Selvaag BlueThink

Kunnskapsbasert Engineering (KBE) med Common Lisp

KBE for semi-automatisk design av leiligheter og bygninger

Kristoffer Kvello

Selvaag BlueThink

Temaer

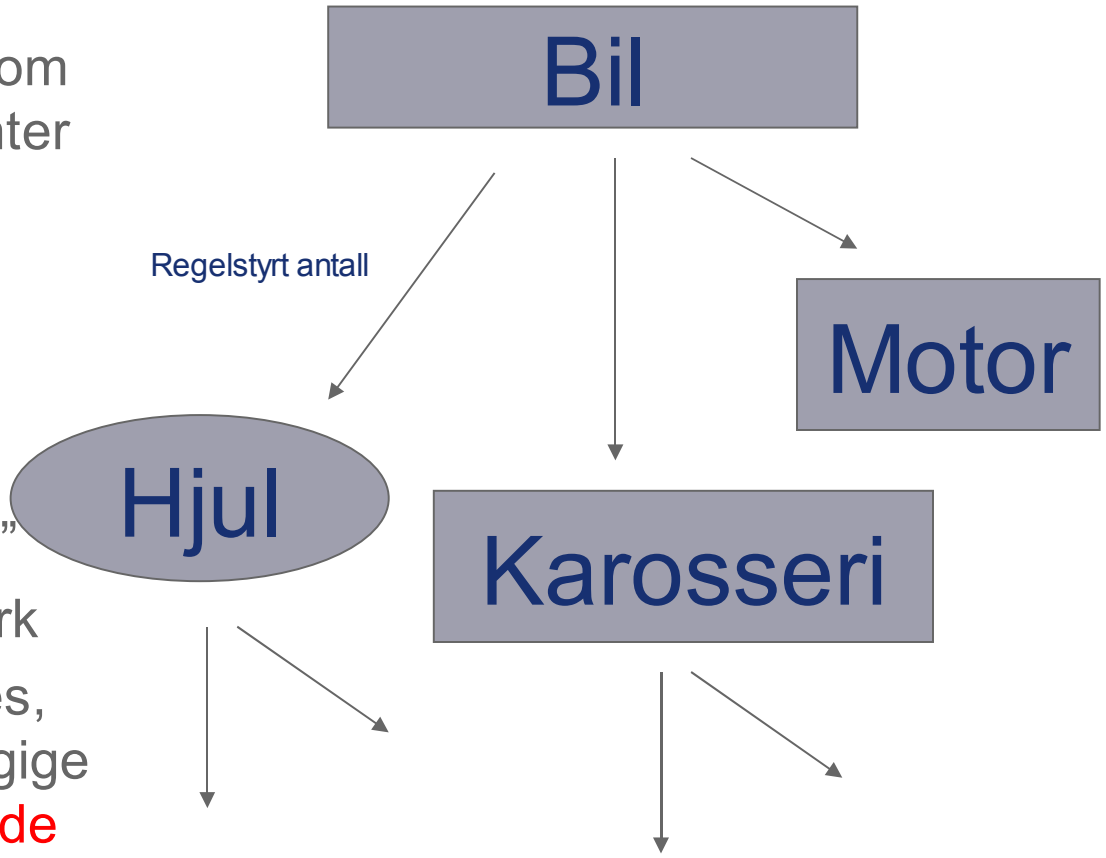
- Hva er Kunnskapsbasert Engineering?
- Lisp
- Hva bruker Selvaag BlueThink KBE til?

Hva er KBE? (Og hva er design?)

- Design:
  - En uttømmende spesifikasjon av noe som skal produseres – en liste av subkomponenter og deres posisjonering, egenskaper. En bruksanvisning man kan bygge etter.
- Kunnskapsbasert Engineering:
  - Programmeringsteknologi og -metodologi for å utvikle programmer som gjør automatisk design basert på regler
  - Disse reglene beskriver hvordan alle vesentlige aspekter av gjenstandene arter seg under ulike forutsetninger og forhold

Kunnskapsbasert Engineering

- Gjenstanden som skal designes modelleres som et tre av subkomponenter
- Foreldre/barn relasjon
- Subkomponentenes egenskaper knyttes til andre egenskaper – i objektet selv, eller i ”forelder” eller ”søsken”
- Objektorientert regneark
- Dersom en verdi endres, reberegnes alle avhengige egenskaper – og bare de



KBE-verktøy

- KBE-verkøyet tar seg av håndteringen av avhengigheter mellom reglene
- Gir en deklarativ følelse

```
(define-attribute  INTO_SPACE (DOOR)
  (ecase ?direction
    (:left ?space_left)
    (:right ?space_right)))
```

- Dette uttrykket skaper en avhengighet mellom attributtene ’into_space’ og ’direction’
  - Skulle ’direction’ endre seg vil ’into_space’ automatisk oppdateres

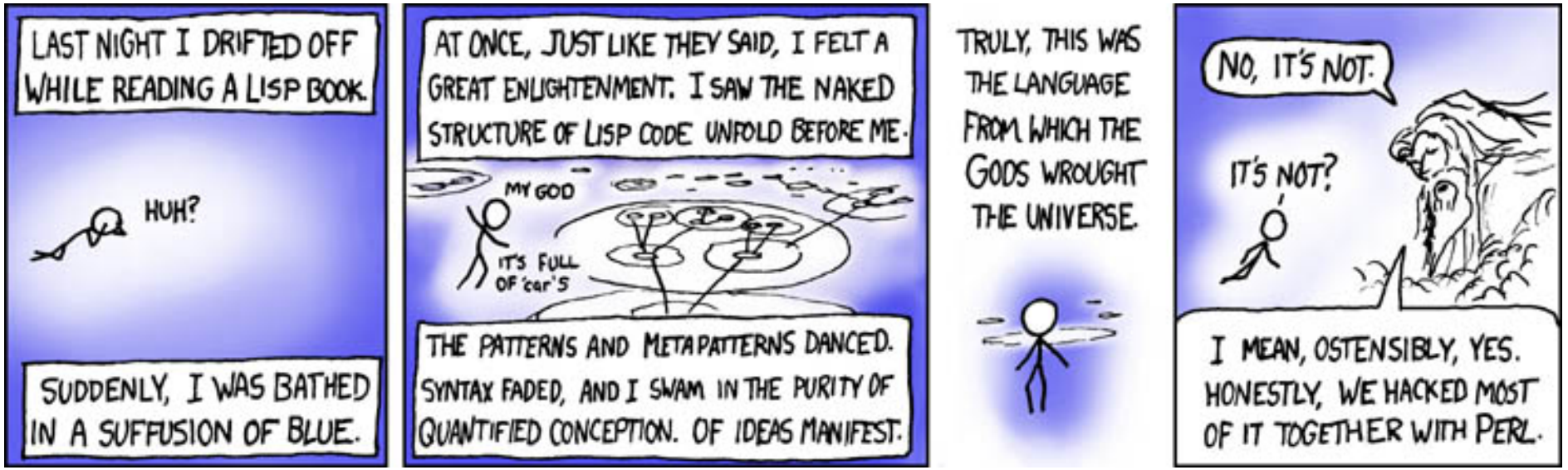
KBE-verktøy

- Altså, attributtene er memoiserte metoder, og en utregnet verdi blir stående helt til dens forutsetninger endrer seg. Når det skjer blir den automatisk reberegnet
- KBE-verktøy tilbyr mer: geometrisk funksjonalitet, kobling til CAD-systemer, men det viktigste er avhengighetshåndteringen

KBE i et nøtteskall

- Objektorientert programmeringsverktøy
- Avhengighetsmaskineri (som i et regneark)
- Modellen er organisert som et tre

Lisp


LAST NIGHT I DRIFTED OFF WHILE READING A LISP BOOK.
HUH?
SUDDENLY, I WAS BATHED IN A SUFFUSION OF BLUE.
AT ONCE, JUST LIKE THEY SAID, I FELT A GREAT ENLIGHTENMENT. I SAW THE NAKED STRUCTURE OF LISP CODE UNFOLD BEFORE ME.
MY GOD
IT'S FULL OF 'car'S
THE PATTERNS AND METAPATTERNS DANCED. SYNTAX FADED, AND I SWAM IN THE PURITY OF QUANTIFIED CONCEPTION. OF IDEAS MANIFEST.
TRULY, THIS WAS THE LANGUAGE FROM WHICH THE GODS WROUGHT THE UNIVERSE.
NO, IT'S NOT.
IT'S NOT?
I MEAN, OSTENSIBLY, YES. HONESTLY, WE HACKED MOST OF IT TOGETHER WITH PERL.

Lisp

- KBE-systemer har tradisjonelt vært basert på Common Lisp
- De tidligste systemene vokste ut av AI-aktivitetene på 70/80-tallet
- Lisp har endel viktige styrker:
  - interaktivitet – man forholder seg til en levende modell
  - makroer – gjør det lettere å skrive mer ingeniør-nære språk oppå det underliggende programmeringsspråket
  - støtte for funksjonell programmering
    - funksjoner som argumenter
    - anonyme funksjoner

Lisp

```
(defun my-member (element list)
  (if (null list)
      nil
      (if (eq element (first list))
          list
          (my-member element (rest list)))))
```

```
D++(6): (my-member 3 (list 1 2 3 4))
(3 4)
```

Lisp – interaktivitet

En leilighets areal er summen av arealet til dens indre rom

```
;;;
;;; Signature:
(define-attribute  AREA (APARTMENT)
  :RULES (:internal)
  (loop for space in ?all_spaces
      when (equal ?space.outside 'NO)
      sum ?space.area))
```

```
-1\--  ACL Idle apartment.lisp   12:58   (Common Lisp; pkg::Desi
D++(11):
D++(11): apartment
#<FRAME APARTMENT_LED.S427655 TEWT>
D++(12):
```

Lisp – interaktivitet

En leilighets areal er summen av arealet til dens indre rom

```
;;; Signature:
(define-attribute  AREA (APARTMENT)
  :RULES (:internal)
  (loop for space in ?all_spaces
      when (equal ?space.outside 'NO)
      sum ?space.area))
```

```
-1\--  ACL Idle apartment.lisp   1:00   (Common Lisp; pkg::Design+H
D++(11):
D++(11): apartment
#<FRAME APARTMENT_LED.S427655 TEWT>
D++(12): ?apartment.area
9.869284e+7
D++(13):
```

Lisp – interaktivitet

Feil! Den riktige regelen sier at man skal legge til 10%.

```
;;;
;;; Signature:
(define-attribute  AREA (APARTMENT)
  :RULES (:internal)
  (* 1.1
     (loop for space in ?all_spaces
         when (equal ?space.outside 'NO)
         sum ?space.area)))
```

```
-1\--  ACL Idle apartment.lisp    1:10    (Common Lisp; pkg::Des
D++(11):
D++(11): apartment
#<FRAME APARTMENT_LED.5427655 TEWT>
D++(12): ?apartment.area
9.869284e+7
D++(13):
```

Lisp – interaktivitet

Den endrede regelen rekompileres, og modellen oppdateres umiddelbart. Kun berørte attributter reberegnes.

```
;;;
;;; Signature:
(define-attribute  AREA (APARTMENT)
  :RULES (:internal)
  (* 1.1
     (loop for space in ?all_spaces
         when (equal ?space.outside 'NO)
         sum ?space.area)))
```

```
-1\--  ACL Idle apartment.lisp    1:21    (Common Lisp; pk
D++(11):
D++(11): apartment
#<FRAME APARTMENT_LED.S427655 TEWT>
D++(12): ?apartment.area
9.869284e+7
D++(13): ?apartment.area
1.0856212400000001e+8
D++(14):
```

Lisp – interaktivitet

Oppsann, en feil er oppstått

```
;;;  These are spaces of other apartments that somehow end up crossing into
-1\--  ACL Idle apartment.lisp    9:25    (Common Lisp; pkg::Design++ CVS:1.406)--L1063--C27--36%-----
D++(11):
D++(11): apartment
#<FRAME APARTMENT_LED.S427655 TEWT>
D++(12): ?apartment.area
9.869284e+7
D++(13): ?apartment.area
1.0856212400000001e+8
D++(14): ?apartment.area
Error: hahaha

Restart actions (select using :continue):
 0: Return to Top Level (an "abort" restart).
 1: Abort entirely from this (lisp) process.
[1] D++(15): :rs
#<FRAME APARTMENT_LED.S427655 TEWT> ALL_SPACES
#<FRAME APARTMENT_LED.S427655 TEWT> AREA
[1] D++(16):
```

Stack'en viser hvilket attributt feilen oppsto i

Lisp – interaktivitet

Den sannsynlige feilen er identifisert

```
;;;   List of all spaces in the apartment.
;;;
;;; Signature:
(define-attribute  ALL_SPACES (APARTMENT)
  :RULES (:internal)
  (remove-duplicates
   (append ?basic_spaces          ;; spaces directly ins
           ?subspaces             ;; spaces instantiated
           ?common_spaces         ;; spaces instantiated
           (error "hahaha")
           ?subapartment_spaces   ;; spaces instantiated
           ?installation_wall_spaces ;; spaces instanti
           )))
```

Lisp – interaktivitet

Feilen repareres og attributtet rekompileres

```
;;; Signature:
(define-attribute  ALL_SPACES (APARTMENT)
  :RULES (:internal)
  (remove-duplicates
   (append ?basic_spaces          ;; spaces directly instantia
           ?subspaces             ;; spaces instantiated by ba
           ?common_spaces         ;; spaces instantiated exter
   ;;        (error "hahaha")
           ?subapartment_spaces   ;; spaces instantiated in su
           ?installation_wall_spaces ;; spaces instantiated b
           )))
```

Lisp – interaktivitet

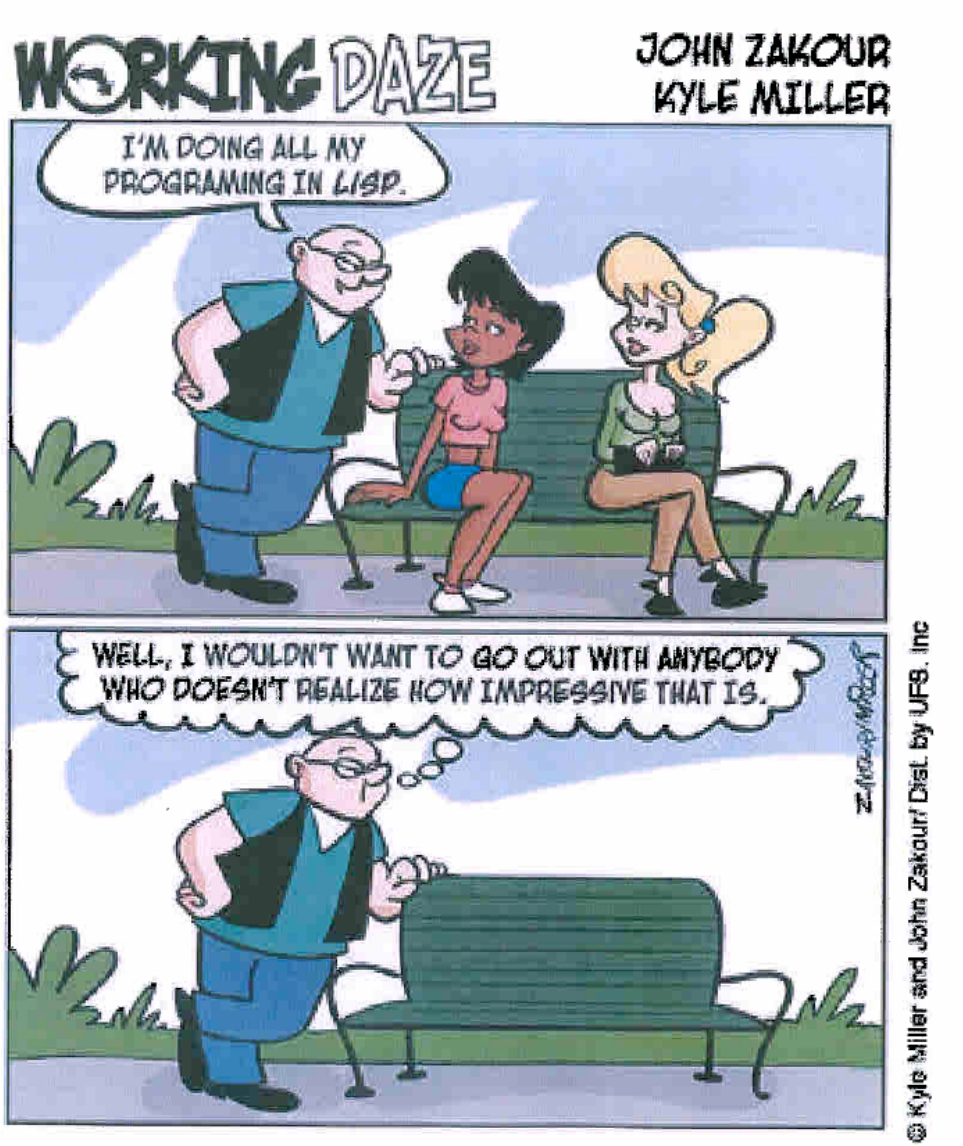
```
;;; Signature:
(define-attribute  ALL_SPACES (APARTMENT)
  :RULES (:internal)
  (remove-duplicates
   (append ?basic_spaces          ;; spaces directly instantiated b
           ?subspaces             ;; spaces instantiated by basic-s
           ?common_spaces         ;; spaces instantiated externally
   ;;        (error "hahaha")
           ?subapartment_spaces   ;; spaces instantiated in sub-ape
           ?installation_wall_spaces ;; spaces instantiated by an
           )))


;;;@+
;;; Author and date: IOK 2005-10-13
;;;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
;;; Description:
;;;  These are spaces of other apartments that somehow end up cros
-1\**  ACL Idle apartment.lisp    9:32    (Common Lisp; pkg::Design++
#<FRAME APARTMENT_LED.S427655 TEWT>
D++(12): ?apartment.area
9.869284e+7
D++(13): ?apartment.area
1.0856212400000001e+8
D++(14): ?apartment.area
Error: hahaha

Restart actions (select using :continue):
 0: Return to Top Level (an "abort" restart).
 1: Abort entirely from this (lisp) process.
[1] D++(15): :rs
#<FRAME APARTMENT_LED.S427655 TEWT> ALL_SPACES
#<FRAME APARTMENT_LED.S427655 TEWT> AREA
[1] D++(16): :restart
1.0856212400000001e+8
D++(17):
```

Deretter ber jeg systemet fortsette der det slapp, og ny korrekt kode eksekveres

Lisp – interaktivitet

Man kan legge attributter til
en levende modell

```
D++(17): (define-attribute test (apartment) "Dette er en test")
TEST
D++(20): ?apartment.test
"Dette er en test"
D++(21):
```


WORKING DAZE
JOHN ZAKOUR
KYLE MILLER
I'M DOING ALL MY PROGRAMING IN LISP.
WELL, I WOULDN'T WANT TO GO OUT WITH ANYBODY WHO DOESN'T REALIZE HOW IMPRESSIVE THAT IS.
© Kyle Miller and John Zakour/ Dist. by UFS, Inc

Lisp – makroer

- Makroer er funksjoner som opererer på kildekoden
- Lar en definere egen syntaks
- Gjør det mulig (lettere) å lage egne språk

Lisp – makroer

Ny syntaks, definert i '?'-makroen

Orginal syntaks

```
D++(22): ?apartment.all_spaces[1].name
"STUE"
D++(23): '?apartment.all_spaces[1].name
(:QUERY (FIRST (:QUERY APARTMENT ALL_SPACES)) NAME)
D++(24): (:QUERY (FIRST (:QUERY APARTMENT ALL_SPACES)) NAME)
"STUE"
D++(25):
```

Lisp – makroer

```
(defpart <navn> (<superklasser>)
   :input-parameters <lisp-kode>
   :attributes <lisp-kode>
   :children <lisp-kode>
)
```

DEFPART ekspanderer til lavere-nivås kode som bruker Lisps eget objektsystem

Lisp har ikke noe begrep om et instans-tre, eller input-parametre eller barn. Dette er verktøy/konstrukter som dannes av det makro-definerte språket.

Når makroen brukes er resultatet lisp-kode fra verktøyet og sluttbruker som forenes, og så kompileres.

Lisp – makroer

- Oppsummert: i lisp har du full tilgang til kildekoden før kompilatoren ser den

Lisp – funksjonell programmering

(map 'first ((1 2) (3 4)) → (1 3)

(remove-if 'oddp (1 2 3 4) → (2 4)

```
D++(43): (mapcar (lambda (o)
                   (if (casa-attribute-exists-p o 'name)
                       ?o.name
                     ?o.icid))
                 (list apartment floor bedroom))
("A" "Floor.1" "Sov 1")
D++(44):
```

Selvaag BlueThink utvikler KBE som gjør

- Semi-automatisk design av leiligheter og bygninger
- Basert på skisser fra arkitekt/ingeniør beregner systemet de nødvendige designmessige konsekvenser
- .. og analyserer ideene gjennom en rekke rapporter
  - Kostnadsoverslag
  - Statikkberegninger
  - Energiberegninger
  - Lysberegninger

Open source Common Lisp for Linux

- editor og utviklingsmiljø: emacs
- kobling mellom emacs og lisp: slime
- lisp: sbcl

- www.cliki.net
- comp.lang.lisp
- no.it.programmering.lisp