Sto på trykk i Elektronikk nr. 10, 2001

Modulær testing basert på universelle lover

I hovedfagsarbeidet mitt viser jeg en ny metode for modulær testing av distribuerte systemer. Metoden er basert på et matematisk prinsipp som kan forstås som en universell lov for systemutvikling. Metoden gjør også bruk av automatisert testing av eksekverbare spesifikasjoner.

Av Mass Soldal Lund

Systemutviklingen som fagfelt er i stor grad preget av pragmatiske metoder, dvs. metoder og teknikker som på en eller annen måte har vist seg å virke noenlunde tilfredsstillende. På sett og vis er pragmatismen den ledende ideologien inne dagens systemutvikling. Dagens datasystemene blir stadig større og mer komplekse, og kravene til sikkerhet og pålitelighet øker. Det ikke gitt at de pragmatiske metodene har et solid nok fundament til å håndtere dette.

Formalisering av systemutviklingen
Formalisering av systemutvikling har mange ulike aspekter, men kalles gjerne formelle metoder med et fellesnavn. De formelle metodene har en historie som er like lang som den elektroniske datateknologien.

På 1930-tallet kom Turing-maskinen og på 1940-tallet Churchs lambdakalkyle, begge deler matematiske modeller for beregninger, som på den tiden ble utført manuelt av mennesker. På 1960-tallet begynner enkelte miljøer av informatikere å lage matematikkbaserte systemer for å kunne resonnere rundt dataprogrammer og programmeringsspråk.

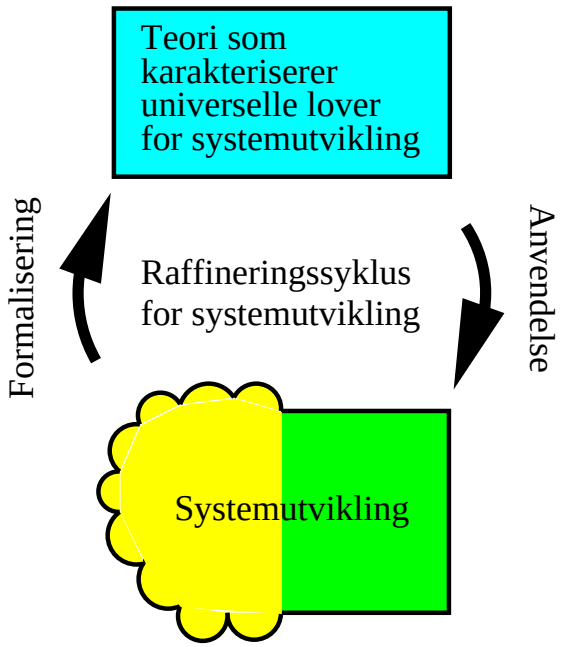
Det meste av arbeidet dreide seg om formalisering, som f. eks. å beskrive semantikken (meningsinnholdet) til et programmeringsspråk utvetydig og matematisk presist, men det har også vært forsøk på å lage formelle systemutviklingsmetoder. Best kjent er antagelig Hoares programlogikk fra slutten av 1960-tallet. I denne Hoare-logikken kan korrektheten av et program kan bevises matematisk ved hjelp av logiske regeler, men den var også ment å brukes til å definere programmeringsspråk.

Fra 1970-tallet og utover begynte man også å se på systemer med parallellitet, og på 1980 og -90-tallet har mye av arbeidet med formalisering vært konsentrert rundt dette.

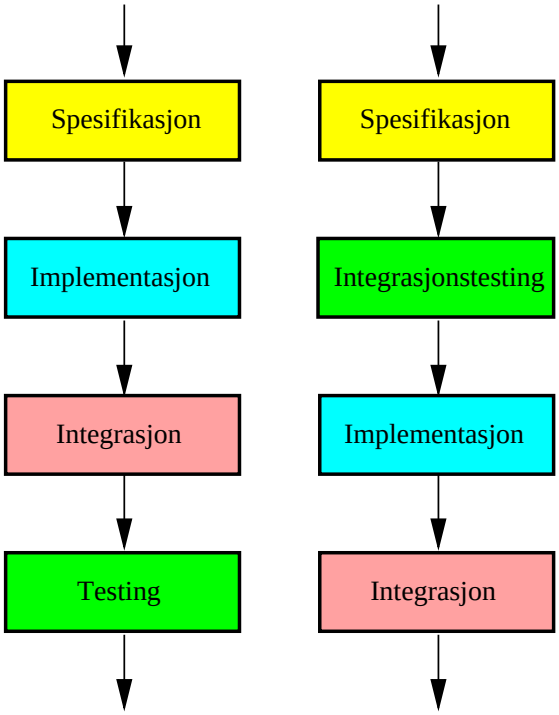
På tross av at det har blitt lagt ned betydelige forskningsresurser i arbeidet med de formelle metodene, har de aldri slått gjenom som verktøy for systemutvikling. Mye av grunnen har vært at de var vanskelige å forstå uten spesiell interesse for logikk, og at de har vist seg å være alt for arbeidskrevende til å anvende i praksis.

Universelle lover
Forsøkene med å anvende de formelle metodene direkte i systemutvikling har vært lite vellykket. Likevel kan man ikke si at arbeidet har vært uten resultater. De teoretiske prinsippene man kom fram til er like gyldige, og kan sies å konstituere universelle lover for systemutvikling.

I figur 1 er formaliseringen av systemutviklingen representert ved pila til venstre. Dersom de universelle lovene kan omsettes til praktisk anvendbare metoder for systemvikling, representert ved pila til høyre, kan dette gi systemutviklingen et mer solid fundament. Som forsøkt vist i figuren kan hele prosessen, eller syklusen om man vil, med formalisering og anvendelse gjøre systemutviklingen mindre diffus. Man endre opp med en dypere forståelse av systemutvikling og mer raffinerte systemutviklingsmetoder.



Figur 1



Figur 2

På samme måte som metodene for konstruksjon av bruer og raketter må basere seg på mekanikkens lover, burde metodene for systemutvikling basere seg på de universelle lovene denne formaliseringen tilbyr oss.

Skal det konstrueres praktisk anvendbare metoder basert på lovene, er det selvfølgelig en forutsetning at matematikken som ligger bak skjules for den som skal anvende metodene. Målet må være at slike metoder skal kunne anvendes av systemutviklere uten spesialkompetanse i logikk.

Et av disse universelle lovene er Martín Abadi og Leslie Lamports prinsipp for komposisjon av kontraktsorienterte spesifikasjoner. Prinsippet er først formulert for det logikkbaserte spesifikasjonsspråket Temporal Logic of Actions og er en regel for forenkling av beviser for egenskapene til komponentbaserte systemer.

Hovedfagsarbeidet mitt går ut på å omsette dette prinsippet til en metode for praktisk testing av komposisjon og vise at denne metoden lar seg realisere i et automatisk testeverktøy basert på eksekverbare spesifikasjoner og test case-generering.

Integrasjonstesting på spesifikasjonsnivå

Alle datasystemer av en hvis størrelse vil nødvendigvis bestå av mange komponenter. Under utvikling av store systemer vil ulike grupper mennesker arbeide på ulike deler av systemet; ulike komponenter kan være satt bort til ulike underleverandører. Bruk av standardkomponenter og gjenbruk komponenter er et ideal på den mer økonomiske siden av et utviklingsprosjekt.

I dette ligger det en stor utfordring i integrasjon av komponentene. Det er ikke nok at en komponent i seg selv har den rette funksjonaliteten, den må også være i stand til å kommunisere med omgivelsene den settes inn i. En vellykket integrasjon forutsetter at alle komponentene i et system havner i forventede omgivelser.

En av tankene som ligger til grunn for prosjektet er at integrasjonstesting bør gjøres på spesifikasjonsnivå. Figur 2(a) viser er utsnitt av en klassisk systemutviklingsprosess. Det er ikke uvanlig at integrasjonstesting ikke blir gjort før komponentene i et system er implementert. Skulle det da vise seg at komposisjonen av komponentene ikke har det den effekten man ønsket seg, risikerer man å møtte gå tilbake og gjøre endringer i implementasjonen.

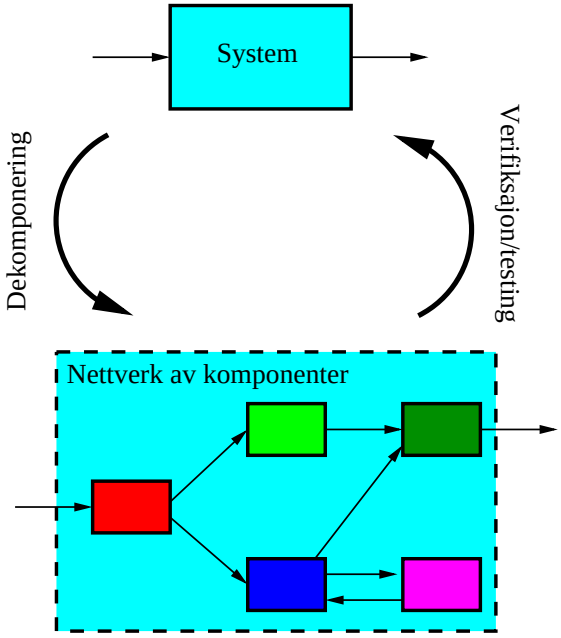
Gjøres integrasjonstestingen på spesifikasjonsnivå, dvs. før implementasjon av komponentene som vist i figur 2(b), reduseres sjansen for at de implementerte komponenten ikke lar seg integrere. Det er grunn til å tro at dette kan gi betydelige besparelser i systemutvikling, og at det gir mer pålitelige systemer.

Kontraktorienterte spesifikasjoner

Metoden er basert på kontraktsorienterte spesifikasjoner. Kontraktorientert spesifikasjoner er, som navnet antyder, spesifikasjoner som er som er formet som en kontrakt. Spesifikasjonen for en komponent består av en antagelse og en garanti. Antagelsen beskriver under hvilke forutsetninger, dvs. i hvilke omgivelser, komponenten virker som spesifisert, mens garantien beskriver komponentens korrekte oppførsel. Kontrakten består i at komponenten skal oppføre seg som spesifisert av garantien så lenge omgivelsene til komponenten oppfyller antagelsen.

Denne måten å lage spesifikasjoner på har klare fordeler. En fordel er at det gir et logisk skille mellom en komponents funksjonalitet og de betingelsene komponenten treger for å virke. En kanskje viktigere fordel er det gir et spesifikasjonsformat som forteller hva man må kreve av omgivelsene for at komponenten skal virke som forutsatt. En måte å si det på kan være at det er et spesifikasjonsformat hvor

man forsøker å unngå ubehagelige overraskelser når komponenter kombineres ved å angi kravene til omgivelsene eksplisitt. For standard- og gjenbrukskomponenter vil det være en opplagt fordel å vite nøyaktig i hvilken kontekst de er tenkt å virke.



Figur 3

Testing av komposisjon
Figur 3 viser spesifikasjonen av et system sett som et hele, dekomponert til et nettverk av spesifikasjoner av komponenter. Dekomponeringen er representert ved pila til venstre. For å forsikre seg om at dekomponeringen er gyldig, trenger man en metode for å undersøke om de ytre egenskapene til den nedre spesifikasjonen er de samme som (eller en forfining av) de ytre egenskapene til den øvre spesifikasjonen. Dette er vist ved pila til høyre.

Denne prosessen kan med fordel anvendes iterativt. Etter en dekomponering er verifisert, kan hver av komponentene igjen dekomponeres. På denne måten kan man arbeide seg fra en spesifikasjon for et system sett som en enhet, ned til spesifikasjoner for komponenter som er detaljerte nok til å implementeres.

Dette er ingen ny ide; den finnes bl. a. i Börje Langefors' fundamentalprinsipp for systemutvikling fra 1966. Derimot finnes det ingen kjent metode som gjør dette automatisk og som har et matematisk-logisk fundament som sikrer at verifikasjonen er korrekt. Spesielt parallellitet i systemer gjør at denne formen for verifikasjon ikke er en triviell oppgave, så en slik metode har sin misjon.

Metoden er generell nok til å ha anvendelse utover software-utvikling. Det er ingen grunn til at den ikke skal kunne benyttes f. eks. ved utvikling av hardware-komponenter. En annen mulig anvendelse kan være testing av operatørprosedyrer for systemer hvor menneskelige operatører spiller viktige roller. Operatørprosedyrer vil ofte la seg beskrive med standard spesifikasjonsteknikker. Metoden kan brukes til å kontrollere om en prosedyre, anvendt av en operatør i samspill med resten av systemet, har den ønskede effekten. Forutsatt at systemet er spesifisert kan dette gjøres på spesifikasjonsnivå, dvs. uten at systemet settes i drift.

Verktøy for automatisert testing
Metoden skal kunne realiseres i et automatisk verktøy. Den er generell nok til å kunne anvende alle typer spesifikasjonsteknikker, så lenge de er eksekverbare. Et naturlig valg er å bruke tilstandsdiagrammer, siden dette er en teknikk som er svært utbredt. En forskjell fra vanlig bruk av tilstandsdiagrammer er at hver komponent må spesifiseres ved hjelp av to diagrammer; et for antagelsen og et for garantien. Komposisjon spesifiseres ved hjelp av dataflytdiagrammer, siden metoden baserer seg på testing av input- og outputstrømmer.

Jeg arbeider for tiden med å prøve ut metoden i et verktøy for automatisert test case-genering fra tilstandsdiagrammer.

Konklusjon
Arbeidet mitt viser at det er mulig å omsette universelle lover fra formalisering av systemutvikling til praktisk anvendbare systemutviklingsmetoder. Mer spesifikt et verktøy for testing av komposisjon basert på Abadi og Lamports prinsipp for komposisjon. Arbeidet gjøres som hovedfagsarbeid ved Institutt for Informatikk, Universitetet i Oslo, og er veiledet av Ketil Stølen ved SINTEF Tele og data.

Systemutviklingen er ennå et ungt fagfelt. Selv om de formelle metodene av mange har blitt avvist som ubrukelige i praksis, kan det være grunn til å tro at de likevel har nytte. Det er et behov for presise og gode systemutviklingsmetoder, og formelle metoder har gitt opphav til et stort antall lover og prinsipper som kan vise seg å være det teoretiske fundamentet systemutviklingen har et helt klart behov for.