


**OO Design & Development III & IV:
Implementation
Modeling
CS577b, Fall 2001**

Ed Colbert
USC Center for Software
Engineering

1 2/26/2002



Goal of Presentation

- Understand how to perform Object-Oriented Design
 - Using
 - MBASE
 - Object-oriented techniques
 - RUP
 - Rational Rose
- Understand how to document analysis
- Presentation is part 3 & 4 of 4 lectures on Detailed Design

OO D&D III & IV, CS577b, Spring 2002 2 2/26/2002

Outline

- **When Last We Met ...**
- Design Process Overview
- Design Process by Example

Life Cycle Objectives (LCO) Guidelines

- **General**
 - Less structured, with information moving around
 - Focus on the strategy or "vision"
 - e.g., for Operational Concept Description & Life Cycle Plan
 - May have some mismatches
 - indicating unresolved issues or items
 - No need for complete forward & backward traceability
 - May still include "possible" or "potential" elements
 - e.g., Entities, Components, ...
 - Some sections could be left as TBD
 - Particularly Construction, Transition, and Support plans
- **System Analysis/Design**
 - Focus on
 - High-level architecture
 - High-risk or complex behaviors

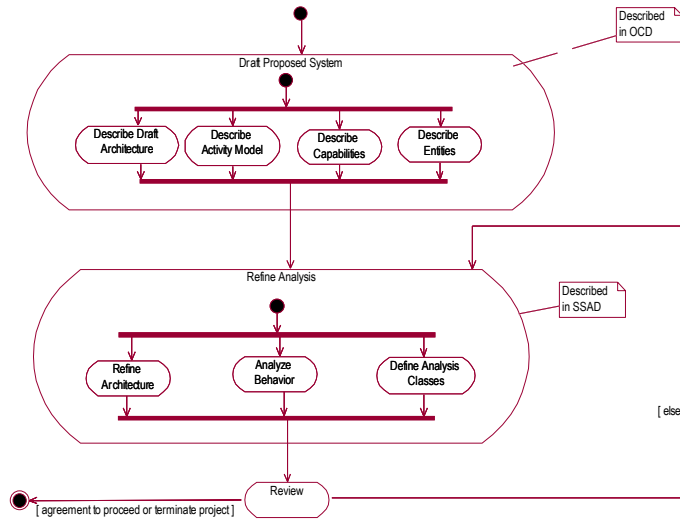
Life Cycle Architecture (LCA) Guidelines

- 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
- System Analysis/Design
 - Stable Architecture
 - Design of objects & classes that implement high-risk or complex behaviors

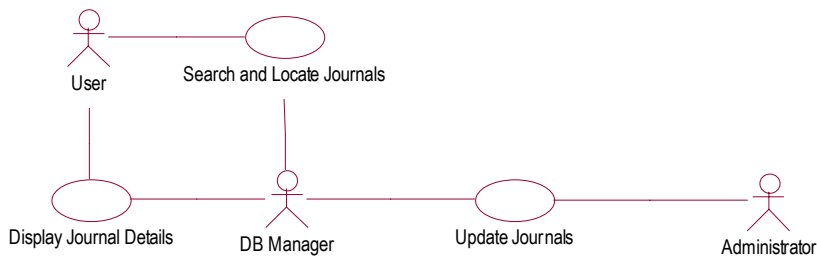
Initial Operating Capability (IOC) Guidelines

- 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
- System Analysis/Design
 - Stable Architecture
 - For all capabilities in iteration
 - Design of objects & classes that implement capabilities
 - Implementation models sufficient to code

System Analysis Process Overview



Analyze Behavior – LCO Use-Case Diagram Example



Analyze Behavior – LCO Use-Case Description Example 1

Use-Case Name	Full-text Journal Title Search
Abstract	
Purpose	To allow a user to search for journals to which USC Libraries subscribes by keyword
Actors	User, DB Manager
Importance	Primary
Requirements	Full-text Journal Title Search
Risks	
High-Risk?	No
Architecturally Significant?	Yes
Development Status	Draft LCO
Overview	User enters search criteria and system returns lists of journals matching criteria
User Interface	
Pre-conditions	Database has been initialized
Post-conditions	Displayed List of all the complete journal titles containing the user's search criteria
Includes	
Extension Points	

Analyze Behavior – LCO Use-Case Description Example 1 (cont.)

■ Typical Course of Action

Seq. #	Actor Actions	System Response
1.	User requests search	
2.		Displays search page
2	User enters search criteria	
3		Queries the database asking for journal titles that match the user's search criteria
4		Displaying the journal list in search result page

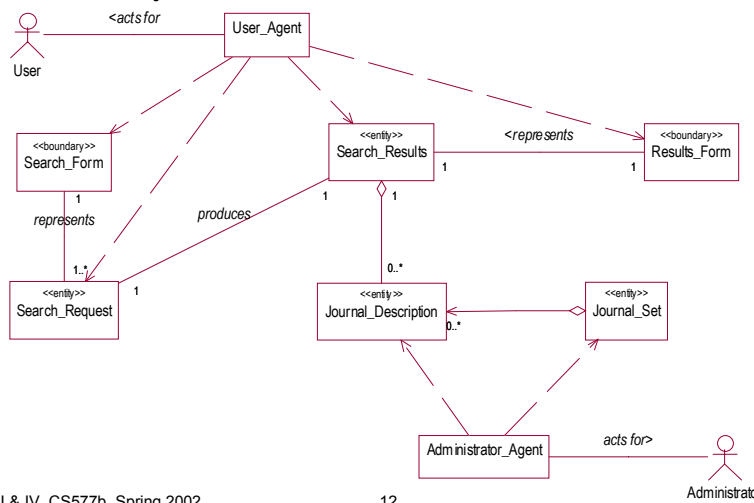
Analyze Behavior – LCO Use-Case Description Example 1 (cont.)

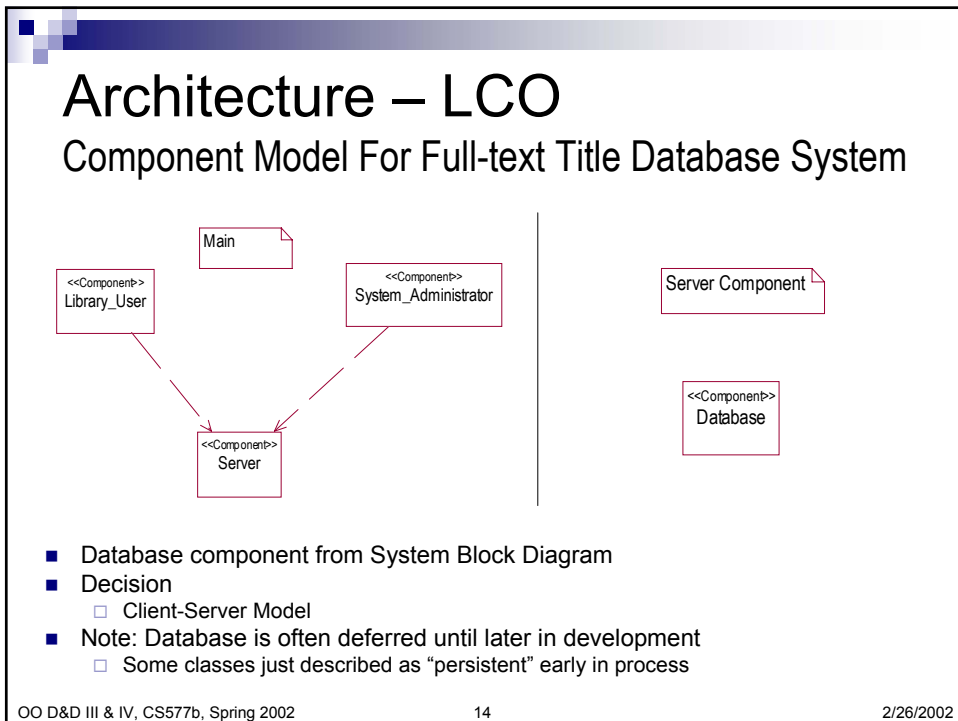
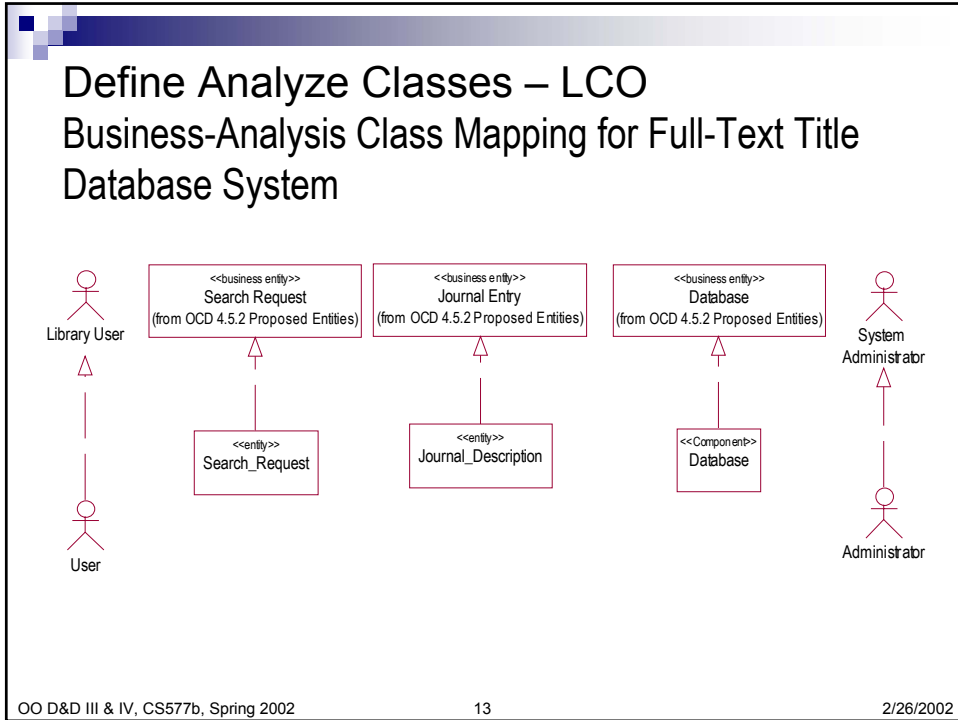
- Alternate Course of Action: No results match search criterion

Seq. #	Actor Actions	System Response
4.		Display error page that asks user to search again

- Exceptional Course of Action: None

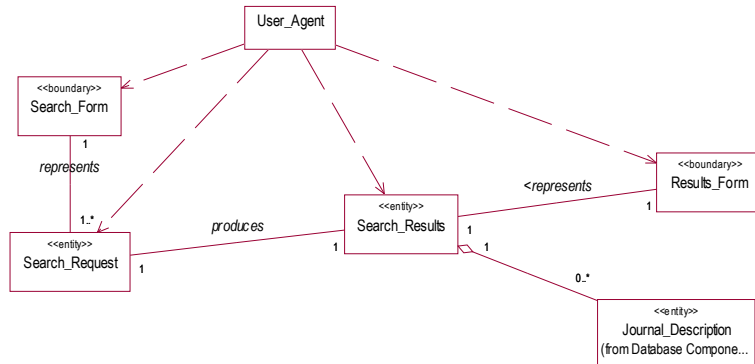
Define Analyze Classes – LCO Enterprise Classification Model for Full-Text Title Database System





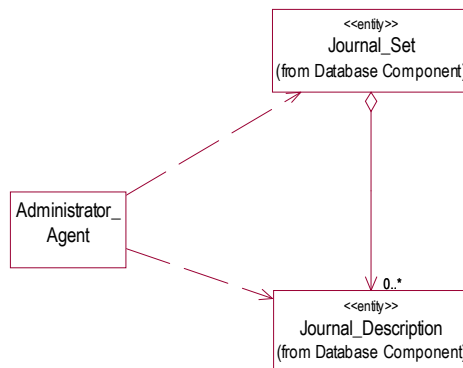
Refine Architecture – LCO

Logical Class Model for Library_User Component



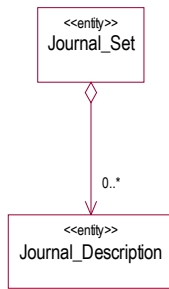
Refine Architecture – LCO

Logical Class Model for System_Administrator Component



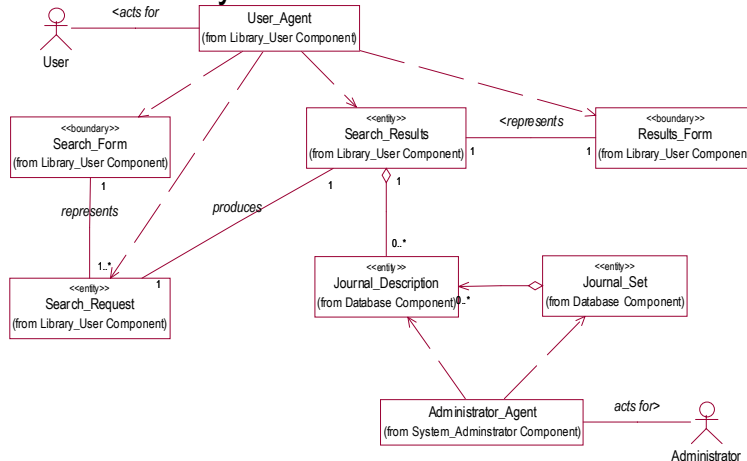
Refine Architecture – LCO

Logical Class Model for Server::Database Component



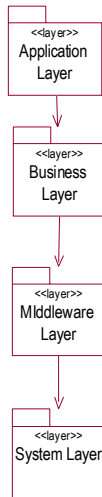
Refine Architecture – LCO

Updated Enterprise Classification Model for Full-Text Title Database System



Refine Architecture – LCO

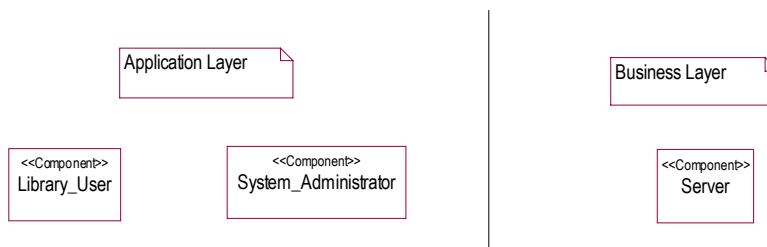
System Topology for Full-Text Title Database System



- Currently only have few classes
- But
 - We've just started
 - Planning for evolution

Refine Architecture – LCO

System Topology for Full-Text Title Database System



- Nothing in other layers
 - yet...

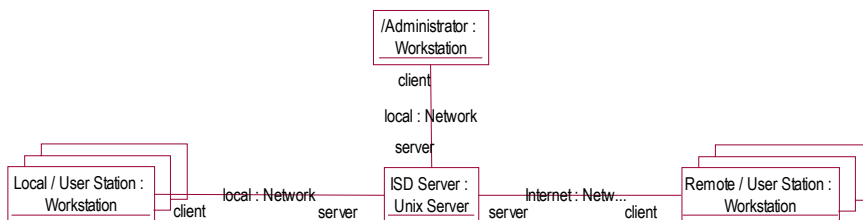
Refine Architecture – LCO

Node Classes for Full-Text Title Database System



Refine Architecture – LCO

Node Configuration for Full-Text Title Database System



Refine Architecture – LCO

Component Configuration for /Administrator :
Workstation



Refine Architecture – LCO

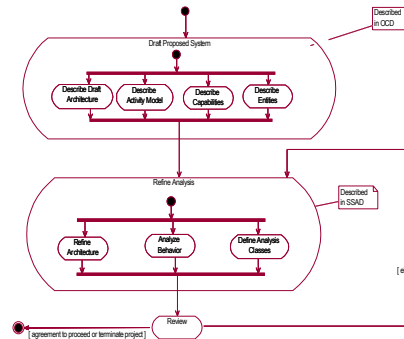
Component Configuration for / User Station :
Workstation



System Analysis

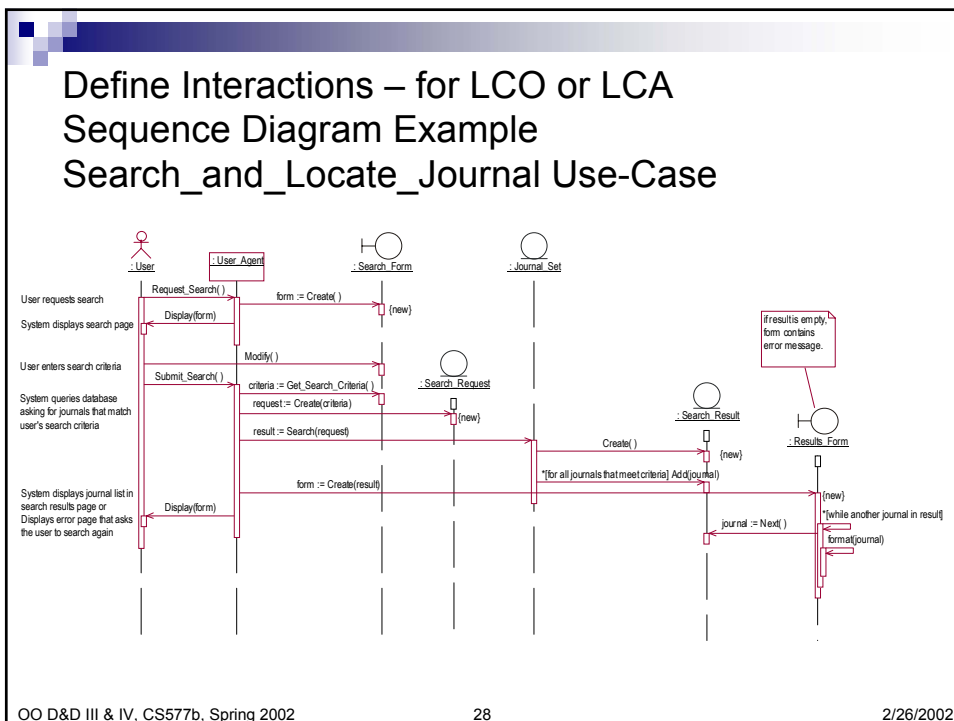
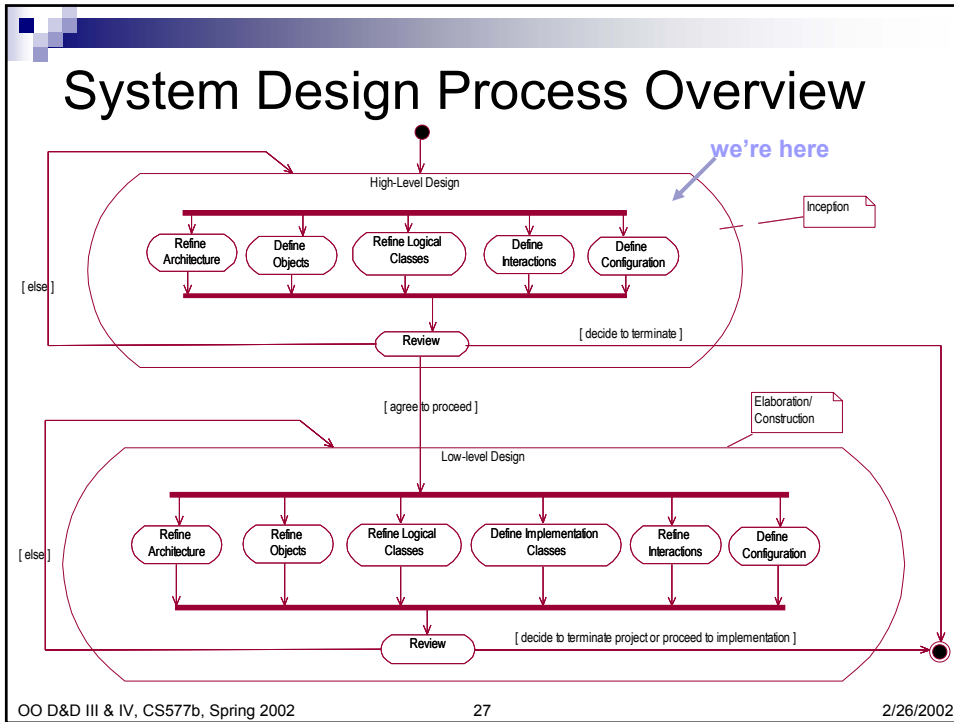
What Have We Done?

- Developed a preliminary
 - Architecture model for our system
 - Behavior Model of system
 - Classes model



System Design

- Describe on how system can be implemented in software
- Describes specific technology solutions that satisfy Project & System requirements
- 2-levels
 - High-level
 - Resolves Analysis issues
 - e.g. 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 & other constraint
 - Low-level
 - Implementation considerations
 - e.g. use of databases, web-servers, hardware, critical algorithms, sequence, significant events, GUI's, etc.



Define Interactions – LCO or LCA Operation Specification Form Journal_Set.Search

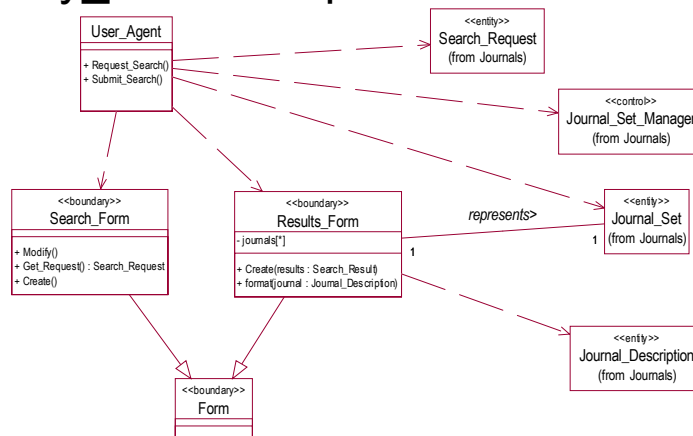
Identifier	Op10
Initiator	User_Agent
Passed parameters	request : Search_Request
Return values	Search_Result
Exception handling	N/A
Guards	N/A
Validation	N/A
Messages	a Request
Exits	
Constraints	Concurrent
Relates to	Search_and_Locate_Journal use-case

OO D&D III & IV, CS577b, Spring 2002

29

2/26/2002

Define Interactions – LCO or LCA Updated Logical Class Model For Library_User Component

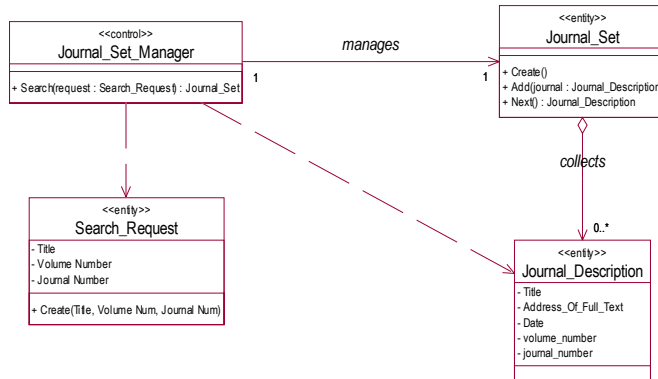


OO D&D III & IV, CS577b, Spring 2002

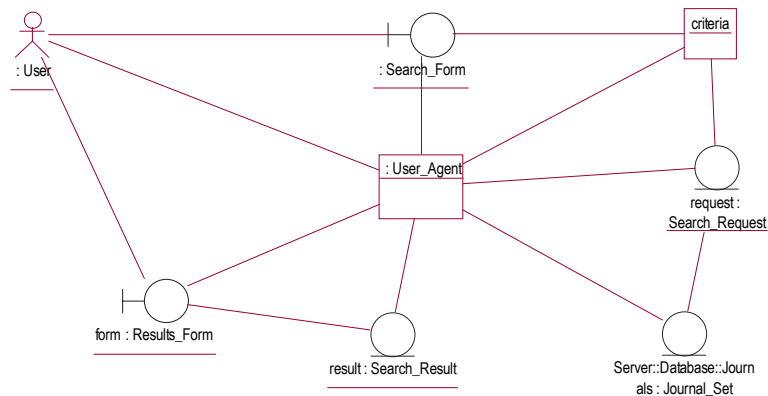
30

2/26/2002

Define Interactions – LCO or LCA Updated Logical Class Model For Server::Database Component



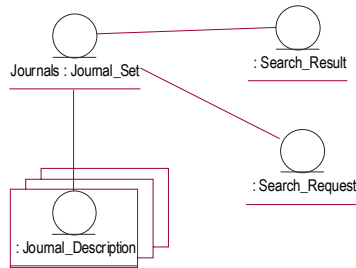
Define Objects – LCO or LCA Draft Object Model For Library_User Component



■ Based on Interaction Model

Define Objects – LCO or LCA

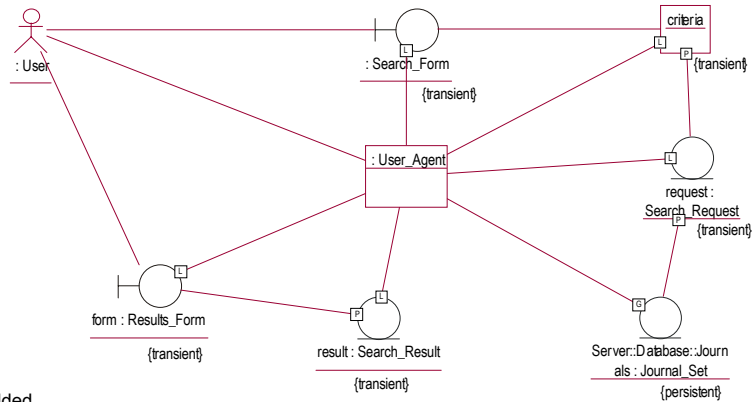
Draft Object Model For Server::Database Component



■ Based on Interaction Model

Define Objects – LCO or LCA

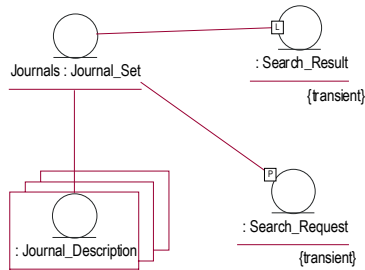
Refined Object Model For Library_User Component



- Added
 - Link "Implementation Stereotypes"
 - Note: Rational Rose
 - Uses letters "P", "L", "G" in box for «Parameter», «Local», «Global» respectively
 - Uses "loop-back" link for «Self»
 - No visible notation for «association»
 - "Persistence" properties that indicate instances which are created &/or destroyed

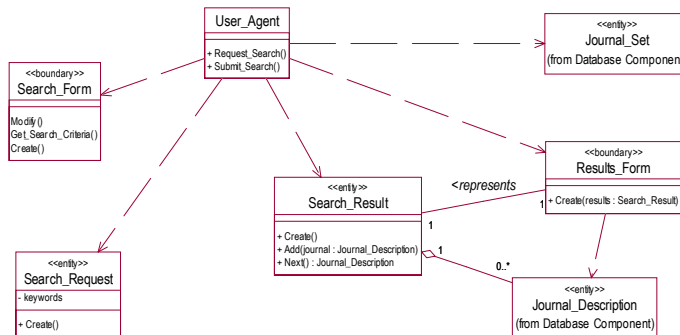
Define Objects – LCO or LCA

Refined Object Model For Server::Database Component



Define Objects – LCO or LCA

Updated Logical Class Model For Library_User Component

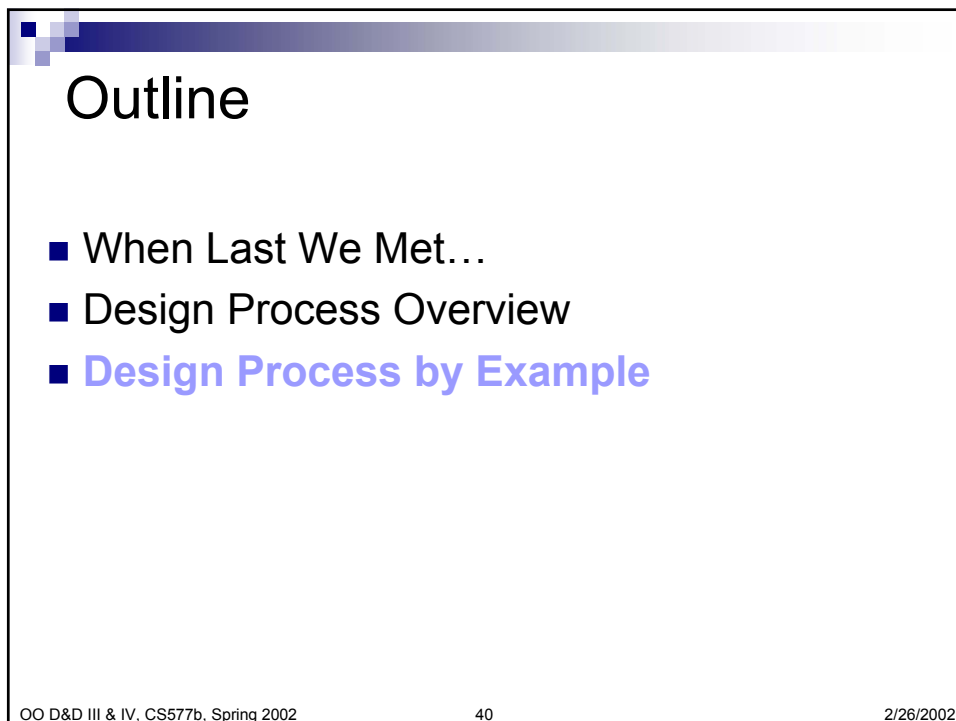
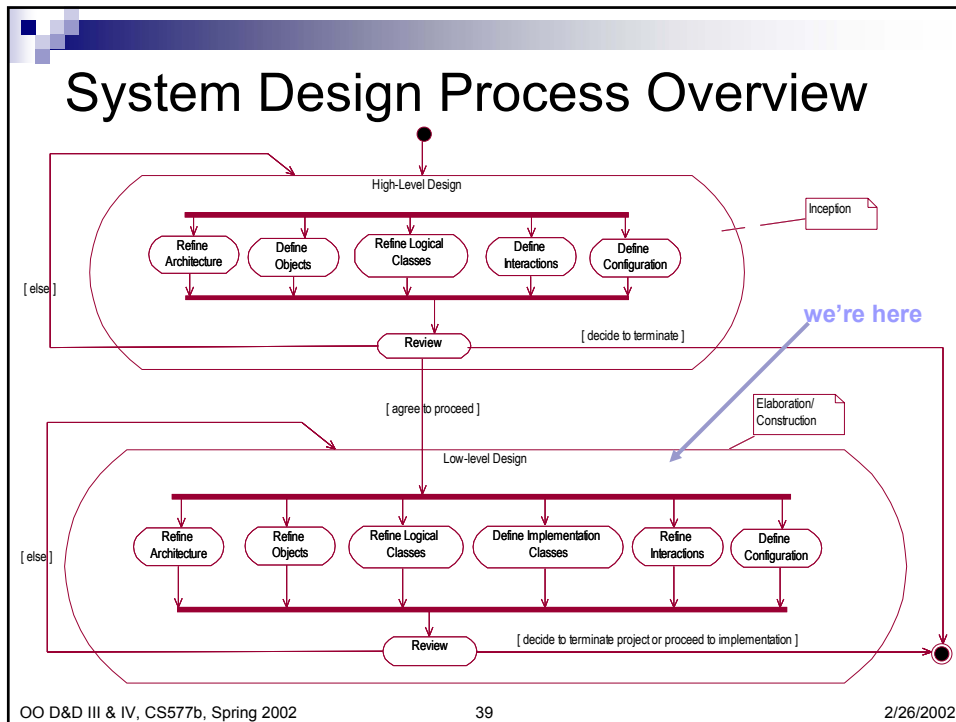


Outline

- When Last We Met...
- **Design Process Overview**
- Design Process by Example

System Design

- Describe on how system can be implemented in software
- Describes specific technology solutions that satisfy Project & System requirements
- 2-levels
 - High-level
 - Resolves Analysis issues
 - e.g. 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 & other constraint
 - Low-level
 - Implementation considerations
 - e.g. use of databases, web-servers, hardware, critical algorithms, sequence, significant events, GUI's, etc.



Refine Architecture – LCA or IOC

- Purpose:
 - Incorporate implementation decisions into 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:
 - LCO Architecture
 - LCO Logical Class Model
 - LCO Object Model
 - LCO Interaction Model
 - Architectural Patterns
 - L.O.S Requirements
 - Evolution Requirements

Refine Architecture – LCO or IOC Artifacts

- Component Model
 - Define Component-Implementation Model
 - Determine whether components are implement by COTS or not
 - Determine interfaces of components
 - Define implementation-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

Refine Architecture – LCO or IOC Component-Implementation Design Model

- Purpose:
 - Describe how the components will be implemented
 - Define their interfaces
 - Describe COTS products that will be used to implement components, and how they are configured
 - 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
 - Interface Class Diagram(s)
 - Implementation Class Model
 - Description of COTS configuration
 - See *MBASE COTS Integration Supplement*

Component-Implementation Design Model For Full-text Title Database System

- Document in SSAD 3.1.2.x
 - Library_User Component
 - Will be implemented with web-browser that access pages generated by Java Server Pages
 - System_Administrator Component
 - Will be implemented 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)

Refine Architecture – LCA or IOC

Component Model For Full-text Title Database System

- Revise to show any changes due to implementation decisions
 - Use of COTS
 - Web Browser
 - Web Server
 - MySQL
 - Dependency on
 - Java Framework
 - Web Framework

OO D&D III & IV, CS577b, Spring 2002 45 2/26/2002

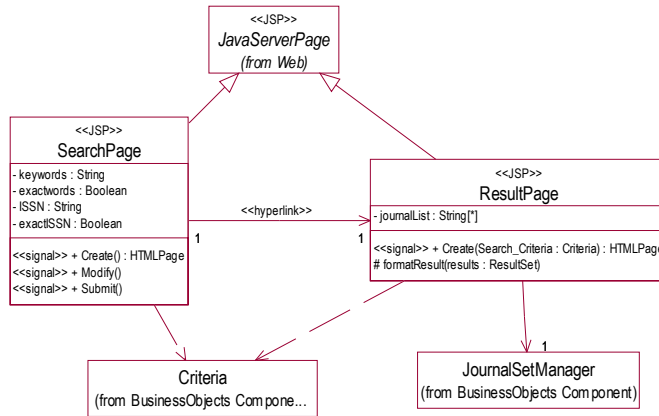
Define Implementation Class Model – LCO or LCA

For Library_User Component

- Define implementation of logical classes

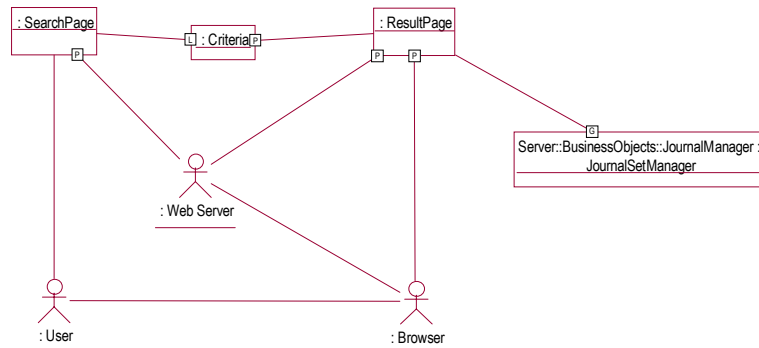
OO D&D III & IV, CS577b, Spring 2002 46 2/26/2002

Define Implementation Class Model – LCO or LCA For Library_User Component



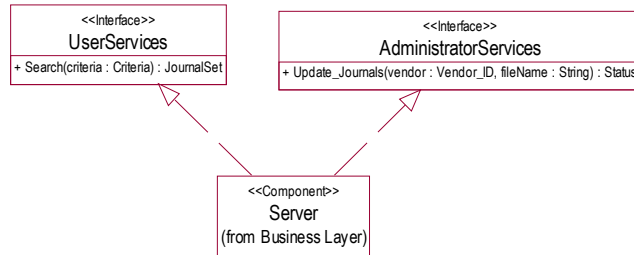
- Define attributes, operations, and relations for implementation classes

Refine Object Model – LCO or LCA For Library_User Component



- Define implementation objects that are used to build component
 - May not need if implementation class model is real close to logical class model

Define Component Interfaces – LCO or LCA For Server Component



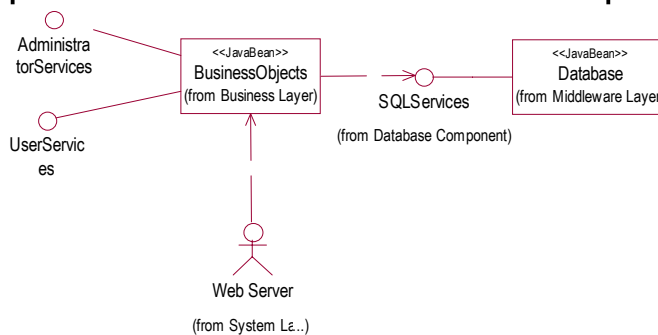
- Define *interfaces* for Component
 - Define operations available in each *interface*
 - Define dependencies on other classes & *interfaces*
- Note: deferring complete AdministratorServices *interface* to later build

OO D&D III & IV, CS577b, Spring 2002

49

2/26/2002

Refine Architecture – LCA or IOC Component Model For **Server** Component



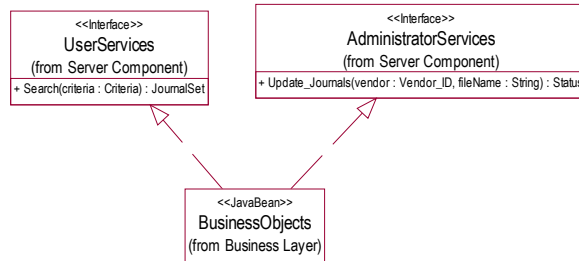
- Revise to show any changes due to implementation decisions
 - Interface named SQLServices supplied by Database component
 - Database component will be Java Bean
 - New BusinessObjects Component (Java Bean)
 - Decouple higher-level components from decisions about Database (e.g. SQL)
 - Realizes AdministratorServices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

OO D&D III & IV, CS577b, Spring 2002

50

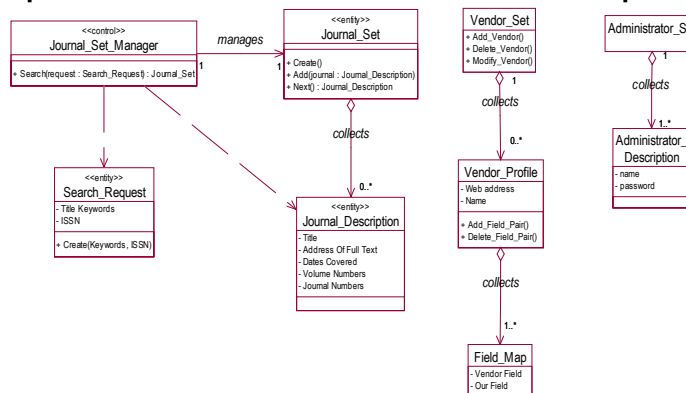
2/26/2002

Define Component Interfaces – LCO or LCA For Server::BusinessObjects Component



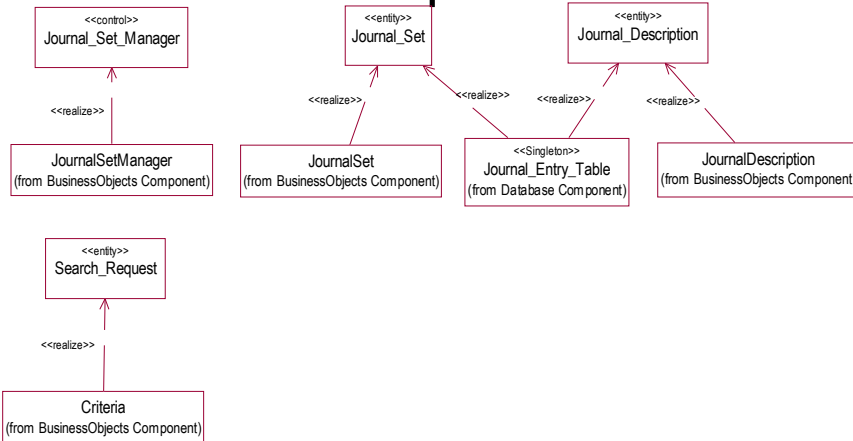
- Define *interfaces* for Component
 - Define operations available in each *interface*
 - Define dependencies on other classes & *interfaces*
- Notes:
 - Deferring complete AdministratorServices *interface* to later build
 - This diagram is shown for completeness
 - Information already resented in Component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

Refined Logical Class Model – LCA or IOC Component Model For Server Component



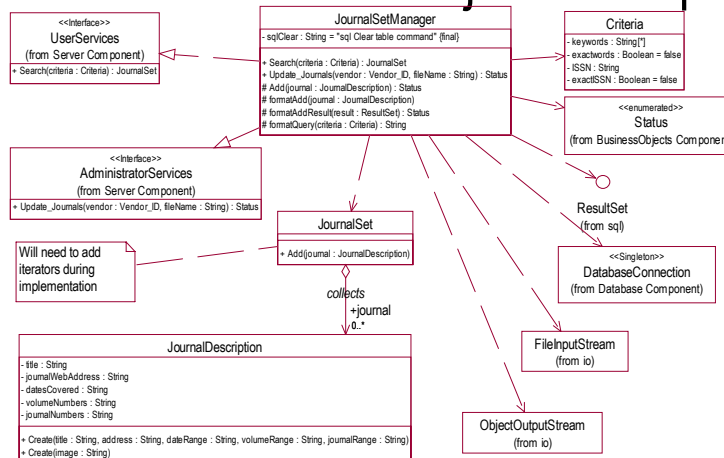
- When add BusinessObjects Component, moved this diagram (and logical Object-Structure Diagram) from Database subcomponent to Server component
 - Realized that this diagram represented classes in both BusinessObject and Database subcomponents
 - Alternative is to duplicate diagram in each component

Define Implementation Class Model – LCA or IOC For Server Component



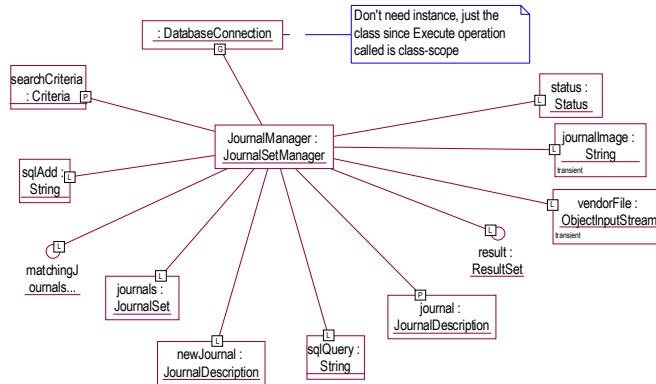
- Define implementation of logical classes
- Note: deferring Vendor_Set & User_Set to later build

Define Implementation Class Model – LCA or IOC For Server::BusinessObjects Component



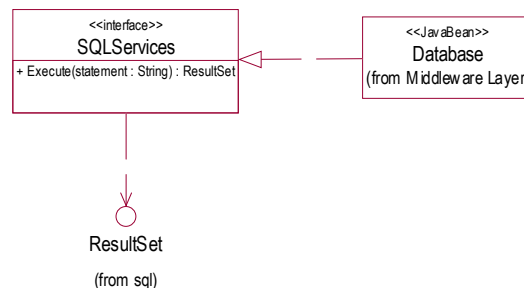
- Define attributes, operations, and relations for implementation classes
- Note: deferring Vendor_Set & User_Set implementations to later build

Refine Object Model – LCA or IOC For Server::BusinessObjects Component



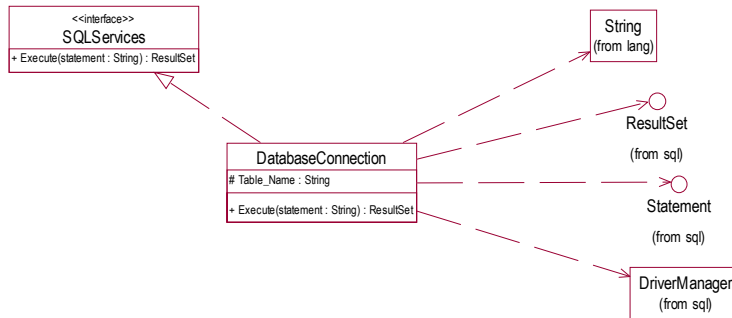
- Define implementation objects that are used to build component
 - May not need if implementation class model is real close to logical class model

Define Component Interfaces – LCA or IOC For Server::Database Component



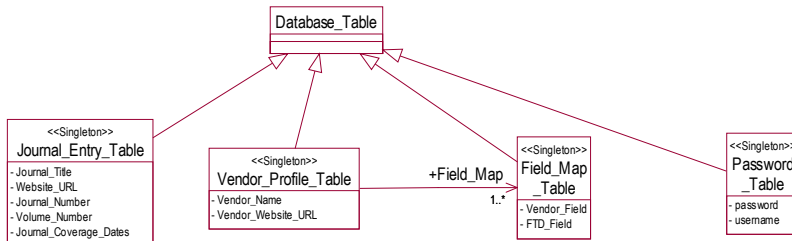
- Define *interfaces* for Component
 - Define operations available in each *interface*
 - Define dependencies on other classes & *interfaces*

Define Implementation Class Model – LCA or IOC For Server::Database Component



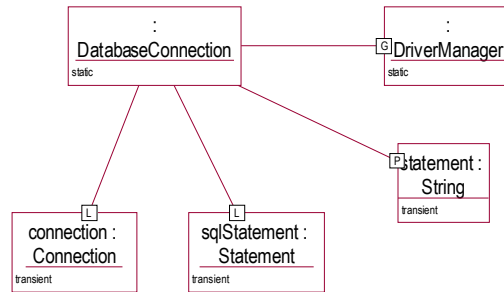
- Define attributes, operations, and relations for implementation classes

Define Implementation Class Model – LCA or IOC For Server::Database Component, Relational Database Schema



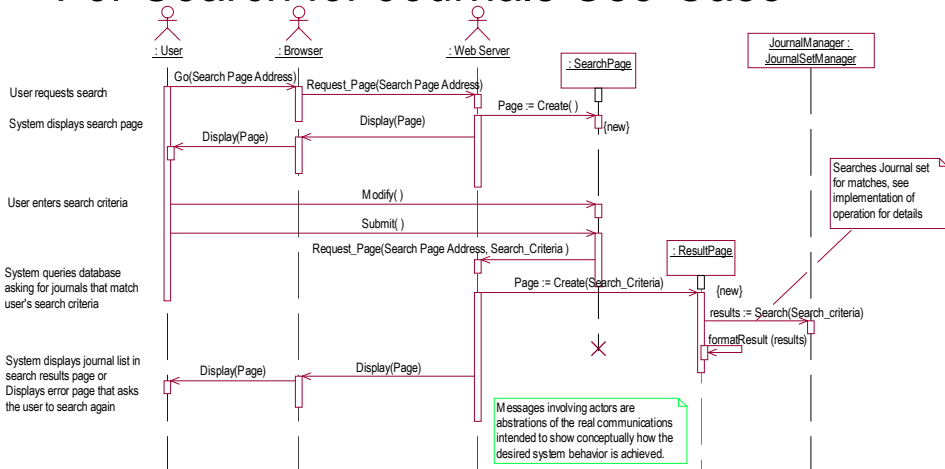
- Define attributes, operations, and relations for implementation classes

Refine Object Model – LCA or IOC For Server::Database Component



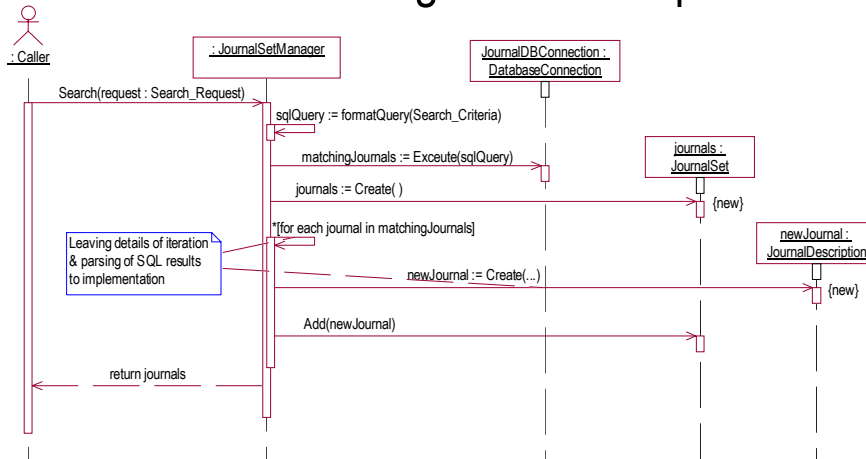
- Define implementation objects that are used to build component
 - May not need if implementation class model is real close to logical class model

Refine Interaction Model – LCA or IOC For Search for Journals Use-Case



- Define how objects interact to achieve use-cases
 - May not need if implementation class model is real close to logical class model

Refine Interaction Model – LCA or IOC For JournalSetManager.Search Operation



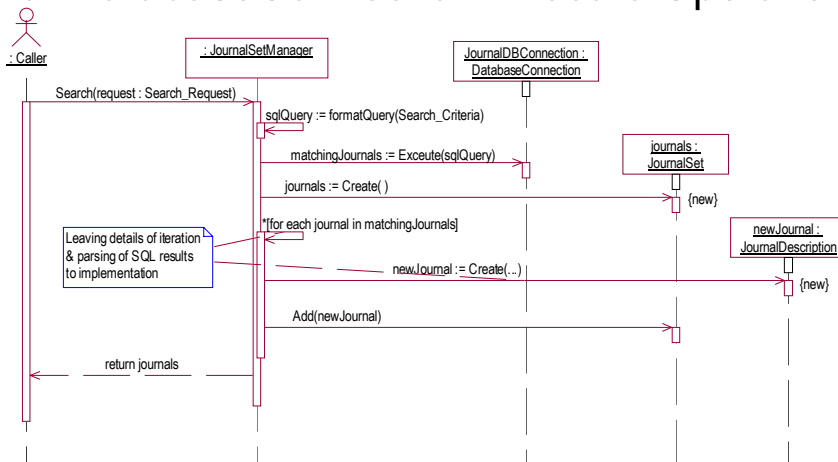
- Define how objects interact to achieve use-cases
 - May not need if implementation class model is real close to logical class model

OO D&D III & IV, CS577b, Spring 2002

61

2/26/2002

Refine Interaction Model – LCA or IOC For DatabaseConnection.Execute Operation



- Define how objects interact to achieve use-cases
 - May not need if implementation class model is real close to logical class model

OO D&D III & IV, CS577b, Spring 2002

62

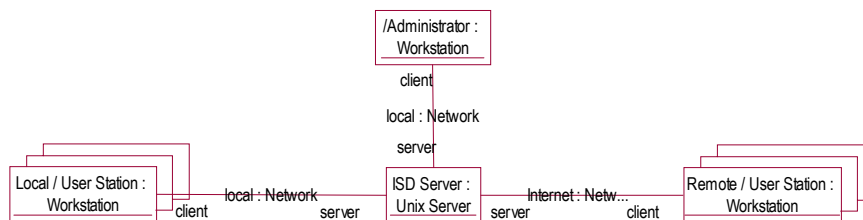
2/26/2002

Refine Deployment Model – LCA or IOC Node Classes for Full-Text Title Database System



- No changes in this example

Refine Deployment Model – LCA or IOC Node Configuration for Full-Text Title Database System



Refine Deployment Model – LCA or IOC Component Configuration for ...

■ /Administrator : Workstation

■ /User : Workstation

■ Define how software components are allocated to hardware nodes

- May not need if Implementation Deployment Model is same to Logical Deployment Model

OO D&D III & IV, CS577b, Spring 2002 65 2/26/2002

Refine Deployment Model – LCA or IOC Component Configuration for : Unix Server

■ Define how software components are allocated to hardware nodes

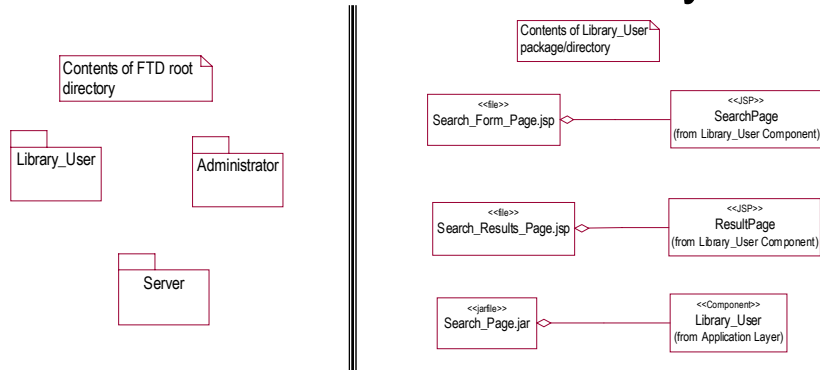
- May not need if Implementation Deployment Model is same to Logical Deployment Model

OO D&D III & IV, CS577b, Spring 2002 66 2/26/2002

Define Artifact Configuration Model

- Purpose:
 - Define *artifacts* that will be produced during implementation
 - Source files
 - Scripts
 - Binary files
 - Dynamic or static link libraries
 - Database Tables
 - Define what components or classes are in each artifact?
 - Define which artifacts “know” each other
- Inputs:
 - Component Model
 - Implementation Class Model
- Artifacts:
 - Configuration Model
- Typically done at LCO & LCA only if creating executable prototype

Configuration Model – LCA or IOC For Full-text Title Database System



- Define
 - File structure
 - What Components or classes are in each file
 - Directory structure

