NYHETSBLAD FÖR SESAM
Försvarssektorns Adaintressenters Användargrupp för Software Engineering

RENDEZVOUS

Nr 2 juni 2000

Innehåll

Försvarssektorns Adaintressenters Användargrupp för Software Engineering

SESAM

Vad är SESAM?

SESAM har tillkommit för att organisera och stimulera samarbete och samverkan inom programvaruområdet mellan försvarsindustrin, FMV och FOA.

Det avtalsfästa syftet med SESAM är ”att genom organiserat samarbete mellan användargruppens medlemmar främja tillförlitlighet och effektivitet i utveckling och vidmakthållande av programvarusystem i Ada inom försvarssektorn”. Inom ramen härför skall SESAM även anpassa, profilera och förnya sin verksamhet med hänsyn till ändrade tekniska och andra omständigheter av betydelse för intresseområdet.

Följande kommer att ske under den närmaste 2-3-årsperioden.

1. SESAM skall allmänt verka för att sprida information om faktorer som påverkar möjligheterna till tillförlitlig och effektiv utveckling och vidmakthållande av programvarusystem. Särskilt skall härvid Adas betydelse i sammanhanget klargöras.

2. SESAM skall i sin verksamhet fortlöpande bevaka möjligheterna att samla, skapa och sprida information om objektiva mät- och andra resultat och erfarenheter vunna vid användning av ”software engineering”-principer och Ada.

3. SESAM behandlar tillvägagångsättet vid utveckling och vidmakthållande av programsystem. Implicit i detta ligger givetvis att använda processer skall tillförsäkra de resulterande produkterna efterfrågade egenskaper. Produktegenskaper som påverkas av processerna är därför av primärt intresse att bevaka i SESAMs verksamhet.

4. SESAM skall i sin verksamhet fästa stor vikt vid att underlätta samexistens mellan Ada-program och programvara skriven i andra språk. Speciellt skall aspekter vid användning av COTS beaktas.

5. SESAM skall där så är möjligt sätta konkretiserade och mätbara mål för sin verksamhet under avgränsade tidsperioder.

SESAM styrs av ett Råd med representanter för gruppens medlemmar. Rådet har till sin hjälp ett Verkställande Utskott (VU) och ett sekretariat.

Rådets ordförande är Claes Wadsten, AerotechTelub, tel 013-231652 .

VU

Andersson Tommy, Ericsson Microwave Systems
tommy.andersson@emw.ericsson.se
Bengtsson Christopher, FMV
chben@tranet.fmv.se
Brandt Roger, FMV
robra@fmv.se
Carlsson Ingemar, adjungerad
ingemar.carlsson@mbox2.swipnet.se
Folkesson Dag, Saab AB
dag.folkesson@saab.se
Johansson Billy, CelsiusTech Electronics AB
bijo@celsiustech.se
Merkell Curt, Bofors
curt.merkell@celsius.se
Wadsten Claes, AerotechTelub
claes.wadsten@aerotechtelub.se

Arbetet utförs i ett två arbetsgrupper:

Ag Metodik
Håkan Edler, CTH/Datorteknik
edler@ce.chalmers.se
Ag Teknik
Vakant

Vilka kan vara med i SESAM?

Medlemmarna i SESAM är svenska företag, organisationer och myndigheter (förvaltningar, utbildningsinstitutioner etc) med anknytning till försvarssektorn. Medlemmarna indelas i följande kategorier
- ordinarie medlemmar
- arbetsgruppsmedlemmar
- informationsmedlemmar.

Enskild person kan endast komma ifråga som informationsmedlem.

Inträde i SESAM

För samtliga medlemskategorier gäller att inträde beslutas av Rådet.

För inträde som ordinarie- och arbetsgruppsmedlem krävs status som leverantör till FMV. Dessutom krävs en skriftlig förbindelse att uppfylla åtagande som ordinarie- och arbetsgruppsmedlem.

För inträde som informationsmedlem (erhåller endast informationsbladet) krävs status som leverantör till FMV eller status som myndighet inom totalförsvaret. Rådet kan emellertid anta annan part som informationsmedlem.

För ansökan om medlemskap i SESAM vänd er till sekretariatet.

SESAM-Sekretariatet
AerotechTelub AB
c/o Kåsjös Kontor
Ytterspåret 14
187 54 TÄBY

Ordföranden har ordet

Betydelsen av att öka kunskapen inom olika områden som berör säkerhetskritiska system får större betydelse i framtiden. Denna framtid står redan för dörren och krav på olika typer av lösningar för säkerhetskritiska system finns redan idag.

Den nya ominriktningen av svenska försvaret kräver att vi bygger system av system. Det totala systemet kräver ”nästan” full funktion trots att vissa delar fallerar.

I system kommer vi att hantera hårvarufunktioner likväl som simulerade funktioner. Dessa skall dessutom var utbytbara med varandra. Dessa krav ställer mycket stora krav på programvaran och programvarulösningar så att inte systemen ”låser” sig eller skapar säkerhetskritiska situationer.

Nya metoder för programutveckling, verifiering och test som gör att vi snabbare kommer fram till ”felfri” kod har stor prioritet. Kunskaper inom nya arkitekturer och systemutformningar är andra viktiga områden.

SESAM kan genom sitt kontaktnät inom svensk försvarsindustri, högskolor och försvarsmakten skapat förutsättningar för att information och tekniska lösningar skall kan utbytas och belysas.

De mikroprojekt som startat i år samt de som avslutades föregående år är exempel på hur SESAM kan hjälpa till med informationsspridning.

Det är min förhoppning att medlemmarna i SESAM även i år skall utföra sina uppdrag på ett entusiastiskt sätt så att hösten seminarie blir lika lyckat som under 1999.

De medlemmar och grupper som haft en sen start av sina projekt måste beakta att resultaten skall redovisas vid höstseminariet. Inget händer om inte alla i en grupp tar sin del av initiativet.

Jag vill med dessa tankar och viss påminnelse önska medlemmarna i SESAM ett fruktsamt samarbete mellan medlemmarna.

Med hälsningar
Claes Wadsten

Status FoTA P12: Överföring till industrin av programvaruteknik för säkerhetskritiska system

De projekt inom Forsknings och teknikutvecklingsprogrammet för anpassning (FoTA), som är inriktade mot programvaruteknik (P3, P4, P7, P9, P10, P11) samt H ProgSäk höll den 3/5 ett gemensamt möte. Vid detta redogjorde Bror-Arne Ersson (FMV sammanhållande för FoTA-programmet) om dess bakgrund, att stärka svensk industris förmåga att stödja ett anpassningsförsvar enl Försvarsbeslut -96.

Annika Ohlsson, EMW och P3 presenterade EMW:s första version av en utvärderingsmodell för COTS & Objektorientering för realtid. Modellen baseras på 3 komponenter (typkonstruktion, plattform, språk). 24 typkonstruktioner inom 4 områden (Process-, Minneshantering, Modularisering, Kommunikation & distribution), som implementeras i 3 språk (Ada95, C++, Java) kommer att praktiskt utvärderas av EMW, Saab, Celsius på olika plattformar (Windows 2000, Linux, VxWorks, Windows CE, EPOC). Vanlig PC med middleware (CORBA; COM osv) i olika varianter används. COTS kommer att ingå i konstruktionsfall, OS och maskinvara. En uppdatering utförs därefter i november. Modellen är tänkt att kunna användas också för framtida språk med dess standardbibliotek.

Synpunkter på modellen kan lämnas nu under våren.

Magnus Sjöland, S&T, gav en statusrapport över P4 (Patterns och komponentåteranvändning). En stor inventering över ämnesområdet planeras m h a exjobs-arbetare (klart i december). Bland företagen ingår Aerotech Telub och EMW. I huvudstudien kommer även CTH och Växjö högskola att deltaga. Tillämpningsområden kan vara förbandssimulering, WAP m m samt designpatterns inom OO.

Bengt Gustafsson, CTS, gick igenom läget inom P7 (COTS & militära ledningsoch infosystem). Den tidigare utgivna rapporten i ämnet (litteraturstudier och erfarenhetsåtervinning från bl a STRIC) har nu kompletteras med information från AUSTACCS-projektet och synpunkter från bl a P12.

Håkan Edler, CTH, redogjorde för teknikområdet P9 (Experimentell verifiering & feltolerans), vars målsättning är att analysera, modellera, implementera, verifiera system m a p pålitlighet. Felinjicering används för att artificiellt införa fel i system, för att verifiera felhanteringsmekanismer och validera tillförlitligheten. Olika strategier finns för var / när injicering skall utföras. I systemet injiceras antingen en felkälla (fault) eller ett efterföljande feltillstånd[1] (error). Inom FoTA P9 kommer tekniken att användas på datorsystemet MACS[2]. Avsikten är bl a

att mäta ett systems förmåga att kunna upptäcka och tolerera fel samt att generalisera provningsmiljön, så att den kan användas på andra typer av system och processorer. Två rapporter[3] blir klara till 00-07-01. Projektet avslutas år 2001.

Göran Anger, Industrilogik, gav en lägesrapport över P11 (Formalisering, krav & analys av krav på säkerhetskritiska system). Kommersiella villkor i referensprojektet har gjort att tilltänkt industripart varit tvungen att dra sig ur. Förutsättningarna att både objekt och resultat skall vara öppna visade sig ej uppfyllda. Diskussioner förs nu med Saab som eventuell part.

Den gemensamma delen avslutades med att datum för höstens samordnade projektdagar fastställdes: **onsdagarna** den **27/9** samt **29/11 kl 13-16**, lokal: **FMV, TrV, Filmsal C**. Förmiddagarna reserveras för enskilda FoTA-projektmöten (två extra rum har bokats, vilka kan reserveras genom kontakt med P12:s PL).

I övrigt påmindes om att FoTA-material för granskning kan sändas till P12:s Huvudleverantör samt PL. Information till/från P12 på dess hemsida (www.aerotech.ffv.se./fota_p12 via password från P12:s Huvudleverantör).

Under förmiddagen den 3/5 höll P12 sitt projektmöte. Medlemmarna redogjorde för utförda uppgifter. Nya uppslag gavs beträffande hemuppgifter (checklistor), tillämpningsområden för teknikprov (P11 på säkerhetsslingor) samt nästkommande dagordning (demo av ett webbaserat verksamhetsledningssystem resp kompletteringar av en programutvecklingsmiljö m a p systemsäkerhet). Till kommande uppgifter hör även att lämna in en specifikation över FoTA-teknik att prova i eget företag.

Inga-Lill Bratteby-Ribbing
AOL FoTA P12
ilbra@fmv.se
tel 018-12 02 63

---

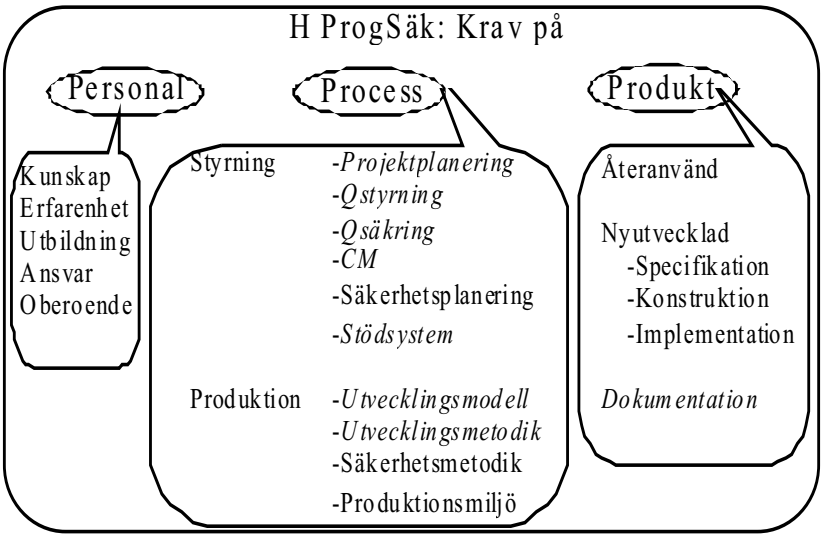
1 För test, där det fordras att exekveringen genomlöper en komplicerad kedja av händelser innan ett visst tillstånd uppnås är felinjicering av tillståndet effektivt jämfört med att trigga den utlösande händelsen / felkällan.
Tekniken är också användbar då det gäller att injicera ett visst tillstånd (ett korrekt sådant) för att kunna testa ut att mostvarande följdhändelser är de avsedda.

2 Förseningar p g a svårigheter att få igång utvecklingsmiljön SEEMACS (bl a flytt från Solaris 6 till Solaris 8).

3 Felmodeller för run-time exekutiven i Gripens systemdator. Hur man fått fart på utvecklingsmiljön SEEMACS.

Handbok för Programvara i säkerhetskritiska tillämpningar (H ProgSäk)

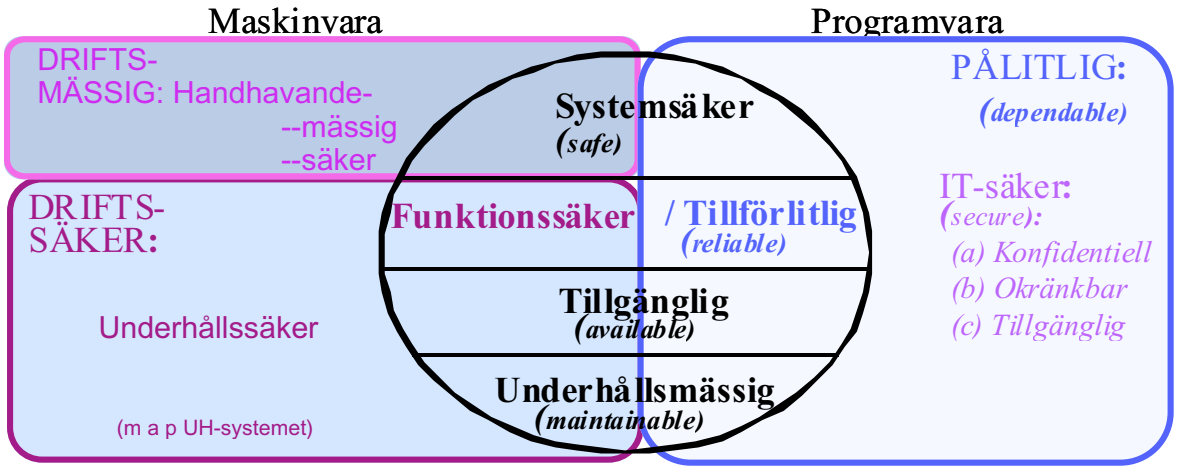
Arbetet med FMV:s handbok för anskaffning av programvara i säkerhetskritiska tillämpningar fortskrider. Ett antal FMV-projekt provar nu handboken för specning av säkerhetskrav på programvara, vid utvärdering av anbud och systemsäkerhetsplaner eller vid säkerhetsverifieringar. Några företag har inlett en inventering för att utröna i vilken utsträckning dess metoder, tekniker och utbildning kan tillgodose krav på system- och programvarusäkerhet när det gäller processer och personalkompetens. Förväntat resultat för företagens räkning är ett antal interna förbättringsförslag, en ökad systematik m a p programvusäkerhet samt ett förstärkt stöd vid arbete med säkerhetskritiska system. Erfarenheterna från dessa prov kommer att återmatas till H ProgSäk.



Intresset för handboken är stor. Över 200 ex av senaste version har på begäran distribuerats. Flera förfrågningar om engelsk version föreligger och även erbjudande att ombesörja en översättning. Denna fråga aktualiseras först i höst.

Halvdagsintroduktioner i ämnet programvarusäkerhet har under våren genomförts inom FM och FMV. Begrepp, programvarusäkerhetsteknik, skillnader program- resp maskinvara, ”fel” och felskattningar av programvara samt återanvändning belyses m h a några studiefall. En sammanfattning av introduktionern finns att tillgå hos undertecknad.

Inga-Lill Bratteby-Ribbing
AOL H ProgSäk, FMV



OBS boka in höstseminariet 18/10
Programvaran i framtidens försvar

Planeringen pågår för höstens SESAM/AiS-seminarium som vid VUs möte den 25 maj fick temat ”Programvaran i framtidens försvar”. Syftet är att få ett grepp om vad de stora förändringar som är på gång inom såväl Försvarsmakten, övriga försvarsmyndigheter som industrin, kan tänkas innebära för den framtida programvaruutvecklingen. Nedanstående beskrivning måste, på känt manér, t v betraktas som en målsättning, eftersom överenskommelser med alla nödvändiga föredragshållare ännu inte träffats.

Avsikten är att försöka få till stånd en logiskt sammanhängande förklaringskedja där man i ett första block går igenom den nya inriktning som beslutats av riksdagen och som omsätts till förbands/ system/materielmålsättningar etc av Försvarsmakten med bistånd av FOA, FMV med flera och sedan till systemarkitekturer, materielspecifikationer och nya samverkans/utvecklings/anskaffningsmodeller m m av FMV m fl. Tanken är här att försöka att också få ordentligt förklarad innebörden av alla de nya ”buzz-words” som är aktuella i sammanhanget, som RMA, DBA, DS, PE, NCW etc.

I ett andra block avses betydelsen av den förväntade tekniska utvecklingen och konsekvenserna för programvaruutvecklingen av denna att skärskådas. Där spekuleras dels kring vilka krav som kan komma att behöva ställas m h t utvecklingen inom informationskrigföring och de nya säkerhetsaspekter som den ökande förekomsten av distribuerade och mobila system, samt sammankoppling mellan ledningssystem och vapensystem medför. Vidare kring vilken betydelse utvecklingen inom system-, hårdvaru- och programvarukonstruktion – och speciellt den konvergens eller sammansmältning mellan dessa områden som pågår - kan få för den framtida programvaruhanteringen.

I det tredje blocket slutligen görs ett försök att mot den beskrivna bakgrunden reda ut vad som kan tänkas karaktärisera framtida programvarusystem och deras framtagning samt vilka krav som kan ställas på hur programvara tas fram i den nya livscykel som blir en följd av anpassningsförsvarsdoktrinen. Vidare beskrivs innebörden av en av de nya anskaffningsmodeller som kan bli aktuell, nämligen simuleringsbaserad utveckling/anskaffning, samt ges exempel på detta inom ett sedan några år pågående internationellt modellerings- och simuleringssamarbete på marinområdet inom NATO/PfP-ramen, där Sverige deltar.

I samband med den efterföljande middagen (enbart för SESAM-medlemmar och försvarsrepresentanter) är meningen att en öppen diskussion skall hållas kring vad som framförts under dagen.

Som synes ett ambitiöst upplägg, som om det kan genomföras fullt ut, skulle kunna kunna ge ett bra tillfälle för främst SESAM-medlemmarna att få en inblick i vad framtiden möjligen kan bjuda, och förhoppningsvis även att kunna påverka vad som planeras betr deras kärnkompetensområde programvaruutveckling.

Ansvarig för seminariets uppläggning och innehåll är Lena Sporre (sporre@sto.foa.se) och Ingemar Carlsson (ingemar.carlsson@mbox2.swipnet.se), *vilka mycket gärna tar emot synpunkter och önskemål på alla aspekter av seminariets utformning*, ju förr dess bättre.

Vad är ”fel” i programvara?

Det talas ofta om fel i programvaran. Vari består dessa?

Många brukar associera till rena kodningsfel, dvs:

- Krävd funktion utförs ej eller levererar inget resultat
- En oönskad händelse inträffar, t ex
    - icke begärd funktion
    - felaktigt svar / respons / beteende / styrinstruktioner
    - avsedd funktionalitet under felaktiga förutsättningar
- Felaktig ordning på begärda händelser
- Rätt händelser vid felaktig tid.

Men de flesta programutvecklare har nog lärt sig bevaka den här typen av misstag och fångar upp merparten under V&V-fasen.

Det har visat sig att rena konstruktionsfel i levererade system endast står för en bråkdel av programvarufelen[1] .

Studerar man vad som brukar betecknas som programvarufel finner man att de
antingen är ***systematiska***

och beror på brister i	a) specifikation av system, driftsförhållanden och gränssnitt
	b) konstruktion
	c) produktionsprocess (som missat identifiera felen)
	d) produktionsverktyg
eller	
på ***felaktig användning***,	
t ex	e) återanvändning under andra förutsättningar
	f) användningsområde skilt från det som specificerats för systemet
	g) handhavande i strid mot specifikationer och instruktioner.

Det som till en början uppfattats som fel i en viss programvarukomponent kan visa sig vara oväntade effekter från en ny kombination av komponenter eller beror på att situation och miljö har förändrats från vad som gällde vid konstruktionstillfället. Sådana fel är naturligtvis mycket svårare att fånga än rena konstruktionsmissar och ställer extra krav på personer inblandade i systemets realisering: ursprunglig systemansvarig, komponentutvecklare, den som står för konstruktion och återanvändning i det nya systemet –och i sista hand– slutanvändare.

Vad som är rätt eller fel beror således på sammanhanget. Ett beteende som i ett visst sammanhang betraktas som korrekt / acceptabelt kan i ett annat visa sig vara otillförlitligt (i strid mot den funktionella specifikationen) eller t o m osäkert / katastrofalt. Det är därför som återanvändning av programvara är vansklig.

---

[1] Analys av tillbud i programvarusystem har visat att mindre än 15% orsakas av fel introducerade under konstruktion och implementation, medan mer än 44% beror på brister i kravspecifikationerna (de senare är f ö de mest kostsamma att åtgärda).

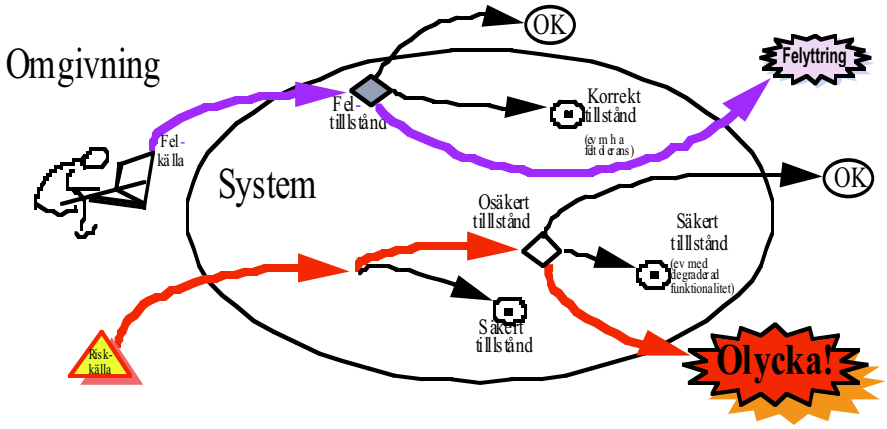
Så var fallet med Ariane 5:s haveri[1] . Tidigt misstänktes att ”felet” berodde på programvaran. Senare konstaterades, att den ursprungliga felkällan låg i gränssnittspecifikationerna (feltyp a ovan). Denna felkälla hade kunna förhindras, om någon av följande kvalitetsbrister ej förelegat:

b) Konstruktion:
- b1) Redundans i form av duplicerad kod.
- b2) Ohanterade *exceptions*
- b3) Exekvering av funktion under inaktuell mod.

c) Produktionsprocess:
- c1) Inga säkerhetsanalyser av ändringars effekt.
- c2) Otillräckliga verifieringsprocedurer.

e) Återanvändning under andra förutsättningar:
- e1) Intrimningsfunktion byggd på implicita och i detta fall inaktuella antaganden om systemet (trajektorybanor).

Programvarukvalitet är väsentlig för säkerhetskritiska system i o m att den ger möjlighet att bryta händelsekedjan från en fel- / riskkälla över till ett felaktigt eller osäkert tillstånd i systemet och vidare till en felyttring / olycka.



Ariane 5 illustrerar ett av problemen med programvara: Det går att plugga in en komponent i ett nytt system, integrera detta och köra systemet även om inte alla förutsättningar är uppfyllda. Dessa kan vara baserade på antaganden eller villkor mer eller mindre självklara för det ursprungliga systemet och därför inte explicit formulerade. I ett nytt system –t o m ett mycket närbesläktat sådant–, där några av dessa villkor ej föreligger, kan exekveringen ta andra vägar än de, som varit möjliga / aktuella för det ursprungliga systemet. Ett ur systemsynpunkt säkert beteende i avsedd omgivning och användning kan med återanvänd programvaruprodukt i nytt system eller systemversion föranleda olycka.

Flera av de säkerhetskrav man bör ställa inför återanvändning av programvara enligt H ProgSäk var ej uppfyllda för det system som ingick i Ariane 5-raketen. En anledning var troligen, att systemet sågs som en smärre vidareutveckling av Ariane 4, vilken varit i drift mer än 10 år utan missöden.

Inga-Lill Bratteby-Ribbing
FMV

---

[1] Den fullständiga 10-sidiga rapporten finns på www.esrin.esa.it/htdocs/tidc/press/Press96/ariane5rep.html.
En 1-sidig sammanfattning finns i ”Programvarusäkerhet –en introduktion”, tillgänglig hos undertecknad.

Vad menas med COTS?

Äntligen ett användbart svar

Av David Carney och Fred Long, Software Engineering Institute

Translated from the original English version and reprinted with permission from IEEE Software March/April 2000
(Tolkning av I Carlsson)

Problemet med användning av det i sig bekväma begreppet COTS, som med det nuvarande stora intresset för kommersiell ”off-the shelf” programvara fått stor spridning, är givetvis att det och dess förkortningskompanjoner, akronymer, är för ”smalt” för att kunna täcka in alla de aspekter, som måste beaktas vid användning av programvara som utvecklats utanför det egna projektet.

Vi vill försöka få fram ett botemedel mot den förvirring som råder betr COTS-akronymerna.

COTS-röran i USA:

COTS: commercial off-the shelf software
GOTS: off-the-shelf software owned by the government
MOTS: modifiable off-the-shelf software
NDI: non developmental item

Dessa begrepp räcker ej till för att tillräckligt precist karaktärisera komponenter. I stället för att fortsätta att försöka få fram en överenskommen definition, föreslår vi en mekanism som gör det möjligt att karaktärisera programvaruprodukter och komponenter på ett rimligt specifikt sätt. Den är enkel att använda och har som enda restriktion, att man har en vilja att undvika att använda enkla etiketter. Diskussionerna om COTS-intensiva system (system som innehåller kommersiella och andra utifrån kommande delar) kan med den mekanismen baseras på en otvetydig, gemensam förståelse, speciellt med referens till de nyanser som kan förekomma när systemet använder diverse komponenter från vitt varierande källor.

Varför nuvarande benämningar är otillräckliga

De hittills tillgängliga benämningarna försöker täcka in för många olika egenskaper, vilka ligger på ortogonala intresse-axlar. T ex, om man försöker skilja mellan COTS och MOTS, gör man samma logiska fel som om man försöker dela in folk i två kategorier, som kvinnor och förfäder. Självklart kan en person vara endera, ingendera eller bådadera.

Således är en viktig aspekt som vissa akronymer försöker att fånga, *källan* till en komponent. T ex kan den komma från en kommersiell leverantör eller ägas av staten; de förra kallas COTS och de senare GOTS. Att man alls skiljer mellan dessa är för att det i något avseende är viktigt, t ex för att göra budgetbeslut. Om i stället den verkliga betydelsen ligger i det förhållandet att komponenten erhölls utifrån, i stället för att utvecklas i projektet, kan den helt enkelt kallas OTS (off the shelf) eller non-developmental item (NDI) ≈ ”del som ej kräver utveckling”. Här bryr man sig inte om den inköptes eller ej, endast att det inte gick åt några egna resurser för att skapa den.

Man gör en mycket annorlunda distinktion när man skiljer mellan komponenter som antingen kan eller måste modifieras; en ännu mer komplex distinktion blir det fråga om, när man också försöker indikera vilket slags modifiering som gjörs. Tyvärr ser man ofta

denna distinktion gjord inte i samband med den föregående, utan som ersättning för den. M a o påståendet ”X är inte COTS, den är MOTS”, innebär ett fundamentalt misstag: källan och modifierbarheten är helt oberoende aspekter av X, komponenten kan mycket väl samtidigt vara COTS och MOTS.

Det finns också många olika graderingar av modifierbarhet och den enda gemensamma akronym som används härför, MOTS, kan omöjligen inrymma alla de subtila skillnaderna. En innebörd av MOTS hänför sig till den allmänna uppfattningen att det inte är bra att ändra en kommersiell produkts källkod, vilket i många fall är sant. Men en stor mängd kommersiella produkters användning är helt beroende av en omfattande anpassning. Den sortens modifiering är inte bara acceptabel, den är nödvändig, och kvalificerar sig som MOTS. Så trots att de många nyanserna av komponentmodifiering, från den nödvändiga anpassningen av produkt X till det skadliga och kortsiktiga manipulerandet med produkt Y, verkligen kan vara väsentliga att förstå, när man skall besluta om användningen av en komponent, gör den enda bokstaven M i MOTS att dessa nyanser döljs.

En enkel lösning

Vår lösning är att skilja de olika attributen längs olika intresseaxlar, så att komponenten kan placeras i en grafisk rymd. Graderingen av axlarna är huvudsakligen spekulativ, och kan lätt justeras om så behövs. Det viktiga är att attributen på olika axlar inte blir hopblandade och att varje axel har flera intressepunkter.

Käll-axeln

Käll-axeln anger komponentens ursprung. Det innebär mer än att ställa komponenter som ej fordrar utveckling i det egna projektet/organisationen, resp. komersiella komponenter mot varandra. T ex vore en extrem, en totalt icke-off-the-shelf, inom projektet eller den egna organisationen nyskriven och produktifierad komponent. Nästa på axeln skulle kunna vara en tidigare utvecklad komponent, antingen från ett befintligt system (”legacy”) eller hämtad från ett återanvändningsorgan. Det kunde vara en komponent från ett annat företag eller organisation, utvecklad på en gemensam beställning.

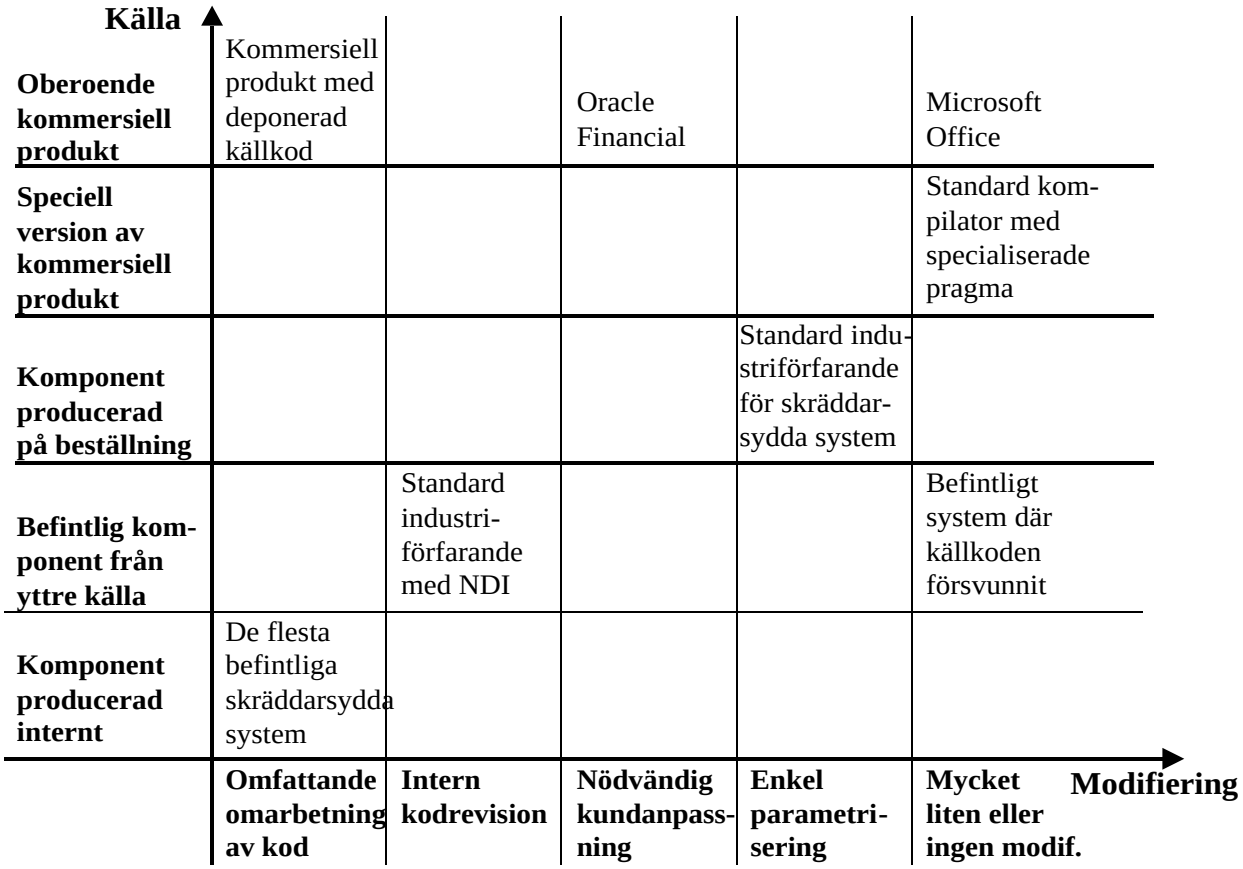
Det finns också flera grader av kommersiell. Något är odisputabelt kommersiellt om det köpts på den anonyma marknaden; säljaren tillhandahåller en produkt till världen och vet inget om den enskilde köparens krav eller begänsningar. Men en lika vanlig innebörd av kommersiell, är om någon ber ett programvaruföretag utveckla en komponent mot en överenskommen ersättning. Mellan dessa båda innebörder av kommersiell, kan ligga att en köpare kommer överens med ett företag att producera en anpassad version av en kommersiellt tillgänglig komponent (som senare kan, eller inte kan, komma att säljas fristående).

Vi inkluderar därför följande steg på källaxeln, från den obestridligt kommersiella, via olika nyanser av ”non-developmental” till rent kundanpassad kod:

- Oberoende kommersiell produkt
- Speciell version av kommersiell produkt
- Komponent producerad på beställning
- Befintlig komponent erhållen från yttre källa (t ex återanvändningsrepositorium)
- Komponent producerad internt i egna projektet/organisationen (”in-house”)

Modifierings-axeln

Den andra axeln, modifierings-axeln beskriver i vilken utsträckning användaren (i regel systemintegratören) måste ändra komponentens kod. Om man t v bortser från alla implikationer av de subtila skillnaderna, kan man konstatera att vissa komponenter praktiskt taget inte tillåter modifiering alls, annan än den som har att göra med de rudimentära detaljerna i dess installation. Vissa komponenter modifieras genom enkel parametrisering, medan en mer kompex modifiering



innefattar anpassning, vilken kan sträcka sig från trivial till omfattande. Det finns vidare modifieringar som leverantören inte har avsett skall göras eller protesterar mot, som kodrevisioner för att använda eller klara av speciella OS-protokoll eller realtidsbegränsningar. En drastisk modifiering kunde vara en större utvidgning av funktionaliteten eller fundamentala ändringar som att ta bort säkerhetsbegränsningar eller interna åtkomstkontroller. Vi tar därför med följande steg på modifieringsaxeln:

- Mycket liten eller ingen modifiering
- Enkel parametrisering
- Nödvändig kundanpassning
- Intern kodrevision för att hantera speciella plattformsegenskaper
- Omfattande funktionell omkodning och omarbetning

I figuren illustreras de båda axlarna med några exempelfall utplacerade.

De välbekanta kommersiella produkter som syns där, visar att vanliga COTS-produkter kan uppvisa mycket olika möjligheter för modifiering. Övriga produkter i figuren identifieras inte explicit, men är hämtade ur våra erfarenheter. Således är exemplet ”standard kompilator med specialiserade pragma” hämtat från två statliga projekt som båda byggde stora och komplexa real-tidssystem i Ada med användninmg av en kommersiell kompilator. Båda projekten var beroende av icke-standardegenskaper i kompilatorerna, så de hade avtalat med kompilatorleverantören om nödvändiga kundanpassningar. Leverantören har sedan inte försökt att sälja denna version på marknaden.

De två exemplen ”standard industriförfarande för skräddarsydda system” resp. ”standard industriförfarande med NDI” är kanske inte vanliga, men de avspeglar vår samlade erfarenhet från flera projekt de senaste åren och ger åtminstone en indikering om vad som kan förekomma.

Exemplet ”kommersiell produkt med deponerad kod” refererar till ett vanligt förfarande hos statliga myndigheter och inom industrin. Det implicerar att den som vill ha koden deponerad, också är villig att ta hand om underhållet av denna kod. Den tidiagre beskrivna specialiserade kompilatorn kunde bli ett sådant fall. Om kompilatorleverantören skulle upphöra att finnas till, och kompilatorn sedan skulle fordra ytterligare kundanpassning, så skulle denna kompilator flyttas hela vägen till vänster i figuren och hamna i kategorin ”omfattande omarbetning av kod” på modifieringsaxeln.

Möjliga användningar av mekanismen

Den omedelbara vinsten är ökad klarhet i beskrivningen, vilken erhålls till priset av att man måste ge upp tilliten till de bekväma akronymerna. Men det här sättet att representera kan också hjälpa till att förstå hur komponenterna används i ett system.

Exempelvis tenderar modifieringsaxeln att också skilja mellan tekniska faktorer. D v s punkterna på denna axel refererar till ändringar, triviala eller eljest, i en komponents funktionella egenskaper. Sådana distinktioner är värdefulla för att göra tidiga prediktioner om komplexiteten i att konstruera och integrera ett system.

På liknande sätt kan käll-axeln avspegla affärs- eller strategiska frågor, som tids- eller budgetuppskattingar, t ex licenskostnader. Sådan kunskaper är naturligtvis användbara för projektplanering, men kan också nyttjas för att skatta andra aspekter av ett system. T ex kunde man klassificera kravbilden m a p vilka delar av den som tillfredsställs av av olika komponenter från olika steg på källaxeln. Detta kan ge perspektiv på de långsiktiga riskfaktorerna för ett system och kan t o m indikera att man bör allokera om vissa krav. Denna klassificering kan också ge indikation för prediktering av resursbehov för underhåll av systemet på sikt, licensförnyelser och komponentuppgraderingar.

Mekanismen kan också vara användbar för att välja utvärderingsstrategier. För komponenter som ligger högre upp på källaxeln, är det nödvändigt med ”in-house black-box” testning för att komplettera komponentleverantörens utvärderingsrapport. De som ligger längre ner på skalan blir beroende av mer traditionella värderingsmetoder. För komponenter som ligger mer till vänster, behöver utvärderingen beakta hur de stämmer med kraven, för de som ligger till höger vore det mer logiskt för utvärderingen att undersöka, om man kan ändra kraven för att passa komponentens funktionalitet.

Slutligen kan representationen användas för att karaktärisera ett system medelst inneboende karaktäristika hos dess beståndsdelar. Som exempel är det vanligt att illustrera ett system som består av ett flertal komponenter, som en samling rutor anordnade och sammanlänkade enligt någon arkitekturprincip. Om man grafiskt skulle representera de olika punkterna längs både käll- och modifieringsaxlarna, skulle det vara möjligt att indikera ett systems profil med avseende på hur det använder både kommersiella och tidigare existerande komponenter. Det kanske inte blir mer än en mycket grov profil, men om den skulle visa sig vara värdefull, kan den endast erhållas om de enskilda komponenterna själva är karaktäriserade enligt ett mer precist system, än att helt enkelt kalla dem för COTS eller NDI.

Tolkens anm: David Carney deltog f ö i det svensk/amerikanska seminariet om återanvändning på DARPA 1991, vilket kanske en del minnesgoda SESAM-iter som deltog där, erinrar sig.)

Mikroprojekt 2000 i arbetsgruppen för metodik

Erfarenheterna från förra året var mycket goda och gruppen fortsätter år 2000 med samma uppläggning på arbetet. Vid avsparksmötet den 19 dec på Chalmers fick vi 13 förslag till projekt och dem samordnade vi till fyra:

- Riskhantering
- Aktuella utvecklingsmetoder
- Komponentbaserad programutveckling
- Granskning av objektorienterade modeller.

Arbetet har pågått under våren och den 12 april hade gruppen ett avstämningsmöte. Resultaten skall redovisas på SESAM-seminariet den 18 - 19 oktober

RISKHANTERING

Deltagare

Lena Sporre, sammankallande
Göran Anger
Juan Alonso

Frågeställning

Man vill undersöka hur risker hanteras i projekt och ställa det i relation till vad man gör för verifiering, validering och ackreditering. Studien gäller produktrisker, alltså risker vid användningen av en produkt, och ej projektrisker. Man kommer att redogöra för ett projekt man drivit i samarbete med kunder och man kommer att studera grundläggande metodik för att värdera risker vid användning av programvarustyrda system. Inriktningen är mer validering än verifiering.

- Hur modellerar man sådana tillämpningar?
- Vilka testfaktorer får man för att kunna prova ett system avseende dessa risker?
- Kan en provning ge svar på frågan om en produkt uppfyller kraven?

AKTUELLA UTVECKLINGS-METODER

Deltagare

Hans Ekberg, sammankallande
Lars Bergfeldt
Billy Johansson
Göran Anger
Håkan Edler

Frågeställning

Seriösa organisationer för utveckling av datorbaserade system arbetar i allt större utsträckning efter väldefinierade processer. Väsentliga inspirationskällor har varit DoD 2167A, DoD 468, ISO 12207 och SEI CMM för att bara nämna några. Andra tycker att processer enligt dessa modeller blir för tunga, ger för mycket byråkrati och allmänt hämmar kreativiteten. Ett antal alternativa sätt att utveckla system har därför föreslagits, prövats och diskuterats. Några är:

- Daily build / smoke test
- Scrum / eXtreme programming
- The Bazaar
- Adaptiva öppna system
- Genetisk programmering

Vi bör studera dessa sätt och se vad de kan tillföra vårt eget arbete. Frågor är:

- Hur seriösa är egentligen dessa ansatser?
- Hur stora är system man använder dem för?
- I vilka avseenden anses de vara bättre än traditionella processmodeller?

Arbetssätt

- Litteraturstudier och diskussioner

KOMPONENTBASERAD PROGRAMUTVECKLING - INKLUSIVE COTS-RELATERADE ASPEKTER

Deltagare

Mari Persson, sammankallande
Bertil Lundgren
Mats Forsman
Harriet Borgman

Frågeställning

- Upphandling - Vad ska man tänka på? Vilka frågor ska man ställa?
- Kravställning, t ex. omgivningskrav.
- Testning/verifiering vid integration av nya releaser, speciella integrationstester för att samordna med funktionella krav?
- Val av nivå för komponentutveckling

Arbetssätt

- Litteratursökning – sammanställning
- Nätsökning – sammanställning
- Standards (för COTS)
- Intervjuer - sammanställning erfarenheter

METODER OCH TEKNIKER FÖR GRANSKNING AV KOD OCH OBJEKTORIENTERADE MODELLER

Deltagare

Claes Norelöv, sammankallande
Hans Linderholt
Torbjörn Jungeby
Pether Camitz.

Frågeställning

Projektet tar upp metoder att granska modeller, ej dokument. Frågor är hur man hanterar kvalitetsaspekter, vilka lämpliga tekniker man kan använda och hur processer och metoder att hantera granskningsresultat kan se ut.

- Vilka problem/brister finns idag genom att man granskar dokument? Dessa är ju enbart en vy av en OO-orienterad modell.
- Hur skall man rent arbetsmässigt gå tillväga när man granskar modeller? Skiljer sig granskningsförfarandet under olika faser av utvecklingen, vilka aspekter av modellen granskas?
- Vilka krav ställs på konfigurationsstyrning? Vad utgör Configuration Items i modellen?

Arbetssätt

- Samla in erfarenheter genom intervjuer i de egna företagen
- Informationssökning

Håkan Edler, CTH
Ordföranden i Ag Metodik

SESAM-möten år 2000

25 aug	VU
5 okt	VU
18-19 okt	Höstseminariet
20 okt	Rådsmöte
8 dec	VU

En varm och skön sommar önskas er alla

Du besöker väl vår websajt?
http://sesam.tranet.fmv.se

Bidrag till Rendezvous och SESAM hemsida efterlyses

Varken Rendezvous eller hemsidan blir bättre än vad medlemmarna gör den till. Håll ögon och öron öppna för intressanta händelser i eller nära SESAM-världen och skicka in en notis till sekretariatet, alkas@tranet.fmv.se. När det är aktuellt att kunna publicera rapporter eller beskrivningar från projekt som Ni är inblandade i, glöm då inte våra egna medier.

SESAM-Sekretariatet: AerotechTelub AB
c/o Kåsjös Kontor
Ytterspåret 14
187 54 TÄBY

Telefon: 08-510 51866
Telefax: 08-510 51932
GSM: 070-716 9702
E-post: alkas@tranet.fmv.se