First Tutorial

Amirhossein Layegh

amlk@kth.se

2025-09-03

- **Course Assistants**
  - Amirhossein Layegh - (Amir)
  - amlk@kth.se
  - Shirin Tahmasebi
  - shirint@kth.se

HW Tutorials Overview

Introduction

- This course focuses on **object-oriented techniques and design** for software development
- The goal => solid foundation in designing software systems
- We focus:
  - Learning to model software systems using **UML**
  - Practically applying these concepts through exercises and assignments that simulate real-world software design problems

Why This Matters

- Good design saves time, improves software quality, and makes the system easier to maintain and scale.
- Even if you work with automatic code generation, having a structured design mindset will make you a skilled software engineer.
- We will learn how to turn requirements into structured, well-designed software.

Tutorial Sessions 1/5

📌 Tutorial #1 (September 3th):

- Work on the Exercise 1
  - Use Case diagram
  - Class diagram
  - Sequence diagram
  - Activity diagram

- Present Homework 1
  - A group of 3 is required for all homework and the final project
  - Book your time slot for the homework presentation
  - **Publish date** 2025-09-03
  - **Due date** 2025-09-10

Tutorial Sessions 2/5

📌 Tutorial #2 (September 10th):

- Work on the Exercise #2
  - Scenarios vs Use Cases
  - Identification of actors
  - Identification of Use Cases
  - Non-functional requirements

- Present Homework 2
  - **Publish date** 2025-09-10
  - **Due date** 2025-09-17

Tutorial Sessions 3/5

- 📌 Tutorial #3 (September 17th):
  - Work on the Exercise #3
    - Entity - Boundary - Control
    - Class diagrams - Relationships
    - CRC Cards
    - State-chart diagrams
  - Present Homework 3
    - **Publish date** 2025-09-17
    - **Due date** 2025-09-24

Tutorial Sessions 4/5

📌 Tutorial #4 (September 23rd):

- Work on the Exercise #4
  - Object Constraint Language (OCL)
- Present the Homework 4
  - **Publish date** 2025-09-23
  - **Due date** 2025-09-30

Bonus Points

- 📌 Delivering each homework and the project in due time gives 1 bonus (getting approval is required).

Bonus Points

- 📌 Delivering each homework and the project in due time gives 1 bonus (getting approval is required).
- 📌 Approval from the first attempt gives 1 bonus for each homework and the project.

Bonus Points

- 📌 Delivering each homework and the project in due time gives 1 bonus (getting approval is required).
- 📌Approval from the first attempt gives 1 bonus for each homework and the project.
- *Please note that in case of late submission of any homework, no bonus points for in-time submission*.

Bonus Points

- 📌 Delivering each homework and the project in due time gives 1 bonus (getting approval is required).
- 📌Approval from the first attempt gives 1 bonus for each homework and the project.
- *Please note that in case of late submission of any homework, no bonus points for in-time submission*.
- 📌Passing all 9quizes gives 7 points in total.

Question?

UML as a Notation

Complexity in SE

- Software systems are complex creations
- They perform many functions
- They comprise many components
- Many participants from different disciplines take part

UML as a Notation 1/10

- Why do we model the system?

UML as a Notation 2/10

- Why do we model the system?
  - Deal with complexity

- How to represent complex aspects of a system, coherently?

UML as a Notation 3/10

- Why do we model the system?
  - Deal with complexity

- How to represent complex aspects of a system, coherently?
  - Employ a standard way of representation which is notation

UML as a Notation 4/10

- Notation advantages
  - It comes with well-defined semantics.
  - It is well understood among project participants.
  - It minimizes the misunderstanding between stakeholders

UML as a Notation 8/10

- UML Vs. Notation?

UML as a Notation 8/10

- UML Vs. Notation?
  - Unified Modeling Language (UML) is a standard notation in industry.
  - Provides a spectrum of notations to represent different aspects of a system.

Overview of UML

- Five UML notations:
  - Use Case Diagrams -> Functional model describes the functionality
  - Class Diagrams -> Object model describes the structure of the system
  - Interaction Diagrams
  - State Machine Diagrams
  - Activity Diagrams

Dynamic model, describes the internal behavior of the system

Use Case Diagrams

- What is a Use case diagram?
- What is the purpose?
- Used in which development phase?

Use Case Diagrams

- Represent the functionality of the system.
- A use case describes a function provided by the system that yields a visible result for an actor.
- Used in requirement elicitation and analysis.

Use Case Diagrams

- Use Case Diagrams Elements:

Use Case Diagrams

- Use Case Diagrams Elements:
  - Actor is any entity outside, that interacts with the system.

Use Case Diagrams

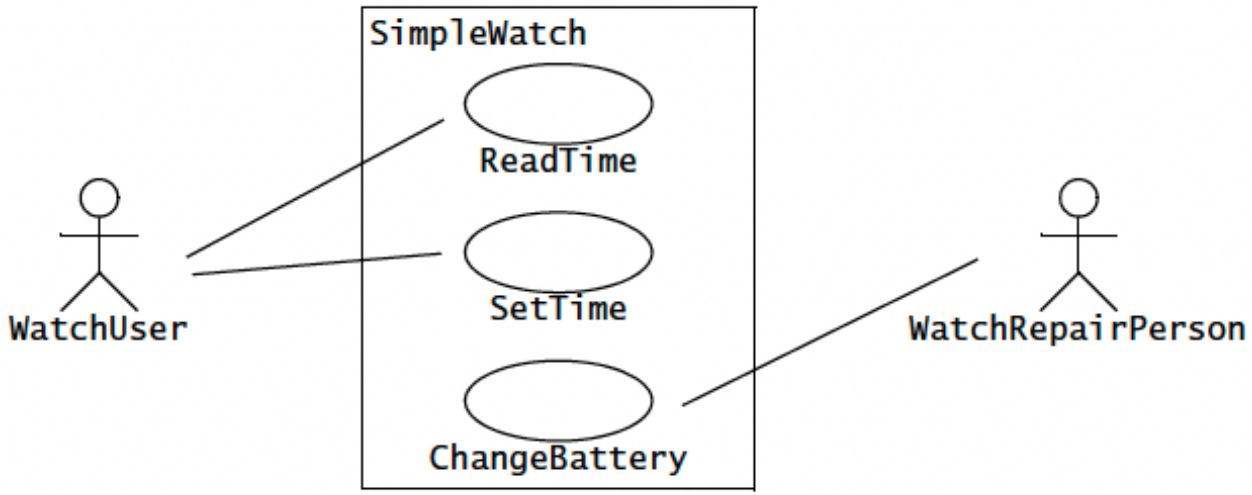
- Use Case Diagrams Elements:
  - Actor is any entity outside, that interacts with the system.
  - Functions, that yield a visible result for an actor.

Use Case Diagrams

- Use Case Diagrams Elements:
  - Actor is any entity outside, that interacts with the system.
  - Functions, that yield a visible result for an actor.
  - Boundary, that differentiates between the system and its environment.

Use Case Diagrams

- Use Case Diagrams Elements:
  - Actor is any entity outside, that interacts with the system.
  - Functions, that yield a visible result for an actor.
  - Boundary, that differentiates between the system and its environment.



Problem Description Q1

In an IT company, software developers usually must manage their vacations with their direct manager. There might be possible schedule conflicts for important project deliveries and priorities among the colleagues. The company would like to automate the communication between the developers and their managers through an internal system that can be used to manage the communication workflow. The system should be able to retrieve schedules of all projects deadlines and order the vacation requests according to date of request and priorities. Figure 1 illustrates the usual communication process.

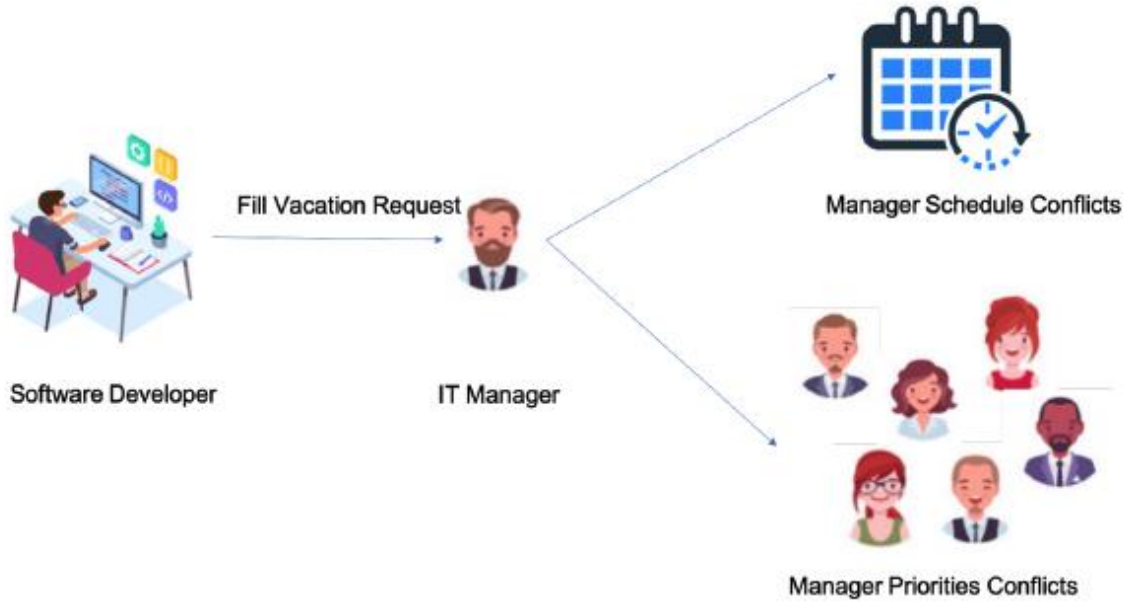


Figure 1: Manage vacation requests

Problem Description Q1

Write a use case description and draw a use case diagram for the “Vacation Request” case, taking into consideration the following details:

- Both software developer and the manager should be registered in the new system and should be able to login and access the system. Ignore the registration use case for this example.
- The application can be approved or rejected. In some cases, it can be suspended for further negotiation.

1. Identification of roles

Role	Responsibilities
Developer	a. Filling vacation request with required details
Fill ?	Fill ? .....

1. Identification of roles

Role	Responsibilities
Developer	a. Filling vacation request with required details
Manager	a. Reviewing, approving or rejecting applications b. Negotiating the application when possible

2. Identification of Use Cases

a.	Login
b.	ValidateUser
c.	CreateVacationRequest
d.	….
e.	….
f.	….
g.	….

2. Identification of Use Cases

a.	Login
b.	ValidateUser
c.	CreateVacationRequest
d.	ReviewApplication
e.	….
f.	….
g.	….

2. Identification of Use Cases

a.	Login
b.	ValidateUser
c.	CreateVacationRequest
d.	ReviewApplication
e.	CheckSchedule
f.	….
g.	….

2. Identification of Use Cases

a.	Login
b.	ValidateUser
c.	CreateVacationRequest
d.	ReviewApplication
e.	CheckSchedule
f.	CheckPriority
g.	….

2. Identification of Use Cases

a.	Login
b.	ValidateUser
c.	CreateVacationRequest
d.	ReviewApplication
e.	CheckSchedule
f.	CheckPriority
g.	FinalizeApplication

3. Identification of Exceptional Use Cases

Many exceptional situations happen in a certain condition not expected by the user

b.	**InvalidCredential**
c.	ScheduleConflict
d.	.....
e.	.....

3. Identification of Exceptional Use Cases

Many exceptional situations happen in a certain condition not expected by the user

b.	**InvalidCredential**
c.	ScheduleConflict
d.	PriorityConflict
e.	SuspendForNegotiation

4. Write Use Cases Description

Name: ManageVacationRequest
Participating actor(s): Fill? … Fill? …
Entry Condition: 1. Developer logins to the system. 2. System validates the developer credentials. 3. Developer accesses the create vacation request functionality.
Exit Condition: Developer and manager have agreed whether to approve, cancel or negotiate the vacation request.
Quality Condition: The system should be available and functioning without unexpected interruptions.
Event Flow: (Next Slide)

4. Write Use Cases Description

Name: ManageVacationRequest
Participating actor(s): a. Developer b. Manager
Entry Condition: 1. Developer logins to the system. 2. System validates the developer credentials. 3. Developer accesses the create vacation request functionality.
Exit Condition: Developer and manager have agreed whether to approve, cancel or negotiate the vacation request.
Quality Condition: The system should be available and functioning without unexpected interruptions.
Event Flow: (Next Slide)

4. Write Use Cases Description

Event Flow:

(Hint) How does the use case start?

1. Developer chooses to create a new vacation request application.
2. System displays the requested form.

(Hint) What does the developer have to fill in the application?

3. ....
4. ...

(Hint) How does the manager get informed about the application?

5. System displays the new request to the manager.
6. Manager reviews the application and chooses to review the schedule.
7. System displays the schedule to the manager.
8. Manager chooses to review the vacation priorities for the other developers.

(Hint) what is the action that should be taken by the manager?

9. ...
10. ...

(Hint) How does the developer know the final decision ?

11. System displays the result of the request to the initiator developer.

4. Write Use Cases Description

Event Flow:

(Hint) How does the use case start?

1. Developer chooses to create a new vacation request application.
2. System displays the requested form.

(Hint) What does the developer have to fill in the application?

3. Developer fills the application with the required date, and reason for vacation.
4. Developer chooses to send the application to the manager.

(Hint) How does the manager get informed about the application?

5. System displays the new request to the manager.
6. Manager reviews the application and chooses to review the schedule.
7. System displays the schedule to the manager.
8. Manager chooses to review the vacation priorities for the other developers.

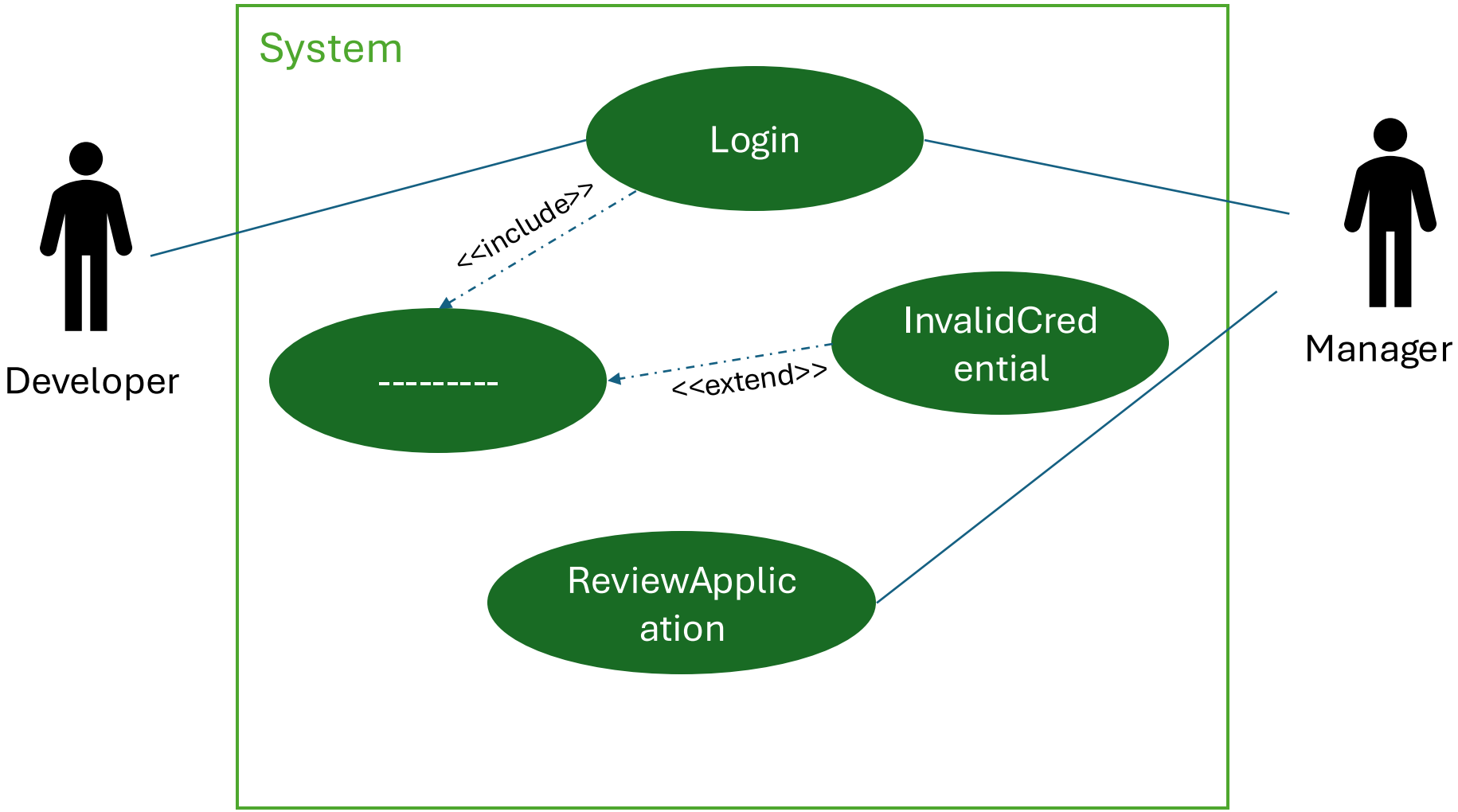
(Hint) what is the action that should be taken by the manager?

9. System displays the list of vacation requests to the manager.
10. Manager reviews the application and chooses to approve, reject or suspend the application.

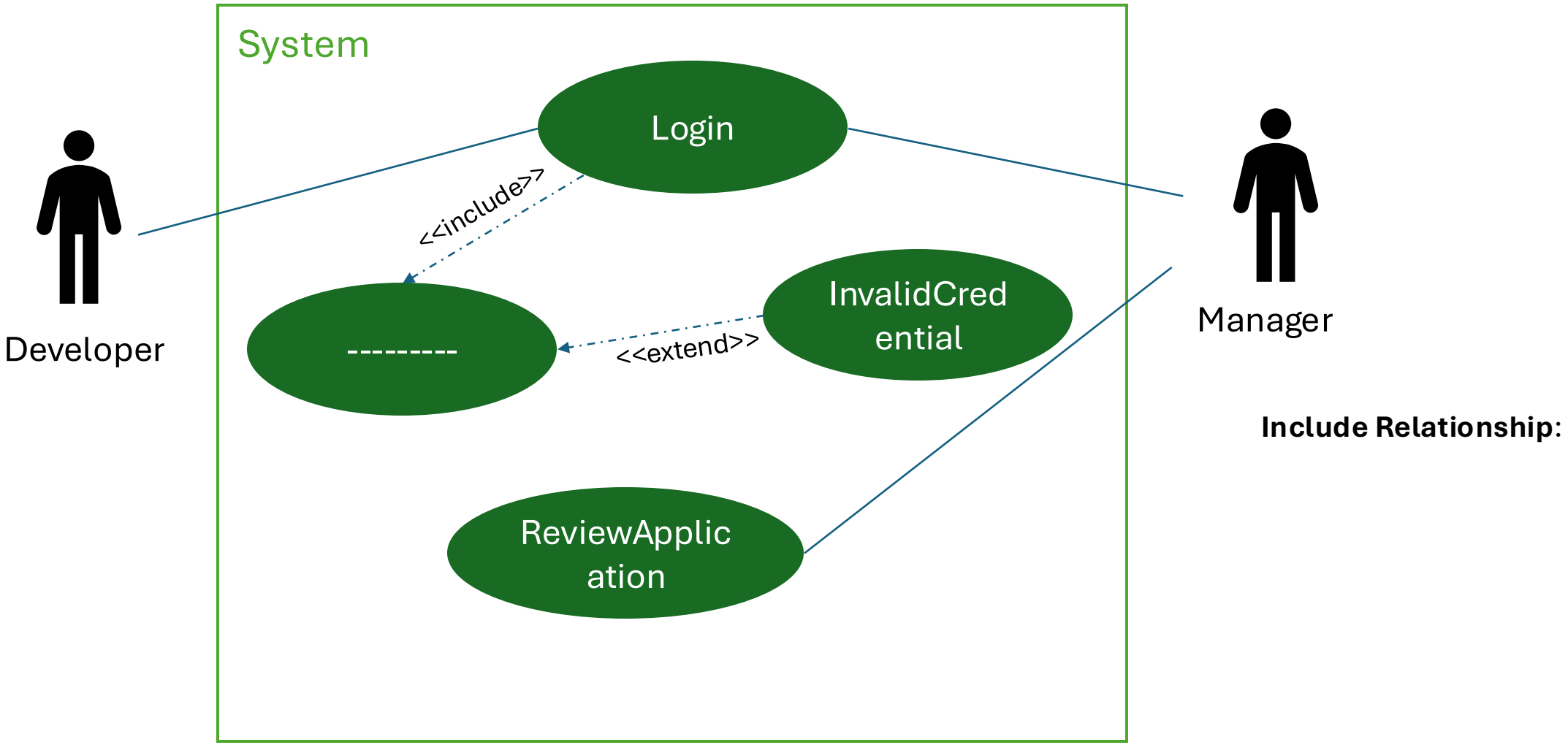
(Hint) How does the developer know the final decision ?

11. System displays the result of the request to the initiator developer.

4. Use Cases Diagram

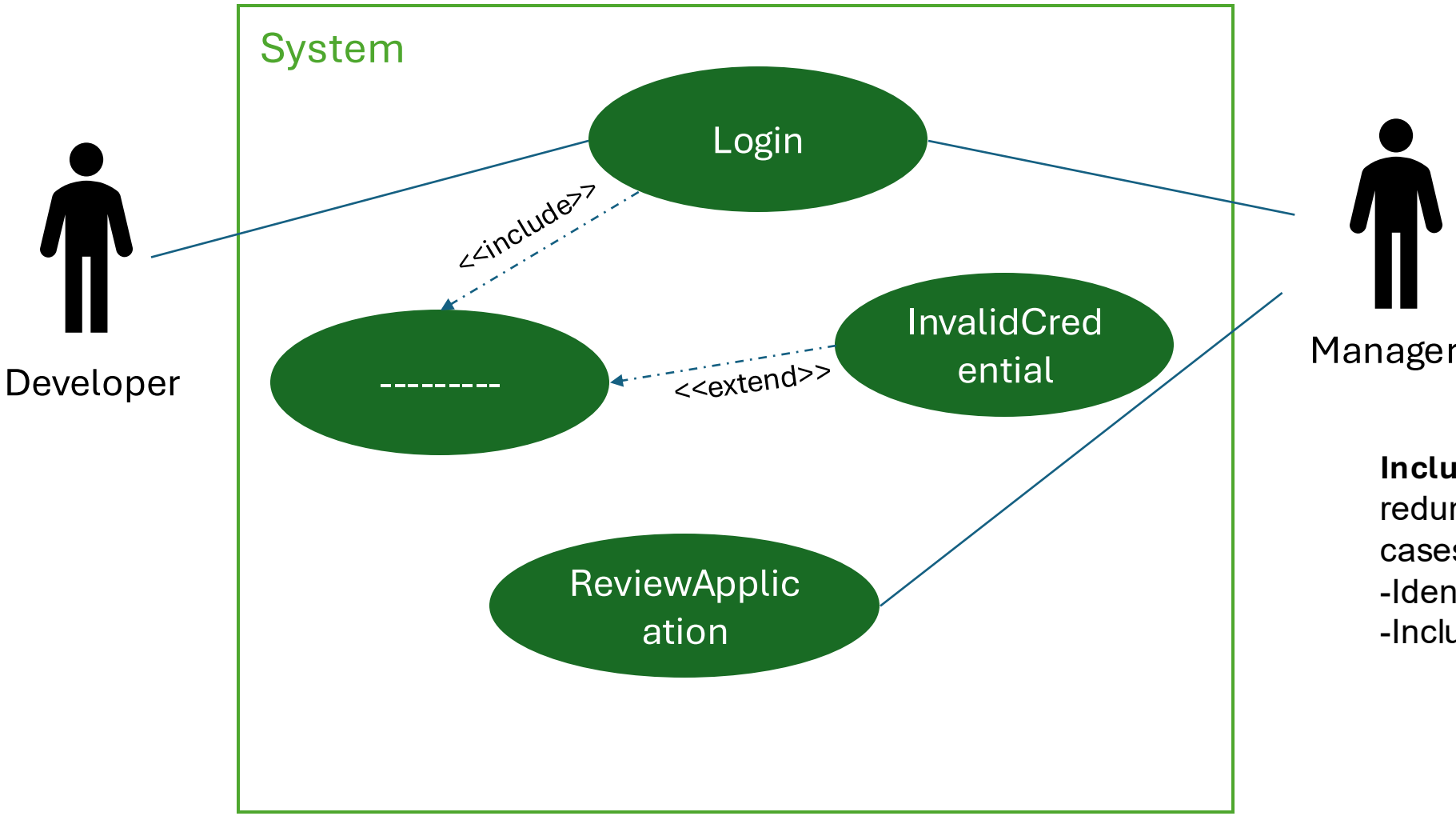


4. Use Cases Diagram



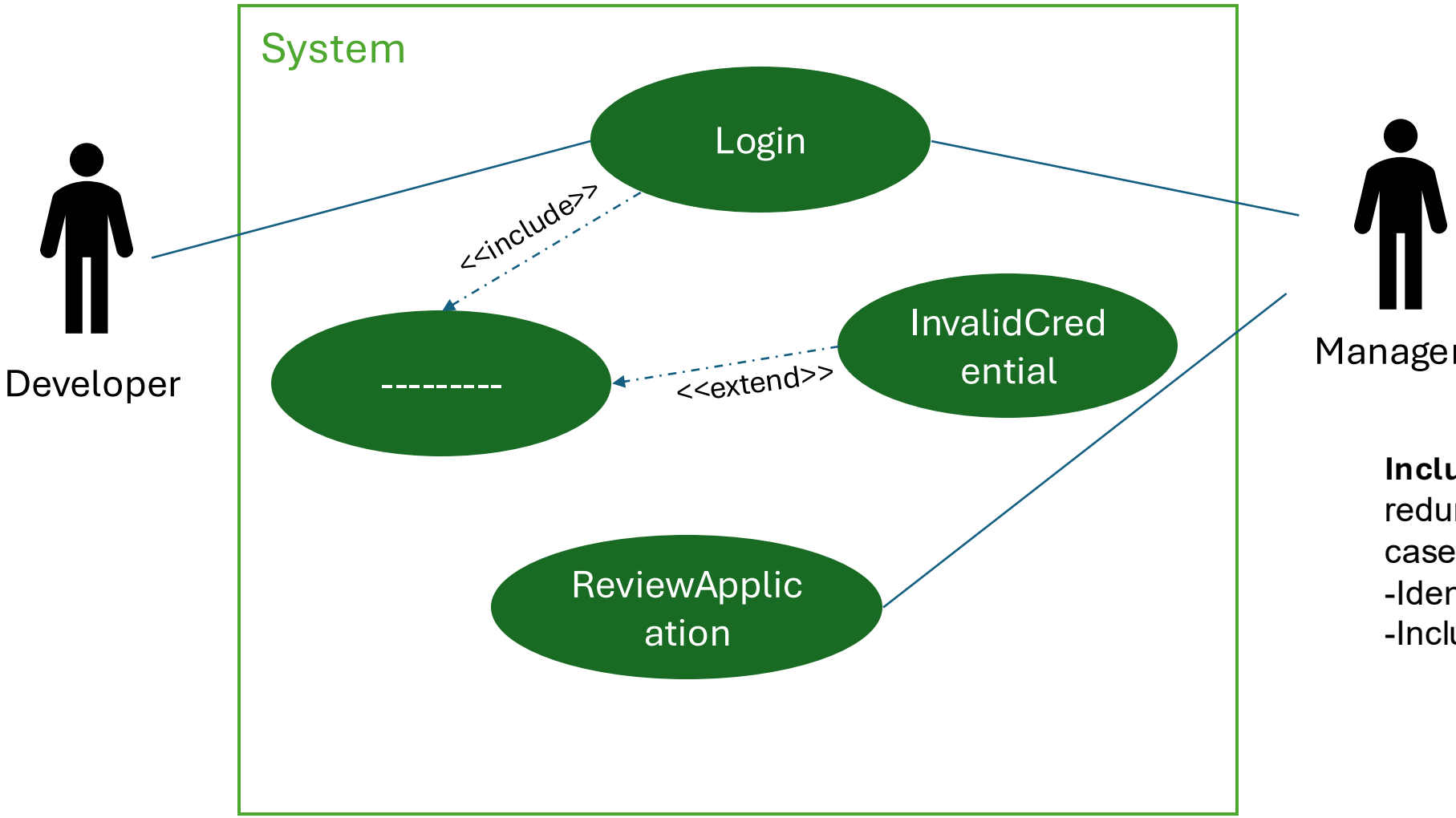
Include Relationship:

4. Use Cases Diagram



Include Relationship: We may have redundancy and some common use-cases inside another use cases.
-Identifying commonalities
-Included use case can be re-used.

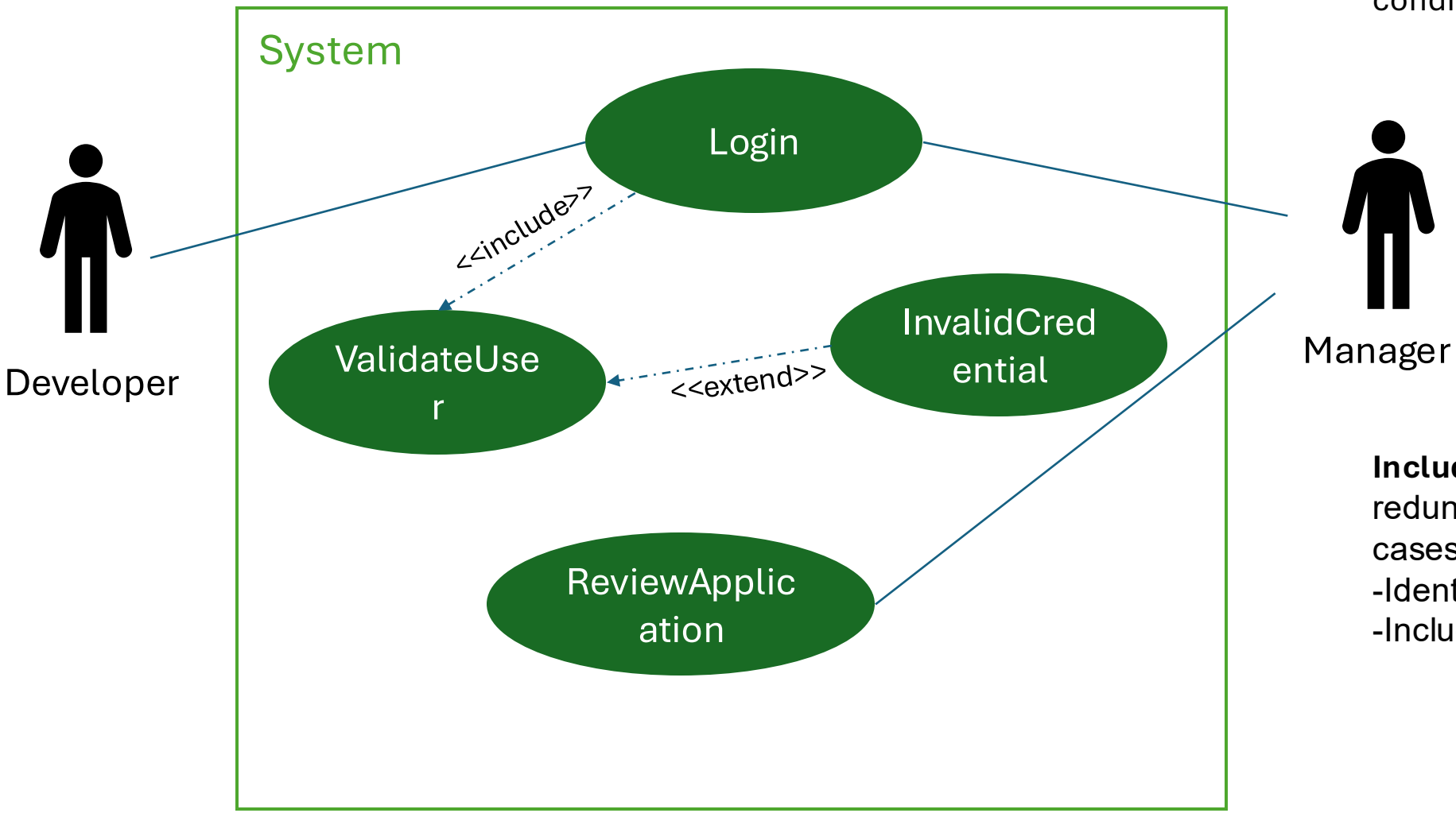
4. Use Cases Diagram



Extend Relationship:

Include Relationship: We may have redundancy and some common use-cases inside another use cases.
-Identifying commonalities
-Included use case can be re-used.

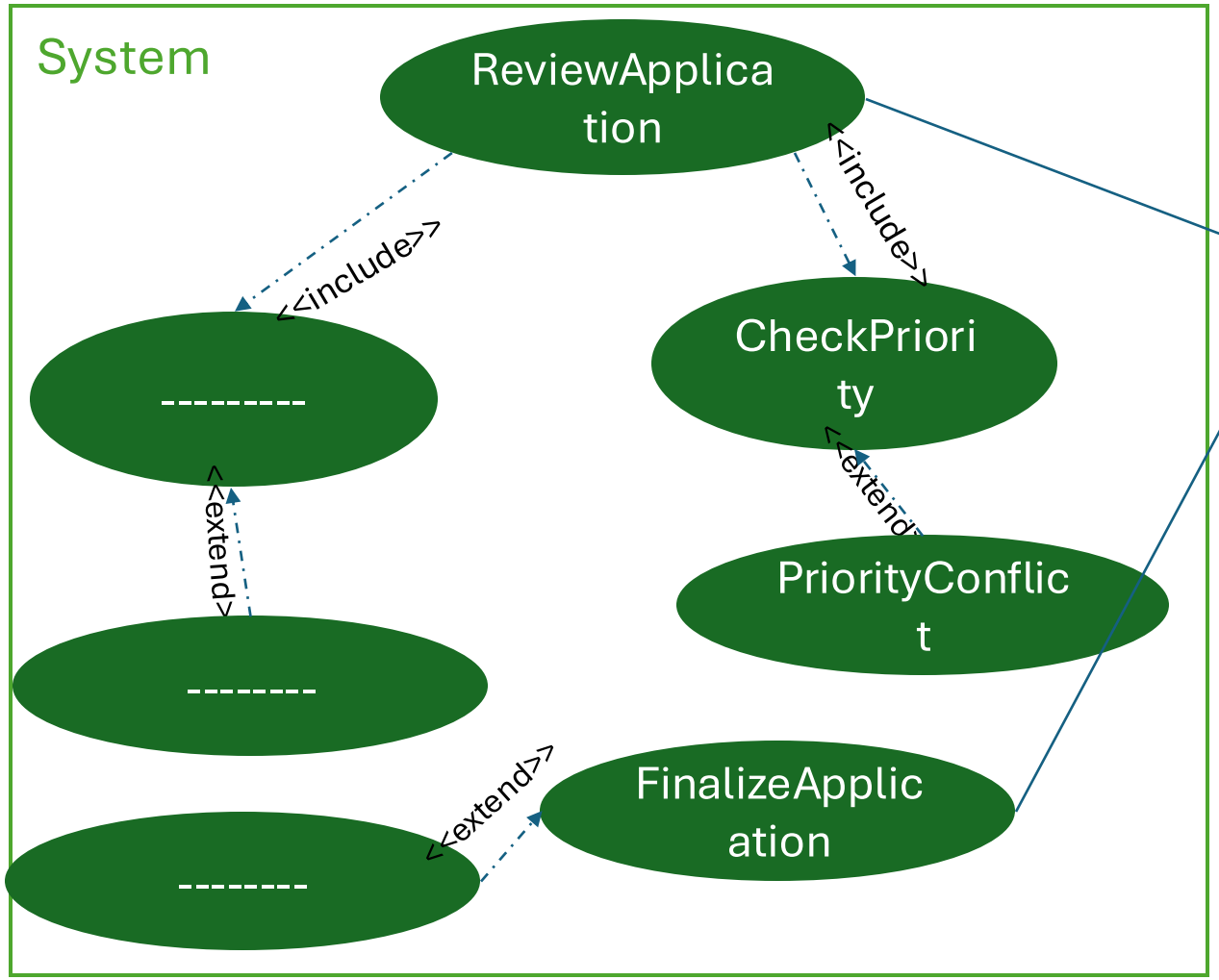
4. Use Cases Diagram



Extend Relationship: Shows the exception of use cases. Instance of a use case can happen under the certain condition.

Include Relationship: We may have redundancy and some common use-cases inside another use cases.
-Identifying commonalities
-Included use case can be re-used.

4. Use Cases Diagram



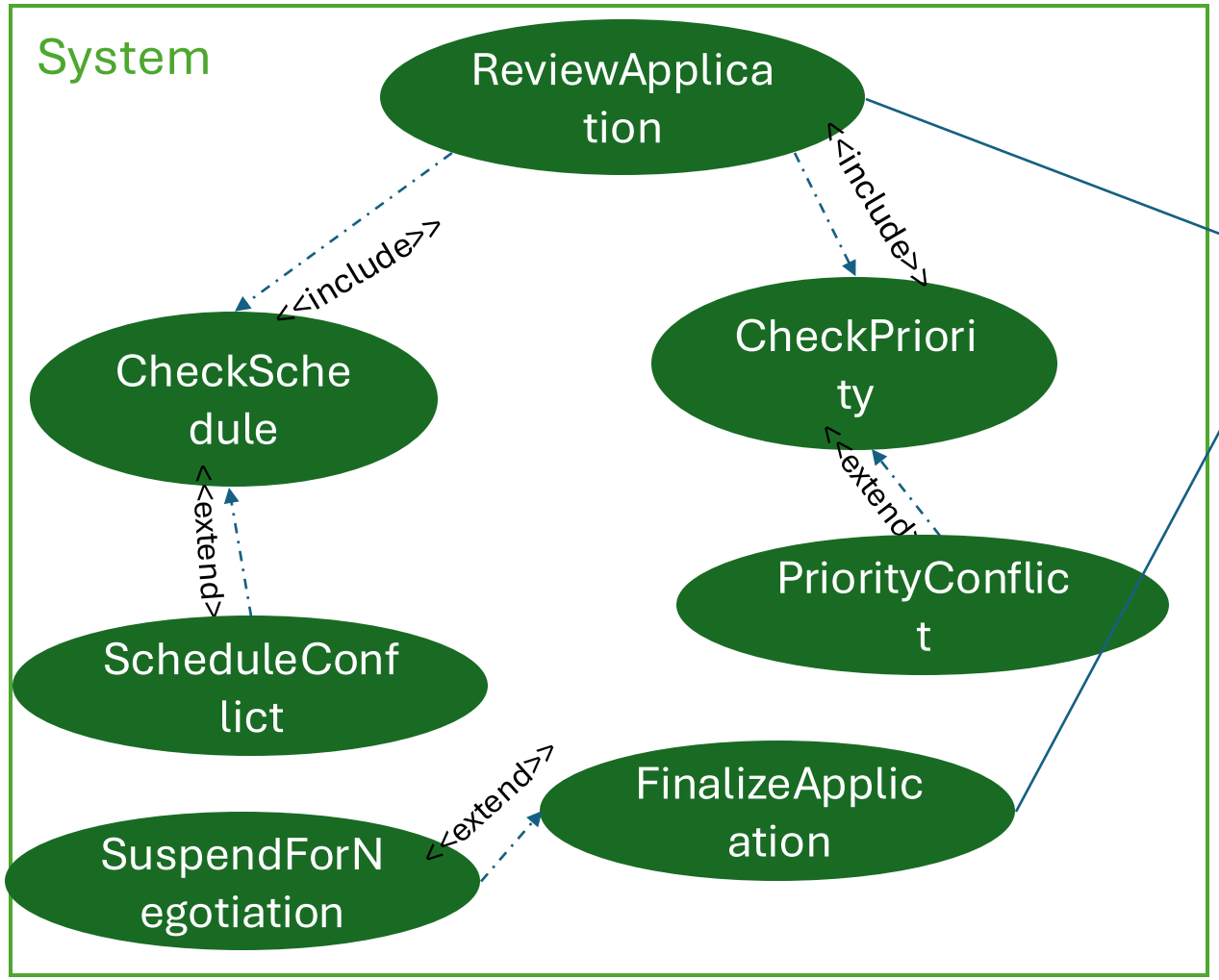
Developer

Manager

Extend Relationship: Shows the exception of use cases. Instance of a use case can happen under the certain condition.

Include Relationship: We may have redundancy and some common use-cases inside another use cases.
-Identifying commonalities
-Included use case can be re-used.

4. Use Cases Diagram



Developer

Manager

Extend Relationship: Shows the exception of use cases. Instance of a use case can happen under the certain condition.

Include Relationship: We may have redundancy and some common use-cases inside another use cases.
-Identifying commonalities
-Included use case can be re-used.

Class Diagrams

Class Diagrams

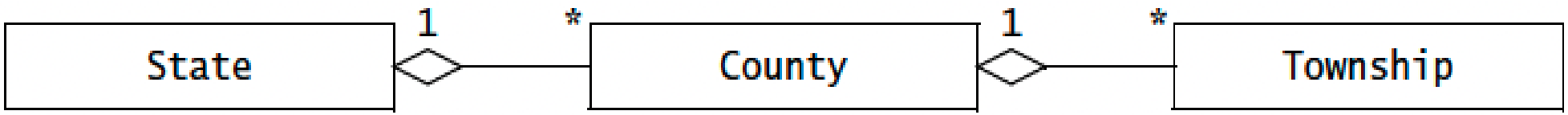
- Describes the Structure of the system, in terms of Classes and Objects that make up the system.

Class Diagrams

- Describes the Structure of the system, in terms of Classes and Objects that make up the system.
- Classes:
  - Abstractions that specify the attributes and behavior of a set of objects.
  - Is a collection of objects that share a set of attributes.
- Objects:
  - An instance of a Class.
  - Are entities that encapsulate state and behavior.

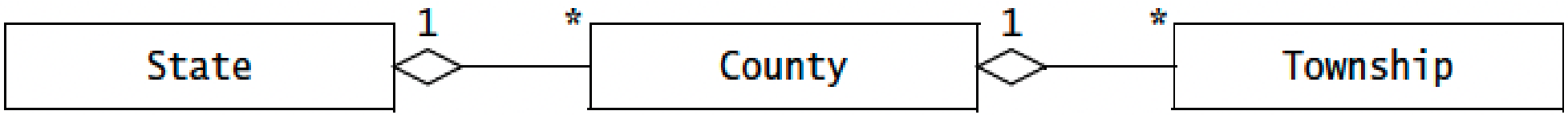
Class Diagram Associations

- **Associations** are relationships between classes and represent connections.
  - **One-to-one Association**
  - **One-to-many Association**
  - **Many-to-many Association**
  - **Aggregation ->** shows the hierarchy of classes



Class Diagram Associations

- **Associations** are relationships between classes and represent connections.
  - **One-to-one Association**
  - **One-to-many Association**
  - **Many-to-many Association**
  - **Aggregation ->** shows the hierarchy of classes
  - **Inheritance ->** denotes ”a type of” relationship



Class Diagram Associations

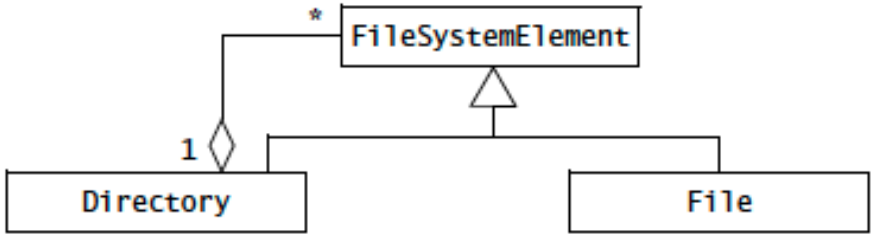


Figure 2-28 Example of a hierarchical file system. A `Directory` can contain any number of *`FileSystemElements`* (a *`FileSystemElement`* is either a `File` or a `Directory`). A given *`FileSystemElement`*, however, is part of exactly one `Directory`.

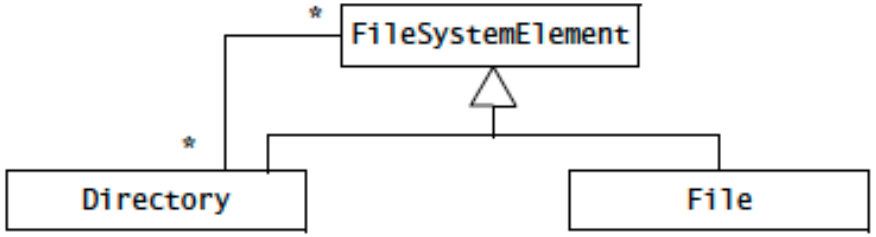


Figure 2-29 Example of a nonhierarchical file system. A `Directory` can contain any number of *`FileSystemElements`* (a *`FileSystemElement`* is either a `File` or a `Directory`). A given *`FileSystemElement`* can be part of many `Directories`.

Problem Description Q2

The developed system has a request which might be a vacation or travel request. Each request has a serial number, name of requester, name of the person to whom it is submitted, request type (vacation or travel request), and date of submission. Vacation request has other specific fields such as: vacation reason, priority, duration of vacation (From - To). Travel request has destination, travel reason, (From -To), and information whether it is paid by the inviting organization or not.
Each request has a status (initiated, approved, rejected, under negotiation).

Sequence Diagrams

Sequence Diagrams

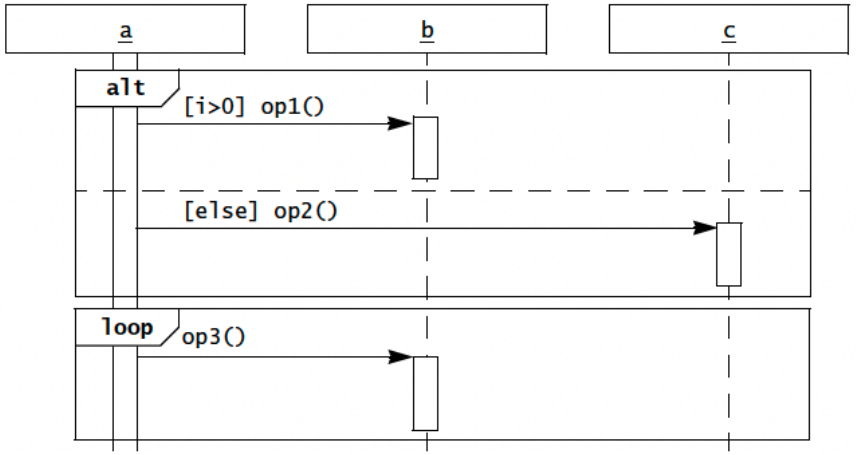
- Describes the dynamic behavior of system by representing:

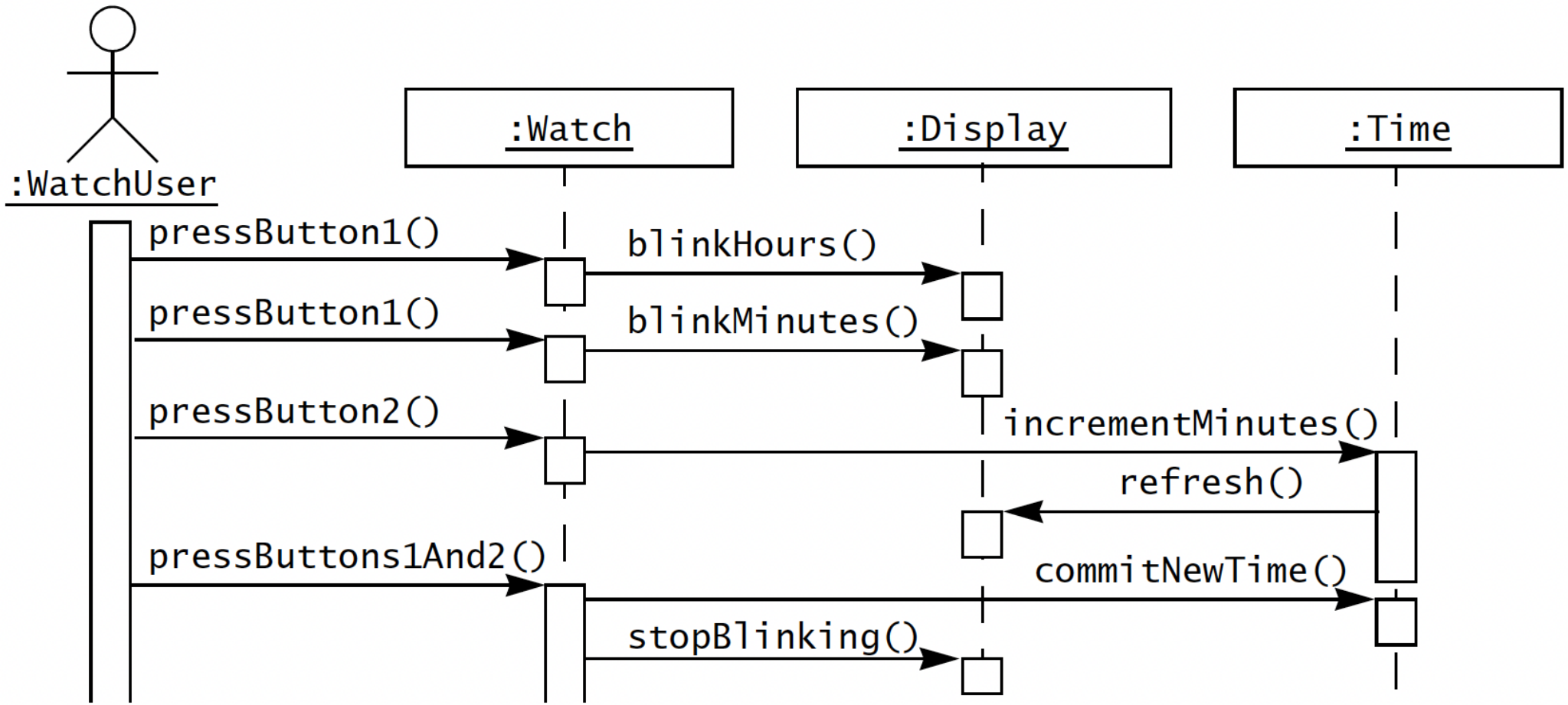
Sequence Diagrams

- Describes the dynamic behavior of system by representing:
  - Participating objects, objects involved in a use case and their interaction -> horizontally
  - Time -> vertically

Sequence Diagrams

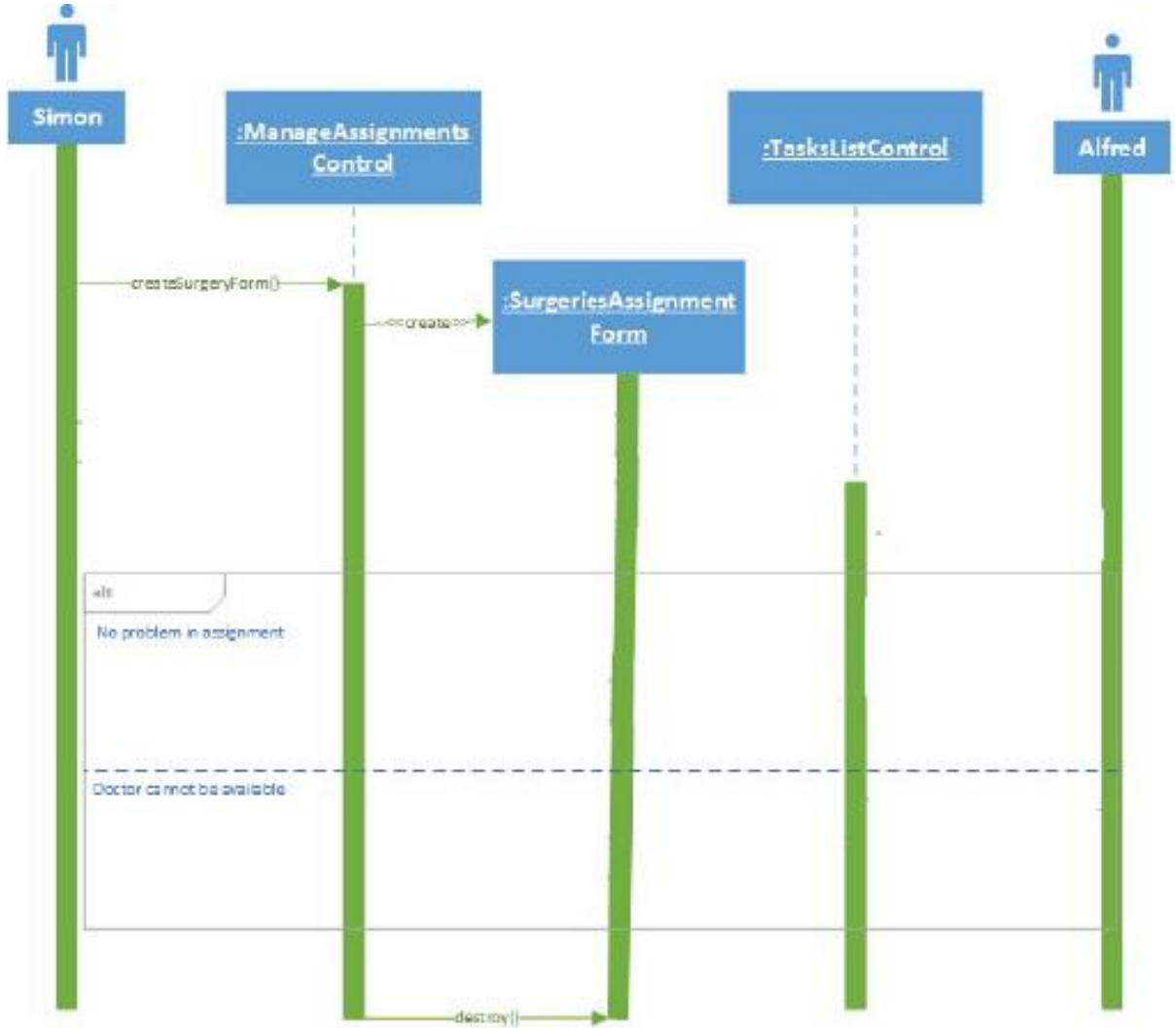
- Sequence diagrams can be useful for identifying additional objects that participate in the use cases
  - Notation for iteration -> combined fragment labeled with the *Loop operator*.
  - Notation for conditional -> combined fragment containing a partition for each *alternative*.

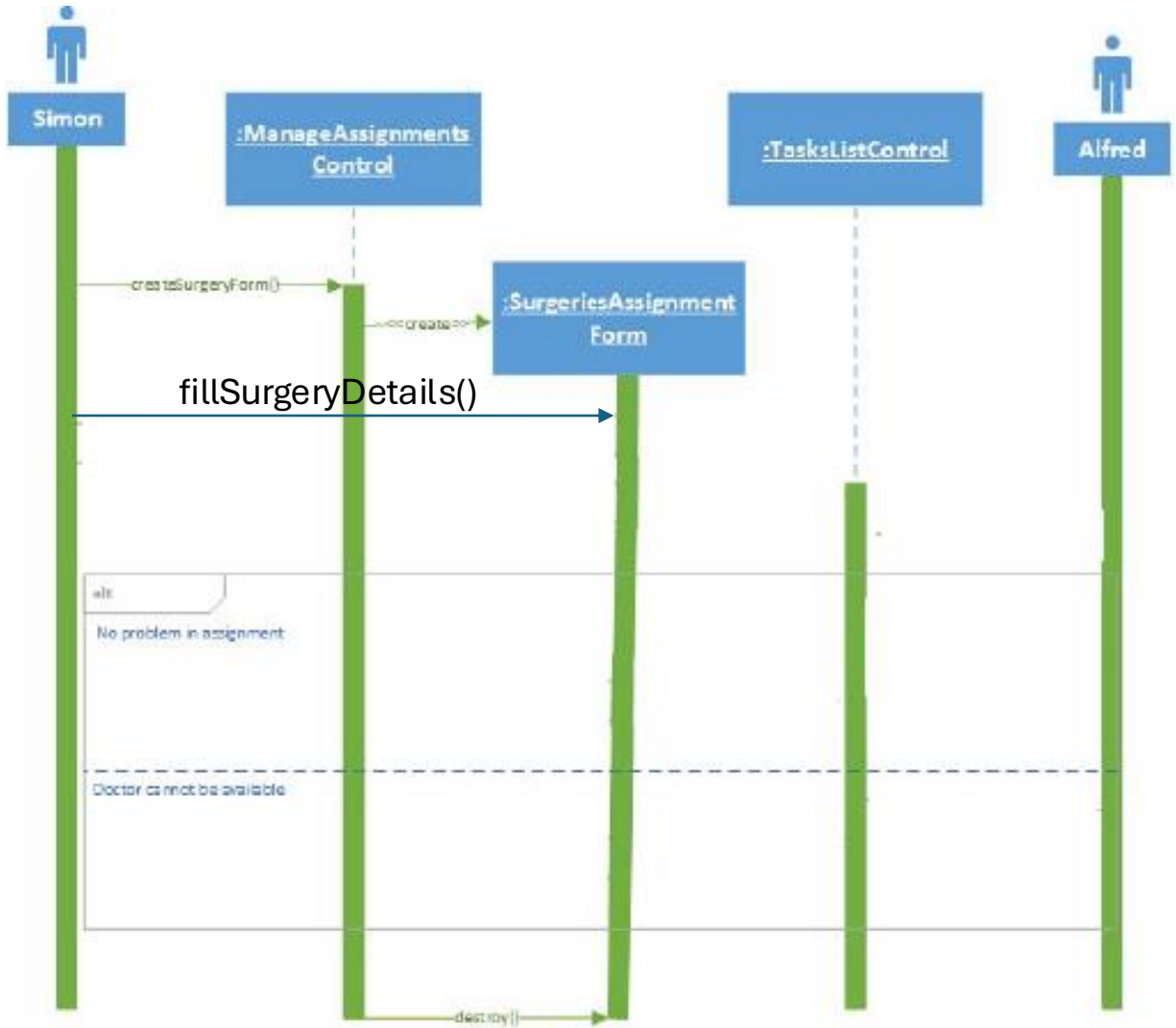


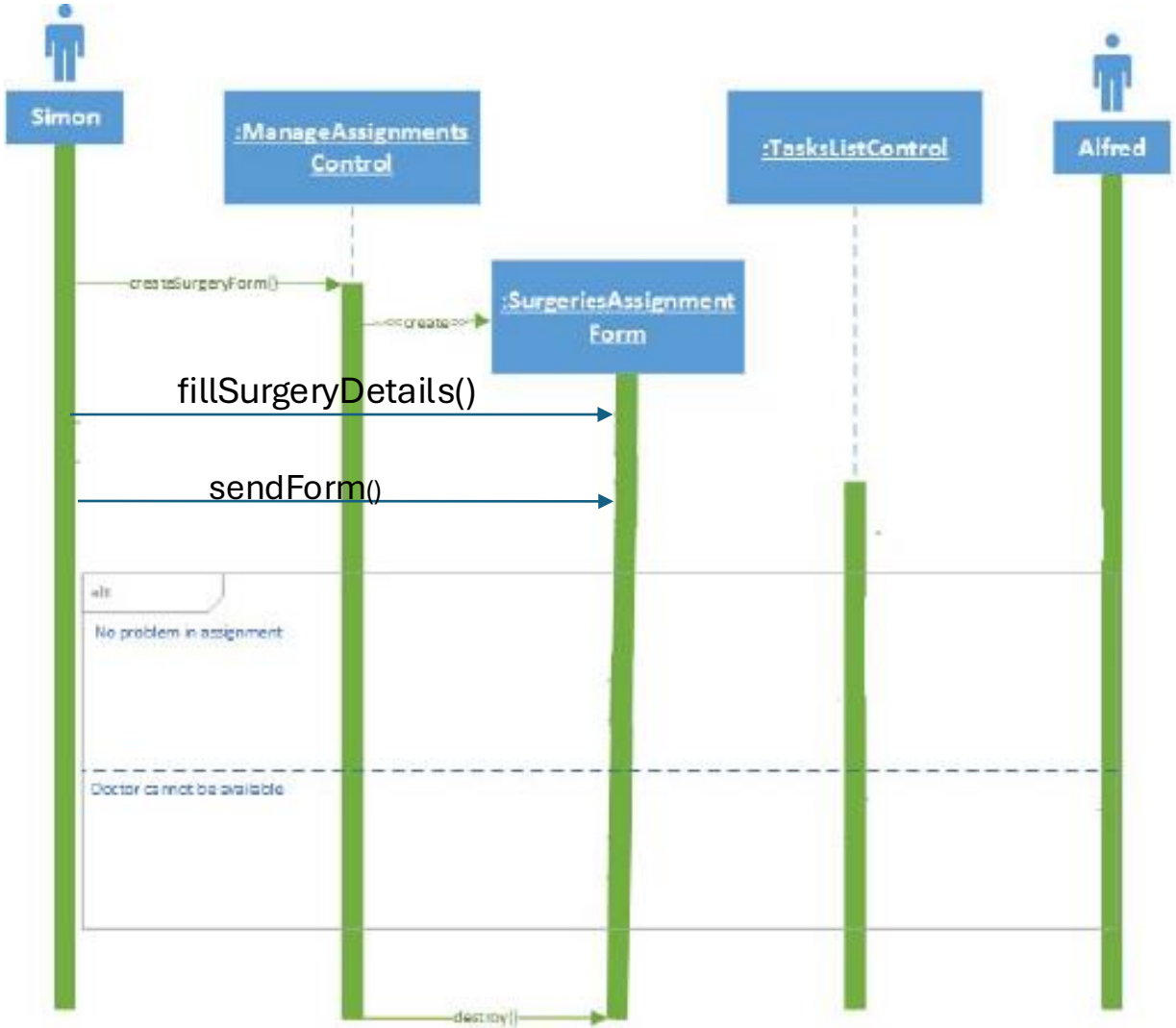

:WatchUser
:Watch
:Display
:Time
pressButton1()
blinkHours()
pressButton1()
blinkMinutes()
pressButton2()
incrementMinutes()
refresh()
pressButtons1And2()
commitNewTime()
stopBlinking()

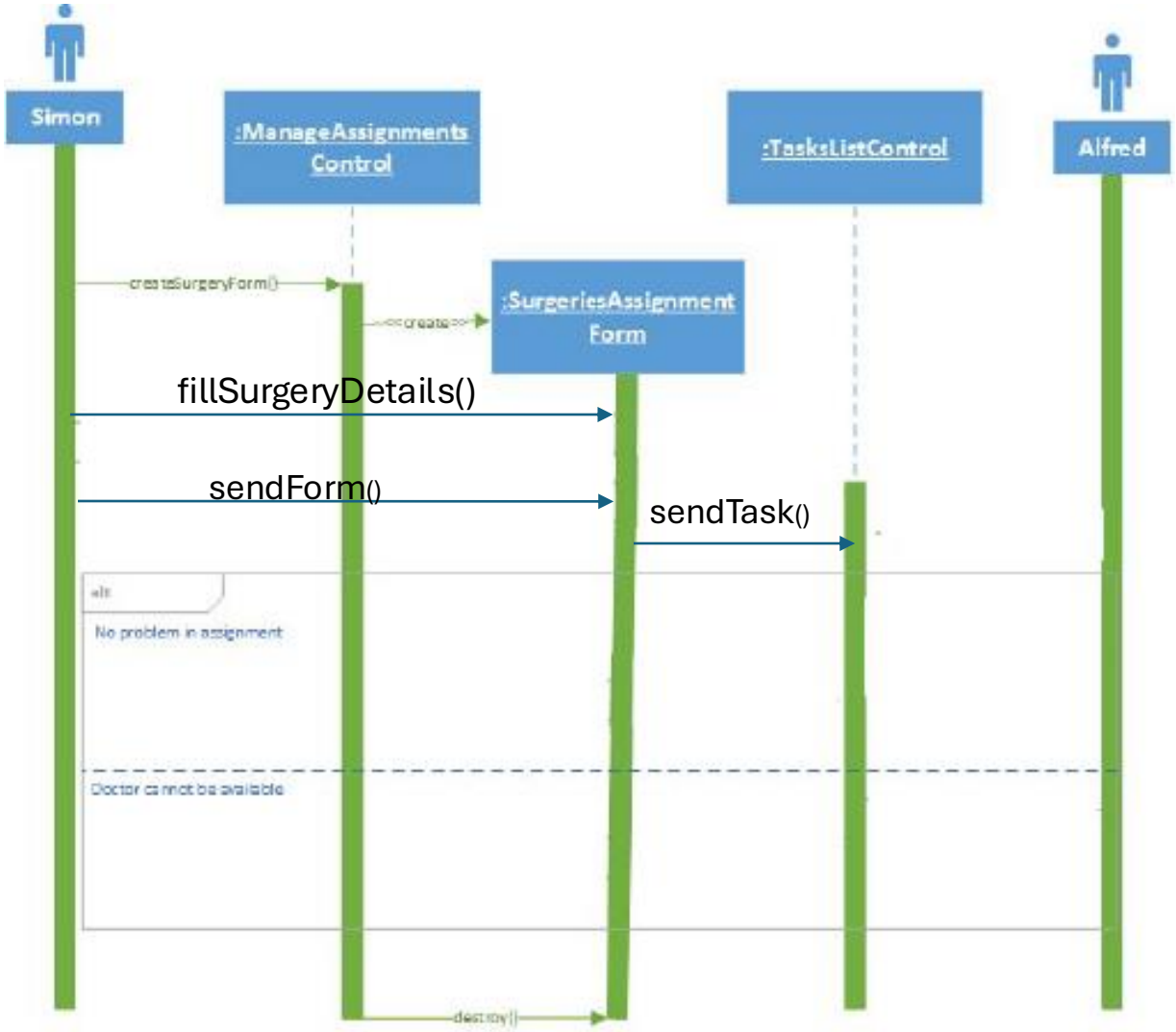
Problem defenition Q3

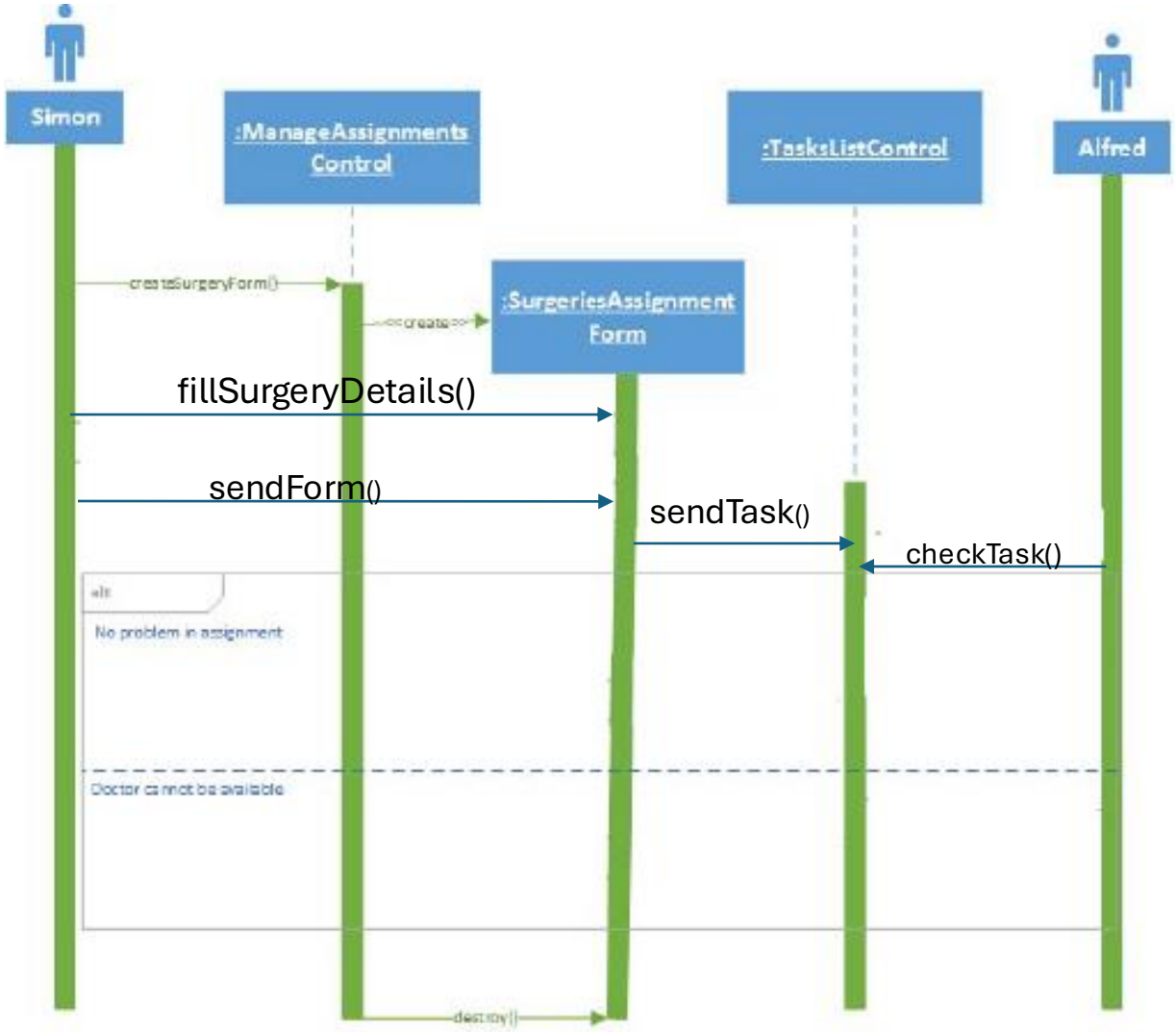
scenario name	AssignDoctorToASurgery
Participating actor instances	Alfred: Doctor Simon: HeadOfUnit
Flow of events	1. Simon logins to the system and chooses to create a new surgery assignment request. 2. Simon selects "Alfred" to be assigned to the surgery, and he fills the other required details in the assignment form and chooses to send the application. 3. System displays the new task in Alfred tasks list. 4. Alfred checks if he can be available for the surgery and approves the application if it is okay. Otherwise, Alfred rejects the assignment with description of the reasons. 5. System updates the status of the form to be approved or rejected and sends the update to Simon. 6. Simon opens the application and checks Alfred feedback. 7. Simon closes the application.

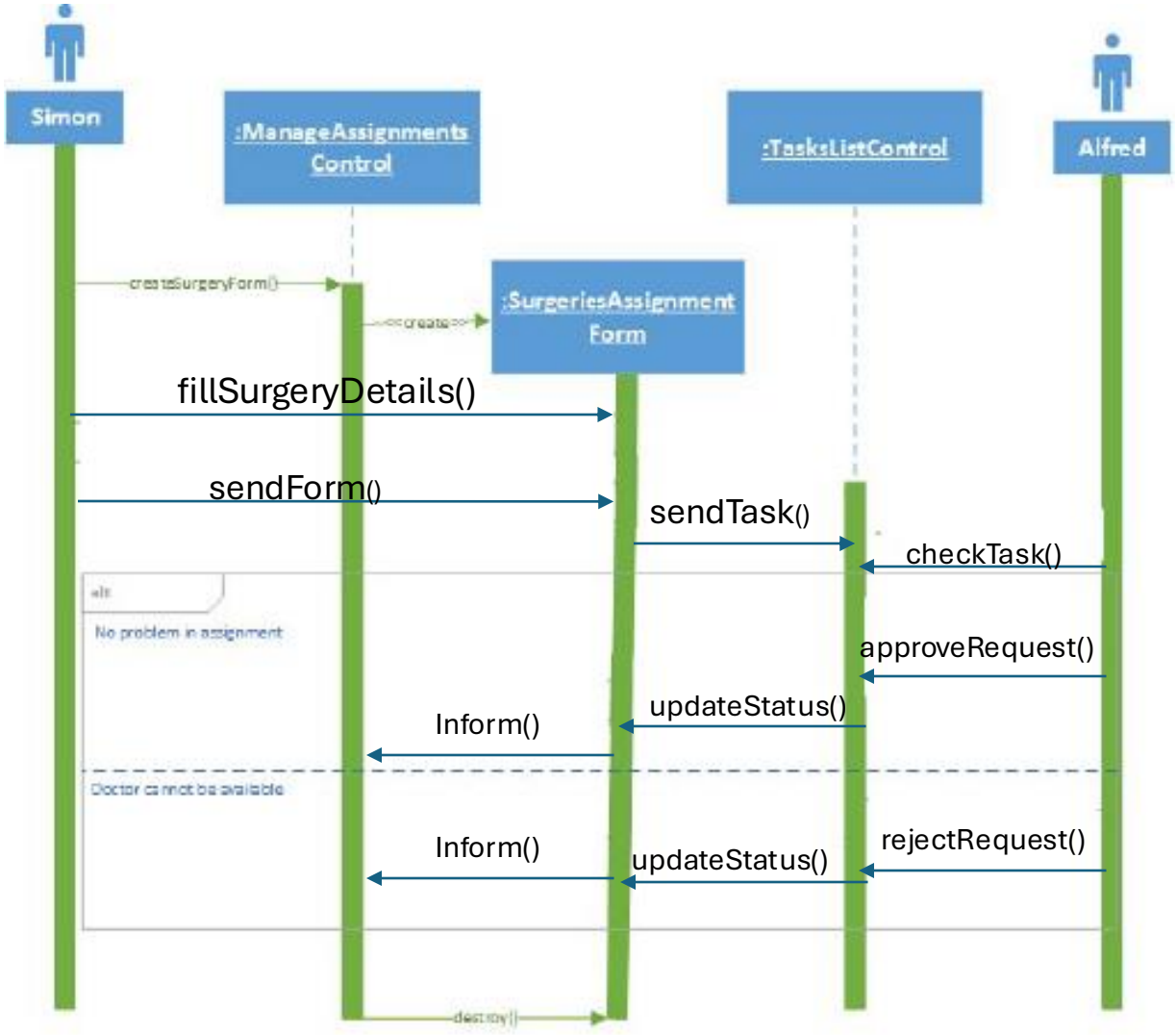

Simon
:ManageAssignments Control
:SurgeriesAssignment Form
:TasksListControl
Alfred
createSurgeryForm()
<<create>>
alt
No problem in assignment
Doctor cannot be available
destroy()

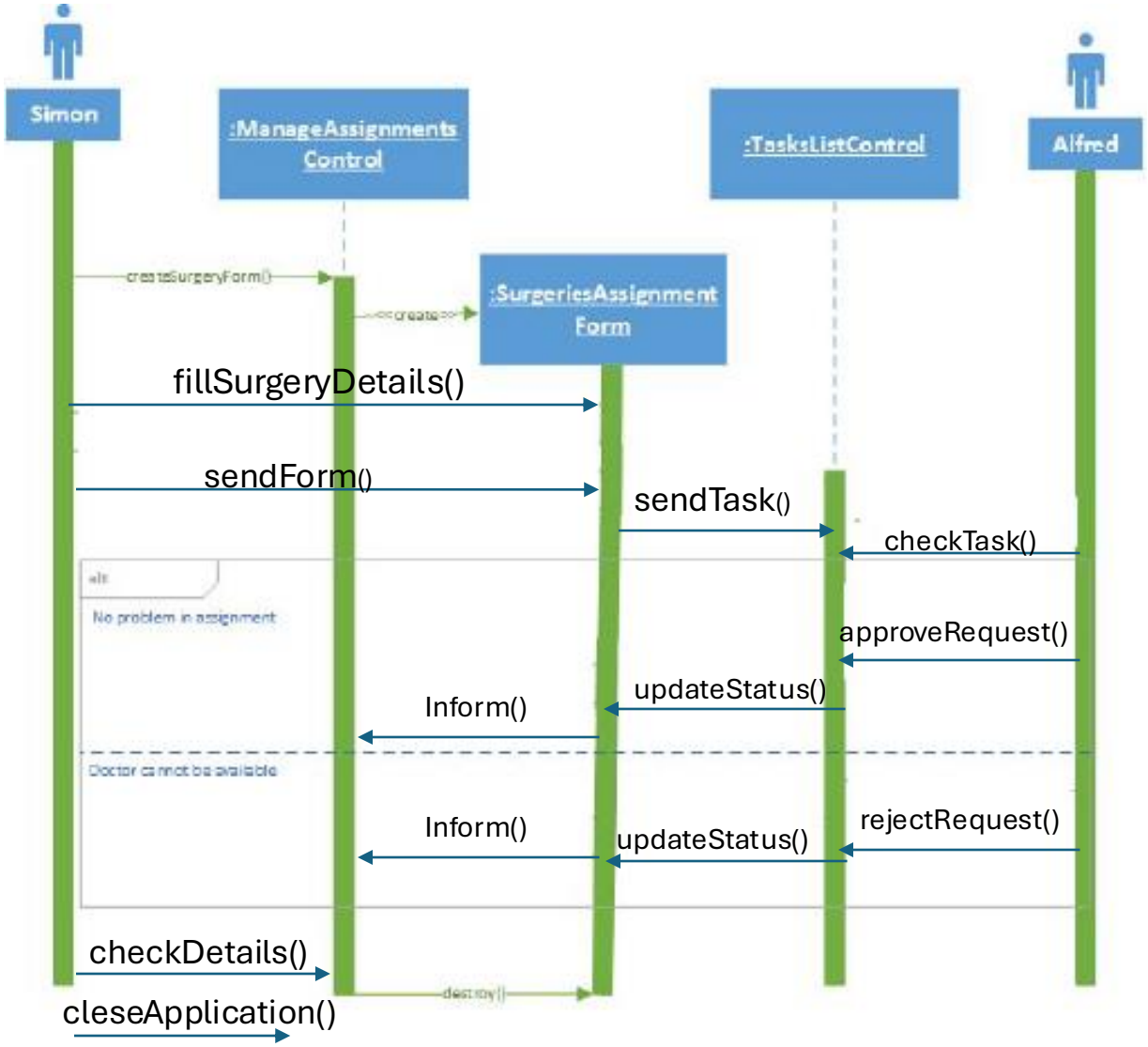

Simon
:ManageAssignments Control
:TasksListControl
Alfred
createSurgeryForm()
<<create>>
:SurgeriesAssignment Form
fillSurgeryDetails()
alt
No problem in assignment
Doctor cannot be available
destroy()


Simon
:ManageAssignments Control
:TasksListControl
Alfred
createSurgeryForm()
<<create>>
:SurgeriesAssignment Form
fillSurgeryDetails()
sendForm()
alt
No problem in assignment
Doctor cannot be available
destroy()


Simon
:ManageAssignments Control
:SurgeriesAssignment Form
:TasksListControl
Alfred
createSurgeryForm()
<<create>>
fillSurgeryDetails()
sendForm()
sendTask()
alt
No problem in assignment
Doctor cannot be available
destroy()


Simon
:ManageAssignments Control
:SurgeriesAssignment Form
:TasksListControl
Alfred
createSurgeryForm()
<<create>>
fillSurgeryDetails()
sendForm()
sendTask()
checkTask()
alt
No problem in assignment
Doctor cannot be available
destroy()


Simon
:ManageAssignments Control
:TasksListControl
Alfred
createSurgeryForm()
<<create>>
:SurgeriesAssignment Form
fillSurgeryDetails()
sendForm()
sendTask()
checkTask()
alt
No problem in assignment
approveRequest()
updateStatus()
Inform()
Doctor cannot be available
Inform()
updateStatus()
rejectRequest()
destroy()


Simon
:ManageAssignments Control
:SurgeriesAssignment Form
:TasksListControl
Alfred
createSurgeryForm()
<<create>>
fillSurgeryDetails()
sendForm()
sendTask()
checkTask()
alt
No problem in assignment
approveRequest()
updateStatus()
Inform()
Doctor cannot be available
Inform()
updateStatus()
rejectRequest()
checkDetails()
destroy()
cleseApplication()

Activity Diagrams

Activity Diagrams

- An activity diagram denotes how a behavior is realized in terms of one or several sequences of activities and the object flows needed for coordinating the activities.
- An activity is made up of either an action or a graph of subactivities and their associated object flow.
- Means of describing workflows.
- Describe business flow and high-level business requirements.

Activity Diagrams

- Activity diagram elements:
  - **Control nodes**: coordinate control flows in an activity diagram, providing mechanisms for representing decisions, concurrency, and synchronization (order of operations).
  - **Decisions**: are branches in the control flow (depicted by diamonds).
  - **Fork and Join**: represent concurrency and synchronization.

Activity Diagrams

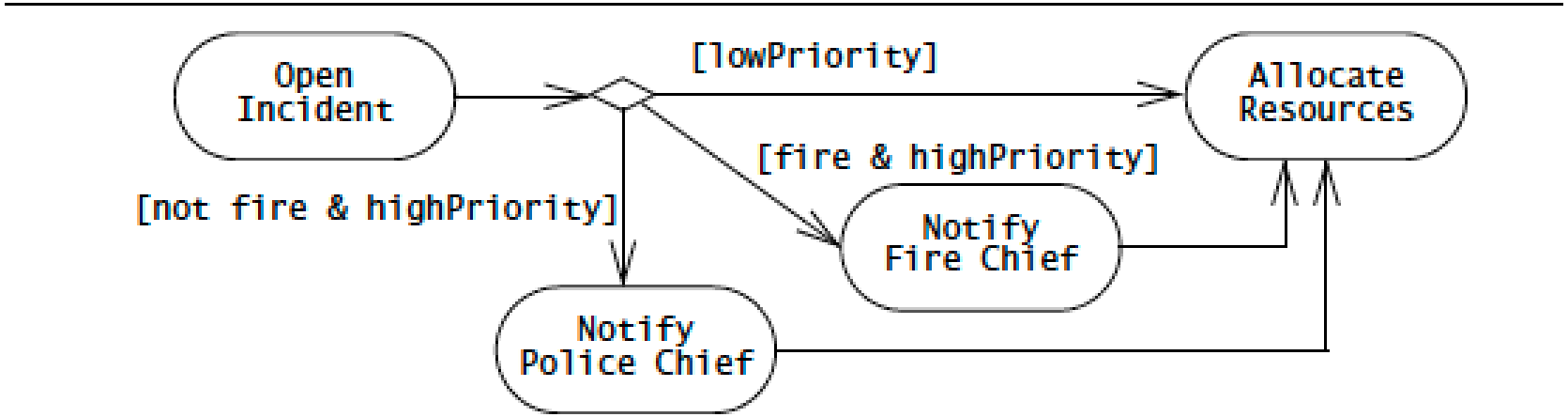


Figure 2-42 Example of decision in the `OpenIncident` process. If the `Incident` is a fire and is of high priority, the `Dispatcher` notifies the `FireChief`. If it is a high-priority `Incident` that is not a fire, the `PoliceChief` is notified. In all cases, the `Dispatcher` allocates resources to deal with the `Incident`.

Activity Diagrams

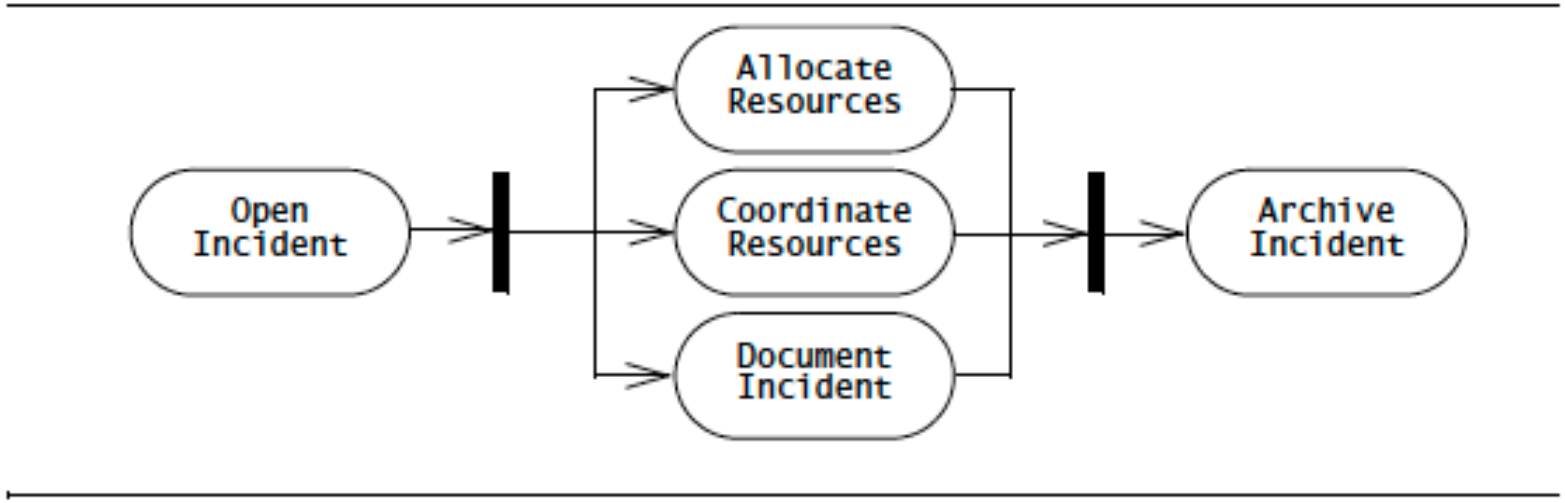
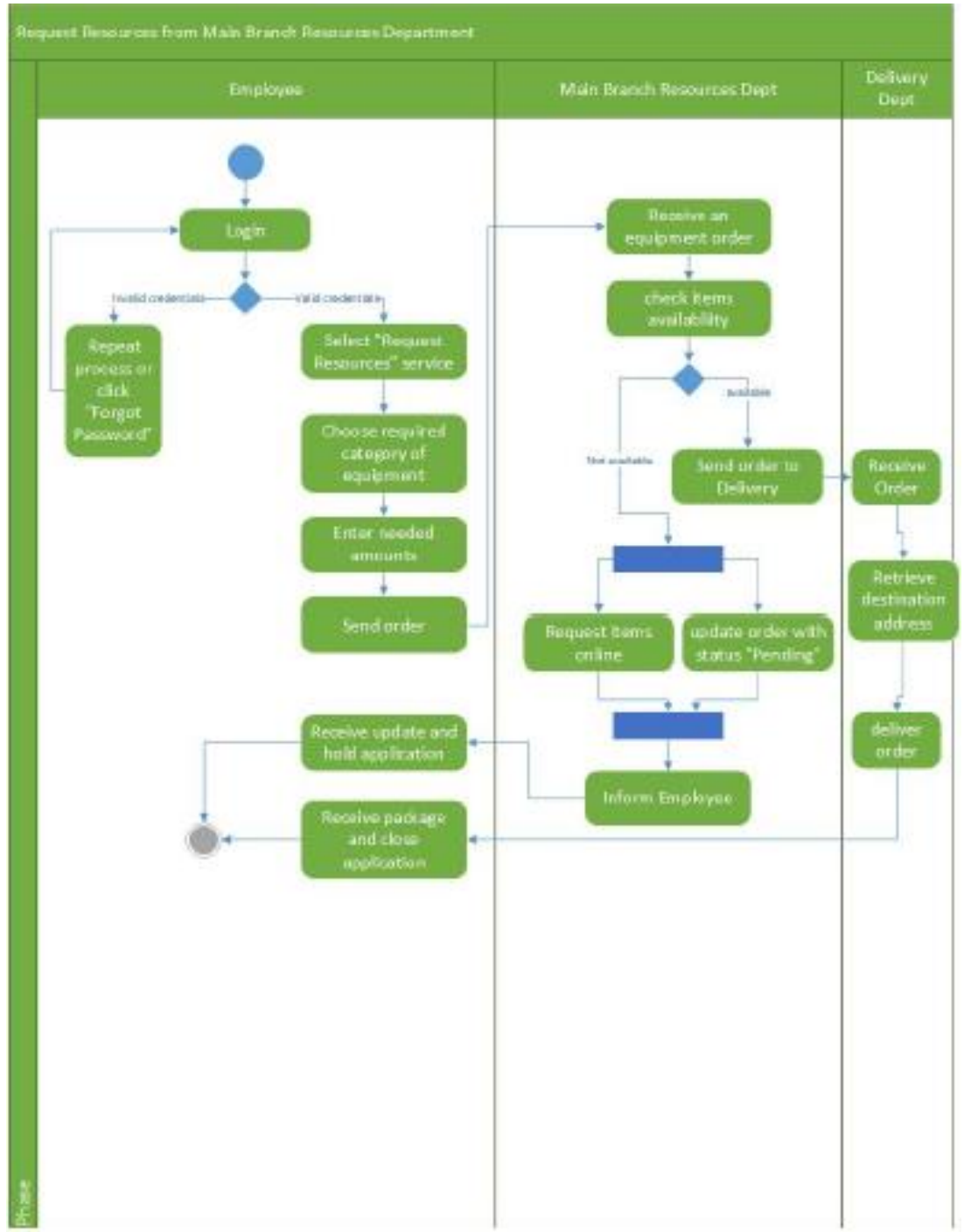


Figure 2-43 An example of fork and join nodes in a UML activity diagram.

Problem definition Q4

Our hospital has a main branch which has more medical materials in its local store. Sometimes, when there is a need to request some resources, a specific employee uses the system to request items from the main branch. If the items are available, the employee sends an order to the delivery department to deliver the package to its branch. Otherwise, they request the items online, and change the status of the request to be in the waiting list (Pending). In this part, draw an activity diagram that visualizes the described procedures. Use parallel flows where possible.

Study the diagram and discuss it with your colleagues


Request Resources from Main Branch Resources Department
Employee
Main Branch Resources Dept
Delivery Dept
Login
Repeat process or click 'Forgot Password'
Select "Request Resources" service
Choose required category of equipment
Enter needed amounts
Send order
Receive update and hold application
Receive package and close application
Receive an equipment order
check items availability
Send order to Delivery
Request items online
update order with status "Pending"
Inform Employee
Receive Order
Retrieve destination address
deliver order