

# **System Design**

## **Design Views, and the SSAD**

CS 577a

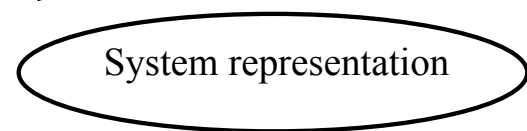
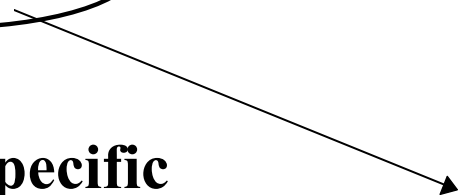
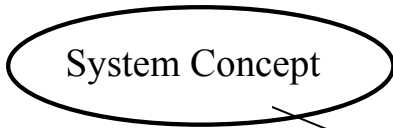
Fall 2001

# 3. System Design

- Details on how the system can be represented in software
- Describes specific technology solutions that meet requirements (both project and system)
  - high-level: resolves Analysis issues
    - how will roles and states be handled, expand bi-directional relationships, break multi-way relationships, handle global and relational attributes, decompose Components into objects, complex dependencies and other constraint
  - low-level: direct implementation considerations
    - use of databases, web-servers, hardware, critical algorithms, sequence, significant events, GUI's, etc.

# People - technology gap

- General
- People

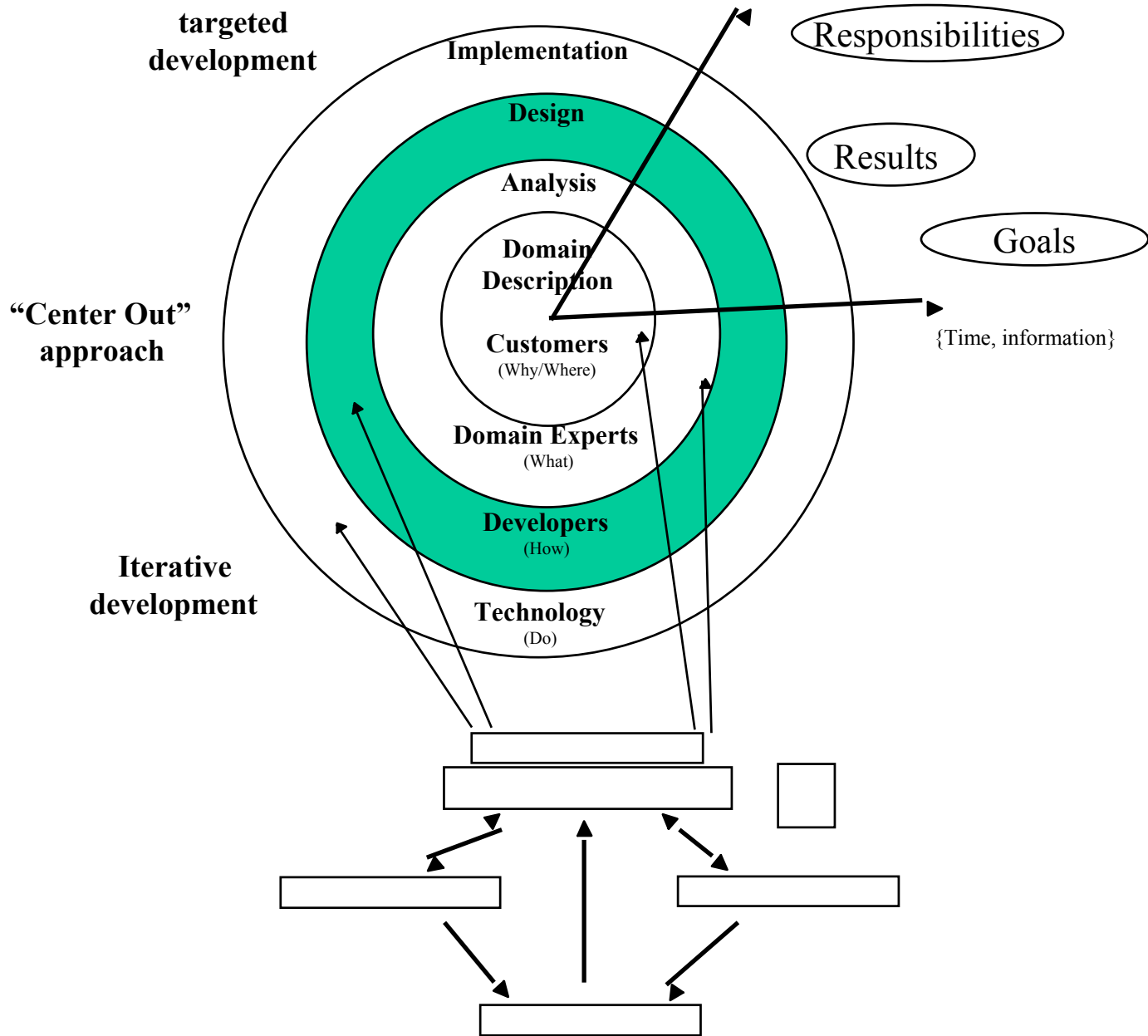


- Specific
- Technology

General	Specific
Domain Description	System Analysis
System Design	System Implementation

**People**

**Technology**



# OO models and abstractions

- Models provide a tangible way to work with groups of complex concepts through abstractions
- Enable management of complex sets of abstractions
- Offer multiple views of abstractions for a variety of purposes

\*\* All models use the language of abstractions \*\*

# What is OOD?

- Mapping Analysis models to software
- Focus on finding software representations and solutions to implementation problems
- May target particular technologies
- Addresses:
  - UI's
  - DB descriptions
  - Design constructs (e.g. custom value objects, layers, ...)

# Purpose of Design

Describes how the system components are to be realized within specific software structures

Answers: “What are the fundamental objects (and their relationships) that can faithfully represent the system components?”

# OOD Main Tools

- Formalized conventions
  - Address many implementation details
- Analysis models
  - Traceable evolution from what is wanted
- Architectural views
  - Indicate how to implement Components in software
- Design Patterns
  - Can help with common complex implementations
- Frameworks and mechanisms
  - Provide simple solutions to common problems
- Documentation (models) for implementers

# Design resolves Analysis issues

- Resolve policies by finding “algorithms” to carry out
- Resolves complex relationships between components
  - bi-directional, multi-way
  - memberships
  - multiplicity (containers, selectors, etc.)
  - relational attributes
- Enforces constraints (especially dependencies)
- Resolves roles, complex states, sub-types, modes

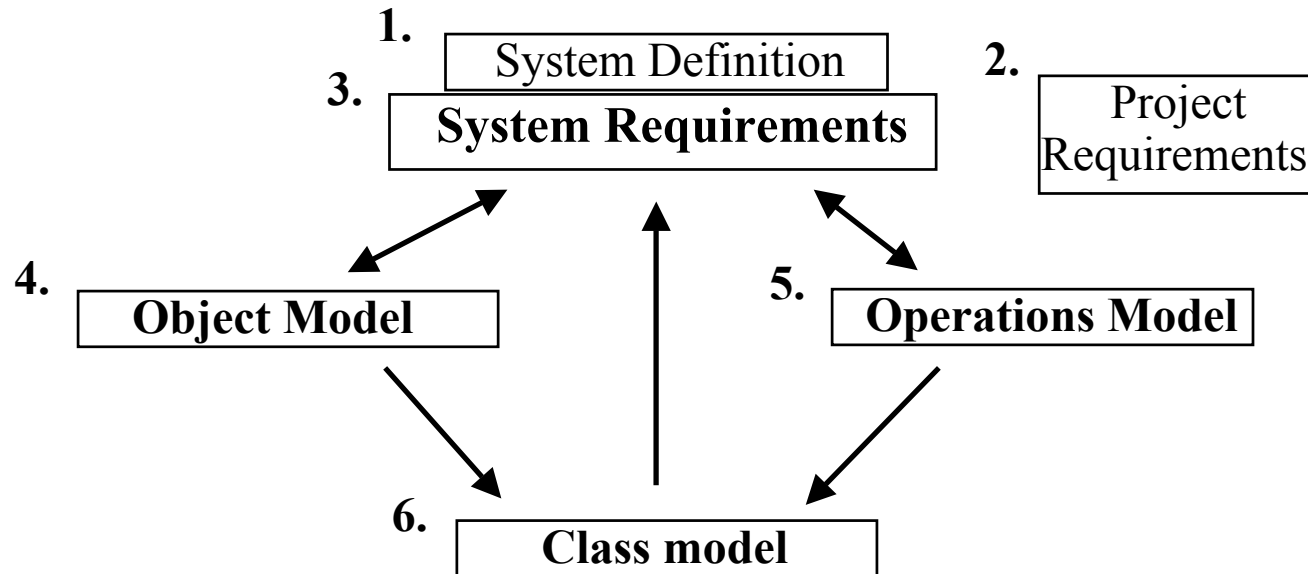
# Design models may include COTS

- COTS solutions
  - mechanisms
  - frameworks
  - API's
  - sub-systems
  - entire applications
  - Software libraries
  - Repositories

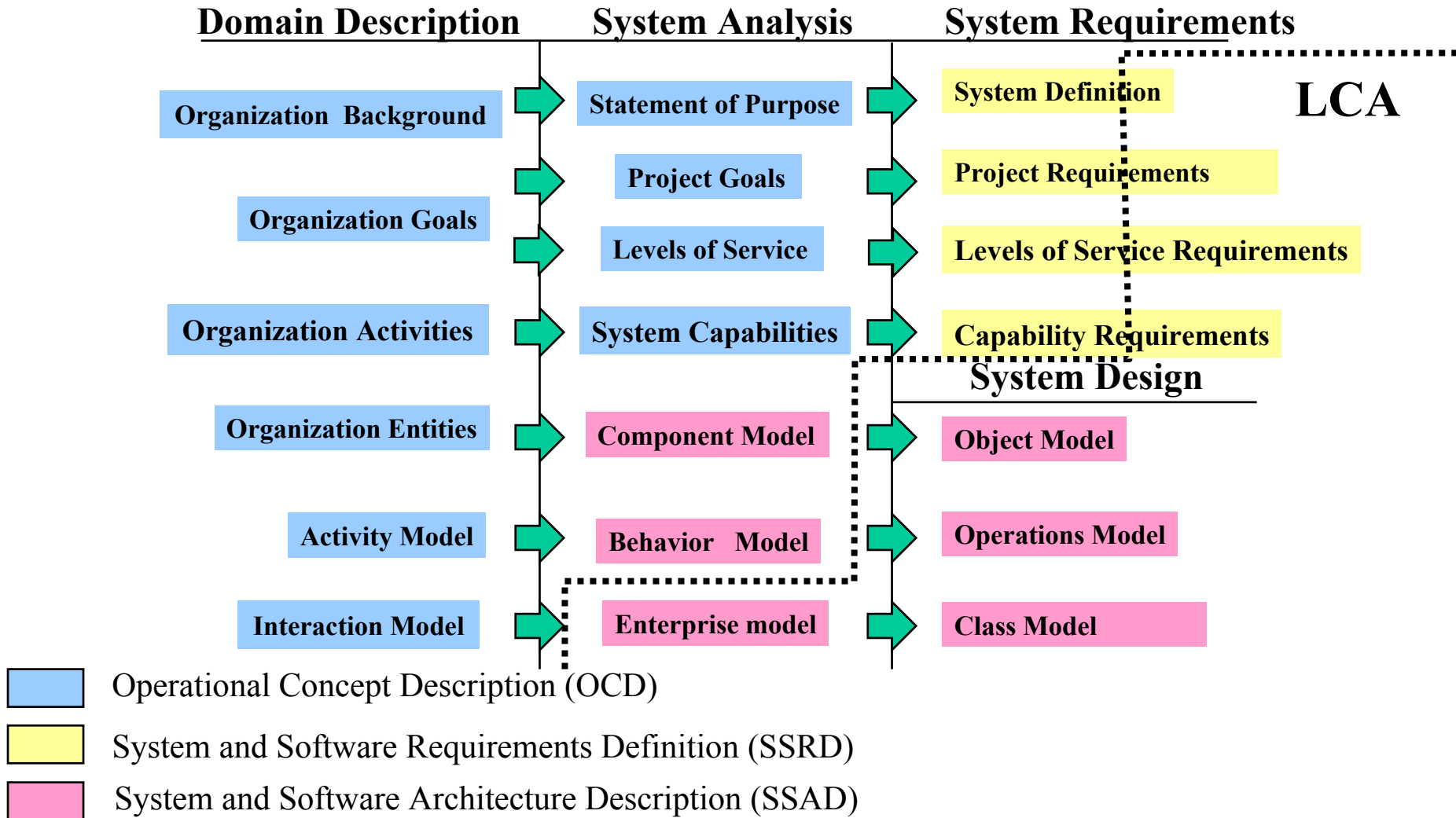
# Design Approaches

- Not an exact science
  - many many approaches, variations, choices, solutions
  - an “art” but has basic principles to follow
  - often no single “right” answer, usually “will work”, “is feasible”, or “best we can do for now”

# Design Model Views



# Integration of MBASE System Definition Elements



# 3.1 Design Views

- Consist of:
  - 3.1.1 System Topology (Layers)
  - 3.1.2 Component Implementation Design
  - 3.1.3 Framework and Protocol Specs
  - 3.1.4 System Deployment
  - 3.1.5 Logical Class

# Component Design Views

- What are they?
  - Describe how system components are mapped into low-level software architecture
- Why?
  - Help identify what **objects** are needed by grouping components into technology representation “clusters” from different perspectives:
    - discover straightforward implementations and design patterns
    - identify “gaps” for which particular system objects must be created to fill (no direct relevance to domain here, makes components “work” in software)

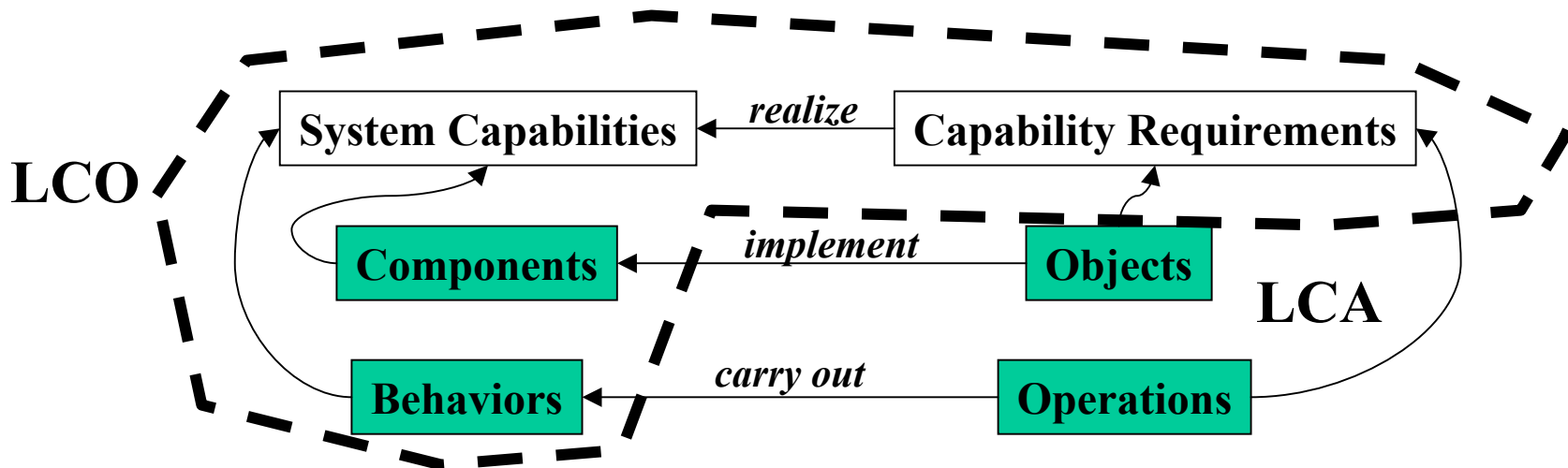
# A Brief Introduction to Objects...

# About Objects...

- A fundamental, semi-autonomous software-level structure that embodies both data and operations to represent or support system components
  - Examples: DB tables, Java classes, HTML pages, files
  - may be introduced simply to handle “technical” issues
  - GUI
    - special buttons, HW/SW constraints, input/output validation (dates, time, SS#, etc.)
    - DB unique ids, primary keys, stored procedures
    - Facilitate integration and communication through APIs, translations, multiple constrained relationships
    - Event handling - exceptions, sequence of operations (precedence, synchronization, model loops)

# Objects in Design

- Objects add implementation refinements to the Component Model (SSAD 2.1), but also may appear earlier in the design views (SSAD 3.1)
- Objects are software elements introduced to directly implement Components, handle component interactions and represent design elements within logical, topological, and physical views, implement complex relationships, attributes, and constraints



# Components vs. Objects

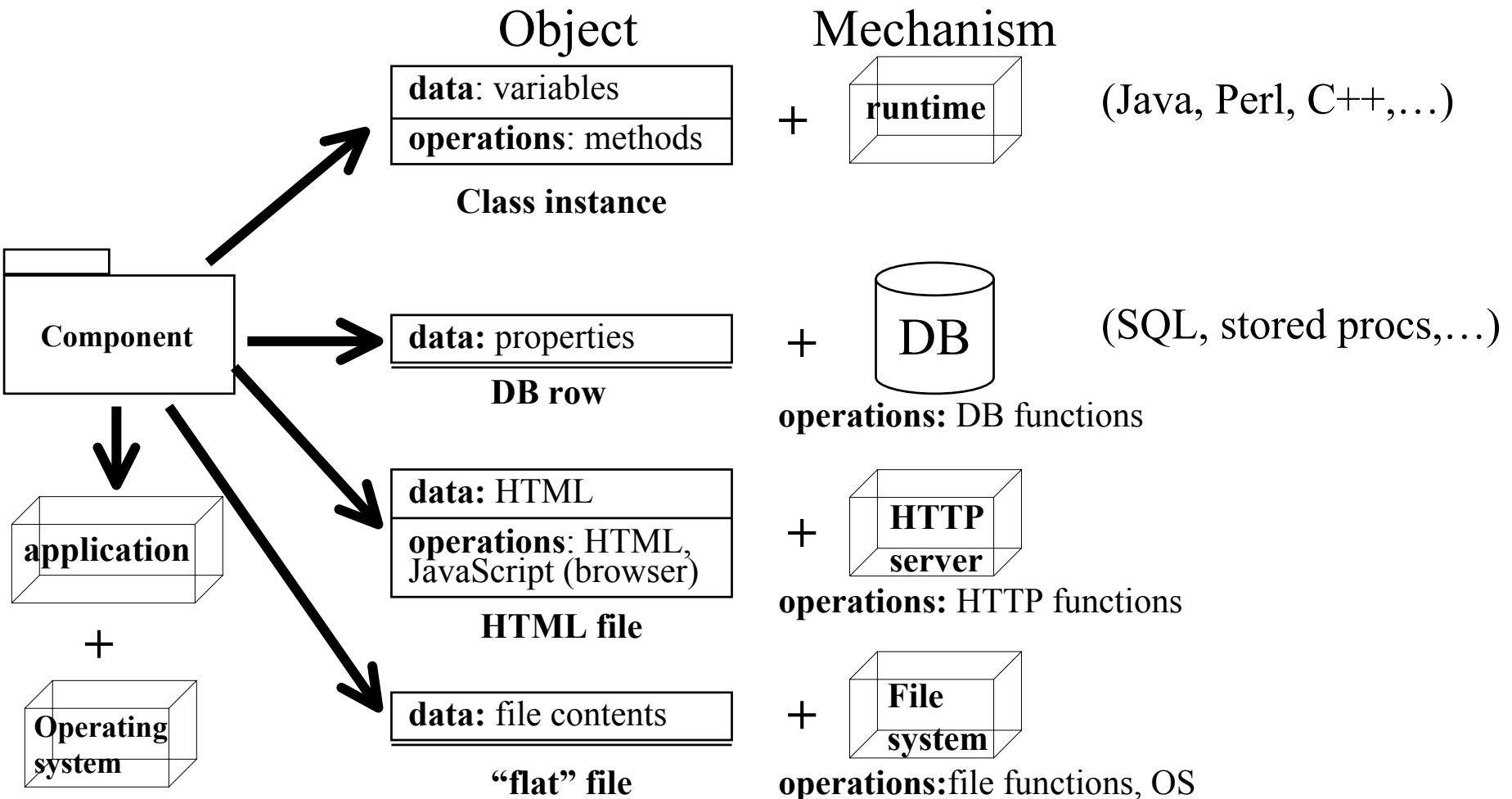
- **Objects** are the smallest (most refined) entity we consider in our models prior to implementation
- **Components** are compositions (membership relationships) of objects with a high degree of *cohesion* within the domain
- **Components** are what you need to describe the system to domain experts at a higher level of abstraction (less detail)
- **Components** are used to describe “parts” in the design views
- **Objects** are usually associated with technology
- **Components** are usually associated with the domain

# Objects

- In the Design phase you decompose components into objects.
- Objects are used to represent the system in software.
- Objects are specialized “parts of” a component.
- An object is an atomic unit for systems analysis purposes.
- Introduce objects in the design views (SSAD 3.1) to handle technology gaps external to components

# Straightforward Component Design

- Often a simple object + mechanism (or framework) suffice to realize a component in software



Now Back To Design Views!

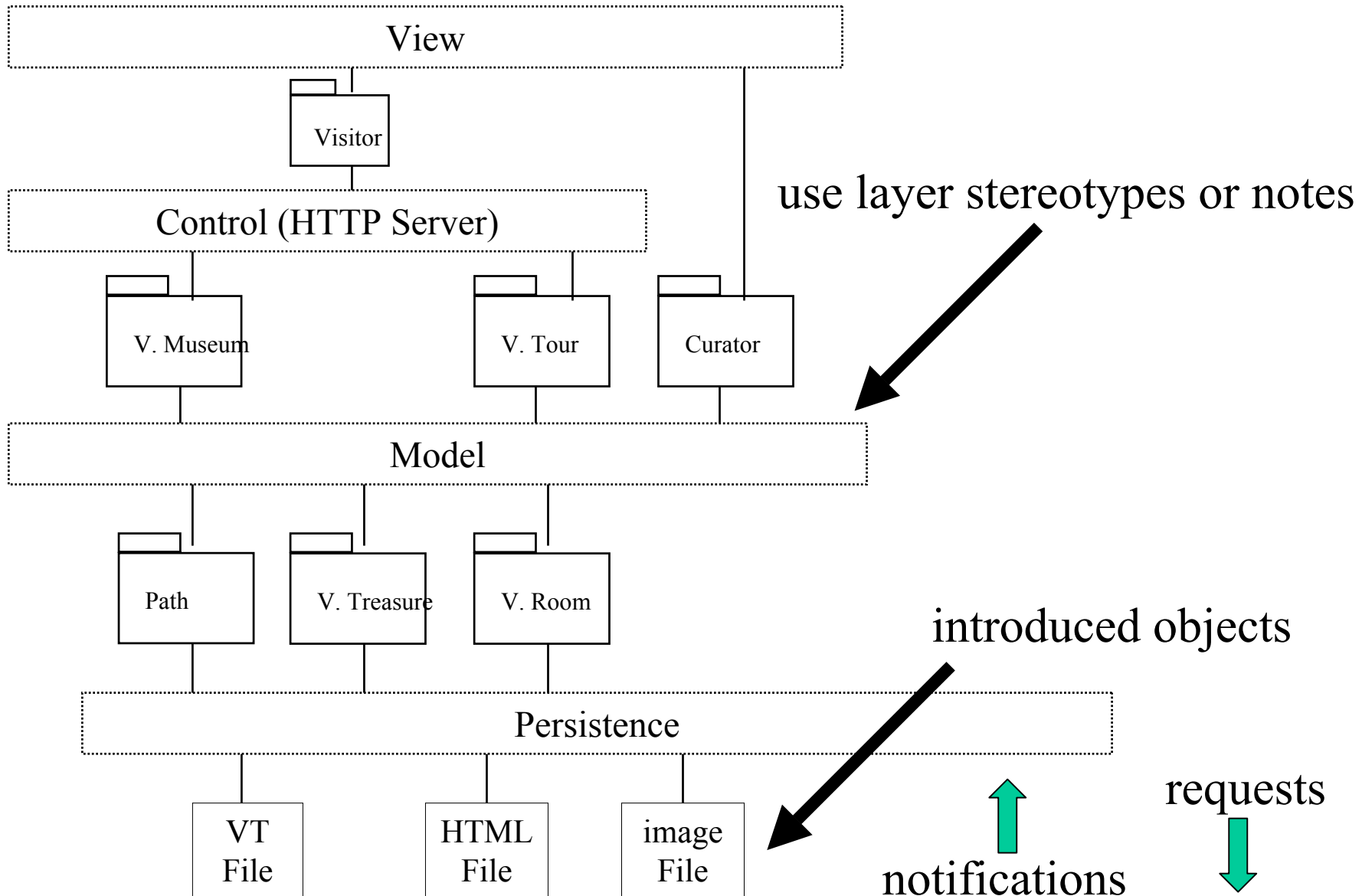
# 3.1.1 System Topology

- System Layered View elements are often called subsystems.
- Subsystems are organized in a hierarchy of layers, where each layer has a well defined interface. Some of the layers that can be found in a system:
  - User Interface (View)
  - Application Specific Packages
  - Reusable Business Packages
  - Key Mechanisms (Persistence, Control, etc.)
  - Hardware and Operating System Packages

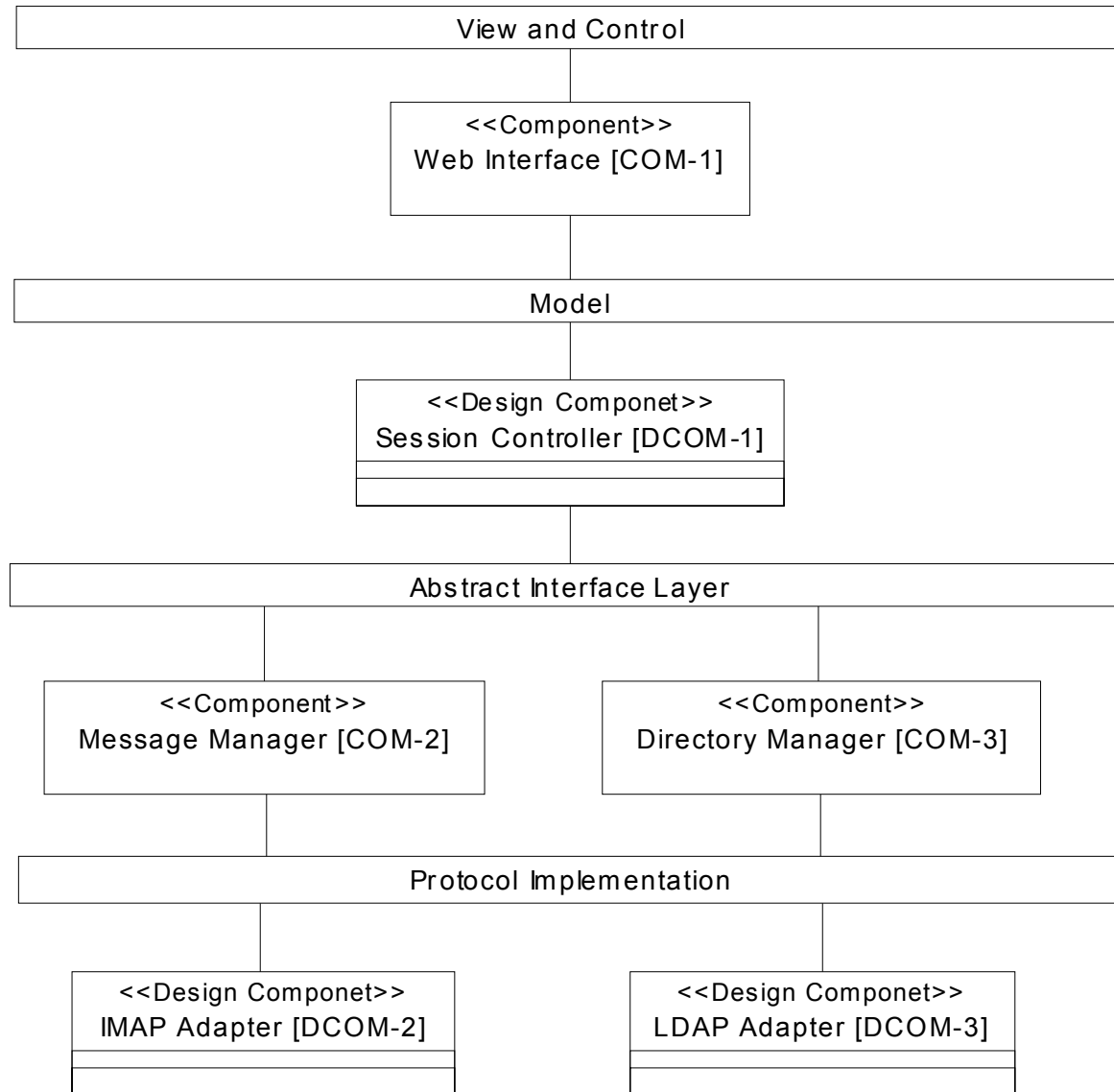
# System Topology

- Assigns components to system “layers”
  - components in the same layer implicitly communicate
  - notifications move upward, requests downward
  - Helps identify useful mechanisms and frameworks (particularly COTS DB’s, file systems, GUI’s)
  - Loosely based on C2 Architectural style (see <http://www.ics.uci.edu/pub/arch/c2.html>)
- Splits system into communication areas
  - very common that components that communicate across communication areas require new objects to facilitate (e.g. JDBC for Java, Apache for file system to WWW browser)

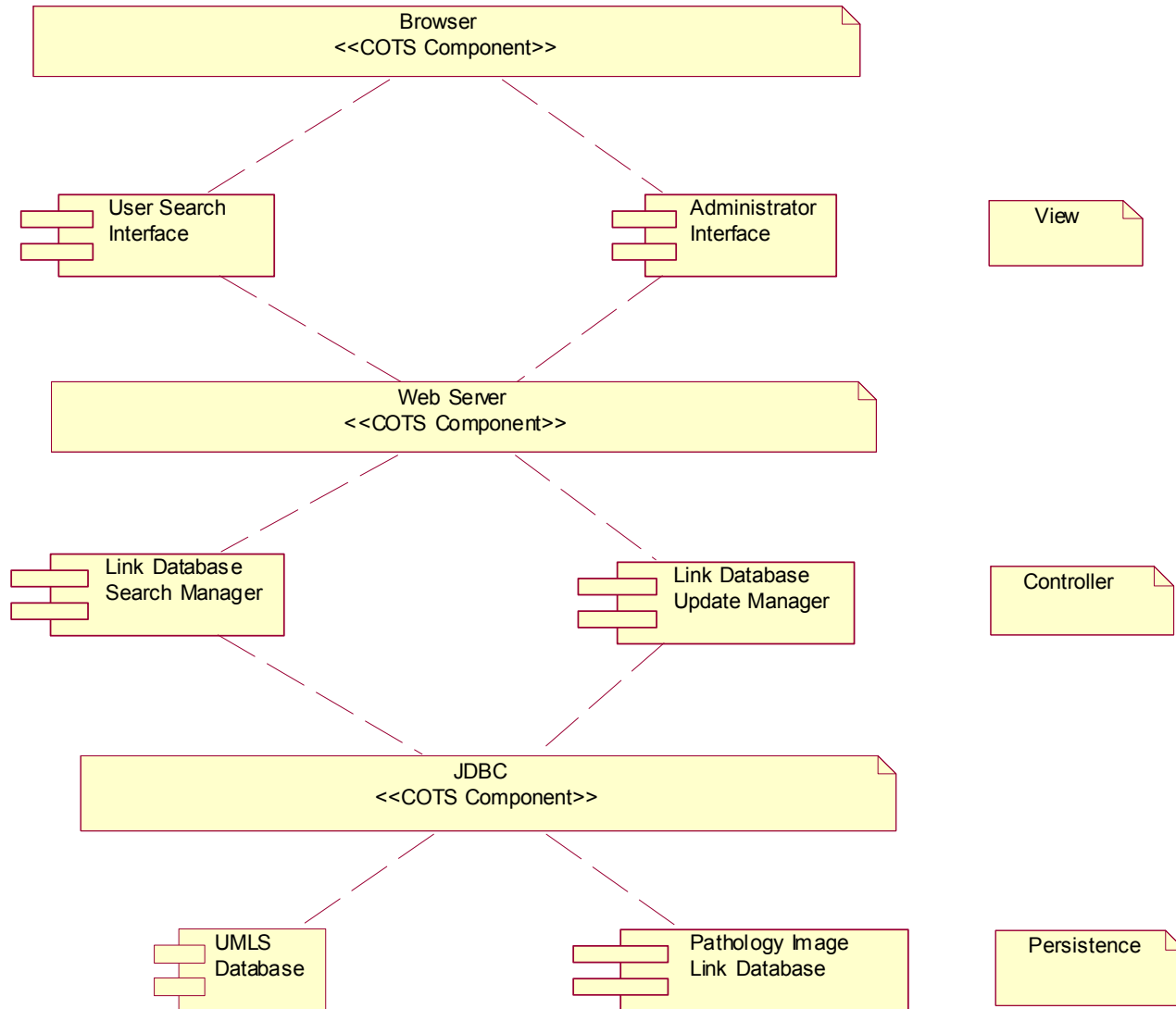
# System Topology View Example



# SSAD 3.1.1 System Topology View (Web Mail)



# SSAD 3.1.1 System Topology View (PISE)



## 3.1.2 Component Implementation

- Describe how the components from the Component Model (SSAD 2.1) will be implemented as software elements
- Add new components as necessary to make the system realizable in software
  - May be implementation (technology) specific
  - Complex relationships of software elements which are never considered independently (e.g., a COTS package with no source code)

# Component Specification

- Component 1

Identity -

Attributes -

Assigned Behaviors -

Relationships (aggregation, association, interface, observer, etc.) -

State Groups -

Possible Roles -

Constraints -

Implementation (Kind Of Object) - e.g., application, server, existing subsystem, COTS package

Relates to component -

# Component Implementation (Web Mail)

<b>Identifier</b>	<b>DCOM-2</b>
Defining quality	A message protocol interface implementation component which handles the details communication with the message server
Name	IMAP Adapter
Attributes	1 Server IP address 2 Server port number
Assigned Behaviors	2 Retrieve user account information 3 Retrieve message folder information 4 Retrieve messages 5 Modify user account information 6 Modify message folder information 7 Send messages
Relationships	COM-2 Message Manager
State Groups	
Constraints	
Implementation	Java Beans with IMAP interface implementation (JavaMail)

# Component Implementation Inspector

DCOM-02 Zaguán

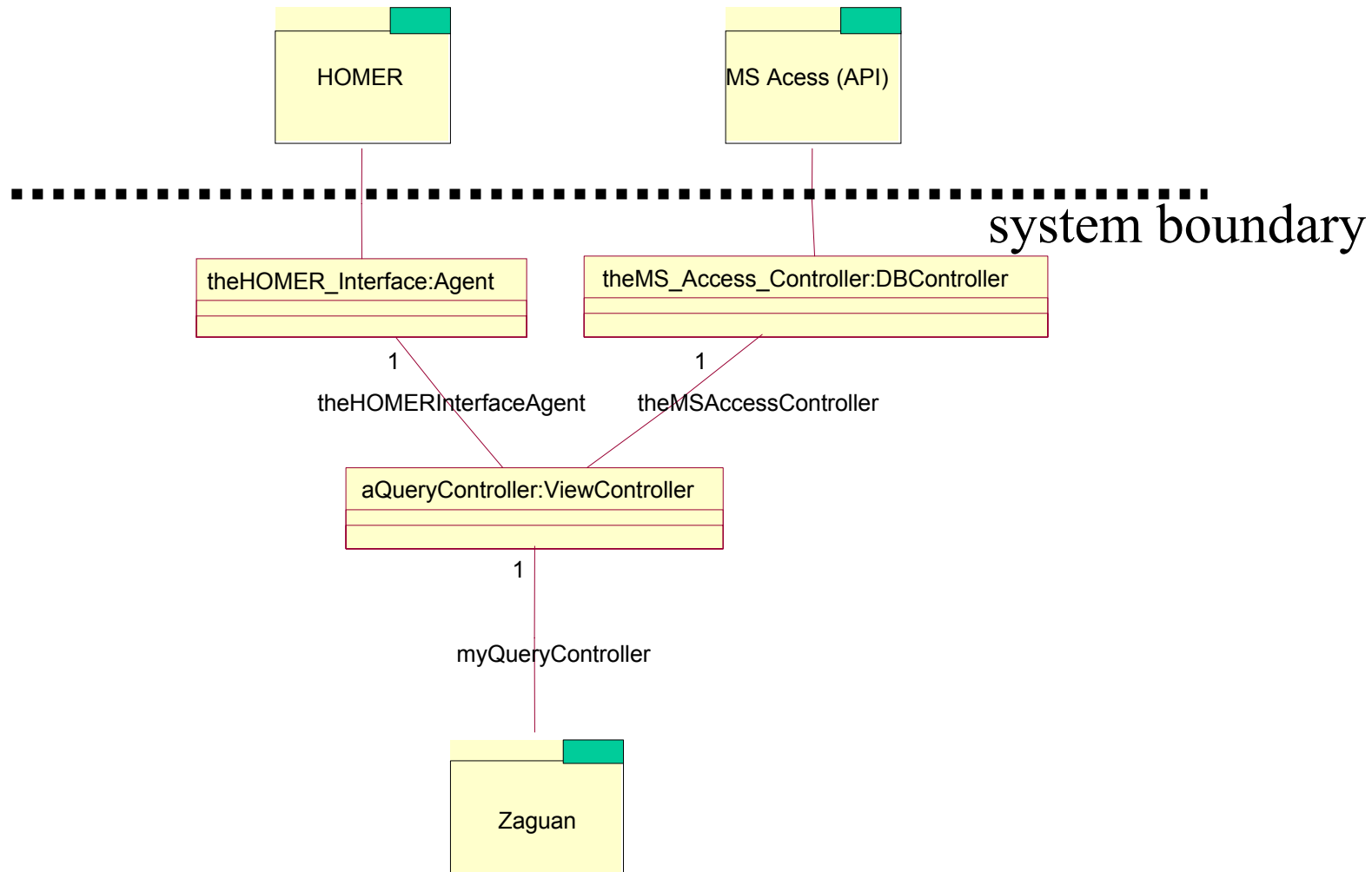
A COTS component that provides an Internet based interface to the patrons into the archive.

<i>Attributes</i>	<ul style="list-style-type: none"> <li>a) URL</li> <li>b) Viewing preferences</li> <li>c) Cache</li> </ul>
<i>Assigned Behaviors</i>	<ul style="list-style-type: none"> <li>a) Back and Forward navigation</li> <li>b) Add and remove bookmarks</li> <li>c) HTTP request (missing behavior references)</li> <li>d) Stop loading</li> </ul>
<i>Relationships</i>	<ul style="list-style-type: none"> <li>a) Archive Web Server</li> <li>b) Patron Interface</li> </ul>
<i>State Groups</i>	HTTP {Lookup, Connecting, Waiting, Displaying}
<i>Possible Roles</i>	<ul style="list-style-type: none"> <li>HTTP request source</li> <li>HTTP reply destination</li> </ul>
<i>Constraints</i>	Should accept HTML 3.2 strict
<i>Implementation</i>	Netscape Navigator 4.0+ or Internet Explorer 4.0+

(Relates to component and requirements missing)

# SSAD 3.1.2

## Component Implementation



# 3.1.3 Framework and Protocol Specifications

- Describe the specific frameworks to be used and the nature of interactions among the logical components (SSAD 2.1) and design components (SSAD 3.1.2) in order to support all the behaviors of the system as described in SSAD 2.2
- Protocols and services are typically independent of the domain and their choice depends upon the various requirements of the system, most notably Level of Service Requirements (SSRD 5).
- Examples of frameworks include CORBA Services and Facilities, Java JDK, TCP/IP and various network protocols as well as security and audit mechanisms.

# 3.1.3 Framework and Protocol Specifications (Web Mail)

## IMAP

The Internet Message Access Protocol is the email access protocol interface in the existing email servers that the components in our system talk to. It also provides the user folder manipulation in which user preferences or user data can be stored through this protocol.

Refer to IMAP official website <http://www.imap.org>

Refer to RFC2060 <http://www.cis.ohio-state.edu/htbin/rfc/rfc2060.html> for IMAP specification.

## LDAP

The Lightweight Directory Access Protocol is the communication protocol that existing directory server provides us to query and retrieve email addresses by using name of the person.

Refer to RFC2251 <http://www.innosoft.com/ldapworld/rfc2251.txt> for LDAP specification.

Refer to iPlanet [http://www.iplanet.com/downloads/developer/detail\\_43\\_356.html](http://www.iplanet.com/downloads/developer/detail_43_356.html) for

Netscape Directory SDK for Java.

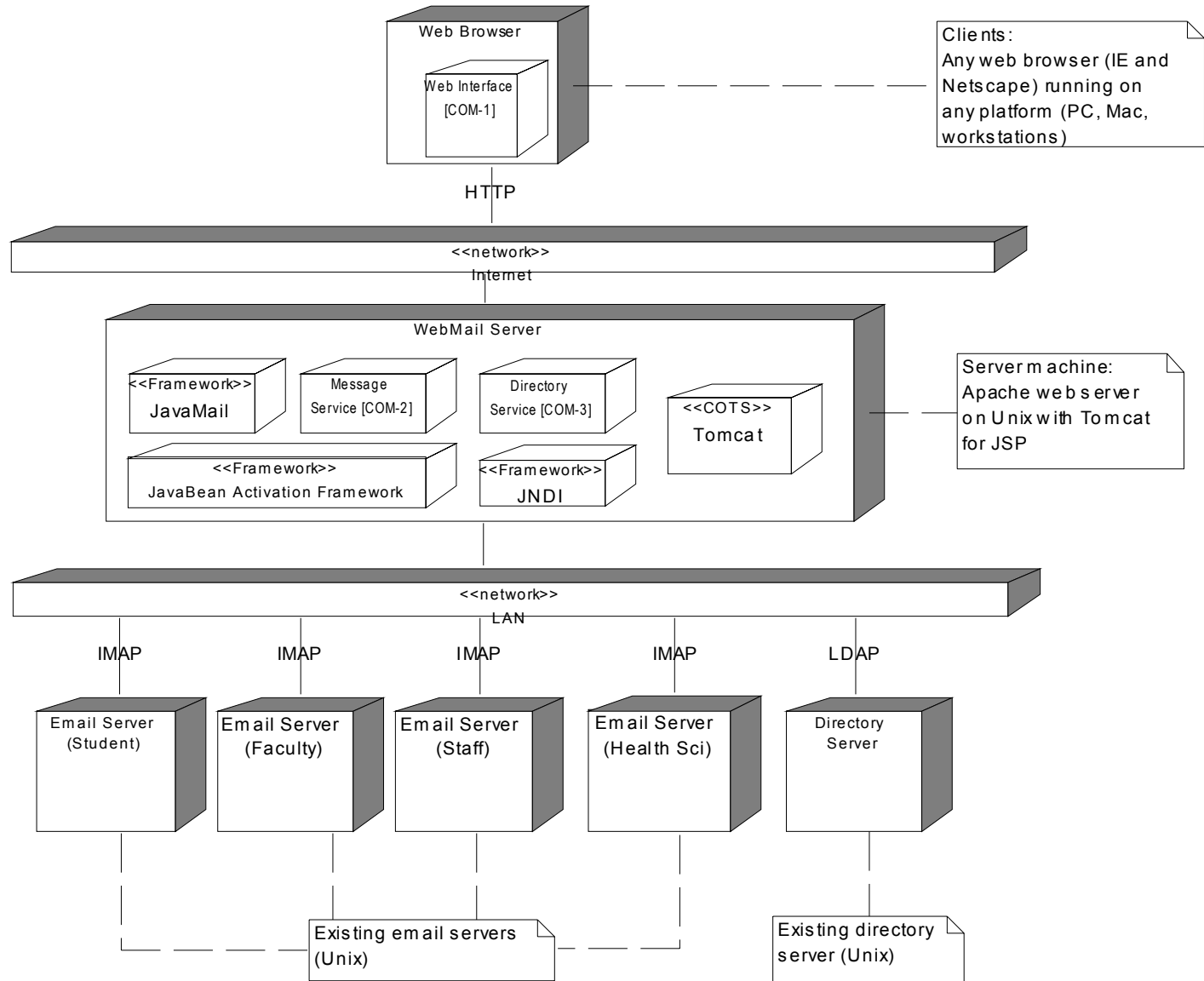
## 3.1.4 Deployment View

- Consistent with System Component View diagrams and the various interface diagrams
- Use the Deployment View to document the physical (i.e., hardware components) architecture of the system and how the logical components are deployed on them.
- Use a UML Deployment Diagram

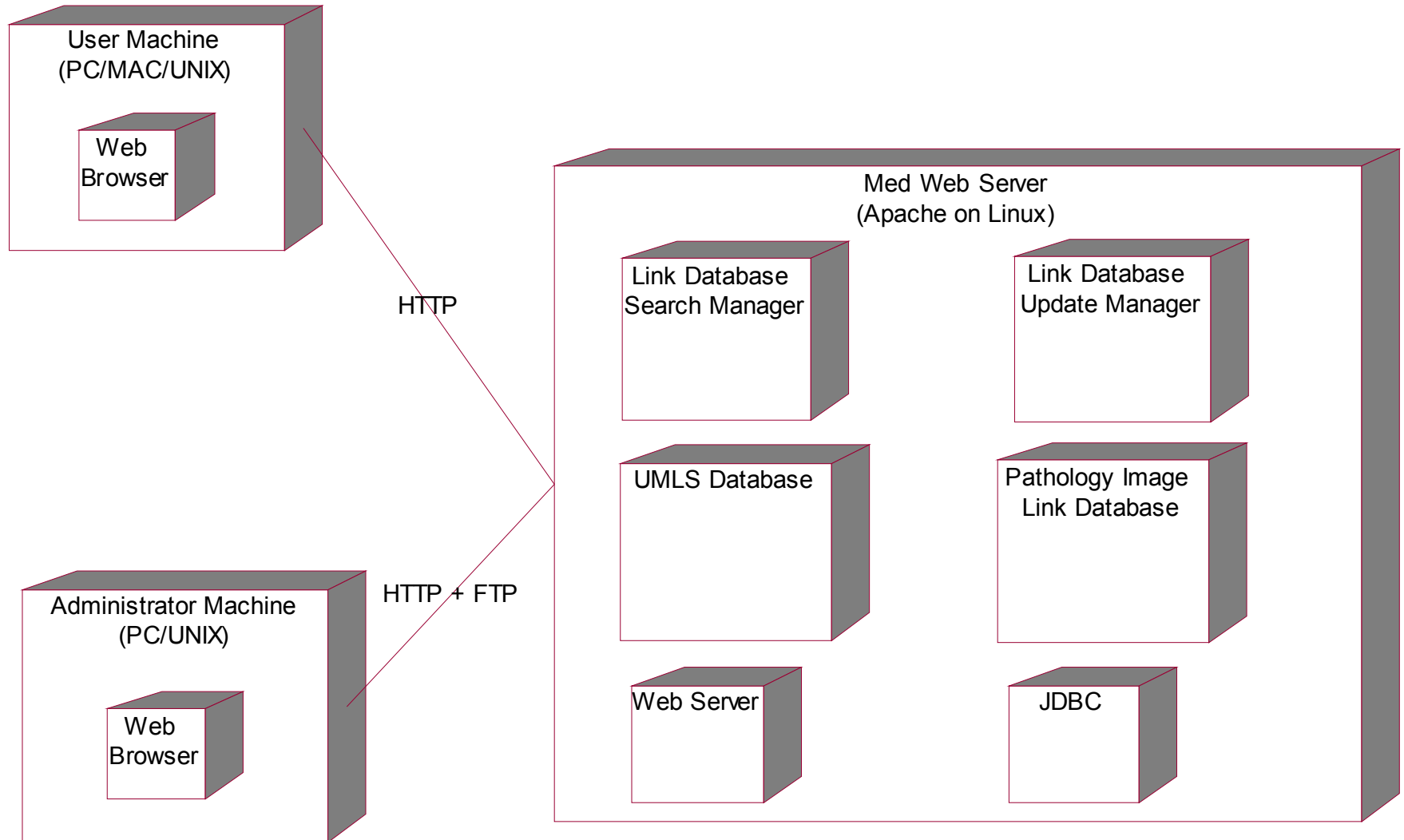
# System Deployment View

- Assigns components to deployed hardware and software
  - if known, includes OS, mechanisms, and frameworks
- Splits system into physical groups
  - very common that components that communicate across physical groups will require “glue” objects

# SSAD 3.1.4 Deployment View (Web Mail)



# SSAD 3.1.4 Deployment View (PISE)



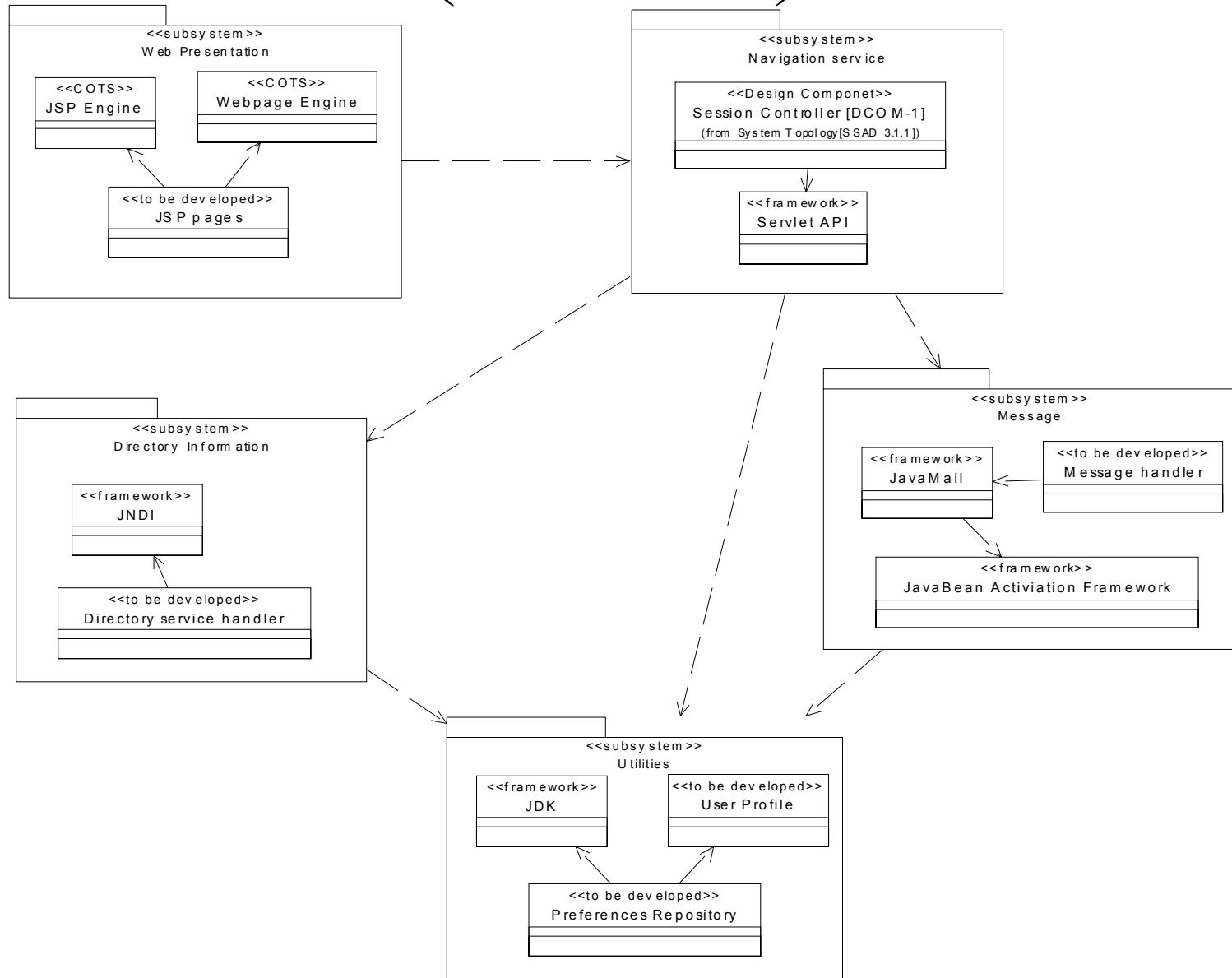
## 3.1.5 Logical Class Model

- Logical description of the system
- Should be consistent with System Definition (SSRD 2.1)
- Start with Block Diagram from SSRD 2.1

# Logical Class Model

- Assigns components to system block diagram or other logical system group
  - conceptual understanding and completeness
  - ensures consistency, accessibility, and necessity of system parts and relationships to outside (system boundaries)
  - may introduce new technology choices (some may exist from block diagram or requirements)

# SSAD 3.1.5 Logical Class Model (Web Mail)



# SSAD 3.1.5 Logical Class Model (PISE)

