**VIGTIGT!!:**

**En væsentlig udfordring i denne del er**:

At få lavet en fornuftig disponering af materialet.

Valg skal diskuteres før konklusionerne bruges i anden sammenhæng osv. 🡪

3. **Designovervejelser:**

Hvordan realiserer man ønskerne i et program?

Hvilke centrale valg foretager man?

hvad er de relevante alternativer af valg?

Overvejelser om hvilke metoder der skal bruges til at løse det?

**Designovervejelser:**

I analysen bør man overveje anvendelsen af centrale algoritmer eller datastrukturer.

F.eks. hvilke datastrukturer der kunne bruges, hvilke algortimer der var relevante,

hvilke klasser og objekter man har valgt at lave - samt hvorfor.

Hvilken begrundelse ligger der bag f.eks. valget af datastrukturer?

Overvej alternativer, for ofte vil alternativerne være mindst lige så interessante for en læser som jeres valgte løsninger.

Bliv ikke for programnær.

Det er det centrale design man skal diskutere og ikke specifikke overvejelser om forskellige faciliteter i sproget.

De tekniske muligheder og problemer man har med at få programmer til at virke er mange overvejelser værd når man endnu ikke er så erfaren programmør, men det er nok ikke så interessant at læse om.

**🡨 VIGTIGT!!:**

**En væsentlig udfordring i denne del er**:

At få lavet en fornuftig disponering af materialet.

Valg skal diskuteres før konklusionerne bruges i anden sammenhæng osv.

**Design Considerations part 1**

Vores:

**Hierarchical Finite State Machines (for the robot's decision making):**

The FSM can change from one state to another in response to some external inputs; the change from one state to another is called a *transition*. A FSM is defined by a list of its states, its initial state, and the conditions for each transition.

**Network behavior anomaly detection** (**NBAD**) provides one approach to [network security](https://en.wikipedia.org/wiki/Network_security) threat detection. It is a complementary technology to systems that detect security threats based on [packet signatures](https://en.wikipedia.org/wiki/Deep_packet_inspection).:

Når den ene robot gør noget så gør den anden ikke det samme og er klar over hvad de andre gør hele tiden for at gøre noget andet.

**Færdighed i at specificere og modellere krav**

Til it-systemers funktionalitet.

**UML Models part 1**

**Der stilles ikke krav om:**

Andre værktøjer (f.eks. referencelinjer, UML).

**Design process part 1**

**Færdighed i at vælge og argumentere For valg af:**

Design til det konkrete projekt

**Fritzing models. Part 1**

**Færdighed i at specificere og modellere krav**

Til it-systemers funktionalitet.

Diskussion til hvordan vi løser problemformuleringen?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

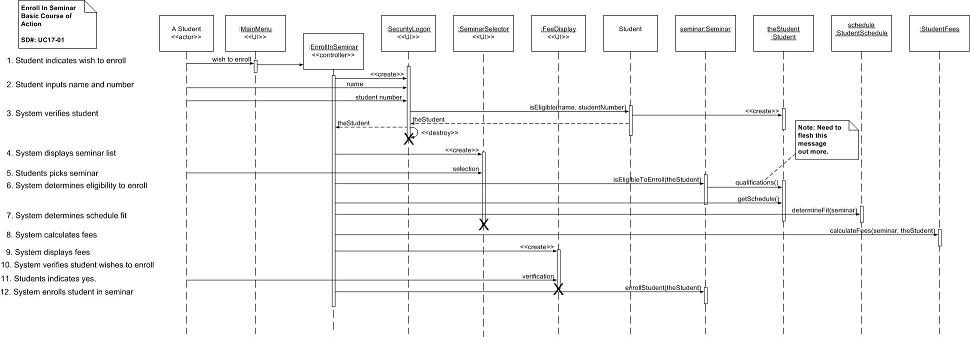
**UML Models part 1**

**Før design part 1. UML Modeller part 1:**

1. Design model (den nederste på s. 275) viser hvad domænet (i vores tilfælde klasserne skal indeholde. (Mere detaljeret forklaring se nederste figur på side 275)
2. Design model med operating contracts (s. 275 Øverste design model) Den viser forskellige domæner og hvad de skal indeholde og hvordan de forskellige domæner **henter og sender samt bruger hinandens data** **fra funktionerne og data som klassen indeholder. (Vores domæner er her klassens funktioner og hvordan de bruger hinanden samt hvordan de deler data).** s. 357 øverste og nederste model dynamic design med loop. Brug eventuelt på design modellen. s. 364 visning af hvad de indeholder.
3. s. 364 visning af hvad de indeholder. + 366 + 367 lidt anderledes modeller. Eventuelt noget vi også kunne bruge.
4. **Sekvensdiagram: Viser hvad vi gerne vil have vores robotter til at gøre og i hvilken rækkefølge samt hvordan forløbet med dets hoved loop med dets funktioner foregår.**
5. **Forskellen på interaktionsdiagram og et design diagram er at i et design diagram er der ikke en bruger! – vigtigt**

Sekvens diagram som viser vores klassers indholder (se side 222). (Giver et godt overblik og en god måde at vise hvad vores robotter kan). + se nedenstående sekvens diagram. SSD Diagram med udestående aktører. S. 409 (Måske vigtigt).

Sekvensdiagram eksempel:

[](http://www.agilemodeling.com/artifacts/sequenceDiagram.htm)

1. **UML modelling:**
2. If we are using some server based code for our Arduino we could make a UML sequence diagram to show what the Arduino is sending and receiving from the server and vice versa. This could be done through a ethernet shield for the Arduino to access the internet and receive the signal.
3. You could also use a sequence diagram to show what happens between a server if you are using one and the actual computer. For example, the server is sending some file to the computer and the computer response back to the server. Or you could use to show different transitions between the weather change you mentioned.
4. Eventuelt i stedet for design modellen (s. 275 nederst): kan vi bruge Domæne model. (S. 283 øverst.)

I domæne modellen viser vi hvor vores program henter data fra og hvor den bruger dem i hvilke funktioner og hvordan de er koblet til hinanden og bruger af hinandens data).  
F.eks. at vores robot består af et setup som har nogle data. Udover et setup er der et loop som indeholder funktioner som indeholder data. Husk også hvordan funktionerne deler dette data imellem sig med pilen eksempel: (phys🡪)

En domæne model (er mere detaljeret) derimod viser f.eks. i vores tilfælde at vores robot har et setup som indeholder data og et loop som indeholder funktioner med forskellige data men derimod ikke hvordan de forskellige funktioner opfører sig i forhold til hinanden og hvordan de har forbindelse med hinanden. (Mere detaljeret)

**Vigtigt: 533 Domæne model udvidet med iteration 3.**

Vigtigt: Mapping design to code (Vigtigt): s. 371, 373, 375, 376 – 388,

Generelt: Klasse hiarki s. 511 + 512 + 513 + 514 + 515 + 518 + 527, **528 (nederst) + 529 (nederst) (store model) + 530 + 531, + 532, (vigtig), + 625, 626 (øverst), 627 (øverst) Tables og objekter. (Vigtigt) + 631 + 632, 633 (Vigtigt) eventuelt: s. 642, databaser: 644 (Hvis vi bruger databaser), eventuelt andet: 668 (tjek det ud)**

**Første side i bogen + 2 side + næst sidste side + sidste side i bogen for alle modellerne tror jeg.**

**Tråde og meddelser i UML i et interaktionsdiagram (usikker) se side 593 (Vigtigt!)**

Vandfaldsmodel/iteraitoner i vores agile model:

**s. 667 + 668 for iterations planer med vores agile metode eller vandfaldsmodellen. (vigtigt!)**

**Undtagelser i et interaktions diagram se s. 597**

**s. 600 for et godt interaktions diagram! (ØVERSTE model) (VIGTIGT)**

1. **Fully dressed use case:**

We could eventually use a “fully dressed use case” to show the action of a user interacting with a screen and our robot and the server part. Eventually show the users goal and success scenario.

Aktivitets diagram s. 478 + 480 + 482 + 483, hvis vi har brugere med.

Hvis vi har interaktion med en bruger: se s. 372 for interaktions diagram. (Mapping design to code)

Eventuelt bruge et state machine diagram se s. 491 nederste figur. Dog kun hvis vi laver en use-case med en bruger

Eventuelt hvis vi har brugere som inteagere med vores robotter så kan vi bruge et Use case diagram se s. 499.

**SSD diagram: s. 536 med loop VIGTIGT + 539 Sammenligning med use case postconditions og SSD diagrammet på s. 538 tror jeg.**

**Pakke diagram med logiske lag, Userinterface (bruger), domæner (klasser) og udestående f.eks. en server (vigtigt) s. 662. (hvis vi har en bruger med)**

**Interaktionsdesign med logisk arkitektur s. 564 (Hvis vi har en bruger med)**

**SSD Diagram med system operationer med lag s. 569 (Hvis vi har en bruger med)**

Godt til robotter og oversigt over pakker + klasser:

**Logisk arkitektur før og efter design. S. 560 + 562 + 563**

**Måske: (ekstra): Data flow view s. 664, eventuelt use case scenarie s. 665,**

**Forklaring af funktioner se s. 606 (bedre end s. 607 model) + 607 (lidt for meget)**

**Forklaring på polymorf metoder. s. 610**

Andet vigtig information: 523, 525 (nederst),

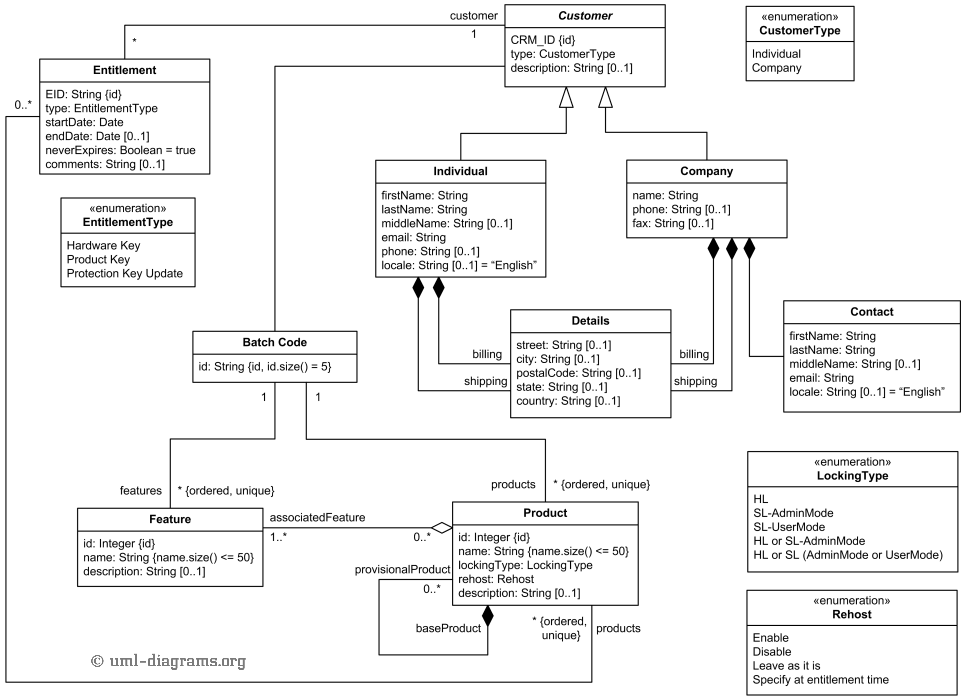
**Eventuelt godt til vores robotter:**

**Eksempel på interaktionsdiagram se side 613. + lidt mere info men ikke så vigtigt s. 614 + 615 (tror jeg) + 616 (Vigtigt), 617 (Vigtigt)**

**Polymorf iteration 3 s. 618. (Vigtigt)**

1. **Eventuelt kan vi bruge et Klassediagram også:**

Et klassediagram viser hvordan vores robotter er bygget op, med setup og hvad det indeholder, hvad vores loop indeholder og hvad funktionerne indeholder. (Ikke sikkert pga. vi også har design modellen i forvejen) (Denne er også for bred. Vi kunne bruge klassediagrammet hvis vi kørte med at robotterne f.eks. kendte til hinandens positioner eller at de fik funktionerne og data fra en server).

[](http://www.uml-diagrams.org/class-diagrams-examples.html)

**Forskellen på domæne model og et klassediagram:**

Et klassediagram (er mere bredt) viser flere områder end en domæne model f.eks. en børnehave og en skole kan have forbindelse til hinanden og hvordan de skal hente og henter noget fra hinanden som de er afhængige af hinanden. Klassediagrammer viser også hvad vores robotter indeholder overordnet set. (Derfor har vi ikke brug for dem medmindre at vi bruger en server og at robotterne kender til hinanden som beskrevet lige ovenover).

You could use some UML diagrams for example package diagram to show which files are dependent on each other.

The UML package diagram can give you a good overlook which parts of the server files and computer files which are sending data between and the a very important thing about this is also to look at the files that are depending on each other and how you can optimize the separation between files and how they can be less depended on each other.

You could use UML Use Case Diagram to show a success scenario when the user is using your program.

**Design considerations:**

**Other suggestions:**

Eventual we have some suggestion to what our robots could be drawing besides the specific requirements**:**

**Forklar om de basale dele vi har brugt – også til vores robotter. – vigtigere end at tilføje alt for mange dele på den og få det til at virke. – dog skal det helst virke. Forklar hvordan det løser vores specific requirements.**

**We could make the robot draw shapes from a screen which the user interacts with.**

**We could also make the robot draw shapes that people are drawing with a tool on screen which is then send to the Arduino over a network based shield on the Arduino.**

We could also make a robot that draws random generated mazes and eventual draws the solution or that another robot could draw the solution to.

We could also implement processing to draw the lines that our robot could register with some sensor and then show the lines on a screen for the user.

**Fritzing models:**

Hjemmeside: <http://fritzing.org/home/>

**Modelling:**

We can use a modelling program called Fritzing. It’s used to make models for the Arduino board with wires, servos, motors, resistance and so on, and include those models as a predesign and final design for our robots.

**Eksempel fra google:**