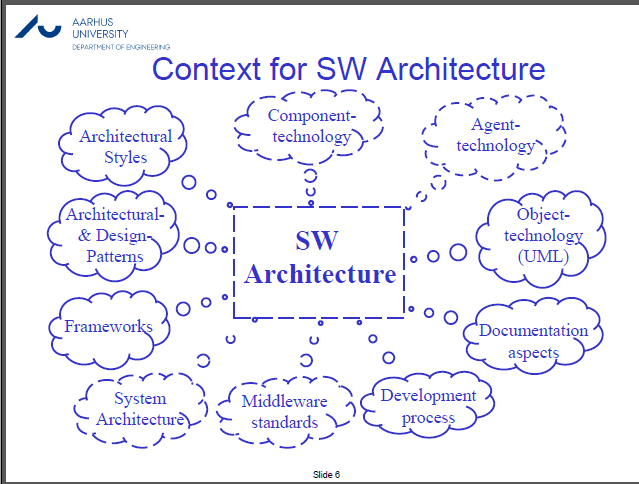
**Question 1: Architectural Views (course week #1)**

* Explain the concept of architectural views and give examples of different view models, compare and discuss them.
* Explain how a view model can be used in an iterative process model.

**Definition på SW arkitektur:**

* Software arkitekturen af en program system er strukturen af strukturer af systemet, som omfatter software komponenter, de eksternt synlige egenskaber af de komponenter og relationerne mellem dem.

UML (Unified Modeling Language) bruges til at visualiserer Software arkitektur design

**Design karakteristik:**

* En analyse model kan implementeres med antal af forskellige design løsninger
* Det er normalt engineering discipline som forslår og investigerer en antal af forskellige løsninger og vælger den bedste overenskomst for en given situation
* Det er også vigtigt at dokumenterer design alternativer

**ROPES 3 design niveauer:**

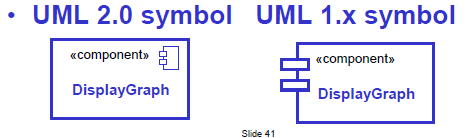
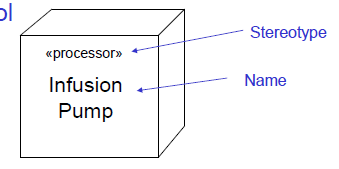
* Arkitektur design
  + Node, pakker (package), komponenter og aktive objekt
* Mekanistisk design
  + Gruppe af samarbejdende objekter
* Detaljeret design
  + Klasser (klassens navn, attributter og operationer (funktioner) )

**Arkitektur views:**

* Adskillige modeller
  + Kruchten, 4+1 view model
  + Nokia model, 3+1 view model
  + Hofmeister, 4 view model
  + Douglas, 5 view model
* Alle forfatter er enige om hoved view konceptet
  + Forskellige view modeller er behøvet
  + Hver view viser en specifik karakteristisk af systemet, men navne og indholdet varierer
* Det er nyttigt at anvende en a+1 model, som beskriver scenario eller use cases

**4+1 view model (Kruchten)**

Denne type af model bruges til at beskrive software arkitekturen, ved at anvende flere samtidige visninger. Dette bruges til at beskrive systemet set fra forskellige interessante, såsom slutbruger, udviklere og projekt leder.

* Logic view (logsike view)
  + Den logiske view vedrører den funktionalitet, som systemet giver til slutbrugeren.
  + Beskriver den logiske design med klasse diagrammer med overordnede organisationer beskrevet af pakker (packages)
  + Klasse, kommunikation og sekvens diagrammer kan bruges her.
  + ROPES anbefalder en logiske arkitekturer baseret af konceptet af domains:
    - En domain er en uafhængig subjekt område
    - Eksampler: Operating system, data management (data styring) , User interface, Hardware
* Devolpment view (udvikling view)
  + Viser systemet fra en programmør perspektiv og inkluderer også software management. Dette view også kendt som en implantation view.
  + Beskriver den statiske software organisation i moduler, fysikse lagring og gruppering af source koden, data filer, komponenter osv
  + Bruger UML component diagram til at beskrive systems komponenter
    - System komponenter beskriver en software enhed, der eksisterer på run-time,
      * En bibliotek
      * En binær komponent
      * En fil (konfiguration fil, html/xml fil)
  + UML diagramer som er anvendt til at præsenterer udviklingen view inkluderer package diagram
* Deployment view (Physical /fysisk)
  + Beskriver computer og forbundet hardware apparater og valgfrit også allokation af run-time komponenter i computer
  + Viewt koncentrerer sig med topologien af software komponenter i den fysiske lager også forbindelser mellem disse.
  + Anvender deployment diagrammer, som viser fysiske apparater og forbindelser
  + UML nodes er anvendt, som præsenterer
    - Fysiske enheder (apparater)
      * Processor, sensorer, aktuar, display memory
    - Udførsels miljø
  + UML symbol
* Process or task view (vigtig for real-tids systemer)
  + Beskriver concurrency eksempel, processer, tasks eller threads og deres kommunikation og synkronisering
  + Anvender klasse diagram
    - Viser primært aktive klasse (processer,task,thrrad) og deres kommunikation mekanisme
      * En aktiv objekt er et objekt som udfører i dens egen thread
      * I real tid UML kaldes for en task eller thread
* Use case / scenario (hændelsesforløb) view
  + Beskrivelsen af en arkitektur er illustreret benyttelse af få sæt af use case, som bliver den femte view.
  + Beskriver sekvenserne og påvirkning mellem objekter og mellem processer
  + Anvender use case diagrammet

**Nokia 3+1” View model:**

* Logical view
  + System, produkt, applikation
* Development view
  + Komponenter, interface og lager
* Scenarios
  + Sammenarbejde, rational design
* Run-Time view
  + Sammenlignet med 4+1 view model er denne view en blanding af Process og deployment view
    - Computer, processer, protokoller, vedvarende objekter

**BDP’s FIve views of physical architecture:**

* Subsystem and component view
  + Ligesom 4+1 view model (Development view)
* Deployment view
  + Ligesom 4+1 view model (Deployment view)
* Concurrency and Resource view
  + Ligesom 4+1 view model (Process view)
* Safety and reliability view
  + Her bliver system redundacy defineret og styret til at øge systemets pålidelighed og sikkherhed
    - En pålidelighed system er en system som forsætter med at opererer over en lang periode
  + Et sikkert system er et system som ikke opretter ulykker, som vil lede til skade, tabet af liv eller skadet ejendom.
  + I safety-critical miljøer, applikationer skal forsætte til at gøre det rigtig ting, selvom en type fejl opstår.
* Distribution view
  + Omhandler hvordan objekter, som kan være forskellige i adresss spaces finder og sammenarbejde med hinanden.
  + Inkluderer også politik for hvordan objekter kommunikerer (kommunikation protokol)