

Forslag til hovedoppgaver

Birger Møller-Pedersen (birger@ifi.uio.no) rom 3301 – rett overfor buerommet.

Verktøy for modellering av systemfamilier

OMS er med i ITEA (EU) prosjektet Families (<http://www.esi.es/en/Projects/Families/>). Ideen i prosjektet er å komme frem til teknikker for å lage hele familier av systemer istedet for enkeltstående systemer, og selvfølgelig teknikker for å generere spesifikke systemer fra en systemfamilie. En systemfamilie vil typisk inneholde en del punkter, hvor forskjellige systemer i familien varierer (variasjonspunkter)

Norske deltakere er SuperOffice, Visma, AkvaSmart, DNV Software, Sintef og Ifi/OMS Internasjonale deltakere er bl.a. Philips, Siemens, Nokia, Thales.

- Oppgaven går ut på å utvikle et verktøy for å generere systemer basert på en UML (2.0) modell av systemfamilien - da i henhold til en gitt modell for variabilitet, f.eks. en modell som er tatt frem av Ifi og Sintef innen ITEA prosjektet Families.
- Verktøyet bør inneholde endel logikk for å hjelpe brukeren til å velge. Dette innebærer bl.a. å vise mulige valg (f.eks. komponent/klasse-kandidater, eller å spesifisere constraints (f.eks. på verdier) ved variasjonspunkter. Verktøyet bør kunne sjekke konflikter mellom valg og kunne generere den nye modellen.
- Oppgaven går også ut på å prøve ut om EMF-rammeverket innen Eclipse (www.eclipse.org) kan brukes til dette formålet. Eclipse er et open source verktøyrammeverk tatt frem av IBM.

Denne oppgave kan gjøres i samarbeide med tilsvarende Families-relatert oppgave for Øystein Haugen, f.eks. felles eksempel.

Kontaktperson Sintef: Jon Oldevik, taskleder for den del av ITEA prosjektet Families som har å gjøre med verktøy.

Fra variasjonsmodellering til variasjonsprogrammering

Innen Families prosjektet studeres *modellering* av systemfamilier: En familie av systemer inneholder variasjonspunkter, hvor de enkelte systemer vil være forskjellige og hvordan de kan være forskjellige. Det er utviklet forskjellige teknikker for å modellere slik variasjoner.

- Oppgaven går ut på finne ut hvordan man gjøre det tilsvarende innen *programmering*, dvs uten å modellere først? En del bedrifter vil gjerne lage familier av systemer, men bruker ikke modellering som en del av deres utvikling, så for dem vil dette være relevant.
- Oppgaven skal svare på om mekanismer som *generiske klasser* og *rammeverk* (frameworks) være nok? Hva med *aspekt-orientering*?
- Oppgaven skal gjennomgå de forskjellige variasjonsmekanismer innen modellering og finne tilsvarende mekanismer innen programmering, dernest applisere disse på et eksempel (en av fler muligheter: Access Control System/ATM/Watch) som er modellert (men ikke implementert) i Families prosjektet
- Oppgaven skal rapportere erfaringer og vurdere om modelleringsmekanismene kan overføres til programmering.

Fra informasjonsmodeller til forretningslogikkmodeller

Denne oppgaven gjøres i samarbeide med DNV (Det Norske Veritas).

I praksis brukes UML modellering mest til å lage informasjonsmodeller eller til å lage kodeskjellletter, hvor kode for forretningslogikk legges inn. DNV bruker UML til modellering av informasjonsmodeller f.eks. for system og funksjonsbeskrivelse av et skip, all skade historie for et skip, inspeksjonsplan for et skip.

Kilde for generering av div. kode: databaseskjema (både MS SQLServer og Oracle), dataaksesskode

DNV er interessert i å ta i bruk UML til modellering av forretningslogikk. De har gode erfaringer med informasjonsmodellering, og vil derfor gjerne bruke UML til noe mer.

Oppgaven går ut på å finne ut hva et første skritt for å få nytte av UML modellering for forretningslogikk bør være, og hvilken del av UML vil dette være?

- Eksempel vil bli valgt i samarbeid med DNV, slik at det blir et relevant eksempel
- Oppgaven skal også vurdere forskjellige verktøy for å generere kode fra en modell av forretningslogikk. Pr idag brukes IBM/Rational Rose, men Visio er også aktuelt. Det vil også være interessant å vurdere MS Visual Studio, som kommer med mer modelleringsstøtte i 2005..
- Oppgaven vil typisk kunne utformes som en anbefaling til DNV.

Kontaktperson DNV: Bjørn Egil Hansen

Språkdefinisjon ved metamodeller kontra grammatikker

En del språk har i den senere tid blitt definert ved hjelp av metamodeller. En metamodel er en objekt-orientert modell av et språk, hvor de forskjellige begreper i språket defineres ved hjelp av klasser og hvor for eksempel subklassing brukes til å definere spesielle språkbegreper.

Oppgaven går ut på å sammenligne bruk av grammatikker og metamodeller til både definisjon og implementasjon av språk. Hva er fordelene og hva er ulempene ved de to måten å definere språk på?

Oppgaven vil typisk inneholde

- Gjennomgang av de to teknikker
- Design av et språk, evt identifikasjon av et eksisterende språk
- Definisjon av dette språket ved hjelp av en metamodel og ved hjelp av en grammatikk
- Implementasjon av (deler av) språket
- Vurdering av fordeler og ulemper

Denne oppgaven egner seg for å bli laget av 2 studenter, hvor hver student tar for seg hver sin teknikk.

Traits + Java = Sant?

Det har blitt foreslått et nytt språkbegrep 'Traits' for å løse problemene med multiple inheritance. Dette begrepet er interessant, men desverre fungerer det bare for språk som ikke er typt, som f.eks. Smalltalk.

Oppgaven går ut på å finne ut om dette begrepet også kan brukes for et typt språk (f.eks. Java) eller om det må gjøres tilpasninger? Hva vil utfordringen være for et typt språk a la Java eller C#?

Oppgaven vil typisk kunne inneholde:

- En undersøkelse av nytten av traits ved å gå gjennom et eksempel på et stort Java bibliotek
- Tilpasning av traits for et typet språk
- Vurdere om traits for Java kan implementeres på eksisterende JVM.

Denne oppgaven er koplet til et mulig PhD prosjekt som det søkes NFR støtte til:
SWAT: Semantics-preserving Weaving - Advancing the Technology

Traits er beskrevet her: <http://www.iam.unibe.ch/~scg/Archive/Papers/Scha03aTraits.pdf>.

Domenespesifikke språk

Det virker som om Microsoft kommer til å satse på verktøystøtte for domenespesifikke språk som et alternativ til APIer i programmeringspråk og som et alternativ til f.eks. UML for modellering. Det finnes allerede en del verktøy for å lage slike språk, og det er noen gode erfaringer, men det virker som om disse domenespråk lages 'fra gang til gang', dvs uten noen bakenforliggende teori.

Oppgaven går ut på å ta frem en slik teori, ved å

- gjennomgå en rekke domenespesifikke språk og forsøke en karakterisering. Gjøre klar forskjellen på domenspråk og vanlige rammeverk (frameworks) eller APIer?
- Vurdere forskjellen ved å kjøre igjennom samme eksempel ved både domenespråk, rammeverk og API.

Oppgaven skal også svare på disse spørsmål:

- Hvordan implementere domenespesifikke språk? Noen fordeler/ulemper ved metamodeller contra grammatikker?
- Hvordan uttrykke at sub-domenespråk (eller fler språk innen samme domene) har felles egenskaper? Dette er pr. idag ikke løst innen dette feltet
- Hvordan kombinere fler domener?

Bakgrunn:

- Metacase (<http://www.metacase.com/>) – finsk bedrift som har spesialisert seg på domenespesifikke språk, som f.eks. brukes av Nokias mobiltelefoner
- OOPSLA workshop on domain languages (<http://www.metacase.com/news/OOPSLA2004.html>)