

# **System Design II**

## **CS577a**

### **Fall 2002**

Ed Colbert

USC Center for Software  
Engineering

# Goal of Presentation

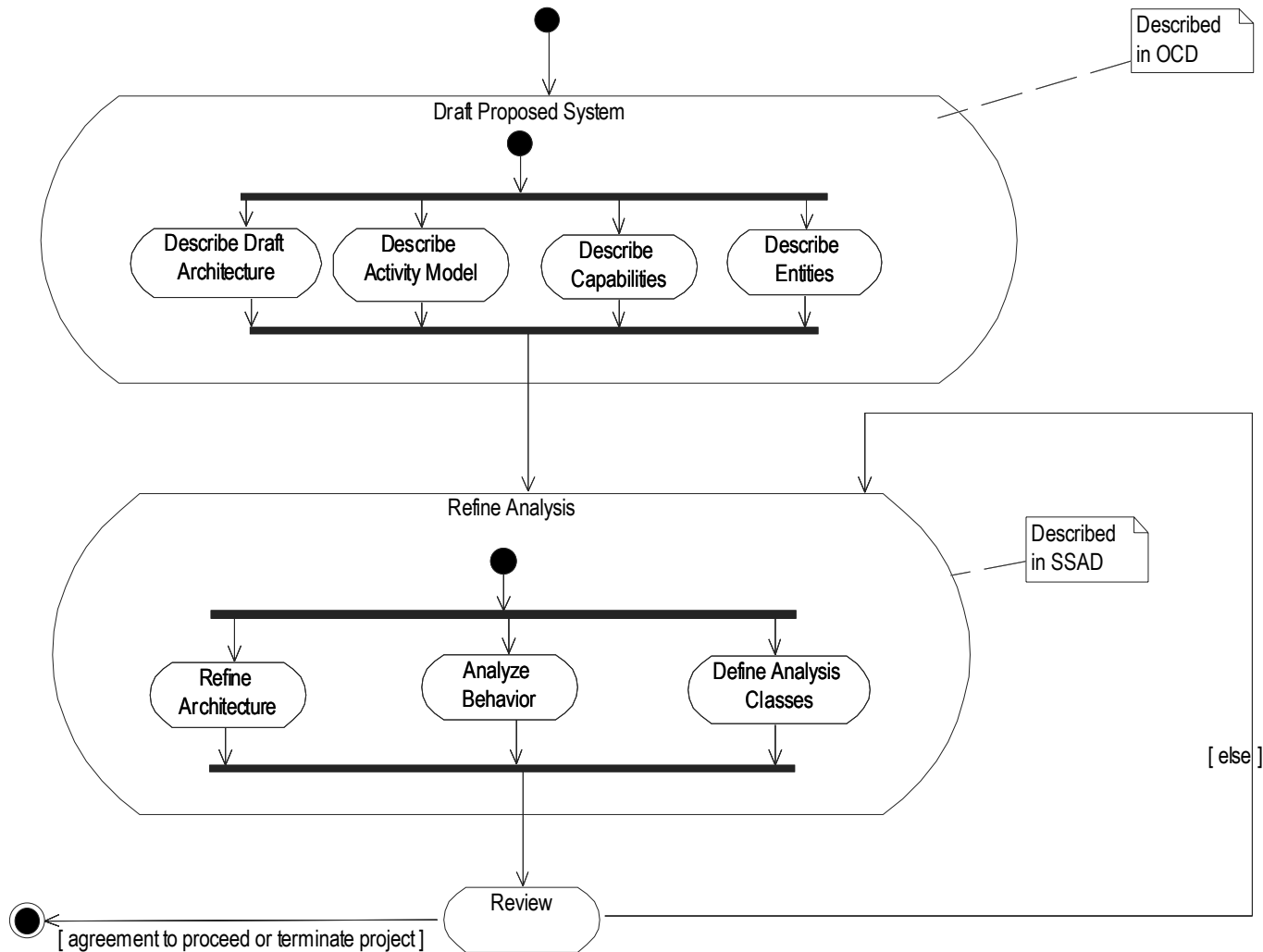
- Understand how to perform System Design
  - Using
    - MBASE
    - Object-oriented techniques
    - RUP
    - Rational Rose
- Understand how to document analysis
- Presentation is part 2 & 3 of 3 lectures on System Design



# Outline

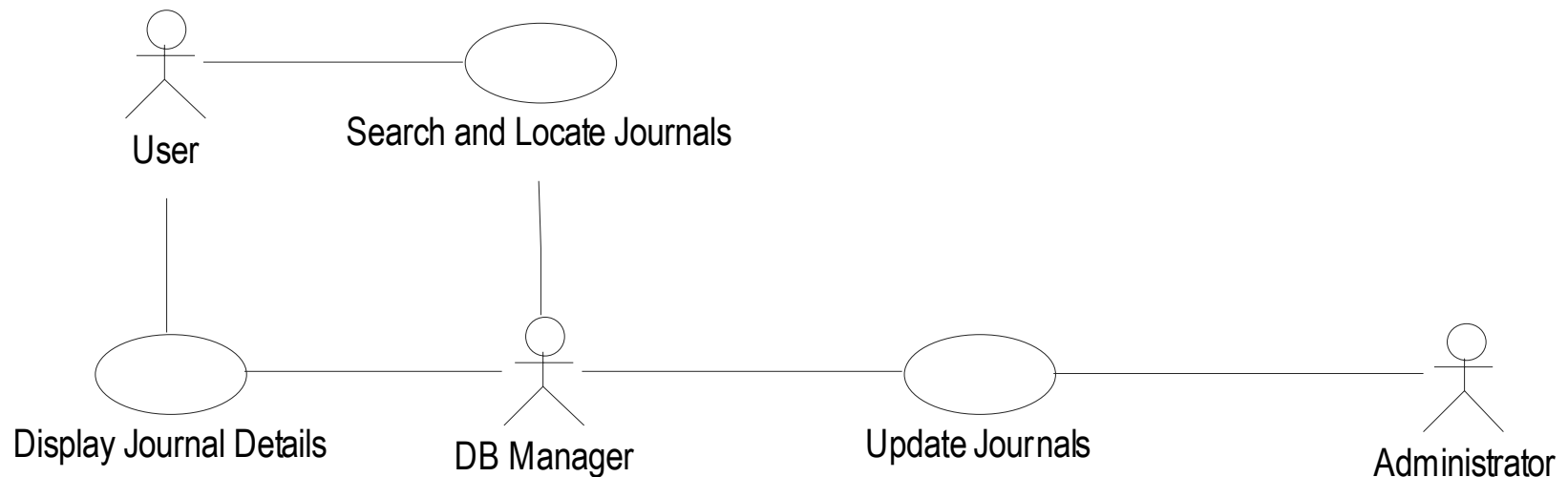
- **When Last We Met ...**
- Design Process Overview
- Design Process by Example

# System Analysis Process Overview



# Analyze Behavior – LCO

## Use-Case Diagram Example



# Analyze Behavior – LCO

## Use-Case Description Example 1

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

# 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.)

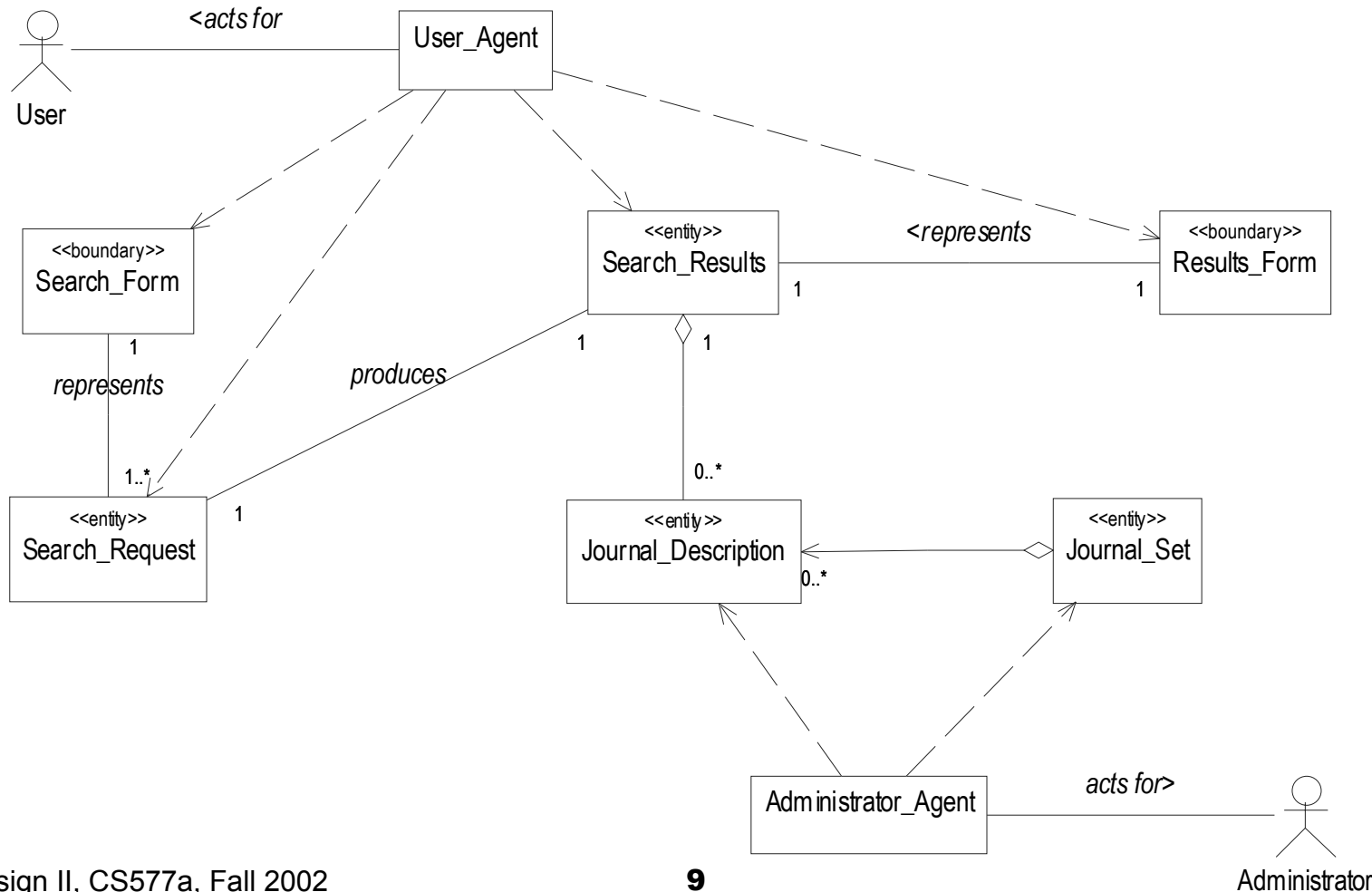
- Alternate Course of Action: No results match search criterion

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

- Exceptional Course of Action: None

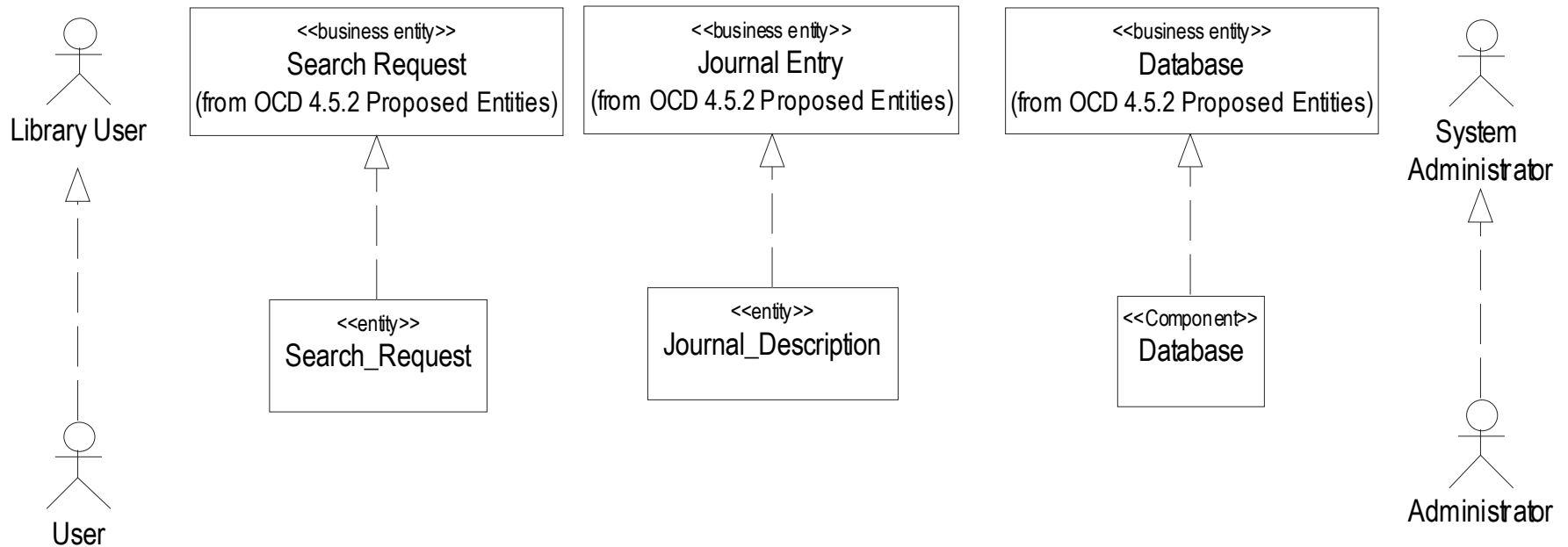
# Define Analyze Classes – LCO

## Enterprise Classification Model for Full-Text Title Database System



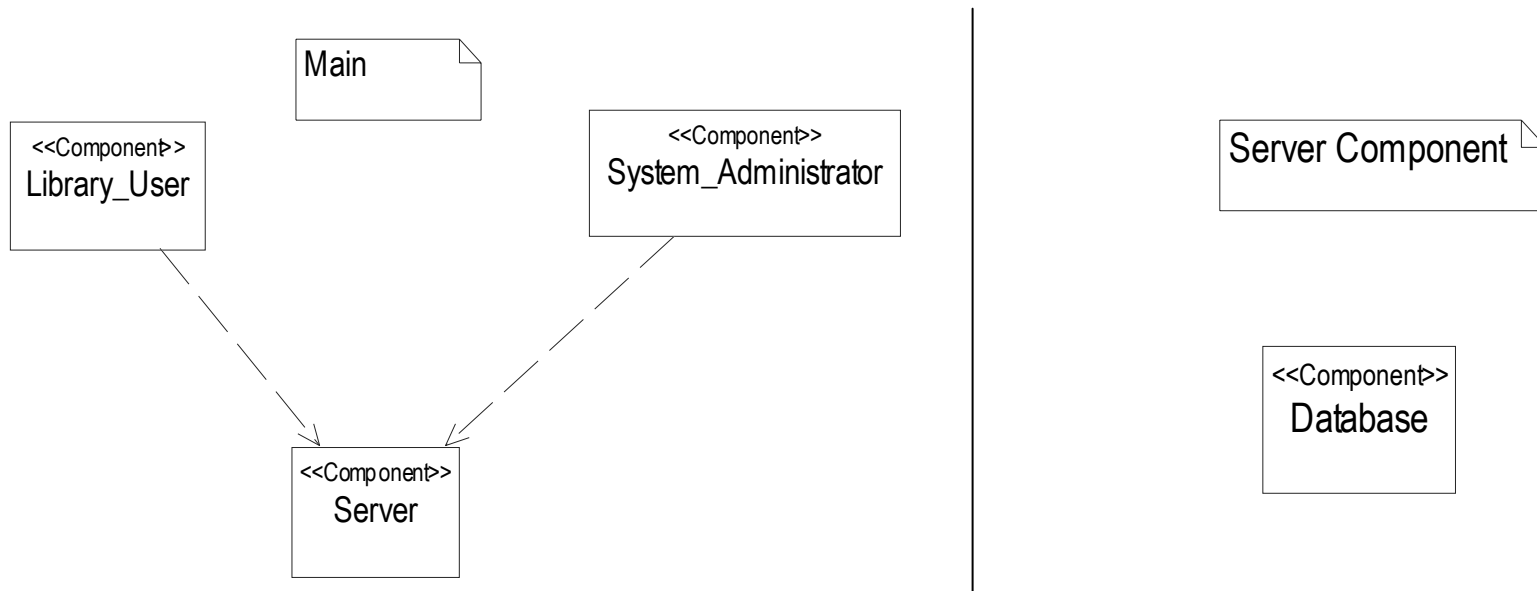
# Define Analyze Classes – LCO

## Business-Analysis Class Mapping for Full-Text Title Database System



# Architecture – LCO

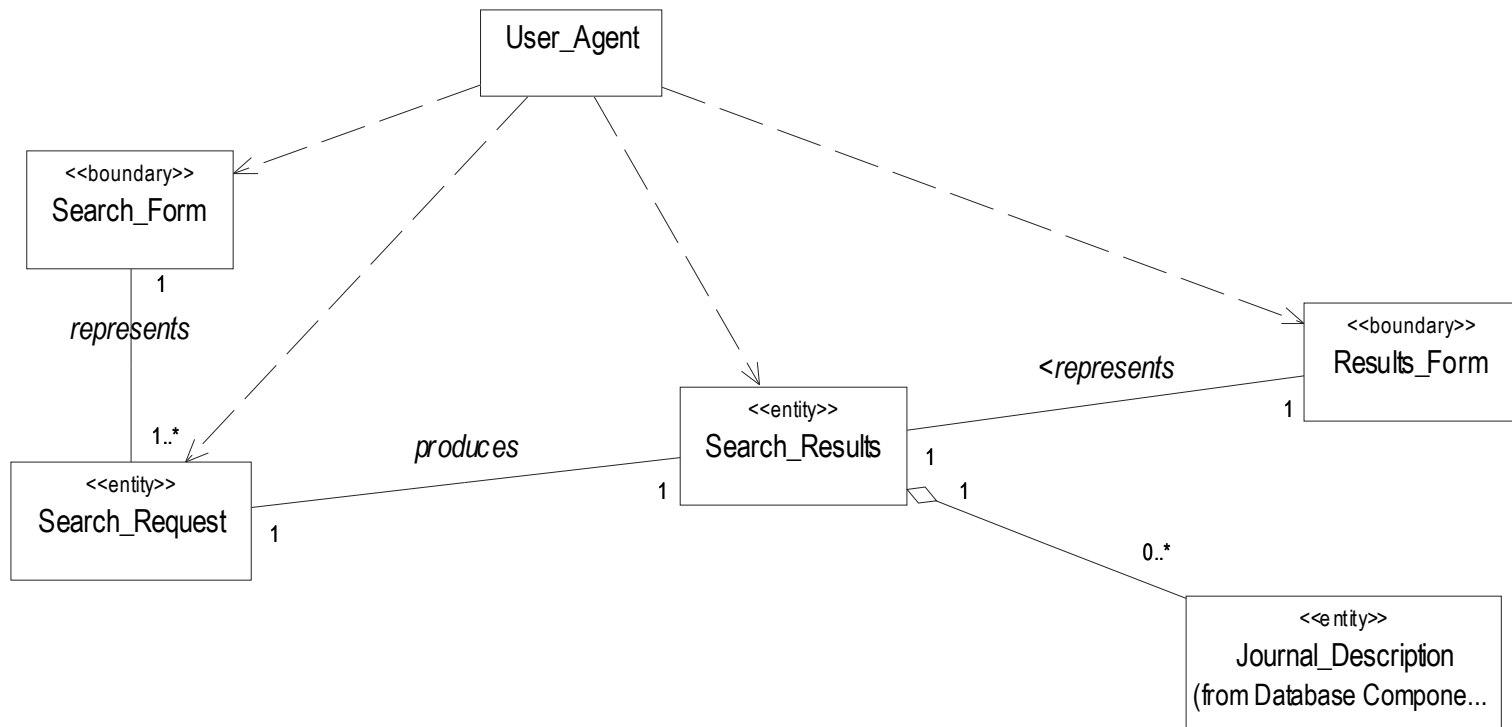
## Component Model For Full-text Title Database System



- Database component from System Block Diagram
- Decision
  - Client-Server Model
- Note: Database is often deferred until later in development
  - Some classes just described as “persistent” early in process

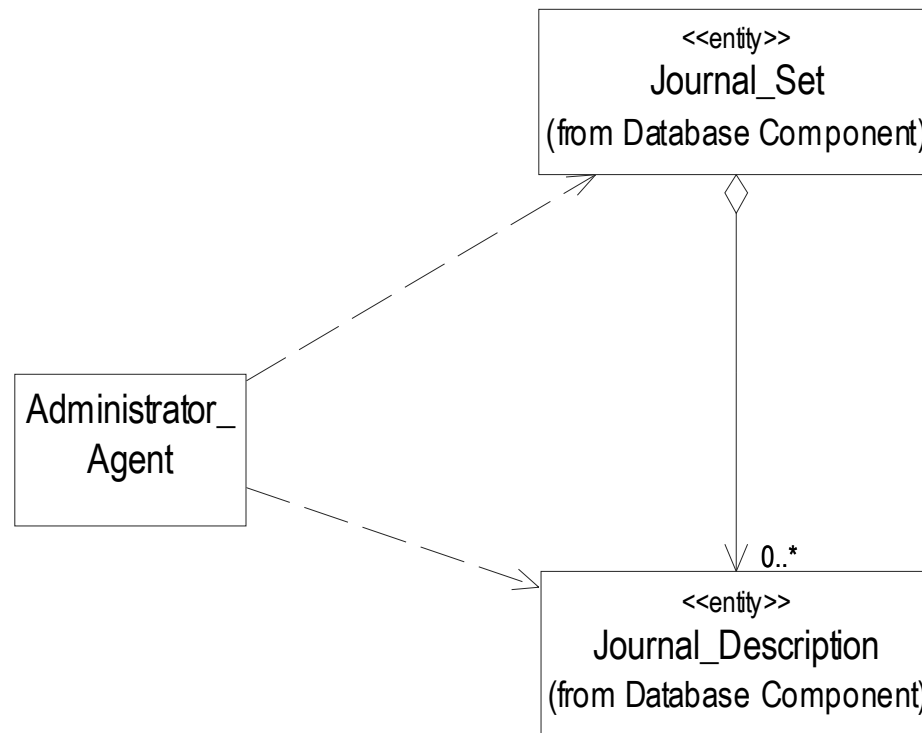
# Refine Architecture – LCO

## Logical Class Model for Library\_User Component



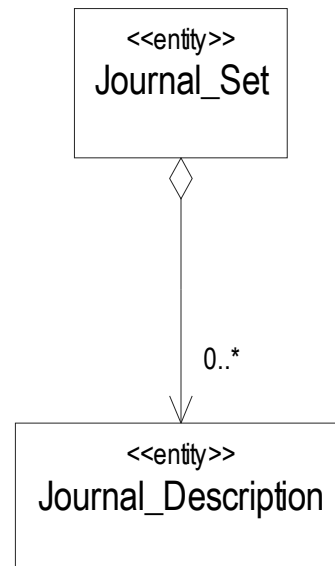
# Refine Architecture – LCO

## Logical Class Model for System\_Administrator Component



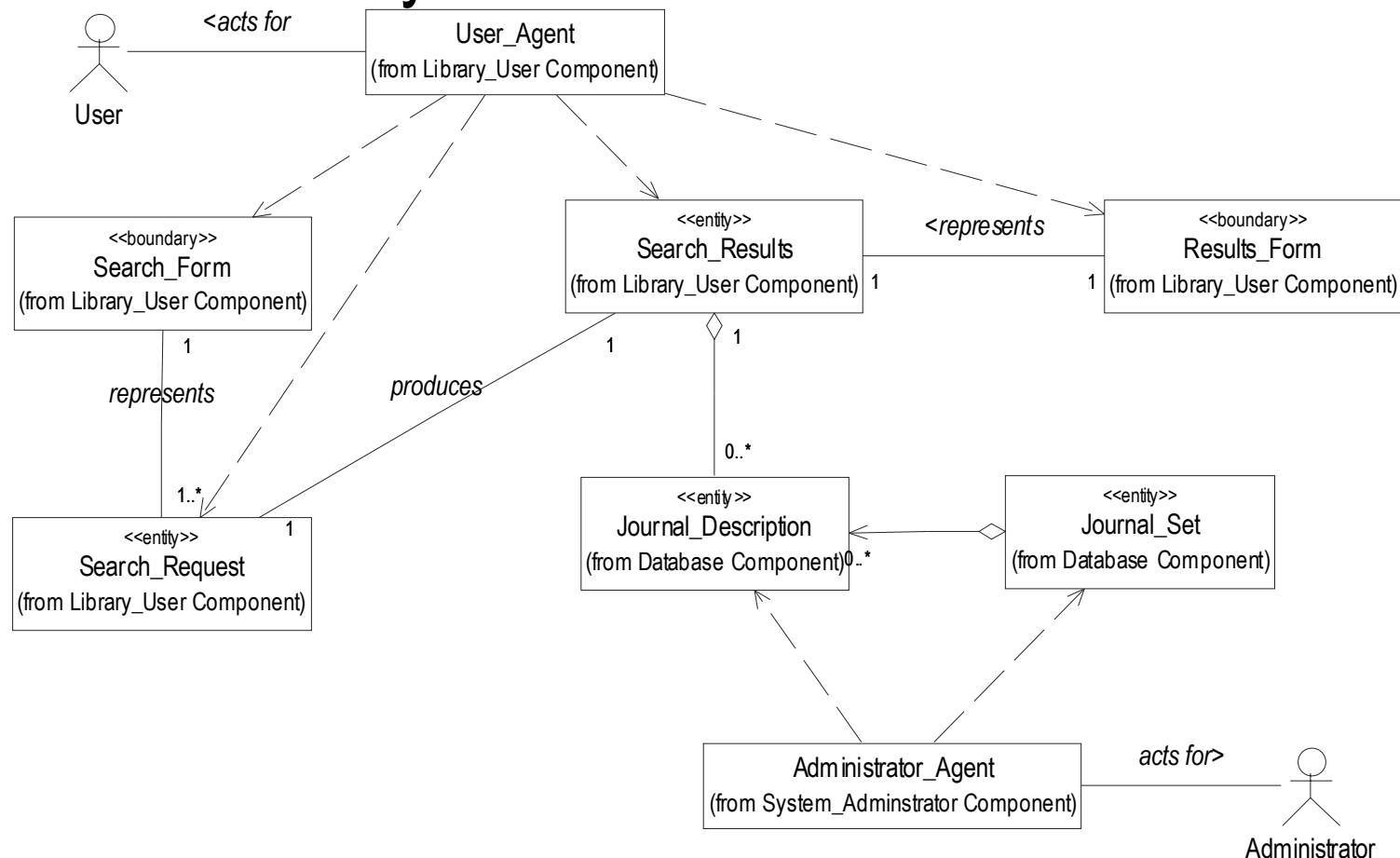
# Refine Architecture – LCO

## Logical Class Model for Server::Database Component



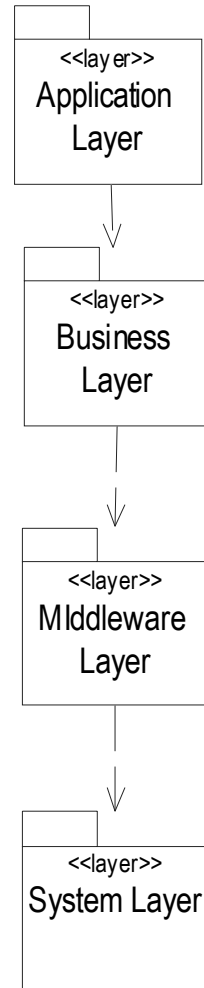
# Refine Architecture – LCO

## Updated Enterprise Classification Model for Full-Text Title Database System



# Refine Architecture – LCO

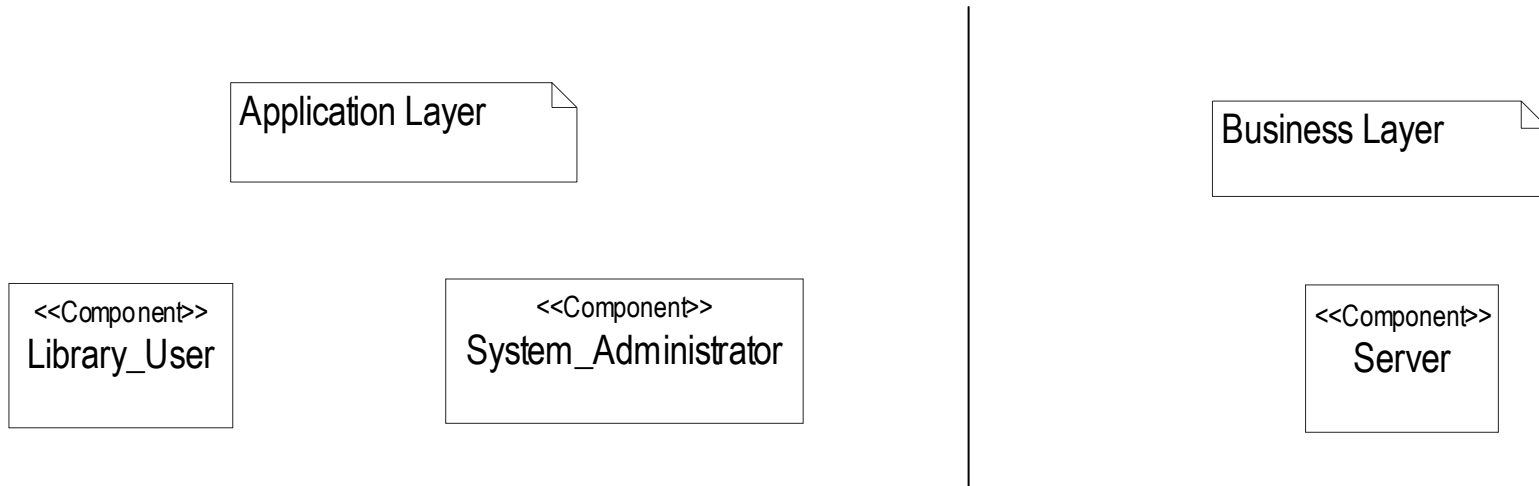
## System Topology for Full-Text Title Database System



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

# Refine Architecture – LCO

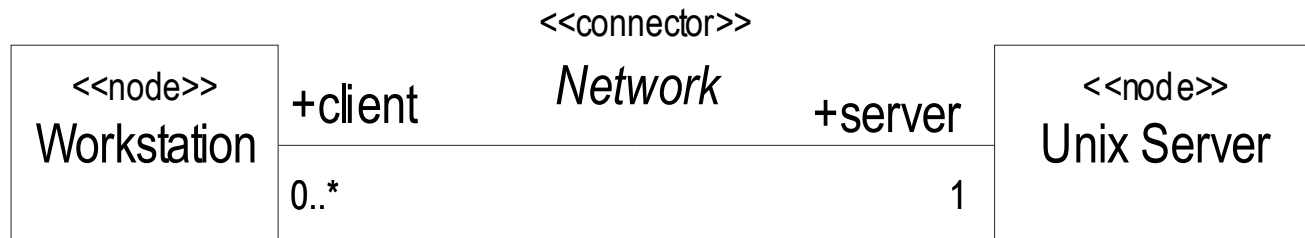
## System Topology for Full-Text Title Database System



- Nothing in other layers
  - yet...

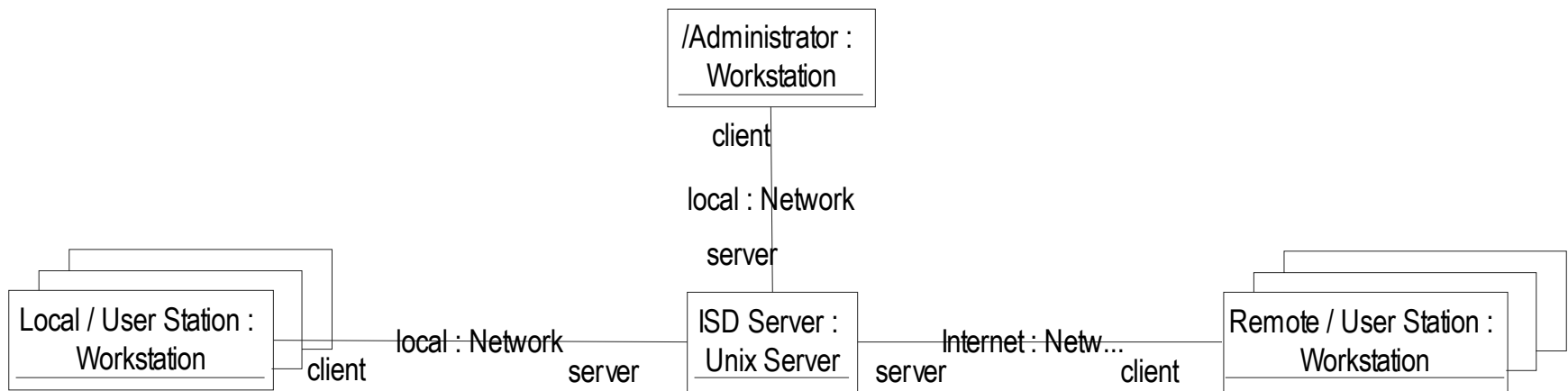
# Refine Architecture – LCO

## Node Classes for Full-Text Title Database System



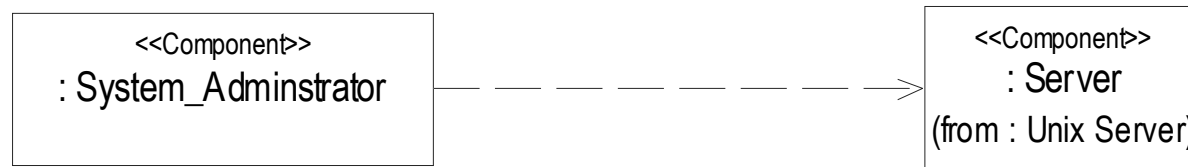
# Refine Architecture – LCO

## Node Configuration for Full-Text Title Database System



# Refine Architecture – LCO

## Component Configuration for /Administrator : Workstation



# Refine Architecture – LCO

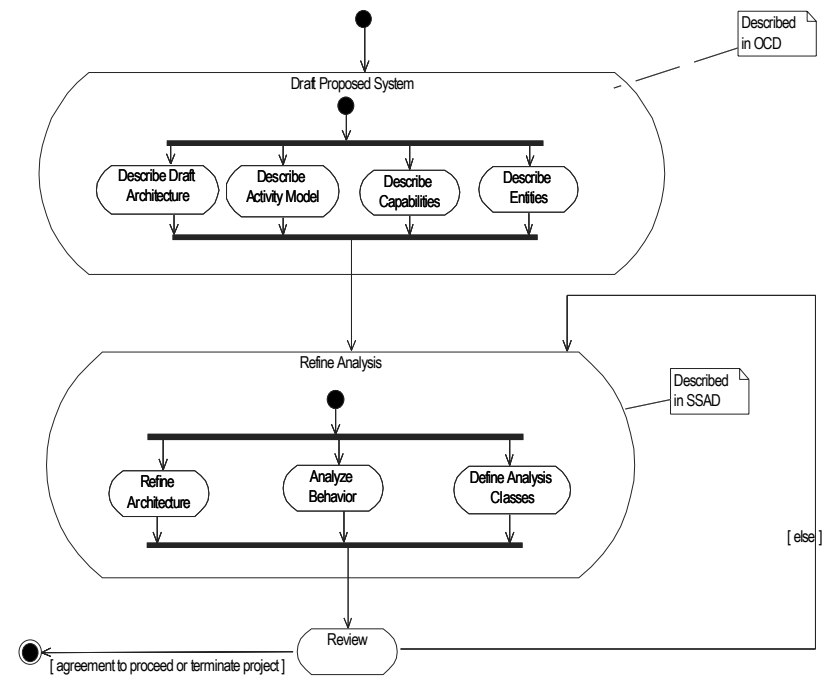
## Component Configuration for / User Station : Workstation



# System Analysis

## What Have We Done?

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





# 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.

# 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

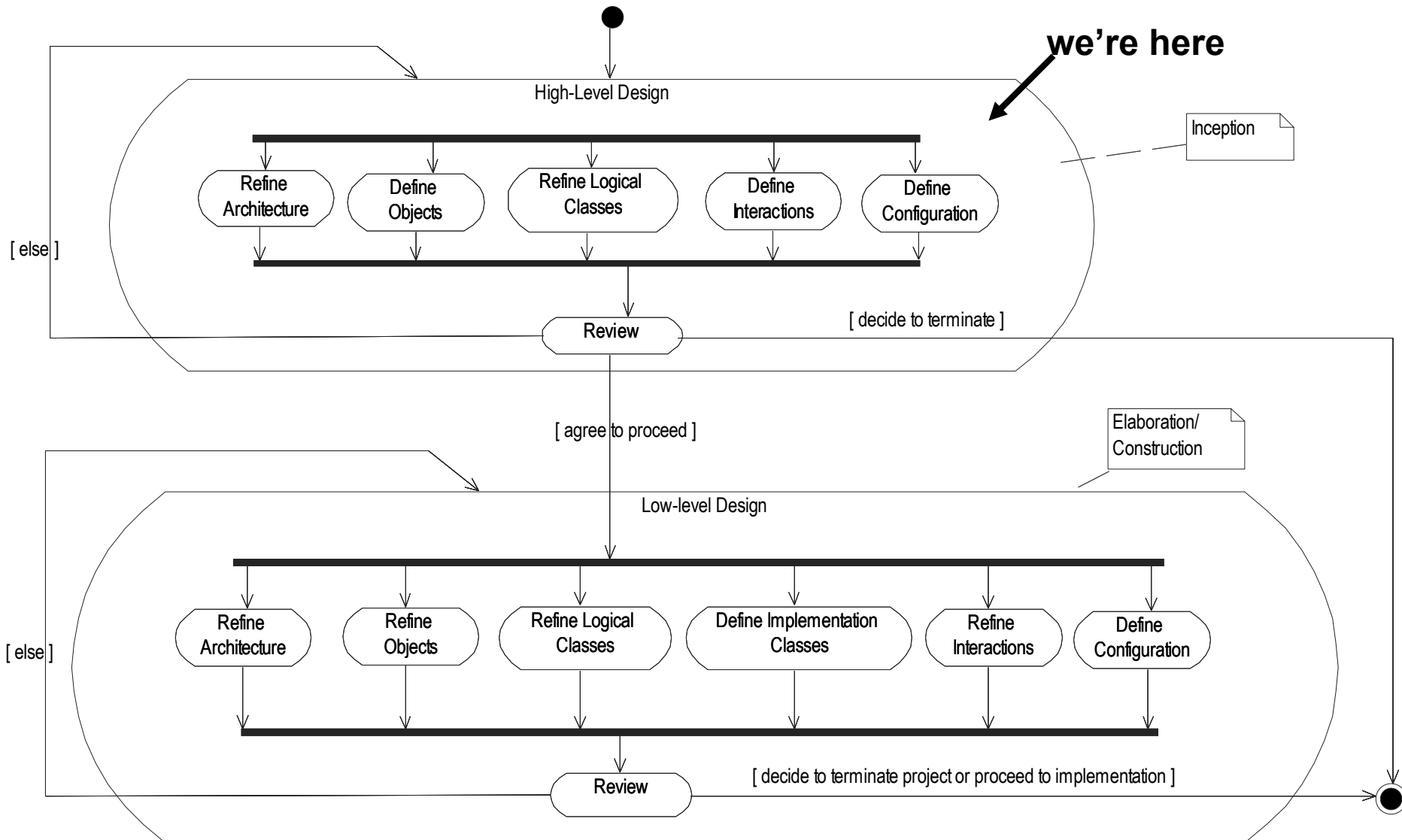
## ■ General

- 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 Design Process Overview





# Outline

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

# Define Interactions

- Purpose:
  - Refine behavior model
  - Validate/Identify objects & their links
  - Identify operations
- Inputs:
  - Use-cases Model
  - Object Model
  - Logical or Implementation Class Model
- Artifacts:
  - Sequence Diagram(s) for each use-case
  - Updated Object Model
  - Updated Class Model
  - Descriptions of
    - Operations
    - Algorithms

# Define Interactions – LCO or LCA

## Guide for Creation of Sequence Diagram(s)

- Determine whether to create
  - 1 Sequence Diagram that merges normal, alternate, & exceptional courses of actions
  - Separate Sequenced Diagrams for each course of action
- For each Sequence Diagram
  - Add descriptions of course(s) of action in left margin of diagram
  - If collaboration of objects already defined in model
    - Add all objects to diagram
    - For each step in course of action
      - Identify
        - Object that request operation
        - Object that performs operation
        - Name of operation
        - Signature of operation
      - Add
        - Operation to class of performing object, if doesn't exist
        - Message from requesting object to performing object
      - If performing object requires other objects do something
        - Identify objects & their operations
        - Add operations messages

# Define Interactions – LCO or LCA

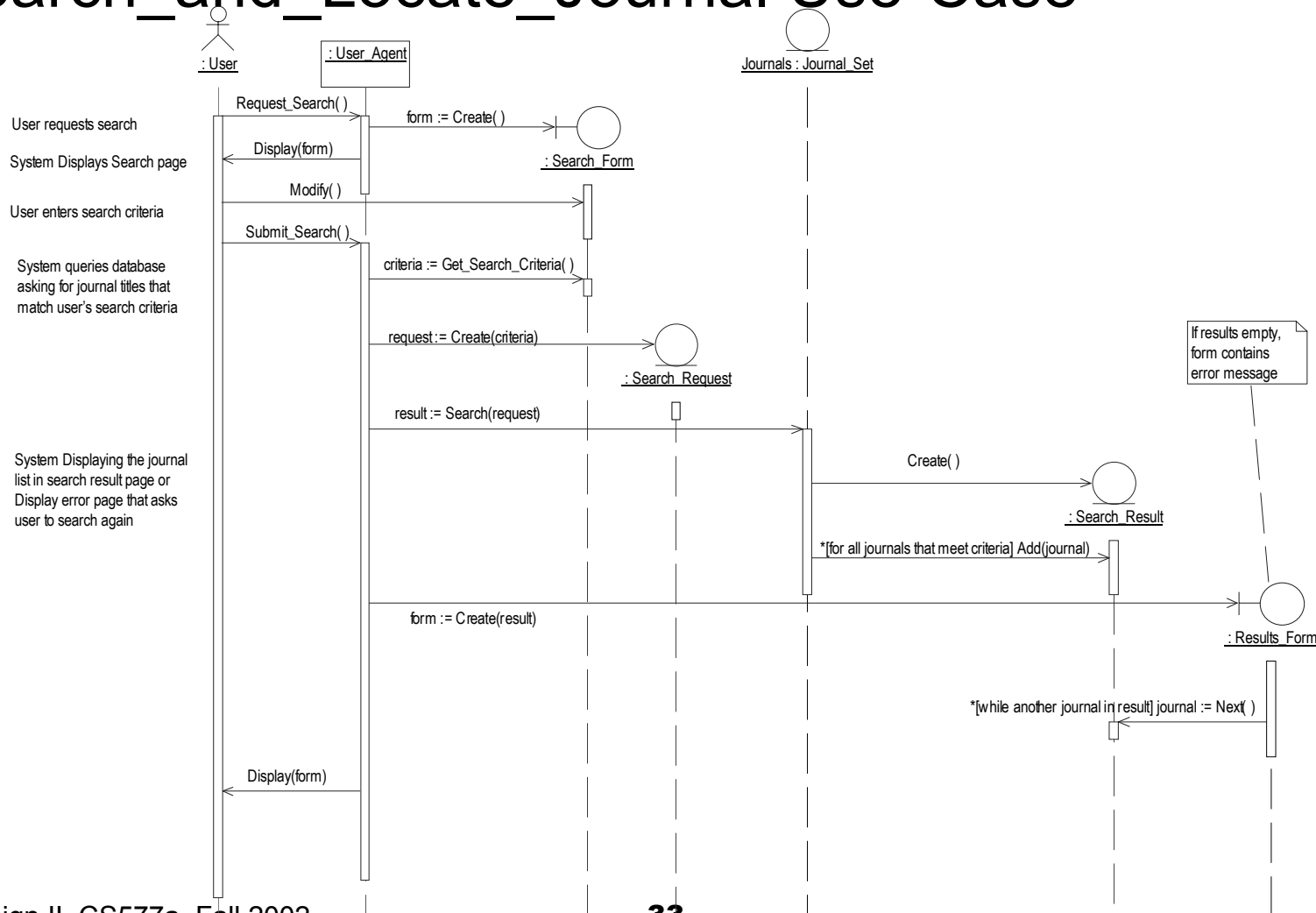
## Guide for Creation of Sequence Diagram(s)

- For each Sequence Diagram (cont.)
  - If collaboration of objects not defined in model
    - Add actors that participate in use–case
      - If separate diagrams for each course, add only actors that participate in that course
    - For each step in course of action
      - Identify object that request operation
      - Define
        - Object that performs operation
        - Name of operation
        - Signature of operation
      - Identify class of performing object
      - Add
        - Operation to class of performing object, if doesn't exist
        - Message from requesting object to performing object
      - If performing object requires other objects do something
        - Identify objects & their operations
        - Add operations messages

# Define Interactions – for LCO or LCA

## Sequence Diagram Example

### Search\_and\_Locate\_Journal Use-Case



# Define Interactions – LCO or LCA

## Updating Models

### ■ Interaction Model

- For non-trivial operations
  - Explain algorithms & policies in SSAD section 3.3.1
  - Describe using form in SSAD section 3.3.2

### ■ Class Model

- All ready added new operations to classes while creating Sequence Diagrams

### ■ Object Model

- If based sequence diagram on existing collaboration of objects
  - For each message
    - A Collaboration Diagram should show
      - Object requesting operation
      - Object performing operation
      - Link between objects
    - Add any missing objects or links

# Define Interactions – LCO or LCA

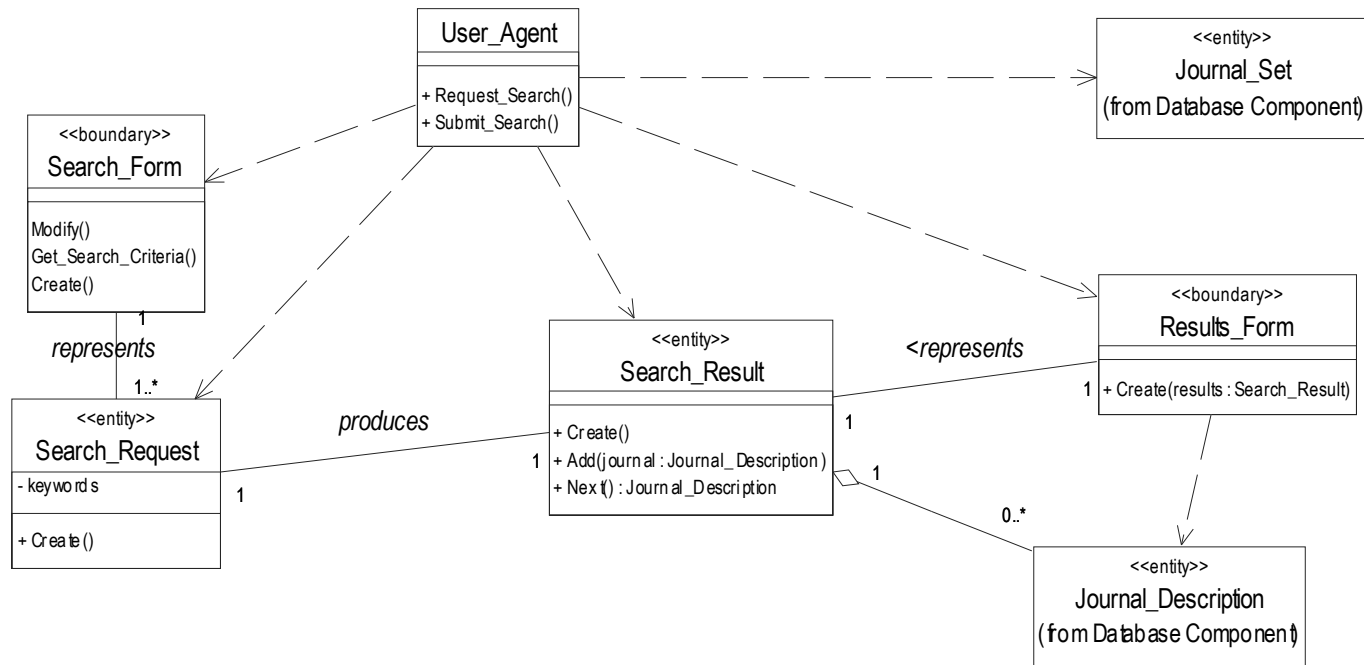
## Operation Specification Form

### 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

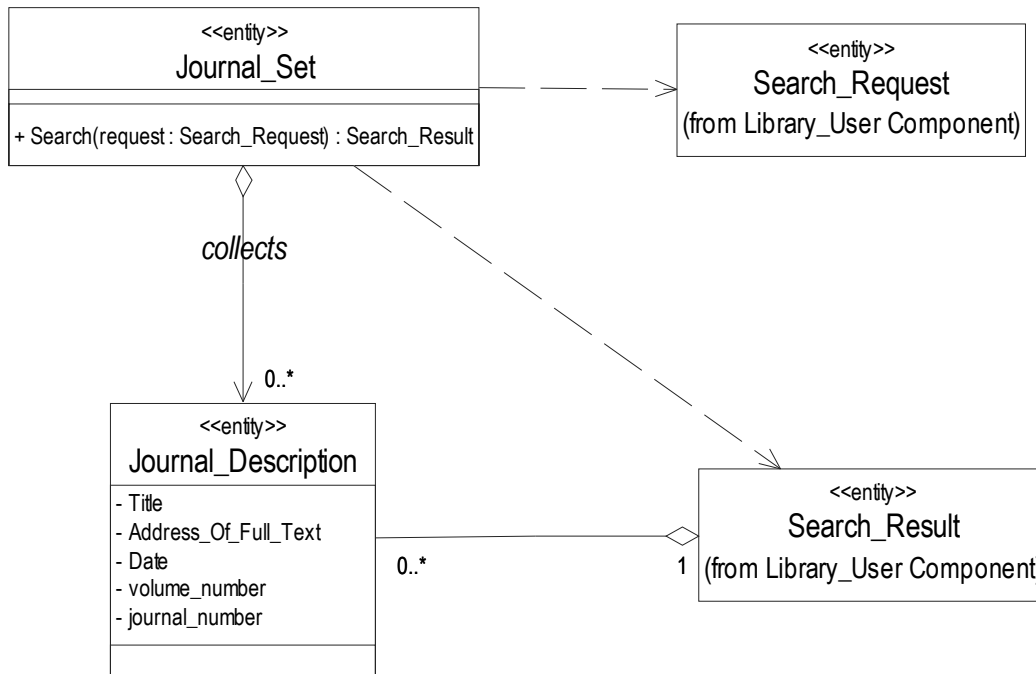
# Define Interactions – LCO or LCA

## Updated Logical Class Model For Library\_User Component



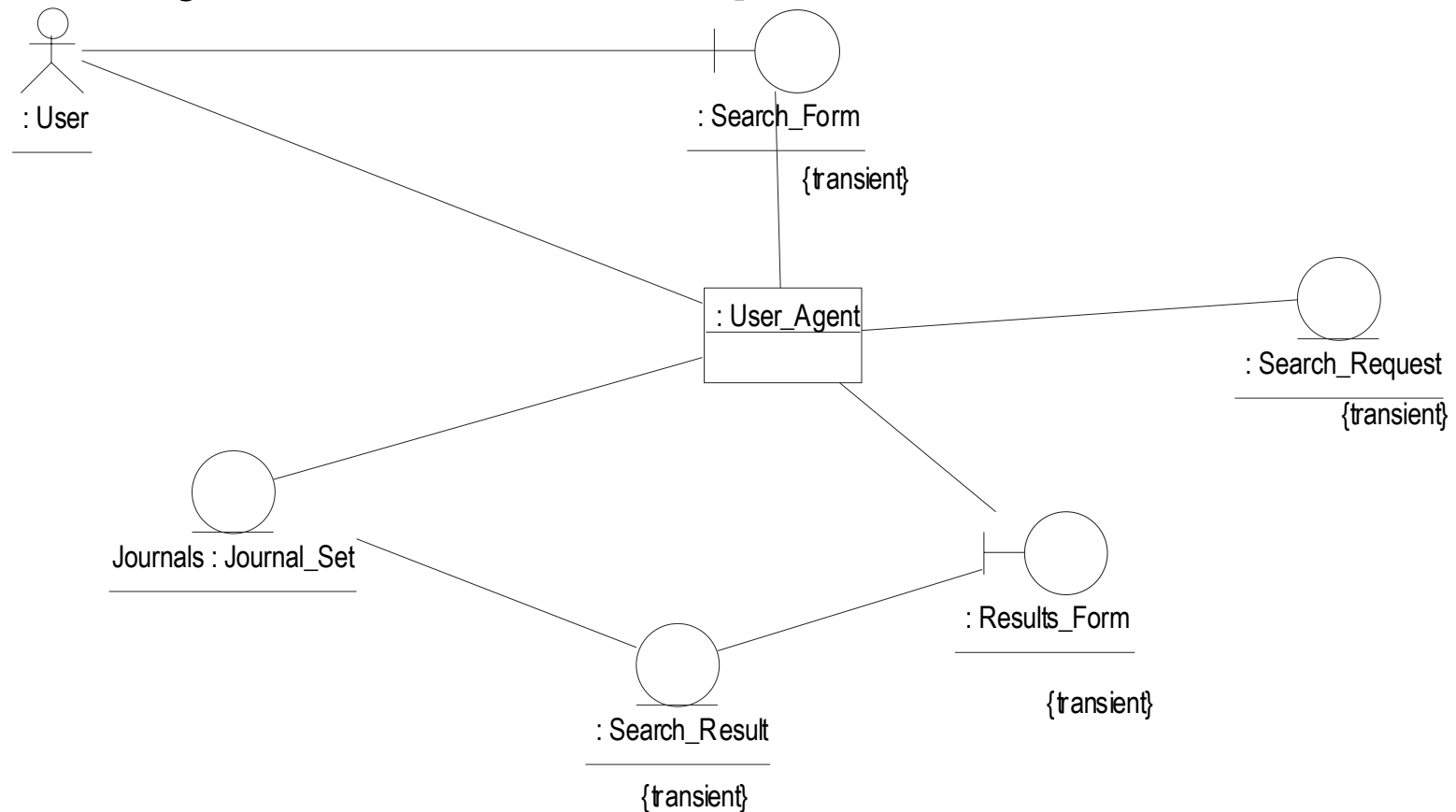
# Define Interactions – LCO or LCA

## Updated Logical Class Model For Server::Database Component



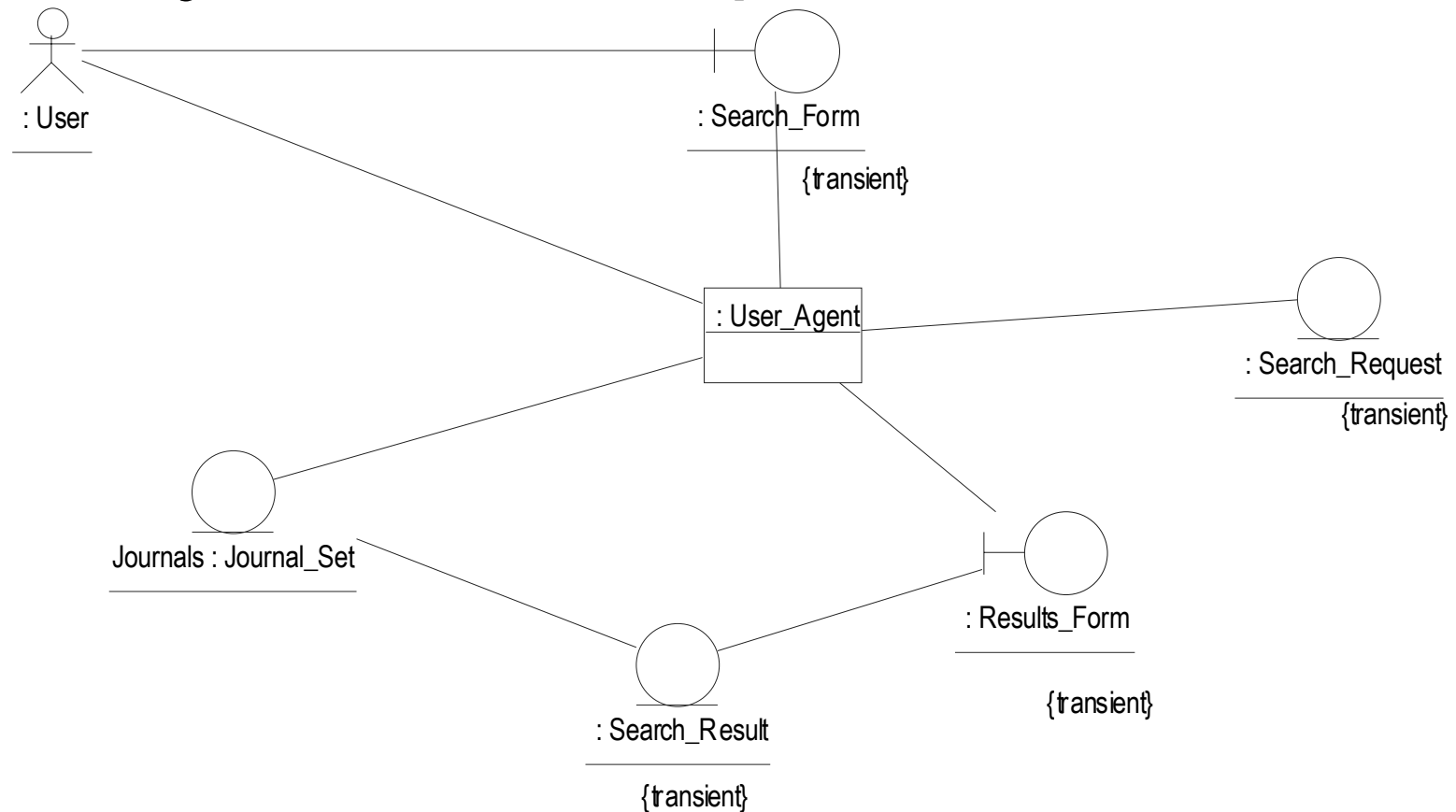
# Define Interactions – LCO or LCA

## Draft Object Model For Library\_User Component



# Define Interactions – LCO or LCA

## Draft Object Model For Library\_User Component



# Define Objects

- Purpose:
  - Define objects that will implement each component
    - What instances of classes belong in component?
    - What objects from other components must be known?
  - Define how object “know” each other
- Inputs:
  - Logical or Implementation Class Model
  - Use-cases Model
  - Interaction Model
  - Design Patterns
- Artifacts:
  - Object Model
  - Updated Class Model

# Define Object Model – LCO or LCA

## Guide for Creation of Collaboration Diagram(s)

- For each component,
  - Identify objects needed to build component
    - Look for objects described in use-case
      - Descriptions (i.e. forms)
      - Sequence Diagrams (from Interaction Model)
        - If created
    - Determine whether to create
      - 1 Collaboration Diagram that merges all objects from all use-case
      - Multiple Collaboration Diagrams that show subsets of objects
        - e.g.
          - Only persistent objects
          - Objects in *Component* that implement particular use-cases
          - Snapshots of objects before & after particular use-cases
        - Selection depends on
          - What you need to communicate
          - Risks

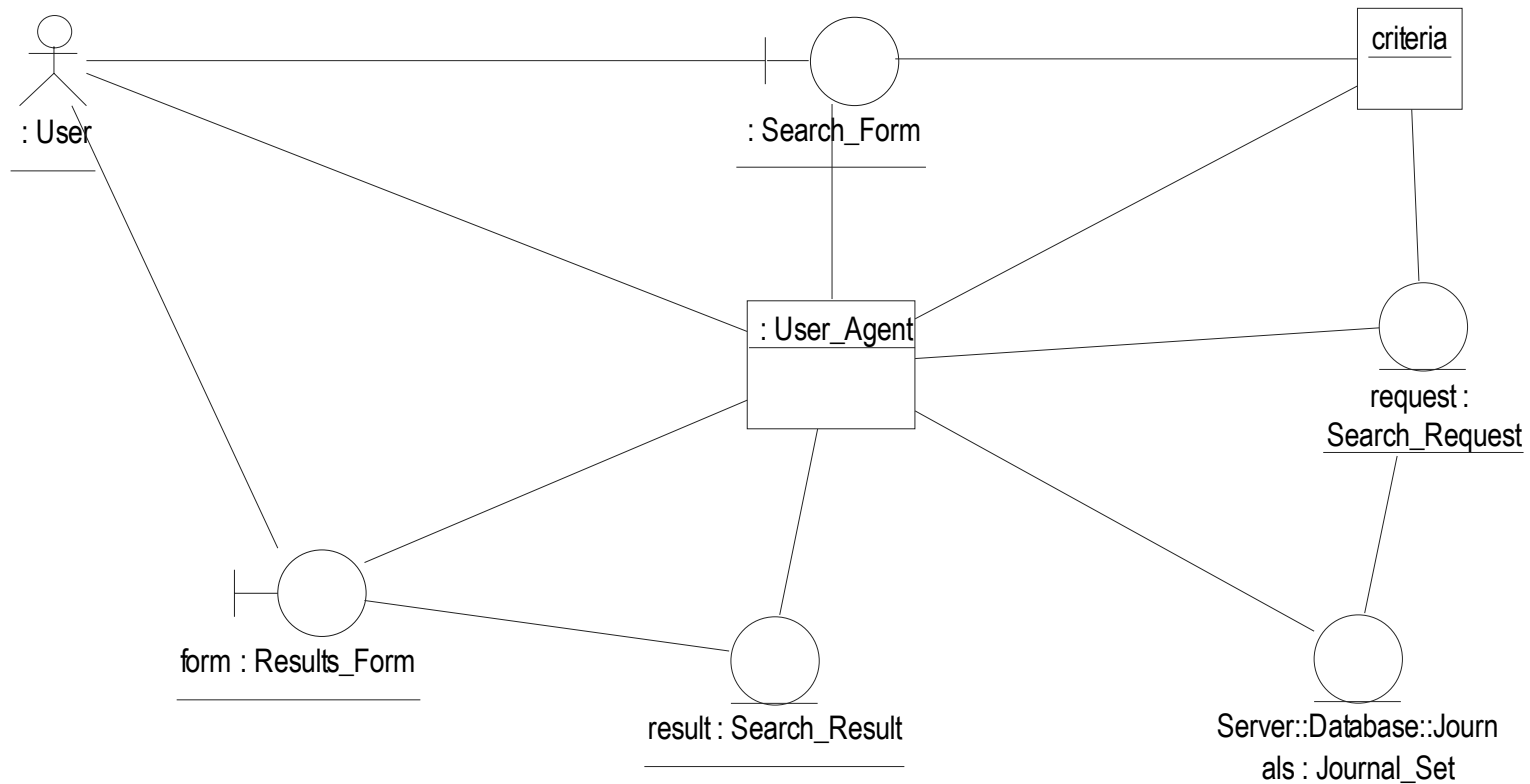
# Define Object Model – LCO or LCA

## Guide for Creation of Collaboration Diagram(s) – cont.

- For each object,
  - Specify
    - It's name &/or role-name
    - It's class
    - Links to other objects
      - In same component
      - In other components

# Define Objects – LCO or LCA

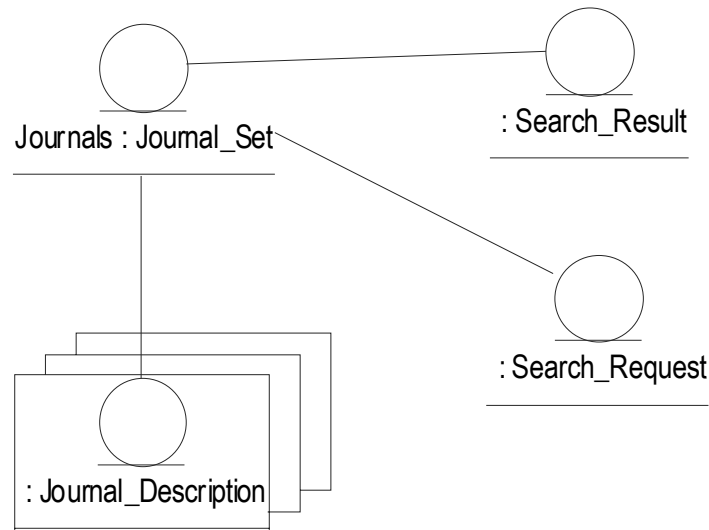
## Draft Object Model For Library\_User Component



### ■ Based on Interaction Model

# Define Objects – LCO or LCA

## Draft Object Model For Server::Database Component



### ■ Based on Interaction Model

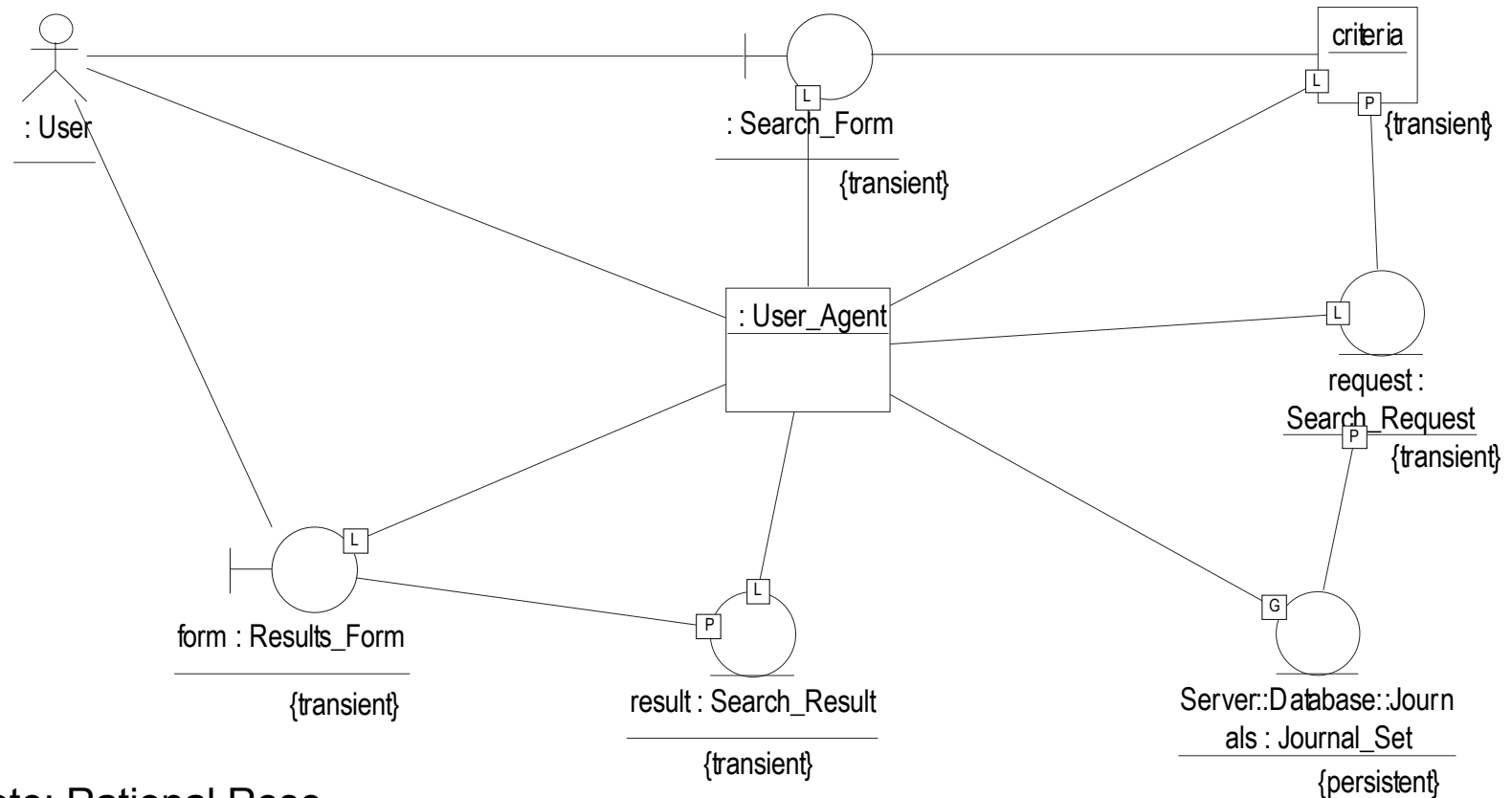
# Define Object – LCO or LCA

## Guide for Creation of Collaboration Diagram(s) – cont.

- For each link
  - UML says links are not named
  - Related an association or dependency relation
    - Specifying relation is tool dependant
  - On each end of link, optionally specify
    - Role names
    - “Implementation Stereotype”
      - «association» -- Link by association (default), inc. aggregation
      - «Parameter» -- Link to method parameter
      - «Local» -- Link to local variable of method
      - «Global» -- Link to Global object
      - «Self» -- Link to self
- For each object or link
  - If created or destroyed during execution time represented by Collaboration
    - e.g. life-time of system, single operation/use-case
    - Show one of following constraints
      - {new} -- created during execution
      - {destroyed} -- destroyed during execution
      - {transient} -- created then destroyed

# Define Objects – LCO or LCA

## Refined Object Model For Library\_User Component

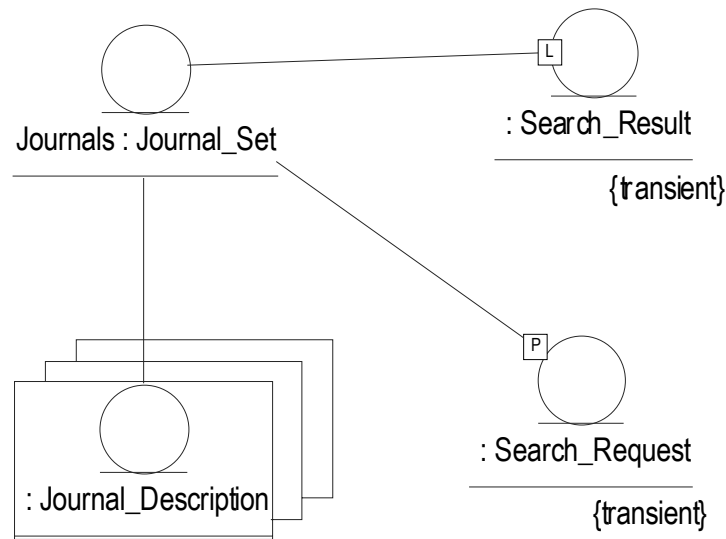


### ■ Note: Rational Rose

- Uses letters “P”, “L”, “G” in box for «Parameter», «Local», «Global» respectively
- Uses “loop-back” link for «Self»
- No visible notation for «association»

# Define Objects – LCO or LCA

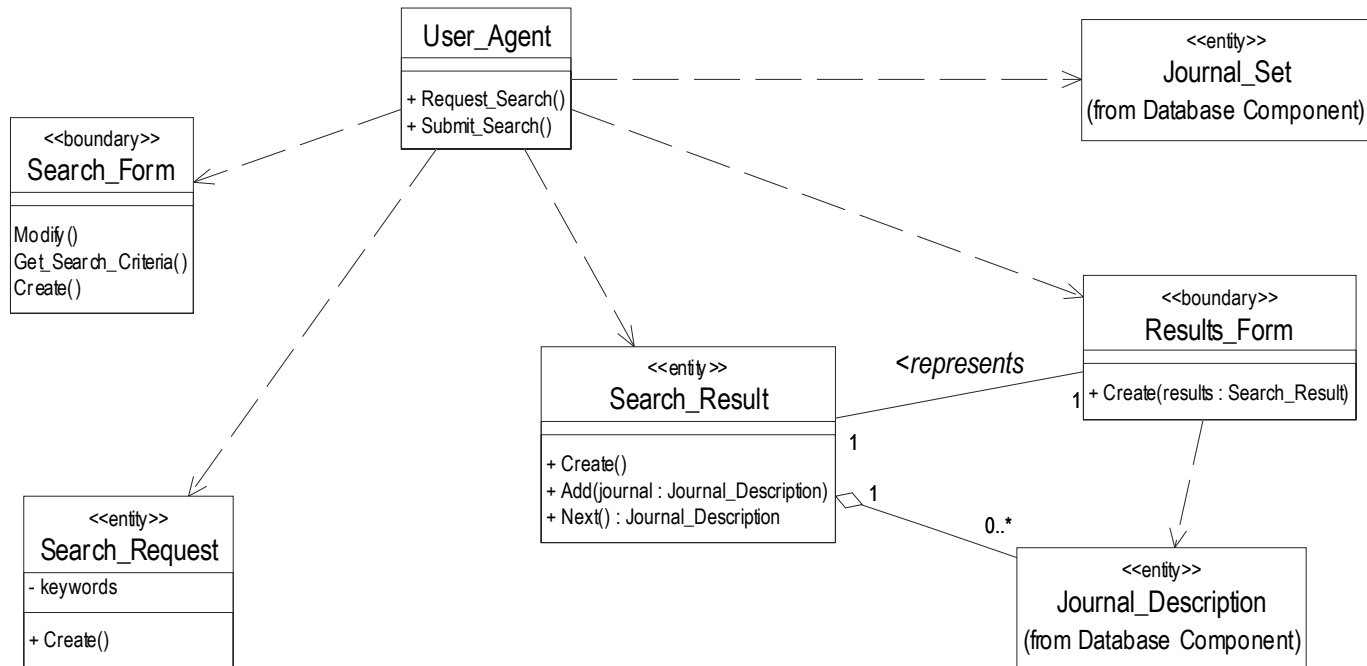
## Refined Object Model For Server::Database Component

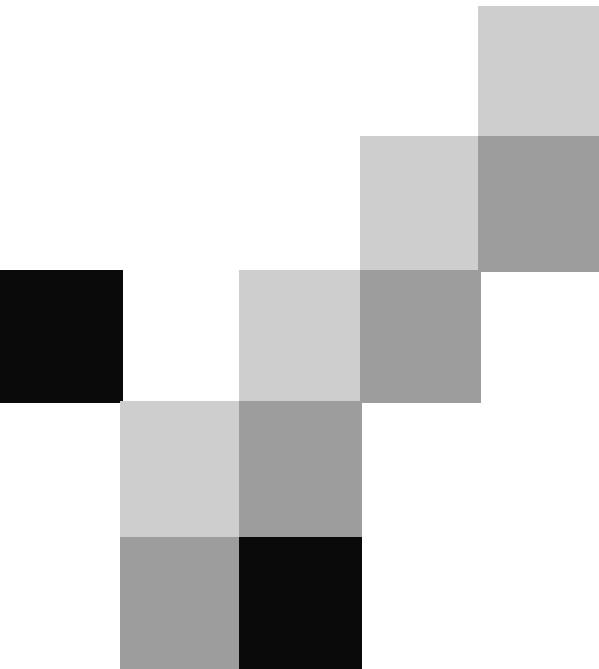


### ■ Based on Interaction Model

# Define Objects – LCO or LCA

## Updated Logical Class Model For Library\_User Component





End of System Design.  
When next we meet,  
we'll discuss  
Implementation Design