System Application Models
Part 2
Systems with subsystems

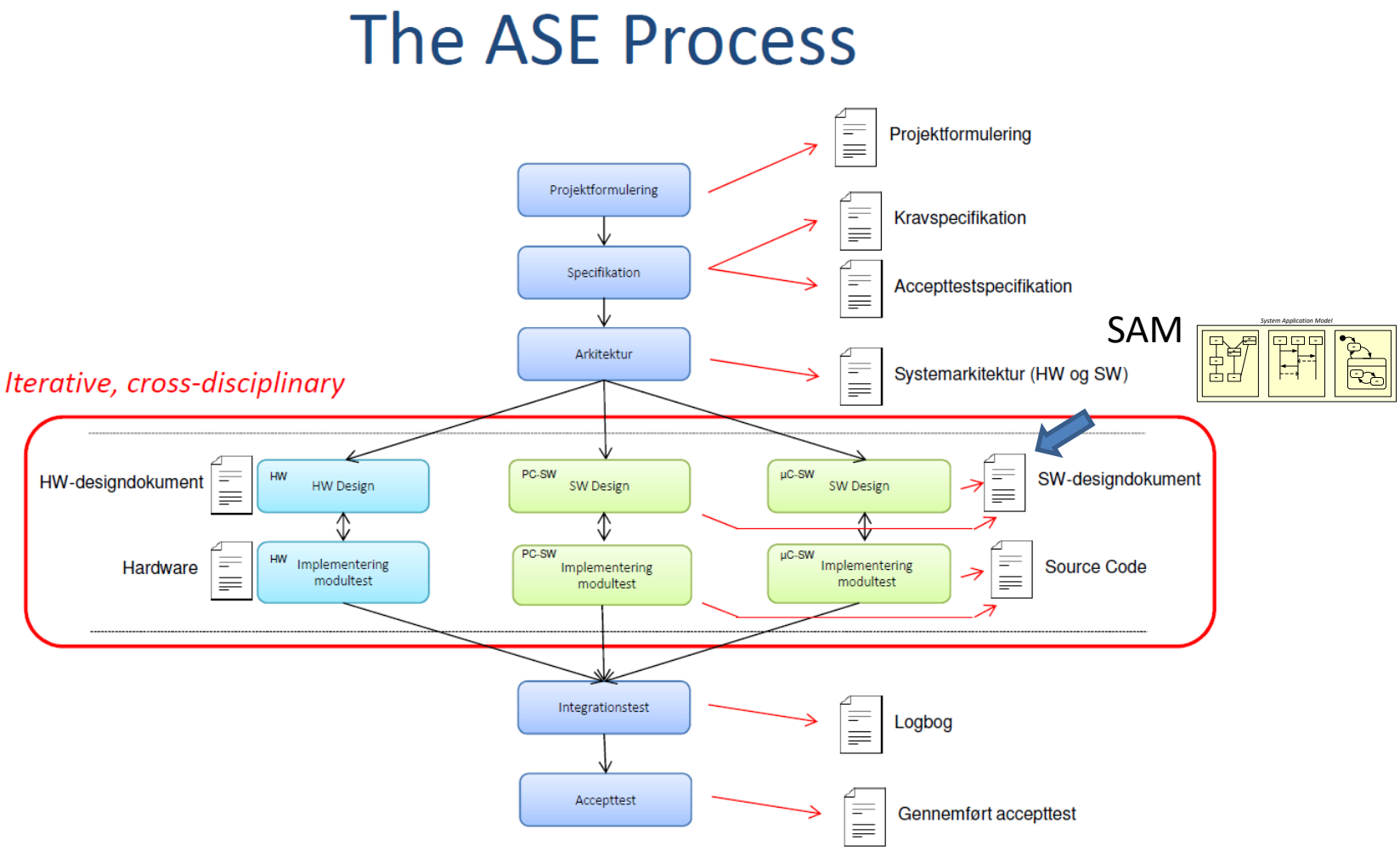
I2ISE

Application models – bridging the gap

- A lot of time has been spent on writing use cases and making domain models. Today, we cash in!
- We will use the UCs to bridge the gap between *what* the system must do (requirements) and *how* it must be done (design)
- In other words, we will use the UC's as *design drivers*
- So it would seem that :

UCs are important!

The SAM's place in the artefacts



Identify the what, the what and the what?!?

- Our application model consists of three different types of classes: *Boundary*, *domain*, and *controller* classes

- *Boundary* classes represent UC *actors*
  - They are the actors' interface(s) to the system (UI, protocol, …)
  - They *present* the system but contain no business logic.
  - Stereotyped «boundary»

- *Domain* classes represent the system's *domain*
  - Memory, domain-specific knowledge, configuration, etc.
  - 1 or more, shared between several UCs

Identify the what, the what and the what?!?

- Our application model consists of three different types of classes: *Boundary*, *domain*, and *controller* classes

- The *Controller* class holds the UC business logic
  - It ”executes” the use case by interacting with the boundary and domain classes.
  - Named after the UC
  - Typically 1 per UC or 1 shared among a couple of UCs
  - Optionally stereotyped «`control`» or «`controller`»

The System Application Model – Step 1

- The application model is constructed incrementally in units of use cases. So, apparently,

UCs are important!

Step 1.1: Select the next fully-dressed UC's to design for (how?)

Step 1.2 a: Identify all actors involved in the UC → *Boundary* classes.

Step 1.2 b: **If you have a BDD/IBD, identifying the actual hardware interfaces: split the *Boundary classes* into relevant hardware interface classes.**

So are BDD/IBD!

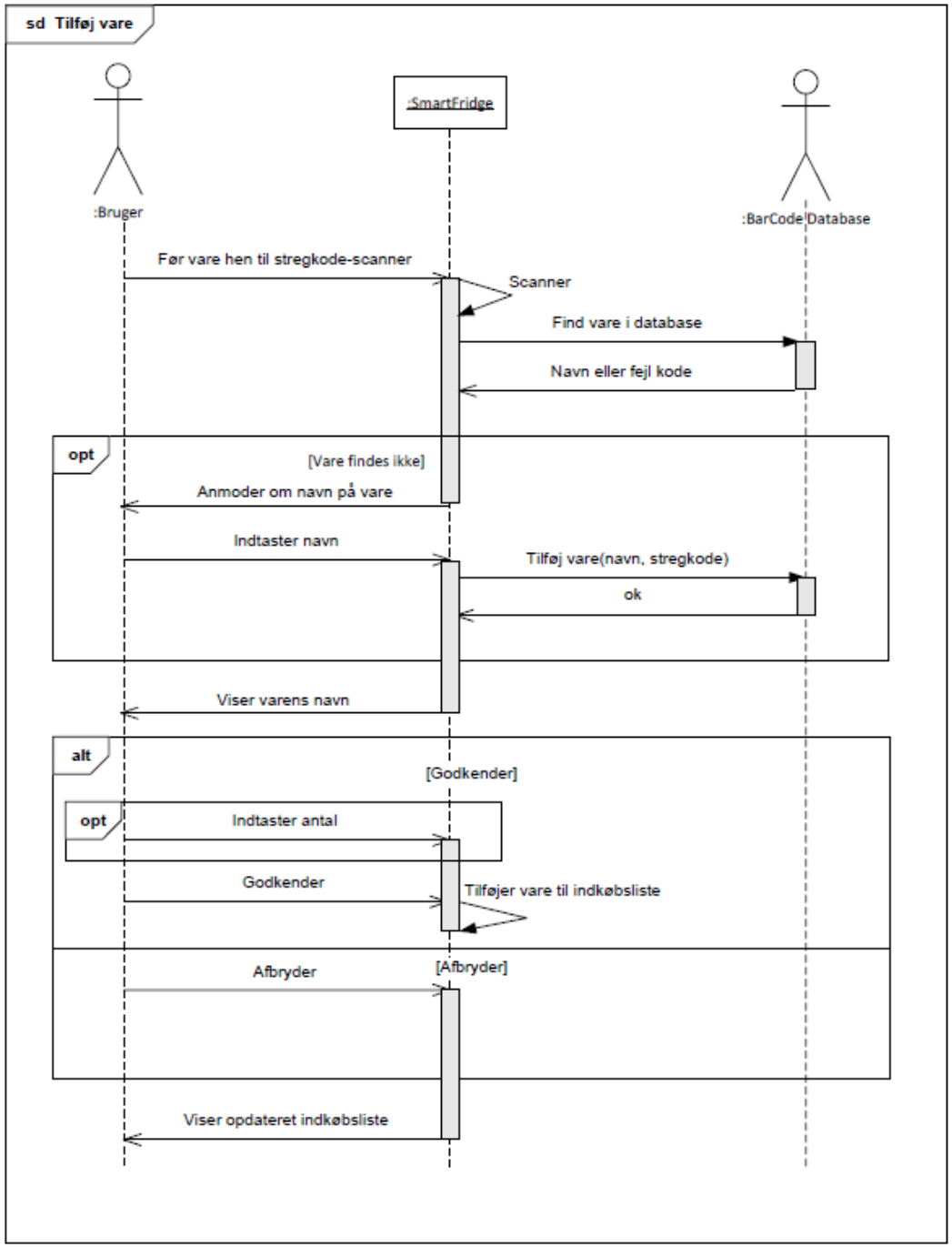
Step 1.3: Identify relevant classes in the domain model involved in the UC → *Domain* classes

So are DM classes!

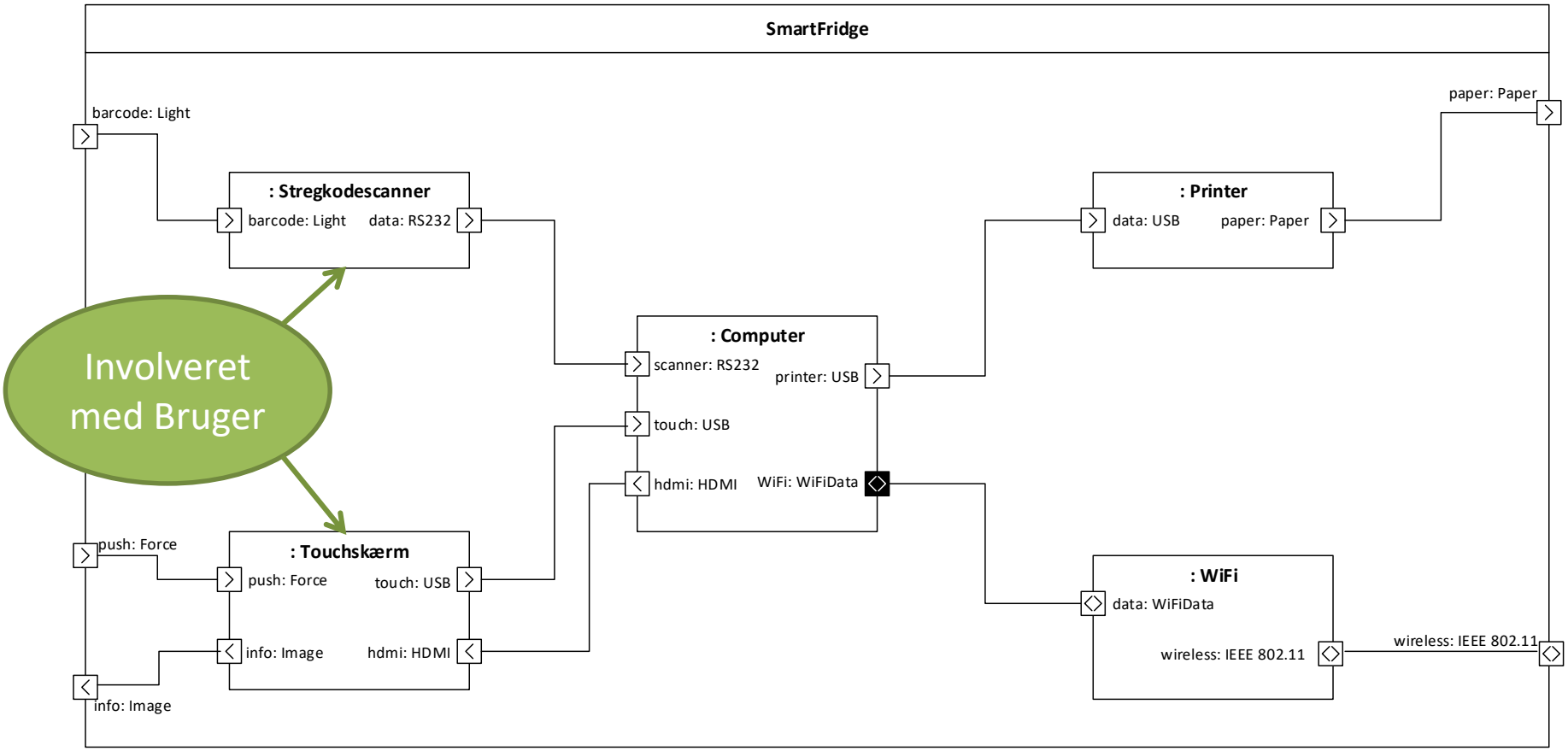
Step 1.4: Identify the UC *controller* → *Controller* class

Boundary classes and hardware interfaces

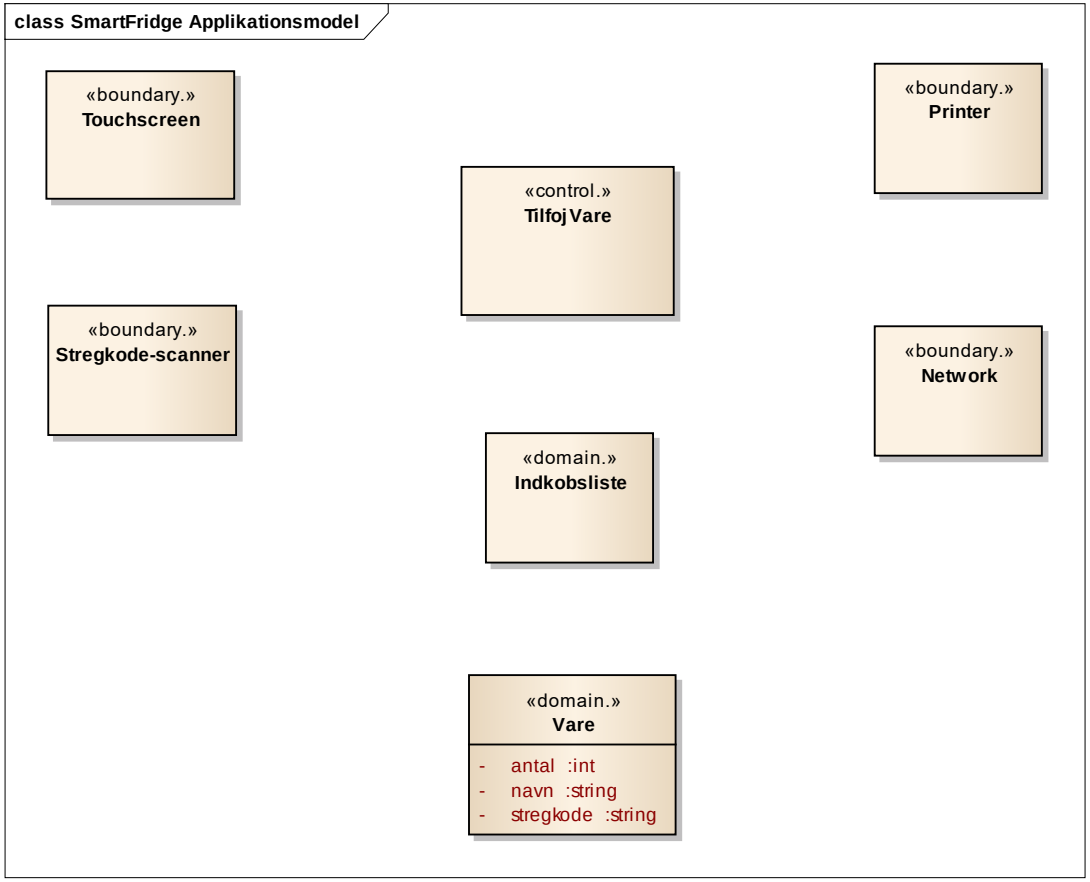
- ***Boundary classes*** are the parts of the software that handles the interfaces to the **Actors**
- The software runs on the computer/microprocessor
- The interfaces to the **Actors** must be the interfaces of the processor!
- Look for hardware interfaces from the processor to the **Actors**!
- Make each into a ***boundary class***!


sd Tilføj vare
:Bruger
:SmartFridge
:BarCode Database
Før vare hen til stregkode-scanner
Scanner
Find vare i database
Navn eller fejl kode
opt
[Vare findes ikke]
Anmoder om navn på vare
Indtaster navn
Tilføj vare(navn, stregkode)
ok
Viser varens navn
alt
[Godkender]
opt
Indtaster antal
Godkender
Tilføjer vare til indkøbsliste
Afbryder
[Afbryder]
Viser opdateret indkøbsliste

Find Boundaryklasser



1. Version SAM klassediagram



The System Application Model – Step 2

- The collaboration between the classes is now explored from the UC description

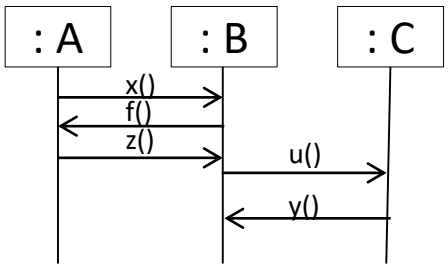
Step 2.1:	Go through the UC main scenario step-by-step and identify collaborations (**actor- or system-initiated**)
Step 2.2:	Update the application model's sequence and class diagrams to reflect the collaboration (relations, operations, attributes)
Step 2.3:	Identify any classes with state-based behavior and update STMs for the classes (states, events, transitions) . *(Step 2.3 is skipped if none classes with state-based behavior)*
Step 2.4:	Verify that the diagrams adhere to the UC (descriptions, test)
Step 2.5:	Repeat 2.1 – 2.4 for all UC exceptions. Refine model.

- Note: All 3 diagrams (class, SEQ, STM) are updated at the *same time* in this process.

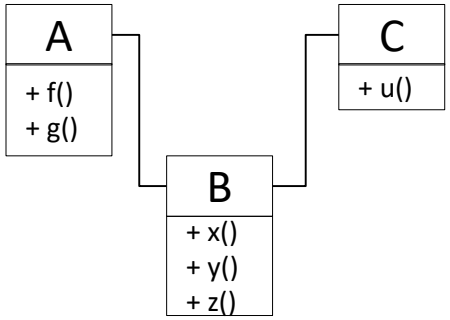
Principle for step 2: Go through main scenario, update collaborations

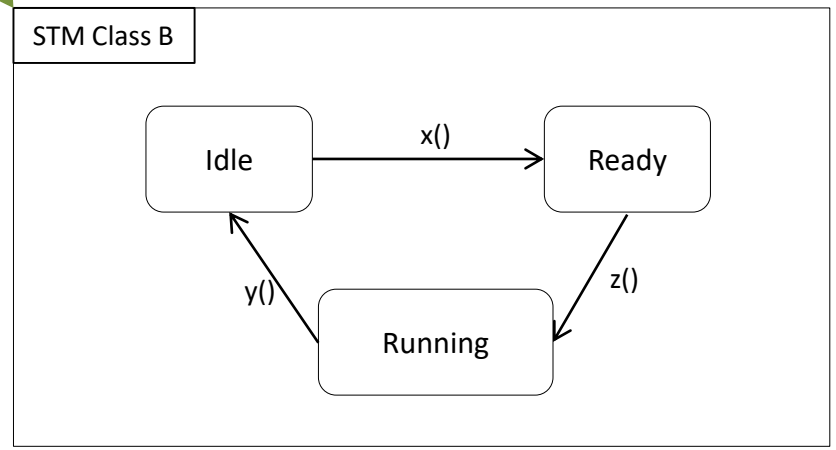
Sequence diagram:
- Add sequence of calls to objects

Class diagram:
- Add operations to classes
- Update associations



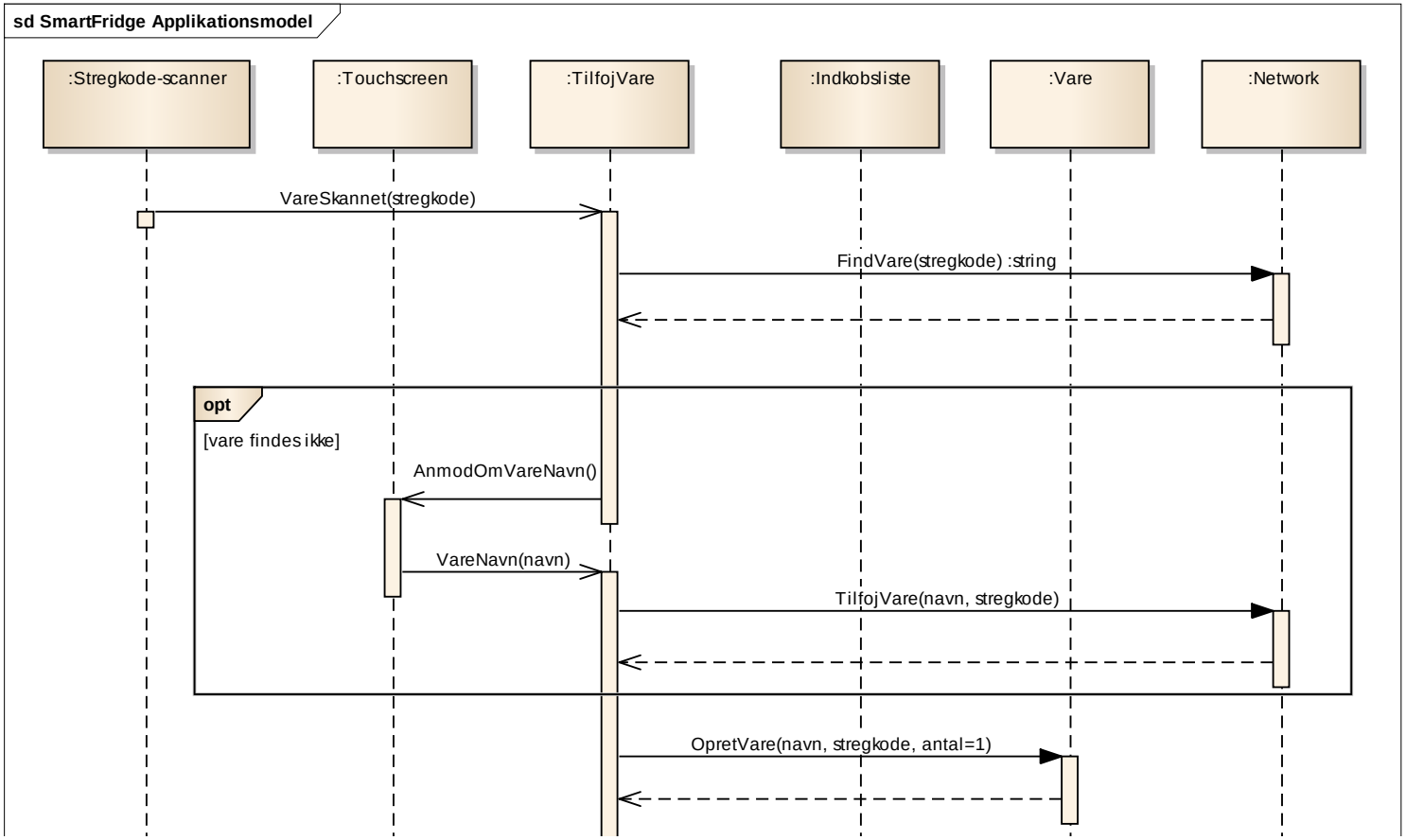
Updated simulta-neously



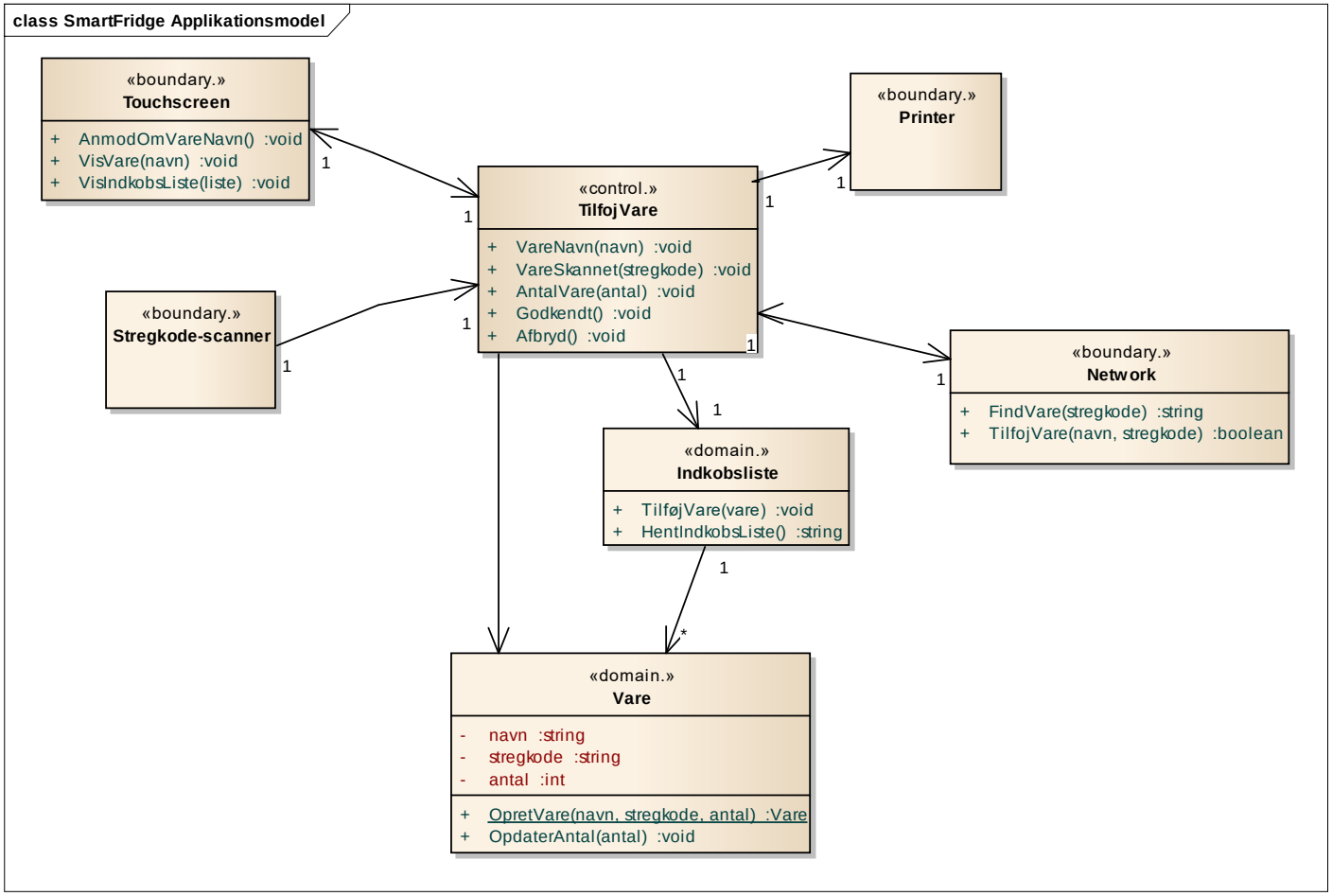


State diagrams:
- Add states to stateful classes
- Add actions (operations) that changes the state

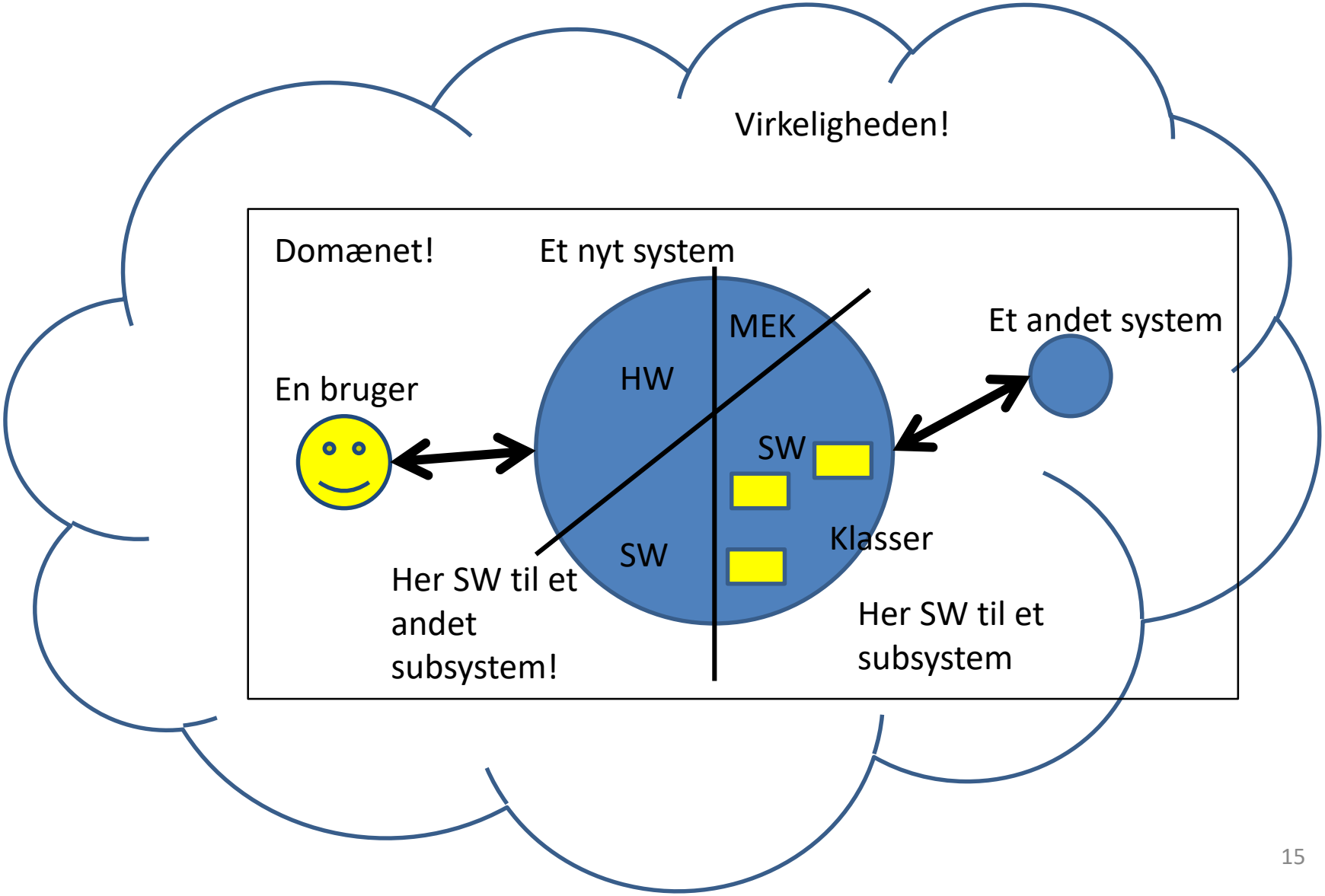
SAM sekvensdiagram



SAM endeligt klassediagram



Virkeligheden og systemet – med subsystemer!



The System Application Model – Step 1 Version 2!

- The application model is constructed incrementally in units of use cases.
- **One for each subsystem!**

Step 1.1: Select the next fully-dressed UC's to design for

Step 1.2: Identify **all interfaces to** actors involved in the UC **and all connections to other subsystems as seen in the ibd** → *Boundary* classes

Step 1.3: Identify relevant classes in the domain model involved in the UC → *Domain* classes

Step 1.4: Identify the UC *controller* → *Controller* class

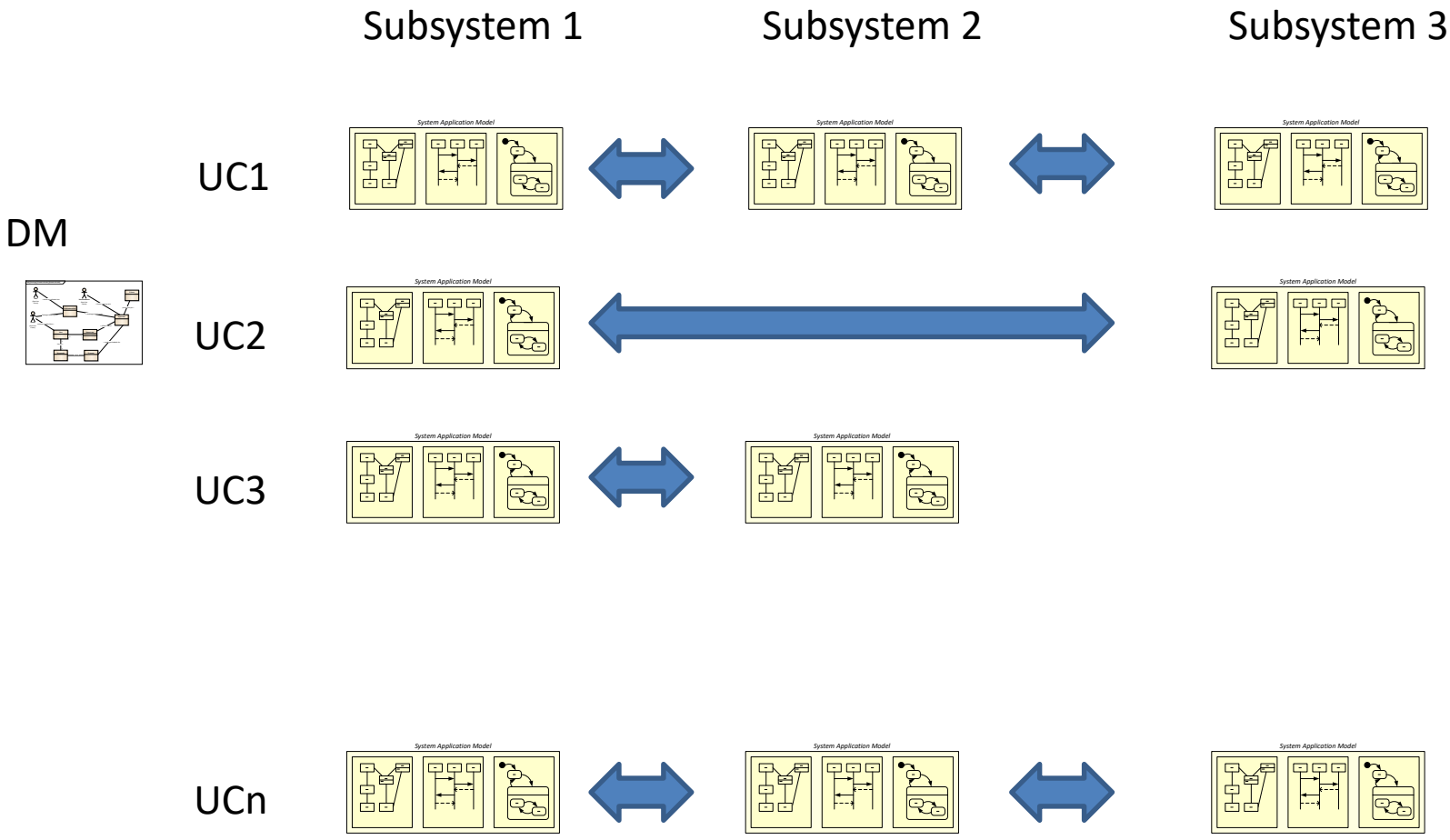
The System Application Model – Step 2 Version 2!

- The collaboration between the classes is now explored from the UC description **and the system sequence diagrams**
- **For each subsystem!**

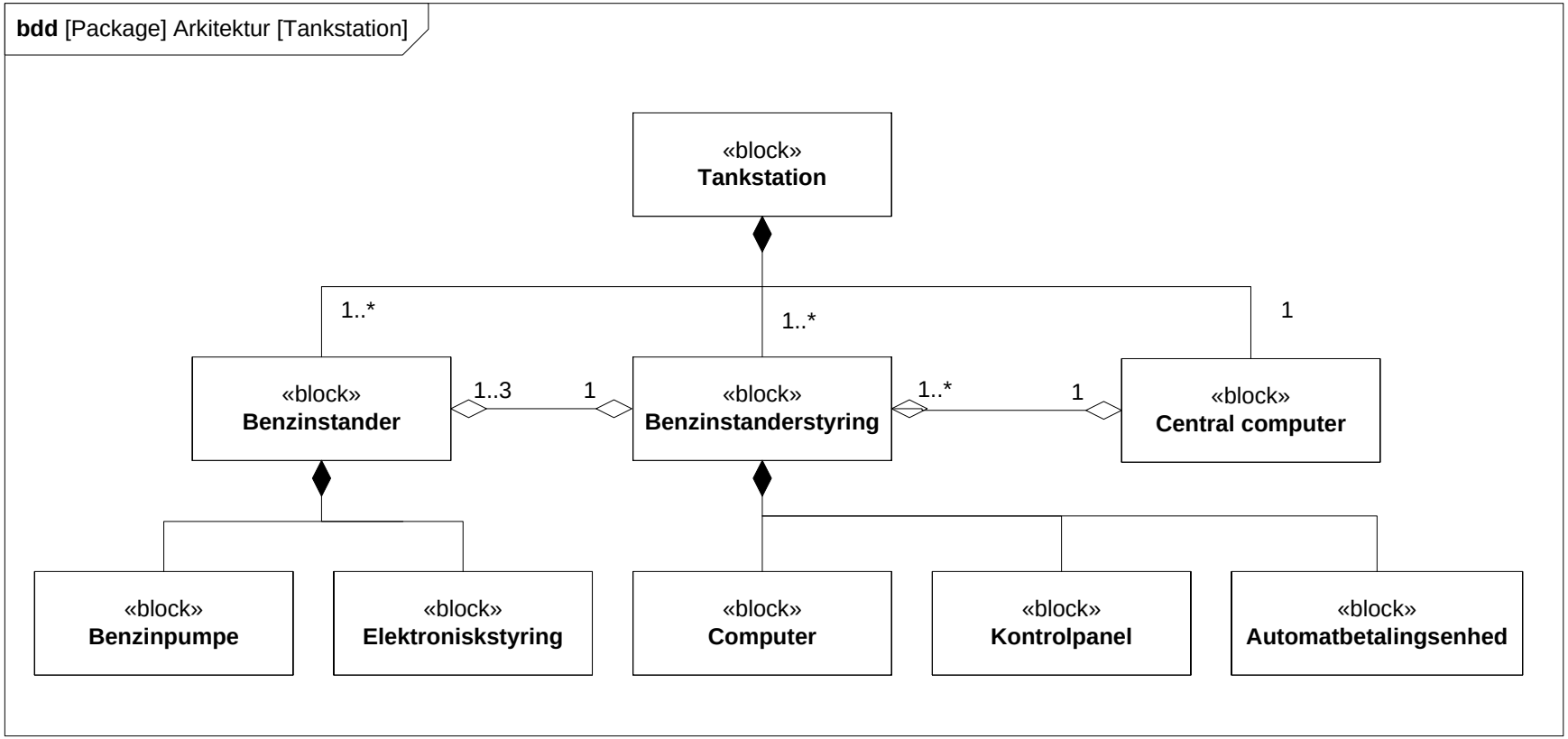
Step 2.1:	Go through the UC main scenario step-by-step and identify collaborations (**actor-, subsystem, or system-initiated**)
Step 2.2:	Update the application model's sequence and class diagrams to reflect the collaboration (relations, operations, attributes)
Step 2.3:	Identify any classes with state-based behavior and update STMs for the classes (states, events, transitions) . *(Step 2.3 is skipped if none classes with state-based behavior)*
Step 2.4:	Verify that the diagrams adhere to the UC (descriptions, test)
Step 2.5:	Repeat 2.1 – 2.4 for all UC exceptions. Refine model.

- Note: All 3 diagrams (class, SEQ, STM) for one **subsystem** are updated at the *same time* in this process.

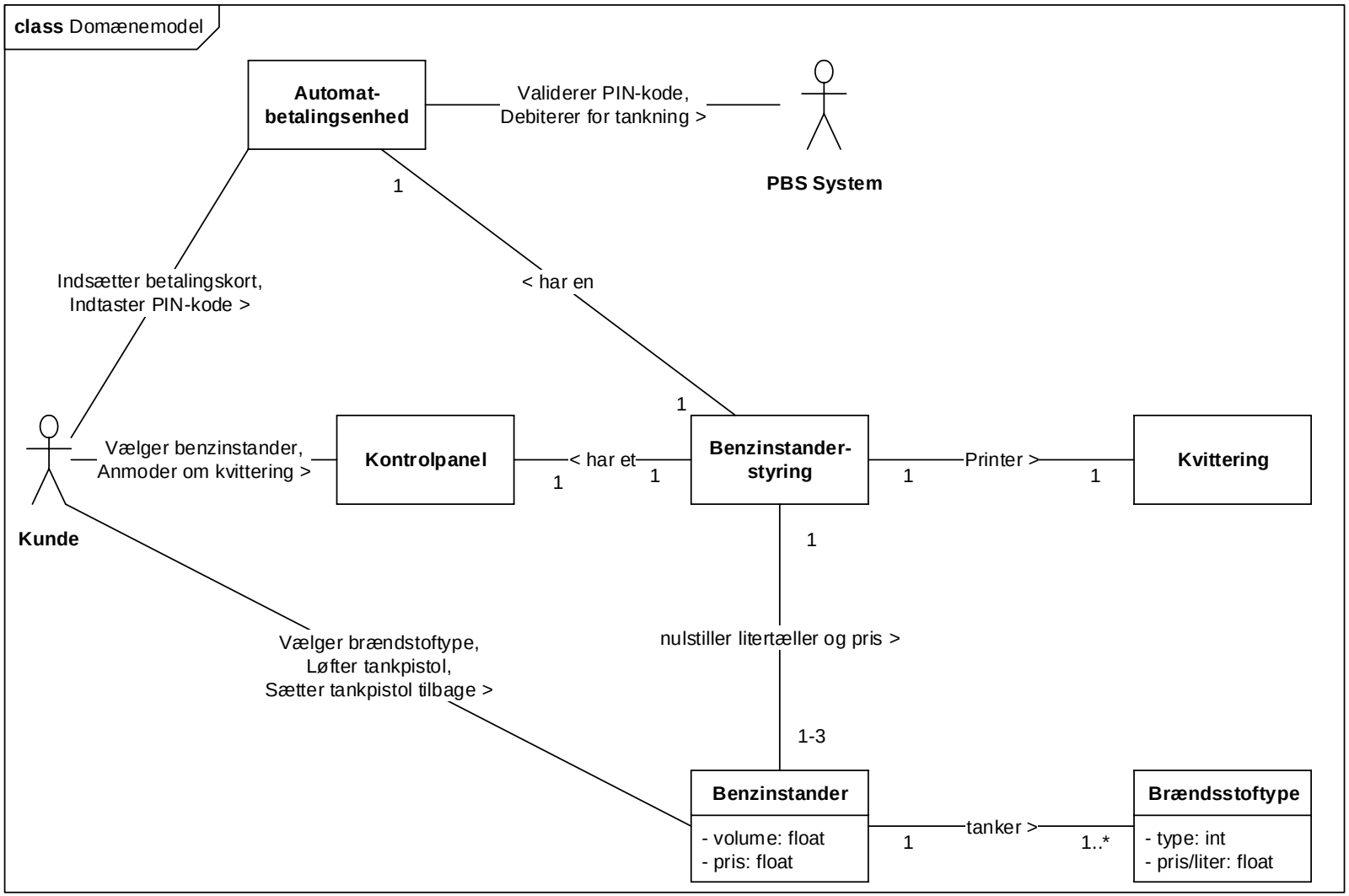
Applikationsmodeller for samarbejdende subsystemer



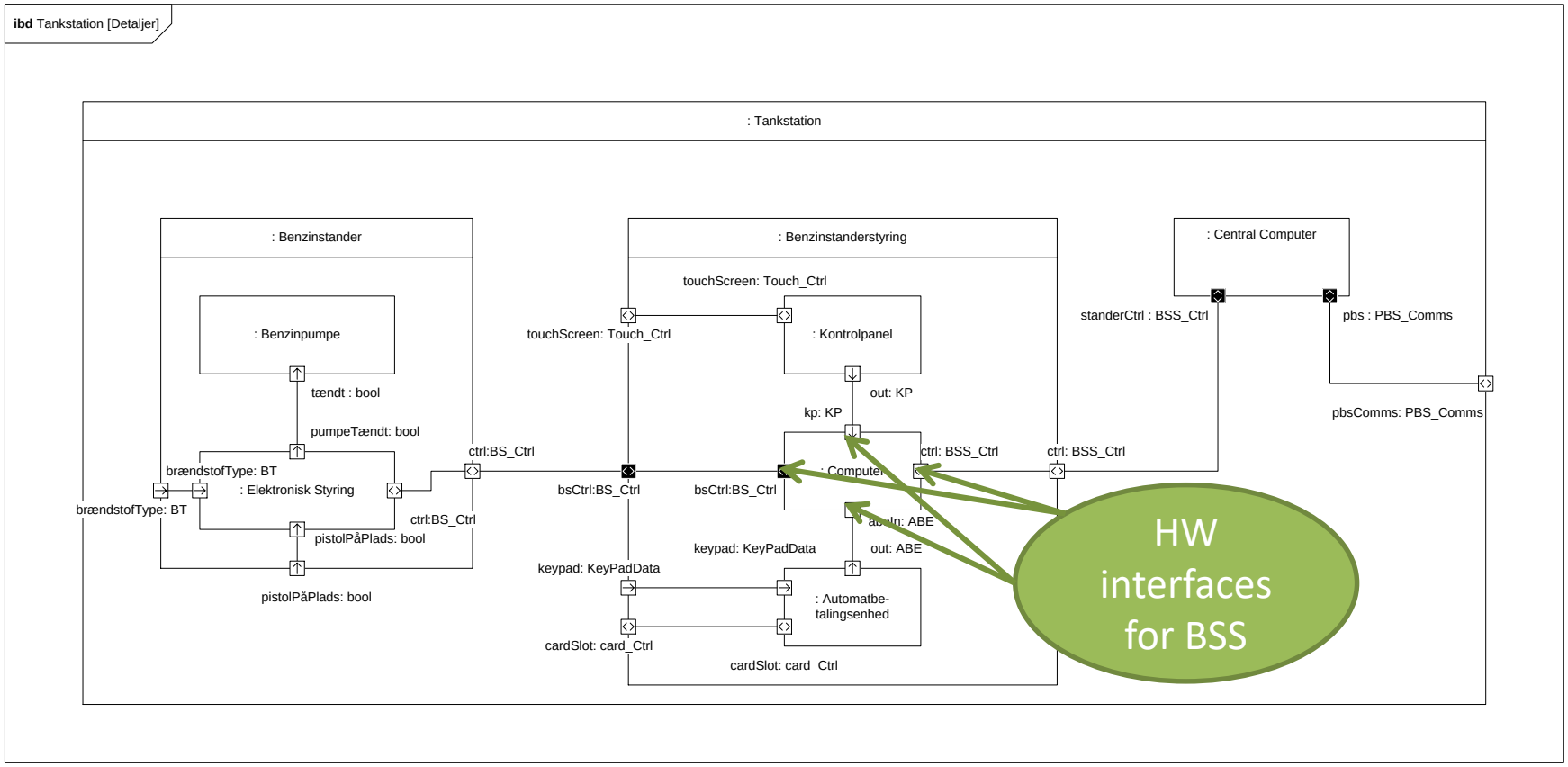
System med subsystemer



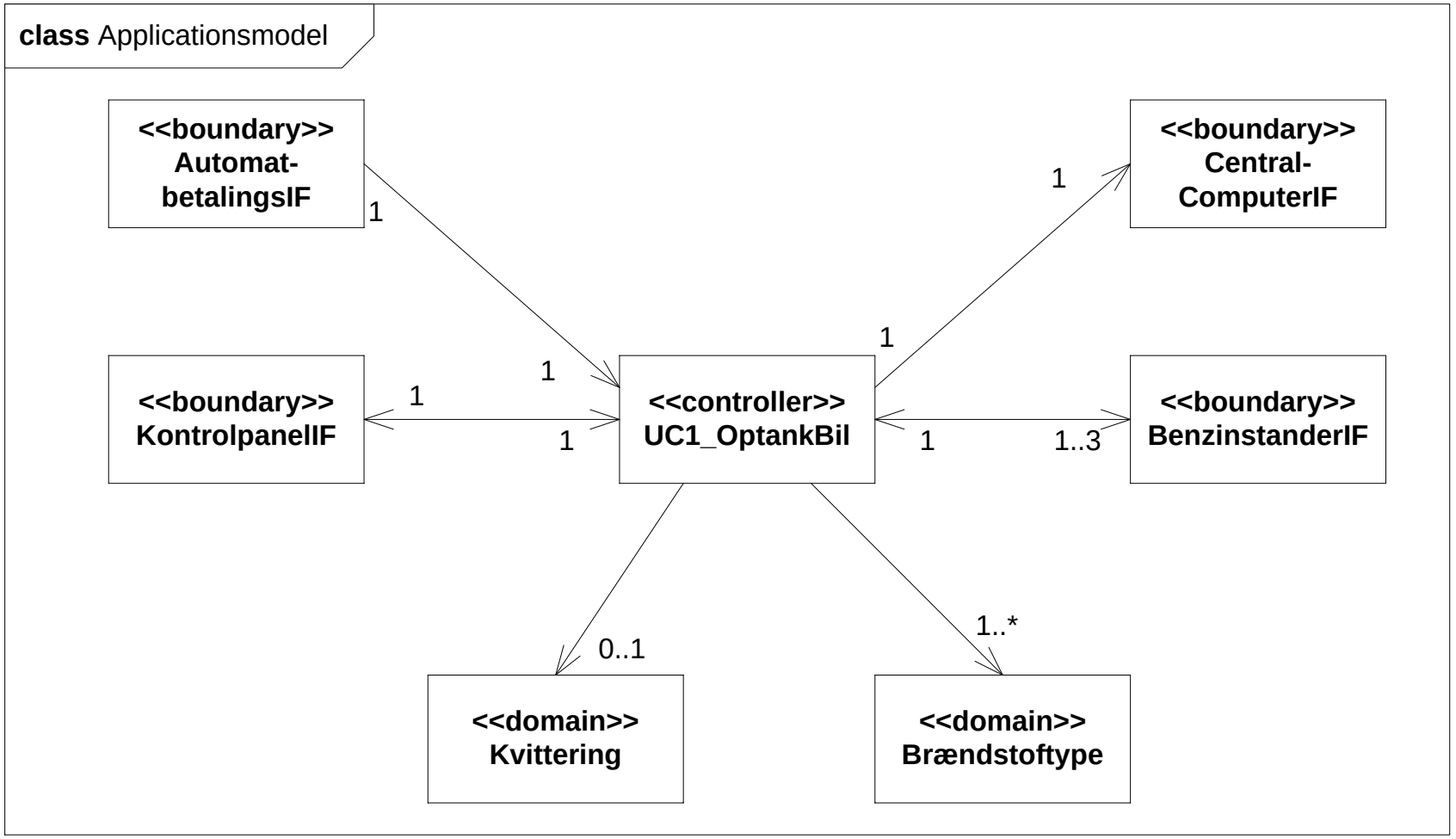
Domænemodellen for hele systemet



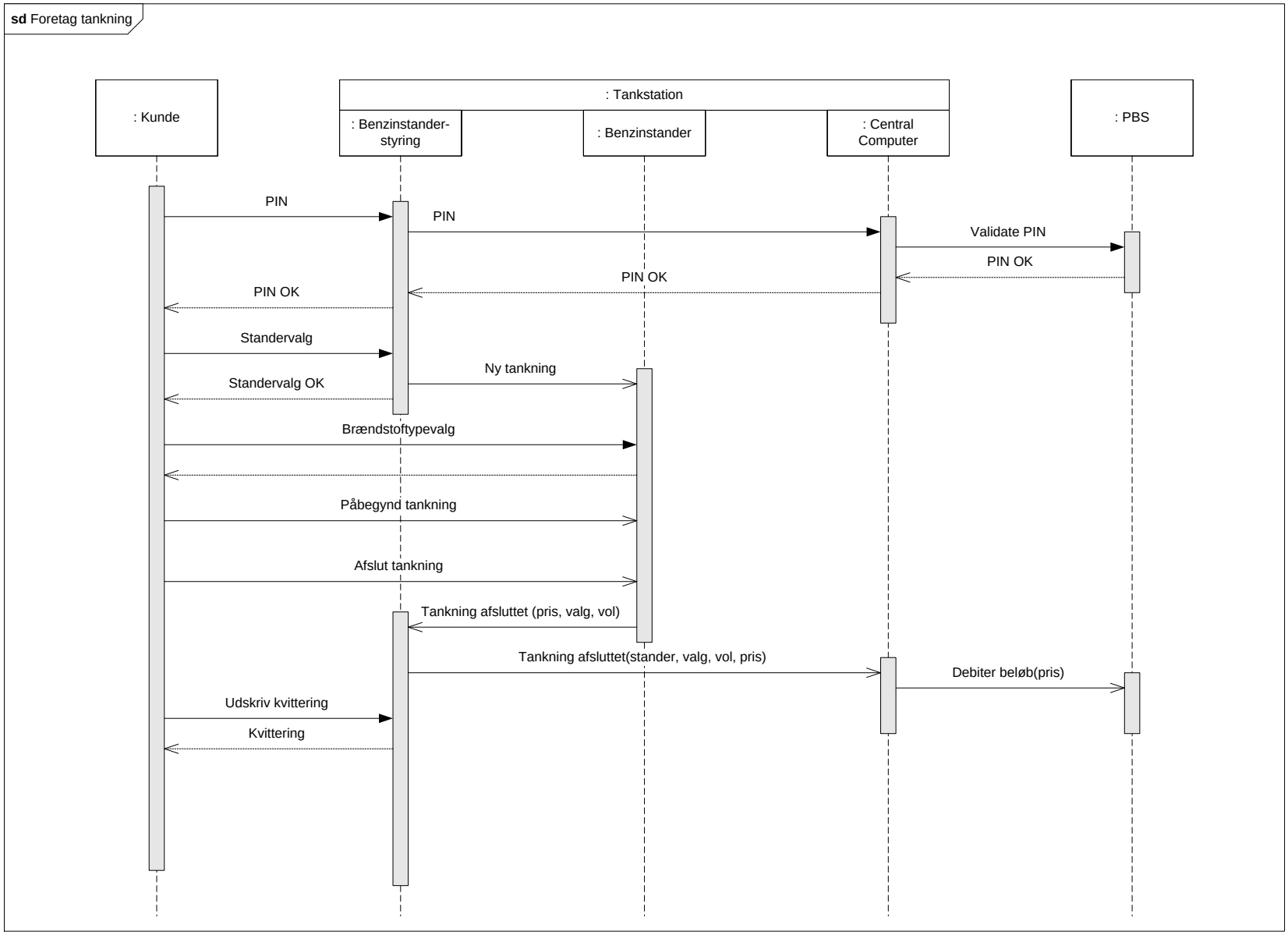
IBD for systemet



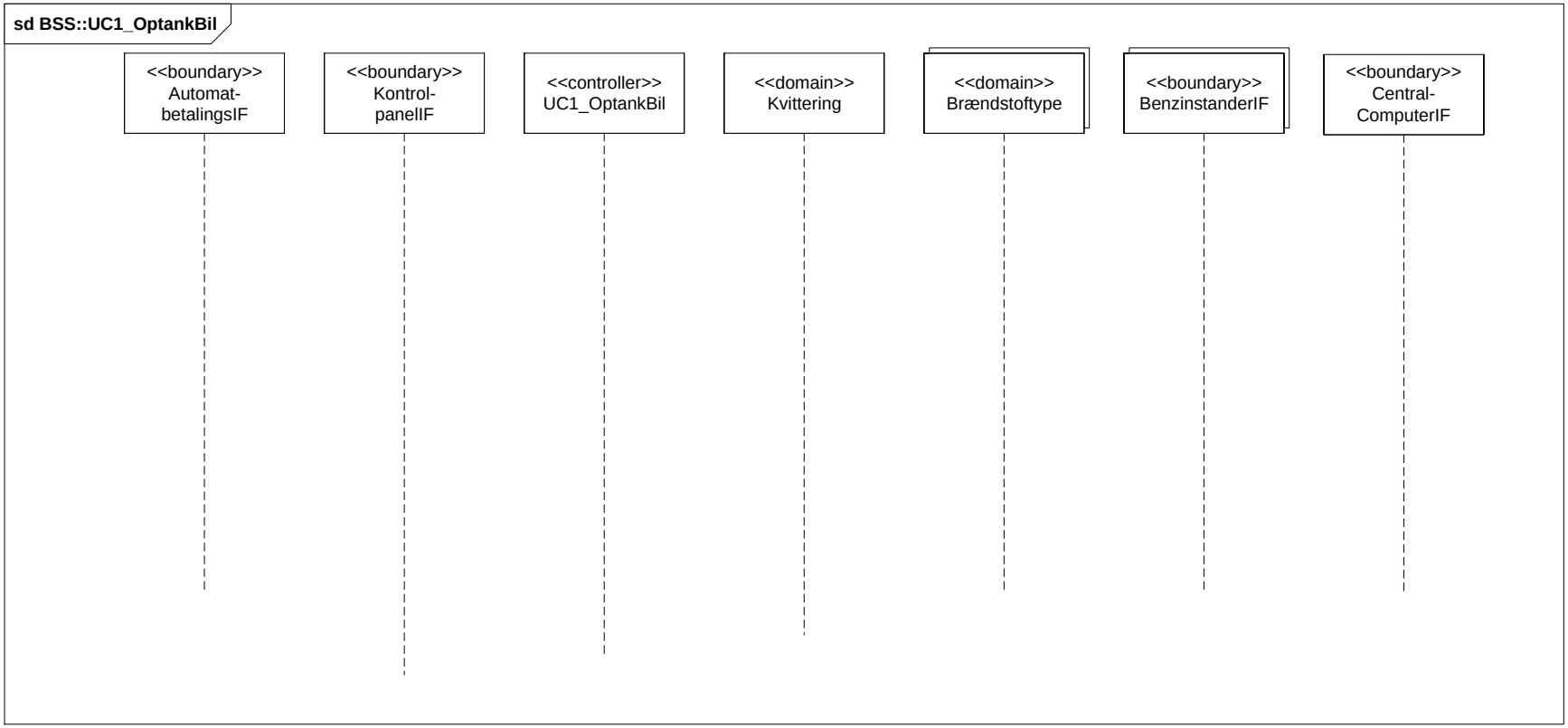
1. Version af SAM klassediagram for Benzinstanderstyringen (BSS)



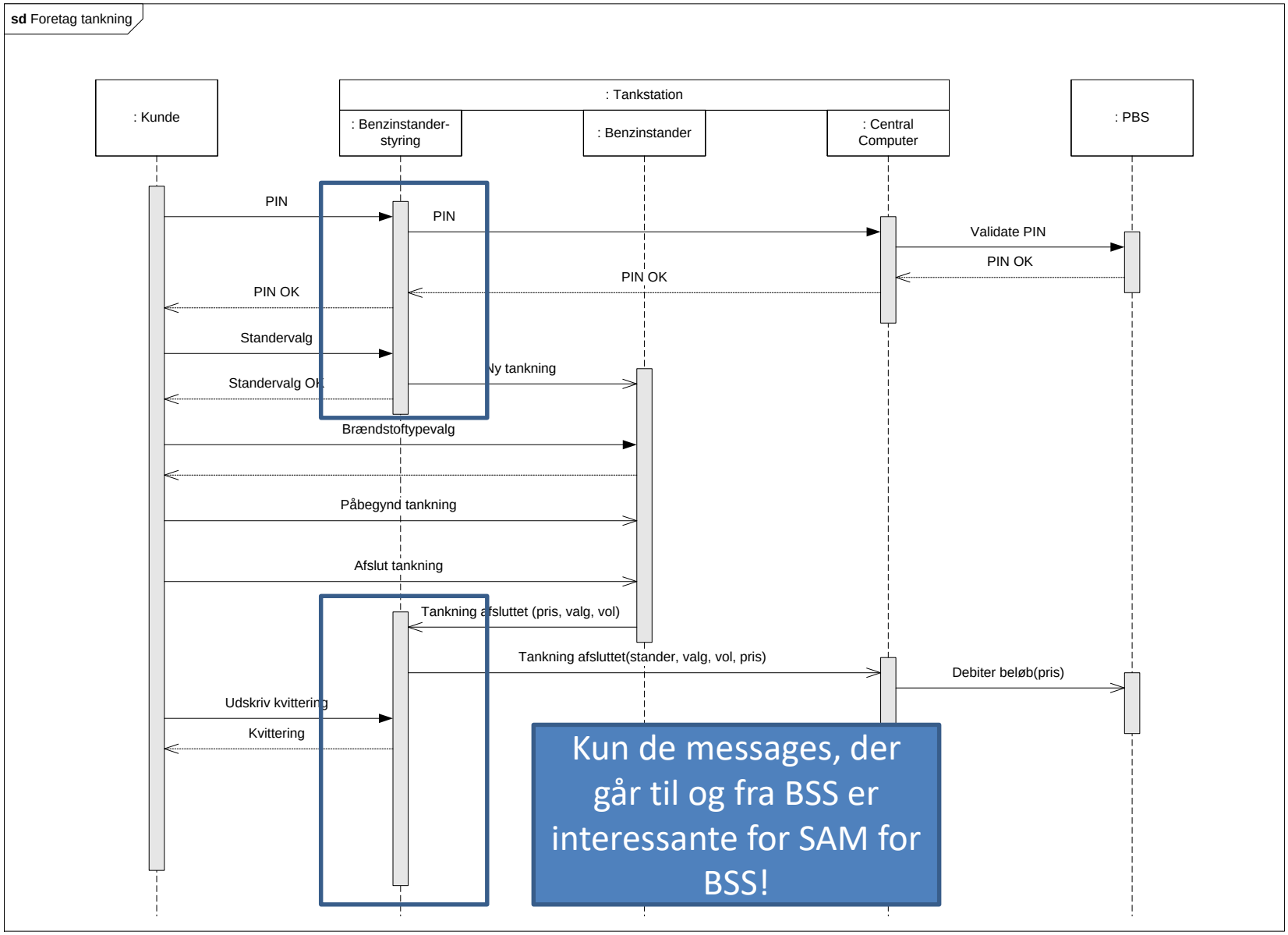
System SD for UC Optank Bil



Start på SAM SD for BSS for UC Optank Bil



System SD for UC Optank Bil



Your turn: System Application Model for Benzinstanderstyring – UC Optank Bil

- Complete the system application model for the Benzinstanderstyring – UC Optank Bil, using
  - bdd and ibd
  - System Sequence Diagram
  - Domain Model
  - initial class diagram

1. Check if steps 1.1-1.4 have been completed for the supplied class diagram for the System Application Model
2. Complete steps 2.1-2.5 for the UC