



**LUT**  
**University**

## Course Q&A

Please ask any question (or post them in Echo360 lecture Q&A area – posted questions will be answered after the lectures latest)

## How we'll proceed today?

- I'll introduce the topic with a 60-ish minutes lecture
- Then, we'll proceed to work on analyzing the course project for the remaining time
- Lastly, groups are welcome to post their analysis on the course project assignment *before the end of day today* (for a small point reward)
- There will be more in-depth videos posted online (on UML and modeling) => remember to watch also the online-only content!

The small point rewards from interactive lectures, weekly exercises, and seminars overlap => it is enough to do one, or part from each

Weekly assignments are the main "big effort" before the course project. (10%)

# Software Engineering

# Models and Modeling

## From Requirements to Models

Antti Knutas

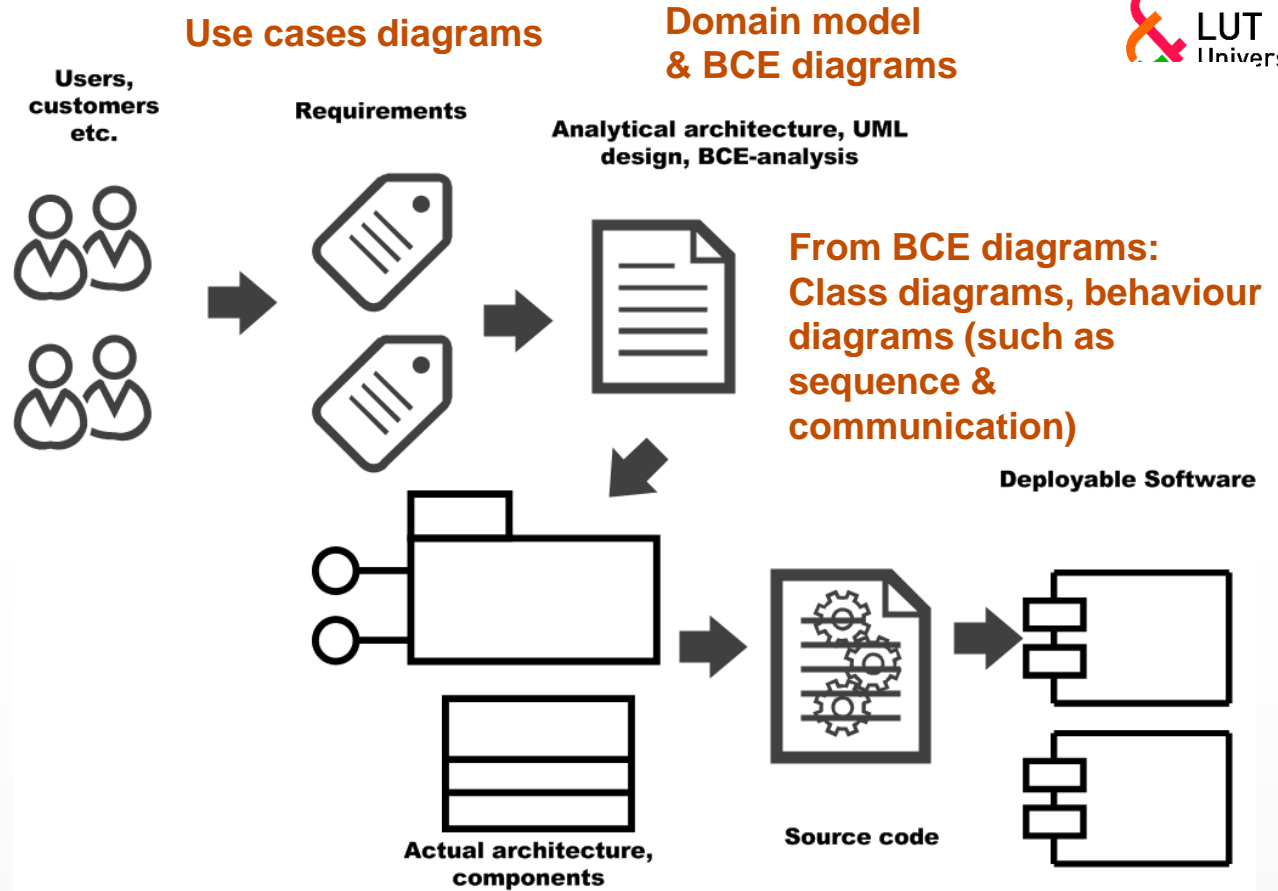
# Overview of the Process

# How to get from requirements, such as FURPS & user stories into models?

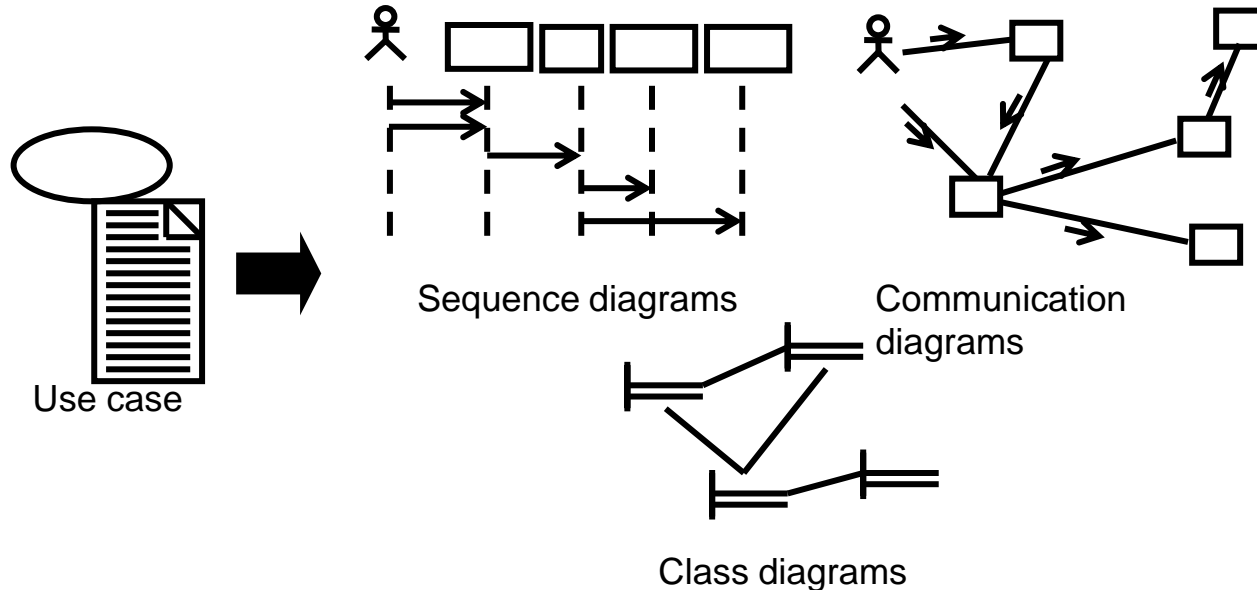
- We start converting user stories (use cases) into use case diagrams
- From use case diagrams, we create BCE models through robustness analysis
- From the analysis classes (boundaries, controllers, entities) we can describe technological objects
- At the same time we can first create a rudimentary domain model and then more in-depth class diagrams as required

Use cases => use case diagrams => BCE model =>  
Domain & class model gets built at the same time

# Context & process



# Realizing use cases (process)

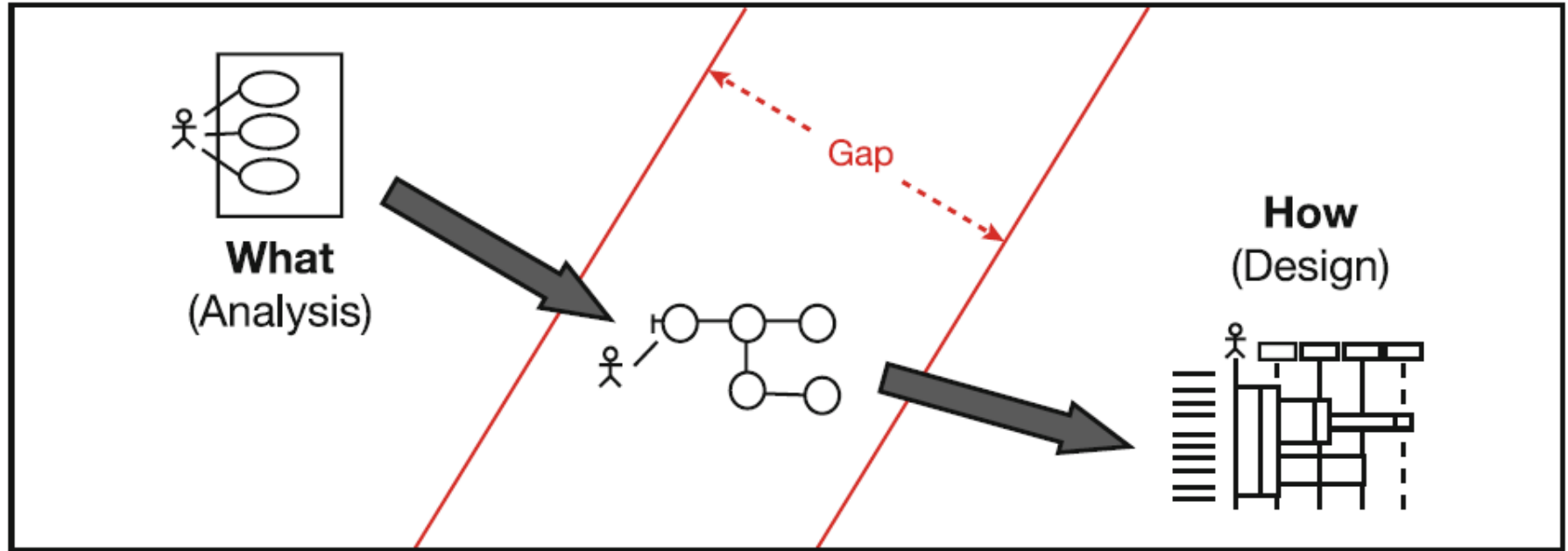


**A use case is defined as a set of classes and the interaction between objects of these classes**

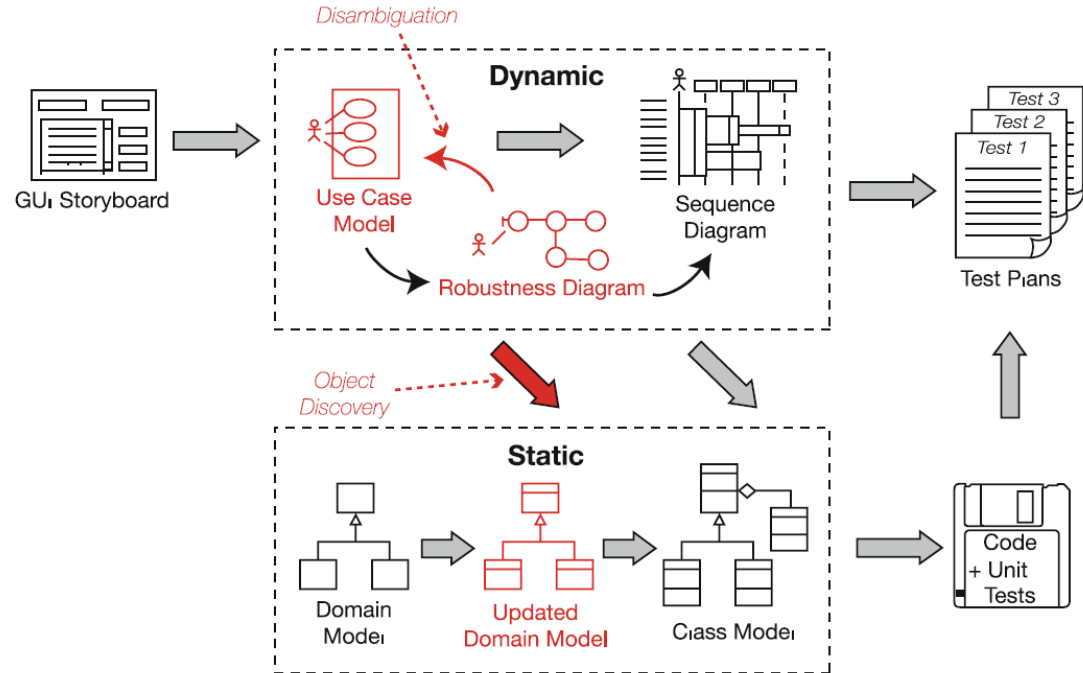
- Interaction diagrams: sequence diagrams and communication diagrams



# From What to How



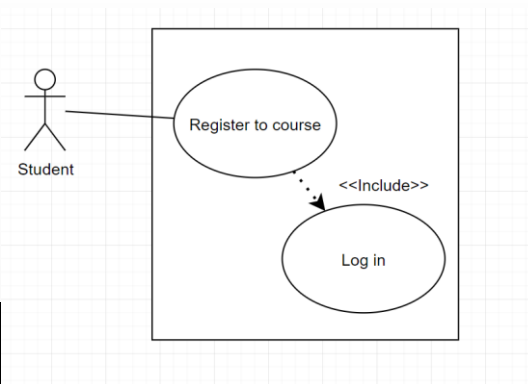
# Context & process, pt. 2



# Pt. 1 Use case and domain model

# Step 1: Use cases

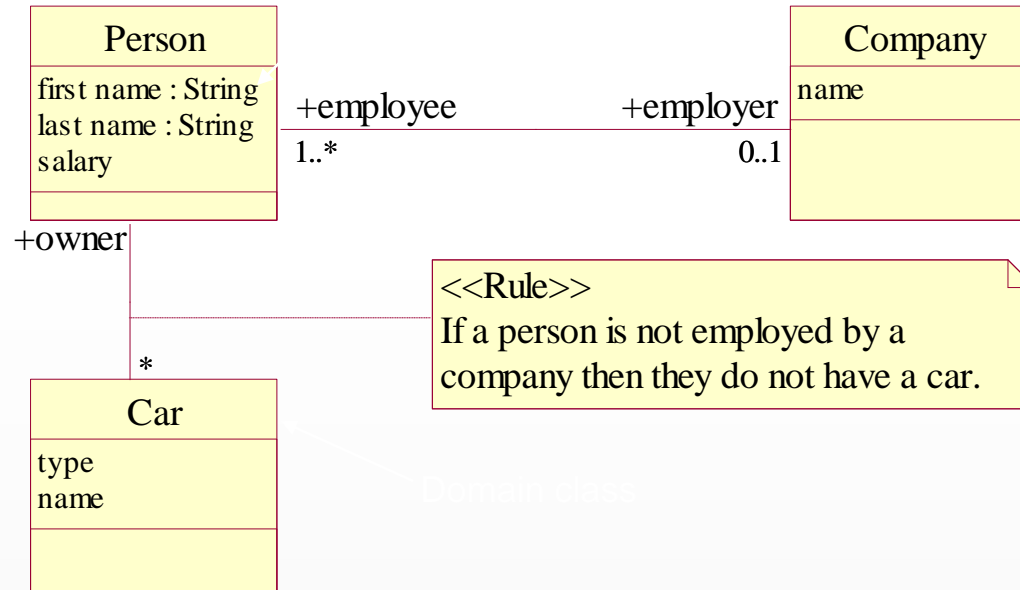
- See the earlier lecture on requirements and use case diagrams
- Gathered through a full requirements engineering process
  - (please also see our requirements engineering course!)
  - But: Analyze the domain by observing, talking to people, reading about existing documents...
  - Then: Condense these into user stories and practical use cases
  - Finally: Create use case diagrams



## Step 2: Domain model

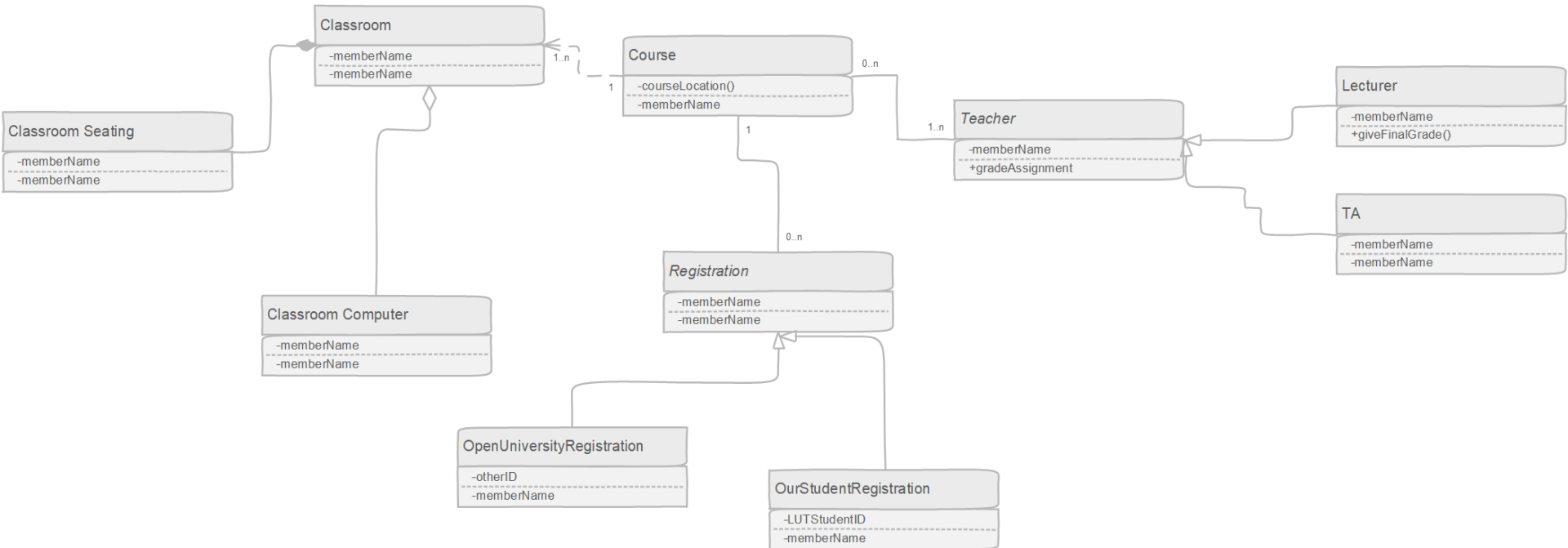
- Describes which objects and concepts exist in the system, or are related to it (in the use case / business case sense)
- Collaboration and connections between these concepts
- The following features enable us to express time invariant static business rules for a domain:
  - **Classes** – each class denotes a type of object.
  - **Attributes** – an attribute is the description of a named slot of a specified type in a domain class; each instance of the class separately holds a value.
  - **Associations** – an association is a relationship between two (or more) domain classes that describes links between their object instances. Associations can have roles, describing the multiplicity and participation of a class in the relationship.
  - **Additional rules** – complex rules that cannot be shown with symbology can be shown with attached notes.

# Simple domain model

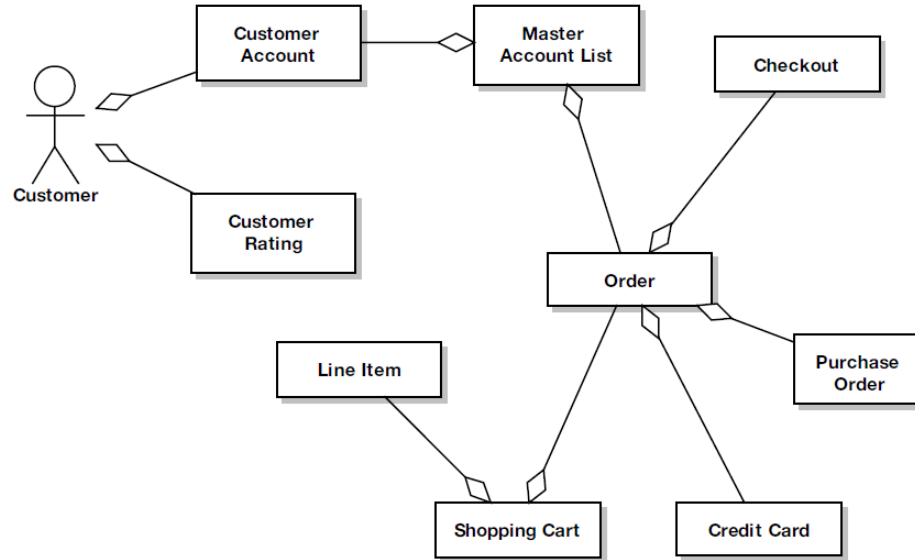


Domain class

# Example: Domain model (UML class diagram)



# Alternative example: Domain models can be even simpler





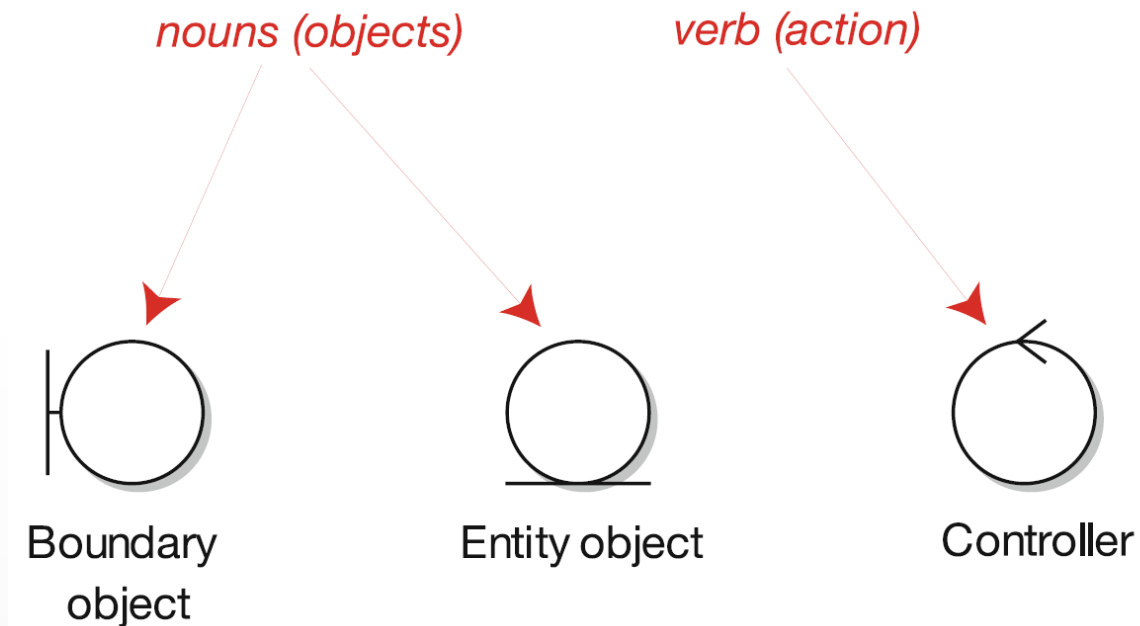
## **Pt. 2 from use case to BCE diagram (robustness analysis)**

## Step 3. Robustness analysis

- Robustness analysis: identifying a **first-guess set of objects for each use case**
- Essentially robustness diagram is an object picture of an use case
- Robustness diagram uses objects from the **domain model**
  - Update the domain model when necessary
- Review the BCE diagram structure from previous lecture if necessary

# Robustness diagram

Just three components.



# Robustness analysis rules

- Boundaries can interact with users (actors) and controllers
- Boundaries cannot connect to other UIs (boundaries) or data (entity)
- Controllers work with all, except users (actors)
- Entities can only connect to controllers

Can connect to	Entity	Boundary	Control	Actor
Entity			X	
Boundary			X	X
Control	X	X	X	
Actor		X		

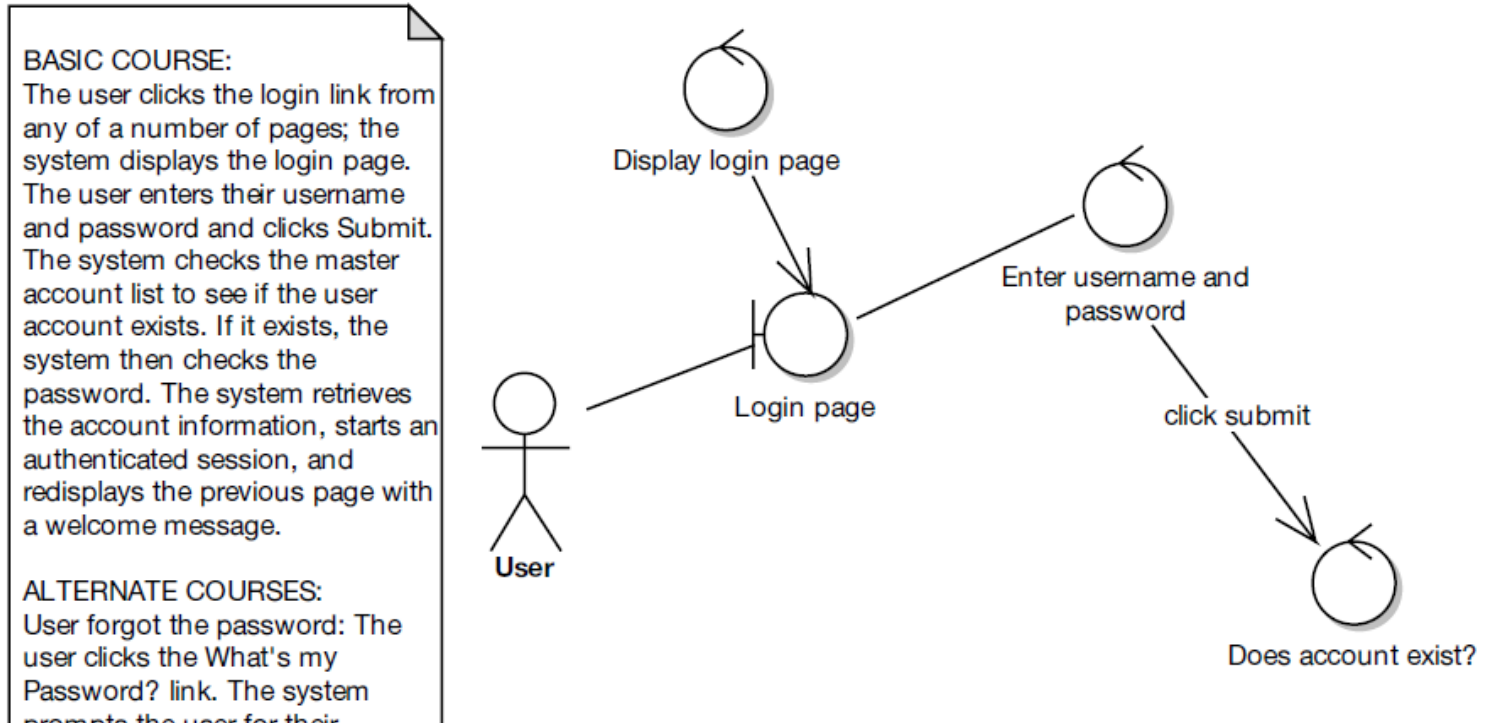
# Robustness analysis principles (selection from Rosenberg & Stephens)

1. Review use case text carefully (even paste it in)
2. Use entity classes from the domain model, update domain model if necessary
3. Rewrite use cases (clarify) if necessary
4. Make a boundary object for each user interface and name them clearly
5. Remember that controllers do not always control real world objects – they are often software logic
6. Don't worry about the direction of arrows too much
7. Robustness diagram is an initial conceptual design, not a literal design
8. Boundary and entity classes often will become object instances in a sequence diagram and controllers will become messages
9. Robustness picture builds a picture of what technological objects are present in an use case (virtual software objects, data entities and interfaces)

# Four steps for creating the analysis (Visual Paradigm)

1. You perform robustness analysis for a use case by walking through the use case text.
2. One sentence at a time, and drawing the actors, the appropriate boundary, entity objects and controllers, and the connections among the various elements of the diagram.
3. You should be able to fit the basic course and all of the alternate courses on one diagram.
4. Anyone who reviews a robustness diagram should be able to read a course of action in the use case text, trace their finger along with the associations on the diagram, and see a clear match between text and picture.

# Robustness analysis example, login to course registration system



## **Pt. 3 From BCE diagrams to Behaviour**



## Step 4. Describing behaviour with sequence diagrams

- Previous parts were about object discovery
- This step is about behaviour discovery

### Three primary goals

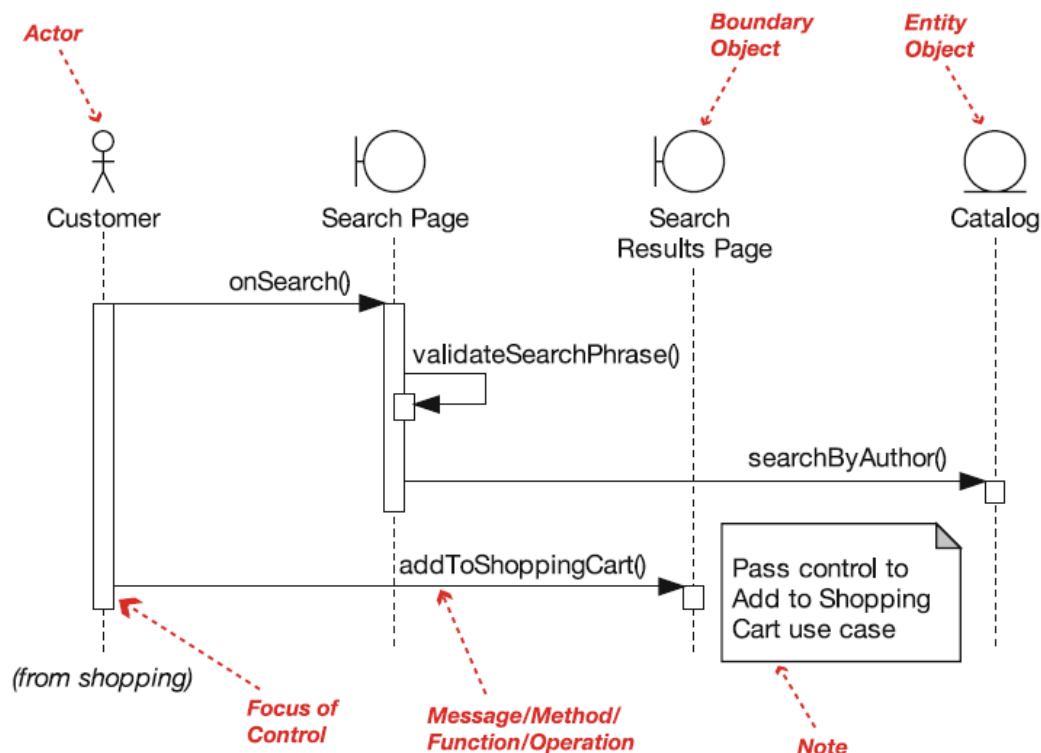
1. Allocate behaviour to classes
2. Show in detail how the classes interact over the lifetime of the use case
3. Finalize the distribution of operations among classes

# Elements of Sequence Diagram

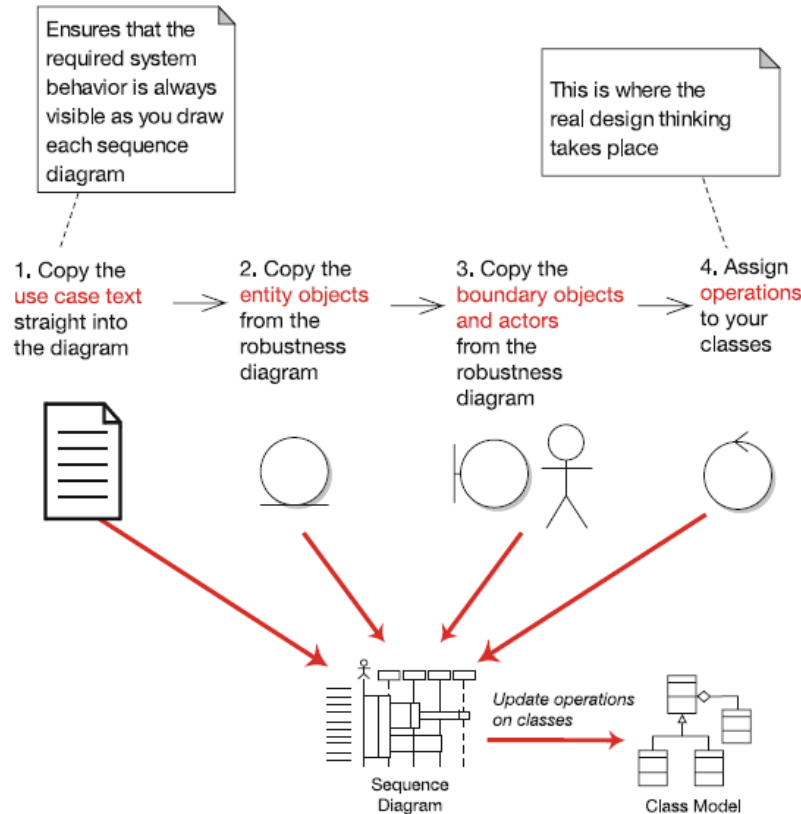
Entities are objects representing system data: Customer, Product, etc.

Boundaries are objects that interface with actors: UserInterface, DataBaseGateway, etc.

Controls are messages, mediating contact



# Step 4. Steps for building a sequence diagram



**Recap, all in one**

# Finally: The point of this process

Systematic process for translating use cases and use case text into both

- Involved objects (logical view: domain model & class diagram)
- Behaviour related to use cases (process view: sequence & communications diagrams)

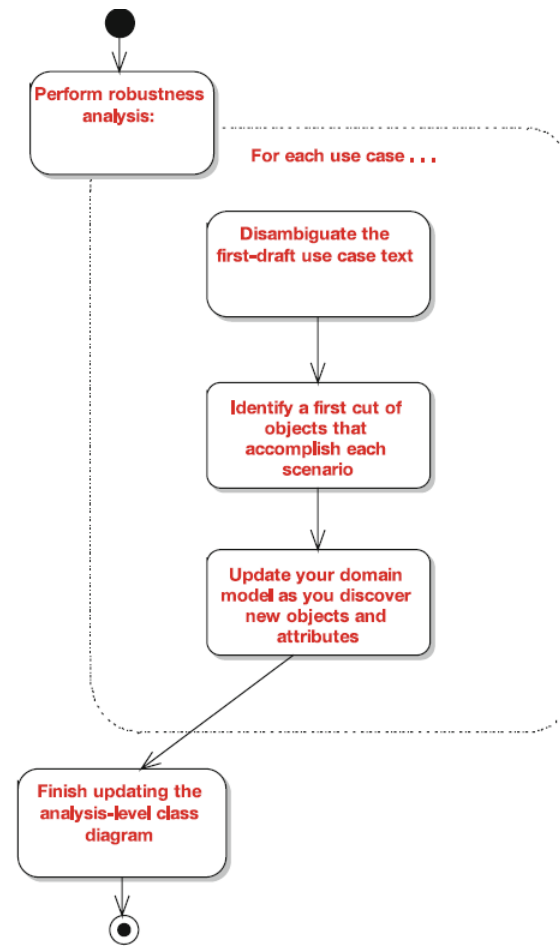
# Three step process

Zero: Define use cases and use them & related data sources (documentation, interviews, observation) as the basis for modeling

1. Create domain model
2. Create BCE models through use cases
3. Start elaborating use cases with behaviour diagrams, such as sequence diagrams, where necessary

Constantly: Update domain models & use cases and detail into class diagram where necessary

# Visually



# **THIS WEEK'S LECTURE CASE & ANALYSIS**

**WHERE THE USE OF GENERATIVE TOOLS SUCH AS LLMS  
WOULD BE MOST PRODUCTIVE**

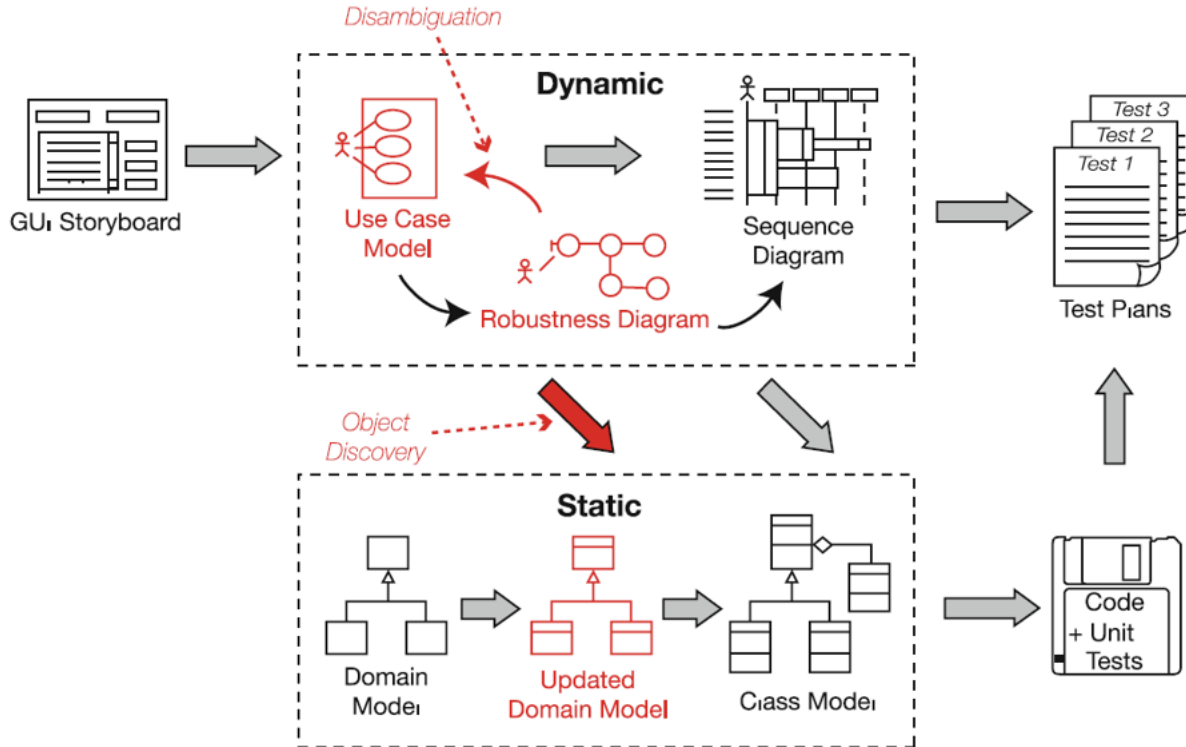


# How we'll proceed

1. I'll review today's tasks
2. Then we analyse in teams or groups
3. Lastly, we'll select one person (the bravest one!) shares the analysis outcomes of their group
4. I also invite everyone in the group to post their answer to the specific forum under lectures tab for a tiny point reward (same point category with exercises and seminars)

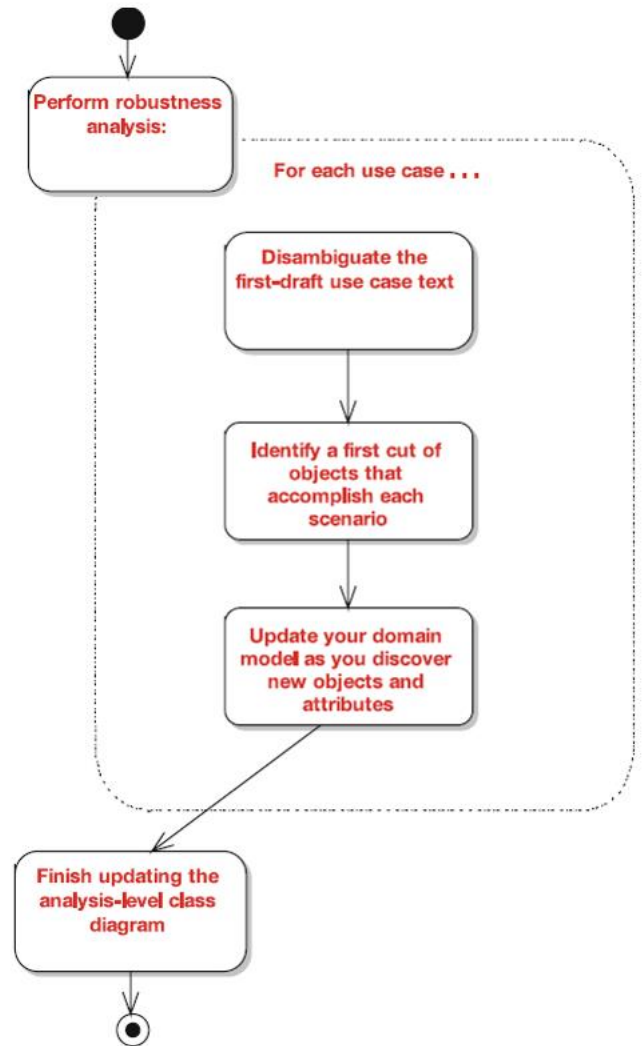
# Task 1/3

- Take a quick look at these “from requirements to models” processes by Rosenberg & Stephens



## Task 2/3

- Take a quick look at these “from requirements to models” processes by Rosenberg & Stephens



## Task 3/3

1. Think about the process of modeling software from requirements, through the process presented by Rosenberg & Stephens and covered in Topic 5 lectures.
2. Select one place where generative AI (such as LLMs) could be used to create genuine value to the customer and help speed up the engineering team's work. Take one of the pictures below and specify where. Also explain your rationale with a few sentences.

## Sharing and review (if there is time, online otherwise)

Which stages in the process were most commonly selected by groups?

How could they be aided by the use of generative tools, such as LLMs?

Let's discuss the findings of few randomly selected groups, before the lecturer weighs in his answer.

# How to continue from here? (What's the use for course project?)

UML class diagrams can be used to present conceptual models in a more formal, standardized way.