A New OOO Digital Watch

- Supposed to develop a new OOO digital watch.

- Let's analyze and design your own new OOO digital watch.
  - OOAD development method : OOPT
  - Use a UML tool
    - Not use Communication, Activity, Package, Deployment Diagrams, for now
  - Basic Requirements & Assumptions :
    - A set of predefined/fixed hardware (1 LCD, 4 buttons, 1 buzzer, 1 SW controller)
    - Dynamic SW Configuration (4 activated in 6 functions)
      - OOO, Timekeeping, Timer, Alarm, Stopwatch, World Time
    - Up to 4 alarms
    - GUI : Web-based UI
  - Instructions
    - Take care of the layered architecture of your system under development
    - Take care of your system context - embedded system
    - Make every assumptions clear, feasible and consistent
    - Our OOAD(OOPT) project focuses on a control SW in your digital watch
    - (Web-based) GUI are implemented on your own, not following the OOAD process.

- **Team activities:**
  1. Stage 1000 : Plan
  2. Stage 2000 > 2030 : Analyze
  3. Stage 2000 > 2040 : Design
  4. Stage 2000 > 2050 : Implementation
  5. System Testing
  6. Static Analysis

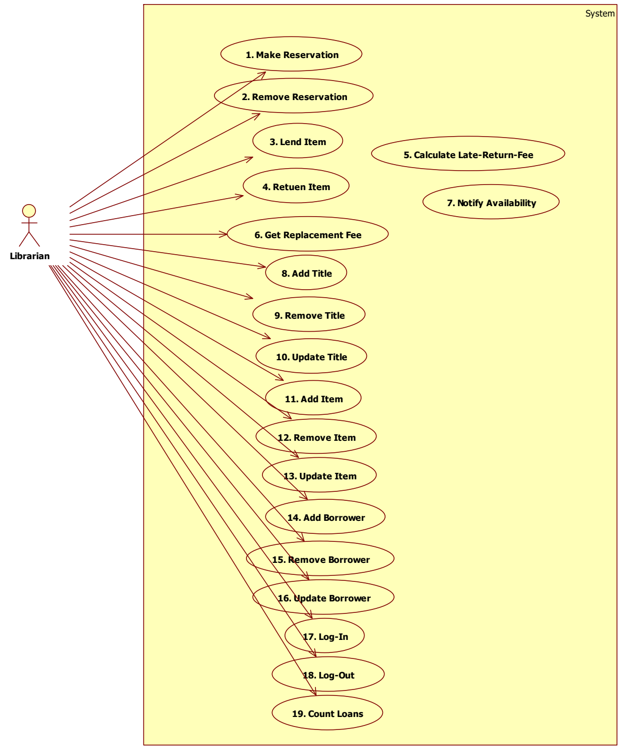
[Team Activity #1] Stage 1000. Planning

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

Functional Requirements ≈ Use Case

Use Case Description (Brief format)

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 Diagram

Overview

Case Study

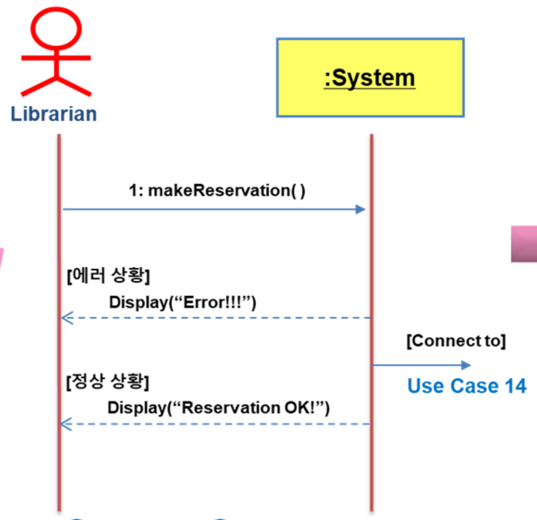
DEPENDABLE SOFTWARE LABORATORY

[Team Activity #2] 2030. Object-Oriented Analysis

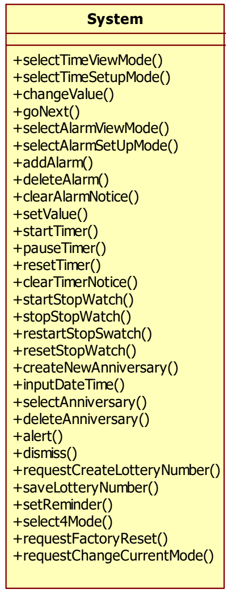
USE CASE: 1. Make Reservation

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 other Use Case)
5. (S) Create reservation information.

Use Cases
(Casual)

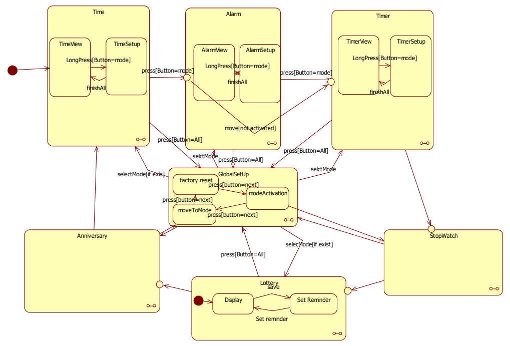


System Sequence Diagrams

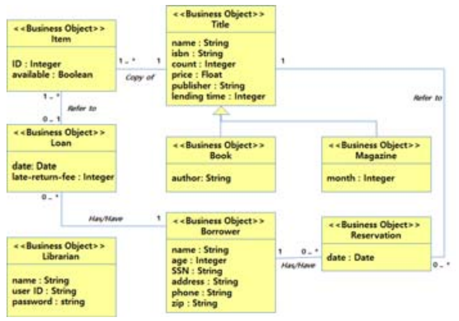


System Operations
(System Interface)

Traceability Table



Statechart Diagram
(optional)



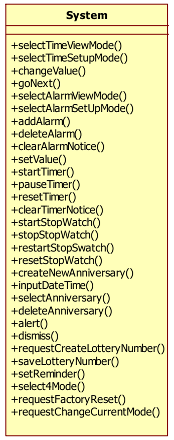
Domain Model

Overview

Case Study

Dependable Software Laboratory

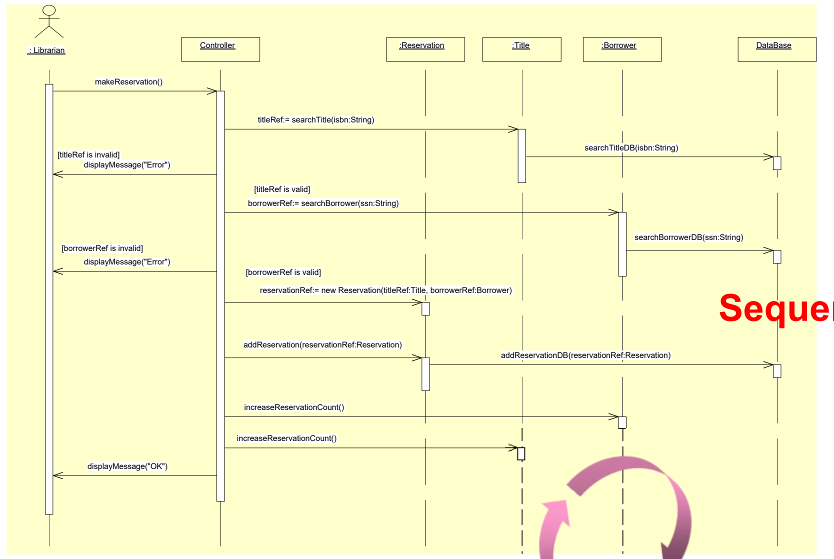
[Team Activity #3] 2040. Object-Oriented Design



System

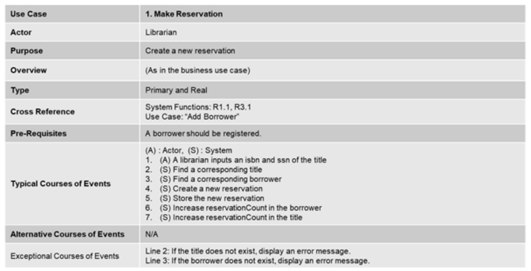
+selectTimeViewMode()
+selectTimeSetupMode()
+changeValue()
+goNext()
+selectAlarmViewMode()
+selectAlarmSetUpMode()
+addAlarm()
+deleteAlarm()
+clearAlarmNotice()
+setValue()
+startTimer()
+pauseTimer()
+resetTimer()
+clearTimerNotice()
+startStopWatch()
+stopStopWatch()
+restartStopSwatch()
+resetStopWatch()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

System Operations



: 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")

Sequence Diagrams



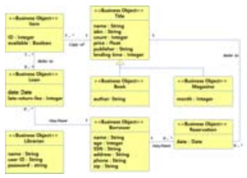
Use Case	1. Make Reservation
Actor	Librarian
Purpose	Create a new reservation
Overview	(As in the business use case)
Type	Primary and Real
Cross Reference	System Functions: R1.1, R3.1 Use Case: "Add Borrower"
Pre-Requisites	A borrower should be registered.
Typical Courses of Events	(A) : Actor, (S) : 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.

Use Cases
(Fully dressed-up)

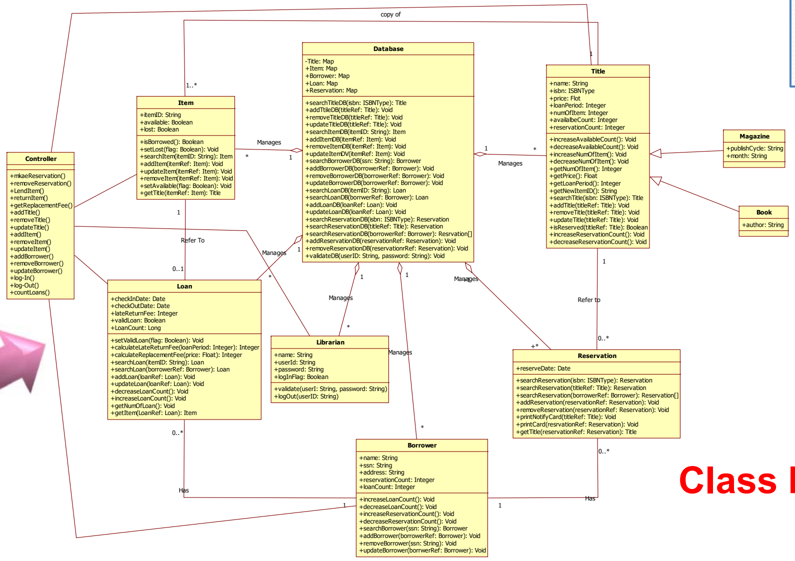
Traceability Table

Overview

Case Study



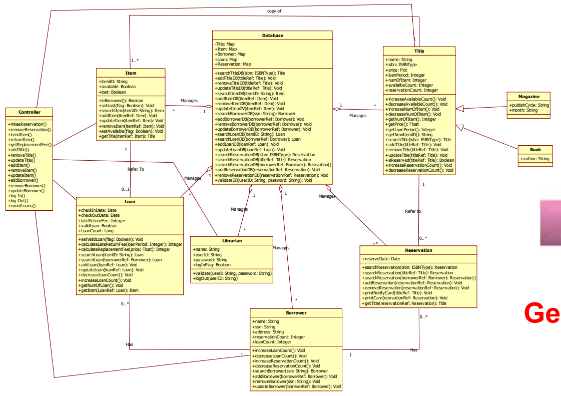
Domain Model



copy of
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
1..*
Manages
1
Refer To
0..1
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
0..*
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
Refer to
0..*
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
Manages
Has

Class Diagram

[Team Activity #4] 2050. Object-Oriented Implementation



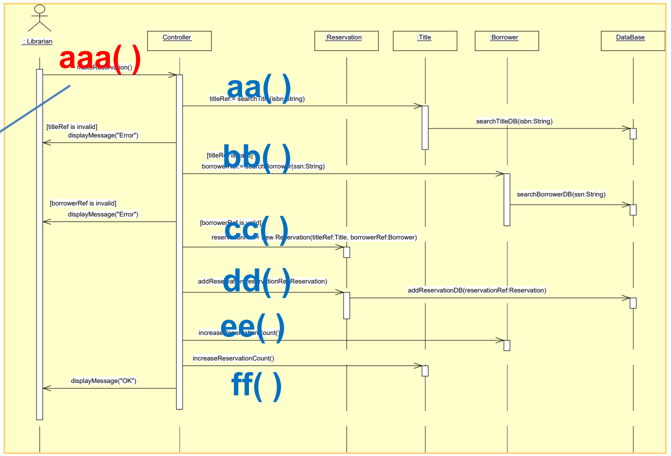
Class Diagram
(Static Model)

Code
Generation

Skeleton Code

```
class  A
{
   variables:

   aaa();
   bbb();
…
}
```

System
Operation



:Librarian
Controller
:Reservation
:Title
:Borrower
DataBase
aaa( )
aa( )
titleRef = searchTitle(isbn:String)
searchTitleDB(isbn:String)
[titleRef is invalid]
displayMessage("Error")
bb( )
borrowerRef = searchBorrower(ssn:String)
searchBorrowerDB(ssn:String)
[borrowerRef is invalid]
displayMessage("Error")
cc( )
new Reservation(titleRef:Title, borrowerRef:Borrower)
dd( )
addReservationDB(reservationRef:Reservation)
ee( )
increaseReservationCount()
ff( )
displayMessage("OK")

Sequence Diagrams
(Dynamic Model)

Realization

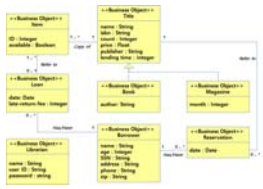
```
class  A
{
   variables:

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

+ 변수 선언
+ 컨디션문
+ 초기화
+ Visibility 확보

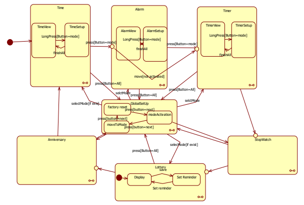
[Team Activity #5 #6] System Testing & Static Analysis

- 4학년 검증팀에서 수행한 System Testing 및 Static Analysis 결과를 검토하여, OOO Digital Watch System을 수정합니다.
  – 결과 반영 및 수정 내용을 각 팀별 CTIP 환경을 사용하여 정리하고, 보고서도 제출합니다.
  – 4학년 검증팀과의 모든 소통은 기본적으로 팀별 CTIP 환경을 사용합니다.

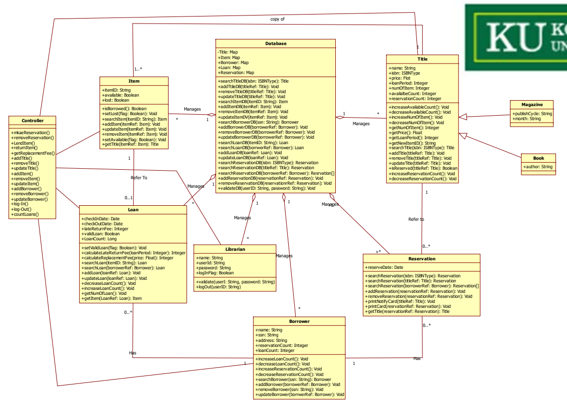
OOPT of OOAD, in Summary



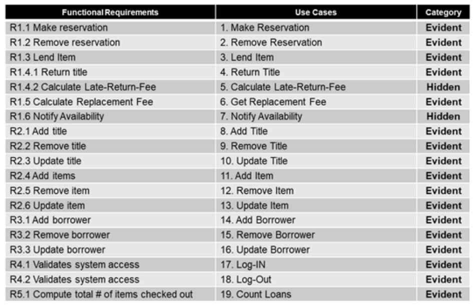
Domain Model



Statechart Diagram



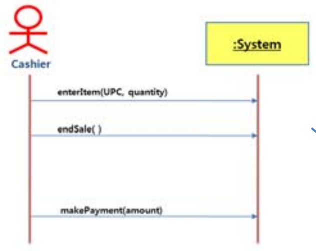
Class Diagram



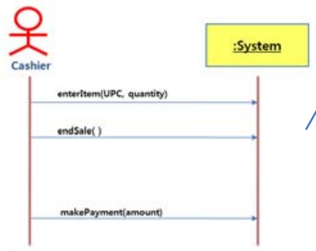
Functional Requirements	Use Cases	Category
R1.1 Make reservation	1. Make Reservation	Evident
R1.2 Remove reservation	2. Remove Reservation	Evident
R1.3 Lend Item	3. Lend Item	Evident
R1.4.1 Return title	4. Return Title	Evident
R1.4.2 Calculate Late-Return-Fee	5. Calculate Late-Return-Fee	Hidden
R1.5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

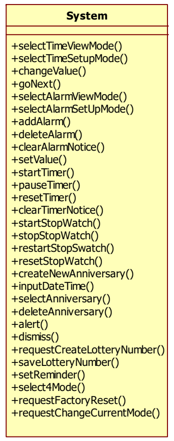
Use Cases



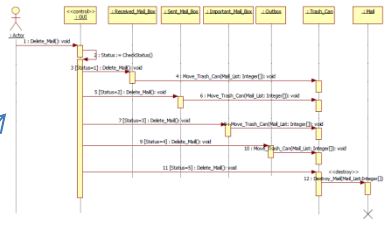
:



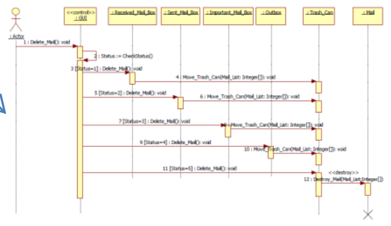
System Sequence Diagrams



System Operations



:



Sequence Diagrams

Traceability Table

Stage 1000. Plan

Stage 2030. Analyze

Stage 2040. Design