Software som Teknologi

Softwareprojekter
Udvikling og Dokumentation

Bertho Stultiens
Viden Djurs - Grenaa

VID GYMNASIER
HHX OG HTX

Computer - definition

- com·put·er (kəm-pyo͞o͞ʹtər)

  n.

  - A magical box performing miracles

Tak til Arthur C. Clarke (1917-2008)

Software Affordance

- Når jeg siger ”Software”, hvad tænker du?

Software Affordance

- Når jeg siger ”Software”, hvad tænker du?

- Program, (kilde-)kode, computer
- ”Noget jeg ikke har forstand på”
- Matematik, datalogi, nørder, hackere

Fakta

- En computer programmeres
- En computer eksekverer et program
- En computer *kan* være interaktiv
- En computer kan være (u)synlig
- Computeren er over det hele!

Computer - revolution

- Hvorfor bruges computeren overalt?
- Teknisk og virksomhedsperspektiv (*årsag*)
  - automatisering(*) - bedre, hurtigere, ensartet
- Samfundsperspektiv (*virkning*)
  - omstrukturering - byrde, ansvar, kommunikation
  - fysisk → mental (videnssamfundet)

(*) Robotter er en anden diskussion

Spørgsmål #1

- Hvad er et ”produkt”
- Er et program et ”produkt”?

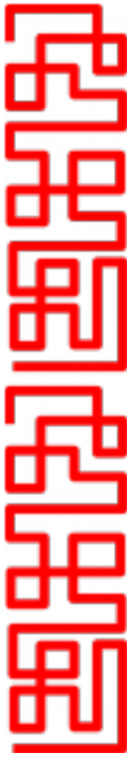
```
#!/bin/sh
echo ”Hello World!”
```

Dette er et fungerende program. Er det et produkt?

Software

- Software er abstrakt
  - en ikke håndgribelig størrelse
  - en mekanisk, dynamisk og æstetisk størrelse
  - en ide om funktion
  - en ide om funktionalitet
  - en mapning af aktion og reaktion

Kunst

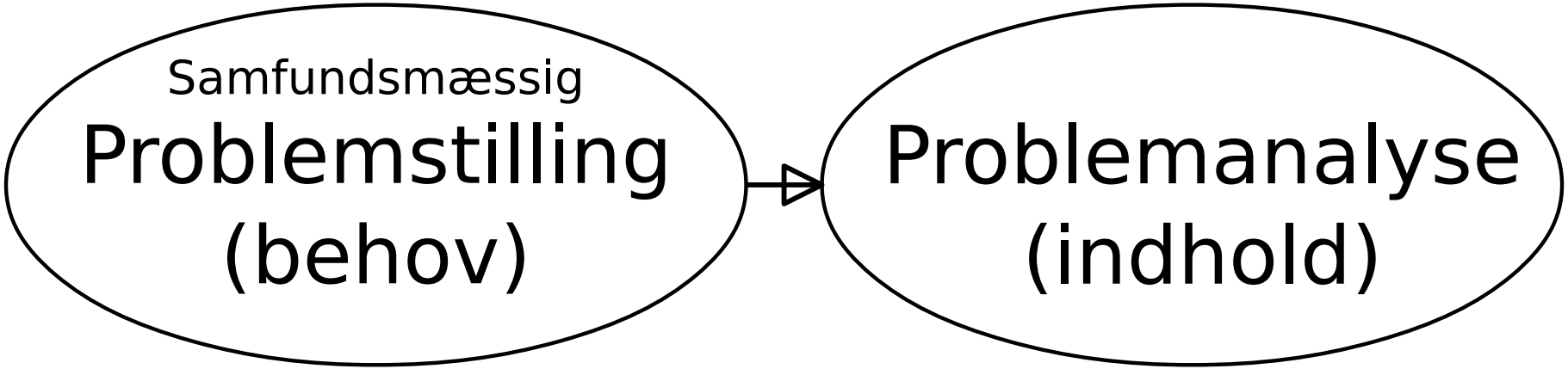


Software eller Program

- Et *program* er det, der kører på computeren
- *Software* er alt, der ligger bag programmet
  - en pakke med hele løsningen

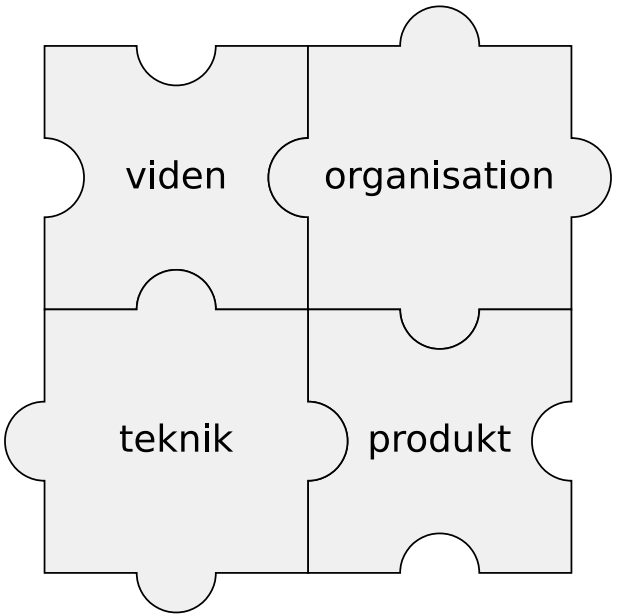
```
4   #include "utilities.h"
5
6   Game game;
7
8   int main(int argc, char **argv) {
9
10   game = Game();
11
12   game.initialize();
13  }
```

Teknologi - udgangspunkt



Software Teknologianalyse

- Hvad perspektiv tager man i analysen?
  - brugerperspektiv
  - udviklingsperspektiv
  - virksomhedsperspektiv

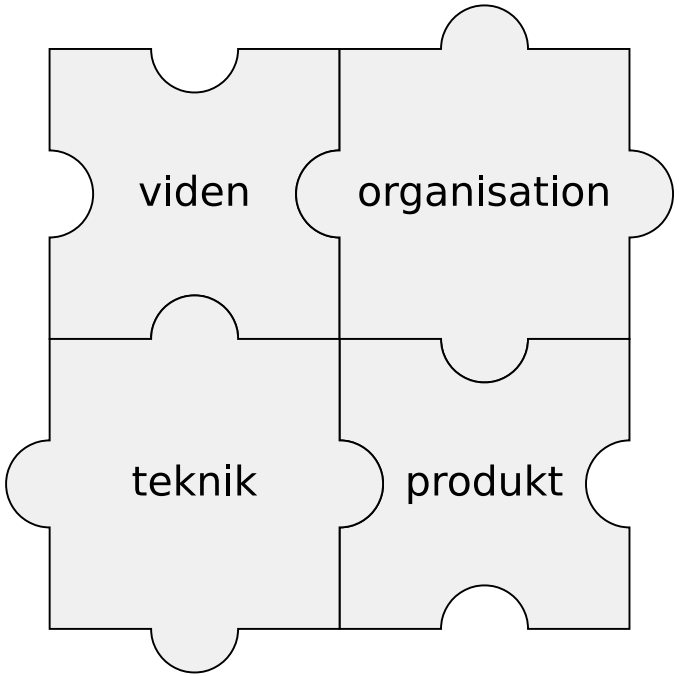


Software som produkt

brugerperspektiv

Funktionalitet

At betjene tastaturet



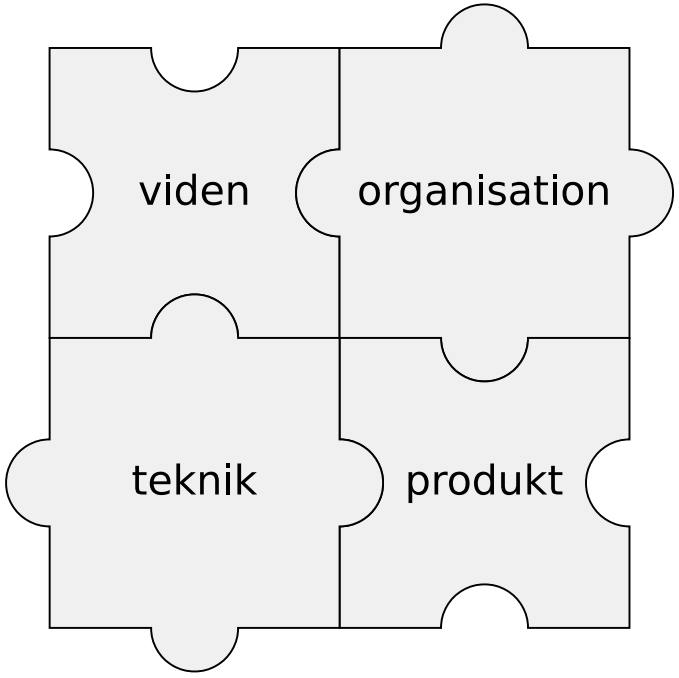
Interaktion
Kommunikation

Programmet
der løser en opgave

Software som produkt

udviklingsperspektiv

Datalogi
Matematik
Interaktionsdesign MMI/HCI



Netværk
Kommunikation
Use case
Modellering
Licensering

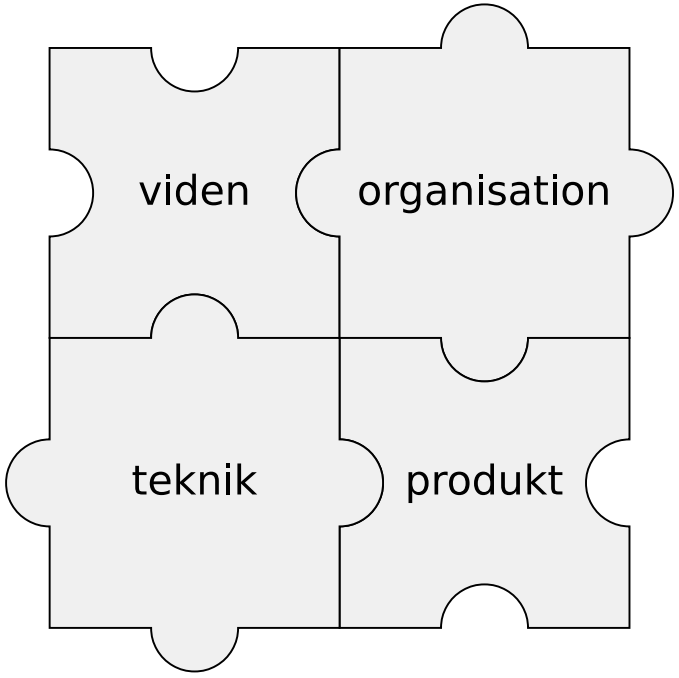
Kreativitet
At betjene tastaturet
Programmeringssprog

Programmet
der udtrykker ideen

Software som produkt

virksomhedsperspektiv

Marked
Administration
Licensering



Infrastruktur
Distribution
Kommunikation
Licensering

Infrastruktur
Administration

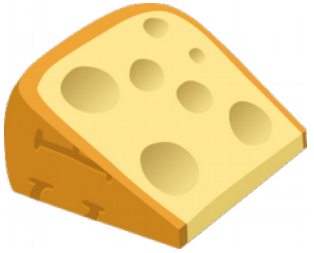
Programmet
som funktionel helhed

Hvad er produktet?

- Programmet, der løser en opgave
- Programmet, der udtrykker en ide
- Programmet, som funktionel helhed

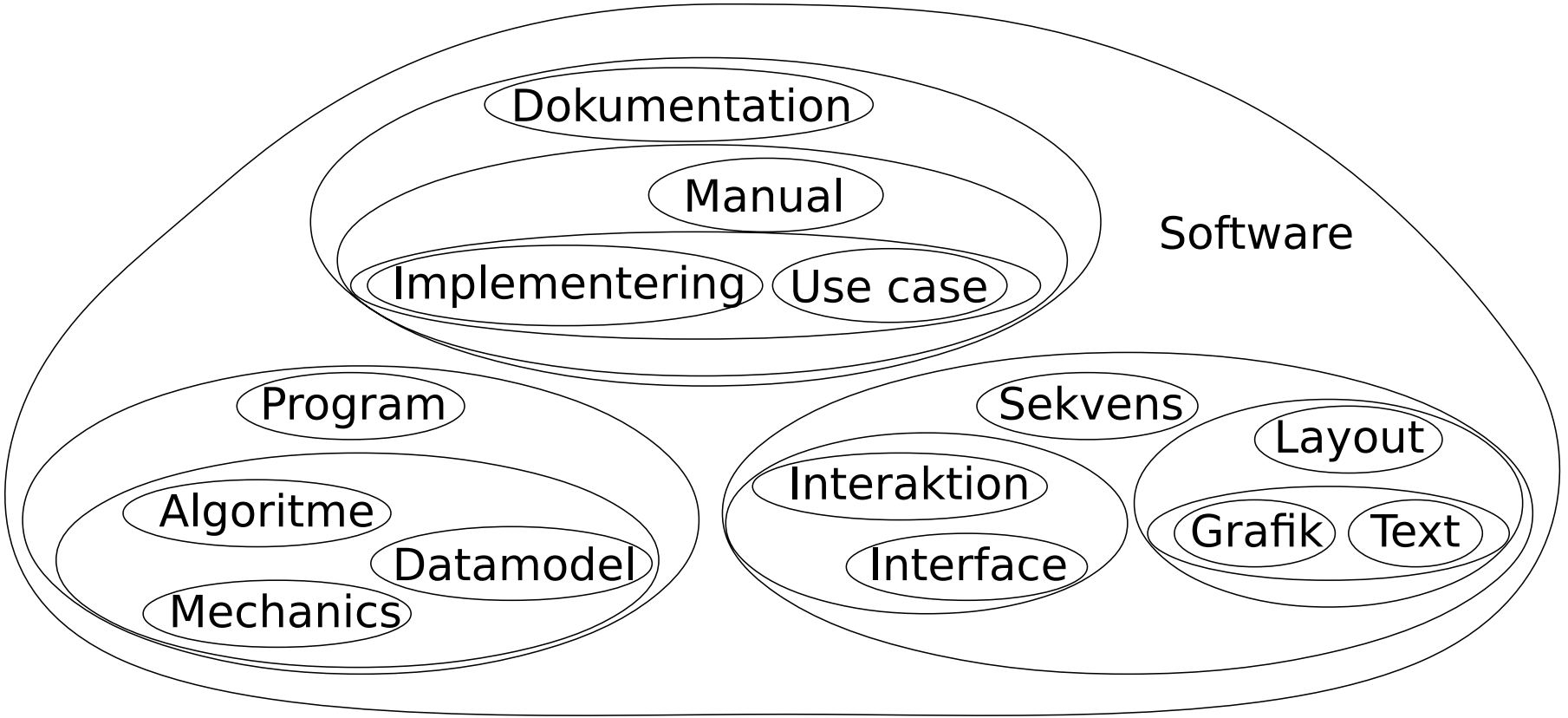
Hvad definerer et produkt

- Det løser et (fiktivt) problem
  - kan anvendes til dokumenteret formål
- Der kan argumenteres for dets eksistens
- Har en beskrevet funktionalitet og virkemåde

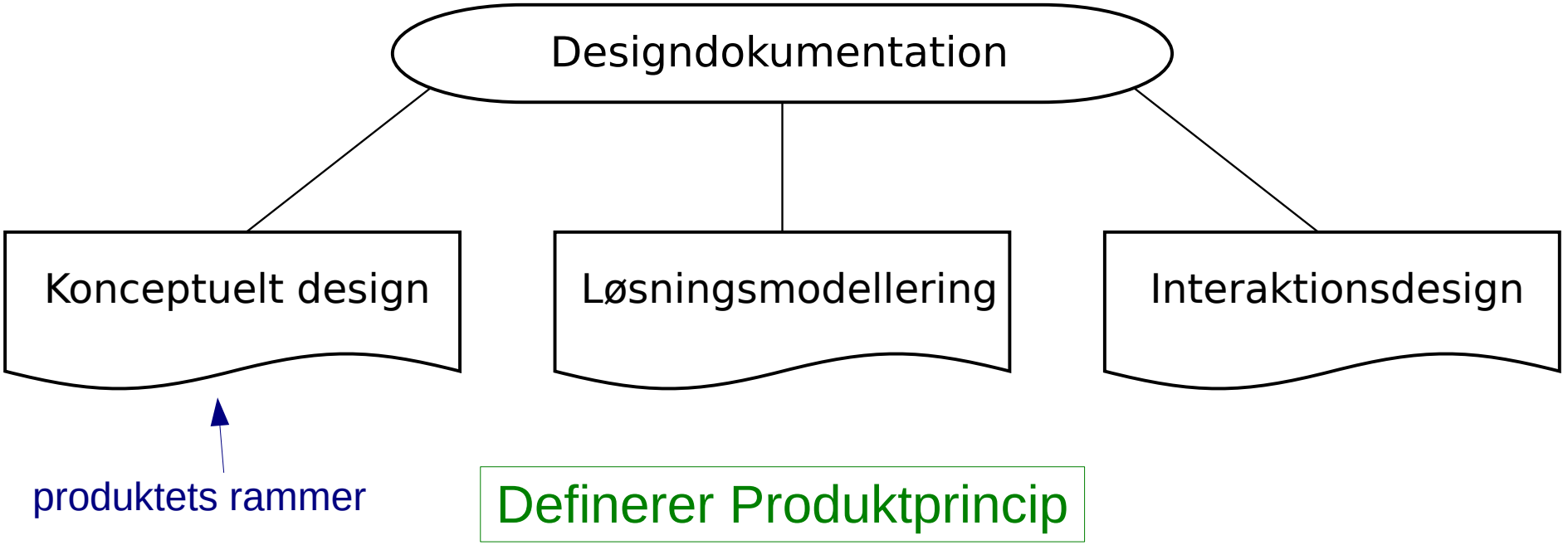


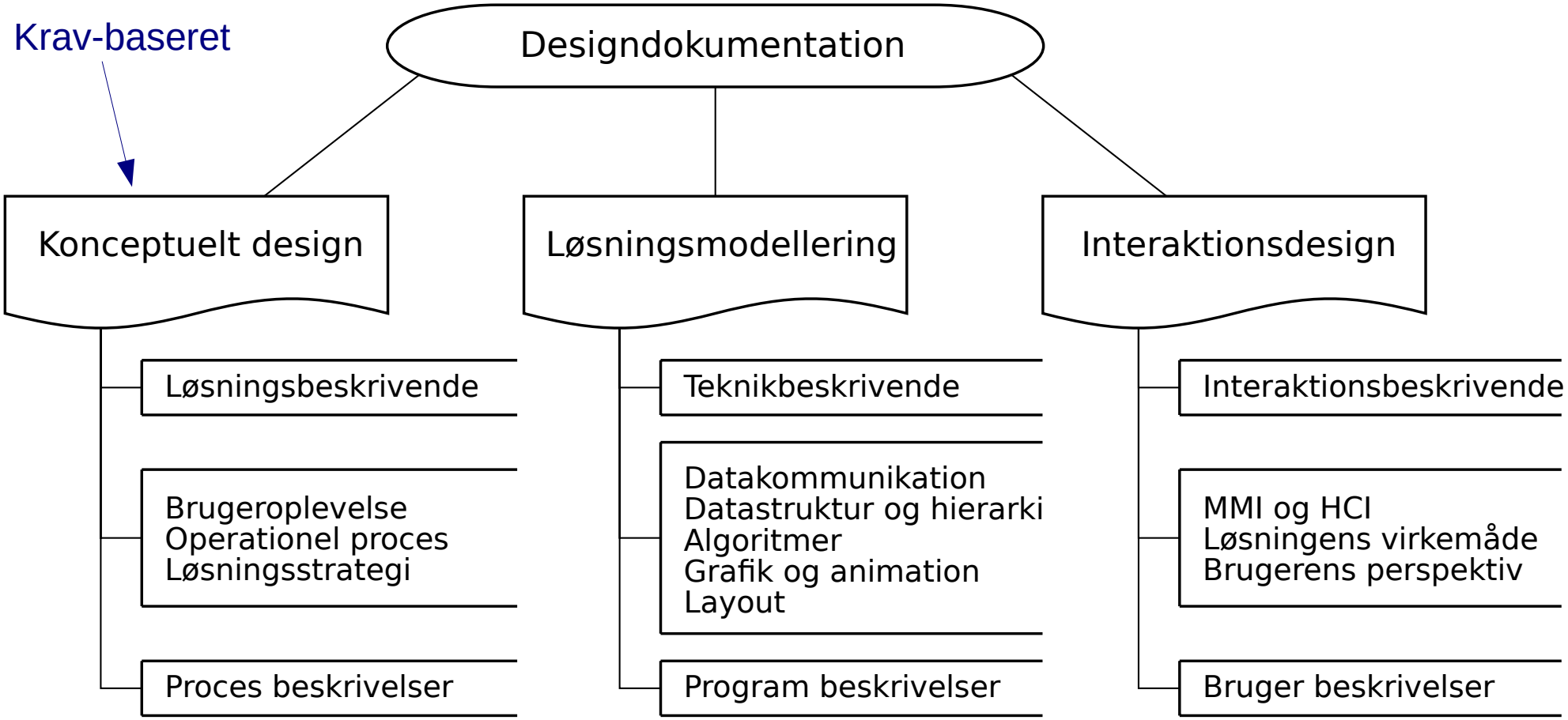
“beskrevet” kan være implicit ved hjælp af en affordance

Software Produkt (minus business)

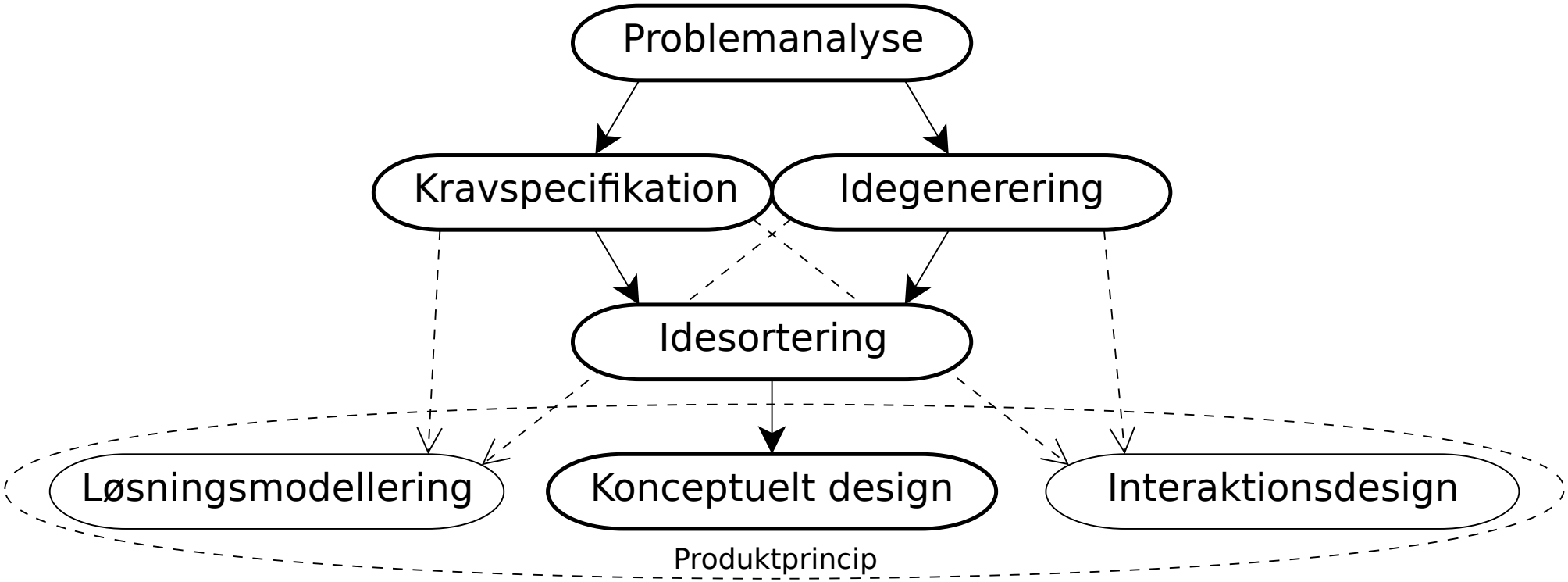


Design af Software

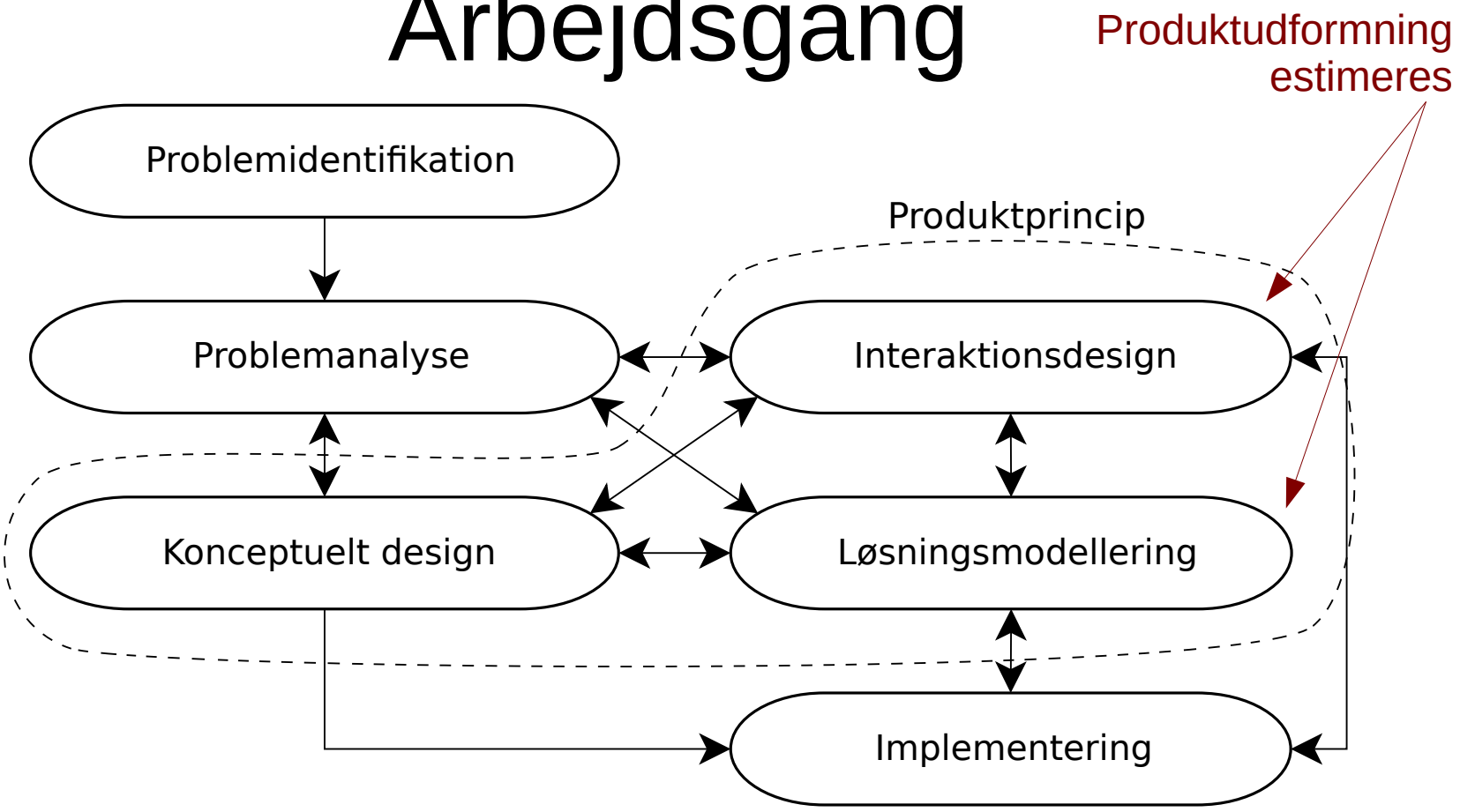



Krav-baseret
Designdokumentation
Konceptuelt design
Løsningsmodellering
Interaktionsdesign
Løsningsbeskrivende
Brugeroplevelse
Operationel proces
Løsningsstrategi
Proces beskrivelser
Teknikbeskrivende
Datakommunikation
Datastruktur og hierarki
Algoritmer
Grafik og animation
Layout
Program beskrivelser
Interaktionsbeskrivende
MMI og HCI
Løsningens virkemåde
Brugerens perspektiv
Bruger beskrivelser

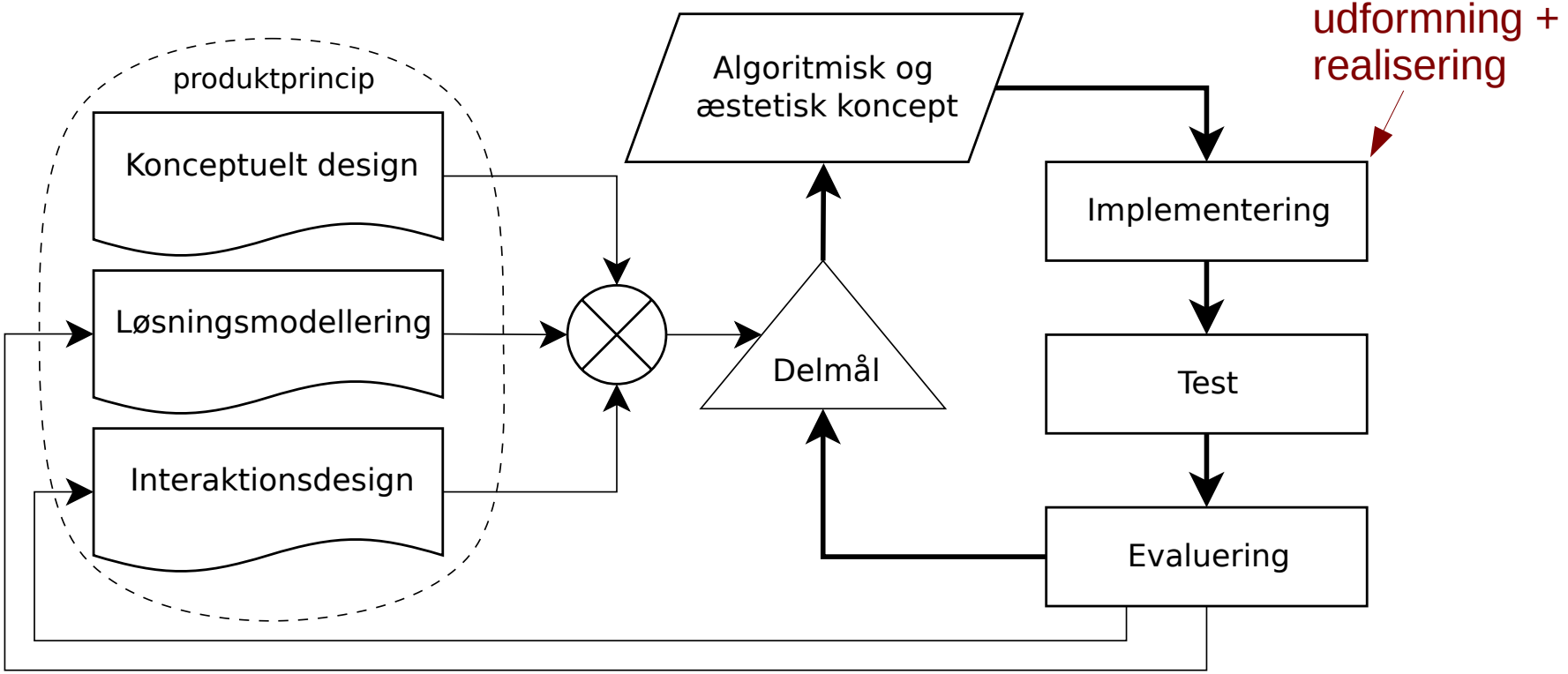
Mapning til teknologi



Arbejdsgang



Iterativ og trinvis udvikling



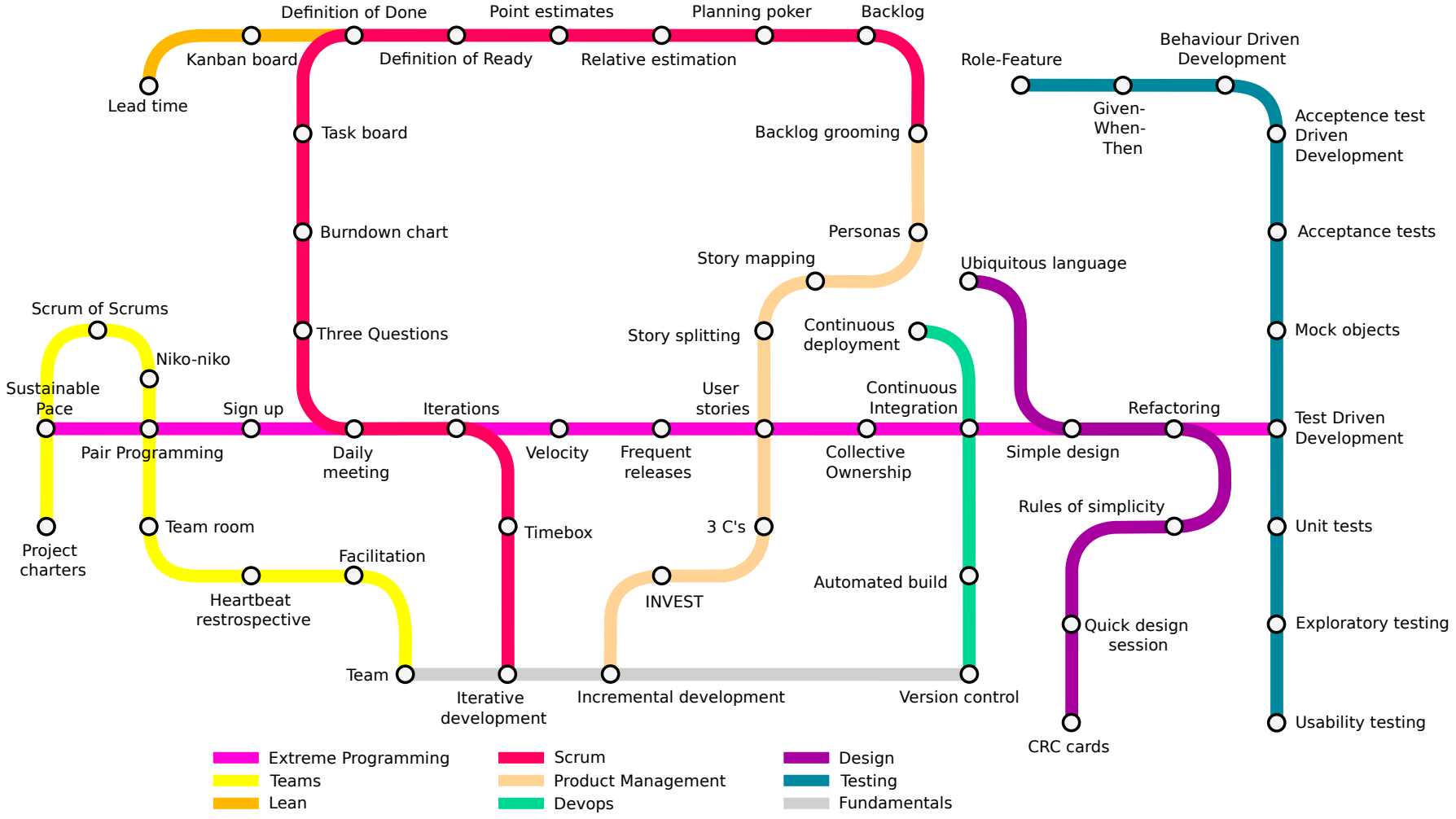
← evaluering indgår produktprincip

Agile = Agile ≠ Agile

- Agil udvikling er et *mindset*
  - der findes mange metoder under navnet
- Formål
  - at strukturere proces
  - at mappe sprog og handling
    - business- vs. teknologiske- vs. udviklings-mål

https://www.agilealliance.org/

https://www.agilealliance.org/agile101/subway-map-to-agile-practices/



Elevforudsætninger

- Vores elever er hverken dataloger eller programmører
- Det tager 10...15 år for at lære de faglige detaljer
  - det gælder for algoritmer, sprog og struktur
- Vi skal have det rigtige fokus, når vi vejleder og vurderer softwareprodukter

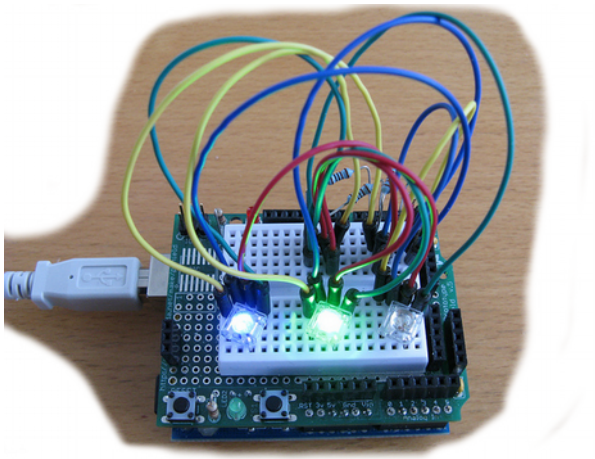
Fokusområde - 1

- Designdokumentet - konceptuelt design
  - det er nærliggende i teknologiforstand
- Skal opfylde teknologistandarden
  - traditionel teknologi
  - softwareteknologi

- Definerer produktets ramme, løsning og virkemåde

Programmeringsværksted

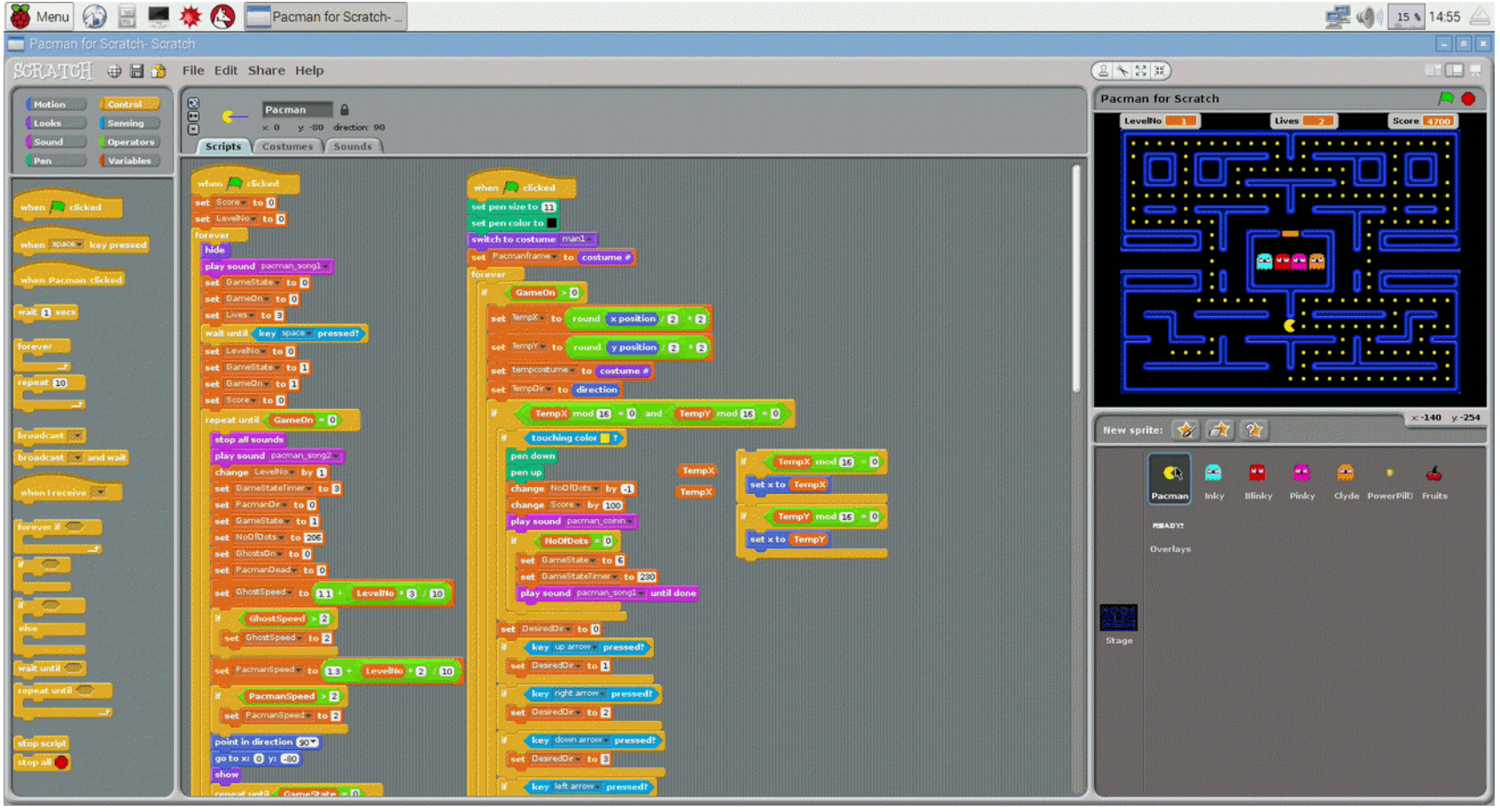
- Det er et *værksted*
- Hvad fokus har vi når eleven bruger:
  - EL-værksted?
  - træ-værksted?
  - plast-værksted?
  - ...



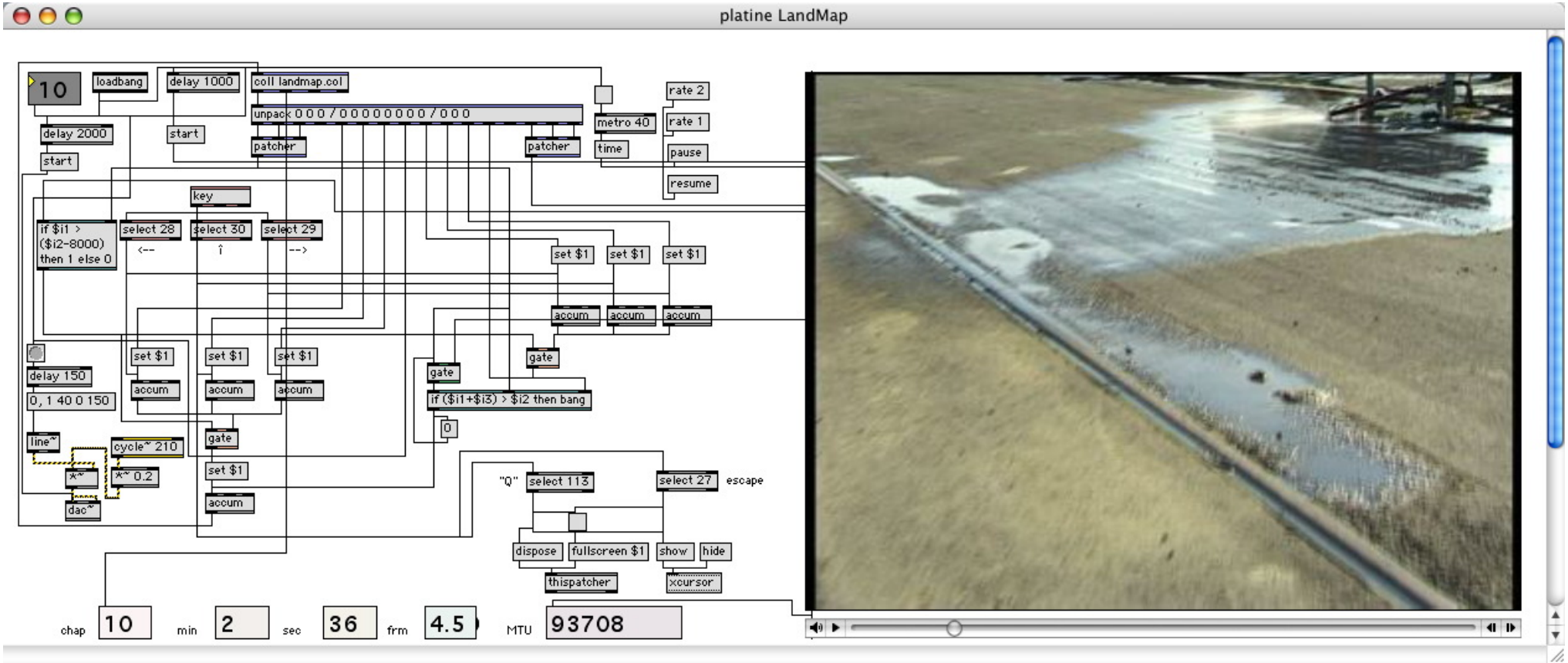
Kildekode

- Kildekode er uinteressant - en *implementeringsdetalje*
  - eleverne kan ikke programmere
    - mangler abstraktion, viden og erfaring
  - et ”lille” program fylder 10^3...10^5 linjer kode
- Programmerinssproget er ligegyldigt
  - sprog A bruger 25 linjer kode til at opnå X
  - sprog B kan opnå X med 10 billeder
  - hvilket sprog er bedre?

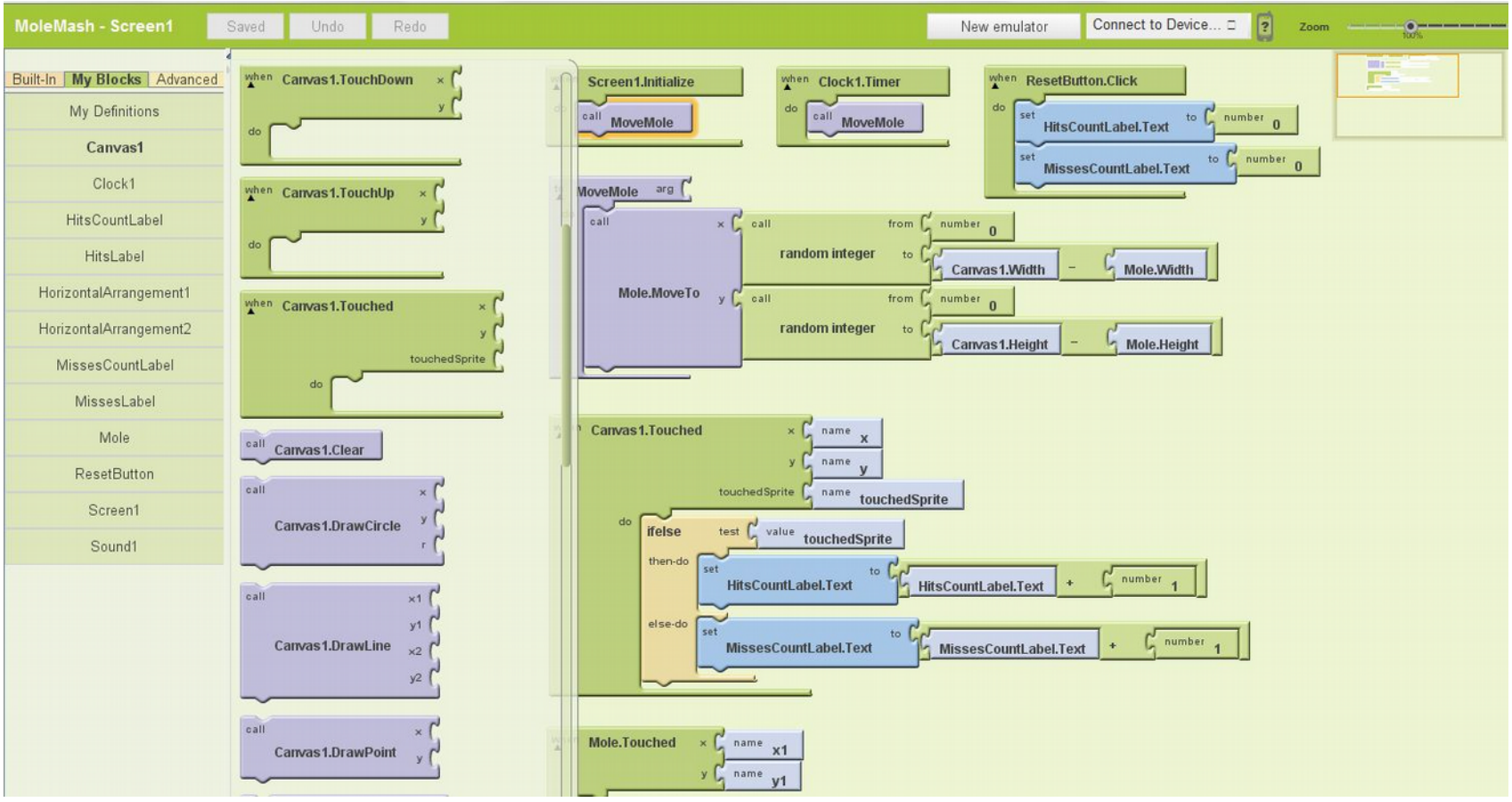
Det der med kildekode...



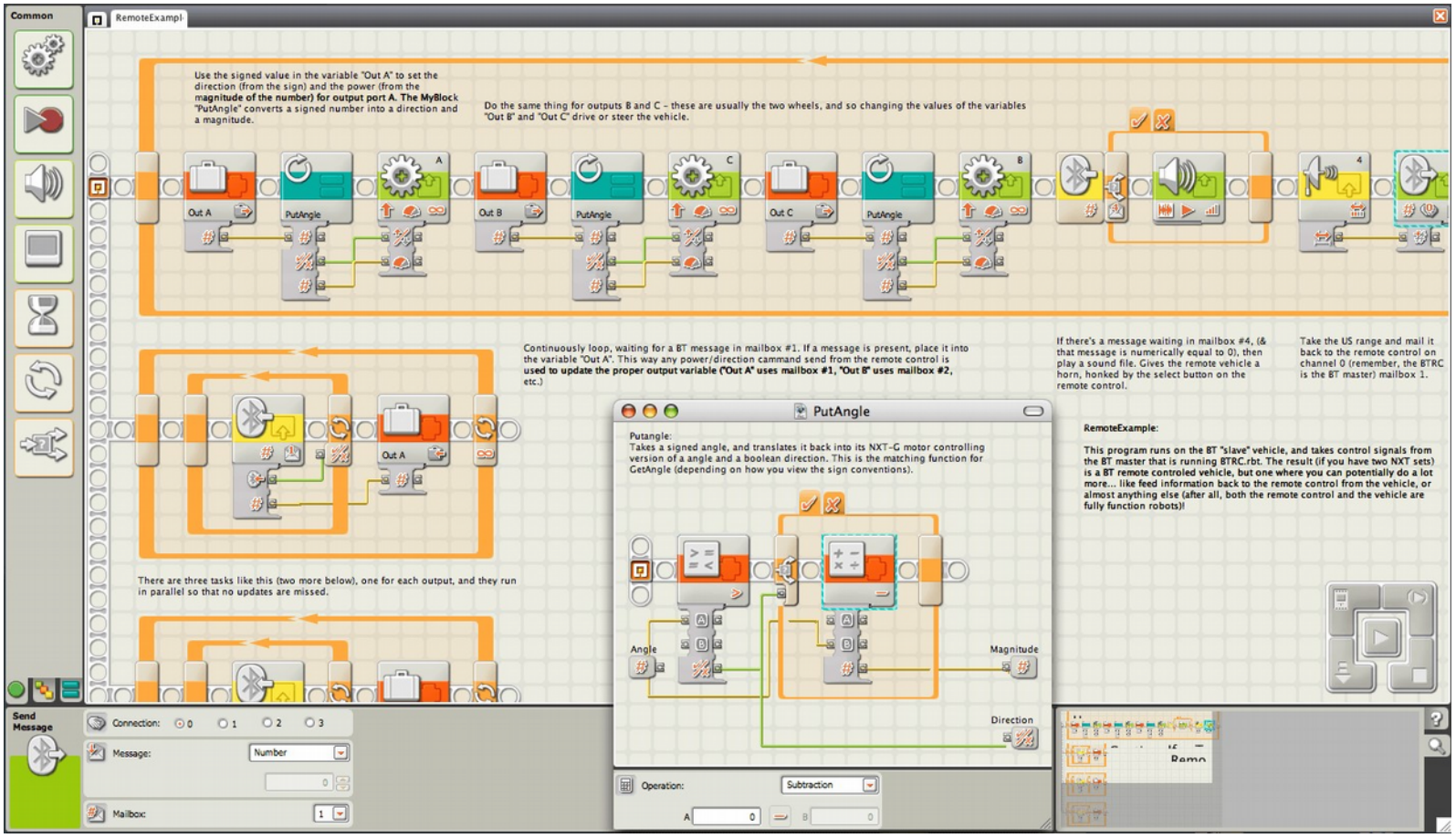
Max/MSP


platine LandMap
10
loadbang
delay 1000
coll landmap.col
unpack 0 0 0 / 0 0 0 0 0 0 0 0 / 0 0 0
delay 2000
start
start
patcher
patcher
metro 40
time
rate 2
rate 1
pause
resume
key
if $i1 > ($i2-8000) then 1 else 0
select 28
select 30
select 29
<--
^
-->
set $1
set $1
set $1
accum
accum
accum
gate
gate
delay 150
0, 1 40 0 150
set $1
set $1
set $1
accum
accum
accum
if ($i1+$i3) > $i2 then bang
0
line~
cycle~ 210
x~
x~ 0.2
dac~
gate
set $1
accum
"0"
select 113
select 27
escape
dispose
fullscreen $1
show
hide
thispatcher
xcursor
chap
10
min
2
sec
36
frm
4.5
MTU
93708

App Inventor



Mindstorm


Common
RemoteExampl
Use the signed value in the variable "Out A" to set the direction (from the sign) and the power (from the magnitude of the number) for output port A. The MyBlock "PutAngle" converts a signed number into a direction and a magnitude.
Do the same thing for outputs B and C – these are usually the two wheels, and so changing the values of the variables "Out B" and "Out C" drive or steer the vehicle.
Out A
PutAngle
Out B
Out C
Continuously loop, waiting for a BT message in mailbox #1. If a message is present, place it into the variable "Out A". This way any power/direction cammand send from the remote control is used to update the proper output variable ("Out A" uses mailbox #1, "Out B" uses mailbox #2, etc.)
If there's a message waiting in mailbox #4, (& that message is numerically equal to 0), then play a sound file. Gives the remote vehicle a horn, honked by the select button on the remote control.
Take the US range and mail it back to the remote control on channel 0 (remember, the BTRC is the BT master) mailbox 1.
There are three tasks like this (two more below), one for each output, and they run in parallel so that no updates are missed.
PutAngle
Putangle:
Takes a signed angle, and translates it back into its NXT-G motor controlling version of a angle and a boolean direction. This is the matching function for GetAngle (depending on how you view the sign conventions).
Angle
Magnitude
Direction
Operation:
Subtraction
RemoteExample:
This program runs on the BT "slave" vehicle, and takes control signals from the BT master that is running BTRC.rbt. The result (if you have two NXT sets) is a BT remote controled vehicle, but one where you can potentially do a lot more... like feed information back to the remote control from the vehicle, or almost anything else (after all, both the remote control and the vehicle are fully function robots)!
Send Message
Connection:
Message:
Number
Mailbox:

At programmere

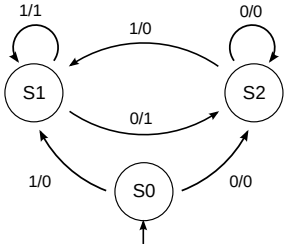
- En hierarkisk og iterativ besvarelse af
  - Hvad skal der ske
  - Hvordan skal det ske
  - Hvornår skal det ske

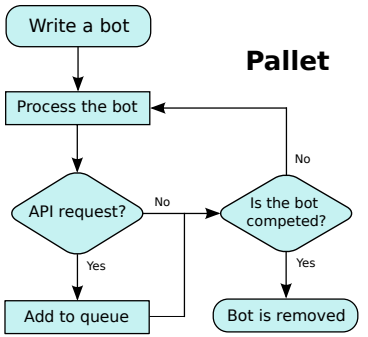
- Først derefter kan det implementeres

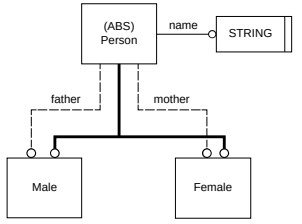
Fokusområde - 2a

- Designdokumentet - løsningsmodellering
  - det er softwarens *tekniske tegning*
- Handler om
  - datamodel og dataflow
  - programmodel og programflow
- Definerer programmets funktionalitet

↳ *Mechanics*



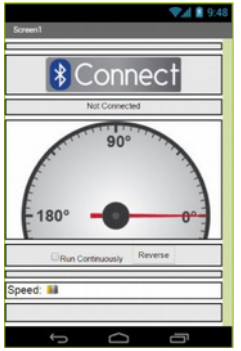




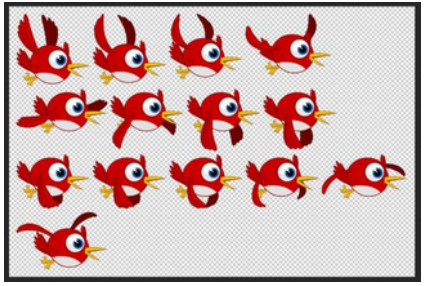
Fokusområde - 2b

- Designdokumentet - løsningsmodellering
  - det er softwarens *grafiske design*
- Handler om
  - tekst, grafik og animation
  - formidling, sanser og velbehag
- Definerer programmets *udseende*

↳*Aesthetics*



Lorum ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.



Interaktionsdesign:

- Peber eller Salt?

Don Norman (*Living with Complexity*, 2010)

Interaktionsdesign

- Design, Human Computer Interaction (HCI) og softwareudvikling

> *The problems with designing computer interfaces are fundamentally different from those that do not include software (e.g., hammers).*
>
> Alan Cooper (*The Inmates Are Running the Asylum*, 2004)

se f.eks. https://en.wikipedia.org/wiki/Interaction_design

Interaktionsdesign

- The Law of Conservation of Complexity

  “Every application has an inherent amount of irreducible complexity. The only question is:
  Who will have to deal with it — the user, the application developer, or the platform developer?”

  Larry Tesler (ca. 1984)

http://www.nomodes.com/Larry_Tesler_Consulting/Complexity_Law.html

Vi skal ”låne” IxD indhold

- Kommunikation og IT (A/C)
- Informatik (B/C)
- Design (B)
- Design og Arkitektur (B)
- Teknikfag (A) Digtalt Design og Udvikling

Fokusområde - 3

- Designdokumentet - interaktionsdesign
  - det er softwarens *use case*
- Handler om
  - input og output, aktion og reaktion
  - (forceret) intention og flow
- Definerer brugervenlighed og brugbarheden
  - user-friendly, usability, effectivity

↳ *Dynamics*

Softwareudvikling i Teknologi

- Fokus på proces 
  - iterativ, trinvis og agil
- Fokus på designdokumentet 
  - konceptuelt design, løsningsmodellering, interaktionsdesign
- Programmeringsværksted er et *værksted* 
  - skal tolkes og behandles som andre værksteder
- Tværfaglighed skal udnyttes 
  - elevforudsætninger skal tages som udgangspunkt

Tak

Penguins were used in this presentation - none got hurt

$42 = 6 \cdot 9$