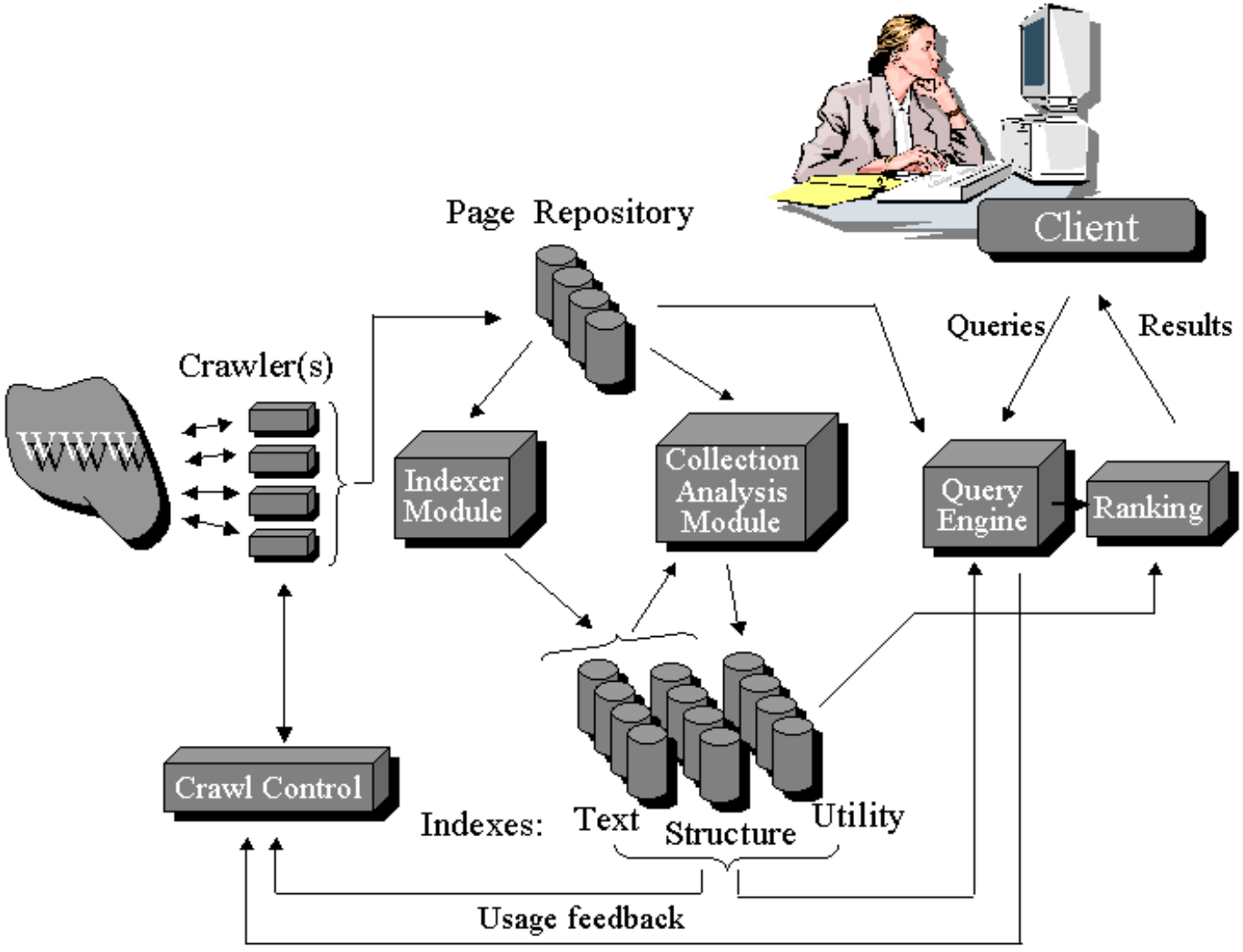
SE 211
Software Specification and Design II

Principles of Software Architecture

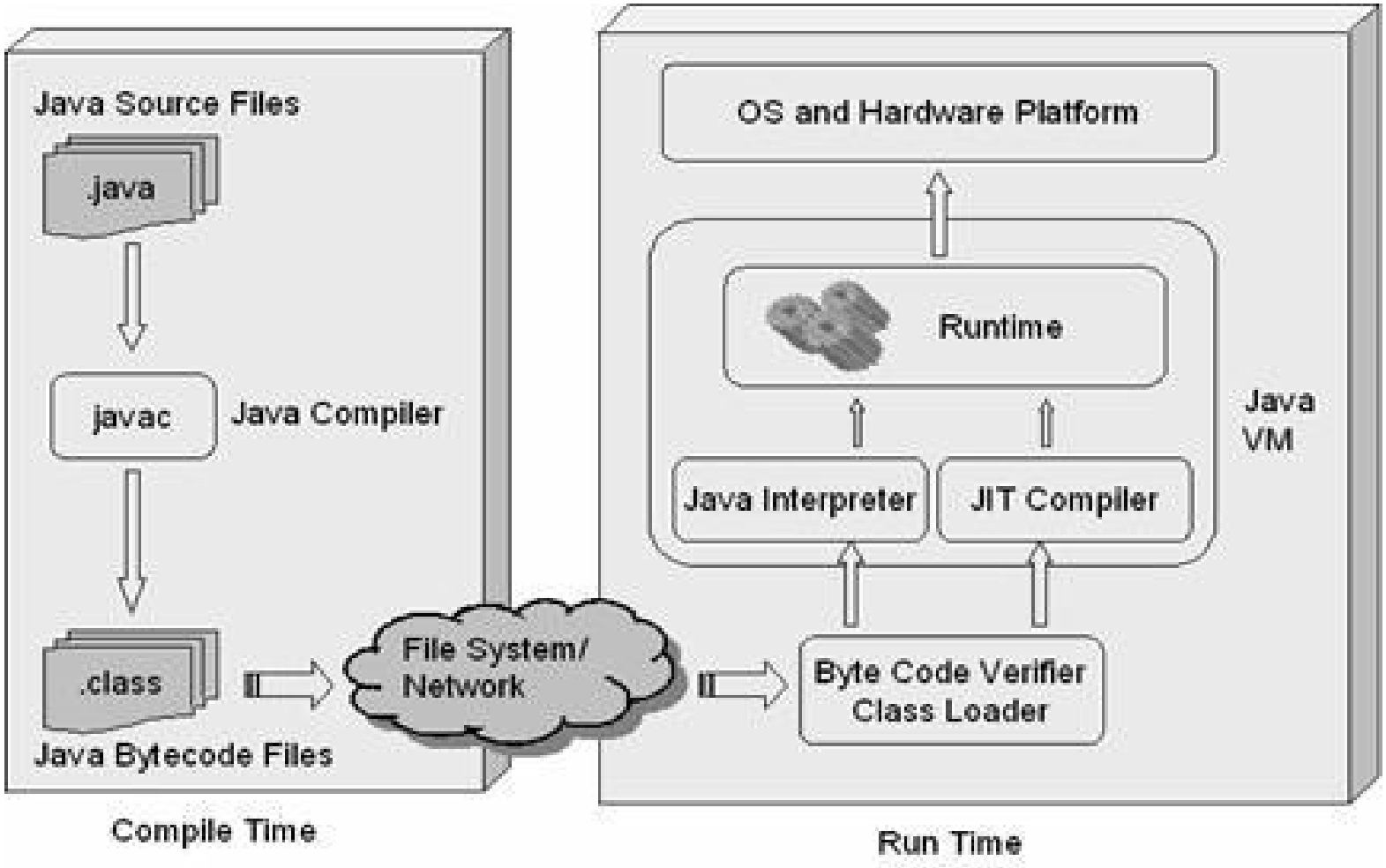
Software Design

- Software Architectural Design
  - The top-level structure and organization of the system is described and the various components are identified.
- Software Detailed Design
  - Each component is sufficiently designed to allow for its coding.

Architectural Design Sample Diagram



Java Architecture



Software Architecture's Elements

- A software system's architecture should be a composition and interplay of different elements
  - Processing
  - Data, also referred as information or state
  - Interaction

Components

- Elements that encapsulate processing and data in a system's architecture are referred to as software components
- A software component is an architectural entity that:
  - Encapsulates a subset of the system's functionality and/or data.
  - Restricts access to that subset via an explicitly defined interface.
  - Has explicitly defined dependencies on its required execution context.
- Components typically provide application-specific services.

Connectors

- A software connector is an architectural building block tasked with effecting and regulating interactions among components.
- In complex systems interaction may become more important and challenging than the functionality of the individual components.
- In many software systems connectors are usually simple procedure calls or shared data accesses.
  - Much more sophisticated and complex connectors are possible!
- Connectors typically provide application-independent interaction facilities.

Examples of Connectors

- Procedure call connectors
- Shared memory connectors
- Message passing connectors
- Streaming connectors

Architecture Defines Structure

- Decomposition of system in to components/modules/subsystems
- Architecture defines:
  - Component responsibilities
    - Precisely what a component will do when you ask it
  - Component interfaces
    - What a component can do
  - Component communications and dependencies

Architecture Specifies Component Communication

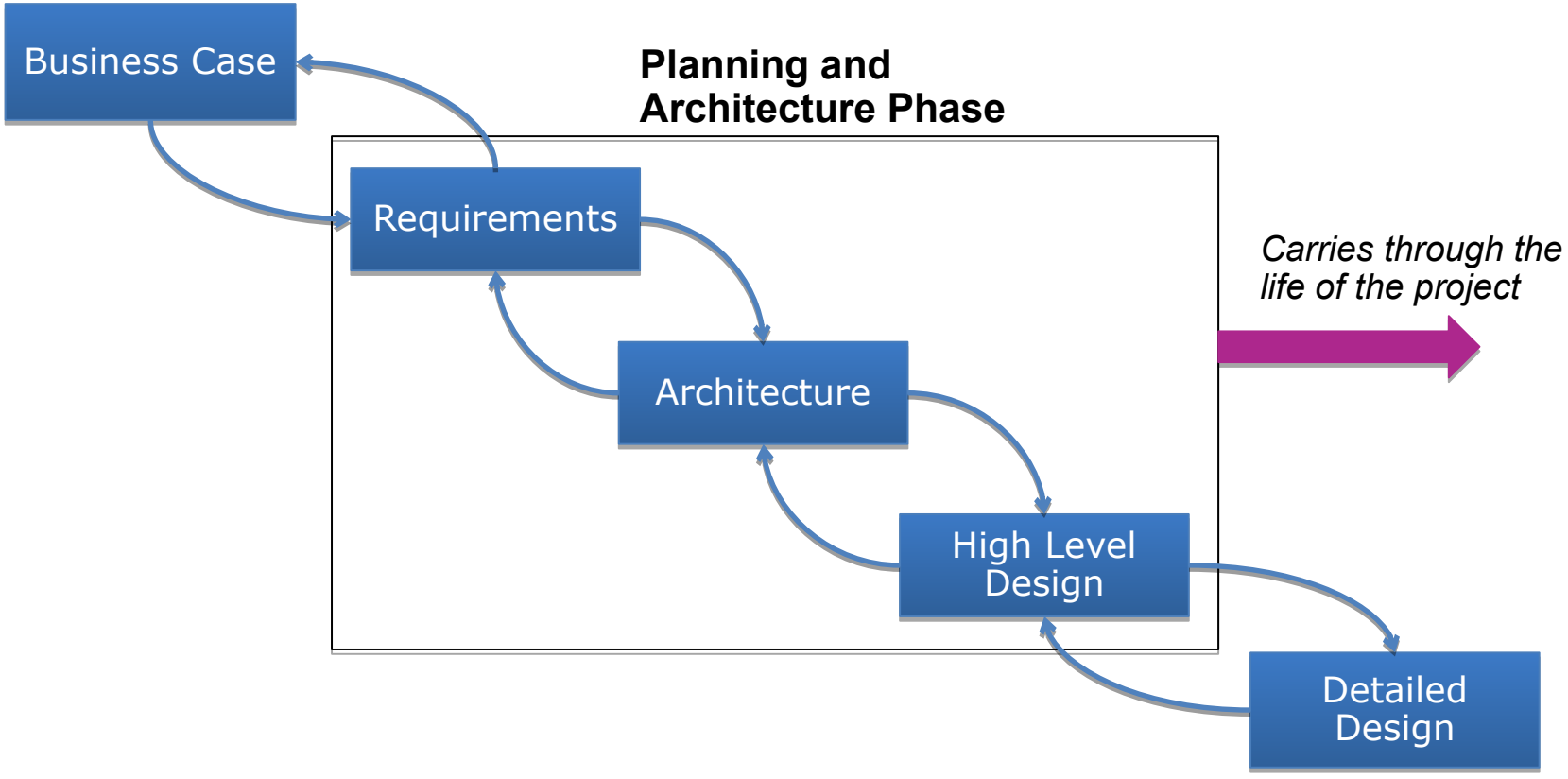
- Communication involves:
  - Data passing mechanisms, e.g.:
    - Remote method invocation
    - Asynchronous message
  - Control flow
    - Sequential
    - Concurrent/parallel

Who Impacts the Architecture?

Architects must:

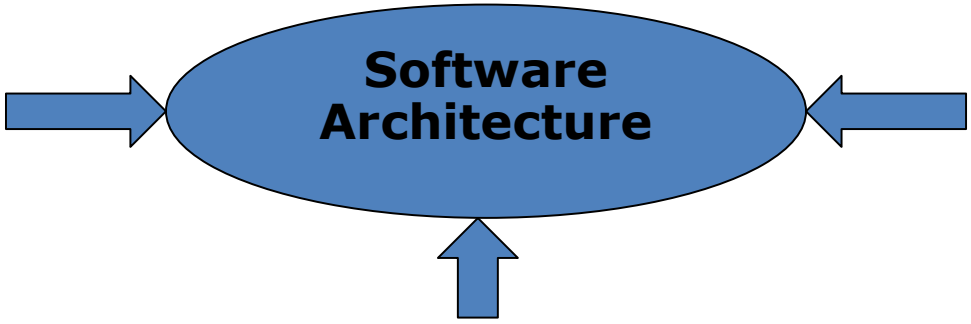
- Work with various application stakeholders:
  Architects play a pivotal liaison role by making sure all the application's stakeholder needs are understood and incorporated into the design.
- Work with the requirements team:
  The architect plays an important role in requirements gathering by understanding the overall systems needs and ensuring that the appropriate quality attributes are explicit and understood.
- Work with the project management:
  Planning, estimates, budgets, schedules.
- Lead the technical design team:
  Defining the application architecture is a design activity.
- Ensure faithful implementation.

Where Does Architecture Fit In?



Requirements That Shape Architecture

Functional Requirements
what the system must do



Quality Attributes
performance, security, modifiability, …

Constraints
decisions made externally (e.g., which OS, which devices, etc.)

Example

- A typical architecture requirement :
  - "*Communications between components must be guaranteed to succeed with no message loss*"

Constraints

Constraint	Architecture Requirement
Business	The technology must run as a plug-in for MS BizTalk, as we want to sell this to Microsoft.
Development	The system must be written in Java so that we can use existing development staff.
Schedule	The first version of this product must be delivered within six months.

- Constraints impose restrictions on the architecture and are (almost always) non-negotiable.
- They limit the range of design choices an architect can make.

Quality Attributes

- QAs are part of an application's Non-Functional Requirements
  - "how" the system achieves its functional requirements
- There are many QAs
- Architect must decide which are important for a given application
  - Understand implications for application
  - Understand competing requirements

Quality Attributes (cont'd)

- Often know as –*ilities*
  - Reliability
  - Availability
  - Portability
  - Scalability
  - Performance (!)

Quality Attribute Requirements

QA	Architecture Requirement
Performance	Application performance must provide sub-four second response times for 90% of requests.
Scalability	The application must be able to handle a peak load of 500 concurrent users during the enrollment period.
Modifiability	The architecture must support a phased migration from the current Forth Generation Language (4GL) version to a .NET systems technology solution.
Availability	The system must run 24x7x365, with overall availability of 0.99.
Resource Management	The server component must run on a low end office-based server with 512MB memory.
Usability	The user interface component must run in an Internet browser.
Reliability	No message loss is allowed, and all message delivery outcomes must be known with 30 seconds.
Security	All communications must be authenticated and encrypted using certificates.

Architecture Views

- A software architecture represents a complex design artifact
- Many possible ‘views’ of the architecture
  - Cf. with buildings – floor plan, external, electrical, plumbing, air-conditioning

Architecture Views (cont'd)

- Logical view
- Process view
- Development view
- Physical view

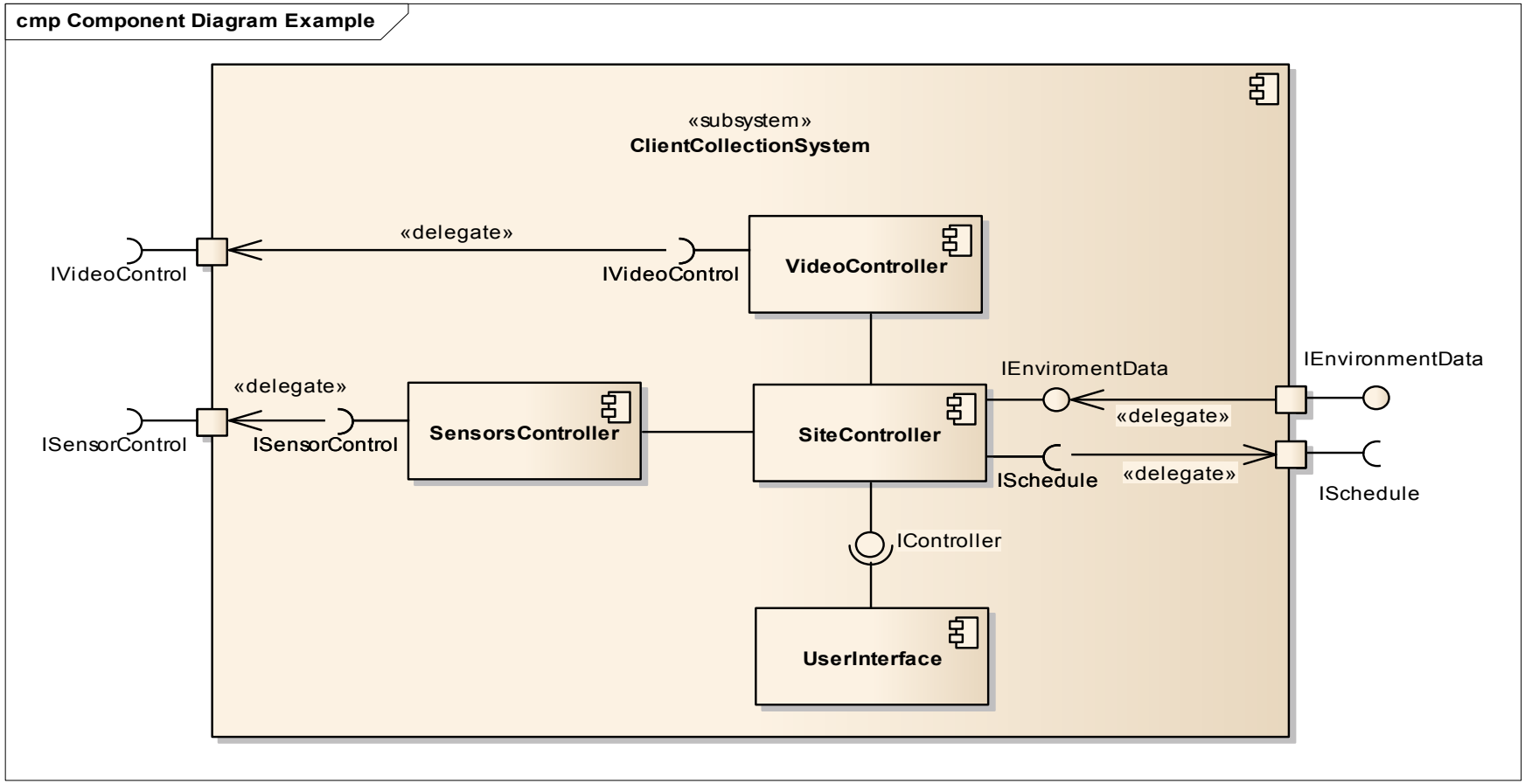
Architecture Views (cont'd)

- *Marketecture*
  - Informal depiction of system's structure and interactions.
  - Portray the design philosophies embodied in the architecture.
- Every system should have a marketecture:
  - Easy to understand.
  - Helps discussion during design, build, review, sales (!) activities.

Designing the Logical Architecture

- The logical view is used to decompose systems into logical components that represent the structural integrity that supports functional and non-functional requirements.
- The logical view can be modeled using:
  - Component diagrams
  - Class diagrams
  - Box-and-line diagrams

Logical Architecture

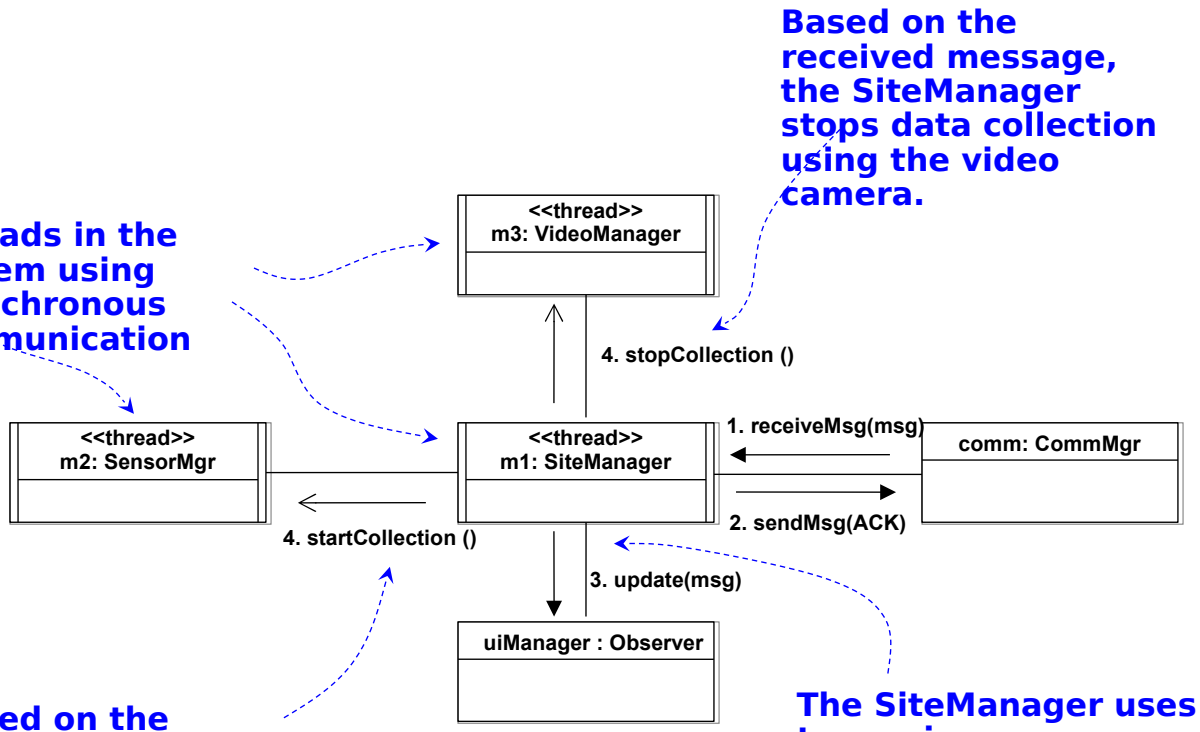


Designing the Process Architecture

- The process view is used to represent the dynamic or behavioral aspects of software systems, where the main units of analysis are processes and threads.
  - With this view, the system is decomposed into processes and threads to address design issues that deal with the dynamic flow of control between architectural elements.

- The process view can be modeled with UML using:
  - Sequence diagrams
  - Communication diagrams

Process Architecture

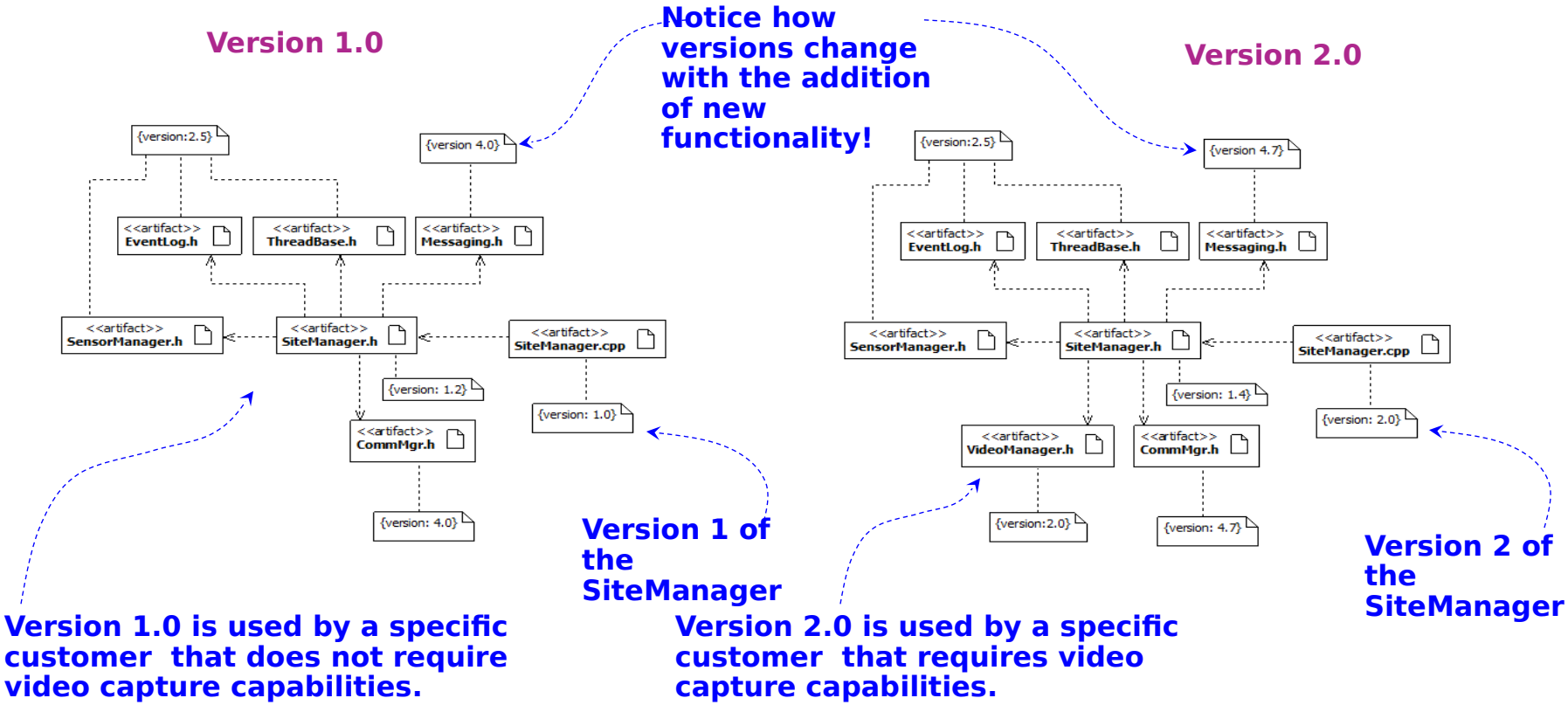


Designing the Development Architecture

- The development view of software systems represents the software development configuration aspects of the software system.
  - The main units of decomposition are actual physical files and directories.

- This view can be used to analyze the system from the perspective of how logical components map to physical files and directories.

Development Architecture

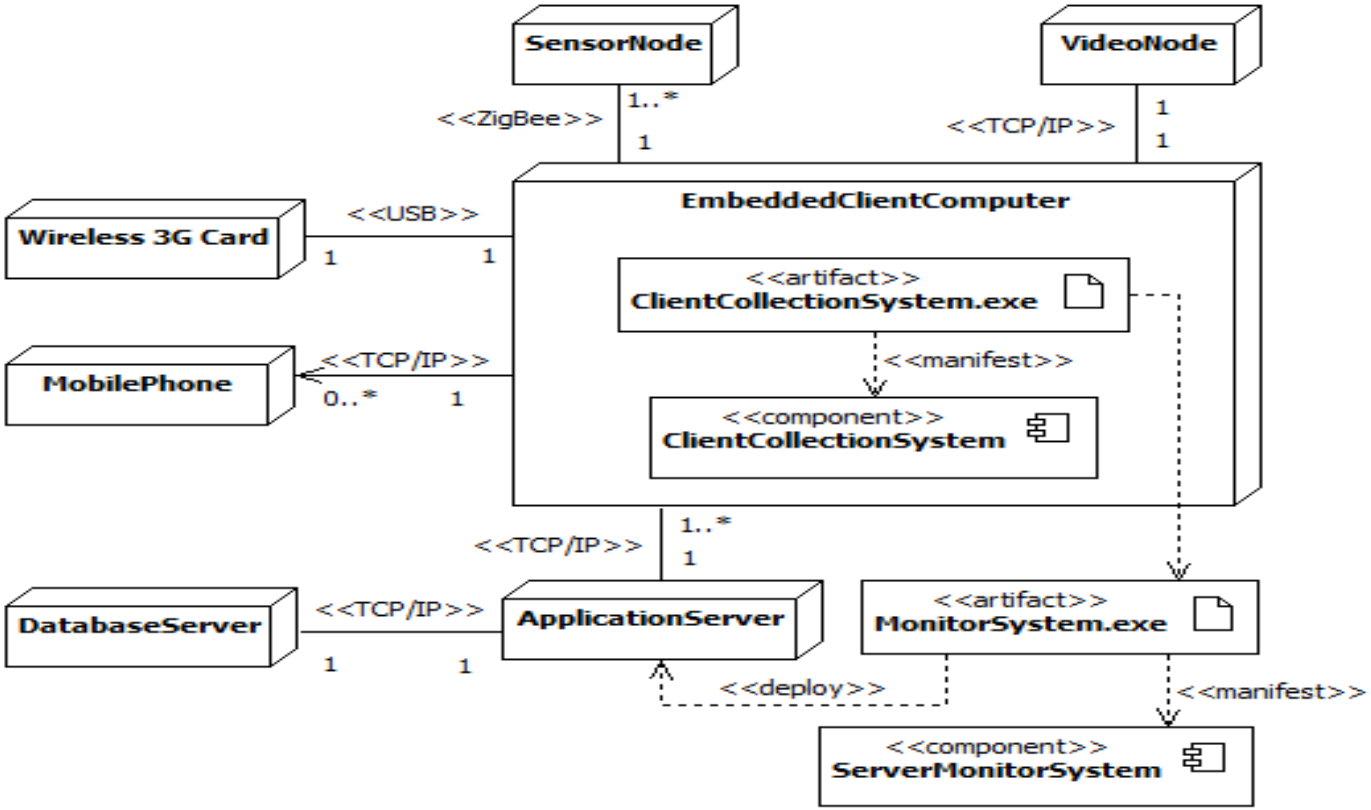


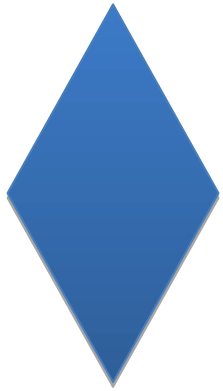
Designing the Physical Architecture

- The physical view represents the deployment aspects of software systems.
  - The main elements of analysis are nodes, connections between nodes, and the mapping of artifacts to these nodes.
  - This view can be used to model the run-time dependencies of the system.

- The physical view can be modeled in UML using:
  - Deployment diagrams

Designing the Physical Architecture



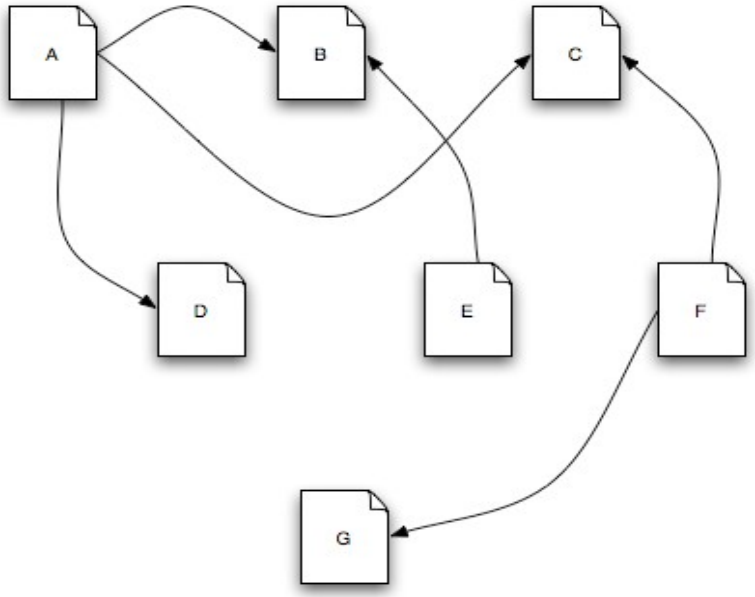


The Power of Architecture

- What is the World Wide Web?
- How is it built?
- How do you explain the Web to a child?
- How do you designing Web-based e-commerce software?

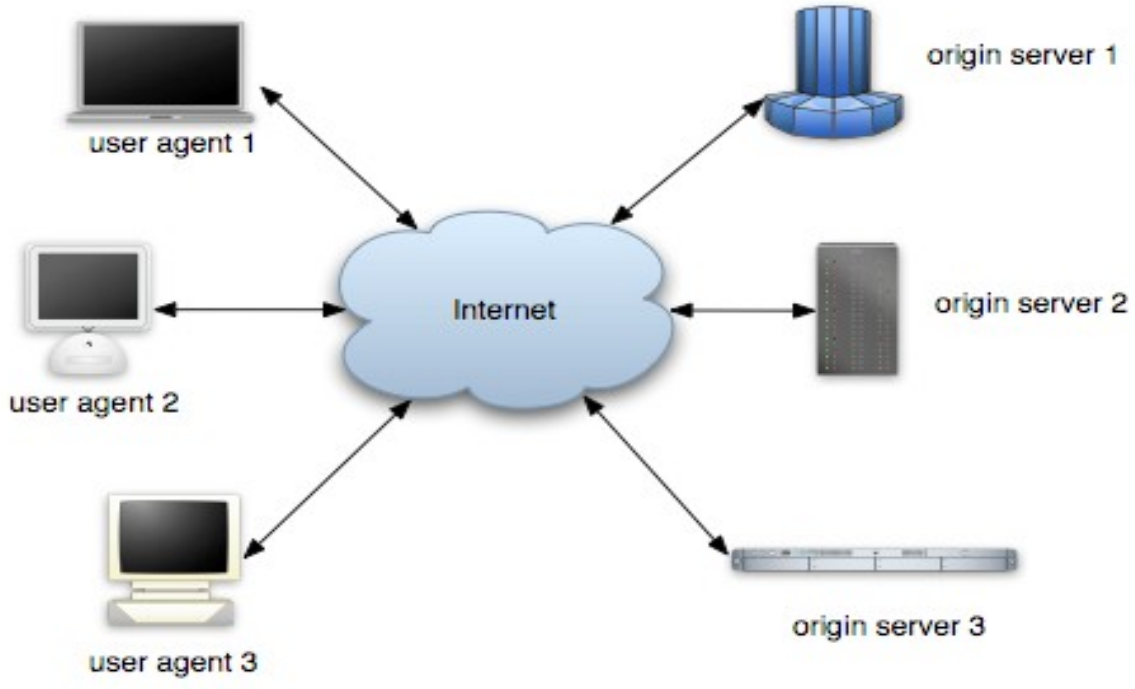
The Architecture of the WWW

- This is the Web – a collection of interrelated pieces of information.



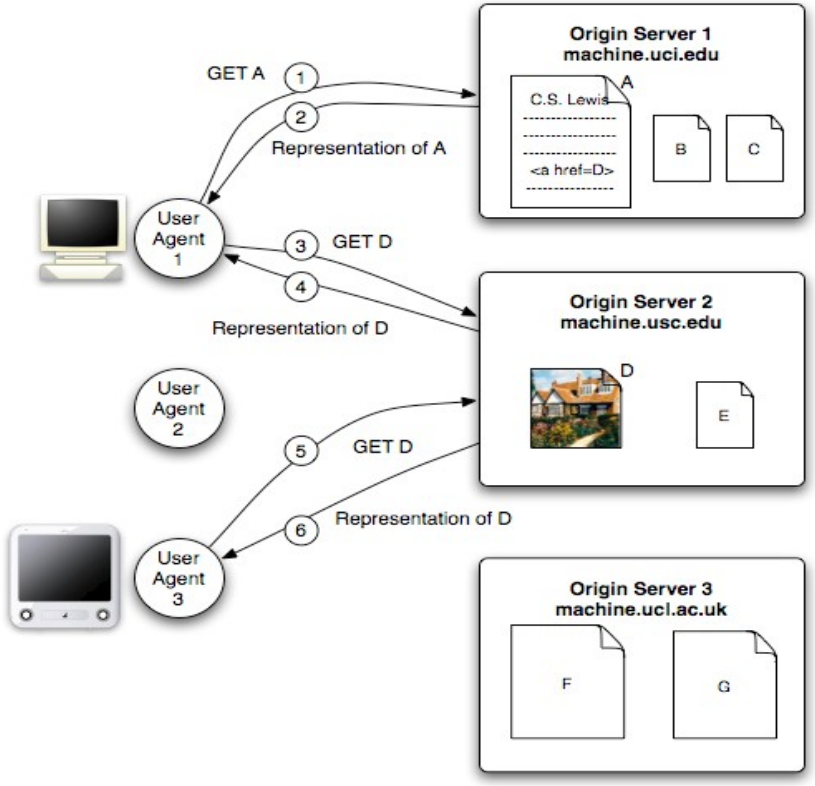
The Architecture of the WWW (cont'd)

- So is this – a collection of computers interconnected through the internet.



The Architecture of the WWW (cont'd)

- And this – the user agents, the origin servers, and the location of the documents.



WWW in a (Big) Nutshell

- The Web is a collection of resources, each of which has a unique name known as a uniform resource locator, or "URL".
- Each resource denotes, informally, some information.
- URLs can be used to determine the identity of a machine on the Internet, known as an origin server, where the value of the resource may be ascertained.
- Communication is initiated by clients, known as user agents, who make requests of servers.
  - Web browsers are common instances of user agents.

WWW in a (Big) Nutshell (cont'd)

- All communication between user agents and origin servers must be performed by a simple, generic protocol (HTTP).
- All communication between user agents and origin servers must be fully self-contained. (So-called "stateless interactions")