

OOAD 객체지향개발방법론

A Brief Guide for Team Projects

JUNBEOM YOO
jbyoo@konkuk.ac.kr

KONKUK UNIVERSITY
<http://dslab.konkuk.ac.kr>

An Overview of Team Projects

The RVC Control SW



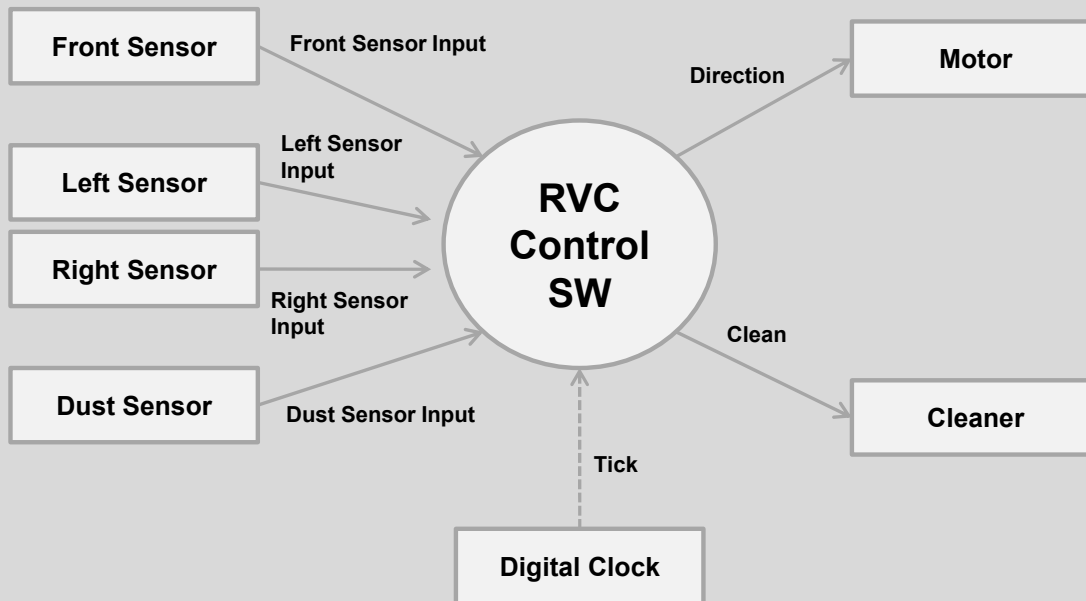
Preliminary Requirements for RVC SW Controller

- An RVC automatically cleans and mops household surface.
- It goes straight forward while cleaning.
- If its sensors found an obstacle, it stops cleaning, turns aside left or right, and goes forward with cleaning.
- If there are obstacles in both front, left and right, it move backward and turn aside left or right, and goes forward.
- If it detects dust, power up the cleaning for a while.
- We do not consider the detail design and implementation on HW controls.
- We only focus on the automatic cleaning function.

Future or Extended Requirements to Consider

- The RVC will add or change sensors.
- It will be able to circulate one spot for a while.
- It will have to communicate with a mobile app.
- It can do machine learning and inferring for more efficient cleaning.

DFD Level 0 from SASD (SE)



Input/ Output Event	Description	Format / Type
Front Sensor Input	Detects obstacles in front of the RVC	True / False , Interrupt
Left Sensor Input	Detects obstacles in the left side of the RVC periodically	True / False , Periodic
Right Sensor Input	Detects obstacles in the right side of the RVC periodically	True / False , Periodic
Dust Sensor Input	Detects dust on the floor periodically	True / False , Periodic
Direction	Direction commands to the motor (go forward / turn left with an angle / turn right with an angle)	Forward / Backward / Left / Right
Clean	Turn off / Turn on / Power-Up	On / Off / Up

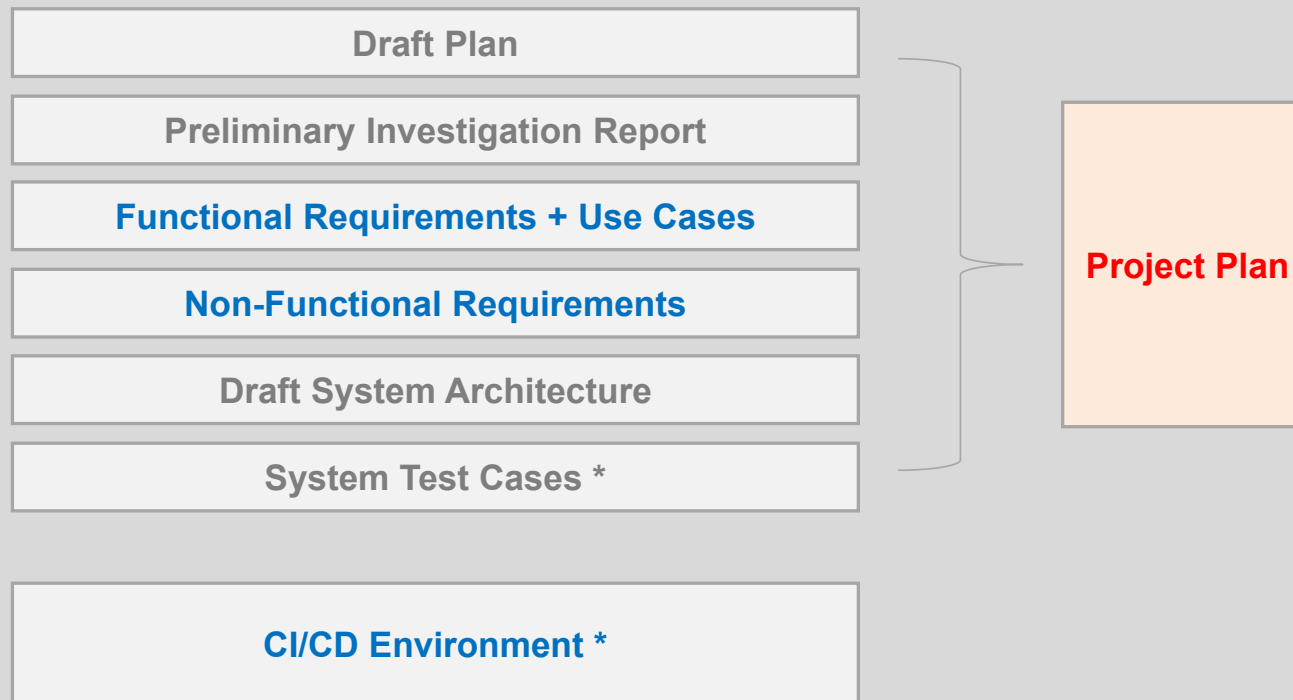
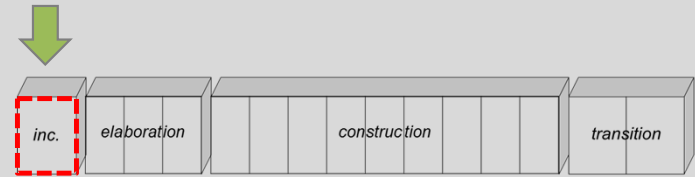
OOAD for RVC Control SW

- Develop a **control software for RVCs** as well as a simulator for testing purpose
 - Develop the control SW on the base of OOAD (UP) and UML
 - Not apply the OOAD to the simulator, which you can develop with any architectures and techniques available
- **The Overall Steps**
 1. **The Inception (UP)**
 - Use cases + CI/CD 환경 구축
 2. **OOA (UP)**
 - Use cases + SSD + Domain model
 3. **OOD (UP)**
 - Use cases + Sequence diagrams + Class Diagram + SOLID 분석 및 적용
 4. **OOI**
 - Unit / System Test 생성 및 실행 (Google Test) + Code Review (수동) + Static Code Analysis
 - Simulator 개발
 5. **OOD with AI**
 - AI를 활용한 SOLID 분석 및 적용
 6. **OOI with AI**
 - AI를 활용한 코드 개발 + AI를 활용한 UT/ST 생성 및 실행 + AI 기반 코드 리뷰
 7. **Proposal of your optimal OOAD approach**

Team Practice #1

Inception - Planning

Inception



* : Not included in the original UP

Out of the scope of this course

Use Cases

Ref. #	Functional Requirements	Use-Case Number & Name	Category	Category
R1.1	Make a new reservation	1. Make Reservation	Primary	Evident
R1.2	Remove an existing reservation	2. Remove Reservation	Primary	Evident
R1.3	Lend an item	3. Lend Item	Primary	Evident
R1.4.1	Return a 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 to reserves	7. Notify Availability	Primary	Hidden
R2.1	Add a title	8. Add Title	Primary	Evident
R2.2	Remove a title	9. Remove Title	Primary	Evident
R2.3	Update a title	10. Update Title	Primary	Evident
R2.4	Add items	11. Add Items	Primary	Evident
R2.5	Remove an item	12. Remove Item	Primary	Evident
R2.6	Update item	13. Update Item	Primary	Evident
R3.1	Add a borrower	14. Add Borrower	Primary	Evident
R3.2	Remove a borrower	15. Remove Borrower	Primary	Evident
R3.3	Update a borrower	16. Update Borrower	Primary	Evident
R4.1	Validates system access (log-in)	17. Log-IN	Secondary	Evident
R4.2	Validates system access (log-out)	18. Log-Out	Secondary	Evident
R5.1	Compute total # of items checked out	19. Count Loans	Secondary	Evident

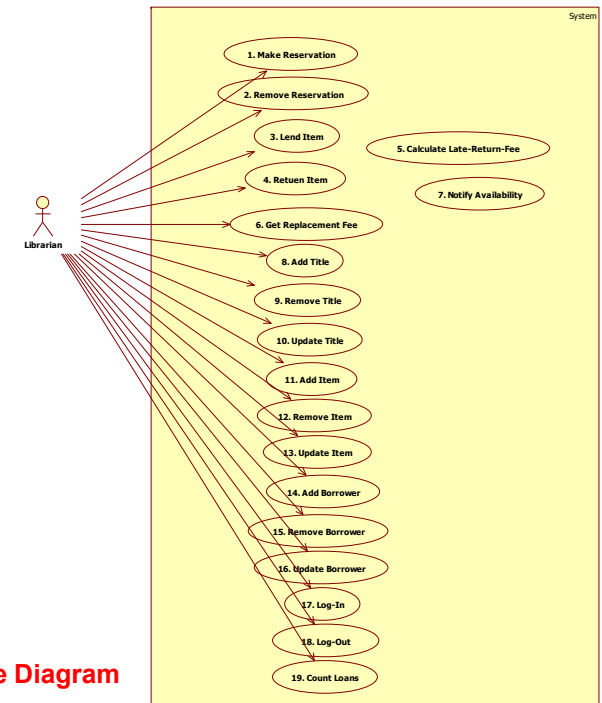
Functional Requirements (≈ Use Cases)

Identified, Derived, Analyzed and Organized
by

Requirements Elicitation & Analysis
at Requirements Analysis Step
in case of Waterfall Model

Use Case Descriptions (Brief format)

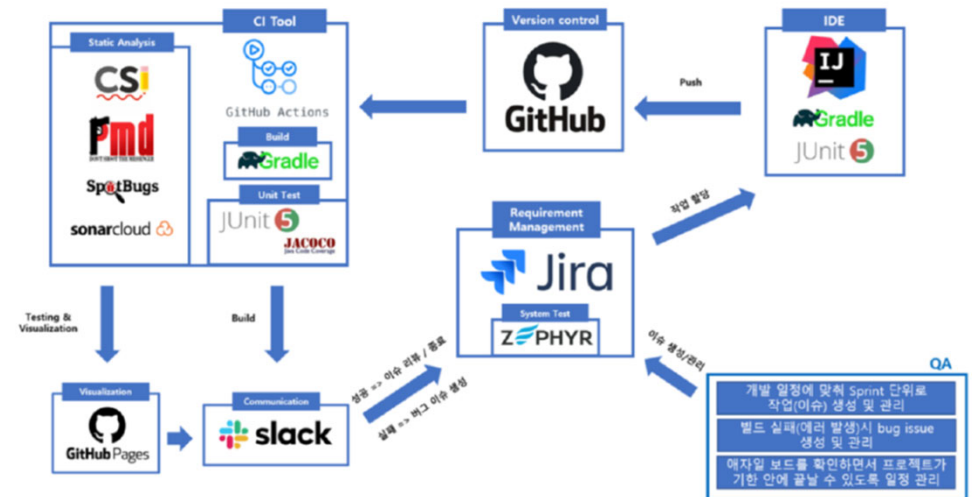
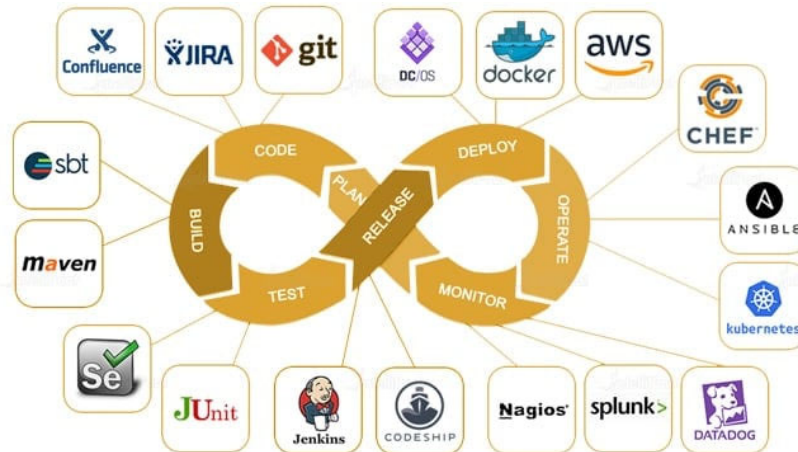
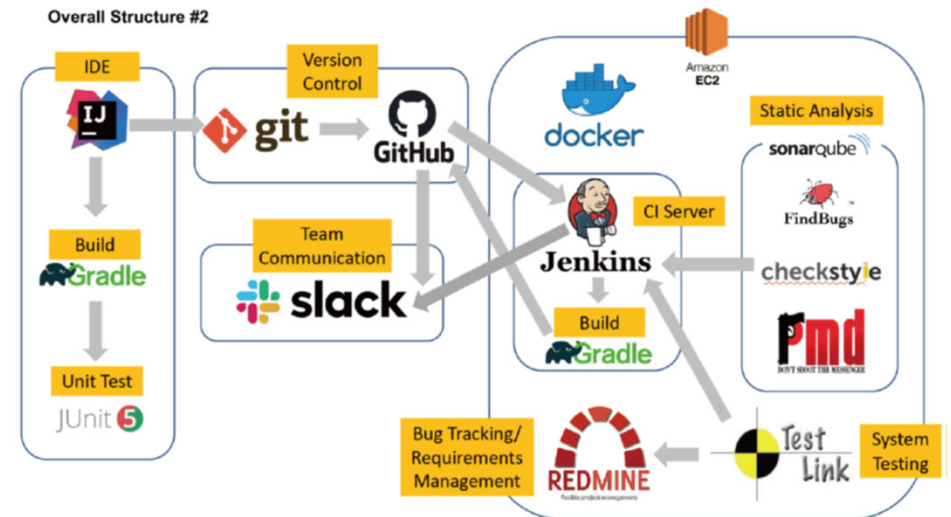
Use Case	1. Make Reservation
Actors	Librarian
Description	<ul style="list-style-type: none"> - This use case begins when a borrower arrives at the counter and then requests reservation. - For a registered borrower, it makes a reservation slip (software-wise). - For an unregistered borrower, the librarian registers the person and makes a reservation for the person.



Use Case Diagram

CI/CD Environment

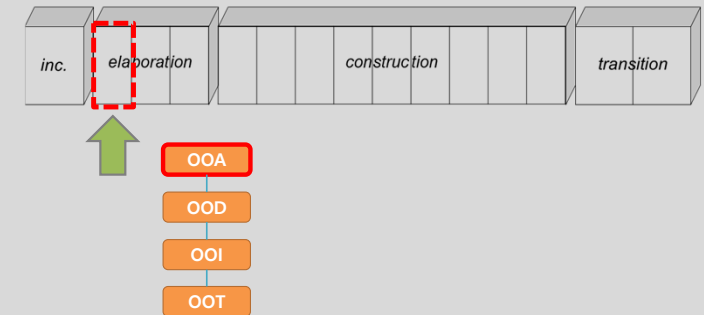
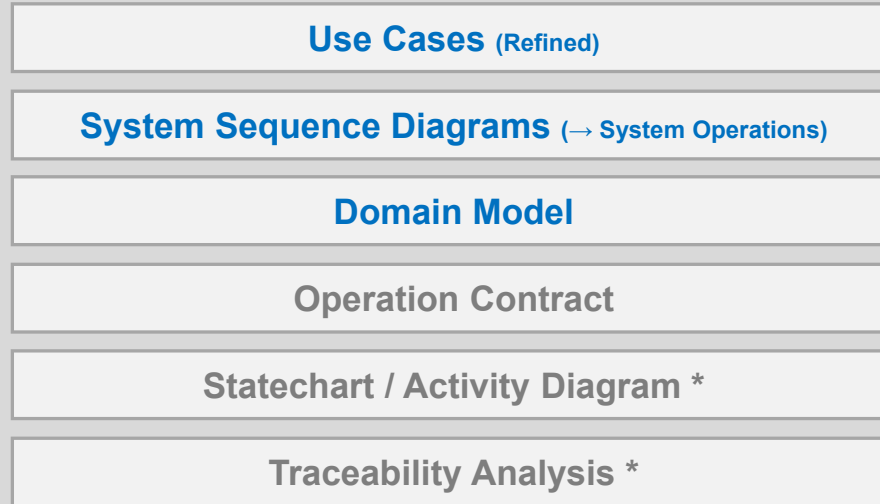
- CI Server
- Code Version Control
- IDE (C++)
- Unit Test Automation (Google Test)
- Static Code Analysis (SonarQube 포함 3개)
- 기타 (요구사항, 협업, 커버리지 계산, 시스템 테스트 지원 등 필요한 만큼)



Practice #2

OOA

OOA in the 1st Elaboration

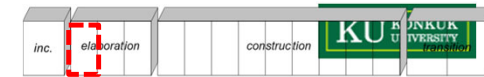


* : Not included in the original UP

Out of the scope of this course

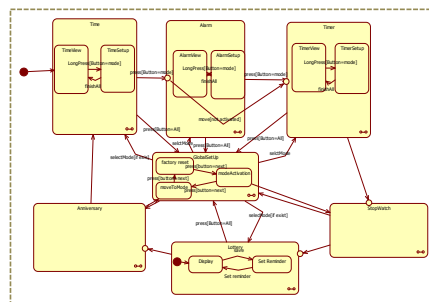
SRS : Software Requirements Specification

OOA

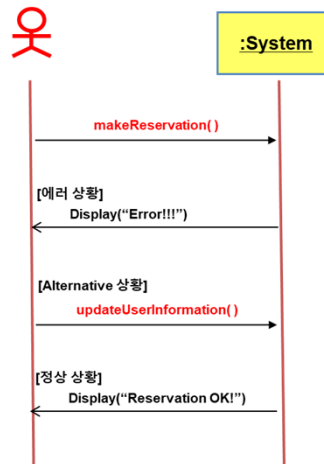


Use Case	1. Make Reservation
Actor	Librarian (Evident)
Purpose	(As in the Inception)
Overview	(As in the Inception)
Type	Primary and Essential
Cross Reference	System Functions: R1.1, R3.1 Use Case: "Add Borrower"
Pre-Requisites	Borrower should have an id_card.
Typical Courses of Events	(A) : Actor, (S) : System 1. (A) A librarian requests the reservation of a title 2. (S) Check if a corresponding title exists 3. (S) Check if a corresponding borrower exists 4. (S) Create a reservation information
Alternative Courses of Events	Line 3: (S) If the borrower's information is out of date, request for the update. (A) A librarian updates up-to-date information of the borrower.
Exceptional Courses of Events	Line 1~3: If invalid reservation information is entered, indicate an appropriate error.

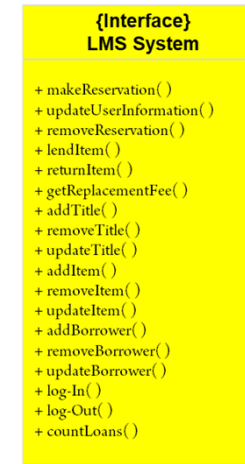
Use Cases



Statechart Diagram (Optional)



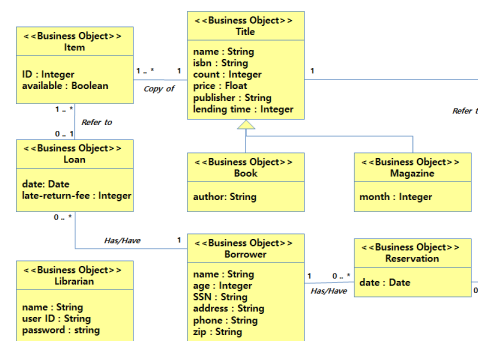
System Sequence Diagrams



System Operations (System Interface)

System Function	Business Object Class	Sequence of Messages to System
Make reservation	+ makeReservation()	+ makeReservation()
Remove reservation	+ removeReservation()	+ removeReservation()
Lend item	+ lendItem()	+ lendItem()
Return item	+ returnItem()	+ returnItem()
Calculate replacement fee	+ getReplacementFee()	+ getReplacementFee()
Add title	+ addTitle()	+ addTitle()
Remove title	+ removeTitle()	+ removeTitle()
Update title	+ updateTitle()	+ updateTitle()
Add item	+ addItem()	+ addItem()
Remove item	+ removeItem()	+ removeItem()
Update item	+ updateItem()	+ updateItem()
Add borrower	+ addBorrower()	+ addBorrower()
Remove borrower	+ removeBorrower()	+ removeBorrower()
Update borrower	+ updateBorrower()	+ updateBorrower()
Log in	+ log-In()	+ log-In()
Log out	+ log-Out()	+ log-Out()
Count loans	+ countLoans()	+ countLoans()

+ Traceability Table

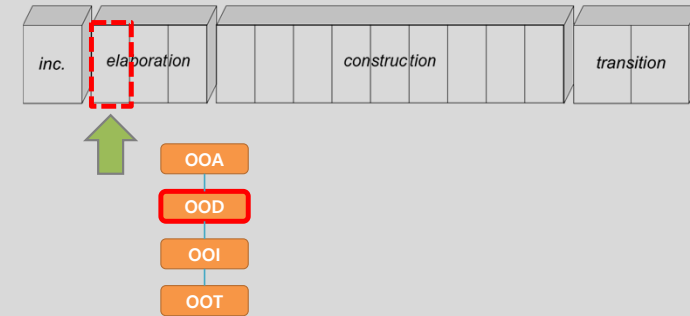
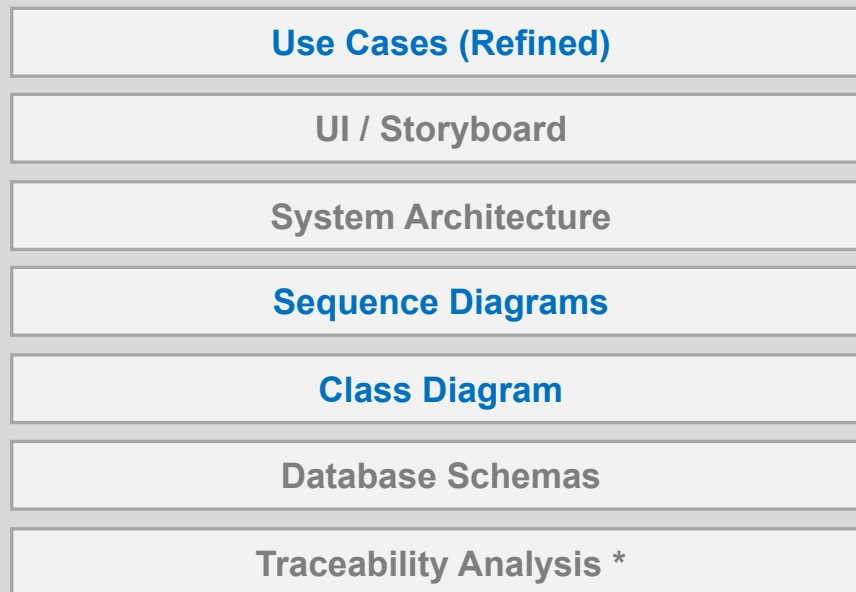


Domain Model

Practice #3

OOD

OOD in the 1st Elaboration

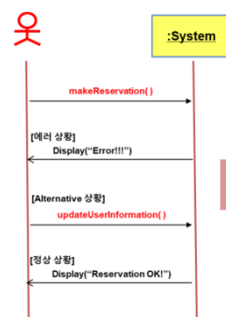


* : Not included in the original UP

Out of the scope of this course

SDS : Software Design Specification

OOD

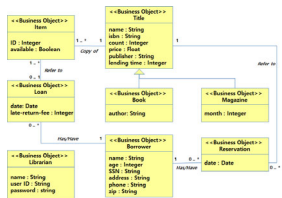


one by one

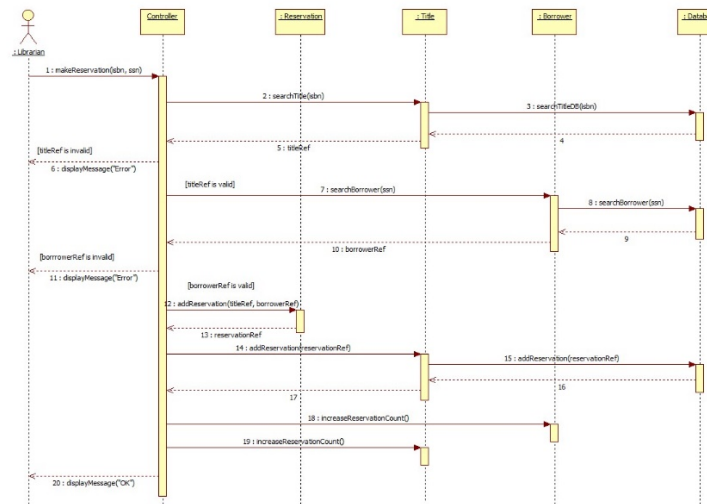
System Operations + SSDs

Use Case	1. Make Reservation
Actor	Librarian (Librarian)
Purpose	(As in the description)
Overview	(As in the description)
Type	Primary and Casual
Cross Reference	System Functions: R1.1, R3.1 Use Case: "Add Borrower"
Pre-Conditions	Borrower should have an ID card.
Typical Courses of Events	(A) Actor: (S) System 1. (S) A librarian requests the reservation of a title. 2. (S) Check if a corresponding title exists. 3. (S) Check if a corresponding borrower exists. 4. (S) Create a reservation information.
Alternative Courses of Events	Line 2: (S) If the borrower's information is out of date, request for the update. (A) A librarian updates up-to-date information of the borrower.
Exceptional Courses of Events	Line 1-3: If model reservation information is entered, indicate an error.

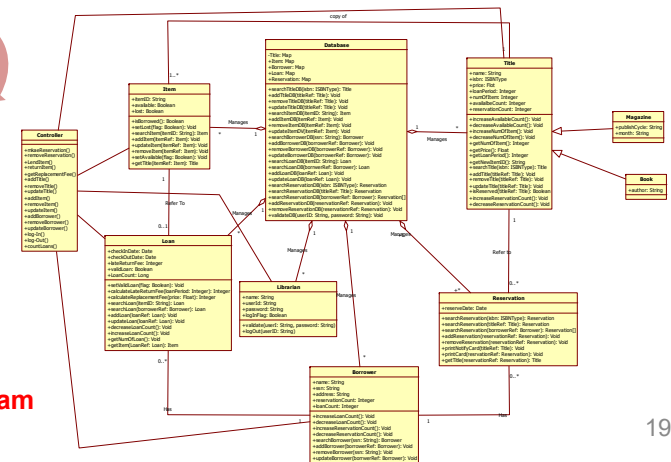
Use Cases



Domain Model



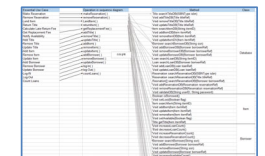
Sequence Diagrams



Class Diagram

System Architecture

Component Diagram

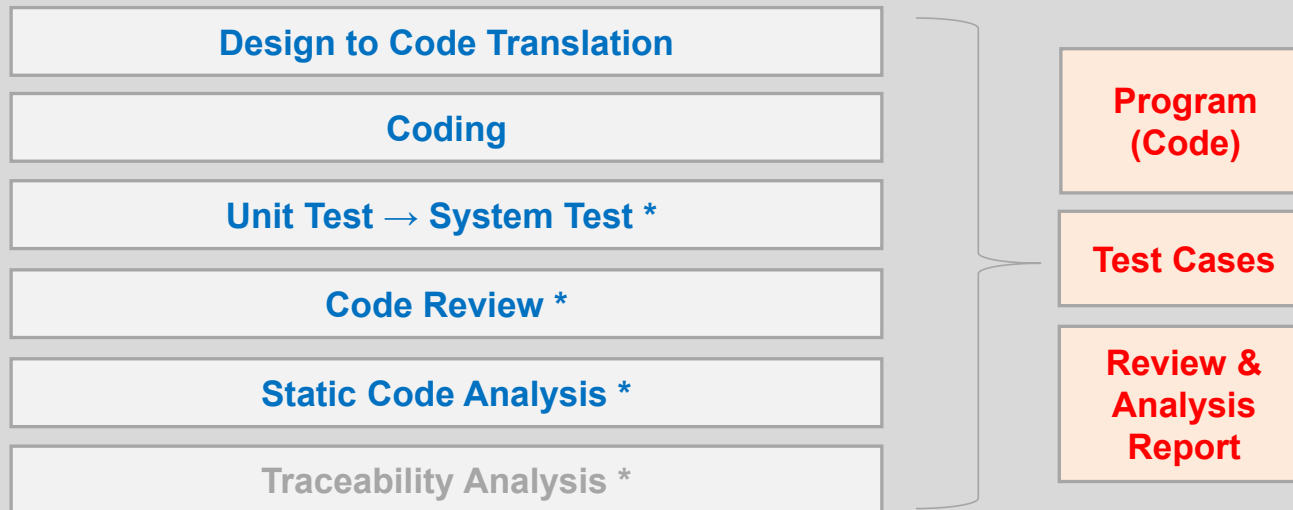
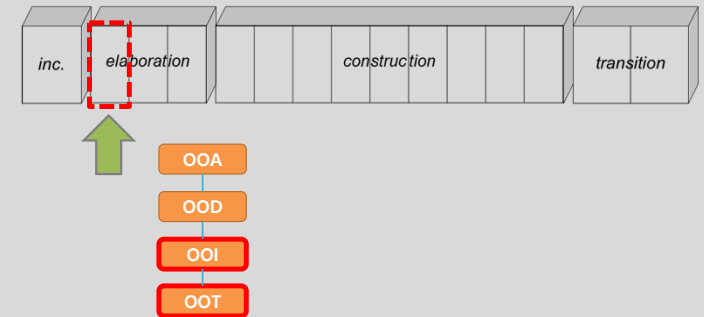


Traceability Table

Practice #4

OOI

OOI in the 1st Elaboration



* : Not included in the original UP

Out of the scope of this course



Code Generation / Translation

Skeleton Code

System Operation

Realization

Sequence Diagrams (Dynamic Model)

with a hint

Code Review

Static Code Analysis

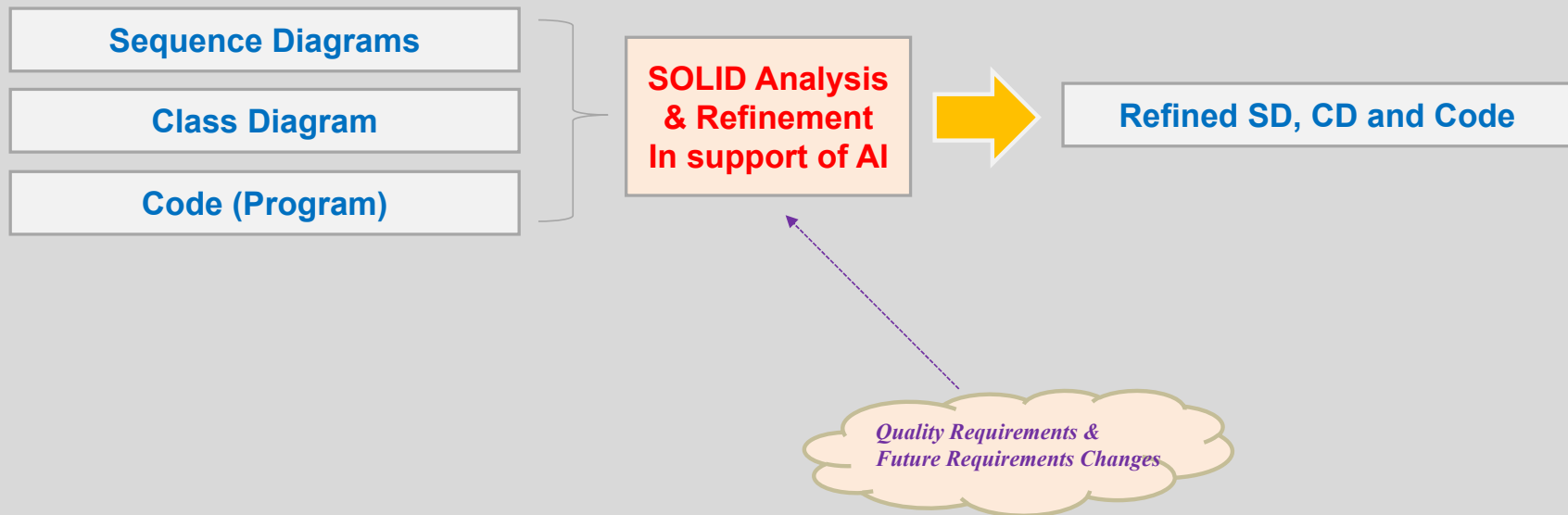
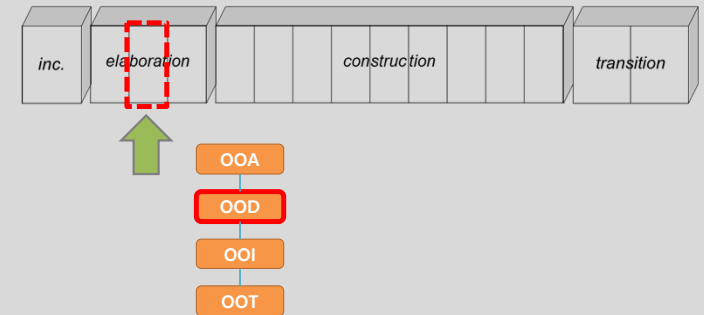
Unit Test 개발

Unit Test 실행

Practice #5

OOD with AI

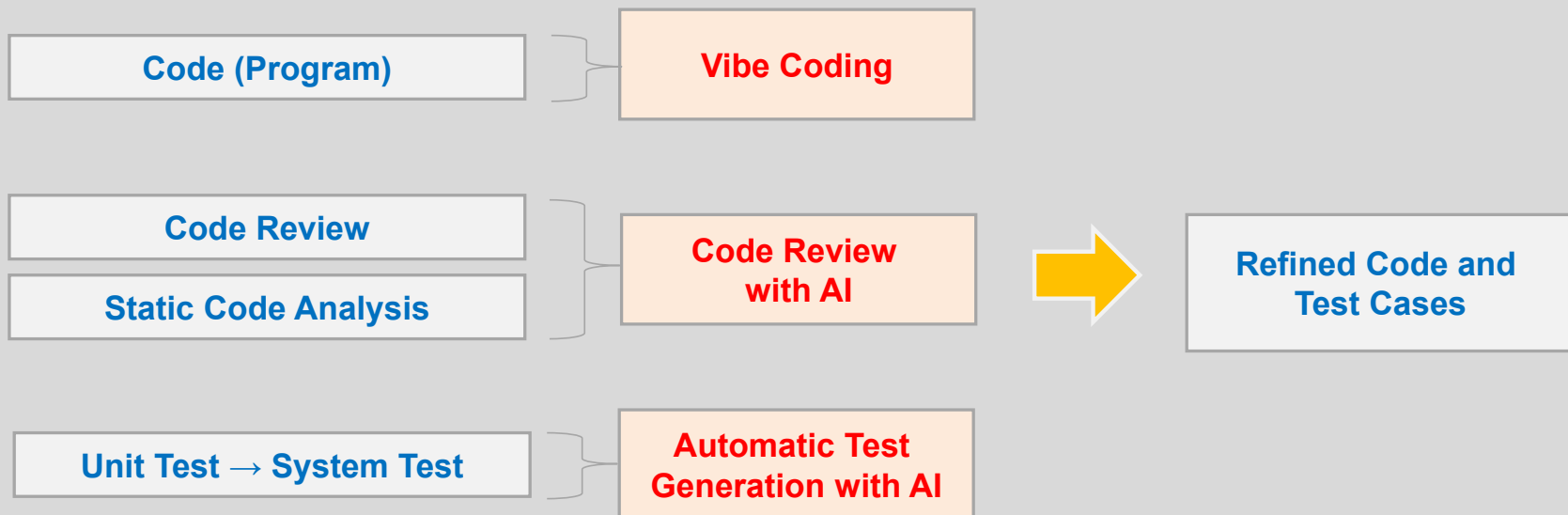
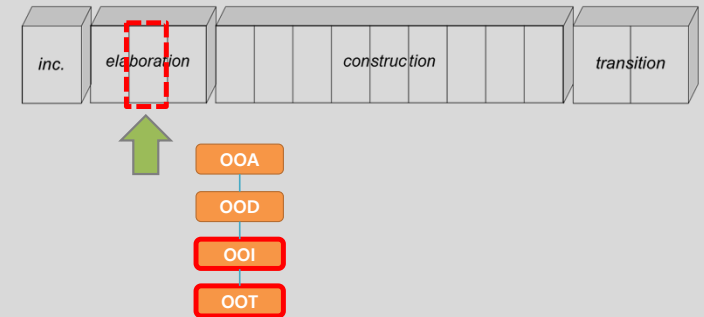
OOD with AI in the 2nd Elaboration



Practice #6

OOI with AI

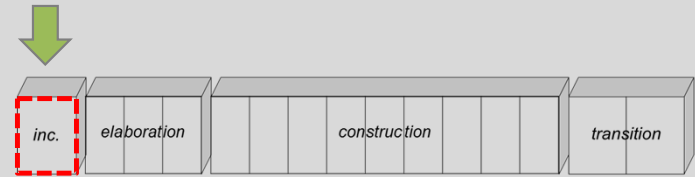
OOI with AI in the 2nd Elaboration



The UP Example

- **LMS** (Library Management System)

Inception



* Not included in the original UP

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 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.

• Resource Estimation

- Human Efforts(Man-Month): 6-10 M/M
- Human Resource: 5M
- Project Duration: 3 Days
- Cost: \$ 3M

• Functional Requirements

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

• Non-Functional Requirements (Quality)

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

• Other Information

- The first version : using 3-Tier Client/Server Architecture
- Future Version : with Web interface

A Preliminary Investigation Report

- **Alternative Solutions**

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

- **Project Justification (Business Demands)**

- From various perspectives such as Cost/Duration/Risk/Effect

- **Risk Reduction Plans**

- First adoption of UP (20) : Try a pilot project using UP
- 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.

- **Risk Management**

Risk	Probability	Significance	Total (PxS)
Lack of OO experience	4	4	16
First adoption of UP	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	1	5	5

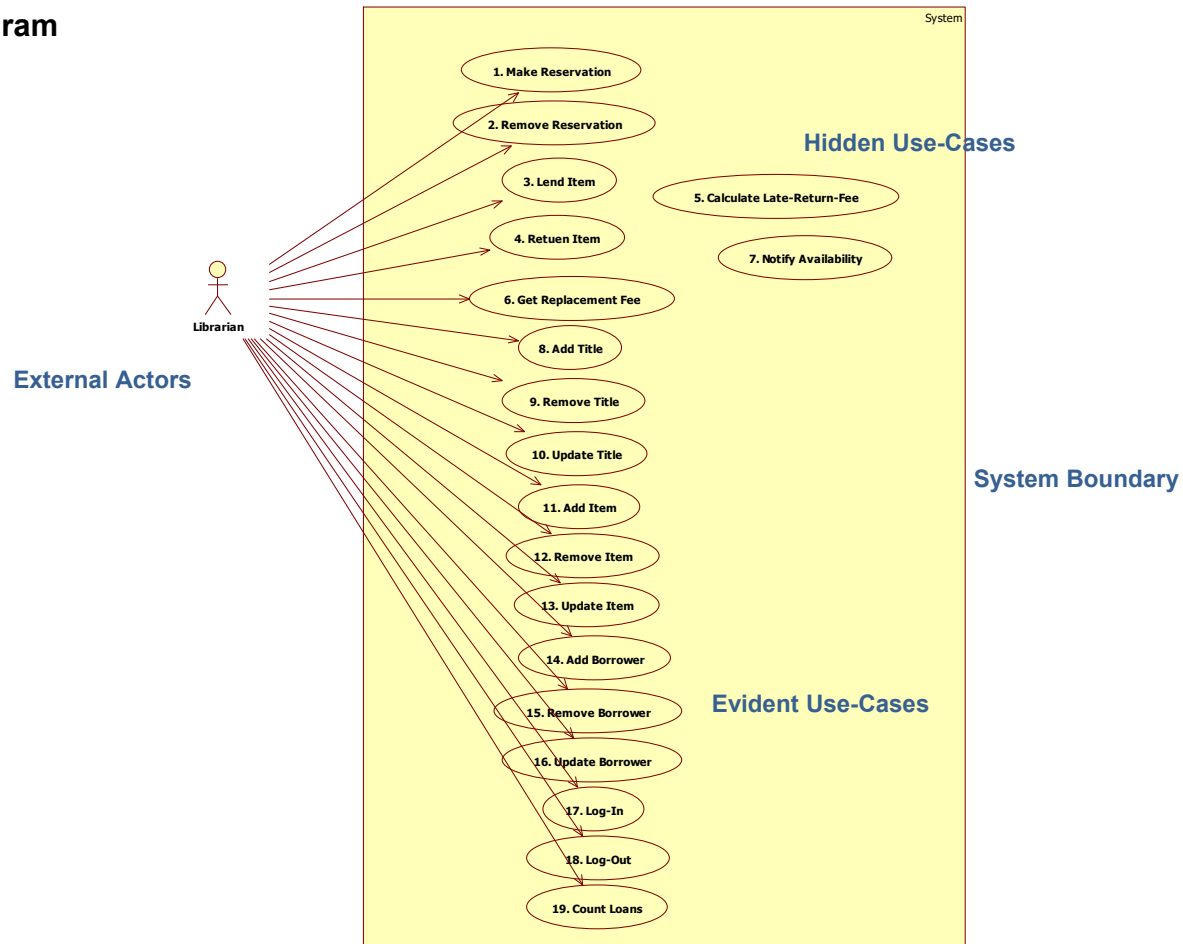
Functional Requirements + Use Cases

- Functional Requirements + Use Cases *

Ref. #	Functional Requirements	Use Case Number & Name	Category	Category
R1.1	Make a new reservation	1. Make Reservation	Primary	Evident
R1.2	Remove an existing reservation	2. Remove Reservation	Primary	Evident
R1.3	Lend an item	3. Lend Item	Primary	Evident
R1.4.1	Return a 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 to reserves	7. Notify Availability	Primary	Hidden
R2.1	Add a title	8. Add Title	Primary	Evident
R2.2	Remove a title	9. Remove Title	Primary	Evident
R2.3	Update a title	10. Update Title	Primary	Evident
R2.4	Add items	11. Add Items	Primary	Evident
R2.5	Remove an item	12. Remove Item	Primary	Evident
R2.6	Update item	13. Update Item	Primary	Evident
R3.1	Add a borrower	14. Add Borrower	Primary	Evident
R3.2	Remove a borrower	15. Remove Borrower	Primary	Evident
R3.3	Update a borrower	16. Update Borrower	Primary	Evident
R4.1	Validates system access (log-in)	17. Log-IN	Secondary	Evident
R4.2	Validates system access (log-out)	18. Log-Out	Secondary	Evident
R5.1	Compute total # of items checked out	19. Count Loans	Secondary	Evident

Functional Requirements + Use Cases

- Use Cases Diagram



Functional Requirements + Use Cases

- Use Cases (Tables in Text)

Use Case	1. Make Reservation
Actors	Librarian
Description	<ul style="list-style-type: none"> - 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	<ul style="list-style-type: none"> - A borrower who made a reservation can cancel his/her reservation. <ul style="list-style-type: none"> • Explicitly cancels the reservation (Evident) - When a borrower checks out an item which he/she previously reserved, this use case is invoked automatically. (Hidden)
Use Case	5. Calculate Late-Return-Fee
Actors	None (Hidden)
Description	<ul style="list-style-type: none"> - This use case can begin when the late items are returned or every midnight. - 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.

Non-Functional Requirements

- **Performance Requirements (Quality)**

- 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.

- **Other Requirements (Quality)**

- The system must control the system access.
- The system should be designed for expandable and maintainable.

- **Operating Environment**

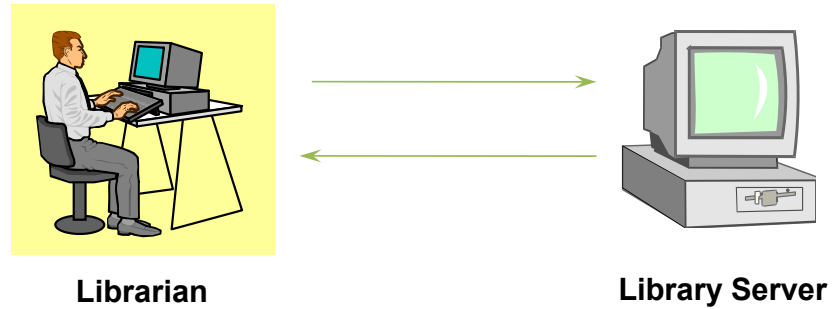
- Microsoft Windows 7 and 10

- **Interface Requirements**

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

A Draft System Architecture

- A Simple Client/Server Architecture



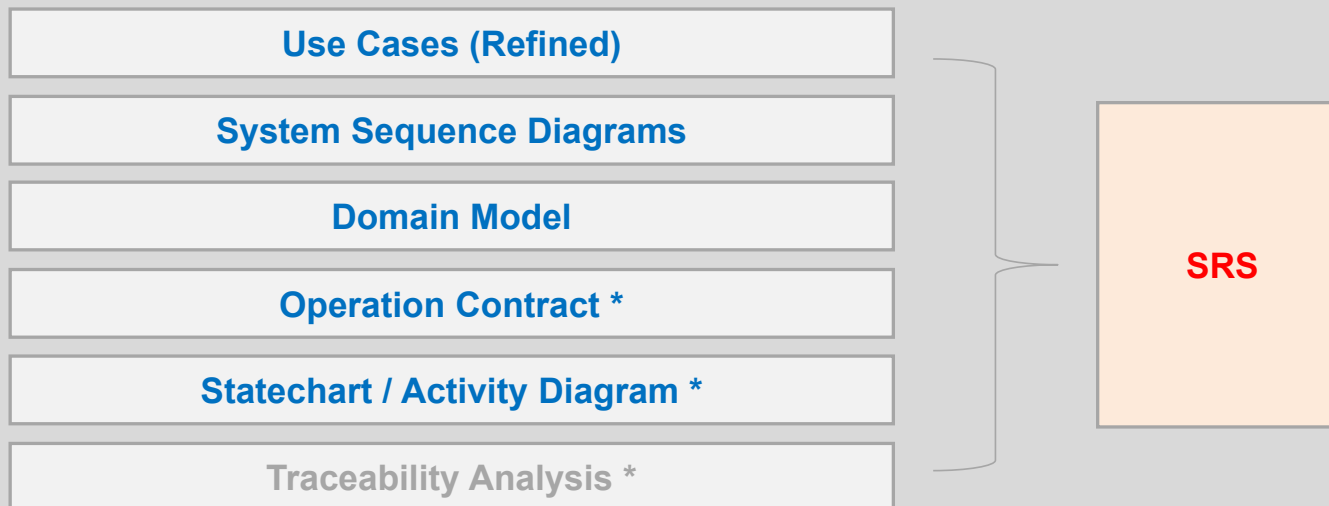
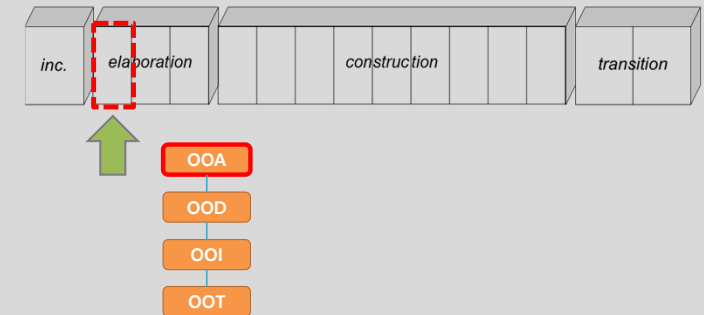
System Test Cases

- System test cases for all or important functional requirements

No.	Functional Requirements (Use Cases)	Test Case Descriptions
1	Make reservation	Correct한 borrower가 correct한 title 예약 (Positive)
2	Make reservation	Correct한 borrower가 incorrect한 title 예약 (Negative)
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
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



OOA in the 1st Elaboration



* Not included in the original UP

* Out of the scope of this course

SRS : Software Requirements Specification

Use Cases (Refined)

- Refined Use Cases (Evident)

Use Case	1. Make Reservation
Actor	Librarian (Evident)
Purpose	(As in the Inception)
Overview	(As in the Inception)
Type	Primary and Casual
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 a title 2. (S) Check if a corresponding title exists 3. (S) Check if a corresponding borrower exists 4. (S) Create a reservation information
Alternative Courses of Events	Line 3: (S) If the borrower's information is out of date, request for the update. (A) A librarian updates up-to-date information of the borrower.
Exceptional Courses of Events	Line 1~3: If invalid reservation information is entered, indicate an error.

Use Cases (Refined)

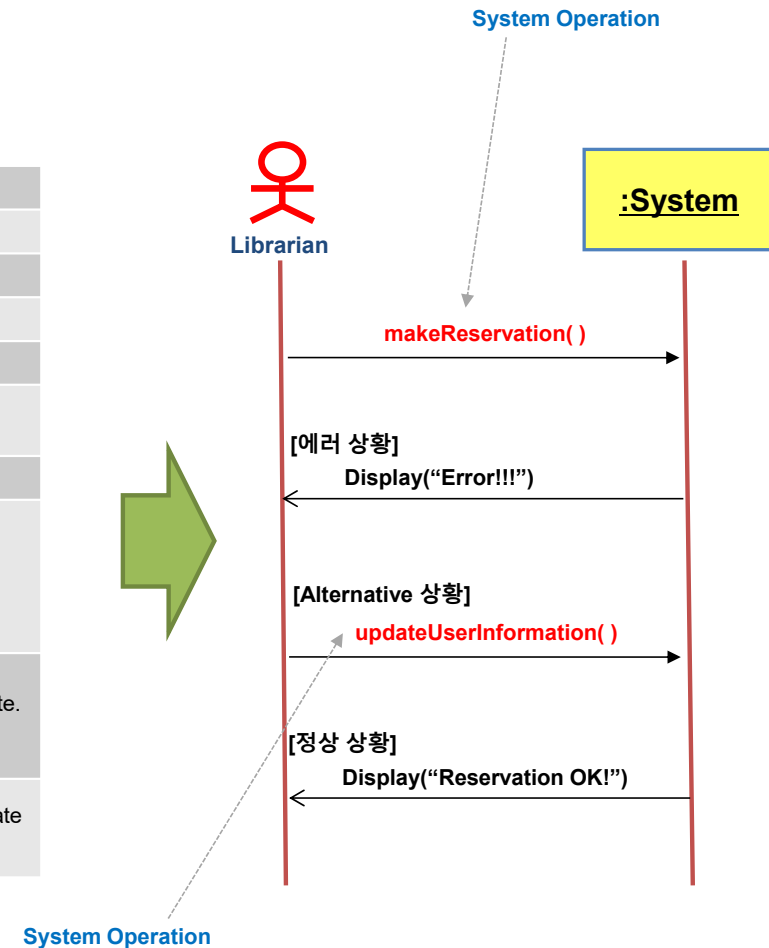
- Refined Use Cases (Hidden)

Use Case	5. Calculate Late-Return-Fee
Actor	None (Hidden)
Purpose	(As in the Inception)
Overview	(As in the inception)
Type	Primary and Casual
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

System Sequence Diagrams

- System Sequence Diagrams (SSD) for all evident Use Cases

Use Case	1. Make Reservation
Actor	Librarian (Evident)
Purpose	(As in the Inception)
Overview	(As in the Inception)
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 a title 2. (S) Check if a corresponding title exists 3. (S) Check if a corresponding borrower exists 4. (S) Create a reservation information
Alternative Courses of Events	Line 3: (S) If the borrower's information is out of date, request for the update. (A) A librarian updates up-to-date information of the borrower.
Exceptional Courses of Events	Line 1~3: If invalid reservation information is entered, indicate an appropriate error.



System Sequence Diagrams

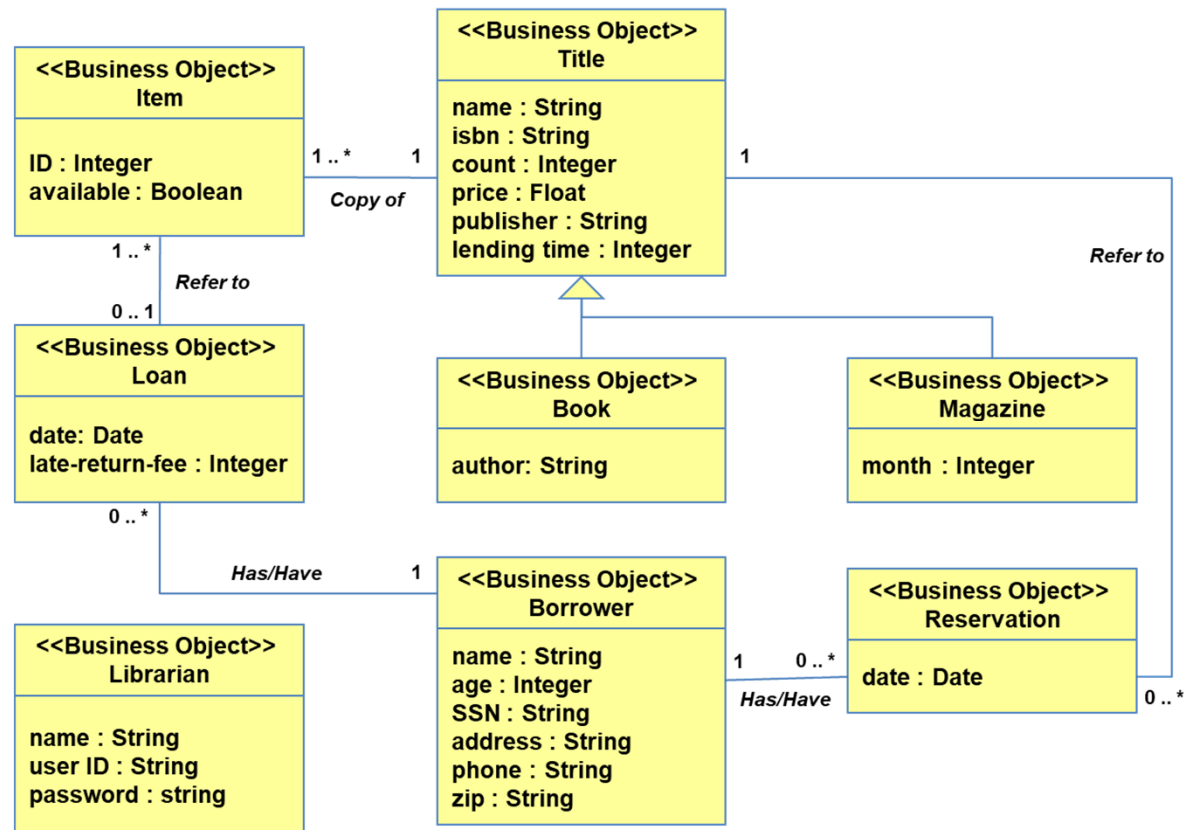
- A set of System Operations → System Interfaces
- Use-Case : System Operation = $N : N$

Use Cass	System Operations
1. Make Reservation	makeReservation() updateUserInformation()
2. Remove Reservation	removeReservation()
3. Lend Item	lendItem()
4. Return Title	returnItem()
5. Calculate Late-Return-Fee	N/A
6. Get Replacement-Fee	getReplacementFee()
7. Notify Availability	N/A
8. Add Title	addTitle()
9. Remove Title	removeTitle()
10. Update Title	updateTitle()
11. Add Items	addItem()
12. Remove Item	removeItem()
13. Update Item	updateItem()
14. Add Borrower	addBorrower()
15. Remove Borrower	removeBorrower()
16. Update Borrower	updateBorrower()
17. Log-IN	log-In()
18. Log-Out	log-Out()
19. Count Loans	countLoans()

{Interface} LMS System
+ makeReservation() + updateUserInformation() + removeReservation() + lendItem() + returnItem() + getReplacementFee() + addTitle() + removeTitle() + updateTitle() + addItem() + removeItem() + updateItem() + addBorrower() + removeBorrower() + updateBorrower() + log-In() + log-Out() + countLoans()

Domain Model

- Domain Model



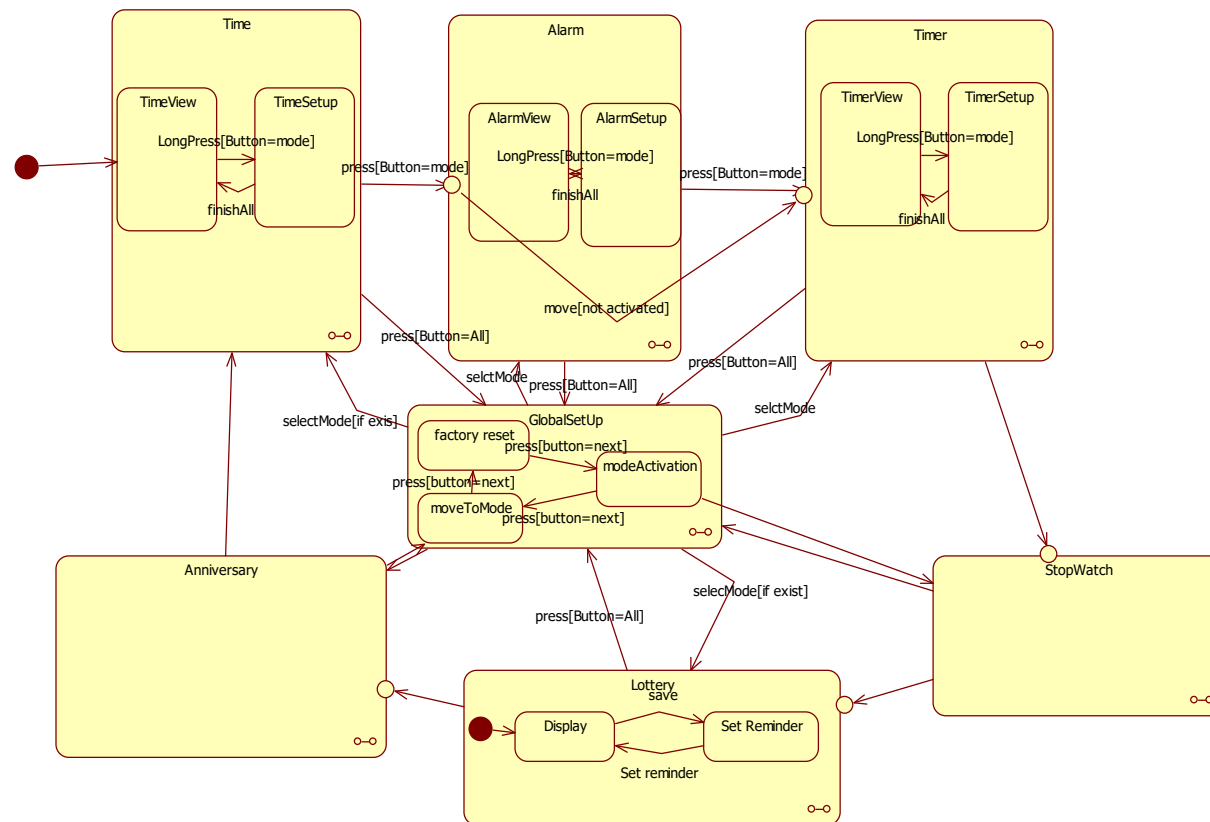
Operation Contracts

- Operation Contracts for all system operations (interfaces)

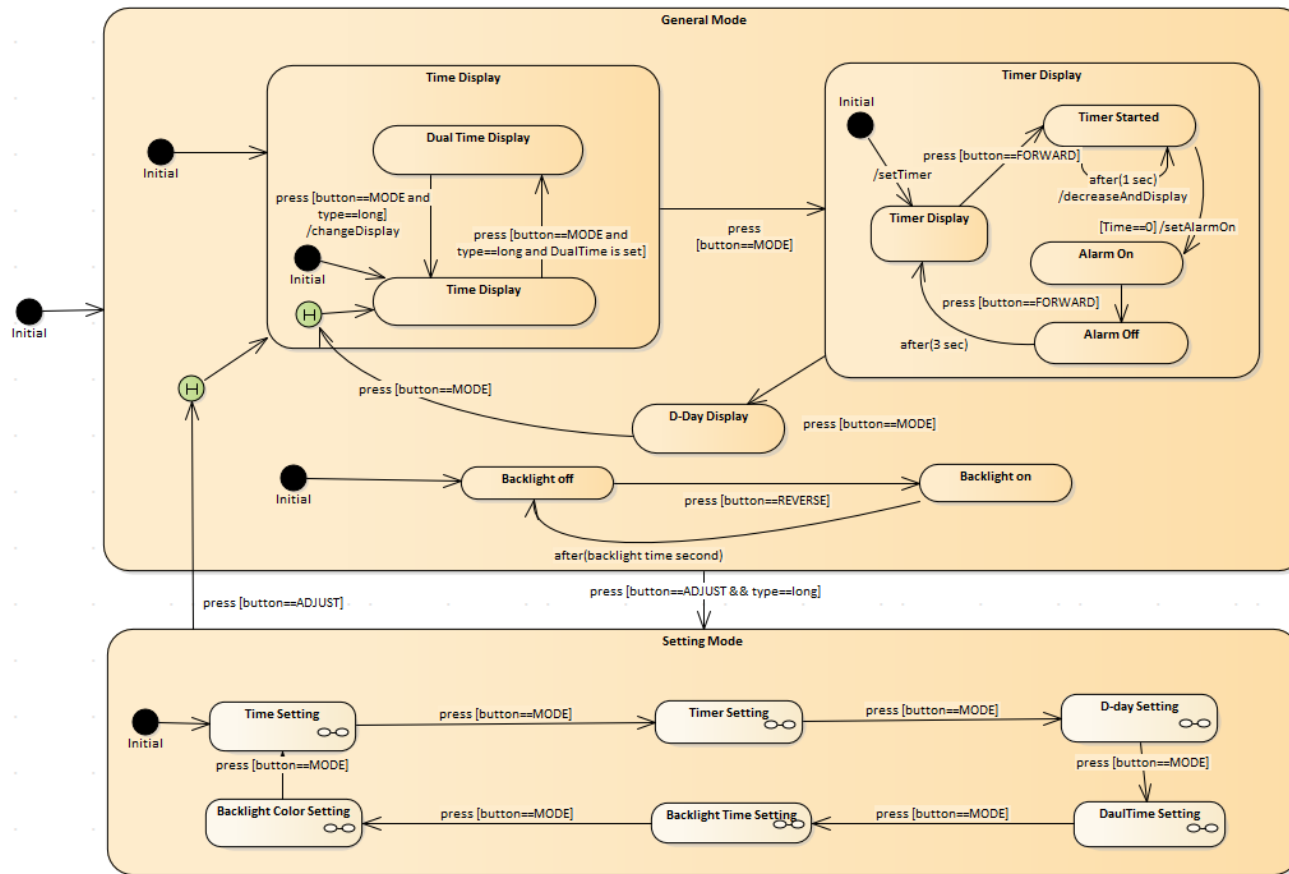
Name	makeReservation()
Responsibilities	Checks if title and borrower information exist, and creates a new reservation
Type	System
Cross References	System functions: R1.1, R2.1
Notes	
Exceptions	N/A
Output	Results from making the reservation
Pre-conditions	Title and Borrower information should be entered.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

Statecharts / Activity Diagram

- Statechart Diagram for state-based systems
- Activity Diagram for business flow-based systems



Statecharts / Activity Diagram



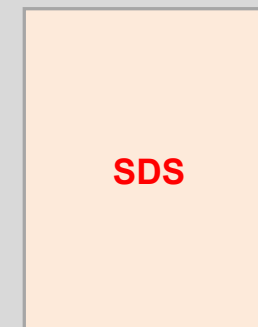
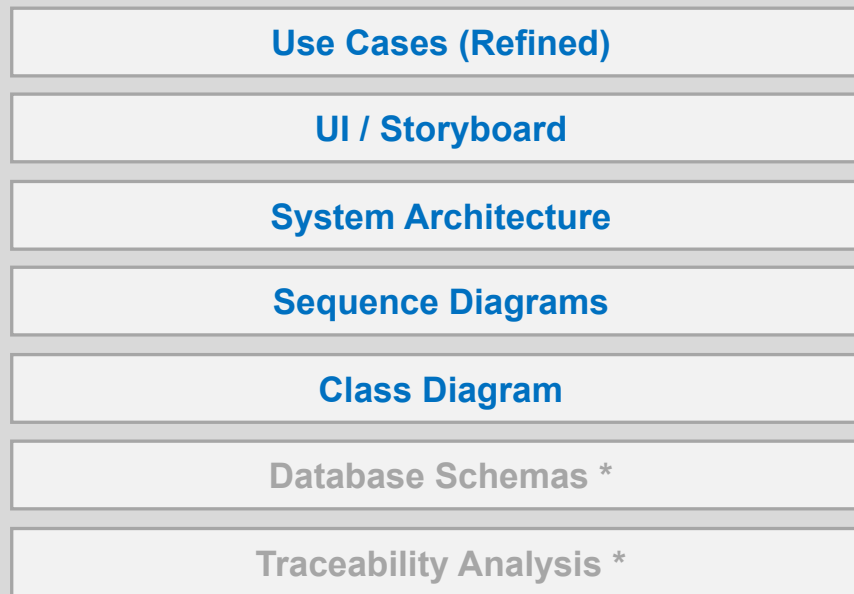
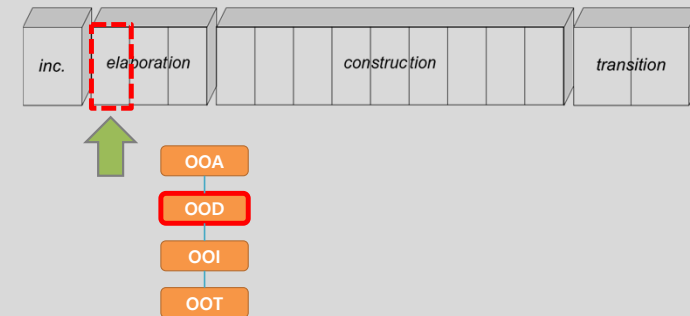
Traceability Analysis

- Traceability Analysis Table

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	UpdateBorrower()
Validates system access	Log-IN	countLoans()
Compute total # of items checked out	Log-Out	
	Count Loans	



OOD in the 1st Elaboration



* Out of the scope of this course

* Not included in the original UP

SDS : Software Design Specification

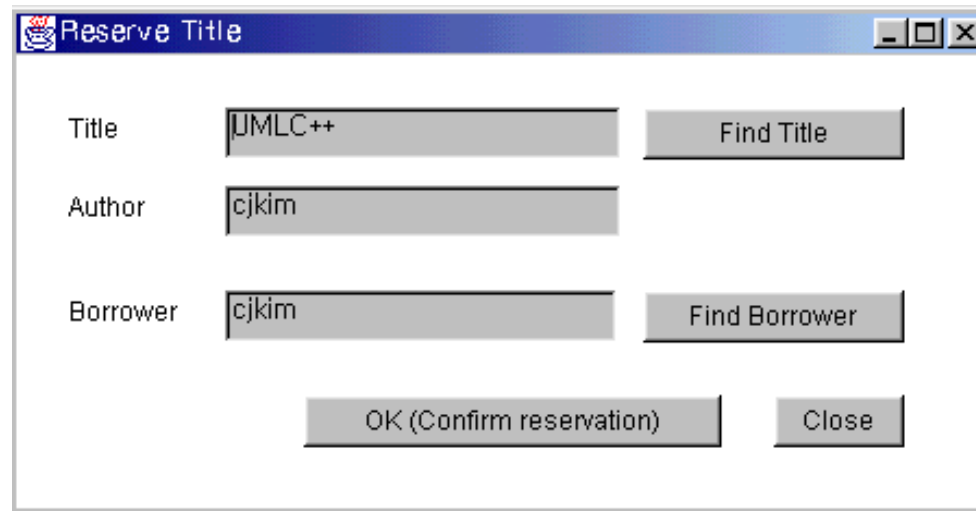
Use Cases (Refined)

- Refined Use Cases

Use Case	1. Make Reservation
Actor	Librarian
Purpose	Create a new reservation
Overview	(As before)
Type	Primary and Fully-Dressed
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 <i>isbn</i> and <i>ssn</i> of the title 2. (S) Find a corresponding <i>title</i> 3. (S) Find a corresponding <i>borrower</i> 4. (S) Create a new <i>reservation</i> 5. (S) Store the new <i>reservation</i> 6. (S) Increase <i>reservationCount</i> in the borrower 7. (S) Increase <i>reservationCount</i> in the title
Alternative Courses of Events	Line 3: (S) If the borrower's information is out of date, request for the update. (A) A librarian updates up-to-date information of the borrower.
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.

UI / Storyboard

- A UI for “Reserve Title”



Reserve Title

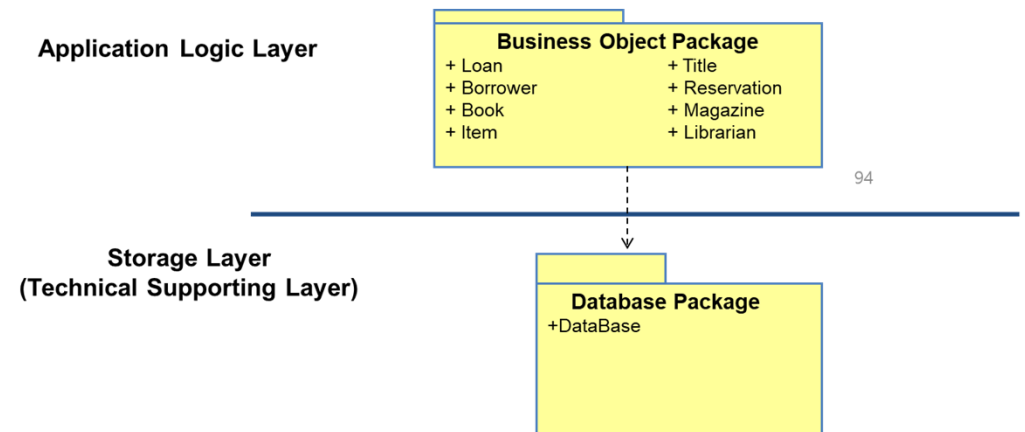
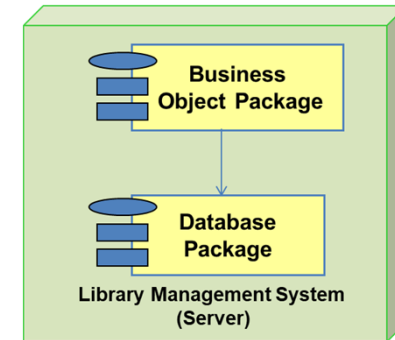
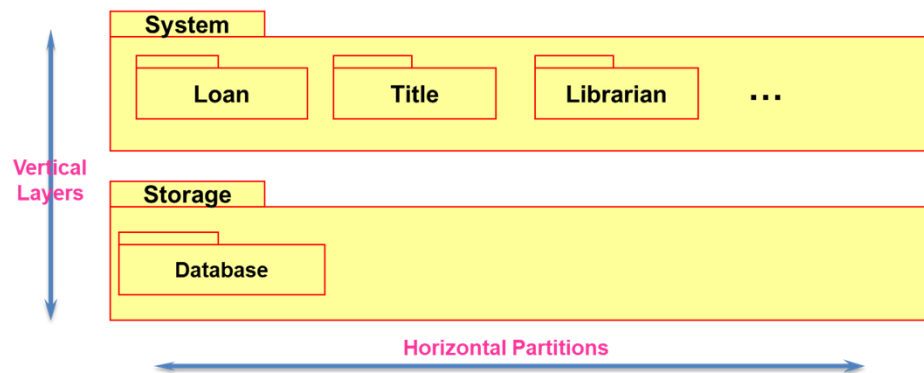
Title

Author

Borrower

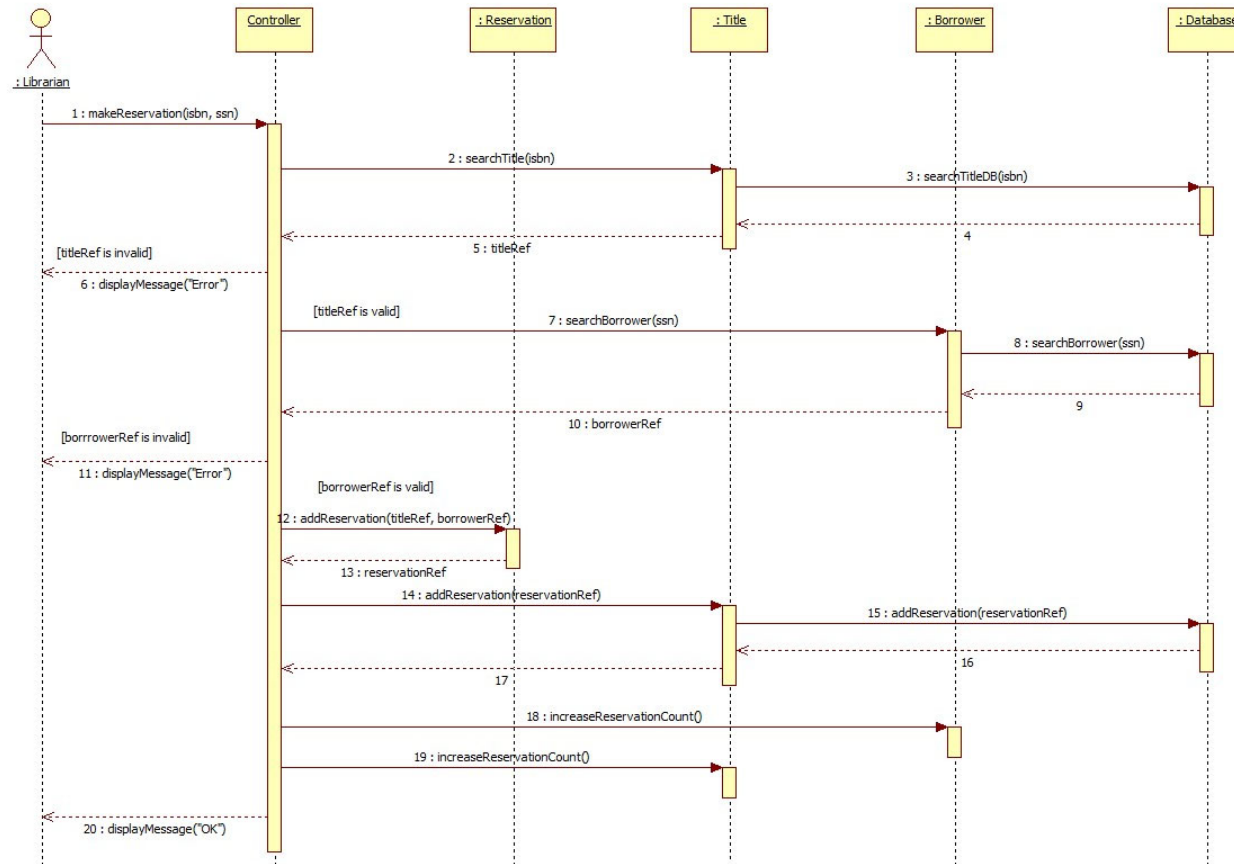
System Architecture

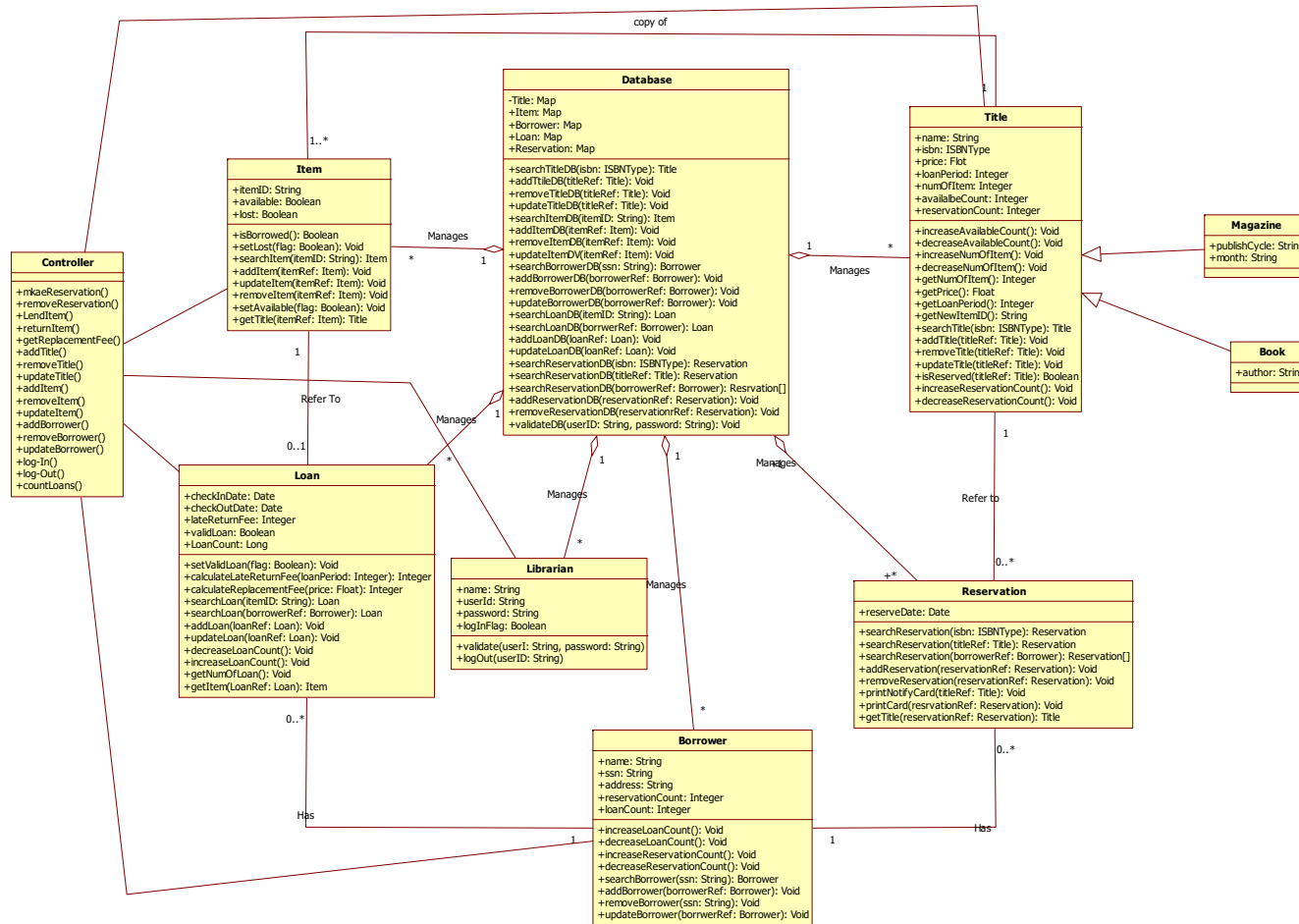
- System Architecture : Client/Server
- Client : A simple terminal
- Server : 2-layered with DB



Sequence Diagrams

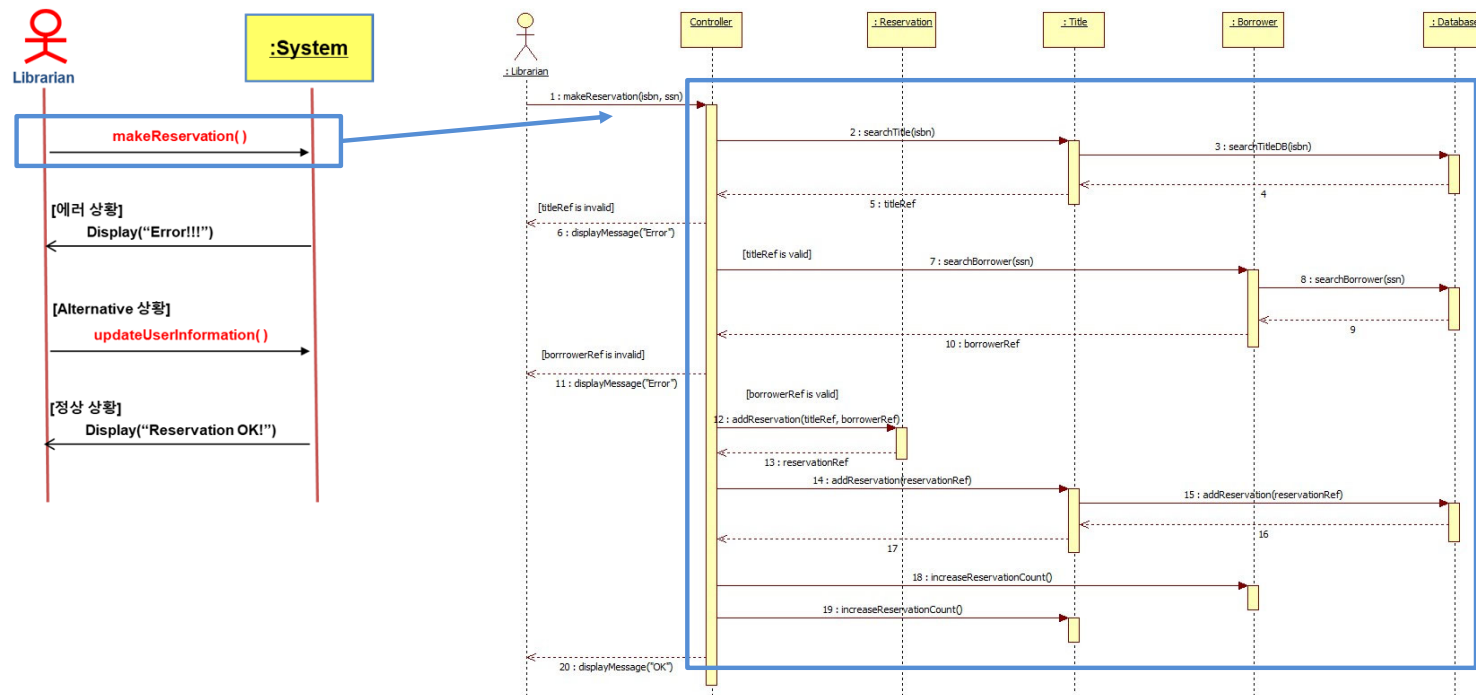
- Sequence Diagrams for all Use Cases (Evident and Hidden)



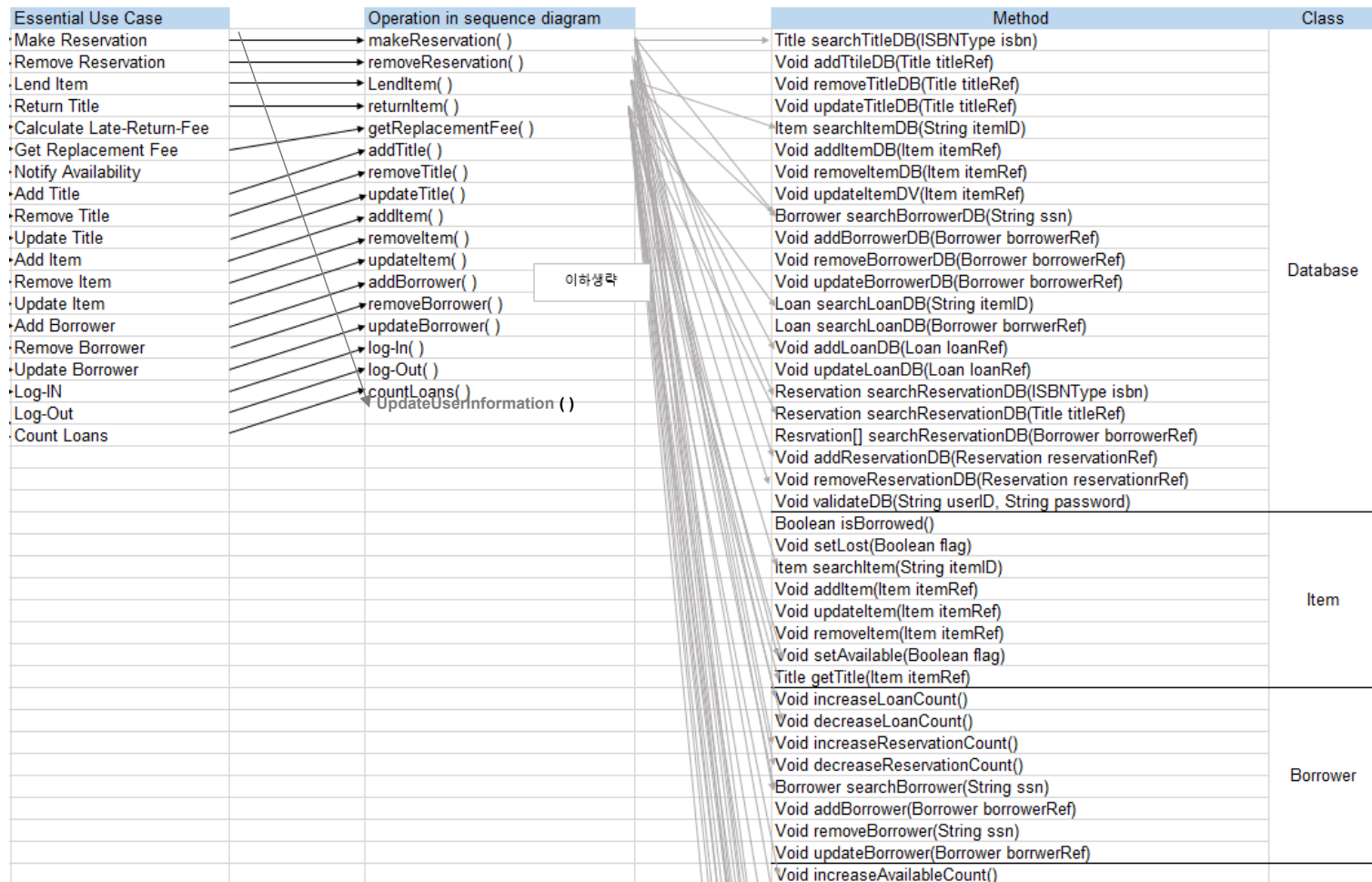


Traceability Analysis

- Traceability Analysis from “a System Operation” to “Operations” consisting it



Traceability Analysis



Traceability Analysis

Essential UseCase	S-Link
시간 확인	S1
시간 설정	S2, S3, S4
알람 표시	S5, S4
알람 설정	S6, S3, S4
알람 추가	S7
알람 삭제	S8
알람 인지	S16
알람 울림 해제	S9
타이머 표시	S10
타이머 설정	S11, S3, S4
타이머 시작	S12
타이머 일시 정지	S13
타이머 초기화	S14
타이머 인지	S17
타이머 울림 해제	S15
스탑워치 시작	S4.1
스탑워치 정지	S4.2
스탑워치 재시작	S4.3
스탑워치 초기화	S4.4
기념일 추가	S5.1, S5.2
기념일 수정	S5.2, S5.3
기념일 삭제	S5.3, S5.4
기념일 알람 해제	S5.5
로또 번호 생성/표시	S6.1
로또 번호 저장/리마	S6.2, S6.3
로또 리마인드 알림	S5, S4
활성 기능 선택	S7.1
공장 초기화	S7.2
현재 모드 표시	
현재 모드 변환	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

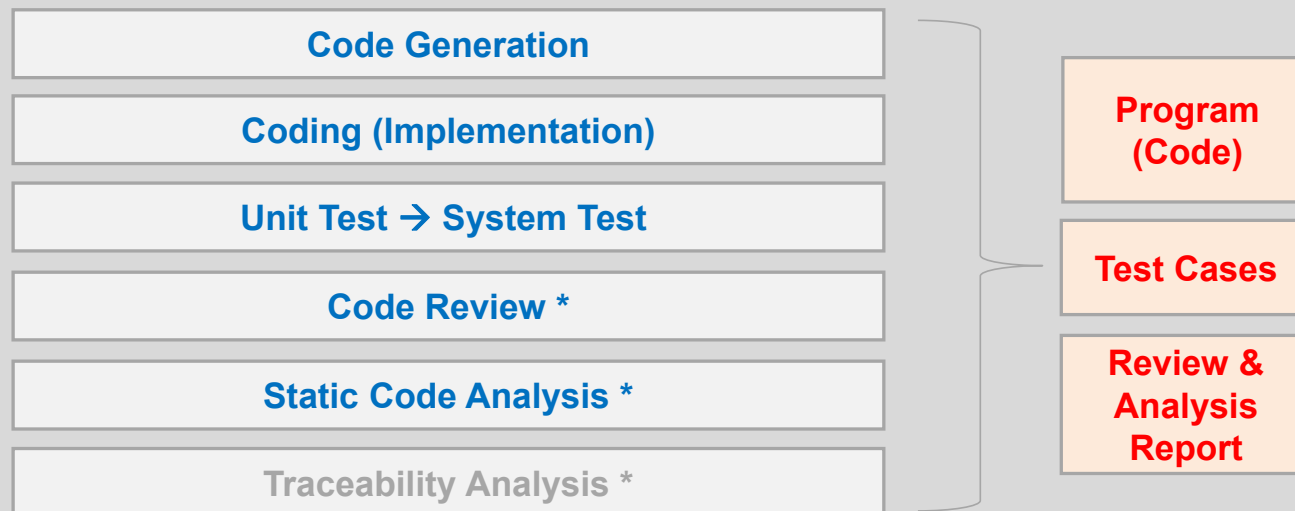
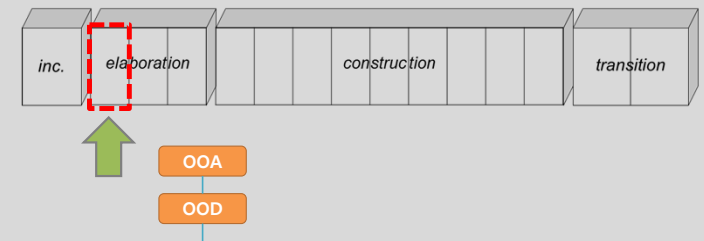
MID	Method	Class
M1	displayCurrentTime	DisplayManager
M2	displaySetupMode	
M3	displayAlarm	
M4	displayNextAlarm	
M5	blinkSetupItem	
M6	displayCurrentMode	
M7	displayTimer	
M8	displaySetupMode	
M9	viewMode	Mode
M10	setupMode	
M11	getPreviousMode	
M12	saveValue	
M13	changeValue	TimeMode
M14	goNext	
M15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

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	displayModelList	
M6.4	updateModelList	
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()

OOI in the 1st Elaboration



* Not included in the original UP

- **Automatic Code Generation with IDE (StarUML)**



Skeleton Codes



Coding (Implementation)

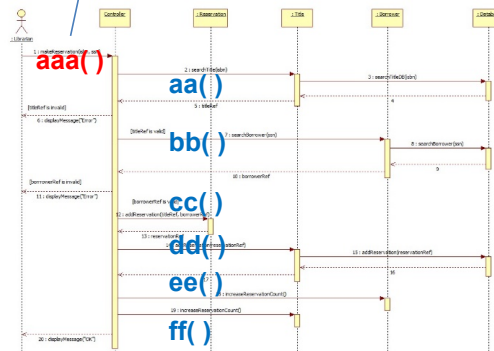
- Realization of the Design into Code

Skeleton Code

```
class A
{
  variables:

  aaa();
  bbb();
  ...
}
```

System
Operation



Sequence Diagrams in OOD

Concrete Code

```
class A
{
  variables:

  aaa()
  {
    aa();
    bb();
    cc();
    dd();
    ee();
    ff();
  }
  bbb();
  ...
}
```

+ 변수 선언
+ 조건문
+ 초기화
+ Visibility 확보



OOAD, in Summary

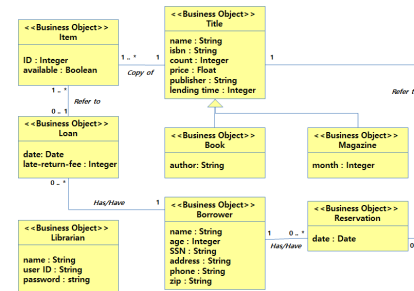
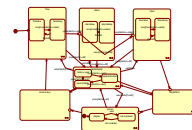
Functional Requirements	Use Cases	Category
R1.1 Make reservation	1. Make Reservation	Evident
R1.2 Remove reservation	2. Remove Reservation	Evident
R1.3 Lend Item	3. Lend Item	Evident
R1.4.1 Return title	4. Return Title	Evident
R1.4.2 Calculate Late-Return-Fee	5. Calculate Late-Return-Fee	Hidden
R1.5 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

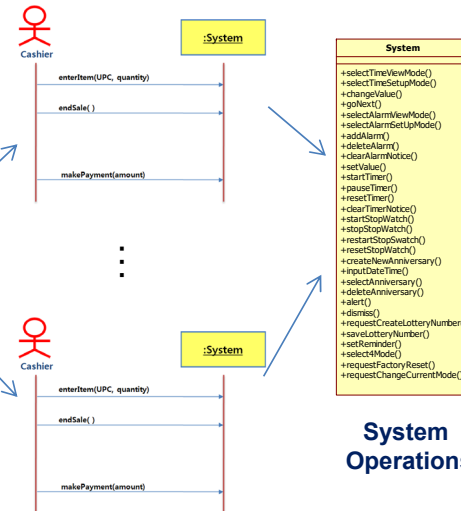
Use Cases

Use Case	1. Make Reservation
Actor	Librarian (Evident)
Purpose	(As in the Inception)
Overview	(As in the Inception)
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 a title 2. (S) Check if a corresponding title exists 3. (S) Check if a corresponding borrower exists 4. (S) Create a reservation information
Alternative Courses of Events	Line 3: (S) If the borrower's information is out of date, request for the update. (A) A librarian updates up-to-date information of the borrower.
Exceptional Courses of Events	Line 1-3: If invalid reservation information is entered, indicate an appropriate error.

Use Cases (refined)

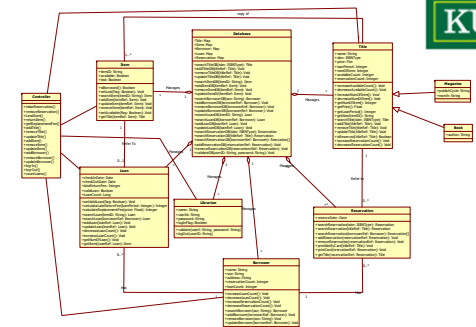


Domain Model

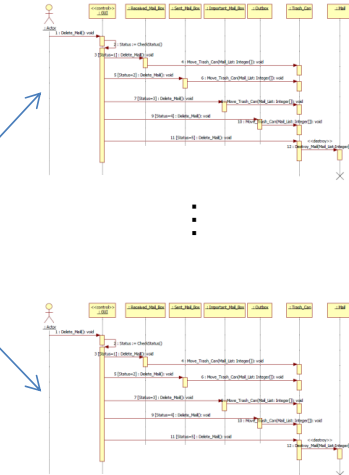


System Sequence Diagrams

System Operations



Class Diagram



Sequence Diagrams

Use Case	Sequence Diagram
1. Make Reservation	SequenceDiagram1
2. Remove Reservation	SequenceDiagram2
3. Lend Item	SequenceDiagram3
4. Return Title	SequenceDiagram4
5. Calculate Late-Return-Fee	SequenceDiagram5
6. Get Replacement Fee	SequenceDiagram6
7. Notify Availability	SequenceDiagram7
8. Add Title	SequenceDiagram8
9. Remove Title	SequenceDiagram9
10. Update Title	SequenceDiagram10
11. Add Item	SequenceDiagram11
12. Remove Item	SequenceDiagram12
13. Update Item	SequenceDiagram13
14. Add Borrower	SequenceDiagram14
15. Remove Borrower	SequenceDiagram15
16. Update Borrower	SequenceDiagram16
17. Log-IN	SequenceDiagram17
18. Log-Out	SequenceDiagram18
19. Count Loans	SequenceDiagram19

Traceability Table

Inception

OOA

OOD

