Repetition
(E2017)

Introduction to Systems Engineering
I2ISE

Topics

- State Machines
  - Sub-states + orthogonal
- Domain and Application Models
  - Domænemodel
  - Sekvensdiagrammer
  - Applikationsdiagrammer og sammenhæng
  - Hvordan finder man domain, control og boundary klasser?
- 4+1 Views
  - Logical + Deployment

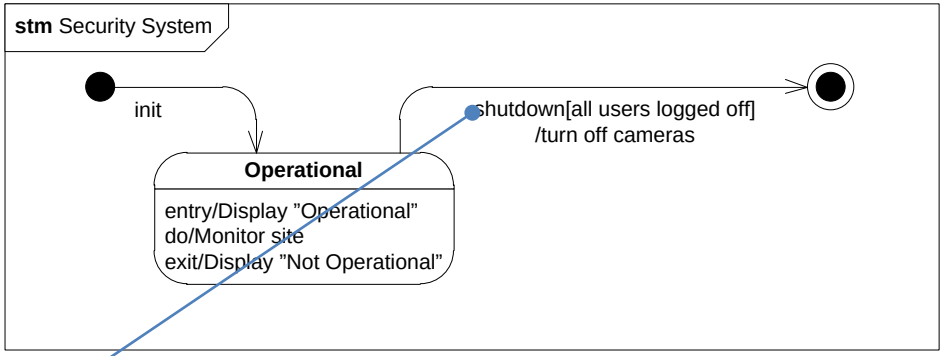
State Machines

- Sub-states + orthogonal

States and transitions

- Transitions consist of *trigger*, *guard* and *effect*: `trigger[guard]/effect`

- When `trigger` occurs, `guard` is evaluated.
  - If `guard` is true, `effect` occurs.
  - If not, `trigger` is consumed without effect



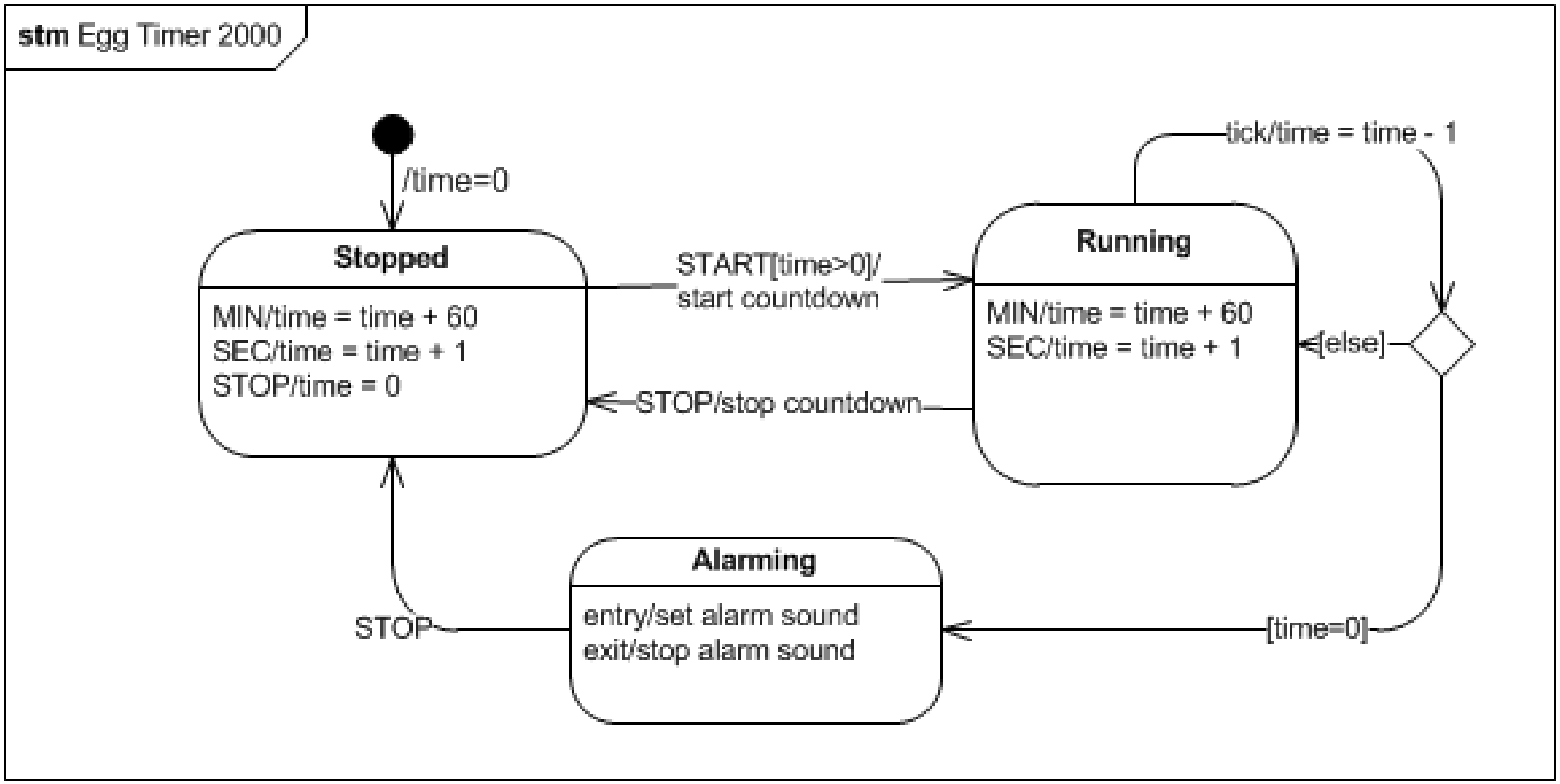
Trigger = shutdown
Guard = all users logged off
Effect = turn off cameras

What happens if some user is still logged on?

Exercise 1: Egg Timer 2000

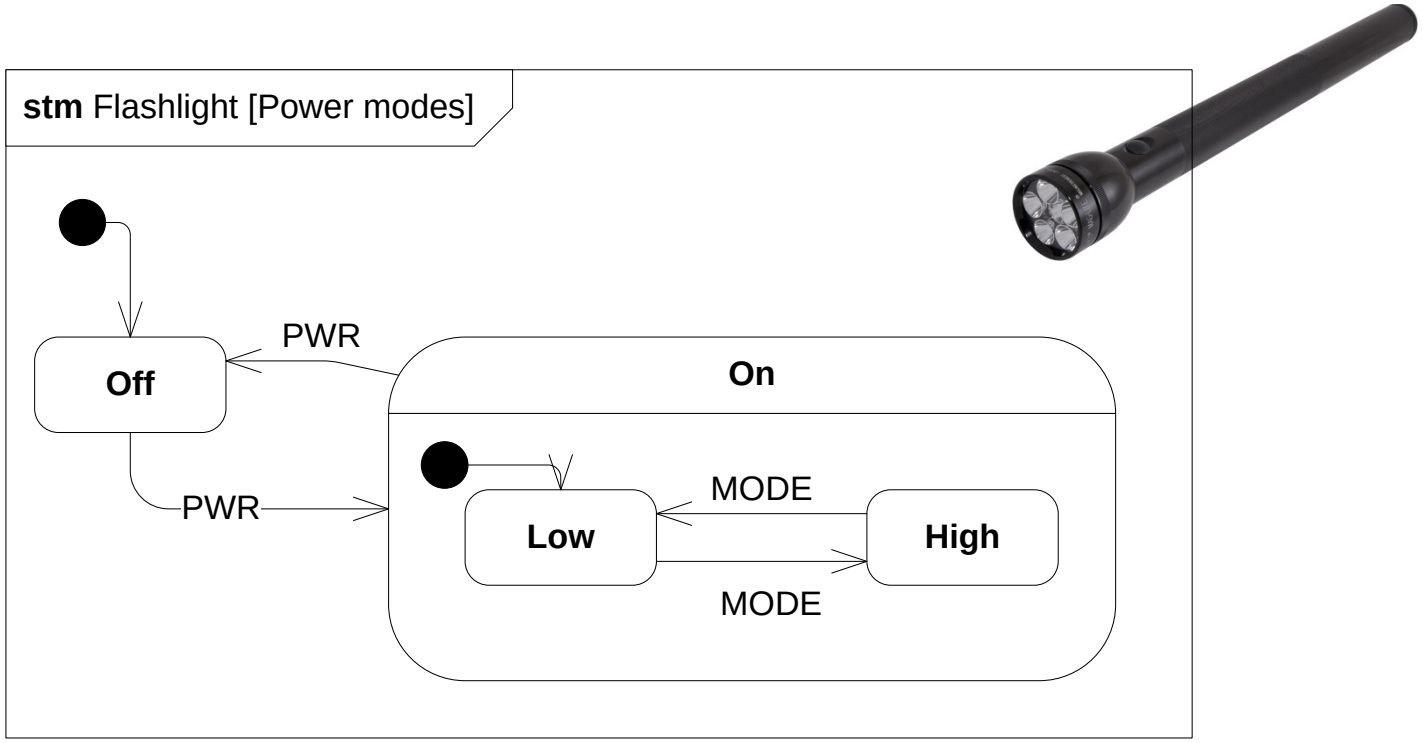
- Create a state machine diagram for an egg timer:
  - The egg timer has four buttons: **`MIN`**, **`SEC`**, **`START`**, **`STOP`**
    - **`MIN`**, **`SEC`**: Increase time by 60 seconds and 1 second, respectively
    - **`START`**: Start countdown
    - **`STOP`**: If running: Stop countdown. If stopped: Clear time. If alarming: Stop alarm
  - Each second, there must be a *tick* event. If ET2000 is running, the remaining number of seconds shall be counted down by 1. If the timer expires, an alarm shall sound.
  - Ignore display updates etc. and concentrate on the setting, counting down and alarming.

Egg Timer 2000 – state machine



States and substates (nested states)

- A state may have substates.
- Example: Flashlight with **PWR** and **MODE** buttons

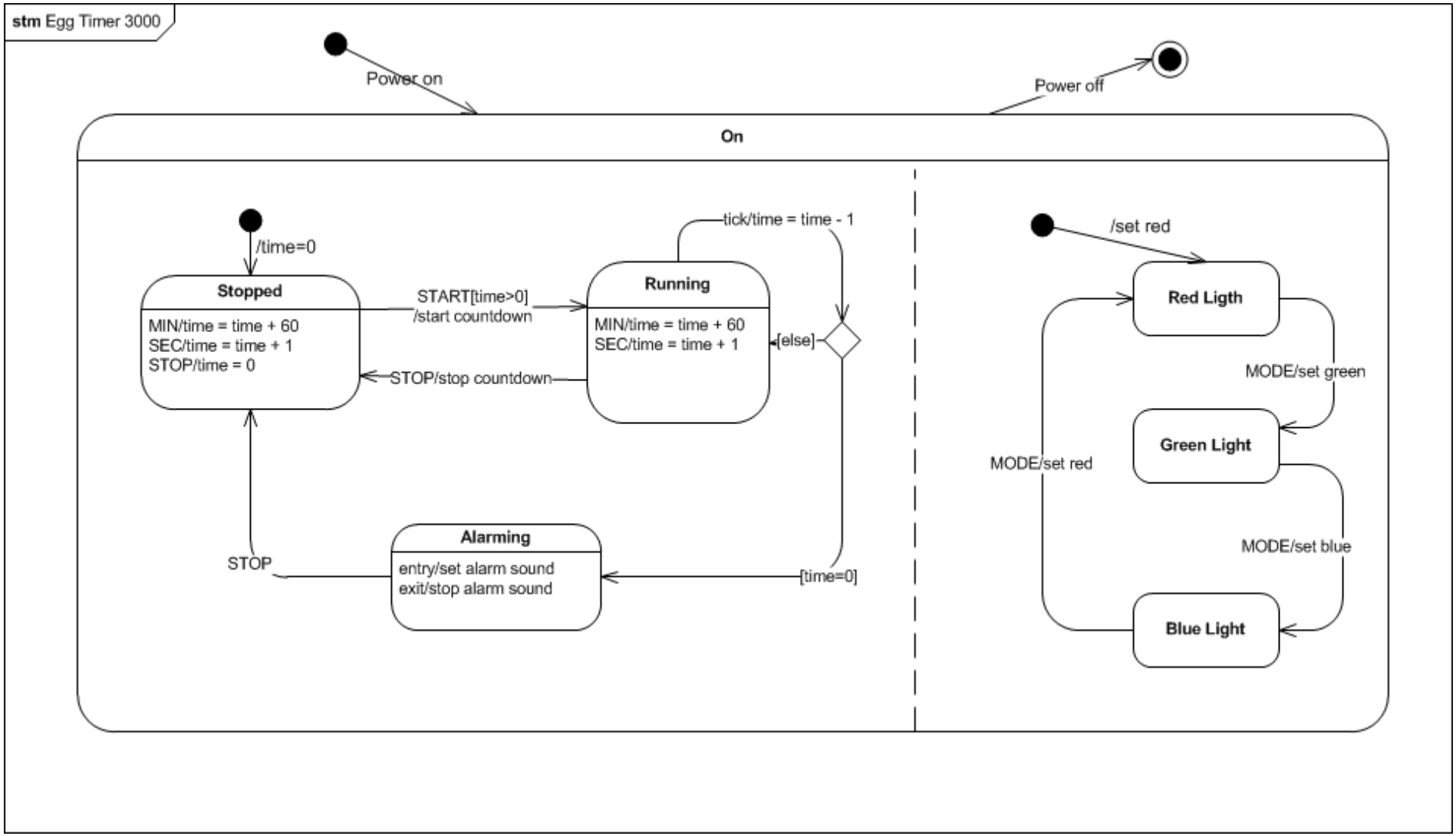


Exercise 2 : Pimped Egg Timer 3000

- PET3000 is like ET2000, but the display can be backlit with either red, green or blue light. This is controlled with the **`MODE`** button which toggles the light.

- Draw it's state machine diagram

Pimped Egg Timer 3000 state machine (multiple regions)



OPGAVE 5 (25 %):
Kontrol klassen "SikkerPassage" i Figur 2 kan realiseres med en tilstandsmaskine, som har følgende fem tilstande:

- Bro lukket
- Bro åbnes
- Bro åben og tom
- Bro åben med skib
- Bro lukkes

- Se BB Eksamenssæt "BSC"

Tilstandsmaskine reagerer på følgende hændelser:

• Åbn bro	kommando fra klassen "UserInterface"
• Luk bro	kommando fra klassen "UserInterface"
• Bro åbnet	status besked fra klassen "VisionSystem"
• Bro lukket	status besked fra klassen "VisionSystem"
• Skib detekteret	status besked fra klassen "VisionSystem"
• Intet skib detekteret	status besked fra klassen "VisionSystem"

Tilstandsmaskinen kan starte følgende aktiviteter.

• Åbn	metode i klassen "Broklapper"
• Luk	metode i klassen "Broklapper"
• Vis besked	metode i klassen "UserInterface"
• Gem besked	metode i klassen "Log-fil"

Lav et State Machine Diagram (STM) for tilstandsmaskinen beskrevet ovenfor.

Domain and Application Models

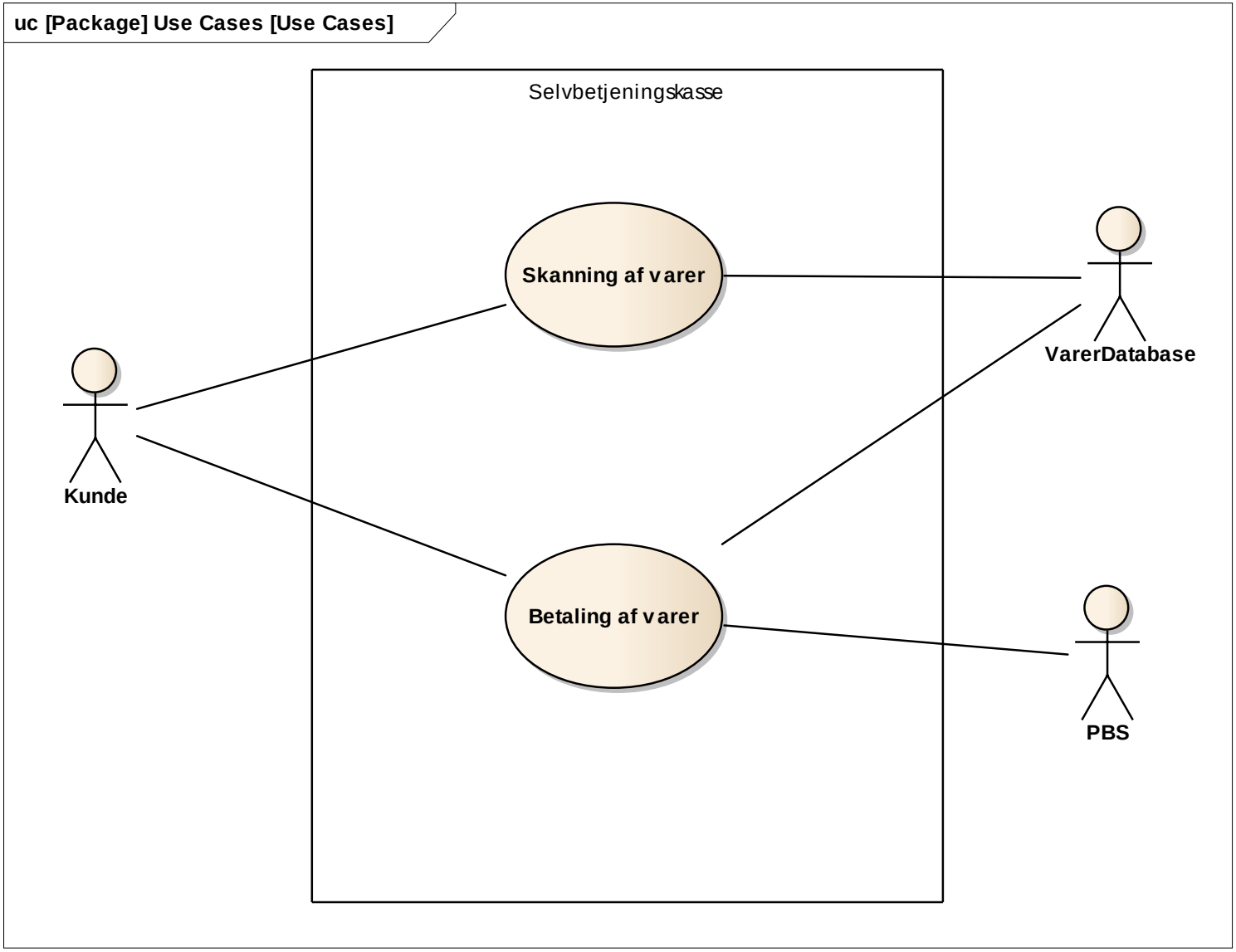
- Domænemodel – retning på pile
- Sekvensdiagrammer – aktiveringer og pile
- Applikationsdiagrammer og sammenhæng
- Hvordan finder man ”domain, control og boundary” klasser?

Beskriv kort formålet med at udarbejde en domænemodel og hvad den beskriver.

Målet med domænemodellen er at skabe et strukturelt overblik over et givent problem/opgaves domæne. Konceptuelle klasser identificeres, som beskrives i et klassediagram med beskrivelse af relationer mellem klasserne med attributter. Modellen beskriver klasser og begreber, som er kendt i problemdomænet. Disse begreber findes ud fra systemspecifikationen med en analyse af navneord, koncepter og begreber. Modellen beskriver ikke implementeringen, som f.eks. software klasser eller hardware komponenter. Domænemodellen danner grundlag for en efterfølgende design processen.

Selvbetjeningskasse

Selvbetjeningskasse – Use Cases



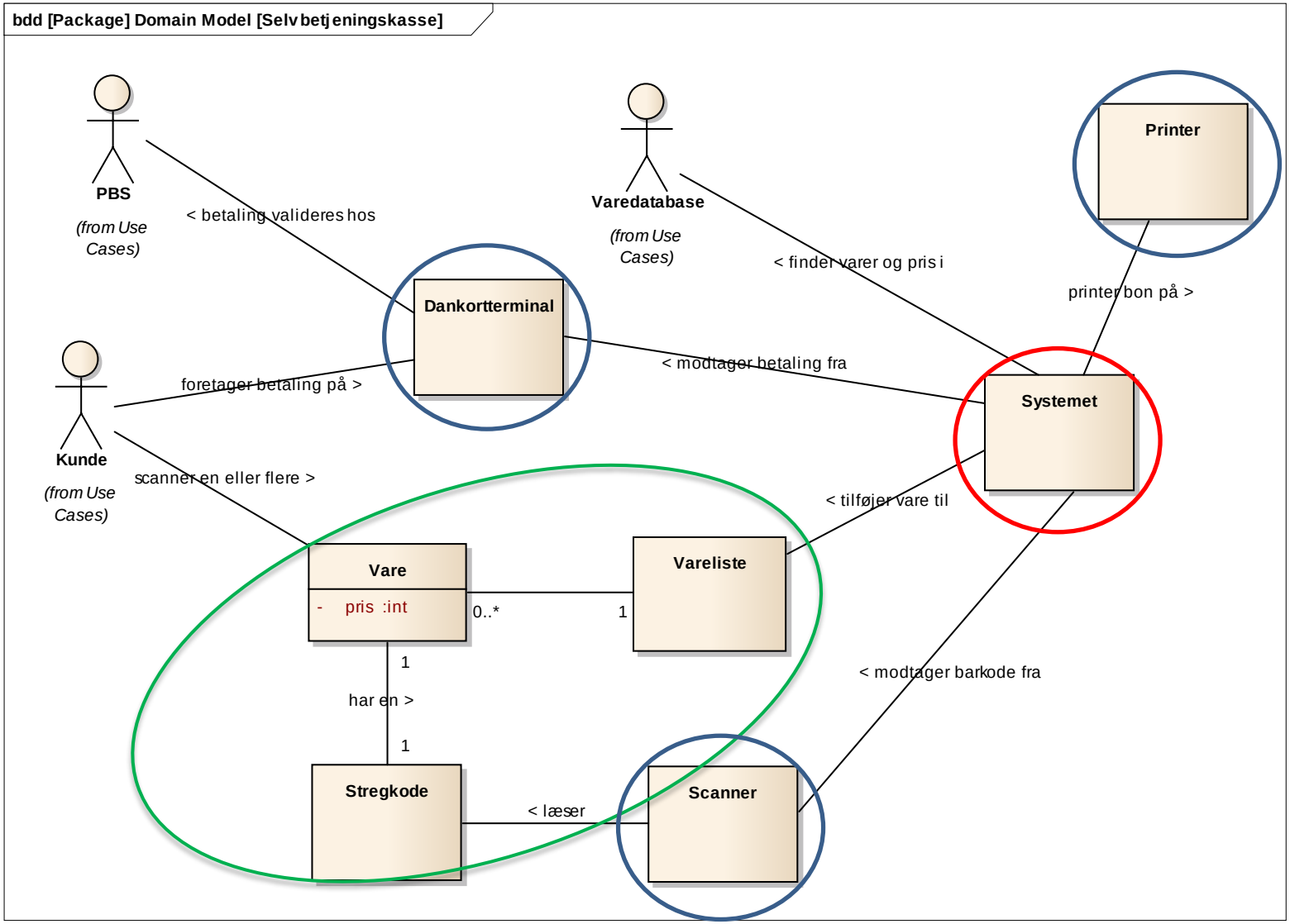
Scanning af Vare (Hovedscenarie)

1. Selvbetjeningskassen anmoder kunden om at skanne vare
2. Kunden placerer vare foran skanner
3. Systemet skanner varens stregkode
4. Systemet finder varens pris i varedatabasen
5. Vare med pris tilføjes til en vareliste
6. Kunden lægger vare i pose på bordet ved siden af skanner
7. Punkterne 1-6 gentages indtil alle varer er skannet
8. Kunden vælger afslut

Betaling af Vare (Hovedscenarie)

1. Kunden vælger betal med dankort på beløbet
2. Kunden indsætter kort i dankortterminalen
3. System viser det totale beløb og anmoder om pinkode
4. Kunden indtaster pinkode
5. Kort og pinkode valideres mod PBS
6. Printer udskriver bon med vareliste

Domain model



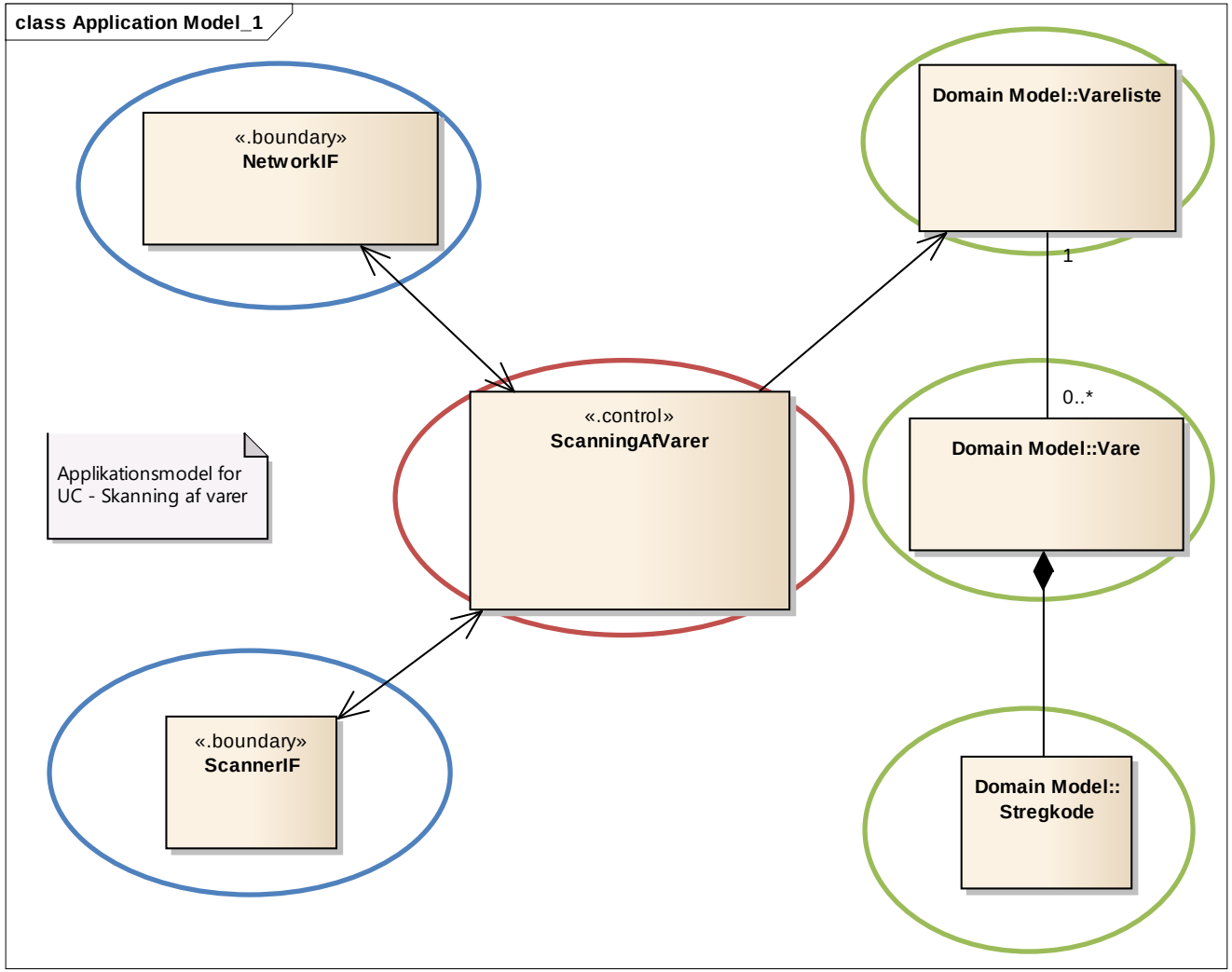
Application Model Classes Purpose

- Domain classes
  - To capture and handle information in the problem domain
- Control classes
  - To realize scenarios specified by Use Cases (Named after the Use Case)
- Boundary classes
  - To interface and communicate with Actors or Devices

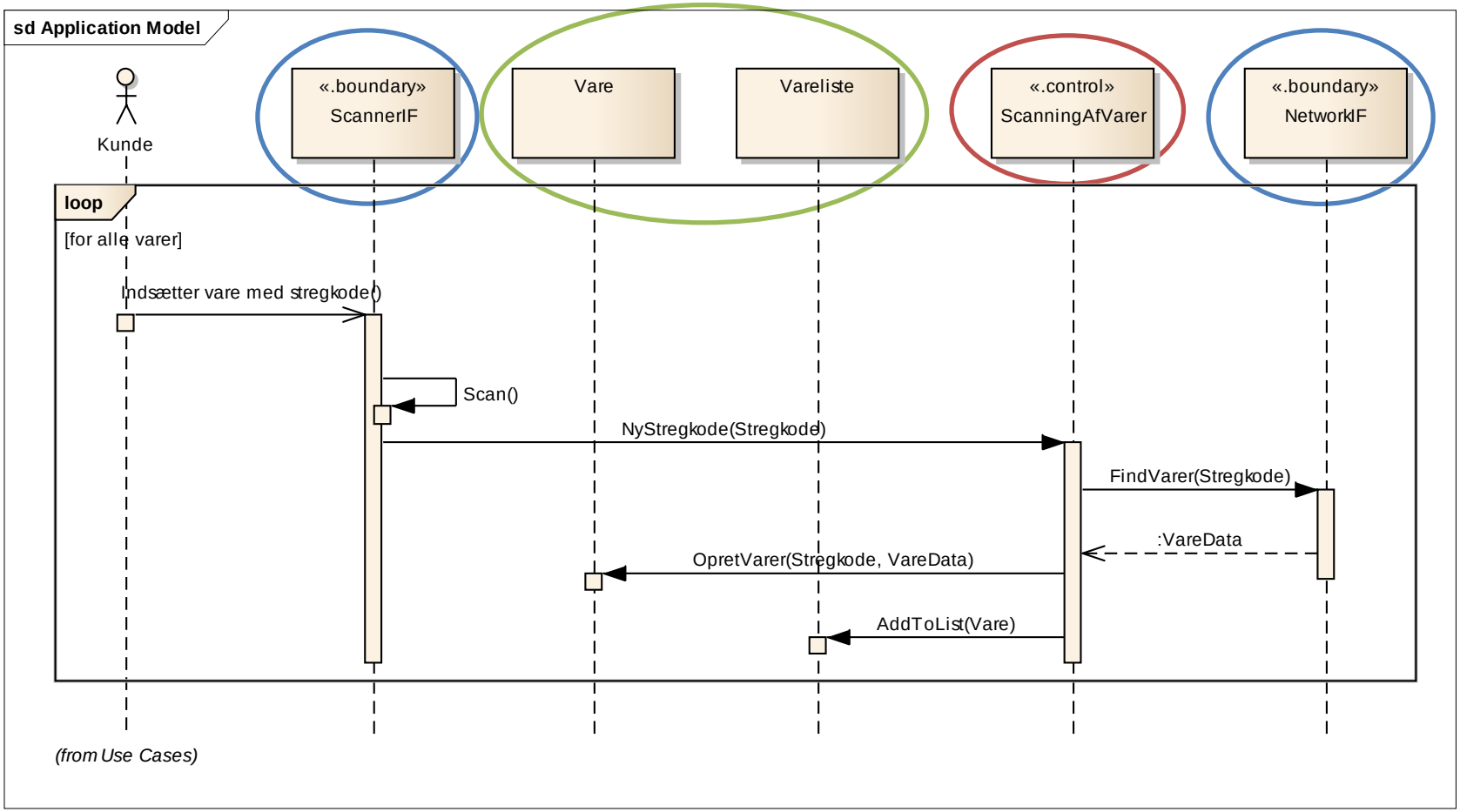
Beskriv kort de tre typer af klasser der indgår i en applikationsmodel?

I applikationsmodellen indgår domæne- (domain), grænseflade- (boundary) og kontrol- (control) klasser. Domæne klasserne findes ud fra domænemodellen. Kontrolklasserne repræsenterer funktionalitet beskrevet i Use Case diagrammet. For hver Use Case oprettes en tilsvarende kontrolklasse, med samme navn som Use Casen. For hver aktør i Use Case diagrammet oprettes en grænsefladeklasse. Målet er en lagdelt arkitektur, hvor kontrol, domæne og grænseflader blive adskilt.

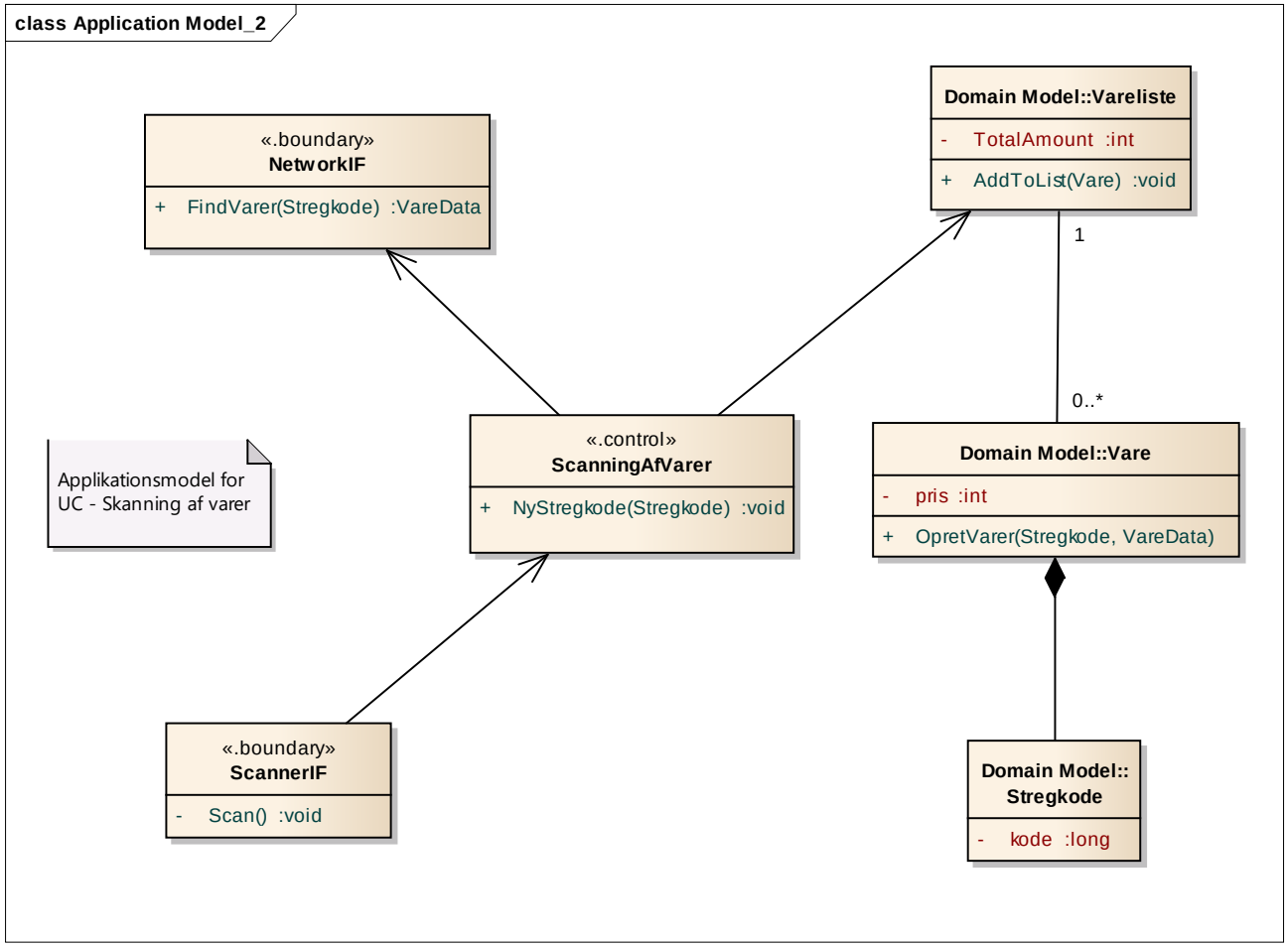
Class Diagram (UC – Scanning af varer)



Sequence Diagram



Class Diagram with Methods

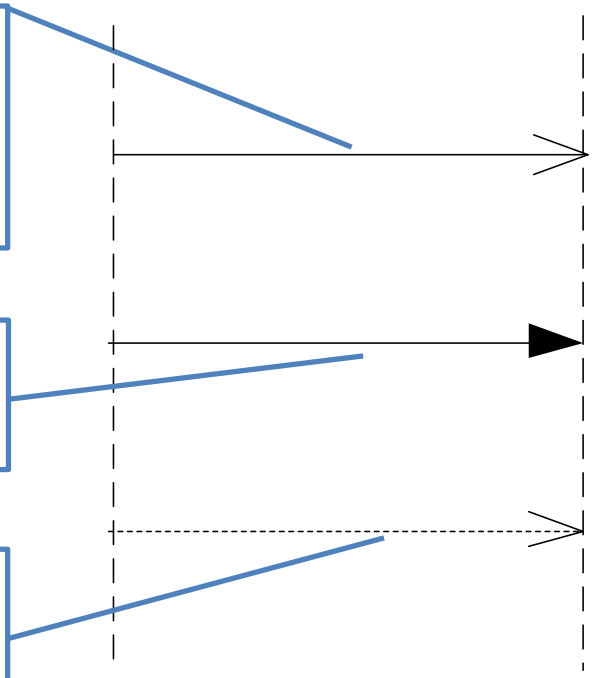


Sequence diagrams and Arrows

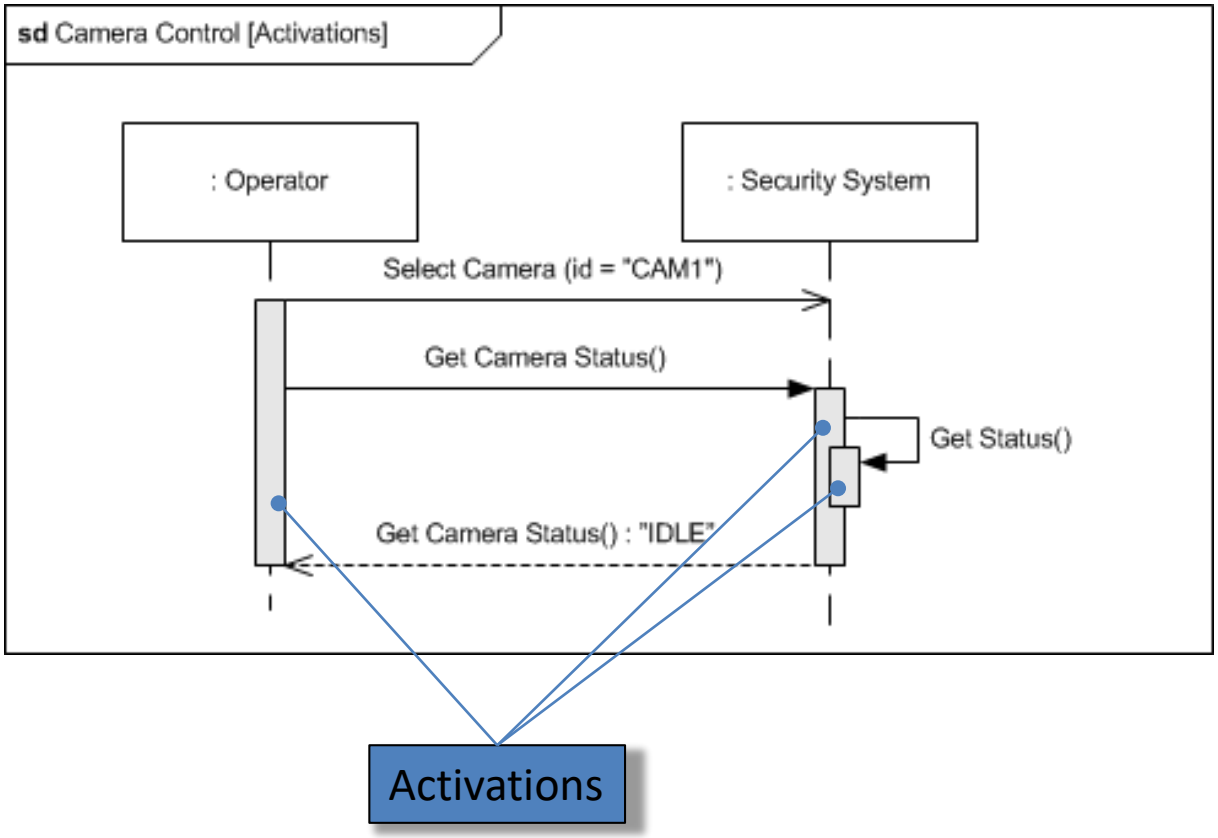
An open arrowhead means an **asynchronous message**. (Sender don't wait for a reply. Sender continues to do other things while message is being received and handled)

A closed arrowhead means a **synchronous message** (Sender waits for completion and reply to message)

An open arrowhead on a dashed line shows a **reply message**. Use only as a reply to a synchronous message.



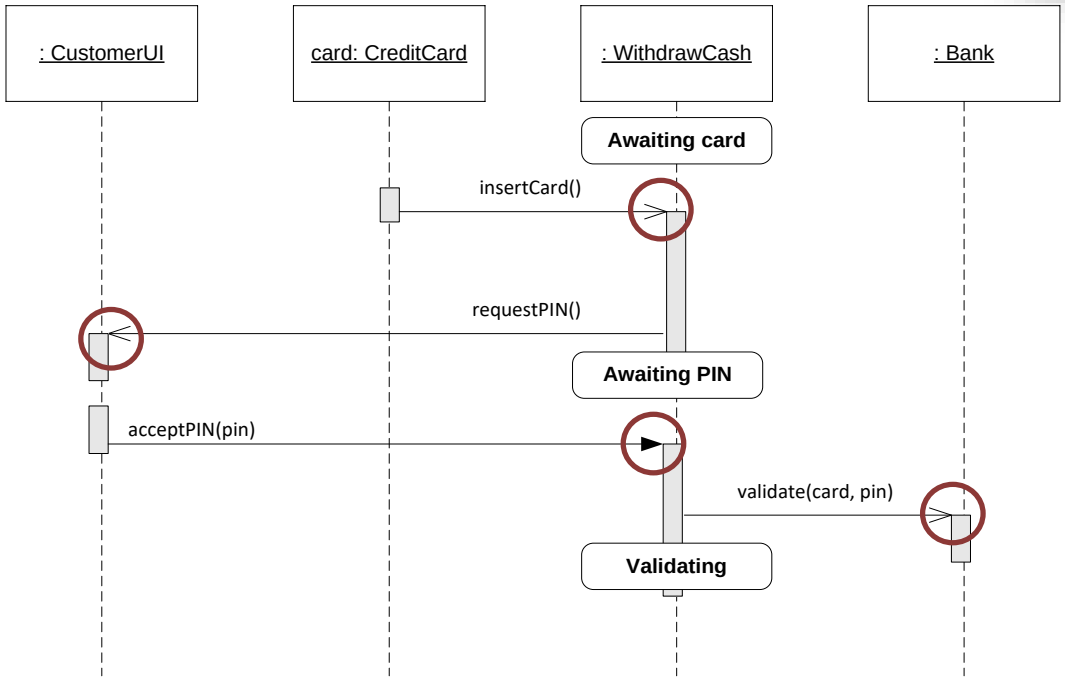
Activations

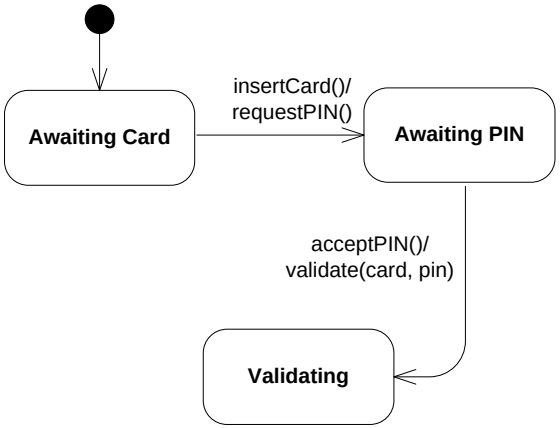


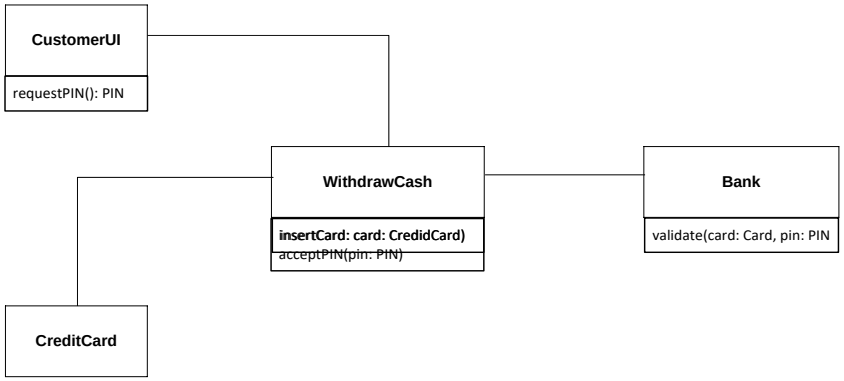
Steps 2.1-2.4 for UC *Withdraw Money*

Main scenario:

1. Customer inserts credit card in System
2. System requests Customer's PIN code
3. Customer enters PIN code
4. System validates card info and PIN code with Bank





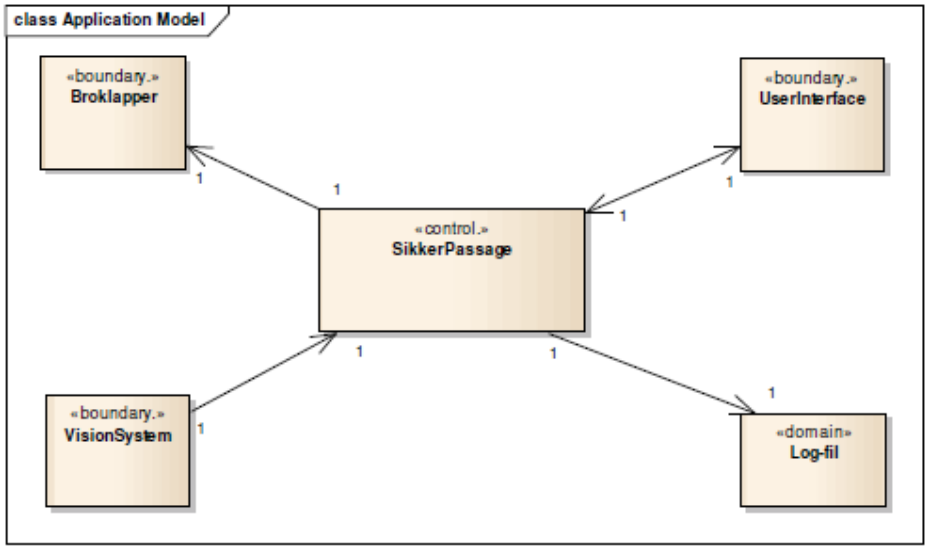


OPGAVE 2 (20 %):
Udarbejd en domænemodel for *Bridge Control System'et* med udgangspunkt i *UC1: Sikker passage for skib* som givet ovenfor. Undtagelserne (*extensions*) skal medtages i modellen.

- Find konceptuelle klasser
- Tegn et klassediagram med associationer
- Angiv associationernes navne og multipliciteter
- Find væsentlige attributter og angiv dem på klassediagrammet

Resultatet er domænemodellen (et klassediagram).

OPGAVE 3 (25 %):
Software skal udvikles til blokken *Controller* i Figur 1, som er den computer, der fortager den centrale styring af broen. Nedenfor er vist et klassediagram for applikationsmodellen med kontrol, domæne- og grænsefladeklasser for hovedscenariet i *UC1: Sikker passage for skib.*



Figur 2 Klassediagram for applikationsmodel for *UC1: Sikker passage for skib*

Tegn et sekvensdiagram, der viser kommunikationen mellem klasserne for "Main Scenario" i UC1 med brug af ovenstående applikationsmodel. Undtagelserne beskrevet i UC1 skal ikke medtages. Find selv på passende navne til funktioner og metoder i klasserne.

OPGAVE 4 (10 %):
Opdatér klassediagrammet i Figur 2 med metoder og evt. attributter, som kan findes ud fra sekvensdiagrammet i løsning af opgave 3.

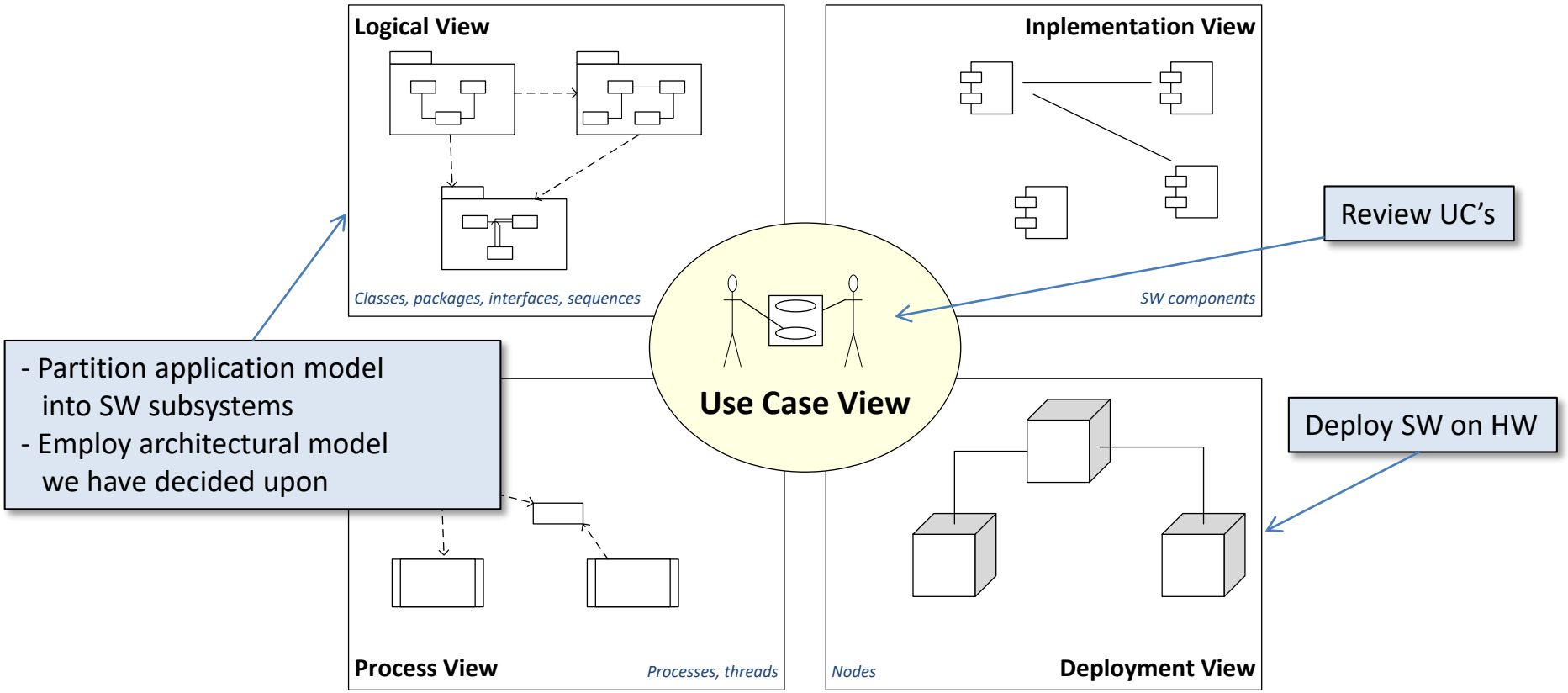
- Se BB
  Eksamenssæt "BSC"
  Eksamenssæt "E2014"

4+1 Views

Logical + deployment

SW architectural design

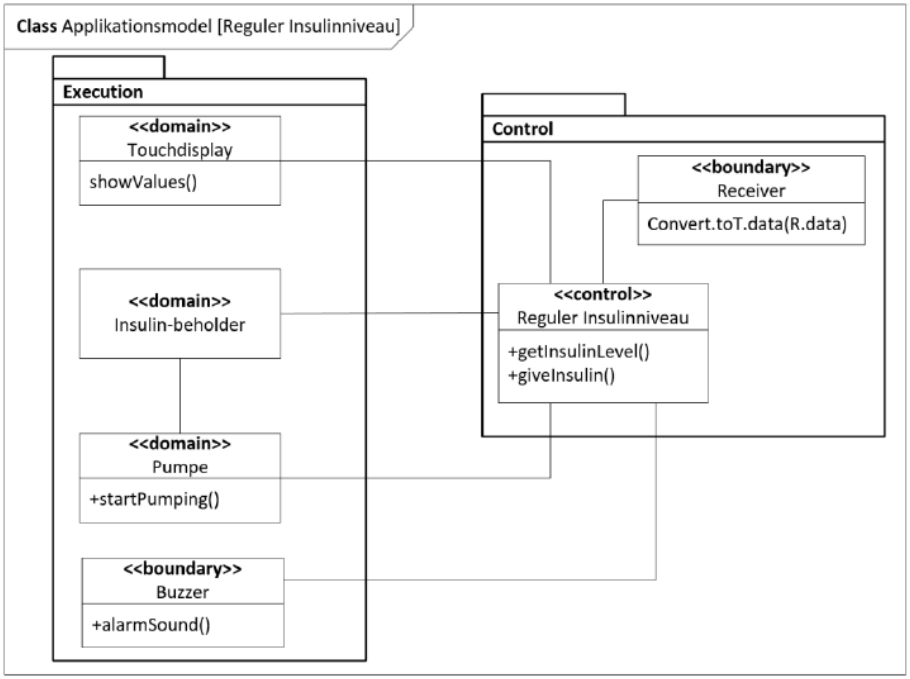
- To support the architectural design activities one can use the ”4+1 Architectural View Model” (Phillipe Krutchen, 1995)
- Describes software architecture using five concurrent views



Logical View

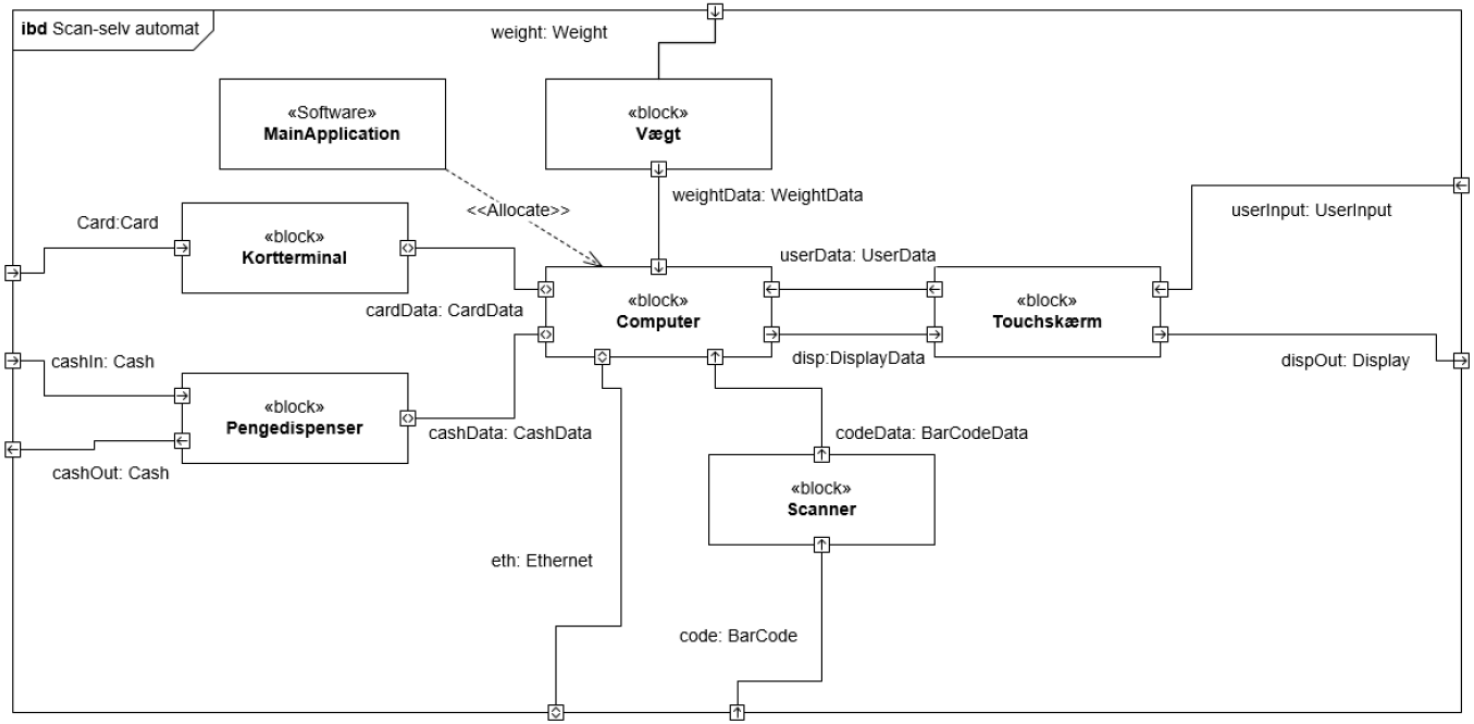
- The view shows the software components of the system as well their interaction and relationship
- **The application model** in terms of UML class, sequence and state diagrams
- **Packages can be used to divide classes into groups of related topics**
- Use cases is driving this view in terms of functional requirements.
- Concerns what the system should provide in terms of service to its users.

Opgave 3.2 – Logical view



Deployment View

- The deployment view concerns understanding the **physical distribution of the software across a set of processing nodes** (computers or micro controllers)
- By use of BDD or IBD diagrams this view illustrates where software is deployed in the system
- Use the **stereotype <<allocate>> to show where software application models** are to be developed and executed



Figur 10 - IBD med software-deployment