Utveckling av datorprogram för driftsäkerhetsanalys

Detta dokument beskriver typiska uppdrag för Trilogik Konsult AB inom ett område som vi har valt att kalla Utveckling av datorprogram för driftsäkerhetsanalys, men som även omfattar närliggande fält som t.ex. underhållsoptimering. Mycket av metodiken inom detta område kan också tillämpas när man har behov av att bygga matematiska modeller inom andra problemområden.

1. Om begreppen Modellering och Simulering

Begreppen, eller ska man kanske hellre säga begrepp**et** Modellering och Simulering syftar på ett arbetssätt där man, oftast inför en viktig beslutssituation, använder **matematiska modeller** för att utforska hur ett komplext system fungerar.

Till stor del handlar Modellering och Simulering naturligtvis om att direkt prova olika beslutsalternativ. Tanken är att man ska göra eventuella **misstag "i modellen" istället för i verkligheten**. En annat resultat av att arbeta med modeller - ibland närmast en "biprodukt" av att prova och jämföra de olika beslutsalternativen - är att man ökar sin förståelse för det studerade systemets egenskaper och beteende. Detta är nog så viktigt, särskilt som sådana mer principiella insikter ofta utgör betydligt säkrare slutprodukter än de enskilda kvantitativa resultaten - "siffrorna" - från den matematiska modellen. I många fall finner man paradoxalt att de insikter modeller ger egentligen är "självklara", i den betydelsen att man kan komma fram till samma insikter med ett relativt enkelt logiskt resonemang. Det betyder dock inte att modellen varit onödig; man behöver dess stöd för att kunna hitta det logiska resonemanget. Dessutom är ju även de kvantitativa resultaten av intresse!

Det här arbetssättet lämpar sig speciellt väl i situationer när det är **svårt eller omöjligt att prova olika alternativ i verkligheten**. Utveckling av ny teknik är ett exempel. Inom försvarssektorn har man av helt andra skäl ofta svårt att "prova i verkligheten". Det är då inte förvånande att man speciellt vid upphandling/utveckling av nya system inom försvarsområdet har funnit att Modellering och Simulering är en lönande verksamhet. I USA pågår ett omfattande arbete inom Department of Defence (DoD) med speciell inriktning just på upphandling och utveckling av vapensystem med stöd av simulering (Simulation Based Acquisition, SBA). I något mindre skala,

men därmed inte nödvändigtvis mindre framgångsrikt, har även Försvarets Materielverk, FMV, använt simulering vid upphandling åt det svenska försvaret. Dessutom finns många framgångsrika tillämpningar som inte har gällt upphandling, utan beslutsstöd för t.ex. underhållsoptimering eller organisationsutveckling inom försvaret.

Trilogiks fokus ligger i detta sammanhang på driftsäkerhet och underhållskostnader, men Modellering och Simulering är givetvis tillämpligt för mycket mer än dessa områden. (I själva verket betraktar väl många våra tillämpningar som lite udda specialfall i den stora mängden av produktionssimulering, simulering av tekniska prestanda, transportsystem etc.) Vi ser ju också att många av de idéer och tekniker vi använder är tillräckligt generella för att kunna tillämpas på helt andra problemområden.

Hos Trilogiks konsulter finns en omfattande erfarenhet av Modellering och Simulering, och då menar vi både utveckling och användning av modellerna. Exempel från försvarssektorn är utveckling av ett antal olika simuleringsmodeller för flygplan/flygbaser (modellen **ASTOR**), fartyg och stridsvagnar.

Inom den civila sektorn är det mest notabla exemplet en simuleringsmodell som användes av SJ vid upphandlingen av snabbtåget X-2000. (Detta skedde givetvis långt innan Trilogik fanns till som företag, vilket för övrigt även gäller flertalet av de militära modellerna.)
Användningen av alla dessa modeller förtjänar ett helt eget kapitel, vilket det i någon mån även har fått (se Dimensionering av underhållssystem). Här ska vi därför helt koncentrera oss på utvecklingen av modellerna.

2. Modellutveckling och specificeringsarbete

Att ta fram en simuleringsmodell för driftsäkerhetsanalys eller liknande omfattar, liksom annat systemutvecklingsarbete, faserna analys, design och implementation (samt test, inte att förglömma). I de senare faserna skiljer sig inte simuleringsmodellen så mycket från andra typer av programvara, och även i analysfasen finns givetvis mycket som är allmängiltigt. I vissa avseenden är dock en simuleringsmodell helt annorlunda än både affärssystem och teknisk programvara.
Till att börja med kan det vara värt att påpeka att ordet simuleringsmodell på detta stadium kan ges en mycket generell mening. Vår erfarenhet, speciellt under senare år, gäller först och främst egentliga simuleringsmodeller, där datorprogrammet på en ofta ganska detaljerad nivå efterliknar verklighetens olika processer och aktörer. Under analysfasen, speciellt dess inledande delsteg, ser dock arbetet ungefär likadant ut om man planerar att använda mer direkta matematiska beräkningsmetoder, dvs analytiska algoritmer och formler. Det kan också vara så att man avser att applicera en optimeringsfunktion på modellen; det förutsätter dock den analytiska lösningsmetoden, eftersom en egentlig simulering är för långsam.

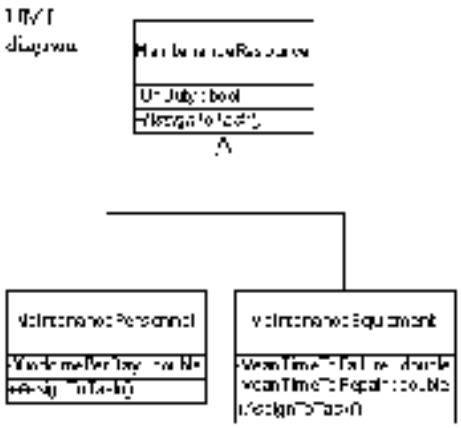
Utveckling av en ny simuleringsmodell, eller vidareutveckling av en existerande modell, inleds med att en beskrivning av det studerade scenariot arbetas fram. Vilka av verklighetens företeelser är det man vill kunna analysera med modellen? På detta stadium behöver beskrivningen inte vara speciellt komplett eller matematiskt stringent, utan önskemålen kan anges översiktligt. Under detta arbete krävs i första hand kunskap om det problemområde som ska studeras, medan specialistkunskap om modellbyggande är mindre viktigt. Det är dock önskvärt att det finns en grundläggande förståelse för de möjligheter som matematiska modeller erbjuder, liksom om deras begränsningar.
Det är en fördel om man redan här kan **precisera syftet med modellen**. Det finns flera frågor som bör besvaras:

- Kommer det att vara viktigt att kunna optimera något (eller några) kvalitetsmått, t.ex. kostnad eller tillgänglighet. Vilka parametrar vill man kunna variera vid optimeringen?
- Vilka delar av modellen vill man i första hand kunna studera? Inom vilka områden kommer man att studera och jämföra resultat för olika data och förutsättningar? Dessa delar av modellen bör göras mer detaljerade. Andra områden kan oftast modelleras mer approximativt, eftersom modellfelen då blir ungefär lika stora för alla analyserade alternativ. Beräkningsfelen i skillnaderna mellan alternativen kommer då att bli små.
- Vilka beräkningstider kan man acceptera? Är hög precision i resultaten viktigare än prestanda?
- Hur generell behöver modellen vara? Kan den skräddarsys för en viss tillämpning eller vill man kunna återanvända modellen i senare tillämpningar?

Nästa delfas är en formalisering av den matematiska modellen. Det grundläggande scenariot ska nu definieras i detalj och dess olika indataparametrar ska identifieras. Här krävs djupa kunskaper i att bygga matematiska modeller, men också kännedom om problemområdet. En nära dialog mellan dessa två kompetensområden är nödvändig och ju mer "kompetensöverlapp" som finns mellan de två specialistkategorierna, desto större är förutsättningarna för ett lyckat resultat. Slutresultatet från denna fas är en matematisk modell "på papper", en modellspecifikation. Modellen beskrivs ofta åtminstone till någon del på textform, men olika grafiska beskrivningssätt är förstås mycket användbara. Ett objektorienterat synsätt brukar falla sig naturligt, speciellt för egentliga simuleringsmodeller.

Till stor del kan man använda **samma modelleringstekniker som för tekniska eller administrativa system**, t.ex. är ett objektorienterat, grafiskt modelleringsspråk som UML mycket användbart. På ett ytligt plan kan UML-diagram för en simuleringsmodell vara mycket lika motsvarande diagram för ett administrativt system inom samma område. Det finns dock vissa subtila men ändå viktiga skillnader i syftet med beskrivningen och i innebörden av de olika modellelementen. Detta hänger främst samman med att man för simuleringsmodellen mer primärt modellerar verkligheten utan någon egentlig relation till den programvara som ska tas fram. Programvaran (den som senare ska utvecklas) är alltså inte en komponent i modellen utan snarare "synonym" med hela UML-modellen.

I programvara som implementerar matematiska modeller är oftast **formler och algoritmer** centrala och betydligt viktigare än i t.ex. ett administrativt datorsystem. Detta gäller förstås speciellt modeller av den analytiska varianten, som helt baseras på formler och algoritmer. Även i egentliga simuleringsmodeller kan det förekomma en hel del algoritmer i form av beslutsregler. Dessa kan motsvara en mänsklig beslutfattare i det verkliga studerade systemet. Ofta blir beslutsreglerna rätt komplexa och måste ibland uttryckas med hjälp av optimeringsalgoritmer. Syftet med sådana inbäddade optimeringsalgoritmer är dock sällan att modellens beslutsregler ska definiera ett helt optimalt agerande utan snarare att undvika att modellen ibland gör riktigt dåliga val. Detta kan vara nog så svårt att uppnå.



I simuleringsmodeller som innehåller slumpmässiga processer (vilket i hög grad gäller inom driftsäkerhetsområdet) förekommer också en hel del sannolikhetsteori och matematisk statistik, dels för att generera slumpprocesserna och dels för att behandla resultatet från simuleringen och presentera det i form av medelvärden, standardavvikelser, kvantiler mm. (Mer om resultatpresentation.)

Som en grundregel bör en modellspecifikation definiera algoritmer tillräckligt noga för att de ska vara matematiskt entydiga. Man behöver däremot inte i detalj beskriva hur en viss matematisk operation ska utföras i programmet, åtminstone inte om det finns kända lösningsmetoder. **Trilogiks styrka** vid utveckling och specificering av simuleringsmodeller är framför allt att kombinera kunskaper i matematik, simuleringsmetoder och generell programutveckling. Vad gäller modeller för driftsäkerhetsanalys har vi dessutom djupa kunskaper och stor erfarenhet inom själva problemområdet.

3. Programdesign och kodutveckling

Implementationen av en simuleringsmodell kan göras i något "skalprogram" med speciellt stöd för simulering ("simuleringsskal") eller med ett generellt programmeringsspråk, där mer eller mindre simuleringsstöd kan finnas inbyggt. (Vi talar här - och fortsättningsvis nedan - om egentliga simuleringsmodeller. För en analytisk modell använder man förstås inte ett skalprogram för simulering, men det finns däremot en hel del programvara med stöd för matematiska beräkningar.) Fördelen med att **använda ett simuleringsskal** är främst att man snabbt kommer igång och får fram en körbar modell. Man kan också bygga en modell utan att behöva så stora kunskaper i traditionell programmeringsteknik. I enklare fall ska man inte alls behöva skriva någon programkod, men i praktiken visar det sig viss kodning ändå behövs. Detta kan bland annat bero på att verktyget (simuleringsskalet) inte i första hand är avsett för att modellera driftsäkerhet.

Man ska också ha klart för sig att kompetens och metodik från traditionell utveckling av programvara krävs vid implementation av större modeller. Om man försummar detta kommer man snart att tappa greppet om modellen och förlora möjligheten att införa modifieringar och nya funktioner. Förtroendet för modellens resultat eroderas då också snabbt. Det är alltså viktigt att åtminstone mer omfattande modeller byggs på ett strukturerat sätt och dokumenteras väl. Modellen ska också testas på samma sätt som konventionell programvara.

Implementation med ett **generellt programmeringsspråk** kräver i allmänhet en betydligt större arbetsinsats innan en körbar modell finns framme. Fördelen är här framför allt att man får bättre detaljkontroll över modellen. Man begränsas inte av ett förbestämt beteende i ett simuleringsskal som man inte kan påverka. Detta gäller framför allt själva modellen, dvs simuleringsförloppet, men även utformingen av indatering och resultatpresentation.

En annan fördel är att man i allmänhet kan uppnå betydligt bättre prestanda än med ett simuleringsskal, vilket ofta är mycket viktigt för större simuleringsmodeller. För stora simuleringsmodeller som förväntas leva länge och vidareutvecklas kan man därför rekommendera angreppssättet med ett generellt programmeringsspråk. Det brukar också visa sig att när man väl har kommit över den första tröskeln med att använda ett generellt programmeringsspråk så är det inte så mycket mer arbetskrävande att införa nya funktioner än när man använder ett simuleringsskal.

En fråga är vilket programmeringsspråk man ska välja för en simuleringsmodell. Det är givetvis en fördel om det finns inbyggt stöd i språket för simulering, men vår erfarenhet är att det går ganska lätt att själv skapa sådana funktioner i vilket generellt språk som helst, åtminstone för programspråk som stödjer objektorientering. **Objektorienterade metoder** visar sig också vara naturliga att använda för realisering av själva modellen, dvs definitionen av beteendet hos modellelementen och deras interaktion. Här är det givetvis en fördel om man redan i analysfasen har använt objektorienterade metoder och beskrivningssätt, men det är inte på något sätt en förutsättning för att använda sådana metoder i kodningen.

Inom Trilogik har vi framför allt arbetat med det generella objektorienterade **programspråket C++** som verktyg för utveckling av simuleringsmodeller. Det har passat vår inriktning mot rätt stora, komplexa och cpu-krävande modeller. Vi har dock stor erfarenhet även av utveckling i högnivåverktyg, t.ex. Anylogic.