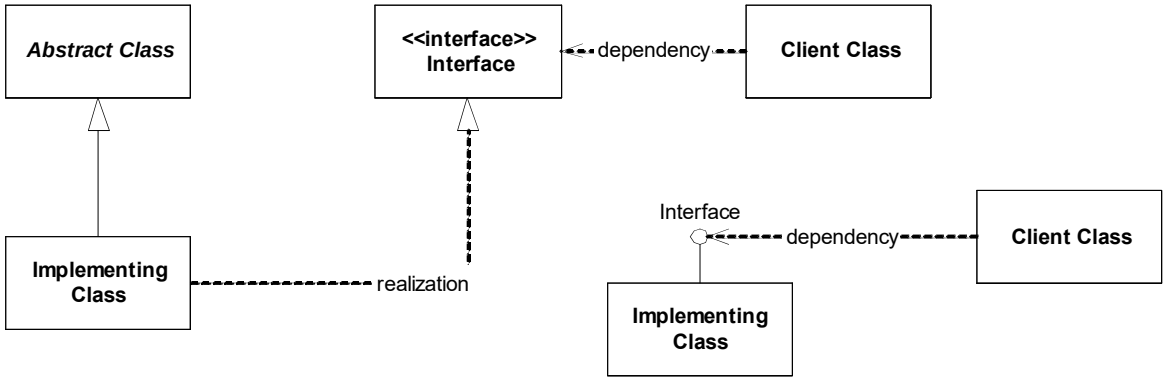
> The software architecture of a computing system is the structures of the system, which comprise software elements, the externally visible properties of those elements, and the relationships among them [Bass et al., 2003]

Basic Module Elements (1)

Packages

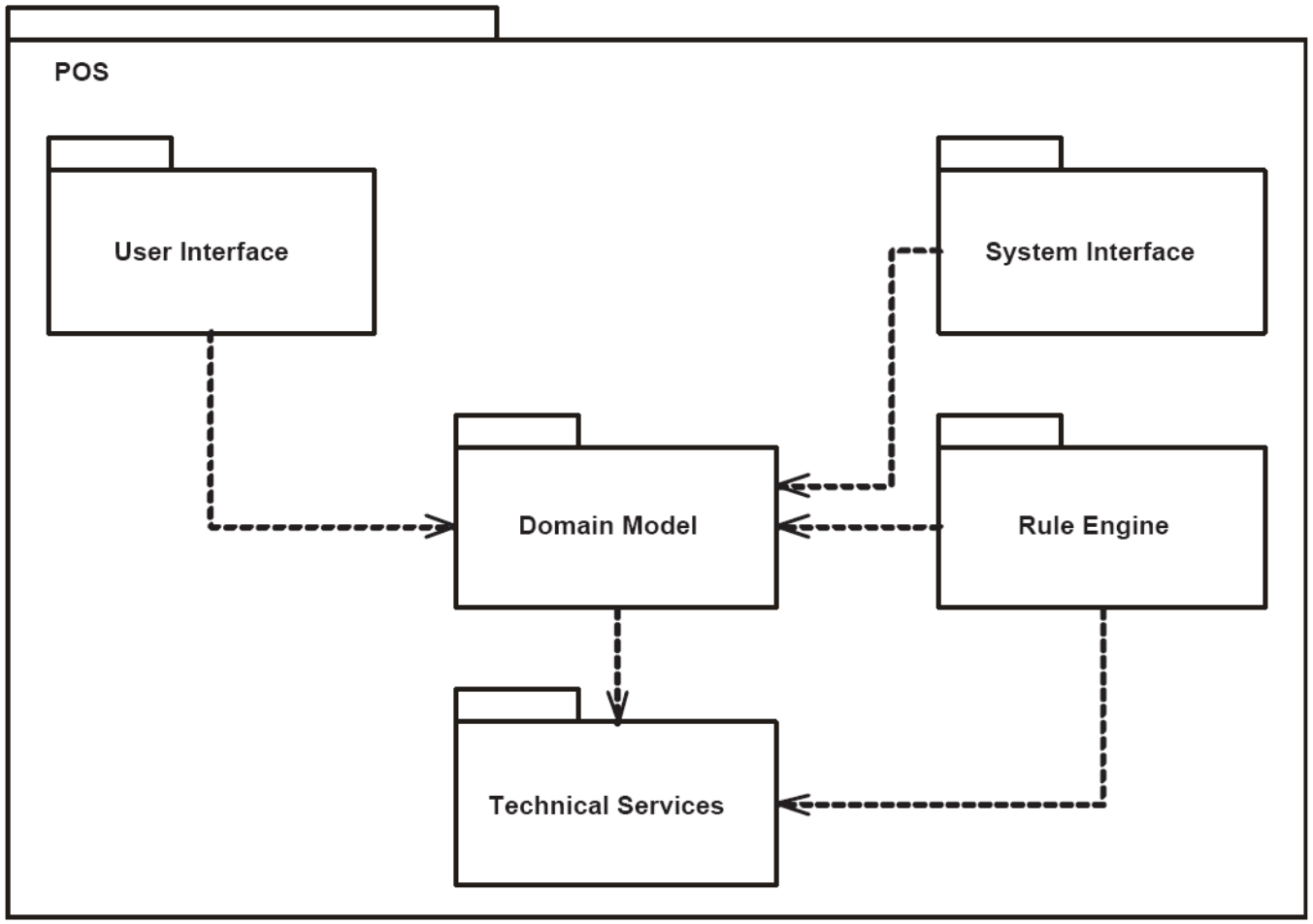


Interfaces

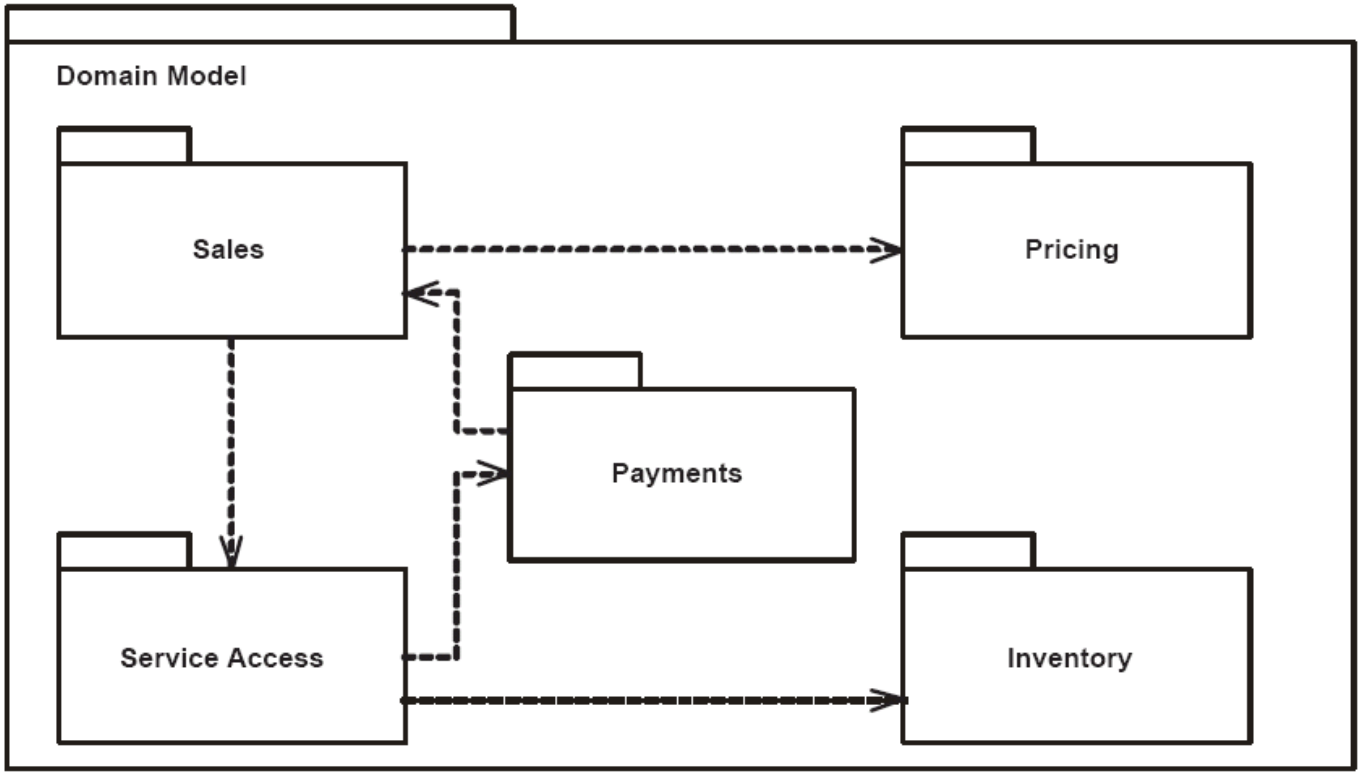


[Fowler, 2000]

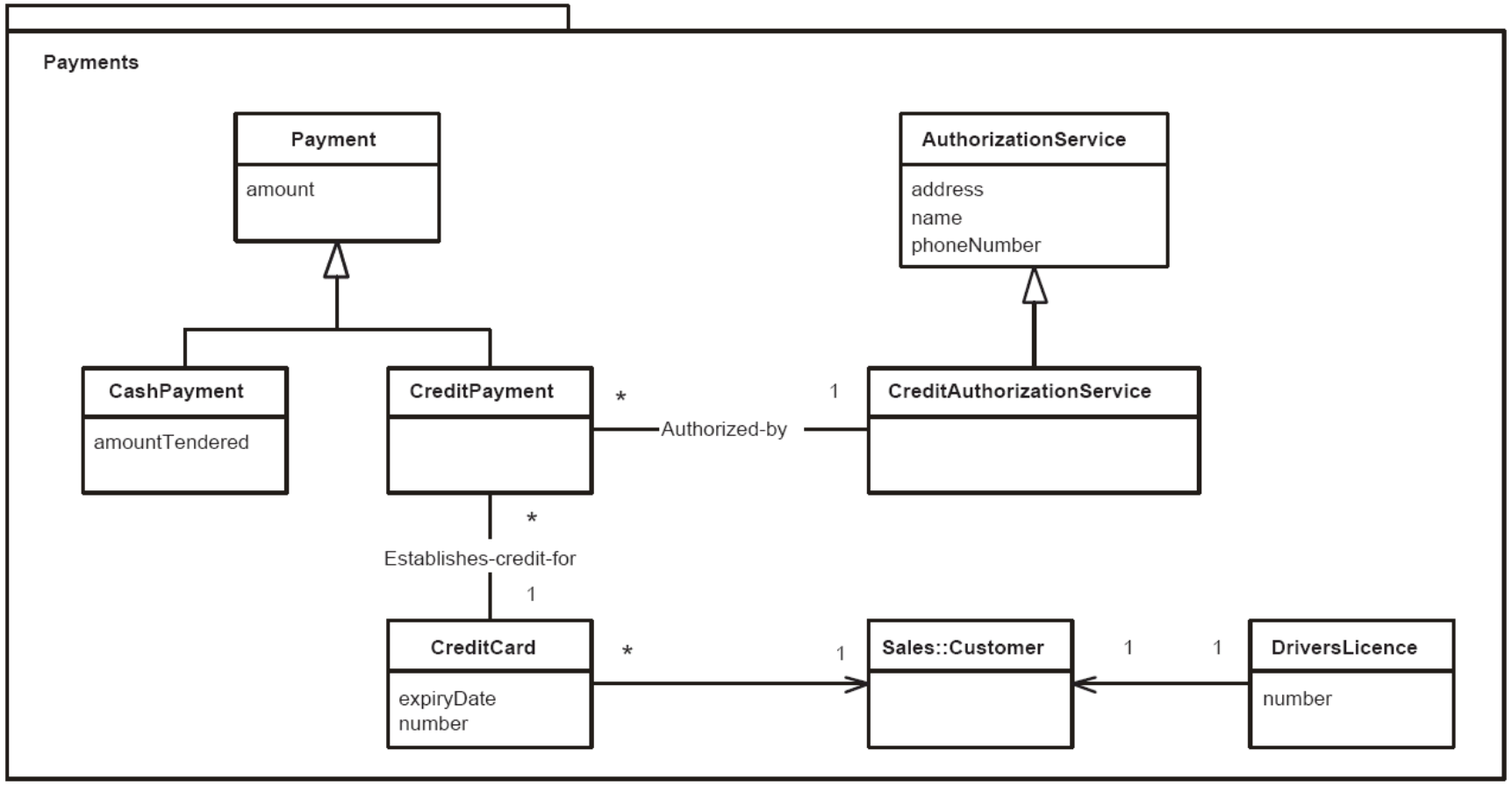
POS Example: Module View (1)



POS Example: Module View (2)



POS Example: Module View (3)



Allocation Viewpoint

Elements

- Software elements: Components, objects
- Environmental elements: Nodes

Relations

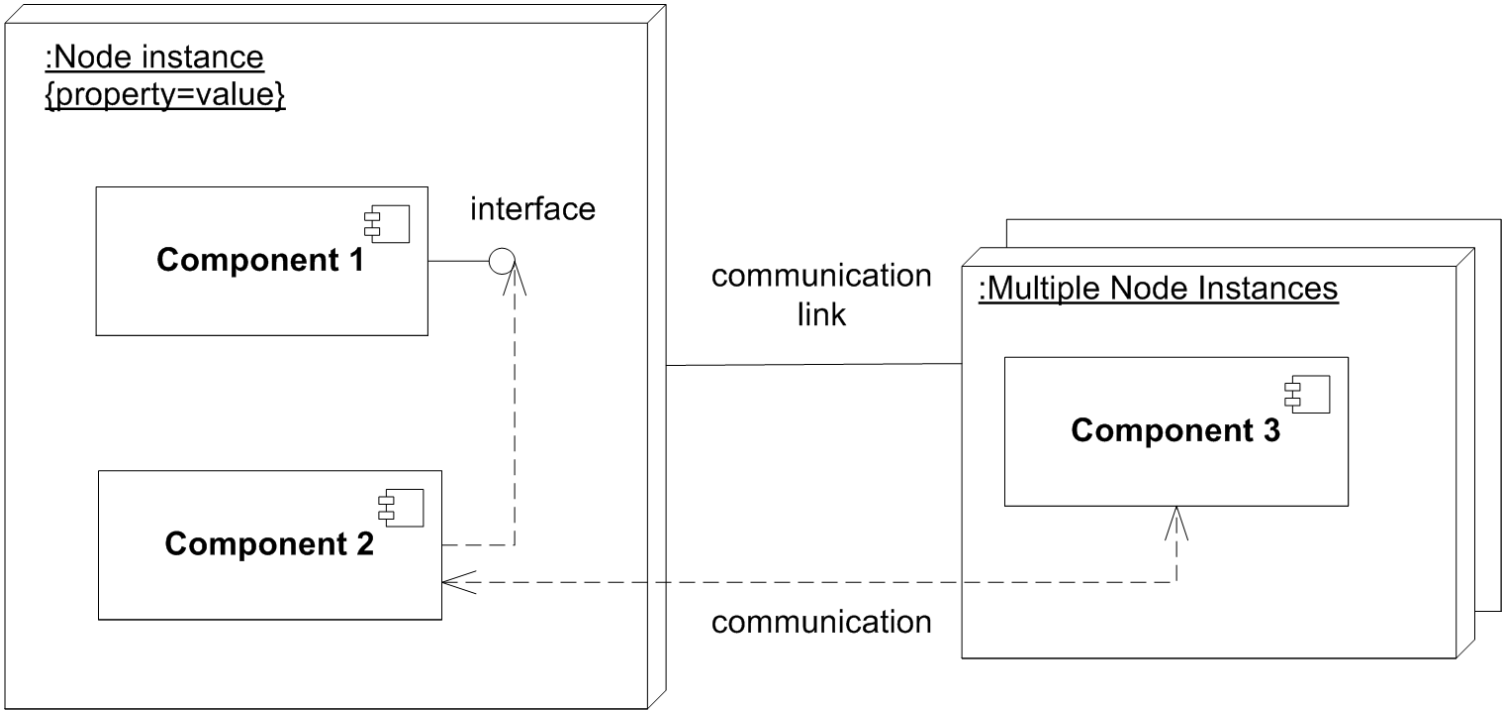
- Allocated-to
- Dependencies
- Connections (communication paths)

Mapping to UML

- Here: focus on deployment
  - Deployment and component diagrams

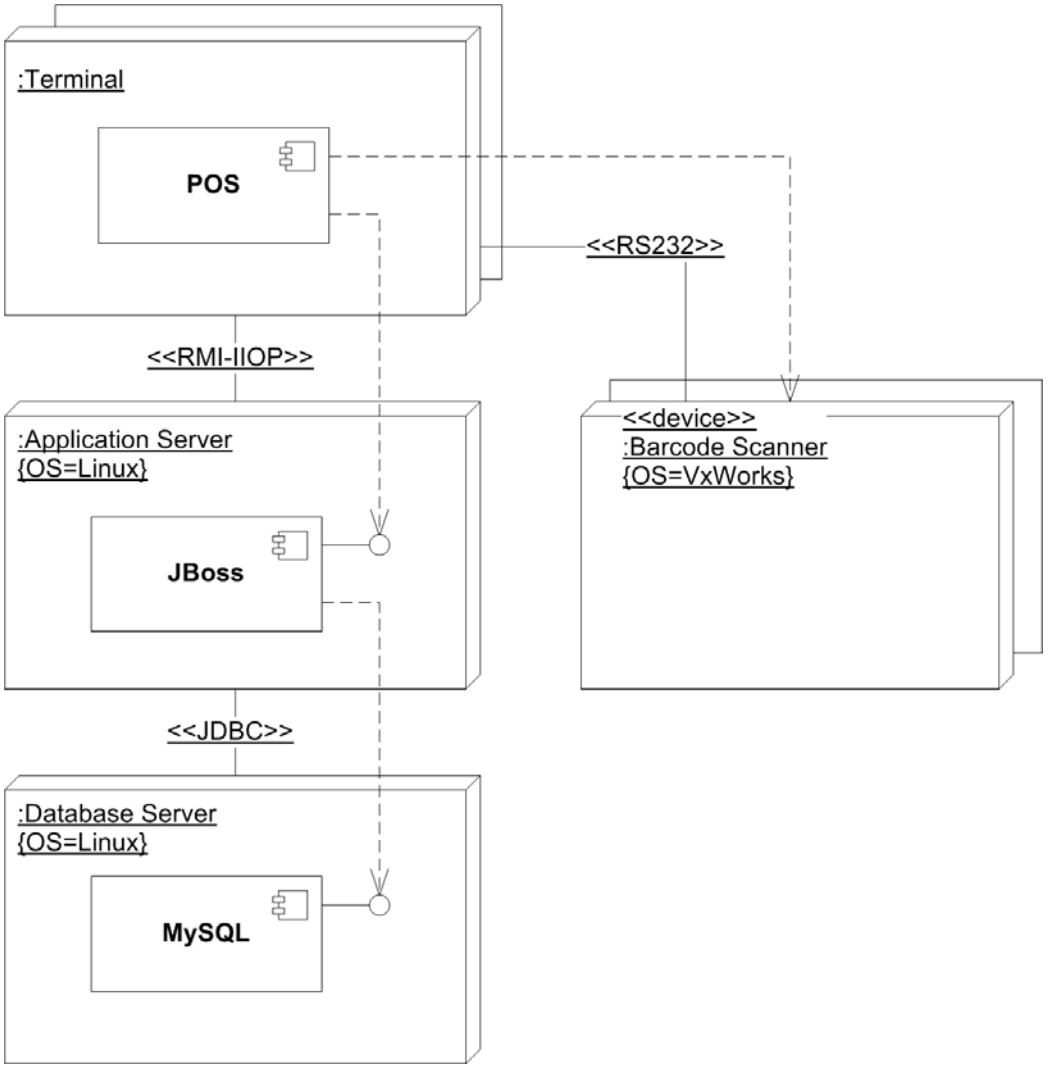
> The software architecture of a computing system is the structures of the system, which comprise software elements, the externally visible properties of those elements, and the relationships among them [Bass et al., 2003]

Basic Allocation Elements



(Note: *UML* components are used here)
[Ambler: *Agile modelling]*

POS Example: Allocation View



Formal Description Languages

Architecture-specific [Medvidovic & Taylor, 2000]

ADL	ACME	Aesop	C2	Darwin	MetaH	Rapide	SADL	UniCon	Weaves	Wright
Focus	Architectural interchange, predominantly at the structural level	Specification of architectures in specific styles	Architectures of highly-distributed, evolvable, and dynamic systems	Architectures of highly-distributed systems whose dynamism is guided by strict formal underpinnings	Architectures in the guidance, navigation, and control (GN&C) domain	Modeling and simulation of the dynamic behavior described by an architecture	Formal refinement of architectures across levels of detail	Glue code generation for interconnecting existing components using common interaction protocols	Data-flow architectures, characterized by high-volume of data and real-time requirements on its processing	Modeling and analysis (specifically, deadlock analysis) of the dynamic behavior of concurrent systems

Other

- E.g., rate-monotonic analysis on architectural components
- E.g., statechart-based analyses

Discussion

Component & Connector viewpoint does not map well to implementation?

- Few languages have interaction as first-class construct
- Neither to the UML – bound tightly to OO implementation
- On the other hand central in many approaches to architectural design

UML is not designed specifically for software architecture description?

- Many (irrelevant) modelling constructs
- But very widely used and supported

Not precise enough for formal analysis (?)

Take a look at the structures of a system architecture structures of a system.pdf