UML 2.0 och dess roll för modellbaserad utveckling

Morgan Björkander

Senior Methods Engineer

mbj@telelogic.com

Telelogic

Agenda

- UML 2.0 översikt
  - översikt
  - nya språkkonstruktioner
- Modellbaserad utveckling
  - UML och MDA (Model Driven Architecture)
  - verktygsstöd

Unified Modeling Language

- UML standardiserades av Object Management Group (OMG) 1997
  - har accepterats av en stor del av mjukvaruindustrin
  - lämpar sig väl för modellbaserad utveckling
- UML är ett grafiskt modelleringsspråk vars främsta syfte varit att specificera och dokumentera källkod
  - många implementationer specialiserade på olika områden (tex. inbyggda system och affärssystem)

Varför behövs UML 2.0?

- Begränsningar i UML 1.x
  - erfarenheter från verktygsmakare och användare visar på bra och dåliga egenskaper
  - konsistensproblem mellan diagram och otillräcklig språkprecision
  - saknar tillräcklig skalbarhet för att kunna hantera större system
  - svårt att använda
- Nya krav
  - verktyg har egna tillägg som användare sedan vill se i standardspråket
  - teknikskifte med komponentbaserad utveckling och exekverbara modeller
  - modellbaserad utveckling (MDA)
  - olika förbättringar av alla områden (klasser, komponenter, samarbeten, tillståndsmaskiner, användningsfall, aktiviteter, sekvenser)

OMG-terminologi för UML 2.0

Infrastructure

Definerar de konstruktioner som används för att beskriva UML (klassdiagram) och utöka språket mot specifika områden

6:e januari, 2003

Superstructure (”UML”)

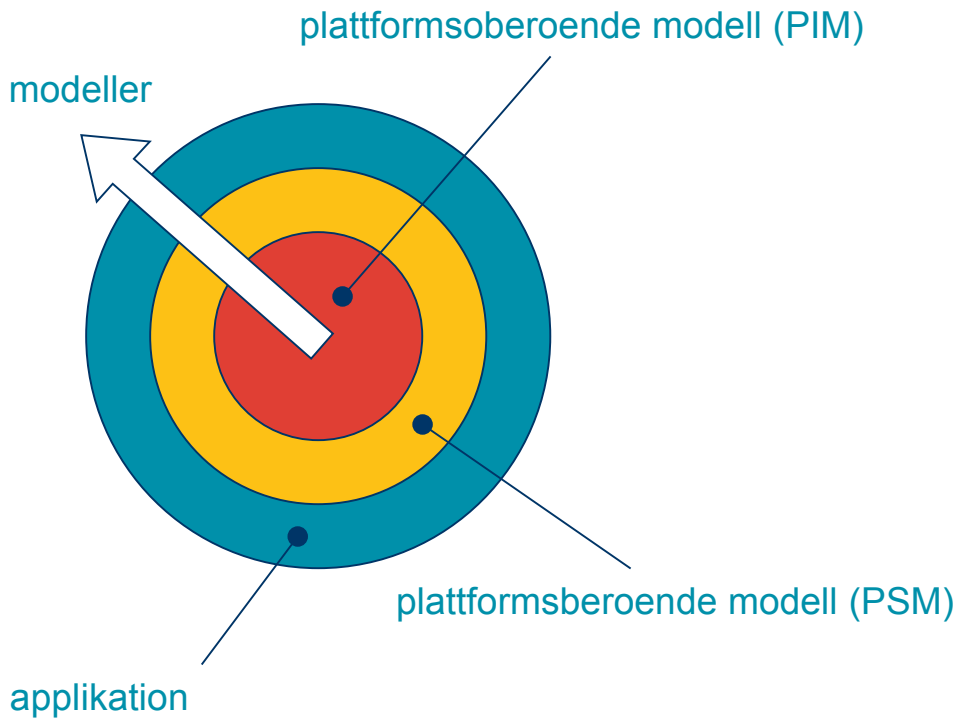
Definierar användarbegrepp för att modellera systemstruktur och beteende; klass diagram, sekvensdiagram, tillståndsdiagram, etc. *Superstructure* inkluderar hela *Infrastructure*.

Object Constraint Language

Diagram Interchange

OMGs Model Driven Architecture (MDA)

- MDA är ett konceptuellt ramverk inom OMG
  - modellbaserad utveckling
- Tre basbegrepp
  - plattformsoberoende modell
    - logisk beskrivning av struktur och funktionalitet
  - plattformsberoende modell
    - hantera specifika implementationsfrågor
  - applikation
    - ”källkod”



MDA är ett konceptuellt ramverk

- Modellen utgör kärnan i ett utvecklingsprojekt
  - olika diagram visar mer eller mindre detaljer om systemet
  - UML
- Språk- och plattformsoberoende
  - det är möjligt att mappa eller till olika teknologier
  - profiler och standardiserade mappningar
  - transformation mellan modeller baserat på olika regler

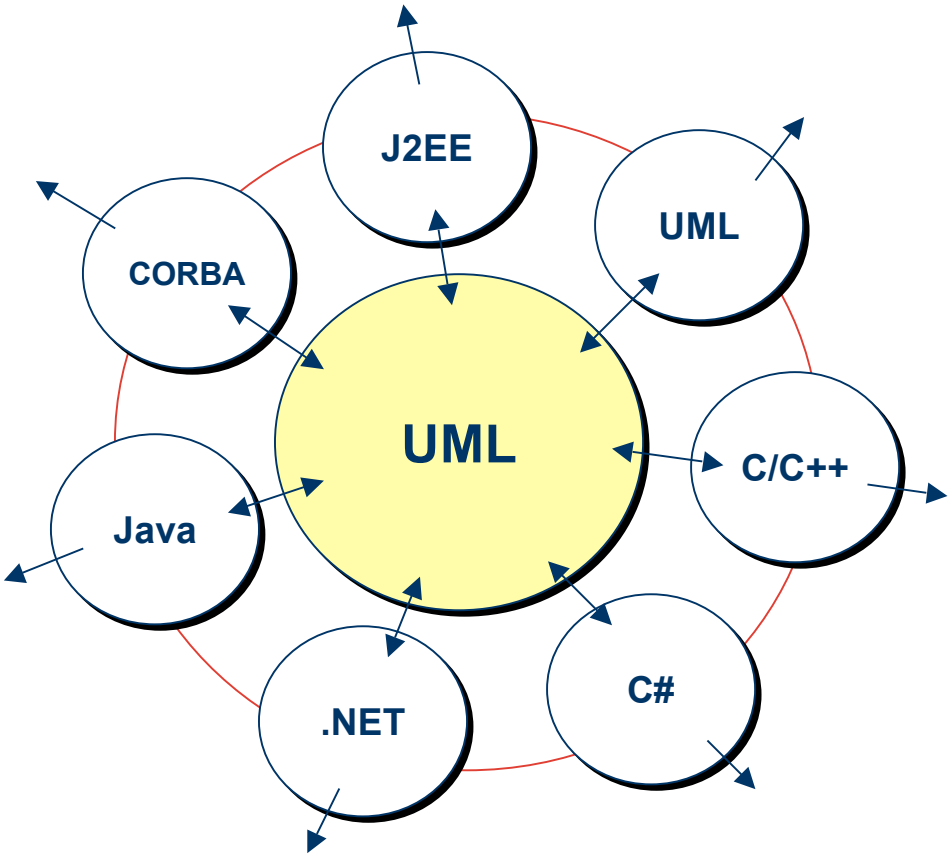
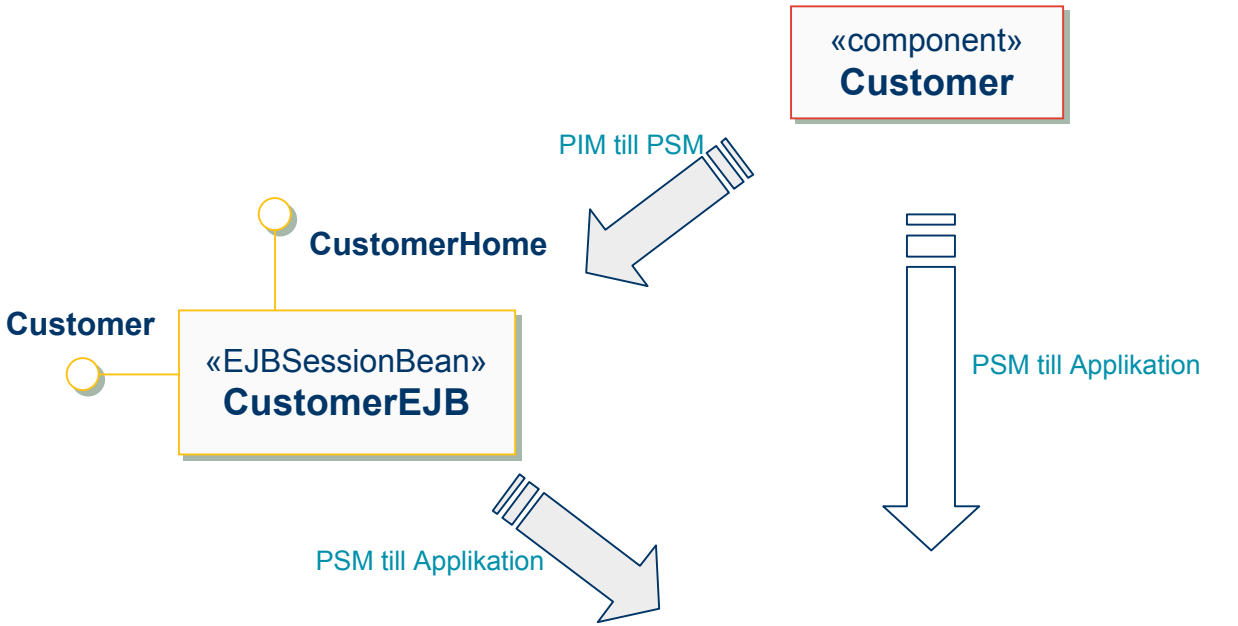


Illustration av MDA



Notera att termerna

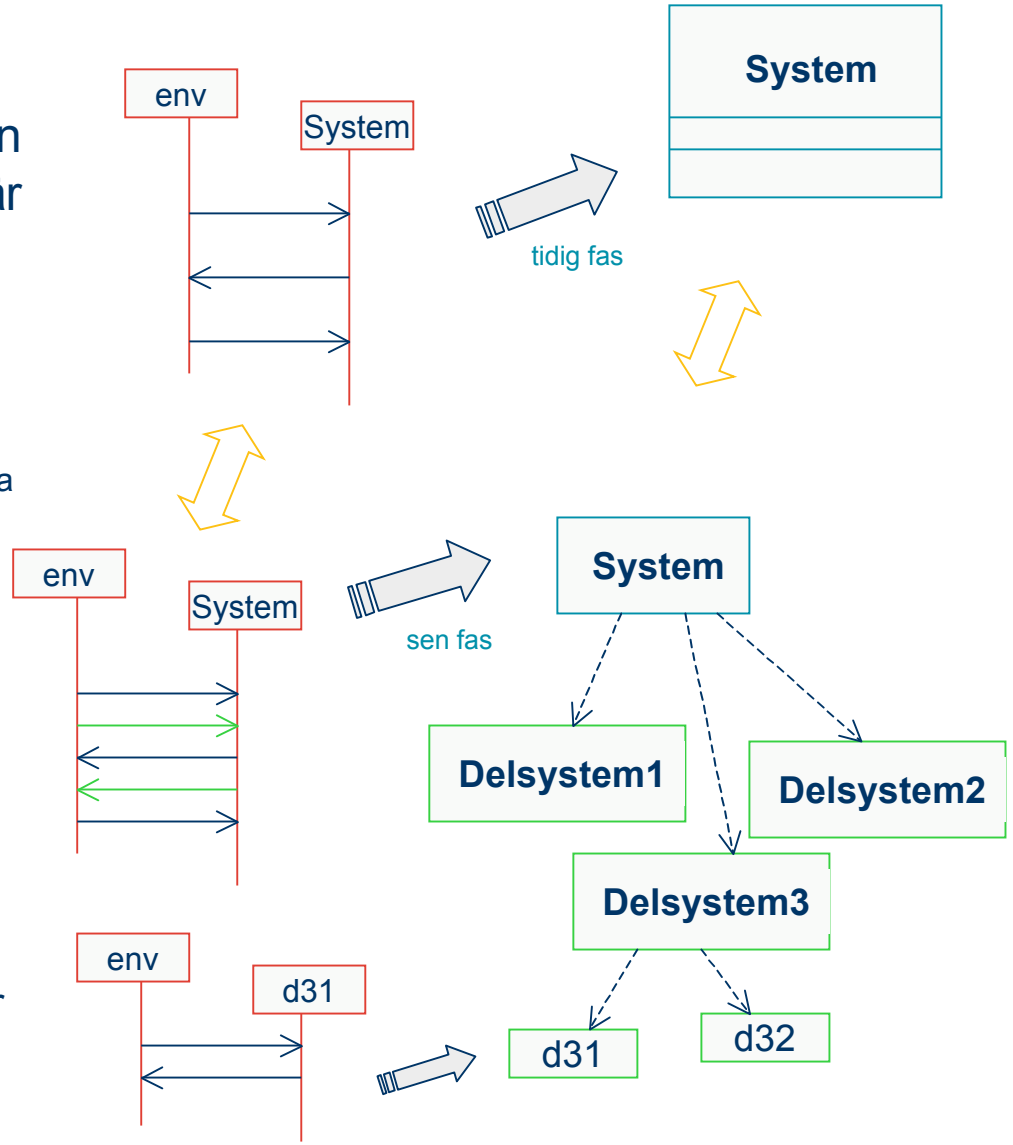
PIM och PSM är relativa.

Några mål med modellbaserad utveckling

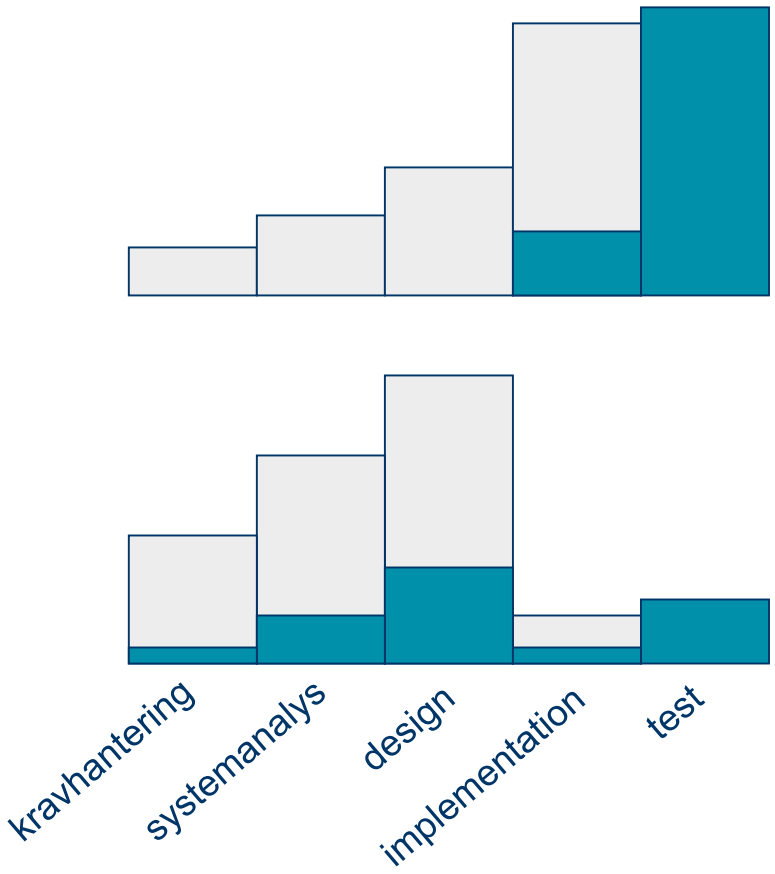
- Minska gapet mellan de olika rollerna i ett utvecklingsprojekt
  - gör det möjligt för alla involverade att tala samma språk
    - arkitekter, programmerare, testare
    - projektledare, chefer
    - *kunder*
    - tredje part
  - gör det lättare att flytta resurser till där de behövs mest för tillfället
- Betrakta utveckling mer som en ingenjörsverksamhet
    - utveckling
    - *underhåll*
- Undvik kodcentrisk utveckling
  - fokusera på systemets funktionalitet
    - att *förstå* systemet är A och O
  - krav, skisser, dokument, designbeslut, kod
    - allt är en del av systemet
  - modellen fungerar som sambandscentral
- Verktyg kan generera mycket eller all kod från en modell
  - låt verktyg ta hand om minneshantering, distribution, etc.
  - undvik tidskrävande lågnivåavlusning

Exekverbara modeller

- Ett av de starkaste argumenten för modellbaserad utveckling är exekverbara modeller
  - beteende och detaljerad funktionalitet kan beskrivas i modeller i stället för i kod
    - om modelleringspråket tillåter detta
  - högre abstraktionsnivå än i programspråk
- Modeller kan exekveras oavsett i vilken utvecklingsfas man befinner sig
  - en vanlig missuppfattning är att det bara går att exekvera detaljerade designmodeller eller implementationsmodeller



Utvecklingscykeln



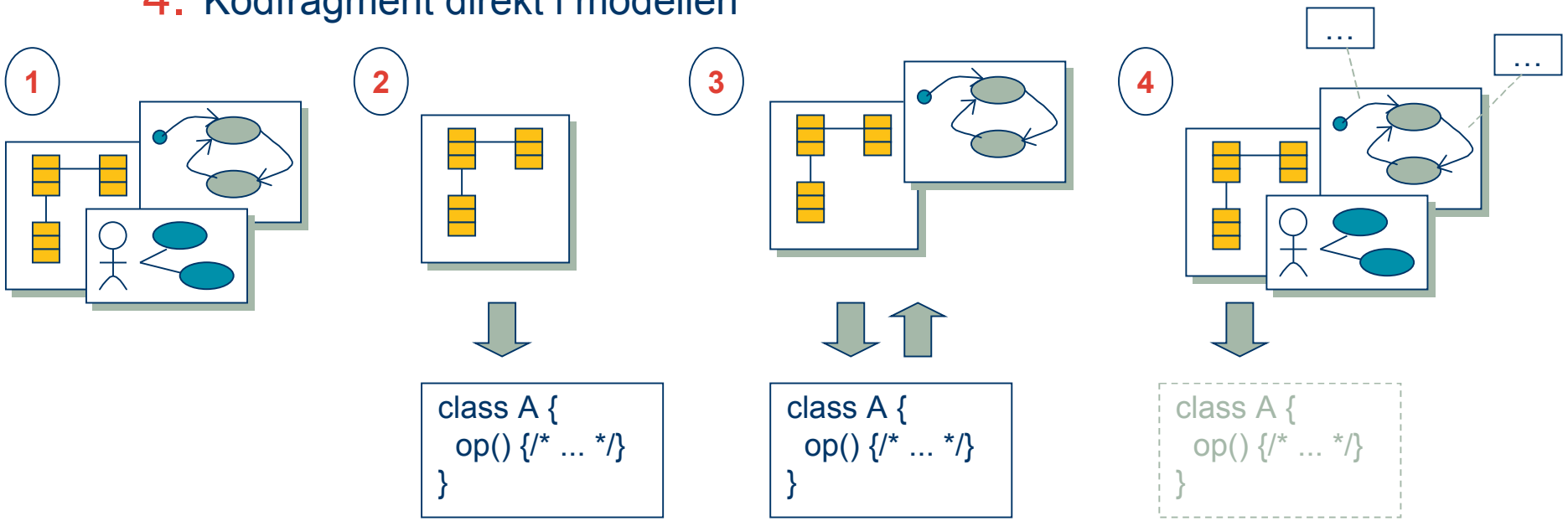
traditionell utveckling

modellbaserad utveckling

Verktygsstöd för UML 1.x

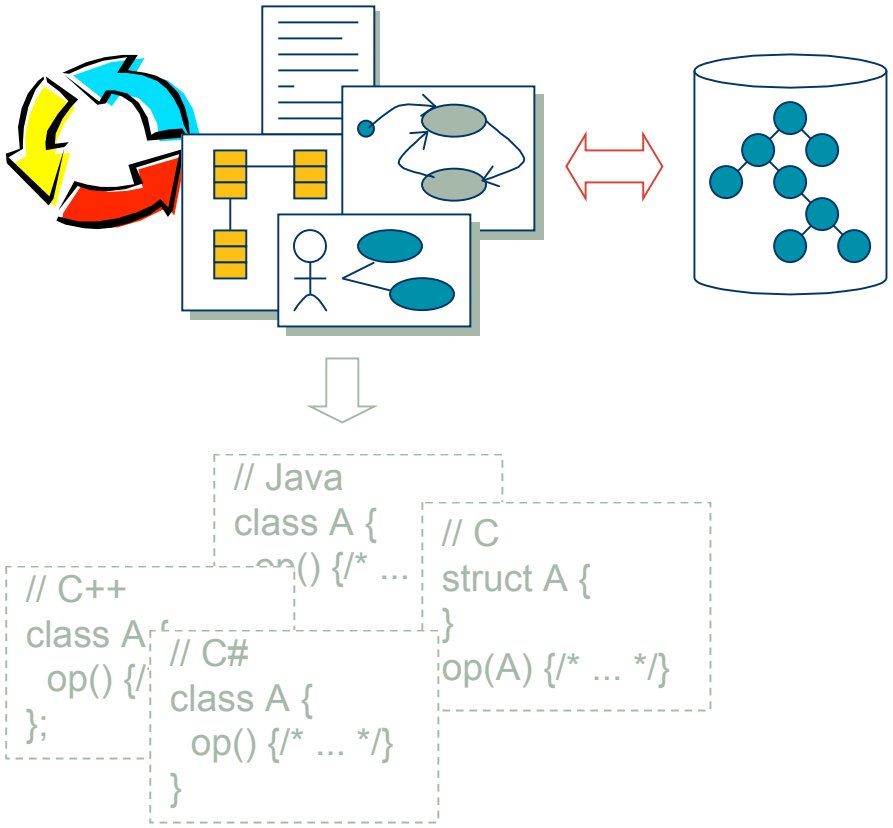
1. Rena ritverktyg
2. Skelettkodsgenerering
3. Skelettkodsgenerering med synkronisering mellan kod och modell
4. Kodfragment direkt i modellen

- Konsistensproblem
  - gäller koden eller modellen?
- Programspråksberoende



Modellbaserad utveckling

- Baserat på en modell
  - varför blanda in programspråk i modellen?
  - utveckla i modellen
- Generera koden (eller andra modeller); modellen innehåller abstraktioner (”patterns”)
  - inbyggda i språket
  - användardefinierade
  - kodgenereringsregler
- Andra hänsyn:
  - kravhantering
  - konfigurationshantering
  - utvecklingsprocess

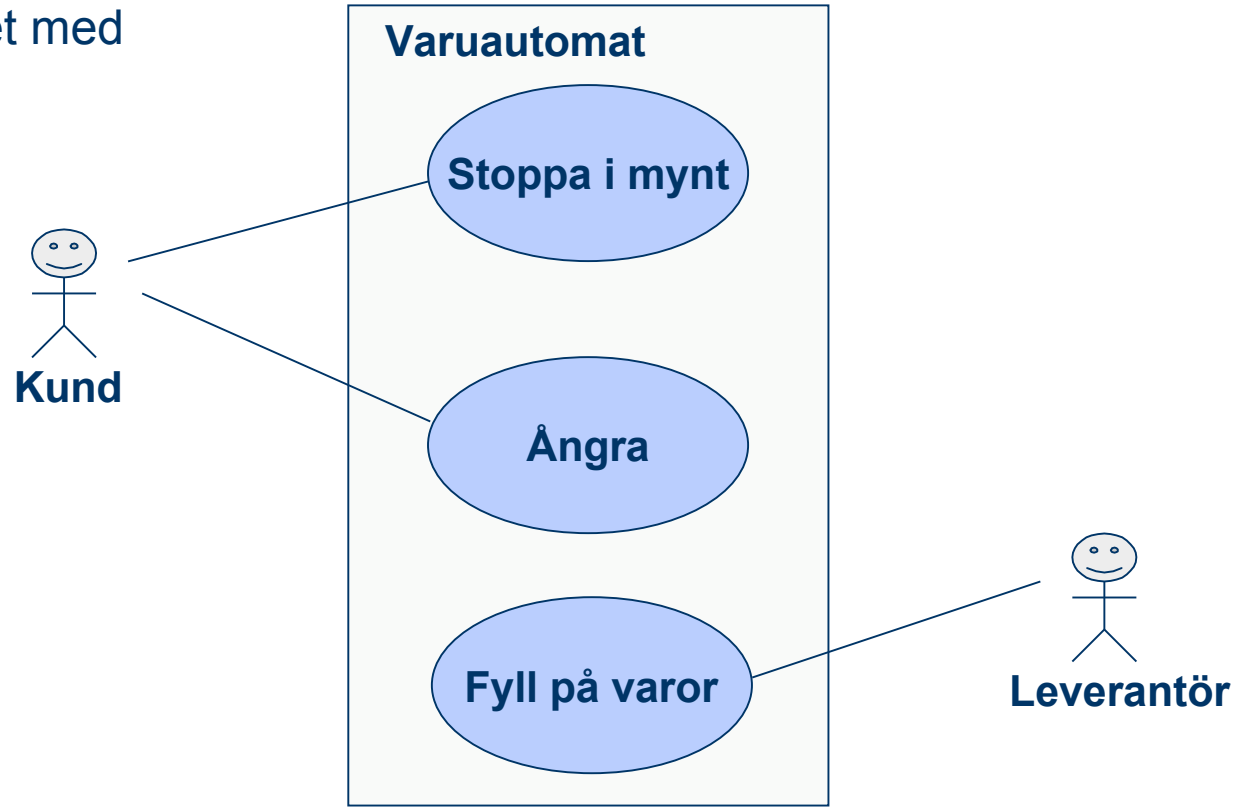


Förslag på UML 2.0

- Det finns flera konsortier som arbetar med olika delar av UML
  - de olika konsortierna samarbetar med varandra
- U2 Partners är det enda konsortium som tagit fram ett komplett förslag för Superstructure (”UML”)
  - www.u2-partners.org
  - www.omg.org
  - de exempel vi tittar på senare baseras på förslaget från U2 Partners

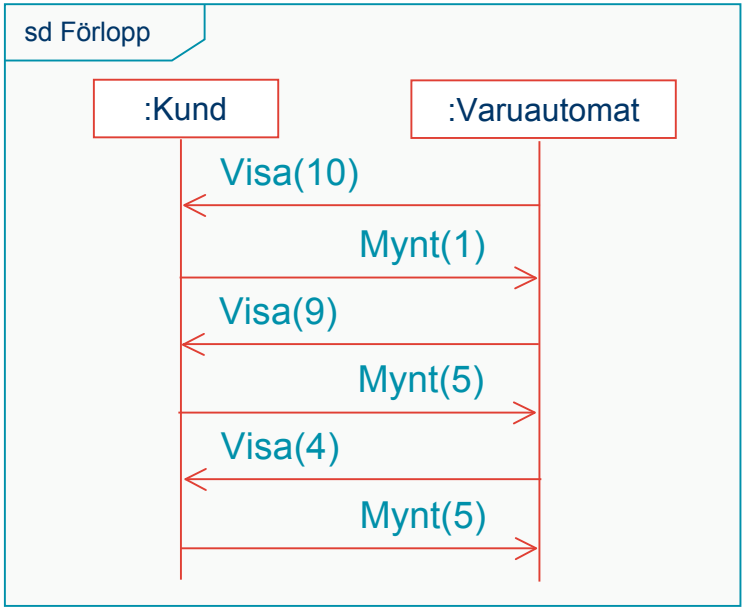
Användningfall

- Notationsmässigt har det inte hänt särskilt mycket med användningsfall



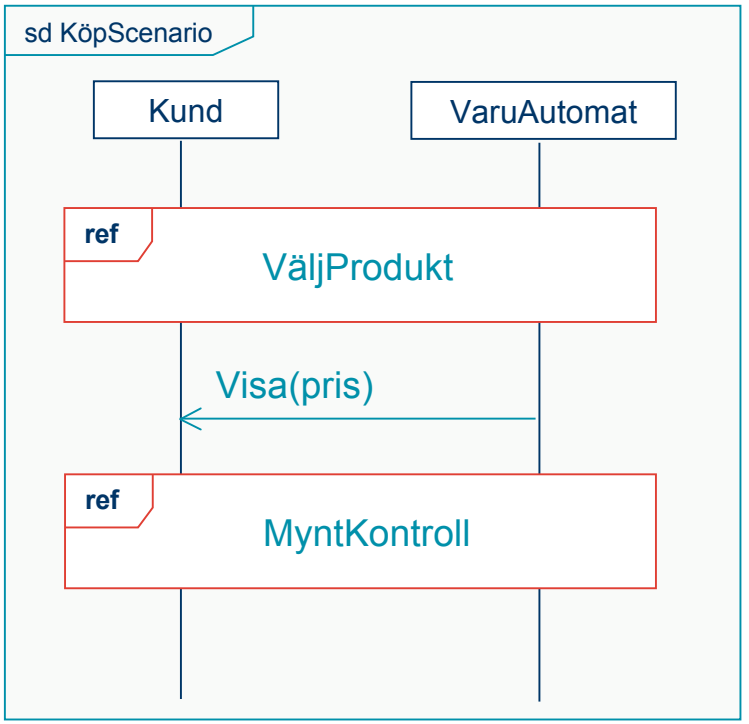
Sekvenser

- Syften:
  - uttrycka händelseförlopp
  - beskriva krav
  - tester och testsviter



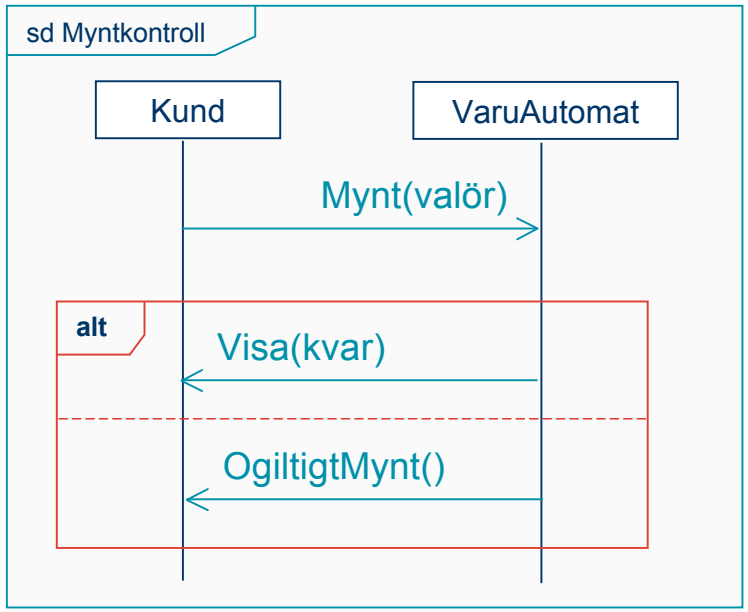
Referenser till andra sekvenser

- För att undvika duplicering av information kan man referera till andra sekvensdiagram
  - gör det möjligt att enkelt och snabbt skapa nya sekvenser (tester/krav) baserade på redan existerande sekvenser



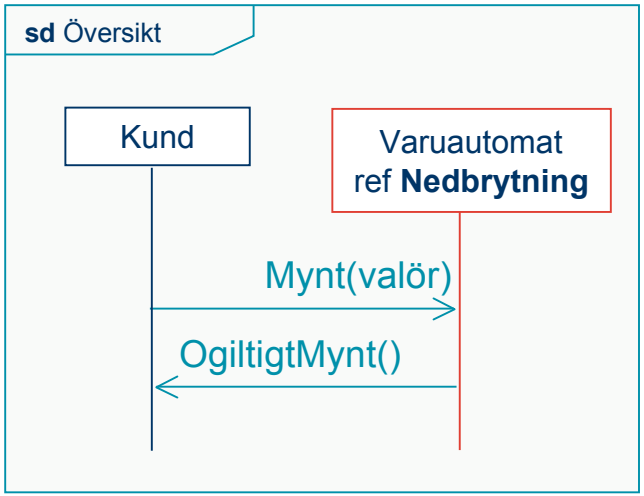
Variationer i sekvenser

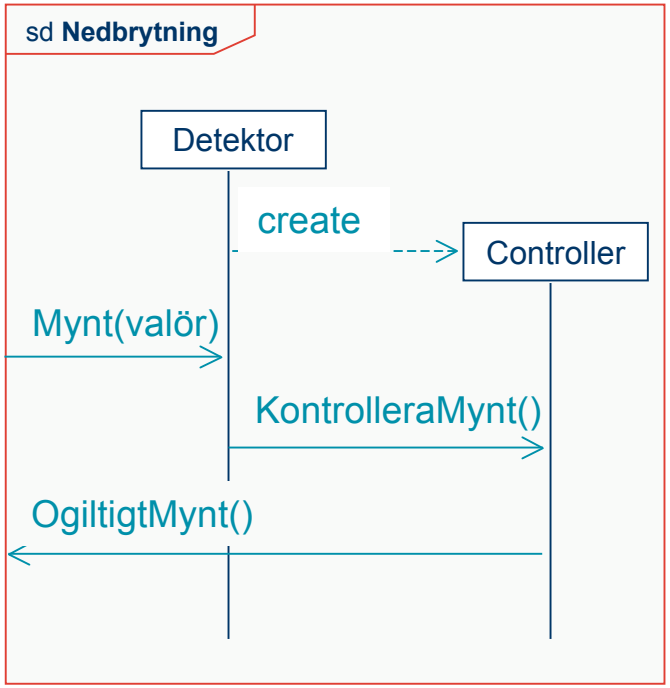
- Reducera antalet nödvändiga sekvensdiagram
  - för att uttrycka systemfunktionalitet hos större system krävs i regel många sekvensdiagram
- Variationer:
  - alternativ
  - iterationer (loopar)
  - möjliga händelser
  - parallellism



Nerbrytning av sekvenser (decomposition)

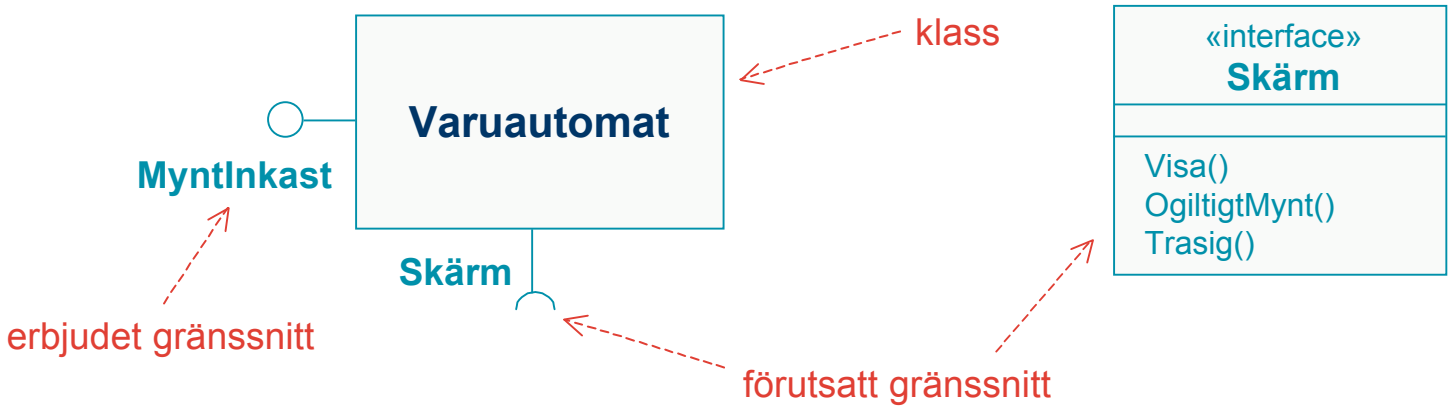
- Nerbrytning av livslinjer gör det möjligt att beskriva hur delsystem interagerar med varandra
  - zooma in för att få mer detaljer
  - zooma ut för att dölja detaljer





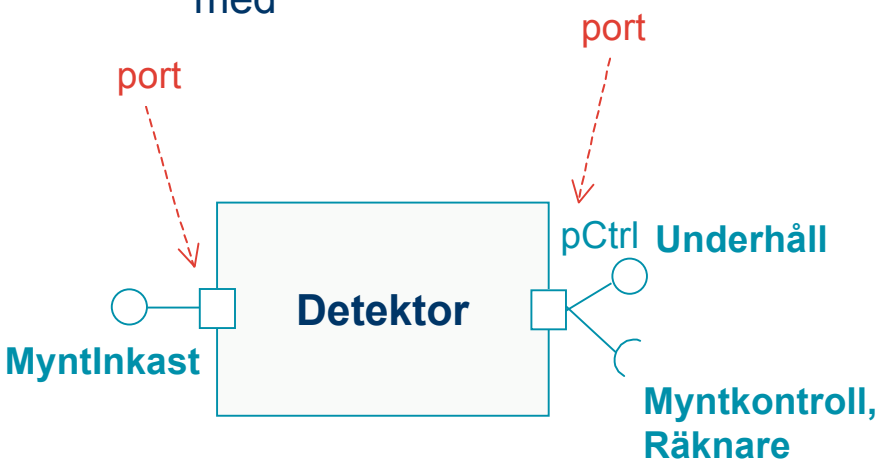
Komponentbaserad utveckling

- Gränsnitt mellan delsystem
  - erbjudna tjänster
    - ”provided interfaces”
  - förutsatta tjänster
    - ”required interfaces”
  - hur kan gränssnitten att användas?
- Inkapsling av implementation
  - struktur och beteende
  - ”black box”

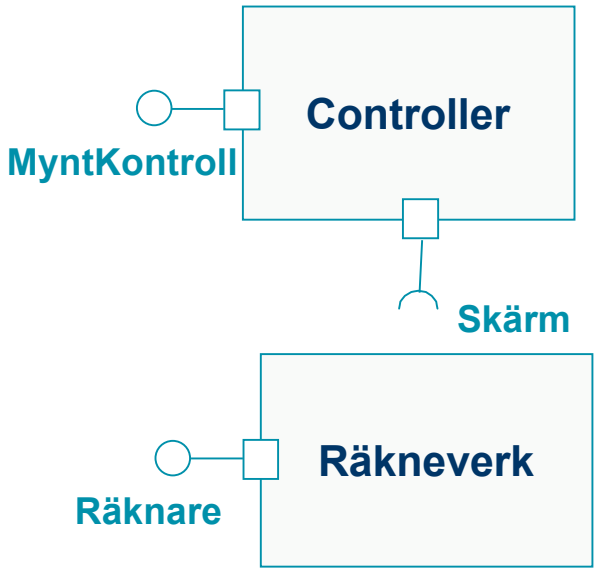


Kommunikation mellan klasser

- Enbart klasser med matchande gränssnitt kan kommunicera med varandra
  - kontrakt beskrivs direkt i modellen
  - möjliggör att avgöra vilka klasser man kan kommunicera med

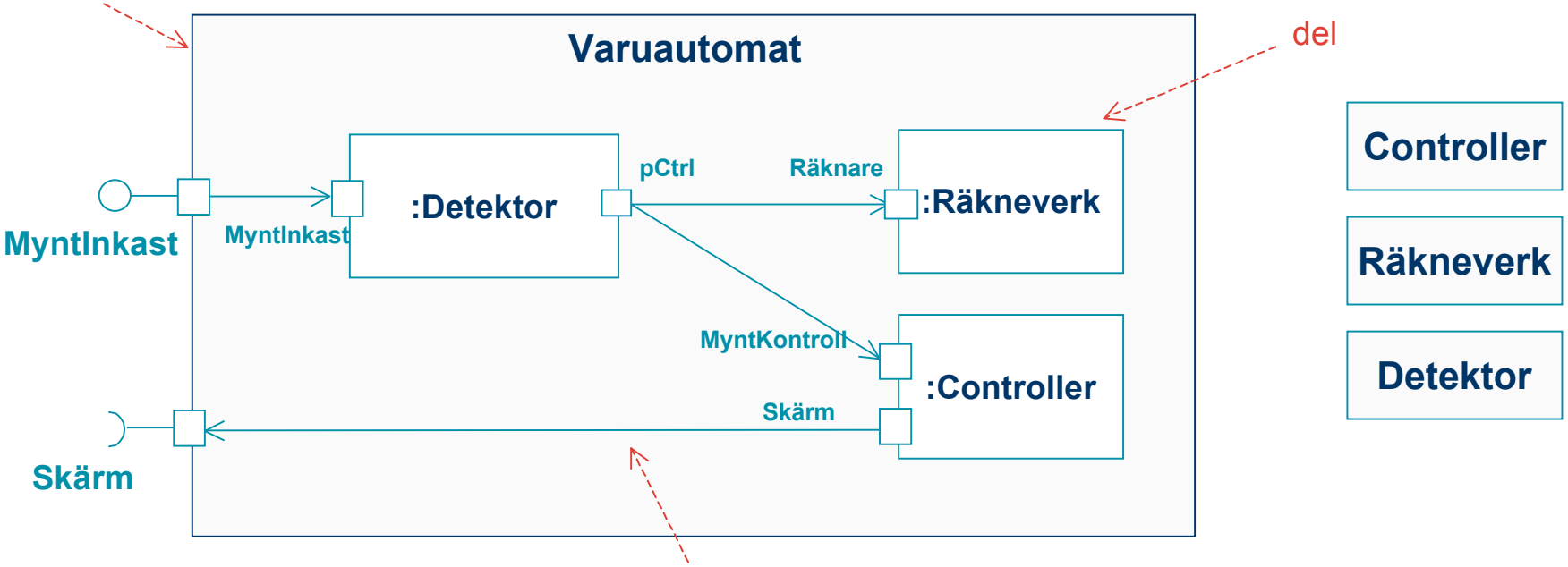


- En port (”port”) har flera roller:
  - knutpunkt för att koppla ihop klasser
  - ger en sammanhållen vy av en klass



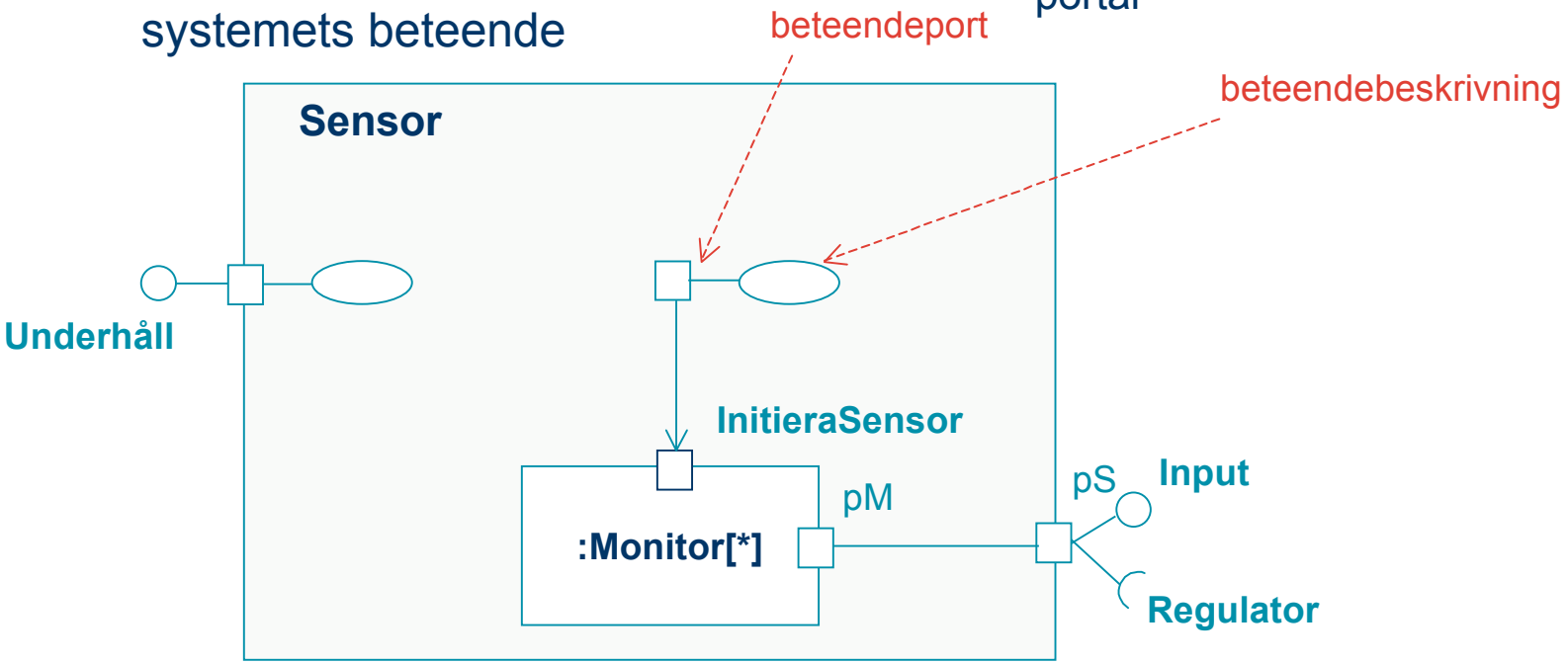
Hierarkisk nerbrytning av klasser

- Ett system kan brytas ner i sina beståndsdelar (”parts”)
  - hierarkisk nerbrytning; delarna kan i sin tur brytas ner ytterligare
  - ”white box”
  - livscykelberoende
- En kanal (”connector”) kopplar ihop delar med varandra
  - kontextberoende association
  - delarna kan kopplas ihop olika beroende på var de används
  - representerar giltiga kommunikationsvägar



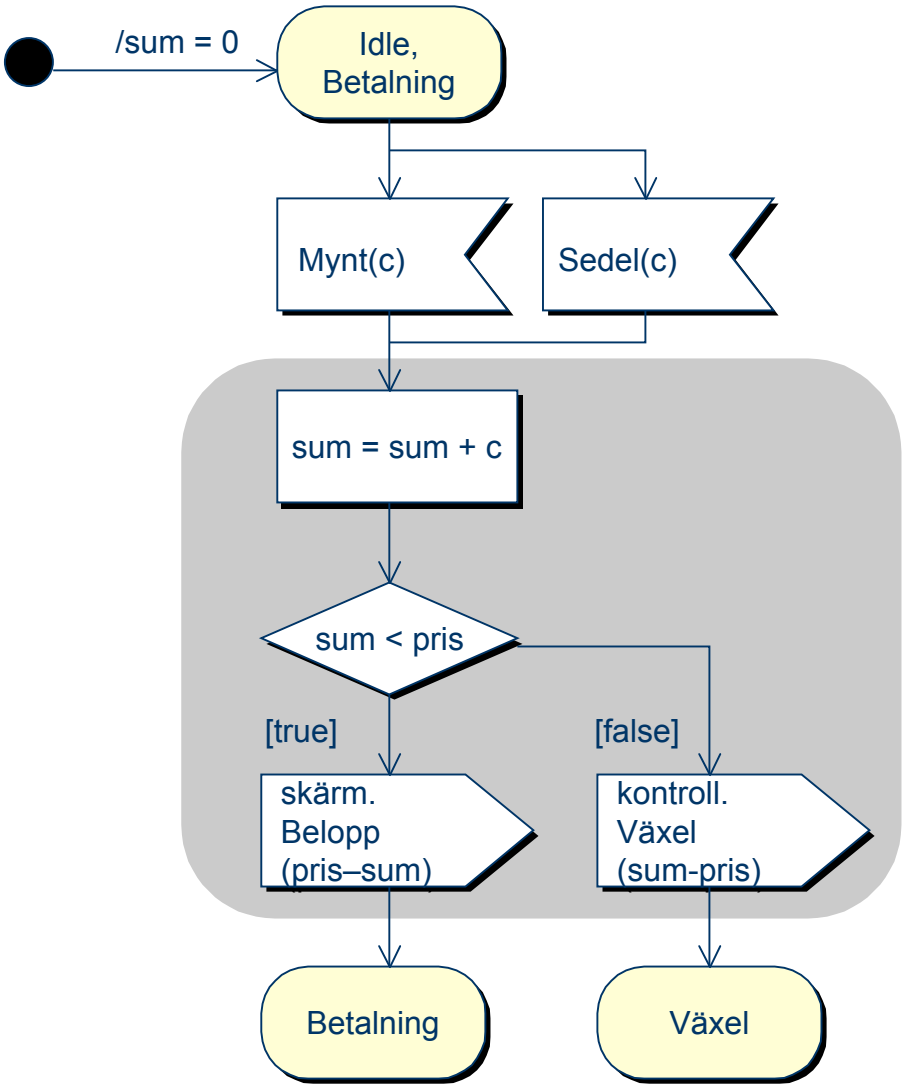
Blandning av struktur och beteende

- Beteendet hos ett system kan ges
  - direkt av systemet
  - av dess delar (delegering)
- En beteendeport kopplar till systemets beteende
- Från systemets omgivning bryr man sig inte om systemets implementation
  - utifrån sett skiljer man inte på beteendeportar och vanliga portar



Att uttrycka detaljerad funktionalitet

- För att uttrycka mer detaljerat beteende kan man beskriva satser (”actions”) på i princip samma nivå som i de flesta programmeringsspråk
  - ”action”-semantik för UML
  - ingen textuell syntax definierad i UML; detta kräver profiler
  - ger grunden för exekverbara modeller i UML
- Gör det möjligt för UML-verktyg att simulera, avlusa och testa modellen



Andra ändringar

- Tillståndsmaskiner
  - inkapslade tillstånd
  - in- och utgångar till tillstånd
- Aktiviteter
  - kopplas loss från tillståndsmaskiner
  - baseras på Petrinät
  - arbetsflöden (kontroll- och dataflöden)
  - verksamhetsmodellering
- Användningsfall
  - Bättre integration med språket i övrigt
- Beteende
  - ”action semantics”
  - bättre precision
  - exekverbara modeller
  - större uttrycksfullhet med fler satser
- Komponenter
  - bättre stöd för komponentramverk som .NET och J2EE
  - deployment
- ...
- Vi strävar efter att bibehålla **bakåtkompabilitet** så långt det går

Sammanfattning

- UML 2.0 ger grunden för modellbaserad utveckling
  - förbättrad användbarhet och skalbarhet
  - adresserar en bredare del av mjukvaruindustrin
  - uttrycker mer på en högre abstraktionsnivå

- Verktygsstöd kommer att avgöra hur väl modellbaserad utveckling faller ut
  - Telelogic Tau Generation2
    - Tau/Architect
    - Tau/Developer