


University of Southern California  
Center for Software Engineering

---

# Introduction to Object- Oriented with Architecture Design, Part 1 CS577a, Fall 2005

Ed Colbert  
USC Center for Software Engineering

1 10/9/2006




University of Southern California  
Center for Software Engineering

---

## Goal of Presentation

- Understand
  - What is an architecture
  - What are *Technology-Independent & Specific Architectures*
    - Why do we care?
  - How to model & document an architecture
    - Using
      - MBASE
      - Object-oriented techniques
        - » Unified Modeling Language (UML) 2
      - Rational System Architect
  - How do we design an architecture?
  - What's a good architecture?

Intro. to OOAD with Arch. Design, Part 1, CS577a, Fall 2005s 2 10/9/2006




University of Southern California  
Center for Software Engineering

---

## Outline

- What is an architecture?
  - And related concepts
- Technology-Independent Architecture
- Technology-Specific Architecture
- Milestone Expectations
- Summary & Pre-class Exercise

Intro. to OOAD with Arch. Design, Part 1, CS577a, Fall 2005s 3 10/9/2006




University of Southern California  
Center for Software Engineering

---

## What's A Software Architecture?

- Organizational structure of a system or component
  - » IEEE Standard Glossary of Software Engineering Terminology [IEEE Std 610.12-1990]
- Fundamental organization of a system embodied in
  - Its components
  - Relationships among components
  - Relationships to environment
  - Principles guiding its design & evolution
    - » IEEE Recommended Practice for Architectural Description of Software-Intensive Systems [IEEE Std 1471-2000]

Intro. to OOAD with Arch. Design, Part 1, CS577a, Fall 2005s 4 10/9/2006



University of Southern California  
Center for Software Engineering

## What's A Software Architecture? (cont.)

Software Architecture =

{ Elements, Form, Rationale }

↑

What

↑


How

↑

Why

» "Foundations for the Study of Software Architectures" [Perry & Wolf 92]

Intro. to OOAD with Arch. Design, Part 1, CS577a, Fall 2005s
5
10/9/2006



University of Southern California  
Center for Software Engineering

## What's A Software Architecture? (cont.)


❑ *Software Architectures* defines for a system

- Computation components
  - Clients
  - Servers
  - Databases
  - Filters
  - Layers

- Interactions among components
  - Subprogram calls
  - Shared data
  - Client–server
  - DB–accessing protocols
  - Asynchronous even multicast
  - Piped streams
  - etc.

» *Software Architectures* [Shaw & Garlan 96]

Intro. to OOAD with Arch. Design, Part 1, CS577a, Fall 2005s
6
10/9/2006




University of Southern California  
Center for Software Engineering

## What's A Software Architecture? (cont.)

- ❑ Shaw & Garlan (cont.)
  - [A level of design that] involves
    - Description of elements from which systems are built
    - Interactions among those elements
    - Patterns that guide their composition
    - Constraints on these patterns

Intro. to OOAD with Arch. Design, Part 1, CS577a, Fall 2005s 7 10/9/2006




University of Southern California  
Center for Software Engineering

## What's A Software Architecture? (cont.)

- ❑ Design & implementation of high-level structure of software
  - Involves
    - Abstraction
    - Decomposition
    - Composition
    - Style
    - Aesthetics

» "Mommy, Where Do Software Architectures Come from?", Phillip Kruchten, Rational, White Paper.

Intro. to OOAD with Arch. Design, Part 1, CS577a, Fall 2005s 8 10/9/2006




University of Southern California  
Center for Software Engineering

---

## Focus Of SW Architectures (cont.)

- ❑ Framework for understanding system-level concerns
  - Global rates of flow
  - Communication patterns
  - Execution control structure
  - Scalability
  - Paths of system evolution
  - Capacity
  - Throughput
  - Consistency
  - Component compatibility

Intro. to OOAD with Arch. Design, Part 1, CS577a, Fall 2005s 9 10/9/2006




University of Southern California  
Center for Software Engineering

---

## Benefits Of Explicit Architectures

- ❑ Framework for satisfying requirements
- ❑ Technical Basis for
  - Design
  - Consistency analysis
  - Dependency analysis
  - Reuse
- ❑ Managerial basis for
  - Cost estimation
  - Process management

Intro. to OOAD with Arch. Design, Part 1, CS577a, Fall 2005s 10 10/9/2006




University of Southern California  
Center for Software Engineering

---

## Key Architectural Concepts

- ❑ Component
  - Unit used as part of some system
  - Locus of computation & state
- ❑ Connector
  - Element that models
    - Interactions among components
    - Rules that govern those interactions
- ❑ Configuration
  - Connected graph of components & connectors which describes architectural structure

Intro. to OOAD with Arch. Design, Part 1, CS577a, Fall 2005s 11 10/9/2006




University of Southern California  
Center for Software Engineering

---

## What's A Component?

- ❑ A constituent part: ingredient
  - » Merriam-Webster's On-line Collegiate Dictionary

Intro. to OOAD with Arch. Design, Part 1, CS577a, Fall 2005s 12 10/9/2006




University of Southern California  
Center for Software Engineering

## What's A Component? (cont.)

- ❑ A unit of computation or a data store
  - Perry & Wolf's processing & data elements
    - » "Foundations for the Study of Software Architectures" [Perry & Wolf 92]

Intro. to OOAD with Arch. Design, Part 1, CS577a, Fall 2005s 13 10/9/2006




University of Southern California  
Center for Software Engineering

## What's A Component?

- ❑ Loci of computation & state
  - clients
  - databases
  - filters
  - servers
  - layers
  - ADTs
- ❑ Has an interface specification that defines its properties
  - » [Shaw & Garlan 96]

Intro. to OOAD with Arch. Design, Part 1, CS577a, Fall 2005s 14 10/9/2006



University of Southern California  
Center for Software Engineering


## What's A Component?

- ❑ A unit of
  - independent deployment
  - third-party composition
- ❑ Has no persistent state
  - » Component Software: Beyond Object-Oriented [Szyperski 97]

\*\*\* Definition conflicts with

- Perry & Wolf
- Shaw & Garland

Intro. to OOAD with Arch. Design, Part 1, CS577a, Fall 2005s 15 10/9/2006




University of Southern California  
Center for Software Engineering

## What's A Component? (cont.)

- ❑ UML v1.4
  - Modular, deployable, & replaceable part of system that
    - Encapsulates implementation
    - Exposes set of interfaces
    - Specified by 1+ classifiers that reside on it
  - May be implemented by 1+ artifacts
    - e.g., binary, executable, or script files

Intro. to OOAD with Arch. Design, Part 1, CS577a, Fall 2005s 16 10/9/2006



University of Southern California  
Center for Software Engineering


## What's A Component? (cont.)

- ❑ A non-trivial, nearly independent, & replaceable part of system that
  - Fulfills clear function in context of well-defined architecture
  - Conforms to & provides physical realization of a set of interfaces

» RUP 2002

- Almost identical to UML

Intro. to OOAD with Arch. Design, Part 1, CS577a, Fall 2005s 17 10/9/2006




University of Southern California  
Center for Software Engineering

## What's A Component? (cont.)

- ❑ Instructor's Definition:
  - A unit used as a part of some system
    - Essentially same as Perry & Wolf
- ❑ Form of unit depends on
  - Abstraction & architectural style employed

Intro. to OOAD with Arch. Design, Part 1, CS577a, Fall 2005s 18 10/9/2006




University of Southern California  
Center for Software Engineering

## What's A Component? (cont.)

- Unit used as component of system should provide capabilities that are of use to system
  - Capabilities = behavior and/or information
  - If unit isn't of use, it should not be component of this system
- A component is expected to work with other components to satisfy requirements of system

Intro. to OOAD with Arch. Design, Part 1, CS577a, Fall 2005s 19 10/9/2006




University of Southern California  
Center for Software Engineering

## Component Goals

- Frequent system goals include
  - Easy maintenance
    - Easy replacement of parts
  - Easy optimization
  - High reliability
  - High security
- To achieve such goals
  - Components' interfaces should be well defined
  - Implementation of components should be hidden
  - Dependence on other components should be minimal

Intro. to OOAD with Arch. Design, Part 1, CS577a, Fall 2005s 20 10/9/2006




University of Southern California  
Center for Software Engineering

## Simple & Composite Component

- ❑ Components may be simple or composite
  - Parts of “Simple” components are different level of abstraction
    - e.g.
      - Heart & Cells
      - Screw & Molecules/Atoms
      - Class & Data & Operations
  - “Composite” component has parts that are same level of abstraction
    - e.g.
      - Body & Organ (e.g. heart)
      - Car & Engine;
      - Process & Thread
    - If composite components are systems,
      - Parent system is “System of Systems”

Intro. to OOAD with Arch. Design, Part 1, CS577a, Fall 2005s 21 10/9/2006




University of Southern California  
Center for Software Engineering

## Connectors

- ❑ A connector is an architectural element that models
  - Interactions among components
  - Rules that govern those interactions
- ❑ Simple interactions
  - Procedure calls
  - Shared variable access
- ❑ Complex & semantically rich interactions
  - Client-server protocols
  - Database access protocols
  - Asynchronous event multicast
  - Piped data streams

**We're not using SW connectors in 577!**

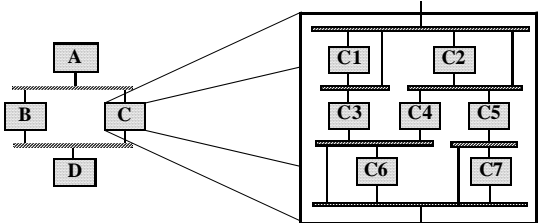
Intro. to OOAD with Arch. Design, Part 1, CS577a, Fall 2005s 22 10/9/2006




University of Southern California  
Center for Software Engineering

## Configurations/Topologies

- ❑ Describes architectural structure
  - Proper connectivity
  - Concurrent & distributed properties
  - Adherence to design heuristics & style rules
- ❑ Represented by connected graph of components & connectors
- ❑ Each composite component has a configuration



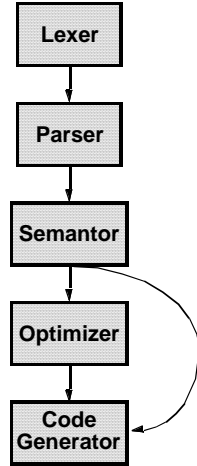
Intro. to OOAD with Arch. Design, Part 1, CS577a, Fall 2005s
23
10/9/2006



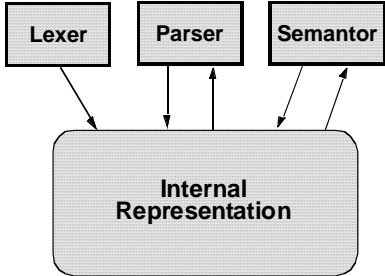
University of Southern California  
Center for Software Engineering

## Example Architecture — Compiler

**Sequential**



**Parallel**



Intro. to OOAD with Arch. Design, Part 1, CS577a, Fall 2005s
24
10/9/2006

USC  
CISE University of Southern California  
Center for Software Engineering

## Example Architecture — Video Game

The diagram illustrates the architecture of a video game. At the top, there are five ADT (Abstract Data Type) components: Clock ADT, Status ADT, Chute ADT, Well ADT, and Palette ADT. Below these are three logic components: Next Tile Placing Logic, Tile Match Logic, and Relative Pos Logic. A Status Logic component is positioned below the logic layer. The next layer consists of four artist components: Status Artist, Well Artist, Chute Artist, and Palette Artist. These are followed by a Tile Artist component, then a Layout Manager, and finally Graphics Binding at the bottom. To the right of the diagram is a screenshot of the game 'KLAX', showing a grid of colored tiles, a score of 105, and 3 lives remaining.


Intro. to OOAD with Arch. Design, Part 1, CS577a, Fall 2005s 25 10/9/2006

USC  
CISE University of Southern California  
Center for Software Engineering

## System Design

- Describes how system (or portions) can be implemented
- Describes specific technology solutions that satisfy project & system requirements
- 2 abstractions
  - Platform/Technology–Independent
  - Platform/Technology–Specific

Intro. to OOAD with Arch. Design, Part 1, CS577a, Fall 2005s 26 10/9/2006




University of Southern California  
Center for Software Engineering

---

## Platform/Technology-Independent TIM

- ❑ Defines
  - Top-level Components of system
  - What are each component's responsibilities
  - How they collaborate
    - Work together
    - Communicate
- ❑ Resolves high-level system issues

Intro. to OOAD with Arch. Design, Part 1, CS577a, Fall 2005s 27 10/9/2006




University of Southern California  
Center for Software Engineering

---

## Platform/Technology-Specific Design

- ❑ Resolves implementation considerations
  - e.g.
    - Use of databases, web-servers, hardware
    - Critical algorithms, sequence, significant events
    - GUI's
    - Implementation technology

Intro. to OOAD with Arch. Design, Part 1, CS577a, Fall 2005s 28 10/9/2006



University of Southern California  
Center for Software Engineering


---

## Sample Considerations for Developing Technology-Independent &/or Specific Design

- ❑ TID only if
  - Tools automate generation of TSD or implementation
  - TSD is totally COTS
- ❑ TSD only if
  - Low likelihood of porting system to different technology in future
  - Not creating product-line
  - No tool generation from
  - Small & short-life product
  - Tool generation from “TSD”  
e.g. Web-site generator

- ❑ TID & TSD if
  - Product is likely to
    - Be ported to different platforms/technology
    - Be implemented using mix technology  
e.g. Mac OS, Windows, Linux
    - Need to integrate different technology
  - Technology may change during development
  - Specific technology needs to be
    - Identified
    - Implemented

Intro. to OOAD with Arch. Design, Part 1, CS577a, Fall 2005s
29
10/9/2006




University of Southern California  
Center for Software Engineering

---

## Outline

- ❑ What is an *architecture*?
  - And related concepts
- ❑ **Technology-Independent Architecture**
- ❑ Technology-Specific Architecture
- ❑ Milestone Expectations
- ❑ Summary & Pre-class Exercise

Intro. to OOAD with Arch. Design, Part 1, CS577a, Fall 2005s
30
10/9/2006




University of Southern California  
Center for Software Engineering

## Define Architecture – LCO

- ❑ Purpose:
  - Define feasible architecture for system
    - Based on experience gained from similar systems or in similar problem domains
  - Identify Components
  - Understand
    - Hardware execution environment
    - Allocation of components to hardware
- ❑ Inputs:
  - Proposed System Analysis
    - Artifact & Information Model
    - Behavior Model
    - Architectural Patterns
    - L.O.S Requirements
    - Evolution Requirements

Intro. to OOAD with Arch. Design, Part 1, CS577a, Fall 2005s 31 10/9/2006




University of Southern California  
Center for Software Engineering

## Define Architecture – LCO Artifacts

- ❑ Component Classifier Model
  - Identify HW & SW component classifiers
    - In UML HW components are called *nodes*
  - Assign capabilities to components
  - Identify dependencies between components
- ❑ **System Deployment Model**
  - Describe configuration of HW nodes
  - Allocate SW component instances to HW nodes

❑ In CS577,  
only  
Deployment  
Model is  
required

Intro. to OOAD with Arch. Design, Part 1, CS577a, Fall 2005s 32 10/9/2006




University of Southern California  
Center for Software Engineering

## Define Architecture – LCO Component Model

- Start with proposed System Context
- Common components
  - Clients
  - Servers
  - Database Management Systems (DBMS)
  - Filters
- Identify dependencies

Intro. to OOAD with Arch. Design, Part 1, CS577a, Fall 2005s 33 10/9/2006




University of Southern California  
Center for Software Engineering

## Define Architecture – LCO Component Model (cont.)

- Look for Components
  - Loci of computation & state
  - **Modular, deployable, & replaceable** part of system
    - i.e. can be independently
      - Ordered, configured, or delivered
      - Developed, as long as the interfaces remain unchanged
      - Deployed across a set of distributed computational nodes
      - Changed without breaking other parts of the systems
  - Units which can provide restricted security over key resources
  - Existing products or external systems in design

Intro. to OOAD with Arch. Design, Part 1, CS577a, Fall 2005s 34 10/9/2006




University of Southern California  
Center for Software Engineering

## Define Architecture – LCO Component Model (cont.)

- Hints from Information Model
  - Look for optional groups
    - i.e. group of classes that represent optional behavior
  - Enclose in subsystem
    - Features which may be removed, upgraded, or replaced with alternatives should be considered independent

Intro. to OOAD with Arch. Design, Part 1, CS577a, Fall 2005s 35 10/9/2006




University of Southern California  
Center for Software Engineering

## Define Architecture – LCO Component Model (cont.)

- Base on past experiences
  - General
    - Buschman, F., R. Meunier, et al. (1996). [A System of Patterns: Pattern-oriented System Architecture](#). John Wiley & Sons, Inc.
    - Fowler, M. (1997). [Analysis Patterns: Reusable Object Models](#). Addison Wesley Longman, Inc.
    - Shaw, M. and D. Garlan (1996). [Software Architecture: Perspectives on an Emerging Discipline](#). Prentice-Hall Inc.
    - Szyperski, C. (1997). [Component Software: Beyond Object-Oriented Programming](#). ACM Press Books.
  - Domain-specific books & papers
  - Technology-specific
    - During design

Intro. to OOAD with Arch. Design, Part 1, CS577a, Fall 2005s 36 10/9/2006




University of Southern California  
Center for Software Engineering

## Define Architecture – LCO

### Considerations for E-Business Systems

- Existing network logical & physical design
- Existing databases & database design
- Existing Web environment
  - Servers, firewalls etc.
- Existing server environment
  - Configuration, software versions, planned upgrades
- Existing standards
  - Network, naming, protocols etc.

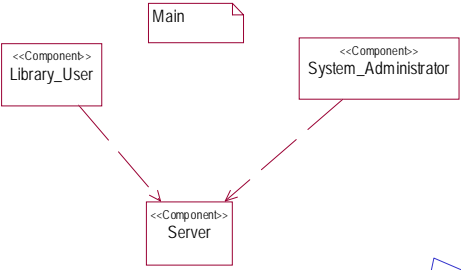
Intro. to OOAD with Arch. Design, Part 1, CS577a, Fall 2005s
37
10/9/2006



University of Southern California  
Center for Software Engineering

## Define Architecture – LCO

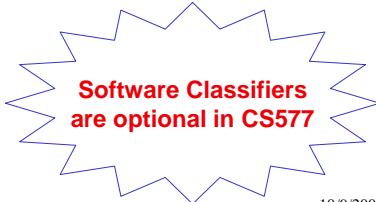
### SW Component Classifier Model For Full-text Title Database System



```


classDiagram
    class Library_User["<<Component>> Library_User"]
    class System_Administrator["<<Component>> System_Administrator"]
    class Server["<<Component>> Server"]
    Library_User --> Server
    System_Administrator --> Server
            
```

- Decision
  - Client-Server Model



Software Classifiers  
are optional in CS577


Intro. to OOAD with Arch. Design, Part 1, CS577a, Fall 2005s
38
10/9/2006



University of Southern California  
Center for Software Engineering


## Define Architecture – LCO

### HW Component Classifier for Full-Text Title Database System



```

classDiagram
    class Workstation["<<node>> Workstation"]
    class UnixServer["<<node>> Unix Server"]
    Workstation "0..*" -- "1" UnixServer : Network
    Workstation --> "client"
    UnixServer --> "server"
  
```




Hardware Classifiers  
are optional in CS577

Intro. to OOAD with Arch. Design, Part 1, CS577a, Fall 2005s

39

10/9/2006



University of Southern California  
Center for Software Engineering

## Define Architecture – LCO

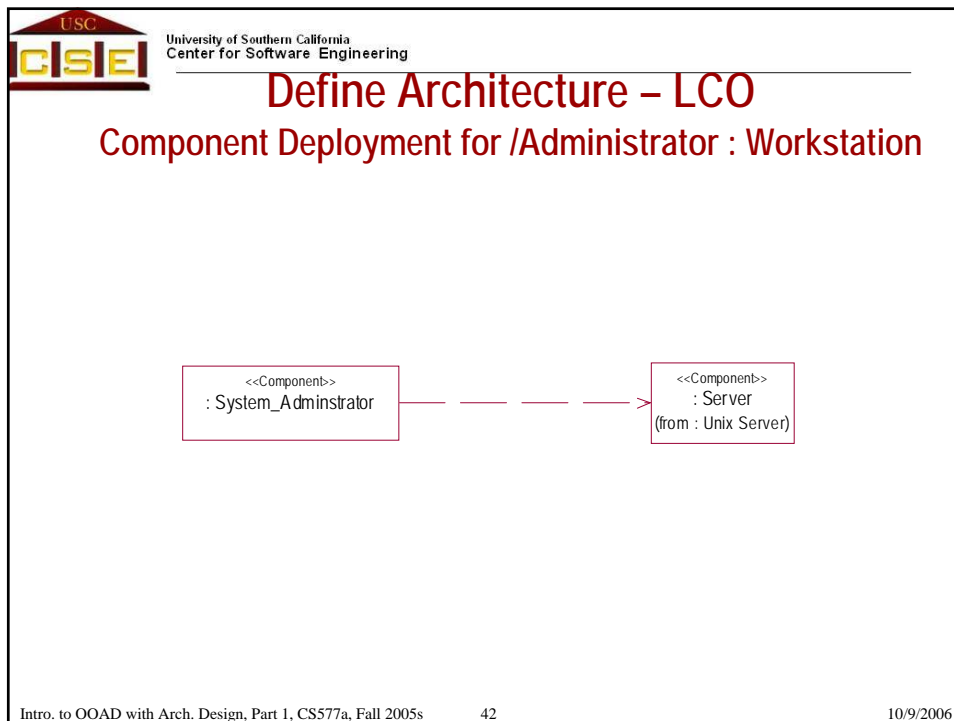
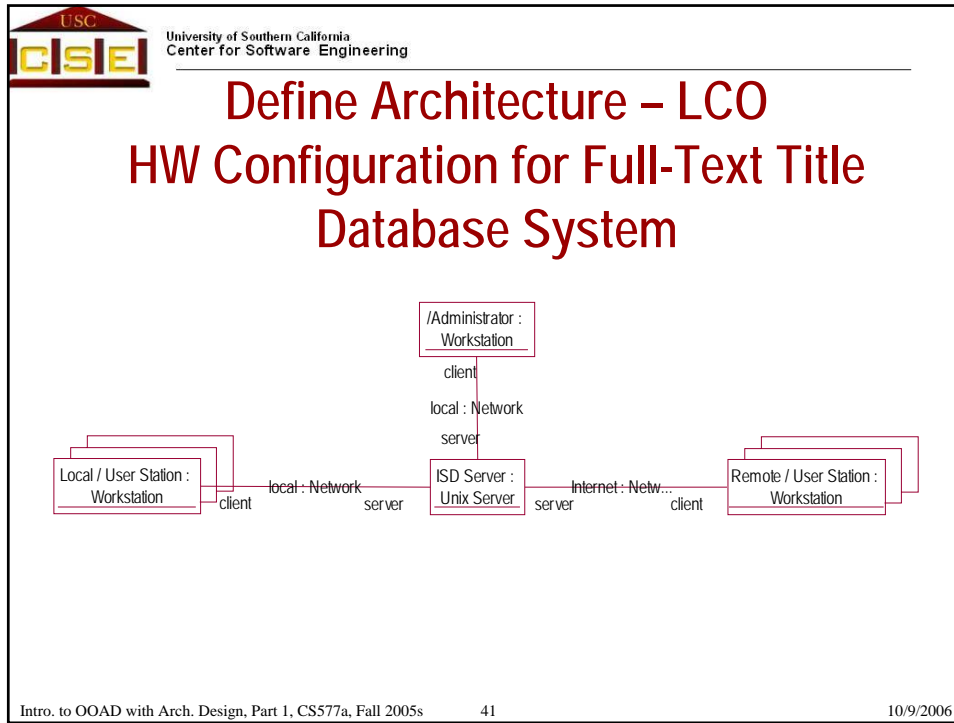
### System Deployment Model


- Goal
  - Understand
    - Execution environment
    - Distribution of software components across environment
- Model describes
  - Configuration(s) of hardware node instances
  - Allocation of SW component instances to nodes

Intro. to OOAD with Arch. Design, Part 1, CS577a, Fall 2005s

40


10/9/2006



 University of Southern California  
Center for Software Engineering


## Define Architecture – LCO

### Component Deployment for / User Station : Workstation



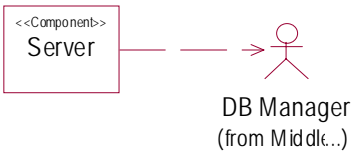
```
graph LR; L["<<Component><br>: Library_User"] -.-> S["<<Component><br>: Server<br>(from : Unix Server)"]
```

Intro. to OOAD with Arch. Design, Part 1, CS577a, Fall 2005s 43 10/9/2006

 University of Southern California  
Center for Software Engineering


## Define Architecture – LCO

### Component Deployment for ISD / Server : Unix Server



```
graph LR; S["<<Component><br>Server"] -.-> DM["DB Manager<br>(from Middle..)"]
```

Intro. to OOAD with Arch. Design, Part 1, CS577a, Fall 2005s 44 10/9/2006



University of Southern California  
Center for Software Engineering

---

## Outline

- What is an *architecture*?
  - And related concepts
- Technology-Independent Architecture
- Technology-Specific Architecture**
- Milestone Expectations
- Summary & Pre-class Exercise

Intro. to OOAD with Arch. Design, Part 1, CS577a, Fall 2005s 45 10/9/2006




University of Southern California  
Center for Software Engineering

---

## Technology-Specific Design Process

- Describe
  - Technology-specific* Architecture Structure
    - *Technology-specific* Information Maintained, Inspected, Produced
    - *Technology-specific* Behavior of Architecture & Components
    - *Technology-specific* L.o.S. Allocation
- Update TIM Models
- Validate correctness

Intro. to OOAD with Arch. Design, Part 1, CS577a, Fall 2005s 46 10/9/2006




University of Southern California  
Center for Software Engineering

---

## Refine Architecture – By LCA

- ❑ Purpose:
  - Incorporate implementation decisions into architecture for system
    - Based on experience gained from similar systems or in similar problem domains
  - Identify implementation-technology specific components
  - Understand
    - Hardware execution environment
    - Allocation of components to hardware
- ❑ Inputs:
  - TIM (LCA)
    - Architecture
    - Artifacts & Data Classes
    - Behavior Model
    - L.O.S Allocation
    - Architectural Patterns
    - Evolution Requirements

Intro. to OOAD with Arch. Design, Part 1, CS577a, Fall 2005s
47
10/9/2006



University of Southern California  
Center for Software Engineering


---

## Refine Architecture – By LCA Project TSM Artifacts

- ❑ **Component Model**
  - Define TSM Component Model
    - Determine whether components are implement by COTS or not
    - Determine interfaces of components
    - Define TSM-specific classes, object, interactions
  - Refine component classes
    - e.g. select specific implementation stereotypes
    - Add implementation specific components
  - Identify dependencies between components
- ❑ **System Topology**
  - Updated as appropriate to reflect changes to component model
- ❑ **System Deployment Model**
  - Updated as appropriate to reflect changes to component model

❑ In CS577,  
only  
Deployment  
Model is  
required

Intro. to OOAD with Arch. Design, Part 1, CS577a, Fall 2005s
48
10/9/2006




University of Southern California  
Center for Software Engineering

## Refine Architecture – By LCA TSM Component Model

- ❑ Purpose:
  - Describe how components will be implemented
    - Subcomponents, objects, classes, functions used
    - COTS products used & how they are configured
  - Define their interfaces
  - Identify development technologies to be used
    - Including database tables, Java, XML/HTML, HTTP servers, API's, class libraries, design patterns
- ❑ Inputs:
  - See revised component model
- ❑ Artifacts
  - Revised Component Model
  - Interface Class Diagram(s)
  - Implementation Class Model
  - Description of COTS configuration
    - See *MBASE COTS Integration Supplement*

Intro. to OOAD with Arch. Design, Part 1, CS577a, Fall 2005s 49 10/9/2006




University of Southern California  
Center for Software Engineering

## TSM Component Model Full-Text Title Database System Decisions

- ❑ Document in SSAD
  - Library\_User Component
    - Will implement with web-browser that access pages generated by Java Server Pages
  - System\_Administrator Component
    - Will implement with web-browser that access pages generated by Java Server Pages
  - Server Component
    - Server::Business Objects
    - Server::Database
      - Will use MySQL as DBMS
    - Will implement classes in Java
- ❑ (Check MBASE Guide & *MBASE COTS Integration Supplement* for details of section)

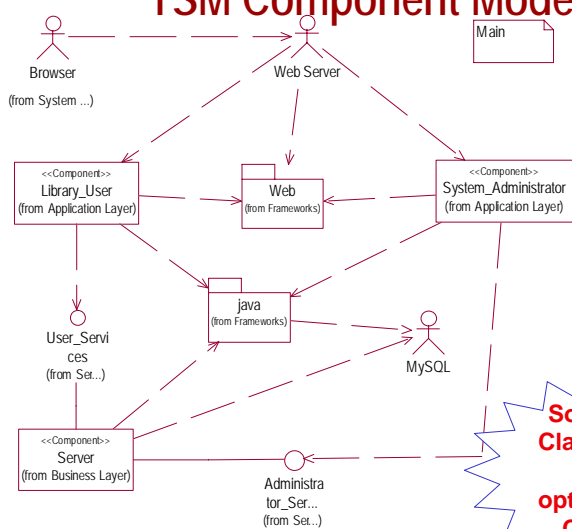
Intro. to OOAD with Arch. Design, Part 1, CS577a, Fall 2005s 50 10/9/2006



University of Southern California  
Center for Software Engineering

## Refine Architecture - By LCA

### TSM Component Model For FTTD




**Software Classifiers are optional in CS577**

❑ Revised to show any changes due to implementation decisions

- Use of COTS
  - Web Browser
  - Web Server
  - MySQL
- Dependency on
  - Java Framework
  - Web Framework

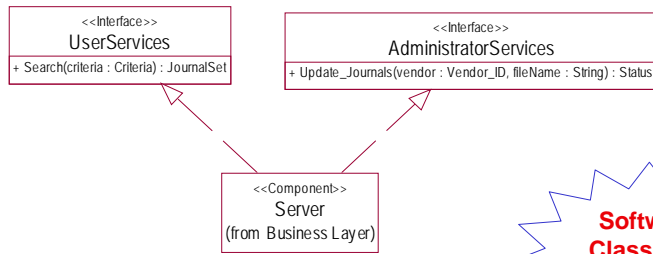
Intro. to OOAD with Arch. Design, Part 1, CS577a, Fall 2005s
51
10/9/2006



University of Southern California  
Center for Software Engineering

## Refine Architecture - By LCA

### Define Interfaces For Server Component




**Software Classifiers are optional in CS577**

❑ Define *interfaces* for Component

- Define operations available in each *interface*
- Define dependencies on other classes & *interfaces*

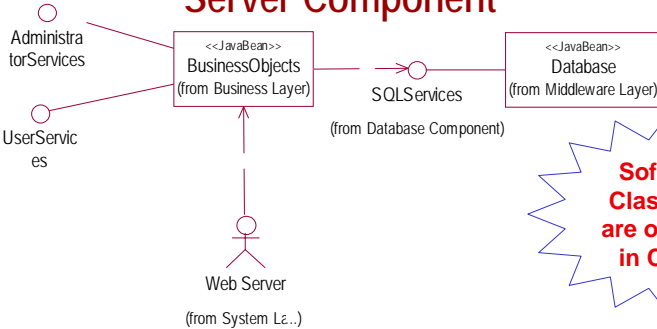
❑ Note: deferring complete AdministratorServices *interface* until next build

Intro. to OOAD with Arch. Design, Part 1, CS577a, Fall 2005s
52
10/9/2006



University of Southern California  
Center for Software Engineering


### Refine Architecture - By LCA TSM Component Model For Server Component



Software Classifiers are optional in CS577

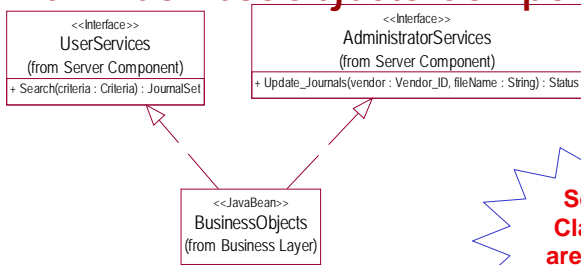
- ❑ Create/Revise to show any changes due to implementation decisions
  - New Database (Sub-)Component (Java Bean)
    - Interface named SQLServices supplied by Database component
  - New BusinessObjects (Sub-)Component (Java Bean)
    - Decouple higher-level components from decisions about database (e.g. SQL)
    - Realizes AdministratorServices & UserServices interfaces

Intro. to OOAD with Arch. Design, Part 1, CS577a, Fall 2005s
53
10/9/2006



University of Southern California  
Center for Software Engineering


### Refine Architecture - By LCA Define Component Interfaces For Server::BusinessObjects Component



Software Classifiers are optional in CS577

- ❑ Define *interfaces* for Component
  - Define operations available in each *interface*
  - Define dependencies on other classes & *interfaces*
- ❑ Notes:
  - Deferring complete AdministratorServices *interface* to later build
  - Diagram is shown for completeness
    - Information already resented in Component Diagram for Server


Intro. to OOAD with Arch. Design, Part 1, CS577a, Fall 2005s
54
10/9/2006




University of Southern California  
 Center for Software Engineering

## Refine Architecture – By LCA TSM Deployment Model

❑ /Administrator : Workstation




❑ /User : Workstation



❑ Define how software components are allocated to hardware nodes

- May not need if Implementation Deployment Model is same as Architecture Deployment Model

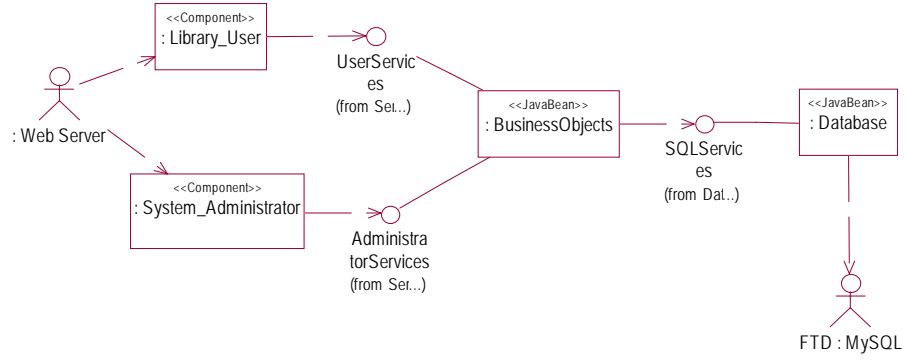
Intro. to OOAD with Arch. Design, Part 1, CS577a, Fall 2005s
55
10/9/2006




University of Southern California  
 Center for Software Engineering

## Refine Architecture Model – By LCA TSM Deployment Model (cont.)

❑ : Unix Server



Intro. to OOAD with Arch. Design, Part 1, CS577a, Fall 2005s
56
10/9/2006




University of Southern California  
Center for Software Engineering

---

## Outline

- What is an *architecture*?
  - And related concepts
- Technology-Independent Architecture
- Technology-Specific Architecture
- Milestone Expectations**
- Summary & Pre-class Exercise

Intro. to OOAD with Arch. Design, Part 1, CS577a, Fall 2005s 57 10/9/2006




University of Southern California  
Center for Software Engineering

---

## Life-Cycle Objectives (LCO) Goal

- Convince customer & other reviewers that you have
  - *Reasonable* understanding of
    - Customer needs (goals & constraint)
      - What customer &/or users want system to do
      - Environment in which system will operate
      - Development constraints
    - Your development capabilities & constraints
  - *Reasonably* defined set of requirements
    - Based on negotiated win-win conditions
  - *Feasible*
    - TIM
    - Development plan




University of Southern California  
Center for Software Engineering

## Life-Cycle Objectives (LCO) Goal What's "Reasonable" mean?

- ❑ Taken from U.S./English Law: "Reasonable Person Standard"
  - Would a person
    - With comparable ability
    - Facing same circumstances
    - Knowing what you know
    - Be content that thing proposed is suitable
  - Generally comes down to a jury of peers
    - e.g. *Architecture Review Board*

Intro. to OOAD with Arch. Design, Part 1, CS577a, Fall 2005s 59 10/9/2006




University of Southern California  
Center for Software Engineering

## Life-Cycle Objectives (LCO) Goal What's "Feasible Architecture"?

- ❑ Architecture described in **sufficient** detail that reasonable technical reviewer would say
  - Likely to satisfy client's need
    - i.e. satisfying current stated requirements
  - Likely to be developable as planned
- ❑ What's "sufficient" depends on risks & benefits
- ❑ Need evidence to convince reviewers
  - Models, prototypes, simulations, schedules, plans, etc.

Intro. to OOAD with Arch. Design, Part 1, CS577a, Fall 2005s 60 10/9/2006




University of Southern California  
Center for Software Engineering

---

## Life-Cycle Objectives (LCO) Guidelines

- General
  - Less structured
  - Focus on strategy or “vision”
    - e.g., Operational Concept Description & Life Cycle Plan
  - May have some mismatches
    - Indicates unresolved issues or items
  - No need for complete forward & backward traceability
  - May have possible or potential elements
    - e.g., Data, Components, ...
  - Some sections could be left as TBD
    - Particularly Construction, Transition, and Support plans

Intro. to OOAD with Arch. Design, Part 1, CS577a, Fall 2005s 61 10/9/2006




University of Southern California  
Center for Software Engineering

---

## Life-Cycle Objectives (LCO) Guidelines for SSAD

- Activities focus on
  - System Analysis
  - Technology–Independent Design or Technology–Specific Design
    - Feasible architecture
    - High–risk or complex behaviors defined

Intro. to OOAD with Arch. Design, Part 1, CS577a, Fall 2005s 62 10/9/2006



University of Southern California  
Center for Software Engineering

---

## Life-Cycle Objectives (LCO) Guidelines for SSAD Documentation

**□ Model of**

- System Analysis
- Technology–Independent &/or  
Technology–Specific Design


**□ MS Word (or equivalent) document**

- Explanation of system analysis
  - Key concepts
  - Refinements relative to OCD
  - Technical Rationale

**□ MS Word document (cont.)**

- Explanation of design
  - Key concepts
    - e.g.
      - » TIM, TSM, or both
      - » Top-level architectural elements
  - Key design decisions
    - e.g.
      - » Selected Deployment
      - » Use of *Patterns, Styles, & Frameworks*
      - » Use of *COTS* (see COTS guide)
  - Technical Rationale
    - Why design decisions?
      - inc. How requirements are satisfied

Intro. to OOAD with Arch. Design, Part 1, CS577a, Fall 2005s
63
10/9/2006



University of Southern California  
Center for Software Engineering


---

## Life-Cycle Architecture (LCA) Goal

**□ Convince customer & other reviewers that you have**

- *Reasonable* understanding of
  - Customer needs (goals & constraint)
    - What customer &/or users want system to do
    - Environment in which system will operate
    - Development constraints
  - Your development capabilities & constraints
- *Reasonably* well–defined set of requirements
  - Based on negotiated win-win conditions
- *Reasonable* Life-Cycle TIM
- *Feasible* Development plan

Intro. to OOAD with Arch. Design, Part 1, CS577a, Fall 2005s
64
10/9/2006




University of Southern California  
Center for Software Engineering

## Life-Cycle Architecture (LCA) Goal

### What's a "Life-cycle Architecture"?

- ❑ Architecture described in sufficient detail that a *reasonable* technical reviewer would confidently say
  - Will satisfy client's need
    - i.e. satisfying current stated goals
  - For expected duration of client's need
    - Able to support any likely future needs that can be predicted
    - Often called "life time" or "life cycle" of system
  - Developable as planned
- ❑ What's "sufficient" depends on risks & benefits
- ❑ Need evidence to convince reviewers
  - Models, prototypes, simulations, schedules, plans, etc.

Intro. to OOAD with Arch. Design, Part 1, CS577a, Fall 2005s 65 10/9/2006




University of Southern California  
Center for Software Engineering

## Life-Cycle Architecture (LCA) Guidelines

- ❑ General
  - More formal
  - Solid tracing upward & downward
  - No major unresolved issues or items
  - Closure mechanisms identified for any unresolved issues or items
    - e.g., "detailed data entry capabilities will be specified once the Library chooses a Forms Management package on February 15"
  - No TBDs expect possibly within Construction, Transition, & Support plans
  - Basic elements from Life Cycle Plan are indicated within Construction, Transition, & Support plans
  - No "possible" or "potential" elements
    - e.g., Entities, Components, ...
  - No more superfluous, unreferenced items
    - Each element should reference or be referenced by another element
    - Elements not referenced should be eliminated or documented as irrelevant

Intro. to OOAD with Arch. Design, Part 1, CS577a, Fall 2005s 66 10/9/2006

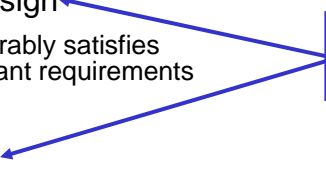


University of Southern California  
Center for Software Engineering


## Life-Cycle Architecture (LCA) Guidelines for SSAD

- ❑ System Analysis
  - System description demonstrably satisfies high-risk, architecturally-significant requirements
- ❑ Technology-Independent Design
  - Life-cycle architecture demonstrably satisfies high-risk, architecturally-significant requirements that's technology-independent
- ❑ Technology-Specific Design
  - Life-cycle architecture demonstrably satisfies high-risk, architecturally-significant requirements that's technology-specific

1 or other  
in CS577



Intro. to OOAD with Arch. Design, Part 1, CS577a, Fall 2005s
67
10/9/2006




University of Southern California  
Center for Software Engineering

## Initial Operating Capability (IOC) Goal

- ❑ Implemented subset of capabilities
  - Implemented applicable portions of architecture
    - For all capabilities in iteration
      - Design of objects & classes in components that implement capabilities
  - Using specific implementation technology
    - Implementation models sufficient to code
- ❑ Stable Technology-Specific Architecture
  - Implementation demonstrates feasibility

Intro. to OOAD with Arch. Design, Part 1, CS577a, Fall 2005s
68
10/9/2006



University of Southern California  
Center for Software Engineering


## Initial Operating Capability (IOC) Guidelines

- ❑ General
  - Complete tracings among models & delivered software
    - e.g. comments in code trace to SSAD design elements
  - MBASE models consistent with delivered system
    - e.g. “as built” OCD, SSRD, SSAD, etc. models
    - Not necessarily complete
  - Core system capability requirements have been implemented & tested
  - At least one construction interaction
  - Complete set of CTS plans & reports consistent with development

Intro. to OOAD with Arch. Design, Part 1, CS577a, Fall 2005s

69

10/9/2006



University of Southern California  
Center for Software Engineering

## Initial Operating Capability (IOC) Guidelines for SSAD


- ❑ System Analysis
  - System description demonstrably satisfies requirements
- ❑ Technology-Independent Design (TID)
  - Life-cycle architecture demonstrably satisfies requirements
  - Complete for portions implement
- ❑ Technology-Specific Design (TSD)
  - Technology-specific Life-cycle architecture demonstrably satisfies requirements
  - Complete for portions implement

1 or other  
in CS577

Intro. to OOAD with Arch. Design, Part 1, CS577a, Fall 2005s

70

10/9/2006




University of Southern California  
Center for Software Engineering

---

## Outline

- What is an *architecture*?
  - And related concepts
- Technology-Independent Architecture
- Technology-Specific Architecture
- Milestone Expectations
- Summary & Pre-class Exercise**

Intro. to OOAD with Arch. Design, Part 1, CS577a, Fall 2005s 71 10/9/2006



University of Southern California  
Center for Software Engineering

---

## Pre-Class Exercise

- Create a Deployment Diagram for the Bookstore Example described in the Simple OOAD lectures
- Explain why you chose this design

Intro. to OOAD with Arch. Design, Part 1, CS577a, Fall 2005s 72 10/9/2006