Software Architecture - What/Why/How

Acknowledgement: Some of the material in these slides is taken from *Software Architecture in Practice, 3rd edition* by Bass, Clements and Kazman.

Outline

- Definitions
- Reference Models and Architectures
- Consequences of Architectural Choice
- Promoting Reuse
- Architectural Structures
- Architecture Business Cycle
- Architecture Process advice
- Rules of Thumb for "Good" Architecture

Definitions of Software Architecture (1/2)

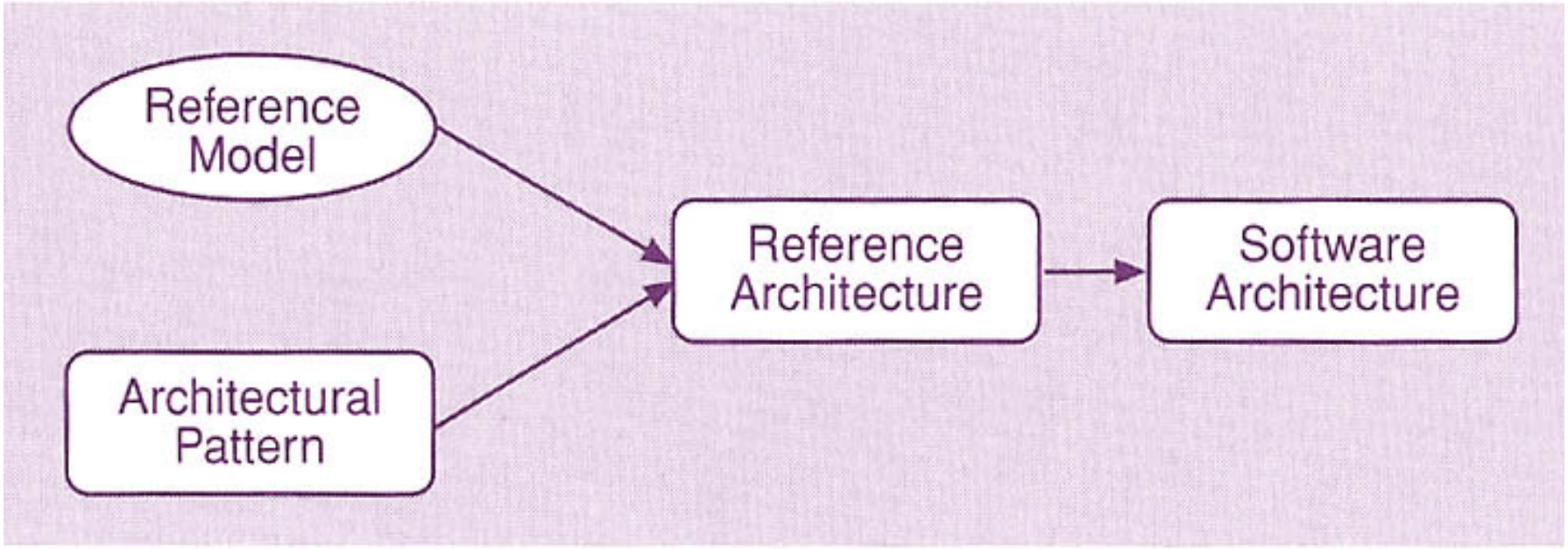
- Many different definitions of software architecture
- See: **http://www.sei.cmu.edu/architecture/definitions.html**

Definitions of Software Architecture (2/2)

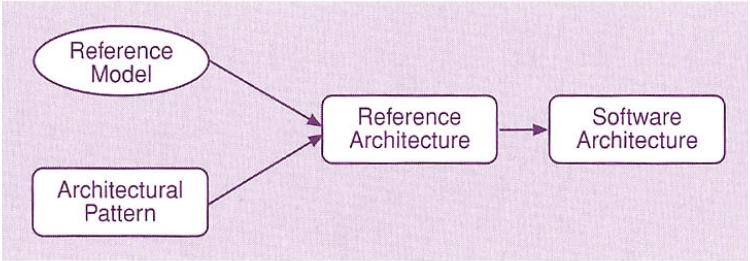
From Bass et al.:

"The software architecture of a program or computing system is the structure or structures of the system, which comprise software elements, the externally visible properties of those elements, and the relationships between them."

Some Related Terms

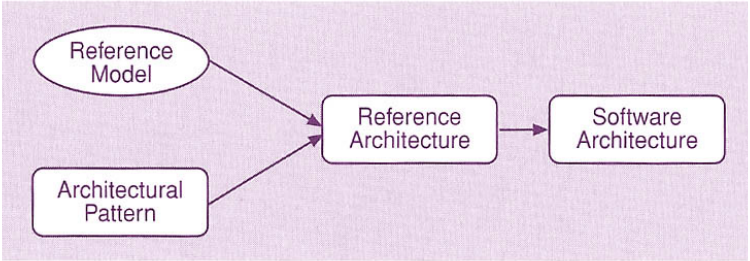


Architectural Pattern



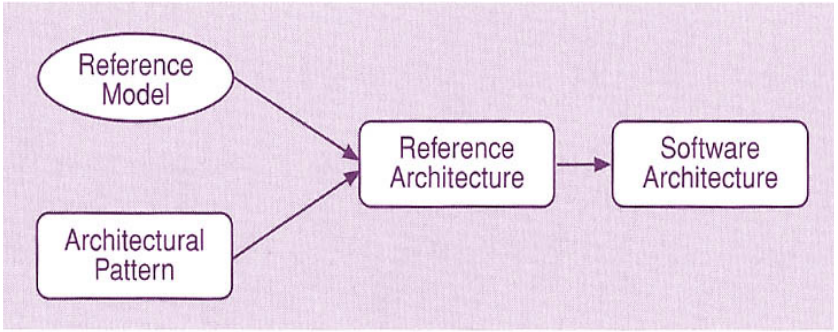
- Also known as "architectural style"
- Description of element and relation types with a set of constraints on how they may be used
- Examples:
  - Pipe and Filter
  - Client-Server
  - Blackboard

Reference Model



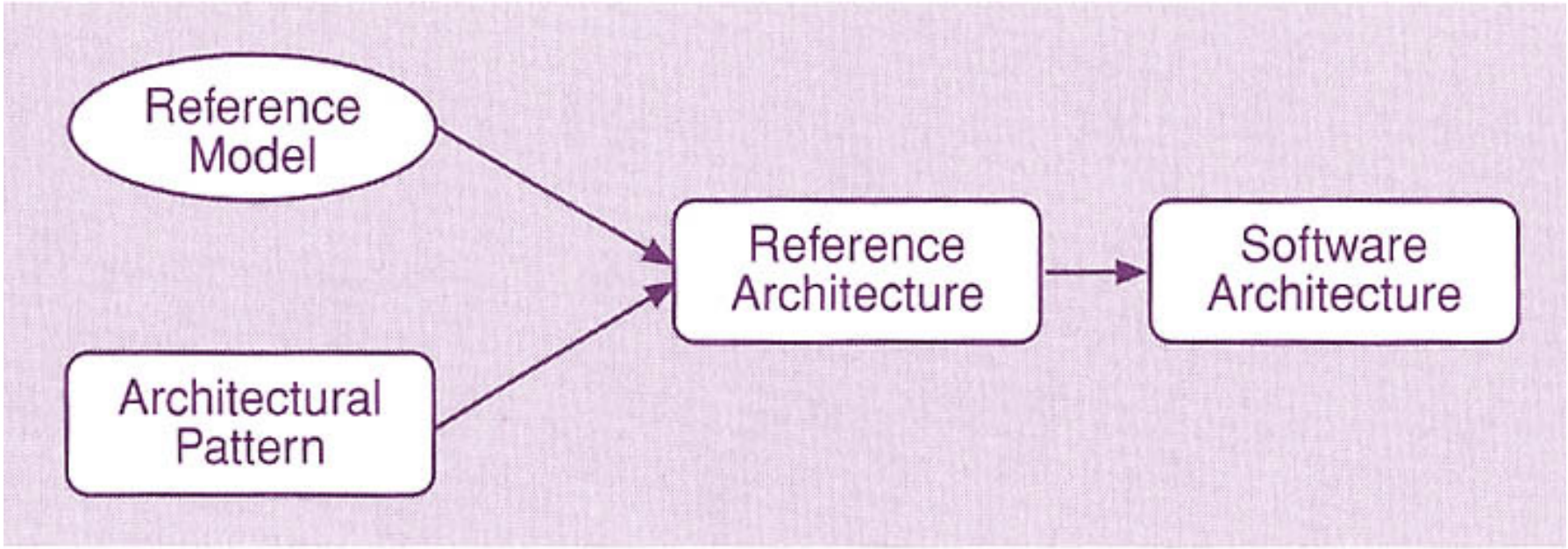
- Division of functionality with data flow between pieces
- Example:
  - Compiler reference model includes a description of parts and data flow between them

Reference Architecture



- Reference model mapped onto software elements and data flows between them

Putting Them All Together



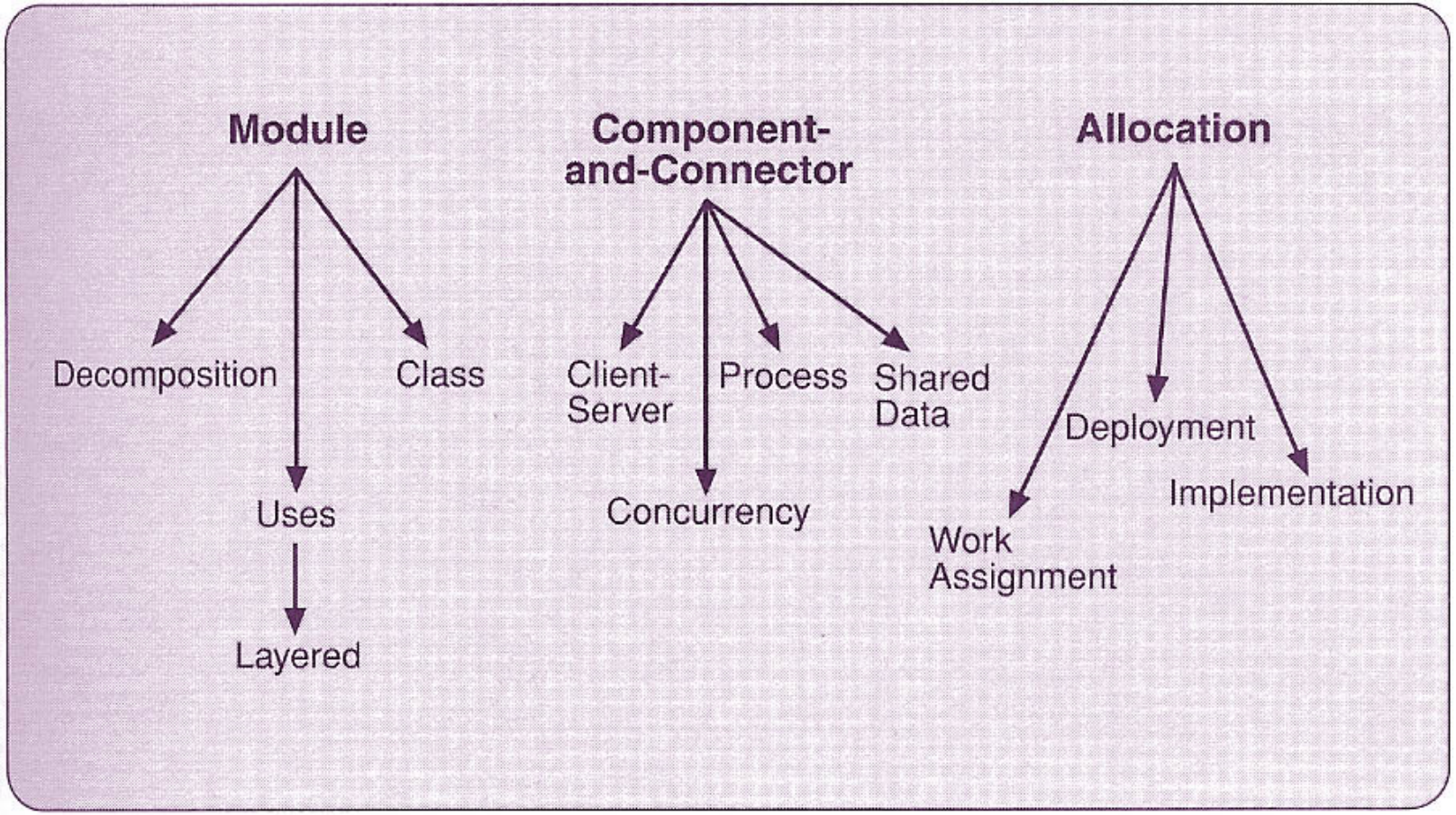
Importance of Software Architecture

- Communication among stakeholders
- Early design decisions (architectural choices)
- Transferable abstraction of a system (promotes reuse)

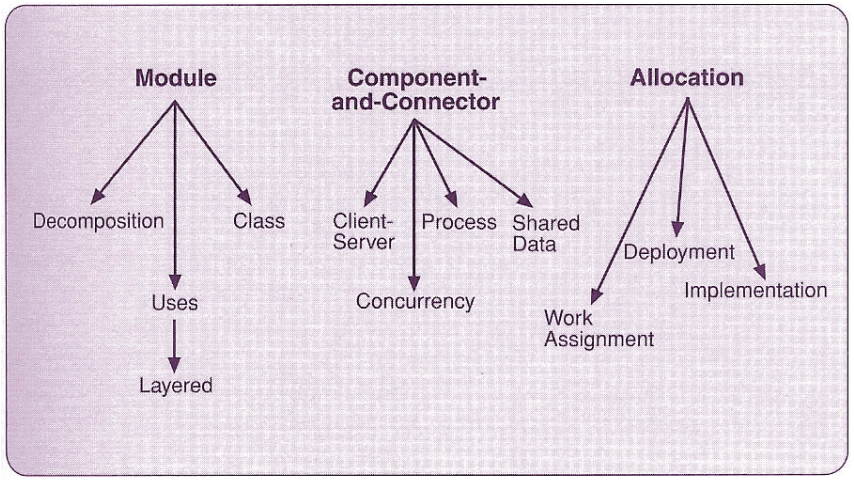
Architecture Promotes Reuse

- Product lines share a common architecture
- Externally-developed elements may be included
- Restrictions encourage reuse of design patterns
- Architecture can be basis for training

Architectural Structures

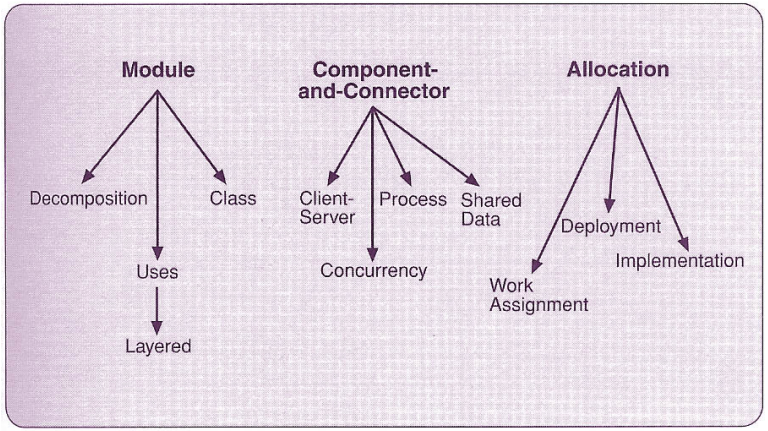


Module Structures



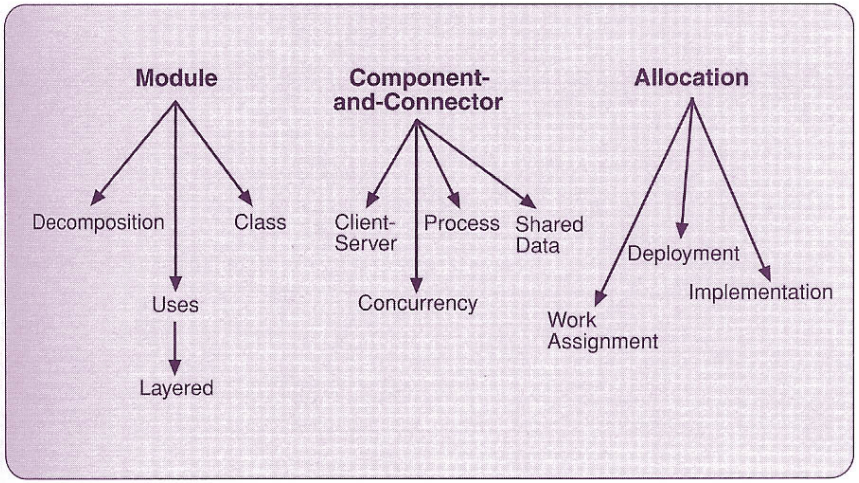
- Decomposition - contains
- Uses - calls
- Layered - controlled access
- Class - inheritance

Component-and-Connector Structures



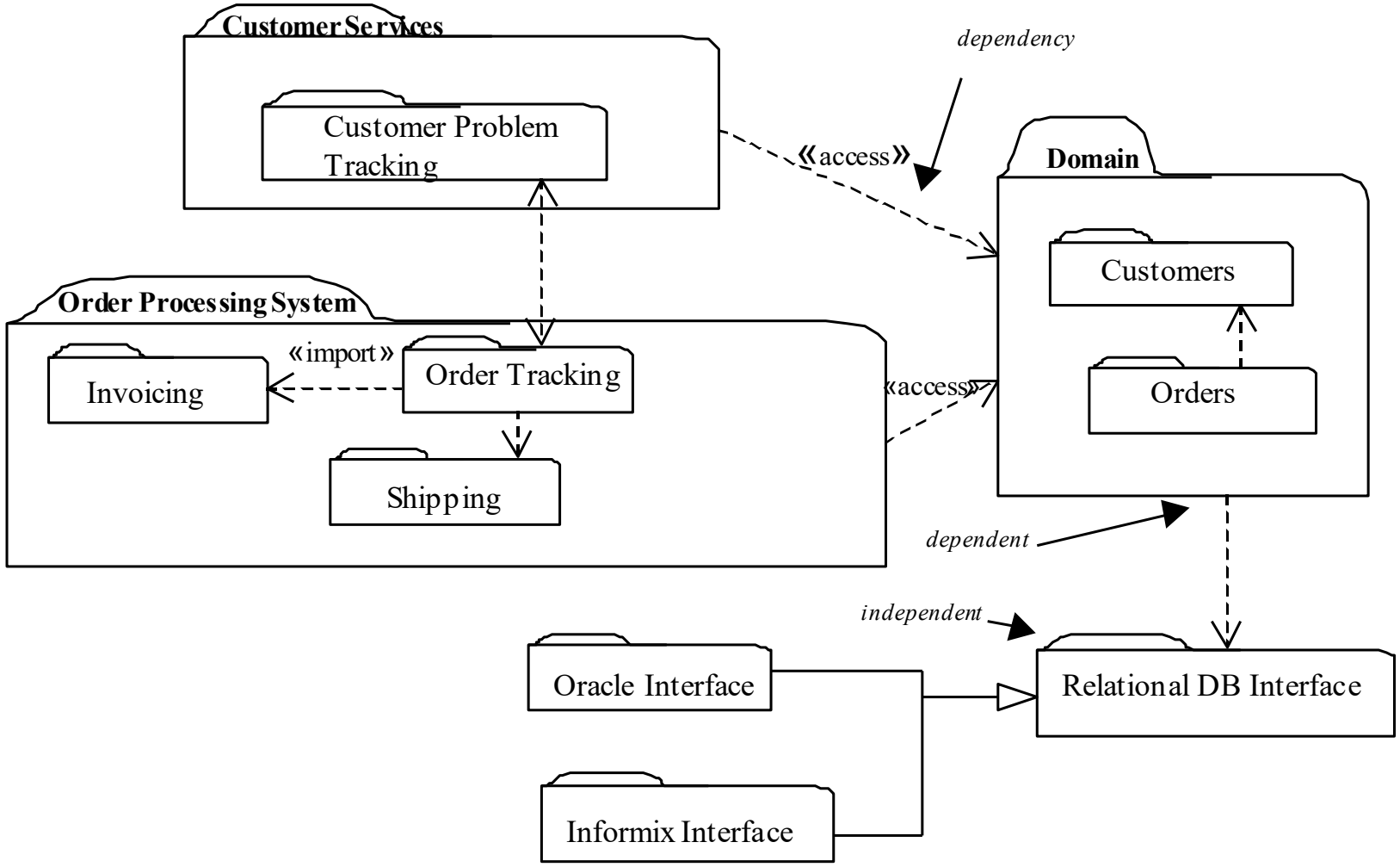
- Client-Server
- Concurrency - parallel execution
- Process - synchronization
- Repository

Allocation Structures



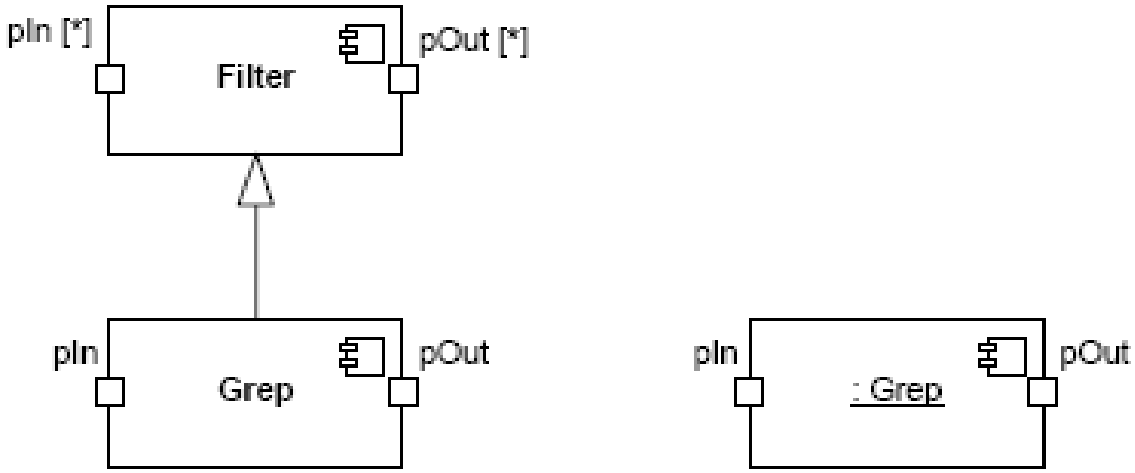
- Work assignment - who does what
- Deployment - allocation to hardware
- Implementation - mapping to files

Example – Module view



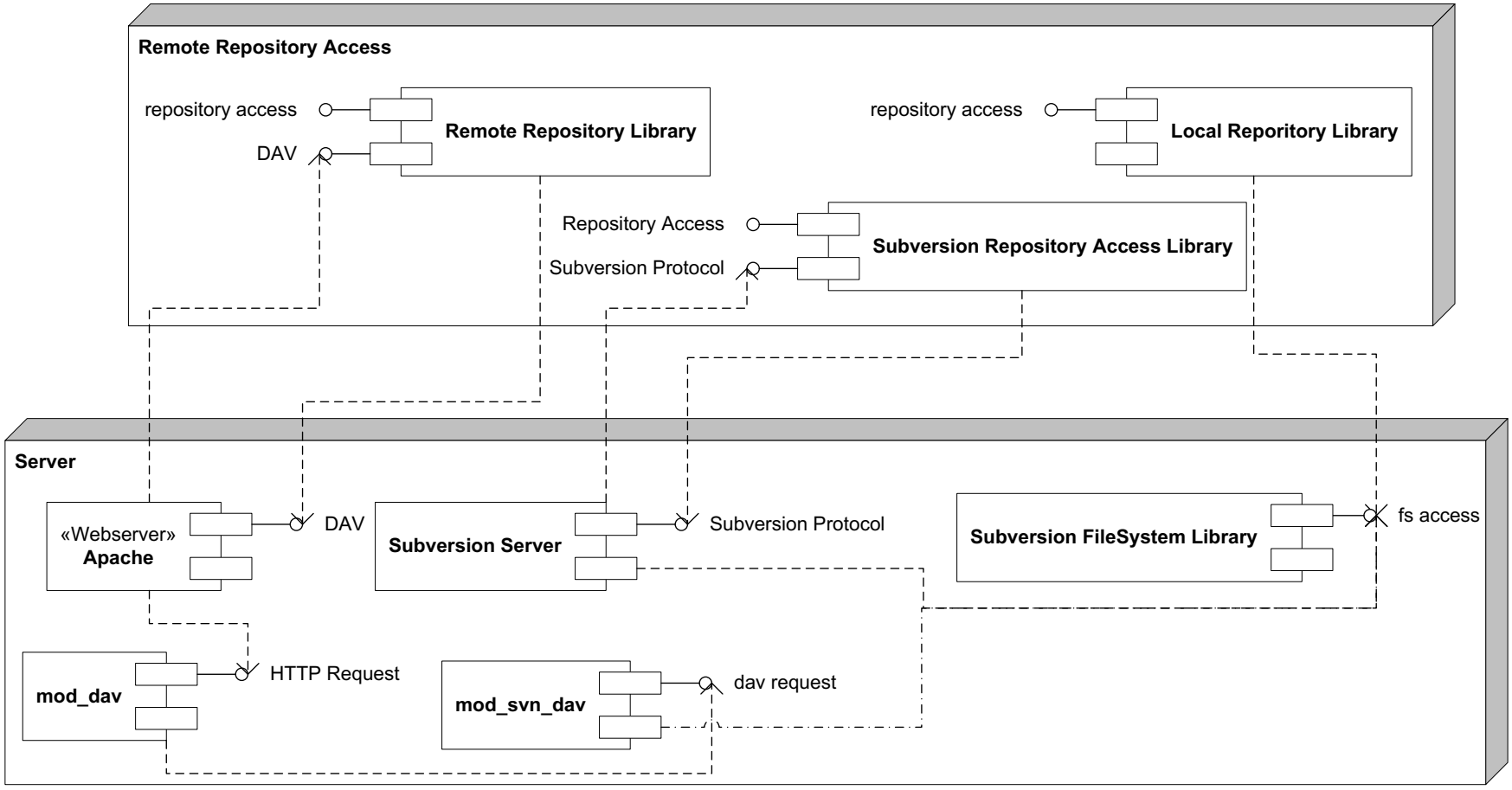
Component and Connector View

Components

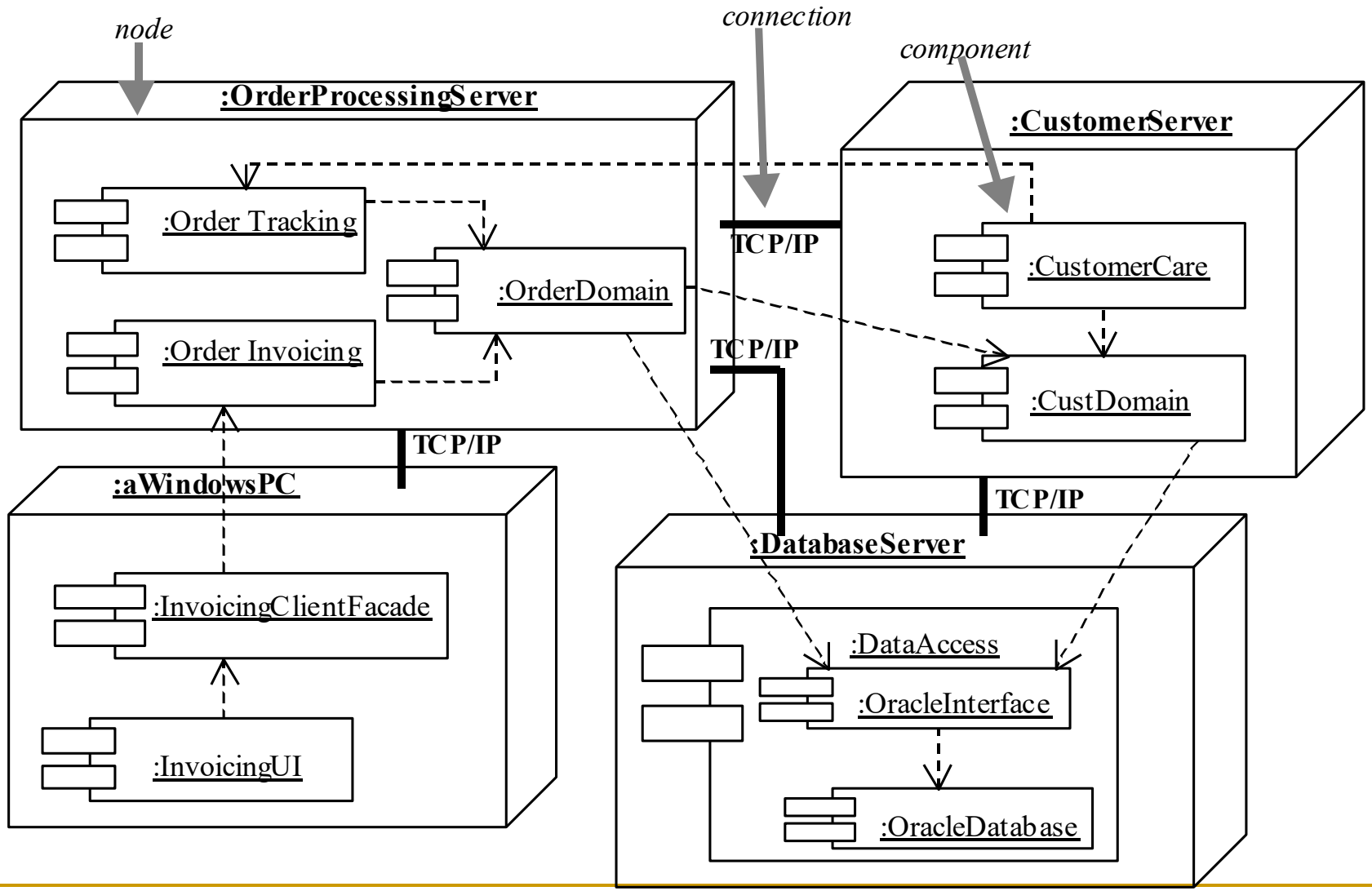


C&C Types as UML Component Types and C&C Instances as UML Component Instances

Component and Connector – Client Server View



Example Allocation view (Deployment)



Architecture Business Cycle

Architecture Business Cycle- Stakeholders

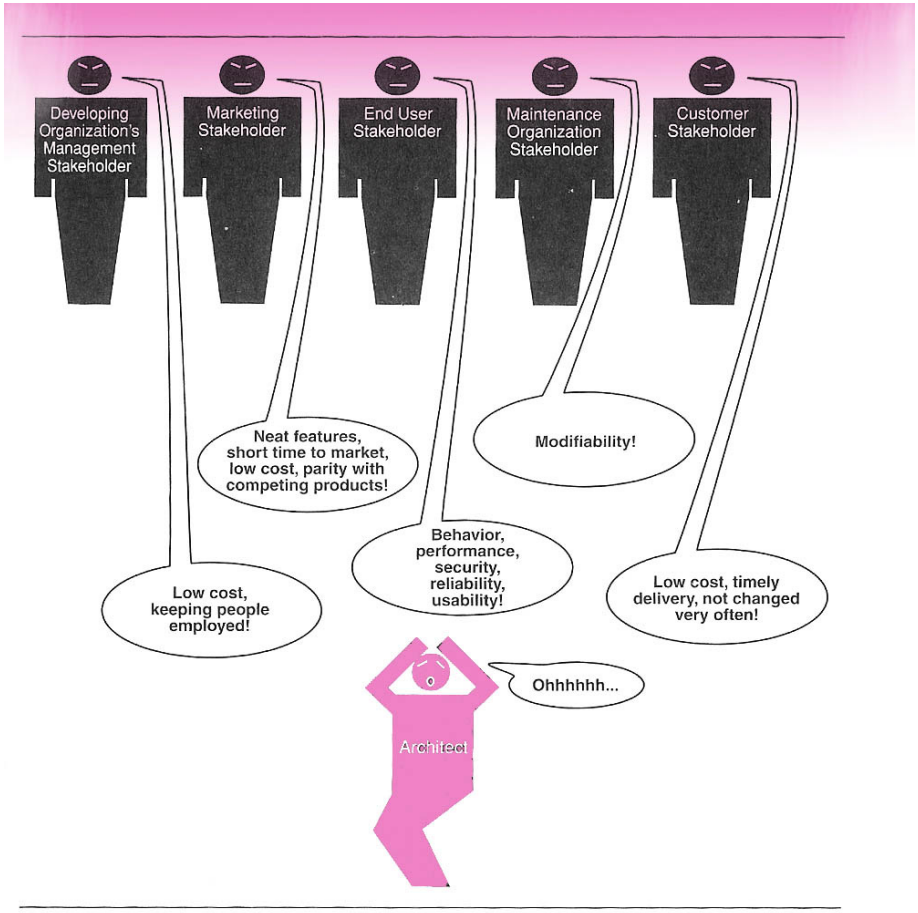
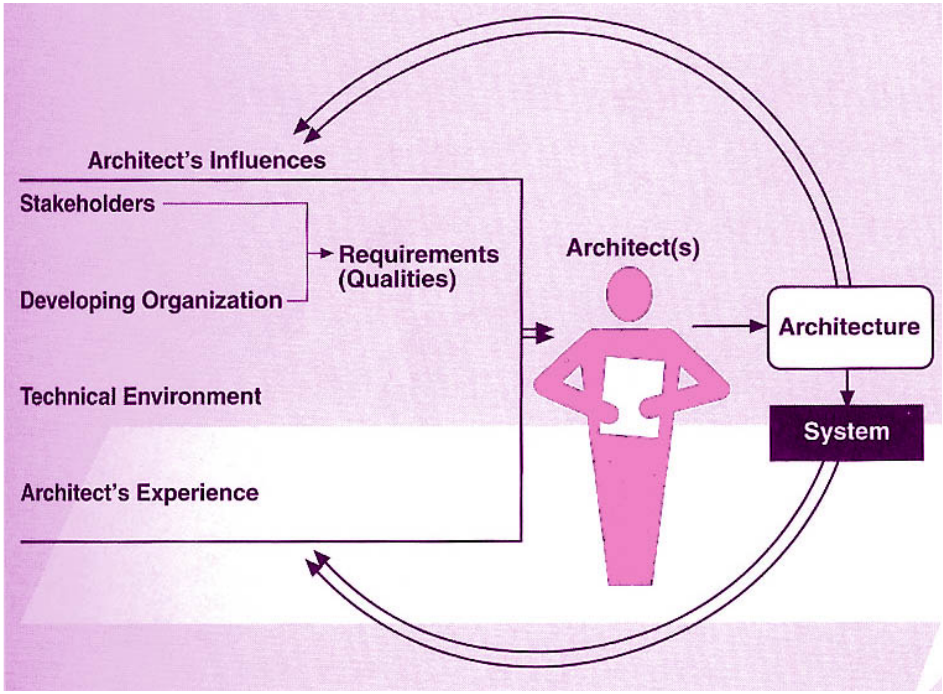


FIGURE 1.2 Influence of stakeholders on the architect

- Developing Organization Management
- Marketing
- End User
- Maintainenace Organization
- Customer

Architecture Business Cycle

- Stakeholders
- Developing Organization
- Technical Environment
- Architect's Experience



Architecture Activities

- Creating business case
- Understanding requirements
- Creating or selecting architecture
- Documenting and communicating architecture
- Analyzing architecture
- Implementing system
- Ensuring conformance of implementation

Architecture Process Advice (1/2)

1. Architecture should be product of a single architect of small group with identified leader
2. Architect should have functional requirements and a prioritized list of quality attributes
3. Architecture should be well-documented with at least one static and one dynamic view

Architecture Process Advice (2/2)

4. Architecture should be circulated to stakeholders, who are active in review
5. Architecture should be analyzed (quantitatively and quality) before it is too late.
6. System should be developed incrementally from an initial skeleton that includes major communication paths
7. Architecture should result in a small number of specific resource contention areas

"Good" Architecture Rules of Thumb (1/2)

1. Use information hiding to hide computing infrastructure
2. Each module should protect its secrets with a good interface
3. Use well-known architecture tactics to achieve quality attributes
4. Minimize and isolate dependence on a particular version of a commercial product or tool.

"Good" Architecture Rules of Thumb (2/2)

4. Separate producer modules from consumer modules.
5. For parallel-processing, use well-defined processes or tasks.
7. Assignment of tasks or processes to processors should be easily changeable (even at runtime)
8. Use a small number of simple interaction patterns