

Videregående Programmering (02324)

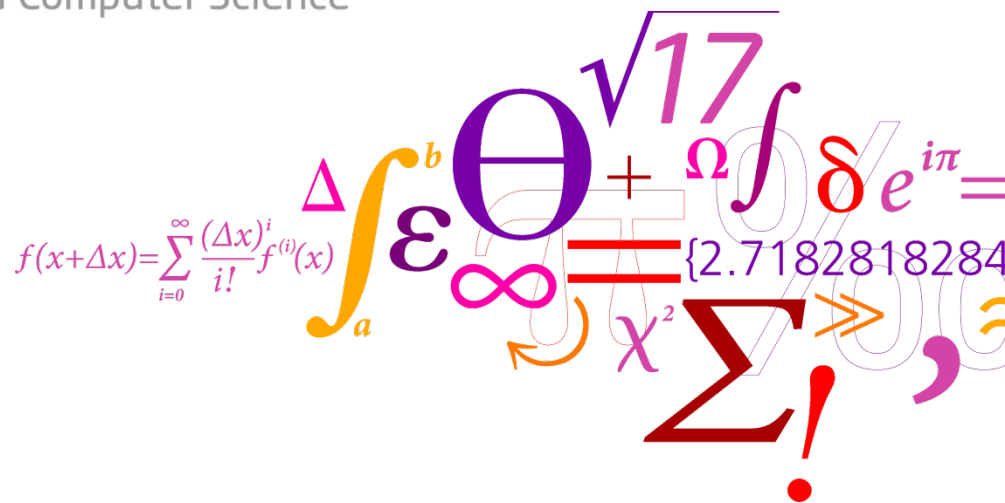
Projekt i software-udvikling (02362)

Forår 2021

Ekkart Kindler

DTU Compute

Department of Applied Mathematics and Computer Science



Meget kort rekapitulation
af uge 1

I. Introduktion

DTU Compute

Department of Applied Mathematics and Computer Science

$$f(x+\Delta x) = \sum_{i=0}^{\infty} \frac{(\Delta x)^i}{i!} f^{(i)}(x)$$

$$\int_a^b \epsilon \Theta + \Omega \int \delta e^{i\pi} = \{2.7182818284\}$$

$$\chi^2 \sum!$$

```
class Pos {  
    int x;  
    int y;  
}  
  
class PosName {  
    String name;  
    Pos pos;  
    public PosName(String name, Pos pos) {  
        this.name = name;  
        this.pos = pos;  
    }  
}
```

Denne kode er ikke pæn!

Kun et teknisk eksempel,
som opfriskning af nogle
begreber og tænkemåder!

```
class Test {  
    public static void main(String[] args) {  
        Pos pos1 = new Pos();  
        pos1.x = 1;  
        pos1.y = 2;  
  
        Pos pos2 = pos1;  
        pos2.x = 3;  
        pos2.y = 4;  
    }  
}
```

Hvilke værdier har
`pos1.x`, `pos2.x`,
`pos1.y` og `pos2.y`
når programmet når her?
Og hvorfor?

→ Se diskussion på tavlen!

```
PosName posName1 = new PosName("position 1", pos1);  
PosName posName2 = new PosName("position 2", pos2);
```

```
Pos[] posArray = new Pos[10];  
for (int i = 0; i < posArray.length; i++) {  
    posArray[i].x = i;  
    posArray[i].y = i;  
}
```

Det ville udløse en
`NullPointerException`

```
posArray[1] = pos1;  
posArray[2] = pos2;
```

```
PosName posName1 = new PosName("position 1", pos1);  
PosName posName2 = new PosName("position 2", pos2);
```

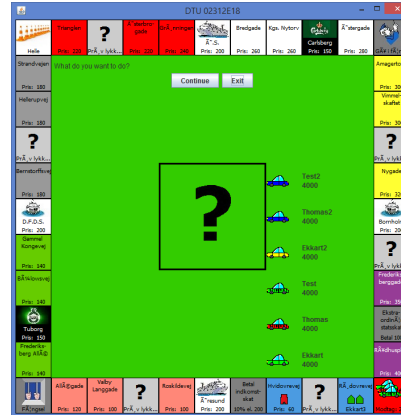
```
Pos[] posArray = new Pos[10];  
for (int i = 0; i < posArray.length; i++) {  
    posArray[i] = new Pos();  
    posArray[i].x = i;  
    posArray[i].y = i;  
}
```

```
posArray[1] = pos1;  
posArray[2] = pos2;
```

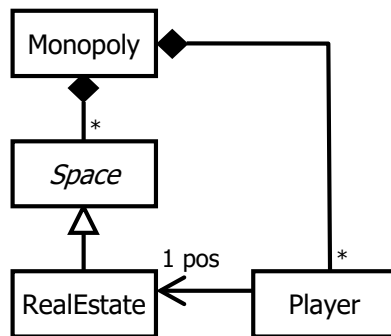
Hvordan ville
værdierne se ud her?

→ Se diskussion på
tavlen!

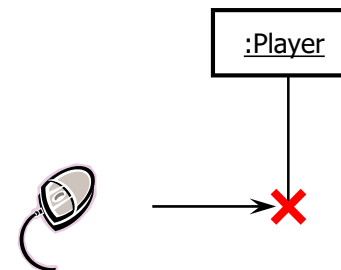
View



Model

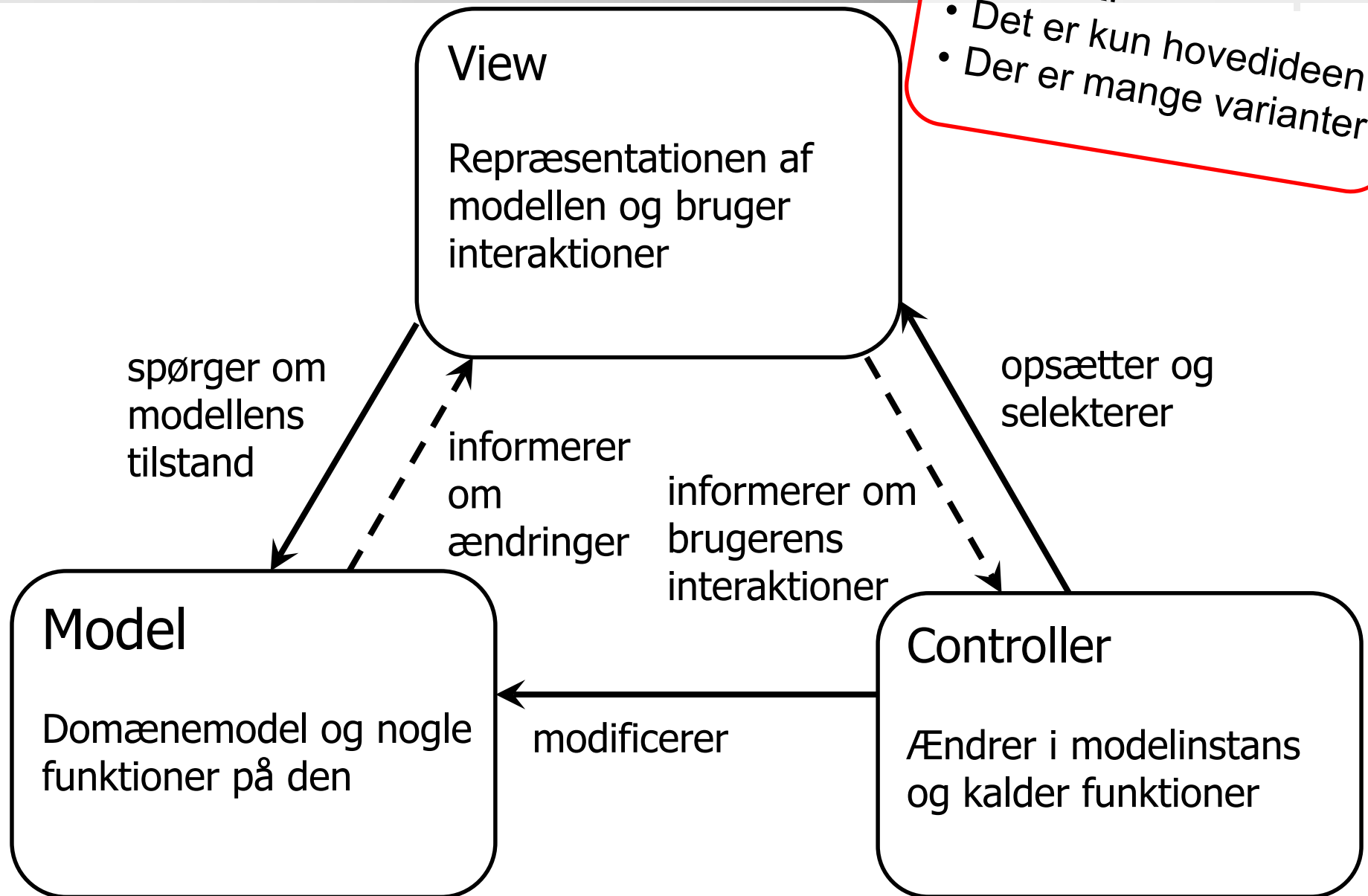


Controller

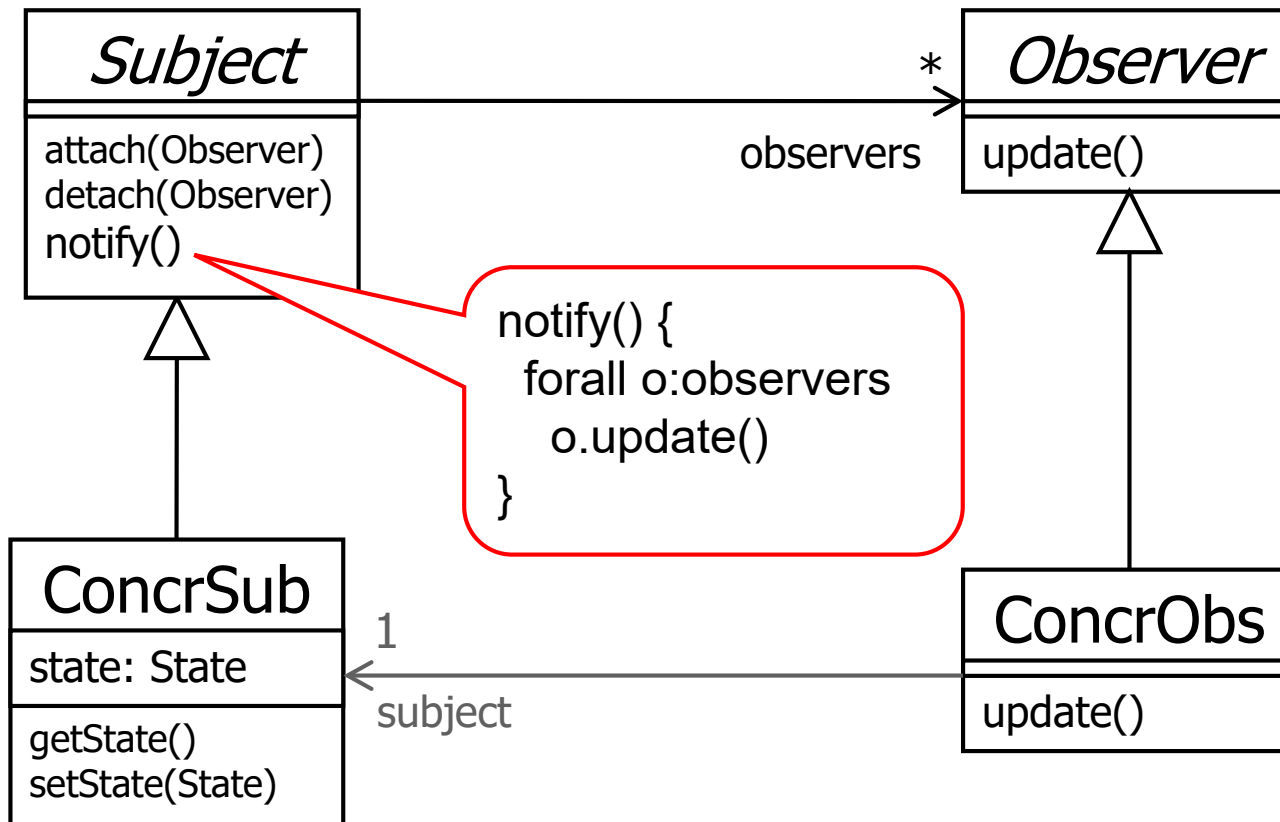


Bemærk:

- Det er kun hovedideen
- Der er mange varianter



Structure



Diskussion: Eksempel

DTU Compute

Department of Applied Mathematics and Computer Science

Ekkart Kindler



```
141 private void continuePrograms() {
142     do {
143         executeNextStep();
144     } while (board.getPhase() == Phase.ACTIVATION && !board.isStepMode());
145 }
146
147 // XXX: V2
148 private void executeNextStep() {
149     Player currentPlayer = board.getCurrentPlayer();
150     if (board.getPhase() == Phase.ACTIVATION && currentPlayer != null) {
151         int step = board.getStep();
152         if (step >= 0 && step < Player.NO_REGISTERS) {
153             CommandCard card = currentPlayer.getProgramField(step).getCard();
154             if (card != null) {
155                 Command command = card.command;
156                 executeCommand(currentPlayer, command);
157             }
158             int nextPlayerNumber = board.getPlayerNumber(currentPlayer) + 1;
159             if (nextPlayerNumber < board.getPlayersNumber()) {
160                 board.setCurrentPlayer(board.getPlayer(nextPlayerNumber));
161             } else {
162                 step++;
163                 if (step < Player.NO_REGISTERS) {
164                     makeProgramFieldsVisible(step);
165                     board.setStep(step);
166                     board.setCurrentPlayer(board.getPlayer(0));
167                 } else {
168                     startProgrammingPhase();
169                 }
170             }
171         } else {
172             // this should not happen
173             assert false;
174         }
175     } else {
176         // this should not happen
177         assert false;
178     }
179 }
180
```

RoboRally 1.1

Designkapitlet (II)
fortsætter vi med
sener!

III. Analyse og konceptuelle modeller

Nu starter vi forfra igen
med analysen eller "hvad"
softwaren skal gøre!

DTU Compute

Department of Applied Mathematics and Computer Science

$$f(x+\Delta x) = \sum_{i=0}^{\infty} \frac{(\Delta x)^i}{i!} f^{(i)}(x)$$

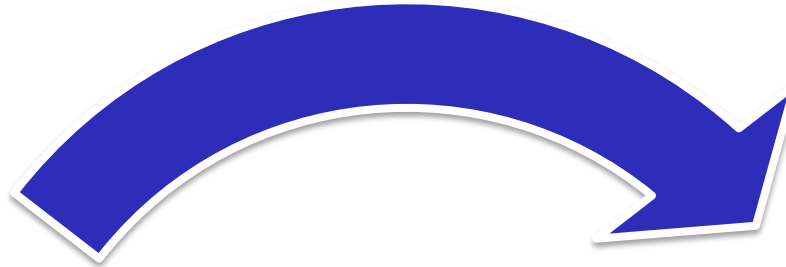
$$\int_a^b \varepsilon \Theta^{\sqrt{17}} + \Omega \int \delta e^{i\pi} = \{2.7182818284\}$$

$$\chi^2 \sum!$$

Analysen drejer sig **kun** om
HVAD (engl. **WHAT**) softwaren skal gøre
(ud fra brugerens synsvinkel)

Analysen siger **ikke noget** om
HVORDAN (engl. **HOW**) softwaren skal
realiseres og implementeres

“Hvad” skal
softwaren
gøre?



WHAT

HOW

Realiteten er mere
kompliceret, men at
adskille **“what”** og
“how” i jeres hoved er
et første.



“Hvordan” er
softwaren
realiseret?

Inden man kan implementere et stykke software er man nødt til af finde ud af **hvad** selve software skal gøre og formulere det fra brugerens synsvinkel:

- Hvilke brugere er der?
- Hvilke begreber (engl. concepts) spiller en rolle
- Hvordan hænger begreberne sammen?
- Hvilke funktioner (use-cases/aktiviteter) er der?
Hvad indebærer de og hvornår kan man gennemføre dem?

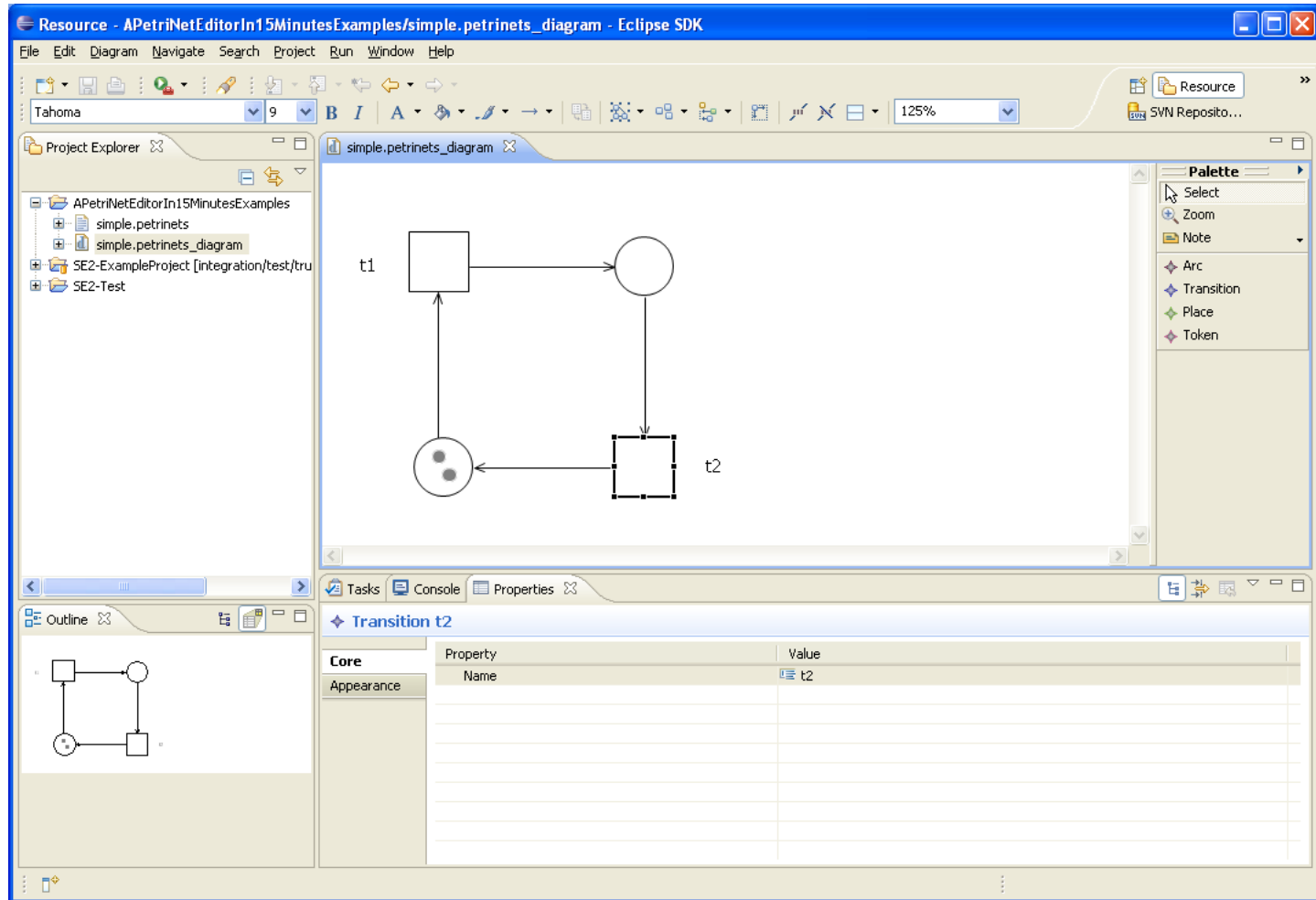
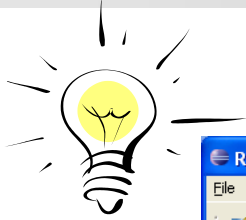
I vores projekt er det mest af
alt: RoboRally's regler

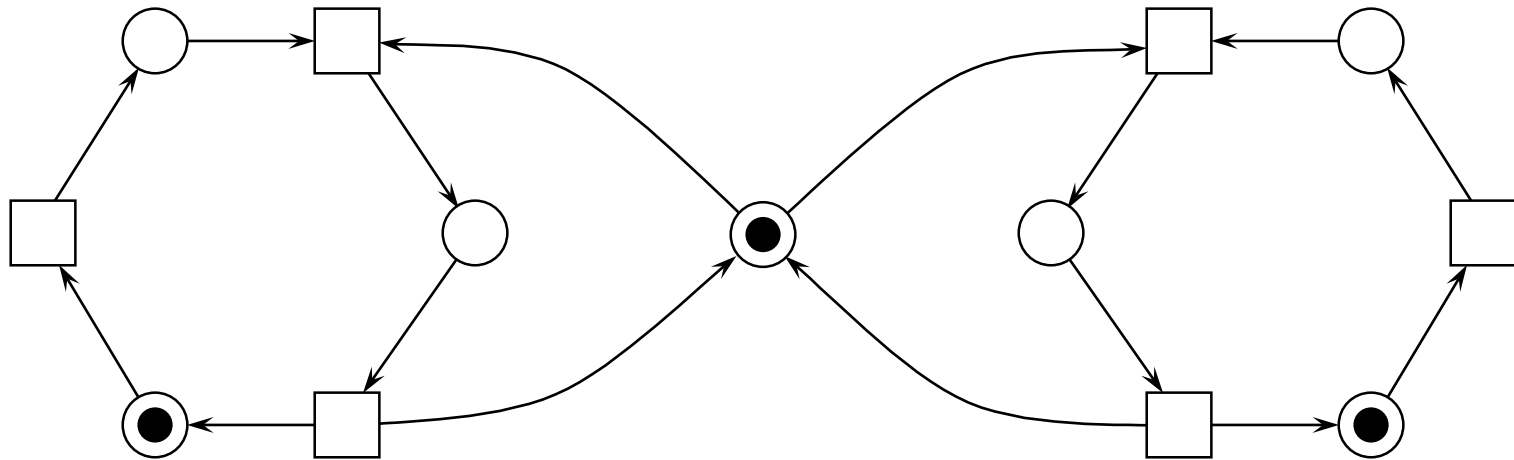
- Find skriftlig dokumentation om området (domænet) og skriv det ud
- Marker alle begreber som synes relevante
- Tal med nogle eksperter fra domænet
- Diskuter indbyrdes og saml informationerne op og strukturer den

- For at samle informationerne op give dem struktur:
 - **Taksonomi**: liste af begreber (ord)
 - **Glossar**: Beskrivelse af begreber deres egenskaber og hvordan de forholder sig til hinanden
 - **Domænemodel**: Begreber, deres attributter og deres relation repræsenteret som klassediagram og beskrivelse af aktiviteter og begrebernes tilstande (aktivitets- og tilstandsdiagrammer)

Disse diagram har ikke noget med programmering at gøre (endnu). Tænk kun på begreber og deres relation!

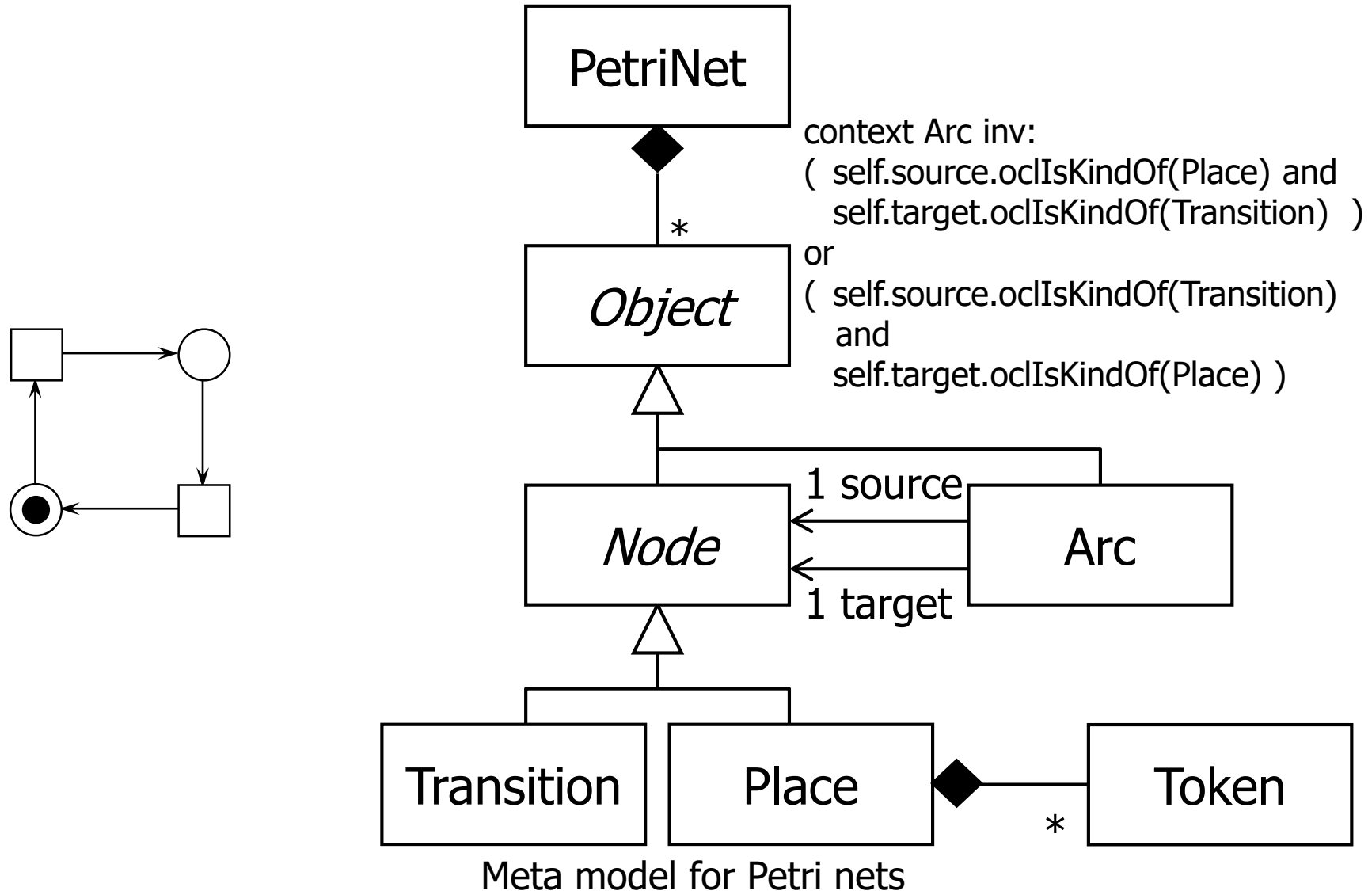
2. Eksempel: Petrinet



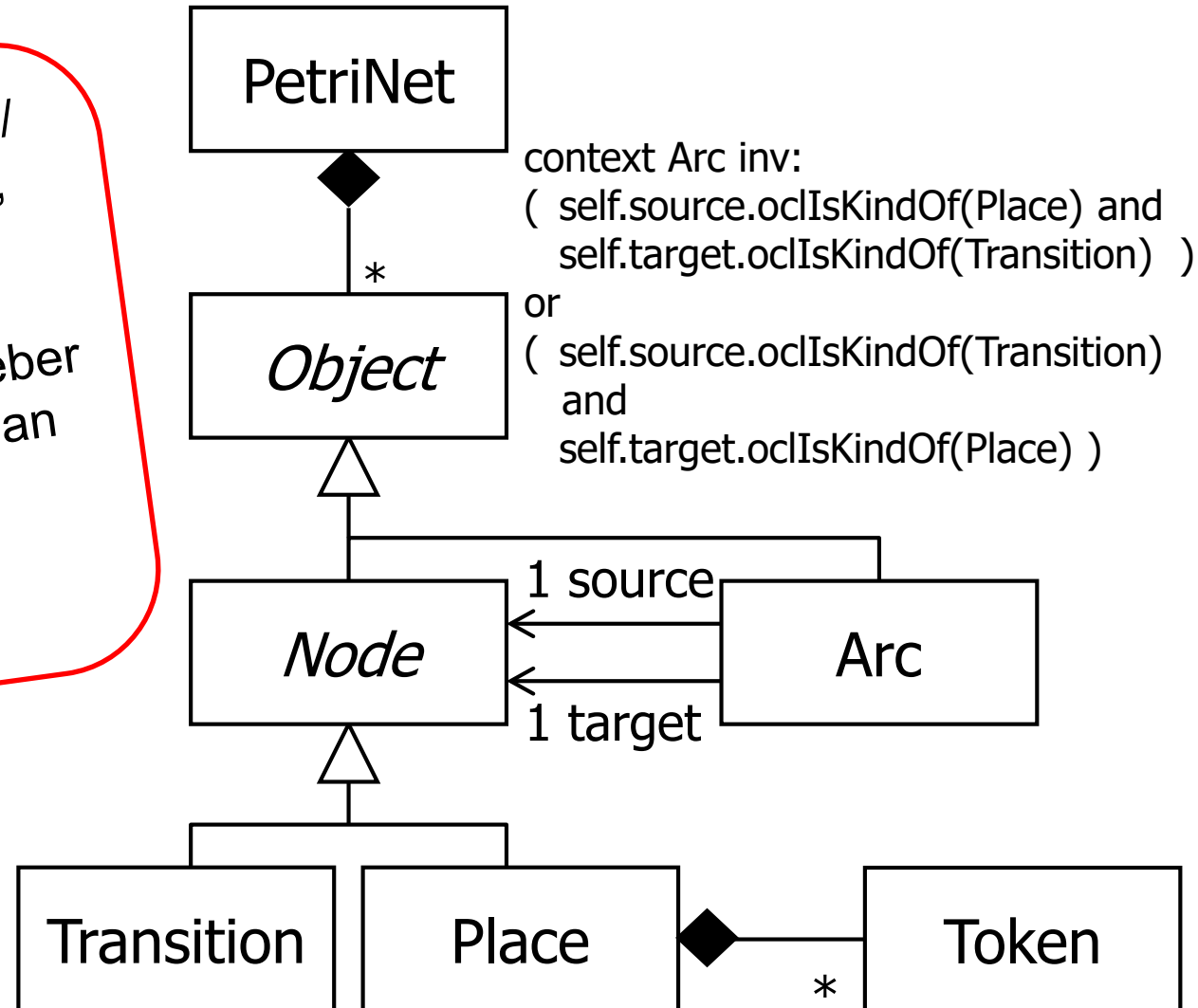


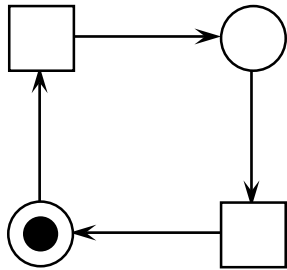
- Eksempler
- Taksonomi (på tavlen)
- Glossar
- Model (på tavlen)

Regel: I skal først starte med at lave et UML diagram, hvis I har kigget på nogle eksempler først og har en liste med hovedbegreberne (taksonomi)!

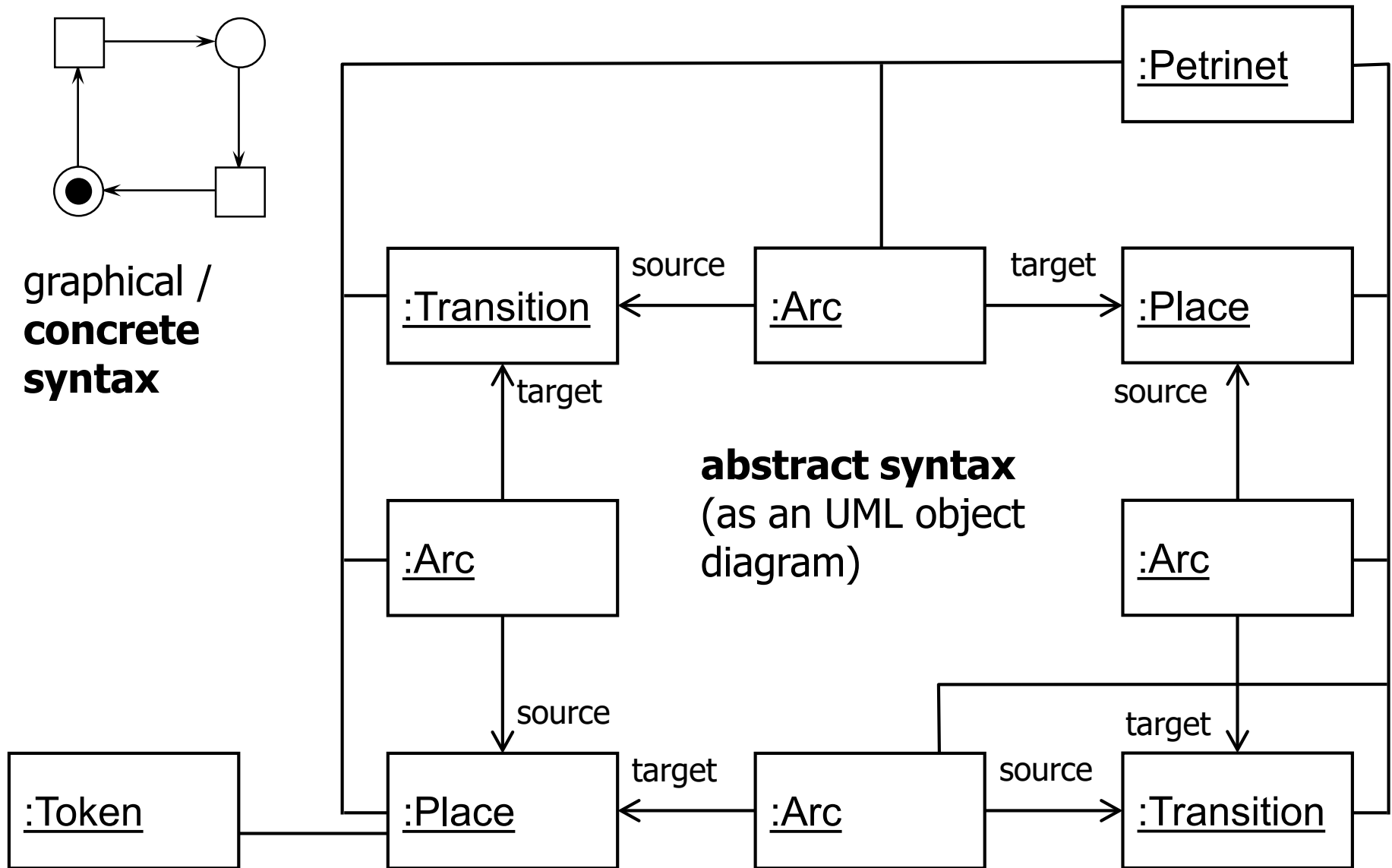


Regel: Under analysen / konceptuel modellering, tænk ikke på programmering eller Java! Det er kun begreber fra domænet og hvordan de relater til hinanden (Petrinet i vores eksempel)!

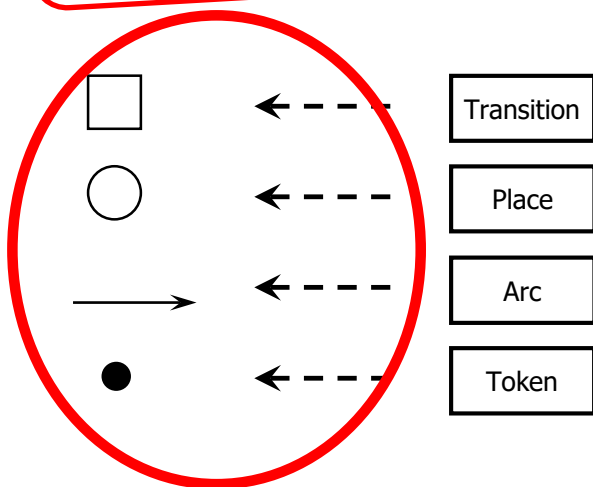




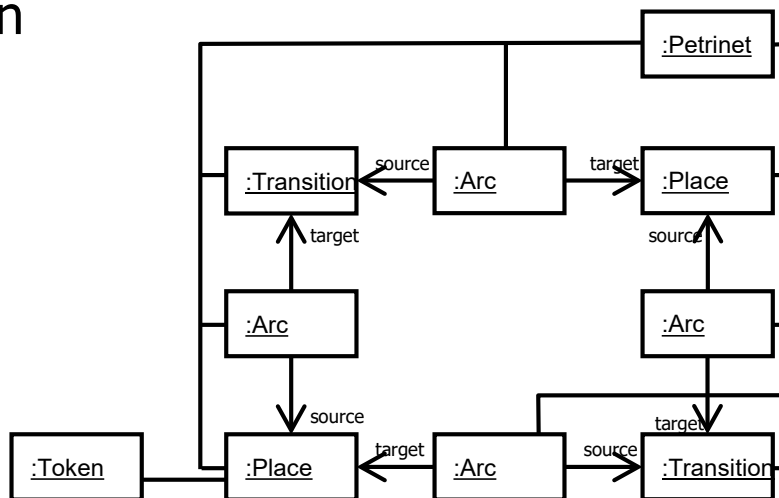
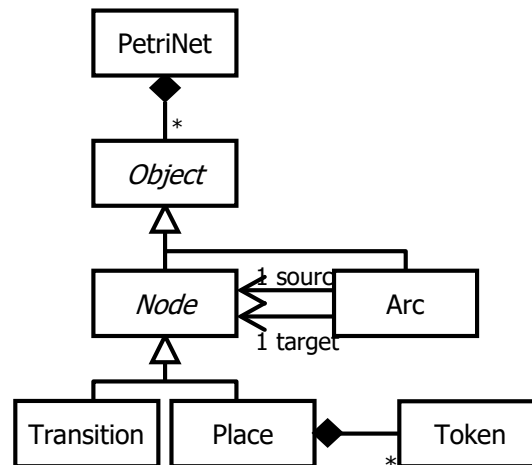
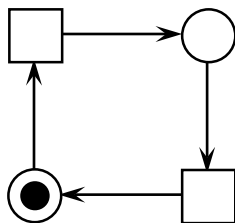
graphical / concrete syntax

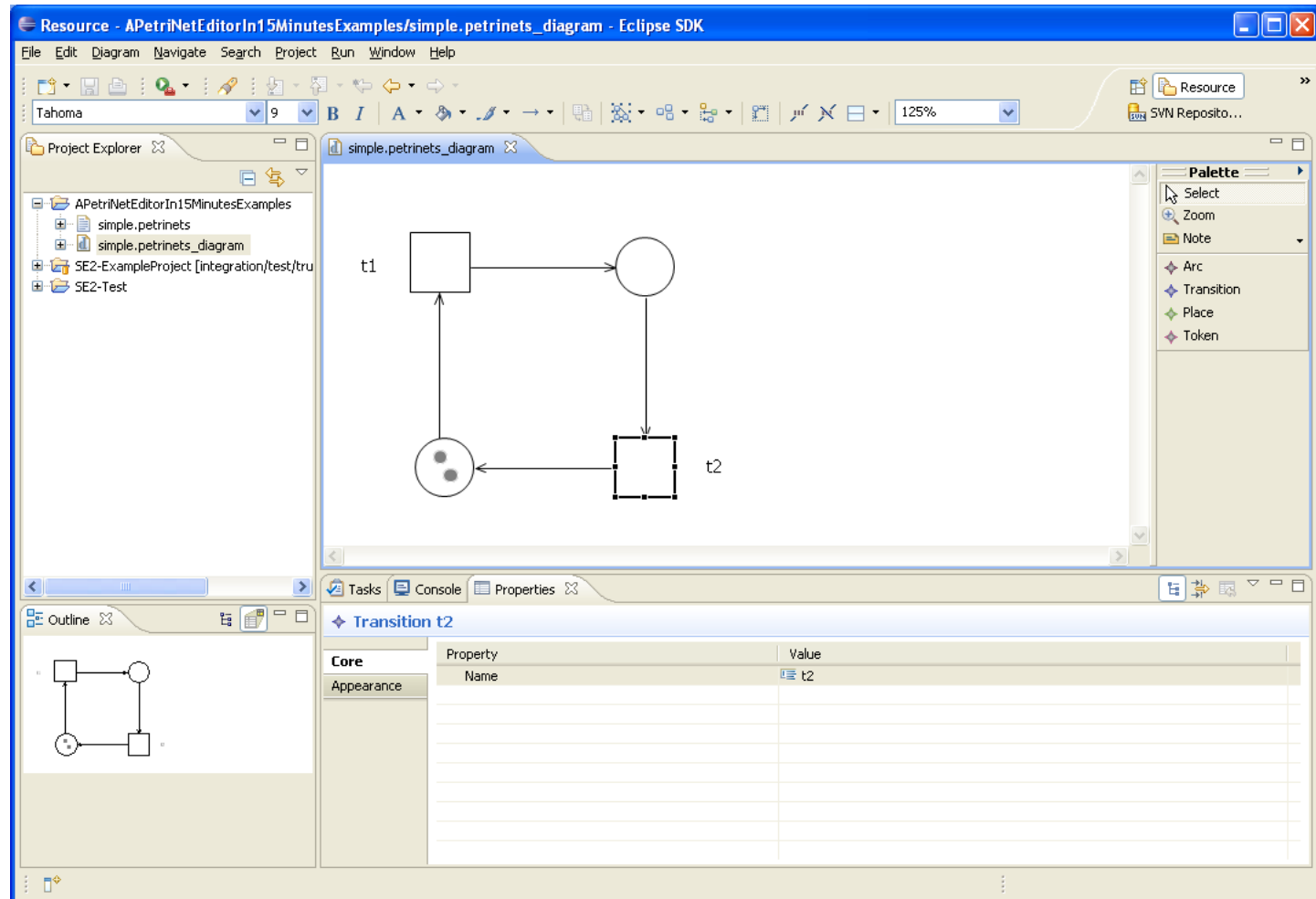


GMF = Teknologi til automatisk generering af editorer (men ikke relevant i dette kursus)!



generate an editor





Vi gennemgår processen med hjælp af udtræk af vores RoboRally-projekt

1. Find beskrivelsen af RoboRally frem (se projektslides og især links til RoboRally's regler)
2. Taksonomi
3. Glossar
4. Modeller (klassediagrammer, tilstandsdiagrammer, aktivitetsdiagrammer)

Det springer vi over her, men det kan gerne være en del af jeres rapport!

Diskuter i jeres i gruppe!

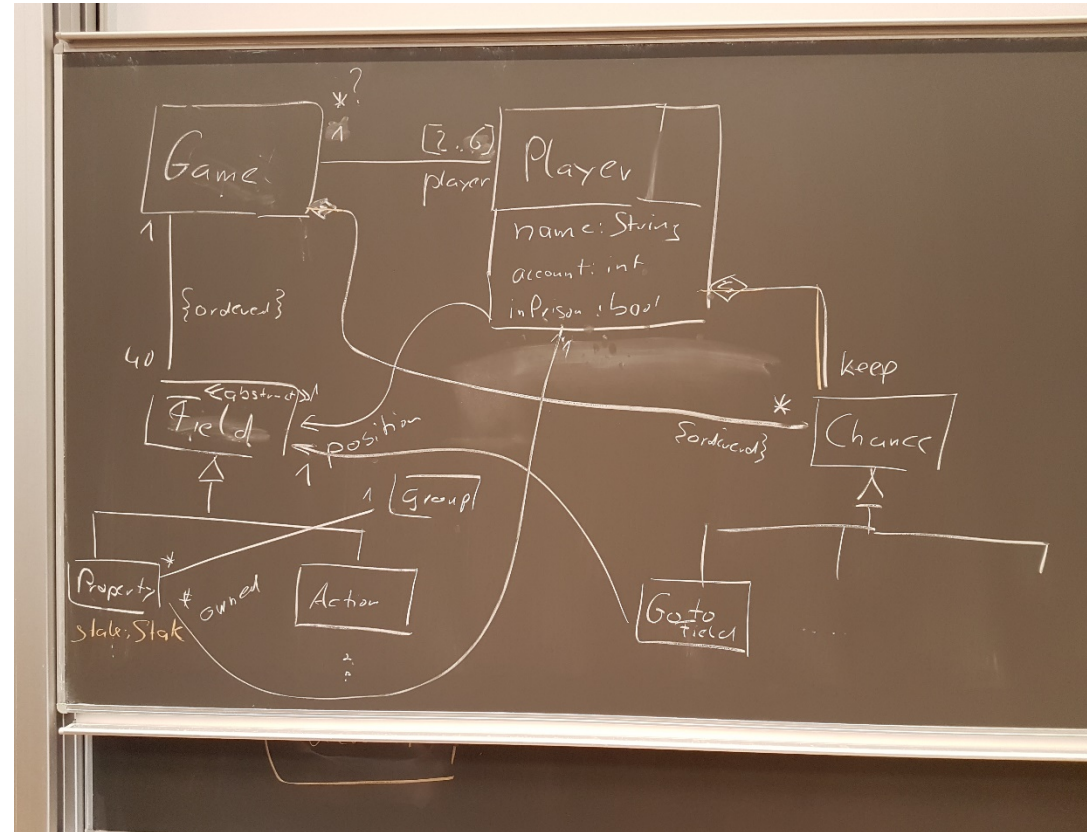
Senere under live-delen :
diskussion med hele klassen

Fra de tidligere år: Monopoly/Matador

Taxonomi

Telt (space/field)	ejendom (property)
Spiller (player)	brilker (token)
Plantsætning	hus
Chancekort	hotel
Chance	fængsel
Konto (par/double)	bank
terning (die dice)	winner
Spilbræt	bankerot

Køb ejendom
 Køb hus/hotel
 betal leje → hus
 fængsel
 chancelkort
 løn (salary)
 sælg grund
 property group



Vi starter med livscyklus af udvalgte objekter, da de er typisk nemmere og tættere på klassediagrammet:

- spil
- spiller (== robot ?)
- felt
- kort
- ...

Diskussion i grupper og på tavlen; vi bruger tilstandsdiagrammer (UML state machines) til det

Faktisk kunne nogle af de der tilstande være et attribut af klasser i klassediagrammet (eller de kan afledes af nogle af objektets andre attributter.

Vi diskuterer udtræk af nogle udvalgte aktiviteter:

- lav et trækk???
- ...

Diskussion i grupper og på tavlen; vi bruger aktivitetsdiagrammer (UML activity diagrams) til det

Bemærk at nogle aktiviteter bliver udført som del af nogle andre aktiviteter; og aktiviteter kan køre også sideløbende med andre aktiviteter

Ud over taksonomien og diagrammerne, skal analysen også indeholde

- Tekst som beskriver diagrammerne
- Diskussion af krav, som ikke er beskrevet igennem taksonomien og diagrammerne (ikke-funktionelle krav):
 - brugbarhed
 - performance
 - vedligeholdbarhed
 - ...
- GUI

I vores projekt kommer hoveddelen af GUI'en fra mig og det er ikke hovedfokus! Derfor kan dens beskrivelse være meget kort (udover nogle særlige features fra jer).

<http://www2.compute.dtu.dk/courses/02362/f21/opgaver/V02/>

Bemærk at opgave A1 beskrives sammen med opgave V2!

Detaljerede analyse/konceptuelle modeller til RoboRally (**afleveringsopgave**):

- Taksonomi (tilstandsbegreber / aktionsbegreber)
- Klassediagrammer
- Tilstandsdiagrammer til relevante klasser (frivilligt)
- Aktivitetsdiagrammer til relevante aktiviteter (frivilligt)

De er først påkrævet med aflevering A2: Projektdefinition

RoboRally 1.1.0 → ???

DTU Compute

Department of Applied Mathematics and Computer Science

Ekkart Kindler



```
141 private void continuePrograms() {
142     do {
143         executeNextStep();
144     } while (board.getPhase() == Phase.ACTIVATION && !board.isStepMode());
145 }
146
147 // XXX: V2
148 private void executeNextStep() {
149     Player currentPlayer = board.getCurrentPlayer();
150     if (board.getPhase() == Phase.ACTIVATION && currentPlayer != null) {
151         int step = board.getStep();
152         if (step >= 0 && step < Player.NO_REGISTERS) {
153             CommandCard card = currentPlayer.getProgramField(step).getCard();
154             if (card != null) {
155                 Command command = card.command;
156                 executeCommand(currentPlayer, command);
157             }
158             int nextPlayerNumber = board.getPlayerNumber(currentPlayer) + 1;
159             if (nextPlayerNumber < board.getPlayersNumber()) {
160                 board.setCurrentPlayer(board.getPlayer(nextPlayerNumber));
161             } else {
162                 step++;
163                 if (step < Player.NO_REGISTERS) {
164                     makeProgramFieldsVisible(step);
165                     board.setStep(step);
166                     board.setCurrentPlayer(board.getPlayer(0));
167                 } else {
168                     startProgrammingPhase();
169                 }
170             }
171         } else {
172             // this should not happen
173             assert false;
174         }
175     } else {
176         // this should not happen
177         assert false;
178     }
179 }
180
```

RoboRally 1.1.0 → ???

DTU Compute

Department of Applied Mathematics and Computer Science

Ekkart Kindler



The screenshot shows an IDE window for the RoboRally project. The left sidebar displays the project structure, including the `GameController` class. The main window shows the RoboRally game interface, which consists of a 10x10 checkered board. The board has several colored triangles representing players: a red triangle at (1,1), a blue triangle at (2,2), a green triangle at (3,3), a yellow triangle at (4,4), and a purple triangle at (5,5). Below the board, there are controls for Player 1, including a 'Program' section with buttons for 'Fwd', 'Fwd', 'Turn Left', and 'Finish Programming'. There are also 'Command Cards' and a 'Phase' indicator showing 'ACTIVATION, Player = Player 1, Step: 2'. The right sidebar shows the code for the `GameController` class, with the following visible code:

```
!board.isStepMode());

ntPlayer != null) {
{
amField(step).getCard();
);
ber(currentPlayer) + 1;
mber()) {
r(nextPlayerNumber));

layer(i 0));
```

The bottom status bar indicates the build completed successfully in 9 sec, 493 ms (8 minutes ago).

<http://www2.compute.dtu.dk/courses/02362/f21/opgaver/V02/>

I får udleveret en enkel implementering af RoboRally-spillet (udvidet fra Opgave V1):

- Tilknyt knapperne (i viewer) til de rigtige metoder i GameController
- Implementer robottens kommandoer i GameController
- Tilføj test til de ny-implementerede metoder i GameController

Flere detaljer om opgaven og løsningen i en separat præsentation