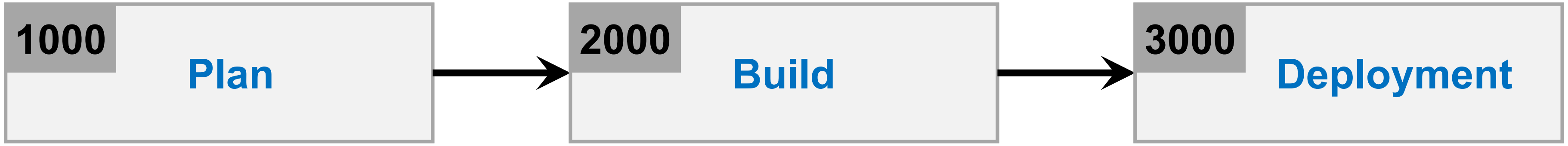
OOPT
(Object-Oriented Process with Traceability)

What is OOPT?

- OOPT (Object-Oriented Process with Traceability)
  - A software development process based on RUP
    - Have been practiced and refined over 10 years
  - Tailored to software engineering classes in universities
    - No risk analysis for architecture : No elaboration phase in UP

- Characteristics of OOPT
  1. 3 Stages
     - Based on the RUP
  2. Iterative
     - Multiple development cycles
  3. Incremental
     - System grows incrementally as each cycle is completed.
  4. Hierarchical Architecture
     - Stage > Cycle > Phase > Activity

1. 3 Stages



1000 Plan → 2000 Build → 3000 Deployment

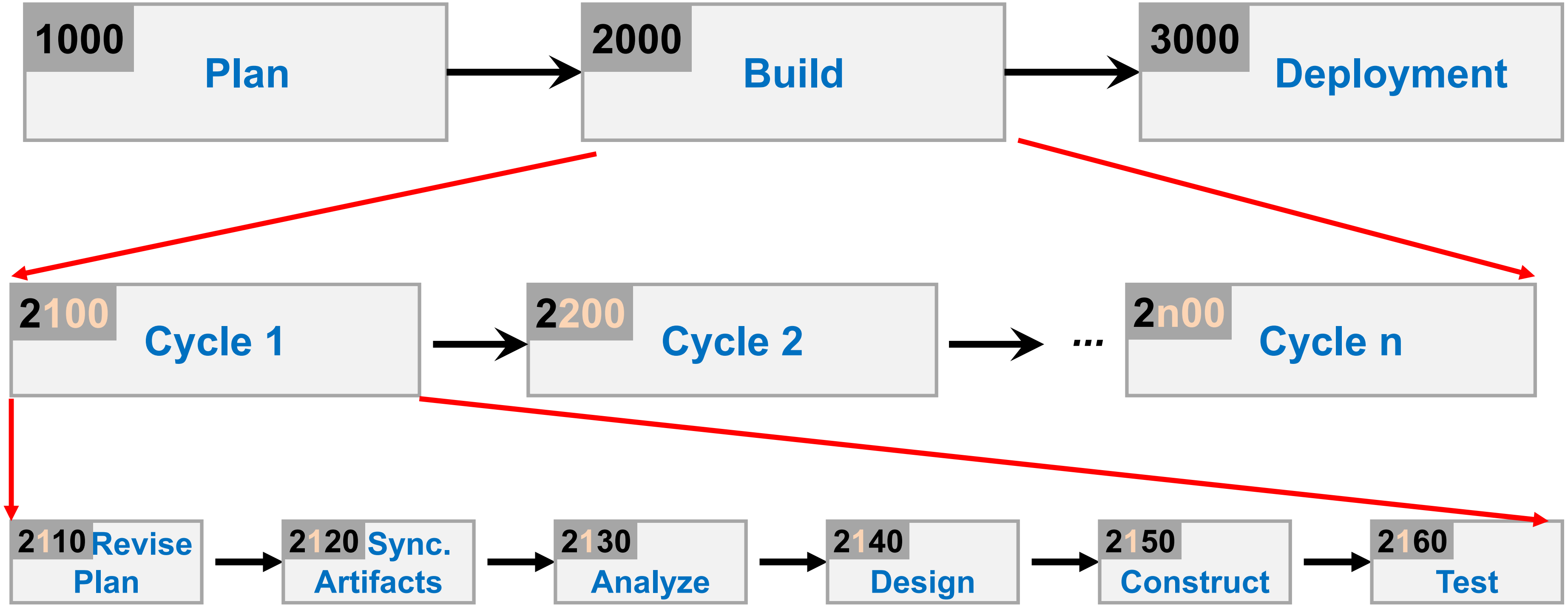
- Stage 1000 : Plan
  - Planning, defining requirements, building prototyping, etc
  - Corresponding to Inception phases in the RUP

- Stage 2000 : Build
  - Elaboration and Construction of the system
  - Corresponding to Elaboration/Construct phase in the RUP

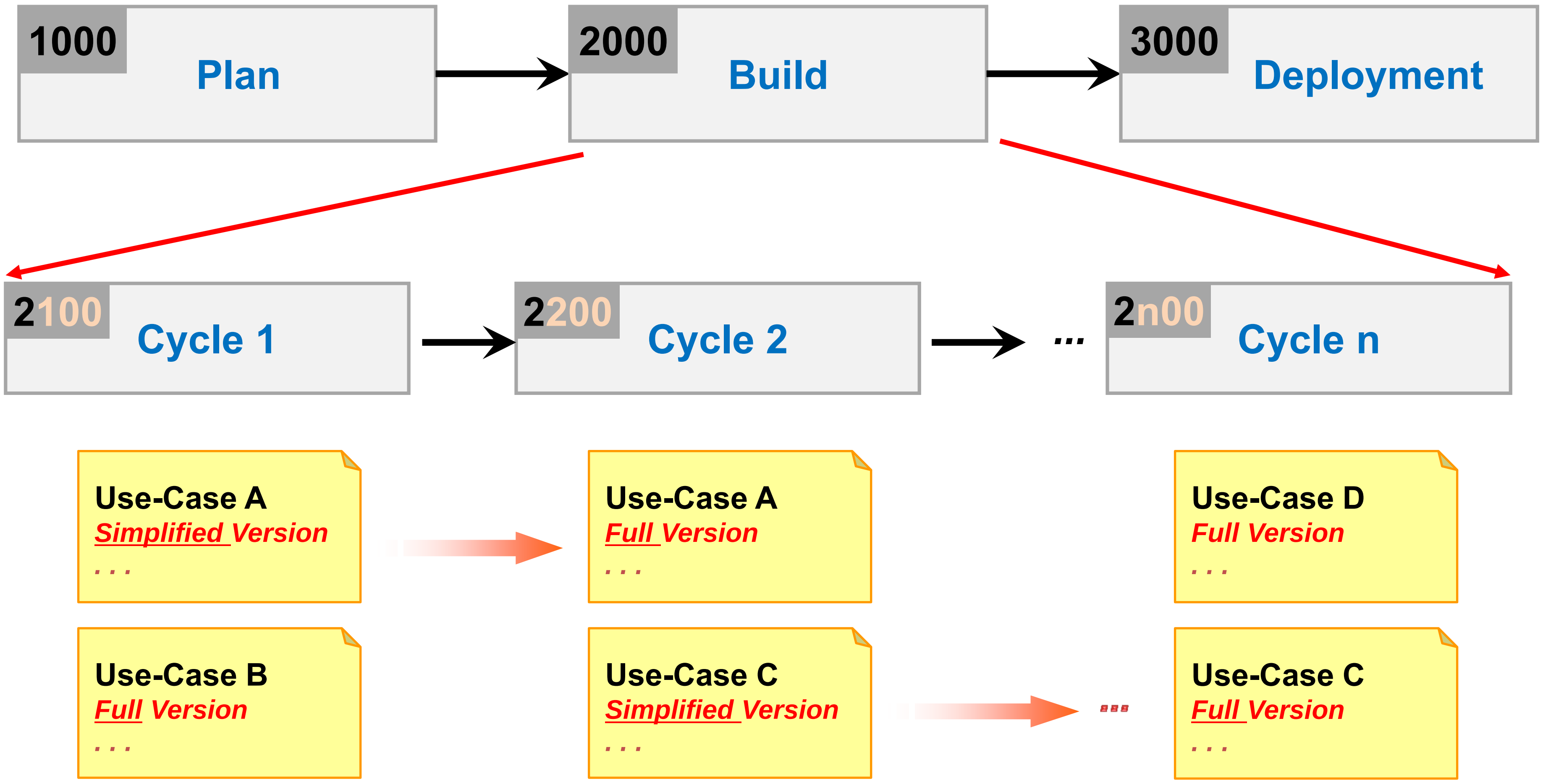
- Stage 3000 : Deployment
  - Implementation of the system into use
  - Corresponding to Transition phase in the RUP

2. Iterative Development

- Multiple iterations in the Build stage
  - Each iteration took about 2 to 4 weeks
  - Corresponding to **iterations** in RUP



3. Incremental Development



1000 Plan

2000 Build

3000 Deployment

2100 Cycle 1

2200 Cycle 2

…

2n00 Cycle n

Use-Case A
Simplified Version
. . .

Use-Case B
Full Version
. . .

Use-Case A
Full Version
. . .

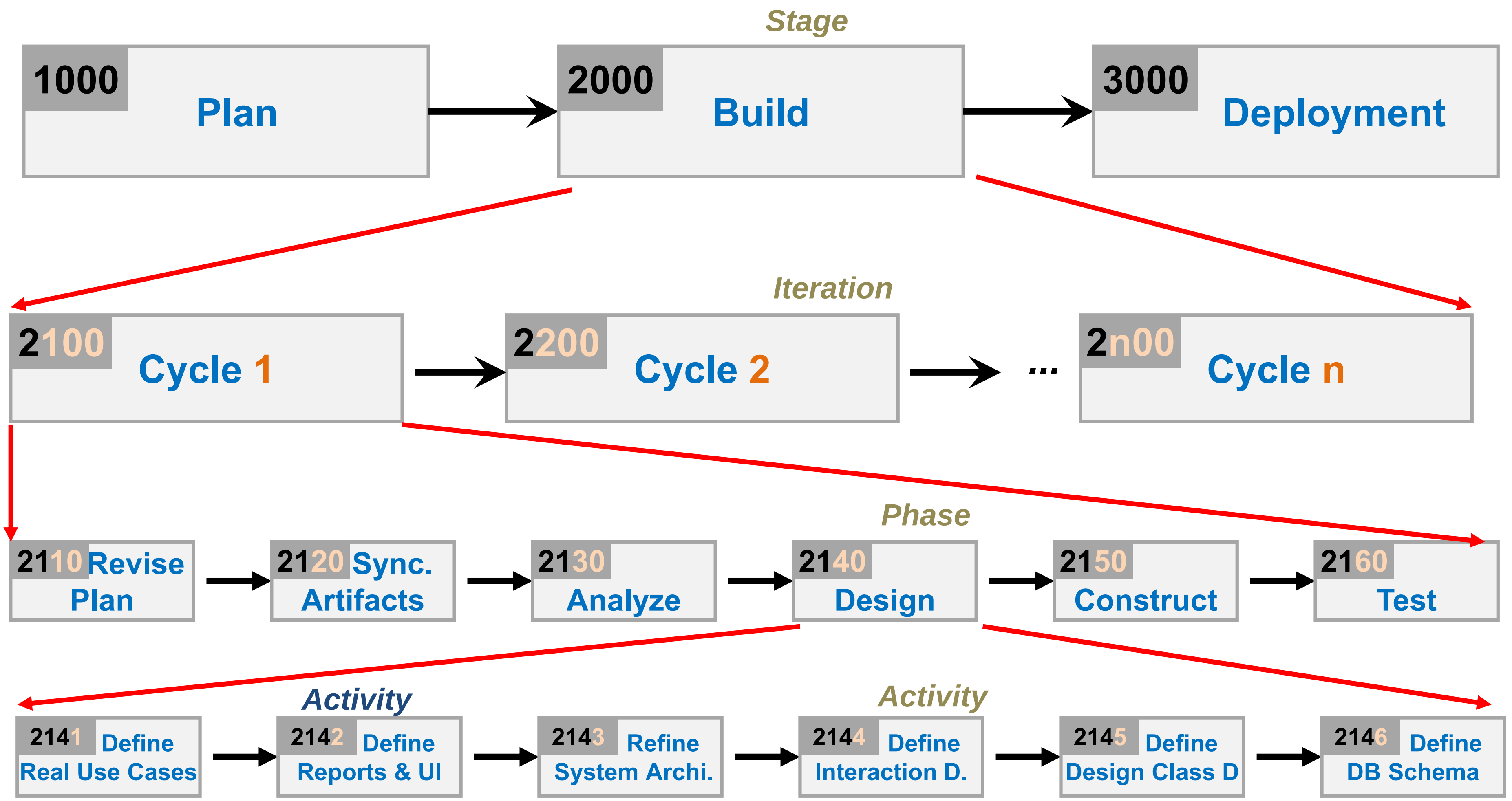
Use-Case C
Simplified Version
. . .

…

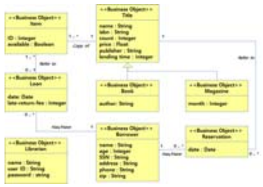
Use-Case D
Full Version
. . .

Use-Case C
Full Version
. . .

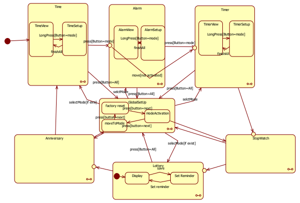
4. Hierarchical Architecture



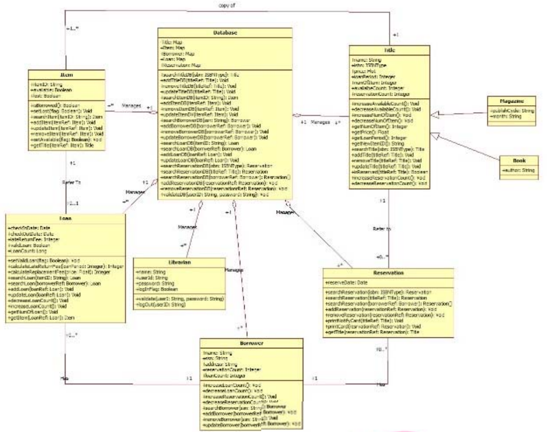
OOPT of OOAD, in Summary



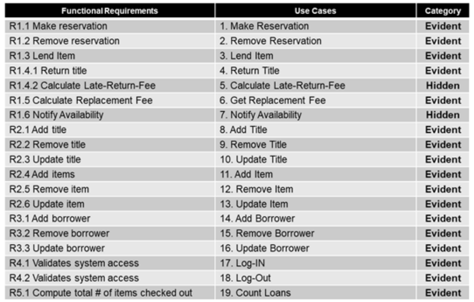
Domain Model



Statechart Diagram



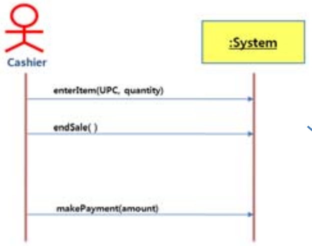
Class Diagram



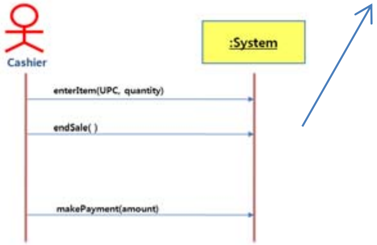
Functional Requirements	Use Cases	Category
R1.1 Make reservation	1. Make Reservation	Evident
R1.2 Remove reservation	2. Remove Reservation	Evident
R1.3 Lend Item	3. Lend Item	Evident
R1.4.1 Return title	4. Return Title	Evident
R1.4.2 Calculate Late-Return-Fee	5. Calculate Late-Return-Fee	Hidden
R1.5 Calculate Replacement Fee	6. Get Replacement Fee	Evident
R1.6 Notify Availability	7. Notify Availability	Hidden
R2.1 Add title	8. Add Title	Evident
R2.2 Remove title	9. Remove Title	Evident
R2.3 Update title	10. Update Title	Evident
R2.4 Add items	11. Add Item	Evident
R2.5 Remove item	12. Remove Item	Evident
R2.6 Update item	13. Update Item	Evident
R3.1 Add borrower	14. Add Borrower	Evident
R3.2 Remove borrower	15. Remove Borrower	Evident
R3.3 Update borrower	16. Update Borrower	Evident
R4.1 Validates system access	17. Log-IN	Evident
R4.2 Validates system access	18. Log-Out	Evident
R5.1 Compute total # of items checked out	19. Count Loans	Evident

User (Functional) Requirements

Use Cases



:

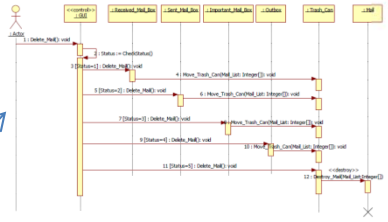


System Sequence Diagrams

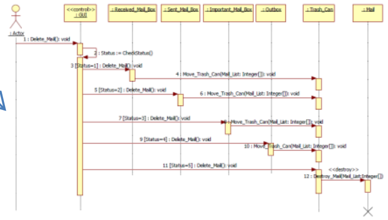
System

+selectTimeViewMode()
+selectTimeSetupMode()
+changeValue()
+goNext()
+selectAlarmViewMode()
+selectAlarmSetUpMode()
+addAlarm()
+deleteAlarm()
+clearAlarmNotice()
+setValue()
+startTimer()
+pauseTimer()
+resetTimer()
+clearTimerNotice()
+startStopWatch()
+stopStopWatch()
+restartStopSwatch()
+resetStopWatch()
+createNewAnniversary()
+inputDateTime()
+selectAnniversary()
+deleteAnniversary()
+alert()
+dismiss()
+requestCreateLotteryNumber()
+saveLotteryNumber()
+setReminder()
+select4Mode()
+requestFactoryReset()
+requestChangeCurrentMode()

System Operations



:



Sequence Diagrams

Traceability Table

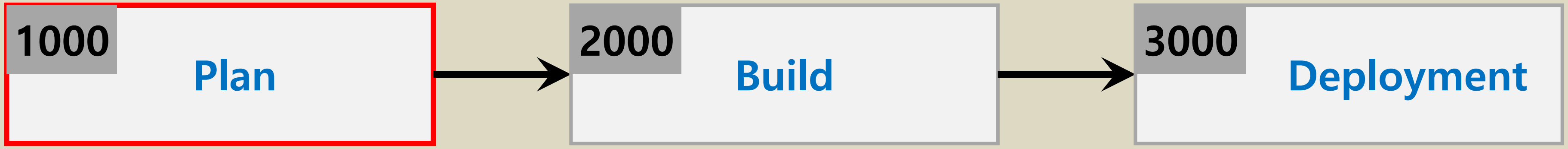
Stage 1000. Plan

Stage 2030. Analyze

Stage 2040. Design

DEPENDABLE SOFTWARE LABORATORY

Stage 1000. Plan



1000 Plan

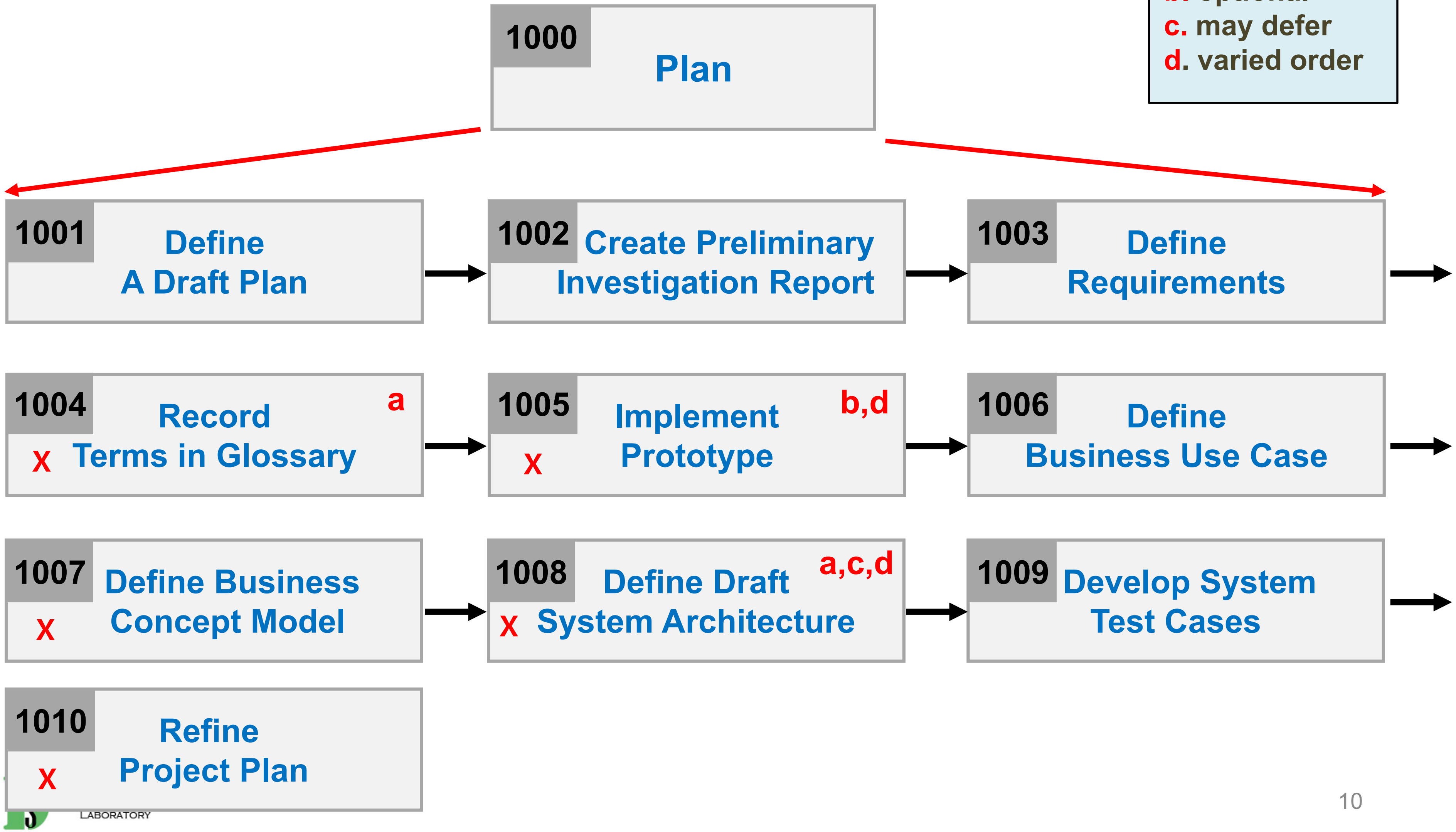
2000 Build

3000 Deployment

Stage 1000. Plan

- 10 Activities

a. ongoing
b. optional
c. may defer
d. varied order



1000 Plan

1001 Define A Draft Plan

1002 Create Preliminary Investigation Report

1003 Define Requirements

1004 Record Terms in Glossary a x

1005 Implement Prototype b,d x

1006 Define Business Use Case

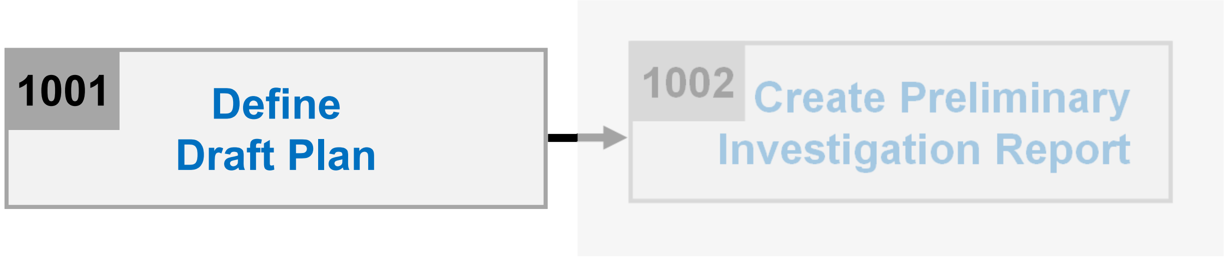
1007 Define Business Concept Model x

1008 Define Draft System Architecture a,c,d x

1009 Develop System Test Cases

1010 Refine Project Plan x

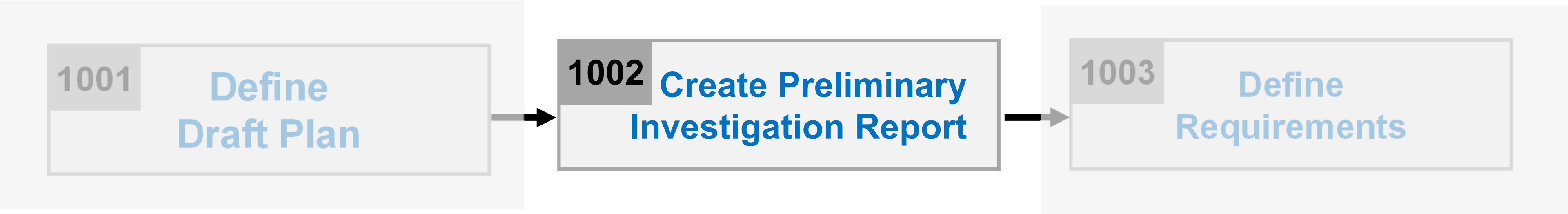
Activity 1001. Define A Draft Plan



- Description
  - Write a draft plan for schedule, resources, budget, objective, etc
  - Input : all related documents of previous similar projects
  - Output : **a draft project plan**

- Steps
  1. Write motivation and objective of project
  2. Write scope of project
  3. Identify and write functional requirements
  4. Identify and write non-functional requirements
  5. Estimate resources (human efforts(M/M), human resources, duration, budget)

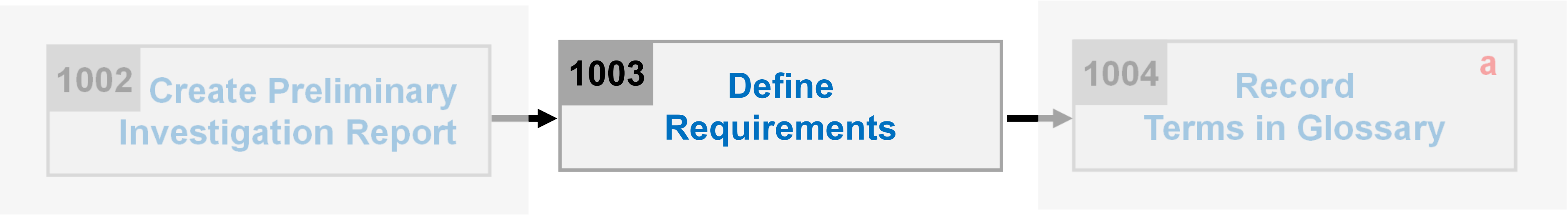
Activity 1002. Create Preliminary Investigation Report



- Description
  - Write an investigation report on alternatives, business needs, risk, etc
  - Input : draft project plan
  - Output : **an investigation report**

- Steps
  1. Write alternative solutions
  2. Write project’s justification (business needs)
  3. Identify and manage risks, and write risk reduction plans
  4. Analyze business market
  5. Write managerial issues

Activity 1003. Define Requirements



- Description
  - Write a requirement specification for a product
  - Input : draft project plan, investigation report
  - Output : **a requirement specification**

- What is a requirement? **(IEEE Std 610.12-1990)**
  - A condition or capability needed by a user to solve a problem or achieve an objective.
  - A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed documents.
  - A documented representation of a condition or capabilities as in (1) or (2)

Activity 1003. Define Requirements

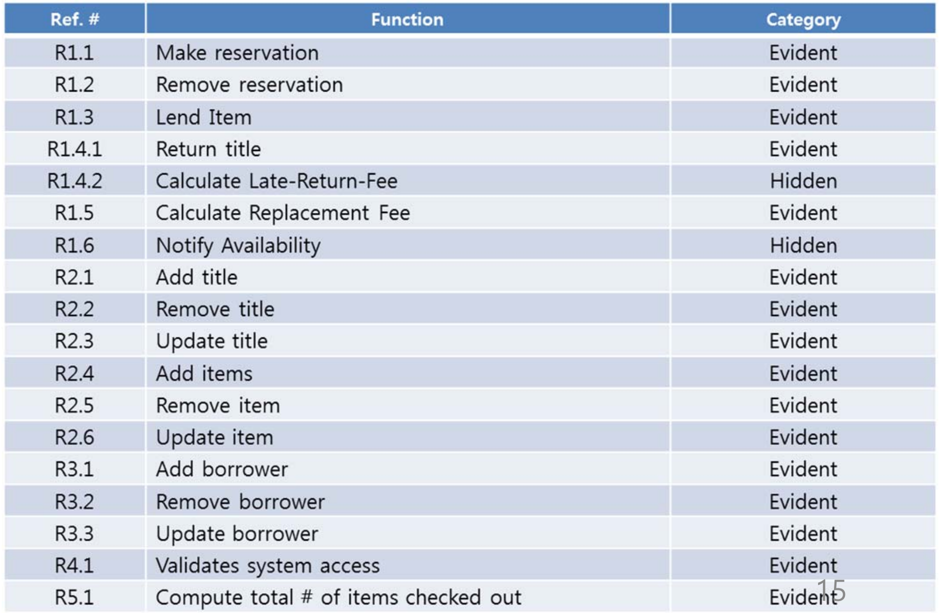
- **Functional requirements**
  - A requirement that specifies a function that a system or system component must be able to perform
  - Analyzed and Realized in Use-Case model, later

- **Non-functional requirements**
  - Constraints on the services or functions offered by the system as timing constraints, constraints on the development process, standards, etc.
  - Portability, Reliability, Usability, Efficiency(Space, Performance)
  - Delivery, Implementation, Standards
  - Ethical, Interoperability, Legislative(Safety, Privacy)

- Recommended reference : IEEE Std. 830-1998

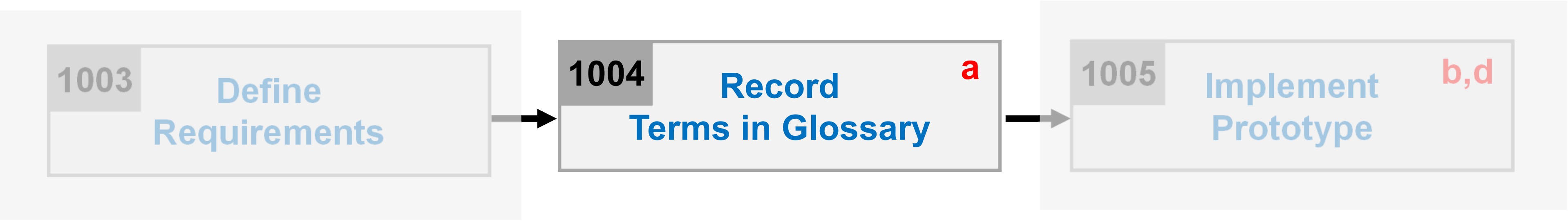
Activity 1003. Define Requirements

- Steps for defining requirements
  1. Gather all kinds of useful documents
  2. Write an overview statement (objective and name of the system, etc.)
  3. Determine customers who use the product
  4. Write goals of the project
  5. Identify system functions
     - Functional requirements
     - Add function references(such as R1.1, …) into the identified functions
     - Categorize identified functions into Event, Hidden, and Optional
  6. Identify system attributes
     - Non-functional requirements
  7. Identify other requirements (Optional)
     - Assumptions, Risks, Glossary, etc.



Ref. #	Function	Category
R1.1	Make reservation	Evident
R1.2	Remove reservation	Evident
R1.3	Lend Item	Evident
R1.4.1	Return title	Evident
R1.4.2	Calculate Late-Return-Fee	Hidden
R1.5	Calculate Replacement Fee	Evident
R1.6	Notify Availability	Hidden
R2.1	Add title	Evident
R2.2	Remove title	Evident
R2.3	Update title	Evident
R2.4	Add items	Evident
R2.5	Remove item	Evident
R2.6	Update item	Evident
R3.1	Add borrower	Evident
R3.2	Remove borrower	Evident
R3.3	Update borrower	Evident
R4.1	Validates system access	Evident
R5.1	Compute total # of items checked out	Evident

Activity 1004. Record Terms in Glossary

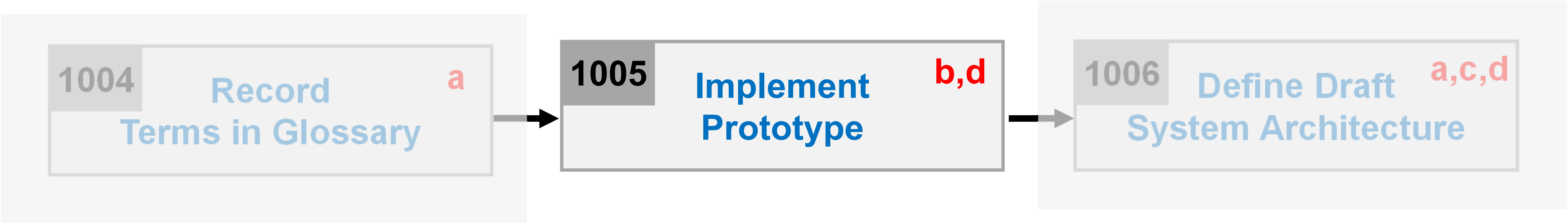


- Description
  - Similar to “Data Dictionary”
  - Dictionary of terms and any associated information(constraints and rules)
  - Input : requirements specification
  - Output : **a term dictionary (glossary)**

- Steps
  1. Describe meaning of terms specified in requirements specification
  2. Write alias of each term

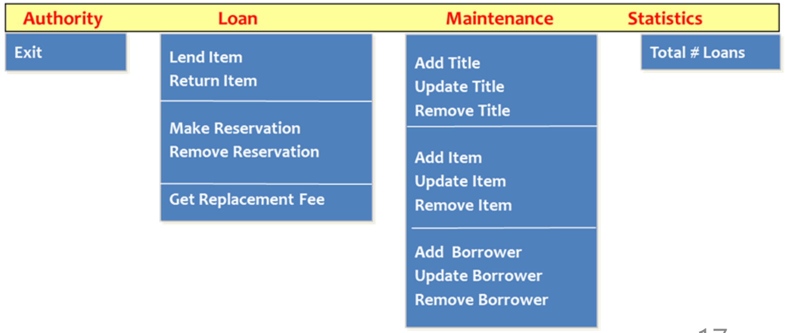
Term	Description	Remarks
Title	Books or Magazines, which are registered in the library system	
Item	Each copy of books or magazines	
Loan	An action of checking out an item from the library	
Librarian	An employee of the library who handles the requests of borrowers.	

Activity 1005. Implement Prototype

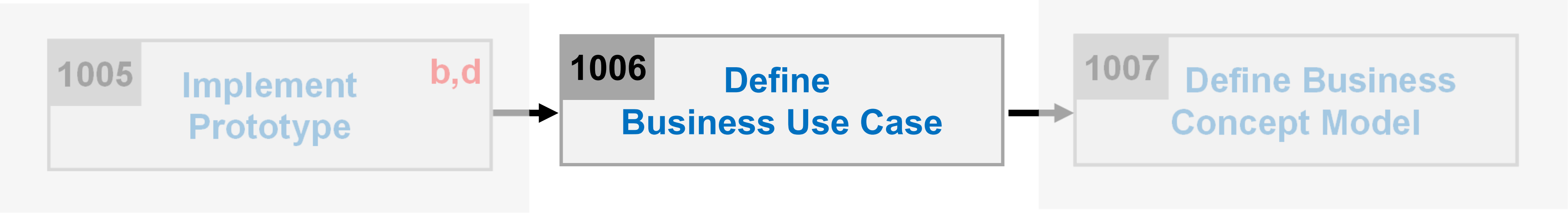


- Description
  - Develop a prototype system to permit use feedback, determine feasibility, or investigate timing or other issues
  - Input : requirements specification
  - Output : **a prototype**

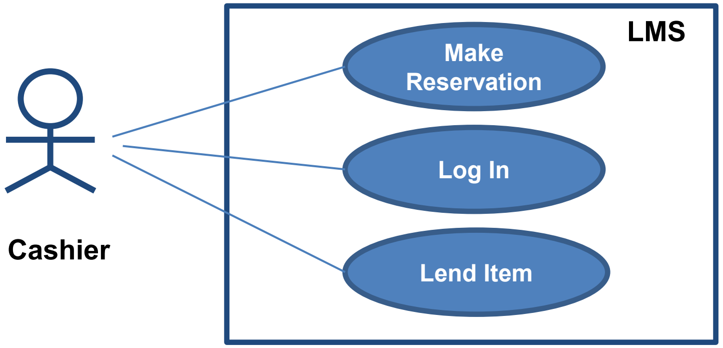
- Steps
  1. Develop a prototype promptly and efficiently



Activity 1006. Define Business Use Case

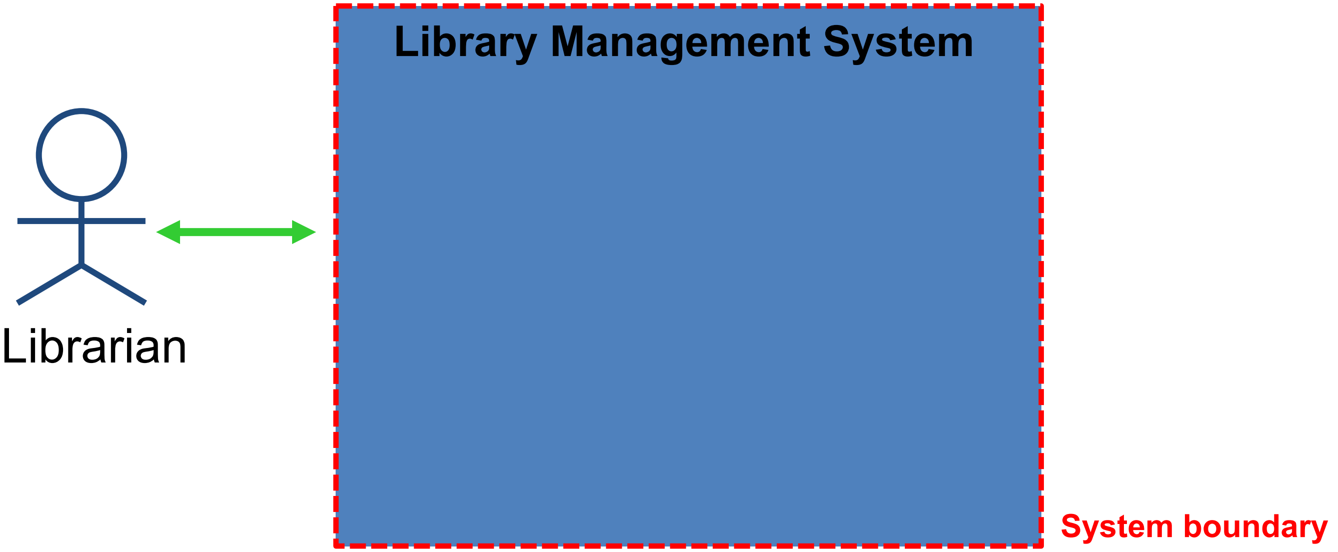


- Description
  - To obtain a deeper understanding of the processes and requirements identified so far
  - Identify business tasks as business use cases, and illustrate their relationships in use case diagrams
  - Input : requirements specification
  - Output : **a business use case model (the brief format of UP)**
    - Business Use Case Diagram and Description



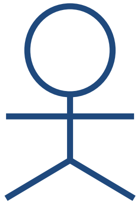
Activity 1006. Define Business Use Case

- Steps
  1. Determine system boundary in order to identify what is external versus internal, and what the responsibilities of the system are
     - Typical system boundary includes:
       - Hardware/Software boundary of a device / computer system
       - Department of an organization
       - Entire organization



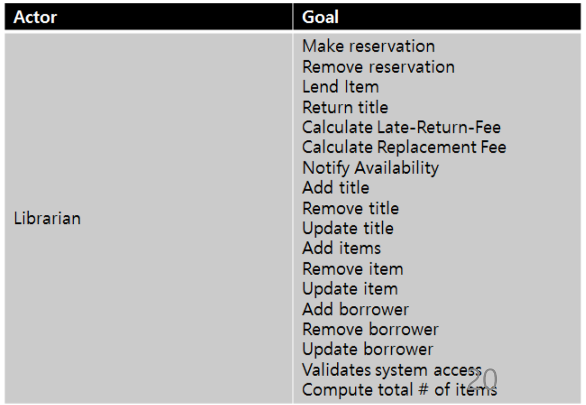
Activity 1006. Define Business Use Case

2. Identify the actors related to a system or organization
    - An actor is anything with behavior, including the system under discussion(SuD) itself when it calls upon the services of other systems
    - Actors are not only the roles played by people, but also organizations, software, and machines
    - Primary Actors
        - Have user's goals fulfilled through using services the system provides
        - Primary actors can be other computer systems (i.e. watchdog)
    - Supporting Actors
        - Provide services to the system under design
        - Often a computer system could be a supporting actor



Actor

3. Identify user goals for each actor
4. Record the primary actors and their goals in an actor-goal list



Actor	Goal
Librarian	Make reservation Remove reservation Lend Item Return title Calculate Late-Return-Fee Calculate Replacement Fee Notify Availability Add title Remove title Update title Add items Remove item Update item Add borrower Remove borrower Update borrower Validates system access Compute total # of items

Activity 1006. Define Business Use Case

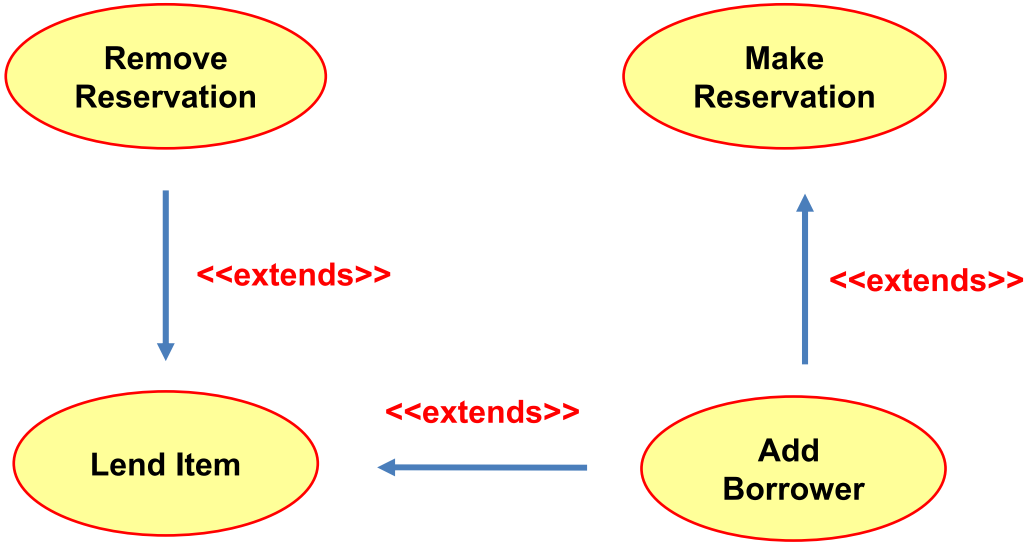
5. Define use cases that satisfy user goals
   - Identify use cases by actor-based
     - For each actor, identify the processes they initiate or participate in
   - Identify use cases by event-based
     - Identify the external events that a system must respond to
     - Related the events to actors and use cases
   - Name them according to their goals

6. Allocate system functions identified during the requirements specification into related use cases

7. Categorize identified use cases into primary, secondary, and optional use cases
   - Primary use cases : major common processes
   - Secondary use cases : minor or rare processes
   - Optional use cases : processes that may not be tackled

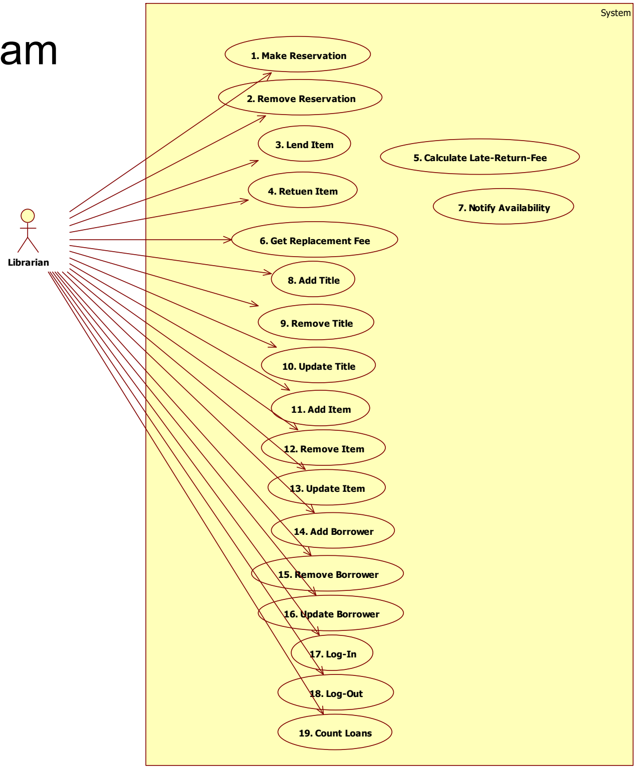
Activity 1006. Define Business Use Case

8. Identify relationships between use cases
   - Write major steps or branching activities of one use case as several separate use cases using “include” relationship, when they are too complex, long, and duplicated to understand
   - Use “extends” relationship when an exceptional activity is occurred in use case (Optional)



Activity 1006. Define Business Use Case

9. Draw a use case diagram



10. Describe use cases

- Describe the detail information of use cases
  - Name, Actor, Description (the brief format of UP)

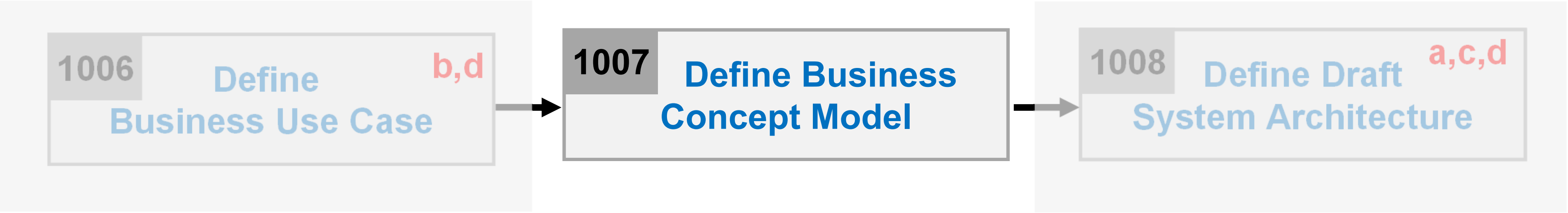
Use Case	The name of use case
Actors	Associated actor
Description	Abstract information of use case

Activity 1006. Define Business Use Case

11. Rank use cases according to the followings:
    a. Significant impact on the architectural design
    b. Significant information and insight regarding the design
    c. Include risky, time-critical, or complex functions
    d. Involve significant research, or new and risky technology
    e. Represent primary line-of-business processes
    f. Directly support increased revenue or decreased costs
        - The ranking scheme may use a simply fuzzy classification such as high-medium-low
        - High ranking use cases need to be tackled in early development cycle

Rank	Use case	Justification
High	Make Reservation …	It reserves the item of the title …
Medium	Validates system access …	Affects security …
Low		

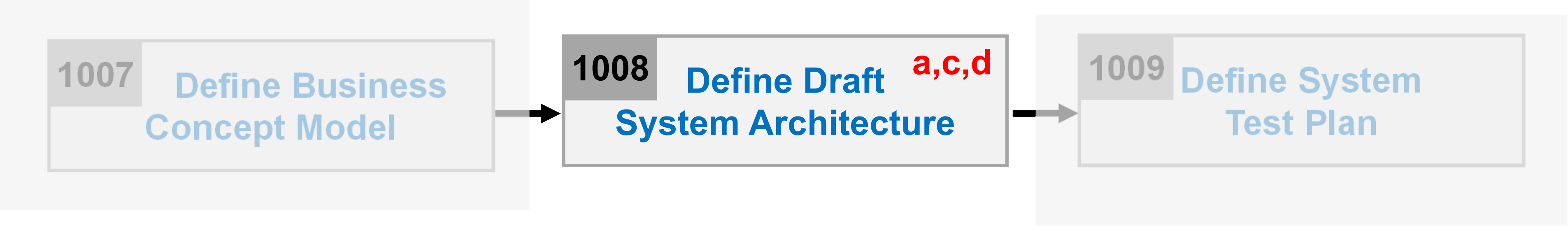
Activity 1007. Define Business Concept Model


1006
Define Business Use Case
b,d
1007
Define Business Concept Model
1008
Define Draft System Architecture
a,c,d

- Description
  - Identify ”business concept” in the target domain which can be candidates for “classes”
  - Input : requirements specification, data dictionary, business use case
  - Output : **a business concept model**

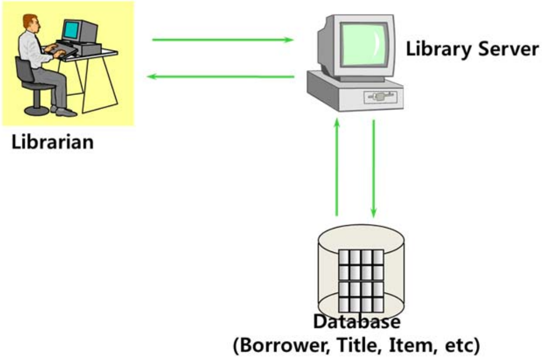
- Steps
  1. Identify business terms or business concepts from requirements specification or through interviews with domain experts
  2. Define identified terms as business concepts
     - Implementation details can’t be business concepts

Activity 1008. Define Draft System Architecture

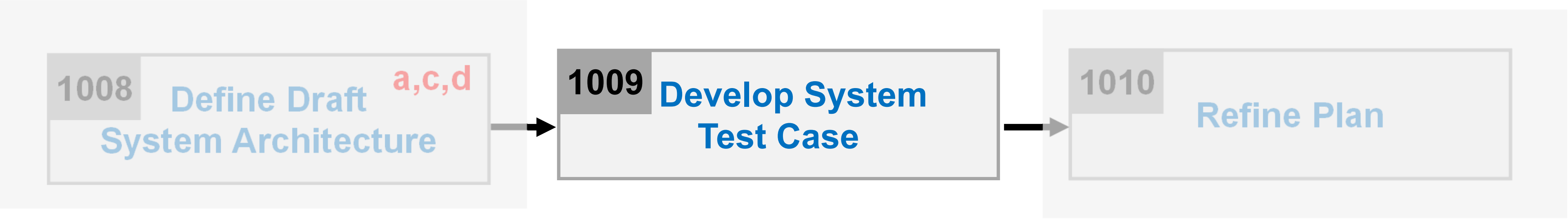

1007
Define Business Concept Model
1008
Define Draft System Architecture
a,c,d
1009
Define System Test Plan

- Description
  - Construct a rough preliminary system architecture model
  - Input : requirements specification
    business use case model
  - Output : **a draft system architecture**

- Steps
  1. Define logical/physical layers of the target system
  2. Separate the whole system into several subsystems
  3. Assign business use cases into each subsystem
  4. Identify and draw up hardware resources



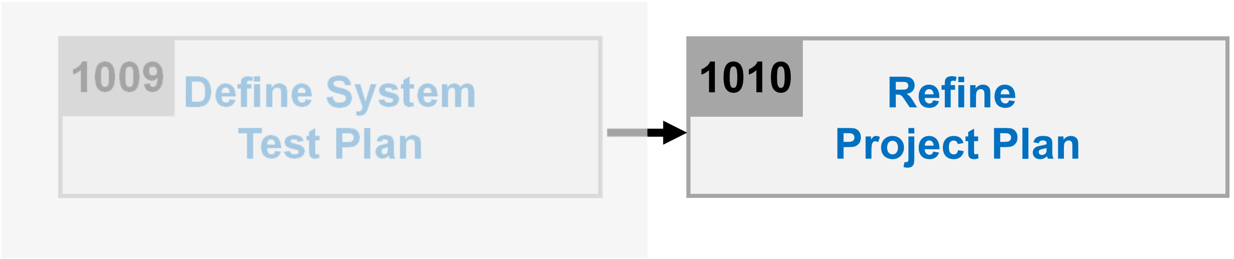
Activity 1009. Develop System Test Case



- Description
  - Develop system test cases
  - Input : requirements specification, business use case, business concept model
  - Output : **a system test plan**

- Steps
  1. Identify important requirements which should be tested later
  2. Develop system test cases with various system testing techniques
     - Category partitioning testing, brute force testing, boundary values, etc.
  3. Check the correspondence between the requirements and system test cases
     - Confirm 100% requirements coverage through tracing all relevant elements

Activity 1010. Refine Project Plan

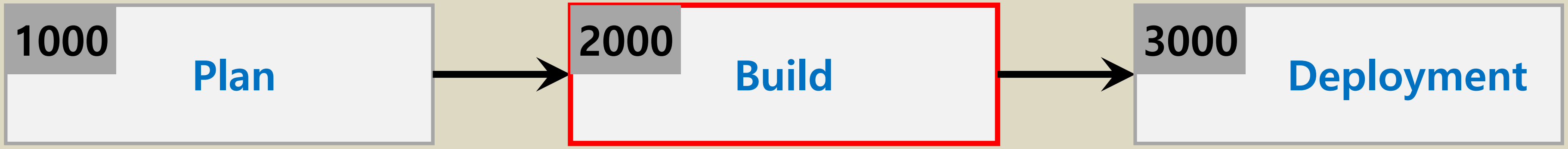


- Description
  - Refine the draft project plan
  - Input : all outputs from OOPT stage 1,000
  - Output: **a refined project plan**

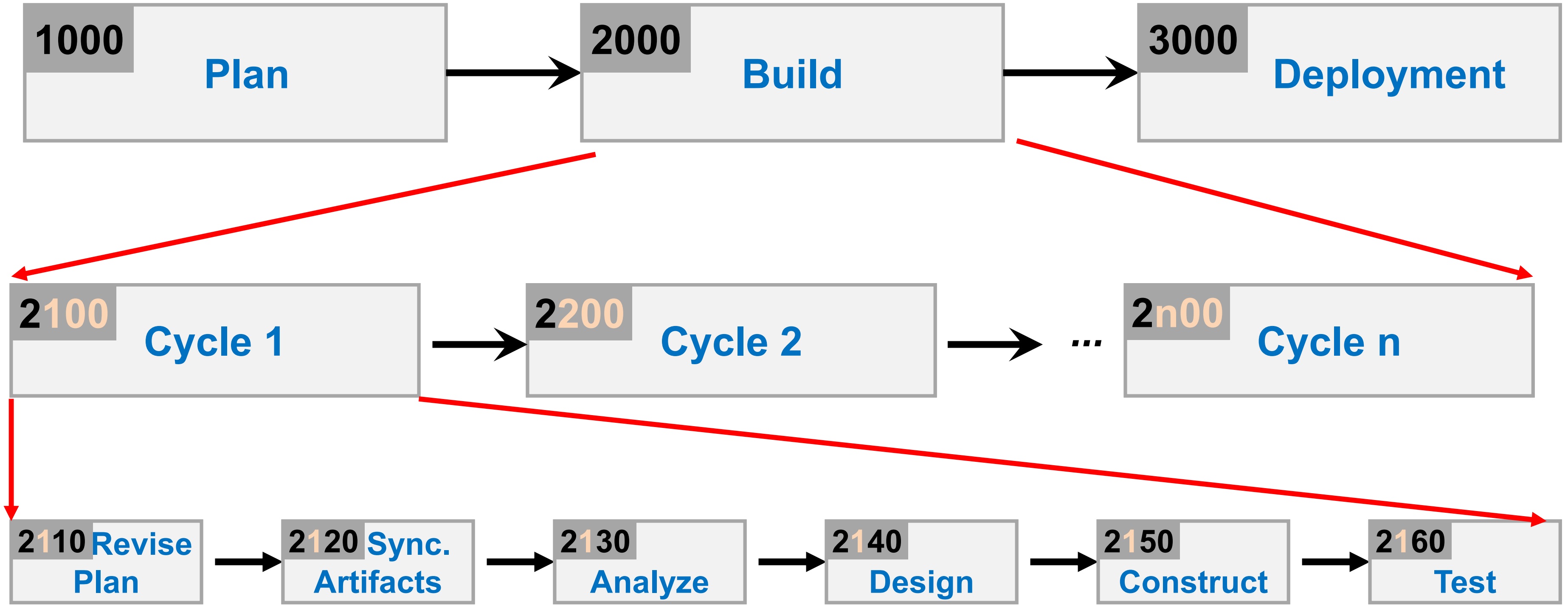
- Steps
  1. Review draft project plan, based on requirements specification, business use case, business concept model and draft system architecture
  2. Refine project's scope, duration, cost, and other resources

Back

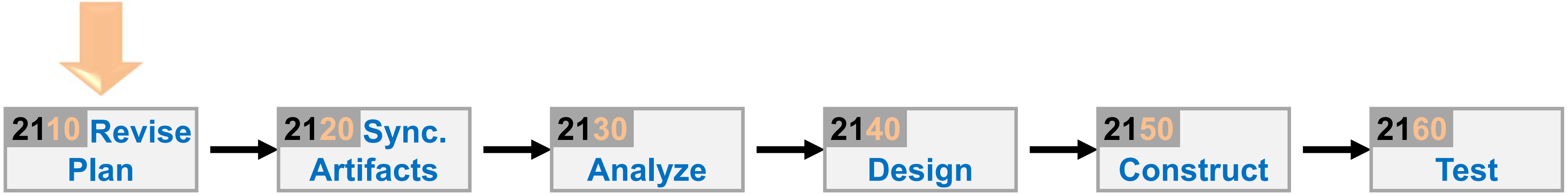
Stage 2000. Build



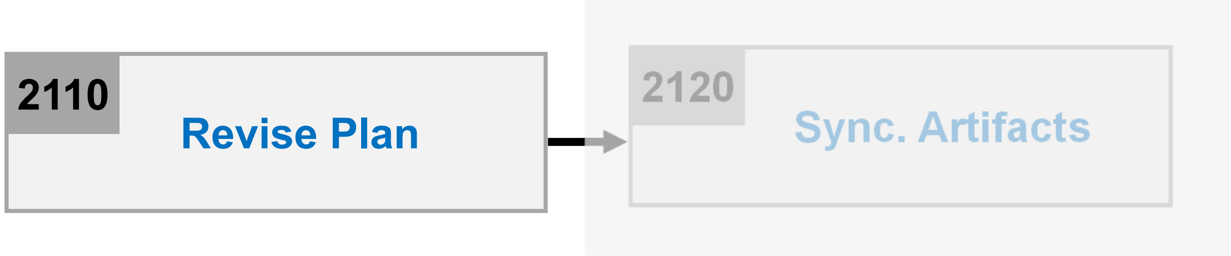
6 Phases of 'Build' Stage



Phase 2010. Revise Plan

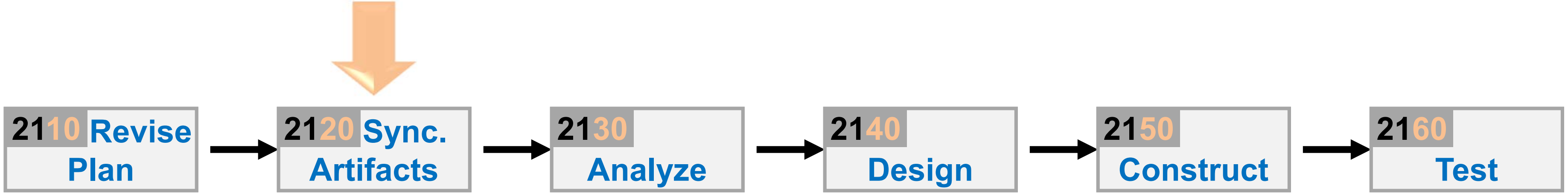


Phase 2010. Revise Plan



- Description
  - Correct and enhance the project plan and requirement definition based on the intermediate deliverables
  - Input : intermediate deliverables
  - Output : a refined project plan

Phase 2020. Synchronize Artifacts



2110 Revise Plan

2120 Sync. Artifacts

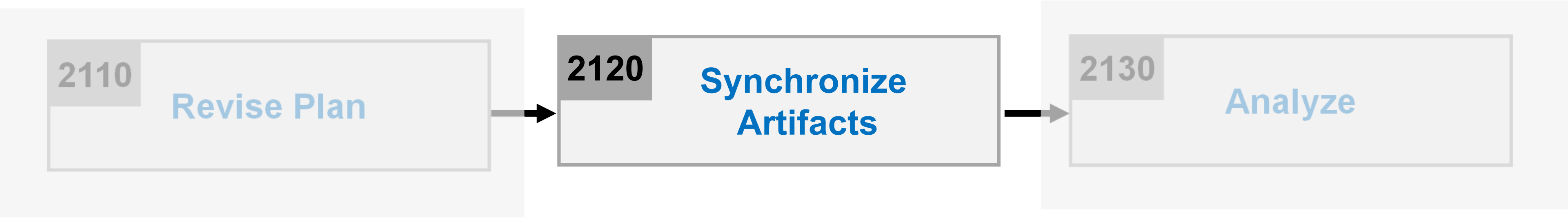
2130 Analyze

2140 Design

2150 Construct

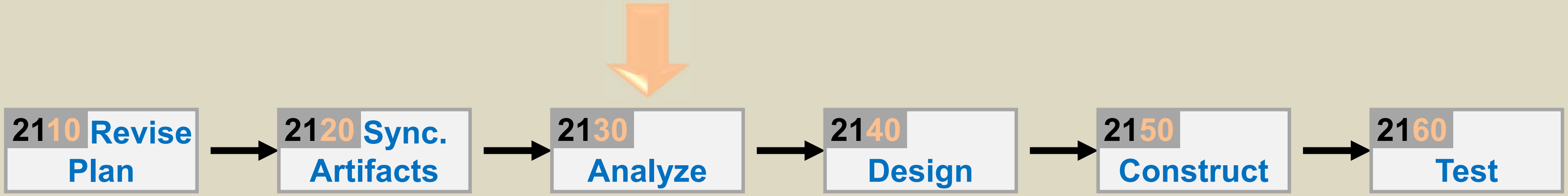
2160 Test

Phase 2020. Synchronize Artifacts



- Description
  - Configure and manage various types of artifacts (Project Repository)
  - Control versions and variations
  - Input : a refined project plan
  - Output : all outputs/deliverables revised

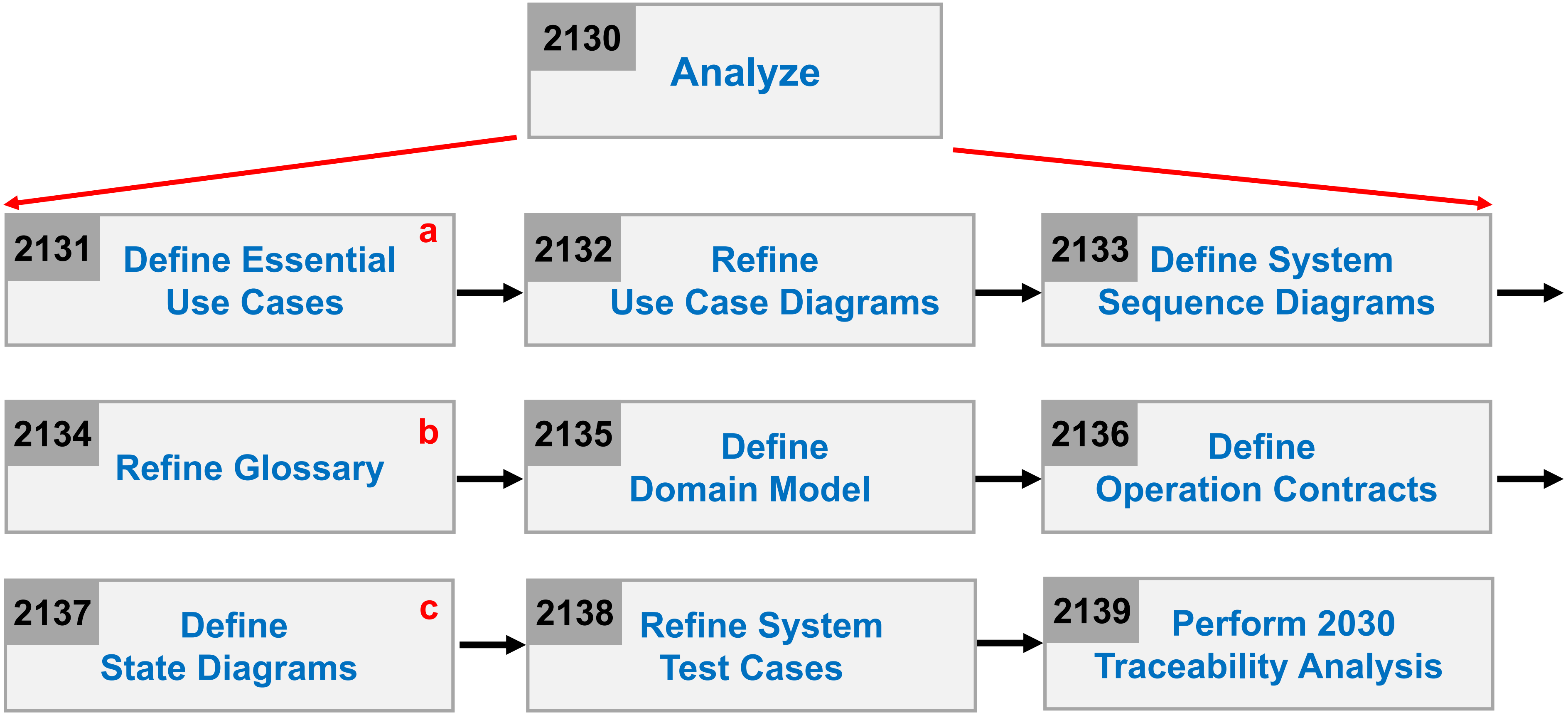
Phase 2030. Analyze



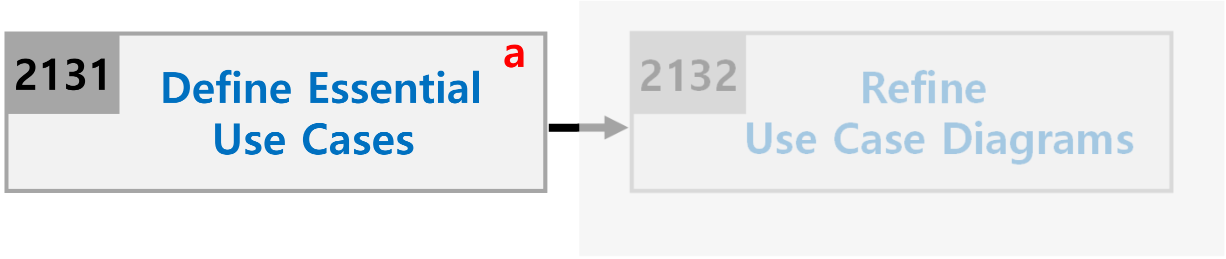
Phase 2030. Analyze

- Phase 2030 Activities

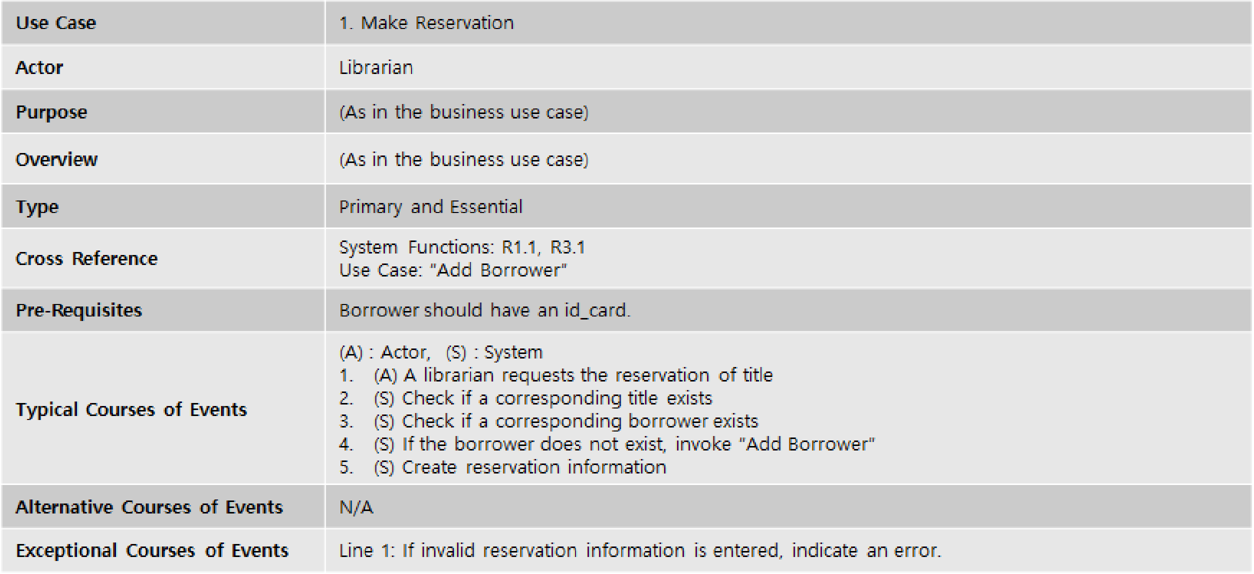
a. if not yet done
b. ongoing
c. optional


2130 Analyze
2131 Define Essential Use Cases
a
2132 Refine Use Case Diagrams
2133 Define System Sequence Diagrams
2134 Refine Glossary
b
2135 Define Domain Model
2136 Define Operation Contracts
2137 Define State Diagrams
c
2138 Refine System Test Cases
2139 Perform 2030 Traceability Analysis

Activity 2031. Define Essential Use Cases



- Description
  - Add event flows to business use case (high-level) descriptions
  - Input : business use case descriptions (activity 1006)
  - Output : **<u>An essential use case descriptions</u> (the causal format of UP)**
    - Standard applied : UML's expanded use case format

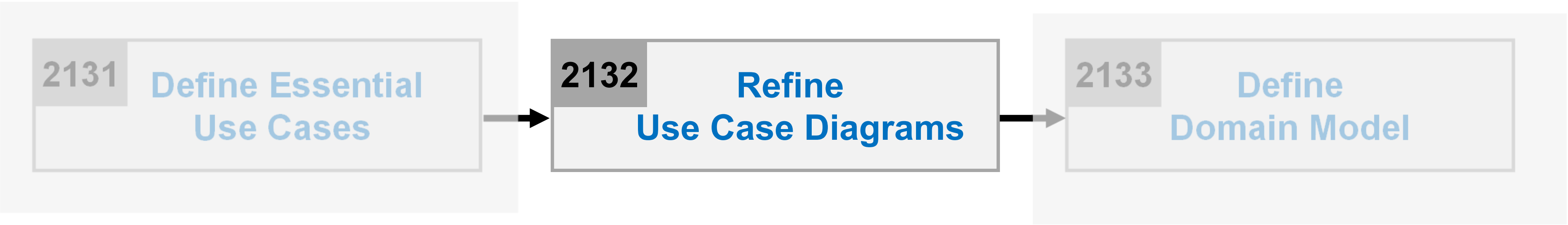


Use Case	1. Make Reservation
Actor	Librarian
Purpose	(As in the business use case)
Overview	(As in the business use case)
Type	Primary and Essential
Cross Reference	System Functions: R1.1, R3.1 Use Case: "Add Borrower"
Pre-Requisites	Borrower should have an id_card.
Typical Courses of Events	(A) : Actor, (S) : System 1. (A) A librarian requests the reservation of title 2. (S) Check if a corresponding title exists 3. (S) Check if a corresponding borrower exists 4. (S) If the borrower does not exist, invoke "Add Borrower" 5. (S) Create reservation information
Alternative Courses of Events	N/A
Exceptional Courses of Events	Line 1: If invalid reservation information is entered, indicate an error.

Activity 2031. Define Essential Use Cases

- Step
  1. Select each use case from business use cases
  2. Identify system functions related to the selected use case from requirements specification
  3. Identify related use cases to the selected use case from business use cases
  4. Identify courses of events for each use case from the requirements specification
     - Typical courses of events (main event flow)
     - Alternative courses of events
     - Exceptional courses of events
  5. Write essential use cases based on typical and alternative courses of events flows by applying expanded use case format
     - Use Case, Actor, Purpose , Overview
     - Type, Cross Reference, Pre-Requisites
     - Typical Courses of Events
     - Alternative/Exceptional Courses of Events

Activity 2032. Refine Use Case Diagrams

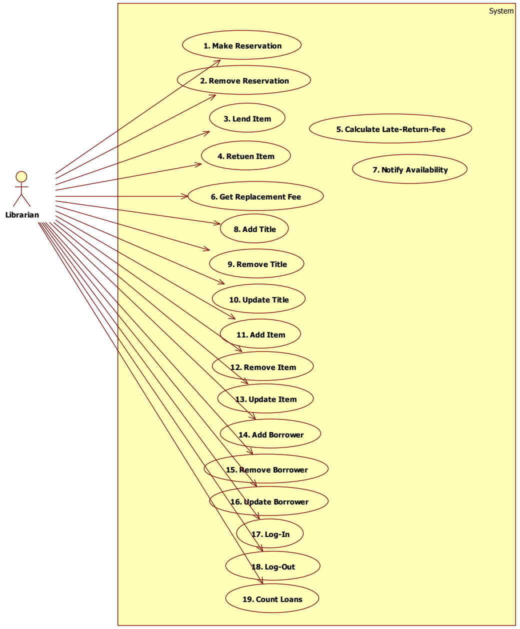


2131 Define Essential Use Cases

2132 Refine Use Case Diagrams

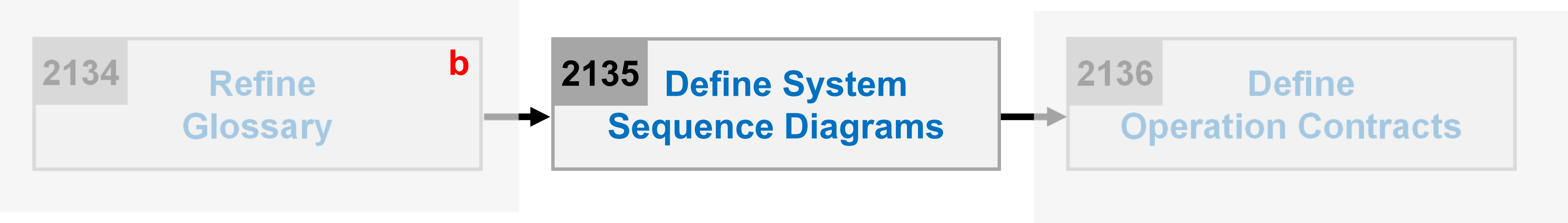
2133 Define Domain Model

- Description
  - – Validate and modify the ‘Business Use-Case Diagram’ (activity 1006)
  - – Input : business use case model, essential use case descriptions
  - – Output : **A refined use case diagram**
    - Standard applied : UML’s use case diagram



- Step
  1. Review business use case diagrams according to essential use case descriptions
  2. Refine use case diagrams by adding or refining use cases and relationships

Activity 2033. Define System Sequence Diagrams



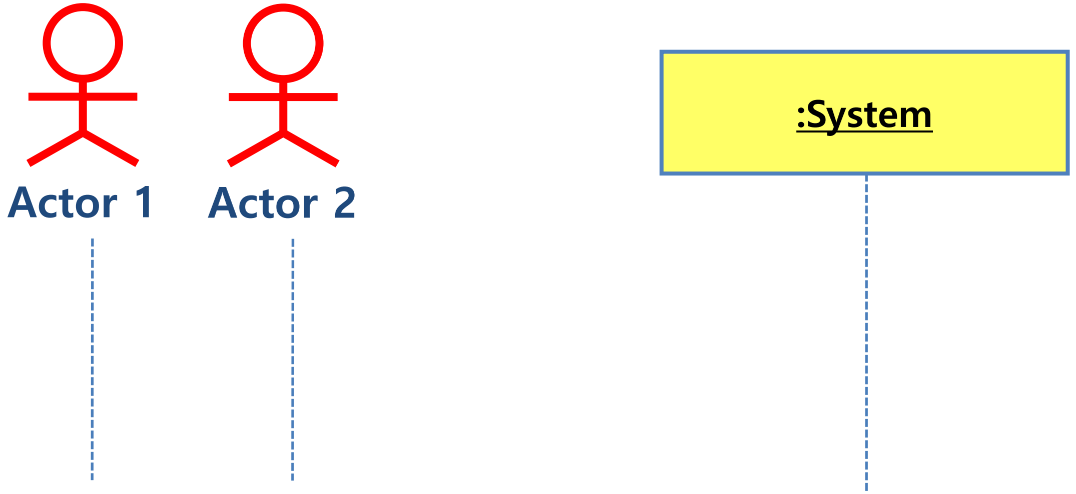
- Description
  - Illustrates events from actors to system under development
  - To investigate the system to build
  - Input : essential use case
  - Output : **A system sequence diagram**

Activity 2033. Define System Sequence Diagrams

- What is a **system sequence diagram (SSD)**?
  - A picture that shows the events that external actors generate, their orders and inter-system events
  - All systems are treated as a black box.
  - The emphasis of the diagram is events that cross the system boundary from actors to systems.
  - SSDs should be defined for
    - Main success scenarios, and then
    - Frequent, complex, or alternative scenarios

Activity 2033. Define System Sequence Diagrams

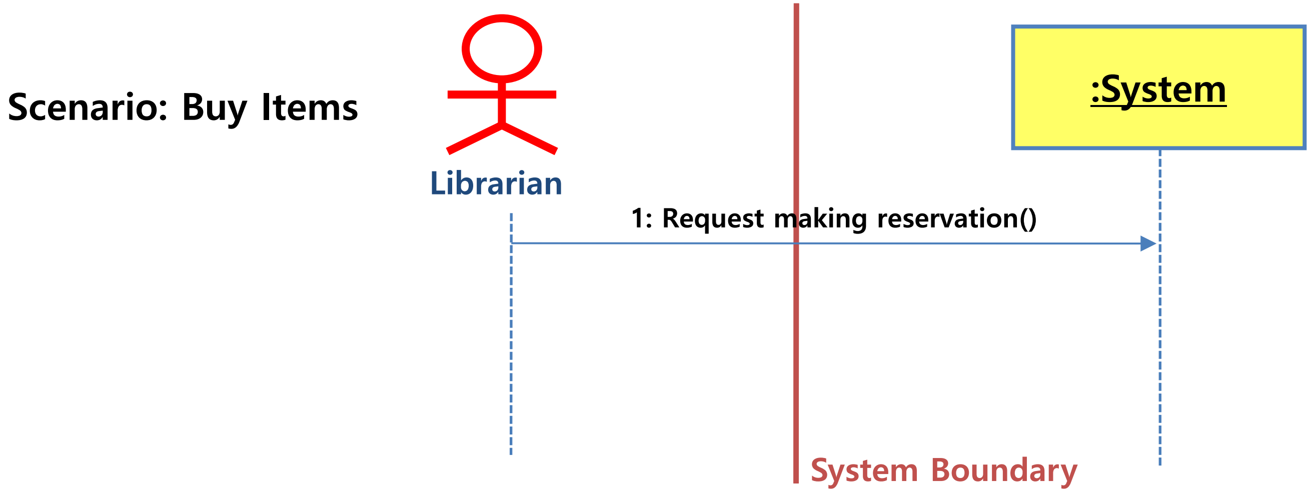
- Step
  1. Draw a black box representing the system based on a use case
  2. Identify each actor that directly operate on the system from the typical (normal) course of events in a use case
     - Draw a line for each actor



Activity 2033. Define System Sequence Diagrams

3. Determine system boundary
   - Hardware/software boundary of a device or computer system
   - Department of an organization or Entire organization

- Identify the system(external) events that each actor generates by according to typical course of events in a use case
- Name the system events
  - Should be expressed at the level of intent rather than of the physics
  - Name a system event with a verb and an objective like "*enterItem*"



Activity 2033. Define System Sequence Diagrams

4. Include the use case text which corresponds to system event to the left of the system sequence diagram

USE CASE: 1. Make Reservation

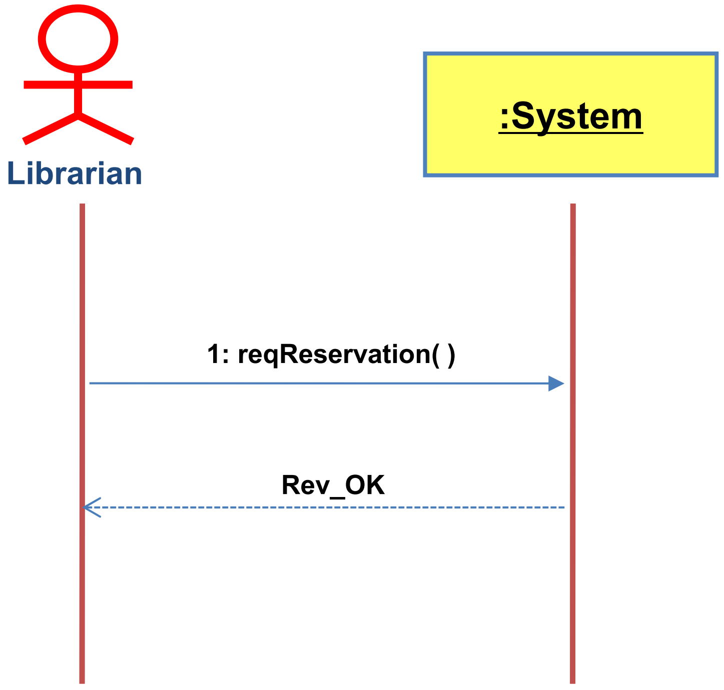
1. (A) A librarian requests the reservation of title.

2. (S) Check if corresponding title exist.

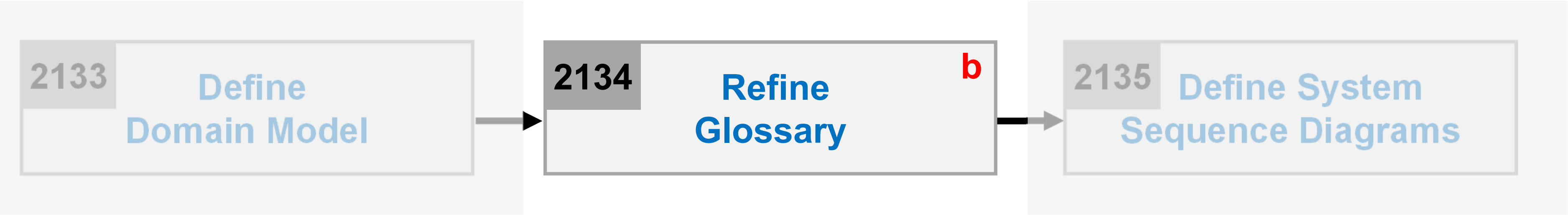
3. (S) Check if corresponding borrower exist.

4. (S) If the borrower does not exist, invoke “Add Borrower”.

5. (S) Create reservation information.



Activity 2034. Refine Glossary

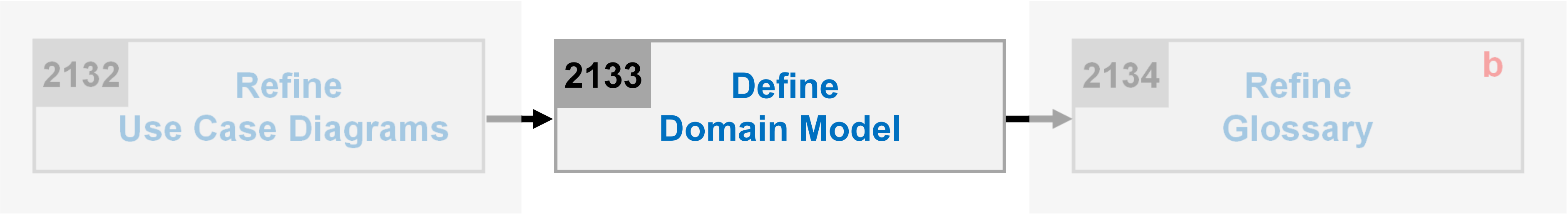


- Description
  - Lists and refines all the terms in order to improve communication and reduce the risk of misunderstanding
  - Input : term dictionary, essential use case descriptions, conceptual class diagram
  - Output : **A refined term dictionary (glossary)**

- Step
  1. Refine terms defined during the stage 1000 and 2000
  2. Record terms as following format:

Term	Category	Comments
Title	**Concept (Class)**	**A type of books or magazines which are registered in the library system**
…	…	…

Activity 2035. Define Domain Model



- Description
  - Define domain concept model by reviewing input artifacts
  - Input : essential use case descriptions, business concept model
  - Output : **A conceptual class diagram**
    - Not class diagram - No operation
    - Standard applied : UML's class diagram

Activity 2035. Define Domain Model

- What is **domain model**?
  - Conceptual models
    - A representation of conceptual classes identified from a real world
    - Illustrates meaningful conceptual classes in a problem domain
  - Widely used as a source of inspiration for designing software objects

- Step
  1. List concepts(domain class) from use cases or business concept model
     - Guideline 1
       - Identify concepts by making a list of candidate concepts from the ‘**Concept Category List**’
     - Guideline 2
       - Identity the **noun and noun phrases** in expanded use cases description and consider them as candidate concepts or attributes

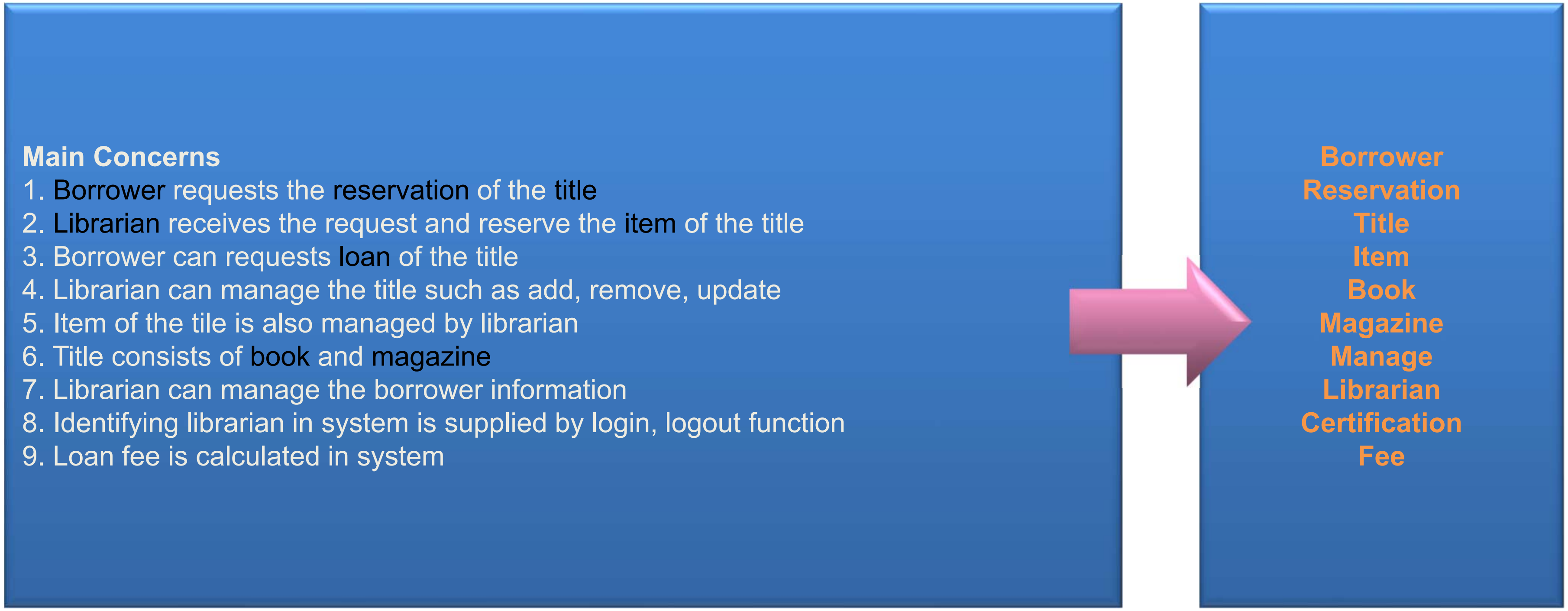
Activity 2035. Define Domain Model

- By using guideline 1
  - ‘Concept Category List’ may contain many common categories that are usually worth to consider.

Concept Category	Examples
Physical or tangible objects	POST
Specifications, designs, or descriptions of things	Product Specification
Places	Store
Transactions	Sale Payment
Transaction line items	Sales Line Item
Roles of people	Cashier
Containers of other things	Store
Things in a container	Item
Other computer or electro-mechanical systems external to our system	Credit Card Authorization System
…	…

Activity 2035. Define Domain Model

- By using guideline 2
  - The fully dressed use cases are an excellent description.
  - Scenario of the use case or use case descriptions can be used.



Main Concerns
1. Borrower requests the reservation of the title
2. Librarian receives the request and reserve the item of the title
3. Borrower can requests loan of the title
4. Librarian can manage the title such as add, remove, update
5. Item of the tile is also managed by librarian
6. Title consists of book and magazine
7. Librarian can manage the borrower information
8. Identifying librarian in system is supplied by login, logout function
9. Loan fee is calculated in system

Borrower
Reservation
Title
Item
Book
Magazine
Manage
Librarian
Certification
Fee

Activity 2035. Define Domain Model

2. Assign class names into concepts
   - Use the existing names in the domain
   - Do not add things that are not there

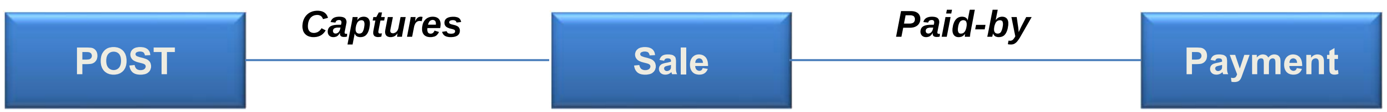
3. Identify associations according to association categories

Association Category	Associations
A is known/logged/recorded/reported/captured in B	Item – Loan Item – Title Loan – Borrower Title – Reservation
A is a line item of B	Item – Title
A is recorded in B	Item – Title
A is related to a transaction of B	Borrower – Loan Borrower – Reservation
A is an organization submit of B	Book – Title Magazine – Title

Activity 2035. Define Domain Model

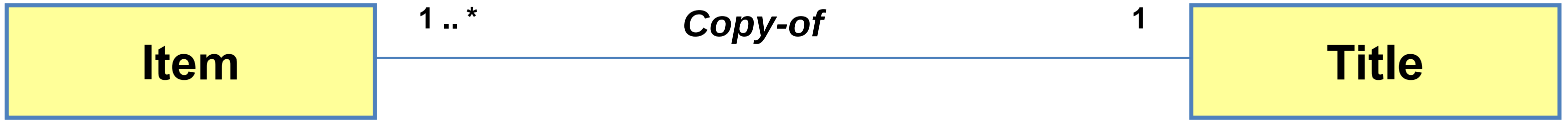
4. Assign <u>priorities</u> into identified associations
   - High priority association categories are
     - A is a physical or logical part of B.
     - A is physically or logically contained in/on B.
     - A is recorded in B.
   - Should avoid showing redundant or derivable associations

5. Assign <u>names</u> into associations
   - “*Type Name” – “Verb Phrase” – “Type Name*”
   - Association names should start with a capital letter.

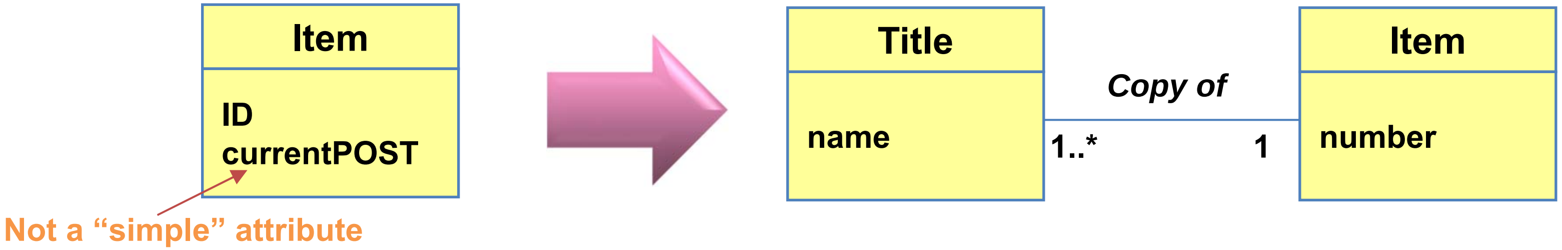


Activity 2035. Define Domain Model

6. Add multiplicity into the ends of an association

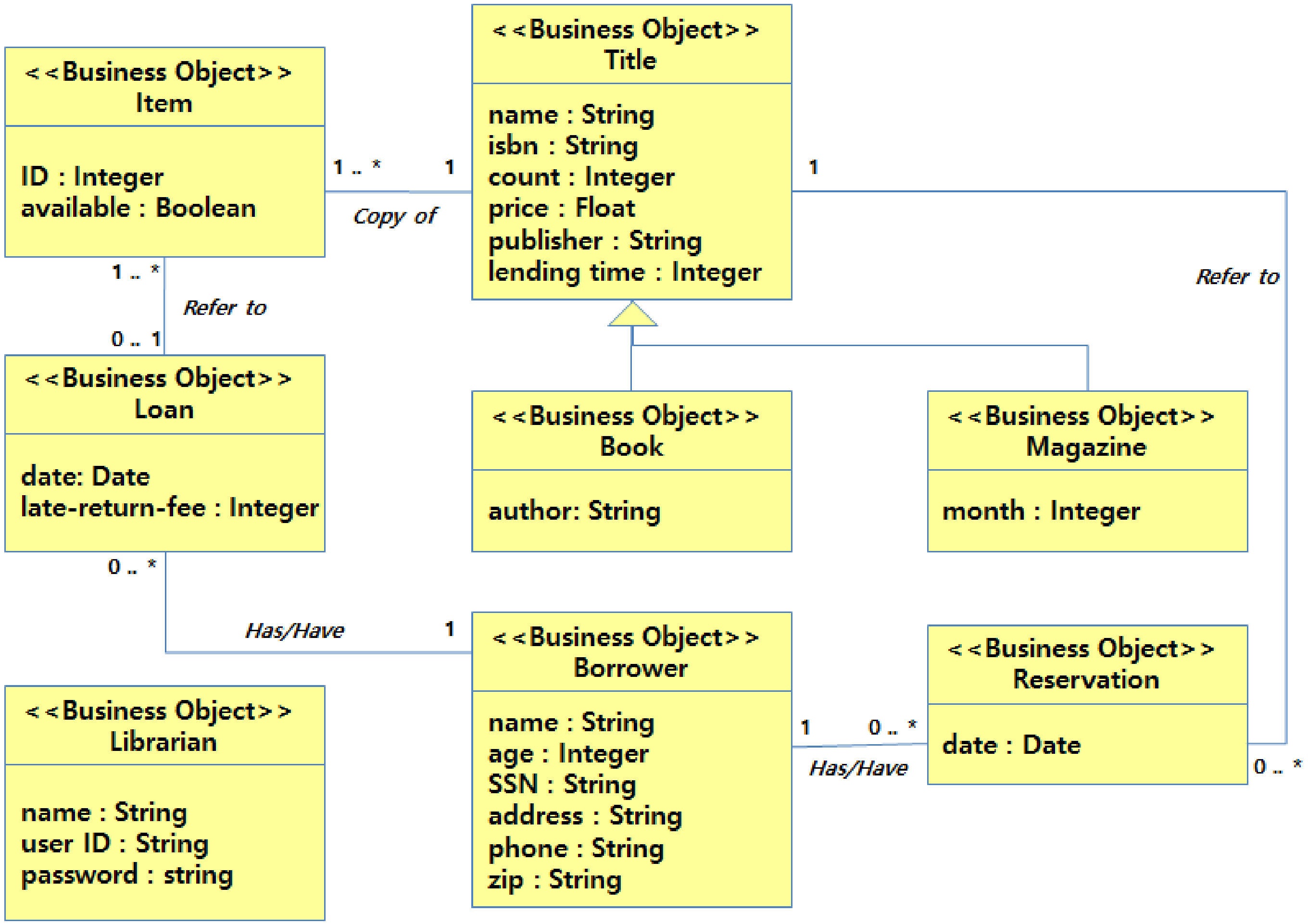


7. Identify attributes by reading
    - requirement specifications, current use cases under consideration, simplification, clarification, and assumption documents
    - Attributes should be simple attributes or pure data values
        - Boolean, Date, Number, String, Time
        - Address, Color, Geometrics(Point, Rectangle,…), Phone Number, Social Security Number, Universal Product Code(UPC), ZIP or postal codes, Enumerated types.

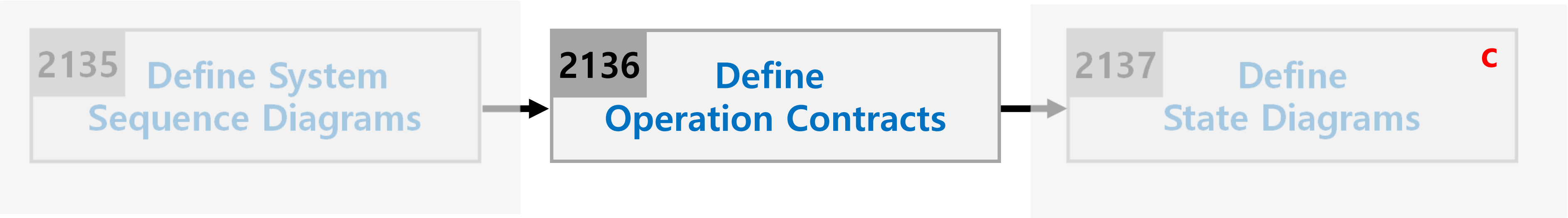


Activity 2035. Define Domain Model

8. Draw them in a conceptual class diagram
   - No operation defined
   - Show basic relationships between business objects



Activity 2036. Define Operation Contracts



- Description
  - Define contracts for system operations
  - Input : essential use case, system sequence diagram, conceptual class diagram
  - Output : **Operation Contracts**

- What is a **contract**?
  - A document that describes what an operation commits to achieve
  - Written for each system operation to describe its behavior
  - System Operation Contract:
    - Describes changes in states of overall system when a system operation is invoked.

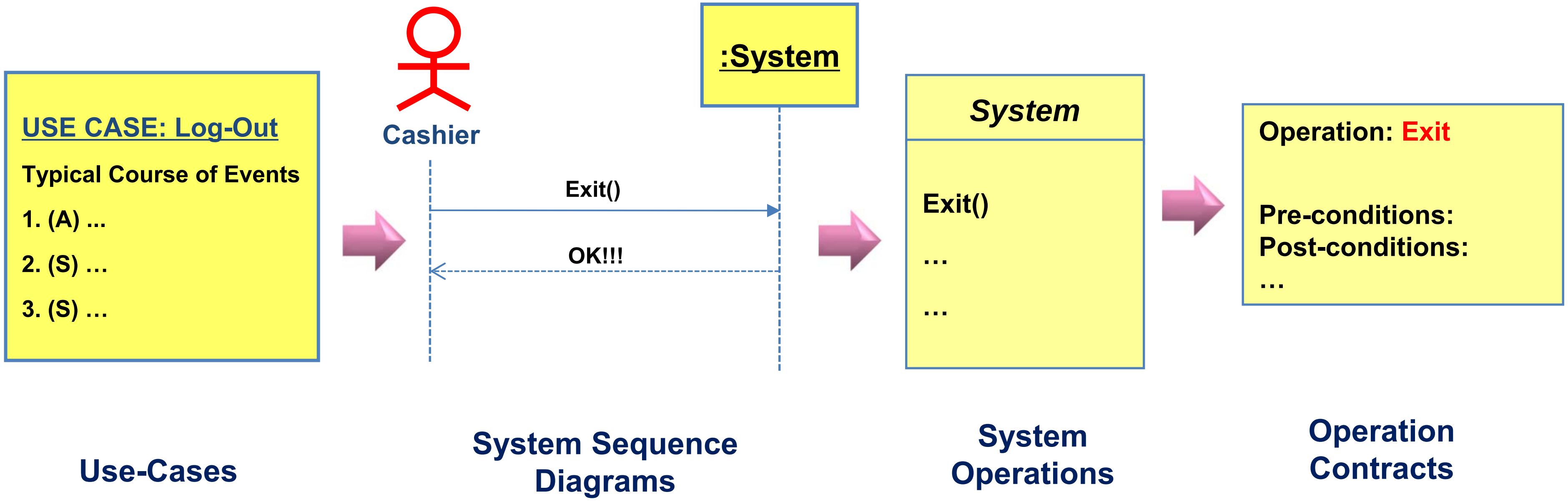
Activity 2036. Define Operation Contracts

- Operation Contracts Format

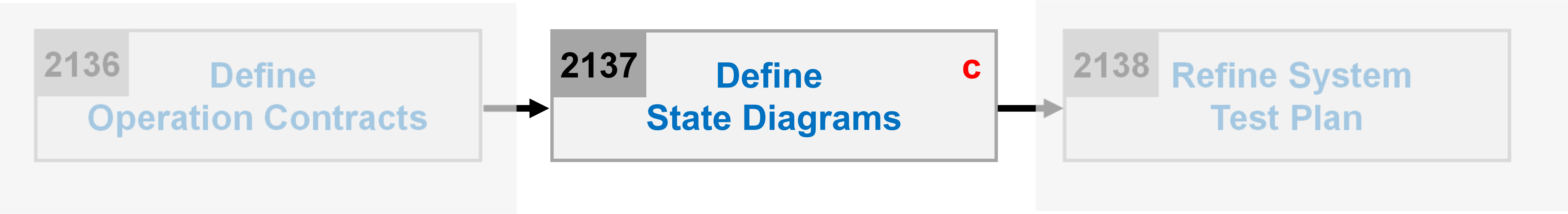
Name	Name of operation, and parameters
Responsibilities	An informal description of the responsibilities that the operation must fill
Type	Name of type(concept, software class, interface)
Cross References	System function reference numbers, use cases, etc.
Notes	Design notes, algorithms, and so on.
Exceptions	Exceptional cases
Output	Non-UI outputs, such as messages or records that are sent outside of the system
Pre-conditions	Assumptions that the state of the system before execution of the operation
Post-conditions	The state of the system after completion of the operation
…	

Activity 2036. Define Operation Contracts

- Operation contracts with other artifacts



Activity 2037. Define State Diagrams



- Description
  - Describes all possible states of the system, use cases, or objects
  - Input : operation contracts, all information available
  - Output : **A state (Statechart) diagrams**

- Three kinds (levels) of State diagrams:
  1. Use case state diagram
  2. System state diagram
  3. Class state diagram

Activity 2037. Define State Diagrams

- Event
  - A significant or noteworthy occurrence
  - Ex) a telephone receiver is taken off the hook

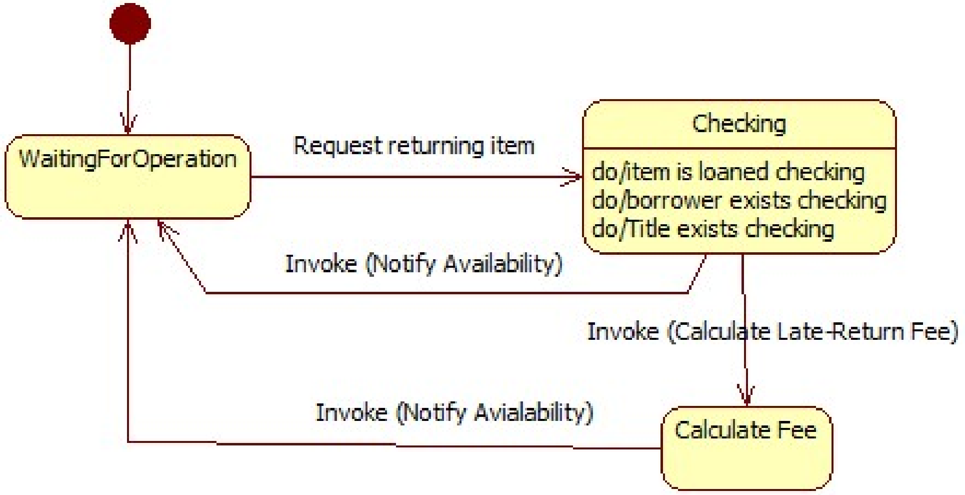
- State
  - Condition of an object at a moment in time
  - Ex) a telephone is in the state of being “idle” after the receiver is placed on the hook and until it is taken off the hook

- Transition
  - A relationship between two states that indicates that when an event occurs and the object moves from one state to another
  - Ex) when the event “off hook” occurs, transition occurs from the “idle” to “active” state

Activity 2037. Define State Diagrams

- State Diagram for Use Case
  - A state diagram that depicts the overall system events and their sequence within a use case

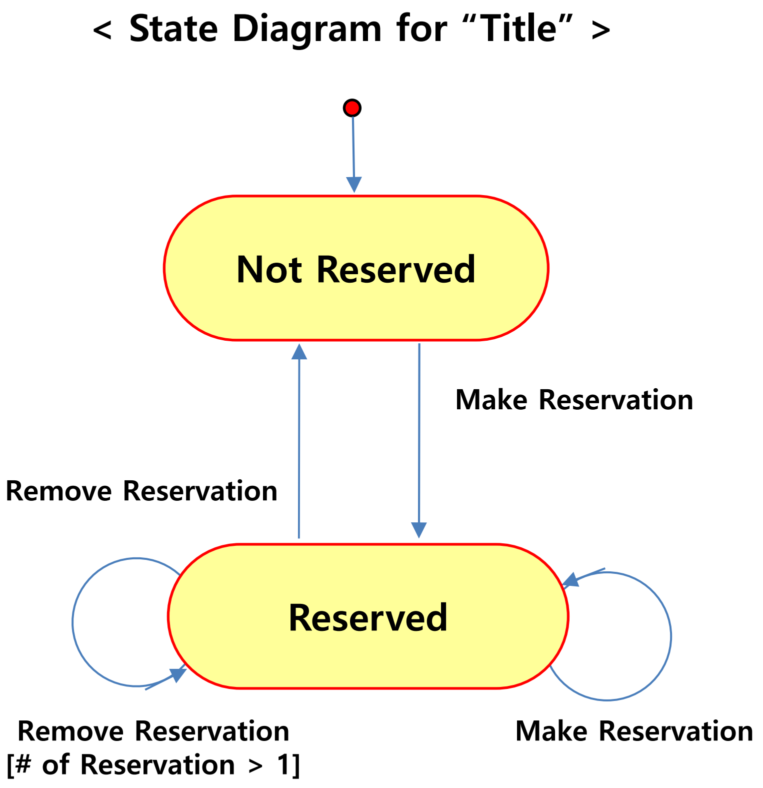
Use Case: Return Item

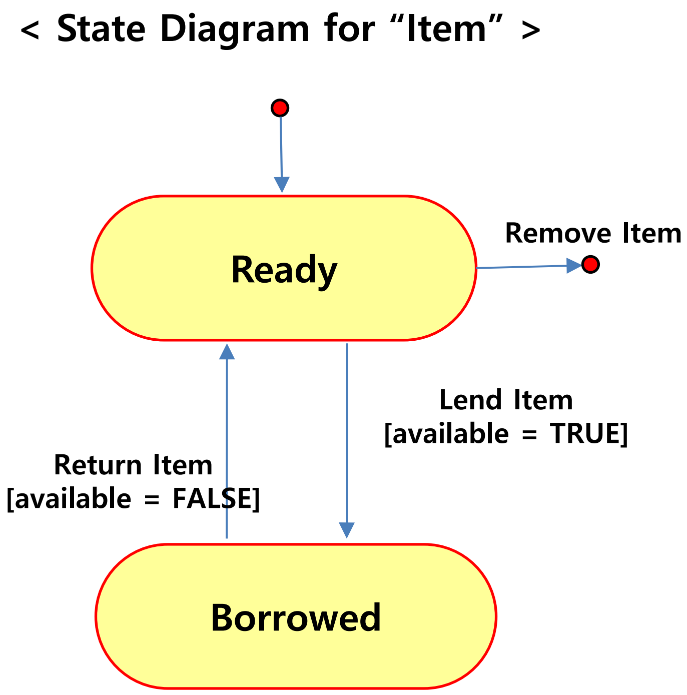


WaitingForOperation
Request returning item
Checking
do/item is loaned checking
do/borrower exists checking
do/Title exists checking
Invoke (Notify Availability)
Invoke (Calculate Late-Return Fee)
Invoke (Notify Avialability)
Calculate Fee

Activity 2037. Define State Diagrams

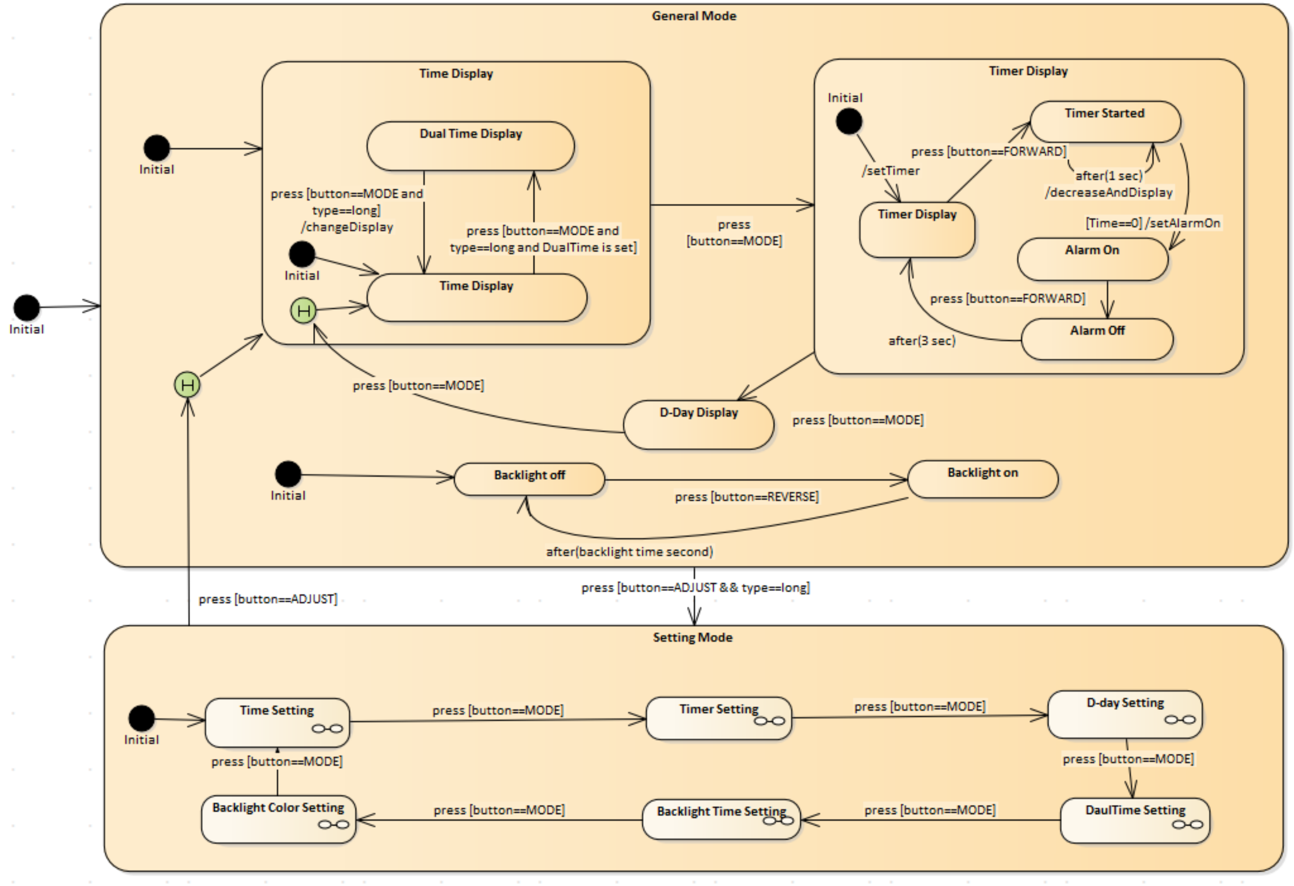
- State Diagram for Class
  - A state diagram that depicts state changes of a class across all the use cases





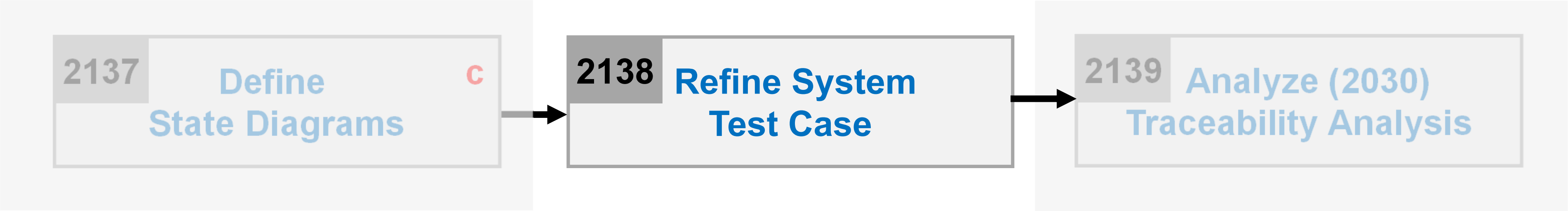
Activity 2037. Define State Diagrams

- State Diagram for Systems
  - Identify system events from system sequence diagram
  - Determine sequence of system events
  - Assign system events into transition of state diagram



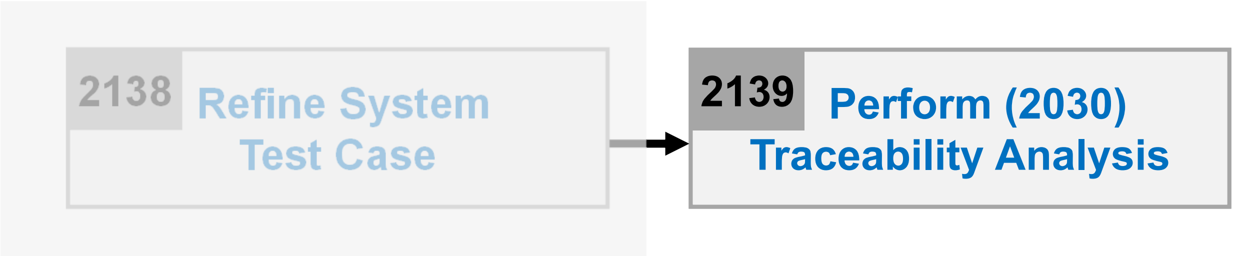
DEPENDABLE SOFTWARE LABORATORY

Activity 2038. Refine System Test Case

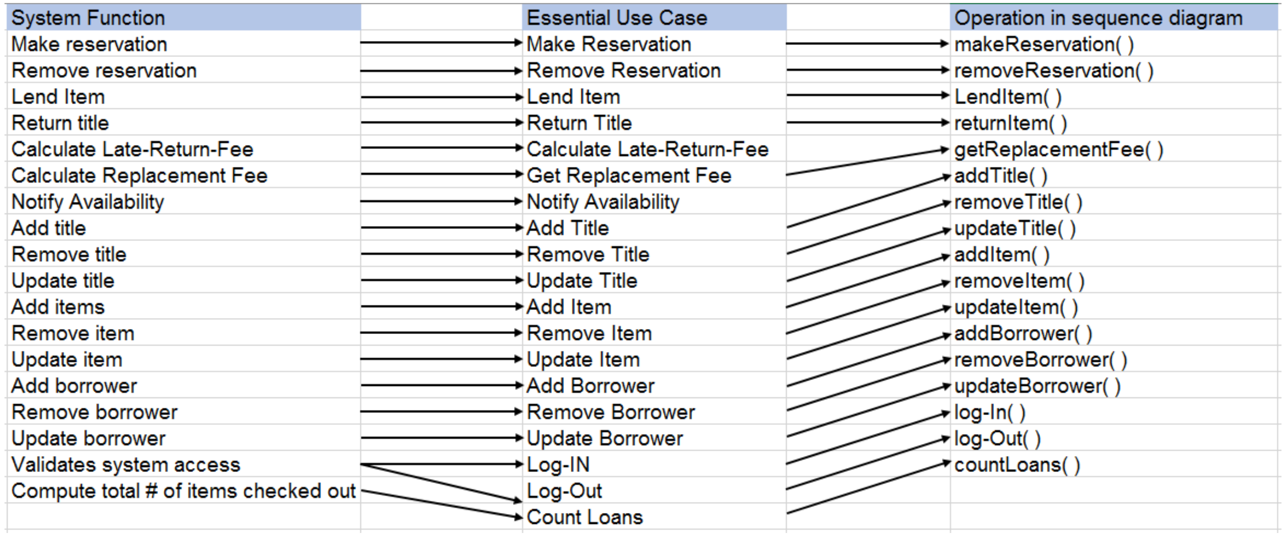


- Description
  - Refine the system test plan by using additional information
  - Input : essential use case, system test plan, system sequence diagram, operation contracts
  - Output : **A refined system test plan**

Activity 2039. Perform 2030 Traceability Analysis



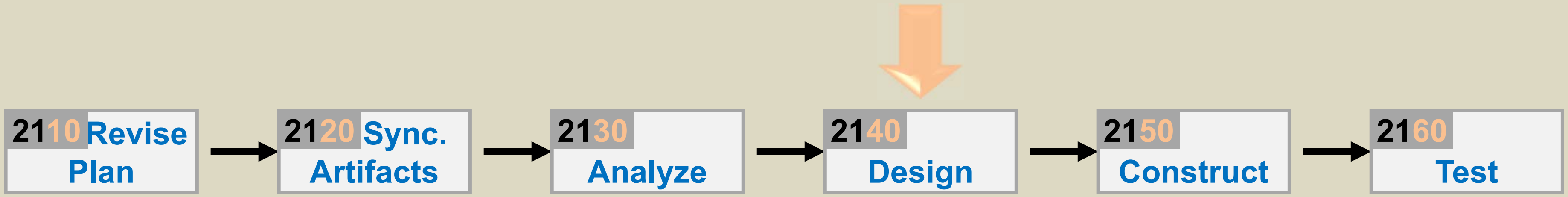
- Description
  - Link all elements from the abstract (requirements and use cases) to details (system operations and system test cases)
  - Input : Requirements specification, essential use case, system sequence diagram, operation contracts, system test cases
  - Output : **A 2030 traceability graph**



System Function	Essential Use Case	Operation in sequence diagram
Make reservation	Make Reservation	makeReservation( )
Remove reservation	Remove Reservation	removeReservation( )
Lend Item	Lend Item	LendItem( )
Return title	Return Title	returnItem( )
Calculate Late-Return-Fee	Calculate Late-Return-Fee	getReplacementFee( )
Calculate Replacement Fee	Get Replacement Fee	addTitle( )
Notify Availability	Notify Availability	removeTitle( )
Add title	Add Title	updateTitle( )
Remove title	Remove Title	addItem( )
Update title	Update Title	removeItem( )
Add items	Add Item	updateItem( )
Remove item	Remove Item	addBorrower( )
Update item	Update Item	removeBorrower( )
Add borrower	Add Borrower	updateBorrower( )
Remove borrower	Remove Borrower	log-In( )
Update borrower	Update Borrower	log-Out( )
Validates system access	Log-IN	countLoans( )
Compute total # of items checked out	Log-Out	
	Count Loans	

Back

Phase 2040. Design

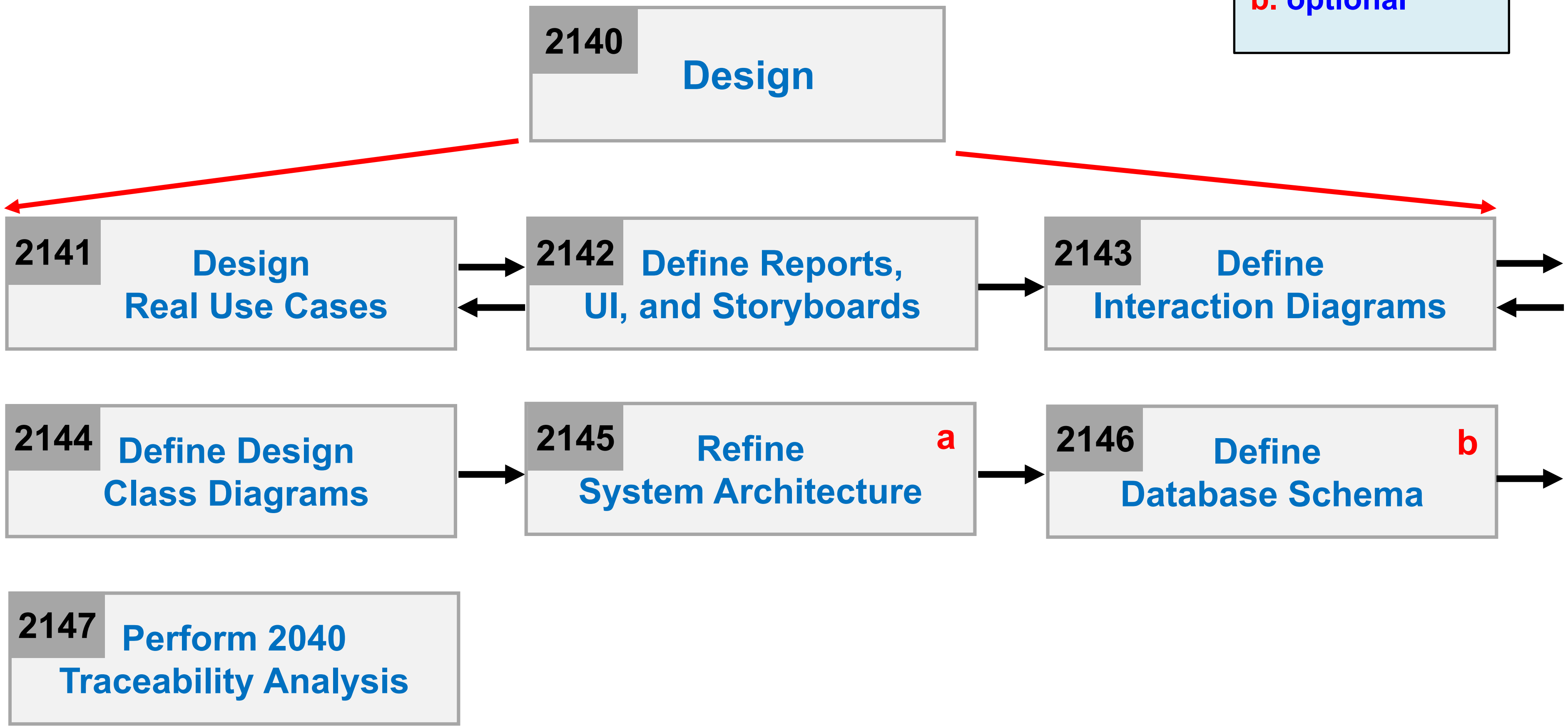


2110 Revise Plan → 2120 Sync. Artifacts → 2130 Analyze → 2140 Design → 2150 Construct → 2160 Test

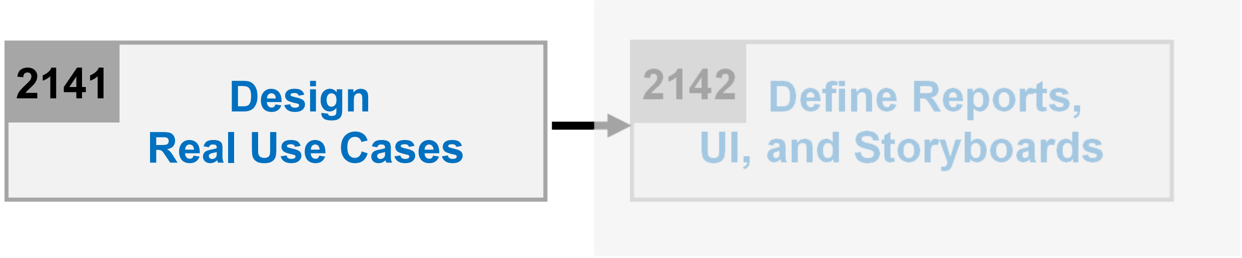
Phase 2040. Design

- 7 Activities

a. Varied order
b. optional


2140
Design
2141
Design Real Use Cases
2142
Define Reports, UI, and Storyboards
2143
Define Interaction Diagrams
2144
Define Design Class Diagrams
2145
Refine System Architecture
a
2146
Define Database Schema
b
2147
Perform 2040 Traceability Analysis

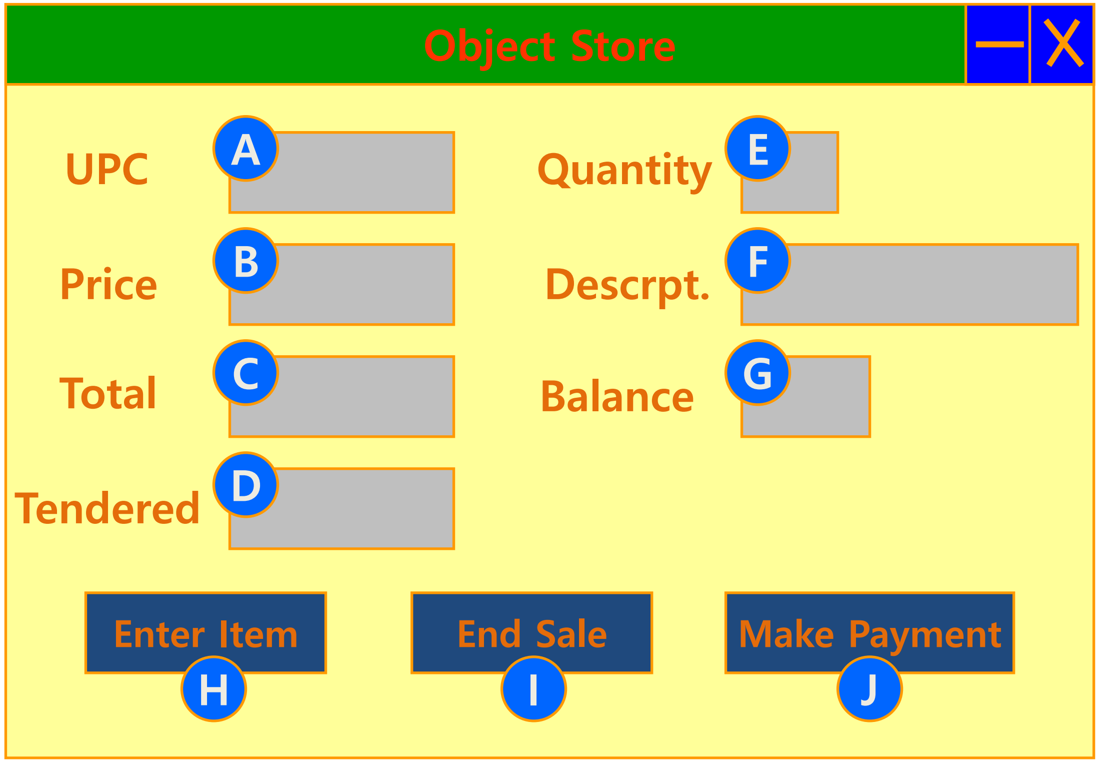
Activity 2041. Design Real Use Cases



- Description
  - It describes real/actual design of the use case in terms of concrete input and output technology and its overall implementation.
  - If a graphical user interface is involved, the real use case will include diagrams of the GUI and discussion of the low-level interactions with interface widgets.
  - Input : Essential Use Case
  - Output : **Real Use Case (the fully dressed format of UP)**

Activity 2041. Design Real Use Cases

- Steps
  1. Select each use case from essential use cases
  2. Add user interface widgets into the expanded format, and concrete implementation details into the typical courses of events



Window-1

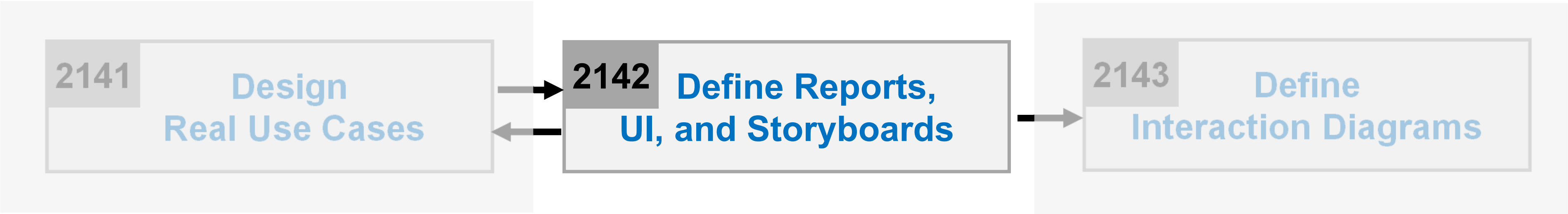
Activity 2041. Design Real Use Cases

Use Case	**Buy Items – Version 1 (Cash only)**
Actor	Customer, Cashier
Purpose	Capture a sale and its cash payment
Overview	A Customer arrives at a checkout with items to purchase. The Cashier records the items and collects cash payment, which may be authorized. On completion, the Customer leaves with the items.
Type	Primary and Real
Cross Reference	Functions: R1.1, R1.2, R1.3, R1.7, R1.9, R2.1 Use Cases: Log In use case
Pre-Requisites	N/A
UI Widgets	Window-1
Typical Courses of Events	(A) : Actor, (S) : System 1. (A) This use case begins when a customer arrives at the POST to checkout with items to purchase. 2. (A) For each item, the Cashier types an UPC in A of Window-1. If there is more than one of an item, the quantity may optionally be entered in E. They press B after each item entry. (E1) 3. (S) Adds the item information to the running sales transaction. The description and price of the current item are displayed in B and F of Window1. 4. (A) The Cashier tells the customer the total.
Alternative Courses of Events	…
Exceptional Courses of Events	E1: If an invalid UPC is entered, indicate an error.

Activity 2041. Design Real Use Cases

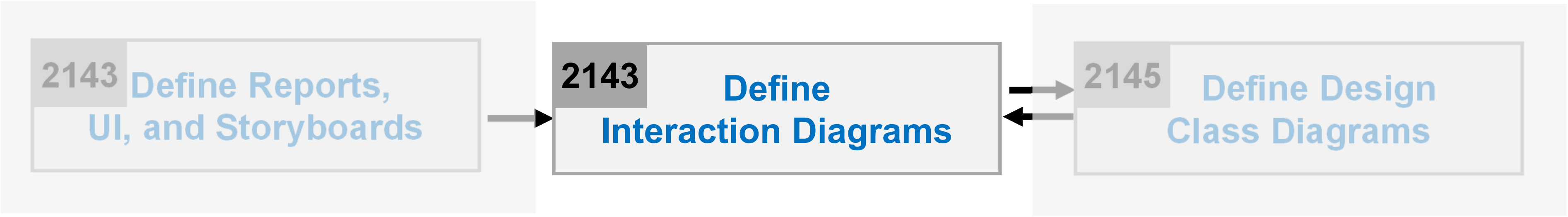
Use Case	**1. Make Reservation**
Actor	Librarian
Purpose	Create a new reservation
Overview	(As in the business use case)
Type	Primary and Real
Cross Reference	System Functions: R1.1, R3.1 Use Case: “Add Borrower”
Pre-Requisites	A borrower should be registered.
Typical Courses of Events	(A) : Actor, (S) : System 1. (A) A librarian inputs an *isbn* and *ssn* of the title 2. (S) Find a corresponding *title* 3. (S) Find a corresponding *borrower* 4. (S) Create a new *reservation* 5. (S) Store the new *reservation* 6. (S) Increase *reservationCount* in the borrower 7. (S) Increase *reservationCount* in the title
Alternative Courses of Events	N/A
Exceptional Courses of Events	Line 2: If the *title* does not exist, display an error message. Line 3: If the *borrower* does not exist, display an error message.

Activity 2042. Define Reports, UI and Storyboards



- Description
  - Design UI storyboard and UI components
  - Input : Requirements Specification, Real Use Case Descriptions
  - Output : UI Storyboard, UI Component Design Specification

Activity 2043. Define Interaction Diagrams



- Description
  - Communication diagrams illustrate object interactions in a graph or network format
  - To illustrate how objects interactions via messages to fulfill tasks

  - Input : Real Use Case Descriptions
  - Output : **An interaction diagram**
    - Standards Applied
      - UML's **Sequence Diagram** , **Communication Diagram** , **Timing Diagram** and **Interaction Overview Diagram**

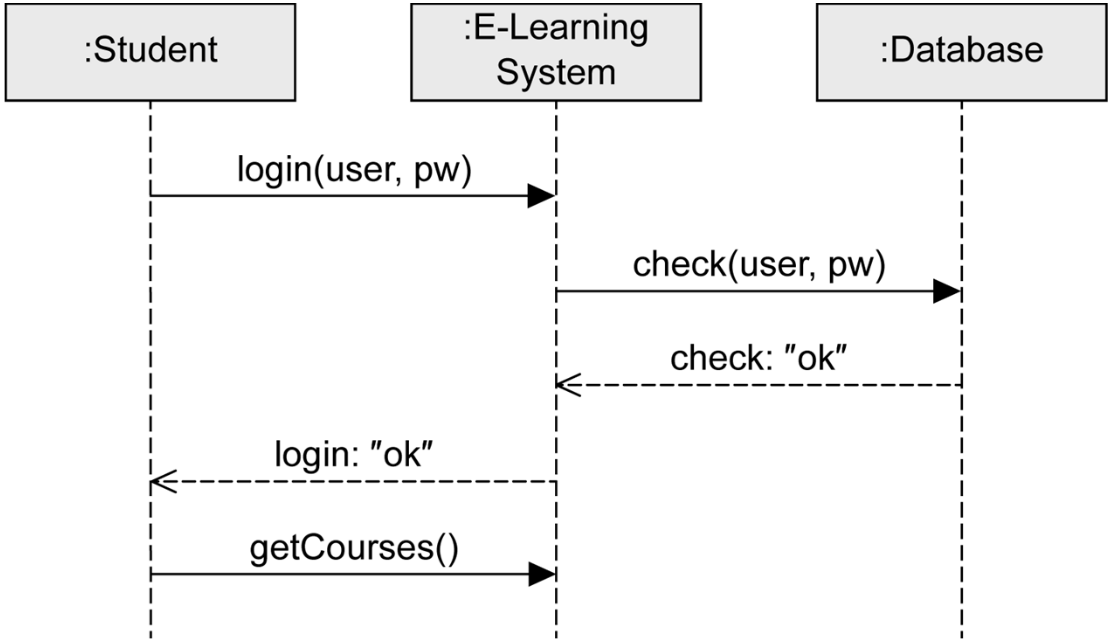
Activity 2043. Define Interaction Diagrams

- Sequence Diagram vs. Communication Diagram
  - Based on the same concepts
  - Generally equivalent for simple interactions, but different focus

Type	Strengths	Weaknesses
Sequence Diagram	Clearly shows sequence or time ordering of messages	Forced to extend to the right, when adding new objects with consuming horizontal space
Communication Diagram	Space economical and flexible to add new objects in two dimensions Better to illustrate complex branching, iteration, and concurrent behavior	Difficult to see sequence of messages

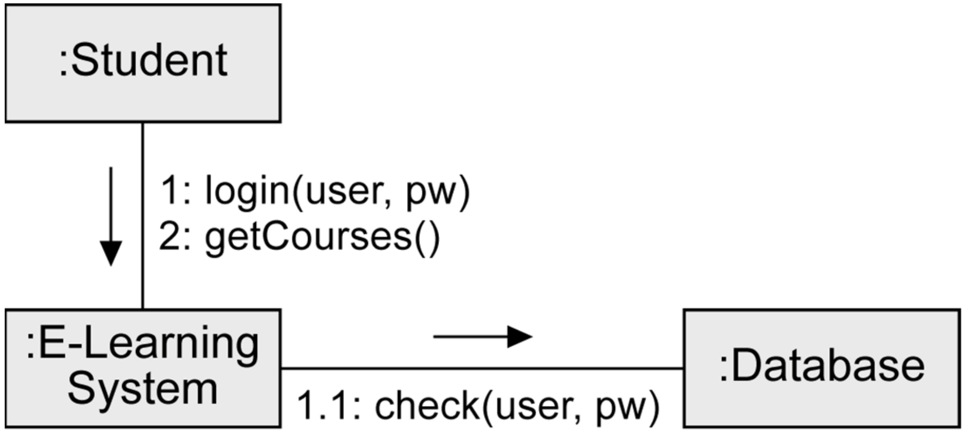
Activity 2043. Define Interaction Diagrams

- **Sequence diagram**
  - Vertical axis: chronological order
  - Horizontal axis: interaction partners



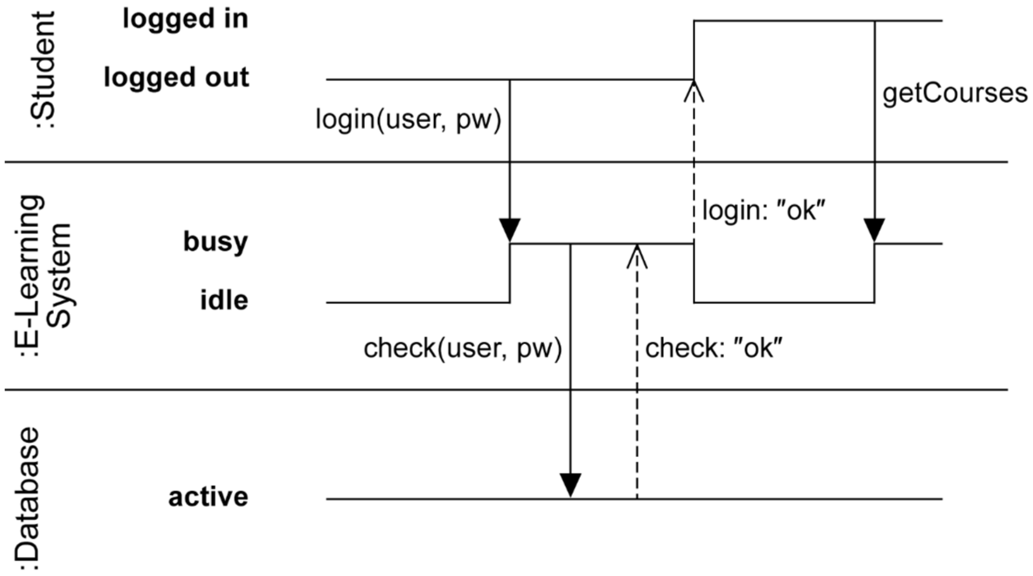
Activity 2043. Define Interaction Diagrams

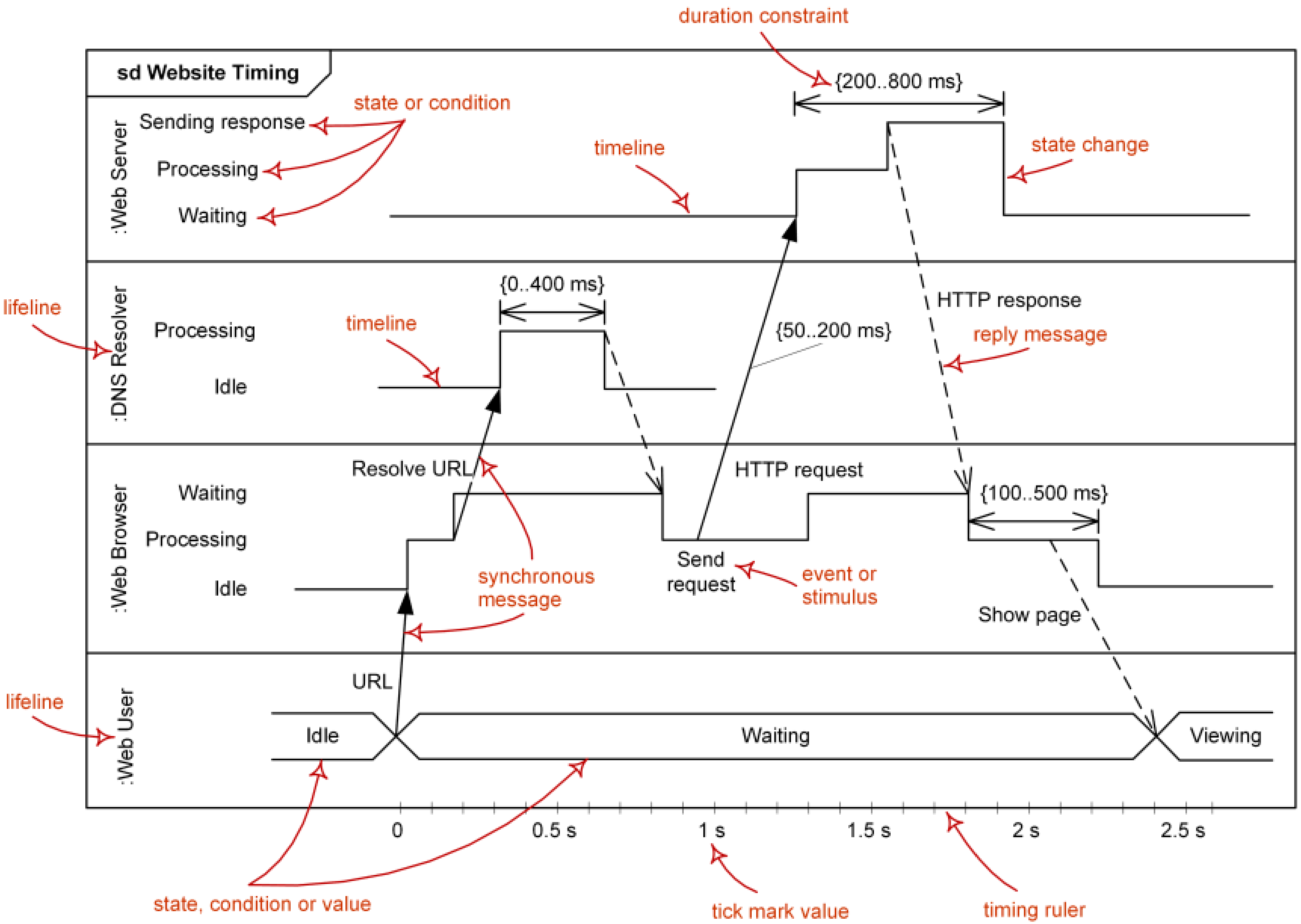
- **Communication diagram**
  - Models the relationships between communication partners
  - Focus: Who communicates with whom
  - Time is not a separate dimension
  - Message order via decimal classification



Activity 2043. Define Interaction Diagrams

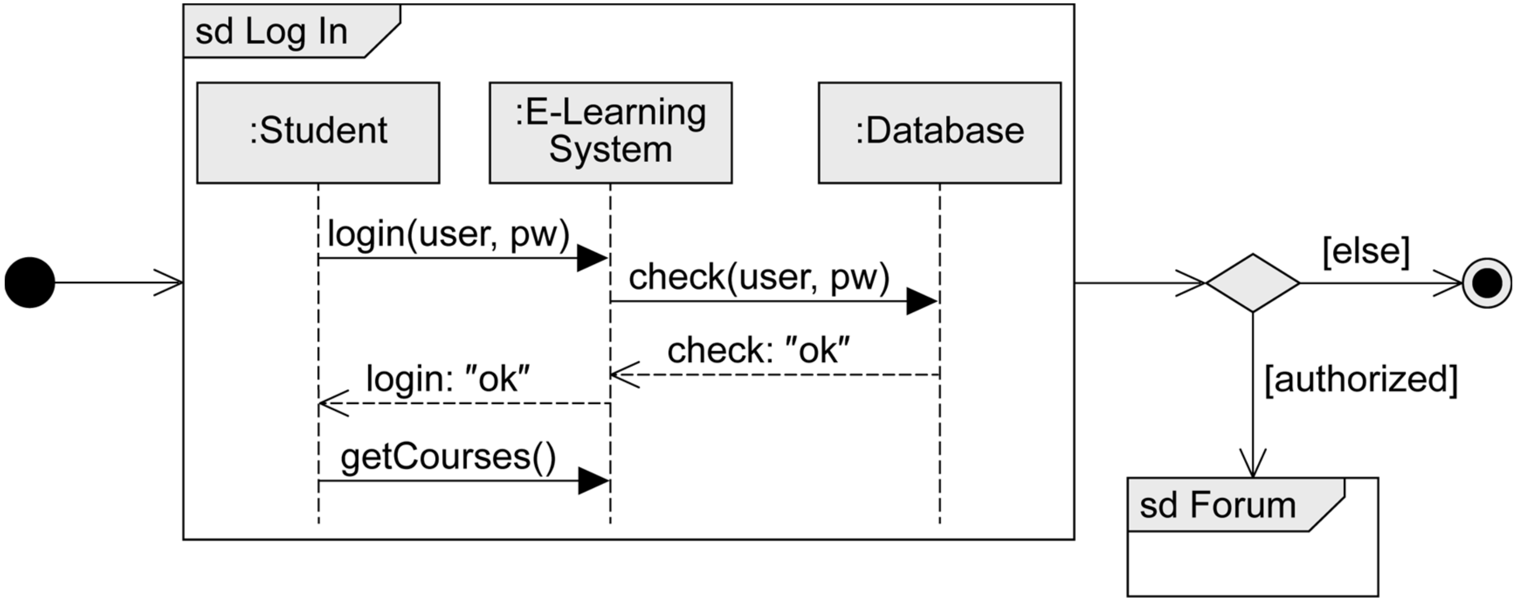
- **Timing diagram**
  - Shows state changes of the interaction partners that result from the occurrence of events
  - Vertical axis: interaction partners
  - Horizontal axis: chronological order



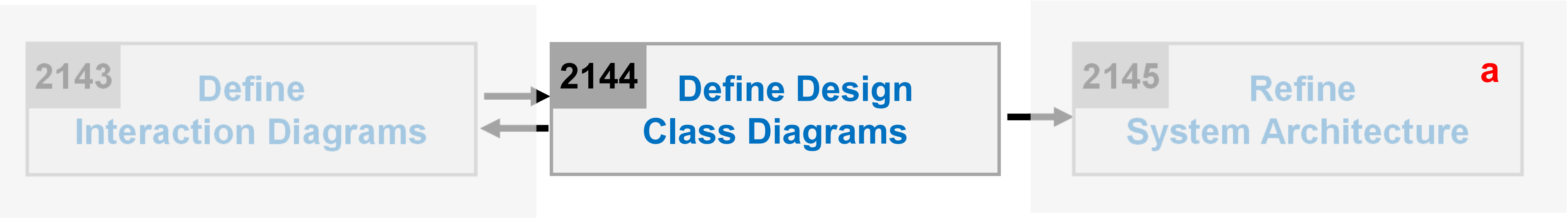

duration constraint
sd Website Timing
{200..800 ms}
state or condition
:Web Server
Sending response
Processing
Waiting
timeline
state change
lifeline
:DNS Resolver
{0..400 ms}
Processing
Idle
timeline
HTTP response
{50..200 ms}
reply message
:Web Browser
Resolve URL
HTTP request
Waiting
Processing
Idle
{100..500 ms}
synchronous message
Send request
event or stimulus
Show page
lifeline
:Web User
URL
Idle
Waiting
Viewing
0
0.5 s
1 s
1.5 s
2 s
2.5 s
state, condition or value
tick mark value
timing ruler

Activity 2043. Define Interaction Diagrams

- **Interaction overview diagram**
  - Visualizes order of different interactions
  - Allows to place various interaction diagrams in a logical order
  - Basic notation concepts of activity diagram



Activity 2044. Define Design Class Diagrams


2143
Define Interaction Diagrams
2144
Define Design Class Diagrams
2145
Refine System Architecture
a

- Description
  - Describes more details in conceptual class diagram
  - Add navigability, dependency, data type, operation signature, parameters, return types, and so on

  - Input : Interaction Diagram, Conceptual Class Diagram
  - Output : **A Design Class Diagram**
    - Standards Applied: UML’s Class Diagram

Activity 2044. Define Design Class Diagrams

- Steps
  1. Identity all classes
  2. Draw them in a class diagram
  3. Add attributes
  4. Add method names
  5. Add type information to the attributes and methods
  6. Add the associations
  7. Add navigability arrows
  8. Add dependency

Activity 2044. Define Design Class Diagrams

- Step 1. Identify all classes
  - by scanning all interaction diagrams
  - listing classes mentioned

Loan

Librarian

Borrower

Reservation

Title

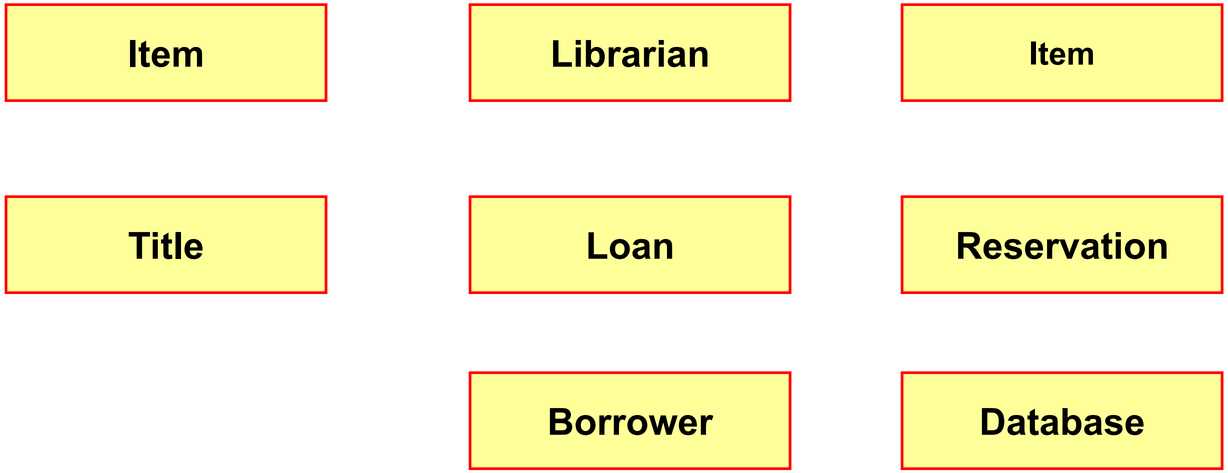
Database

Book

Magazine

Activity 2044. Define Design Class Diagrams

- Step 2. Draw a class diagram
  - includes classes found in Step 1



Item

Librarian

Item

Title

Loan

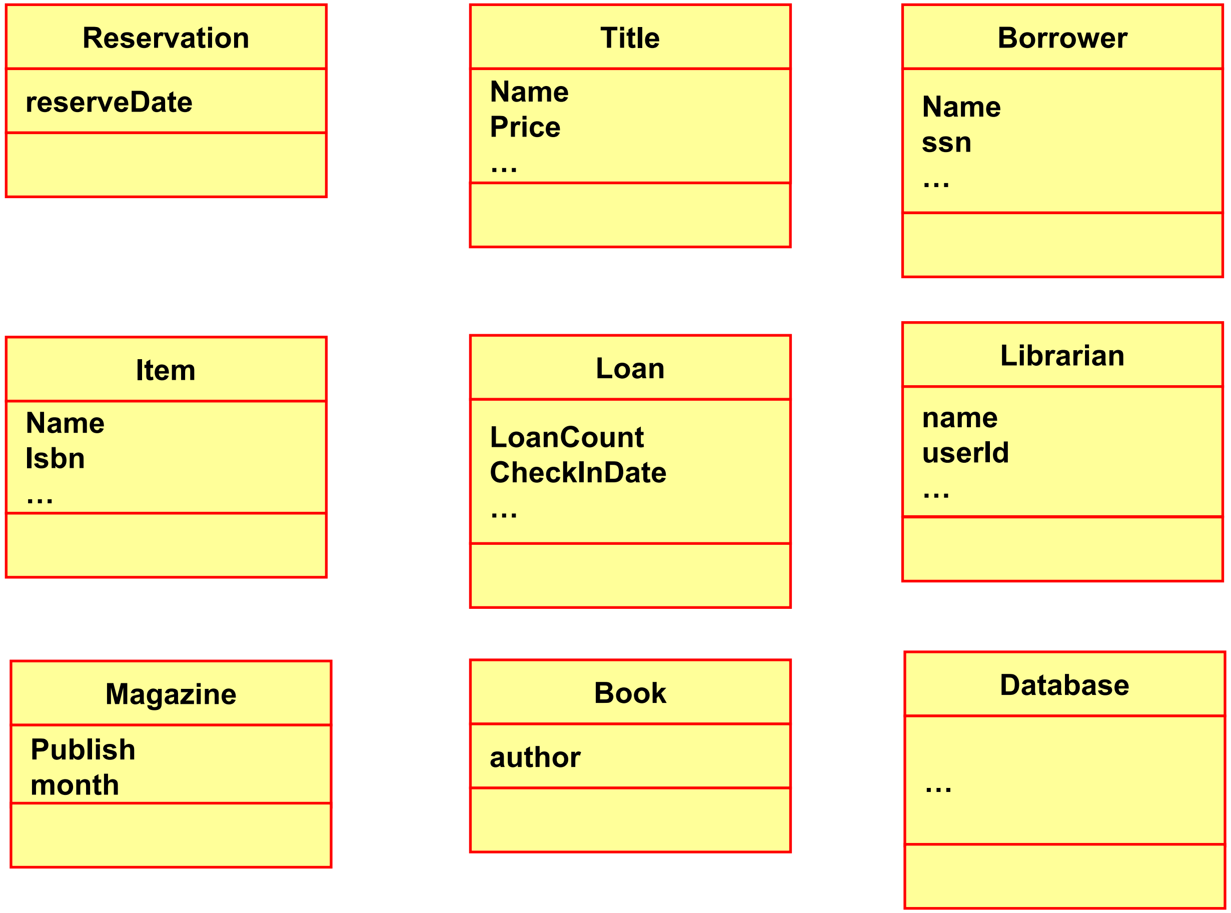
Reservation

Borrower

Database

Activity 2044. Define Design Class Diagrams

- Step 3. Add attributes
  - Include the attributes previously identified in the conceptual class diagram that are also used in the design



Reservation
reserveDate

Title
Name
Price
…

Borrower
Name
ssn
…

Item
Name
Isbn
…

Loan
LoanCount
CheckInDate
…

Librarian
name
userId
…

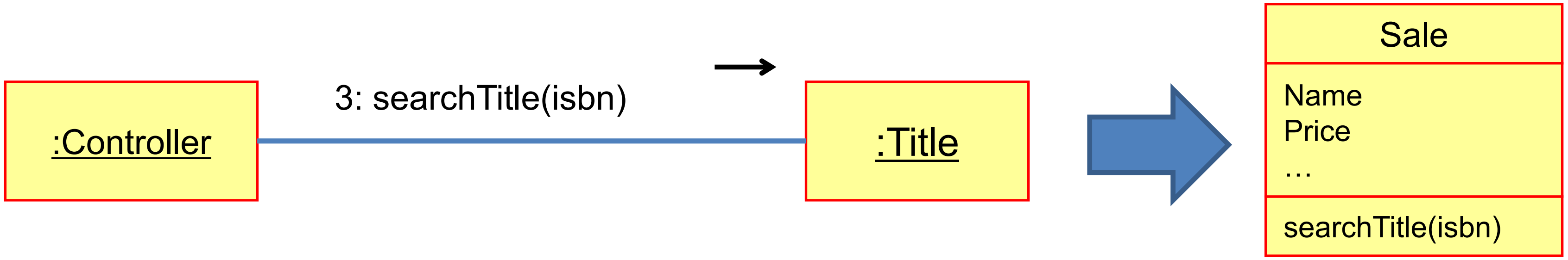
Magazine
Publish
month

Book
author

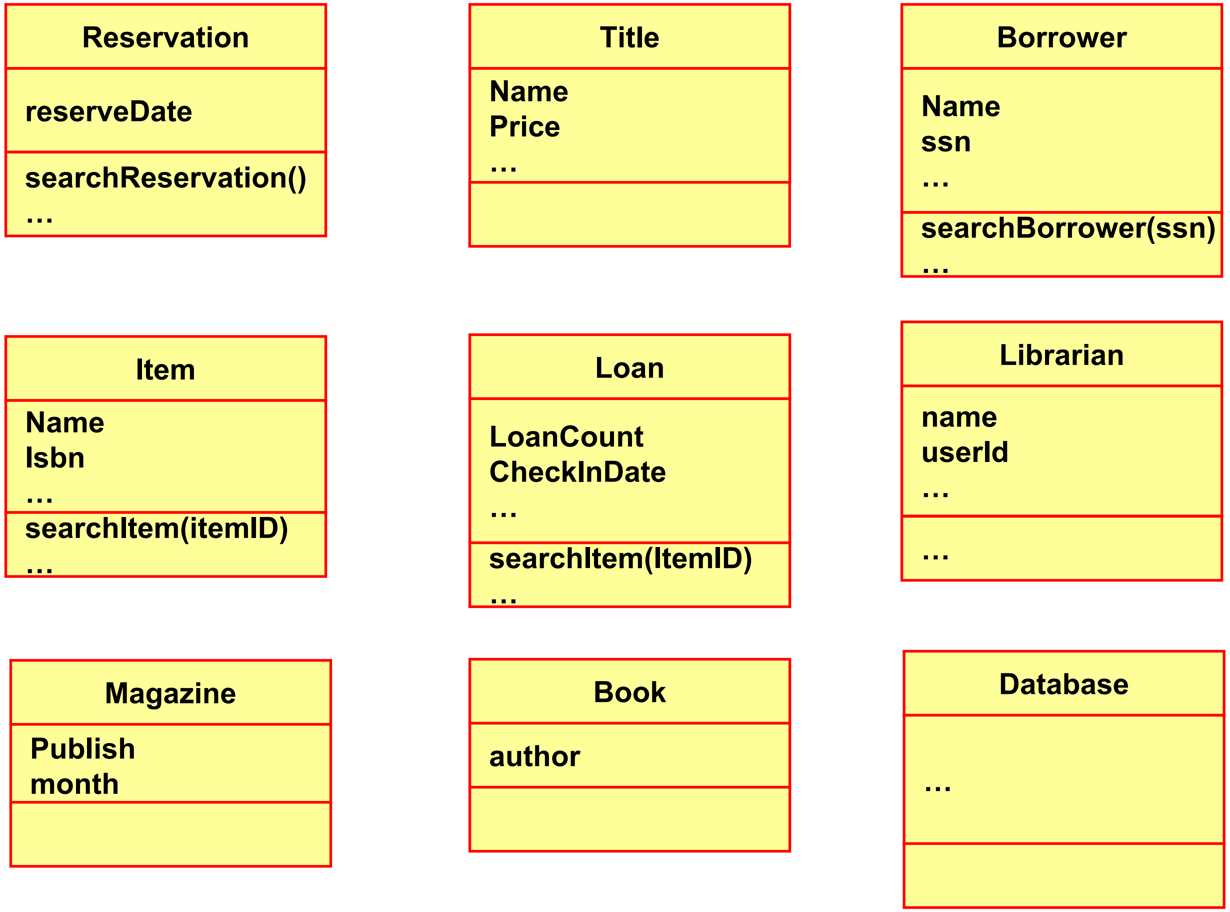
Database
…

Activity 2044. Define Design Class Diagrams

- Step 4. Add method names
  - Identify method of each class by scanning the interaction diagrams
  - The messages sent to a class in interaction diagrams must be defined in the class
  - Don't add
    - creation methods and constructors
    - accessing methods
    - messages to a multi-object



Activity 2044. Define Design Class Diagrams



Reservation

reserveDate

searchReservation()
…

Title

Name
Price
…

Borrower

Name
ssn
…

searchBorrower(ssn)
…

Item

Name
Isbn
…

searchItem(itemID)
…

Loan

LoanCount
CheckInDate
…

searchItem(ItemID)
…

Librarian

name
userId
…

…

Magazine

Publish
month

Book

author

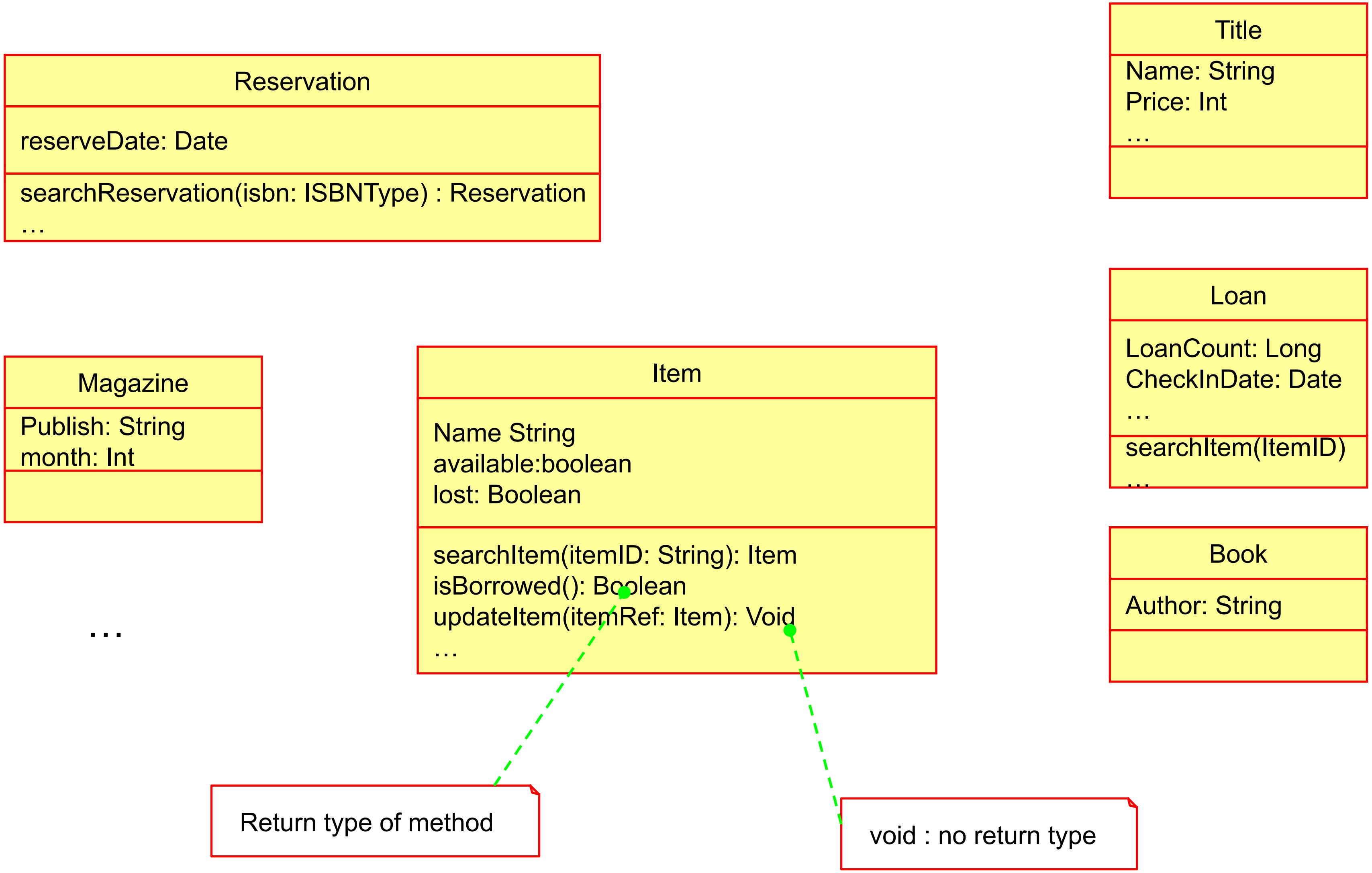
Database

…

Activity 2044. Define Design Class Diagrams

- Step 5. Add type information
  - Show types of attributes, method parameters, and method return values optionally.
  - Determine whether to show type information or not
    - When using a CASE tool with automatic code generation, exhaustive details are necessary
    - If it is being created for software developers to read, exhaustive detail may adversely effect the noise-to-value ratio

Activity 2044. Define Design Class Diagrams



Reservation

reserveDate: Date

searchReservation(isbn: ISBNType) : Reservation
…

Title

Name: String
Price: Int
…

Magazine

Publish: String
month: Int

…

Item

Name String
available:boolean
lost: Boolean

searchItem(itemID: String): Item
isBorrowed(): Boolean
updateItem(itemRef: Item): Void
…

Loan

LoanCount: Long
CheckInDate: Date
…

searchItem(ItemID)
…

Book

Author: String

Return type of method

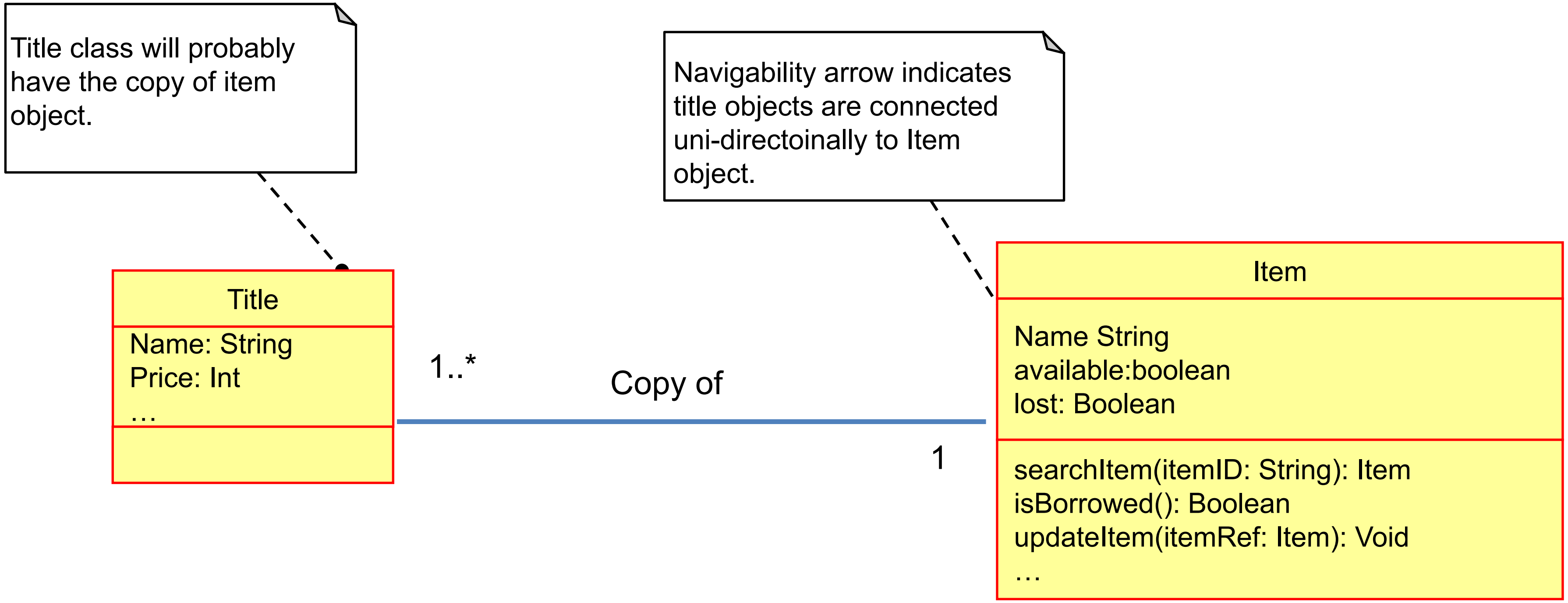
void : no return type

Activity 2044. Define Design Class Diagrams

- Step 6. Add associations
  - Choose associations by software-oriented need-to-know criterion from the interaction diagrams

- Step 7. Add navigability arrows
  - According to the interaction diagram
  - Common situations to define navigability
    - A sends a message to B
    - A creates an instance B
    - A needs to maintain a connection to B

Activity 2044. Define Design Class Diagrams

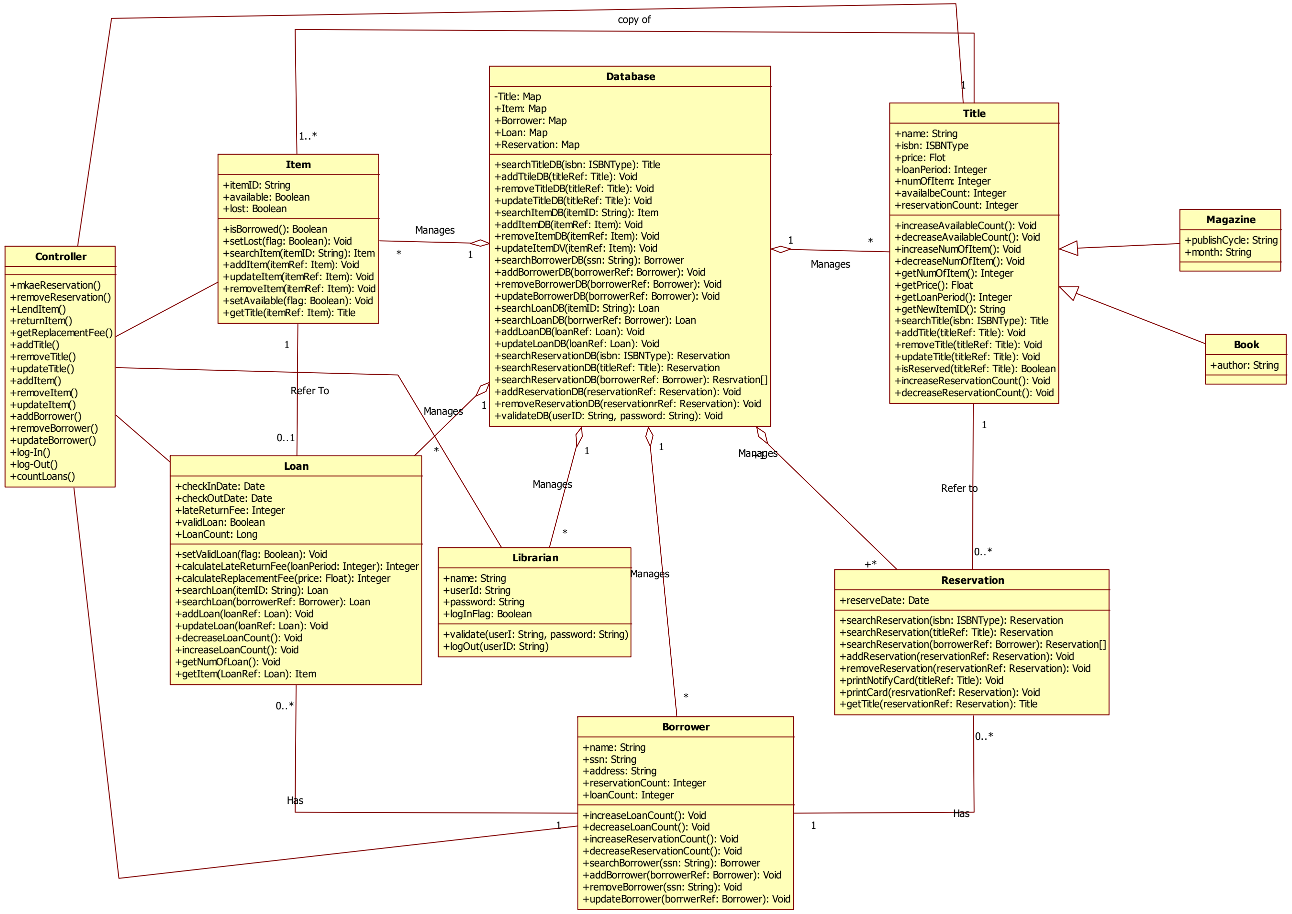


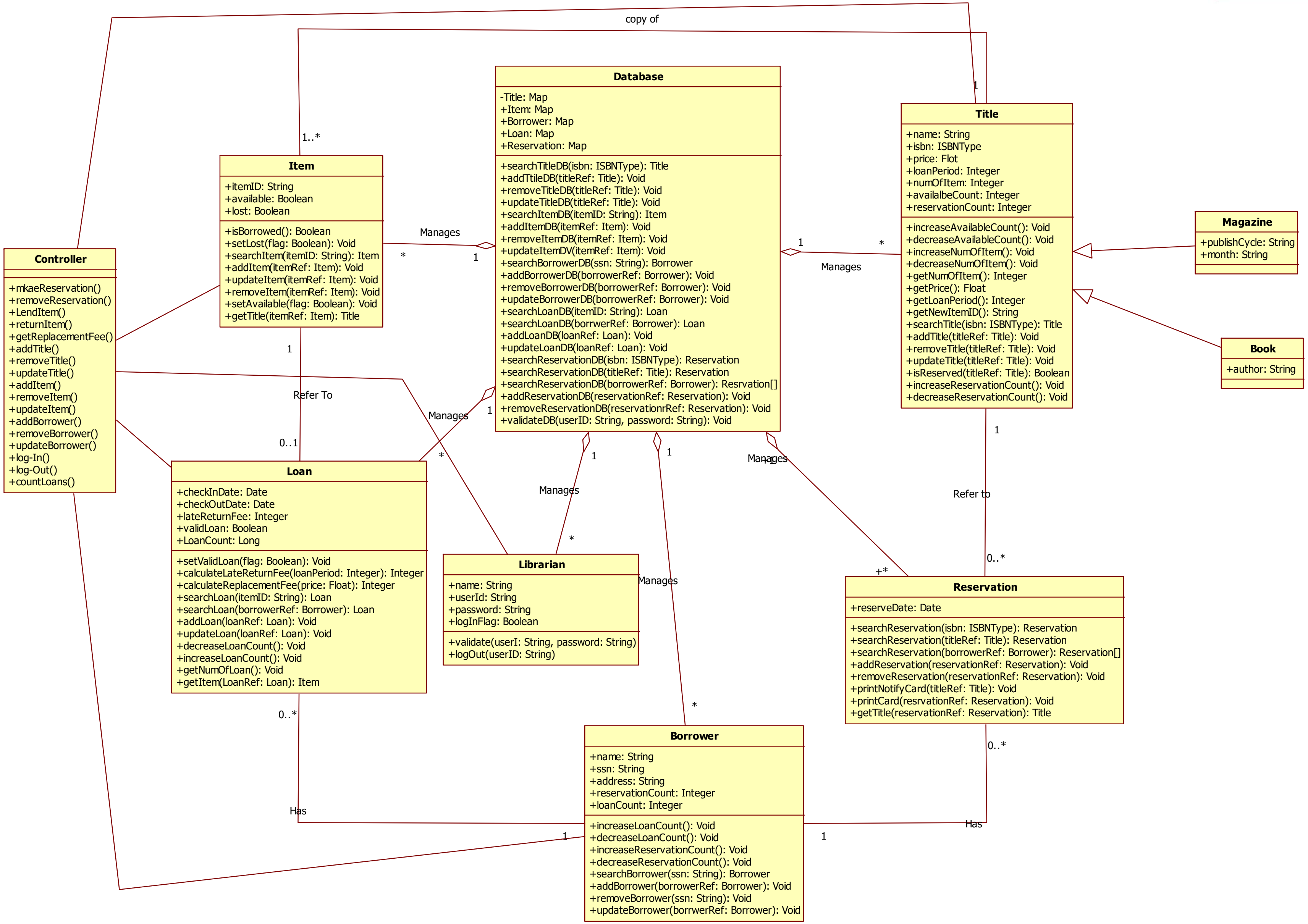
Title class will probably have the copy of item object.

Navigability arrow indicates title objects are connected uni-directoinally to Item object.

Activity 2044. Define Design Class Diagrams

- Step 8. Add dependency relationship
  - when there is non-attribute visibility between classes
  - Non-attribute visibility : parameter, global, or locally declared visibility


copy of
Database
-Title: Map
+Item: Map
+Borrower: Map
+Loan: Map
+Reservation: Map
+searchTitleDB(isbn: ISBNType): Title
+addTtileDB(titleRef: Title): Void
+removeTitleDB(titleRef: Title): Void
+updateTitleDB(titleRef: Title): Void
+searchItemDB(itemID: String): Item
+addItemDB(itemRef: Item): Void
+removeItemDB(itemRef: Item): Void
+updateItemDV(itemRef: Item): Void
+searchBorrowerDB(ssn: String): Borrower
+addBorrowerDB(borrowerRef: Borrower): Void
+removeBorrowerDB(borrowerRef: Borrower): Void
+updateBorrowerDB(borrowerRef: Borrower): Void
+searchLoanDB(itemID: String): Loan
+searchLoanDB(borrwerRef: Borrower): Loan
+addLoanDB(loanRef: Loan): Void
+updateLoanDB(loanRef: Loan): Void
+searchReservationDB(isbn: ISBNType): Reservation
+searchReservationDB(titleRef: Title): Reservation
+searchReservationDB(borrowerRef: Borrower): Resrvation[]
+addReservationDB(reservationRef: Reservation): Void
+removeReservationDB(reservationrRef: Reservation): Void
+validateDB(userID: String, password: String): Void
1
Title
+name: String
+isbn: ISBNType
+price: Flot
+loanPeriod: Integer
+numOfItem: Integer
+availalbeCount: Integer
+reservationCount: Integer
+increaseAvailableCount(): Void
+decreaseAvailableCount(): Void
+increaseNumOfItem(): Void
+decreaseNumOfItem(): Void
+getNumOfItem(): Integer
+getPrice(): Float
+getLoanPeriod(): Integer
+getNewItemID(): String
+searchTitle(isbn: ISBNType): Title
+addTitle(titleRef: Title): Void
+removeTitle(titleRef: Title): Void
+updateTitle(titleRef: Title): Void
+isReserved(titleRef: Title): Boolean
+increaseReservationCount(): Void
+decreaseReservationCount(): Void
Magazine
+publishCycle: String
+month: String
Book
+author: String
1..*
Item
+itemID: String
+available: Boolean
+lost: Boolean
+isBorrowed(): Boolean
+setLost(flag: Boolean): Void
+searchItem(itemID: String): Item
+addItem(itemRef: Item): Void
+updateItem(itemRef: Item): Void
+removeItem(itemRef: Item): Void
+setAvailable(flag: Boolean): Void
+getTitle(itemRef: Item): Title
Manages
*
1
1
*
Manages
Controller
+mkaeReservation()
+removeReservation()
+LendItem()
+returnItem()
+getReplacementFee()
+addTitle()
+removeTitle()
+updateTitle()
+addItem()
+removeItem()
+updateItem()
+addBorrower()
+removeBorrower()
+updateBorrower()
+log-In()
+log-Out()
+countLoans()
1
Refer To
0..1
Manages
1
*
1
1
Manages
Loan
+checkInDate: Date
+checkOutDate: Date
+lateReturnFee: Integer
+validLoan: Boolean
+LoanCount: Long
+setValidLoan(flag: Boolean): Void
+calculateLateReturnFee(loanPeriod: Integer): Integer
+calculateReplacementFee(price: Float): Integer
+searchLoan(itemID: String): Loan
+searchLoan(borrowerRef: Borrower): Loan
+addLoan(loanRef: Loan): Void
+updateLoan(loanRef: Loan): Void
+decreaseLoanCount(): Void
+increaseLoanCount(): Void
+getNumOfLoan(): Void
+getItem(LoanRef: Loan): Item
Manages
*
Librarian
+name: String
+userId: String
+password: String
+logInFlag: Boolean
+validate(userI: String, password: String)
+logOut(userID: String)
Manages
1
Refer to
0..*
+*
Reservation
+reserveDate: Date
+searchReservation(isbn: ISBNType): Reservation
+searchReservation(titleRef: Title): Reservation
+searchReservation(borrowerRef: Borrower): Reservation[]
+addReservation(reservationRef: Reservation): Void
+removeReservation(reservationRef: Reservation): Void
+printNotifyCard(titleRef: Title): Void
+printCard(resrvationRef: Reservation): Void
+getTitle(reservationRef: Reservation): Title
0..*
*
0..*
Borrower
+name: String
+ssn: String
+address: String
+reservationCount: Integer
+loanCount: Integer
+increaseLoanCount(): Void
+decreaseLoanCount(): Void
+increaseReservationCount(): Void
+decreaseReservationCount(): Void
+searchBorrower(ssn: String): Borrower
+addBorrower(borrowerRef: Borrower): Void
+removeBorrower(ssn: String): Void
+updateBorrower(borrwerRef: Borrower): Void
Has
1
1
Has



Database

-Title: Map
+Item: Map
+Borrower: Map
+Loan: Map
+Reservation: Map

+searchTitleDB(isbn: ISBNType): Title
+addTtileDB(titleRef: Title): Void
+removeTitleDB(titleRef: Title): Void
+updateTitleDB(titleRef: Title): Void
+searchItemDB(itemID: String): Item
+addItemDB(itemRef: Item): Void
+removeItemDB(itemRef: Item): Void
+updateItemDV(itemRef: Item): Void
+searchBorrowerDB(ssn: String): Borrower
+addBorrowerDB(borrowerRef: Borrower): Void
+removeBorrowerDB(borrowerRef: Borrower): Void
+updateBorrowerDB(borrowerRef: Borrower): Void
+searchLoanDB(itemID: String): Loan
+searchLoanDB(borrwerRef: Borrower): Loan
+addLoanDB(loanRef: Loan): Void
+updateLoanDB(loanRef: Loan): Void
+searchReservationDB(isbn: ISBNType): Reservation
+searchReservationDB(titleRef: Title): Reservation
+searchReservationDB(borrowerRef: Borrower): Resrvation[]
+addReservationDB(reservationRef: Reservation): Void
+removeReservationDB(reservationrRef: Reservation): Void
+validateDB(userID: String, password: String): Void

Title

+name: String
+isbn: ISBNType
+price: Flot
+loanPeriod: Integer
+numOfItem: Integer
+availalbeCount: Integer
+reservationCount: Integer

+increaseAvailableCount(): Void
+decreaseAvailableCount(): Void
+increaseNumOfItem(): Void
+decreaseNumOfItem(): Void
+getNumOfItem(): Integer
+getPrice(): Float
+getLoanPeriod(): Integer
+getNewItemID(): String
+searchTitle(isbn: ISBNType): Title
+addTitle(titleRef: Title): Void
+removeTitle(titleRef: Title): Void
+updateTitle(titleRef: Title): Void
+isReserved(titleRef: Title): Boolean
+increaseReservationCount(): Void
+decreaseReservationCount(): Void

Magazine

+publishCycle: String
+month: String

Book

+author: String

Item

+itemID: String
+available: Boolean
+lost: Boolean

+isBorrowed(): Boolean
+setLost(flag: Boolean): Void
+searchItem(itemID: String): Item
+addItem(itemRef: Item): Void
+updateItem(itemRef: Item): Void
+removeItem(itemRef: Item): Void
+setAvailable(flag: Boolean): Void
+getTitle(itemRef: Item): Title

Controller

+mkaeReservation()
+removeReservation()
+LendItem()
+returnItem()
+getReplacementFee()
+addTitle()
+removeTitle()
+updateTitle()
+addItem()
+removeItem()
+updateItem()
+addBorrower()
+removeBorrower()
+updateBorrower()
+log-In()
+log-Out()
+countLoans()

Loan

+checkInDate: Date
+checkOutDate: Date
+lateReturnFee: Integer
+validLoan: Boolean
+LoanCount: Long

+setValidLoan(flag: Boolean): Void
+calculateLateReturnFee(loanPeriod: Integer): Integer
+calculateReplacementFee(price: Float): Integer
+searchLoan(itemID: String): Loan
+searchLoan(borrowerRef: Borrower): Loan
+addLoan(loanRef: Loan): Void
+updateLoan(loanRef: Loan): Void
+decreaseLoanCount(): Void
+increaseLoanCount(): Void
+getNumOfLoan(): Void
+getItem(LoanRef: Loan): Item

Librarian

+name: String
+userId: String
+password: String
+logInFlag: Boolean

+validate(userI: String, password: String)
+logOut(userID: String)

Reservation

+reserveDate: Date

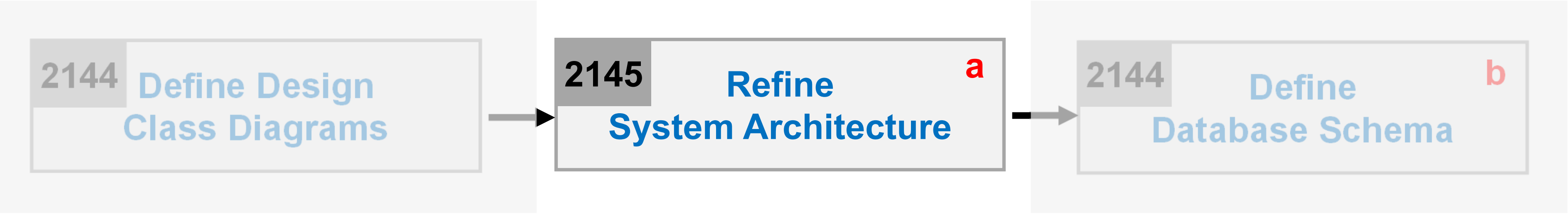
+searchReservation(isbn: ISBNType): Reservation
+searchReservation(titleRef: Title): Reservation
+searchReservation(borrowerRef: Borrower): Reservation[]
+addReservation(reservationRef: Reservation): Void
+removeReservation(reservationRef: Reservation): Void
+printNotifyCard(titleRef: Title): Void
+printCard(resrvationRef: Reservation): Void
+getTitle(reservationRef: Reservation): Title

Borrower

+name: String
+ssn: String
+address: String
+reservationCount: Integer
+loanCount: Integer

+increaseLoanCount(): Void
+decreaseLoanCount(): Void
+increaseReservationCount(): Void
+decreaseReservationCount(): Void
+searchBorrower(ssn: String): Borrower
+addBorrower(borrowerRef: Borrower): Void
+removeBorrower(ssn: String): Void
+updateBorrower(borrwerRef: Borrower): Void

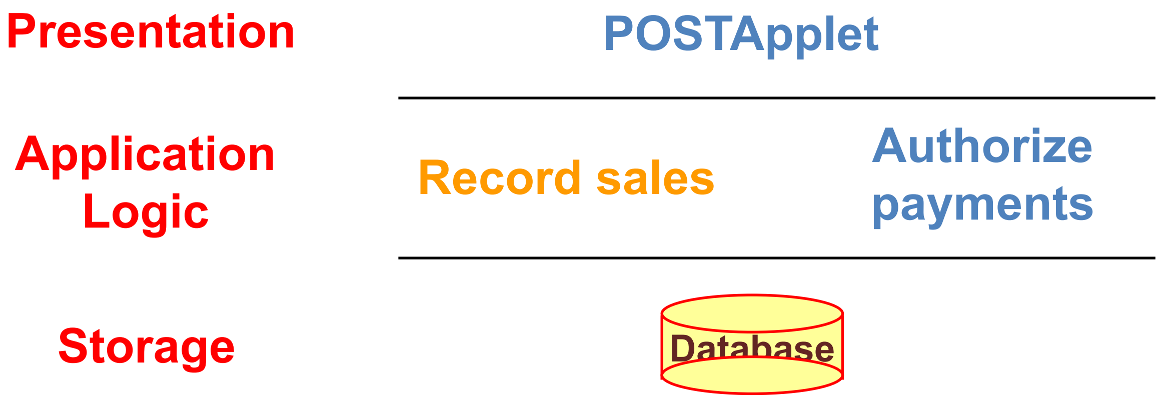
Activity 2045. Refine System Architecture



- Description
  - Refine draft system architecture developed in the plan stage
  - Input : Draft System Architecture
  - Output : **A package diagram**, **a deployment diagram**
    - Standards Applied : UML's Package Diagram and UML's Deployment Diagram

Activity 2045. Refine System Architecture

- Steps (1~3: Deployment diagram , 4~7: Package diagram)
  1. Define a 3-tier layered system architecture
     - Presentation Layer : Windows, Reports, and so on
     - Application Logic Layer : Tasks and rules that govern the process
     - Storage Layer : Persistent storage mechanism



Presentation

POSTApplet

Application Logic

Record sales

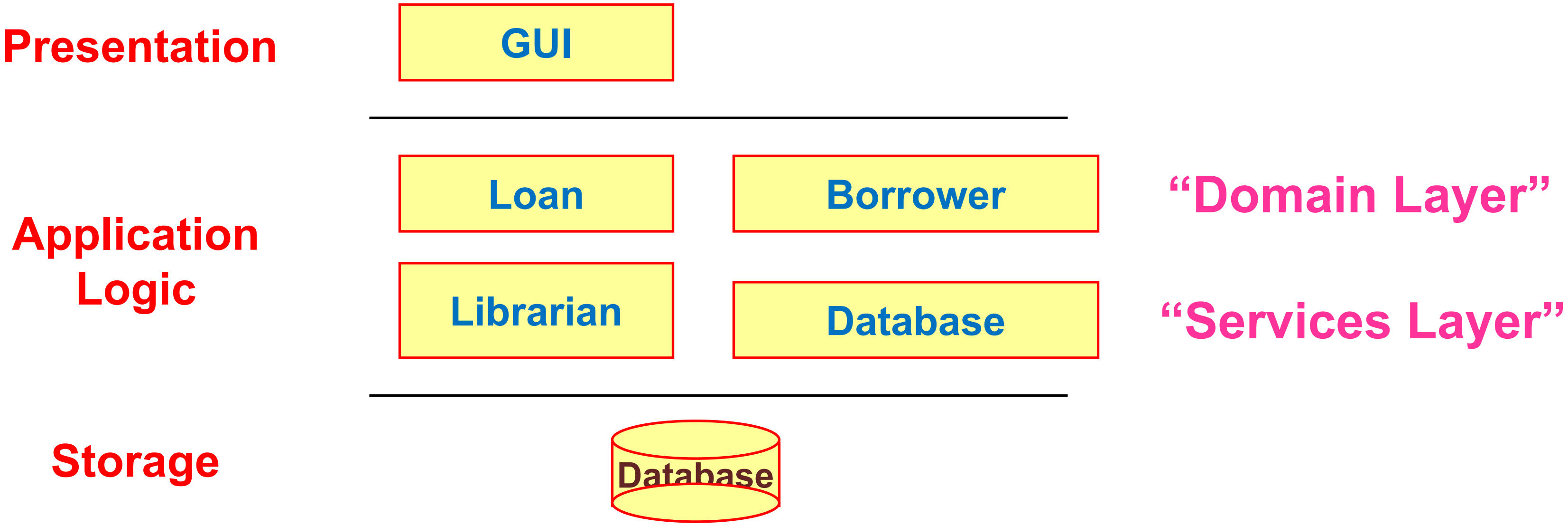
Authorize payments

Storage

Database

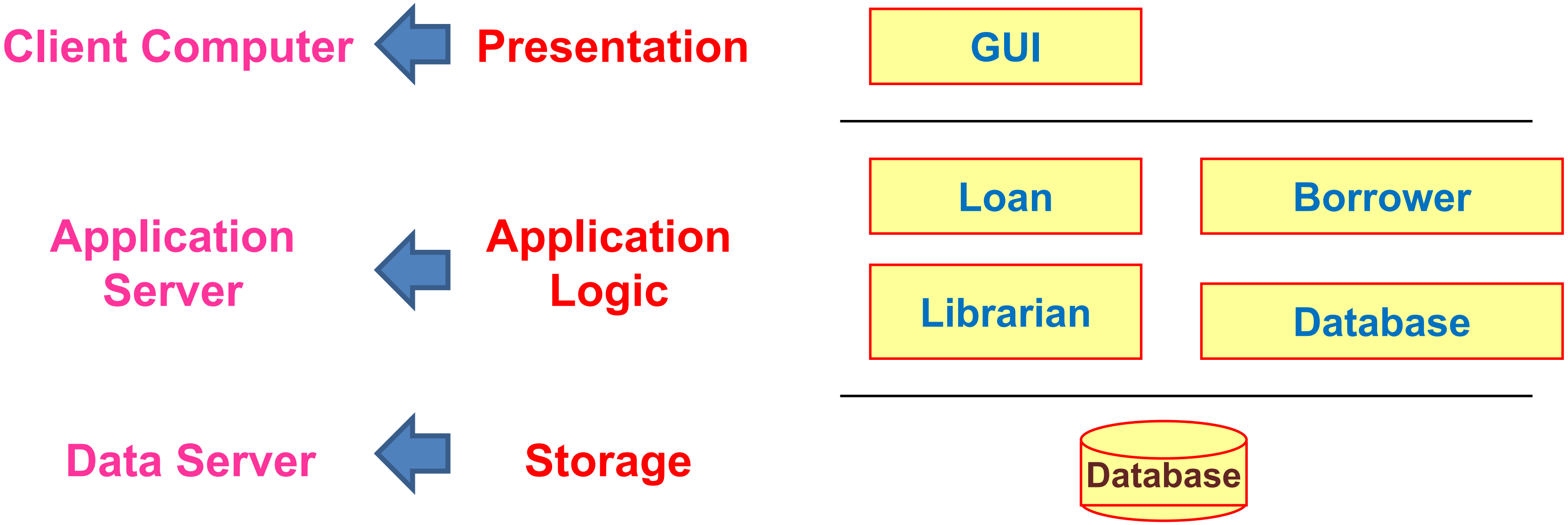
Activity 2045. Refine System Architecture

2. Decompose the application logic tier into finer layers
   - Domain object layer
     - Classes representing domain concepts
   - Service layer
     - Service objects for functions such as database interaction, reporting, communications, security, and so on



Activity 2045. Refine System Architecture

3. Assign each tier into different physical computing nodes, and/or different processes



Client Computer ← Presentation

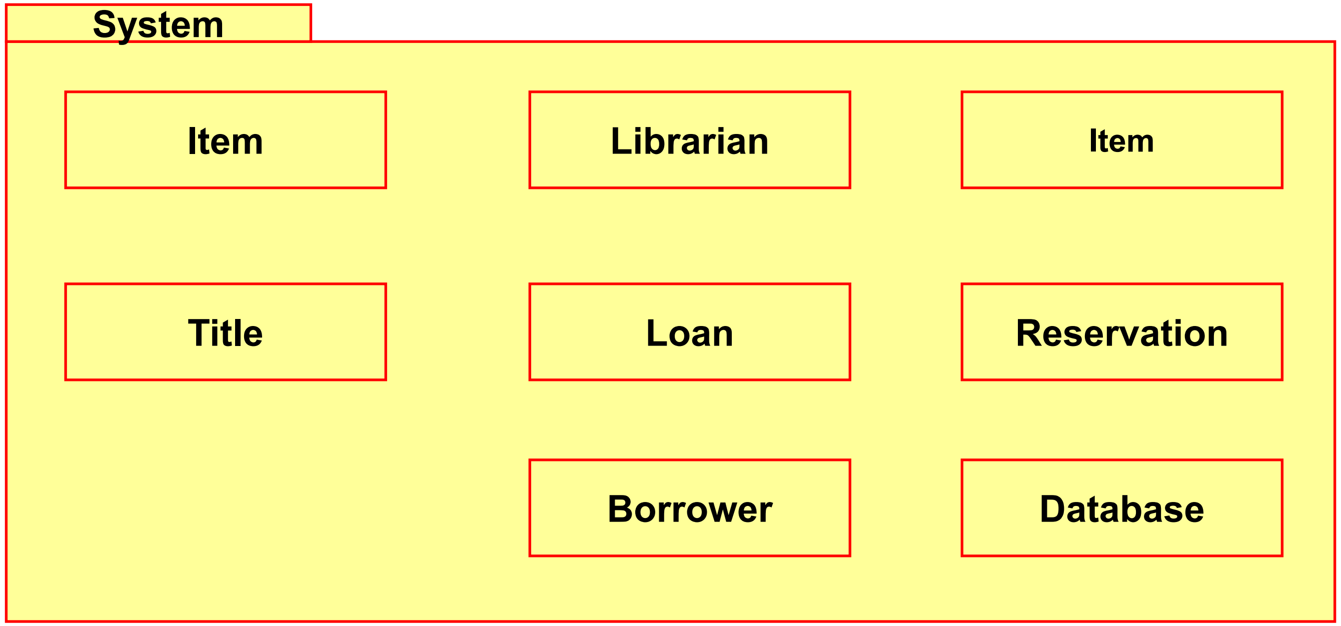
Application Server ← Application Logic

Data Server ← Storage

Activity 2045. Refine System Architecture

4. Identify packages

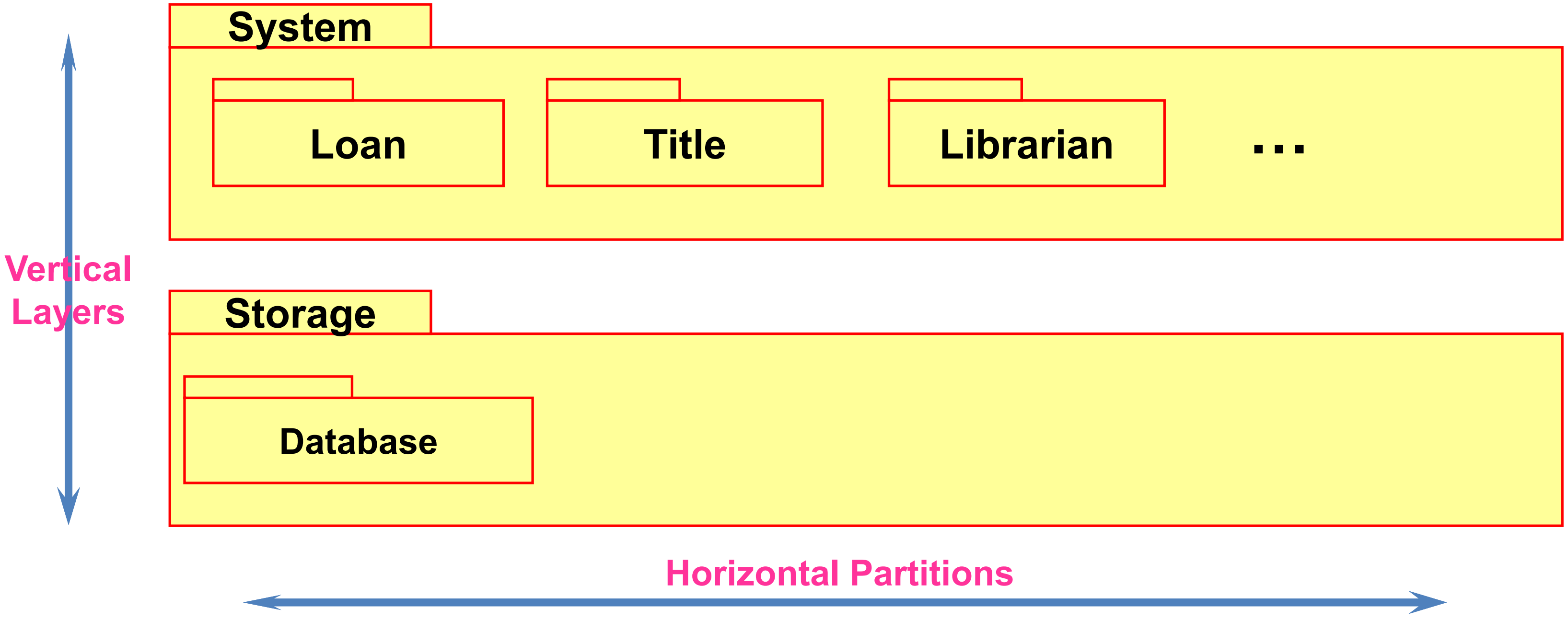
- Place elements together
  - that are in the same subject area-closely related by concept or purpose, or that are in a type hierarchy together
  - that participate in the same use cases or
  - that are strongly associated



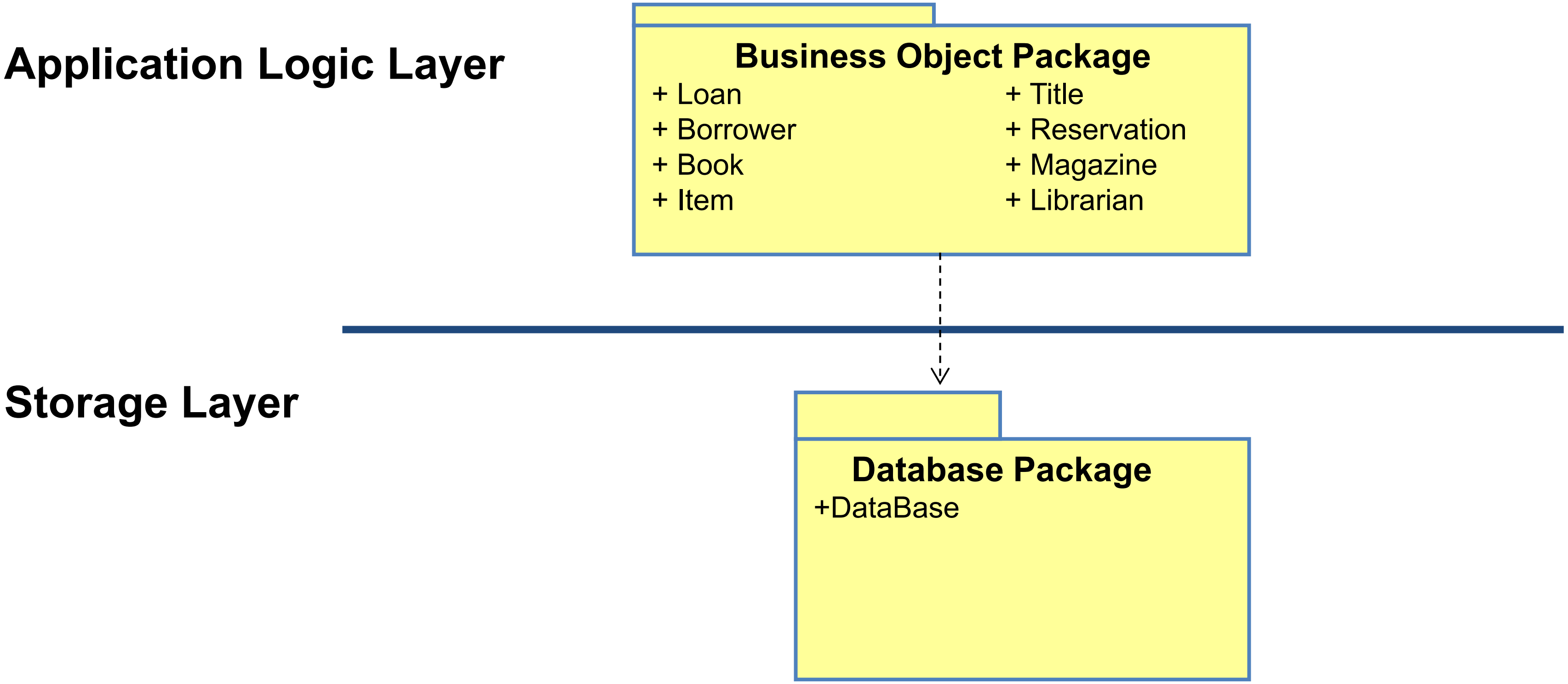
Activity 2045. Refine System Architecture

5. Layers of the architecture : vertical tiers

   Partitions of the architecture : horizontal division of relatively parallel subsystems

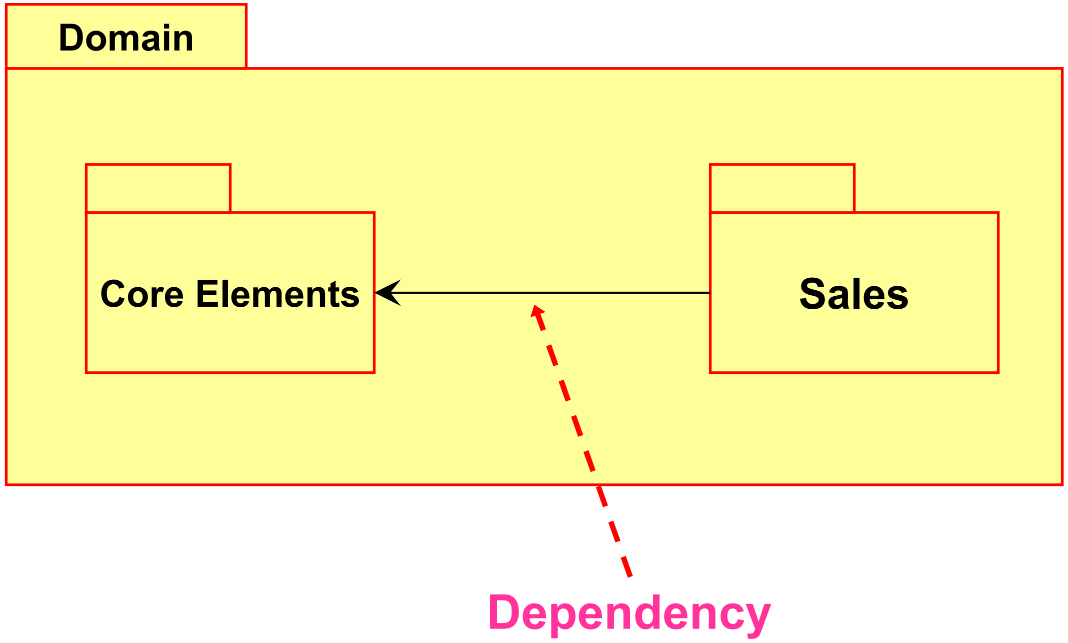


Activity 2045. Refine System Architecture



Activity 2045. Refine System Architecture

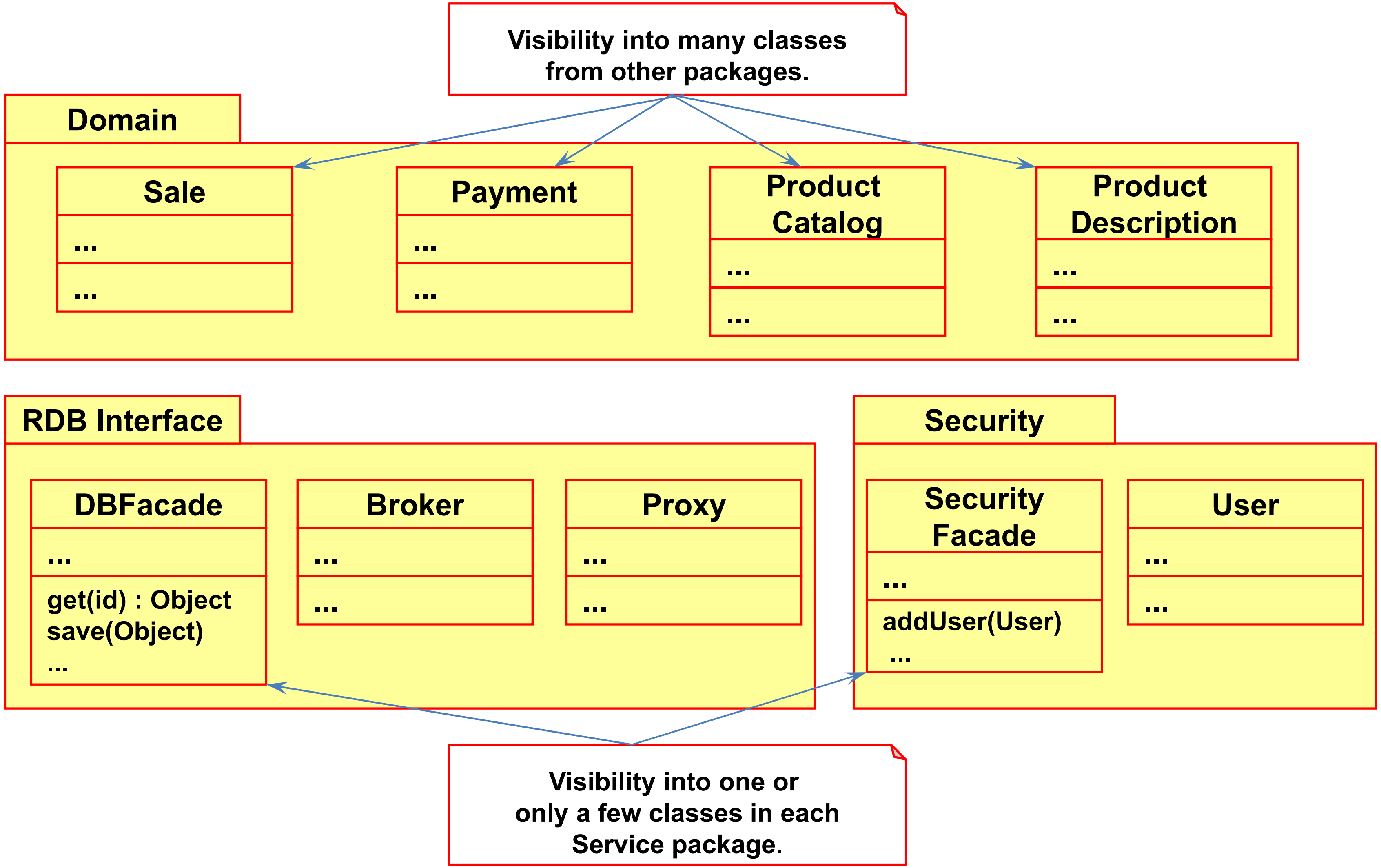
6. Determine package dependencies
   - Dependency relationships indicates coupling between packages.



Activity 2045. Refine System Architecture

7. Assign visibility between package classes
   - Access into the Domain packages
     - Some packages, typically the presentation package, have visibility into many of the classes representing domain concepts
   - Access into the Service packages
     - Some packages, typically the Domain and Presentation packages, have visibility into only one or a very few classes in each particular Service package
   - Access into the Presentation packages
     - No other packages have direct visibility to the Presentation layer

Activity 2045. Refine System Architecture



Visibility into many classes from other packages.

Domain

Sale
...
...

Payment
...
...

Product Catalog
...
...

Product Description
...
...

RDB Interface

DBFacade
...
get(id) : Object
save(Object)
...

Broker
...
...

Proxy
...
...

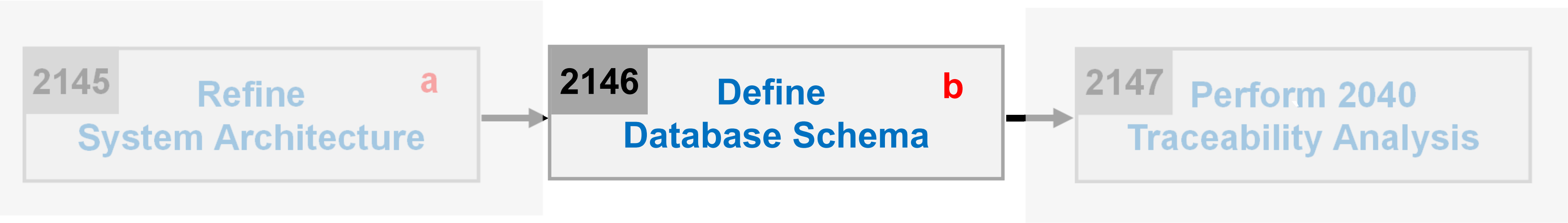
Security

Security Facade
...
addUser(User)
...

User
...
...

Visibility into one or only a few classes in each Service package.

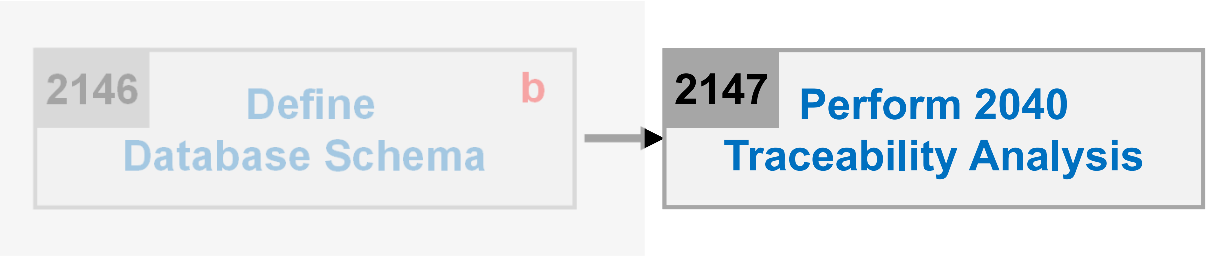
Activity 2046. Define Database Schema


2145
Refine System Architecture
a
2146
Define Database Schema
b
2147
Perform 2040 Traceability Analysis

- Description
  - Design database, table, and records
  - Map classes into tables
  - Input : Design Class Diagram
  - Output : A Database Schema

- Steps:
  1. Map classes into tables
  2. Map relationships between classes into relations between tables
  3. Map attributes into fields of tables
  4. Design Schema

Activity 2047. Perform 2040 Traceability Analysis



- Description
  - Link all elements from the abstract at 2030 to details at 2040 (design class diagram and system test cases)
  - Express all traces from requirements to system test cases
  - Input : Real use cases, functional requirements, design class diagram, operation contracts, system test cases
  - Output : **A 2040 traceability graph**

Activity 2047. Perform 2040 Traceability Analysis

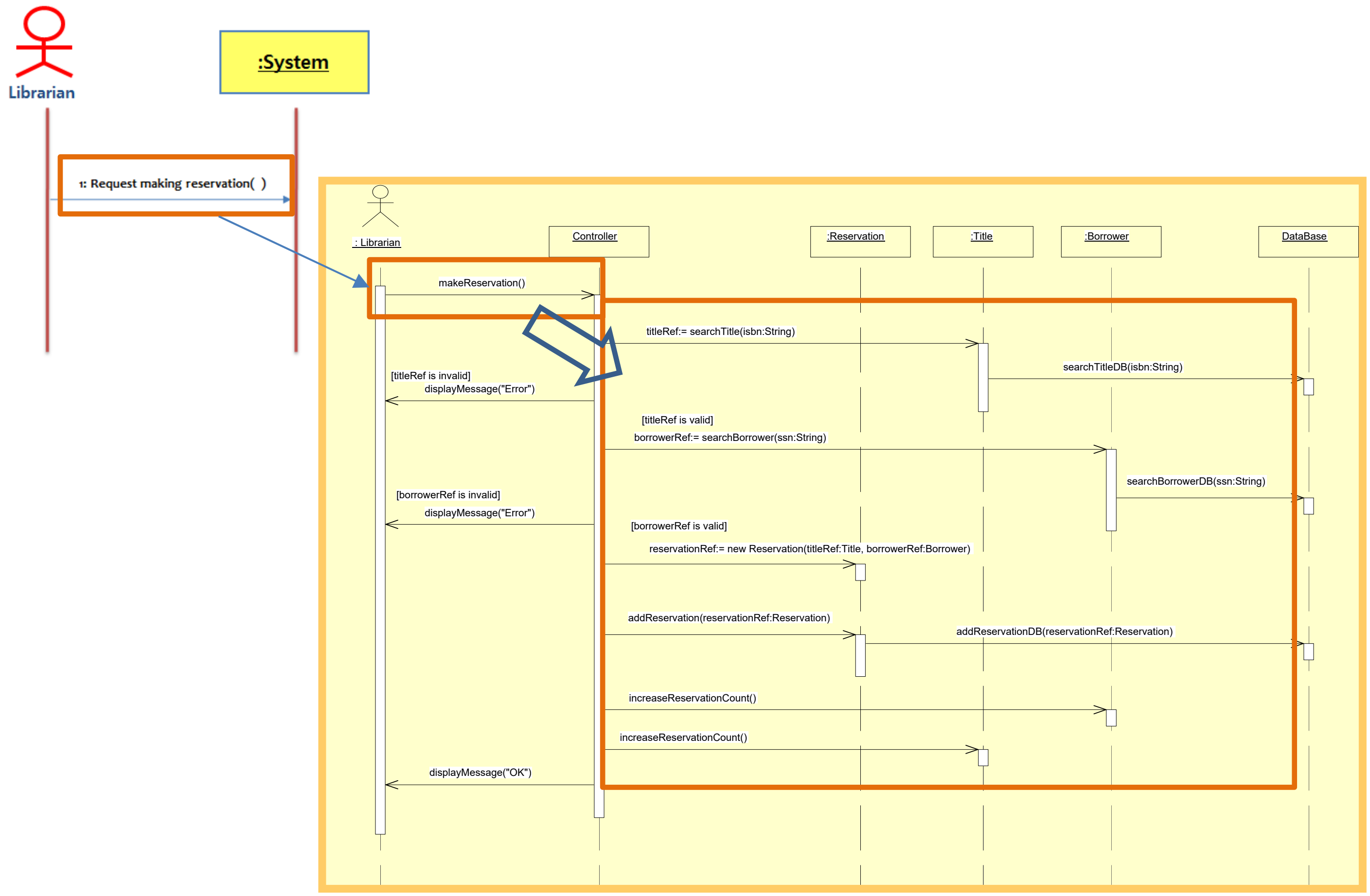
- Step 1.
  - Identify the system operation, system interaction diagram, and class diagram

- Step 2.
  - Identify the operations which are connected with system operation and others

- Step 3.
  - Grasp the relations between methods extracted by interaction diagram and system operations in sequence diagram
  - Classify the connection system operation directly and others

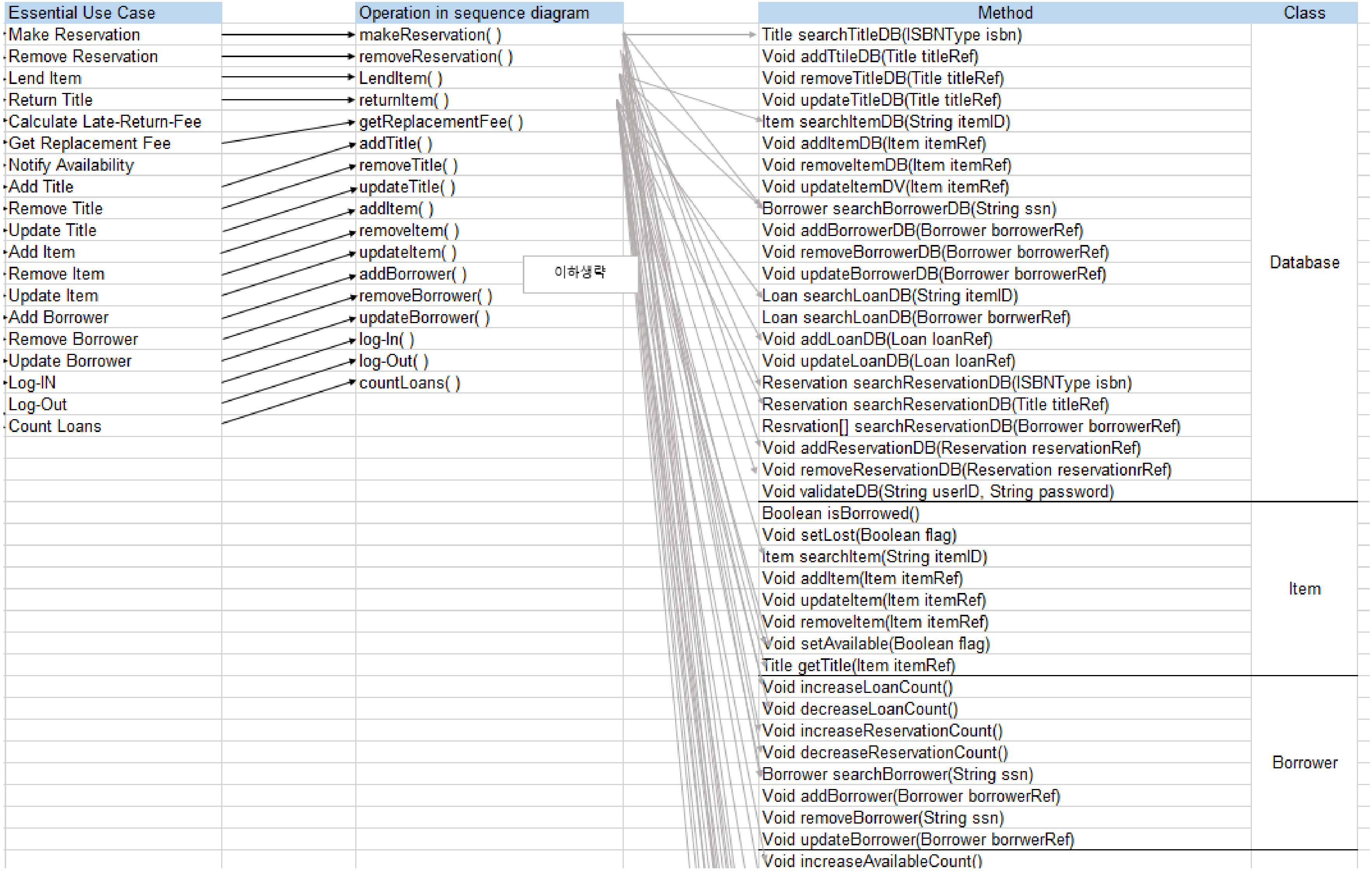
Activity 2047. Perform 2040 Traceability Analysis

- Draw up the traces between a system operation (2035) and operations in interaction diagrams (2043)



Librarian
:System
1: Request making reservation( )
: Librarian
Controller
:Reservation
:Title
:Borrower
DataBase
makeReservation()
titleRef:= searchTitle(isbn:String)
searchTitleDB(isbn:String)
[titleRef is invalid]
displayMessage("Error")
[titleRef is valid]
borrowerRef:= searchBorrower(ssn:String)
searchBorrowerDB(ssn:String)
[borrowerRef is invalid]
displayMessage("Error")
[borrowerRef is valid]
reservationRef:= new Reservation(titleRef:Title, borrowerRef:Borrower)
addReservation(reservationRef:Reservation)
addReservationDB(reservationRef:Reservation)
increaseReservationCount()
increaseReservationCount()
displayMessage("OK")

Activity 2047. Perform 2040 Traceability Analysis



Essential Use Case
Make Reservation
Remove Reservation
Lend Item
Return Title
Calculate Late-Return-Fee
Get Replacement Fee
Notify Availability
Add Title
Remove Title
Update Title
Add Item
Remove Item
Update Item
Add Borrower
Remove Borrower
Update Borrower
Log-IN
Log-Out
Count Loans

Operation in sequence diagram
makeReservation( )
removeReservation( )
LendItem( )
returnItem( )
getReplacementFee( )
addTitle( )
removeTitle( )
updateTitle( )
addItem( )
removeItem( )
updateItem( )
addBorrower( )
removeBorrower( )
updateBorrower( )
log-In( )
log-Out( )
countLoans( )

이하생략

Method	Class
Title searchTitleDB(ISBNType isbn)	Database
Void addTtileDB(Title titleRef)	
Void removeTitleDB(Title titleRef)	
Void updateTitleDB(Title titleRef)	
Item searchItemDB(String itemID)	
Void addItemDB(Item itemRef)	
Void removeItemDB(Item itemRef)	
Void updateItemDV(Item itemRef)	
Borrower searchBorrowerDB(String ssn)	
Void addBorrowerDB(Borrower borrowerRef)	
Void removeBorrowerDB(Borrower borrowerRef)	
Void updateBorrowerDB(Borrower borrowerRef)	
Loan searchLoanDB(String itemID)	
Loan searchLoanDB(Borrower borrwerRef)	
Void addLoanDB(Loan loanRef)	
Void updateLoanDB(Loan loanRef)	
Reservation searchReservationDB(ISBNType isbn)	
Reservation searchReservationDB(Title titleRef)	
Resrvation[] searchReservationDB(Borrower borrowerRef)	
Void addReservationDB(Reservation reservationRef)	
Void removeReservationDB(Reservation reservationrRef)	
Void validateDB(String userID, String password)	
Boolean isBorrowed()	Item
Void setLost(Boolean flag)	
Item searchItem(String itemID)	
Void addItem(Item itemRef)	
Void updateItem(Item itemRef)	
Void removeItem(Item itemRef)	
Void setAvailable(Boolean flag)	
Title getTitle(Item itemRef)	
Void increaseLoanCount()	Borrower
Void decreaseLoanCount()	
Void increaseReservationCount()	
Void decreaseReservationCount()	
Borrower searchBorrower(String ssn)	
Void addBorrower(Borrower borrowerRef)	
Void removeBorrower(String ssn)	
Void updateBorrower(Borrower borrwerRef)	
Void increaseAvailableCount()	

Essential UseCase	S-Link
시간 확인	S1
	S2, S3, S4
	S5, S4
	S6, S3, S4
	S7
	S8
	S16
	S9
	S10
	S11, S3, S4
	S12
	S13
	S14
	S17
	S15
	S4.1
	S4.2
	S4.3
	S4.4
	S5.1, S5.2
	S5.2, S5.3
	S5.3, S5.4
	S5.5
시	S6.1
과	S6.2, S6.3
님/	S5, S4
	S7.1
	S7.2
연새 보느 번완	S7.3

SID	Operation in Sequence Diagram	M-Link
S1	selectTimeViewMode	M15,M1
S2	selectTimeSetupMode	M16,M2,M12,M5
S3	changeValue	M13,M12
S4	goNext	M14,M12
S5	selectAlarmViewMode	M18,M3,M14
S6	selectAlarmSetupMode	M17,M4,M5,M12
S7	addAlarm	M20,M17,M12,M3
S8	deleteAlarm	M19,M3
S9	clearAlarmNotice	M21,M11,M6
S10	selectTimerViewMode	M26,M7
S11	selectTimerSetupMode	M27,M8,M5,M12
S12	startTimer	M25,M7
S13	pauseTimer	M22,M7
S14	resetTimer	M23,M7
S15	clearTimerNotice	M24,M7
S16	alarmBeep	M31,M3
S17	timerBeep	M31,M7
S4.1	startStopWatch()	M4.1, M4.2, M4.3, M4.4, M4.5
S4.2	stopStopWatch()	M4.6, M4,7
S4.3	restartStopWatch()	M4.2, M4.3, M4.4, M4.5, M4.8
S4.4	resetStopWatch()	M4.5, M4.9, M4.10
S5.1	createNewAnniversary()	M5.1M5.2
S5.2	inputDateTime()	M5.3M5.4M5.5M5.6M5.7M5.8
S5.3	selectAnniversary()	M5.9, M5.2
S5.4	deleteAnniversary()	M5.10, M5.11, M5.12
S5.5	dismiss()	M5.13, M5.14, M5.15
S6.1	requestCreateLotteryNumber	M6.1,M6.6, M6.7, M6.10, M6.11
S6.2	saveLotteryNumber	M6.5
S6.3	setReminder	M6.6
S7.1	select4Mode	M6.2, M6.3, M6.
S7.2	requestFactoryReset	M6.9
S7.3	requestChangeCurrentMode	M6.13

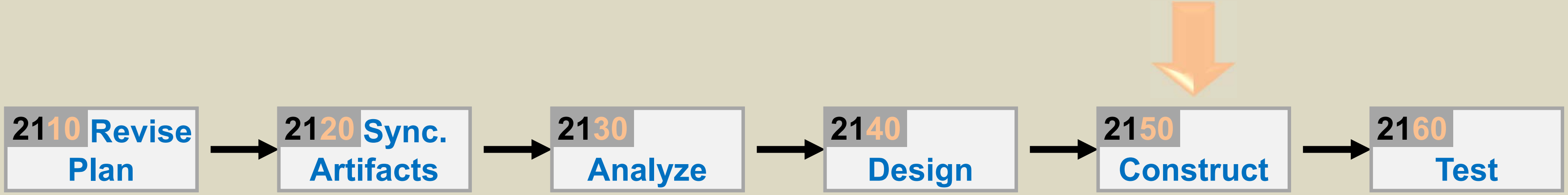
MID	Method	Class
M1	displayCurrentTime	DisplayManager
M2	displaySetupMode	
M3	displayAlarm	
M4	displayNextAlarm	
M5	blinkSetupItem	
M6	displayCurrentMode	
M7	displayTimer	
M8	displaySetupMode	
M9	viewMode	Mode
M10	setupMode	
M11	getPreviousMode	
M12	saveValue	
M13	changeValue	
M14	goNext	
M15	selectTimerViewMode	TimeMode
M16	selectTimeSetupMode	
M17	selectAlarmSetupMode	AlarmMode
M18	selectAlarmViewMode	
M19	deleteCurrentAlarm	
M20	addNewAlarm	
M21	clearAlarm	
M22	pauseTimerVlaue	TimerMode
M23	resetTimerValue	
M24	clearTimer	
M25	runTimer	
M26	selectTimerViewMode	
M27	selectTimerSetupMode	
M28	registerTickObserver	TimeManager
M29	setTime	
M30	tick	TickObserver
M31	beep	BeepManager
M32	(Input Event 생성)	InputProcessor

M4.1	startStopWatch()	StopWatchMode
M4.2	registerTickObserver()	TimeManager
M4.3	startTick()	TimeManager
M4.4	tick()	TimeManager
M4.5	updateTime()	DisplayManager
M4.6	stopStopWatch()	StopWatchMode
M4.7	stopTick()	TimeManager
M4.8	restartStopWatch()	StopWatchMode
M4.9	resetStopWatch()	StopWatchMode
M4.10	unregisterTick()	TimeManager
M5.1	createNewAnniversary()	AnniversaryMode
M5.2	getSlot()	AnniversaryStorage
M5.3	inputDateTime()	AnniversarySlot
M5.4	updateDateTime()	AnniversarySlot
M5.5	save()	AnniversarySlot
M5.6	setAlarm()	AlarmManager
M5.7	updateDate()	DisplayManager
M5.8	updateTitle()	DisplayManager
M5.9	selectAnniversary()	AnniversaryMode
M5.10	deleteAnniversary()	AnniversaryMode
M5.11	deleteSlot()	AnniversaryStorage
M5.12	deleteAlarm()	AlarmManager
M5.13	dismiss()	AnniversaryAlarm
M5.14	stop()	LightBuzzerManager
M5.15	turnOff()	LightBuzzerManager
M6.1	displayLotteryNumber	DisplayManager
M6.2	select4Mode	
M6.3	displayModeList	
M6.4	updateModeList	
M6.5	saveLotteryNumber	LotteryStorage
M6.6	sortLotteryNumber	Lottery
M6.7	setReminder	LotteryAlarm
M6.8	save4Mode	SettingsStorage
M6.9	resetData	
M6.10	sortLotteryNumber	Lottery
M6.11	generateLotteryNumber	RandomGenerator
M6.13	changeCurrentMode	ModeManager

중복 methods

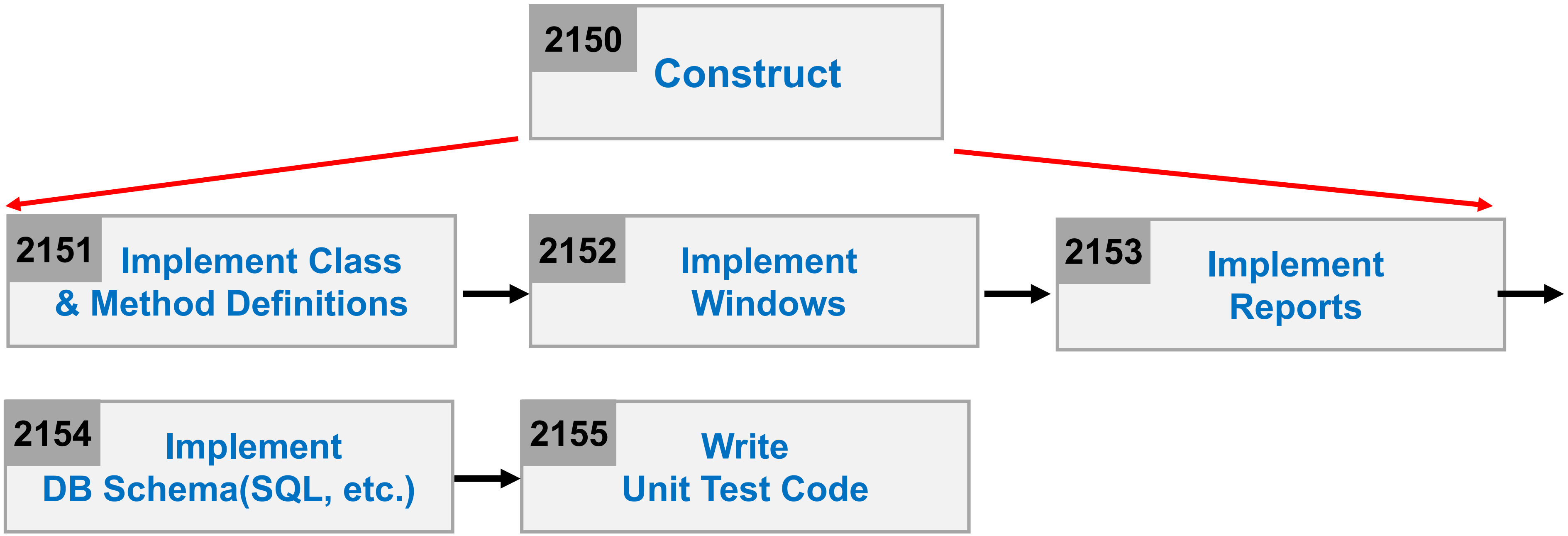
M2 M8	displaySetupMode
M28, M	registerTickObserver()
M15, M	selectTimerViewMode
M6.6 M	sortLotteryNumber
M30, M	tick()

Phase 2050. Construct



Phase 2050. Construct

- Phase 2050 Activities



2150 Construct

2151 Implement Class & Method Definitions

2152 Implement Windows

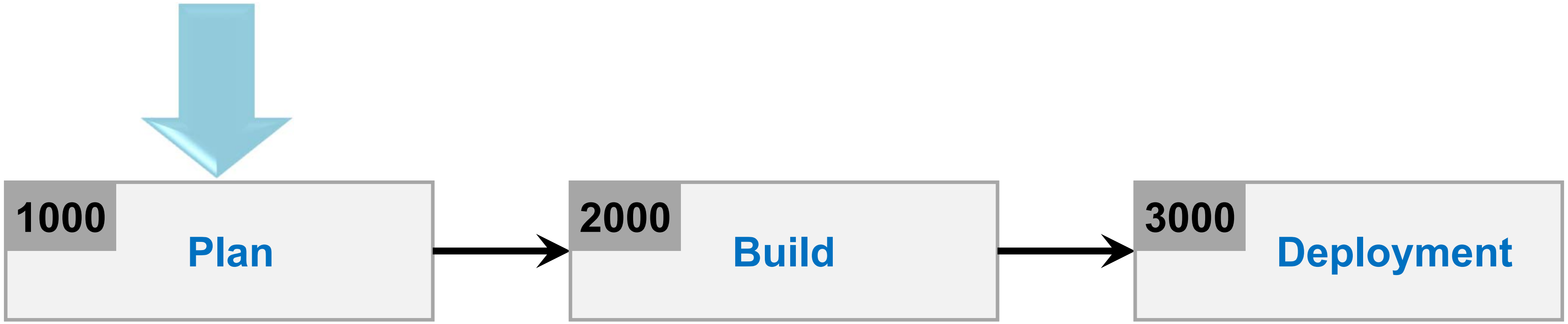
2153 Implement Reports

2154 Implement DB Schema(SQL, etc.)

2155 Write Unit Test Code

OOPT - Case Study
Library Management System (LMS)

Stage 1000. Plan

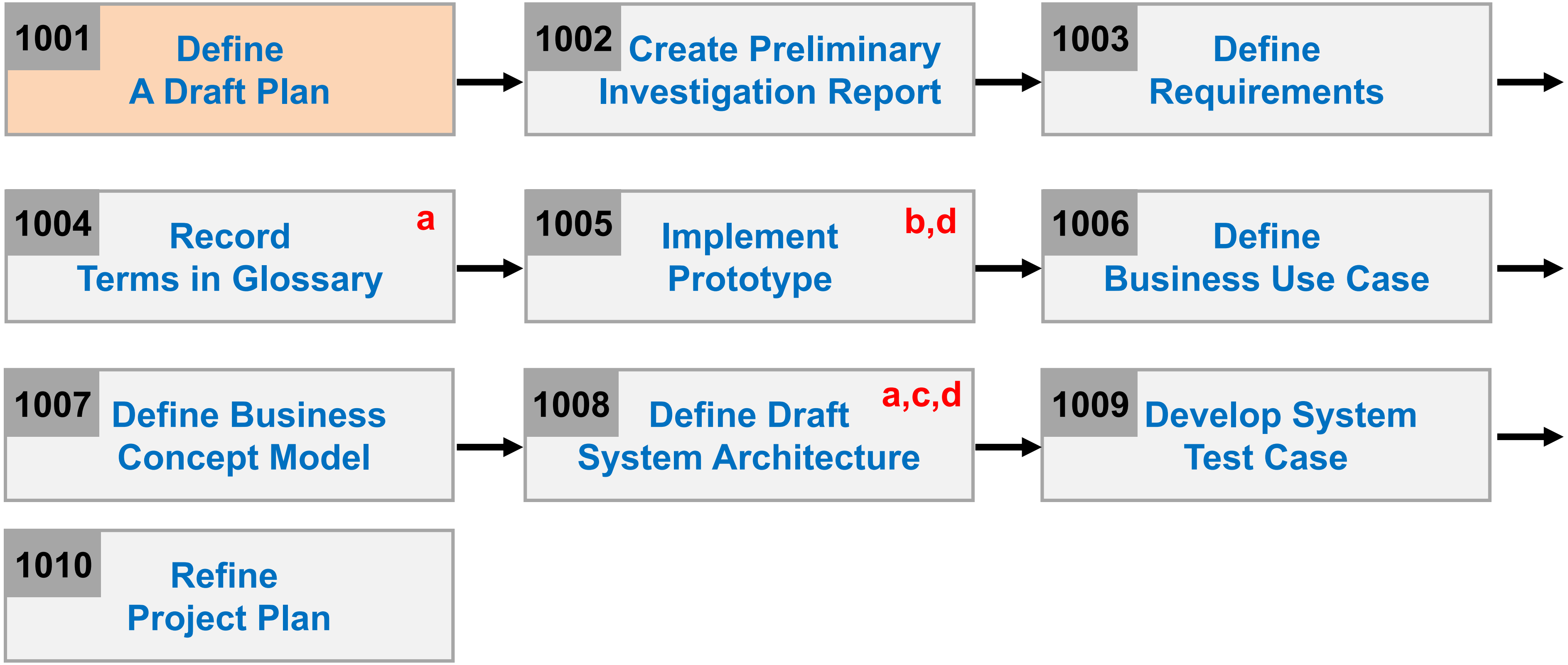


1000 Plan

2000 Build

3000 Deployment

Activity 1001. Define a Draft Plan



1001 Define A Draft Plan

1002 Create Preliminary Investigation Report

1003 Define Requirements

1004 Record Terms in Glossary a

1005 Implement Prototype b,d

1006 Define Business Use Case

1007 Define Business Concept Model

1008 Define Draft System Architecture a,c,d

1009 Develop System Test Case

1010 Refine Project Plan

Activity 1001. Define a Draft Plan

- Motivation
  - The size of title volumes and the number of users for a city library are sharply increasing.
  - Hence, the city wants to develop a 'Library Management System' in order to automate most of the library operations.
  - Among the various library operations, they want to automate the most commonly used operations such as loan, reservation, purchase, discarding old books, and simple statistics.

- Project Objectives
  - To develop a computerized library management software, that provides typical library operations such as:
    - Lend and return books, Reserve books, Maintaining Borrow information, and Purchasing new books.
  - The new software should be easy to learn and use, and efficient.

Activity 1001. Define a Draft Plan

- Functional Requirements
  - Lend titles.
  - Return titles.
  - Reserve titles.
  - Purchase new titles.
  - Discard old titles.
  - Maintain borrower information.

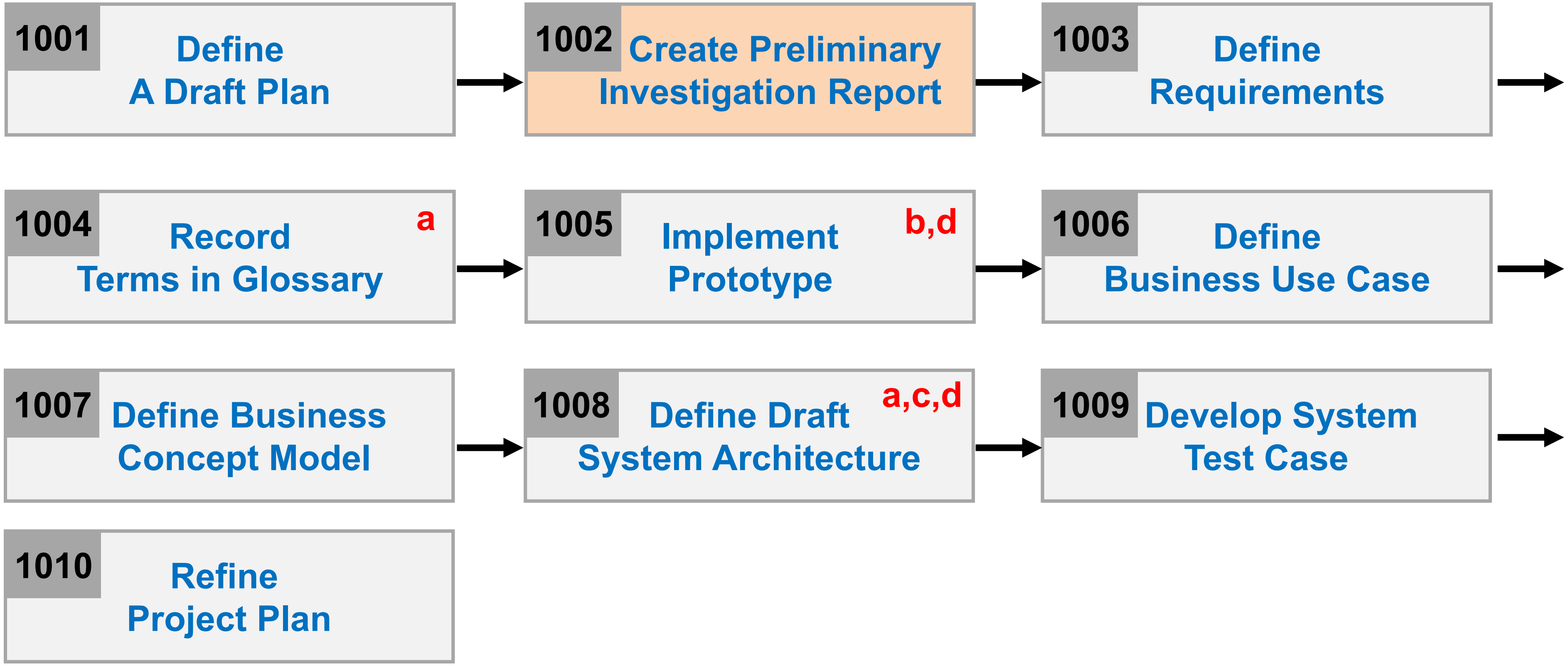
- Non-Functional Requirements
  - The average response time for front desk operations should be less than 5 seconds.
  - The system should be designed to expandable and maintainable.

Activity 1001. Define a Draft Plan

- Resource Estimation
  - Human Efforts(Man-Month): 6-10 M/M ?
  - Human Resource:
  - Project Duration:
  - Cost:

- Other Information
  - Future Version
    - Adopt 3-Tier Client/Server Architecture.
    - Add Web Interface.

Activity 1002. Create Preliminary Investigation Report



1001 Define A Draft Plan

1002 Create Preliminary Investigation Report

1003 Define Requirements

1004 Record Terms in Glossary a

1005 Implement Prototype b,d

1006 Define Business Use Case

1007 Define Business Concept Model

1008 Define Draft System Architecture a,c,d

1009 Develop System Test Case

1010 Refine Project Plan

Activity 1002. Create Preliminary Investigation Report

- Alternative Solutions
  - Purchasing such a library managing software, if available.
  - Outsourcing
  - Other Options

- Project Justification (Business Demands)
  - Cost
  - Duration
  - Risk
  - Effect

Activity 1002. Create Preliminary Investigation Report

- Risk Management

Risk	Probability	Significance	Weight
Lack of OO experience	4	4	16
First adoption of OSP	4	5	20
Lack of domain knowledge	1	5	5
Team communication	3	3	9
Problem of requirements change	1	4	4
Lack of tool skill	2	2	4
Wandering	3	5	15

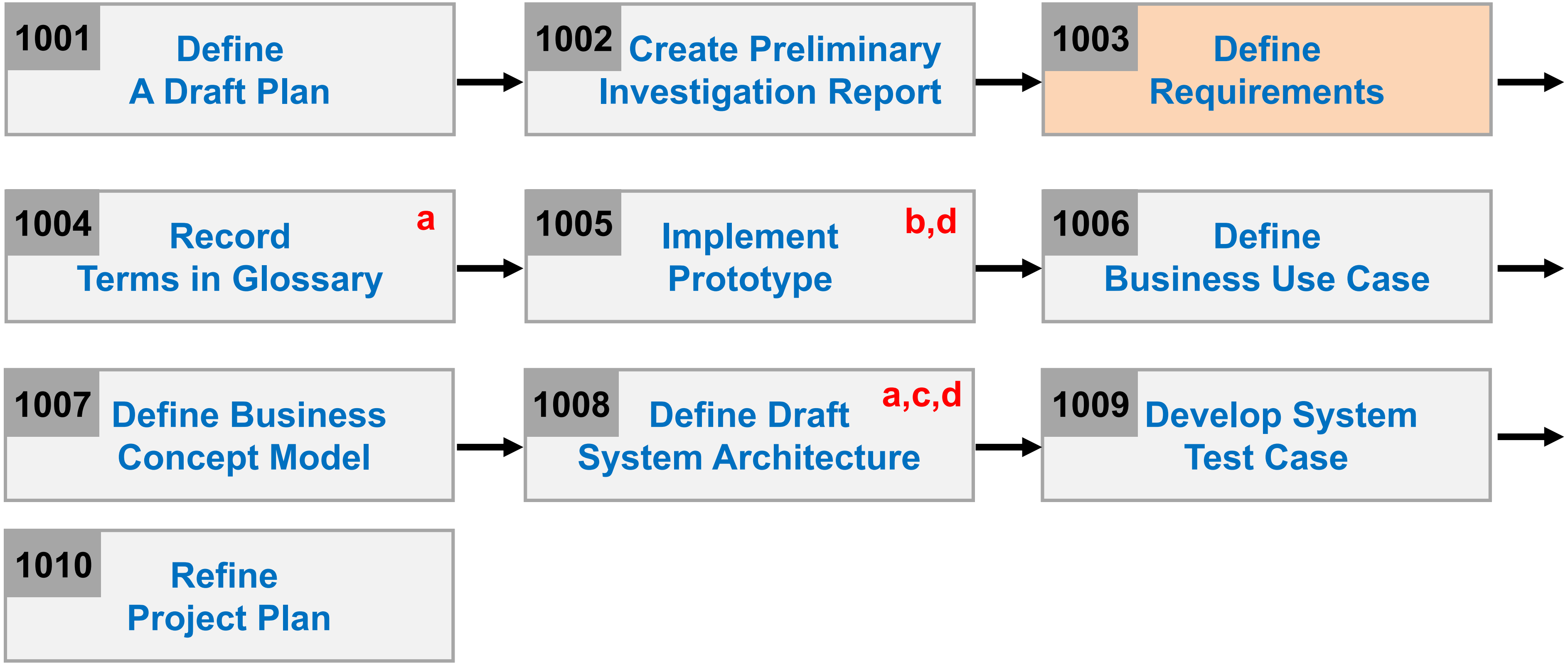
Activity 1002. Create Preliminary Investigation Report

- Risk Reduction Plan
  - First adoption of OSP (20) : Try a pilot project using OSP
  - Lack of OO Project Experience (16) : Take part in a study group
  - Team Communication (9) : Have a team meeting on every Friday night

- Market Analysis
  - A few generic packages are available, however too expensive.
  - May be able to market the software to other similar-scaled libraries.

- Other Managerial Issues
  - The project should be completed by June, 2008.
    - Plan to participate in a SW exhibition.

Activity 1003. Define Requirements



1001 Define A Draft Plan

1002 Create Preliminary Investigation Report

1003 Define Requirements

1004 Record Terms in Glossary a

1005 Implement Prototype b,d

1006 Define Business Use Case

1007 Define Business Concept Model

1008 Define Draft System Architecture a,c,d

1009 Develop System Test Case

1010 Refine Project Plan

Activity 1003. Define Requirements

- Functional Requirements (Version 0.9)
  - A library lends books and magazines to borrowers, who are registered in the system.
  - A library handles the purchase of new titles. Popular titles are bought in multiple copies.
  - Old books and magazines are removed when they are out of date or in poor condition.
  - The librarian is an employee of the library, who interacts with the customers and whose work is supported by the system.
  - A borrower can reserve a book or magazine that is not currently available in the library, so that when it's returned or purchased by the library, that person is notified.
  - The reservation is canceled
    - when the borrower checks out the book or magazine, or
    - through a explicit canceling procedure.
  - The library can easily create, update, and delete information about the titles, borrowers, loans, and reservations in the system.

Activity 1003. Define Requirements

- User Interviews

Index	Question	Answer
1	Direct Interface with Borrower?	No, indirect
2	Can borrower search books on-line?	No, next version
3	Charge a fee for late return?	Yes, it just calculates the fee, and no direct interface with accounting software.
4	Charge a fee for lost books?	Yes, it just calculates the fee.
5	How to handle unregistered borrower?	First register and then lend items.
6	Is a notification available?	Yes, it can be printed on cards.
7	Calculate total number of titles checked out?	Yes
8	Specify max number of loans per borrower?	Yes
9	Specify max number of days for loans?	Yes
10	Send a kindly-reminder(SMS/Email) for return due?	No
11	Classify adult boos?	Yes
12	Specify qualification for valid borrower?	No
13	Maintain reliable database?	Yes
14	Can control any system access?	Yes, through login and logout.

Activity 1003. Define Requirements

- Functional Requirements (Version 1.0)
  - A library lends books and magazines to borrowers, who are registered in the system.
  - If the person has not been registered, the system first register the person. Then, lend titles.
  - A library handles the purchase of new titles. Popular titles are bought in multiple copies.
  - Old books and magazines are removed when they are out of date or in poor condition.
  - The librarian is an employee of the library who interacts with the customers(borrowers) and whose work is supported by the system.
  - A borrower can reserve a book or magazine that is not currently available in the library, so that when its returned or purchased by the library, that person is notified.
  - The system automatically prints 'post-cards' to notify the availability of the books. Then, the librarians mail them at the post office.

Activity 1003. Define Requirements

- Functional Requirements (Version 1.0)
  - For unregistered person, the system first register the person. Then, make reservations
  - The reservation is canceled when the borrower checks out the book or magazine or through a explicit canceling procedure.
  - The library can easily create, update, and delete information about the titles, borrowers, loans, and reservations in the system.
  - Upon request, the system calculates the total # of items checked out.
  - For any over-due items, a late-return fee is calculated and charged.
  - For any items lost, a replacement-fee is computed and charged.
  - The system validates the system access through librarian IDs and passwords.
  - For each title, the librarians specify the maximum number of days that can be held by the borrowers.

Activity 1003. Define Requirements

- Functional Requirements (Categorized Table)

Ref. #	Function	Category
R1.1	Make reservation	Evident
R1.2	Remove reservation	Evident
R1.3	Lend Item	Evident
R1.4.1	Return title	Evident
R1.4.2	Calculate Late-Return-Fee	Hidden
R1.5	Calculate Replacement Fee	Evident
R1.6	Notify Availability	Hidden
R2.1	Add title	Evident
R2.2	Remove title	Evident
R2.3	Update title	Evident
R2.4	Add items	Evident
R2.5	Remove item	Evident
R2.6	Update item	Evident
R3.1	Add borrower	Evident
R3.2	Remove borrower	Evident
R3.3	Update borrower	Evident
R4.1	Validates system access	Evident
R5.1	Compute total # of items checked out	Evident

Activity 1003. Define Requirements

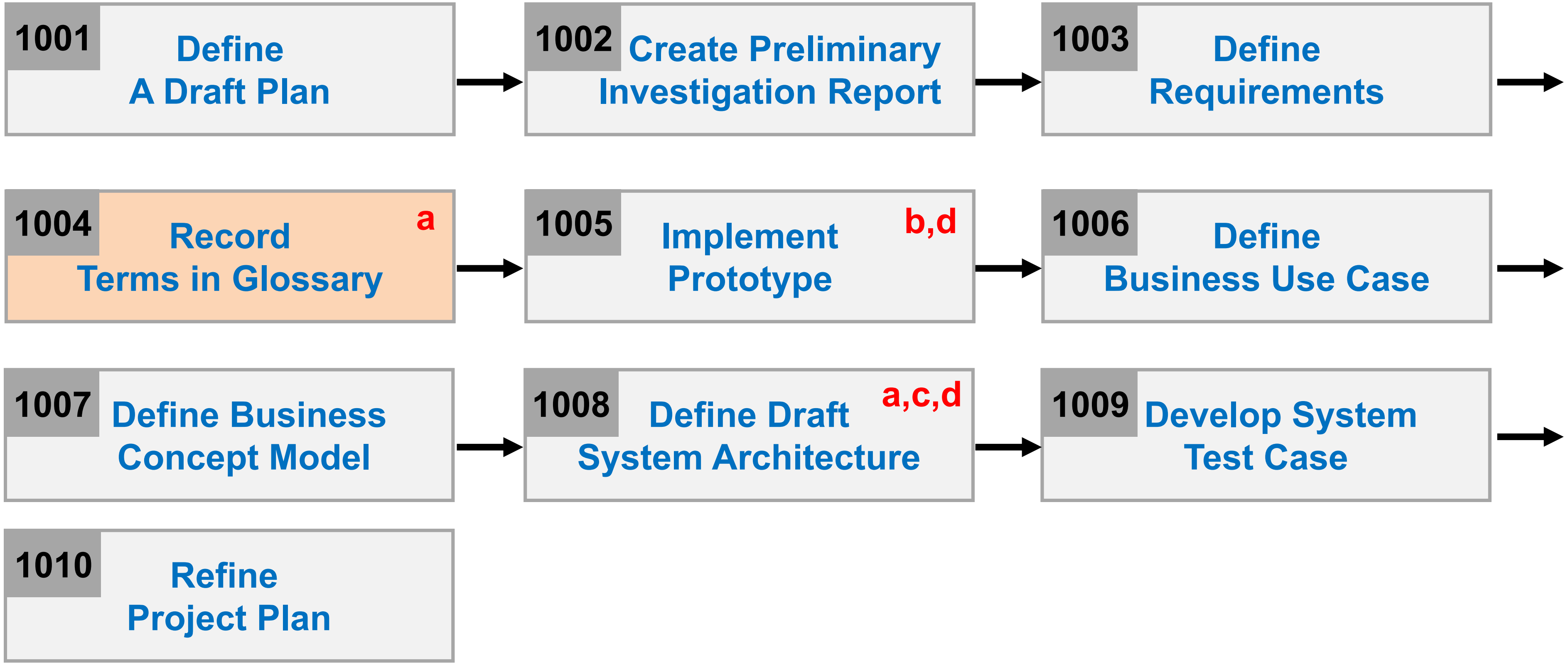
- Performance Requirements
  - The average response time for front desk operations should be less than 5 seconds.
  - The post-card to notify availability must be printed out immediately after the reserved book becomes available.

- Operating Environment
  - Microsoft Windows 7 and 10

- Interface Requirements
  - The current version may incorporate a menu-driven approach.
  - Next version incorporates windows metaphor.

- Other Requirements
  - The system must control the system access.

Activity 1004. Record Terms in Glossary



1001 Define A Draft Plan

1002 Create Preliminary Investigation Report

1003 Define Requirements

1004 Record Terms in Glossary a

1005 Implement Prototype b,d

1006 Define Business Use Case

1007 Define Business Concept Model

1008 Define Draft System Architecture a,c,d

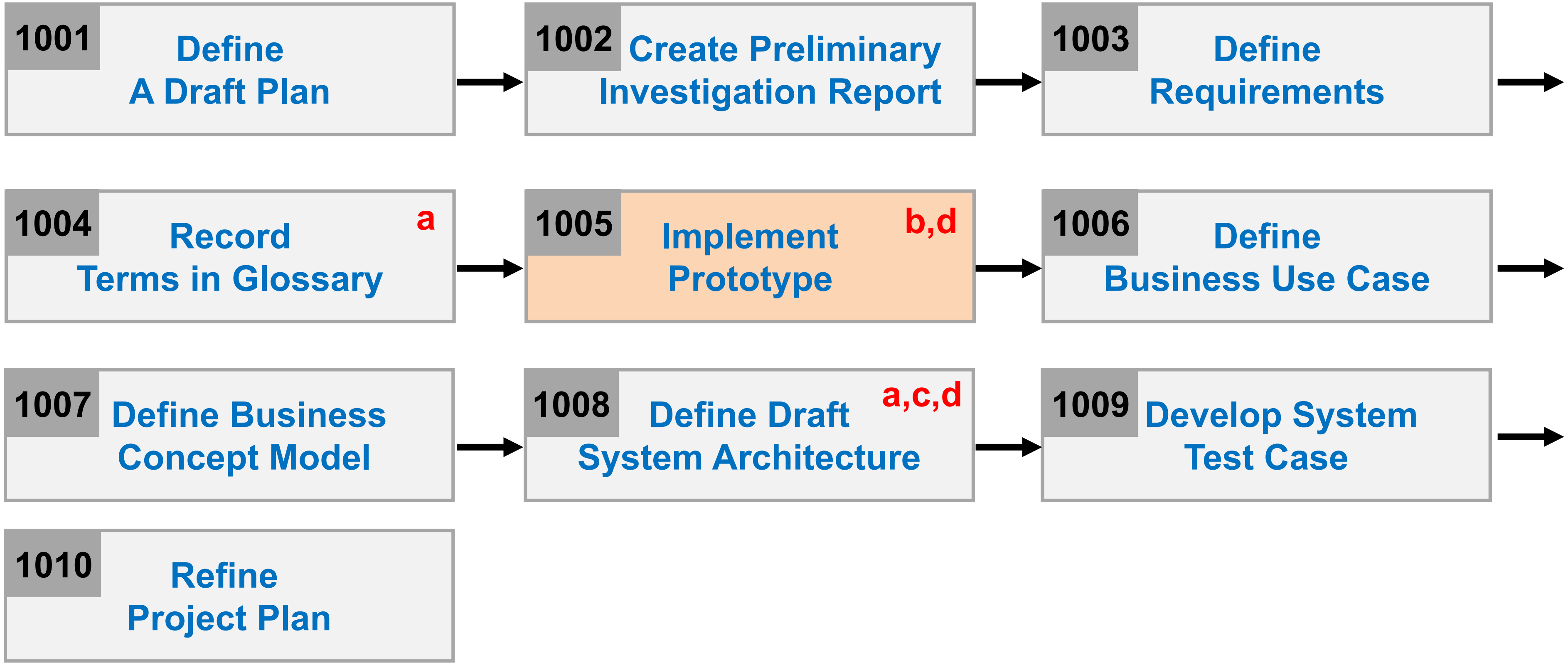
1009 Develop System Test Case

1010 Refine Project Plan

Activity 1004. Record Terms in Glossary

Term	Description	Remarks
Title	Books or Magazines, which are registered in the library system	
Item	Each copy of books or magazines	
Loan	An action of checking out an item from the library	
Librarian	An employee of the library who handles the requests of borrowers.	
…	…	

Activity 1005. Implement Prototype



1001 Define A Draft Plan

1002 Create Preliminary Investigation Report

1003 Define Requirements

1004 Record Terms in Glossary a

1005 Implement Prototype b,d

1006 Define Business Use Case

1007 Define Business Concept Model

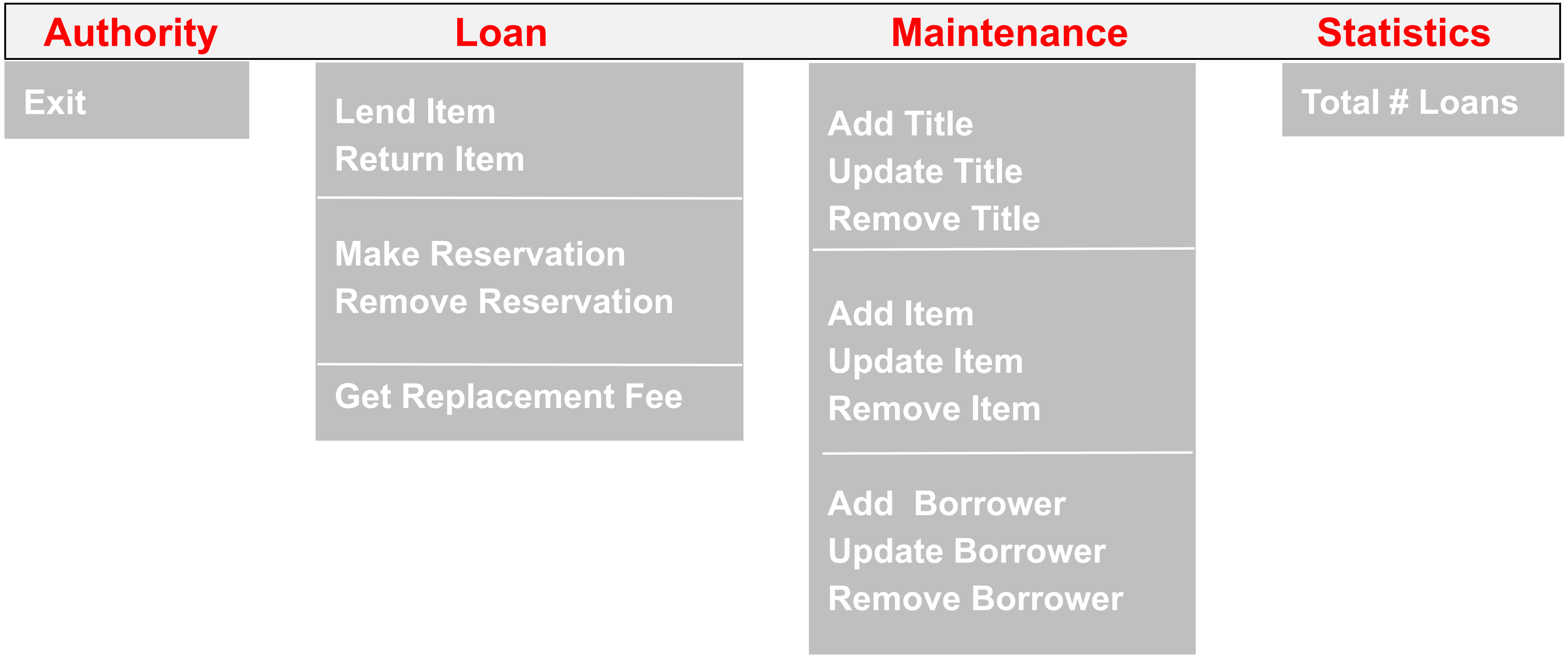
1008 Define Draft System Architecture a,c,d

1009 Develop System Test Case

1010 Refine Project Plan

Activity 1005. Implement Prototype

- User-Interface is sufficient for this LMS project



Authority

Exit

Loan

Lend Item
Return Item

Make Reservation
Remove Reservation

Get Replacement Fee

Maintenance

Add Title
Update Title
Remove Title

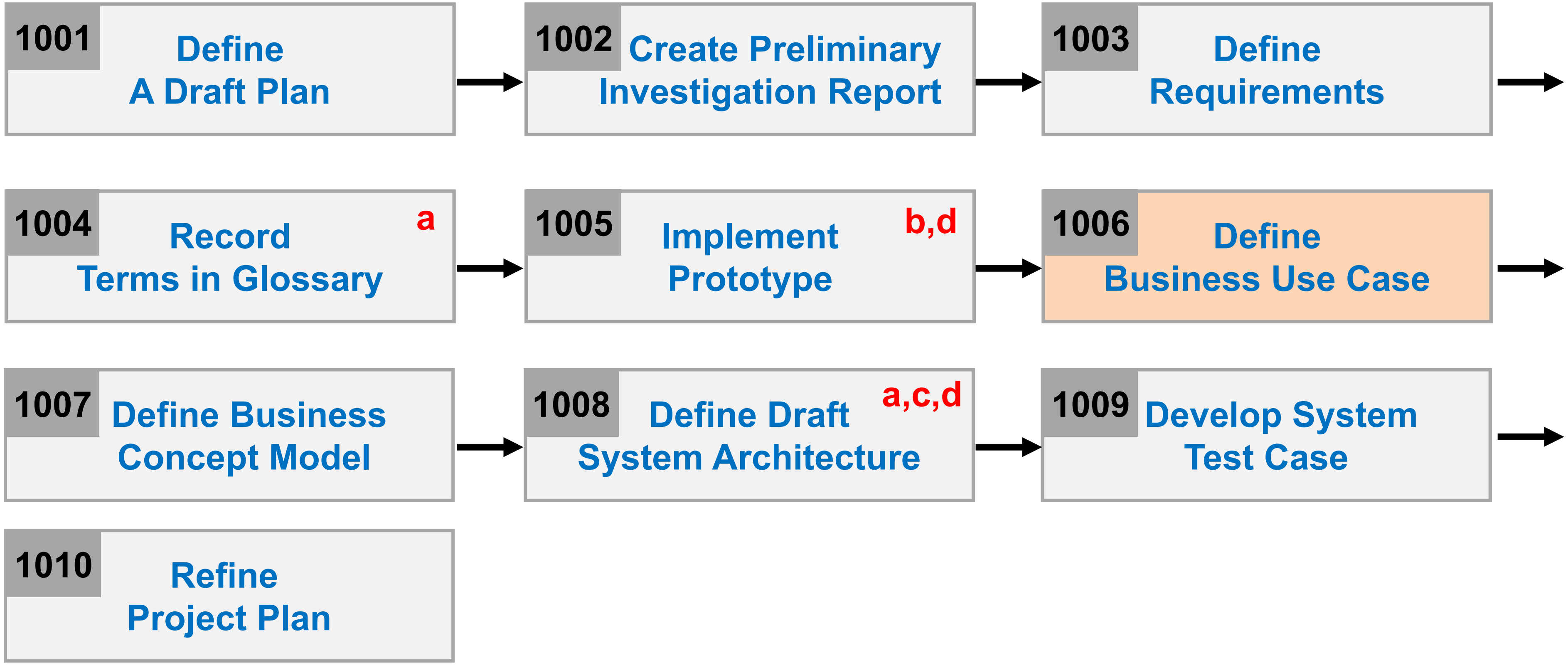
Add Item
Update Item
Remove Item

Add Borrower
Update Borrower
Remove Borrower

Statistics

Total # Loans

Activity 1006. Define Business Use Case


1001 Define A Draft Plan

1002 Create Preliminary Investigation Report

1003 Define Requirements

1004 Record Terms in Glossary a

1005 Implement Prototype b,d

1006 Define Business Use Case

1007 Define Business Concept Model

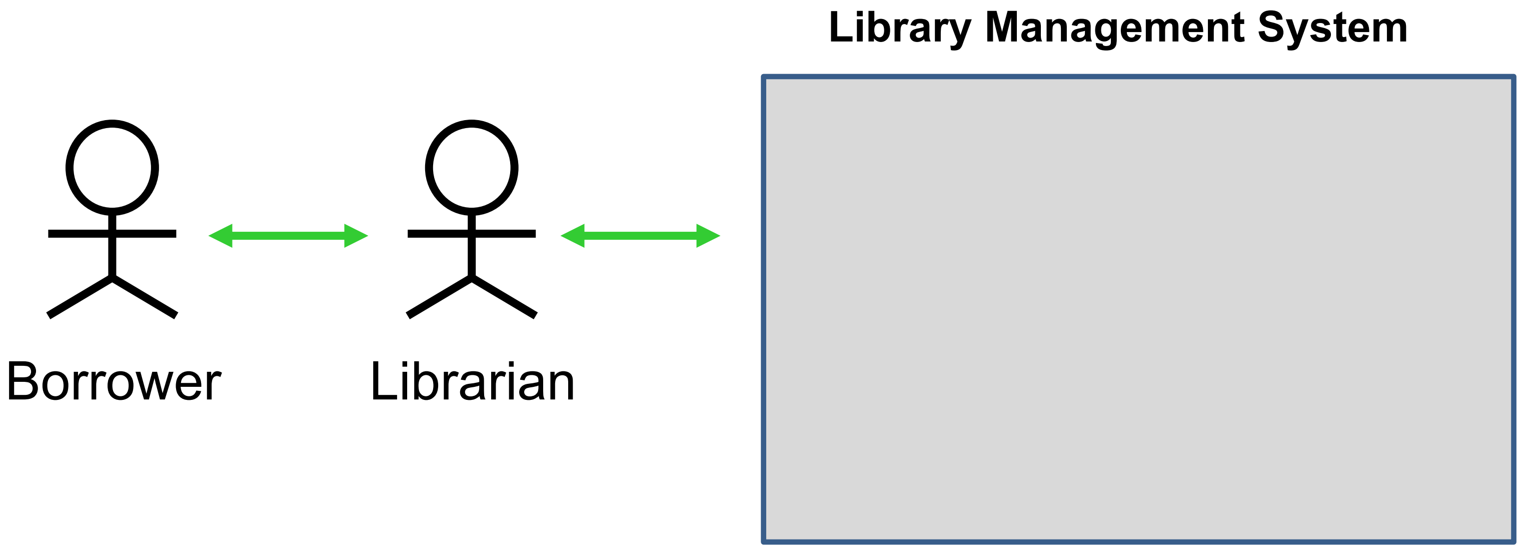
1008 Define Draft System Architecture a,c,d

1009 Develop System Test Case

1010 Refine Project Plan

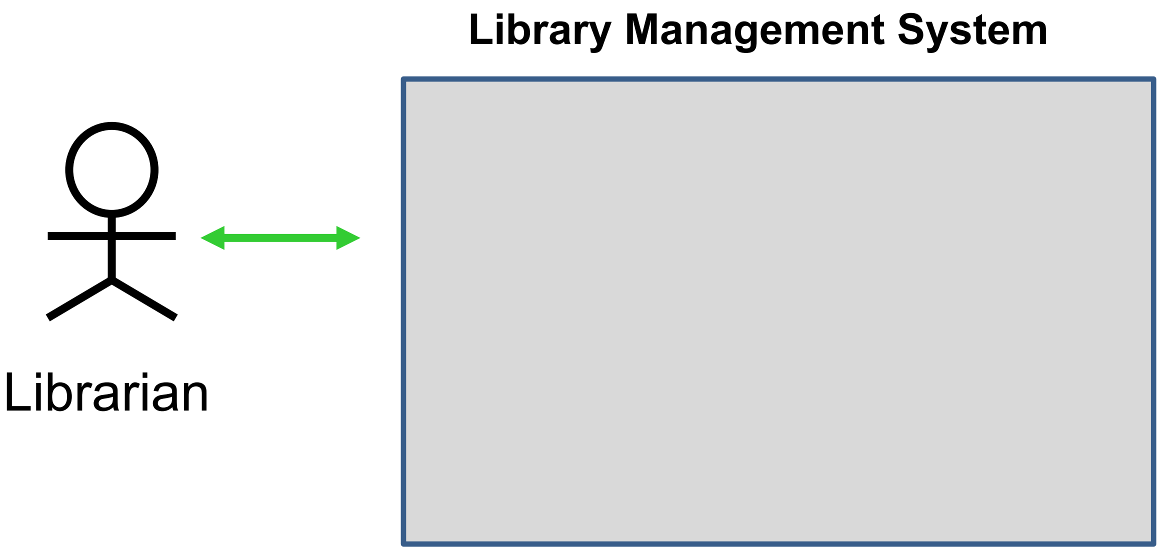
Activity 1006. Define Business Use Case

- Step 1. Define system boundary
  - All the functions defined earlier are inside the system boundary.



Activity 1006. Define Business Use Case

- Step 2. Identify the actors related to a system/organization
  - **Librarian** : an employee of the library who interacts with the customers(borrowers) and whose work is supported by the system.



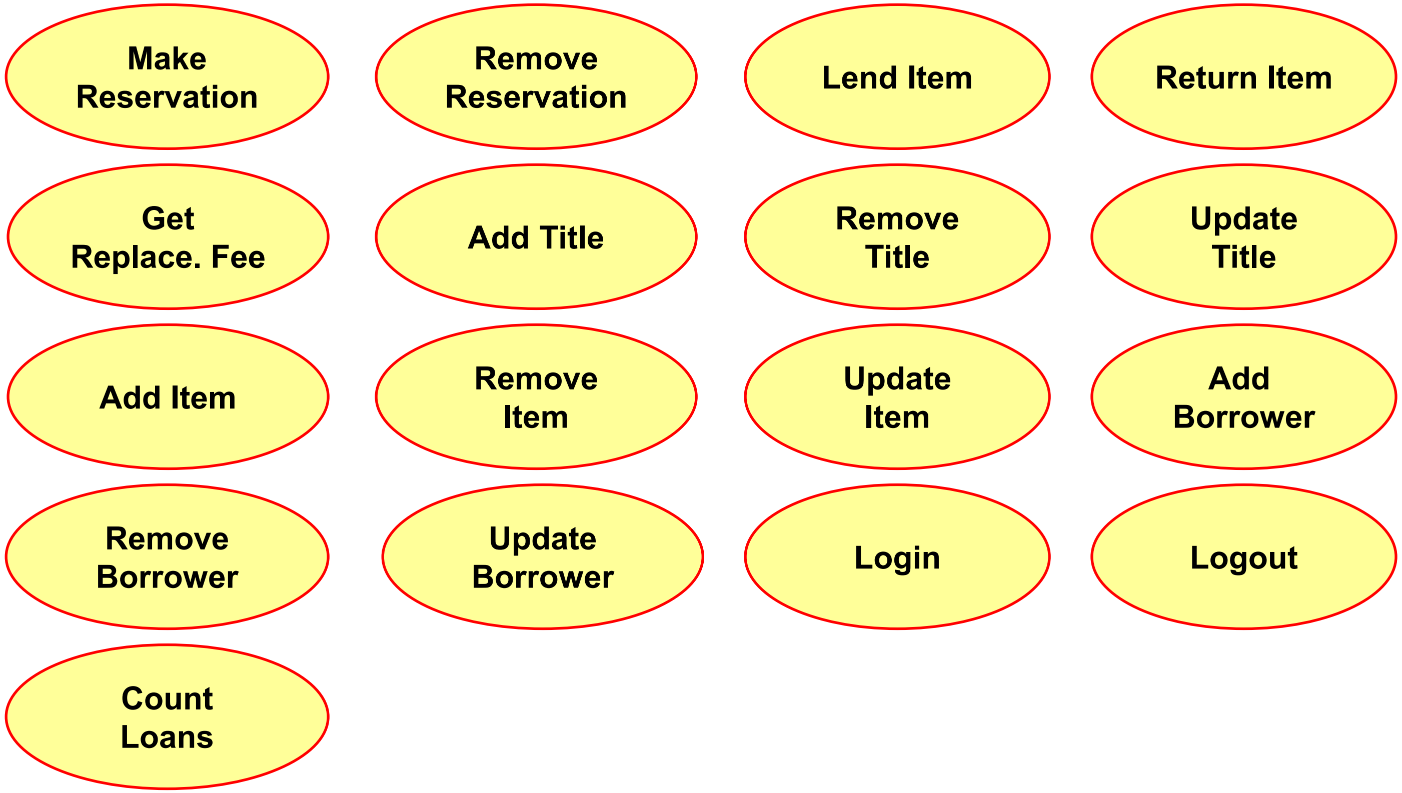
Activity 1006. Define Business Use Case

- Step 3. Identify user goals for each actor
- Step 4. Record the primary actors and their goals in an actor-goal list

Actor	Goal
Librarian	Make reservation Remove reservation Lend Item Return title Calculate Late-Return-Fee Calculate Replacement Fee Notify Availability Add title Remove title Update title Add items Remove item Update item Add borrower Remove borrower Update borrower Validates system access Compute total # of items

Activity 1006. Define Business Use Case

- Step. 5 Define use cases that satisfy user goals
  - Actor-based use cases



Make Reservation

Remove Reservation

Lend Item

Return Item

Get Replace. Fee

Add Title

Remove Title

Update Title

Add Item

Remove Item

Update Item

Add Borrower

Remove Borrower

Update Borrower

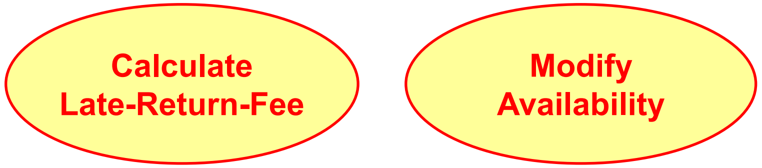
Login

Logout

Count Loans

Activity 1006. Define Business Use Case

- Step. 5 Define use cases that satisfy user goals
  - Event-based use cases



Activity 1006. Define Business Use Case

- Step 6. Allocate system functions into related use cases

Ref. #	Function	Use Case Number & Name
R1.1	Make reservation	1. Make Reservation
R1.2	Remove reservation	2. Remove Reservation
R1.3	Lend Item	3. Lend Item
R1.4.1	Return title	4. Return Title
R1.4.2	Calculate Late-Return-Fee	5. Calculate Late-Return-Fee
R1.5	Calculate Replacement Fee	6. Get Replacement Fee
R1.6	Notify Availability	7. Notify Availability
R2.1	Add title	8. Add Title
R2.2	Remove title	9. Remove Title
R2.3	Update title	10. Update Title
R2.4	Add items	11. Add Item
R2.5	Remove item	12. Remove Item
R2.6	Update item	13. Update Item
R3.1	Add borrower	14. Add Borrower
R3.2	Remove borrower	15. Remove Borrower
R3.3	Update borrower	16. Update Borrower
R4.1	Validates system access	17. Log-IN
R4.2	Validates system access	18. Log-Out
R5.1	Compute total # of items checked out	19. Count Loans

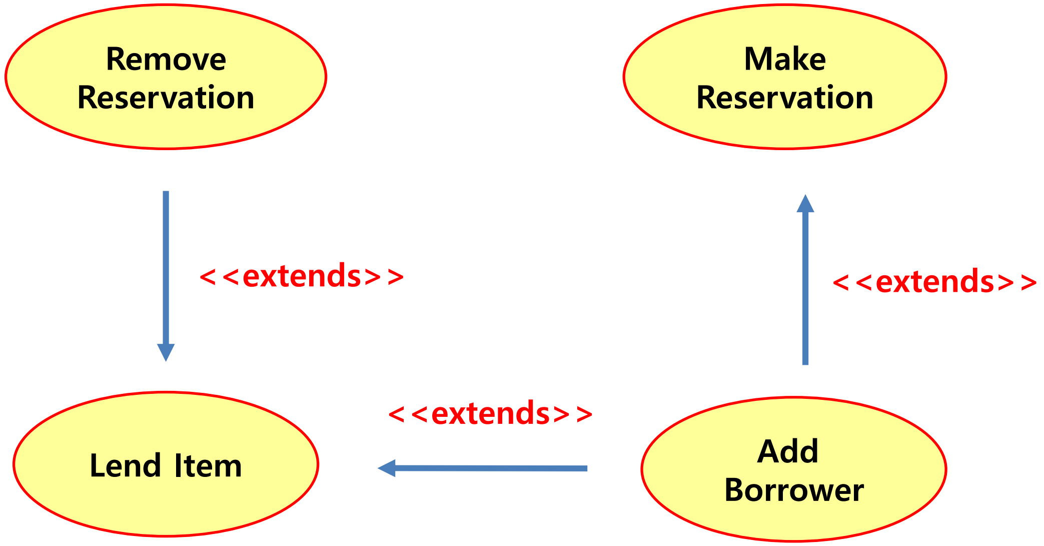
Activity 1006. Define Business Use Case

- Step 7. Categorize use cases

Ref. #	Function	Use Case Number & Name	Category	Category
R1.1	Make reservation	1. Make Reservation	Primary	**Evident**
R1.2	Remove reservation	2. Remove Reservation	Primary	**Evident**
R1.3	Lend Item	3. Lend Item	Primary	**Evident**
R1.4.1	Return title	4. Return Title	Primary	**Evident**
R1.4.2	Calculate Late-Return-Fee	5. Calculate Late-Return-Fee	Primary	**Hidden**
R1.5	Calculate Replacement Fee	6. Get Replacement Fee	Primary	**Evident**
R1.6	Notify Availability	7. Notify Availability	Primary	**Hidden**
R2.1	Add title	8. Add Title	Primary	**Evident**
R2.2	Remove title	9. Remove Title	Primary	**Evident**
R2.3	Update title	10. Update Title	Primary	**Evident**
R2.4	Add items	11. Add Item	Primary	**Evident**
R2.5	Remove item	12. Remove Item	Primary	**Evident**
R2.6	Update item	13. Update Item	Primary	**Evident**
R3.1	Add borrower	14. Add Borrower	Primary	**Evident**
R3.2	Remove borrower	15. Remove Borrower	Primary	**Evident**
R3.3	Update borrower	16. Update Borrower	Primary	**Evident**
R4.1	Validates system access	17. Log-IN	Secondary	**Evident**
R4.2	Validates system access	18. Log-Out	Secondary	**Evident**
R5.1	Compute total # of items checked out	19. Count Loans	Secondary	**Evident**

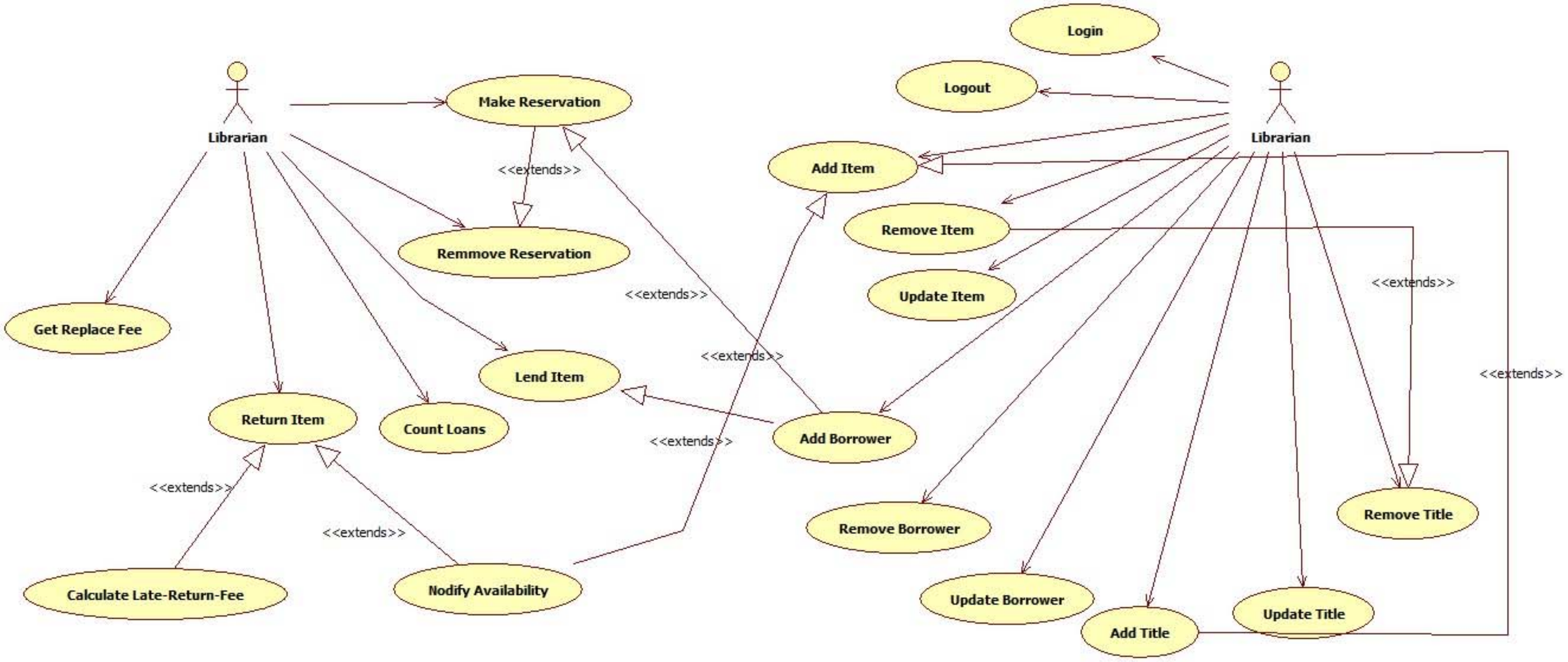
Activity 1006. Define Business Use Case

- Step 8. Identify relationships between use cases (Optional)

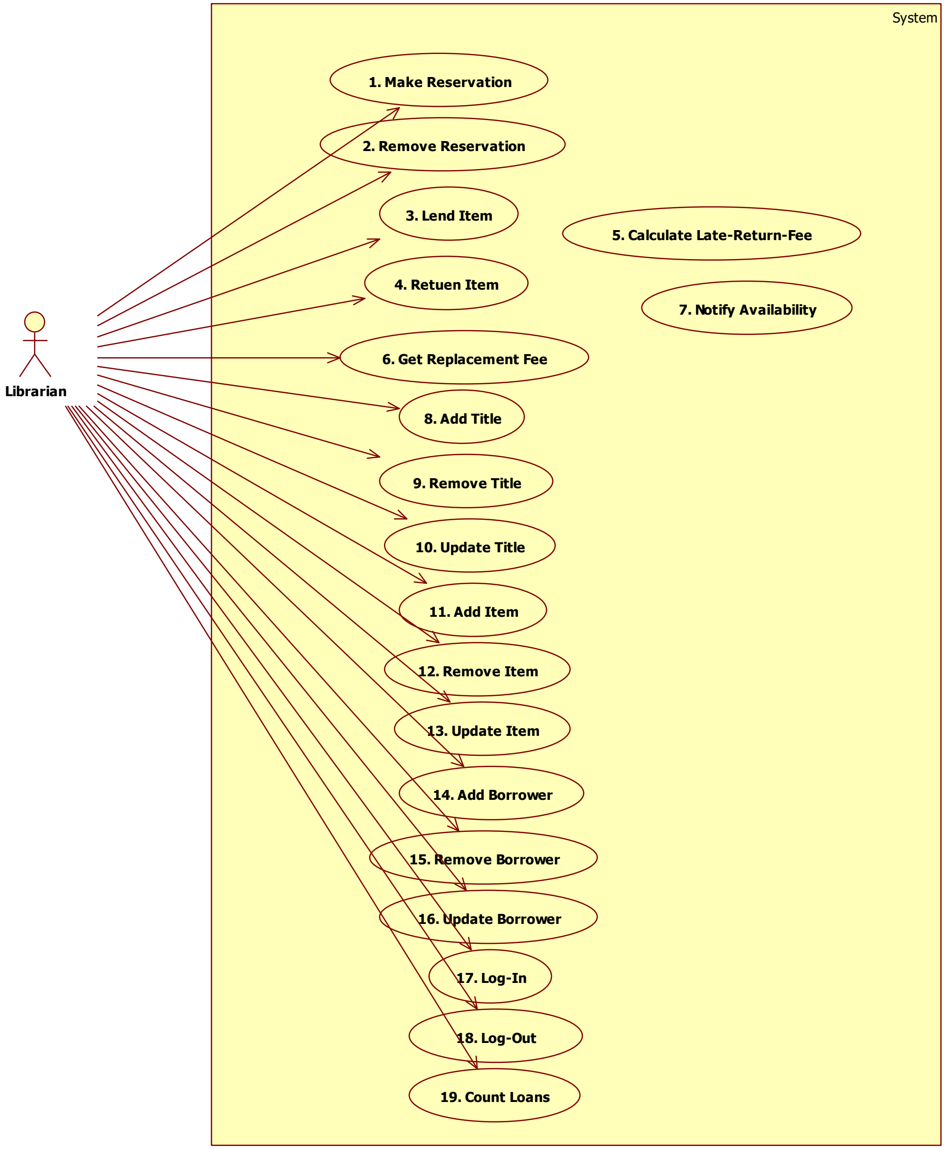


Activity 1006. Define Business Use Case

- Step 9. Draw a use case diagram
  - Defining system boundary (context) is referable.



Activity 1006. Define Business Use Case



Activity 1006. Define Business Use Case

- Step 10. Describe use cases

Use Case	1. Make Reservation
Actors	Librarian
Description	- This use case begins when a borrower arrives at the counter and then requests reservation. - For a registered borrower, it makes a reservation slip (software-wise). - For an unregistered borrower, the librarian registers the person and makes a reservation for the person.

Use Case	2. Remove Reservation
Actors	Librarian
Description	- A borrower who made a reservation can cancel his/her reservation. • Explicitly cancels the reservation. (Evident) - When a borrower checks out an item which he/she previously reserved, this use case is invoked automatically. • Hidden system function

Activity 1006. Define Business Use Case

Use Case	3. Lend Item
Actors	Librarian
Description	- This use case begins when the borrower arrives at the front desk with items to lend. - If a borrower does not registered, register first his/her information in the system. - This use case records the date, borrower ID, item ID and other relevant information for this loan.

Use Case	4. Return Item
Actors	Librarian
Description	- This use case begins when a borrower returns items at the counter. - If the item is returned past due date, a late-return-fee is computed, so that the borrower should pay the penalty.

Activity 1006. Define Business Use Case

Use Case	5. Calculate Late-Return-Fee
Actors	None
Description	- This use case computes the penalty amount for items returned late. - It first computes the number of extra days held by the borrower, then multiplies it by a pre-determined daily rate for late returns.

Use Case	6. Get Replacement Fee
Actors	Librarian
Description	- This use case computes the cost for replacing the lost book. - It first finds out the current price of the lost book, and add the handling cost to the book price.

Use Case	7. Notify Availability
Actors	None
Description	- This use case prints the book title that just became available, number of days held by the library, the name and address of the person who reserved on a post-card. - The actual mailing will be done manually by the librarian.

Activity 1006. Define Business Use Case

Use Case	8. Add Title
Actors	Librarian
Description	- Whenever a new kind of book is purchased, the book information is recorded into the system. - Then, it invokes 'Add Item' use case to record the number of copies purchased.

Use Case	9. Remove Title
Actors	Librarian
Description	- Some old books are selected for removal by the librarians. - This use case deletes the information of the book to be removed. - And, it will be no longer available for loans.

Use Case	10. Update Title
Actors	Librarian
Description	- This use case will change the recorded information of the title. - What actual kinds of information?

Activity 1006. Define Business Use Case

Use Case	11. Add Item
Actors	Librarian
Description	- When additional copies (of the currently available title) are purchases, this updates the total number of copies for each title. • Date, Price, Bookstore, Available, etc. - When a reservation has been made for this title, this use case invokes ‘notify availability’ use case.

Use Case	12. Remove Item
Actors	Librarian
Description	- This use case will update the number of items for each title. - If no more item is remaining after removal, this use case will invoke ‘Remove Title’ use case.

Use Case	13. Update Item
Actors	Librarian
Description	- This use case updates the information of the items. - What actual kinds of information will be updated ?

Activity 1006. Define Business Use Case

Use Case	14. Add Borrower
Actors	Librarian
Description	- This use case will record the information of the new borrower such as name, address, phone, loan priority, etc.

Use Case	15. Remove Borrower
Actors	Librarian
Description	- This use case deletes the information of borrower from the system, so that the person can no longer check out titles. - This may happen if the borrower has a bad return history or has not been using the library longer than 2 years.

Use Case	16. Update Borrower
Actors	Librarian
Description	- This use case updates the information of the borrower such as new address and phone.

Activity 1006. Define Business Use Case

Use Case	17. Log-In
Actors	Librarian
Description	- This use case reads the user ID and password of the librarian, and verifies. - If an invalid information is entered, it will re-prompt and read the ID and password. - After 3 successive failures of login, it records this ‘attach’ information and automatically returns to the initial menu.

Use Case	18. Log-Out
Actors	Librarian
Description	- This use case records the date and time of the current logout, and returns to the initial menu.

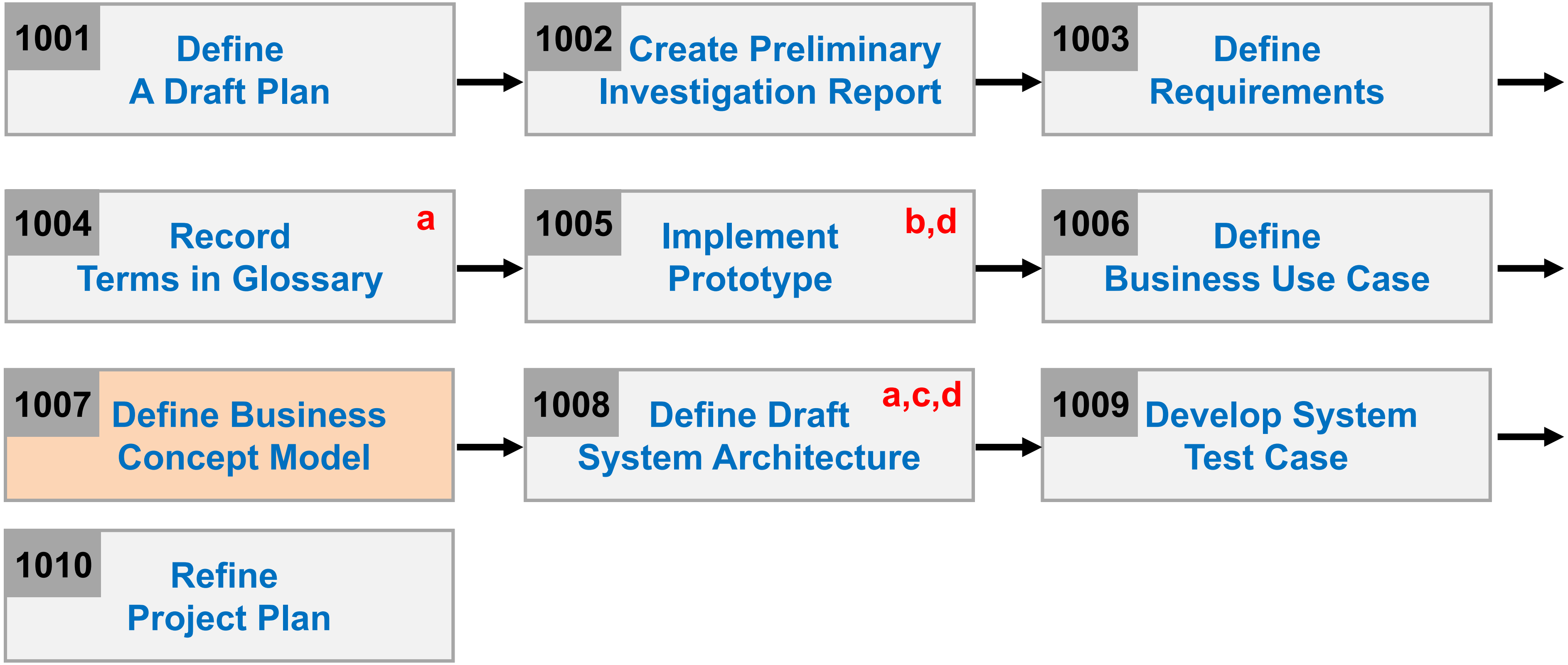
Use Case	19. Count Loans
Actors	Librarian
Description	- This use cases computes the total number of items checked out.

Activity 1006. Define Business Use Case

- Step 11. Rank use cases

Ref. #	Function	Use Case Number & Name	Category	Rank	Category
R1.1	Make reservation	1. Make Reservation	Primary	High	Evident
R1.2	Remove reservation	2. Remove Reservation	Primary	High	Evident
R1.3	Lend Item	3. Lend Item	Primary	High	Evident
R1.4.1	Return title	4. Return Title	Primary	High	Evident
R1.4.2	Calculate Late-Return-Fee	5. Calculate Late-Return-Fee	Primary	High	Hidden
R1.5	Calculate Replacement Fee	6. Get Replacement Fee	Primary	High	Evident
R1.6	Notify Availability	7. Notify Availability	Primary	High	Hidden
R2.1	Add title	8. Add Title	Primary	High	Evident
R2.2	Remove title	9. Remove Title	Primary	High	Evident
R2.3	Update title	10. Update Title	Primary	High	Evident
R2.4	Add items	11. Add Item	Primary	High	Evident
R2.5	Remove item	12. Remove Item	Primary	High	Evident
R2.6	Update item	13. Update Item	Primary	High	Evident
R3.1	Add borrower	14. Add Borrower	Primary	High	Evident
R3.2	Remove borrower	15. Remove Borrower	Primary	High	Evident
R3.3	Update borrower	16. Update Borrower	Primary	High	Evident
R4.1	Validates system access	17. Log-IN	Secondary	Medium	Evident
R4.2	Validates system access	18. Log-Out	Secondary	Medium	Evident
R5.1	Compute total # of items checked out	19. Count Loans	Secondary	Medium	Evident

Activity 1007.
Define Business Concept Model


1001 Define A Draft Plan
1002 Create Preliminary Investigation Report
1003 Define Requirements
1004 Record Terms in Glossary
a
1005 Implement Prototype
b,d
1006 Define Business Use Case
1007 Define Business Concept Model
1008 Define Draft System Architecture
a,c,d
1009 Develop System Test Case
1010 Refine Project Plan

Activity 1007.
Define Business Concept Model

- Identify 'Concepts' in the target domain.

Title

Book

Magazine

Item

Reservation

Borrower

Loan

Librarian

Customer

Library

Return

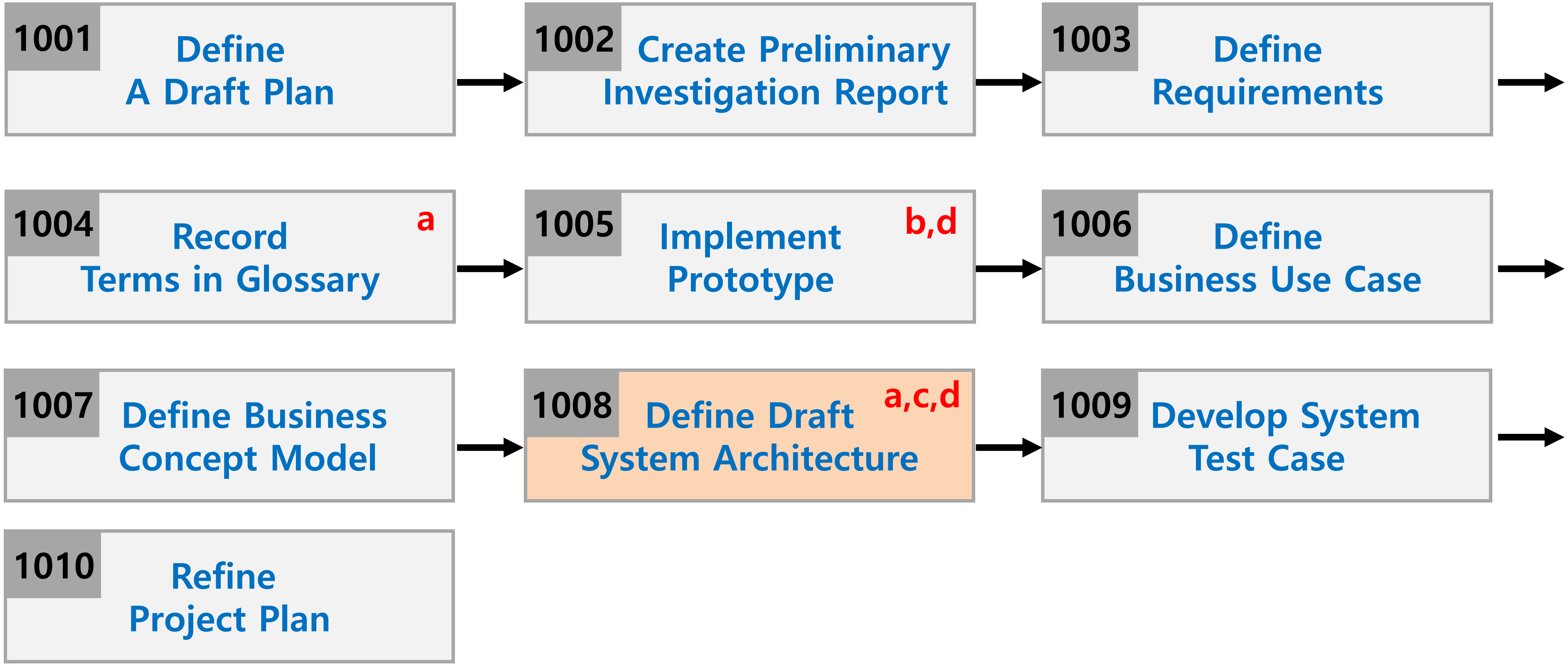
Registration

Notification

Late-Return-Fee

Check-Out

Activity 1008.
Define Draft System Architecture



1001 Define A Draft Plan

1002 Create Preliminary Investigation Report

1003 Define Requirements

1004 Record Terms in Glossary a

1005 Implement Prototype b,d

1006 Define Business Use Case

1007 Define Business Concept Model

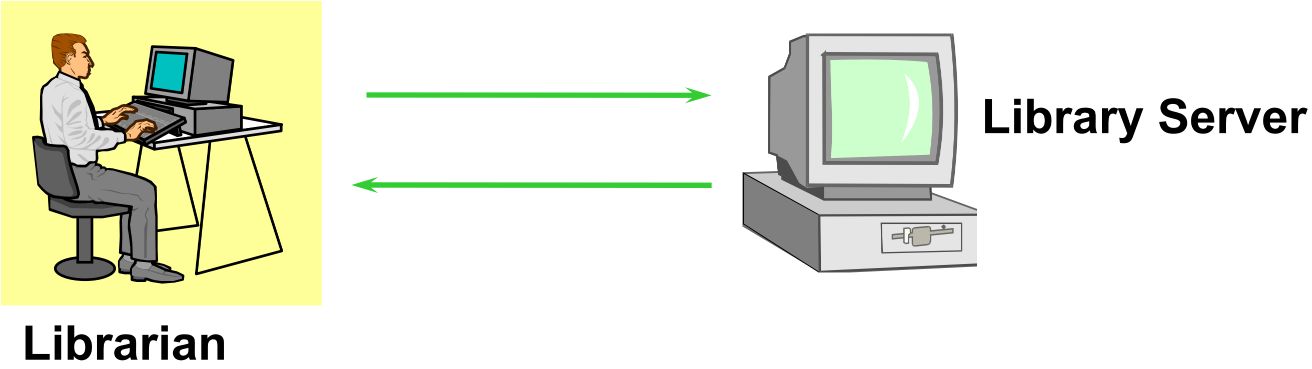
1008 Define Draft System Architecture a,c,d

1009 Develop System Test Case

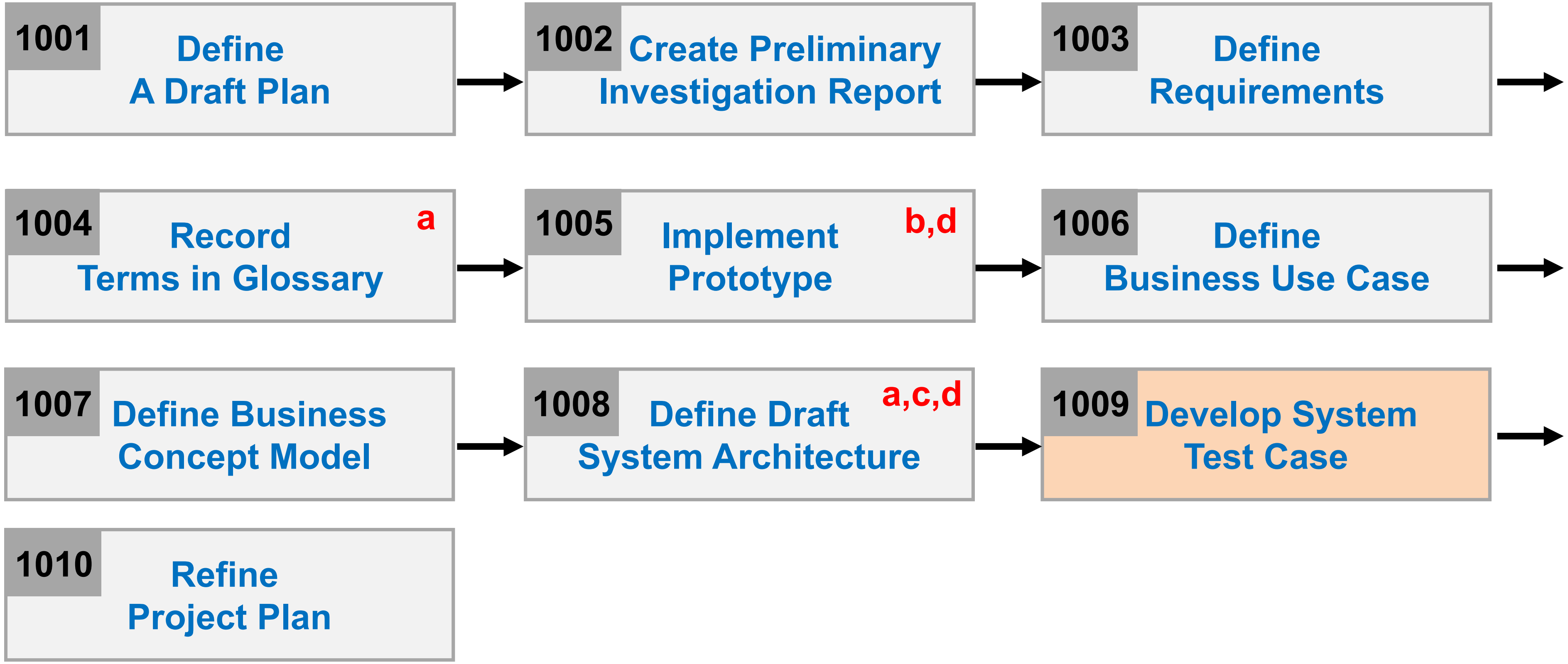
1010 Refine Project Plan

Activity 1008.
Define Draft System Architecture

- Define system architecture



Activity 1009. Develop System Test Case



1001 Define A Draft Plan

1002 Create Preliminary Investigation Report

1003 Define Requirements

1004 Record Terms in Glossary a

1005 Implement Prototype b,d

1006 Define Business Use Case

1007 Define Business Concept Model

1008 Define Draft System Architecture a,c,d

1009 Develop System Test Case

1010 Refine Project Plan

Activity 1009. Develop System Test Case

- Step 1. Identify important requirements

Ref. #	Function	Category
R1.1	**Make reservation**	Evident
R1.2	**Remove reservation**	Evident
R1.3	**Lend Item**	Evident
R1.4.1	**Return title**	Evident
R1.4.2	Calculate Late-Return-Fee	Hidden
R1.5	**Calculate Replacement Fee**	Evident
R1.6	Notify Availability	Hidden
R2.1	**Add title**	Evident
R2.2	**Remove title**	Evident
R2.3	**Update title**	Evident
R2.4	**Add items**	Evident
R2.5	**Remove item**	Evident
R2.6	**Update item**	Evident
R3.1	**Add borrower**	Evident
R3.2	**Remove borrower**	Evident
R3.3	**Update borrower**	Evident
R4.1	**Validates system access**	Evident
R5.1	**Compute total # of items checked out**	Evident

Activity 1009. Develop System Test Case

- Step 2. Develop system test cases with various system testing techniques
  - First, brute force testing

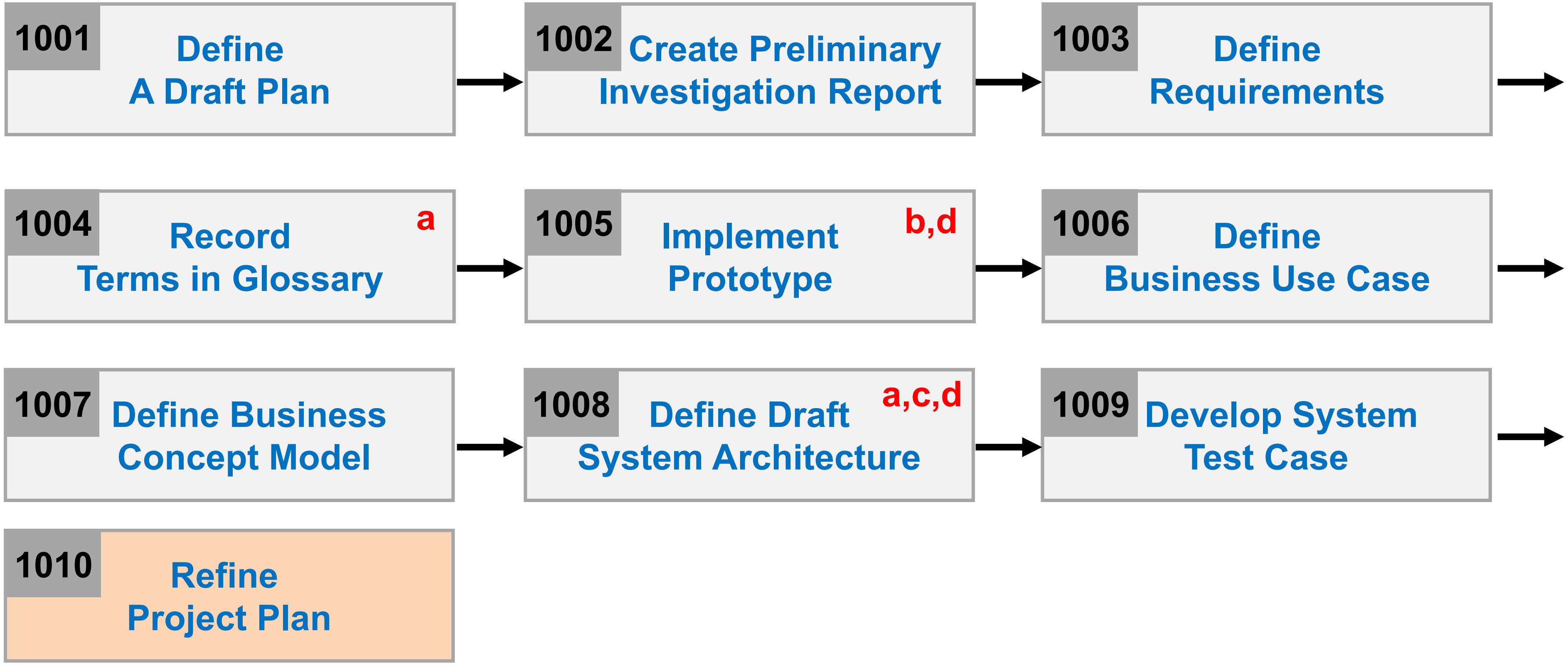
No.	Tests	Description
1	**Make reservation**	Correct한 borrower가 correct한 title 예약
2	**Make reservation**	Correct한 borrower가 incorrect한 title 예약
3	**Make reservation**	Correct한 borrower가 대여중인 title 예약
4	**Make reservation**	Incorrect한 borrower가 예약
5	**Remove reservation**	Correct한 borrower가 예약 취소
6	**Remove reservation**	Incorrect한 borrower가 예약 취소
7	**Lend Item**	Correct한 borrower가 대여 가능한 title 대여
8	**Lend Item**	Correct한 borrower가 incorrect한 title 대여
9	**Lend Item**	Correct한 borrower가 모두 대여중인 title 대여
10	**Lend Item**	Incorrect한 borrower가 대여
11	**Return title**	Borrower가 title 반납
12	**Return title**	Borrower가 연체된 title 반납
13	**Add title**	새 title 추가
14	**Remove title**	기존의 title 제거
15	**Remove title**	존재하지 않는 title 제거
16	**Update title**	Title 정보 update
17	**Add item**	Title item 추가
18	**Add item**	존재하지 않는 title의 item추가

Activity 1009. Develop System Test Case

- Step 2. Develop system test cases with various system testing techniques
  - First, brute force testing

No.	Tests	Description
19	**Remove item**	Title의 item제거
20	**Remove item**	존재하지 않는 title의 item제거
21	**Update item**	올바른 item의 정보 update
22	**Update item**	Title에 존재하지 않는 item update
23	**Add borrower**	Borrower 추가
24	**Remove borrower**	Borrower 삭제
25	**Update borrower**	기존의 borrower update
26	**Update borrower**	삭제된 borrower update
27	**Validates system access**	Correct id/pw로 로그인
28	**Validates system access**	Incorrect id/pw로 로그인
29	**Validates system access**	로그아웃
30	**Compute total # of items checked out**	계산 시도

Activity 1010. Refine Project Plan



DEPENDABLE SOFTWARE LABORATORY

Activity 1010. Refine Project Plan

- Project Scope
  - The library management software automates typical library operations; reservation, lending item, adding, removing, and updating the information of title, item, and borrower.

- Project Objectives
  - To develop a computerized library management software, that provides typical library operations such as:
    - Lend and return books, Reserve books, Maintaining Borrow information, and Purchasing new books.
  - The new software should be easy to learn and use, and efficient.

Activity 1010. Refine Project Plan

- Functional Requirements

Ref. #	Function	Category
R1.1	Make reservation	Evident
R1.2	Remove reservation	Evident
R1.3	Lend Item	Evident
R1.4.1	Return title	Evident
R1.4.2	Calculate Late-Return-Fee	Hidden
R1.5	Calculate Replacement Fee	Evident
R1.6	Notify Availability	Hidden
R2.1	Add title	Evident
R2.2	Remove title	Evident
R2.3	Update title	Evident
R2.4	Add items	Evident
R2.5	Remove item	Evident
R2.6	Update item	Evident
R3.1	Add borrower	Evident
R3.2	Remove borrower	Evident
R3.3	Update borrower	Evident
R4.1	Validates system access	Evident
R5.1	Compute total # of items checked out	Evident

Activity 1010. Refine Project Plan

- Performance Requirements
  - When making reservations, the information of reservation will appear within 5 seconds.
  - When lending items, the content of lending item will appear within 5 seconds.
  - When returning items, the content of returning item will appear within 5 seconds.

- Operating Environment
  - Microsoft Windows 7 and 10

- User Interface Requirements
  - Menu-driven approach
  - Should be designed for upgrading to ‘Window-based’ version.

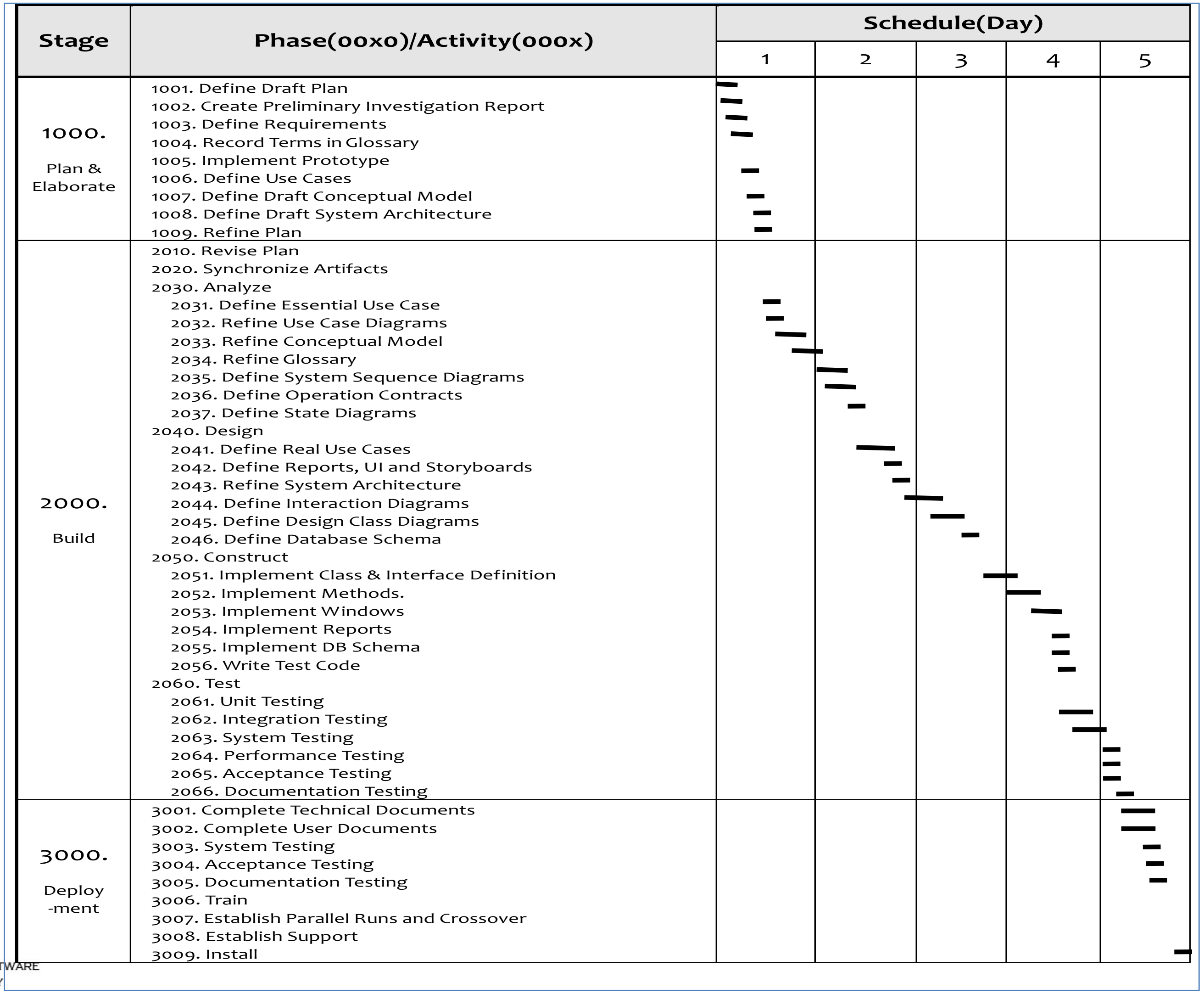
Activity 1010. Refine Project Plan

- Other Requirements
  - The content of database should be maintained reliably.
  - System should control the system access.

- Resources
  - Man Month : 6 Persons
    - A Team Leader
    - A Document Manager
    - 3-4 Engineers
  - Period : 5 Days (Around 40 Hours)
  - Hardware : skylake processor
  - Software
    - OS : Windows 7/10
    - Programming Language : Java
    - Case Tools : Rational Rose, Paradigm Plus

Activity 1010. Refine Project Plan

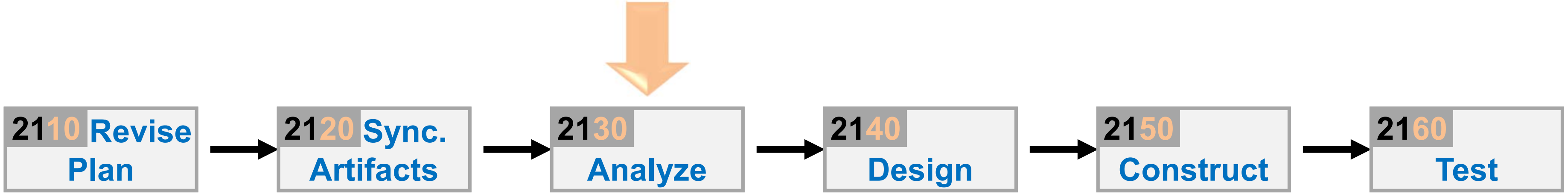
- Scheduling



Stage	Phase(00x0)/Activity(000x)	Schedule(Day)				
		1	2	3	4	5
1000. Plan & Elaborate	1001. Define Draft Plan					
	1002. Create Preliminary Investigation Report					
	1003. Define Requirements					
	1004. Record Terms in Glossary					
	1005. Implement Prototype					
	1006. Define Use Cases					
	1007. Define Draft Conceptual Model					
	1008. Define Draft System Architecture					
	1009. Refine Plan					
2000. Build	2010. Revise Plan					
	2020. Synchronize Artifacts					
	2030. Analyze					
	2031. Define Essential Use Case					
	2032. Refine Use Case Diagrams					
	2033. Refine Conceptual Model					
	2034. Refine Glossary					
	2035. Define System Sequence Diagrams					
	2036. Define Operation Contracts					
	2037. Define State Diagrams					
	2040. Design					
	2041. Define Real Use Cases					
	2042. Define Reports, UI and Storyboards					
	2043. Refine System Architecture					
	2044. Define Interaction Diagrams					
	2045. Define Design Class Diagrams					
	2046. Define Database Schema					
	2050. Construct					
	2051. Implement Class & Interface Definition					
	2052. Implement Methods.					
	2053. Implement Windows					
	2054. Implement Reports					
	2055. Implement DB Schema					
	2056. Write Test Code					
	2060. Test					
	2061. Unit Testing					
	2062. Integration Testing					
	2063. System Testing					
	2064. Performance Testing					
	2065. Acceptance Testing					
	2066. Documentation Testing					
3000. Deploy -ment	3001. Complete Technical Documents					
	3002. Complete User Documents					
	3003. System Testing					
	3004. Acceptance Testing					
	3005. Documentation Testing					
	3006. Train					
	3007. Establish Parallel Runs and Crossover					
	3008. Establish Support					
	3009. Install					

Back

Phase 2030. Analyze



2110 Revise Plan

2120 Sync. Artifacts

2130 Analyze

2140 Design

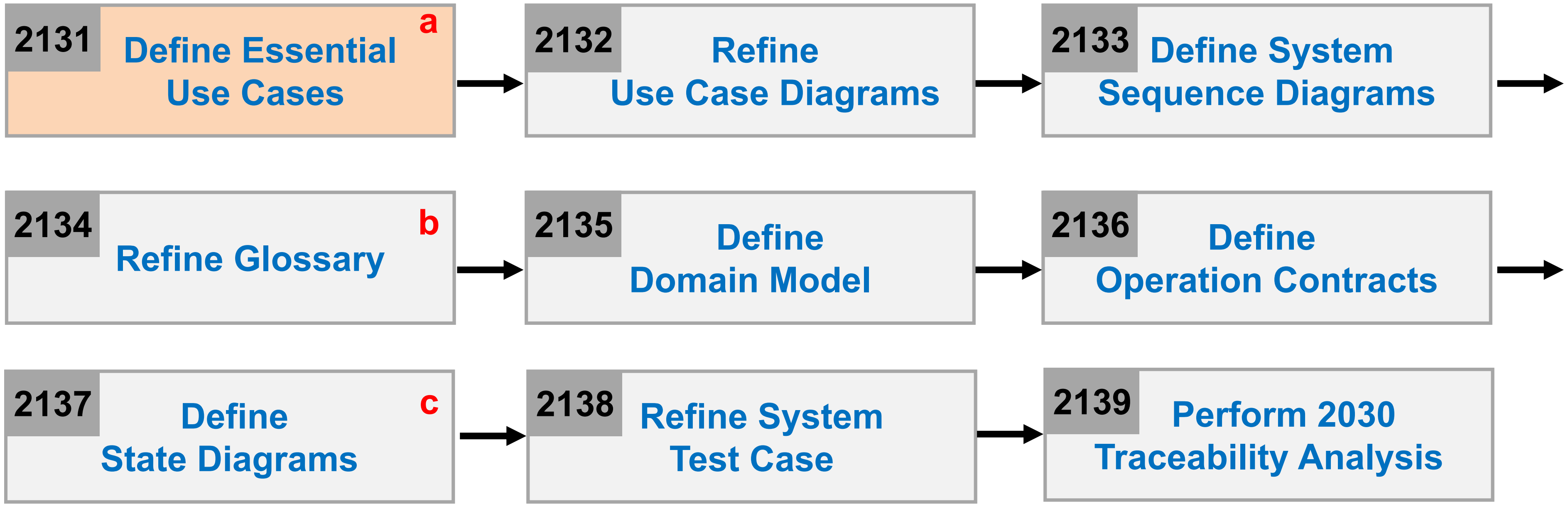
2150 Construct

2160 Test

Activity 2031. Define Essential Use Cases

- Phase 2030 Activities

a. if not yet done
b. ongoing
c. optional



2131 Define Essential Use Cases a

2132 Refine Use Case Diagrams

2133 Define System Sequence Diagrams

2134 Refine Glossary b

2135 Define Domain Model

2136 Define Operation Contracts

2137 Define State Diagrams c

2138 Refine System Test Case

2139 Perform 2030 Traceability Analysis

Activity 2031. Define Essential Use Cases

- 1. Make Reservation

Use Case	1. Make Reservation
Actor	Librarian
Purpose	(As in the business use case)
Overview	(As in the business use case)
Type	Primary and Essential
Cross Reference	System Functions: R1.1, R3.1 Use Case: “Add Borrower”
Pre-Requisites	Borrower should have an id_card.
Typical Courses of Events	(A) : Actor, (S) : System 1. (A) A librarian requests the reservation of title 2. (S) Check if a corresponding title exists 3. (S) Check if a corresponding borrower exists 4. (S) If the borrower does not exist, invoke “Add Borrower” 5. (S) Create reservation information
Alternative Courses of Events	N/A
Exceptional Courses of Events	Line 1: If invalid reservation information is entered, indicate an error.

Activity 2031. Define Essential Use Cases

- 2. Remove Reservation

Use Case	2. Remove Reservation
Actor	Librarian
Purpose	(As in the business use case)
Overview	(As in the business use case)
Type	Primary and Essential
Cross Reference	System Functions: R1.2, R1.3 Use Case: “Lend Item”
Pre-Requisites	N/A
Typical Courses of Events	(A) : Actor, (S) : System 1. (A) A librarian requests removing reservation of the title 2. (S) Check if a corresponding title exists 3. (S) Check if a corresponding borrower exists 4. (S) Find the reservation 5. (S) Remove the reservation
Alternative Courses of Events	N/A
Exceptional Courses of Events	Line 1: If invalid reservation information is entered, indicate an error.

Activity 2031. Define Essential Use Cases

- 3. Lend Item

Use Case	3. Lent Item
Actor	Librarian
Purpose	(As in the business use case)
Overview	(As in the business use case)
Type	Primary and Essential
Cross Reference	System Functions: R1.3, R1.2 Use Cases: “Remove Reservation”, “Add Borrower”
Pre-Requisites	Borrower should have id_card.
Typical Courses of Events	(A) : Actor, (S) : System 1. (A) A librarian requests lending item 2. (S) Check if a corresponding title exists 3. (S) Check if a corresponding item is available 4. (S) If the item was reserved, invoke “Remove Reservation” 5. (S) Check if corresponding borrower exists 6. (S) If the borrower does not exist, invoke “Add Borrower” 7. (S) Create new loan
Alternative Courses of Events	N/A
Exceptional Courses of Events	Line 1: If invalid lending information is entered, indicate an error.

Activity 2031. Define Essential Use Cases

- 4. Return Item

Use Case	4. Return Item
Actor	Librarian
Purpose	(As in the business use case)
Overview	(As in the business use case)
Type	Primary and Essential
Cross Reference	System Functions: R1.4.1, R1.4.2, R1.6 Use Cases: “Calculate Late-Return-Fee”, “Notify Availability”
Pre-Requisites	N/A
Typical Courses of Events	(A) : Actor, (S) : System 1. (A) A librarian requests returning item 2. (S) Check if a corresponding title exists 3. (S) Check if a corresponding borrower exists 4. (S) Check if a corresponding item is loaned 5. (S) Find the borrower of the item 6. (S) Check whether the returning due-date is over or not 7. (S) If the returning due-date is over, invoke “Calculate Late-Return-Fee” 8. (S) Remove the loan 9. (S) If the item is reserved, invoke “Notify Availability”
Alternative Courses of Events	N/A
Exceptional Courses of Events	Line 1: If invalid returning information is entered, indicate an error

Activity 2031. Define Essential Use Cases

- 5. Calculate Late-Return-Fee

Use Case	5. Calculate Late-Return-Fee
Actor	b
Purpose	(As in the business use case)
Overview	(As in the business use case)
Type	Primary and Essential
Cross Reference	System Functions: R1.4.1, R1.4.2 Use Case: “Return Item”
Pre-Requisites	Lending due-date should be over.
Typical Courses of Events	(A) : Actor, (S) : System 1. (S) Compute late-return time 2. (S) Compute late-return fee 3. (S) Print the late-return fee
Alternative Courses of Events	N/A
Exceptional Courses of Events	N/A

Activity 2031. Define Essential Use Cases

- 6. Get Replacement-Fee

Use Case	6. Get Replacement-Fee
Actor	Librarian
Purpose	(As in the business use case)
Overview	(As in the business use case)
Type	Primary and Essential
Cross Reference	System Functions: R1.5 Use Case: -
Pre-Requisites	Title should be lost.
Typical Courses of Events	(A) : Actor, (S) : System 1. (A) A librarian inputs a title’s information 2. (S) Check if a corresponding title exists 3. (S) Find the price of the title 4. (S) Compute replacement-fee
Alternative Courses of Events	N/A
Exceptional Courses of Events	N/A

Activity 2031. Define Essential Use Cases

- 7. Notify Availability

Use Case	7. Notify Availability
Actor	**None**
Purpose	(As in the business use case)
Overview	(As in the business use case)
Type	Primary and Essential
Cross Reference	System Functions: R1.4.1, R1.6, R2.4 Use Cases: “Return Item”, “Add Item”
Pre-Requisites	The title should be returned or new title should be added.
Typical Courses of Events	(A) : Actor, (S) : System 1. (S) Notify the availability of the item 2. (S) Print a post-card
Alternative Courses of Events	N/A
Exceptional Courses of Events	N/A

Activity 2031. Define Essential Use Cases

- 8. Add Title

Use Case	8. Add Title
Actor	Librarian
Purpose	(As in the business use case)
Overview	(As in the business use case)
Type	Primary and Essential
Cross Reference	System Functions: R2.1, R2.4 Use Case: “Add Item”
Pre-Requisites	N/A
Typical Courses of Events	(A) : Actor, (S) : System 1. (A) A librarian inputs a title’s information 2. (S) Check if a corresponding title exists 3. (S) Add a new title 4. (S) Invoke “Add Item”
Alternative Courses of Events	N/A
Exceptional Courses of Events	Line 1: If invalid title information is entered, indicate an error.

Activity 2031. Define Essential Use Cases

- 9. Remove Title

Use Case	9. Remove Title
Actor	Librarian
Purpose	(As in the business use case)
Overview	(As in the business use case)
Type	Primary and Essential
Cross Reference	System Functions: R2.2 Use Case: -
Pre-Requisites	N/A
Typical Courses of Events	(A) : Actor, (S) : System 1. (A) A librarian inputs a title's information to deleted 2. (S) Check if a corresponding title exists 3. (S) Remove the items of the title 4. (S) Remove the title
Alternative Courses of Events	N/A
Exceptional Courses of Events	Line 1: If invalid title information is entered, indicate an error.

Activity 2031. Define Essential Use Cases

- 10. Update Title

Use Case	10. Update Title
Actor	Librarian
Purpose	(As in the business use case)
Overview	(As in the business use case)
Type	Primary and Essential
Cross Reference	System Functions: R2.3 Use Case: -
Pre-Requisites	N/A
Typical Courses of Events	(A) : Actor, (S) : System 1. (A) A librarian inputs a title's information to change 2. (S) Check if a corresponding title exists 3. (S) Update the title's information
Alternative Courses of Events	N/A
Exceptional Courses of Events	Line 1: If invalid title information is entered, indicate an error.

Activity 2031. Define Essential Use Cases

- 11. Add Item

Use Case	11. Add Item
Actor	Librarian
Purpose	(As in the business use case)
Overview	(As in the business use case)
Type	Primary and Essential
Cross Reference	System Functions: R1.6, R2.1, R2.4 Use Cases: “Notify Availability”, “Add Title”
Pre-Requisites	N/A
Typical Courses of Events	(A) : Actor, (S) : System 1. (A) A librarian inputs a item to add 2. (S) Check if a corresponding title exists 3. (S) Add the item 4. (S) Invoke “Notify Availability”
Alternative Courses of Events	N/A
Exceptional Courses of Events	Line 1: If invalid title information is entered, indicate an error.

Activity 2031. Define Essential Use Cases

- 12. Remove Item

Use Case	12. Remove Item
Actor	Librarian
Purpose	(As in the business use case)
Overview	(As in the business use case)
Type	Primary and Essential
Cross Reference	System Functions: R2.1, R2.5 Use Case: “Remove Title”
Pre-Requisites	N/A
Typical Courses of Events	(A) : Actor, (S) : System 1. (A) A librarian inputs an item’s information to remove 2. (S) Check if a corresponding title exists 3. (S) Check if a corresponding item exists 4. (S) Remove the item 5. (S) If there is no remaining item, invoke “Remove Title”
Alternative Courses of Events	N/A
Exceptional Courses of Events	Line 1: If invalid title information is entered, indicate an error.

Activity 2031. Define Essential Use Cases

- 13. Update Item

Use Case	13. Update Item
Actor	Librarian
Purpose	(As in the business use case)
Overview	(As in the business use case)
Type	Primary and Essential
Cross Reference	System Functions: R2.6 Use Case: -
Pre-Requisites	N/A
Typical Courses of Events	(A) : Actor, (S) : System 1. (A) A librarian inputs an item's information to update 2. (S) Check if a corresponding title exists 3. (S) Check if a corresponding item exists 4. (S) Update the item's information
Alternative Courses of Events	N/A
Exceptional Courses of Events	Line 1: If invalid title information is entered, indicate an error.

Activity 2031. Define Essential Use Cases

- 14. Add Borrower

Use Case	14. Add Borrower
Actor	Librarian
Purpose	(As in the business use case)
Overview	(As in the business use case)
Type	Primary and Essential
Cross Reference	System Functions: R1.1, R1.3, R3.1 Use Cases: “Make Reservation”, “Lend Item”
Pre-Requisites	N/A
Typical Courses of Events	(A) : Actor, (S) : System 1. (A) A librarian inputs borrower’s information such as SSN, name, address, zip code, phone number, and age. 2. (S) Check if the corresponding borrower exists 3. (S) Add New borrower
Alternative Courses of Events	N/A
Exceptional Courses of Events	Line 1: If invalid borrower information is entered, indicate an error.

Activity 2031. Define Essential Use Cases

- 15. Remove Borrower

Use Case	15. Remove Borrower
Actor	Librarian
Purpose	(As in the business use case)
Overview	(As in the business use case)
Type	Primary and Essential
Cross Reference	System Functions: R3.2 Use Case: -
Pre-Requisites	N/A
Typical Courses of Events	(A) : Actor, (S) : System 1. (A) A librarian inputs a borrower's information to remove 2. (S) Check if a corresponding borrower exists 3. (S) If there is a loan of the borrower, remove the loan. 4. (S) Remove the borrower's information
Alternative Courses of Events	N/A
Exceptional Courses of Events	Line 1: If invalid borrower information is entered, indicate an error.

Activity 2031. Define Essential Use Cases

- 16. Update Borrower

Use Case	16. Update Borrower
Actor	Librarian
Purpose	(As in the business use case)
Overview	(As in the business use case)
Type	Primary and Essential
Cross Reference	System Functions: R3.2 Use Case: -
Pre-Requisites	N/A
Typical Courses of Events	(A) : Actor, (S) : System 1. (A) A librarian inputs a borrower's information to change 2. (S) Check if a corresponding borrower exists 3. (S) Update the borrower's information
Alternative Courses of Events	N/A
Exceptional Courses of Events	Line 1: If invalid borrower information is entered, indicate an error.

Activity 2031. Define Essential Use Cases

- 17. Log-In

Use Case	17. Log-In
Actor	Librarian
Purpose	(As in the business use case)
Overview	(As in the business use case)
Type	Primary and Essential
Cross Reference	System Functions: R4.1 Use Case: -
Pre-Requisites	A librarian should have user name and password.
Typical Courses of Events	(A) : Actor, (S) : System 1. (A) A librarian enters his(her) user name and password into the system 2. (S) Check if the user name and password are correct
Alternative Courses of Events	N/A
Exceptional Courses of Events	Line 1: If invalid user name and password entered, indicate an error.

Activity 2031. Define Essential Use Cases

- 18. Log-Out

Use Case	18. Log-Out
Actor	Librarian
Purpose	(As in the business use case)
Overview	(As in the business use case)
Type	Primary and Essential
Cross Reference	System Functions: R4.1 Use Case: -
Pre-Requisites	A librarian should have user name and password.
Typical Courses of Events	(A) : Actor, (S) : System 1. (A) A librarian exits the system 2. (S) Log the librarian’s information
Alternative Courses of Events	N/A
Exceptional Courses of Events	Line 1: If invalid user name and password entered, indicate an error.

Activity 2031. Define Essential Use Cases

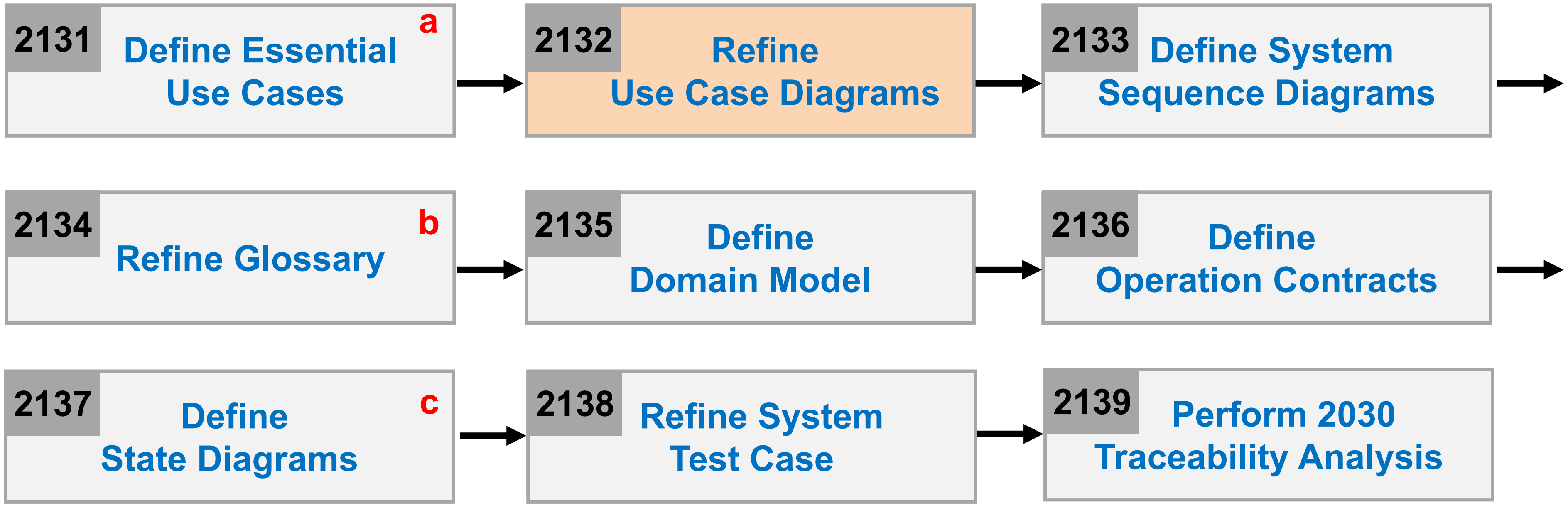
- 19. Count Loans

Use Case	19. Count Loans
Actor	Librarian
Purpose	(As in the business use case)
Overview	(As in the business use case)
Type	Primary and Essential
Cross Reference	System Functions: R5.1 Use Case: -
Pre-Requisites	A librarian should have user name and password.
Typical Courses of Events	(A) : Actor, (S) : System 1. (A) A librarian requests total counts of titles checked out 2. (S) Find loan information 3. (S) Calculate total counts of titles checked out 4. (S) Print total counts.
Alternative Courses of Events	N/A
Exceptional Courses of Events	Line 1: If invalid user name and password entered, indicate an error.

Activity 2032. Refine Use Case Diagrams

- Phase 2030 Activities

a. if not yet done
b. ongoing
c. optional



2131 Define Essential Use Cases a

2132 Refine Use Case Diagrams

2133 Define System Sequence Diagrams

2134 Refine Glossary b

2135 Define Domain Model

2136 Define Operation Contracts

2137 Define State Diagrams c

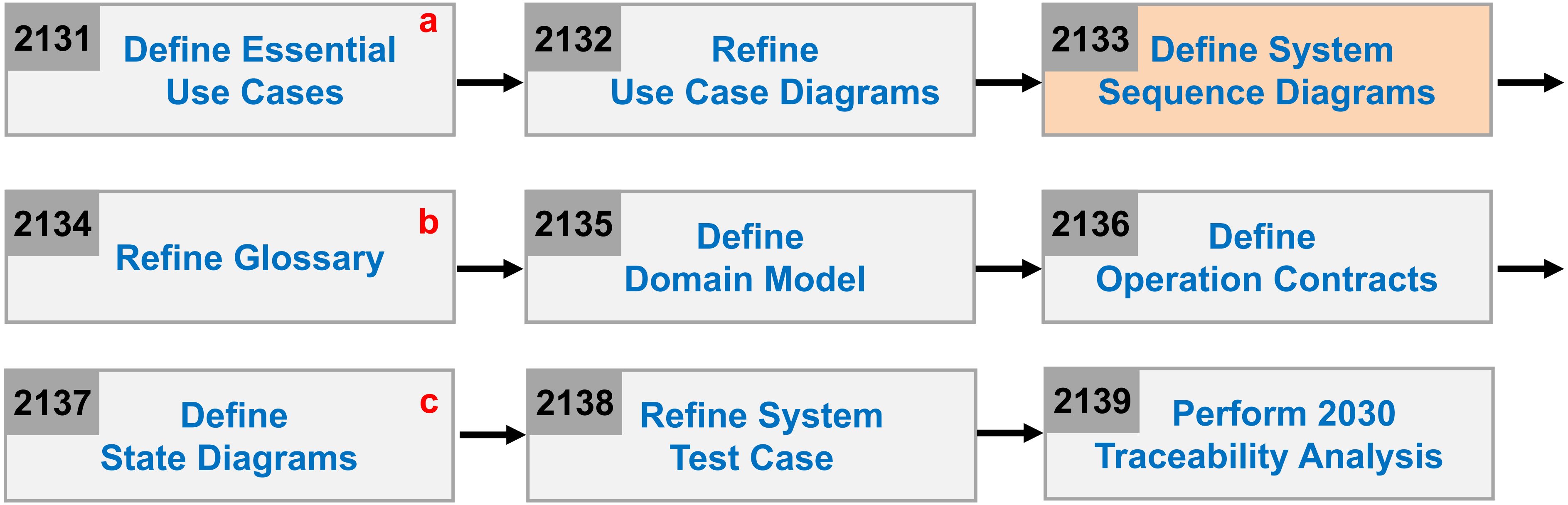
2138 Refine System Test Case

2139 Perform 2030 Traceability Analysis

Activity 2033. Define System Sequence Diagrams

- Phase 2030 Activities

a. if not yet done
b. ongoing
c. optional



2131 Define Essential Use Cases a

2132 Refine Use Case Diagrams

2133 Define System Sequence Diagrams

2134 Refine Glossary b

2135 Define Domain Model

2136 Define Operation Contracts

2137 Define State Diagrams c

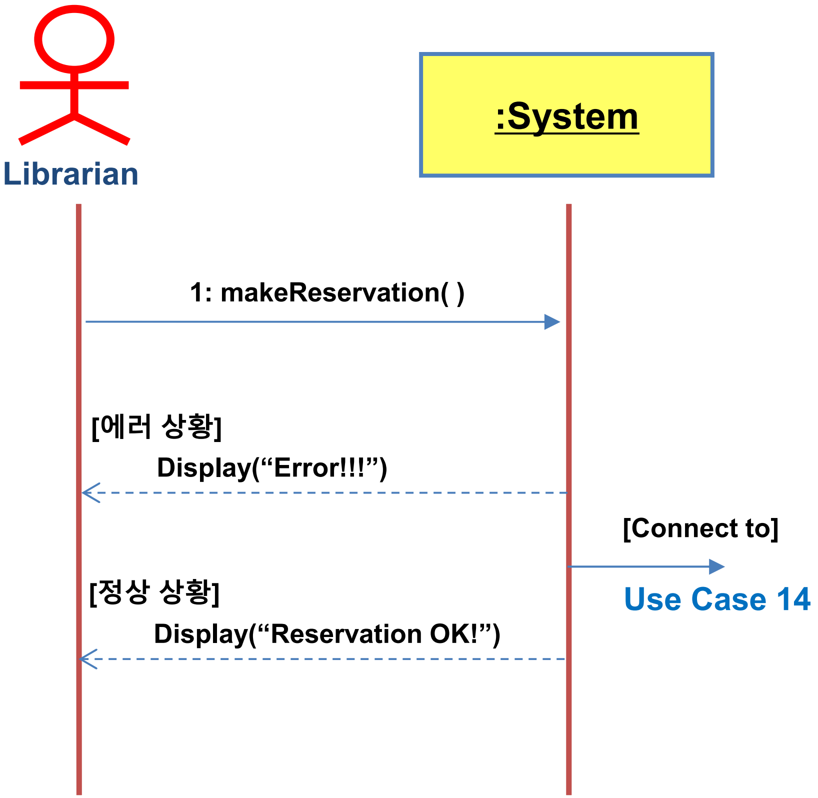
2138 Refine System Test Case

2139 Perform 2030 Traceability Analysis

Activity 2033. Define System Sequence Diagrams

USE CASE: 1. Make Reservation

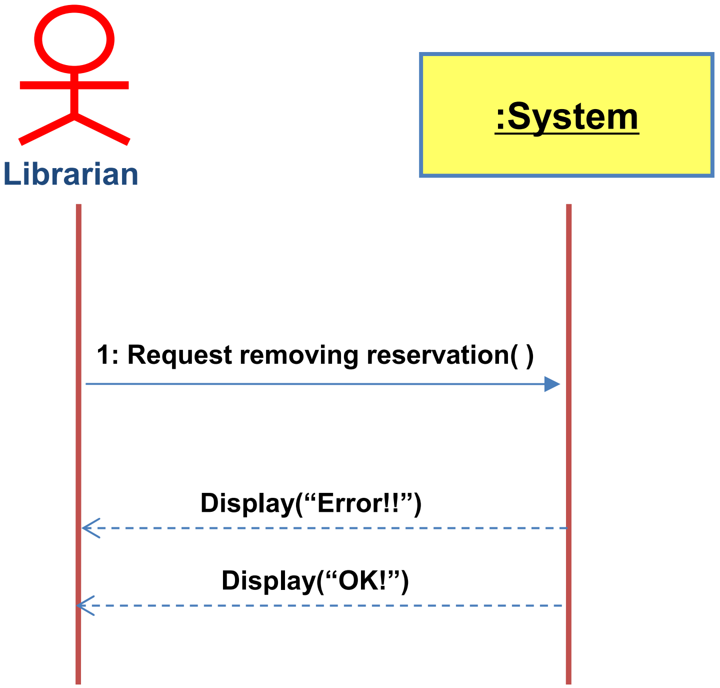
1. (A) A librarian requests the reservation of title.
2. (S) Check if corresponding title exist.
3. (S) Check if corresponding borrower exist.
4. (S) If the borrower does not exist, invoke "Add Borrower".
   (→ connect to the other Use Case)
5. (S) Create reservation information.



Activity 2033. Define System Sequence Diagrams

USE CASE: 2. Remove Reservation

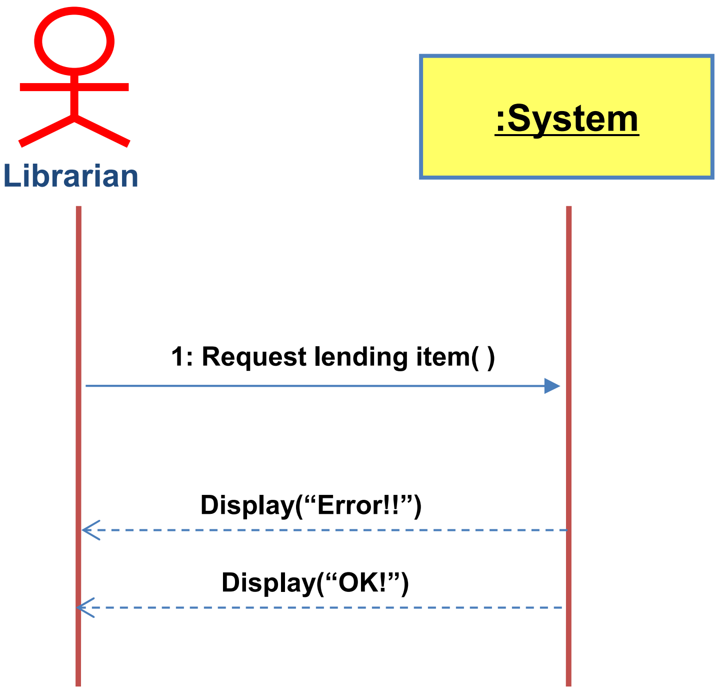
1. (A) A librarian requests removing reservation.
2. (S) Check if corresponding title exist.
3. (S) Check if corresponding borrower exist.
4. (S) Find the reservation.
5. (S) Remove the reservation.



Activity 2033. Define System Sequence Diagrams

USE CASE: 3. Lend Item

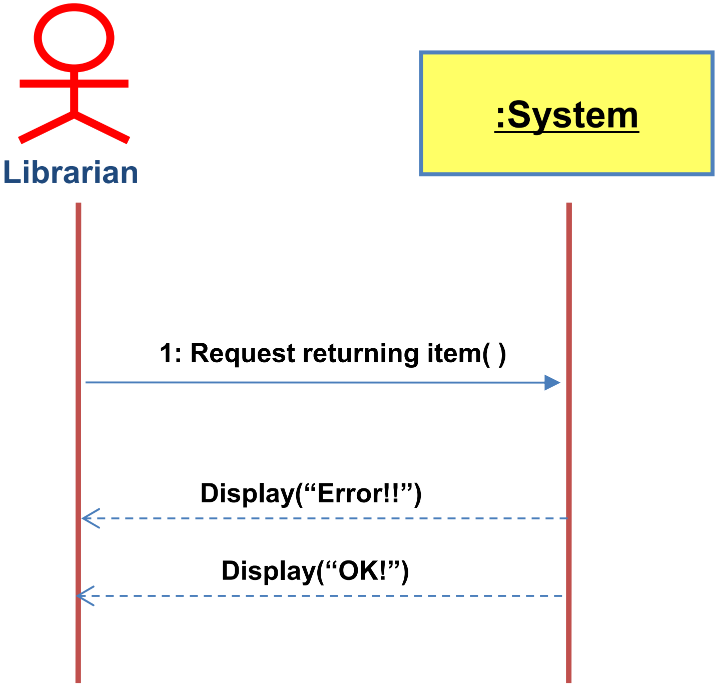
1. (A) A librarian requests lending item.
2. (S) Check if corresponding title exist.
3. (S) Check if corresponding item is available.
4. (S) If the item was reserved, invoke "Remove Reservation".
5. (S) Check if corresponding borrower exist.
6. (S) If the borrower does not exist, invoke "Add Borrower".
7. (A) Create new loan.



Activity 2033. Define System Sequence Diagrams

USE CASE: 4. Return Item

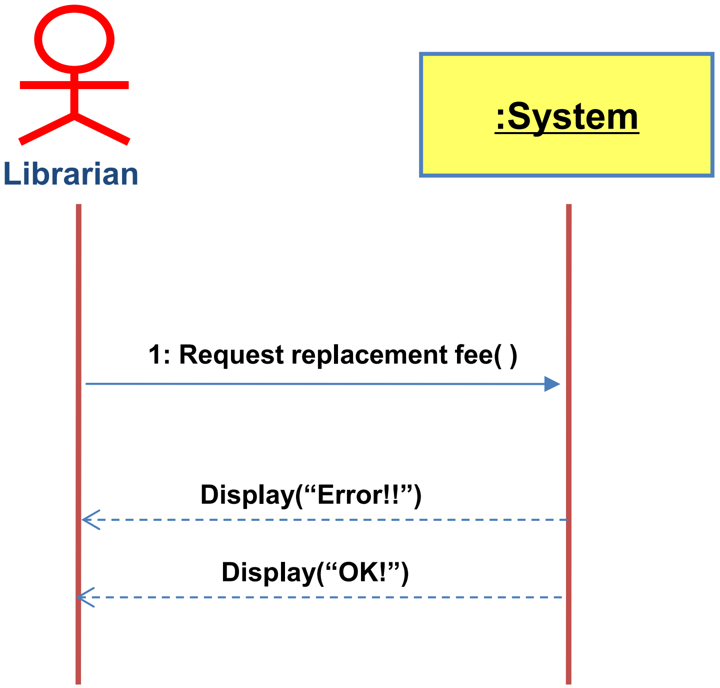
1. (A) A librarian requests returning item
2. (S) Check if a corresponding title exists
3. (S) Check if a corresponding borrower exists
4. (S) Check if a corresponding item is loaned
5. (S) Find the borrower of the item
6. (S) Check whether the returning due-date is over or not
7. (S) If the returning due-date is over, invoke “Calculate Late-Return-Fee”
8. (S) Remove the loan
9. (S) If the item is reserved, invoke “Notify Availability”



Activity 2033. Define System Sequence Diagrams

USE CASE: 6. Get Replacement Fee

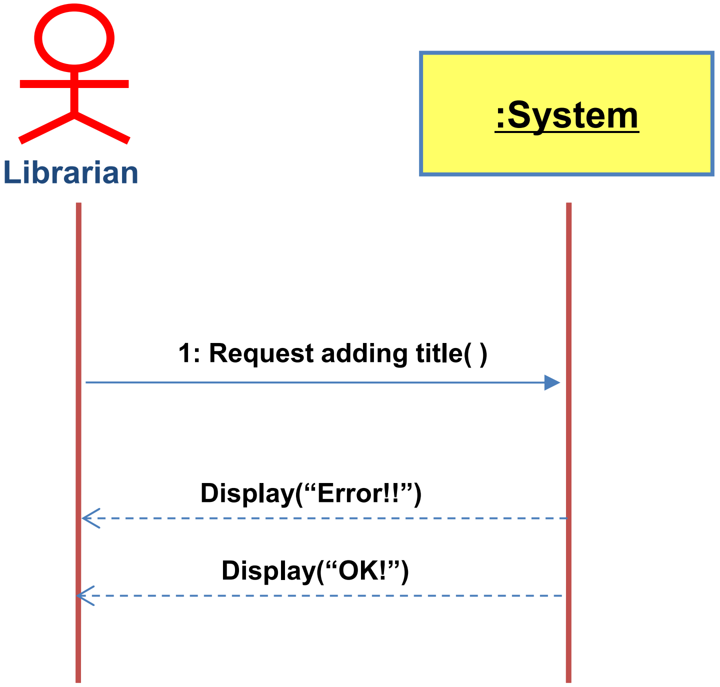
1. (A) A librarian inputs a title's information
2. (S) Check if a corresponding title exists
3. (S) Find the price of the title
4. (S) Compute replacement-fee



Activity 2033. Define System Sequence Diagrams

USE CASE: 8. Add Title

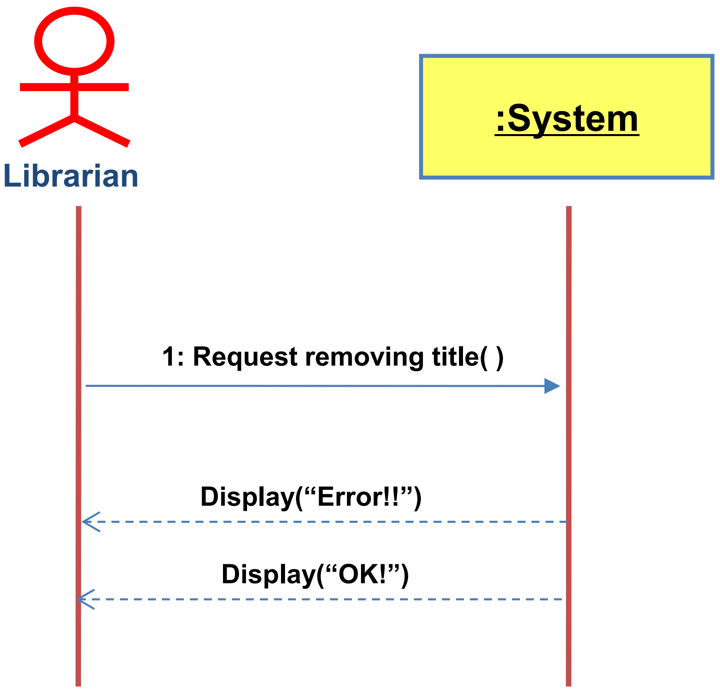
1. (A) A librarian inputs a title's information
2. (S) Check if a corresponding title exists
3. (S) Add a new title
4. (S) Invoke "Add Item"



Activity 2033. Define System Sequence Diagrams

USE CASE: 9. Remove Title

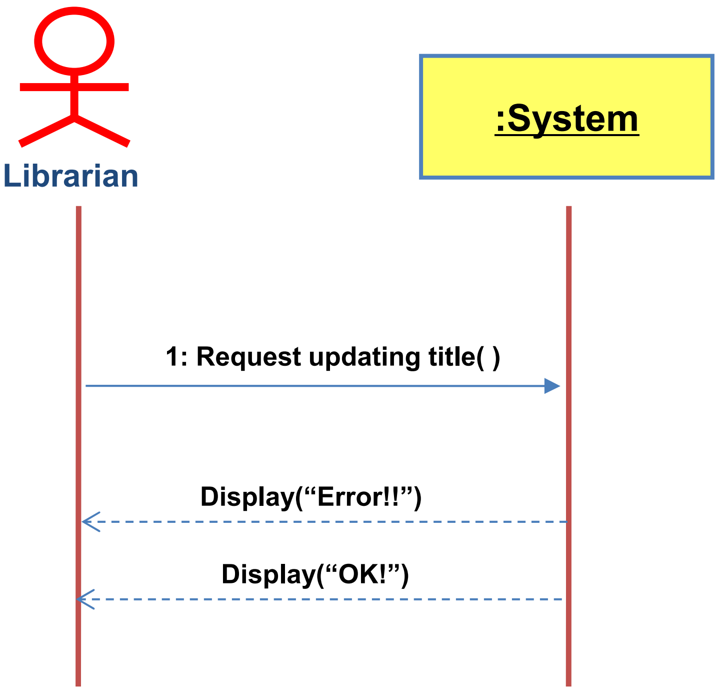
1. (A) A librarian inputs a title's information to deleted
2. (S) Check if a corresponding title exists
3. (S) Remove the items of the title
4. (A) Remove the title



Activity 2033. Define System Sequence Diagrams

USE CASE: 10. Update Title

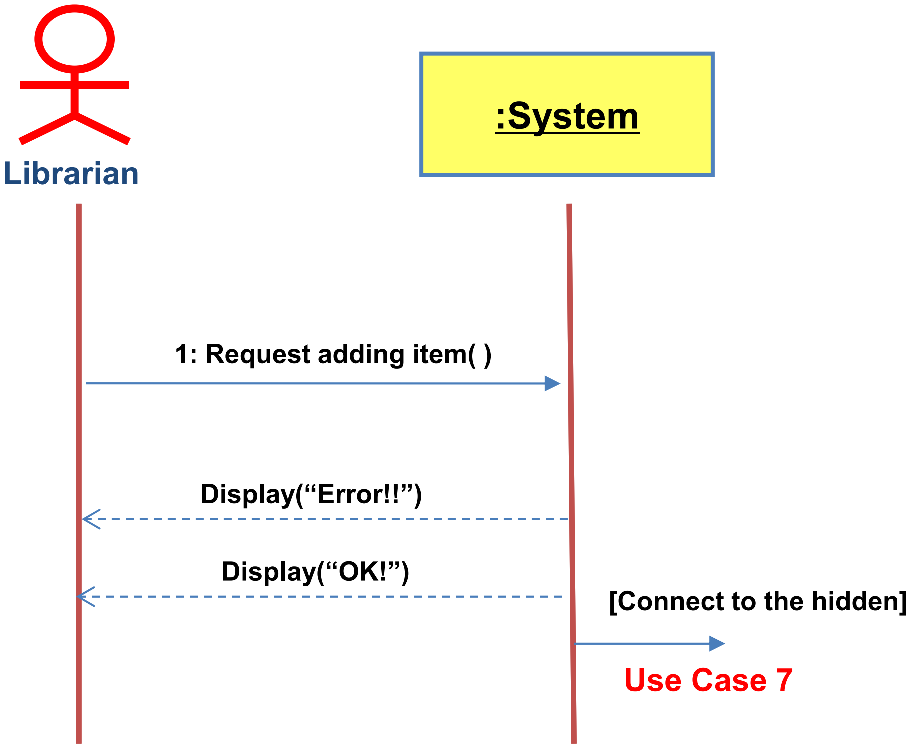
1. (A) A librarian inputs a title's information to change
2. (S) Check if a corresponding title exists
3. (S) Update the title's information



Activity 2033. Define System Sequence Diagrams

USE CASE: 11. Add Item

1. (A) A librarian inputs a item to add
2. (S) Check if a corresponding title exists
3. (S) Add the item
4. (S) Invoke "Notify Availability"



Librarian

:System

1: Request adding item( )

Display("Error!!")

Display("OK!")

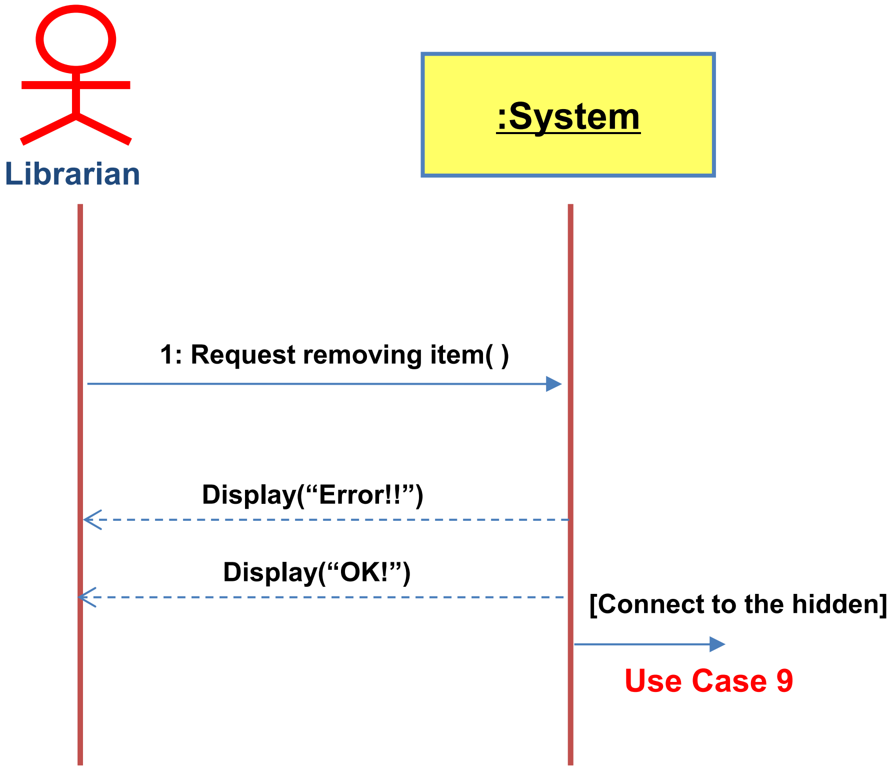
[Connect to the hidden]

Use Case 7

Activity 2033. Define System Sequence Diagrams

USE CASE: 12. Remove Item

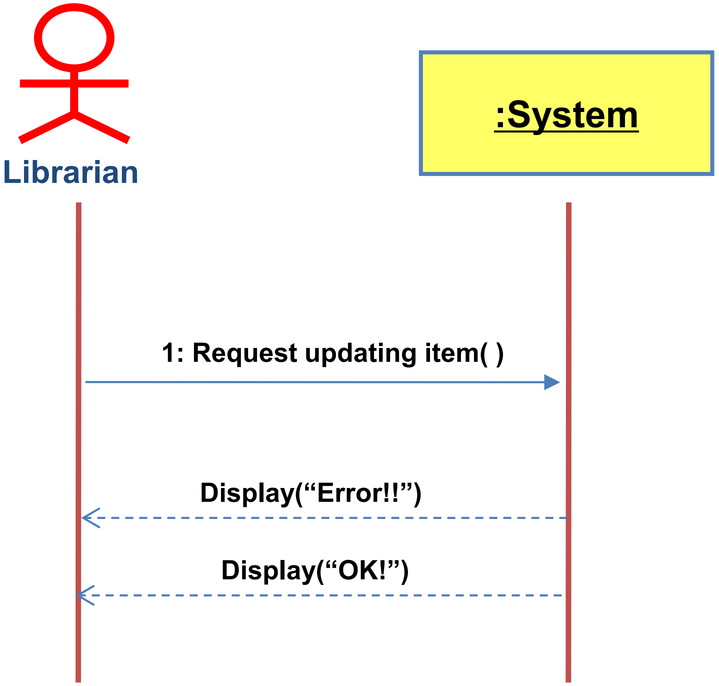
1. (A)A librarian inputs an item’s information to remove
2. (S) Check if a corresponding title exists
3. (S) Check if a corresponding item exists
4. (S) Remove the item
5. (S) If there is no remaining item, invoke “Remove Title”



Activity 2033. Define System Sequence Diagrams

USE CASE: 13. Update Item

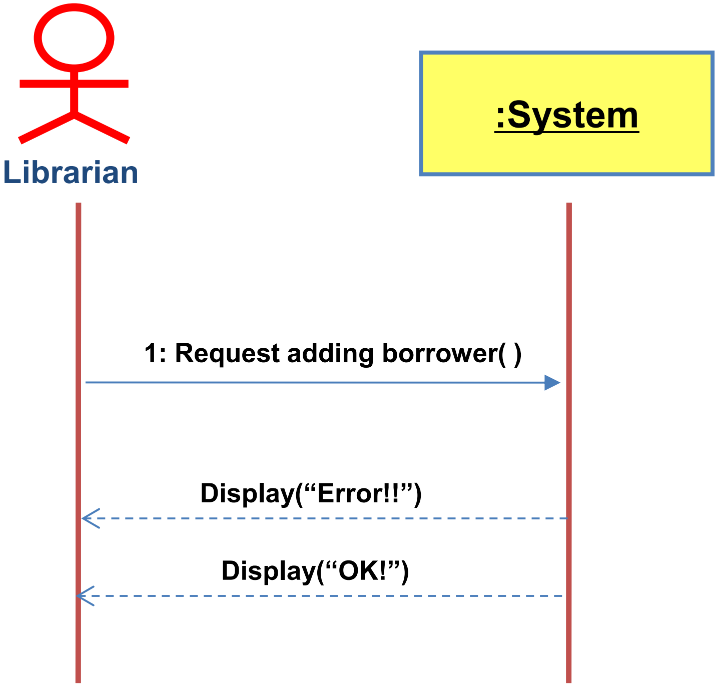
1. (A) A librarian inputs an item's information to update
2. (S) Check if a corresponding title exists
3. (S) Check if a corresponding item exists
4. (S) Update the item's information



Activity 2033. Define System Sequence Diagrams

USE CASE: 14. Add Borrower

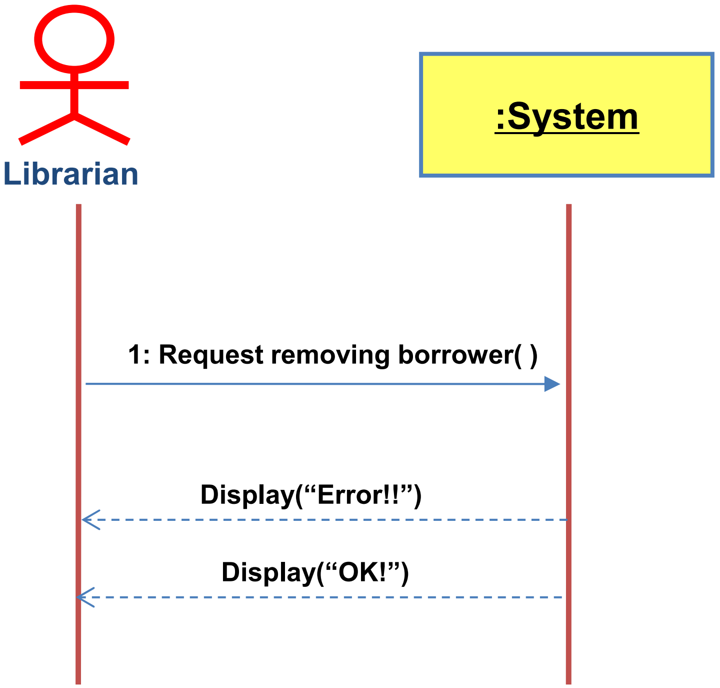
1. (A) A librarian inputs borrower's information such as SSN, name, address, zip code, phone number, and age.
2. (S) Check if the corresponding borrower exists
3. (S) Add New borrower



Activity 2033. Define System Sequence Diagrams

USE CASE: 15. Remove Borrower

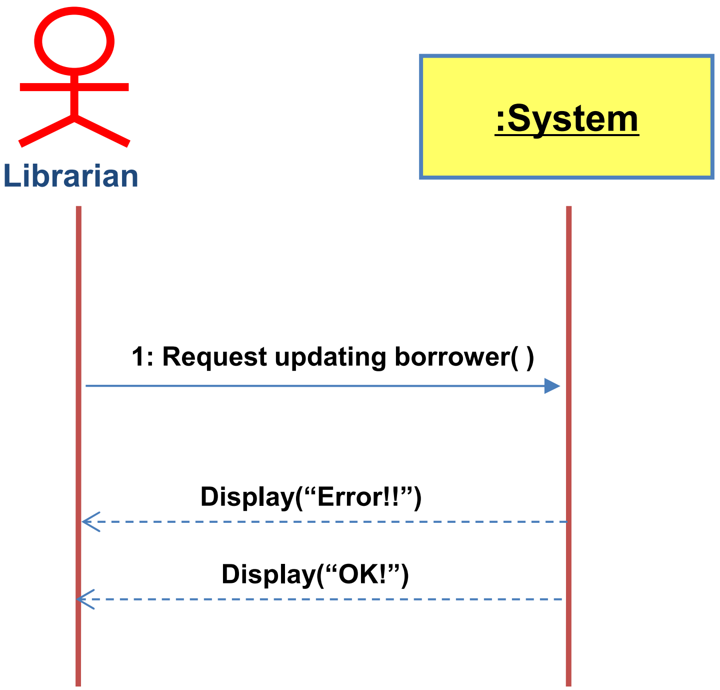
1. (A) A librarian inputs a borrower's information to remove
2. (S) Check if a corresponding borrower exists
3. (S) If there is a loan of the borrower, remove the loan.
4. (S) Remove the borrower's information



Activity 2033. Define System Sequence Diagrams

USE CASE: 16. Update Borrower

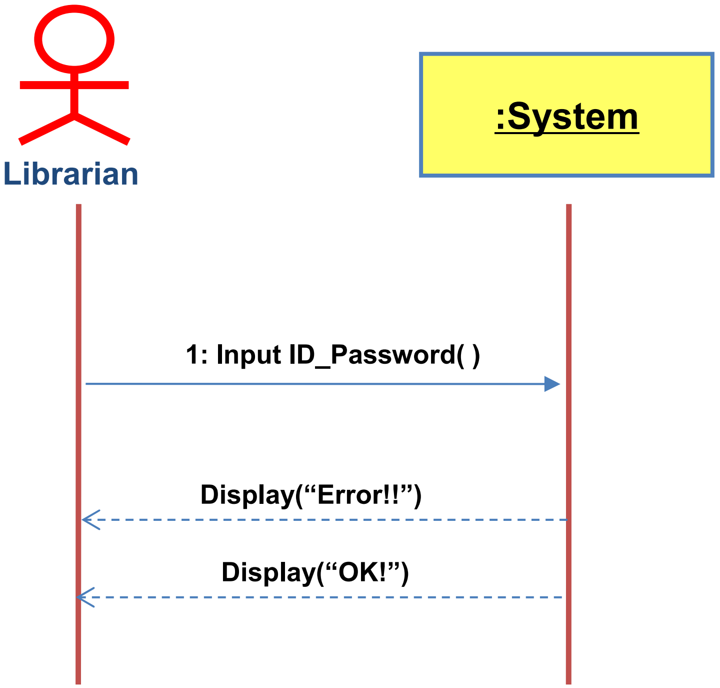
1. (A) A librarian inputs a borrower's information to change
2. (S) Check if a corresponding borrower exists
3. (S) Update the borrower's information



Activity 2033. Define System Sequence Diagrams

USE CASE: 17. Log-In

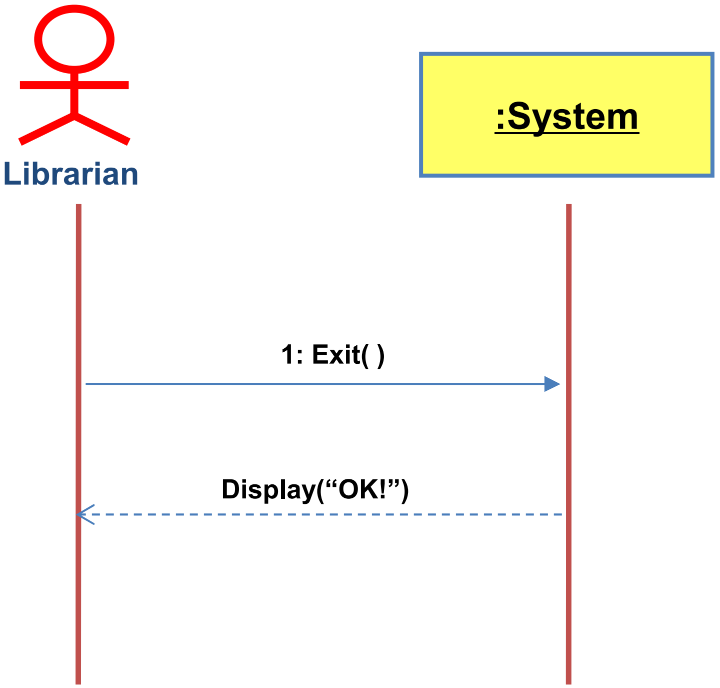
1. (A) A librarian enters his(her) user name and password into the system
2. (S) Check if the user name and password are correct



Activity 2033. Define System Sequence Diagrams

USE CASE: 17. Log-Out

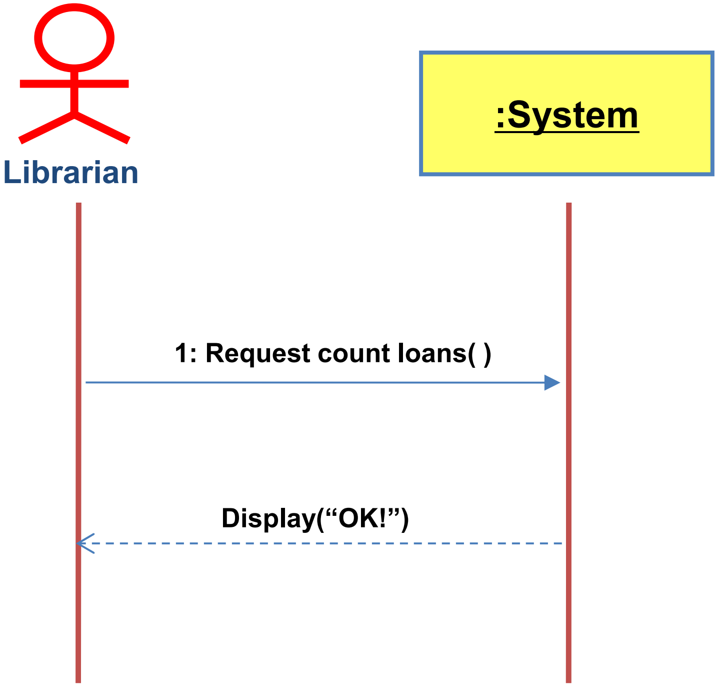
1. (A) A librarian exits the system
2. (S) Log-off the librarian's information



Activity 2033. Define System Sequence Diagrams

USE CASE: 18. Count Loans

1. (A) A librarian requests total counts of titles checked out
2. (S) Find loan information
3. (S) Calculate total counts of titles checked out
4. (S) Print total counts.

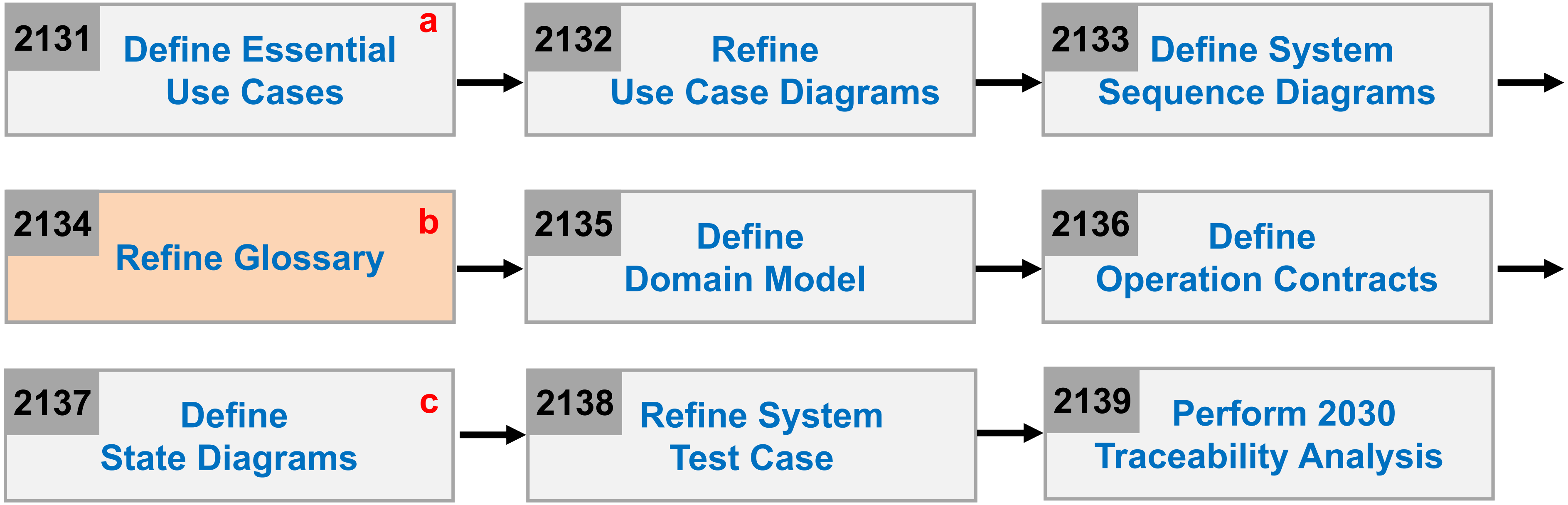


Use Case	Name of Actor-Activated Event	System Operations
1. Make Reservation	1: Request making reservation( )	1. makeReservation( )
2. Remove Reservation	1: Request removing reservation( )	2. removeReservation( )
3. Lend Item	1: Request lending item( )	3. LendItem( )
4. Return Item	1: Request returning item( )	4. returnItem( )
5. Calculate Late-Return-Fee	N/A	N/A
6. Get Replacement Fee	1: Request replacement fee( )	5. getReplacementFee( )
7. Notify Availability	N/A	N/A
8. Add Title	1: Request adding title( )	6. addTitle( )
9. Remove Title	1: Request removing title( )	7. removeTitle( )
10. Update Title	1: Request updating title( )	8. updateTitle( )
11. Add Item	1: Request adding item( )	9. addItem( )
12. Remove Item	1: Request removing item( )	10. removeItem( )
13. Update Item	1: Request updating item( )	11. updateItem( )
14. Add Borrower	1: Request adding borrower( )	12. addBorrower( )
15. Remove Borrower	1: Request removing borrower( )	13. removeBorrower( )
16. Update Borrower	1: Request updating borrower( )	14. updateBorrower( )
17. Log-In	1: Input ID_Password( )	15. log-In( )
18. Log-Out	1: Exit( )	16. log-Out( )
19. Count Loans	1: Request count loans( )	17. countLoans( )

Activity 2034. Refine Glossary

- Phase 2030 Activities

a. if not yet done
b. ongoing
c. optional



2131 Define Essential Use Cases a

2132 Refine Use Case Diagrams

2133 Define System Sequence Diagrams

2134 Refine Glossary b

2135 Define Domain Model

2136 Define Operation Contracts

2137 Define State Diagrams c

2138 Refine System Test Case

2139 Perform 2030 Traceability Analysis

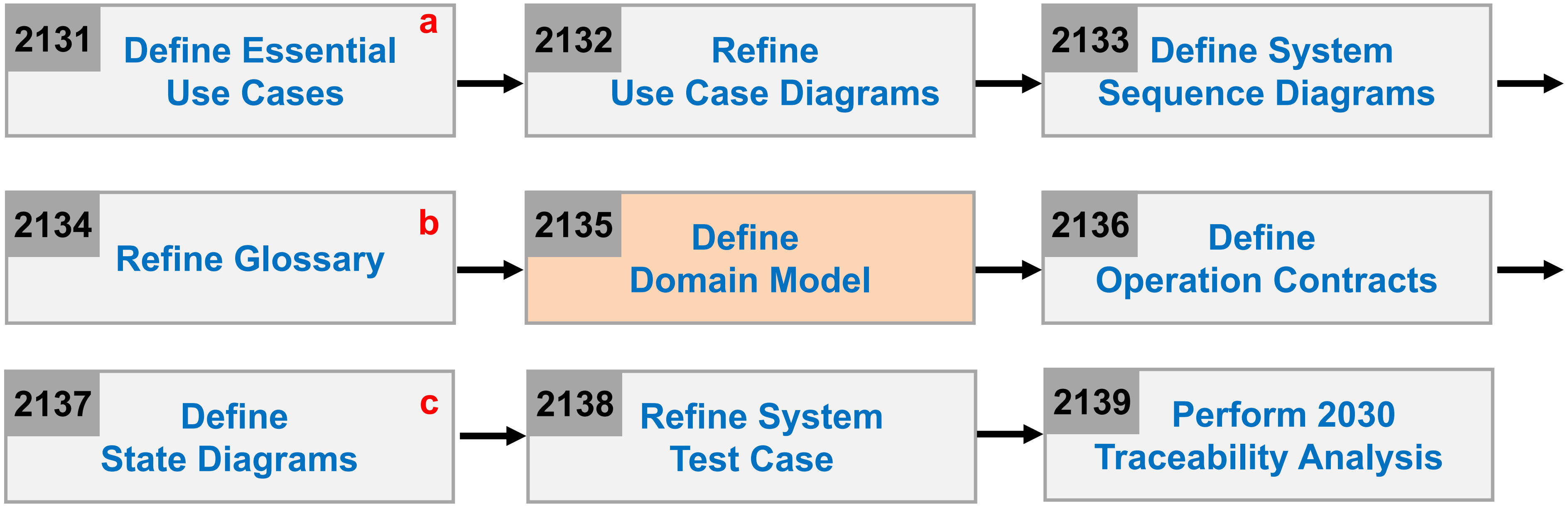
Activity 2034. Refine Glossary

Term	Category	Remarks
Title	Class	A type of books or magazines which are registered in the library system.
Item	Class	Each copy of book or magazine.
Reservation	Class	An action of lending title that is available for use when it is needed.
Borrower	Class	A person that lends, returns item.
Loan	Class	An action of lending a book/magazine from the library.
Librarian	Class	An employee of the library who interacts with the borrower.
Librarian.name	Attribute	The name of librarian.
Librarian.userId	Attribute	The user name of librarian.
Librarian.password	Attribute	The password of librarian.
Title.name	Attribute	The title of a book or a magazine.
Title.publisher	Attribute	The publishing company of the title.
Title.isbn	Attribute	The International Standard Book Number of title.
Title.price	Attribute	The price of title.
Title.count	Attribute	The number of item contained in a title.
Title.lendingtime	Attribute	The lending time of a title.
Book.author	Attribute	The author name of a book.
Magazine.month	Attribute	The publication cycle of a magazine.
Reservation.date : Date	Attribute	The date of reservation.
Item.id : Integer	Attribute	Item number.
Loan.date : Date	Attribute	Lending date of an item.
Loan.late-return-fee	Attribute	Over lending time of an item.
Borrower.SSN	Attribute	The resident registration number
Borrower.name	Attribute	A borrower name.
Borrower.address	Attribute	A borrower address.
Borrower.zip	Attribute	A zip code of borrower.
Borrower.age	Attribute	A borrower age.

Activity 2035. Define Domain Model

- Phase 2030 Activities

a. if not yet done
b. ongoing
c. optional



2131 Define Essential Use Cases a

2132 Refine Use Case Diagrams

2133 Define System Sequence Diagrams

2134 Refine Glossary b

2135 Define Domain Model

2136 Define Operation Contracts

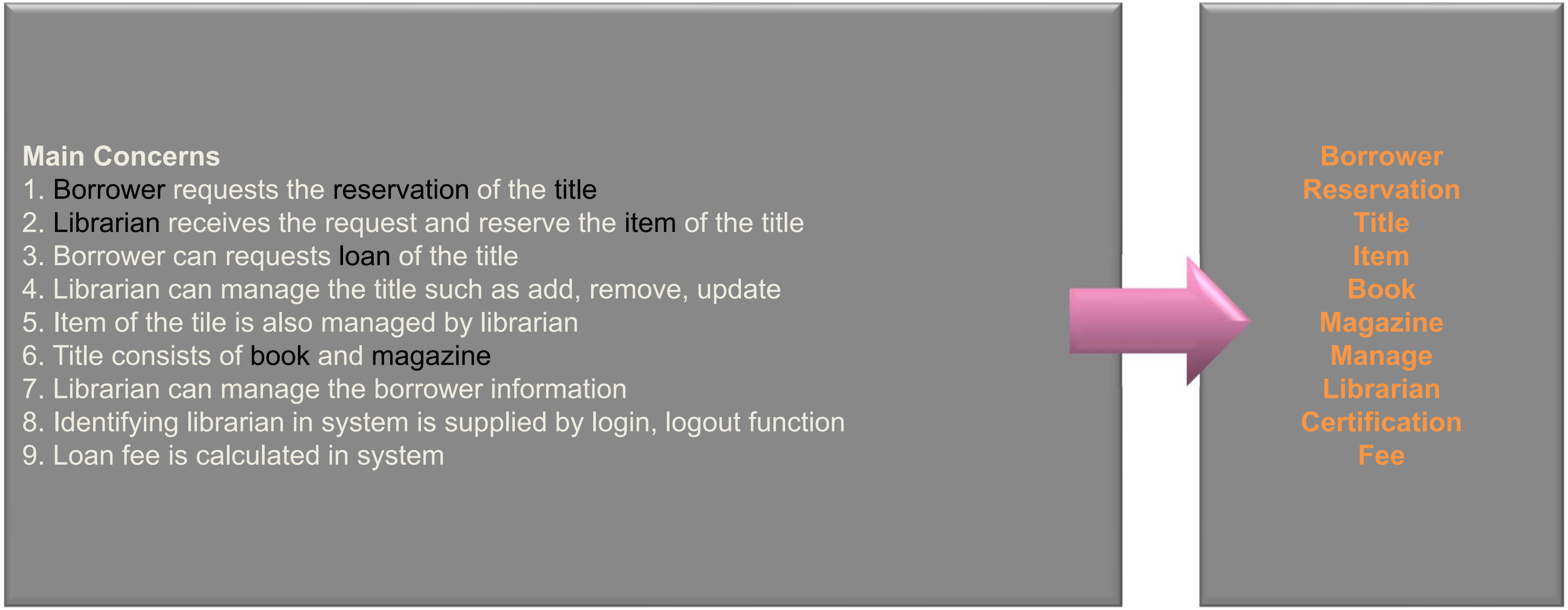
2137 Define State Diagrams c

2138 Refine System Test Case

2139 Perform 2030 Traceability Analysis

Activity 2035. Define Domain Model

- Step 1. List concepts
  - Guideline 2



Main Concerns

1. Borrower requests the reservation of the title
2. Librarian receives the request and reserve the item of the title
3. Borrower can requests loan of the title
4. Librarian can manage the title such as add, remove, update
5. Item of the tile is also managed by librarian
6. Title consists of book and magazine
7. Librarian can manage the borrower information
8. Identifying librarian in system is supplied by login, logout function
9. Loan fee is calculated in system

Borrower
Reservation
Title
Item
Book
Magazine
Manage
Librarian
Certification
Fee

Activity 2035. Define Domain Model

- Step 2. Assign class names into concepts
  - Title
  - Librarian
  - Book
  - Magazine
  - Loan
  - Reservation
  - Borrower
  - Item

Activity 2035. Define Domain Model

- Step 3. Identify associations according to association categories

Association Category	Associations
A is known/logged/recorded/reported/captured in B	Item – Loan Item – Title Loan – Borrower Title – Reservation
A is a line item of B	Item – Title
A is recorded in B	Item – Title
A is related to a transaction of B	Borrower – Loan Borrower – Reservation
A is an organization submit of B	Book – Title Magazine – Title

Activity 2035. Define Domain Model

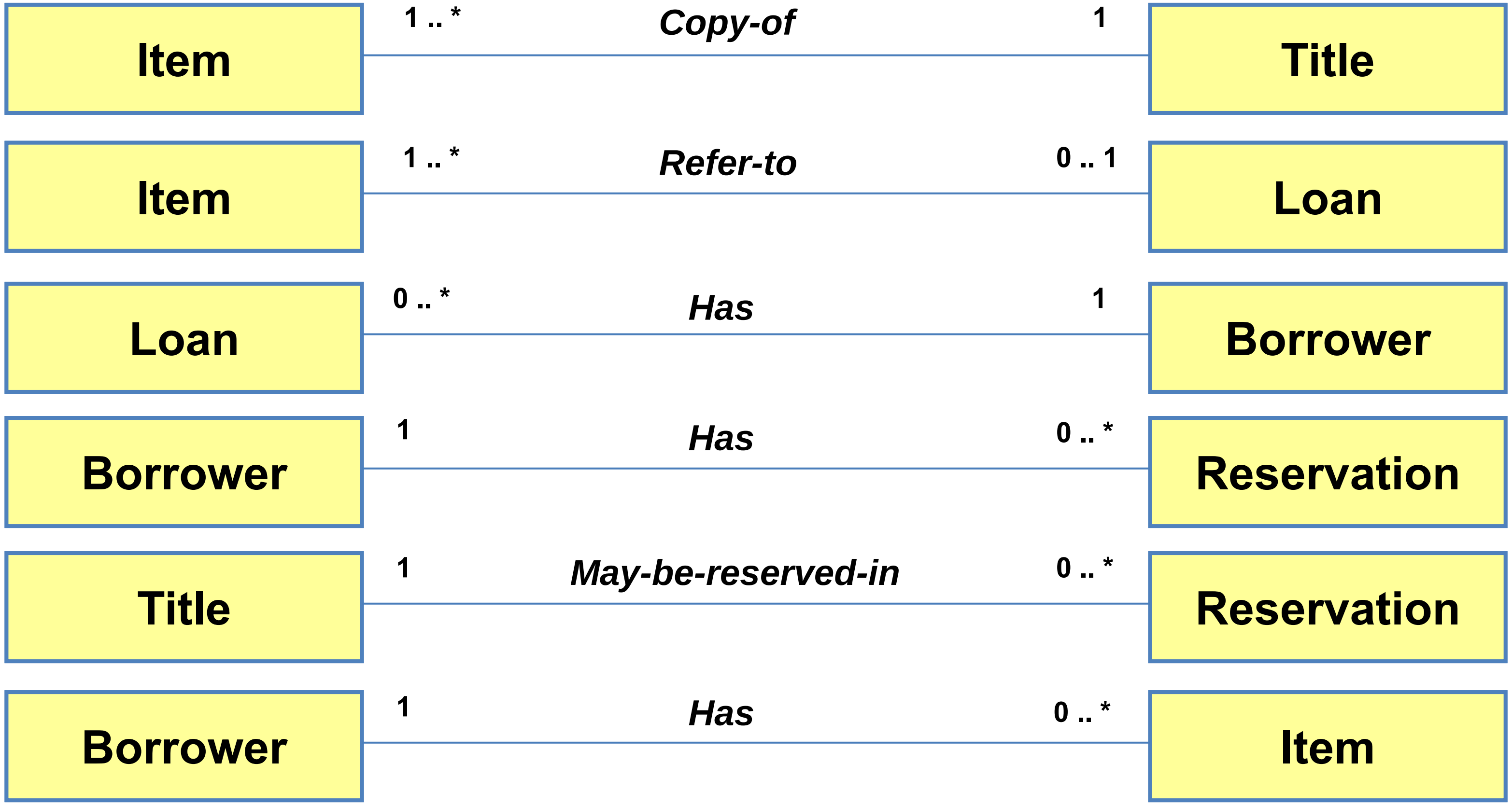
- Step 4. Assign priorities into identified associations

Association Name	Priority
Item – Title	High

- Step 5. Assign names into associations
  - Item *Copy-of* Title
  - Item *Refer-to* Loan
  - Loan *Has* Borrower
  - Borrower *Has* Reservation
  - Title *May-be-reserved-in* Reservation
  - Borrower *Has* Item

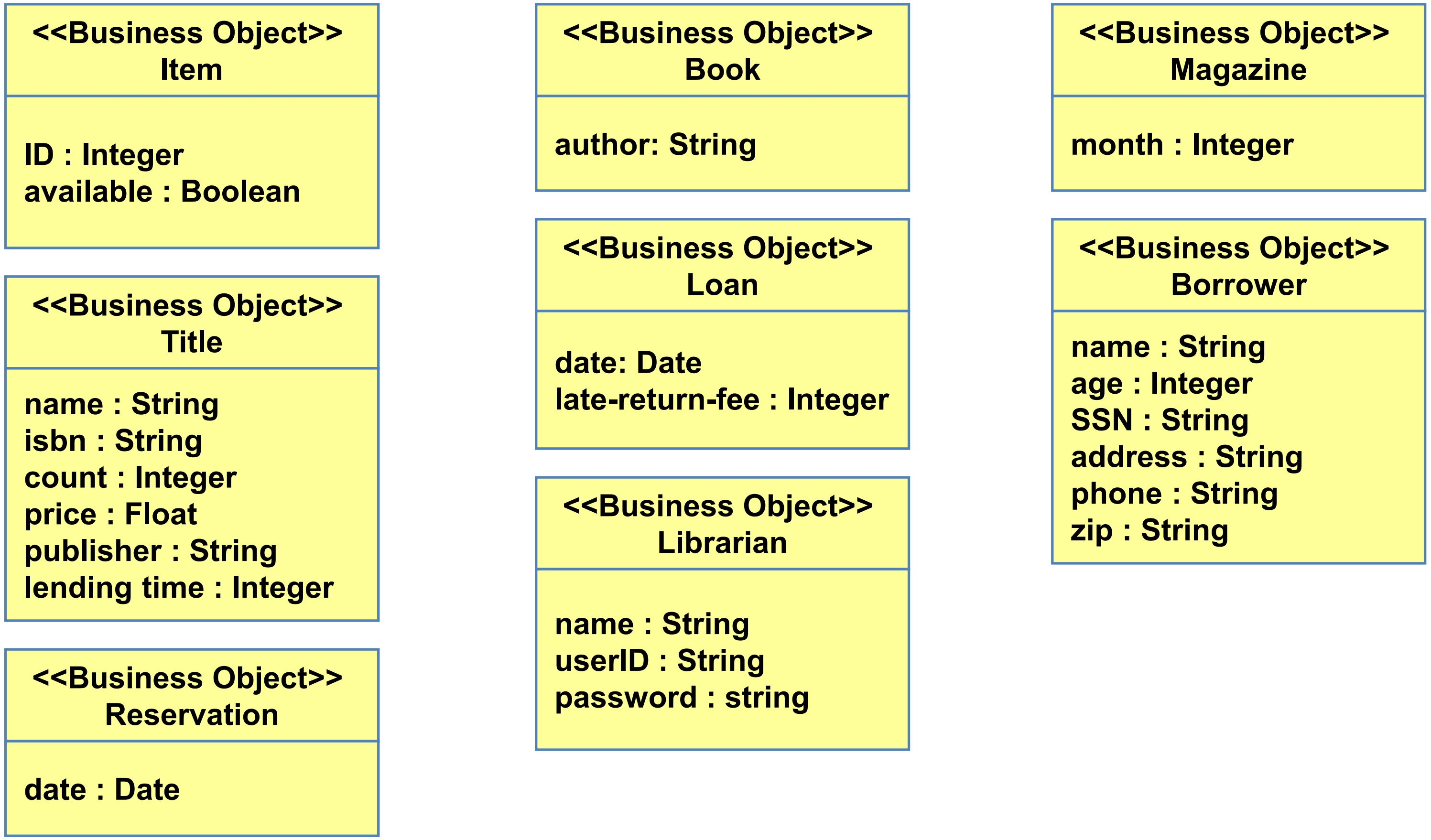
Activity 2035. Define Domain Model

- Step 6. Add multiplicity into the ends of an associations



Activity 2035. Define Domain Model

- Step 7. Identify attributes by reading



<<Business Object>>
Item

ID : Integer
available : Boolean

<<Business Object>>
Title

name : String
isbn : String
count : Integer
price : Float
publisher : String
lending time : Integer

<<Business Object>>
Reservation

date : Date

<<Business Object>>
Book

author: String

<<Business Object>>
Loan

date: Date
late-return-fee : Integer

<<Business Object>>
Librarian

name : String
userID : String
password : string

<<Business Object>>
Magazine

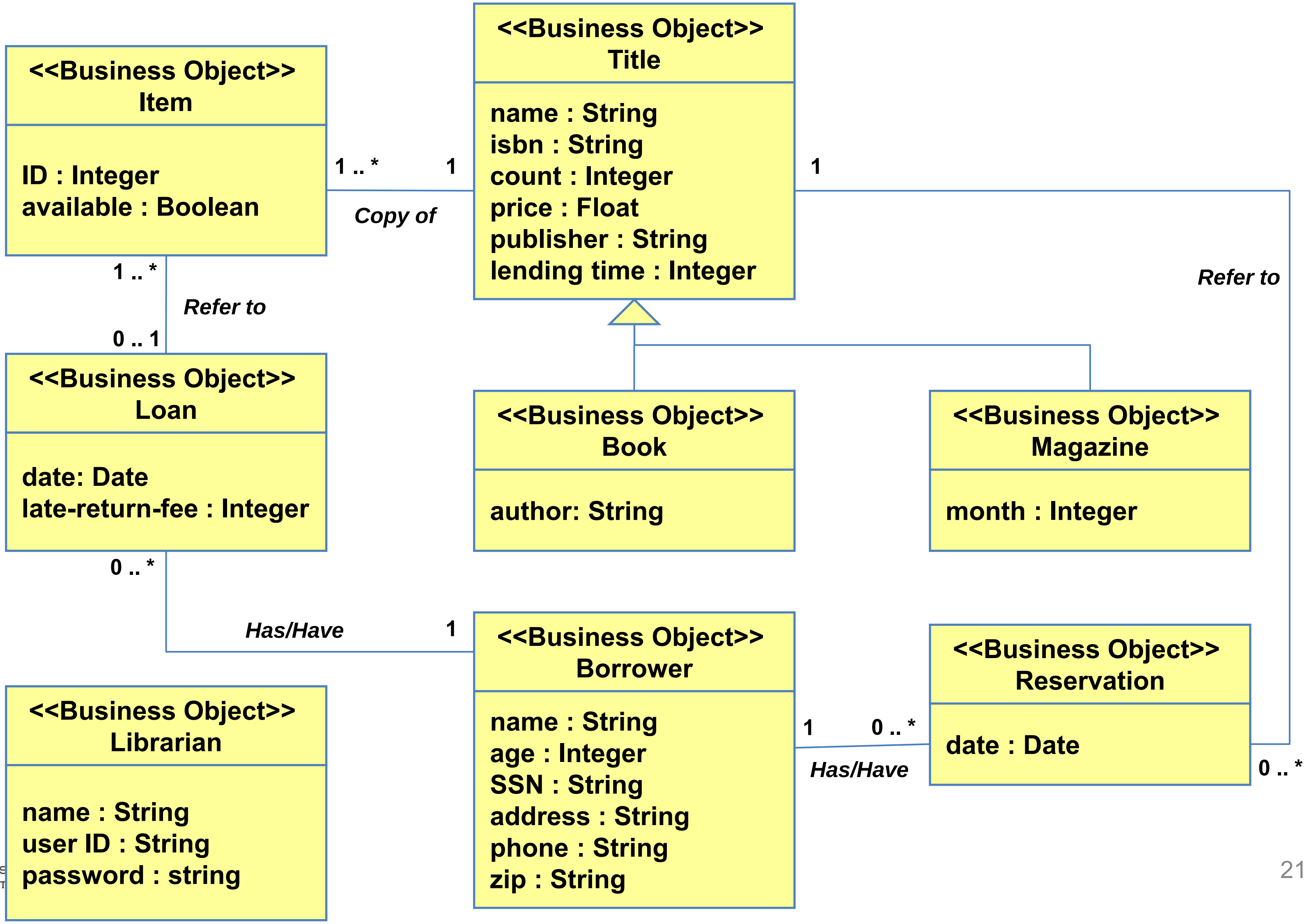
month : Integer

<<Business Object>>
Borrower

name : String
age : Integer
SSN : String
address : String
phone : String
zip : String

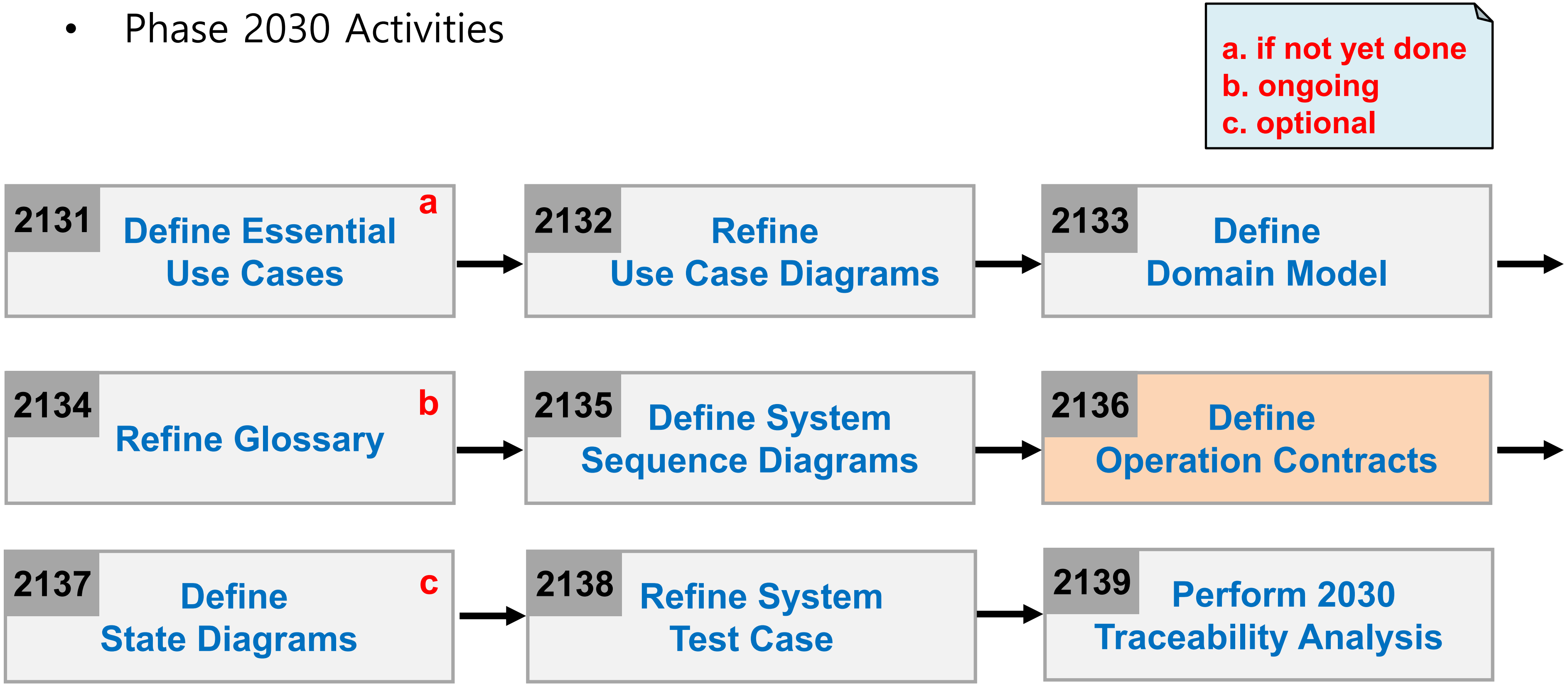
Activity 2035. Define Domain Model

- Step 8. Draw them in a conceptual class diagram



Activity 2036. Define Operation Contracts

- Phase 2030 Activities



a. if not yet done
b. ongoing
c. optional

2131 Define Essential Use Cases a
2132 Refine Use Case Diagrams
2133 Define Domain Model
2134 Refine Glossary b
2135 Define System Sequence Diagrams
2136 Define Operation Contracts
2137 Define State Diagrams c
2138 Refine System Test Case
2139 Perform 2030 Traceability Analysis

Activity 2036. Define Operation Contracts

Use Case	Name of Actor-Activated Event	System Operations
1. Make Reservation	1: Request making reservation( )	1. makeReservation( )
2. Remove Reservation	1: Request removing reservation( )	2. removeReservation( )
3. Lend Item	1: Request lending item( )	3. LendItem( )
4. Return Item	1: Request returning item( )	4. returnItem( )
5. Calculate Late-Return-Fee	N/A	N/A
6. Get Replacement Fee	1: Request replacement fee( )	5. getReplacementFee( )
7. Notify Availability	N/A	N/A
8. Add Title	1: Request adding title( )	6. addTitle( )
9. Remove Title	1: Request removing title( )	7. removeTitle( )
10. Update Title	1: Request updating title( )	8. updateTitle( )
11. Add Item	1: Request adding item( )	9. addItem( )
12. Remove Item	1: Request removing item( )	10. removeItem( )
13. Update Item	1: Request updating item( )	11. updateItem( )
14. Add Borrower	1: Request adding borrower( )	12. addBorrower( )
15. Remove Borrower	1: Request removing borrower( )	13. removeBorrower( )
16. Update Borrower	1: Request updating borrower( )	14. updateBorrower( )
17. Log-In	1: Input ID_Password( )	15. log-In( )
18. Log-Out	1: Exit( )	16. log-Out( )
19. Count Loans	1: Request count loans( )	17. countLoans( )

Activity 2036. Define Operation Contracts

Name	**makeReservation( )**
Responsibilities	Checks if title and borrower information exist, and creates a new reservation
Type	System
Cross References	System functions: R1.1, R2.1
Notes	
Exceptions	N/A
Output	Results from making the reservation
Pre-conditions	Title and Borrower information should be entered.
Post-conditions	A new reservation has created. Reservation.title has set to the title. Reservation.borrower has set to the borrower. The Reservation is associated with the Title. The Reservation is associated with the Borrower.

Activity 2036. Define Operation Contracts

Name	**removeReservation( )**
Responsibilities	Receive reservation information from a librarian and removes the reservation information
Type	System
Cross References	System functions: R1.2 Use case: “Remove Reservation”
Notes	
Exceptions	N/A
Output	Results from removing the reservation
Pre-conditions	The title should be reserved.
Post-conditions	The Reservation has deleted. The Reservation is associated with Title. (Why?) The Reservation is associated with Borrower. (Why?)

Activity 2036. Define Operation Contracts

Name	**lendItem( )**
Responsibilities	Checks whether the item to lend exists or not and lends the item
Type	System
Cross References	System functions: R1.2, R1.3 Use case: “Lent Item”, “Make Reservation”, “Remove Reservation”
Notes	
Exceptions	N/A
Output	Results from lending the item
Pre-conditions	The title of the item should exist.
Post-conditions	A new loan has created. The Loan is associated with the Item. The Loan is associated with the Borrower.

Activity 2036. Define Operation Contracts

Name	**returnItem( )**
Responsibilities	Receives an item’s information and returns the item
Type	System
Cross References	System functions: R1.4.1, R1.4.2 Use case: “Return Item”, “Calculate Late-Return-Fee”
Notes	
Exceptions	N/A
Output	Results from returning the item
Pre-conditions	Information of the item to return should be entered.
Post-conditions	Item.loan was set to the loan. The Item is associated with the Loan. The Loan has deleted. The Loan is associated with the Borrower.

Activity 2036. Define Operation Contracts

Name	**getReplacementFee( )**
Responsibilities	Requests to calculate for lost items or items in a poor condition
Type	System
Cross References	System functions: R1.5 Use case: “Get Replacement Fee”
Notes	
Exceptions	N/A
Output	Data on the calculated replacement fee
Pre-conditions	ISBM of the lost item should be entered.
Post-conditions	Item.lost has set to a true value. A count of the title has decremented. An available count of the title has decremented A Loan has deleted. (Why?)

Activity 2036. Define Operation Contracts

Name	**addTitle( )**
Responsibilities	Adds a new title
Type	System
Cross References	System functions: R2.1, R2.4 Use case: “Add Title”, “Add Item”
Notes	
Exceptions	If the title already exists, indicate an error.
Output	Results from returning the item
Pre-conditions	ISBM of the lost item should be entered.
Post-conditions	A new Title has created. Title.name has set to the name. Title.isbn has set to the isbn. Title.price has set to the price. Title.numOfCount has set to the numOfCount. Title.availableCount has set to the availableCount. Title.publisher has set to the publisher. Title.loanPeriod has set to the loanPeriod. Title.reservationCount has set to the reservationCount. Title is associated with Item.

Activity 2036. Define Operation Contracts

Name	**removeTitle( )**
Responsibilities	Removes an old book or magazine
Type	System
Cross References	System functions: R2.2 Use case: “Remove Title”
Notes	
Exceptions	If the title does not exist, indicate an error.
Output	Results from removing the title
Pre-conditions	Information of the title should be entered.
Post-conditions	The Title has deleted. The Title is associated with an Item, Reservation, Loan has deleted.

Activity 2036. Define Operation Contracts

Name	**updateTitle( )**
Responsibilities	Updates an old book or magazine
Type	System
Cross References	System functions: R2.3 Use case: “Update Title”
Notes	
Exceptions	If the title does not exist, indicate an error.
Output	Results from updating the title
Pre-conditions	Information of the title should be entered.
Post-conditions	The Title has updated. The Title is associated with an Item, Reservation, Loan has updated.

Activity 2036. Define Operation Contracts

Name	**removeItem( )**
Responsibilities	Removes an item
Type	System
Cross References	System functions: R2.5 Use case: “Remove Item”
Notes	
Exceptions	If the item’s title does not exist, indicate an error.
Output	Information of the removed item
Pre-conditions	Information of the title and item should be entered.
Post-conditions	The Item has removed. The Item is associated with Title, Loan has removed.

Activity 2036. Define Operation Contracts

Name	**updateItem( )**
Responsibilities	Updates an item
Type	System
Cross References	System functions: R2.6 Use case: “Update Item”
Notes	
Exceptions	If the item’s title does not exist, indicate an error.
Output	Information of the updated item
Pre-conditions	Information of the title and item should be entered.
Post-conditions	The Item has updated. The Item is associated with Title. The Item is associated with Loan.

Activity 2036. Define Operation Contracts

Name	addBorrower( )
Responsibilities	Adds a new borrower’s information
Type	System
Cross References	System functions: R3.1 Use case: “Add Borrower”
Notes	
Exceptions	If the borrower exists, indicate an error.
Output	Results from adding the new borrower
Pre-conditions	Information of the borrower should be entered.
Post-conditions	A new Borrower has created. Borrower.SSN has set to the SSN. Borrower.name has set to the name. Borrower.address has set to the address. Borrower.reservationCount has set to reservationCount. Borrower.loanCount has set to loanCount. Borrower is associated with Loan. Borrower is associated with Reservation.

Activity 2036. Define Operation Contracts

Name	**removeBorrower( )**
Responsibilities	Removes a borrower’s information
Type	System
Cross References	System functions: R3.2 Use case: “Remove Borrower”
Notes	
Exceptions	If the borrower does not exist, indicate an error.
Output	Results from removing the borrower
Pre-conditions	Information of the borrower should be entered.
Post-conditions	A Borrower has deleted. Borrower is associated with Loan, Reservation has deleted.

Activity 2036. Define Operation Contracts

Name	**updateBorrower( )**
Responsibilities	Updates a borrower’s information
Type	System
Cross References	System functions: R3.3 Use case: “Update Borrower”
Notes	
Exceptions	If the borrower does not exist, indicate an error.
Output	Results from updating the borrower
Pre-conditions	Information of the borrower should be entered.
Post-conditions	A Borrower has updated. Borrower is associated with Loan. Borrower is associated with Reservation.

Activity 2036. Define Operation Contracts

Name	**Log-In( )**
Responsibilities	Inputs an ID and Password of a librarian
Type	System
Cross References	System functions: R4.1 Use case: “Log-In”
Notes	Authentication information consists of ID and password
Exceptions	If the librarian does not exist, indicate an error.
Output	Approval information
Pre-conditions	Authentication information should be entered.
Post-conditions	The authentication information is associated with the librarian.

Activity 2036. Define Operation Contracts

Name	**Log-Out( )**
Responsibilities	Logouts from the system
Type	System
Cross References	System functions: R4.1 Use case: “Log-Out”
Notes	
Exceptions	N/A
Output	Exits from the system
Pre-conditions	-
Post-conditions	-

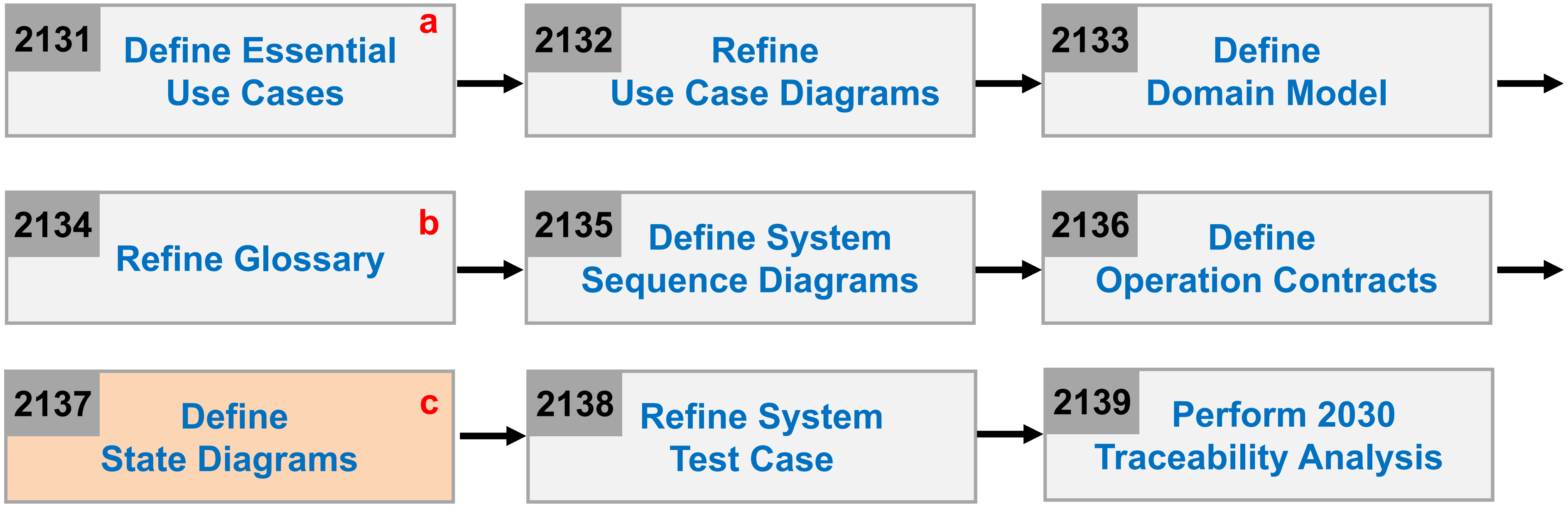
Activity 2036. Define Operation Contracts

Name	**countLoans( )**
Responsibilities	Requests for calculating a total counts of all titles checked
Type	System
Cross References	System functions: R5.1 Use case: “Count Loans”
Notes	
Exceptions	N/A
Output	Calculated data on the loans
Pre-conditions	It should calculate only the number of titles checked out.
Post-conditions	Number of titles checked out has calculated.

Activity 2037. Define State Diagrams

- Phase 2030 Activities

a. if not yet done
b. ongoing
c. optional



2131 Define Essential Use Cases a

2132 Refine Use Case Diagrams

2133 Define Domain Model

2134 Refine Glossary b

2135 Define System Sequence Diagrams

2136 Define Operation Contracts

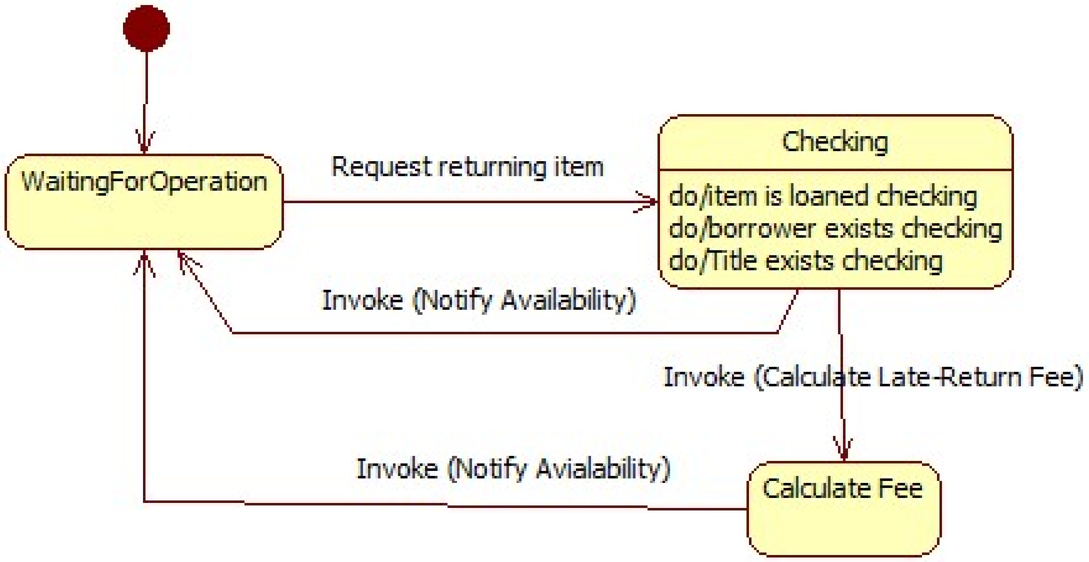
2137 Define State Diagrams c

2138 Refine System Test Case

2139 Perform 2030 Traceability Analysis

Activity 2037. Define State Diagrams

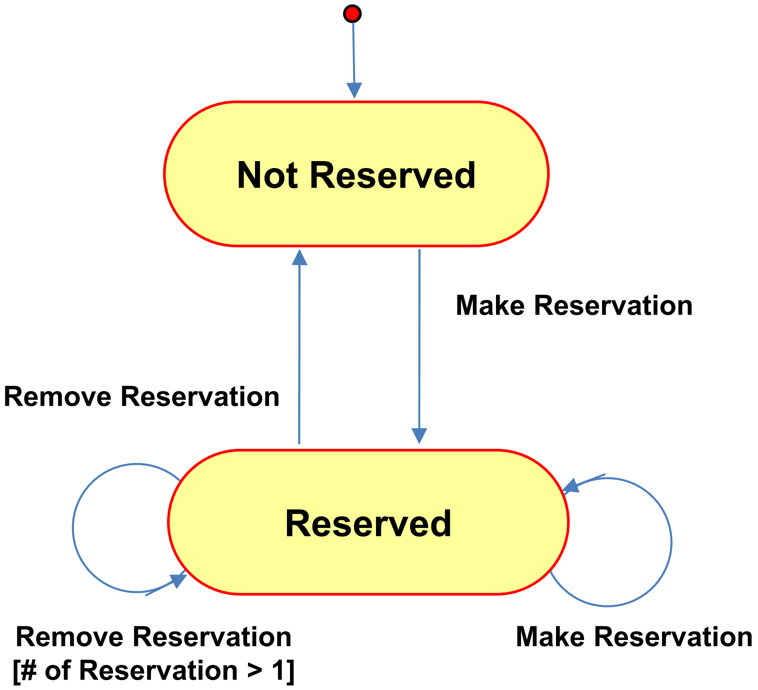
- State Diagram for Use case


WaitingForOperation
Request returning item
Checking
do/item is loaned checking
do/borrower exists checking
do/Title exists checking
Invoke (Notify Availability)
Invoke (Calculate Late-Return Fee)
Invoke (Notify Avialability)
Calculate Fee

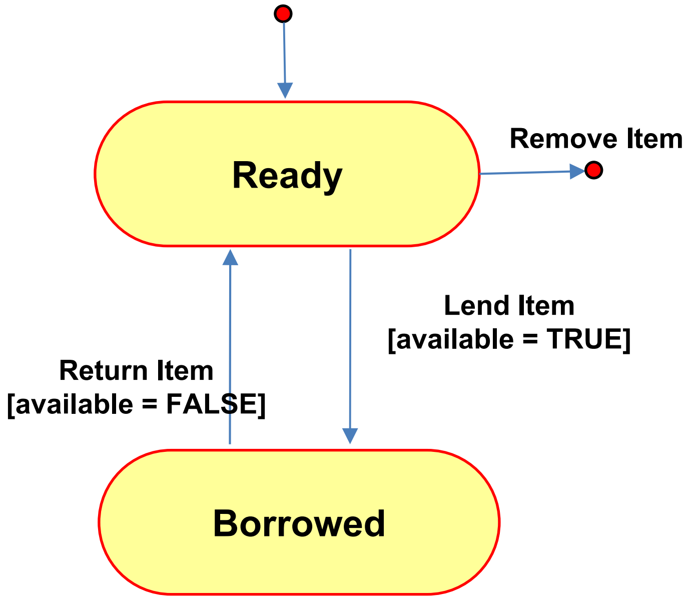
Activity 2037. Define State Diagrams

- State Diagram for Domain Model

< State Diagram for "Title" >



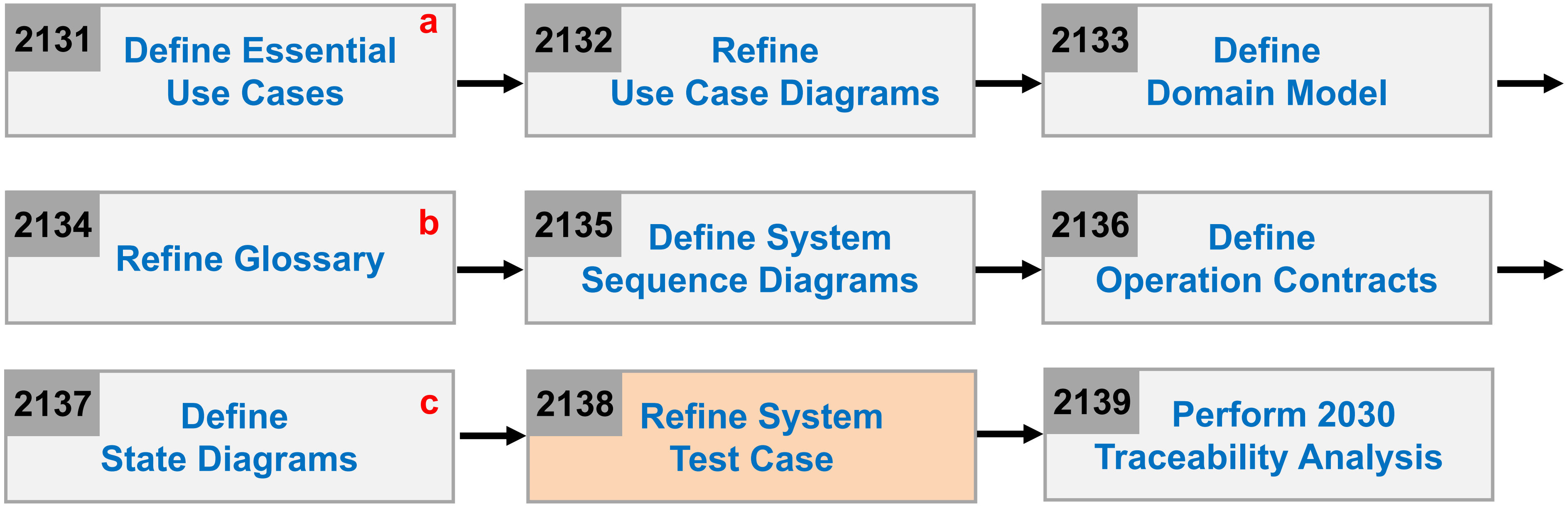
< State Diagram for "Item" >



Activity 2038. Refine System Test Case

- Phase 2030 Activities

a. if not yet done
b. ongoing
c. optional



2131 Define Essential Use Cases a

2132 Refine Use Case Diagrams

2133 Define Domain Model

2134 Refine Glossary b

2135 Define System Sequence Diagrams

2136 Define Operation Contracts

2137 Define State Diagrams c

2138 Refine System Test Case

2139 Perform 2030 Traceability Analysis

Phase 2038. Refine System Test Case

- Step 1. Identify important requirements

Ref. #	Function	Category
R1.1	**Make reservation**	Evident
R1.2	**Remove reservation**	Evident
R1.3	**Lend Item**	Evident
R1.4.1	**Return title**	Evident
R1.4.2	Calculate Late-Return-Fee	Hidden
R1.5	**Calculate Replacement Fee**	Evident
R1.6	Notify Availability	Hidden
R2.1	**Add title**	Evident
R2.2	**Remove title**	Evident
R2.3	**Update title**	Evident
R2.4	**Add items**	Evident
R2.5	**Remove item**	Evident
R2.6	**Update item**	Evident
R3.1	**Add borrower**	Evident
R3.2	**Remove borrower**	Evident
R3.3	**Update borrower**	Evident
R4.1	**Validates system access**	Evident
R5.1	**Compute total # of items checked out**	Evident

Activity 2038. Refine System Test Case

- Step 2. Develop system test cases with various system testing techniques
  - First, brute force testing

No.	Tests	Description
1	**Make reservation**	Correct한 borrower가 correct한 title 예약
2	**Make reservation**	Correct한 borrower가 incorrect한 title 예약
3	**Make reservation**	Correct한 borrower가 대여중인 title 예약
4	**Make reservation**	Incorrect한 borrower가 예약
5	**Remove reservation**	Correct한 borrower가 예약 취소
6	**Remove reservation**	Incorrect한 borrower가 예약 취소
7	**Lend Item**	Correct한 borrower가 대여 가능한 title 대여
8	**Lend Item**	Correct한 borrower가 incorrect한 title 대여
9	**Lend Item**	Correct한 borrower가 모두 대여중인 title 대여
10	**Lend Item**	Incorrect한 borrower가 대여
11	**Return title**	Borrower가 title 반납
12	**Return title**	Borrower가 연체된 title 반납
13	**Add title**	새 title 추가
14	**Remove title**	기존의 title 제거
15	**Remove title**	존재하지 않는 title 제거
16	**Update title**	Title 정보 update
17	**Add item**	Title item 추가
18	**Add item**	존재하지 않는 title의 item추가

Activity 2038. Refine System Test Case

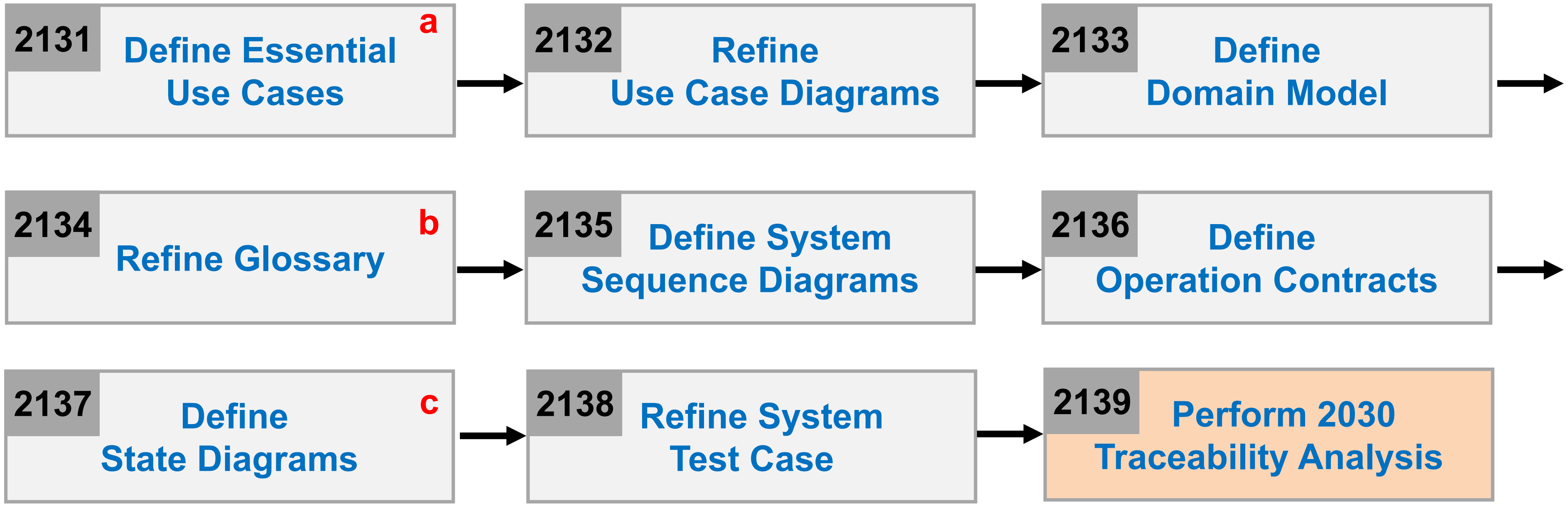
- Step 2. Develop system test cases with various system testing techniques
  - First, brute force testing

No.	Tests	Description
19	**Remove item**	Title의 item제거
20	**Remove item**	존재하지 않는 title의 item제거
21	**Update item**	올바른 item의 정보 update
22	**Update item**	Title에 존재하지 않는 item update
23	**Add borrower**	Borrower 추가
24	**Remove borrower**	Borrower 삭제
25	**Update borrower**	기존의 borrower update
26	**Update borrower**	삭제된 borrower update
27	**Validates system access**	Correct id/pw로 로그인
28	**Validates system access**	Incorrect id/pw로 로그인
29	**Validates system access**	로그아웃
30	**Compute total # of items checked out**	계산 시도

Activity 2039. Perform 2030 Traceability Analysis

- Phase 2030 Activities

a. if not yet done
b. ongoing
c. optional



2131 Define Essential Use Cases a

2132 Refine Use Case Diagrams

2133 Define Domain Model

2134 Refine Glossary b

2135 Define System Sequence Diagrams

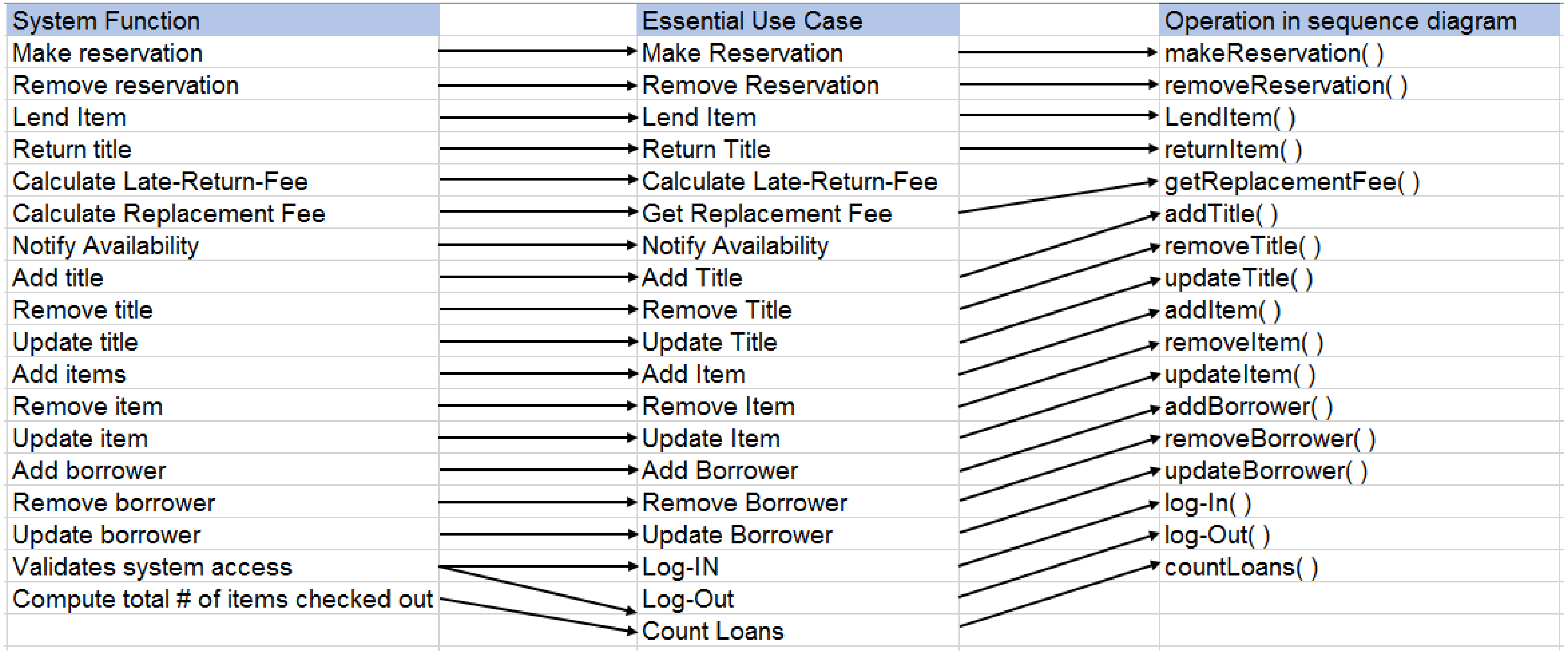
2136 Define Operation Contracts

2137 Define State Diagrams c

2138 Refine System Test Case

2139 Perform 2030 Traceability Analysis

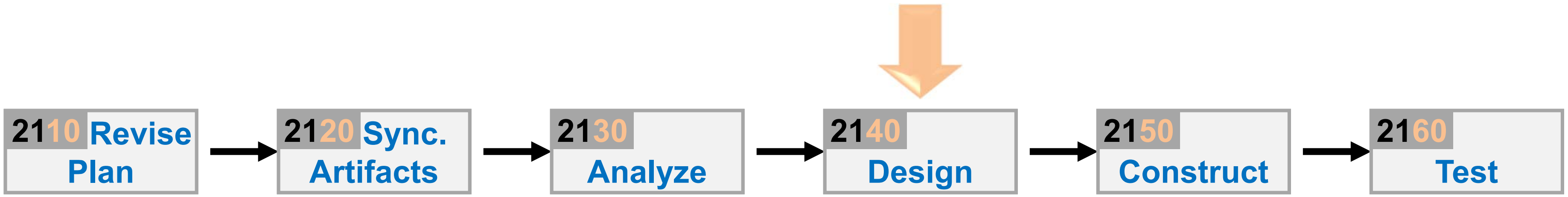
Activity 2039. Perform 2030 Traceability Analysis



System Function		Essential Use Case		Operation in sequence diagram
Make reservation	→	Make Reservation	→	makeReservation( )
Remove reservation	→	Remove Reservation	→	removeReservation( )
Lend Item	→	Lend Item	→	LendItem( )
Return title	→	Return Title	→	returnItem( )
Calculate Late-Return-Fee	→	Calculate Late-Return-Fee		getReplacementFee( )
Calculate Replacement Fee	→	Get Replacement Fee	↗	addTitle( )
Notify Availability	→	Notify Availability		removeTitle( )
Add title	→	Add Title	↗	updateTitle( )
Remove title	→	Remove Title	↗	addItem( )
Update title	→	Update Title	↗	removeItem( )
Add items	→	Add Item	↗	updateItem( )
Remove item	→	Remove Item	↗	addBorrower( )
Update item	→	Update Item	↗	removeBorrower( )
Add borrower	→	Add Borrower	↗	updateBorrower( )
Remove borrower	→	Remove Borrower	↗	log-In( )
Update borrower	→	Update Borrower	↗	log-Out( )
Validates system access	→ ↘	Log-IN	↗	countLoans( )
Compute total # of items checked out	↘	Log-Out	↗	
		Count Loans	↗	

Back

Phase 2040. Design



2110 Revise Plan

2120 Sync. Artifacts

2130 Analyze

2140 Design

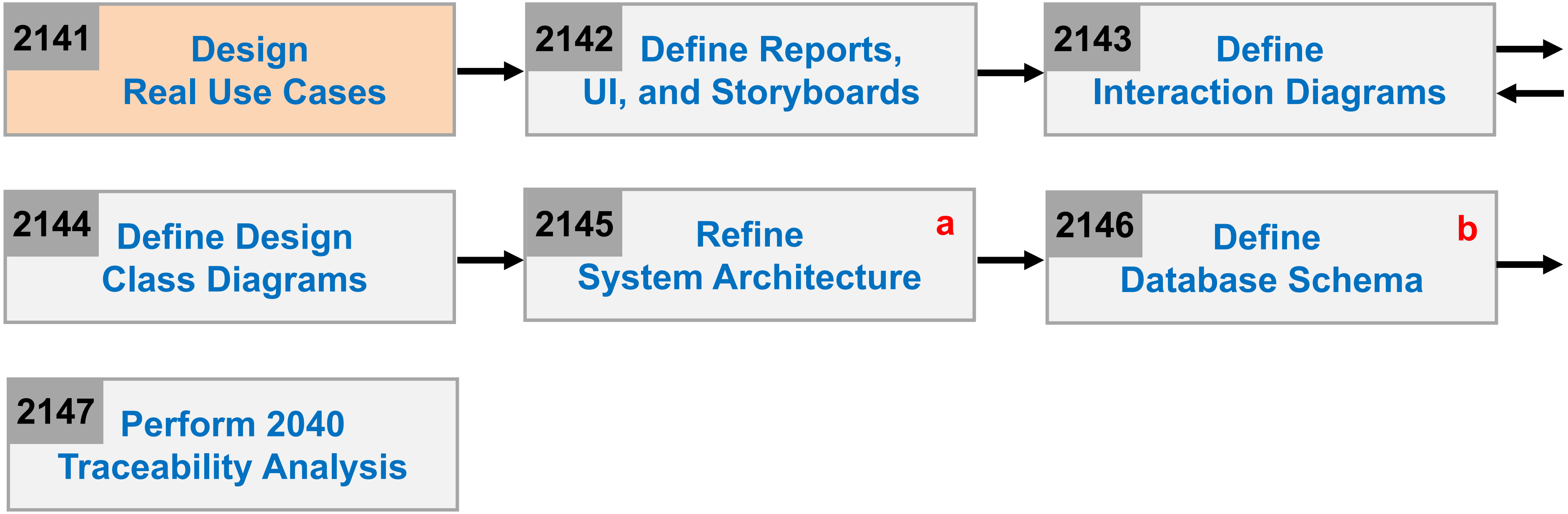
2150 Construct

2160 Test

Phase 2041. Design Real Use Cases

- 7 Activities

a. Varied order
b. optional



2141 Design Real Use Cases

2142 Define Reports, UI, and Storyboards

2143 Define Interaction Diagrams

2144 Define Design Class Diagrams

2145 Refine System Architecture a

2146 Define Database Schema b

2147 Perform 2040 Traceability Analysis

Phase 2041. Design Real Use Cases

- Make Reservation

Use Case	**1. Make Reservation**
Actor	Librarian
Purpose	Create a new reservation
Overview	(As in the business use case)
Type	Primary and Real
Cross Reference	System Functions: R1.1, R3.1 Use Case: “Add Borrower”
Pre-Requisites	A borrower should be registered.
Typical Courses of Events	(A) : Actor, (S) : System 1. (A) A librarian inputs an isbn and ssn of the title 2. (S) Find a corresponding title 3. (S) Find a corresponding borrower 4. (S) Create a new reservation 5. (S) Store the new reservation 6. (S) Increase reservationCount in the borrower 7. (S) Increase reservationCount in the title
Alternative Courses of Events	N/A
Exceptional Courses of Events	Line 2: If the title does not exist, display an error message. Line 3: If the borrower does not exist, display an error message.

Phase 2041. Design Real Use Cases

- Remove Reservation

Use Case	2. Remove Reservation
Actor	Librarian
Purpose	Remove a reservation information
Overview	(As in the business use case)
Type	Primary and Real
Cross Reference	System Functions: R1.2, R1.3 Use Case: “Lend Item”
Pre-Requisites	A borrower should be registered. A title should have been reserved.
Typical Courses of Events	(A) : Actor, (S) : System 1. (A) A librarian inputs an isbn of the title 2. (S) Find a corresponding reservation 3. (S) Remove the reservation 4. (S) Decrease reservationCount of the borrower 5. (S) Decrease reservationCount of the title
Alternative Courses of Events	N/A
Exceptional Courses of Events	Line 2: If the reservation doe not exist, display an error message.

Phase 2041. Design Real Use Cases

- Lend Item

Use Case	3. Lent Item
Actor	Librarian
Purpose	Lend items to a borrower
Overview	(As in the business use case)
Type	Primary and Real
Cross Reference	System Functions: R1.3, R1.2, R3.1 Use Cases: “Remove Reservation”, “Add Borrower”
Pre-Requisites	An item should exist.
Typical Courses of Events	(A) : Actor, (S) : System 1. (A) A librarian inputs an item’s ID and ssn of the borrower 2. (S) Find a corresponding borrower 3. (S) Find a corresponding item 4. (S) Create a new loan 5. (S) Store the new loan 6. (S) Set validLoan to true 7. (S) Increase loanCount of borrower 8. (S) Set available to false 9. (S) Decrease AvailableCount of the title
Alternative Courses of Events	N/A
Exceptional Courses of Events	Line 2: If the borrower does not exist, invoke “Add Borrower” use case.

Phase 2041. Design Real Use Cases

- Return Item

Use Case	4. Return Item
Actor	Librarian
Purpose	Return items loaned
Overview	(As in the business use case)
Type	Primary and Real
Cross Reference	System Functions: R1.4.1, R1.4.2, R1.6 Use Cases: “Calculate Late-Return-Fee”, “Notify Availability”
Pre-Requisites	An item should have been loaned.
Typical Courses of Events	(A) : Actor, (S) : System 1. (A) A librarian inputs an item’s ID 2. (S) Find a corresponding loan 3. (S) Get item information from the loan 4. (S) Get title information from the item 5. (S) Get loanPeriod from the title 6. (S) Compute calculateLateReturnFee 7. (S) Check reservationCount of the title. 8. (S) If the title is reserved, find the corresponding reservation 9. (S) Decrease loanCount of the loan. 10. (S) Decrease loanCount of the Borrower. 11. (S) Set validLoan of the borrower to false. 12. (S) Set available of the item to true. 13. (S) Increase AvailbaleCount of the title.
Alternative Courses of Events	N/A
Exceptional Courses of Events	Line 2: If the loan does not exist, display an error message.

Phase 2041. Design Real Use Cases

- Calculate Late-Return-Fee

Use Case	5. Calculate Late-Return-Fee
Actor	None
Purpose	Compute late-return fee for an item returned late
Overview	(As in the business use case)
Type	Primary and Real
Cross Reference	System Functions: R1.4.1, R1.4.2 Use Cases: “Return Item”
Pre-Requisites	Lending time of an item should have expired
Typical Courses of Events	(A) : Actor, (S) : System 1. (S) Calculate Late-Return-Fee of the item 2. (S) Display the Late-Return-Fee
Alternative Courses of Events	N/A
Exceptional Courses of Events	N/A

Phase 2041. Design Real Use Cases

- Get Replacement-Fee

Use Case	6. Get Replacement-Fee
Actor	Librarian
Purpose	Compute replacement-fee for a lost title
Overview	(As in the business use case)
Type	Primary and Real
Cross Reference	System Functions: R1.5 Use Cases: -
Pre-Requisites	A title should be lost.
Typical Courses of Events	(A) : Actor, (S) : System 1. (A) A librarian inputs an item's ID 1. (S) Find a corresponding loan 2. (S) Get an item from the loan 3. (S) Get a title from the item 4. (S) Get price of the title 5. (S) Compute replacementFee 6. (S) Set validLoan to false 7. (S) Update the loan 8. (S) Decrease loanCount of the borrower. 9. (S) Set the isborrowed of the item to false. 10. (S) Decrease numOfItem of the title.
Alternative Courses of Events	N/A
Exceptional Courses of Events	Line 2: If the loan doe not exist, display an error message.

Phase 2041. Design Real Use Cases

- Notify Availability

Use Case	7. Notify Availability
Actor	None
Purpose	Notify availability of a reserved item
Overview	(As in the business use case)
Type	Primary and Real
Cross Reference	System Functions: R1.4.1, R1.6, R2.4 Use Cases: “Return Item”, “Add Item”
Pre-Requisites	An item should have been returned or a new item should have been added.
Typical Courses of Events	(A) : Actor, (S) : System 1. (S) Print a post-card
Alternative Courses of Events	N/A
Exceptional Courses of Events	N/A

Phase 2041. Design Real Use Cases

- Add Title

Use Case	**8. Add Title**
Actor	Librarian
Purpose	Register a new title
Overview	(As in the business use case)
Type	Primary and Real
Cross Reference	System Functions: R2.1, R2.4 Use Case: “Add Item”
Pre-Requisites	N/A
Typical Courses of Events	(A) : Actor, (S) : System 1. (A) A librarian inputs title information such as name, isbn, price, publisher, loanPeriod (Book: author, Magazine:month, publishCycle) 2. (S) Find a corresponding title 3. (S) Create a new title 4. (S) Store the new title 5. (S) Invoke “Add Item”
Alternative Courses of Events	N/A
Exceptional Courses of Events	Line 1: If the title already exists, display an error message.

Phase 2041. Design Real Use Cases

- Remove Title

Use Case	9. Remove Title
Actor	Librarian
Purpose	Delete information of a title
Overview	(As in the business use case)
Type	Primary and Real
Cross Reference	System Functions: R2.2 Use Case: -
Pre-Requisites	N/A
Typical Courses of Events	(A) : Actor, (S) : System 1. (A) A librarian inputs a title's isbn to remove 2. (S) Find a corresponding title 3. (S) Check if the corresponding title is reserved. 4. (S) If the title is reserved, Remove the reservation 5. (S) Check the item of the title is loaned. 6. (S) Remove the title
Alternative Courses of Events	N/A
Exceptional Courses of Events	Line 2: If the title does not exist, display an error message. Line 5: If the item of the title is loaned, display an error mesasge.

Phase 2041. Design Real Use Cases

- Update Title

Use Case	10. Update Title
Actor	Librarian
Purpose	Update information of a title
Overview	(As in the business use case)
Type	Primary and Real
Cross Reference	System Functions: R2.3 Use Case: -
Pre-Requisites	N/A
Typical Courses of Events	(A) : Actor, (S) : System 1. (A) A librarian inputs a title's isbn and information of the title to change 2. (S) Find a corresponding title 3. (S) Update the title
Alternative Courses of Events	N/A
Exceptional Courses of Events	Line 2: If the item does not exist, display "Not Existing Title". Error message. Line 3: If the isbn is changed, then update items too,

Phase 2041. Design Real Use Cases

- Add Item

Use Case	11. Add Item
Actor	Librarian
Purpose	Add a new item
Overview	(As in the business use case)
Type	Primary and Real
Cross Reference	System Functions: R2.4 Use Cases: “Add Title”
Pre-Requisites	N/A
Typical Courses of Events	(A) : Actor, (S) : System 1. (A) A librarian inputs an item’s id 2. (S) Find a corresponding title 3. (S) Get an item’s ID from the title 4. (S) Create a new item 5. (S) Store the new item 6. (S) Increase numOfItem of the title 7. (S) Increase availablecount of the item
Alternative Courses of Events	N/A
Exceptional Courses of Events	Line 2: Line 2: If the title does not exist, display an error message.

Phase 2041. Design Real Use Cases

- Remove Item

Use Case	12. Remove Item
Actor	Librarian
Purpose	Remove information of an item
Overview	(As in the business use case)
Type	Primary and Real
Cross Reference	System Functions: R2.1, R2.5 Use Case: “Remove Title”
Pre-Requisites	N/A
Typical Courses of Events	(A) : Actor, (S) : System 1. (A) A librarian inputs an item’s ID 2. (S) Find a corresponding item 3. (S) Check if the item is borrowed 4. (S) If the item is borrowed, decrease numOfItem of the title 5. (S) Decrease availableCount of the title 6. (S) Remove the item
Alternative Courses of Events	N/A
Exceptional Courses of Events	Line 2: If the item does not exist, display an error message. Line 3: If the item was already borrowed, display an error message

Phase 2041. Design Real Use Cases

- Update Item

Use Case	13. Update Item
Actor	Librarian
Purpose	Update information of an item
Overview	(As in the business use case)
Type	Primary and Real
Cross Reference	System Functions: R2.6 Use Case: -
Pre-Requisites	N/A
Typical Courses of Events	(A) : Actor, (S) : System 1. (A) A librarian inputs the item's id and information to change 2. (S) Find A corresponding item 3. (S) Update the item 4. (S) If a lost of the item is true, decrease numOfItem of the title. 5. (S) Decrease the availableCount of the title. 6. (S) If a lost of the item is false, increase numOfItem of the title. (What? Only for these cases "Update Item" are used?) 7. (S) Increase availableCount of the title
Alternative Courses of Events	N/A
Exceptional Courses of Events	Line 2: If the item does not exist, display an error message.

Phase 2041. Design Real Use Cases

- Add Borrower

Use Case	**14. Add Borrower**
Actor	Librarian
Purpose	Register a new borrower
Overview	(As in the business use case)
Type	Primary and Real
Cross Reference	System Functions: R1.1, R1.3, R3.1 Use Cases: “Make Reservation”, “Lend Item”
Pre-Requisites	N/A
Typical Courses of Events	(A) : Actor, (S) : System 1. (A) A librarian inputs a borrower’s name, ssn, and address. 2. (S) Find a corresponding borrower 3. (S) Create a new borrower 4. (S) Store the new borrower
Alternative Courses of Events	N/A
Exceptional Courses of Events	Line 2: If the borrower exists already, display an error message.

Phase 2041. Design Real Use Cases

- Remove Borrower

Use Case	15. Remove Borrower
Actor	Librarian
Purpose	Remove information of a borrower
Overview	(As in the business use case)
Type	Primary and Real
Cross Reference	System Functions: R3.2 Use Case: -
Pre-Requisites	N/A
Typical Courses of Events	(A) : Actor, (S) : System 1. (A) A librarian inputs the borrower's ssn 2. (S) Find a corresponding borrower 3. (S) Find a loan of the borrower 4. (S) If the loan is invalid, find a reservation 5. (S) Get the title of the reservation 6. (S) Decrease reservationCount of the title 7. (S) Remove borrower
Alternative Courses of Events	N/A
Exceptional Courses of Events	Line 2: If the borrower does not exist, display an error message. Line 3: If the loan is still valid, display an error message.

Phase 2041. Design Real Use Cases

- Update Borrower

Use Case	16. Update Borrower
Actor	Librarian
Purpose	Update information of a borrower
Overview	(As in the business use case)
Type	Primary and Real
Cross Reference	System Functions: R3.3 Use Case: -
Pre-Requisites	N/A
Typical Courses of Events	(A) : Actor, (S) : System 1. (A) A librarian inputs a borrower’s ssn and information to change 2. (S) Find a corresponding borrower 3. (S) Update the borrower
Alternative Courses of Events	N/A
Exceptional Courses of Events	Line 2: If the borrower does not exist, display an error message.

Phase 2041. Design Real Use Cases

- Log-In

Use Case	17. Log-In
Actor	Librarian
Purpose	Check access authority of a librarian
Overview	(As in the business use case)
Type	Secondary and Real
Cross Reference	System Functions: R4.1 Use Case: -
Pre-Requisites	A librarian should have user name and password.
Typical Courses of Events	(A) : Actor, (S) : System 1. (A) A librarian inputs an userID and password 2. (S) Check if the userID and password are correct
Alternative Courses of Events	N/A
Exceptional Courses of Events	Line 2: If the userID and password are not correct, display an error message.

Phase 2041. Design Real Use Cases

- Log-Out

Use Case	18. Log-Out
Actor	Librarian
Purpose	Exit the library management system
Overview	(As in the business use case)
Type	Secondary and Essential
Cross Reference	System Functions: R4.1 Use Case: -
Pre-Requisites	A librarian should have user name and password.
Typical Courses of Events	(A) : Actor, (S) : System 1. (A) A librarian selects “LogOut” 2. (S) Check if the userID is correct and then exit the system
Alternative Courses of Events	N/A
Exceptional Courses of Events	Line 2: If the userID is incorrect, display an error message.

Phase 2041. Design Real Use Cases

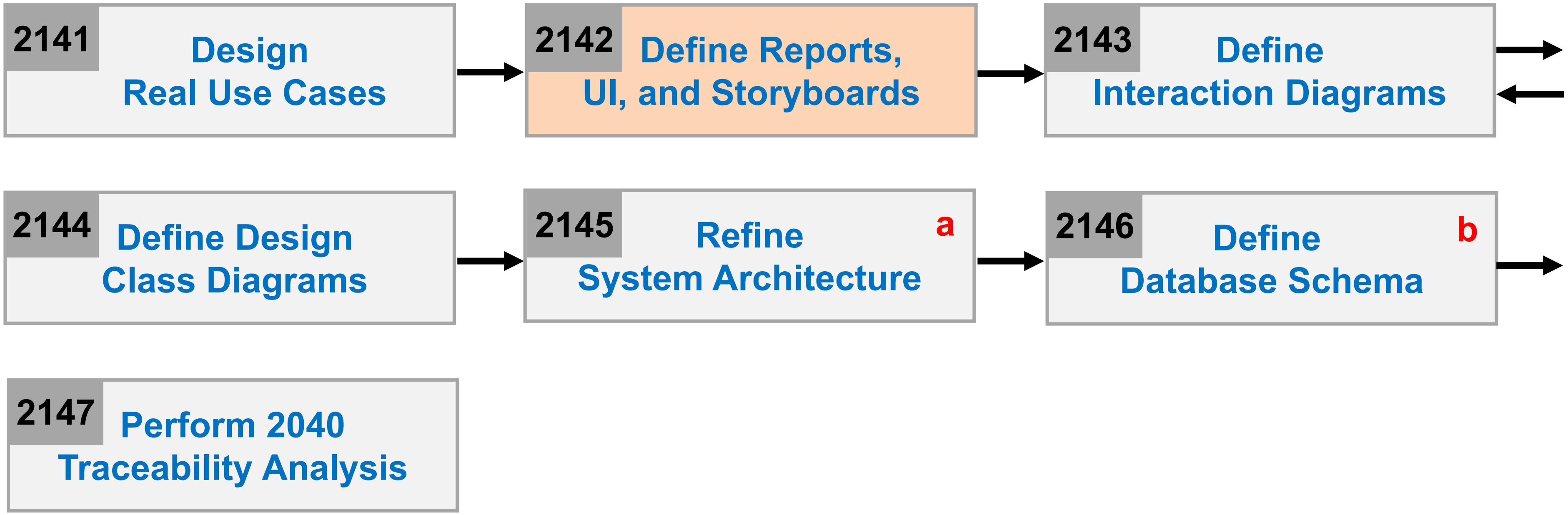
- Count Loans

Use Case	19. Count Loans
Actor	Librarian
Purpose	Compute total count of the titles checked out
Overview	(As in the business use case)
Type	Secondary and Essential
Cross Reference	System Functions: R5.1 Use Case: -
Pre-Requisites	A librarian should have user name and password.
Typical Courses of Events	(A) : Actor, (S) : System 1. (A) A librarian requests loan count 2. (S) Get numOfLoan of the loan
Alternative Courses of Events	N/A
Exceptional Courses of Events	N/A (Really?)

Phase 2042. Define Reports, UI, and Storyboards

- 7 Activities

a. Varied order
b. optional



2141 Design Real Use Cases

2142 Define Reports, UI, and Storyboards

2143 Define Interaction Diagrams

2144 Define Design Class Diagrams

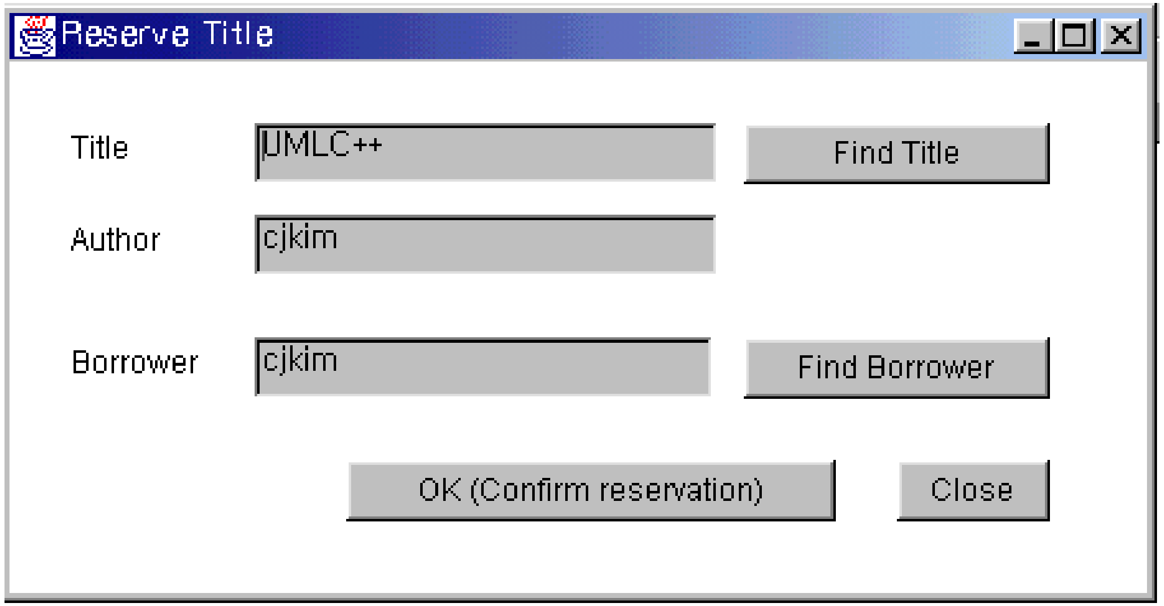
2145 Refine System Architecture a

2146 Define Database Schema b

2147 Perform 2040 Traceability Analysis

Phase 2042. Define Reports, UI, and Storyboards

- Make Reservation



Reserve Title

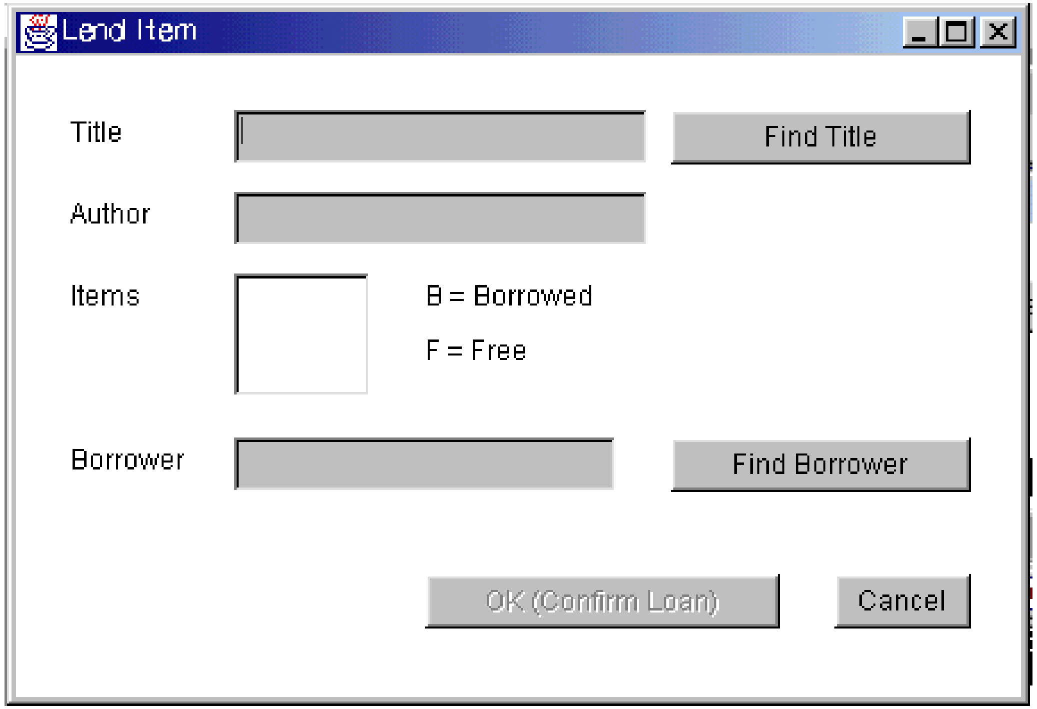
Title: UMLC++

Author: cjkim

Borrower: cjkim

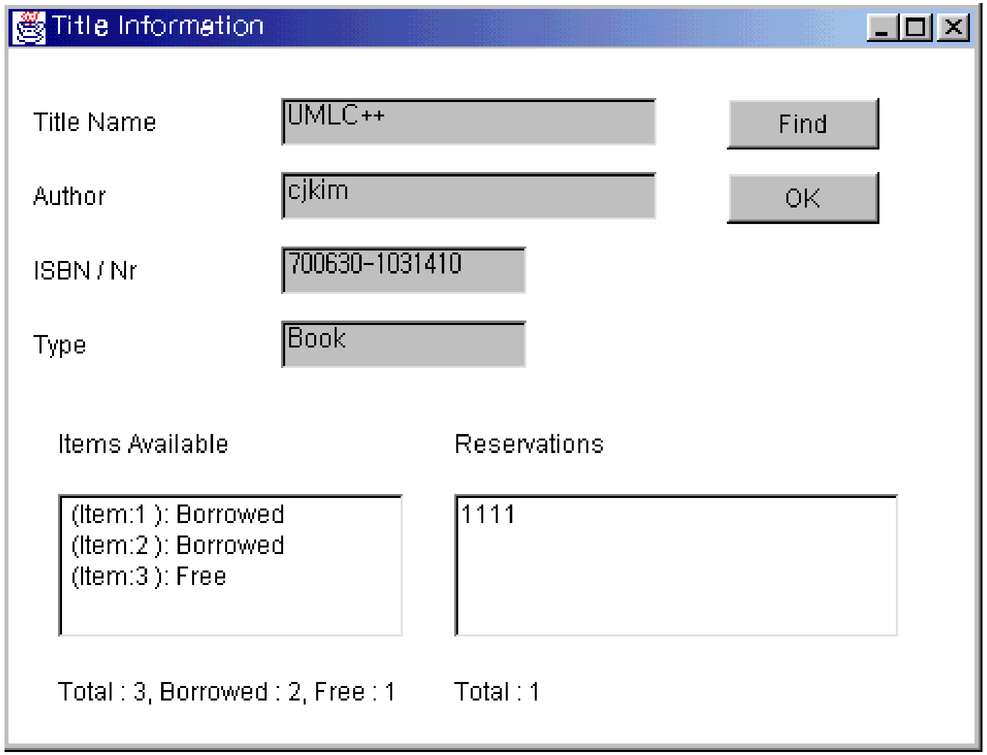
Phase 2042. Define Reports, UI, and Storyboards

- Lent Item



Phase 2042. Define Reports, UI, and Storyboards

- Count Loans



Title Name: UMLC++

Author: cjkim

ISBN / Nr: 700630-1031410

Type: Book

Items Available

(Item:1): Borrowed
(Item:2): Borrowed
(Item:3): Free

Total : 3, Borrowed : 2, Free : 1

Reservations

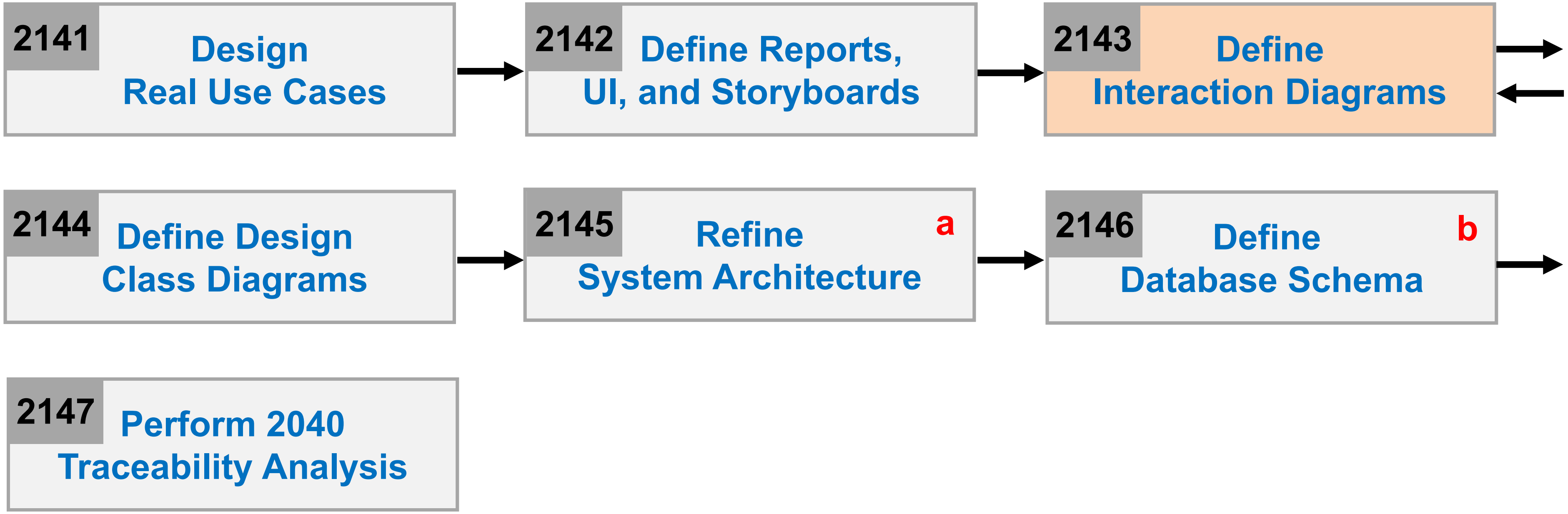
1111

Total : 1

Phase 2043. Define Interaction Diagrams

- 7 Activities

a. Varied order
b. optional



2141 Design Real Use Cases

2142 Define Reports, UI, and Storyboards

2143 Define Interaction Diagrams

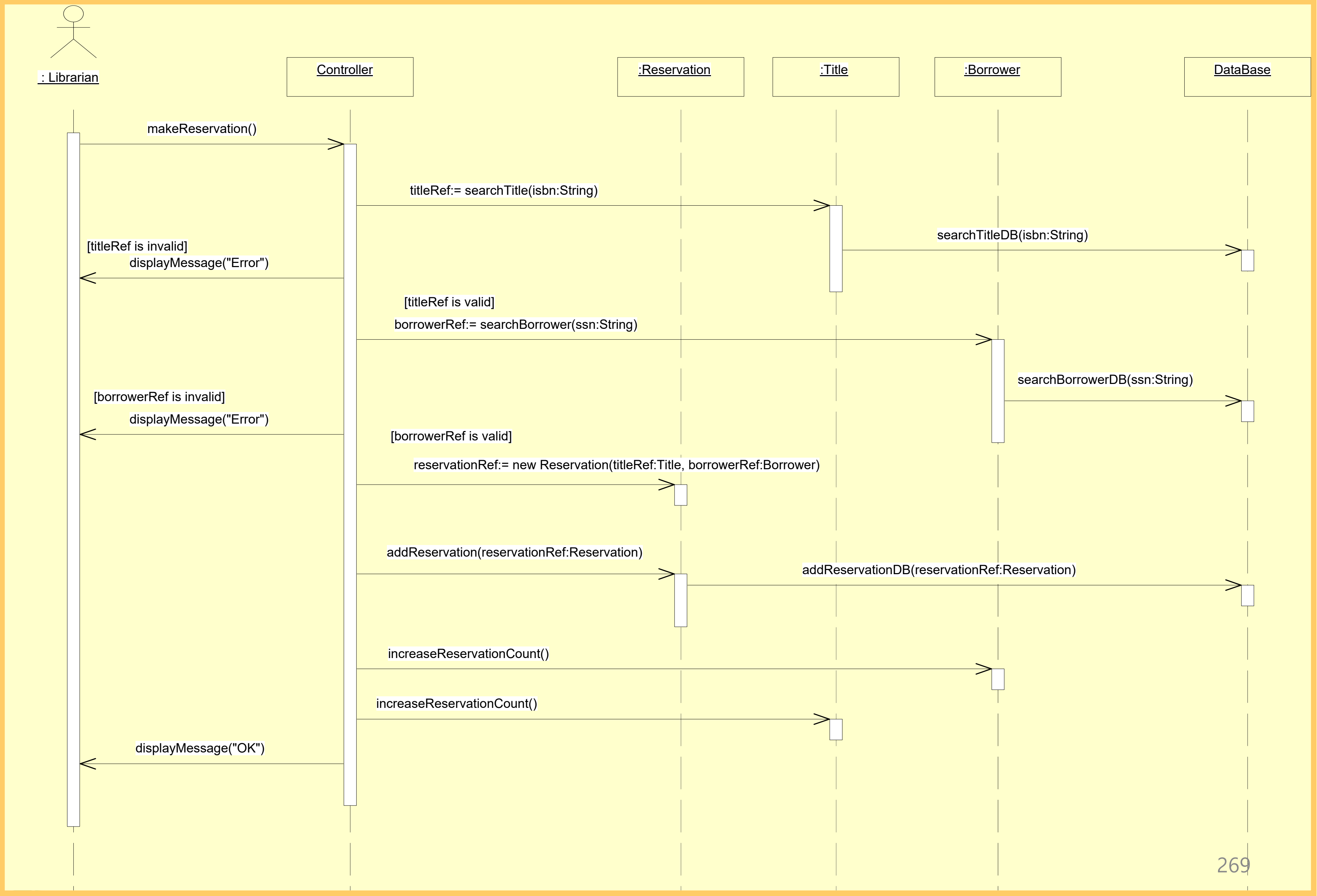
2144 Define Design Class Diagrams

2145 Refine System Architecture a

2146 Define Database Schema b

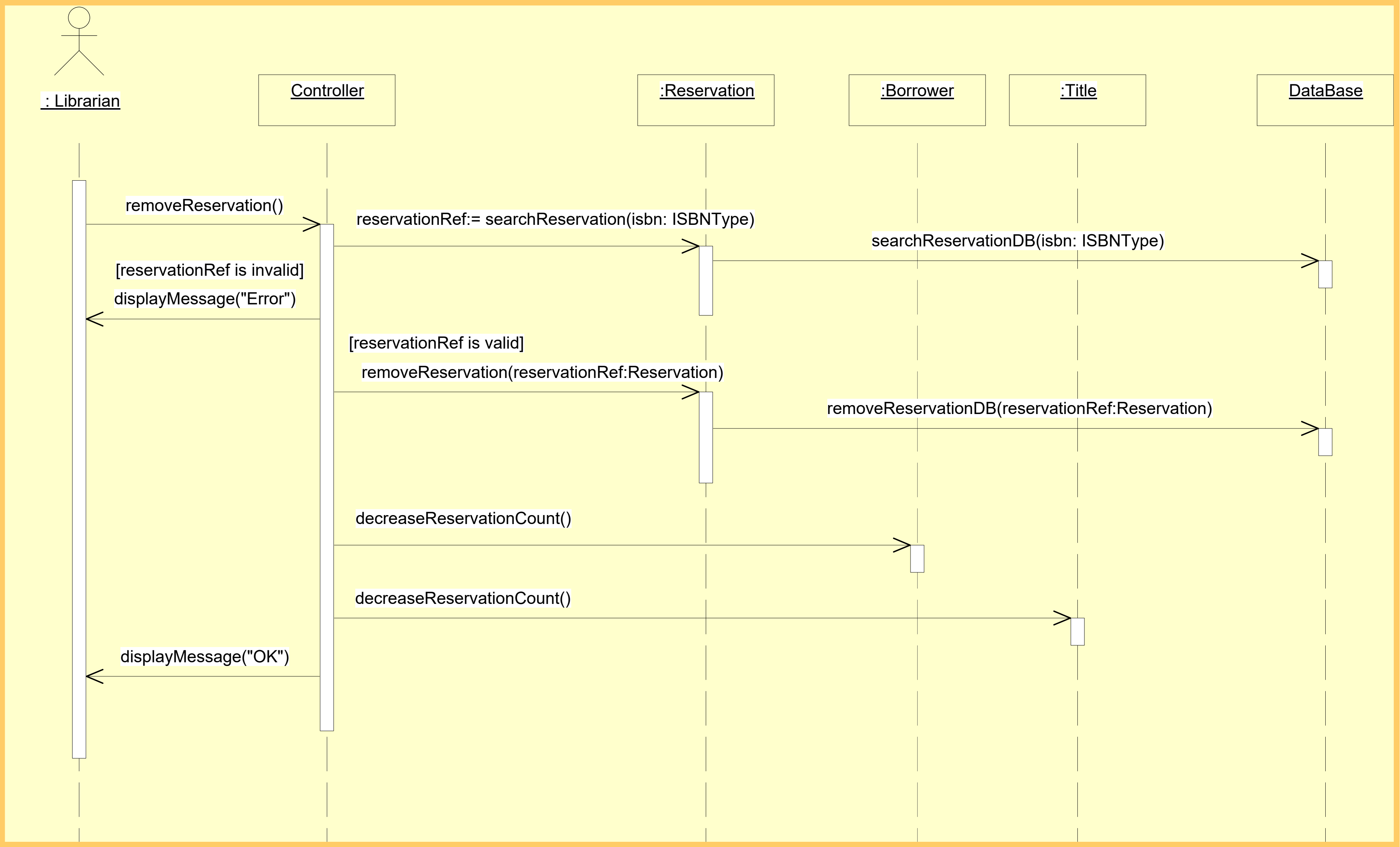
2147 Perform 2040 Traceability Analysis

1. Make Reservation

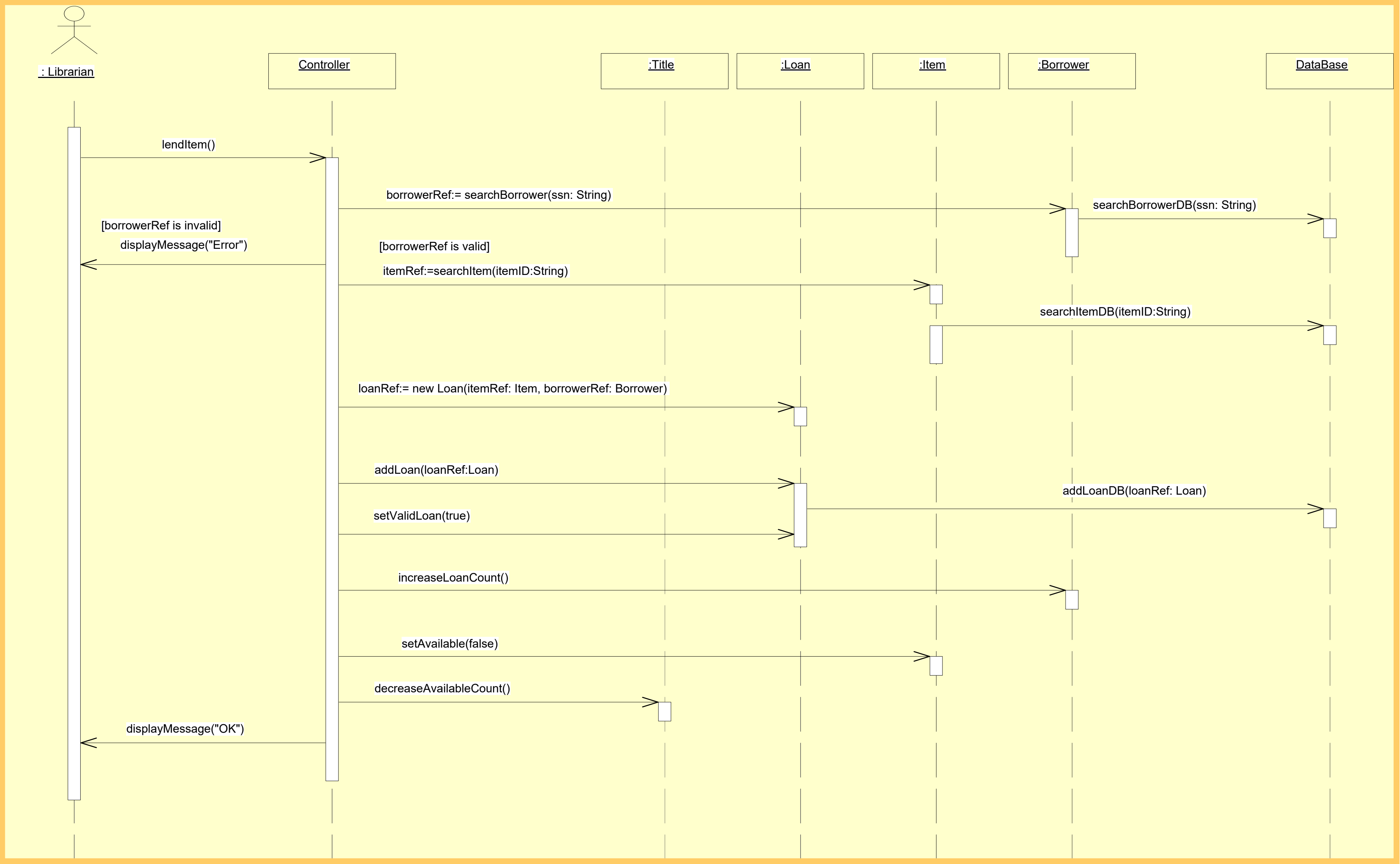


: Librarian
Controller
:Reservation
:Title
:Borrower
DataBase
makeReservation()
titleRef:= searchTitle(isbn:String)
searchTitleDB(isbn:String)
[titleRef is invalid]
displayMessage("Error")
[titleRef is valid]
borrowerRef:= searchBorrower(ssn:String)
searchBorrowerDB(ssn:String)
[borrowerRef is invalid]
displayMessage("Error")
[borrowerRef is valid]
reservationRef:= new Reservation(titleRef:Title, borrowerRef:Borrower)
addReservation(reservationRef:Reservation)
addReservationDB(reservationRef:Reservation)
increaseReservationCount()
increaseReservationCount()
displayMessage("OK")

2. Remove Reservation

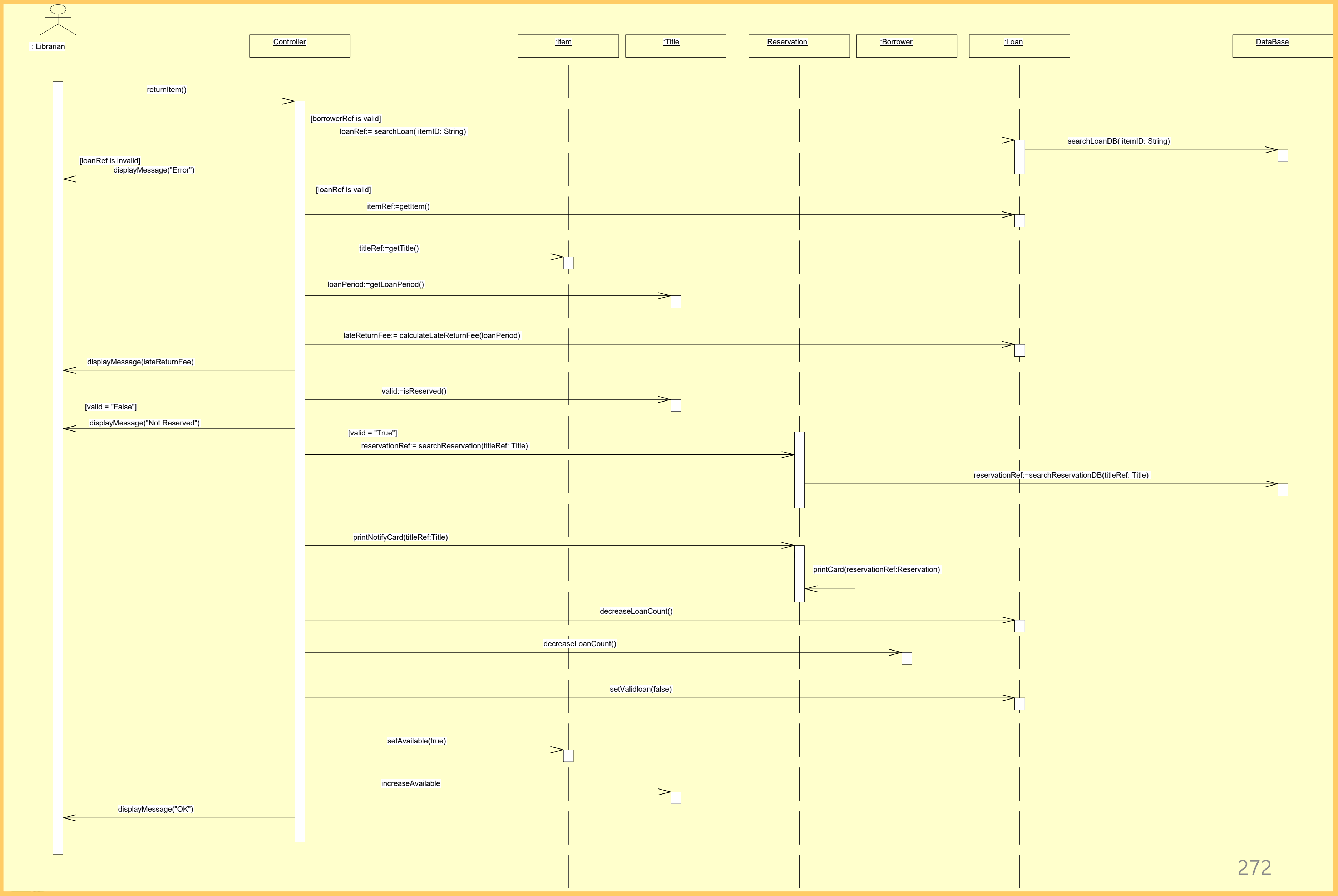

: Librarian
Controller
:Reservation
:Borrower
:Title
DataBase
removeReservation()
reservationRef:= searchReservation(isbn: ISBNType)
searchReservationDB(isbn: ISBNType)
[reservationRef is invalid]
displayMessage("Error")
[reservationRef is valid]
removeReservation(reservationRef:Reservation)
removeReservationDB(reservationRef:Reservation)
decreaseReservationCount()
decreaseReservationCount()
displayMessage("OK")

3. Lend Item



: Librarian
Controller
:Title
:Loan
:Item
:Borrower
DataBase
lendItem()
borrowerRef:= searchBorrower(ssn: String)
searchBorrowerDB(ssn: String)
[borrowerRef is invalid]
displayMessage("Error")
[borrowerRef is valid]
itemRef:=searchItem(itemID:String)
searchItemDB(itemID:String)
loanRef:= new Loan(itemRef: Item, borrowerRef: Borrower)
addLoan(loanRef:Loan)
addLoanDB(loanRef: Loan)
setValidLoan(true)
increaseLoanCount()
setAvailable(false)
decreaseAvailableCount()
displayMessage("OK")

4. Return Item



: Librarian

Controller

:Item

:Title

Reservation

:Borrower

:Loan

DataBase

returnItem()

[borrowerRef is valid]
loanRef:= searchLoan(itemID: String)

searchLoanDB(itemID: String)

[loanRef is invalid]
displayMessage("Error")

[loanRef is valid]

itemRef:=getItem()

titleRef:=getTitle()

loanPeriod:=getLoanPeriod()

lateReturnFee:= calculateLateReturnFee(loanPeriod)

displayMessage(lateReturnFee)

valid:=isReserved()

[valid = "False"]
displayMessage("Not Reserved")

[valid = "True"]
reservationRef:= searchReservation(titleRef: Title)

reservationRef:=searchReservationDB(titleRef: Title)

printNotifyCard(titleRef:Title)

printCard(reservationRef:Reservation)

decreaseLoanCount()

decreaseLoanCount()

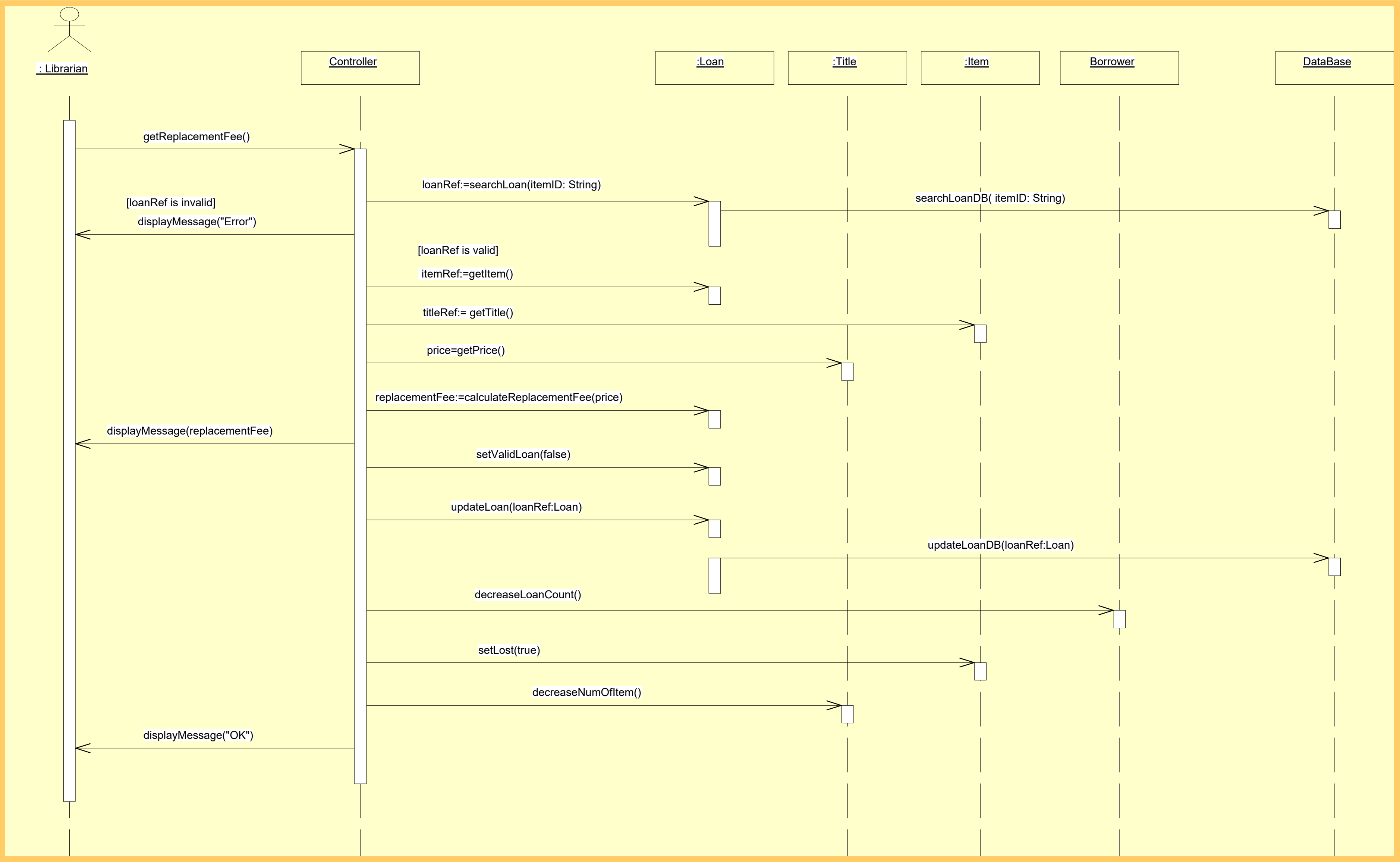
setValidloan(false)

setAvailable(true)

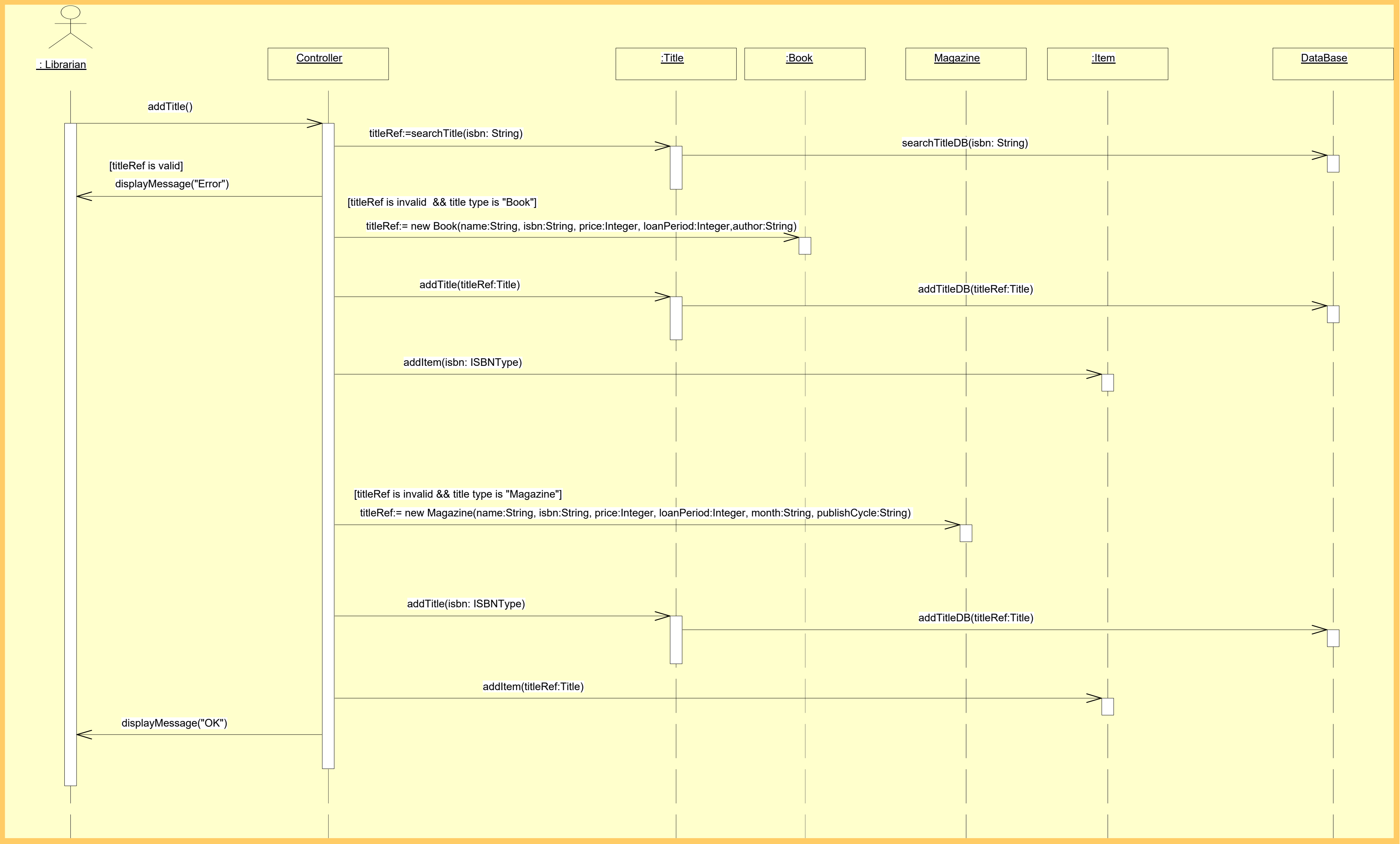
increaseAvailable

displayMessage("OK")

6. Get Replacement-Fee



8. Add Title



: Librarian

Controller

:Title

:Book

Magazine

:Item

DataBase

addTitle()

titleRef:=searchTitle(isbn: String)

searchTitleDB(isbn: String)

[titleRef is valid]

displayMessage("Error")

[titleRef is invalid && title type is "Book"]

titleRef:= new Book(name:String, isbn:String, price:Integer, loanPeriod:Integer,author:String)

addTitle(titleRef:Title)

addTitleDB(titleRef:Title)

addItem(isbn: ISBNType)

[titleRef is invalid && title type is "Magazine"]

titleRef:= new Magazine(name:String, isbn:String, price:Integer, loanPeriod:Integer, month:String, publishCycle:String)

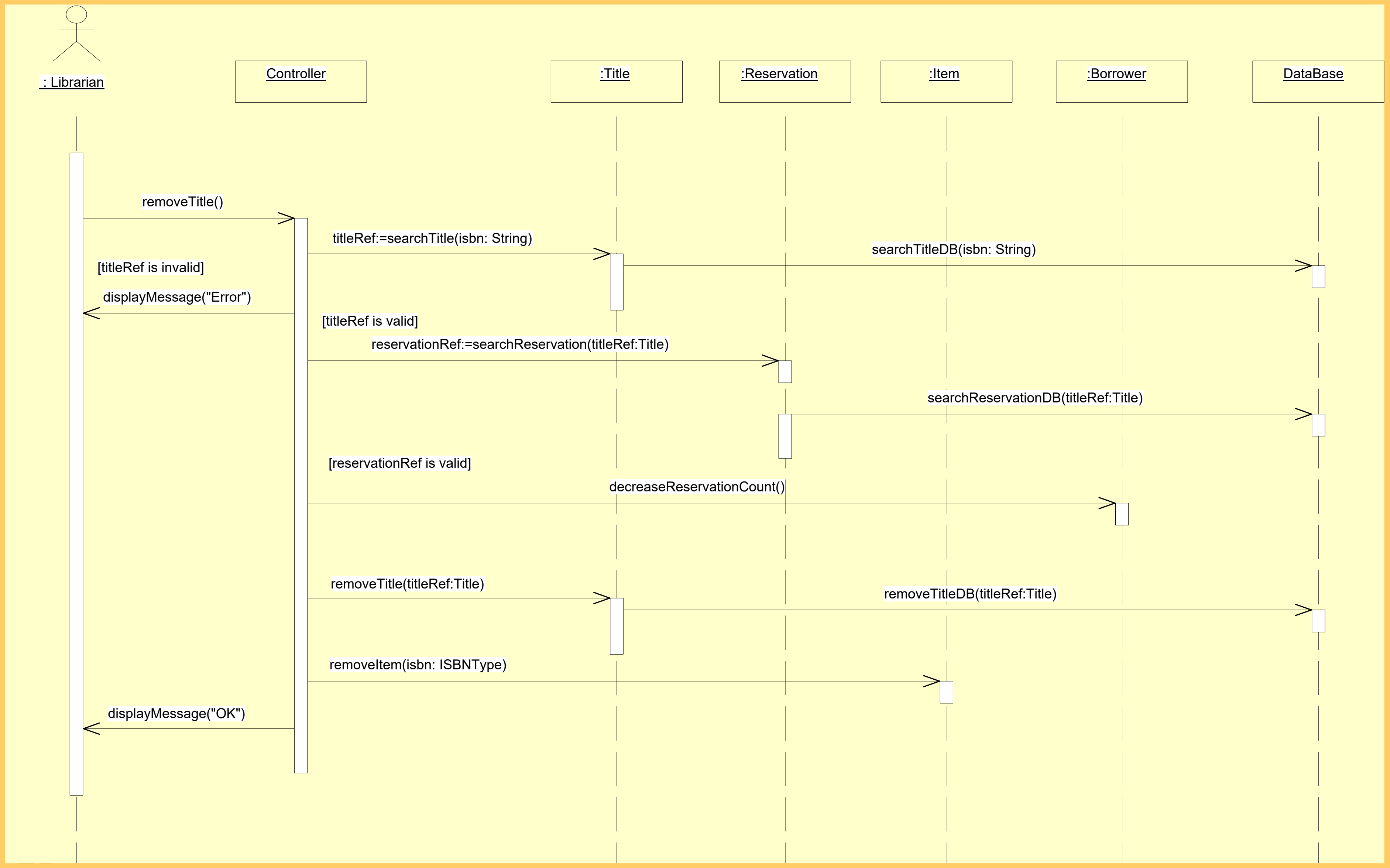
addTitle(isbn: ISBNType)

addTitleDB(titleRef:Title)

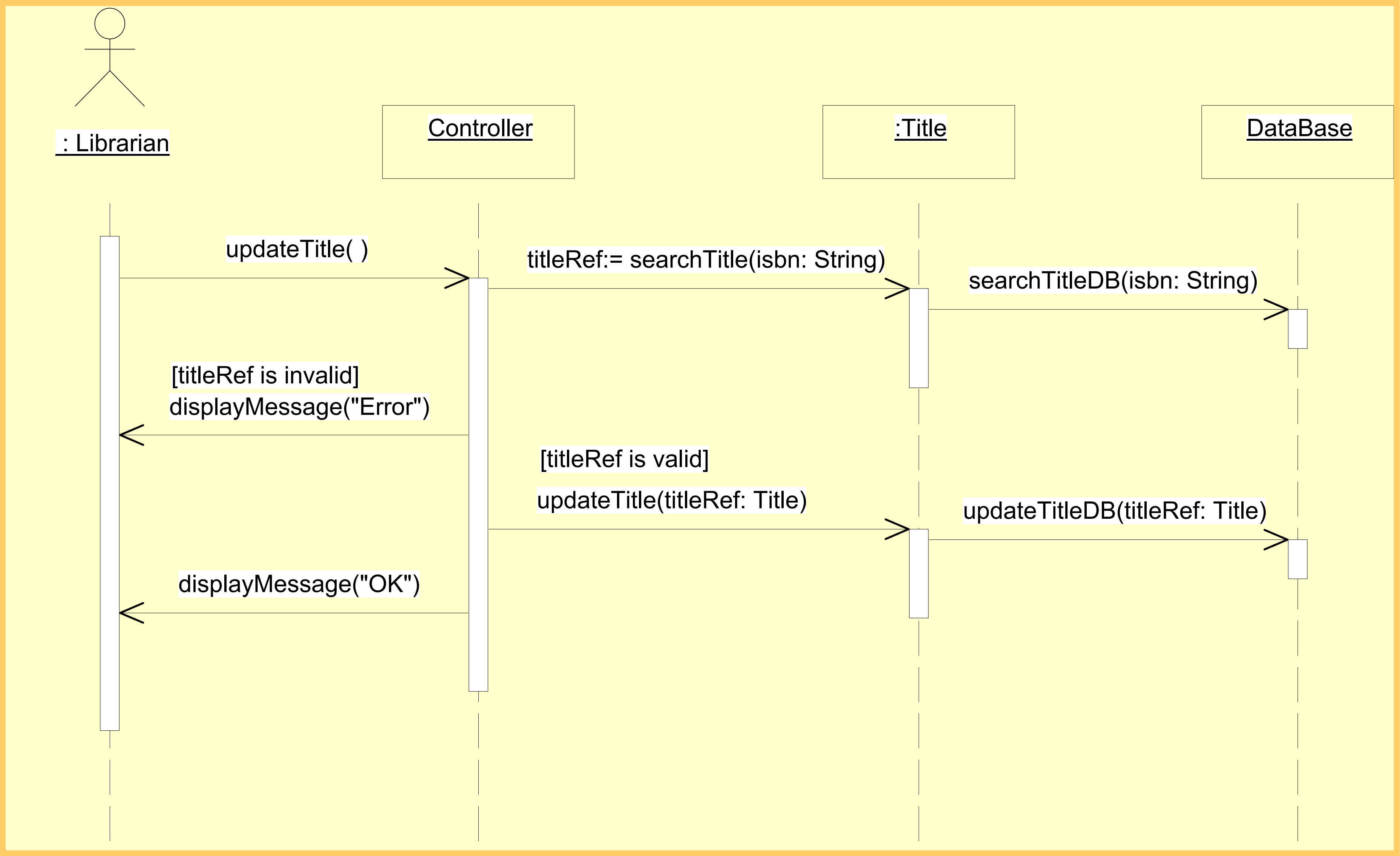
addItem(titleRef:Title)

displayMessage("OK")

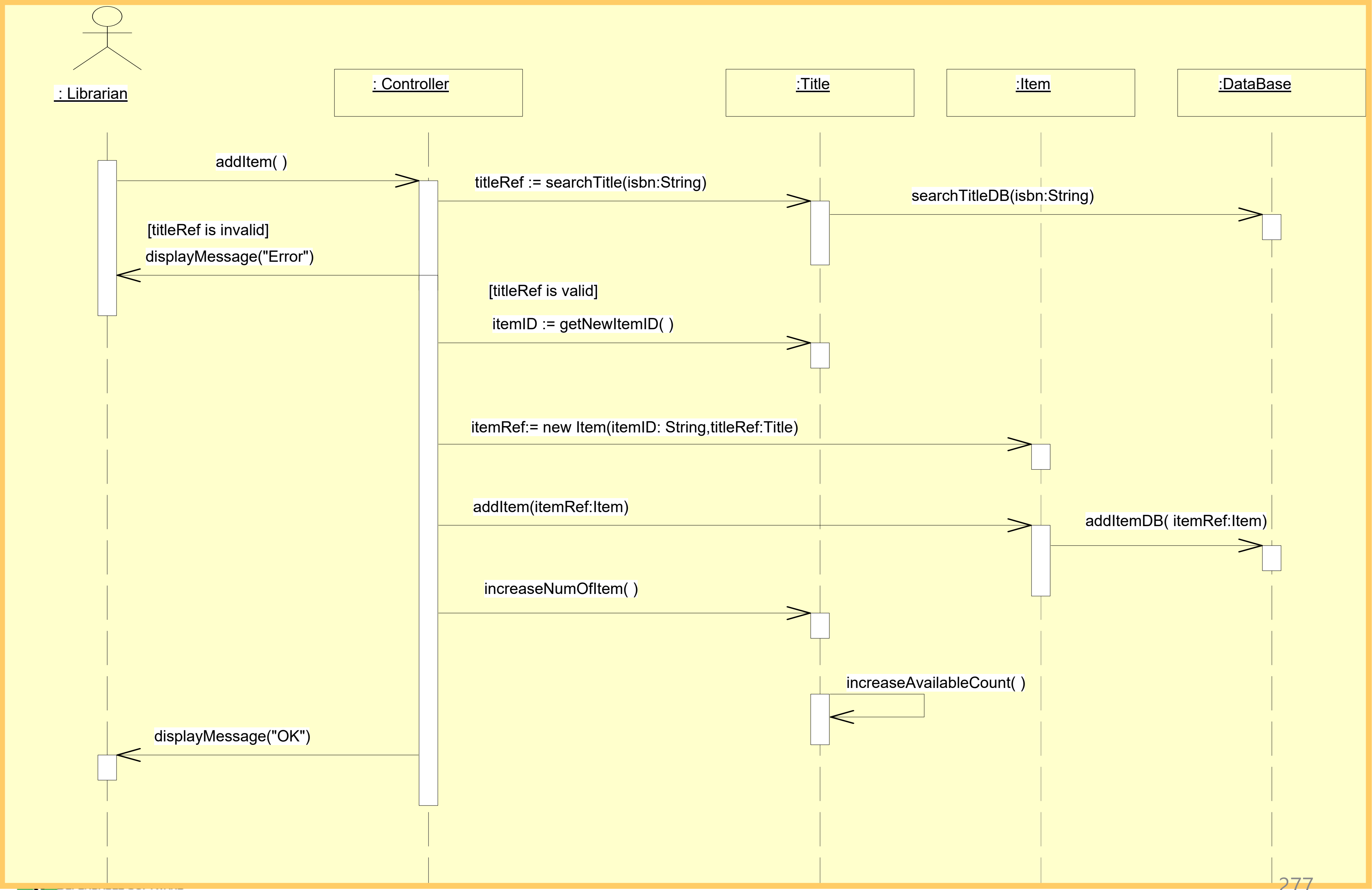
9. Remove Title


: Librarian
Controller
:Title
:Reservation
:Item
:Borrower
DataBase
removeTitle()
titleRef:=searchTitle(isbn: String)
searchTitleDB(isbn: String)
[titleRef is invalid]
displayMessage("Error")
[titleRef is valid]
reservationRef:=searchReservation(titleRef:Title)
searchReservationDB(titleRef:Title)
[reservationRef is valid]
decreaseReservationCount()
removeTitle(titleRef:Title)
removeTitleDB(titleRef:Title)
removeItem(isbn: ISBNType)
displayMessage("OK")

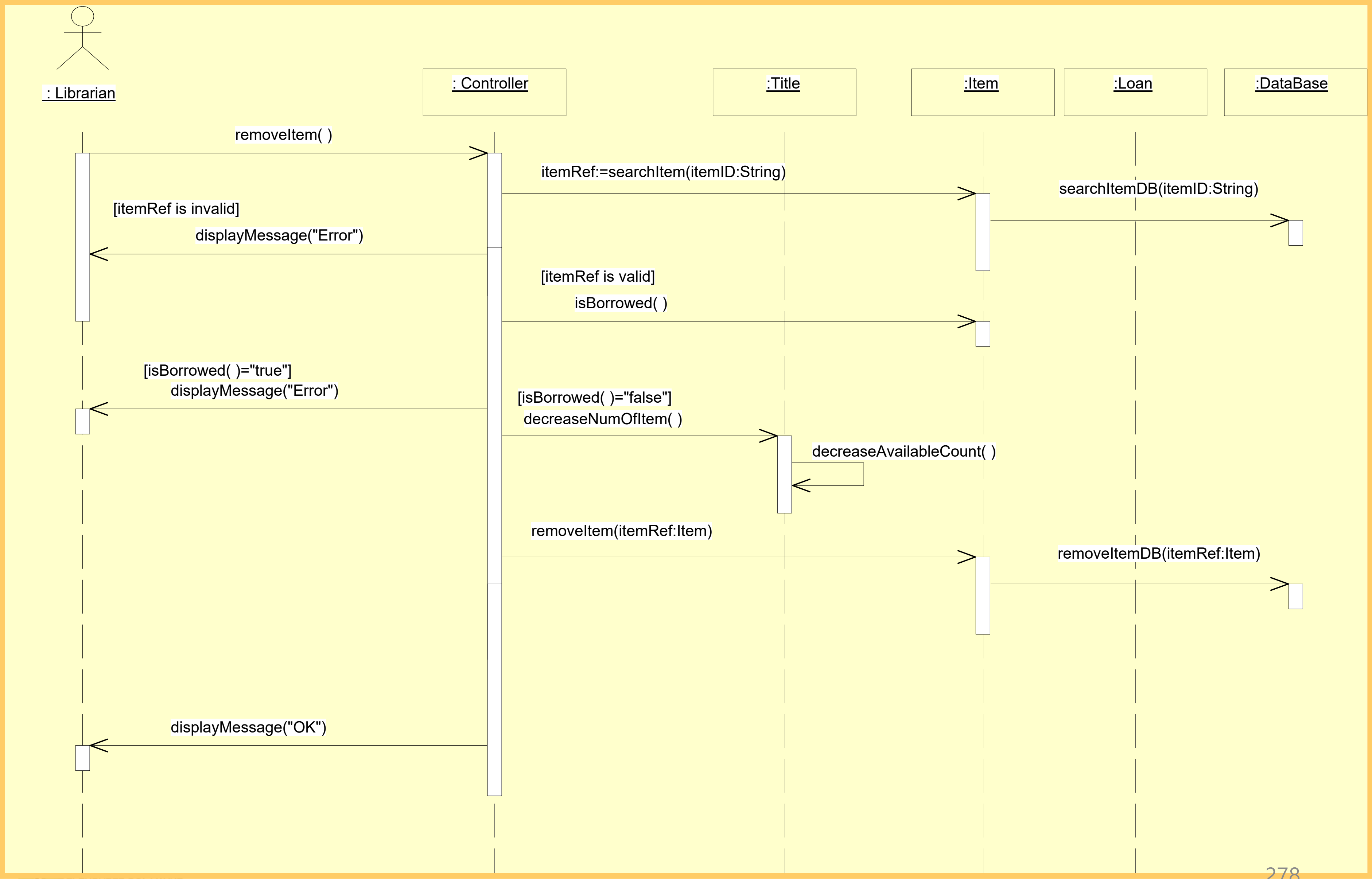
10. Update Title



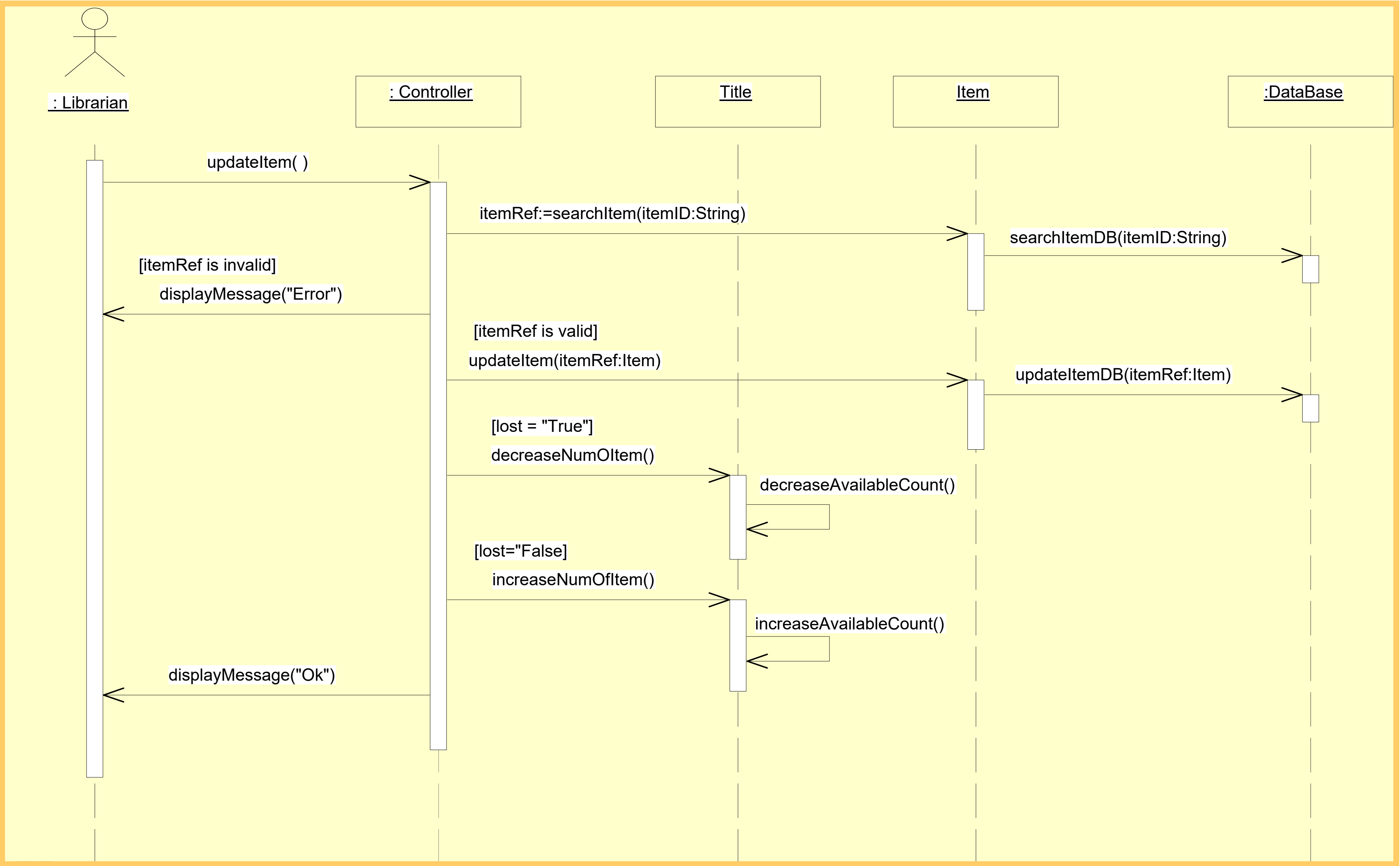
11. Add Item


: Librarian
: Controller
:Title
:Item
:DataBase
addItem( )
titleRef := searchTitle(isbn:String)
searchTitleDB(isbn:String)
[titleRef is invalid]
displayMessage("Error")
[titleRef is valid]
itemID := getNewItemID( )
itemRef:= new Item(itemID: String,titleRef:Title)
addItem(itemRef:Item)
addItemDB(itemRef:Item)
increaseNumOfItem( )
increaseAvailableCount( )
displayMessage("OK")

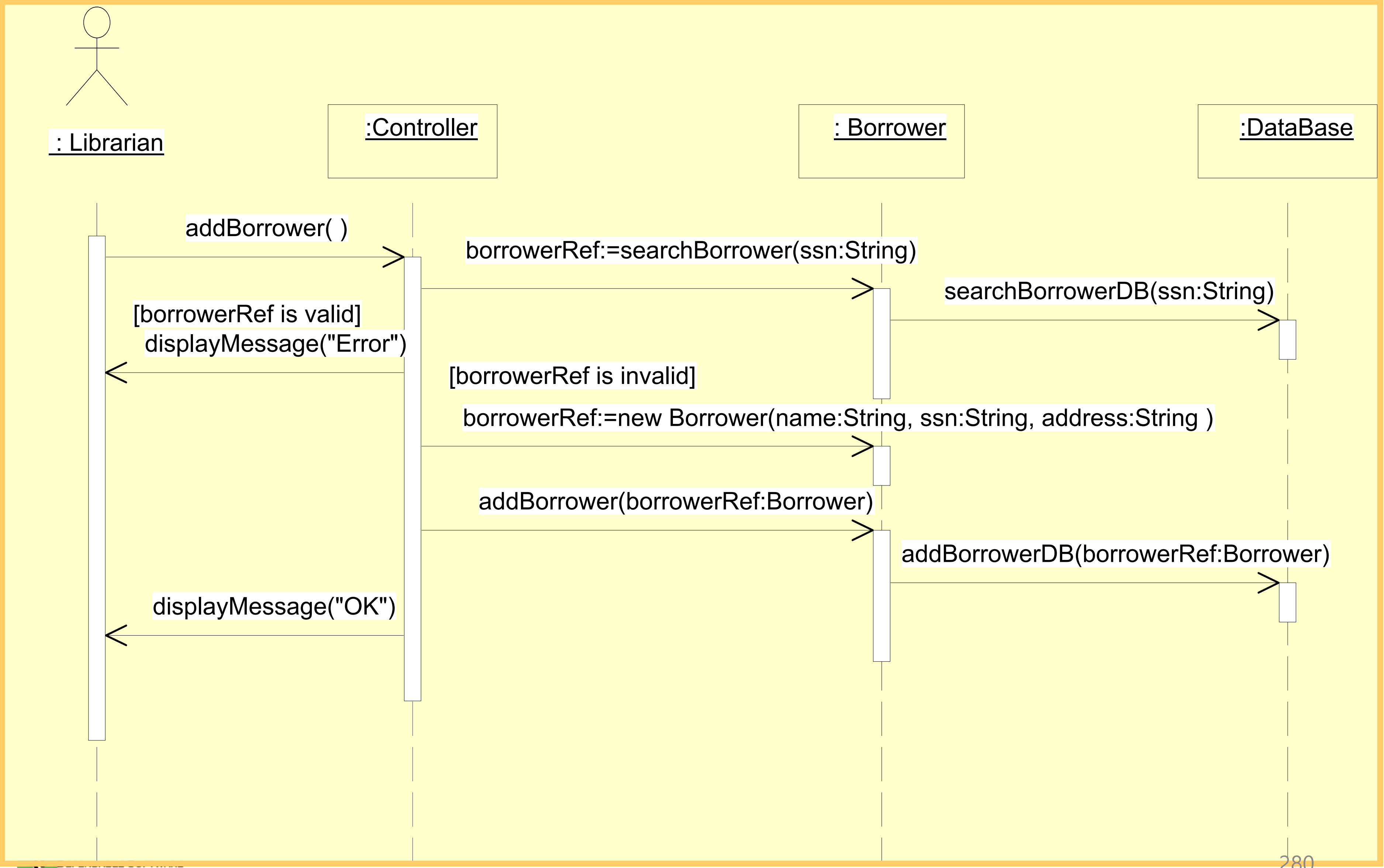
12. Remove Item



12. Update Item

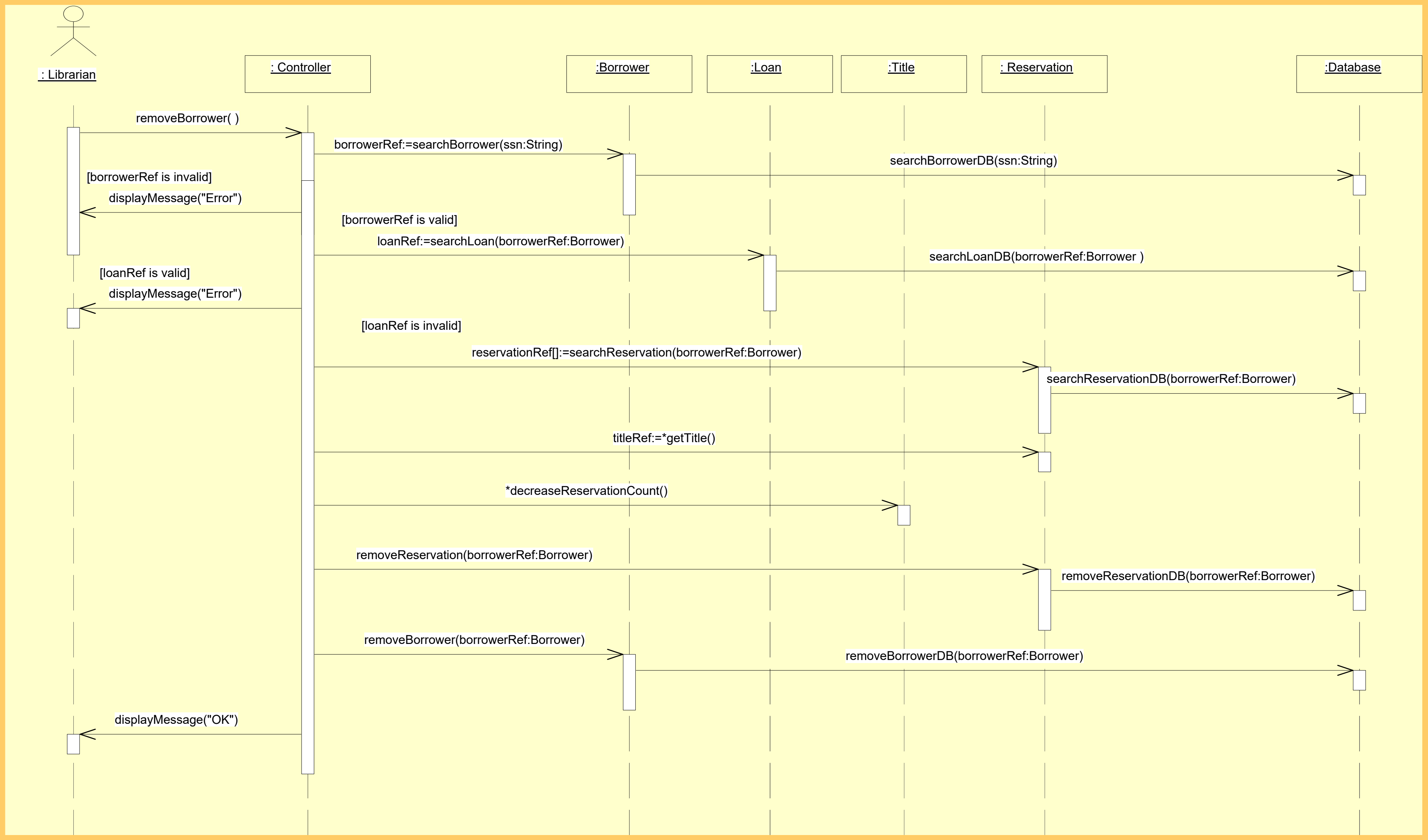


14. Add Borrower

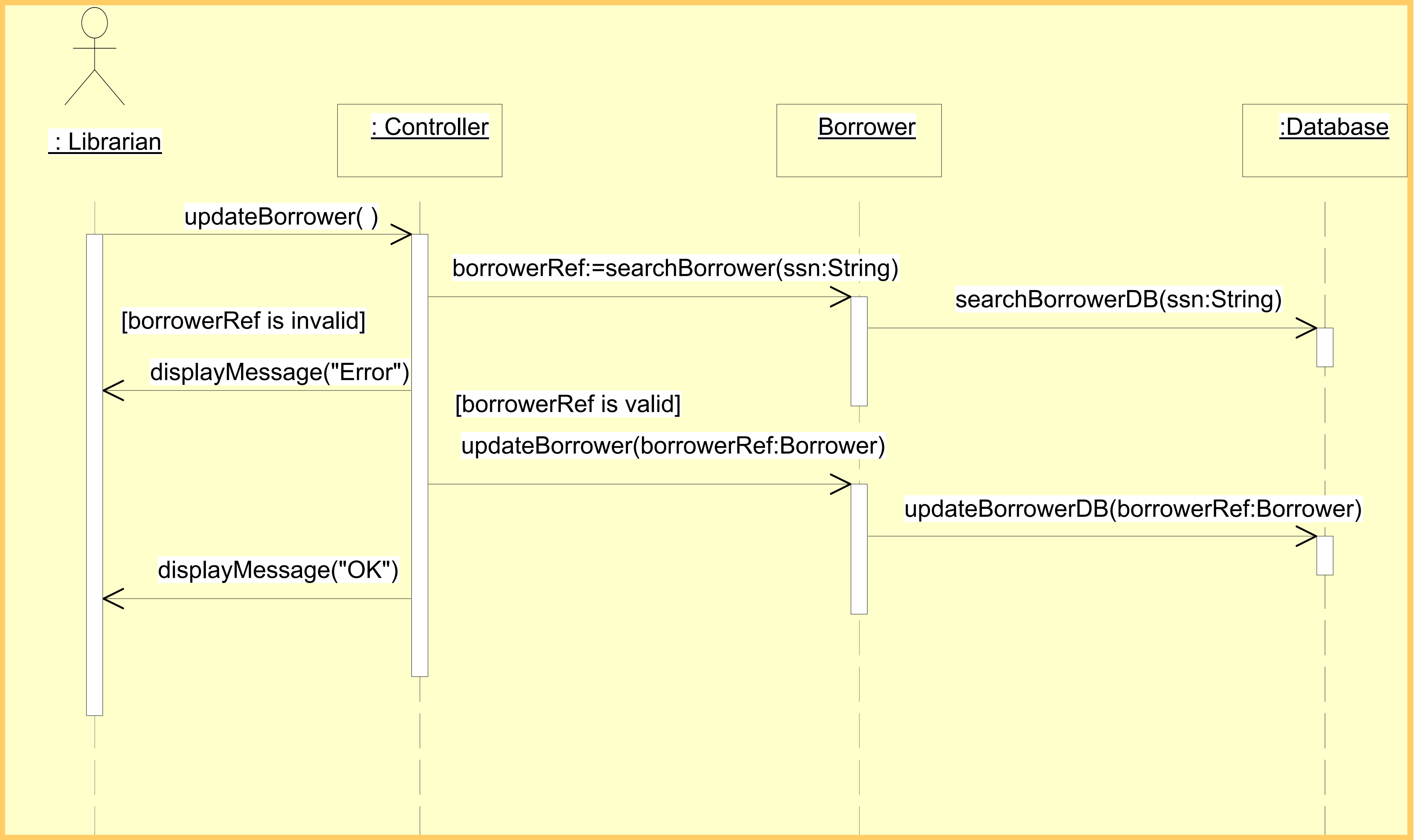


: Librarian
:Controller
: Borrower
:DataBase
addBorrower( )
borrowerRef:=searchBorrower(ssn:String)
searchBorrowerDB(ssn:String)
[borrowerRef is valid]
displayMessage("Error")
[borrowerRef is invalid]
borrowerRef:=new Borrower(name:String, ssn:String, address:String)
addBorrower(borrowerRef:Borrower)
addBorrowerDB(borrowerRef:Borrower)
displayMessage("OK")

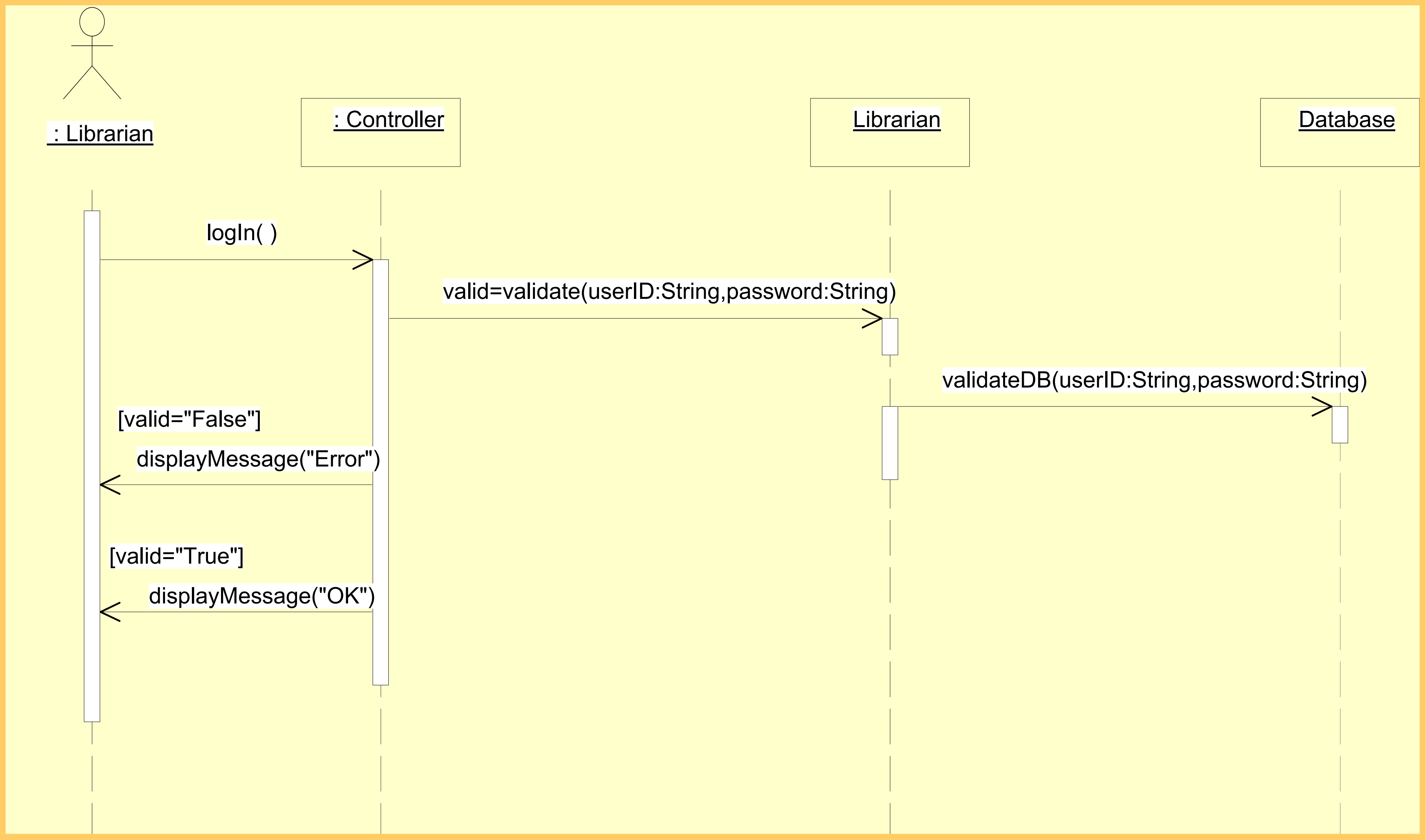
15. Remove Borrower


: Librarian
: Controller
:Borrower
:Loan
:Title
: Reservation
:Database
removeBorrower( )
borrowerRef:=searchBorrower(ssn:String)
searchBorrowerDB(ssn:String)
[borrowerRef is invalid]
displayMessage("Error")
[borrowerRef is valid]
loanRef:=searchLoan(borrowerRef:Borrower)
searchLoanDB(borrowerRef:Borrower)
[loanRef is valid]
displayMessage("Error")
[loanRef is invalid]
reservationRef[]:=searchReservation(borrowerRef:Borrower)
searchReservationDB(borrowerRef:Borrower)
titleRef:=*getTitle()
*decreaseReservationCount()
removeReservation(borrowerRef:Borrower)
removeReservationDB(borrowerRef:Borrower)
removeBorrower(borrowerRef:Borrower)
removeBorrowerDB(borrowerRef:Borrower)
displayMessage("OK")

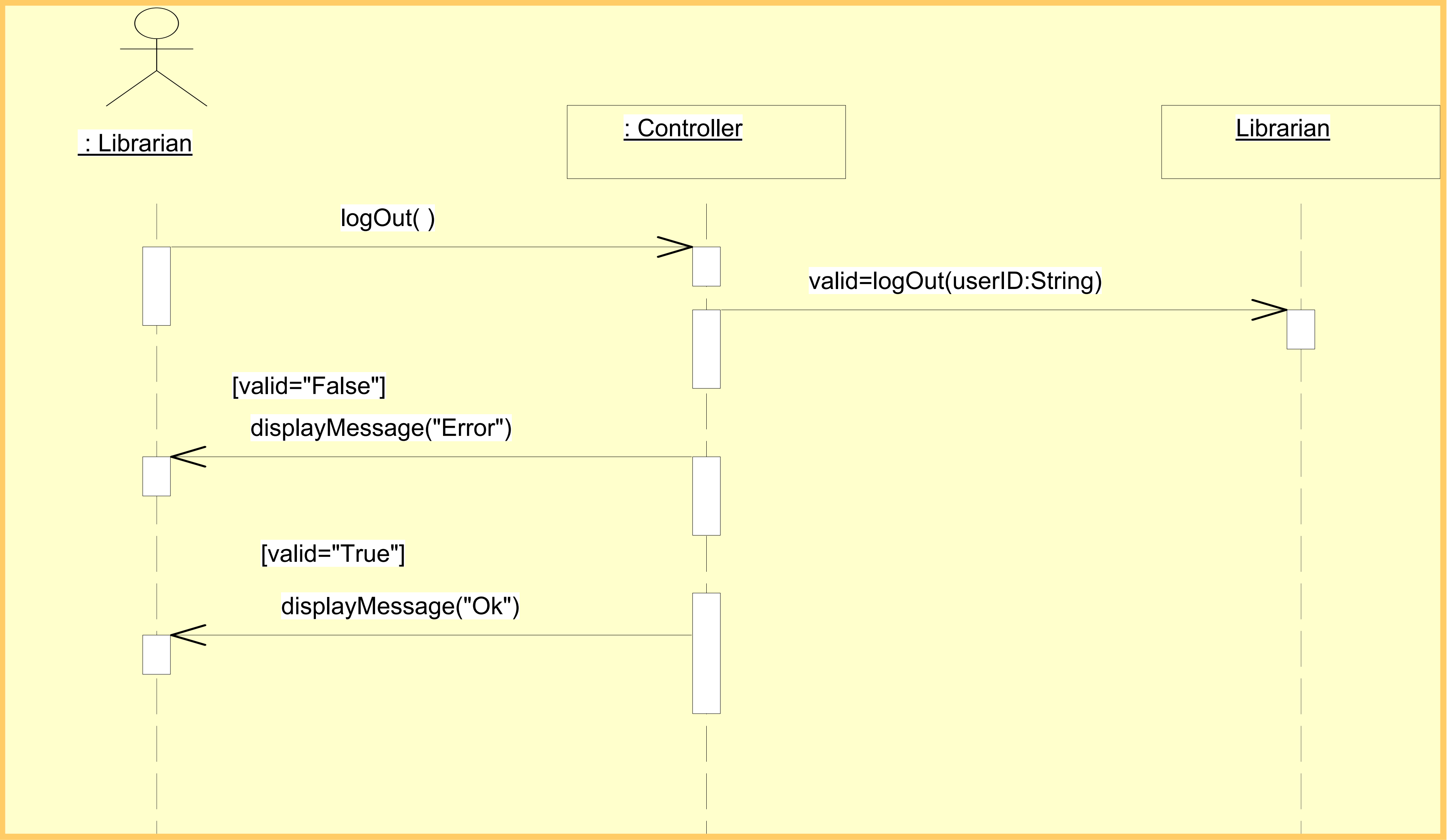
16. Update Borrower



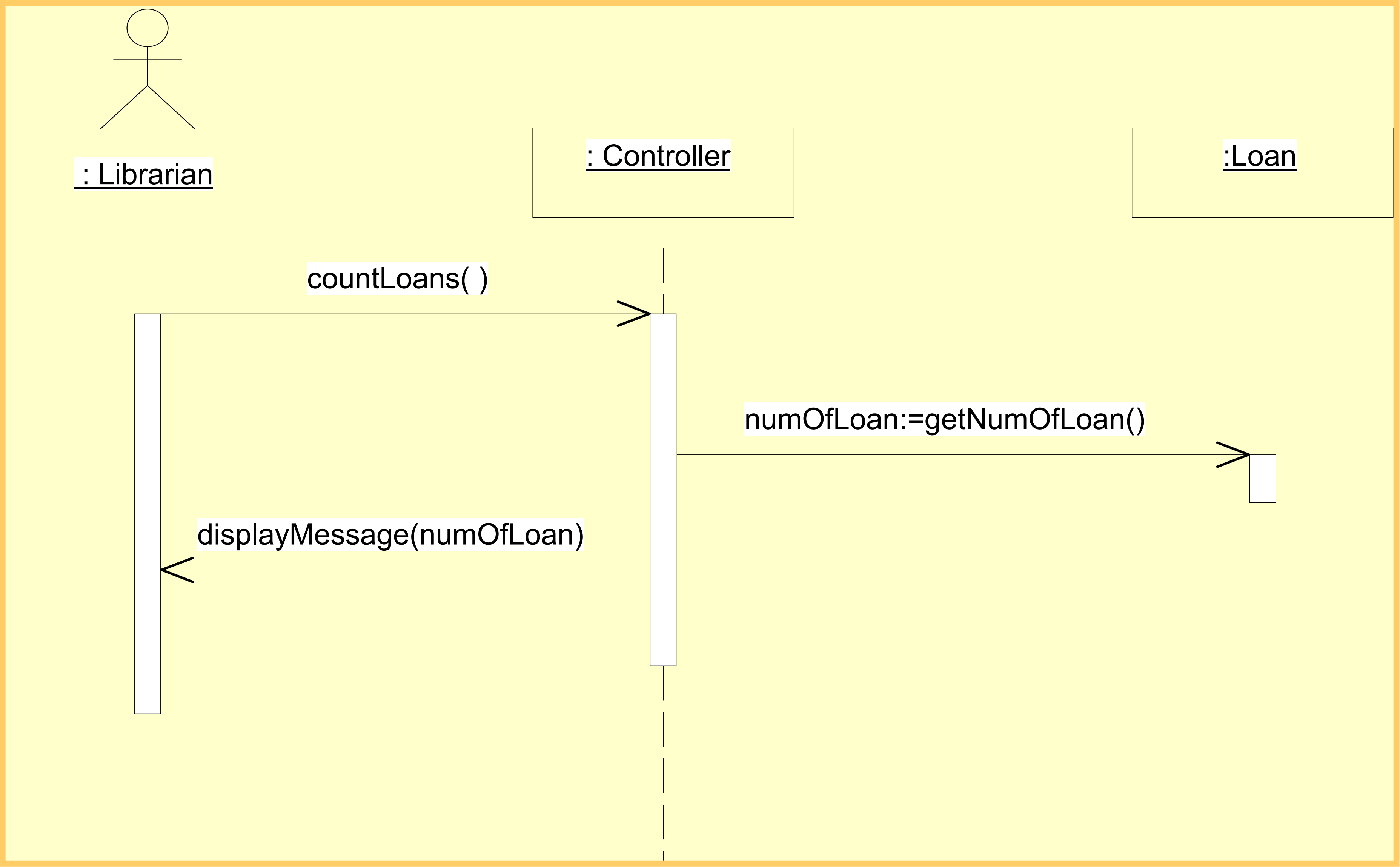
17. Log-In


: Librarian
: Controller
Librarian
Database
logIn( )
valid=validate(userID:String,password:String)
validateDB(userID:String,password:String)
[valid="False"]
displayMessage("Error")
[valid="True"]
displayMessage("OK")

18. Log-Out


: Librarian
: Controller
Librarian
logOut( )
valid=logOut(userID:String)
[valid="False"]
displayMessage("Error")
[valid="True"]
displayMessage("Ok")

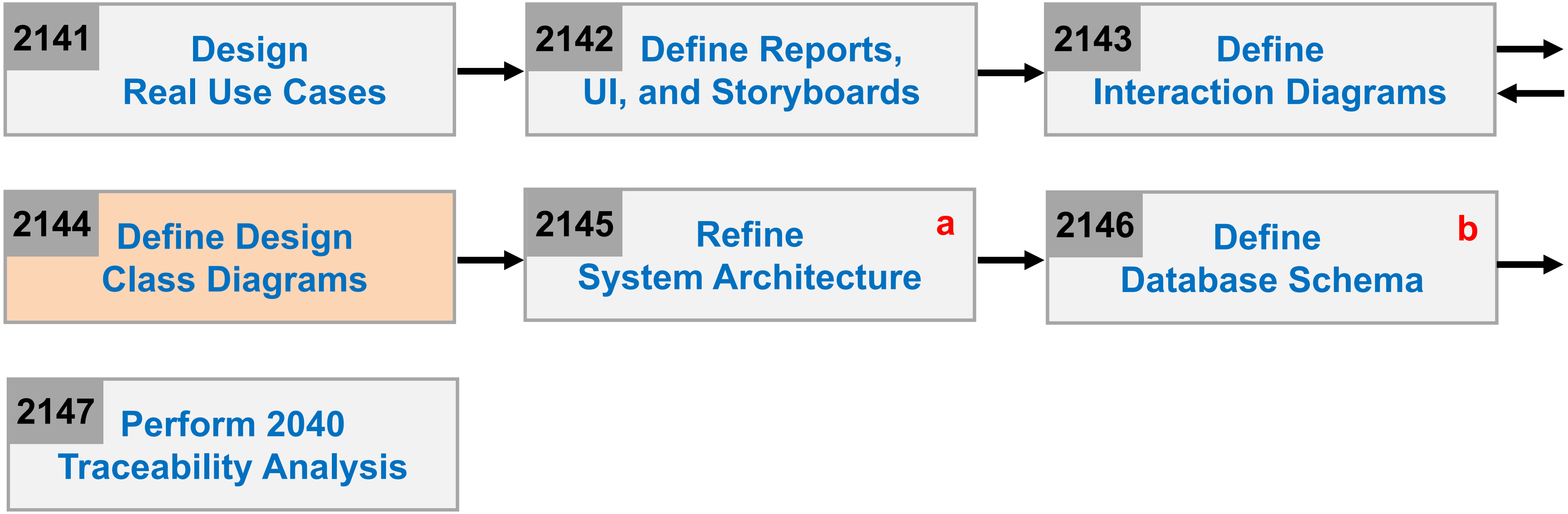
19. Count Loans



Phase 2044. Define Design Class Diagram

- 7 Activities

a. Varied order
b. optional



2141 Design Real Use Cases

2142 Define Reports, UI, and Storyboards

2143 Define Interaction Diagrams

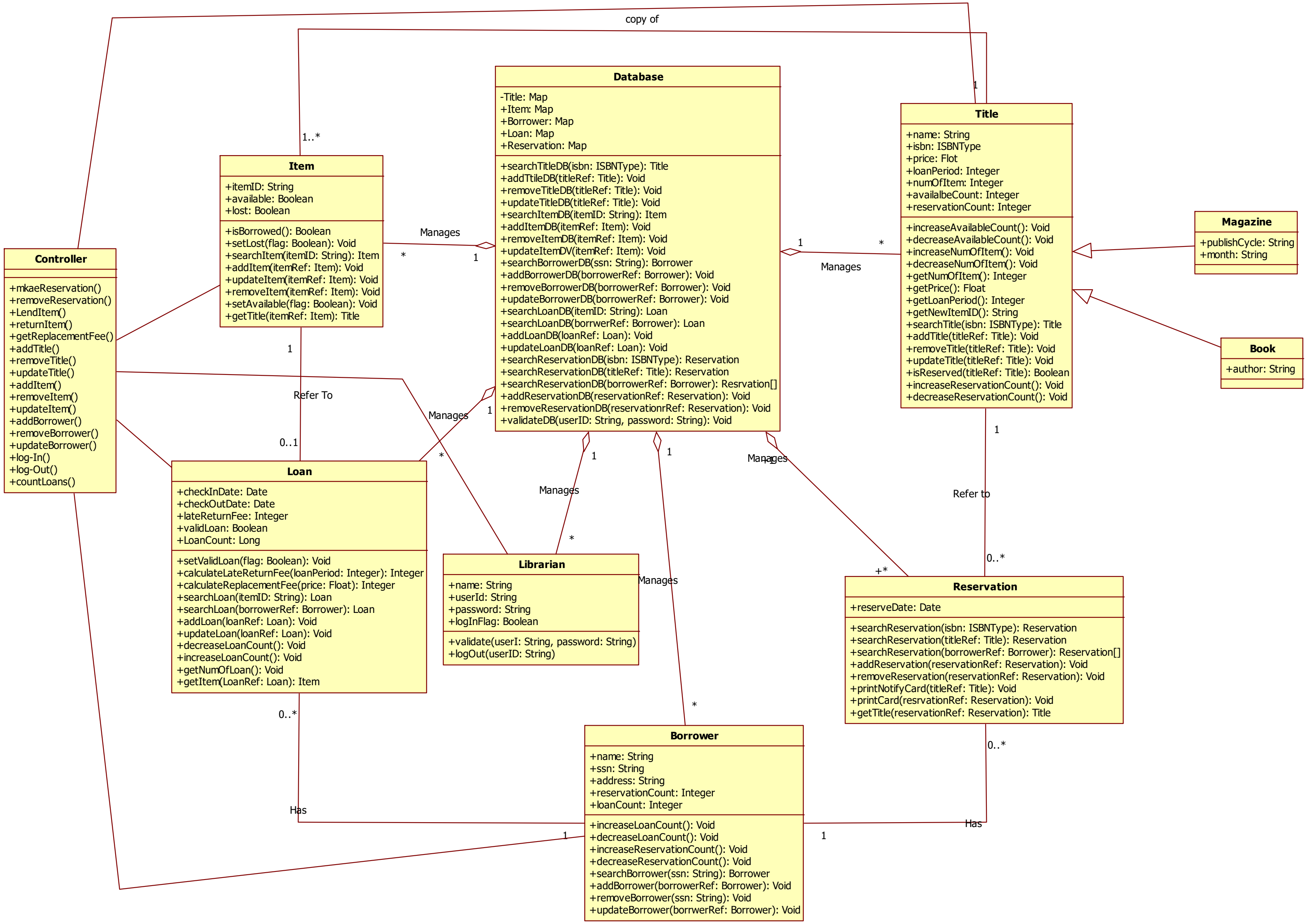
2144 Define Design Class Diagrams

2145 Refine System Architecture a

2146 Define Database Schema b

2147 Perform 2040 Traceability Analysis

Phase 2044. Define Design Class Diagram



Database

-Title: Map
+Item: Map
+Borrower: Map
+Loan: Map
+Reservation: Map

+searchTitleDB(isbn: ISBNType): Title
+addTtileDB(titleRef: Title): Void
+removeTitleDB(titleRef: Title): Void
+updateTitleDB(titleRef: Title): Void
+searchItemDB(itemID: String): Item
+addItemDB(itemRef: Item): Void
+removeItemDB(itemRef: Item): Void
+updateItemDV(itemRef: Item): Void
+searchBorrowerDB(ssn: String): Borrower
+addBorrowerDB(borrowerRef: Borrower): Void
+removeBorrowerDB(borrowerRef: Borrower): Void
+updateBorrowerDB(borrowerRef: Borrower): Void
+searchLoanDB(itemID: String): Loan
+searchLoanDB(borrwerRef: Borrower): Loan
+addLoanDB(loanRef: Loan): Void
+updateLoanDB(loanRef: Loan): Void
+searchReservationDB(isbn: ISBNType): Reservation
+searchReservationDB(titleRef: Title): Reservation
+searchReservationDB(borrowerRef: Borrower): Resrvation[]
+addReservationDB(reservationRef: Reservation): Void
+removeReservationDB(reservationrRef: Reservation): Void
+validateDB(userID: String, password: String): Void

Item

+itemID: String
+available: Boolean
+lost: Boolean

+isBorrowed(): Boolean
+setLost(flag: Boolean): Void
+searchItem(itemID: String): Item
+addItem(itemRef: Item): Void
+updateItem(itemRef: Item): Void
+removeItem(itemRef: Item): Void
+setAvailable(flag: Boolean): Void
+getTitle(itemRef: Item): Title

Title

+name: String
+isbn: ISBNType
+price: Flot
+loanPeriod: Integer
+numOfItem: Integer
+availalbeCount: Integer
+reservationCount: Integer

+increaseAvailableCount(): Void
+decreaseAvailableCount(): Void
+increaseNumOfItem(): Void
+decreaseNumOfItem(): Void
+getNumOfItem(): Integer
+getPrice(): Float
+getLoanPeriod(): Integer
+getNewItemID(): String
+searchTitle(isbn: ISBNType): Title
+addTitle(titleRef: Title): Void
+removeTitle(titleRef: Title): Void
+updateTitle(titleRef: Title): Void
+isReserved(titleRef: Title): Boolean
+increaseReservationCount(): Void
+decreaseReservationCount(): Void

Magazine

+publishCycle: String
+month: String

Book

+author: String

Controller

+mkaeReservation()
+removeReservation()
+LendItem()
+returnItem()
+getReplacementFee()
+addTitle()
+removeTitle()
+updateTitle()
+addItem()
+removeItem()
+updateItem()
+addBorrower()
+removeBorrower()
+updateBorrower()
+log-In()
+log-Out()
+countLoans()

Loan

+checkInDate: Date
+checkOutDate: Date
+lateReturnFee: Integer
+validLoan: Boolean
+LoanCount: Long

+setValidLoan(flag: Boolean): Void
+calculateLateReturnFee(loanPeriod: Integer): Integer
+calculateReplacementFee(price: Float): Integer
+searchLoan(itemID: String): Loan
+searchLoan(borrowerRef: Borrower): Loan
+addLoan(loanRef: Loan): Void
+updateLoan(loanRef: Loan): Void
+decreaseLoanCount(): Void
+increaseLoanCount(): Void
+getNumOfLoan(): Void
+getItem(LoanRef: Loan): Item

Librarian

+name: String
+userId: String
+password: String
+logInFlag: Boolean

+validate(userI: String, password: String)
+logOut(userID: String)

Reservation

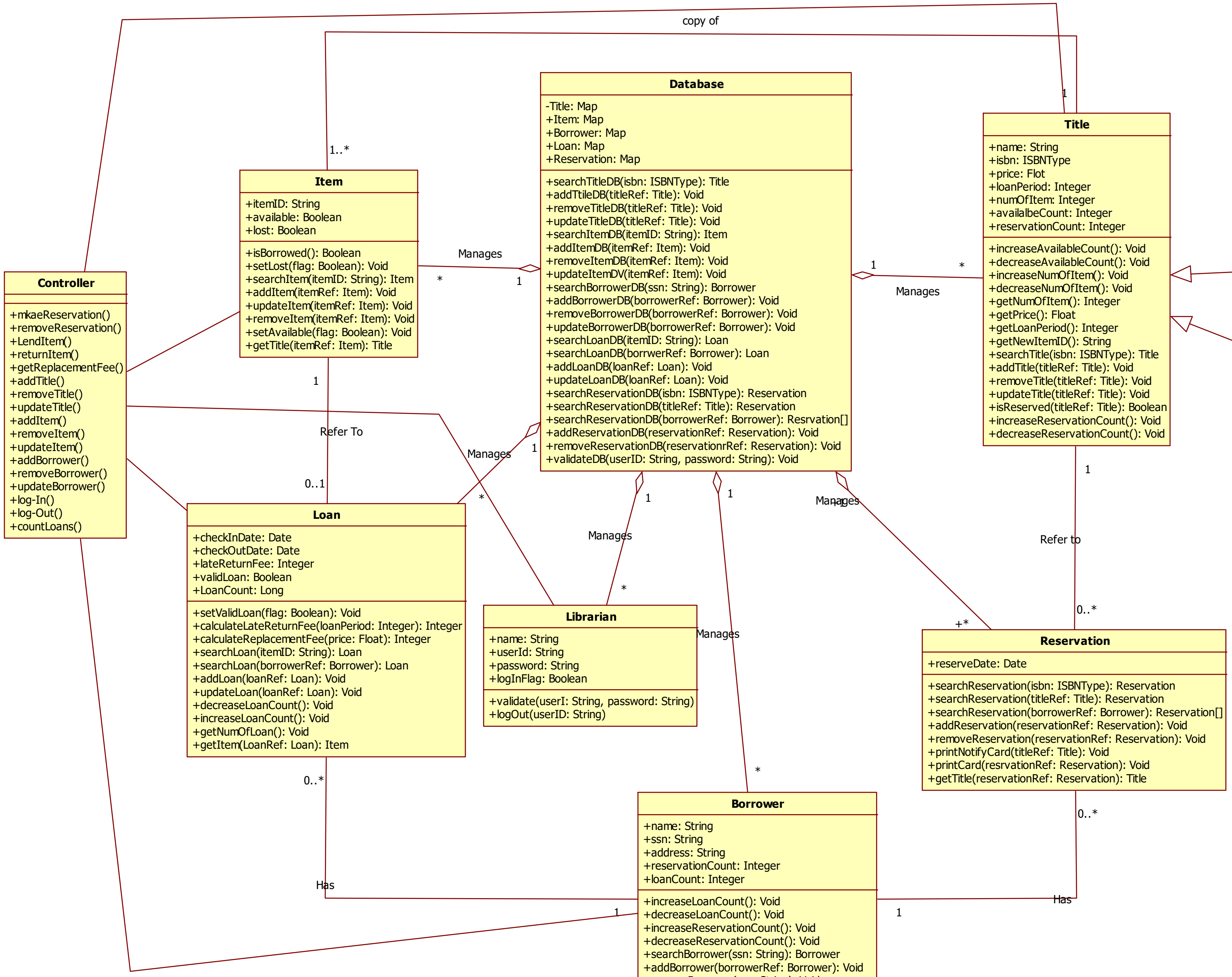
+reserveDate: Date

+searchReservation(isbn: ISBNType): Reservation
+searchReservation(titleRef: Title): Reservation
+searchReservation(borrowerRef: Borrower): Reservation[]
+addReservation(reservationRef: Reservation): Void
+removeReservation(reservationRef: Reservation): Void
+printNotifyCard(titleRef: Title): Void
+printCard(resrvationRef: Reservation): Void
+getTitle(reservationRef: Reservation): Title

Borrower

+name: String
+ssn: String
+address: String
+reservationCount: Integer
+loanCount: Integer

+increaseLoanCount(): Void
+decreaseLoanCount(): Void
+increaseReservationCount(): Void
+decreaseReservationCount(): Void
+searchBorrower(ssn: String): Borrower
+addBorrower(borrowerRef: Borrower): Void
+removeBorrower(ssn: String): Void
+updateBorrower(borrwerRef: Borrower): Void



Controller

+mkaeReservation()
+removeReservation()
+LendItem()
+returnItem()
+getReplacementFee()
+addTitle()
+removeTitle()
+updateTitle()
+addItem()
+removeItem()
+updateItem()
+addBorrower()
+removeBorrower()
+updateBorrower()
+log-In()
+log-Out()
+countLoans()

Item

+itemID: String
+available: Boolean
+lost: Boolean

+isBorrowed(): Boolean
+setLost(flag: Boolean): Void
+searchItem(itemID: String): Item
+addItem(itemRef: Item): Void
+updateItem(itemRef: Item): Void
+removeItem(itemRef: Item): Void
+setAvailable(flag: Boolean): Void
+getTitle(itemRef: Item): Title

Database

-Title: Map
+Item: Map
+Borrower: Map
+Loan: Map
+Reservation: Map

+searchTitleDB(isbn: ISBNType): Title
+addTtileDB(titleRef: Title): Void
+removeTitleDB(titleRef: Title): Void
+updateTitleDB(titleRef: Title): Void
+searchItemDB(itemID: String): Item
+addItemDB(itemRef: Item): Void
+removeItemDB(itemRef: Item): Void
+updateItemDV(itemRef: Item): Void
+searchBorrowerDB(ssn: String): Borrower
+addBorrowerDB(borrowerRef: Borrower): Void
+removeBorrowerDB(borrowerRef: Borrower): Void
+updateBorrowerDB(borrowerRef: Borrower): Void
+searchLoanDB(itemID: String): Loan
+searchLoanDB(borrwerRef: Borrower): Loan
+addLoanDB(loanRef: Loan): Void
+updateLoanDB(loanRef: Loan): Void
+searchReservationDB(isbn: ISBNType): Reservation
+searchReservationDB(titleRef: Title): Reservation
+searchReservationDB(borrowerRef: Borrower): Resrvation[]
+addReservationDB(reservationRef: Reservation): Void
+removeReservationDB(reservationrRef: Reservation): Void
+validateDB(userID: String, password: String): Void

Title

+name: String
+isbn: ISBNType
+price: Flot
+loanPeriod: Integer
+numOfItem: Integer
+availalbeCount: Integer
+reservationCount: Integer

+increaseAvailableCount(): Void
+decreaseAvailableCount(): Void
+increaseNumOfItem(): Void
+decreaseNumOfItem(): Void
+getNumOfItem(): Integer
+getPrice(): Float
+getLoanPeriod(): Integer
+getNewItemID(): String
+searchTitle(isbn: ISBNType): Title
+addTitle(titleRef: Title): Void
+removeTitle(titleRef: Title): Void
+updateTitle(titleRef: Title): Void
+isReserved(titleRef: Title): Boolean
+increaseReservationCount(): Void
+decreaseReservationCount(): Void

Magazine

+publishCycle: String
+month: String

Book

+author: String

Loan

+checkInDate: Date
+checkOutDate: Date
+lateReturnFee: Integer
+validLoan: Boolean
+LoanCount: Long

+setValidLoan(flag: Boolean): Void
+calculateLateReturnFee(loanPeriod: Integer): Integer
+calculateReplacementFee(price: Float): Integer
+searchLoan(itemID: String): Loan
+searchLoan(borrowerRef: Borrower): Loan
+addLoan(loanRef: Loan): Void
+updateLoan(loanRef: Loan): Void
+decreaseLoanCount(): Void
+increaseLoanCount(): Void
+getNumOfLoan(): Void
+getItem(LoanRef: Loan): Item

Librarian

+name: String
+userId: String
+password: String
+logInFlag: Boolean

+validate(userI: String, password: String)
+logOut(userID: String)

Reservation

+reserveDate: Date

+searchReservation(isbn: ISBNType): Reservation
+searchReservation(titleRef: Title): Reservation
+searchReservation(borrowerRef: Borrower): Reservation[]
+addReservation(reservationRef: Reservation): Void
+removeReservation(reservationRef: Reservation): Void
+printNotifyCard(titleRef: Title): Void
+printCard(resrvationRef: Reservation): Void
+getTitle(reservationRef: Reservation): Title

Borrower

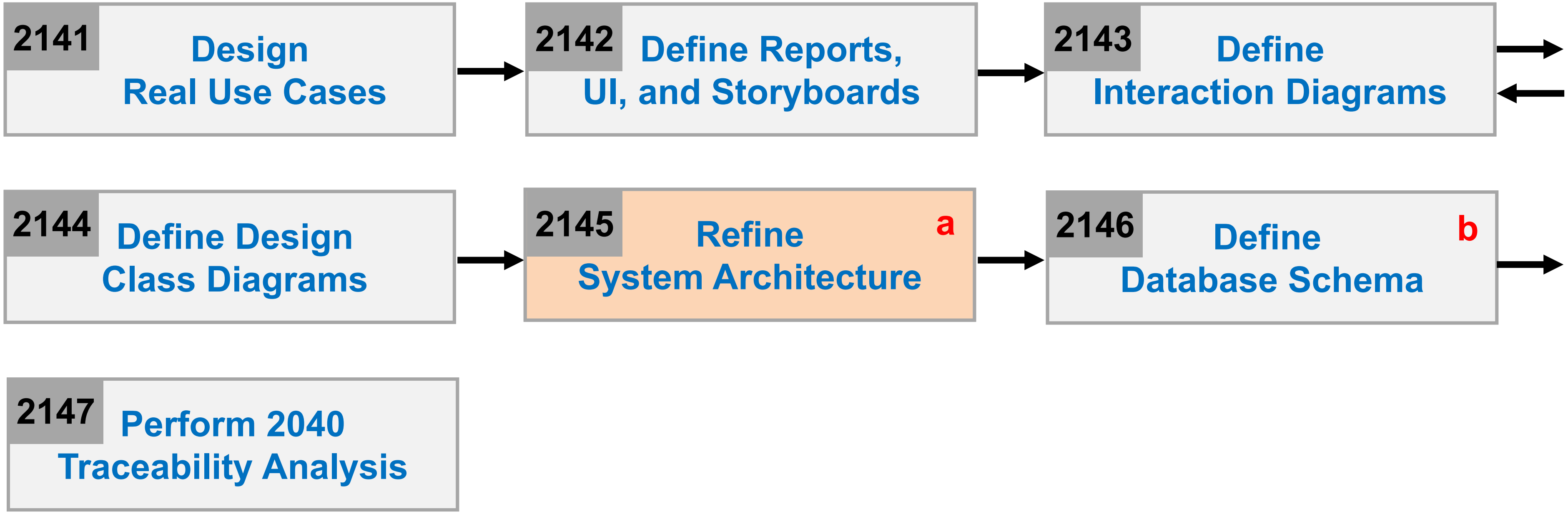
+name: String
+ssn: String
+address: String
+reservationCount: Integer
+loanCount: Integer

+increaseLoanCount(): Void
+decreaseLoanCount(): Void
+increaseReservationCount(): Void
+decreaseReservationCount(): Void
+searchBorrower(ssn: String): Borrower
+addBorrower(borrowerRef: Borrower): Void
+removeBorrower(ssn: String): Void
+updateBorrower(borrwerRef: Borrower): Void

Phase 2045. Refine System Architecture

- 7 Activities

a. Varied order
b. optional



2141 Design Real Use Cases

2142 Define Reports, UI, and Storyboards

2143 Define Interaction Diagrams

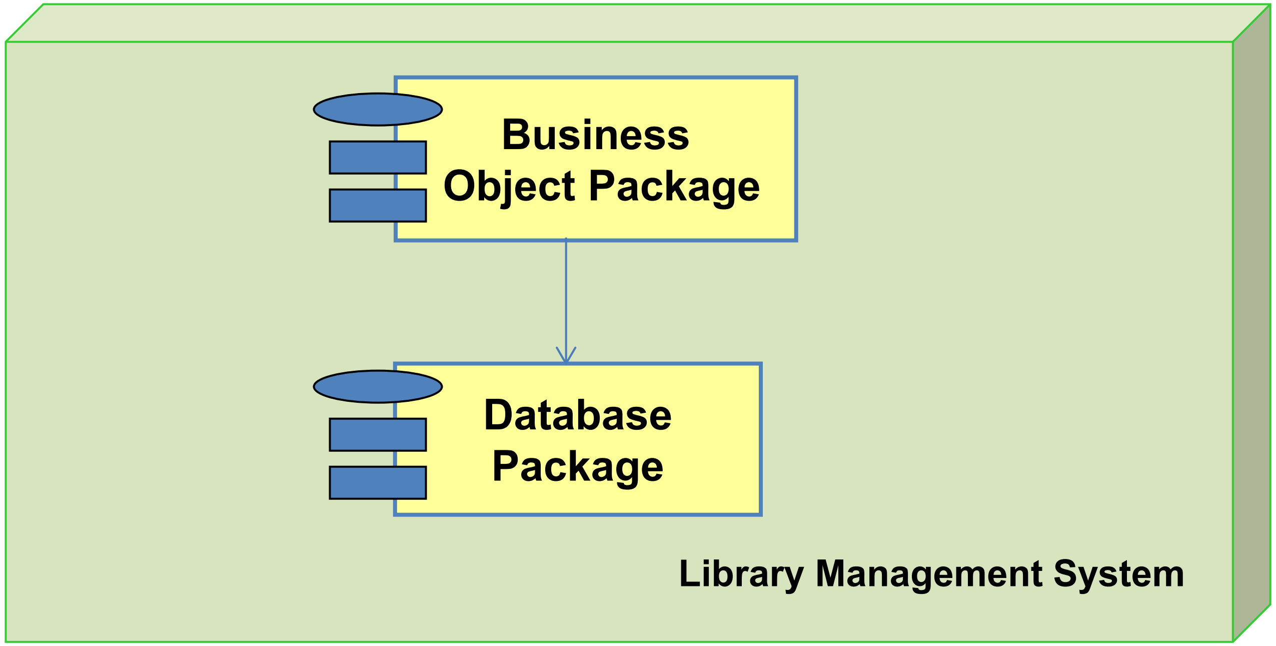
2144 Define Design Class Diagrams

2145 Refine System Architecture a

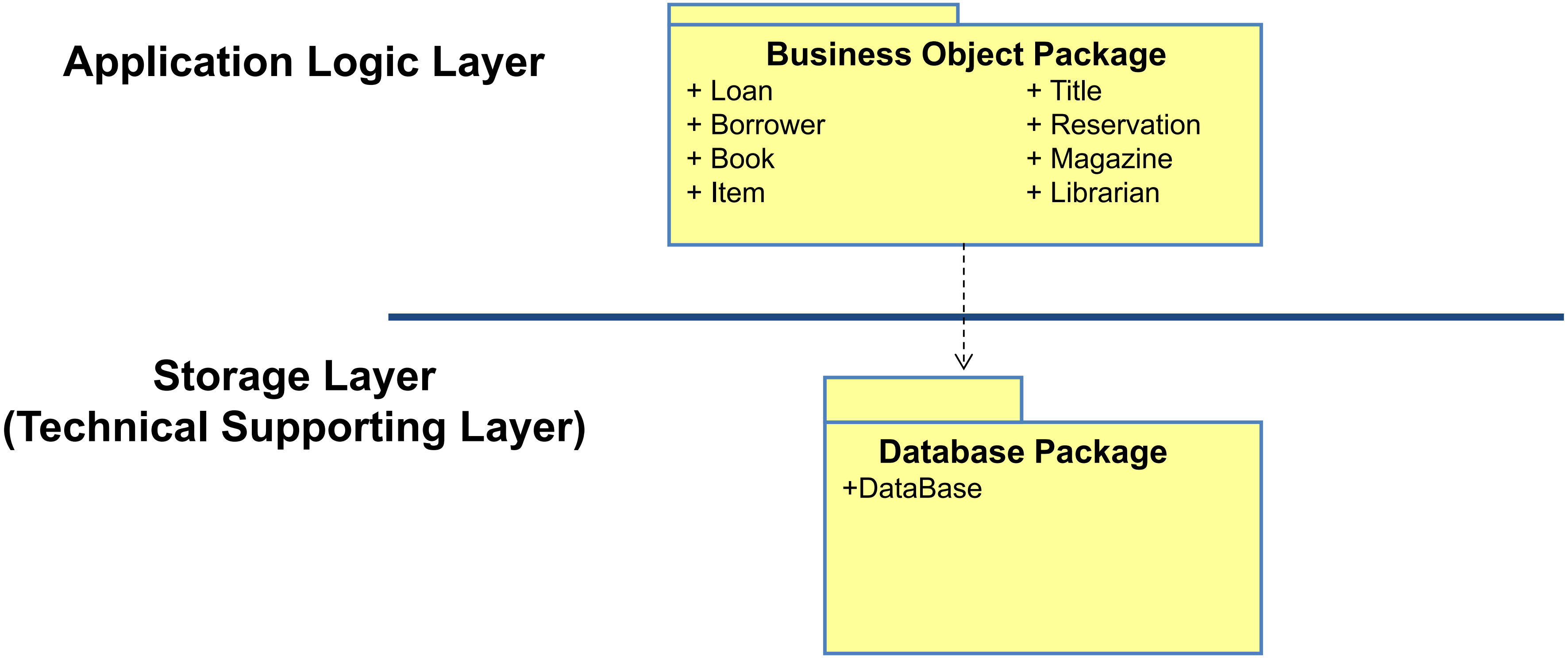
2146 Define Database Schema b

2147 Perform 2040 Traceability Analysis

Phase 2045. Refine System Architecture



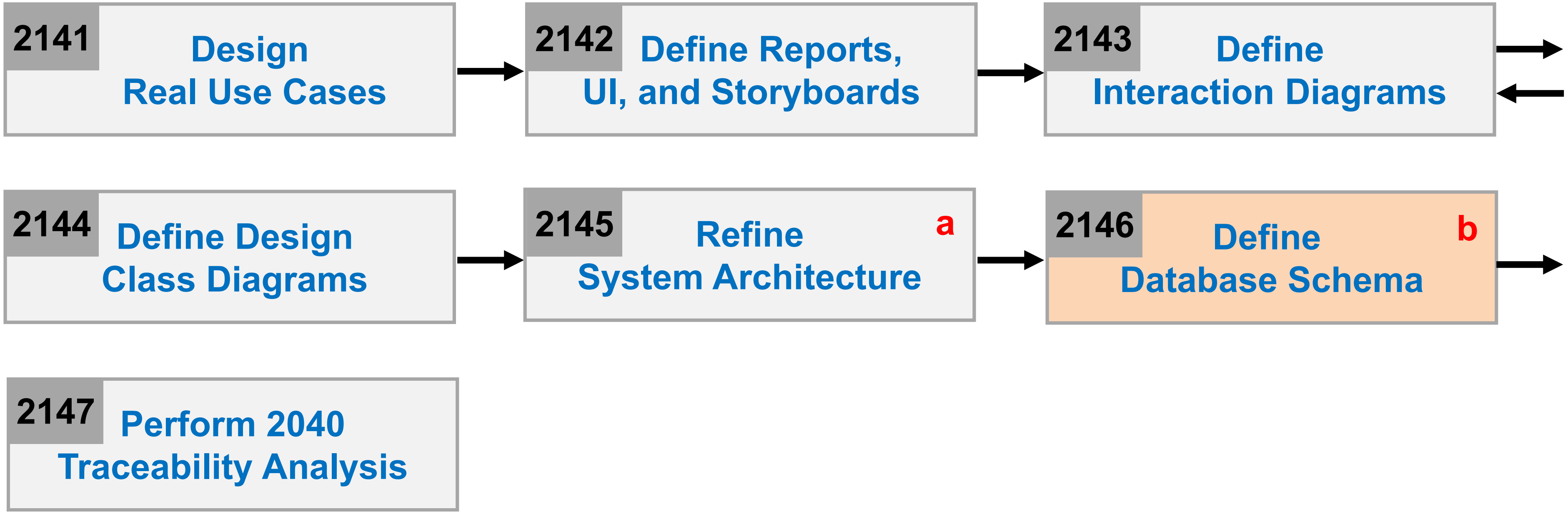
Phase 2045. Refine System Architecture



Phase 2046 Define Database Schema

- 7 Activities

a. Varied order
b. optional

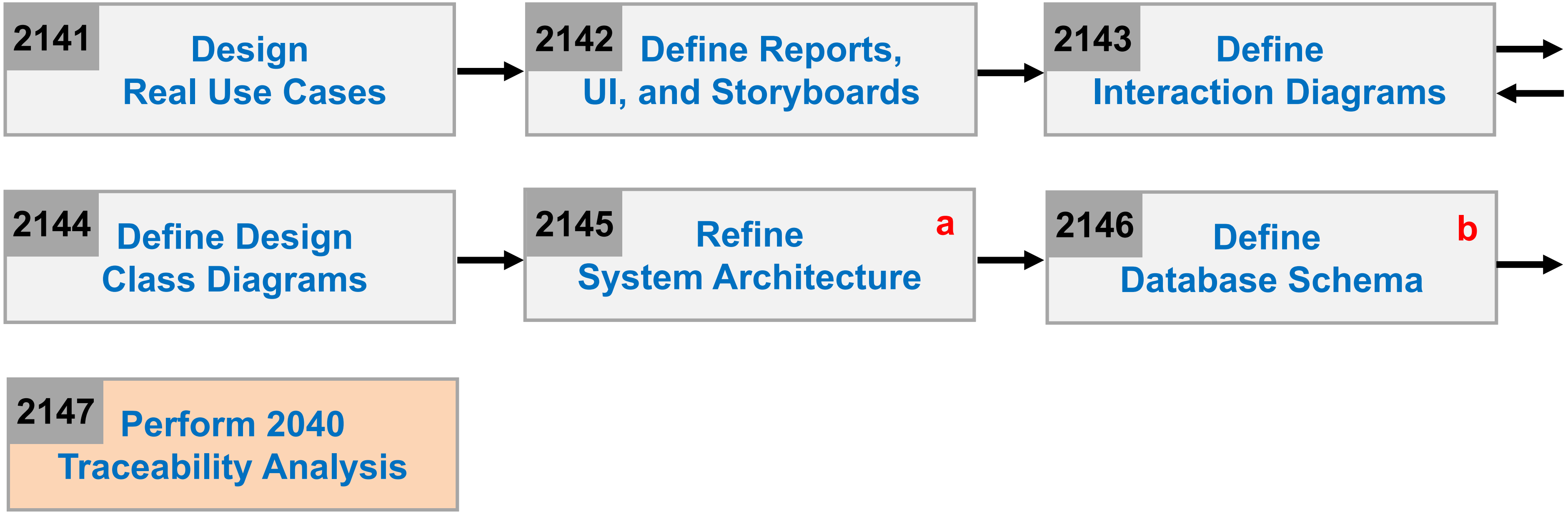


2141 Design Real Use Cases
2142 Define Reports, UI, and Storyboards
2143 Define Interaction Diagrams
2144 Define Design Class Diagrams
2145 Refine System Architecture
a
2146 Define Database Schema
b
2147 Perform 2040 Traceability Analysis

Phase 2047. Perform 2040 Traceability Analysis

- 7 Activities

a. Varied order
b. optional



2141 Design Real Use Cases

2142 Define Reports, UI, and Storyboards

2143 Define Interaction Diagrams

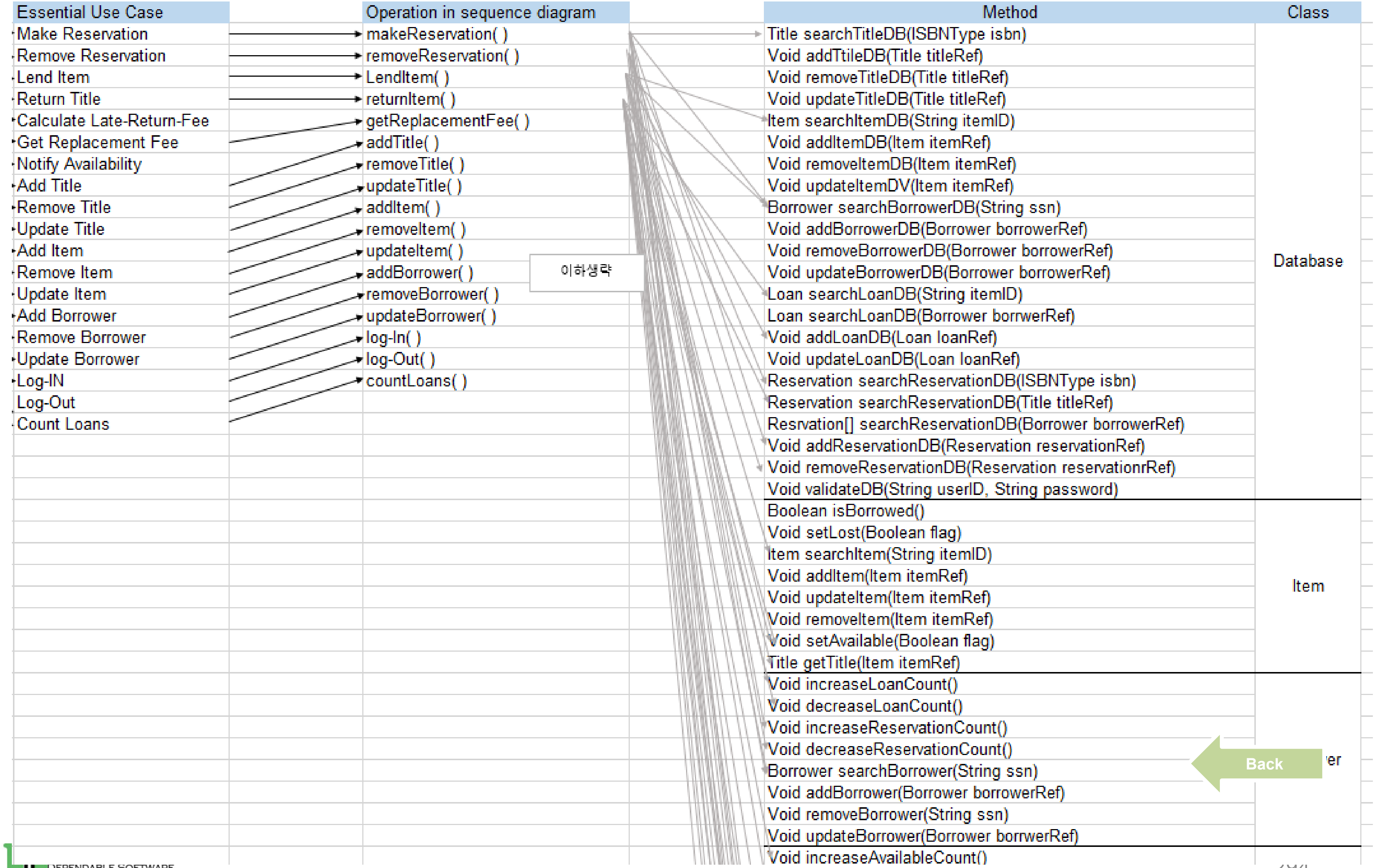
2144 Define Design Class Diagrams

2145 Refine System Architecture a

2146 Define Database Schema b

2147 Perform 2040 Traceability Analysis

Phase 2047. Perform 2040 Traceability Analysis



Essential Use Case	Operation in sequence diagram	Method	Class
Make Reservation	makeReservation( )	Title searchTitleDB(ISBNType isbn)	Database
Remove Reservation	removeReservation( )	Void addTtileDB(Title titleRef)	
Lend Item	LendItem( )	Void removeTitleDB(Title titleRef)	
Return Title	returnItem( )	Void updateTitleDB(Title titleRef)	
Calculate Late-Return-Fee	getReplacementFee( )	Item searchItemDB(String itemID)	
Get Replacement Fee	addTitle( )	Void addItemDB(Item itemRef)	
Notify Availability	removeTitle( )	Void removeItemDB(Item itemRef)	
Add Title	updateTitle( )	Void updateItemDV(Item itemRef)	
Remove Title	addItem( )	Borrower searchBorrowerDB(String ssn)	
Update Title	removeItem( )	Void addBorrowerDB(Borrower borrowerRef)	
Add Item	updateItem( )	Void removeBorrowerDB(Borrower borrowerRef)	
Remove Item	addBorrower( )	Void updateBorrowerDB(Borrower borrowerRef)	
Update Item	removeBorrower( )	Loan searchLoanDB(String itemID)	
Add Borrower	updateBorrower( )	Loan searchLoanDB(Borrower borrwerRef)	
Remove Borrower	log-In( )	Void addLoanDB(Loan loanRef)	
Update Borrower	log-Out( )	Void updateLoanDB(Loan loanRef)	
Log-IN	countLoans( )	Reservation searchReservationDB(ISBNType isbn)	
Log-Out		Reservation searchReservationDB(Title titleRef)	
Count Loans		Resrvation[] searchReservationDB(Borrower borrowerRef)	
		Void addReservationDB(Reservation reservationRef)	
		Void removeReservationDB(Reservation reservationrRef)	
		Void validateDB(String userID, String password)	
		Boolean isBorrowed()	Item
		Void setLost(Boolean flag)	
		Item searchItem(String itemID)	
		Void addItem(Item itemRef)	
		Void updateItem(Item itemRef)	
		Void removeItem(Item itemRef)	
		Void setAvailable(Boolean flag)	
		Title getTitle(Item itemRef)	
		Void increaseLoanCount()	er
		Void decreaseLoanCount()	
		Void increaseReservationCount()	
		Void decreaseReservationCount()	
		Borrower searchBorrower(String ssn)	
		Void addBorrower(Borrower borrowerRef)	
		Void removeBorrower(String ssn)	
		Void updateBorrower(Borrower borrwerRef)	
		Void increaseAvailableCount()	

이하생략

Back