Tutorial #1 – Requirements Elicitation – Functional / Non-Functional Requirements

1. Atomise the following descriptions for an Inventory and Asset Tracking System as functional requirements. Check that the atomized requirements are also verifiable.

   The System maybe queried. A query may contain a user number or a serial number. If the query contains a user number, all equipment assigned to that user is reported. If the query contains a serial number, the assignment for that computer is reported. All query results show the name, office and user number of a user, followed by the serial numbers and types of all computers assigned to that user, and the date of each assignment. If a user is not assigned a computer or a computer not assigned to a user, this is reported.

2. Identify several non-functional requirements for the above Inventory and Asset Tracking System.

1) 1 User must input query
1.1 query must contains user number
1.1.1 all equipment assigned to that user is reported
1.2 query must contains serial number
1.2.1 assignment for that computer is reported
1.3 query results must show name, office n user number of user
Followed by serial numb n type of all Comp assigned to user
1.4 System must report user is not assigned to Comp
1.8 System must report Comp not assigned to user

2) Non Functional requirement
↳ query must respond within 1 minute
↳ System take 2 minute to boot

Tutorial 1: Requirements Elicitation – Functional/Non-Functional Requirements

Question 1: Atomize the requirements for an Inventory and Asset Tracking System.

Check that the atomized requirements are also verifiable.

The System may be queried. A query may contain a user number or a serial number. If the query contains a user number, all equipment assigned to that user is reported. If the query contains a serial number, the assignment for that computer is reported. All query results show the name, office and user number of a user, followed by the serial numbers and types of all computers assigned to that user, and the date of each assignment. If a user is not assigned a computer or a computer not assigned to a user, this is reported.

Question 2: List several non-functional requirements for the above Inventory and Asset Tracking System

Question 1: Atomize the requirements

This tutorial is an exercise in splitting (atomizing) requirements statements and organizing them in a logical hierarchy. There should also be consideration for how the requirements can be verified.

Requirements specification heuristics:

- Write complete, simple sentences in the active voice.
- Define terms clearly and use them consistently.
- Group related material into sections.
- Express all requirements using the words “must” or “shall”.
- Write verifiable requirements.
- Write atomized requirements.

1. Users shall be able to query the System.

    1.1. Users must query the System by user number or by serial number.

        1.1.1. A report for a query containing a user number must report all equipment assigned to the user with that user number.

        1.1.2. A report for a query containing a serial number must report the assignment for the computer with that serial number.

    1.2. All computer assignment reports must have the same format.

        1.2.1. A report of computer assignments must display user data followed by computer data.

1.2.2. The user data display must consist of information about a user, or an indication that the computer is unassigned.

1.2.2.1. The user data displayed for an assigned computer must consist of user's name.

1.2.2.2. The user data displayed for an assigned computer must consist of office number.

1.2.2.3. The user data displayed for an assigned computer must consist of user's number.

1.2.2.4. If the user is not assigned a computer, the user data display must indicate that no computer is assigned to that user.

1.2.3. The computer data display must consist of a list of data for each computer assigned.

1.2.3.1. The data for an assigned computer must consist of the computer's serial number.

1.2.3.2. The data for an assigned computer must consist of the types of computers.

1.2.3.3. The data for an assigned computer must consist of the date of the assignment.

1.2.3.4. If the list of assigned computers is empty, the computer data must indicate that no computer is assigned to the user.

Question 2: Non-functional requirements

Nonfunctional Requirements (NFRs) define system attributes such as security, reliability, performance, maintainability, scalability, and usability. They serve as constraints or restrictions on the design of the system

Security: Authentication

Reliability: Consistency, accuracy

Performance: Response time

Maintainability: Easy to restart / recover

Scalability: Up capacity easily

Usability: Friendliness

Grain Elevator System (GES*) Case Study

*** Some tutorial questions refer to the GES as the subject system.**

The Grain Elevator System is a program for tracking the grain stored in a grain elevator as it arrives in trucks from farms, and is shipped out in trains from the elevator. See http://en.wikipedia.org/wiki/Storage_silo and http://en.wikipedia.org/wiki/Grain_elevator for an explanation of silo and grain elevator operations.

- Silo
- Common
- Seacot
- System

A. Harvested grain is trucked from farms to a central storage elevator where it is placed in silos. The grain is eventually moved from the silos to railroad cars that take it to processing plants. The elevator relies on a software system to track the grain.

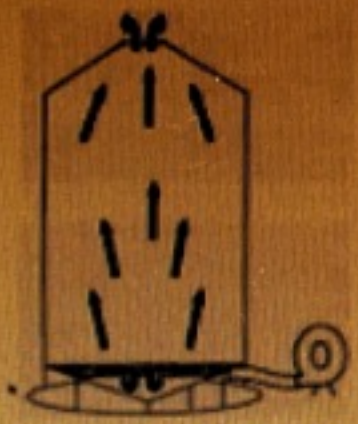
B. The elevator accepts shipments of wheat, barley, long grain rice, short grain rice, oats, and hops. Each type of grain has two grades, high and low. An empty silo may store any kind of grain, but a silo with grain in it can store only grain of the same type and grade.

C. There are 12 silos in the elevator: silos 1-6 silos hold 8,000 bushels each, and silos 7-12 hold 12,000 bushels each. Each truck carries between 150 and 180 bushels. A single railroad car holds 2,000 bushels.

D. Grain only arrives in trucks from growers selling the grain, and only leaves in rail cars taking it to processing plants buying the grain. Each truck has a plate number, a driver, and a grain seller. Each rail car has a serial number, a conductor, and a grain buyer.

E. Grain stored in the silo must be kept dry. Therefore, the humidity and temperature of each silo are closely monitored. Humidity and temperature sensors in each silo send data to the system periodically. If the temperature or humidity of a silo exceeds normal levels, the humidity and temperature sensors will send an alert to the system, as well as, to an external grain dryer. Until its temperature and humidity levels are brought back to normal, the silo cannot accept any more grain.

Diagram of airflow through a silo



F. For accounting purposes, the system maintains a transaction log that records, for each shipment in or out of the elevator, all the information about the shipment, namely: the quantity, type, and grade of grain, the time and date, the elevator manager, the type of shipment (arrival or departure), the truck or rail car identifier, the driver or conductor, and the seller or buyer.

G. When a truck arrives at the elevator with a load of grain, the elevator manager informs the system of the type of grain, its grade, and its quantity. The system must find one or more silos to store to store the grain and tell the elevator manager which silos it has chosen, and how much grain goes in each one. The system may accept only part of a load if there is not room for the entire load, or none of it if there is no place to store it.

H. The elevator manager may accept or overrule the system's choice of silos for an arriving load of grain. The elevator manager must inform the system how much grain is actually deposited in each silo, and enter data about the truck, the driver, and the seller. The system should acknowledge receipt of this data.

I. As a train is loaded, the elevator manager must tell the system how much grain has been removed from each silo, the rail cars loaded, the conductor, and the buyer. The system should acknowledge receipt of this data.

J. Upon request from the elevator manager, the system must produce a complete report of the state of the elevator. This report should list, for each silo, the type of grain stored, the amount stored, and the remaining capacity of the silo. The report should also list the total remaining capacity of the elevator for each type of grain currently stored, and the total capacity of the elevator not currently committed to any type of grain.

K. Upon request from the elevator manager, the system must produce a chronological listing of the transaction log.

Tutorial #2 – Requirements Analysis – Use Case & Conceptual Model in a Class Diagram

1. Refer to the description of the Grain Elevator System (GES).

   (a) Identify the Actors and their main Use Cases and draw a complete Use Case diagram for the Grain Elevator System. Wherever appropriate, use generalization, include and extend relationships.

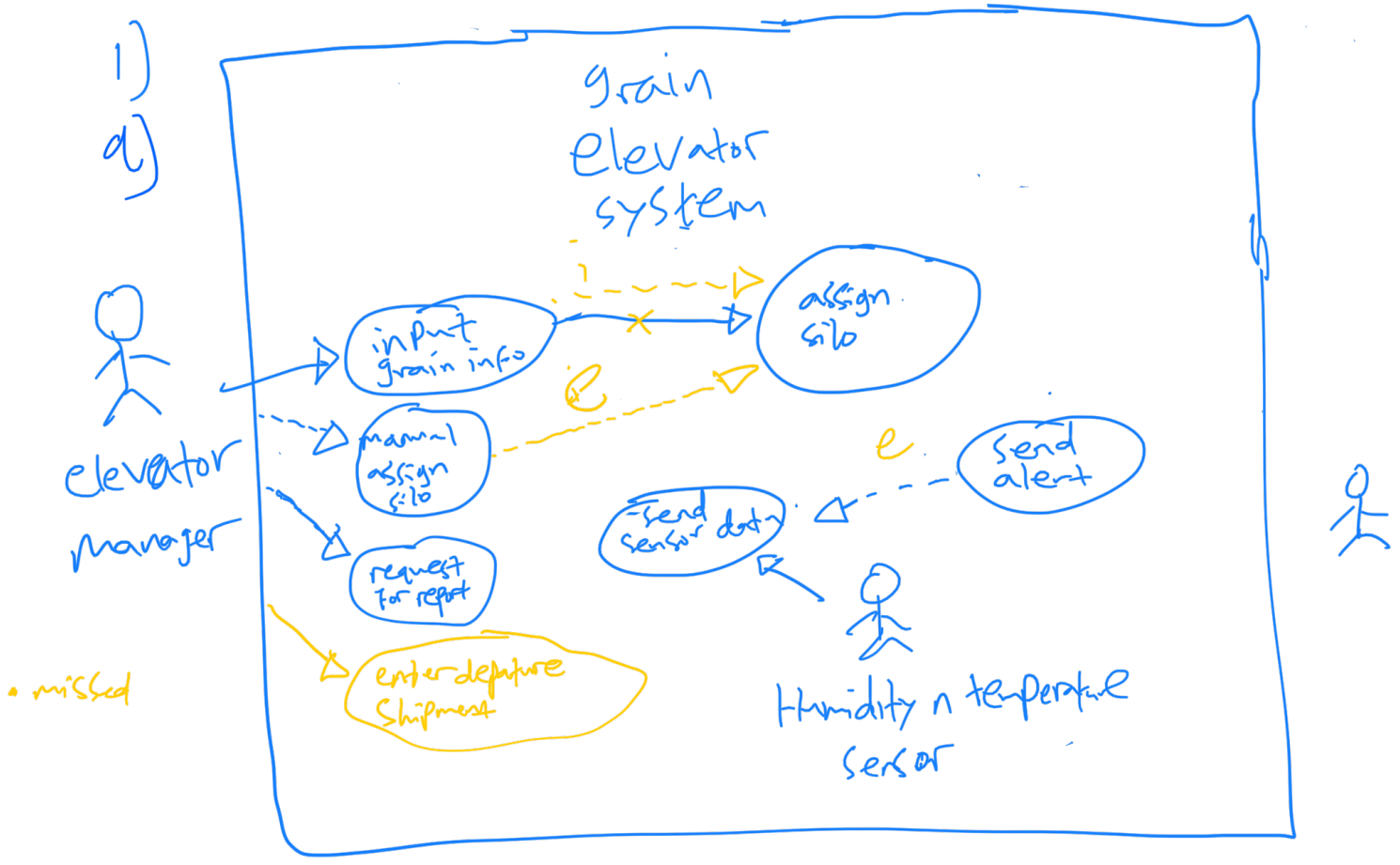
   (b) Write the Use Case description for the use case(s) that encompasses the grain shipment arrival and subsequent allocation functionality.

   (c) Determine and document important terms in the Data Dictionary

2. Refer to the shipment arrival and allocation functionality obtained in the above question.

   (a) From the Use Case description and Data Dictionary, identify the classes to produce a conceptual model.

   (b) Depict the classes identified in part (a) and the associations/dependencies between them in a Class Diagram.



b)

c)

term	definition
elevator manager	human
silo	hold 1 type of grain
Transaction log	

2) a)

Tutorial 2: Requirements Elicitation – Use Case

Question 1: Grain Elevator System (GES)

(a) Make a complete use case model for the Grain Elevator System. Wherever appropriate, use include and extend relationships.

Points to take note:

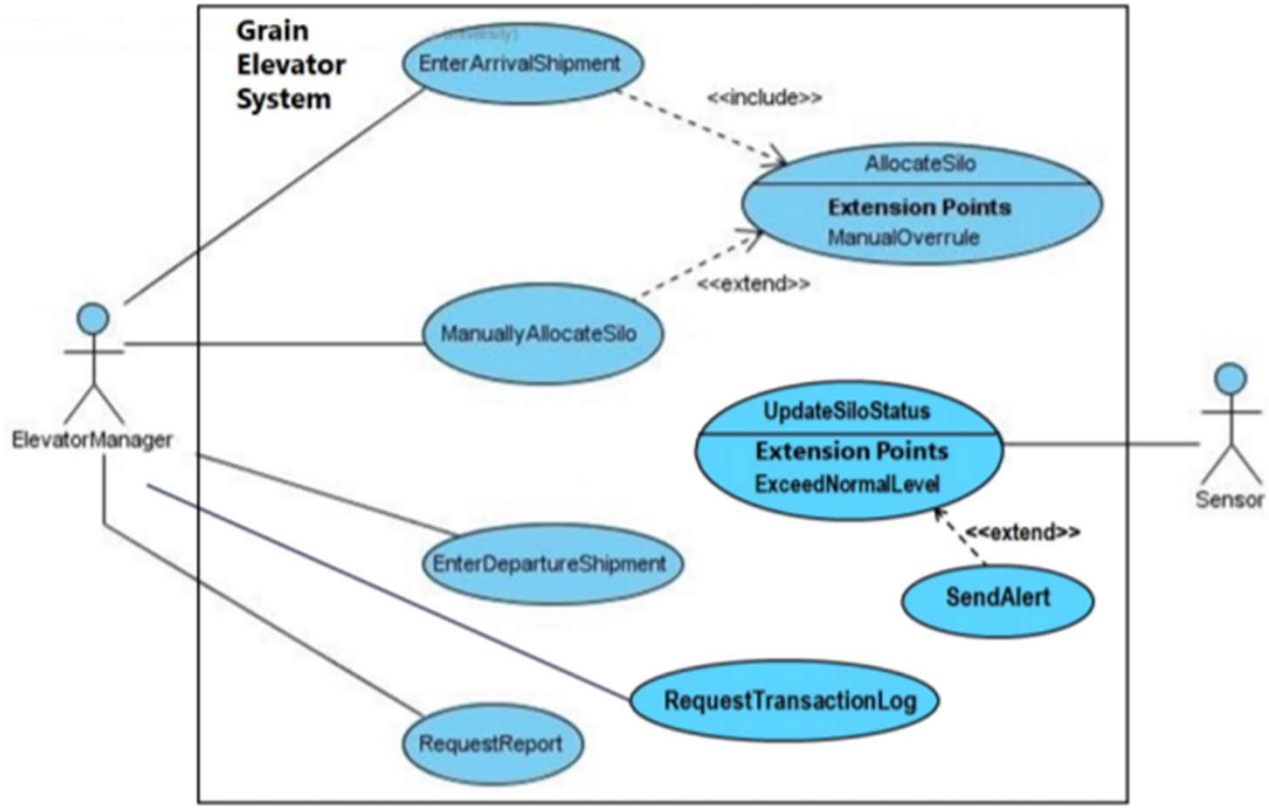
Pay attention to the following points:

- The directions of <<include>> and <<extend>> relationships.
- An actor can be a human, a device (or an equipment), or an external system interacts with the current system, the Grain Elevator System.
- The system itself (the system the use case diagram is modeling, i.e. Grain Elevator System) cannot be drawn as an actor (stick figure in the use case diagram). Only external system (if any) interacting with the system (you are modeling) is drawn as an actor.
- For <<extend>> relationship, extension point to be clearly specified.
- Use case diagram is not unique.

Tutorial 4: Requirements Elicitation – Use Case

Question 1: Grain Elevator System (GES)

(a) Make a complete use case model for the Grain Elevator System. Wherever appropriate, use include and extend relationships.



Question 1: Grain Elevator System (GES)
(b) Use Case Description

Pay attention to the following points:

- Terms from the data dictionary to be used in description.
- Actor does something, GES (the system) responds.
- Specify whether the actor is an initiating actor or a participating actor.
- Textual representation of <<include>> relationships (refer to step 5 in Flow of Events of *EnterArrivalShipment* use case and Entry Condition of *AllocateSilo* use case).
- For the textual representation of <<extend>> relationships, please refer to Bruegge text pg. 48 Figure 2-18 example.

Question 1: Grain Elevator System (GES)
(b) Use Case Description (cont'd)

EnterArrivalShipment Use Case
Initiating Actor: ElevatorManager

Flow of Events:

1. ElevatorManager selects CreateNewShipment on UI.
2. System displays form for Shipment details – GrainType, GrainQuantity, Farm, TruckId, ArrivalDateTime.
3. ElevatorManager enters required information and submits form.
4. System validates data.
5. System allocates the Shipment to be stored in the Silo using the included use case AllocateSilo.
6. System displays the allocation result (success or failure) for the Shipment.

Question 1: Grain Elevator System (GES)
(b) Use Case Description (cont'd)

AllocateSilo Use Case
Entry Condition: Invocation as an included use case by EnterArrivalShipment.

Flow of Events:

1. System provides GrainType and GrainQuantity.
2. System cycles through the Silo list to determine a match for GrainType.
3. When a match is found, System determines if the Silo can contain the GrainQuantity.
4. If the Silo cannot contain the full GrainQuantity, System will fill the current Silo and continue to cycle through the rest of the Silos to deposit the remaining GrainQuantity.
5. If the full GrainQuantity is deposited, System returns a successful allocation. Else System return a failed allocation status.

Points to take note:

Note that the use case descriptions provided above do not include other fields such as Description, Exit Condition (or Postcondition), Alternative Flows, Exceptions, Includes, etc. Please refer to the Use Case template in NTULearn for the complete fields.

Question 1: Grain Elevator System (GES)
(c) Data Dictionary

Term	Definition
Elevator Manager	Human user of system to manage grain storage
Grain	Delivered by truck, stored in silo, distributed by train 6 types: wheat, barley, long grain rice, short grain rice, oats, hops 2 grades: high, low
Railroad Car	Carries 1 type of grain of certain grade from silo to processing plant
Report	Requested by Elevator Manager to reflect status of grain elevator, can be presented per silo or per grain type
Sensor	Monitors humidity and temperature in silo Sends data and alert to the System
Shipment	Arrival Shipment or Departure Shipment Records the type / grade of grain & quantity, together with transport information
Silo	Holds 1 type of grain of certain grade 2 sizes: 8000 bushels, 12000 bushels
Transaction Log	List of Shipments, can be sorted in chronological order
Truck	Carries 1 type of grain of certain grade from farm to silo

Tutorial 4: Requirements Analysis - Conceptual Model in ClassDiagram

Question 2: Refer to the shipment arrival and allocation functionality use cases

(a) From the use-case description and data dictionary (see Question 1), identify the classes to produce a conceptual model.

Points to node:

- **The conceptual model class diagram is developed during the analysis phase of the SDLC. Thus, it is not a detail class diagram which is to be developed during the design phase.**
- **The main focus of the conceptual model is to identify the boundary, control, and entity classes, and the associations and/or dependencies between these classes.**
- **Attributes and operations are not mandatory to be included in the conceptual model.**

(a) From the use-case description and data dictionary, identify the (analysis) classes to produce a conceptual model. (cont'd)

Communication allowed:

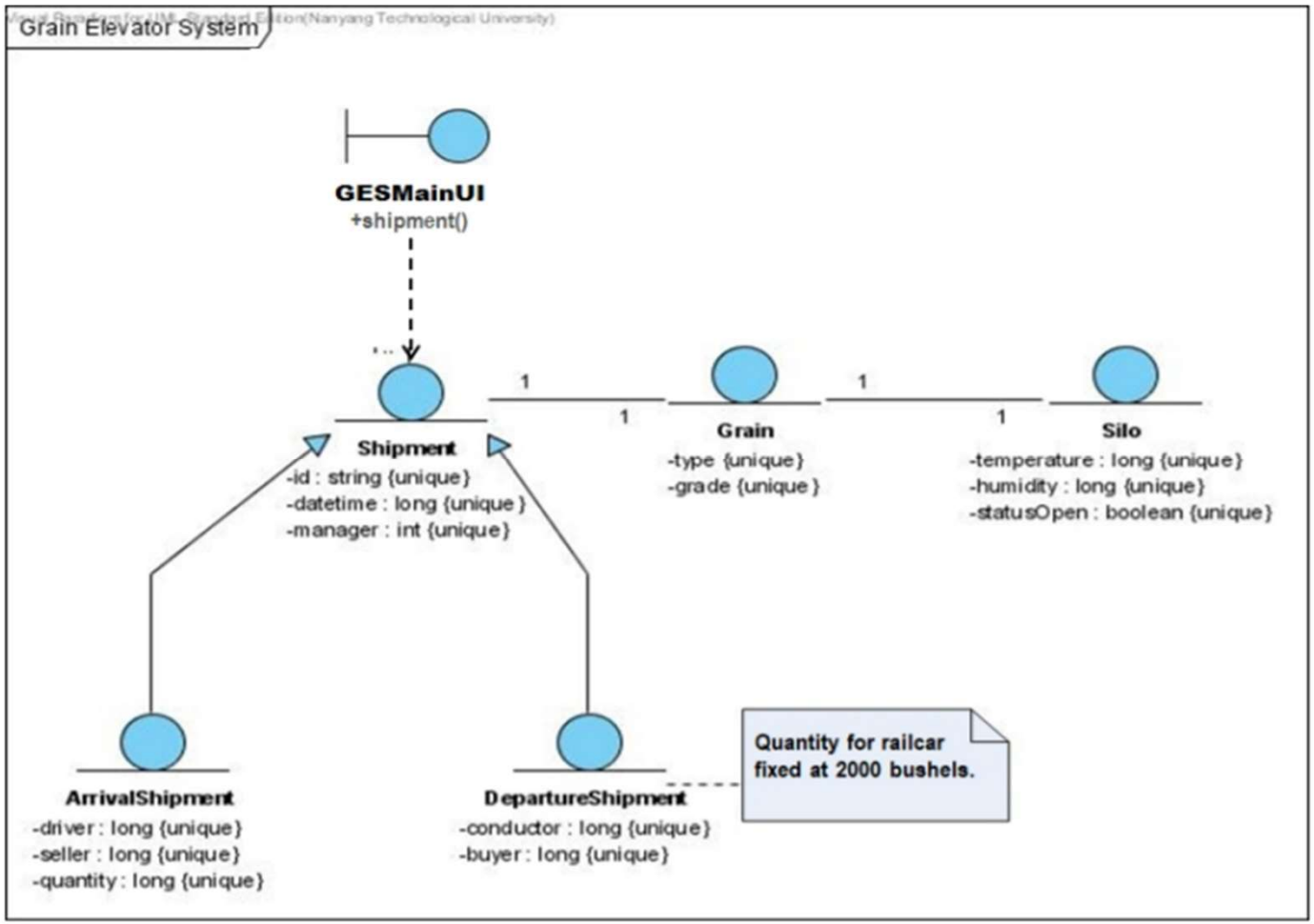
	Entity	Boundary	Control
Entity	X		X
Boundary			X
Control	X	X	X

Entities: Objects representing system data, often from the domain model.

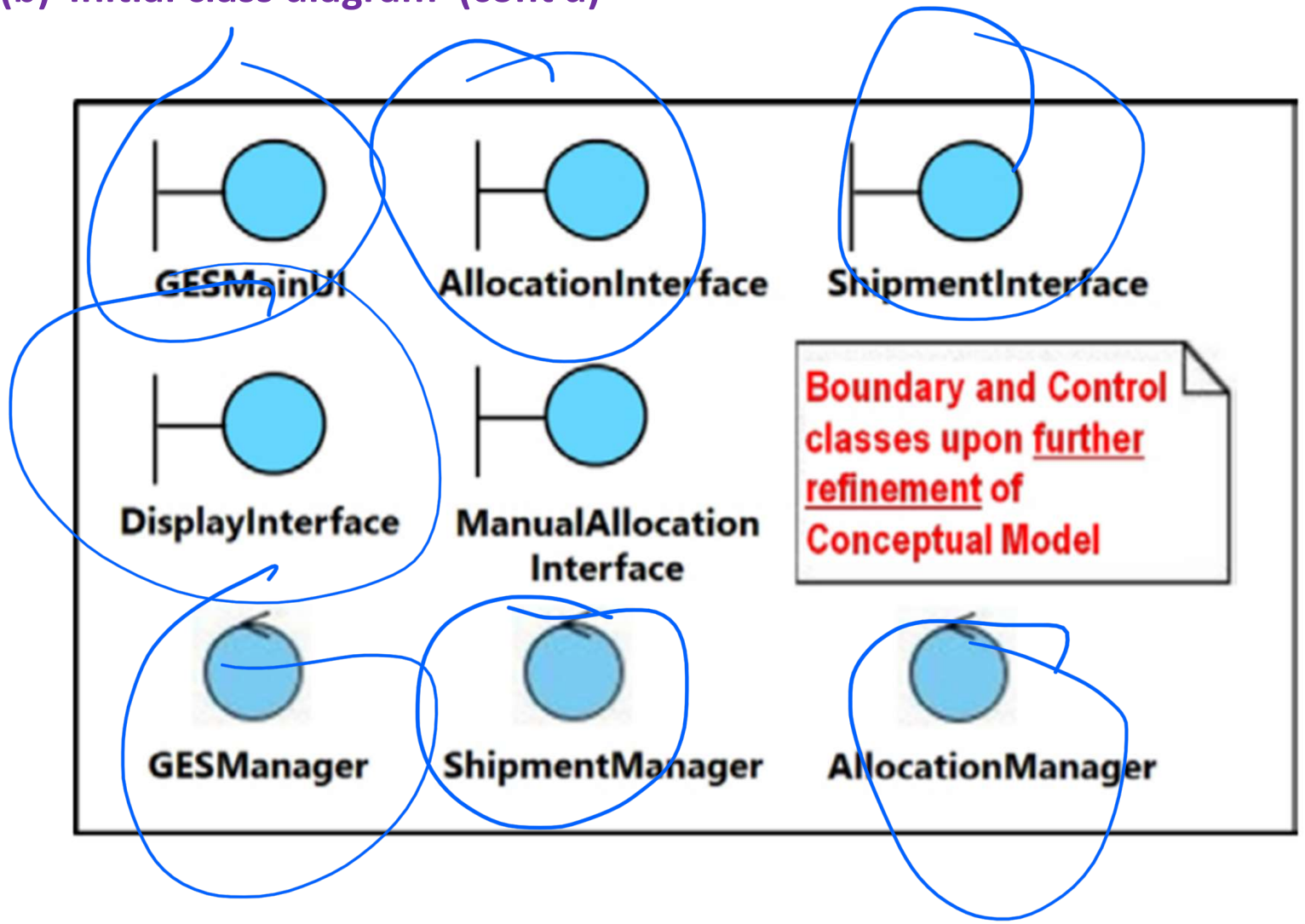
Boundaries: Objects that interface with system actors (e.g. a user or external service). [Windows, screens and menus are examples of boundaries that interface with users.]

Controls: Objects that mediate between boundaries and entities. These serve as the glue between boundary elements and entity elements, implementing the logic required to manage the various elements and their interactions.

Question 2: Grain Elevator System (GES)
(b) Initial class diagram



(b) Initial class diagram (cont'd)



Tutorial #3 – Requirements Analysis – Dynamic Models in Sequence Diagram and State Machine Diagram

1. Grain Elevator System (GES). Refer to the shipment arrival and allocation functionality obtained in Tutorial #4.

   (a) In a Sequence Diagram, model the interaction amongst boundary / control / entity classes to enact the shipment arrival functionality.
   Demonstrate how the messages passed between objects enact the Use Case's flow of events.

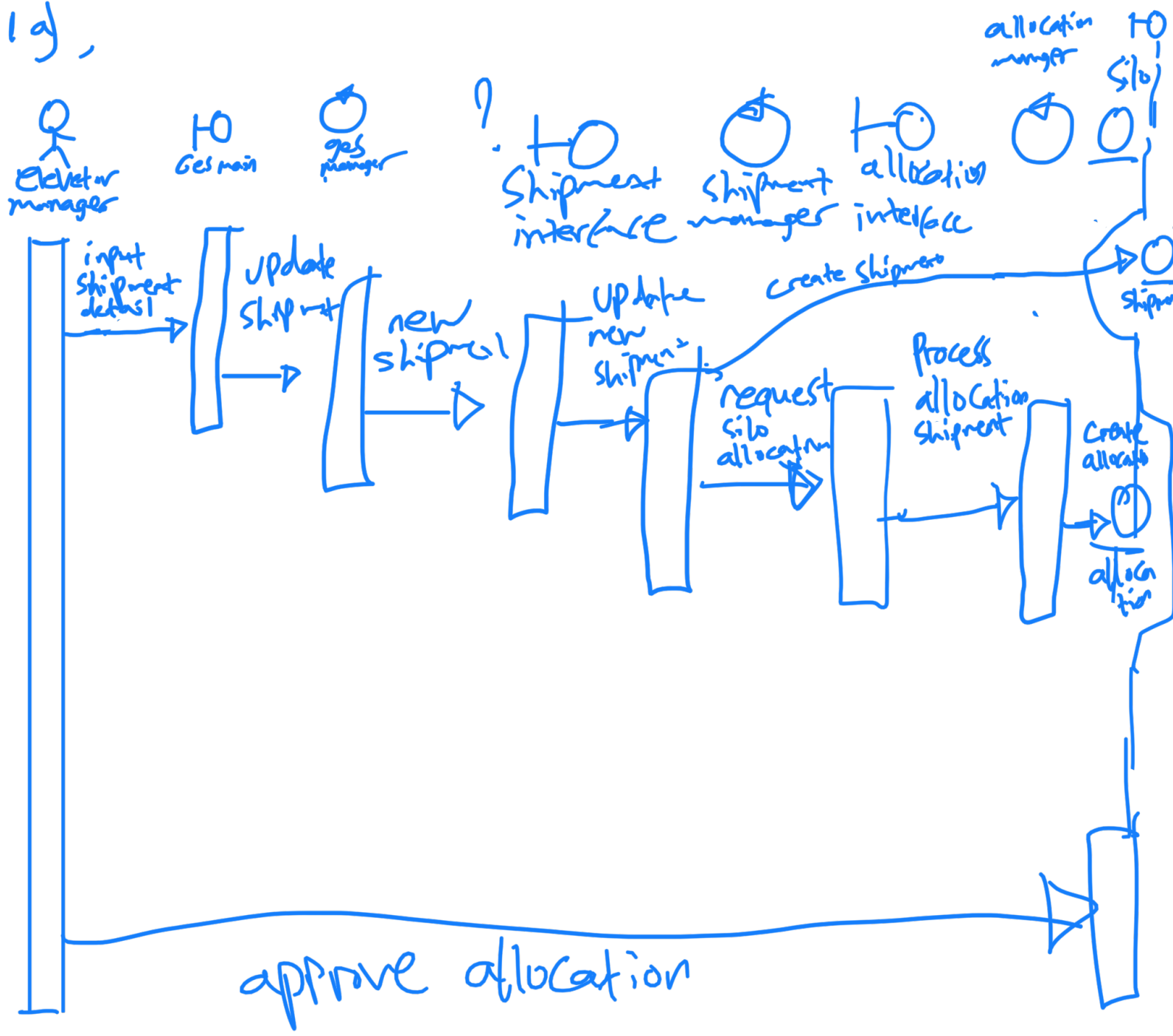
   (b) Refine the conceptual model obtained in Tutorial #4 using details uncovered in question 1(a) above by adding:
   - associations,
   - generalization relationships,
   - properties / attributes,
   - operations, and
   - new classes (if any)

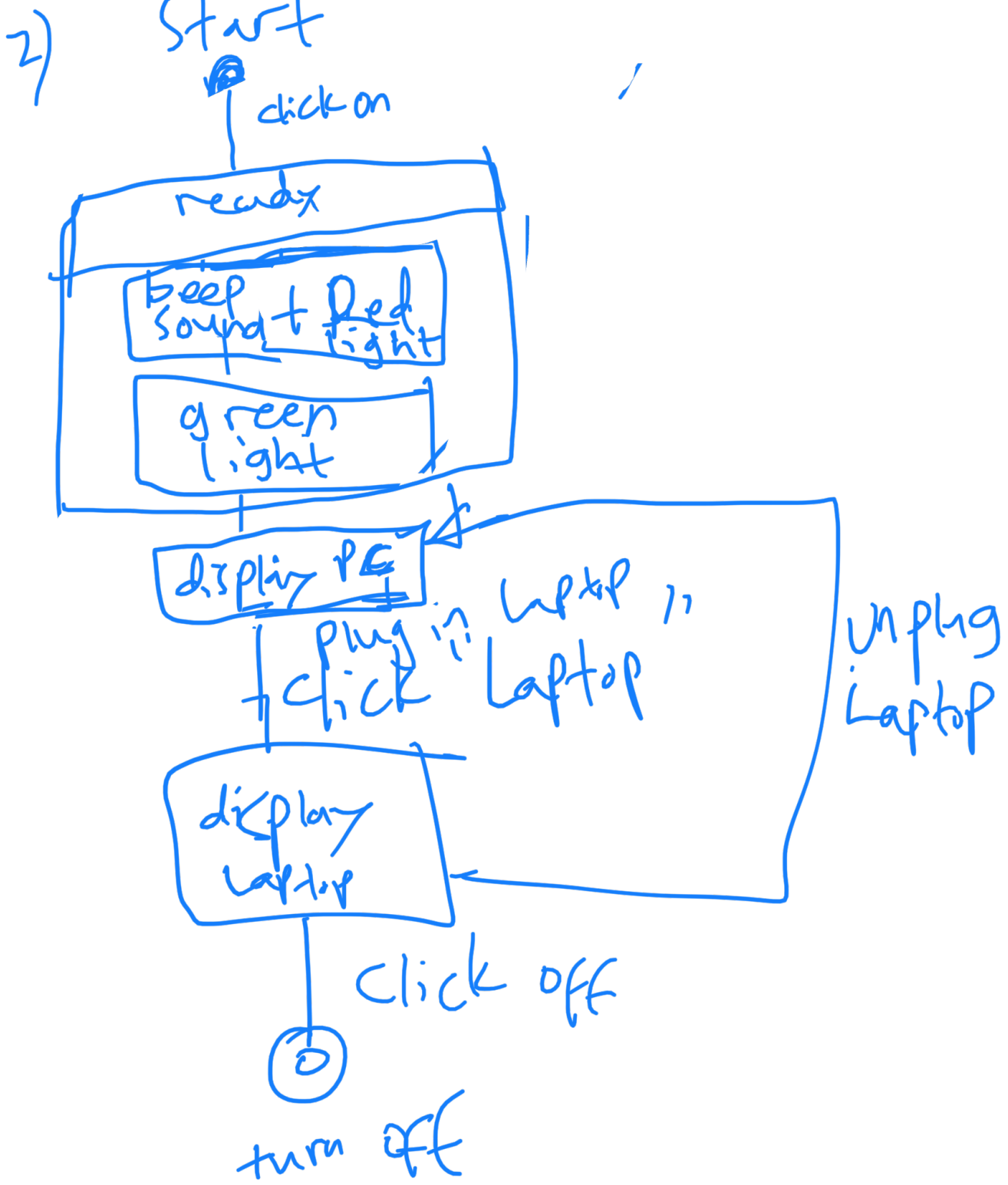
2. 8:30am lectures are usually frustrating for the lecturer. Projectors, having been turned off at the end of the previous day and take a long time to become ready for use.

   Here's what Professor Jane does at the Projector Controller Panel, in order to project slides from her laptop:-

   - Turn on the projector
   - As the projector warms up, it first gives a beep (sound) and a red indicator light continues to blink
   - When ready, the indicator light turns to green; the projector takes input from the PC by default
   - Jane presses the "LAPTOP" switch

   After the lecture, Jane could turn off the projector by pressing the "OFF" button. However, to save her colleagues the angst, she decides to leave it ON.
   Model the projector's system behavior using a State Machine diagram (can also be called a dialog map.


1 g).
allocation manager
Silo
Elevator manager
Ges main
gas manager
Shipment interface
Shipment manager
allocation interface
input shipment detail
update shipment
new shipment
update new shipment
create shipment
request silo allocation
Process allocation shipment
approve allocation

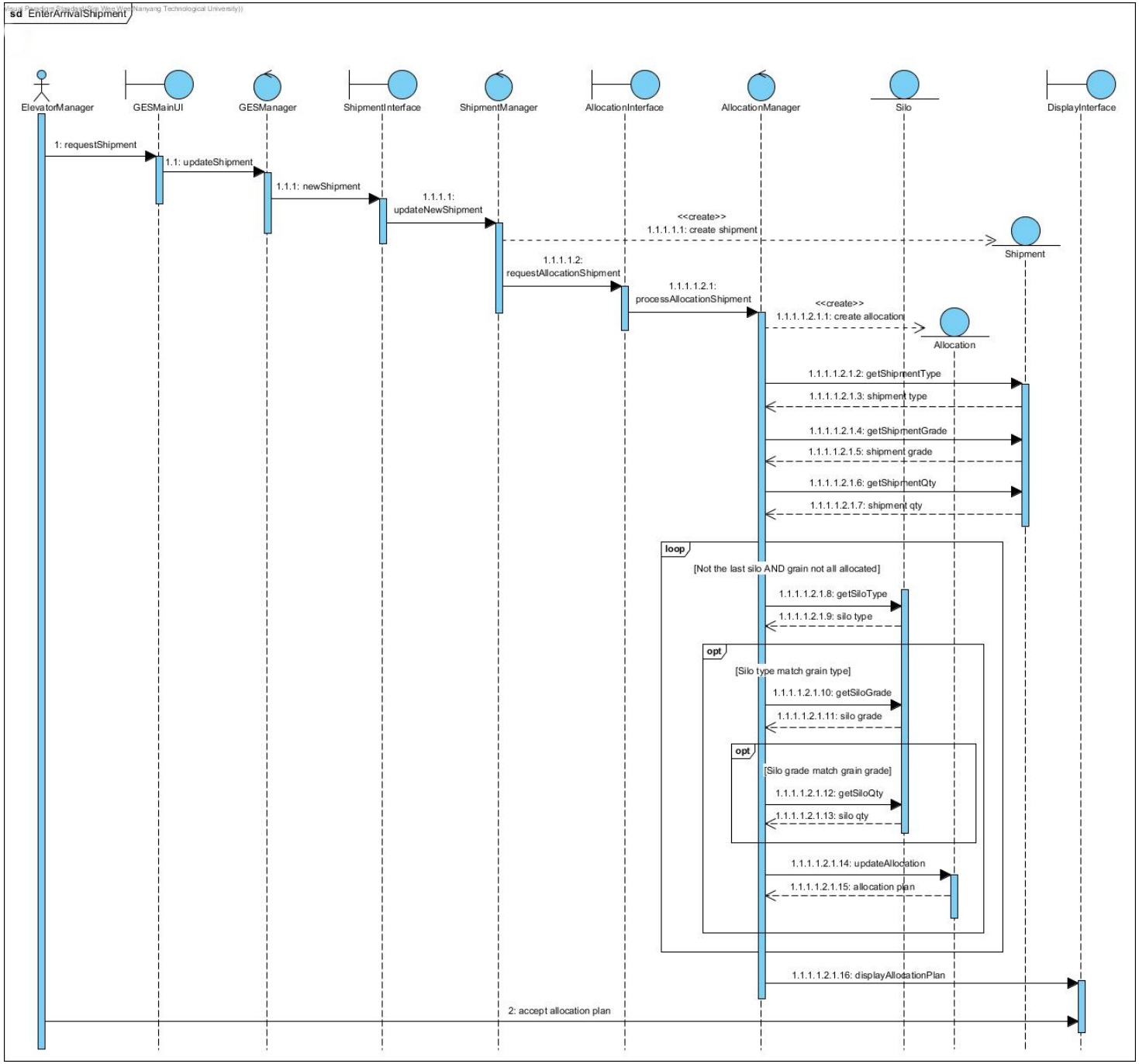

2)
Start
click on
ready
beep sound + Red light
green light
display PE
Plug in Laptop
click
Laptop
Unplug Laptop
display Laptop
Click off
turn off

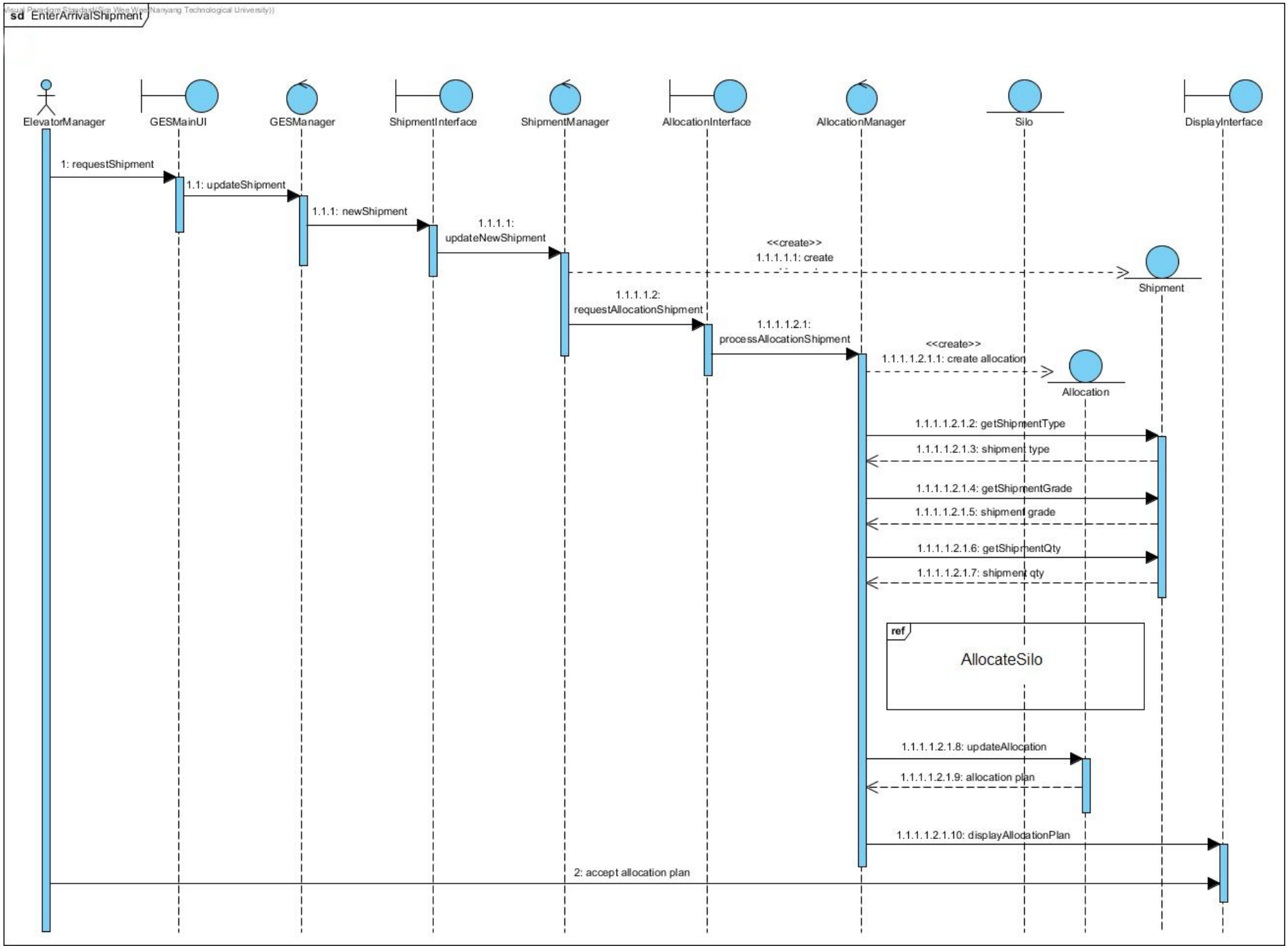
Tutorial 3: Requirements Analysis – Dynamic Models

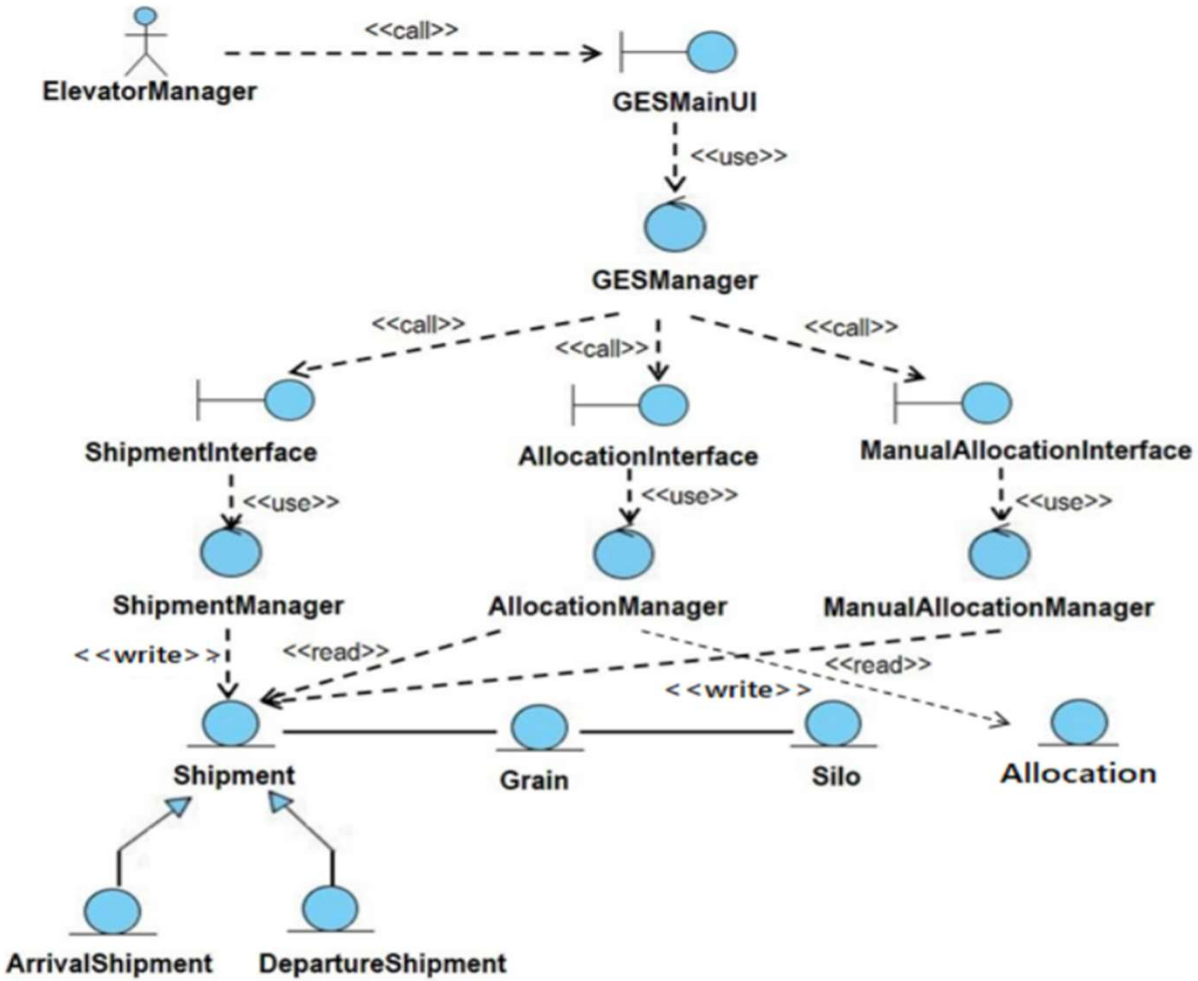
Question 1: Grain Elevator System (GES)

(a) Draw a sequence diagram to model the interaction amongst the classes to enact the shipment arrival functionality.

(b) Refine the conceptual model from last week tutorial.


sd EnterArrivalShipment
ElevatorManager
GESMainUI
GESManager
ShipmentInterface
ShipmentManager
AllocationInterface
AllocationManager
Silo
DisplayInterface
Shipment
Allocation
1: requestShipment
1.1: updateShipment
1.1.1: newShipment
1.1.1.1: updateNewShipment
<<create>>
1.1.1.1.1: create shipment
1.1.1.1.2: requestAllocationShipment
1.1.1.1.2.1: processAllocationShipment
<<create>>
1.1.1.1.2.1.1: create allocation
1.1.1.1.2.1.2: getShipmentType
1.1.1.1.2.1.3: shipment type
1.1.1.1.2.1.4: getShipmentGrade
1.1.1.1.2.1.5: shipment grade
1.1.1.1.2.1.6: getShipmentQty
1.1.1.1.2.1.7: shipment qty
loop
[Not the last silo AND grain not all allocated]
1.1.1.1.2.1.8: getSiloType
1.1.1.1.2.1.9: silo type
opt
[Silo type match grain type]
1.1.1.1.2.1.10: getSiloGrade
1.1.1.1.2.1.11: silo grade
opt
[Silo grade match grain grade]
1.1.1.1.2.1.12: getSiloQty
1.1.1.1.2.1.13: silo qty
1.1.1.1.2.1.14: updateAllocation
1.1.1.1.2.1.15: allocation plan
1.1.1.1.2.1.16: displayAllocationPlan
2: accept allocation plan

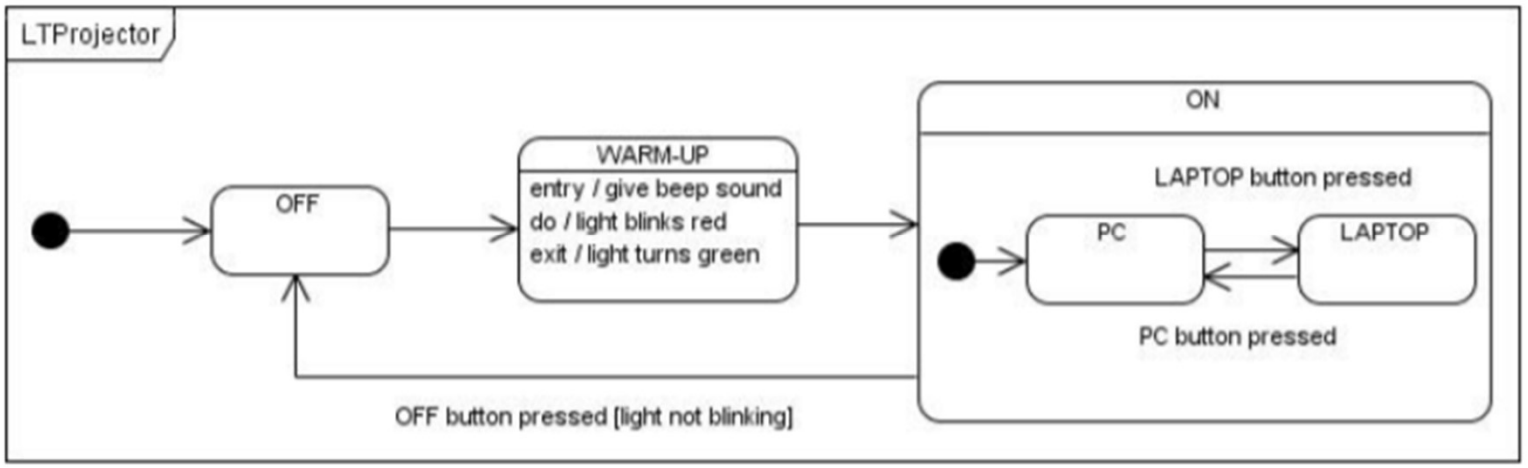

sd EnterArrivalShipment
ElevatorManager
GESMainUI
GESManager
ShipmentInterface
ShipmentManager
AllocationInterface
AllocationManager
Silo
DisplayInterface
1: requestShipment
1.1: updateShipment
1.1.1: newShipment
1.1.1.1: updateNewShipment
<<create>>
1.1.1.1.1: create
Shipment
1.1.1.1.2: requestAllocationShipment
1.1.1.1.2.1: processAllocationShipment
<<create>>
1.1.1.1.2.1.1: create allocation
Allocation
1.1.1.1.2.1.2: getShipmentType
1.1.1.1.2.1.3: shipment type
1.1.1.1.2.1.4: getShipmentGrade
1.1.1.1.2.1.5: shipment grade
1.1.1.1.2.1.6: getShipmentQty
1.1.1.1.2.1.7: shipment qty
ref
AllocateSilo
1.1.1.1.2.1.8: updateAllocation
1.1.1.1.2.1.9: allocation plan
1.1.1.1.2.1.10: displayAllocationPlan
2: accept allocation plan


ElevatorManager
<<call>>
GESMainUI
<<use>>
GESManager
<<call>>
<<call>>
<<call>>
ShipmentInterface
AllocationInterface
ManualAllocationInterface
<<use>>
<<use>>
<<use>>
ShipmentManager
AllocationManager
ManualAllocationManager
<<write>>
<<read>>
<<read>>
<<write>>
Shipment
Grain
Silo
Allocation
ArrivalShipment
DepartureShipment

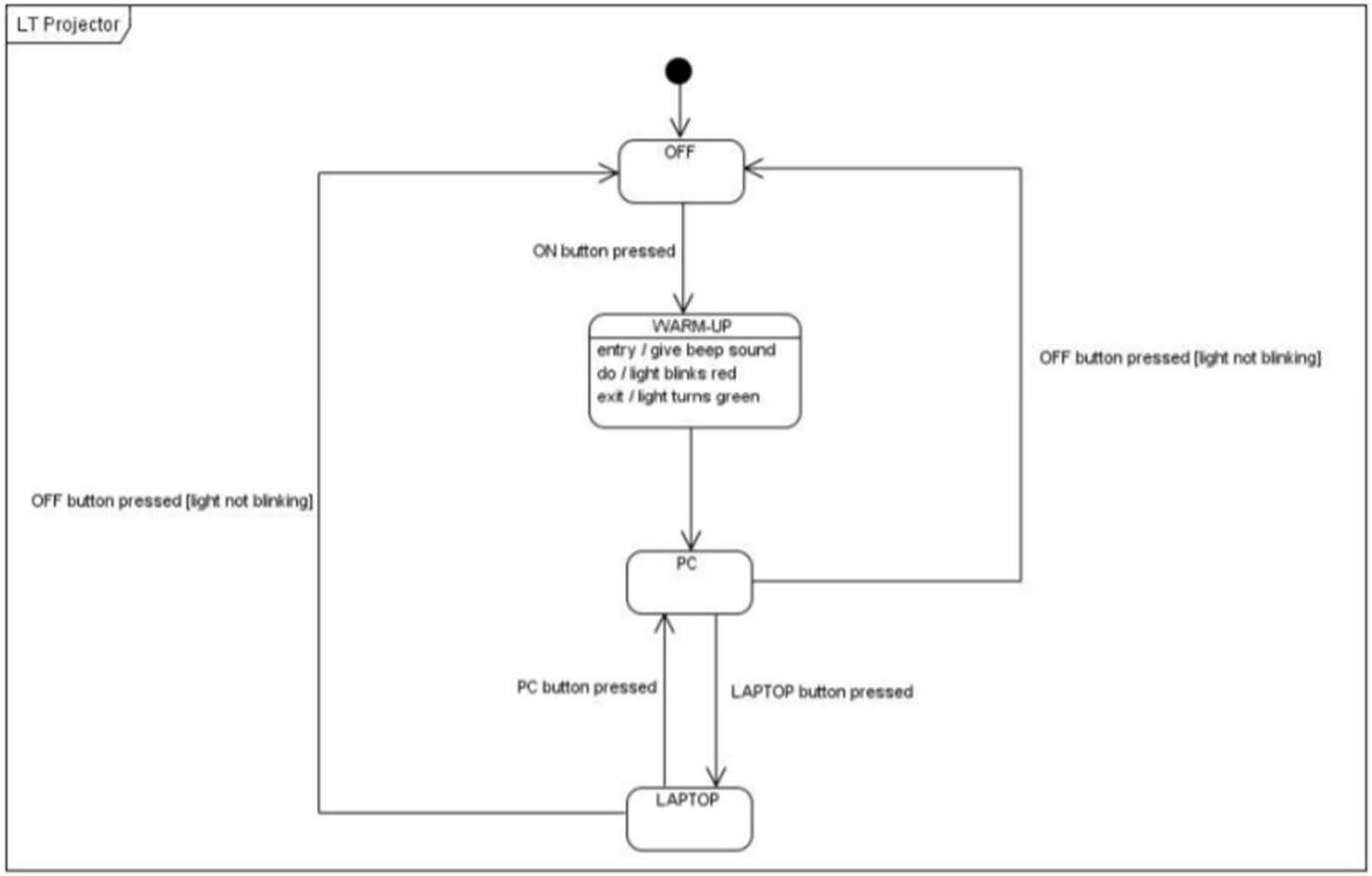
Tutorial 5: Requirements Analysis – Dynamic Models

Question 2: Projector Controller Panel

Model the projector's system behavior using a state machine diagram (dialog map).



An alternate state machine diagram



Tutorial #4 – Software Processes

1. Why are software development costs so high?

2. Why do we spend so much time and effort maintaining existing software programs?

3. Discuss on the characteristics of software that affect its development.

4. What are the two distinct differences between software development and hardware manufacturing?

5. Sum up Agile manifesto in four sentences, then discuss the key Agile principles.

6. Discuss the following characteristics of eXtreme Programming (XP):
   i. When to use XP
   ii. Pair programming
   iii. Test-driven development (TDD)
   iv. Code refactoring

Tutorial 4

Question 1: Why are software development costs so high?

Ans:

1. Unacceptably low quality requires more rework (design, implementation, and testing) than expected.
2. Development of the wrong software functions requires redesign and implementation.
3. Development of the wrong user interface results in redesign and implementation.
4. Development of extra software functions that are not required extends the schedule.

1

Question 2: Why do we spend so much time and effort maintaining existing software programs?

Ans:

1. Software systems are subject to continuing change even after it is built.
2. There are many staff and organizational reasons that make maintenance difficult: limited understanding, morale and management priorities.
3. Technical problems also affect maintenance productivity: inadequate design specification, low-quality programs and documentation, testing difficulties.
4. Real-time and highly synchronized systems are more difficult to change.
5. A system designed to last many years is likely to require more maintenance.
6. System dependent on its hardware's characteristics is likely to require many changes.

2

Question 3: Discuss on the characteristics of software that affect its development.

Ans:

1. **Malleability** - Software is easy to change (all programmers are often tempted to 'tweak' their code). This malleability creates a constant pressure for software to be changed rather than replaced.
2. **Complexity** - Software is often complex. Complexity can usually be recognized, but it is less easy to define. One item of software can be considered more complex than another if the description of the first requires more explanation than that of the second. Part of the complexity arises from the potential variety of pathways between the components of a system.
3. **Size** - It is likely that there will be more errors in a large piece of software than there will be in a small one. Evidence suggests that the number of errors increases roughly in proportion to the square of the size of the software. All things being equal, an item of software that is twice as big as another is likely to have four times the number of errors.

3

Question 4: What are the two distinct differences between software development and hardware manufacturing?

Ans:

1. In software development, we do not produce the same product again and again based on the same process and design, but we do that in manufacturing: In manufacturing, for producing an identical copy of the product, we need to run the same process based on the same design. As such, we produce the same product again and again based on the same design and process. Proper statistical analysis can be carried out to analyze the earlier feedbacks and experiences to improve the latter production. In software development, for producing an identical product, we just need to make another copy without any further effort. When we carry out a new software development, it is surely that we will base on different requirement and/or different design. It is hard for the later development to benefit from the earlier experiences and feedbacks through proper statistical analysis as they are carried out on different basis.

2. Software does not wear out, as a result, maintenance does not mean component replacement: As such, there is no clear cut timing to phase out a piece of software. As a result, users are likely to pressurize software engineers to incorporate more and more new requirements after a system is implemented. This is likely to corrupt the overall structure of the software and deteriorate its quality.

Due to the above key differences, requirements engineering are carried out more frequently in software development.

4

Question 5: Sum up Agile manifesto in four sentences, then discuss the key Agile principles.

Ans:

Individuals and interactions over **processes and tools**

Working software over **comprehensive documentation**

Customer collaboration over **contract negotiation**

Responding to change over **following a plan**

5

Question 5: Sum up Agile manifesto in four sentences, then discuss the key Agile principles. (cont'd)

Ans:

(http://agilemanifesto.org/principles.html)

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4. Business people and developers must work together daily throughout the project.
5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
7. Working software is the primary measure of progress.
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9. Continuous attention to technical excellence and good design enhances agility.
10. Simplicity--the art of maximizing the amount of work not done--is essential.
11. The best architectures, requirements, and designs emerge from self-organizing teams.
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

6

Question 6: Discuss the following characteristics of eXtreme Programming (XP):

i. When to use XP
ii. Pair programming
iii. Test-driven development (TDD)
iv. Code refactoring

Ans:

i.

- Involve new or prototype technology, where the requirements change rapidly, or some development is required to discover unforeseen implementation problems.
- Are research projects, where the resulting work is not the software product itself, but domain knowledge.
- Are small and more easily managed through informal methods.

7

Question 6: Discuss the following characteristics of eXtreme Programming (XP):

i. When to use XP
ii. Pair programming
iii. Test-driven development (TDD)
iv. Code refactoring

Ans:

ii.

- All production software in XP is built by two programmers, sitting side by side, at the same machine.
- While the first developer focuses on writing, the other one reviews code, suggests improvements, and fixes mistakes along the way.
- All production code is reviewed by at least one other programmer, and results in better design, better testing, and better code and ultimately a higher-quality software.
- Faster knowledge sharing.

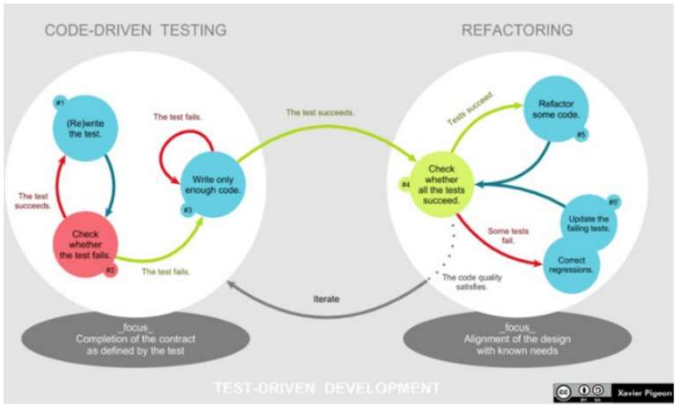
8

Question 6: Discuss the following characteristics of eXtreme Programming (XP):

i. When to use XP
ii. Pair programming
iii. Test-driven development (TDD)
iv. Code refactoring

Ans:
iii.

- Refers to a style of programming in which 3 activities are tightly interwoven: coding, testing (in the form of writing unit tests) and design (in the form of refactoring).



Source: https://en.wikipedia.org/wiki/Testdriven_development#/media/File:TDD_Global_Lifecycle.png

9

Question 6: Discuss the following characteristics of eXtreme Programming (XP):

i. When to use XP
ii. Pair programming
iii. Test-driven development (TDD)
iv. Code refactoring

Ans:
iv.

- The refactoring process focuses on:
  - Removal of duplication
  - Increasing the "cohesion" of the code, while lowering the "coupling"
- A well designed code must be HIGH cohesion and LOW coupling

10

Tutorial #5 – Scrum

1. What are the considerations crucial to deciding whether to adopt Agile/Scrum development in a project?

2. Discuss the ways of estimating Velocity in Scrum.

3. What are the characteristics of a good product backlog?

4. What does the INVEST mnemonic of Agile software development stand for?

5. Under Scrum methodology, a Product Increment is a piece of software that is both complete and potentially shippable. Why is it important to have a working software delivered at the end of each Sprint?

6. The total size (in points) of a list of product backlog items is 64. The Scrum Team consists of Product Owner, Scrum Master and six members in the Development Team. The cost of each member in the Scrum Team for each Sprint is $2K. Eight Sprints is determined to be needed to complete the project. Calculate the velocity and cost of this project.

Tutorial 5

Question 1: What are the considerations crucial to deciding whether to adopt Agile/Scrum development in a project?

Tutorial 1

Question 1: What are the considerations crucial to deciding whether to adopt Agile/Scrum development in a project?

Ans:

- **Team Experience & capability**
  - o Matured developers that can take multiple roles in the Development Team (cross functional) or other valid justifications
  - o Agile Development should be built on motivated individuals. Having the right people is essential for agile projects since it depend on the individuals' capability.
- **Customer involvement & commitment**
  - o Customers can provide continuous feedback to the developer team which help them to deliver frequent releases of working software to customers.

Tutorial 1

Question 1: What are the considerations crucial to deciding whether to adopt Agile/Scrum development in a project?

Ans: (cont'd)

- **Requirement definition***
  - o Constantly changing or other valid justifications
- **Change**
  - o Frequent fluctuation of change or other valid justifications
- **Organizational Culture**
  - o Having a dynamic culture to respond to frequent changes during the agile development lifecycle
  - o An organizational mindset shift on developing a project with frequent engagement and interaction from customers.

Tutorial 1

Question 1: What are the considerations crucial to deciding whether to adopt Agile/Scrum development in a project?

Ans: (cont'd)

- **Documentation**
  - o More flexibility to streamline minimum documentation or other valid justifications

Question 2: Discuss the different ways of estimating Velocity in Scrum.

Question 2: Discuss the different ways of estimating Velocity in Scrum.

Ans:

Three ways to estimate velocity:

- **Use a Proxy Project**
  - **Take a similar recent project to estimate the size of items in Product Backlog and the velocity.**
- **Best Case / Worst Case**
  - **Simulate the number of sprints required based on best and worst case scenario to derive the estimated velocity.**
- **Simulate three sprints (capacity-based sprint planning)**
  - **Simulate 2-3 sprints to understand the realistic capacity of the team so as to derive the estimated velocity.**

Question 3: What are the characteristics of a good product backlog?

Question 3: What are the characteristics of a good product backlog?

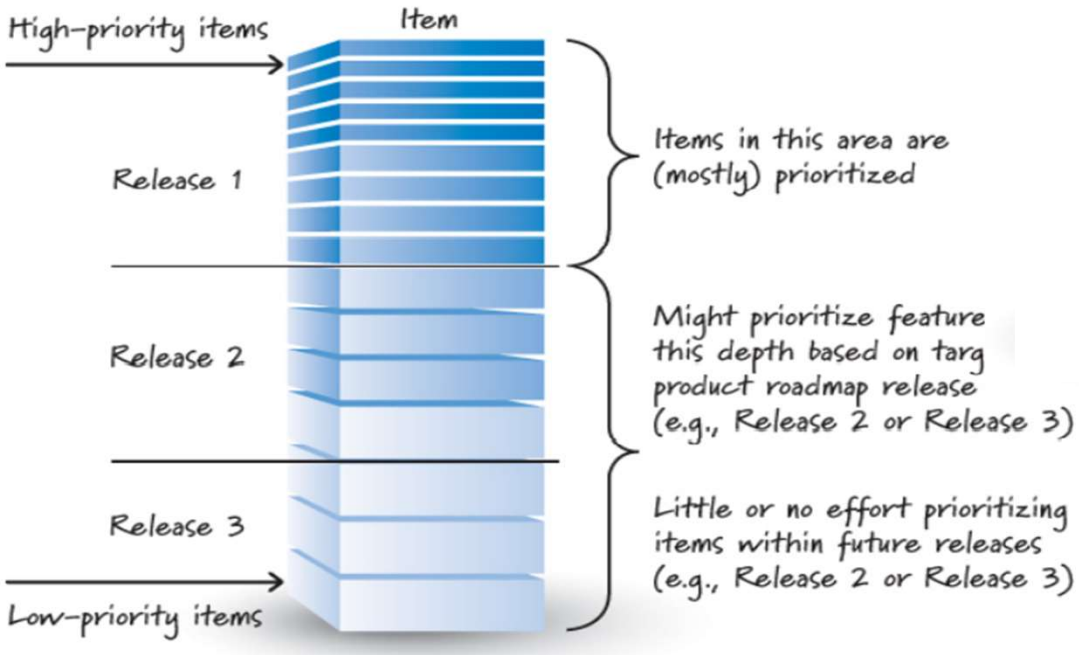
Ans:

- **Detailed appropriately**
  - **Not all items in a PB will be at the same level of detail at the same time.**
  - **PB items that we plan to work on soon should be at the top of the backlog, small in size and very detail so that they can be work on in the near sprint. BPIs that will be worked on later can be larger and less detail.**
- **Emergent**
  - **The product backlog is never complete or frozen. Instead, it is continuously updated based on a stream of economically valuable information that is constantly arriving.**
- **Estimated**
  - **Each PBI has a size estimate corresponding to the effort required to develop the item. Product owner uses these estimates as one of several inputs to help determine a PBI's priority (and therefore position) in the PB.**

Question 3: What are the characteristics of a good product backlog?

Ans: (cont'd)

- **Prioritized**
  - **Useful to prioritize the near-term items that are destined for the next few Sprints.**



Question 4: What does the INVEST mnemonic of Agile software development stand for?

Question 4: What does the INVEST mnemonic of Agile software development stand for?

Ans:

I	**Independent** The requirement is still meaningful on its own. This allows for user stories to be freely re-arranged.
N	**Negotiable** User stories should also be general enough for the development team and client to work around their implementation. They should capture the essence of what is desired, while remembering that requirements could change.
V	**Valuable** User stories should bring value to the client as indicated in the "so that <value>" clause (of the user story).
E	**Estimable** It should be possible to estimate how much time it would take to design and implement the requirement in the user story so that it an be properly prioritized.
S	**Small** A user story should be small because it is meant to be developed in a short time period.
T	**Testable** User stories should be verifiable against a set of criteria in order to determine if it is "done", meaning that the user story has accomplished what it set out to do. This is usually accomplished with acceptance tests.

Question 5: Under Scrum methodology, a Product Increment is a piece of software that is both complete and potentially shippable. Why is it important to have a working software delivered at the end of each Sprint?

Question 5: Under Scrum methodology, a Product Increment is a piece of software that is both complete and potentially shippable. Why is it important to have a working software delivered at the end of each Sprint?

Ans:

Possible benefits:

- **Working software encourages feedback**

  A team can collect more and better feedback if it shows (or better, gives) a functioning though partial product (i.e.: system) to users than producing a documentation about the product will do. This working software is to be demonstrated to the stakeholders at the end of each Sprint during the Sprint Review Meeting.

Question 5: Under Scrum methodology, a Product Increment is a piece of software that is both complete and potentially shippable. Why is it important to have a working software delivered at the end of each Sprint?

Ans: (cont'd)

Possible benefits:

- **Working software helps a team to gauge its progress**

  One of the risks of a project is not knowing how much work remains to be done. When too much of a system is in an unfinished state, it is very difficult to know how much effort will be required to bring the system to a shippable state. By emphasizing working software in each Sprint, Scrum Team avoid this problem.

- **Working software allows the product to release early if desired**

  In today's competitive and rapidly changing world, the option to release early (even if delivering fewer features) can be very valuable for customers.

Question 6: The total size (in points) of a list of product backlog items is 64. The Scrum Team consists of Product Owner, Scrum Master and six members in the Development Team. The cost of each member in the Scrum Team for each Sprint is $2K. Eight Sprints is determined to be needed to complete the project. Calculate the velocity and cost of this project.

Question 6: The total size (in points) of a list of product backlog items is 64. The Scrum Team consists of Product Owner, Scrum Master and six members in the Development Team. The cost of each member in the Scrum Team for each Sprint is $2K. Eight Sprints is determined to be needed to complete the project. Calculate the velocity and cost of this project.

Ans:

Velocity = 64 / 8 = 8

Cost per sprint = (1 + 1 + 6) x $2K = $16K

Cost of project = $16K x 8 = $128K

Tutorial #6 – Software Architecture and Strategy Pattern

1. Grain Elevator System (GES)

   a) Refer to the conceptual model developed in Question 1 of Tutorial#4.

      Use proper architectural styles to model dependencies among boundary, control, and entity objects.

2. Grain Elevator System (GES)

   The elevator manager uses a mobile device to process shipment and allocation. This mobile device must deal with a variety of network access protocols (LAN, WiFi, Bluetooth, 3G, 4G). Furthermore, we want to be able to deal with further network protocols with minimal impacts on the application.

   a) Identify the design problem and apply appropriate design pattern to address this problem

   b) Draw a UML class diagram depicting the classes in the design pattern and explain their roles

Tutorial #7 –Observer Pattern and Factory Pattern

1. Grain Elevator System (GES)

   Once grain arrives or leaves, GES will notify processing plants. The processing plants can register their interests in specific types of grains. GES will notify the registered processing plants using the plant preferred communication means, including email, SMS, or the combination of these means.

   Note that the interests and preferred communication means of the processing plants vary greatly, and the processing plants can change their interests and preferred communication means over time. Furthermore, GES should be easily extended to support new communication means once they become available.

   a) Please identify the design problem in this feature and suggest a design pattern for addressing this problem.

   b) Illustrate your solution in a class diagram and explain the roles of each class.

2. Refer to the design pattern solution developed in Question 2 of Tutorial#6.

   a) Discuss what is missing in the solution

   b) Add a relevant design pattern to the design solution to complete the design

Tutorial #8 – Equivalence Class and Boundary Value Testing

1. Grain Elevator System

   A silo cannot accept any more grain if its temperature or humidity exceeds normal levels. Humidity ranges 0-100%, the normal humidity is 35 – 50%.
   Temperature ranges -10 – 100 C, the normal temperature is 40-60 C.

   a) Determine equivalence classes and boundary values for humidity and temperature

   b) Systematically design a set of test cases to test whether silo can accept grain or not based on its temperature and humidity.

Tutorial #9 – Control Flow Testing

1. A Java method to compute the sum of 1 .. n is shown in the following Java code below:

```
public int sum(int n, int upperbound){
   1  int result, i;
   2  result = 0;
   3 i = 0;
   4 if(n < 0) {
   5    n = -n;
   6  }
   7 while(i<n && result <= upperbound){
   8    i = i + 1;
   9    result = result + i;
   10 }
   11 if(result <= upperbound){
   12   System.out.println("The sum is " + result);
   13 } else {
   14   System.out.println("The sum is too large!");
   15 }
   16 return result;
}
```

   a) Draw the corresponding control flow graph for the `sum(int n, int upperbound)` method

   b) Calculate the Cyclomatic Complexity of the `sum(int)` method

   c) Design test cases to achieve 100% statement coverage and 100% branch coverage

   d) List the basis set of linearly independent paths, along with a test case (input parameters) and expected outcome for each of the path in the basis path set