Software Arkitektur

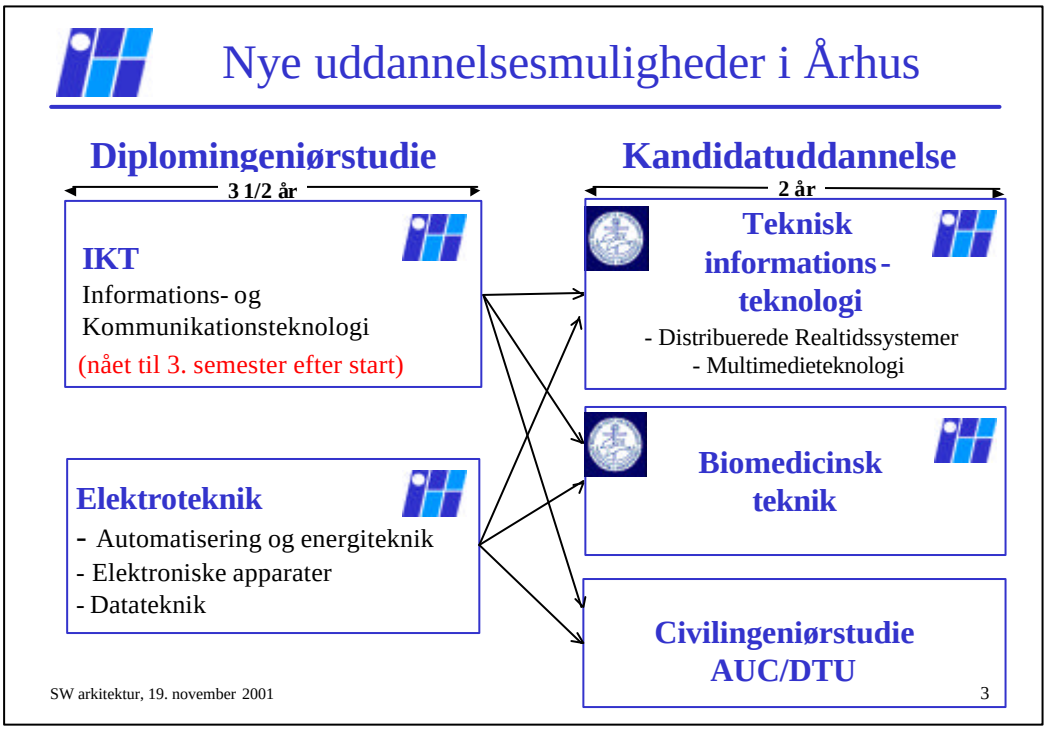
Softwareudvikling-på-tværs konference

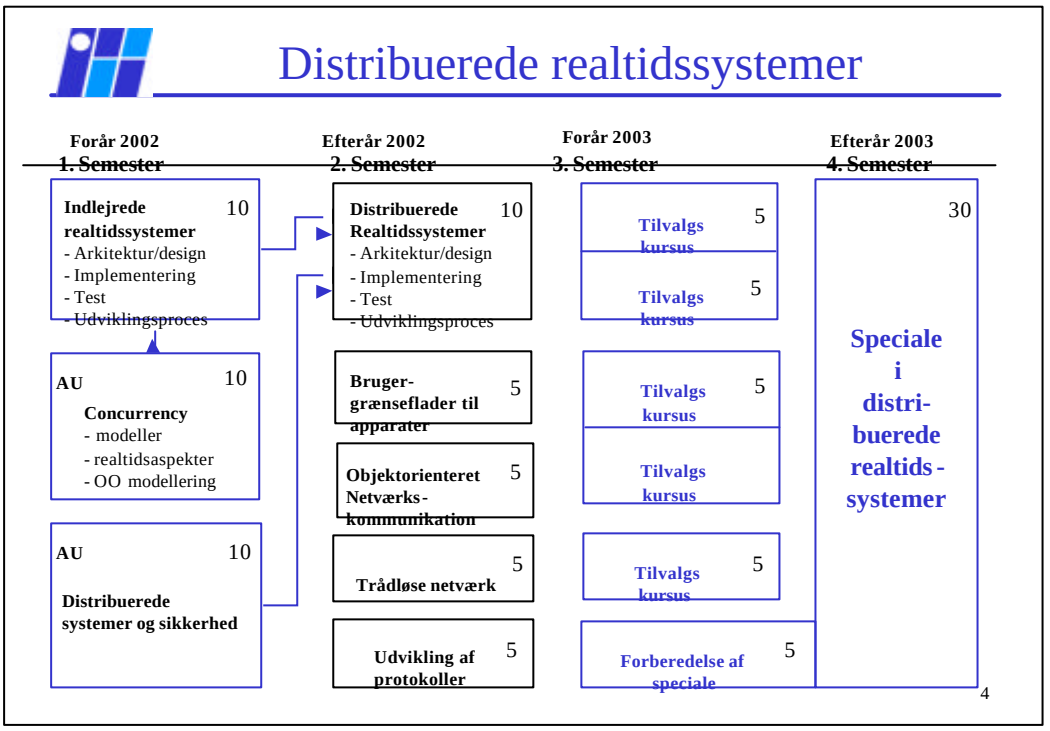
København den 19. november 2001

Finn Overgaard Hansen
Ingeniørhøjskolen i Århus
Elektro- og IKT-afdelingen
foh@e.iha.dk

Agenda

- Introduktion
  - Hvorfor er arkitektur vigtig
- Udvikling inden for SW design
- State of the Art
  - *Architectural Styles*
  - *Architectural- and Design Patterns*
  - *Frameworks*
- Dokumentation og udviklingsproces
- Opsummering


Nye uddannelsesmuligheder i Århus
Diplomingeniørstudie
3 1/2 år
IKT
Informations- og Kommunikationsteknologi
(nået til 3. semester efter start)
Elektroteknik
- Automatisering og energiteknik
- Elektroniske apparater
- Datateknik
Kandidatuddannelse
2 år
Teknisk informations-teknologi
- Distribuerede Realtidssystemer
- Multimedieteknologi
Biomedicinsk teknik
Civilingeniørstudie AUC/DTU
SW arkitektur, 19. november 2001
3


Distribuerede realtidssystemer
Forår 2002
1. Semester
Efterår 2002
2. Semester
Forår 2003
3. Semester
Efterår 2003
4. Semester
Indlejrede realtidssystemer 10
- Arkitektur/design
- Implementering
- Test
- Udviklingsproces
AU 10
Concurrency
- modeller
- realtidsaspekter
- OO modellering
AU 10
Distribuerede systemer og sikkerhed
Distribuerede Realtidssystemer 10
- Arkitektur/design
- Implementering
- Test
- Udviklingsproces
Bruger-grænseflader til apparater 5
Objektorienteret Netværks-kommunikation 5
Trådløse netværk 5
Udvikling af protokoller 5
Tilvalgs kursus 5
Tilvalgs kursus 5
Tilvalgs kursus 5
Tilvalgs kursus
Tilvalgs kursus 5
Forberedelse af speciale 5
30
Speciale i distri-buerede realtids-systemer
4

Udfordringer inden for SW udvikling

- Stigende kompleksitet
- Reduktion af *Time to Market*
- Større projekter
  - flere skal arbejde sammen
  - mere specialisering
- Fra stand alone systemer til netværk af distribuerede systemer
- Fra enkeltstående apparater til produktlinier
- WWW tilkobling (overvågning, konfigurering og opdatering)
- Sørre krav om genbrug - det er blevet for dyrt at starte forfra
- Mange nye og konkurrerende middelware teknologier
  - CORBA, DCOM, RMI, .NET, Jini, SOAP
  - Real Time udgaver på vej af RT-Linux, RT-CORBA, RT-Java

SW arkitektur, 19. november 2001 5

Definering af SW arkitektur

- Et eksempel på definition af SW arkitektur:
  - *"The software architecture of a program or computing system is the structure or structures of the system, which comprise software components, the externally visible properties of those components and the relationships among them"*
  - *Kilde: Software Architecture in Practice,* L.Bass, Addison-Wesley 1998
- Eller en pragmatisk udlægning:
  - Et systems SW arkitektur er den overordnede beskrivelse af hvorledes softwaren er organiseret i komponenter og hvorledes disse er indbyrdes forbundet
  - SW arkitektur- og designbeskrivelser udgør en SW vedligeholders vigtigste arbejdsredskab

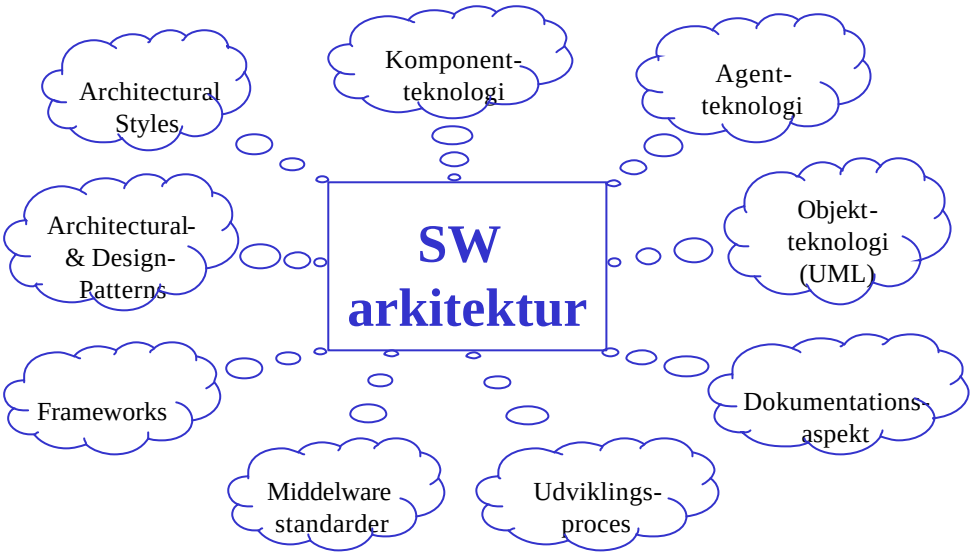
SW arkitektur, 19. november 2001 6

En længere definition på SW arkitektur

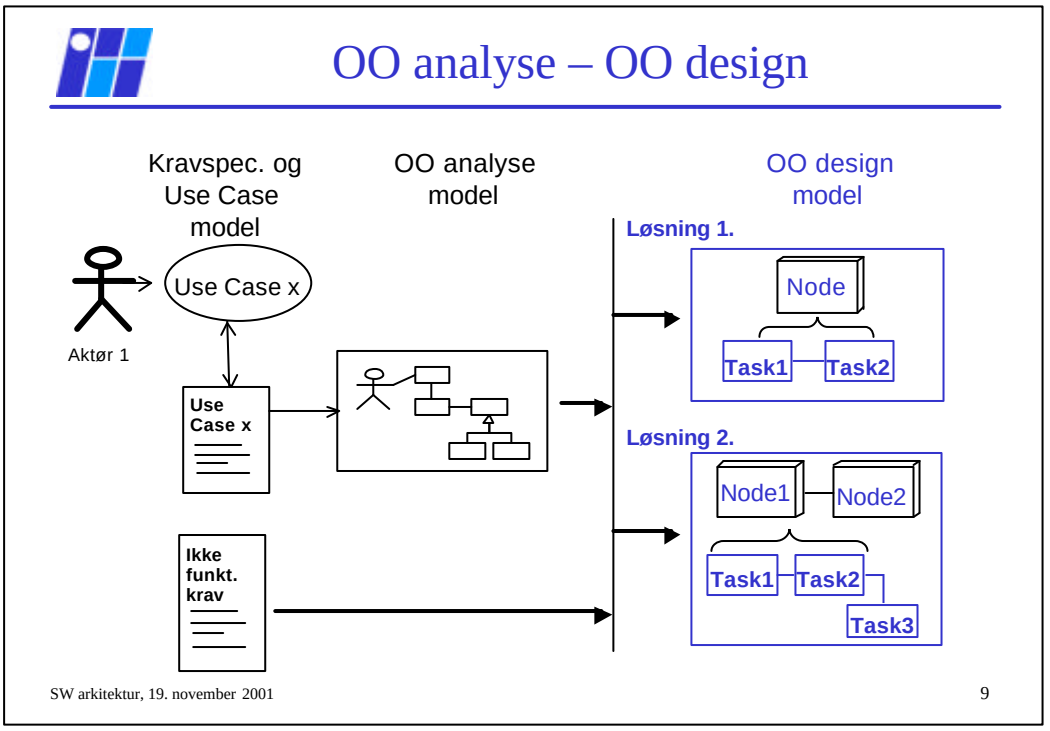
- *Software Architecture*:
  - *"is a set of concepts and design decisions about the structure and texture of sofware that must be made prior to concurrent engineering to enable effective satisfaction of architectural significant explicit functional and quality requirements and implicit requirements of the product family, the problem and the solution domains"*
  - *Kilde: Software Architecture for Product families*, Jazayeri, Addison-Wesley2000

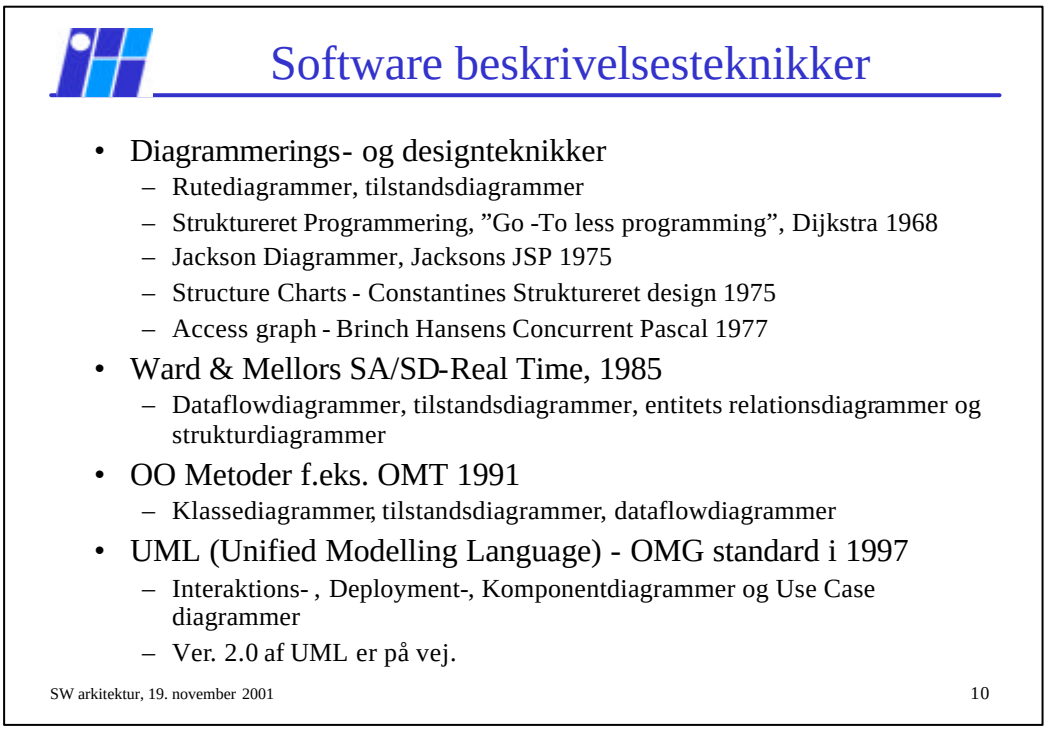
SW arkitektur, 19. november 2001 7

Kontekst for SW arkitektur



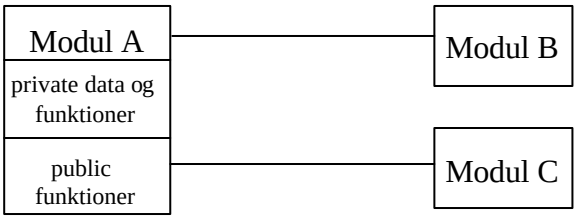
SW arkitektur, 19. november 2001 8


OO analyse – OO design
Kravspec. og Use Case model
OO analyse model
OO design model
Use Case x
Aktør 1
Use Case x
Ikke funkt. krav
Løsning 1.
Node
Task1
Task2
Løsning 2.
Node1
Node2
Task1
Task2
Task3
SW arkitektur, 19. november 2001
9


Software beskrivelsesteknikker
• Diagrammerings- og designteknikker
– Rutediagrammer, tilstandsdiagrammer
– Struktureret Programmering, ”Go -To less programming”, Dijkstra 1968
– Jackson Diagrammer, Jacksons JSP 1975
– Structure Charts - Constantines Struktureret design 1975
– Access graph - Brinch Hansens Concurrent Pascal 1977
• Ward & Mellors SA/SD-Real Time, 1985
– Dataflowdiagrammer, tilstandsdiagrammer, entitets relationsdiagrammer og strukturdiagrammer
• OO Metoder f.eks. OMT 1991
– Klassediagrammer, tilstandsdiagrammer, dataflowdiagrammer
• UML (Unified Modelling Language) - OMG standard i 1997
– Interaktions- , Deployment-, Komponentdiagrammer og Use Case diagrammer
– Ver. 2.0 af UML er på vej.
SW arkitektur, 19. november 2001
10

Udvikling inden for SW design 1.

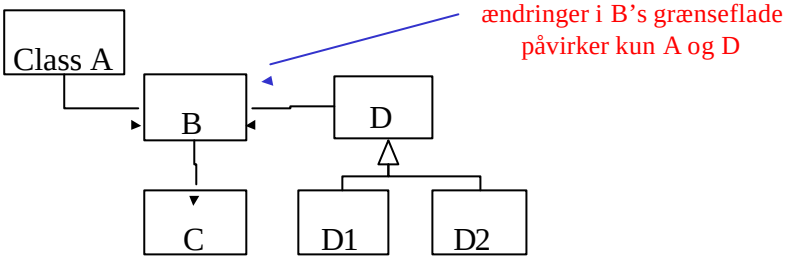
- Indkapsling
  - Abstrakte datatyper, objektbaseret programmering
  - Eksterne/public funktioner, private funktioner og data
  - Dokumentation: moduldiagrammer
  - Realiseret i f.eks. Assembler, PLM, C, Pascal



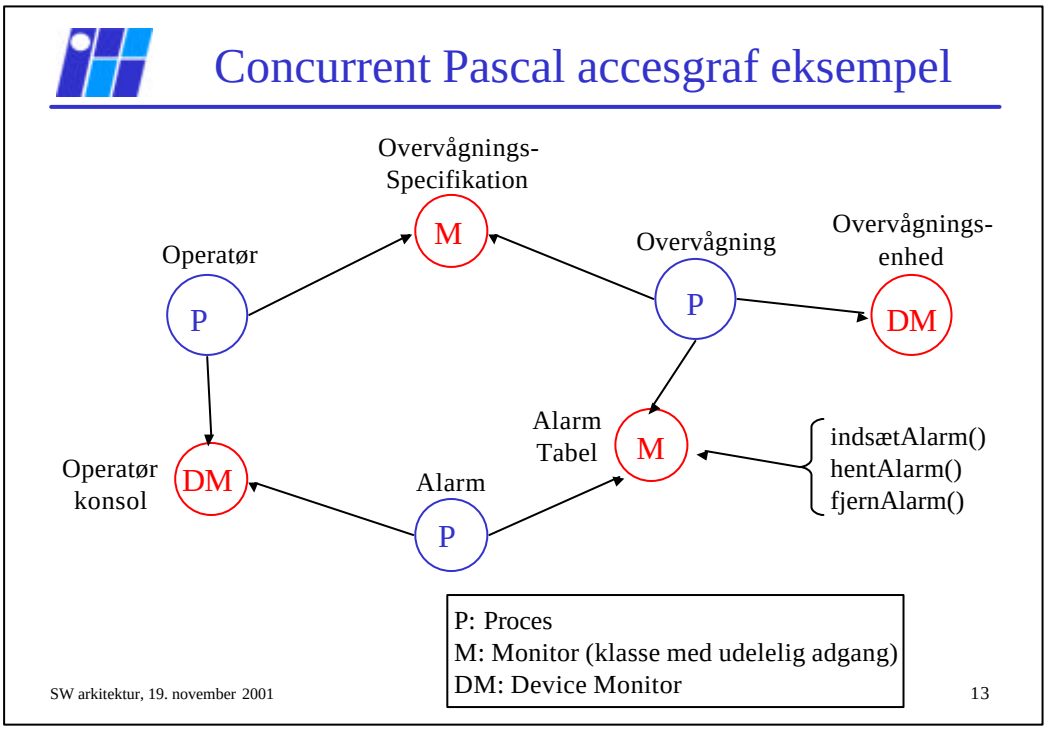
SW arkitektur, 19. november 2001 11

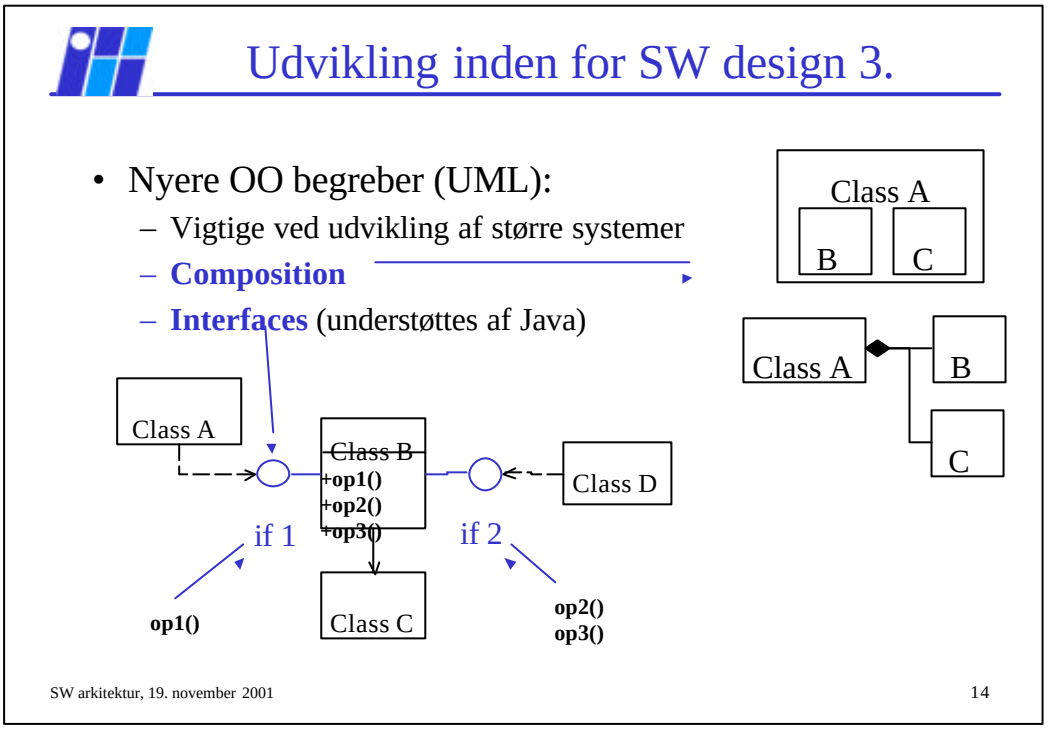
Udvikling inden for SW design 2.

- Objektorienteret Programmering
  - Klasser og associationer
  - Nedarvning og polymorfi
  - Dokumentation: Klassediagrammer (UML)
  - Eksempler på sprog: Smalltalk, C++, Java, C#



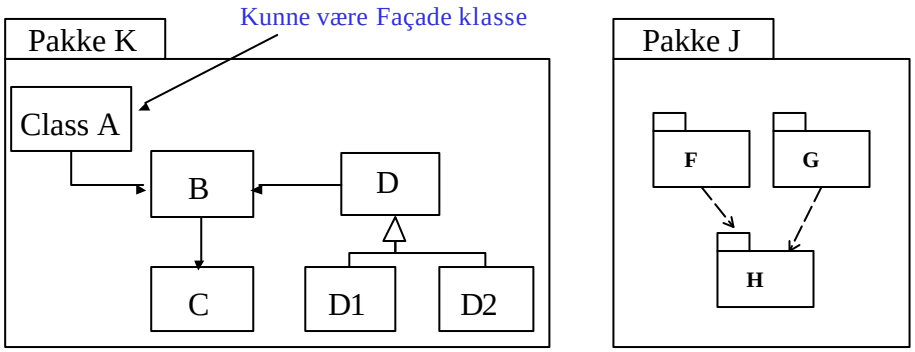
SW arkitektur, 19. november 2001 12


Concurrent Pascal accesgraf eksempel
Overvågnings-
Specifikation
M
Operatør
P
Overvågning
P
Overvågnings-
enhed
DM
Alarm
Tabel
M
indsætAlarm()
hentAlarm()
fjernAlarm()
Operatør
konsol
DM
Alarm
P
P: Proces
M: Monitor (klasse med udelelig adgang)
DM: Device Monitor
SW arkitektur, 19. november 2001
13


Udvikling inden for SW design 3.
• Nyere OO begreber (UML):
– Vigtige ved udvikling af større systemer
– Composition
– Interfaces (understøttes af Java)
Class A
B
C
Class A
B
C
Class A
Class B
+op1()
+op2()
+op3()
Class D
if 1
if 2
op1()
Class C
op2()
op3()
SW arkitektur, 19. november 2001
14

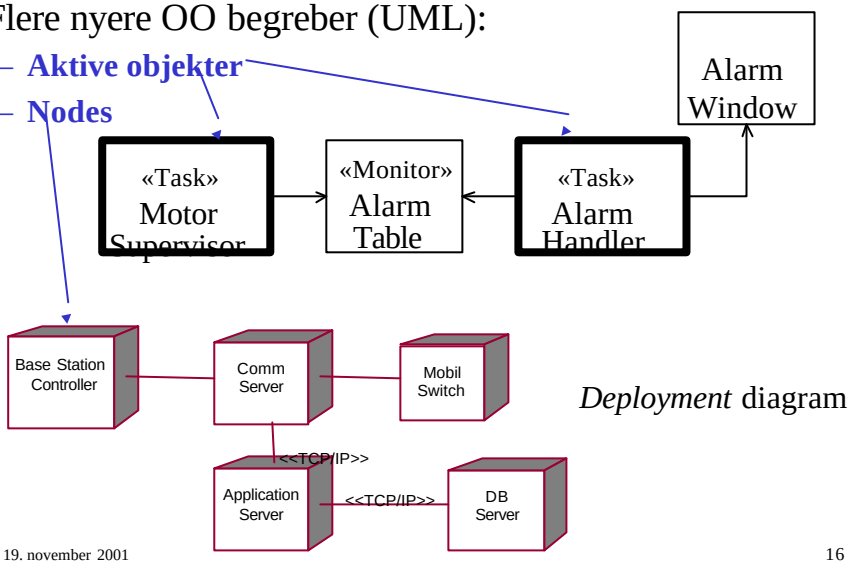
Udvikling inden for SW design 4.

- Flere nyere OO begreber (UML):
  - **Pakker** (understøttes af Java)
  - Anvendes til at vise både logisk og fysisk indkapsling

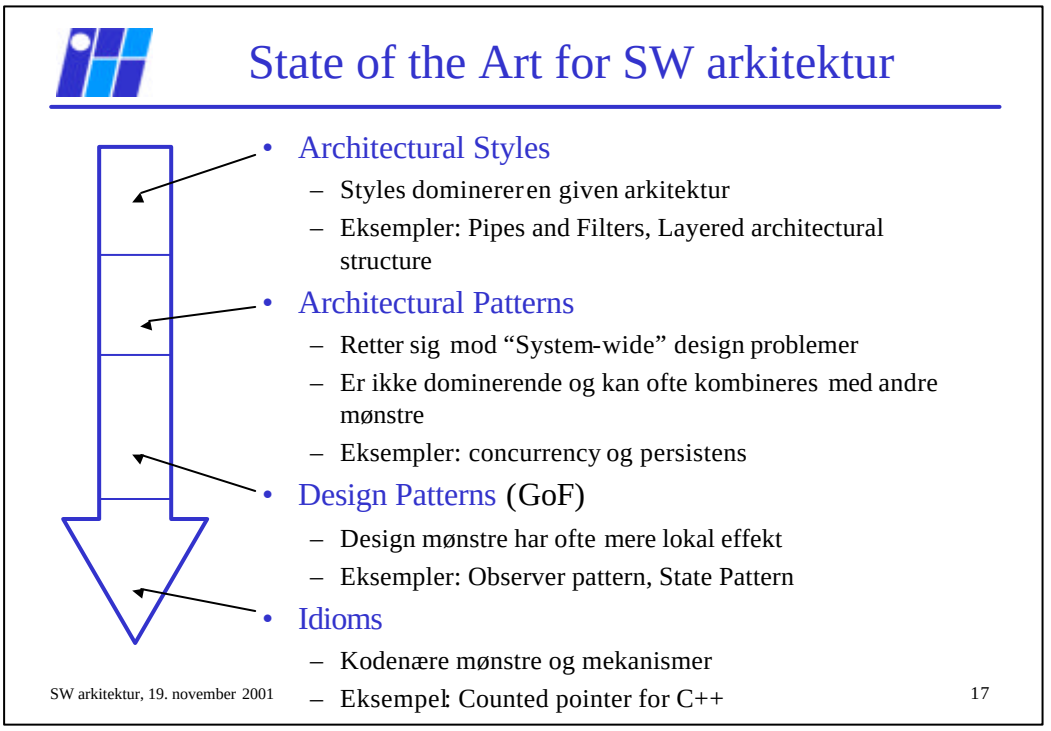


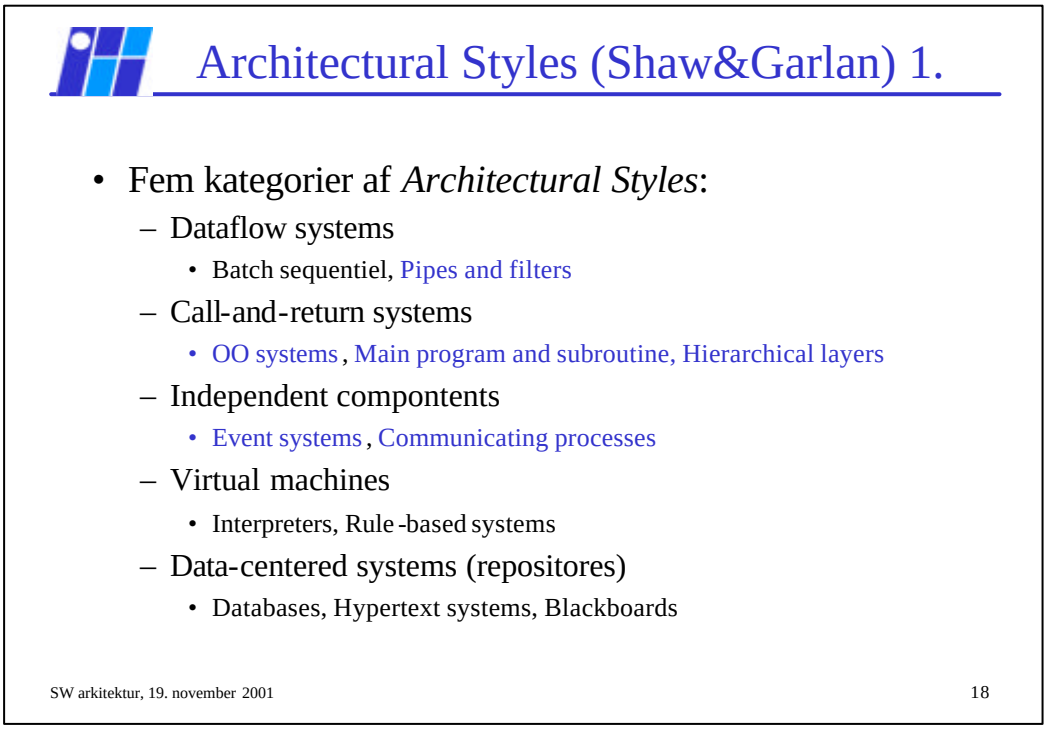
Udvikling inden for SW design 5.

- Flere nyere OO begreber (UML):
  - **Aktive objekter**
  - **Nodes**



Deployment diagram


State of the Art for SW arkitektur
Architectural Styles
Styles dominerer en given arkitektur
Eksempler: Pipes and Filters, Layered architectural structure
Architectural Patterns
Retter sig mod "System-wide" design problemer
Er ikke dominerende og kan ofte kombineres med andre mønstre
Eksempler: concurrency og persistens
Design Patterns (GoF)
Design mønstre har ofte mere lokal effekt
Eksempler: Observer pattern, State Pattern
Idioms
Kodenære mønstre og mekanismer
Eksempel: Counted pointer for C++
SW arkitektur, 19. november 2001
17


Architectural Styles (Shaw&Garlan) 1.
Fem kategorier af Architectural Styles:
Dataflow systems
Batch sequentiel, Pipes and filters
Call-and-return systems
OO systems, Main program and subroutine, Hierarchical layers
Independent compontents
Event systems, Communicating processes
Virtual machines
Interpreters, Rule-based systems
Data-centered systems (repositores)
Databases, Hypertext systems, Blackboards
SW arkitektur, 19. november 2001
18

Architectural Styles (Shaw&Garlan) 2.

- Eksempler på *Architectural Styles*:
  - Pipes and filters
  - Data abstraction and Object-Oriented organization
  - Event-based, implicit invocation
  - Layered systems
    - OSI model, Adm. system: præsentation, logik og model lag
  - Repositores
  - Interpreters
  - Process control

SW arkitektur, 19. november 2001 19

Buschmann's mønster kategorier

Arkitektur mønstre:

- Layers
- Pipes & Filters
- Blackboard

} Styles

- Broker
- Model-View-Controller
- Presentation-Abstraction-Control (PAC)
- Microkernel
- Reflection

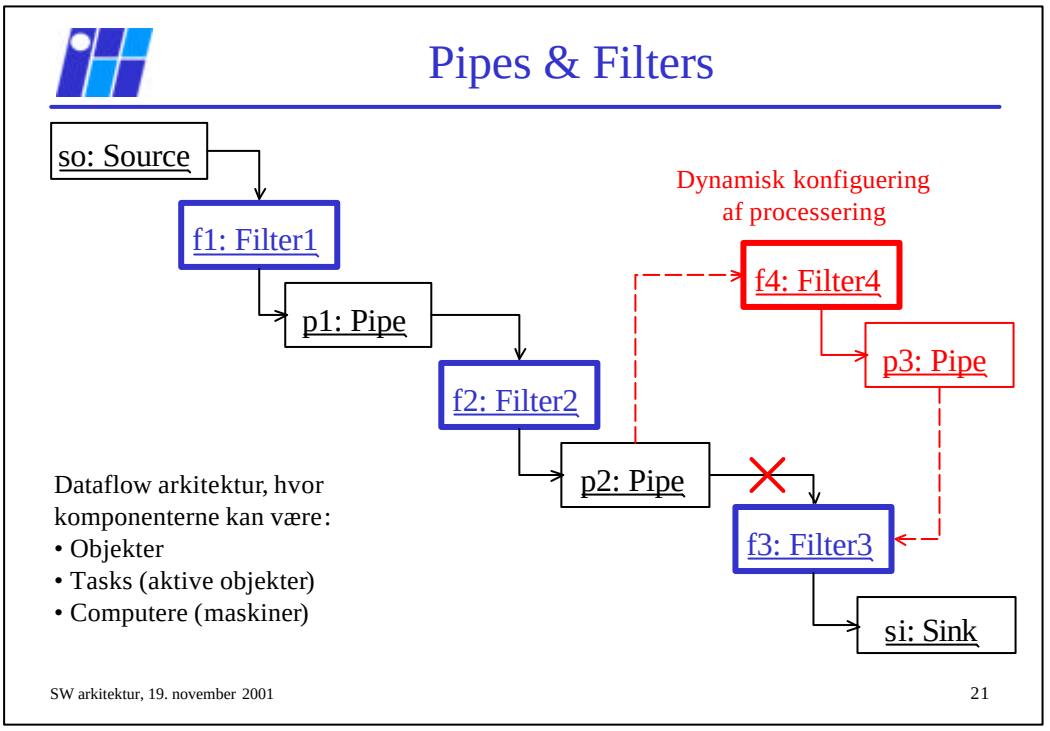
Design mønstre:

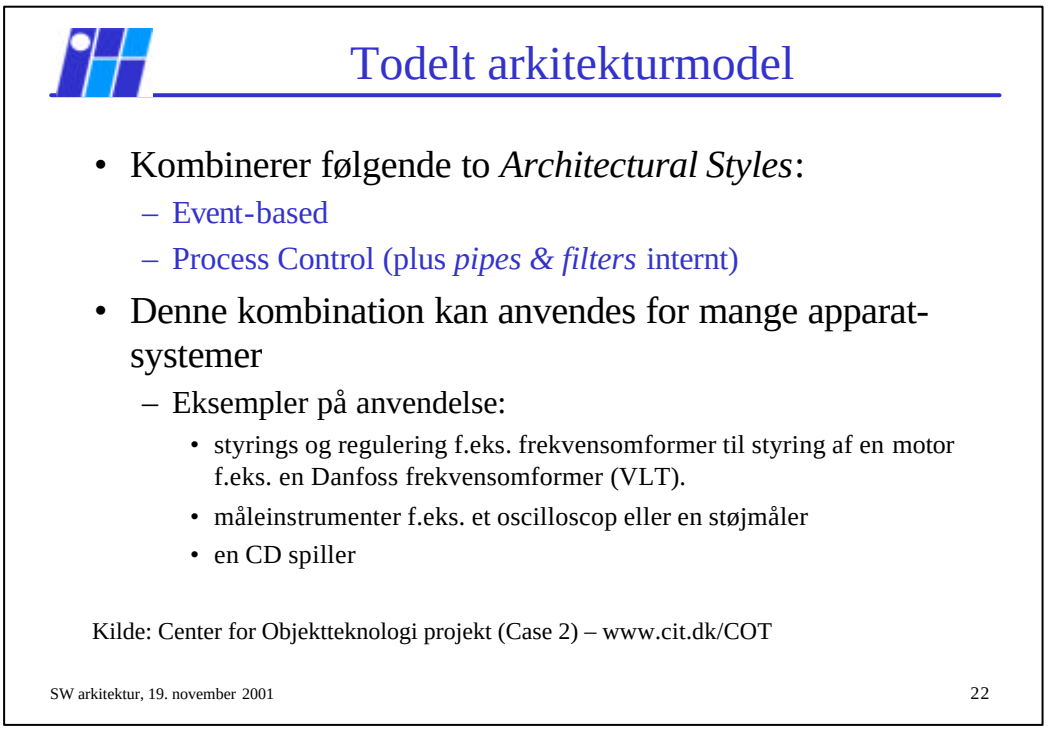
- Observer (GoF)
- Publisher-subscriber
- Strategy (GoF)
- Composite (GoF)
- Abstract Factory (GoF)
- Bridge (GoF)
- Proxy (GoF)
- Command Processor
- View Handler
- Master-slave

Idioms:

- Singleton (GoF)
- Factory Method (GoF)
- Counted pointer, Handle-Body
- Envelope-Letter

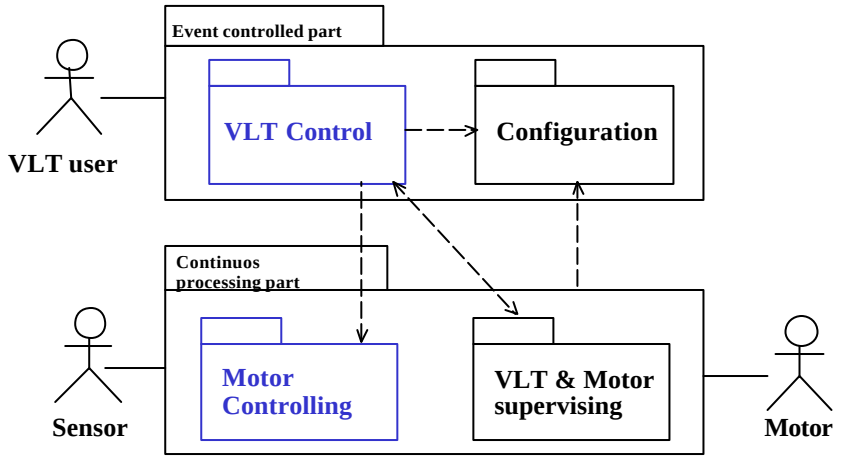
SW arkitektur, 19. november 2001 20


Pipes & Filters
so: Source
f1: Filter1
p1: Pipe
f2: Filter2
p2: Pipe
f3: Filter3
si: Sink
Dynamisk konfiguering
af processering
f4: Filter4
p3: Pipe
Dataflow arkitektur, hvor
komponenterne kan være:
• Objekter
• Tasks (aktive objekter)
• Computere (maskiner)
SW arkitektur, 19. november 2001
21


Todelt arkitekturmodel
• Kombinerer følgende to *Architectural Styles*:
– Event-based
– Process Control (plus *pipes & filters* internt)
• Denne kombination kan anvendes for mange apparat-systemer
– Eksempler på anvendelse:
• styrings og regulering f.eks. frekvensomformer til styring af en motor f.eks. en Danfoss frekvensomformer (VLT).
• måleinstrumenter f.eks. et oscilloscop eller en støjmåler
• en CD spiller
Kilde: Center for Objektteknologi projekt (Case 2) – www.cit.dk/COT
SW arkitektur, 19. november 2001
22

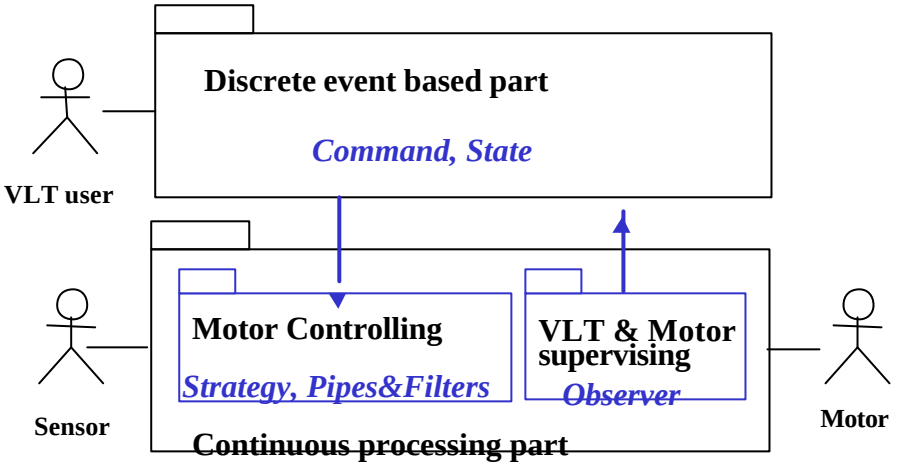
Todelt arkitekturmodel - et eksempel

Danfoss frekvensomformer (VLT)



SW arkitektur, 19. november 2001 23

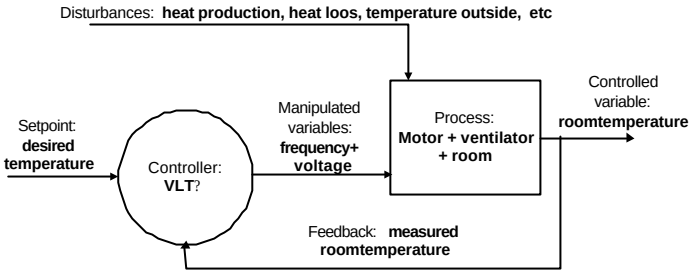
Anvendte Design Patterns i en OO-VLT



SW arkitektur, 19. november 2001 24

Ventilator control as process control

Danfoss frekvensomformer (VLT)



Example of "*closed-loop feedback*" control

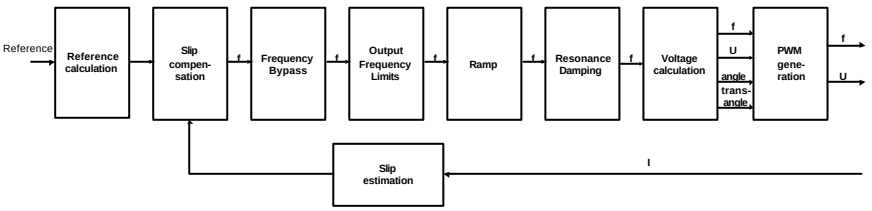
Shaw & Garlan – process control style

SW arkitektur, 19. november 2001 25

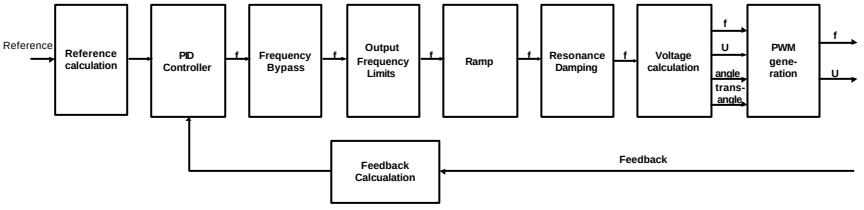
Frekvensomformer eksempel

Blokdiagram for to forskellige driftsformer

Speed Open loop controller:

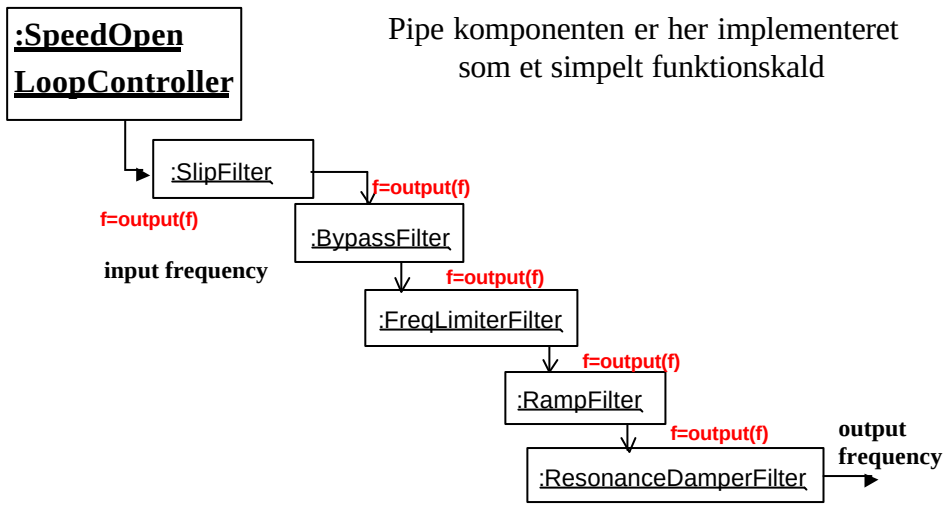


Process Closed Loop controller:



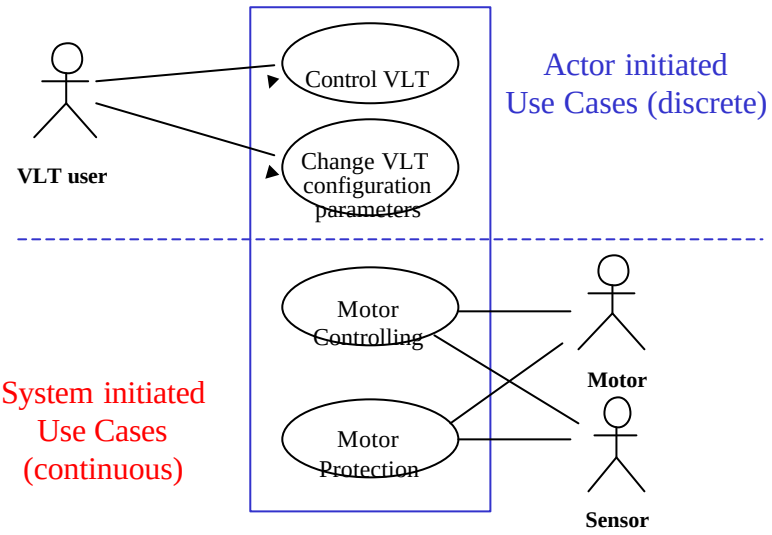
SW arkitektur, 19. november 2001 26

Objektdiagram for SpeedOpenLoopController



Pipe komponenten er her implementeret som et simpelt funktionskald

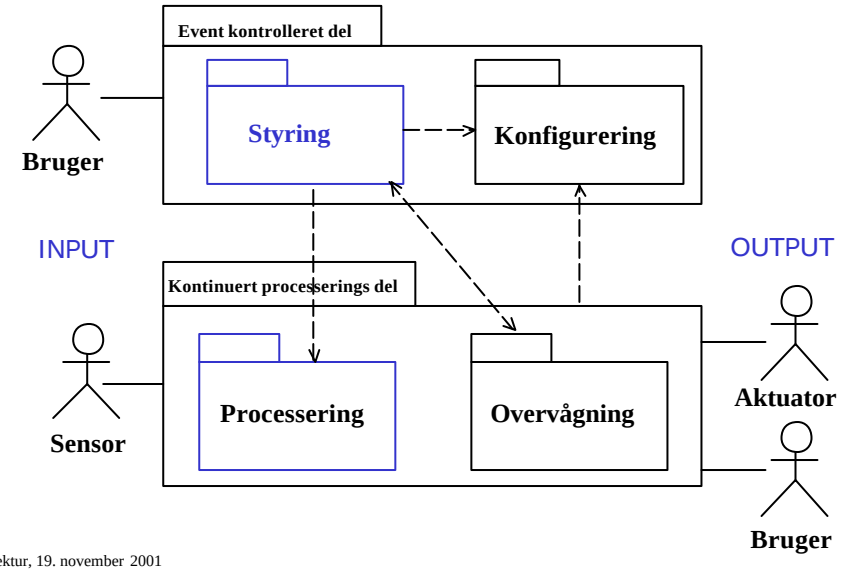
Two types of VLT Use Cases



Actor initiated Use Cases (discrete)

System initiated Use Cases (continuous)

Architectural Style: todelt arkitekturmodel



SW arkitektur, 19. november 2001 29

Taksonomi for Real-Time arkitektur

- Der anvendes i dag fire basale arkitektur typer for real-time systemer:
  - Timeline
  - Event-driven
  - Pipeline
  - Client-Server

Kilde: Doug Locke, TimeSys Corporation
- Embedded System Conference, San Francisco, 2001

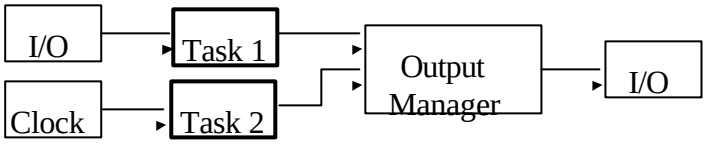
SW arkitektur, 19. november 2001 30

Timeline arkitektur

- Timeline
  - Systemet designes som et antal procedurer, der hver har et bestemt tidskrav (p1: hver 25 ms, p2: hver 200 ms)
  - Procedurer kaldes med faste tidsintervaller, hvis timeren er 25 ms - så aktiveres p1 hver gang og p2 kun hver fjerde gang. Lange procedurer må opdeles i mindre.
  - Eksempel: Flight Control Computer
    - 2-5 processorer, 1 MB RAM
    - 1 task for hver computer, ingen synkronisering
    - De fleste krav er *Hard Real Time* krav, typisk 20 ms
    - Anvendes f.eks. i nyere Airbus fly og i Boeing 747

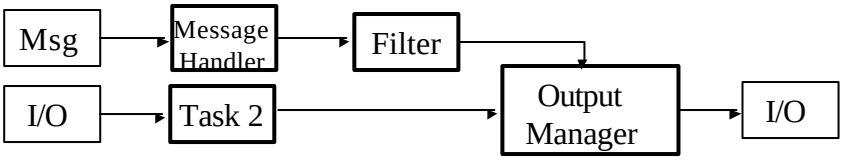
Event-driven arkitektur

- Event-driven
  - Her anvendes afslutning af I/O operationer og timer events til at igangsætte ventende task
  - Der anvendes operativsystem, hvor hvert Task har en prioritet
  - Kræver synkronisering mellem task
  - Eksempel: Aircraft Mission Processor
    - 1-20 RISC processorer, 32-96 MB RAM
    - Soft Real Time krav, typisk i området fra 1 ms - 100 ms



Pipeline arkitektur

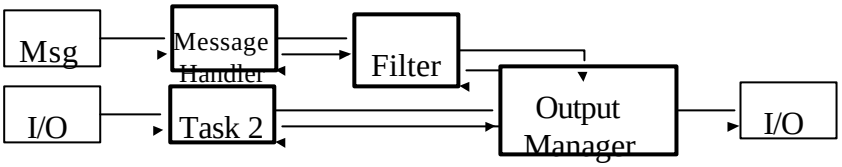
- Pipeline
  - Her anvendes også inter-proces meddelelser ud over afslutning af I/O operationer og timer events til at igangsætte ventende task
  - En hændelse sendes gennem systemet fra *source* til *destination* og bevirker et sæt af task aktiveringer
  - Eksempel: Air Traffic Control
    - 50-300 processorer, 64-256 MB RAM
    - Soft Real Time krav, typisk i området fra 100 ms - 6 sek



SW arkitektur, 19. november 2001 33

Client-server arkitektur

- Client-Server
  - Som for pipeline anvendes også her inter-proces meddelelser ud over afslutning af I/O operationer og timer events til at igangsætte ventende task
  - I modsætning til pipeline - så forbliver kontrollen her på en given node (klienten)
  - Eksempel: Vehicle Training System
    - 1-200 processorer, 32-128 MB RAM
    - Soft Real Time krav, typisk i området fra 33 ms - 150 ms

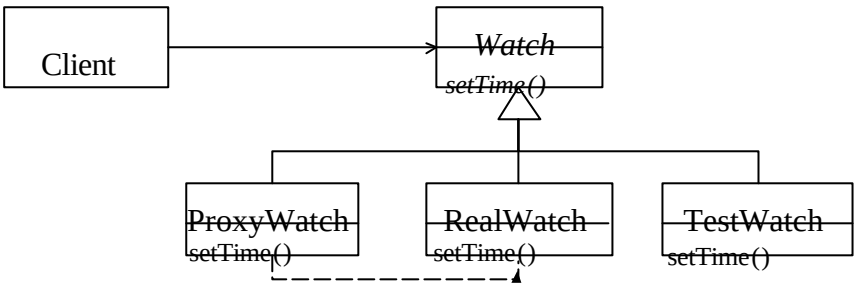


SW arkitektur, 19. november 2001 34

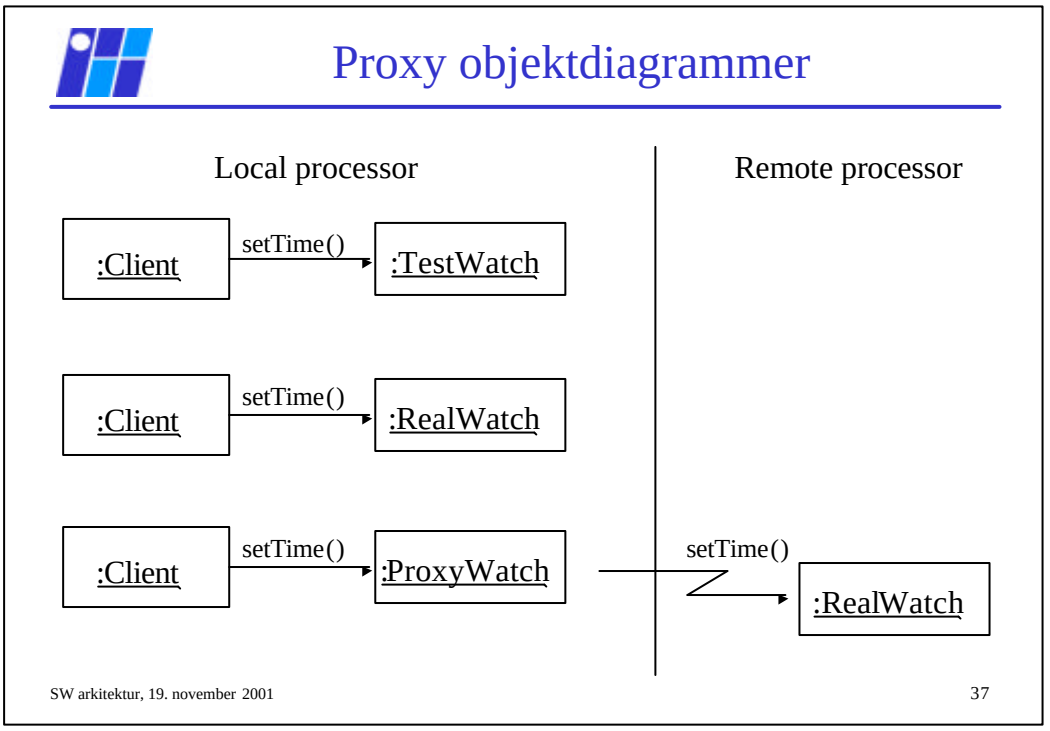
To vigtige designprincipper

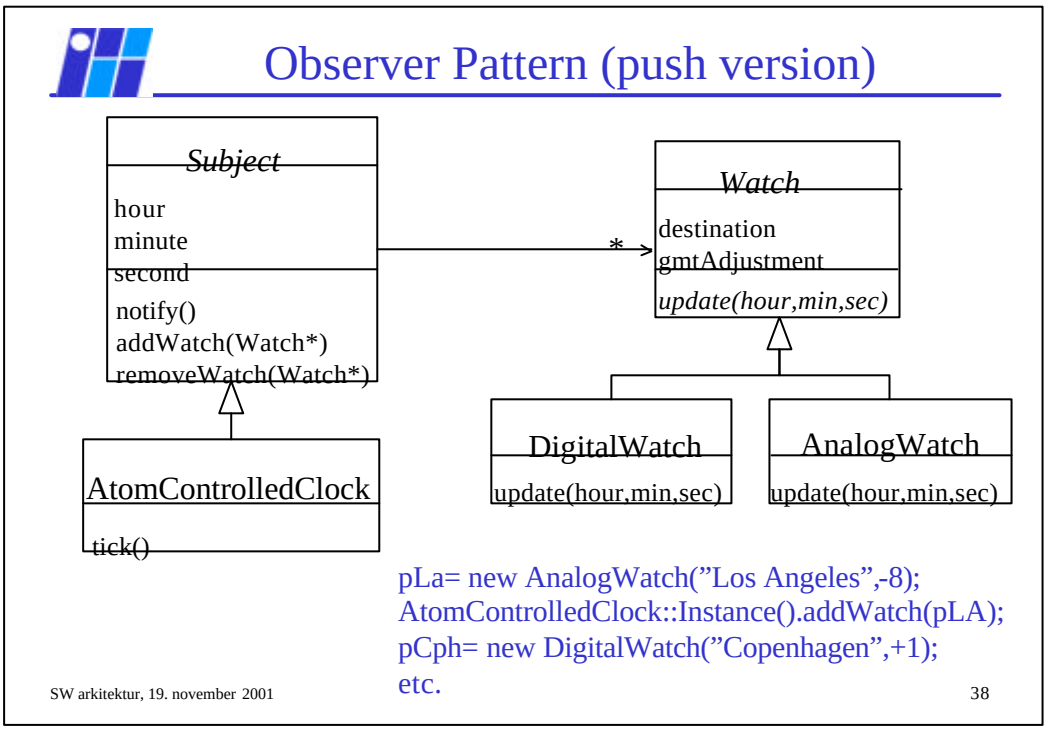
- Bertrand Meyers *Open-Closed* principle:
  - "Software entities (Classes, Modules, Functions etc) should be open for extension, but closed for modification"
  - *Object Oriented Software Construction*, B. Meyer, 1988
- *Liskovs Substitution Principle* (LSP):
  - "Functions that use pointers or references to base classes must be able to use objects of derived classes without knowing it" (den pragmatiske udgave).
  - *"Data Abstraction and Hierarchy"*, Barbara Liskov, SIGPLAN Notices, May 1988
- LSP anvendes til at realisere *Open-Closed* princippet.

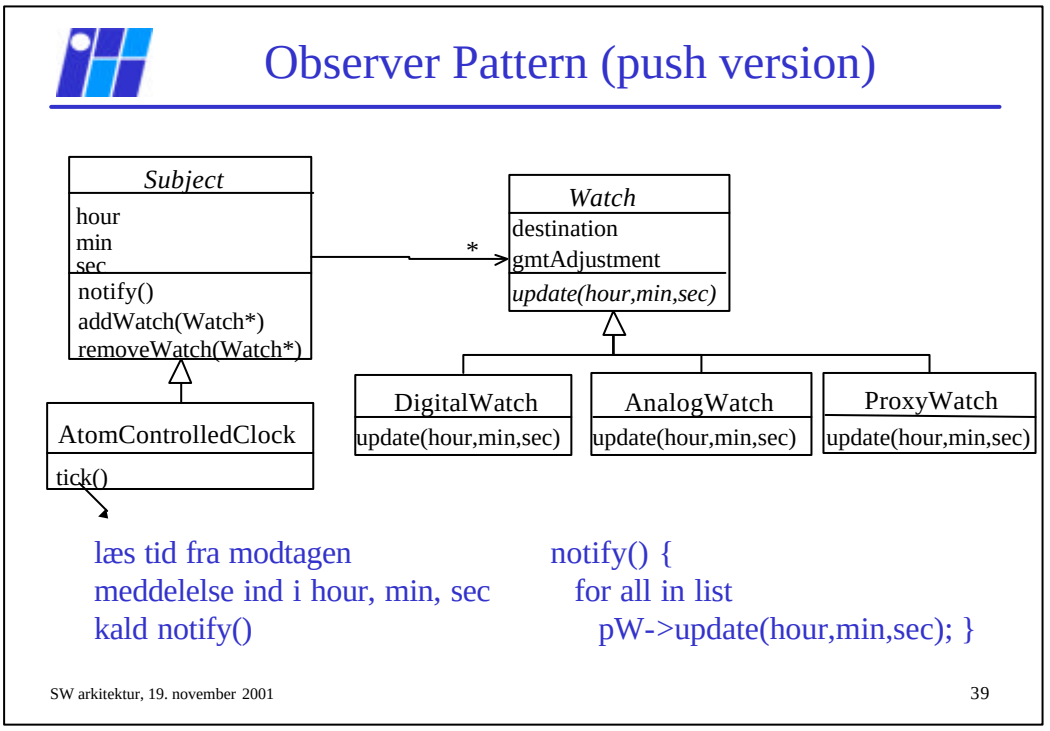
Proxy (stedfortræder) Pattern

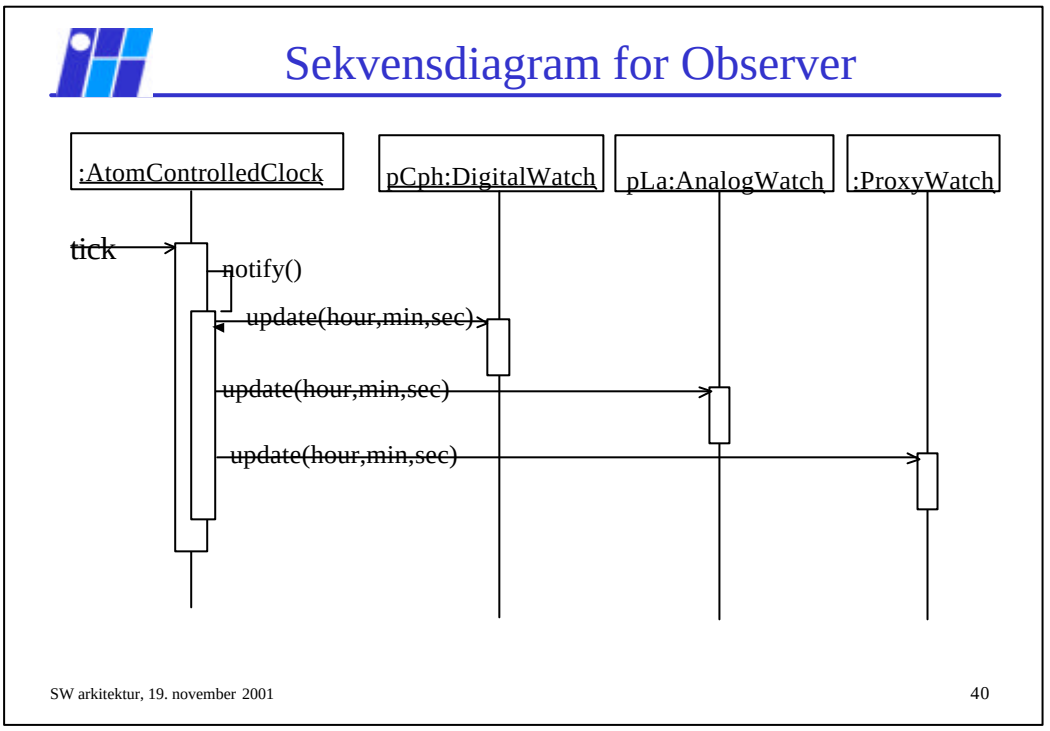


Objekter af RealWatch klassen befinder sig på en anden maskine end Client objektet.
Et eksempel på en Remote Proxy - der kan karakterisers som et Architectural Pattern.


Proxy objektdiagrammer
Local processor
Remote processor
:Client
setTime()
:TestWatch
:Client
setTime()
:RealWatch
:Client
setTime()
:ProxyWatch
setTime()
:RealWatch
SW arkitektur, 19. november 2001
37


Observer Pattern (push version)
Subject
hour
minute
second
notify()
addWatch(Watch*)
removeWatch(Watch*)
*
Watch
destination
gmtAdjustment
update(hour,min,sec)
AtomControlledClock
tick()
DigitalWatch
update(hour,min,sec)
AnalogWatch
update(hour,min,sec)
pLa= new AnalogWatch(”Los Angeles”,-8);
AtomControlledClock::Instance().addWatch(pLA);
pCph= new DigitalWatch(”Copenhagen”,+1);
etc.
SW arkitektur, 19. november 2001
38


Observer Pattern (push version)
Subject
hour
min
sec
notify()
addWatch(Watch*)
removeWatch(Watch*)
*
Watch
destination
gmtAdjustment
update(hour,min,sec)
AtomControlledClock
tick()
DigitalWatch
update(hour,min,sec)
AnalogWatch
update(hour,min,sec)
ProxyWatch
update(hour,min,sec)
læs tid fra modtagen
meddelelse ind i hour, min, sec
kald notify()
notify() {
for all in list
pW->update(hour,min,sec); }
SW arkitektur, 19. november 2001
39


Sekvensdiagram for Observer
:AtomControlledClock
pCph:DigitalWatch
pLa:AnalogWatch
:ProxyWatch
tick
notify()
update(hour,min,sec)
update(hour,min,sec)
update(hour,min,sec)
SW arkitektur, 19. november 2001
40

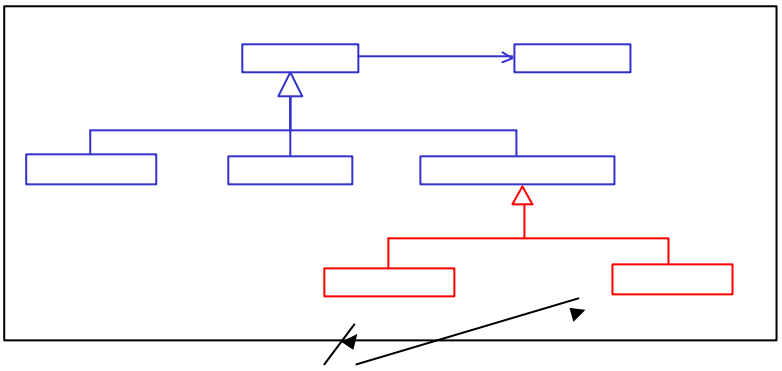
Frameworks

- Frameworks opbygges vha. mønstre
- Forskellige slags framework:
  - Applikationsframework ofte til et specifikt domæne
  - Serviceorienterede frameworks som f.eks.
    - Kommunikationsframework
    - Databaseframework
    - Multiprogrammeringsframework
- Disse kan være implementeret som:
  - Whitebox framework
  - Blackbox framework

SW arkitektur, 19. november 2001 41

Whitebox Framework

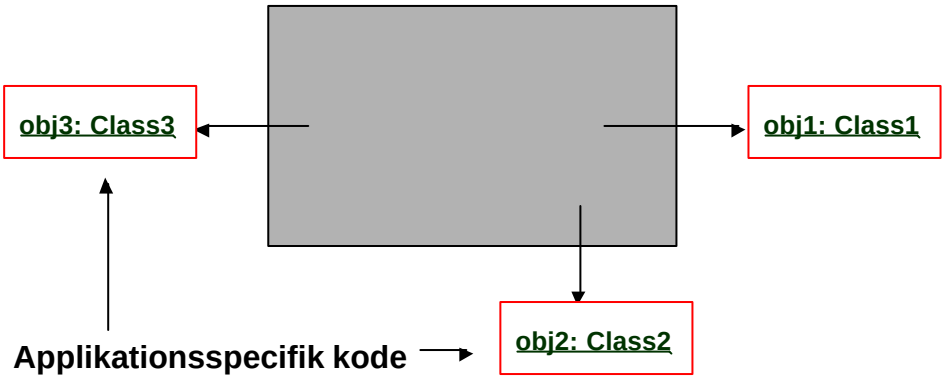
- Her skal man kende den indre struktur for at kunne anvende Frameworket til en konkret applikation f.eks. ved at man tilføjer nye specialiserede klasser til de eksisterende klassehierarkier



SW arkitektur, 19. november 2001 42

Blackbox Framework

- Her er det ikke nødvendigt at kende detaljer i Frameworket, da man tilføjer den ønskede funktionalitet vha. Composition dvs. at man instantierer objekter der ”hægtes” på Frameworket



SW arkitektur, 19. november 2001 43

Erfaring med anvendelse af et framework

Fordele:

- Genbrug af basisklasser og struktur i Frameworket
- Genbrug af serviceklasser
- Kan opbygges vha. GoF design patterns (er veldokumenterede)
- Samme SW struktur for forskellige protokoller
- Letter dokumentation
- Letter vedligeholdelsen pga. færre klasser og samme struktur
- Muliggør en hurtigere protokolimplementering, da man kan koncentrere sig om det protokolspecifikke
- Fremmer inkrementel udvikling

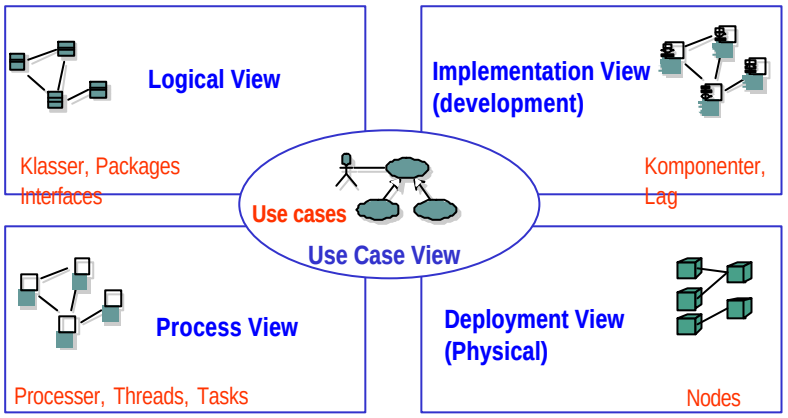
SW arkitektur, 19. november 2001 1/28/2002 44

Framework erfaringer - fortsat

Ulemper:

- Kræver indgående kendskab til Frameworket for at kunne genbruge dette til en ny protokol
- Frameworket bliver først stabilt efter mindst 2 implementationer
- Senere ændringer til Framework strukturen bliver dyrere, da flere projekters SW skal tilpasses
- Kaldestrukturen er mere kompliceret og dermed sværere at teste og debugge

"4+1 View" model for SW arkitektur



Ref.: Philippe Kruchten, "The 4+1 View of Architecture,"IEEE Software, 12(6) Nov. 1995

Udviklingsproces og arkitektur

- Use Case drevet udviklingsproces
  - Udvælg de Use Cases, der har størst betydning for fastlæggelse af systemets arkitektur
- Foretag en iterativ og inkrementel udvikling styret af Use Cases
- Concurrent Engineering (SW + HW udvikling)
  - Arkitektur definerer systemet dvs. både HW og SW
- Fokuser på udvikling af infrastruktur tidligt i et projektforløb
- Muligt at wrappe eksisterende kode ind i klasser
- Udvikling af produktfamilier (HP, Nokia, DD)

SW arkitektur, 19. november 2001 47

Opsummering

- Arkitektur er vigtigere end nogensinde
- Abstraktioner og notationer er ved at være på plads
- Arkitektur *styles* udvikles i disse år
- Mønstre (Patterns) spiller en afgørende rolle på såvel det overordnede arkitekturniveau som på det mere lokale designniveau
- Udviklingsprocesser bør understøtte og lægge vægt på arkitekturaktiviteten
- Arkitektur- og designdokumentation er sammen med kravspecifikationen den vigtigste udviklingsdokumentation

SW arkitektur, 19. november 2001 48

Samarbejdsmuligheder med IHA

- Ingeniørpraktik – ½ år på 5 semester
- Afgangsprojekter på diplomingeniørstudiet
  - udført i virksomheden eller på IHA - vejleder fra virksomheden eller fra IHA
  - omfang 2/3 af sidste semester (25 ECTS point)
- På længere sigt – specialeprojekter på overbygningsuddannelsen (start forår 2003)
- PhD projekter
- Deltagelse i enkeltfag på IHA's kurser under Åben Uddannelse
- Eksterne lektorer
- Eksterne censorer (udpeges for 4 årig periode)
- Vi søger også nye kollegaer til E og IKT linien og til overbygnings-uddannelser

Referencer

- *Design Patterns, Elements of Reusable Object-Oriented Software*
  - Eric Gamma et. al., Addison-Wesley, 1995
- *Software Architecture, Perspectives on an Emerging Discipline*
  - Mary Shaw, David Garlan, Prentice-Hall, 1996
- *Pattern-Oriented Software Architecture - A System of Patterns,*
  - Frank Bushmann et. al., John Wiley & Sons, 1996
- *Design and Use of Software Architecture*
  - Jan Bosch, Addison-Wesley, 2000
- *Software Architecture for Product Families*
  - Mehdi Jazayeri, Alexander Ran, Frank van der Linden, Addison-Wesley, 2000
- *The 4+1 View Model of Architecture*
  - Philippe Kruchten, *IEEE Software*, 12 (6), November 1995, IEEE
- *Center for Objekt Teknology (COT)*, har flere rapporter om SW arkitektur
  - http://www.cit.dk/COT/ - se under Report Series