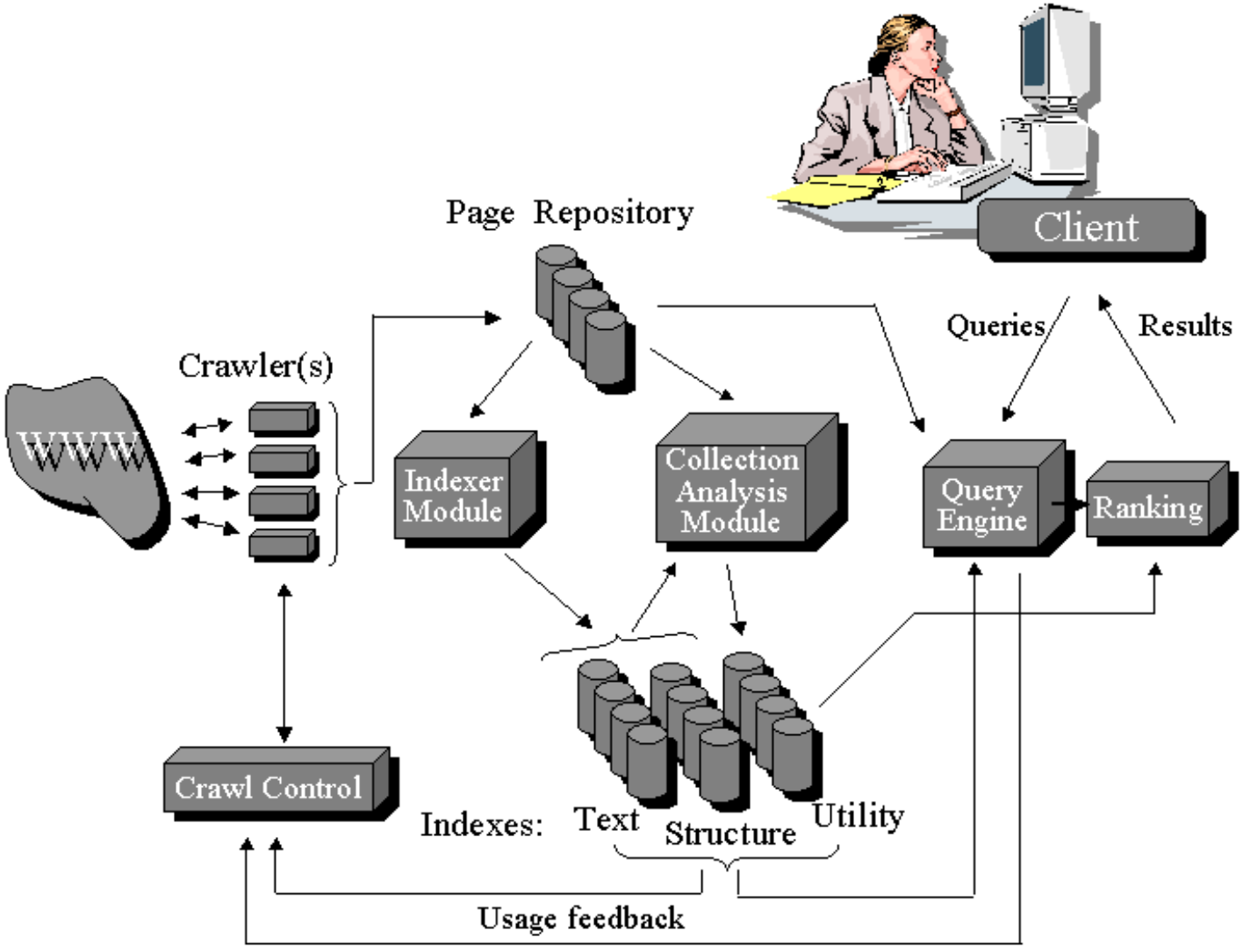
SE 211
Software Specification and Design II

Principles of Software Architecture

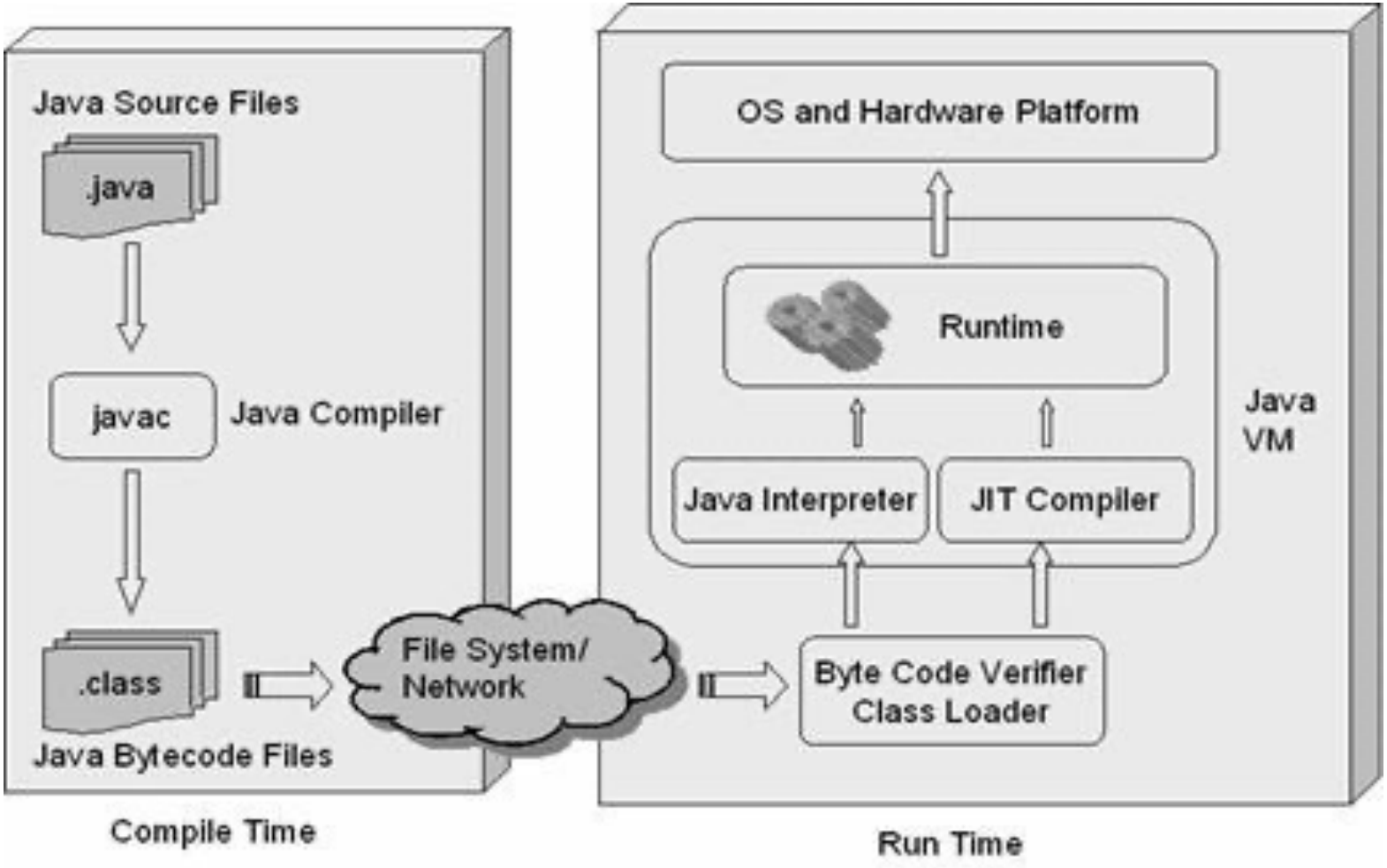
Software Design

- 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

What is Software Architecture?

A software system architecture comprises:

- A collection of software and system components, connections, and constraints.
- A collection of system stakeholders' need statements.
- A rationale which demonstrates that the components, connections, and constraints define a system that, if implemented, would satisfy the collection of system stakeholders' need statements.

Boehm, et al., 1995

What does a Software Architect do?

- Interact with stakeholders to make sure their needs are being met.

What does a Software Architect do?

- Interact with stakeholders to make sure their needs are being met.
- Craft the right architecture to solve the problem at hand.
  - Make sure the right modeling is being done, to know that qualities like performance are going to be met.
  - Make sure that the architecture is not only the right one for operations, but also for deployment and sustainment.

What does a Software Architect do?

- Interact with stakeholders to make sure their needs are being met.
- Craft the right architecture to solve the problem at hand.
  - Make sure the right modeling is being done, to know that qualities like performance are going to be met.
  - Make sure that the architecture is not only the right one for operations, but also for deployment and sustainment.
- Document, and communicate it.

What does a Software Architect do?

- Interact with stakeholders to make sure their needs are being met.
- Craft the right architecture to solve the problem at hand.
  - Make sure the right modeling is being done, to know that qualities like performance are going to be met.
  - Make sure that the architecture is not only the right one for operations, but also for deployment and sustainment.
- Document, and communicate it.
- Give input as needed to issues like tool and environment selection.

What does a Software Architect do?

- Interact with stakeholders to make sure their needs are being met.
- Craft the right architecture to solve the problem at hand.
  - Make sure the right modeling is being done, to know that qualities like performance are going to be met.
  - Make sure that the architecture is not only the right one for operations, but also for deployment and sustainment.
- Document, and communicate it.
- Give input as needed to issues like tool and environment selection.
- Manage risk identification and risk mitigation strategies associated with the architecture.

What does a Software Architect do? (cont'd)

- Make sure everyone is using it and using it correctly.

What does a Software Architect do? (cont'd)

- Make sure everyone is using it and using it correctly.
- Make sure the software and system architectures are in synchronization.

What does a Software Architect do? (cont'd)

- Make sure everyone is using it and using it correctly.
- Make sure the software and system architectures are in synchronization.
- Understand and plan for evolutionary paths.

What does a Software Architect do? (cont'd)

- Make sure everyone is using it and using it correctly.
- Make sure the software and system architectures are in synchronization.
- Understand and plan for evolutionary paths.
- Plan for new technology insertion.

Why is Software Architecture Important?

- Architecture focuses on issues that will be difficult/impossible to change once the system is built.

Architecture Done Poorly …

Get it Right the First Time !

Major Areas of Concern for S/W Architecture

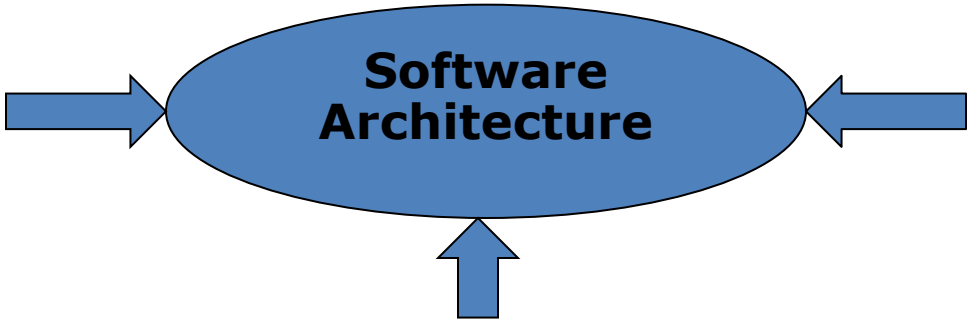
- Requirements
  - Both *functional* and *non-functional*
- OA&M (Operation Administration & Maintenance)
  - The set of procedures for initializing and administering the system.
- Error Recovery
  - How does the system handle *rainy day* scenarios?

Requirements

- Traditional SE suggests requirements analysis should remain unsullied by any consideration for a design.
- However, without reference to existing architectures it becomes difficult to assess practicality, schedules, or costs.
- Requirements analysis and consideration of design must be pursued at the same time.

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.)

Non-Functional Requirements

- Non-Functional Requirements (NFRs) define "how" a system achieves its functional requirements.
- NFRs are also known as "architecture requirements"
  - Must be elicited by architect

- NFRs include:
  - Quality attributes
    - Application performance must provide sub-four second response times for 90% of requests.
  - Technical constraints
    - The system must be written in Java so that we can use existing development staff.
  - Business constraints
    - "We want to work closely with and get more development funding from *MegaHugeTech Corp*, so we need to use their technology in our application."

OA&M

- OA&M includes all the steps necessary to get and keep the system operational.
  - Manufacture the software for delivery and installation
  - Install the software
  - Convert required data
  - Initialize the database
  - Prime the tables
  - Start up the system
  - Connect to the outside world
  - Take it down for backups
  - Restart it after a failure
  - Update the software in the field
  - Etc., Etc., Etc.

OA&M? (cont'd)

- OA&M is one of the most overlooked parts of the architecture.
- Many systems leave this last and assume that the operators in the field will do what they have to do to keep the system operational.
- Some view it as just the set of instructions that the system administrator is given, but this is just the tip of the iceberg.
- OA&M has to be included in the original architecture because it is difficult (if not impossible) to add later.

Error Recovery

- The system has to be designed from the standpoint of errors.
- The sunny day paths are easy; the error handling is what complicates the system.
- Must have a consistent approach for handling errors.
- Error recovery (like OA&M) is often overlooked in the system architecture.

Error Recovery: Kinds Of Errors

- Communication
  - Intermittent / interrupted communication
  - Unexpected / incorrect network changes
- Process
  - Abnormal termination
  - Asleep waiting (*patiently*) for an event (which may never occur, or that has been missed).
  - Resource leaks
- Coordination
  - Someone forgot to tell the up/downstream systems about the new data format.
  - Other systems in the network do not react kindly to a system going down due to a failure or for a scheduled upgrade

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.

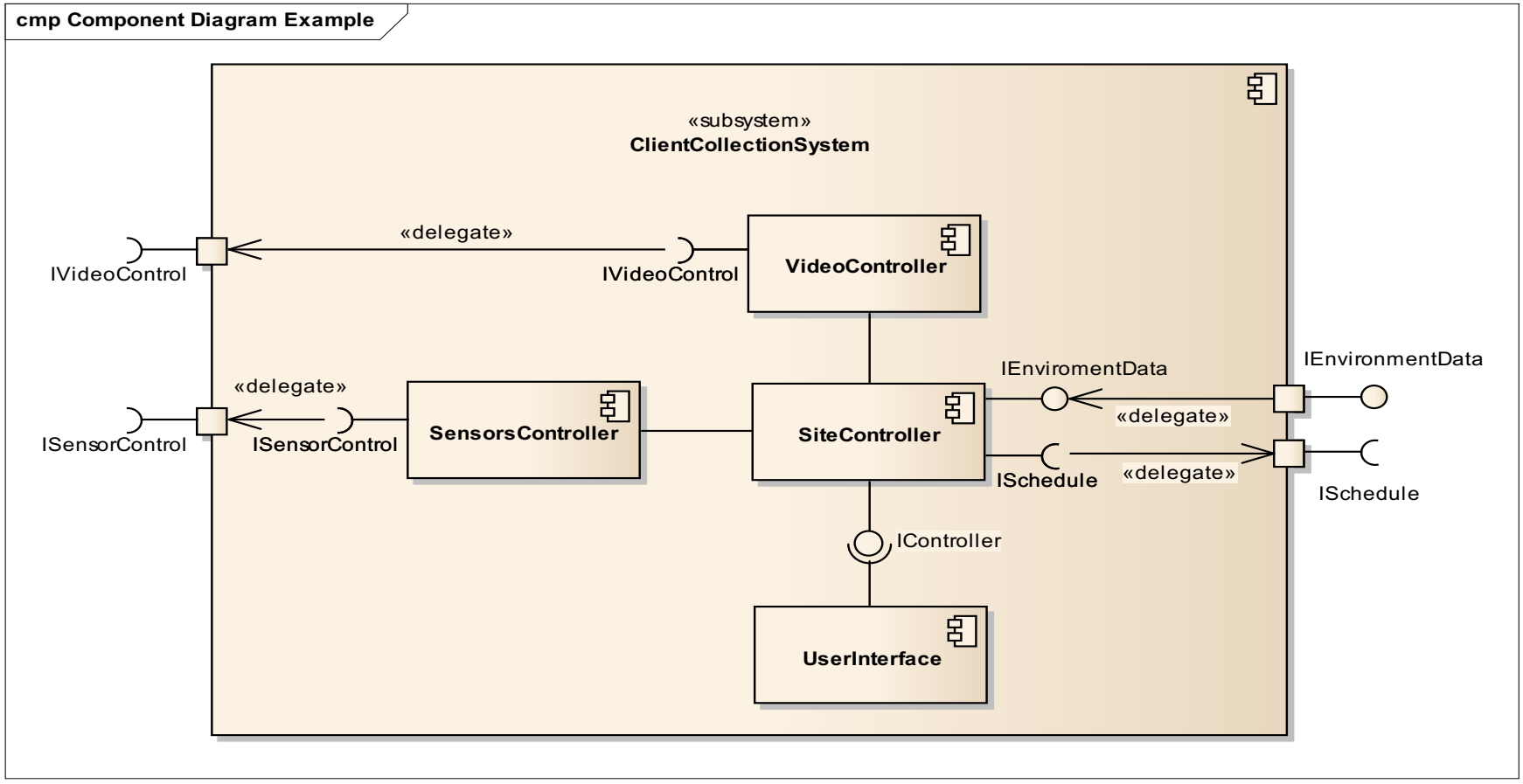
Logical View of the Architecture

- 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:

Logical View of the 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 View of the Architecture



Process View of the 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.

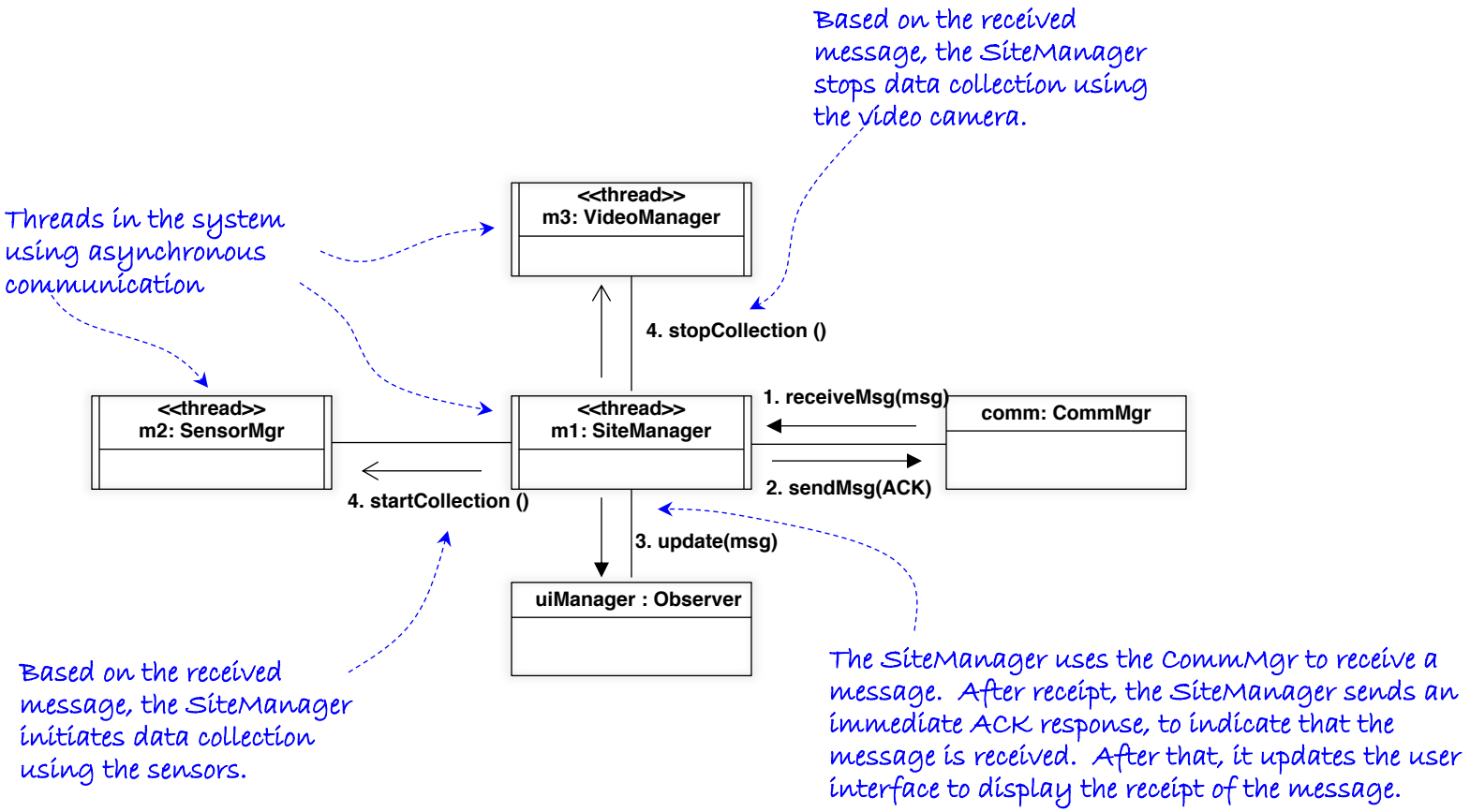
- The process view can be modeled with UML using:

Process View of the 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 View of the Architecture

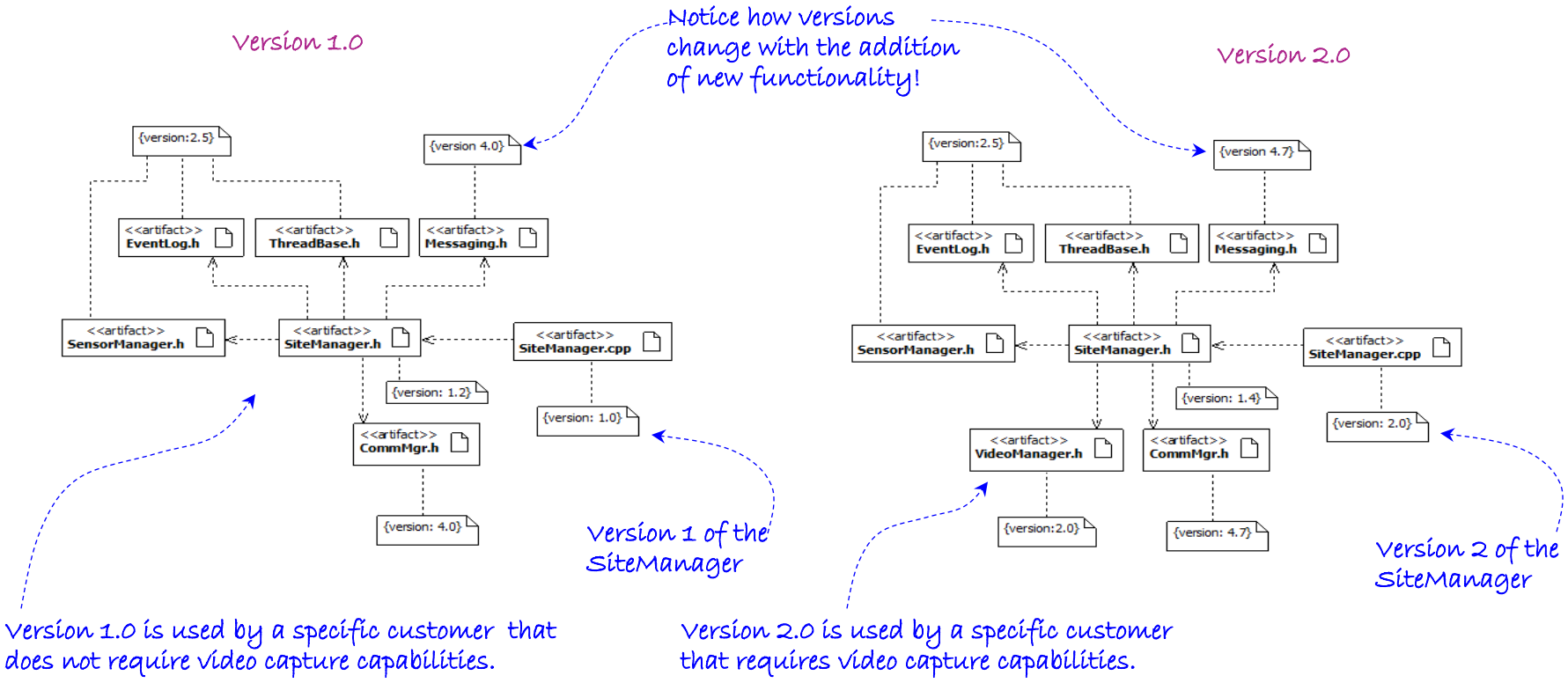


Development View of the Architecture

- 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 View of the Architecture

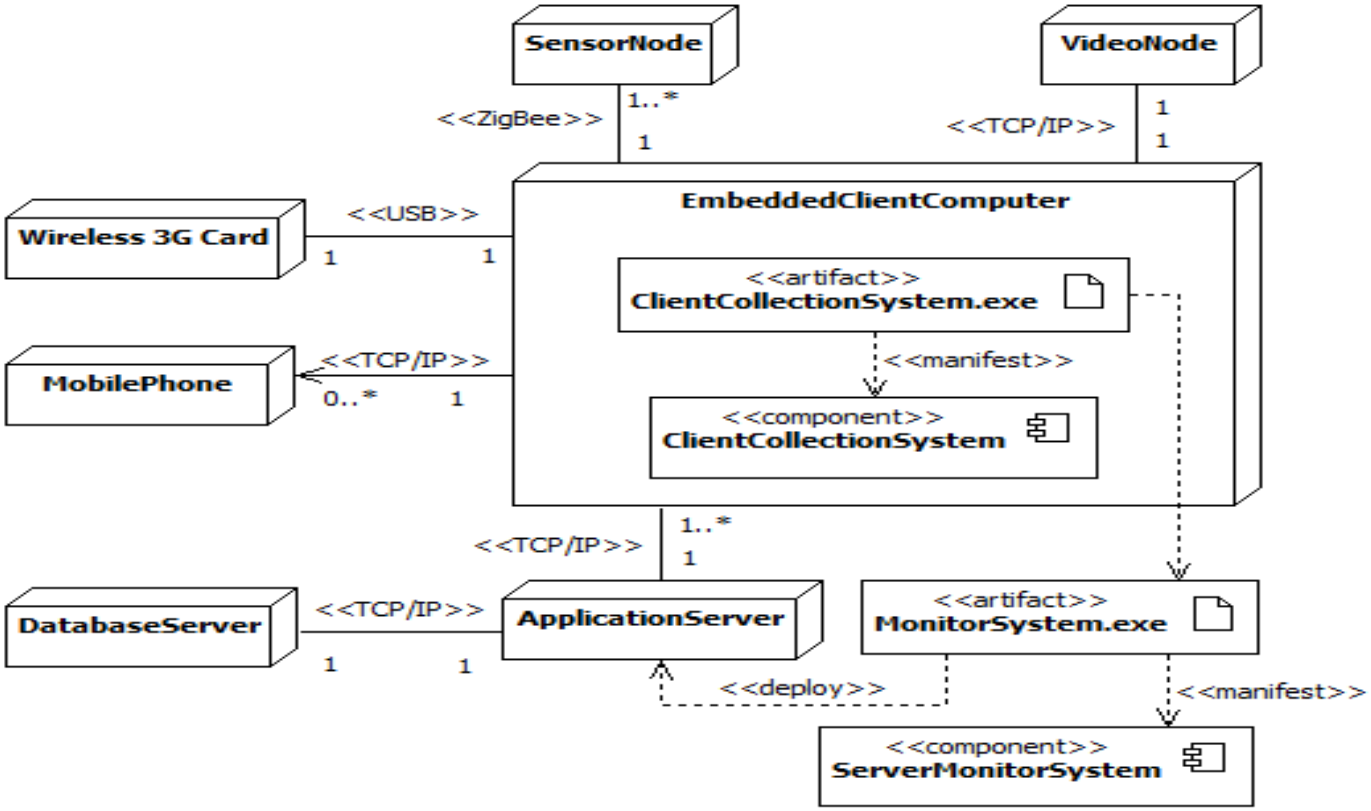


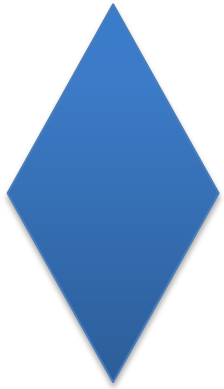
Physical View of the Architecture

- 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.

Physical View of the Architecture



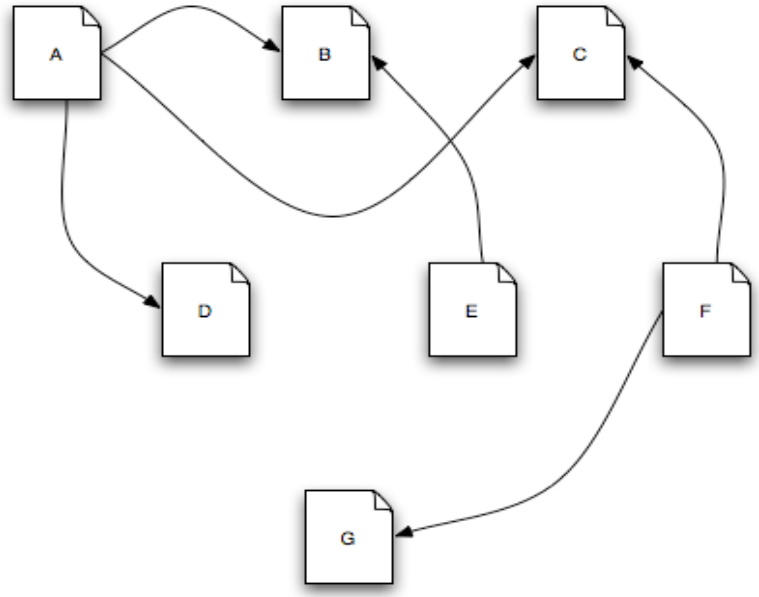


The Power of Architecture

- What is the World Wide Web?
- How is it built?
- How do you explain the Web to a child?
- How do you design Web-based e-commerce software?
- One of the world's most successful applications is only understood adequately from an architectural vantage point.

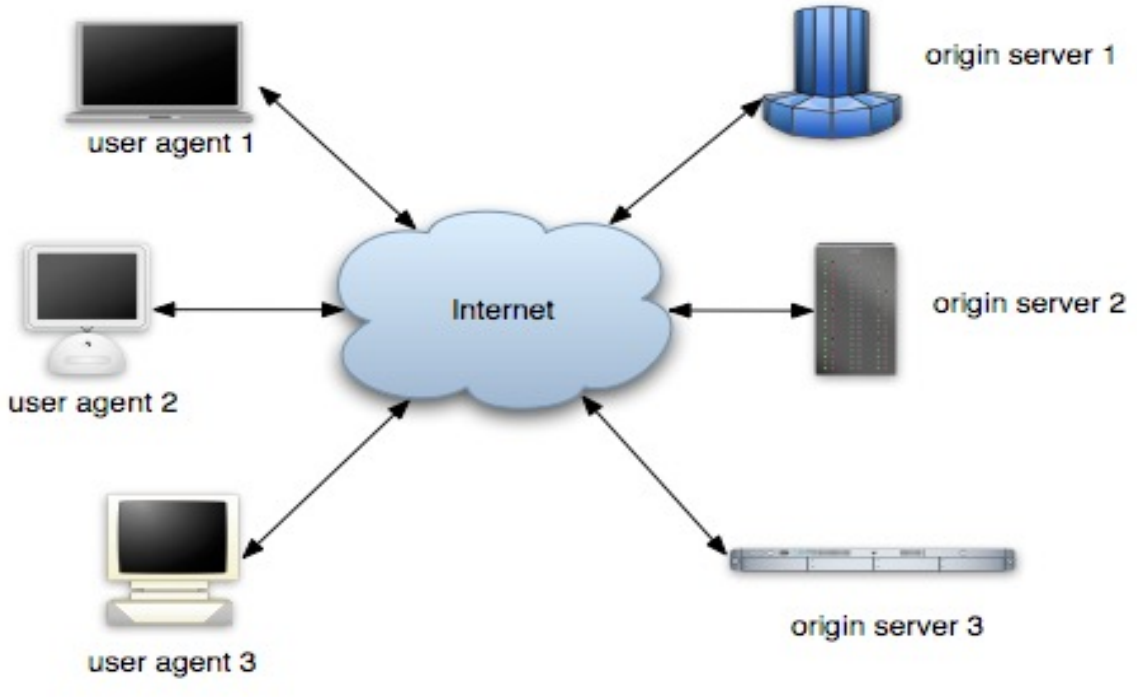
The Architecture of the WWW

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



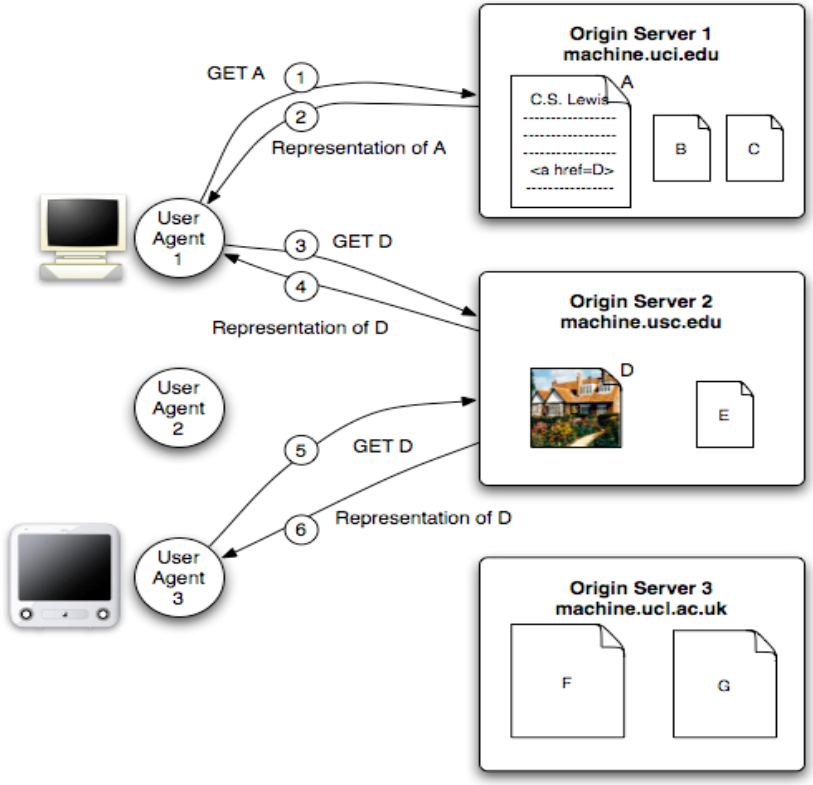
The Architecture of the WWW (cont'd)

- 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")