NYHETSBLAD FÖR SESAM
Försvarssektorns Adaintressenters Användargrupp för Software Engineering

RENDEZVOUS

Nr 4 december 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 två arbetsgrupper:

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

Ag Teknik
Lars Asplund, Uppsala Universitet
asplund@docs.uu.se

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

Vi kan nu se tillbaka på ännu ett verksamhetsår inom SESAM. Det har varit ett givande år även om vi inte på alla punkter uppnått de intentioner som vi hade innan året började.

Det är inom Ag TEKNIK vi haft dålig verksamhet under året. Jag hoppas att detta skall kunna rättas till under nästa verksamhetsår, då vi fått en förstärkning inom gruppen med Lars Asplund, Uppsala universitet. Får vi samma aktivitet inom denna grupp som vi är vana att Håkan Edler har inom Ag METODIK så ser jag med spänning på nästkommande år.

För övrigt har året innehållit många intressanta händelser/aktiviteter. Jag tänker då på beslut vi tagit för att starta en aktivare teknikbevakning med vår allas ”Mr SESAM” Ingemar Carlsson i spetsen. Även höstseminariet var lyckat där några av våra trogna medlemmar fick fram ett intressant program och i vissa fall också deltog som föreläsare. Vi har också fått ett antal stora och små företag som nya medlemmar.

För att genomföra försvarets stora ominriktning kommer ett samarbete mellan svenska företag och försvarsmakten att bli allt viktigare. Vissa försvarsföretag jobbar redan idag i Integrerade Projekt Team (IPT) med försvarsmakten. I framtiden kommer fler och fler beställningar från försvarsmakten använda denna arbetsform. Det är ju så vi arbetar inom SESAM.

Vi skall vara mycket rädda om SESAM:s verksamhet. En förening där leverantörer och försvarsmakten kan arbeta tillsammans med intressanta idéer inom datorområdet (såväl HW som SW). Alla medlemmar som betalar sin medlemsavgift (eget arbete enligt avtalet) har samma inflytande i SESAM oavsett företagets storlek.

Som ordförande vill jag härmed önska alla gamla och nya medlemmar en God Jul och ett Gott Nytt År. Vi syns år 2001 med nya krafter och idéer.

Med hälsningar
Claes Wadsten
Ordf. SESAM

SESAMs höstseminarium 2000

Höstsemniariet ”Programvaran i det nya försvaret” ägde rum på Näsby Slott den 18 oktober och lockade ca 80 deltagare. Föredragsmaterialet finns att hämta ner från hemsidan (än så länge fattas ett par avsnitt).

Den efterföljande diskussionen över ämnet ”Det framtida försvarsindustriella komplexet i Sverige; kommer det att finnas, hur kan det se ut, under vilka förhållanden har det att verka?” riktade sig till SESAM-intressenter och intresserade från andra försvarsmyndigheter.

De fyra inledarna belyste frågeställningen från sina litet olika infallsvinklar.

Gunnar Lindqvist, bl a f d chef för FMVs Huvudavdelning för flygmateriel, anlade mer övergripande perspektiv på säkerhetspolitiken och materielförsörjningen. Han skrivna anförande återges längre fram.
En kommentar från Gunnar Lindqvist med, som det skulle visa sig, särskild bäring på den efterföljande diskussionen, var att en kompetent kund också kunnat ta risker med ny teknik, vilket har bidragit till de framgångsrika resultat som svensk försvarsindustri har uppnått. Han hade bl a också synpunkter på faran med att för mycket fokusera på modebegrepp som RMA och därigenom kanske förleda beslutsfattare att tro att det finns genvägar till billigare och effektivare försvar.

Lennart Jannerö, marknadschef vid Ericsson Microwave, beskrev bl a hur Ericsson-koncernen med sitt kunnande i världsklass inom bl a mikrovågsteknik, sensorer och inte minst mobilt internet, mycket konstruktivt kan hjälpa till att realisera det nätverksförsvar som nu diskuteras. För att Försvarsmakten och FMV skall kunna bygga upp ett sådant försvar fordras närhet till leverantörena och detta kan man få med svensk industri. Sådana komplexa system som det är fråga om måste tas fram i nära samverkan mellan parterna. Jannerö efterlyste en återgång till det sätt man samarbetat på tidigare.

Gert Schyborger, Vice VD i Saab AB och ordförande i affärsområdet Saab Infomatics, beskrev den breda kompetens och de omfattande resurser som den nya Saab-koncernen besitter, men konstaterade att samverkan med Ericsson är nödvändig för att få fram det nya försvarskonceptet. Han konstaterade att tyvärr har man fått dåligt med förutsättningar för framtiden från den politiska sfären, men att man inte kommer att få några bättre. Därför är det bara att sätta igång och jobba. Saab hade konstaterat att hemmamarknaden är krympande, alltså fordras konsolidering, att tendensen på marknaden är globalisering, alltså måste man agera globalt o s v och dragit slutsatserna av detta i sitt handlande. Schyborgers tes var att industrier som agerar överlever, de som reagerar går under.

Schyborger eller Jannerö motstod framgångsrikt, måste erkännas, diskussionsordförandens försök att sortera in deras företag i ”dinosaurie”-fållan.

Magnus Sjöland, grundare av Sjöland & Thyselius Datakonsulter och ”representant” för de mindre/små försvarsföretagen, uppstickarna om man så vill, konstaterade att dessa företag redan spelade en viktig, om än kanske inte så känd, roll inom försvarsmaterielförsörjningen. T ex påpekade han, är S&T underleverantör på programvaruområdet till både Ericsson och Saab och besitter en hel del för de större försvarsindustrierna strategiskt kunnande inom hightech-områden. Folk måste få ha roligt när de jobbar och det har dom på S&T; därför har de inga svårigheter att rekrytera goda krafter. Han trodde att

det var olika typer av folk som drogs till stora och små företag. Den nya generationen vill jobba på ett annorlunda sätt och det får de möjlighet till på de mindre företagen. Sjöland ansåg samverkan på försvarssystemområdet mellan företagen helt nödvändig i Sverige, med de begränsade resurser som finns. Slutligen undrade han vad som händer på FMV; många slutar.

Den efter middagen följande livliga allmänna diskussionen kom att mest kretsa kring FMVs framtida roll och betydelse och det förhållandet att många uppfattat att FMV gör stora förluster av kompetent personal. De flesta verkade överens om att det är mycket oroande att FMV förlorar så mycket kompetens, även inom direkt försvarstekniskt strategiska områden. En åsikt som framfördes vara att om Sverige skall ha en utvecklande industri, måste FMV ha kompetent personal, eljest måste man handla genom NATO, som övriga Skandinavien.

Någon noterade att det inte bara var FMV som förlorade personal, även ”dinosaurierna” gör det; de små konsultföretagen bjuder över. Betr FMV trodde någon med erfarenheter därifrån, att arbetsmetoderna var för invecklade och arbetet för abstrakt, för att FMV skulle vara en bra arbetsplats för nyutexaminerade. En FMVare påpekade att man tvingas acceptera att nyutexaminerade slutar på FMV efter ett par år, men att det ibland finns intresse av återanställning efter några år och det bör FMV ta vara på. Från industrisidan framhölls att alla företag numera måste acceptera att nyanställda inte stannar mer än ett par år och inrätta sig efter det.

Propåer om att det kanske var dags, som på 60-talet, att bygga upp ett nytt i princip leverantörsoberoende, konsultföretag som TUAB, som stöd till FMV, möttes med blandade reaktioner. Från representant för en av de stora företagen framhölls att industrin självt kunde bidra med mycket av den kompetens som FMV tappat och att systemet med integrerade produktlag (IPT) som börjat tillämpas var lovande och kunde tillgodose bägge parters behov. En annan åsikt var att FMV hellre borde ta stöd av de mer neutrala småföretagen vid systemdefinition och specificering etc, inför större upphandlingar från industrin. Det framhölls från en tidigare FMV-anställd att FMV måste koncentrera sina resurser till de egenutvecklade och speciella teknikområdena där man måste ha egen kompetens och på övriga områden, t ex telekommunikation, köpa kompetens från den kommersiella industrin. Samtidigt måste FMV förbättra och förenkla sina regelverk.

Det påpekades också att det finns behov av att utveckla bättre former för kunskapssamverkan och utvecklingsledning i de inter-organisatoriska nätverk som kommer att bli vanliga inom försvarssektorn.

Diskussionsledaren frågade slutligen varför Sverige inte kunde bli lika dominant på det internationella försvarsmaterielområdet, som Ericsson är på telekom. Detta med hänsyn till de utomordentligt internationellt framstående produkter (JAS, kustkorvetter, radarsystem, robotar etc) som man fått fram till unikt låga kostnader de senaste 10-20 åren. Gert Schyborger trodde inte att det skulle gå på totalsystem, men kanske på vissa delsystem. Ett färskt exempel var den digitala bandspelare som Saab (ESB) just sålt till spanska flygvapnets F18-flotta och som rönt stort intresse även på andra håll. Utsikterna att sälja större system på export finns främst i länder utanför Europa och USA. En positiv klimatändring som skett i försvarsmaterielpolitiken, är att företagen numera får bra stöd för exporten från försvarsdepartementet.

Ingemar Carlsson och Dag Folkesson (m h a ofullständiga minnesnoteringar)

ANFÖRANDE VID SESAM-KONFERENS 18/10 2000

Av Gunnar Lindqvist

Hur har vår försvarsindustri sett ut? Kommer det att finnas en försvarsindustri? Hur kommer den att se ut?

HUR DET HAR VARIT?

Sverige utvecklades under 1800-talet och första delen av 1900-talet till en rik industrination. Detta var möjligt tack vare goda naturtillgångar, en kunnig och idog arbetarstam samt tekniskt kunniga företagsledare. Under mitten av det förra seklet betydde också samspelet mellan ett antal kvalificerade leverantörer och kunniga, statliga kunder en stor roll. Det gällde både långsiktiga teknologiska framsteg och tillgång på riskvilligt kapital. Exempel: LME-Televerket, ASEA-Vattenfall, ASEA-SJ, BOFORS-Försvaret, SAAB-Försvaret m fl.

Efter andra världskriget valde Sverige fortsatt alliansfrihet. Detta i kontrast till övriga länder som också varit lika alliansfria och neutrala som vi, men som ändå tvingats in i kriget. Dessa länder drog den naturliga slutsatsen att det var tryggast att ingå i en allians av demokratiska stater, dvs Nato. Till följd av vår således mycket diskutabla säkerhetspolitik fortsatte vi logiskt nog att bygga en försvarsindustri av rätt stor omfattning för att vara rimligt oberoende av leveranser utifrån. Det var erfarenheterna från 1939 som låg till grund.

Sedan stormaktstiden har vi haft en tradition av inhemsk vapentillverkning. Industrialiseringen och neutralitetspolitiken resulterade i en omfattande försvarsindustri, unik för ett land av vår storlek. Vi kunde utveckla och tillverka vapen, missiler, flygplan, fartyg, stridsvagnar, lednings- och sambandssystem med ingående mekanik och elektronik. Till följd av att vi sällan hade råd med flera konkurrerande leverantörer blev en ömsesidig långsiktig planering nödvändig. Kund och leverantör blev beroende av varandra.

Materielen gavs ofta en sk svensk profil. Motivet till detta var dels att våra krav på materielen ofta inte stämde överens med stormakternas och dels att det var en fördel i telekrigföringen. Nackdelen var att materielen i vissa fall inte heller passade kraven från en del potentiella utlandskunder. På grund av en välmotiverad exportpolitik för krigsmateriel blev marknaden mycket begränsad före Warszawa-paktens upplösning.

Helt oberoende från material och materiel utifrån var vår industri aldrig. Men graden av självförsörjning var rätt stor fram till 60- och 70-talet.

Den erforderliga tekniska nivån på krigsmaterielen bestäms av stormakternas ambitioner. Vi behöver naturligtvis inte ha allt, men det vi skaffar måste ha en kvalitativ högstående nivå. Den tekniska utvecklingen har medfört att utvecklings- och seriekostnaderna har ökat kraftigt med tiden. Inom en viss given kostnadsram har alltså möjligt antal enheter minskat. Detta mer vid fall av inhemsk utveckling och tillverkning jämfört med utländska alternativ där endast en del av utvecklingskostnaderna skulle drabba oss.

Människor glömmer så lätt och vill sällan lära sig av andras erfarenhet. Satsningar på försvaret har successivt minskats från 1958 års försvarsbeslut fram till idag. Det har medfört mindre ekonomiska ramar för försvarets materiel. Dessutom ville politikerna inte ha några långsiktiga bindningar mellan försvarets myndigheter och industrin. Den senare skall stå på egna ben och ta det övergripande ekonomiska ansvaret, vilket företagsekonomiskt inte är möjligt för stora försvarsmaterielsystem med stor andel av tekniskt nytänkande. Långsiktigheten

i planeringen försvann.

Antalet möjliga enheter som går att skaffa har alltså klämts från två håll. Detta har medfört nedläggning av flera försvarsindustrier. Vi är nu för större delen av våra materielleveranser beroende av utlandet; antingen för hela system eller delsystem, apparater, komponenter, material mm.

BEHOV AV FÖRSVARSMATERIEL

Behovet av försvarsmateriel bestäms av det hot vi kan tänkas utsättas för och den prioritering av erforderliga medel för att möta detta hot i förhållande till övriga offentliga behov.

Låt oss börja med hotet. Positivt är att det kalla kriget upphört. Just nu finns inga omedelbara motsättningar i vårt närområde. Många latenta risker finns som inom några år kan blossa upp till heta konflikter. Historien visar också att när ett spänningstillstånd upphör kan strax ett nytt uppstå. Ser vi globalt är situationen allt annat än ljus. FN's befolkningsstatistik pekar på att vi omkring 2030 kommer att vara ca 10 miljarder invånare på denna planet, större delen av befolkningen lever i länder med diktatoriskt styre, etniska och religiösa konflikter tilltar, naturresurserna blir allt knappare, spridningen av ABC-vapen till icke demokratiska stater ökar. Den tusenåriga freden har alltså inte inträffat.

Positivt är utvecklingen på kommunikationsområdet, vilket kan bidra till bättre förståelse mellan olika folkgrupper samt möjlighet att snabbare sätta in preventiva åtgärder mot potentiella konflikter.

Vad för slags konflikt vi kan komma att utsättas för är mycket osäkert. Under det gångna århundradet har krig och kriser på olika nivåer förekommit, alltifrån kärnvapenkrig, krig med kemiska vapen, krig med i huvudsak insats av fjärrvapen, gerillakrig, inbördeskrig etc till ekonomiska sanktioner. Krig förs ofta under politiska restriktioner. Allt detta innebär att krig kan föras med olika grader av teknisk nivå. Vad just vi kan komma att utsättas för är osäkert. Vi måste vara beredda på det mesta.

Ett problem med försvarsmaterieltillförseln är att det säkerhetspolitiska läget kan ändra sig snabbare än den tid det tar att i ett kritiskt läge anskaffa erforderlig materiel, än mindre att bygga upp industriella resurser.

VAD FÖR SLAGS FÖRSVARSMATERIEL SKALL VI ANSKAFFA

Vad och hur mycket vi skall anskaffa är beroende på vilket hot vi kan föreställa oss, vilken teknisk utveckling vi ser framför oss, vilka ekonomiska ramar politikerna avsätter till försvaret etc.

Den tekniska utvecklingen har pågått ända sedan urminnes tider: Elden, hjulet mm. Den genom alla tider hittills mest för krigsvetenskapen revolutionerande tekniska utvecklingen är införandet av krutet. Det medförde inte bara revolution av krigföringen utan även ett helt annat mönster vad beträffar maktbalansen mellan utvecklade länder och andra mer "barbariska". Ångmaskinen, förbränningsmotorn, u-båten, flygplanet, missilerna, telefonen, radion, radarn, datorn, internet etc utgör alla stora steg.

Möjligheterna till tekniska nyheter är idag stora. Det är ekonomin inte tekniken som är begränsande, åtminstone till viss del.

Ett försvar byggs upp av informations-, kommunikations-, lednings-, stridsförbands- och underhållsresurser. Här inkluderas personal och materiel. Ingen av resurserna har större prioritet än de andra. Hela systemet måste vara i balans. Det skall vara stridsekonomiskt optimalt avvägt. Det innebär att anskaffning av materiel såsom vapen, missiler, vapenbärare, kommunikationssystem, sensorer och ledningssystem alla är lika viktiga. En ensidig satsning på en viss materieltyp är inte optimal.

Det går lätt att hitta på nya sofistikerade system, men tyvärr har de ofta medfört en större grad av sårbarhet för försvaret. Framstegen inom informationsteknologien bör mer inriktas på robusta system än ökade informations-

flöden, vilket inte alltid leder till bättre beslut.

Just nu pågår en vulgärdebatt som går ut på att det viktigaste är IT-baserade system och precisionsvapen. De är viktiga men löser inte alla problem. Auditoriet här blir kanske besviket på mig när jag framför detta påstående. Men inte desto mindre är det viktigaste att man får ett stridsekonomiskt väl avvägt försvar. Många modeord och fraser, såsom exempelvis RMA, är nu också i omlopp. Det är inte ofarligt. Olika beslutfattare kan förledas tro att det finns en genväg till ett billigare och effektivare försvar.

FRAMTIDA FÖRUTSÄTTNINGARNA FÖR VÅR FÖRSVARSINDUSTRI

Vilka yttre förutsättningar finns det för vår försvarsindustri? Den mycket oklara hotbilden har belysts förut. Industriellt sker en minskning av kapaciteten i de flesta länderna. Undantag finns. Minskningen i efterfrågan har krävt en omfattande rationalisering inom försvarsindustrin världen över. Exempelvis är antal länder med en flygindustri kapabel att utveckla och tillverka krigsflygplan mycket få och än mindre är det antal länder som självständigt utvecklar flygplan. Detsamma gäller för större delen av krigsmaterielindustrin världen över.

Vi har sett ett antal fusioner, omstruktureringar och nedläggningar. Antalet leverantörer krymper. De återstående samarbetar för att få en jämnare beläggning på sina utvecklingsresurser. Det senare är, jämte bättre utnyttjande av stödresurser, det bärande skälet för samarbete och inte alls möjligheten att för kunden att minska Life-Cycle-Cost för respektive system.

På olika sätt måste alltså svensk försvarsindustri samverka med andra länders industri för att överleva. Det svenska försvaret kan och får inte stödja den inhemska industrin i fortsättningen på det sätt som hittills skett. Samarbete medför naturligtvis risker. Det kan leda till för snäv specialisering. Uppköp utifrån kan inom kort tid innebära nedläggning. Svårigheter att bibehålla en tillräckligt bred och djup kunskap inom respektive industris område blir svårare.

De inre förutsättningar framgår av de senaste försvarsbesluten. Beskeden är tyvärr långt ifrån klara på en del punkter.

Vår alliansfria politik fastställes gälla utan vidare diskussion trots att det säkerhetspolitiska läget ändrats radikalt. Myterna om att vi klarat oss från två världskrig tack vare vår alliansfria politik framhävs åter. Det eventuella konflikthotet tonas ned. Vi förutsätter även i fortsättningen att Nato skall avvärja krigshotet mot vårt område, samt att Nato kommer till vår hjälp om det kärvar till sig. Vi skall deltaga så mycket som möjligt i Nato's PFP-övningar, vilket är bra. Vi skall ingå i EU's insatsstyrkor för krishantering. Också bra. Men vi förbinder oss inte på något sätt att solidariskt samverka med de andra länderna. Vilken risk är våra politiker beredda att ta?

Svårigheterna att i en krissituation skaffa erforderlig materiel och utbilda personal tonas också ned, trots erfarenheterna från 1939. Logiken av att ha en någotsånär självförsörjande försvarsindustri lämnas. I stället satsar man nu på modeordet "Anpassning". Det innebär att man räknar med att hinna rusta upp i tid i ett kritiskt läge. Detta trots erfarenheterna från 1939. Den gången tog det 10-15 år att rusta upp. Nu är materielen mer komplicerad, vår industri är mer beroende av leveranser utifrån samt vår kvarvarande försvarsindustri är mindre täckande.

För att råda bot på de problem som sammanhänger med dessa fakta skall i huvudsak två åtgärder vidtagas. Dels räknar man med att satsa på forskning och teknikutveckling inom vissa delar av försvarsmaterielsidan och dels genom att delta i bilaterala eller multinationella samarbetsprojekt.

Det är givetvis bra att medel kan disponeras för forskning och teknikutveckling. Men eftersom, man samtidigt inte vill betala en tillverkning blir det en halv-

hjärtad satsning. Det är lång väg mellan teknikutveckling över produktutveckling, utprovning och produktionsuppsättning till att en tillverkning med tillräcklig kvalitet kan åstadkomma leveranser. Det är fråga om flera år såvida inte produkten är mycket lik en i tillverkning varande civil produkt.

Att gå in i samarbetsprojekt med andra länder har givetvis fördelar som förut nämnts. Men för att säkra leveranser i kris- och krigstid krävs ömsesidiga garantier. Det innebär att Sverige påtar sig skyldigheten att leverera krigsmateriel till krigförande länder. Solidariteten kan ju inte bara vara ensidig till vår fördel. Går ett sådant förfarande ihop med vår alliansfria politik? Och tål frågan en inrikes debatt? Det vore mycket beklagligt om svensk industri går in i ett samarbete men sedan tvingas dra sig ur av inrikespolitiska skäl. Liknande har hänt förut.

Men lovvärda initiativ tas nu beträffande både det sk 4-nations- och 6-nationsinitiativet. Måtte det dock snart bli mer konkreta resultat och inte bara avtalstexter.

En annan svårighet som inte bara drabbar försvarsindustrin är att det blir mindre och mindre stora nationella projekt. Det blir färre riskvilliga och kunniga statliga förvaltningar. En del är privatiserade, vilket naturligtvis har sina fördelar, andra har liksom försvaret mindre medel till förfogande. Någon långsiktig planering mellan industri och förvaltningar är inte längre möjlig.

Men det finns väl inte bara negativa sidor? Naturligtvis inte. Positivt är att vår svenska försvarsindustri har betydligt större möjligheter till samarbete med övriga världen. Före 1990 var endast de neutrala länderna tillåtna och de hade sällan viljan att bibehålla eller att etablera en egen försvarsindustri.

Den av statsmakterna tillåtna exportmarknaden har avsevärt ökat vilket ger kraftigt ökade möjligheter för svensk industri.

SLUTSATSER OM SVENSK FÖRSVARSMATERIELINDUSTRIS MÖJLIGHETER

En inhemsk försvarsindustri måste som grund bl a ha en klart utsagd säkerhetspolitik. Någon diskussion om en sådan, framtida politik förs inte i tillräcklig grad eller inte alls, trots en radikalt ändrad säkerhetspolitisk situation. Orsaken är nog främst att många vill försvara en förlegad och felaktig politik men också att försvarspolitik inte ger många röster.

Sverige vill vara med i många internationella sammanhang och visa vår solidaritet med andra länder som har bekymmer, dock bara med länder på betryggande avstånd. Solidariteten mot våra grannar har sedan länge varit lika med noll. Är det detta som kallas svensk solidaritet? I långa loppet kan det vara mycket farligt. Vi måste nog visa betydligt större ansvar än bara medlarens roll om vi skall bli respekterade i fortsättningen.

Konsekvenserna för industrin på grund av det senaste försvarsbeslutet har enligt min mening inte fullständigt klart redovisats. Speciellt inte de långsiktiga effekterna. Jag tycker inte man kan säga att regeringens försvarsindustripolitik stämmer med den officiella alliansfria politiken. Hur skall industrin reagera.?

Med dessa osäkerheter som bakgrund vågar nog ingen i dag försöka skissa på en möjlig framtida svensk försvarsindustris struktur

Personligen tycker jag också att några nya projekt borde startas av Försvarsmakten inom försvarsmaterielsidan för att få bättre stadga åt planeringen.

Positivt är att svensk försvarsindustri har fått avsevärt ökade marknader och att samarbete inom detta område blir klart möjligt. Det finns dock en del fallgropar att se upp med, som jag förut nämnt.

Kanske mest positivt är att vi fortfarande har, åtminstone några, kvalificerade försvarsindustrier med mycket kunnig personal.

G.L.

Välkänd SESAM-profil får sin specialist-status officiellt bekräftad

Inga-Lill Bratteby-Ribbing utnämndes vid ett högtidligt certifieringsseminarium på FMV den 11 dec till strategisk specialist inom området Programvarusäkerhet.

Sedan 1990 tillämpas vid FMV en policy för specialistutveckling med syftet att specialisten med sitt kunnande och sin erfarenhet skall tillföra verket, andra myndigheter, och Försvarsmakten specialistkompetens inom ett för försvarsmaterielförsörjningen strategiskt område. Specialisten medverkar till att det aktuella teknikområdet utvecklas och att kunskaper förmedlas såväl nationellt som internationellt.

Inga-Lill är den första personen som certifierats som specialist inom programvaruområdet. Inga-Lill började sin programvarubana vid FOA för att senare gå över till Bofors Electronics, som det väl då hette, där hon arbetade med bl a kvalitetsfrågor och återanvändning inom de stora Ada-projekten. Hon kom till FMV och Elektroniksystemavdelningen 1995. Förutom inom programvarusäkerhet, har Inga-Lill bl a alltså varit mycket aktiv inom återanvändning och var en period ordförande i SESAM Ag Återanvändning.

De senaste åren har Inga-Lill koncentrerat sig på programvarusäkerhetsfrågorna och ansvarar bl a för framtagningen av en handbok för anskaffning av programvara i säkerhetskritiska tillämpningar (H ProgSäk). Inga-Lill är också uppföljningsansvarig för FoTA P.12, ”Överföring till industrin av Programvaruteknik för säkerhetskritiska system” från de olika FoT-projekt för anpassning som f n pågår vid industrin.

Officiant för seminariet var Per-Arne Öhman, Chef Tekniskt Kompetenscenter

Före utnämningsceremonin talade K-G Lövstrand, själv den förste som (1991) utnämnts till specialist vid FMV och numera Chief Scientist, om ”Rollen som strategisk specialist”, varav framgick att Inga-Lill kan se fram mot en intressant, men knappast arbetsfri tillvaro som specialist.

Därefter överlämnade stf GD Jan Edelswärd specialistcertifikatet till Inga-Lill.

Efter ceremonin följde ett seminarium med intressanta föredrag förutom av Inga-Lill själv om ”Programvarusäkerhet”, av Bud Lawson om ”System- och programvarusäkerhet ur ett arkitekturperspektiv” och av Erik Sandewall om ”UAV - vilka systemsäkerhetsutmaningar finns för programvaran”.

Rendezvous sällar sig till gratulanterna!

OOPSLA ’00

Lennart.Bie@emw.ericsson.se och Kent.Petersson@emw.ericsson.se
Software Laboratory, Ericsson Microwave Systems

Den stora, årliga konferensen inom objektorienterad programmering heter OOPSLA och brukar gå i oktober eller november. Årets instans av denna konferens ägde rum mellan den 13 och 19 oktober i Minneapolis, USA. OOPSLA är en förkortning av Object-Oriented Programming, Systems, Languages, and Applications vilket betyder att i princip allt som har någon som helst koppling till objektorienterad programmering har sin plats på konferensen. Detta medför att konferensen till sin utformning är bred snarare än djup. OOPSLA är en stor konferens. I år var det totalt cirka 2500 deltagare varav många från Sverige. Enligt uppgift kom de flesta deltagarna från USA och Kanada men sedan kom Sverige som tredje land.

Konferensen har fått ett upplägg som har varit ganska stabilt under de senaste åren. Man börjar med två dagar som bara innehåller tutorials, workshops etc., sedan pågår den ”stora” konferensen under tre dagar tillsammans med en utställning där diverse olika företag visar sina alster. En intressant del av utställningen är bokförlagens utställning och försäljning av böcker. Här finns möjlighet att se och bläddra i nya och gamla böcker inom hela det datalogiska området.

Vi kommer i denna lilla beskrivning av OOPSLA’00 naturligtvis inte att kunna ge en täckande beskrivning av konferensen i stort utan kommer bara att ta upp några enskilda områden som vi uppfattar som intressanta ”trender” inom objektorientering och i övrigt hänvisa till konferensproceedings och web-sidor.

Agenter

Agenter och agentbaserad programutveckling är ett ”hett” område. I många fall har ordet agenter blivit sammankopplat med andra ord såsom intelligent, evolution, reproduktion, mutation, etc. vilket har medfört att det har varit svårt att se om det varit något substantiellt bakom all ”hype”. Ett mycket intressant föredrag var Nick Jennings *Agent-Oriented Software Engineering* som var mycket jordnära och där han visade kopplingen mellan agentbaserade program och ”vanliga” löst kopplade distribuerade system samt pekade på interaktionen mellan agenter som mycket viktig. Om man skall kunna utveckla självständiga agenter måste man försöka generalisera den enskilda kommunikationen mellan speciella agenter och börja fundera på hur interaktionen skall fungera i stort.

Nick Jennings och flera andra talare på konferensen påpekade hur komplexiteten i programvarusystem ständigt ökar. Även andra krav ökar, t.ex. ständigt kortare utvecklingstider, ändrad kravbild under utvecklingsprocessen, ökade krav på dynamik och öppenhet m.m För att möta dessa ökade krav behövs ständigt en utveckling av metoder och processer. Flera talare trodde att agentteknologin kommer att få större betydelse vid utveckling av framtida distribuerade system.

Designmönster och mönsterspråk

Designmönster i olika former har varit ett tema som varit intressant under flera tidigare OOPSLA-konferenser. Intresset är fortfarande mycket stort och många tutorials handlade om designmönster. En trend verkar vara att man mer eller mindre betraktar det som standard praxis att använda designmönster i mjukvarukonstruktion och att nuvarande forskning främst handlar om hur man skall se till att olika mönster passar ihop. Intresset verkar ha gått från enskilda mönster till mönsterspråk (pattern languages) i vilka man skall kunna beskriva hela konstruktioner snarare än enskilda delar.

Utvecklingsprocesser

Ett område som tilldrog sig stort intresse och många diskussioner var programutvecklingsprocessen. En trend som kändes stark var intresset för ”lättviktsprocesser”. Bakgrunden till detta intresse är att man i IT-världen pratar väldigt mycket om saker som förändring, adaptivitet, snabba beslut osv. och att nuvarande ”tunga” processer inte hanterar dessa begrepp på ett tillfredsställande sätt. Flera talare dömde ut de tunga processer många företag använder sig av i dag (med undantag för system med hög säkerhet rymd,

medicin och liknande). Ett intressant föredrag *Adaptive Software Development* hölls av Jim Highsmith i vilket han påpekade faran är att endast använda beprövade metodiker och processer. Historien har visat att det dyker upp grupper med omstörtande (Disruptive) teknologier som radikalt förändrar utvecklingen. Utmaningen i dag är att: ”Snabbt ta fram kompletta stora system som är forskningsinriktade samtidigt som de är företagskritiska i en turbulent affärs- och teknologomgivning”. Han såg tunga processer som CMM, ISO, RUP m.fl. som farliga för företagens överlevnad. Teser som ”Gör rätt från böjan” (icke evolutionärt) sänder fel meddelande enligt Highsmith.

En av de främsta representanterna för lättviktsprocesser är XP, eXtreme Programming . Det är inte lätt att med några få ord karaktärisera XP, men några nyckelbegrepp är: programmering i par, mycket korta iterationer, lite intresse för stor planläggning, stort intresse för testning och framförallt automatiserad testning, samt omstrukturering (refactoring) som en naturlig del av konstruktionen.

Referenser

OOPSLA’00 konferensens hemsida är:
http://oopsla.acm.org/oopsla2k/.
Här finns bland annat OH- bilder från några av de inbjudna föredragshållarna.

En websida bland många andra som man kan börja på om man vill se mer om Extreme Programming är:
http://www.extremeprogramming.org/.

En startsida för att komma åt information om designmönster och mönsterspråk är:
http://hillside.net/patterns/patterns.html.

Nick Jennings hemsida når man på:
http://www.ecs.soton.ac.uk/~nrj/.

Jim Highsmith nås via
http://www.cutter.com/consortium.

Här är första delen av Krister Bergenfeldts rapport från Ada UK konferens den 27/11 om distribuerade programvarusystem. Denna del omfattar tre av de fem föredragen; de återstående två kommer i nästa Rendezvous.

RAPPORT FRÅN DISTRIBUTED SOFTWARE SYSTEMS SYMPOSIUM

Rapporten sammanfattar de föredrag som hölls på konferensen ”Distributed Software Systems Symposium”. Konferensen anordnades av Ada Language UK den 27 november 2000. Platsen var Institution of Electrical Engineers i London, England.

DISTRIBUTED SYSTEMS WITH ADA

Talare var Thomas Quinot från ENST Paris University. Han har bl.a varit med tagit fram orben AdaBroker.

DISTRIBUTION

Inriktningen på anförandet var olika sätt att implementera distribuerade programsystem. De klassiska sätten är:

a Meddelandeöverföring mha av operativssystemet vanligen implementerat med socket-kommunikation. Applikationen hanterar all upp/nedkoppling, anpassning av data till kommunikationsprotokollet mm.

b Middleware, inkapslar och gömmer hanteringen av kommunikationskanalen i standardbibliotek. Ett exempel är Corba.

c Distribuerat programspråk. Språket har inbyggda mekanismer för distribution. Exempel är Java RMI, Modula3 och Ada DSA.

Resten av föredraget inriktade sig på att jämföra Ada DSA och Corba.

CORBA VERSUS ADA DSA

Vid användning av Corba specifierar man gränsnitten mellan objekt mha av OMG's IDL. Från specifikationen genererar man kod för sin målmiljö som hanterar kommunikationen. Genereringen är helt automatiserad och standardiserad styrd av OMG's målspråkmappningar för IDL. För närvarande finns mappningar för bl.a Ada, Java, C++. Det finns också implementationer av Corba för flera olika datorplattformar tex Unix, WindowsNT och VxWorks. Kommunikationen är standardiserad, (GIOP, IIOP), vilket möjliggör användning av Corbaprodukter från flera olika leverantörer tillsammans.

Motsvarande handhavande behövs inte för Ada DSA då språket själv beskriver gränsnitten. Man slipper hantera genererad kod. Ada DSA har andra fördelar som den starka typningen, mer generell modell med bl.a delat data (Shared Passive). Nackdelen med Ada DSA är att implementationen ej är standardiserad dvs Ada DSA från olika leverantörer kan inte kommunicera med varandra.

CORBA FLEXIBILITET FÖR DSA ANVÄNDARE

Corba erbjuder flexibilitet genom att stödja olika programspråk och genom att man under exekveringen dynamiskt kan hitta objekt och dess gränsnitt. Ett pågående forskingsprogram, CIAO, på ENST försöker bringa fram dessa fördelar vid användningen av Ada DSA.

CIAO (Corba interface for Ada distributed Objects). Detta är ett verktyg för att skapa Corba-gränssnitt för tjänster skapade mha Ada DSA. CIAO möjliggör för Corba-klienter, oavsett implementationsspråk, att utnyttja tjänsterna skapade med Ada DSA.

FRAMTIDEN FÖR DISTRIBUERAD ADA

Föredragaren såg en integration av gemensamma funktionaliteter från distribuerade middlewares. Han nämde bl.a:

> reimplemention av Ada DSA över en generisk ORB ...
>
> erbjuder direkt support för GIOP protokoll,
>
> vilket skapar interoperabillitet med Corba baserade distribuerade system.

REAL-TIME CORBA TUTORIAL

Talare var Cathy Hrustich från Objective Interface Systems (OIS), USA. Bland OIS produkter kan nämnas ORBexpress som är en realtids-ORB för implementation av CORBA kommunikation i Ada och C++. OIS medverkar också i OMG vid framtagningen av standarden för Real-Time Corba.

GRUNDERNA I CORBA

Cathy beskrev grunderna i Corba. Jag kommer inte att ta upp det i denna rapport utan hänvisar istället till OMG's hemsida (www.omg.org). Men man kan nämna att Corba's styrka är att sammankoppla applikationer byggda med olika programspråk, operativssystem och hårdvaror dvs heterogena system.

REALTIDSSYSTEM

Realtid är inte detsamma som snabbhet och effektivitet. Realtid handlar om tidsbegränsningar. Att i förväg kunna säga hur lång tid en viss aktivitet kommer ta att utföra. Realtidssystem är ett system som vars korrekthet beror på uppfyllandet av tidsbegränsningarna.

I alla system har man begränsat antal resurser. I realtidssystem har man flera parallellt exekverande processer som utför olika aktiviteter. Processerna tävlar om att få del av gemensamma resurserna. Processerna har prioriteter tilldelade sig beroende på hur viktig deras aktivetet är och hur lång tid den vanligtvis tar att utföra. En högre priorite-

rad process exekveras före en lägre prioriterad process om båda är redo att exekveras. Detta hanteras av operativssystemets schedulering.

Det finns flera olika algoritmer för schedulering av processer i ett realtidssystem. Det som algoritmerna försöker göra att exekvera processerna i systemet så alla uppfyller sina tidsbegränsningar. Det finns fall då lägre prioriterade processer blockerar exekveringen av högre prioriterade processer. Ett exempel är när den lågprioriterade processen A har allokerat en resurs R. Process B som har högre prioritet avbryter process A. Process B försöker sen allokera resurs R men blir blockerad då resurs R redan är upptagen av process A. Process A måste först exekvera för att kunna släppa resurs R innan process B skall kunna exekvera och allokera resurs R. Detta kallas prioritetsinversion. Detta är inte bra eftersom den högre prioriterade processen då får svårare att uppfylla sin tidsbegränsning. Ett sätt att minska skadan prioritetsinversion är att höja process A's prioritet till process B's så kallad prioritetsärvning. Detta gör att processer med prioritet mellan A's och B's prioritet inte kan avbryta process A och tjäla ytterligare tid från process B.

Realtidssystem är inte enkelt i en dator men blir mångdubbelt svårare när systemet är distribuerat mellan flera datorer som kanske har också olika konfiguration vad gäller hårdvara och operativsystem. Processernas prioriteter ser tex olika ut beroende på olika OS.

REAL-TIME CORBA

Realtids-Corba standarden ska hjälpa systemkonstruktörer med att bygga distribuerade Realtidssystem. Detta genom att tillhandahålla mekanismer för hantera tidsbegränsningar på aktiviteter som spänner över flera datorer och erbjuda möjligheten att kontrollera orbens resurser. RT-Corba är ett tillägg till Corbastandarden. Den har samma förhållande till Corba som Corba Services (naming, event mm) har till Corba. Det betyder att orbar som följer RT-Corba även kan kommunicera med icke RT-Corba orbar. Man kan då naturligtvis inte utnyttja den nya funktionaliteten i RT-Corba.

RT-Corba tillför följande funktionalitet till utvecklarens hjälp:

Corba prioritet. Detta är en aktivitets prioritet som är samma i hela det distribuerade systemet. Corba prioriteten mappas mot prioriteterna i OS på varje enskilt dator. Det finns en standard mappning för varje plattform från orbleverantören men man kan också som användare bestämma en egen mappning. Det finns ett standardiserat gränsnitt för att hantera Corba prioriteterna i systemet. Det finns två modeller för Corba-prioriteten: Klientpropagerad, klientens prioritet förs över till servern vid anrop, och Serverdeklarerad, servern har en fast prioritet oberoende av vilken klient som anropar den.

Styrning av uppkopplingspolicy. Man kan styra hur en uppkoppling av kommunikationen mellan klient och server ska ske. Det finns flera alternativa uppkopplingspolicies: Multiplexad; flera klient-server par delar på samma uppkoppling mellan två datornoder. Privat; varje klient-server par har en egen uppkoppling. Prioritetsstyrd; Det finns flera uppkopplingar mellan datornoder, vilken som väljs beror på aktivitetens Corba-prioritet.

Pooler av lättviktsprocesser (threadpools). Man kan förallokera lättviktsprocesser, trådar, för att exekvera aktiviteter på serversidan. Man definierar statiska trådar som skapas och tillförs poolen vid start av systemet. Man kan också definiera dynamiska trådar som skapas i runtime om alla de statiska trådarna är upptagna. Man kan sätta stackstorlek och startprioritet på trådarna.

Mutexsemaforer. RT-Corba definierar ett interface till mutexsemaforer som kan användas för att skydda resurser från samtidig användning. Mutexsemaforerna ska implementera någon sorts strategi för prioritetsärvning. Det kan vara en enkel algoritm tex en lågprioriterade tråd höjs upp till samma prioritet som en hög-

prioriterad tråd som försöker ta samma mutexsemafor. RT-Corba standarden kräver att implementationen av mutexsemaforer ska vara samma som orben själv använder. På så sätt får man en enhetlig strategi för prioritetsärvning genom hela systemet.

Utbytbara transportprotokoll. De flesta icke-RT-Corba implementationer stödjer bara TCP i dagsläget. RT-Corba inför gränssnitt för val och konfiguration av tranportprotokoll. Flera transportprotokoll kan användas samtidigt. RT-Corba standardiserar dock inte gränssnittet mellan orb och transportprotokollet utan det är leverantörsberoende för närvarande.

INTEGRATING ADA INTO A DISTRIBUTED SYSTEMS ENVIRONMENT

Talare var David Humphris från Aonix Europe Ltd.

HUR DISTRIBUERA

Hur sammanfogar man Ada med applikationer på olika plattformar skrivna i olika programspråk? Det finns områden där andra programspråk är mer lämpade än Ada. Två arkitekturer som tillhandahåller distribution är Corba och Java.

Det finns några orbar för Ada, tex ORBexpress från OIS och ORBAda från TopGraph'X. Det finns också ett par freeware orbar tex AdaBroker från ENST. För närmare beskrivning av Corba hänvisas till www.omg.org.

Java har Remote Method Invocation, RMI, som stödjer distribution. Tillsammans med Java Native code Interface, JNI, kan man integrera Ada och Java i en distribuerad miljö. Java tillhandahåller flera paket med stöd för distribution. Ett par exempel är java.rmi, java.net, javax.ejb, java.sql, javax.jms, javax.naming, javax.servlet och javax.transaction. Dessutom finns det många paket för support av grafiska användargränsnitt.

ADA OCH JAVA

Hur kopplar man ihop Ada med Java? Ett lösning är att kompilera Ada-koden till Java Byte Code och exekvera den i en Java Virtual Machine, JVM. Nackdelen är att det blir en långsamt system då JVM är interpreterande.

En lösning som ger ett snabbare system är att använda Sun's Java Native Interface (JNI) C bibliotek. Det ger C-gränsnitt till Java objects metoder. Ada kan utan större problem fås att anropa C-kod. Man får då följande fördelar; säker kompilerad Adakod (man slipper "osäker" JVM), inkapslad interaktion med Java gränssnitten (ett Ada paket för varje Java klass) och kompletta bindningar mot alla Java klasser.

Aonix har tagit fram verktyg för bindningen från Ada till Java, AdaJNI, som de kallar JAB, Java Ada Binding generator. Verktyget genererar kompletta Adabindningar mot en alla Java klasser utan handkodning eller efterjusteringar av koden. Den hanterar alla Javas språkkonstruktioner och gränssnitt. Den bevarar källkodsinformation såsom namn och kommentarer. Den hanterar runtime företeelser såsom exception som propageras från Java till Ada och samtidig användning av JVM av multipla Ada tasks. Verktyget sägs också hantera händelser i Java som via "call-backs" processas i Ada procedurer. Enda nackdelen är att verktyget för närvarande bara är integrerat i Aonix utvecklingsmiljö ObjectAda.

REFERENSER

Ada Language UK Ltd, www.adauk.org.uk

ENST Paris, www.enst.fr

BAE Systems, www.baesystems.com

Objective Interface Systems, OIS, www.ois.com

Aionix Europe Ltd, www.aoinix.com

***

Konferensrapport: SIGAda 2000

Björn Källberg, SaabTech Systems

Otraditionell konferens

SIGAda, den amerikanska Adakonferensen, motsvarande Ada-Europe-konferensen, fd. Tri-Ada, hölls den 12 —16 november. Platsen var Laurel, mellan Washington och Baltimore. Organisatoriskt innehåll den en hel del nyheter. För att göra konferensen mer attraktiv, men också billigare fanns följande nyheter

- Alla talare var inbjudna talare. Vetenskapliga artiklar som granskas, gallras och sorteras av en programkommitté fanns inte.
- Proceedings från konferensen var inte tryckta, utan levererades som en Cd-skiva.
- Allt arbete gjordes av frivilliga krafter
- Konferensen hölls på ett universitet, Johns Hopkins University.

Proceedings på en Cd-skiva är praktiskt. Man slipper tynga bagage och bokhyllor med de 2-kilosluntor som proceedings inklusive kursmaterial från SigAda tidigare utgjort. Nackdelen är ju att man måste ha en bärbar PC för att kunna läsa dokumentationen under eller före föredraget. Större delen av föredragen kommer också att finnas på SIGAdas hemsida (www.acm.org/sigada)

Att alla talare var inbjudna var också positivt. Det innebar en hög kvalitet på föredragen, delvis på grund av talarnas kunskaper, men också genom att de fått mer tid än den halvtimme som annars brukar vara norm. Nackdelen med inbjudna talare är att de inte alltid har gjort föredraget i god tid innan, och tillhörande dokumentation därför kommer sent, eller inte alls. En konsekvens av detta är att alla deltagare kommer att få en ny, uppdaterad CD-skiva när alla bidrag kommit in.

En stor nackdel var dock separation mellan föreläsningslokaler och hotell. Det fanns visserligen bussar, men de gick bara några turer på morgonen och kvällen.

Nedan kommer jag att ge intryck och referat från en del av fördragen. De hölls delvis i parallella sessioner, vilket gjorde det omöjligt att närvara vid alla. En fullständig lista med titlar på alla föredrag finns nedan.

Kurser

Konferensen inleddes med ett stort antal hel- och halvdagskurser under söndag och måndag, totalt 17 kurstitlar. Genom det stora antalet kurser blev deltagarantalet på varje kurs begränsat, nästan alltid mindre än 10 personer. Detta ger naturligtvis en god möjlighet till större interaktivitet än en kurs med många deltagare, då det bara blir ett ensidigt informationsflöde från talarstolen.

Bland kurstitlarna fann jag följande mest intressanta

- ”The personal software process”
- ”Implementing design patterns in Ada95”
- ”Real time and non real time Corba”
- ”Java for Ada programmers”
- ”GtkAda, An Ada95 object oriented graphic toolkit and GUI builder”

Rena Adakurser fanns också.

Utställning

Det fanns givetvis en utställning med ca 10 utställare. Främst märktes Ada-kompilatorleverantörerna.

ACT

Ada Core Technologies, som i huvudsak utvecklar den fritt tillgängliga GNAT-kompilator fanns med. De var den kompilatorleverantör som hade flest tekniska föredrag, bl.a. om Open Source kontra Free Software. Observera att GNAT inte är Free Software, utan skyddas av en licens.

Aonix

Aonix tillverkar ju kompilatorer främst för PC och Unix-system, men också för inbyggda system. Aonixkompilatorn använder den frontända som utvecklats av Averstar under namnet AdaMagic. Den har liksom GNAT den kompileringsmodell som bygger helt på filsystemet, saknar ett stort och monolitiskt Adabibliotek, och i stället kompilerar om alla ”withade” filer.

Man satsar stort på säkerhetskritiska system, bl.a. genom stöd för den erkända Ravenscar-profilen (Genom att begränsa främst tasking, blir systemen möjliga att verifiera fullständigt, även för de högsta säkerhetsklasserna).

DDC-I

DDC:s profil är inbyggda system, ofta säkerhetskritiska, för ett flertal processorer. Deras Ada95-teknik bygger på den i Europa utvecklade ANDF-tekniken. (Achitecture Neutral Distribution Format). ANDF innebär en standardiserad mellankod, från vilken det är lätt att göra den slutliga kodgenereringen. Portering till en ny processor blir därmed lätt, och genom att formatet också används av andra kompilatorer så blir integrationen mellan olika språk god (T.ex. mellan C och Ada).

Green-Hills

Green-Hills är en relativt stor tillverkare på kompilatormarknaden. I huvudsak görs system för C och C++, men även Fortran och Cobol finns, liksom givetvis Ada. Man använder också AdaMagic-frontändan.

Genom att man använder samma bakända och debugger för alla språk får man en fin språkintegration.

Irvine Computer

Irvine Computers produkt kallas ICC-Ada. Den är specialiserad på flygande tillämpningar. Bland pågående projekt finns Joint Strike Fighter (JSF), F22, F18E, Boeing 777 och flottans TAC-3.

OCS, OC-systems

OCS specialiserar sig på kompilatorer och andra verktyg för PowerPC.

Bl.a. tillverkar man ett verktyg, Adaprobe, som gör det möjligt att till länkad kod lägga till olika s.k. Probar. Dessa kan läsa och modifiera variabler, och också skriva ut. Både fördefinierade probar och egendefinierade finns. De egendefinierade probarna skrivs i ett relativt fullständigt programmeringsspråk. Adaprobe blir ett utmärkt verktyg både för debuggning och uttestning.

Nytt var att den nu också finns för Windows NT, på Intelarchitekturer.

Rational

Rational tillverkar givetvis fortfarande kompilatorer för en mängd maskiner. Deras huvudinriktning verkar dock vara andra utvecklingsverktyg, t.ex. designverktyg och CM-hanteringsverktyg. En uppdaterad version av Rose visades.

Övriga

Övriga utställare var mest av lokalt (dvs inom USA) intresse. T.ex. fanns Boeing där. Jag försökte köpa en 777, men det var inte den divisionen. Istället ville de rekrytera adakunnig personal till sin helikopterutveckling, bl.a. den existerande V22 Osprey (mellanting mellan helikopter och flygplan, med gigantiska propellrar som kan vridas mellan vågrätt och lodrätt) och den kommande helikoptergenerationen Comanche. Även andra mindre kända företag som ville rekrytera fanns, liksom en stor bokaffär.

Ada lever och har hälsan

Ben Brosgol, SIGAdas ordförande, inledde konferensen med ett optimistiskt anförande. Han menade, att den kritiska perioden för Adas överlevnad inträffade under de första åren efter det att DoD slutat med sitt direkta stöd. Nu är denna kritiska period passerad, och Adas framtidsutsikter är goda. Ny infrastruktur i ställlet för den Dod tillhandahöll finns, t.ex. för validering och informationsspridning.

Intressant och kanske mindre känt är stora, intressanta och säkerhetskritiska tillämpningar som görs i Ada, bl.a. tunnelbanan i både London och New York och ett TV-

musystem. Det är en lång lista, för fullständighet se SIGAdas hemsida, där alla OH-bilder finns.

Ovanliga Adaapplikationer

Flera föredrag handlade om tillämpningar av Ada. Ada är ju främst använt för militära ändamål, men också stora säkerhetskritiska system, t.ex. Boeing 777, diverse flygtrafikledningssystem och järnvägssystem. Flera föredrag på konferensen handlade dock om användning av Ada inom områden som ligger utanför dessa.

Ada för halvledartillverkning

ITEC, ett Philipsföretag, tillverkar halvledarkomponenter. Man tar de färdiga chipsen, fäster dem på en stabil ram, sätter på yttre anslutningar och förbinder dem med chipset, gjuter in i plast, och testar. För 5 år sedan moderniserades tillverkningsprocessen. Tidigare utrustning använde RTL2 och Pascal. Man bestämde sig för att använda Ada95, på Windows NT-maskiner, och GNAT-kompilatorn.

Systemen är realtidssystem. I en typisk utrustning exekverar 30-50 trådar (tasks), och svarstidskraven på externa avbrott är mindre än 1 ms. Typiskt får man med Windows NT avbrottstider på 40 mikrosekunder, och alltid bättre än kraven 1 ms. Systemen är uppbyggda som en client/server-arkitektur, där servern exekverar med realtidsprioritet och kommunicerar med tillverkningsprocessen. Klienterna svarar för man-maskingränsytan, mot operatörerna. För klienterna har man använt olika Adabaserade GUI-verktyg, bl.a. CLAW och gnatcom/gwindows.

Man är mycket nöjd med valet av Ada. Stödet för Software Engineering, inbyggd taskingmodell, och utökad säkerhet genom Adas kontroller anger man som de största fördelarna. Även utvecklingsmiljön har fungerat mycket bra. Nackdelen är, att få personer kan Ada, och att man ändå måste ha viss C-kunskap.

Canal+ TV-distribution

Canal+ är den största betal-TV-koncernen i Europa. För tekniken svarar ett dotterbolag, Canal+ Technologies. De svarar både för de svarta lådor, dekodrar, som konsumenterna har, och för serversidan (sändarsidan).

För konsumentlådorna använder man numera enbart Java. Totalt har man distribuerat ca 7 miljoner sådana lådor.

På sändarsidan duger inte Java. Av tillgänglighetsskäl, tillförlitlighet och lastskäl använder man där Ada. Ursprungligen var det Ada83 på VAX, men numer har man gått över till Ada95 med Gnatkompilatorn. Det nya systemet är ett flerplattformsystem.

Gigabit IP-switch

Firman TopLayer tillverkar en switch för Internet. Man kallar det för en application switch, dvs. den arbetar inom lagren 4-7 enligt OSI-modellen. Den tillför tjänster för att kunna undvika congestion, ge hög tillgänglighet, ge säkerhet, och garantera prestanda (QoS, quality of service). Vidare finns olika övervaknings- och loggningsfunktioner. En sådan switch kan klara av 12 st 10/100-Mbitsportar och 2 st 1-Gigabitsportar, med upp till 256000 parallella flöden.

Systemet är kodat i Ada och i assembler, för de riktigt tidskritiska delarna. För Adadelarna har man använt Ravenscarprofilen, en delmängd för högtillgängliga system, som bl.a. begränsar komplexiteten av taskinstrukturerna. Processorerna är dels Motorola MPC860, och dels ett specialchip, ARC.

GVD, GNU Visual Debugger

Den nya, visuella debuggern, som distribueras i den allmänna GNU-distributionen är skriven helt i Ada. Det är alltså inte en debugger bara för Ada, utan den generella GNU-debuggern för alla stödda språk.
Den bygger på gtkAda, en bindning till verktyget gtk+. Bindningnen är en tjock bindning, som ger elegant adakod och undviker många av svårigheterna med gtk+.

Ada och Java

Ada och Java är en intressant kombination. Syntaxmässigt sett är ju skillnaderna mycket stora, med Java som har en syntax som är C++-lik. Semantiskt sett är Ada och Java mycket lika. Det är möjligt att kompi-

lera ett godtyckligt Adaprogram till Javas bytekod. Detta är inte möjligt för t.ex. C++, där Javas säkerhetsmekanismer och avsaknad av okontrollerade pekare gör det omöjligt att kompilera C++ till Java. Det finns åtminstone två olika Adakompilatorer som genererar Java bytekod, dels JGNAT och dels ObjectAda.

Flera föredrag handlade om Java.

Högpresterande Javahårdvara för inbyggda system

Ett företag, Ajile, tillverkar ett chips för inbyggda system. Chipset exekverar enbart Java bytekod, som alltså används till allt, från drivrutiner och uppåt. Det är alltså en JVM (Java Virtual Machine) i hårdvara

Prestanda på chipset är mycket bra. För normal, rak kod exekveras Javakod 3-4 gånger snabbare än en Pentium med motsvarande Mhz-tal. Det är dock speciellt rörande trådbyten (taskswitchningstider) som prestanda blir enastående. Ett 100Mhz-chips exekverade trådbytestider ca 100 ggr snabbare än en 300 Mhz Pentium. Dessutom är det strömsnålt, 1 mW per Mhz.

Det finns också möjlighet att partitionera chipset i flera JVM:er, så att man är garanterad att processerna med låg prioritet, som t.ex. används för presentation eller webläsning, inte stör realtidsprocesserna.

Chipset är inte beroende av något programspråk, utan tar ren Java bytekod från klassfiler och laddar ner dem. Därigenom är det fullt möjligt att programmera detta chips i Ada.

Ada vs Jada

Ett annat föredrag belyste fördelarna med Ada jämfört med Java. Det gäller t.ex. realtidsprimitiver, subtyper, olika integertyper, uppräkningstyper, utökad läsbarhet genom tydligare syntax. I enlighet med Adas ursprungsideer kommer fördelarna främst fram vid underhåll, när man skall läsa någon annans program eller ett eget gammalt.

Bl.a. gavs följande exempel. De är alla legal Javakod, men är det ett fel, eller ville programmeraren åstadkomma exakt det han skrev?

```
A=B---C;
if (Bool_1 = Bool_2) ...
X=y; some_func(z);
```

Ada för realtidsLinux

Detta föredrag hade jag sett fram emot. Det hölls av Ted Baker vid FSU, Florida State University. Där har man bl.a. utvecklat runtimesystemet till GNAT. Ada för realtidslinux bygger på GNAT och implementerar Ravenscarprofilen. Dessvärre har utvecklingen inte gått så långt som jag hade hoppats. Det går att få en kodkopia, men Ted rekommenderade den ännu inte för produktion. Bl.a. nämndes följande svårigheter.

- Ravenscar restriktionerna är för begränsade
- Begränsad stabilitet
- Det finns många RT linuxes. Stöder bara RT -Linux.
- RT linux V2 portering är inte klar
- **MEN**: RT linux är en bra platform för Ada.

Säkra system

Ada används ju främst för olika säkerhetskritiska system. Ett flertal föredrag belyste detta.

Georg Romanski talade om olika säkerhetsstandarder för flygprogramvara. Han poängterade att om runtimesystemet exekverar i samma adressutrymme som applikationen så måste det verifieras tillsammans med applikationen. En sådan verifikation inkluderar bl.a. en täckningsanalys. På den lägsta nivån, C, behövs ”endast” fullständig täckning på satsnivå, för nivå B måste man också säkerställa villkorstäckning. Den högsta nivån A fordrar mätning av täckning på maskinkodsnivå.

John Barnes talade om metoden och verktyget Spark. Man använder en delmängd av Ada, kombinerad med tilläggsnotation i en speciell Sparksyntax. Sparkverktyget kan då göra en statisk analys, som vida överstiger det som kan göras utan denna notation.

En praktisk användning av Spark presenterades i ett separat föredrag av Roderick Chapman (Denna presentation gavs också på Ada i Sveriges årsmöte 2000 i Uppsala)

Lars Asplund talade om formell verifiering med hjälp av verktyget Uppål (gemensamt framtaget av universiteten i Uppsala och Ålborg). Han presenterade också ett projekt med fotbollsspelande robotar som pågår i Uppsala.

Föredragen om Ravenscar, utvecklingsprocess och kvalitet hör också till denna kategori.

Reviderad Manual

Technical Corrigendum 1 är ett dokument, med förtydliganden och korrigeringar till Adamanualen. Det innebär dock inga språkändringar. Arbetet med detta dokument är nu avslutat. Formellt beräknas den bli godkänd av ISO/IEC SC22 före årets slut.

Det finns nu också en manual, där dessa ändringar är inarbetade. Den finns tillgänglig på www.ada-auth.org/~acats/arm.html. Det finns också en PDF-version, men HTML-versionen har korsreferenser, index m.m

Ada0y

Man har börjat diskutera nästa revision av Ada, som kallas Ada0y (Kan uttalas adaoj, or ada zero why). Det är inte meningen att det skall bli någon liknande aktivitet som Ada9x, utan endast mycket små ändringar. Vissa är redan implementerade, t.ex. i GNAT.

De viktigare ändringarna som börjar diskuteras är följande:

Ömsesidigt beroende mellan paketspecifikationer

Detta är något som finns t.ex. i Java. Behovet är t.ex om man vill implementera ett system för att handha anställd i ett företag.. En naturlig modularisering är att ha ett paket Employees och ett annat Departments. Problemet är då att Departments måste kunna referera till de anställda som finns där, och för en anställd vill man lämpligen kunna referera till avdelningen. Exemplet enligt detta ”pattern” dyker ofta upp: Ett annat exempel är schackbräde och schackpjäser.

Den lösning som finns i GNAT är konstruktionen ”with type”. Den möjliggör att man kan referera till typen genom en accesstyp, som t.ex. kan läggas i en record.

Multipelt arv a la Java

Java implementerar multipelt arv genom konstruktionen ”Interface”. Ett interface kan inte ärva kod, utan programmeraren måste implementera all kod som implementerar interfacet från början. Detta är ett säkert sett, som undviker de svårigheter som bl.a. C++ har och som var grunden till att direkt multipelt arv inte finns i Ada.

En implementering av ”interface” i Ada skulle vara ett mer rättframt sätt att implementera multipelt arv än de olika patterns, som nu används i Ada.

Objekt.operation-syntax

Många som använder andra objektorienterade språk finner det svårt att anpassa sig till Adas syntax från vanligaste objektorienterade syntaxen. Adas syntax börjar ju med Operationen (metoden), följt med en parameterlista, där objektet (n) finns med. Den vanligare sättet har ju objektet först, följt av operationen, och sedan eventuellt övriga parametrar

```
Carpkg.Drive(Car, Speed)      --a la Ada
Car.Drive(Speed);             --a la Java
```

Förslaget är att parallellt införa Javasyntaxen. Semantiken mellan dessa två sätt är ju exakt densamma, så länge det bara finns ett objekt, och detta är det första i parameterlistan. I sådant fall skulle man kunna använda Javasyntaxen, om man så vill.

Som syns ovan är det inte bara s.k. syntaktiskt socker, utan har ju också den reella fördelen, att man slipper upprepa paketnamnet.

Detta trevliga föredrag om framtiden, som presenterades av Tucker Taft, Ada95:s huvudarkitekt, avslutade en trevlig konferens

Tabell 1. Fullständigt konferensprogram

Whither Ada? State of the Ada Address *Ben Brosgol (Ada Core Technologies),*
A Perspective on the State of Distributed Real-Time Research and Practice: *Douglas Jensen (MITRE)*

Distributed Systems
Real-time and Embedded CORBA Technology Update for Ada *Brad Balfour (Objective Interface Systems, Inc.)*

Ada & Software Engineering
Describing Architectures *David Emery (MITRE)*
A Successful Example of a Layered-Architecture Based Embedded Development with Ada 83 for Standard-Missile Control *Kelly L. Spicer (Raytheon Missile Systems)*

Plenary Session
Ada & Embedded/Real Time Linux *Professor Ted Baker (Florida State University)*

Standardization Issues / Ada & Academia
Ada Standardization Status and Issues James *W. Moore (MITRE)*
Ada as a Foundation Programming Language: Starting Off on the Right Foot *Michael B. Feldman (George Washington University)*

Ada & Java
Why Your Next Embedded CPU Should Provide Hardware Support for Multitasking and Multiple Virtual Machines *David S. Hardin (aJile Systems, Inc.)*
Ada 95 on the JVM: Tea for Two and Two for Tea *Franco Gasperoni (ACT Europe)*

Plenary Session
The ACM Position on Licensing Software Engineers *David Emery (MITRE)*
Open Source / Free Software Issues Robert *B. K. Dewar (Ada Core Technologies)*

High Integrity Systems
High Integrity Software for High Integrity Systems *George Romanski (Verocel)*
Safety Critical Systems Based on Formal Models *Lars Asplund (Asplund Data AB)*

Ada Experience
Using Ada for Semiconductor Assembly Equipment: *Wiljan Derks (Industrial and Technological Engineering Centre)*
Using Ada in Interactive Digital Television Systems *Thierry Lelegard (Canal+ Technologies)*

Plenary Session
Air Traffic Management in the 21st Century Judith Klein (Lockheed Martin), *Jonathan Dehn (Lockheed Martin)*
The SPARK Way to Correctness is Via Abstraction *John Barnes*

High Integrity Systems
Industrial Experience with SPARK *Dr. Roderick Chapman (Praxis Critical Systems Limited)*
Building Partitioned Architectures Based on the Ravenscar Profile *Brian Dobbing (Aonix, Inc.)*

Tools
Generation of Documentation using ASIS Tools *Steven V. Hovater (Rational Software)*
Gtk Marries Ada: The GUI Technology Revolution *Arnaud Charlet (ACT Europe)*

Real-Time
Real-Time Systems Programming with GNAT and OpenRavenscar *Juan Antonio de la Puente (Technical University of Madrid)*
New Developments in Run-Time Profiles in Ada 95 *Joyce Tokar (DDC-I)*

Ada & Software Engineering
Why We Still Have Difficulty Achieving Software Quality *David A. Cook (C. S. Draper Laboratory)*
Ada Code Analysis: Technology, Experience, and Issues. *C. Daniel Cooper (Boeing)*

Real-Time
Ada Tasking: From the Ravenscar Profile to Dynamic Scheduling *Professor Alan Burns (University of York)*
Using Ada95 to Build Software for a Gigabit Layer 7 IP Networking Device: Ada's No Big Deal Anymore *Mike Kamrad (Top Layer Networks, Inc.)*

Plenary Session
Language Issues for Ada's future. *S. Tucker Taft (Averstar Inc)*

Dr Robert C. Leif i San Diego, som har sin yrkeshemvist i medicintekniken, är en av de mest övertygade och vältaliga Ada-förespråkarna och en välkänd personlighet på Ada och SE konferenser i USA. Med tillstånd av honom återger vi här hans referat, infört i Ada Letters i våras, av vad som sades vid en workshop på förra årets SIGAda om hur man skulle kunna påskynda den civila (kommersiella) användningen av Ada. På köpet får vi en jämförande beskrivning av ekonomiska och andra faktorer för olika berörda i samband med Red Hats Linux-satsning.

SIGAda 99, Workshop: How do We Expedite the Commercial Use of Ada?

Robert C. Leif, Ada_Med, a Division of Newport Instruments,
5648 Toyon Road, San Diego CA 92115
(619)582-0437, rleif@rleif.com

The focus of this Workshop, which occurred on October 20, 1999, was on extending the use of Ada into the commercial off-the-shelf (COTS) domain. The goal of this Workshop is to determine what should we do to both make Ada the dominant language for COTS and profit by doing so? The contents of Sections 2. Red Hat, Where the Money Went, a Case Summary and 3. How to Commercialize Ada and Profit have been updated.

1. Speakers:

Robert C. Leif, Vice President Ada_Med (rleif@rleif.com), "Red Hat, Where the Money Went, a Case Summary." (Please see Section 2.)

Mike Kamrad, Senior Engineer Top Layer Networks, Inc. (kamrad@TopLayer.com) "Developing Data Communication Software for a Layer 7 Switch in Ada95". An elegant combination of Ada software, specialized hardware, and knowledge of the Internet market was described. The Top Layer Net- works' Layer switch is able to determine the type of application and use that information to control the prior- ity of transmission of a data packet.

Randall L., Brukardt, Director of Tech- nical Operations R.R. Software, Inc. (Randy@rrsoftware.com), "Ada 95 and Commercial Applications: How and Why?". A major problem with the marketing of commercial off the shelf products, COTS, written in Ada is the creation of a demand for reliable software products. Un- fortunately, the customers do not yet realize that it is possible to produce products that are not bug ridden.

Michael P. Card, Lockheed Martin OR&SS (michael.p.card@lmco.com), "FIRM is Ready" - A Real Time Object Database. The FIRM database, which is written in Ada, has incredible performance compared to the standard commercial competitors. It demonstrates how the combination of a well engineered design and the use of Ada can produce a product at incredibly low cost compared with the present commercial C and C++ based products. This talk also demonstrated the extreme diffi- culty of technology transfer from the Defense Industry to the commercial sector.

Steven Blake, Principal Ada Consultant, Aonix (sblake@AONIX.com), "An ASIS based Tool to Mea- sure what is in an Ada program". Traditional scientific and engineering disciplines are significantly based on the capacity to measure items in their domains. This new Ada technology makes possible, the objective measure- ment for software. For instance it is now possible in Ada to measure the contributions of individual packages to the linked executable. This technology also has the unique commercial utility of with a small addition providing an objective measurement of the contri- butions of individual programmers or

organizations to a large project. This in turn can be employed to divide up royalties without the necessity of employing the expensive services of accountants and lawyers.

Robert C. Leif, Vice President Ada_Med (rleif@rleif.com). "How to Commercialize Ada and Profit" (Please see Section. 3).

2. Red Hat, Where the Money Went, a Case Summary

A summary of the financials of a corporation that markets software based on the GNU Public License, GPL, is a necessary preface to a discussion on "How to Commercialize Ada and Profit". Red Hat was chosen both because Linux is the commercially most significant example of a GPL product and the valuation of Red Hat (RHAT symbol on NASDAQ) is spectacular.

The following information was obtained from the (SEC) Security and Exchange Commission's Edgar web site.

From: http://www.sec.gov/Archives/edgar/data/1087423/0001047469-99-031070-index.html

- Red Hat Linux represented approximately 56% of new license shipments of Linux-based server operating systems in 1998,
- Approximately $10.8 million in revenue for the fiscal year ended February 28, 1999, primarily from the sale of Official Red Hat Linux.
- Strategic alliances or investment relationships include Compaq, Dell, IBM, Intel, Netscape, Novell, Oracle, SAP and Silicon Graphics.
- Customers who purchase our Official Red Hat Linux Version 6.0 product receive 30 days of telephone installation support or 90 days of e-mail installation support at no additional charge.
- Customers' telephone support agreements range in price from $995 for up to three incidents to $60,000 for 24-hour-a-day unlimited support for one year.
- Red Hat has registered the trademark "Red Hat" in the United States; pending in the Australian, Cana-dian and European Union trademark offices.
- Red Hat "Shadow Man" logo registered in the U.S., European Union and Australia and have registra-tions pending for it in the Canadian trademark office.

In terms of Intellectual Property, Red Hat has from a business perspective an absolutely brilliant approach. They obtain much of their software for free by acquiring software covered by the GNU Public License[1],[2] and they protect the corporations assets by employing proprietary trademarks.

2.1. Possible problems:

Red Hat Linux in compressed form consists of approximately 573 megabytes of code (approximately 500 megabytes have been developed by independent third parties). There are 645 distinct software components developed by thousands of individual programmers which Red Hat must assemble and test before a new version can be released. Approximately 10 megabytes of code is contained in the Linux kernel. Without reliable components Red Hat Linux could fail. This would result in serious damage to the company's reputation and have the potential for litigation.

Basing a business on intellectual property where one has no control over many of the developers has its hazards. The release of major product upgrades of Red Hat Linux on a timely basis depends on individ-uals who do not have a relationship or interest in Red Hat. The Linux kernel, is maintained by third parties. Linus Torvalds, the original developer of the Linux kernel, and a small group of independent engineers are primarily responsible for the development and evolution of the Linux kernel. Actually, Red Hat's financial success is the source of its greatest hazard. The monkey see monkey do rule is directly applicable to many investors including venture capitalists. The Linux creators

and maintainers are now very marketable candidates for a start up and other competitors can also market Linux. Caldera has just gone public and Corel is selling its own version of Linux, which will run Corel's software applications.

2.2. Financials:

2.2.1. Initial Public Offering: At the request of Red Hat, the underwriters had reserved up to 800,000 shares of common stock for sale at the initial public offering price through a directed share pro-gram, to directors, officers and employees of Red Hat and to open source software developers and other persons that Red Hat believes have contributed to the success of the open source software community and the growth of Red Hat. An optimistic assumption is that 25% of these 800,000 shares or 200,000 were allocated to the open source developers, and that these developers had the money to purchase these shares. It is rumored that some of the developers did not even qualify to open an account at E*Trade, the brokerage that was charged with making these initial public offering shares available.

If the underwriters sold more than 6,000,000 shares of common stock, the underwriters had the option to purchase up to an additional 900,000 shares from Red-Hat at the initial public offering price less the underwriting discount.

The number of shares of common stock after the offering is shares outstanding on July 31, 1999 did not include:

- 5,762,188 shares of common stock issuable upon the exercise of stock options outstanding on July 31, 1999 with a weighted average exercise price of $2.51 per share
- 3,197,450 shares of common stock warrants outstanding on July 31, 1999 with an exercise price of $.0001 per share.
- 8,075,910 shares reserved as of July 31, 1999 for future stock option grants and purchases under Employee Benefit Plans.

Nominally after the initial public offering, the existing stockholders owned 90.9% of the stock. directors, executive officers and their affiliates beneficially owned approximately 67.7% of common stock. How-ever, the underwriters' bonus purchase of 900,000 shares would have easily been subscribed at a large profit;

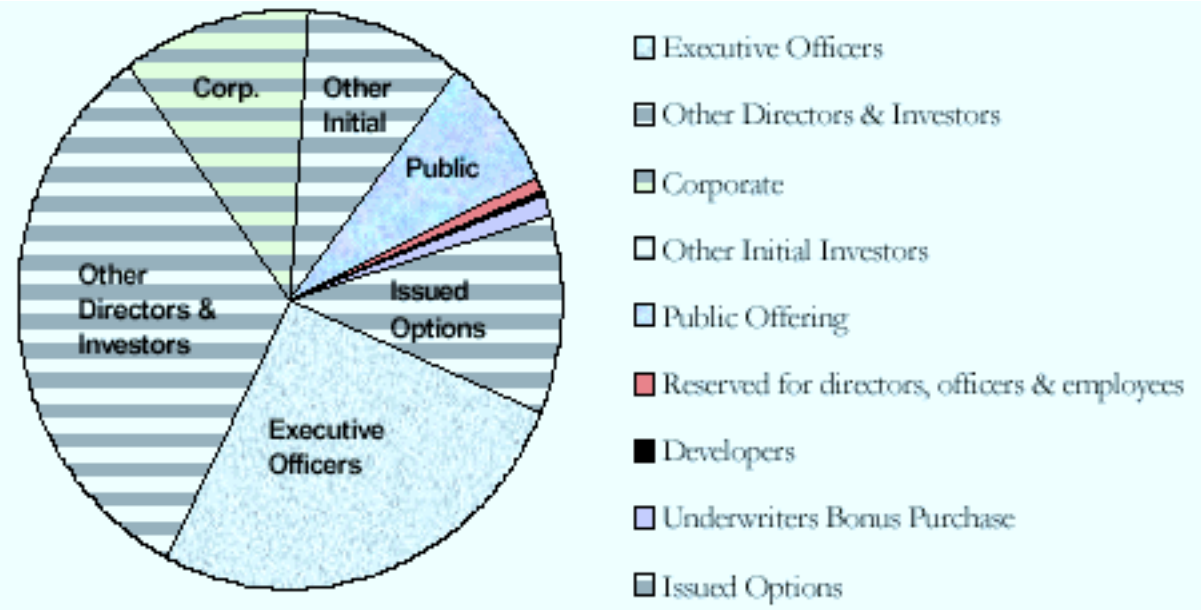

Figure 1. Pie chart showing the distribution of ownership of Red-Hat just subsequent to the com-pletion of its initial public offering. Please notice the thin, solid black sector at 2 o'clock. This is based on the estimate that the original Linux developers both received and exercised the right to purchase 200,000 shares at the open.

and the already Issued 8,959,638 very low cost options will be eventually converted into common stock. In any event even using the valuation, which corresponds to 20 March, 2000, the date of today's writing and the public offering of its competitor, Caldera, the total valuation of Red-Hat, not allowing for its acquisitions is about $9,514,000,000. Of this at best, $23,700,000 or 0.25% of the stock would have gone to the developers.

A simple calculation will indicate that developers would have been far wiser to rely on owning their own intellectual property rather than relying on corporate charity. There were 645 software components. A reasonable, perhaps somewhat conservative, assumption is to assume 25% of the value of a corporation like Red Hat to be based on its commercial software. It must be emphasized, that the relative value of delivered working software where the corporation has no up front capital investment is significantly higher than that of software internally funded by a corporation. Twenty-five percent of the total value of Red Hat is approximately 2,378,000,000. This divided by 645 equals $3,800,000 per component on 20 March, 2000 as compared with $23,700,000 to be shared by some fraction of the developers.

Red Hat went public and immediately had a value of $925 Million, which has increased to a maximum of about 14 billion and now decreased to about 9.5 billion dollars. This was all based on Free Software, Linux! In short, even Karl Marx would never have prognosticated that the workers would get at best 0.25% (virtually nothing) and the capitalists billions. Under the GNU public license[1] some of us (the dis-tributors) can be much more equal than others (the software engineers). Please see George Orwell's Animal Farm[3].

3. How to Commercialize Ada and Profit

3.1. The Past Need Not Predict the Present:

As a commercial technology, Ada has had three misfortunes that normally should have killed it. These were: 1) Ada arrived too soon. 2) Ada was a purely technology motivated undertaking, and 3) no significant attempt has been made to commercially market Ada. The lack of interest in marketing of Ada is demonstrated by the fact that none of the 20 attendees of this meeting had marketing as their primary responsibility. Ada's survival can only be attributed to its technological excellence compared to its com-petitors. The present situation is entirely different from the initial period at Ada's introduction, which should have been its window of opportunity. Later on, it was widely stated that Ada could not replace the dominant object-oriented language, C++. Sun demonstrated with Java that a market oriented company could very effectively introduce a new language to compete with and quite possibly displace C++.

3.2. The Time is Now Right for Ada

3.2.1. Then and Now: When the first validated Ada compilers were released, they were expen-sive, slow to compile and for PCs required a significant amount of expensive memory. The situation today is very different. No special hardware requirements exist for compiling Ada. A PC that can satis-factorily run Microsoft Office 2000 is sufficient for Ada development. The converse may not be true for the combination of Microsoft Office 2000 and Windows 2000 (http://www.microsoft.com/windows2000/guide/professional/sysreq/default.asp).

3.2.2. Software Products have radically changed: Initially, commercial software products fit on a few 1.2 megabyte floppy diskettes and had sufficiently few features that many of these products could be used by normal people. Today, similar products are now sufficiently complex and large that the they require both CD-ROMs and manuals that can be over one thousand pages long. The major vendors employ very large numbers of programmers and testers to produce these complex products. This type of large product development and its

associated maintenance requirements were the initial motivation for the creation of Ada.

3.2.3. Commercial Software Opportunity: The cost of popular software products has increased in significance because it has not decreased while powerful, personal computers can be purchased today for $500 to $1,000. The sum of Windows 2000 and Office is $715 (WWW.CDW.Com).

Windows 2000 and Office 2000 are not suited for home use. Besides being too expensive, they are difficult to learn. For instance, Microsoft Word is a very powerful document creation tool. It contains an enormous complement of features, which are completely beyond the capabilities of a normal user. Unfortunately, most users employ it as a glass-typewriter. They do not create specific paragraph formats for each type of document entity. The full capabilities of paragraph numbering or cross-references are seldom used.

Products like Microsoft Word and Excel are victims of their own history. Object-oriented design and programming languages should be employed to simplify the use of a program. Instead, features continue to be added. In fact, products like Microsoft Works are much better suited for today's nontechnical cus-tomer.

3.2.4. Ada Technology is Now Commercially Relevant. One of the best demonstrations of the fallacy of today's C++ based software technology is described in a book, "Project Oberon"[4]. Wirth's latest language, Oberon, like Ada is an object oriented descendant of Pascal. Both the Oberon language and Operating system are small enough to be described in one 548 page book including source and indices etc.

The combined use of Ada and Software Engineering could extend and capitalize on the discontinuity that Linux has created in the process of commercial software development. Software development in the Windows era has been based on a combination of the large-scale manufacturing and heroic programmer models. Linux has been a distributed venture based on a virtual organization. The CLAW graphics packages and screen builder are a very good example for the organization of future Ada software development. The collaborators live and work in geographically separate states, Wisconsin and California. Admittedly, the initial design and creation of many of the package specifications requires close collabora-tion. However, the creation of most of the package bodies can be reasonably independent.

Ada technology also will permit a major change in the financing of software projects. A distributed development environment eliminates the need to house an army of workers. The Redmond-Bellview Microsoft facility is an example of a very large, expensive commercial building enterprise. If this type of infrastructure cost is minimized and the employees are compensated in large part by royalties, the cost of entry will be greatly reduced. Open-source when combined with software-engineering and Ada will finally permit the replacement of large monolithic programs by modular constructs. If individuals or small groups of developers can independently create and sell these modules, then the Ada software products will achieve the necessary critical mass to be competitive with today's commercial products. The quality of these Ada products will be greatly enhanced by 1) being released as open source and 2) being subject to competition between developers for bodies and enhancements.

3.3. Ada Commercial Products, Operating System in Ada

One of the major reasons for the popularity of the C language and its descendants is that they have traditionally been used to write operating systems starting with UNIX. The ability to directly and simply link application code directly with operating system code is a significant advantage over other languages including

Ada. Ada, of course, has many advantages that in most cases more than make up for these disadvantages. However, the full utility of Ada particularly for real-time applications will be best realized by linking Ada applications to an operating system written in Ada. To that end, plans are underway to write an operating system in Ada (AdaOS-list@adapower.com).

3.3.1. A well designed Operating System should be small. An operating system can be defined as a collection of software components that can work together and be employed to create new software components. Simplicity is an excellent goal for any design process, particularly software. Human frailty often results in costly errors, requires expensive manuals, on-line help, and technical support. A good simple design can reduce development costs and result in products of sufficiently small size to permit distribution via the Internet.

The Oberon operating system developed by Wirth and his colleagues[4] has already proven that a small, simple operating system can be created at minimal cost. If the Oberon operating system had been written in Ada and the graphical user interface had been based on XML (see Section 3.3.4), a very large part of an Ada operating system would have been already completed. However, as in the previous experience with evolution of Pascal to Ada, much of the Oberon design can be reused. The operating system should maximize the present capabilities of Ada and its annexes, in particular those related to real-time operations.

3.3.2. Ada Real-Time Kernel: The first item to be developed should be an operating system independent Ada real-time kernel. This would both fulfill a very significant need for Ada real-time software development and could be marketed as a stand-alone item. The use of the Florida State Ada Linux Kernel Module[5] should eliminate language mismatch with resulting increases in efficiency and reliability. Porting this Kernel to other operating systems such as Windows CE and the Palm Operating system should significantly extend the use of Ada for commercial embedded systems.

Ada software embedded system vendors should very seriously consider combining their technical abilitywith existing companies that have experience and competence in marketing to embedded system developers. These companies interest in real-time embedded kernels and their APIs should be increased by Red-Hat's recent purchase of Cygnus Solutions[6] for shares and options valued on 20 March, 2000 at $787,000,000. Cygnus Solutions is developing EL/IX, an Embedded Linux API based on POSIX.

3.3.3. Existing Operating System Components in Ada: The presentations by both Michael P. Card from Lockheed Martin and Randall L. Brukardt from R.R. Software, Inc. were both highly relevant to the development of a modern operating system in Ada. The FIRM database was described in the Software Technology meetings[7],[8]. Ada needs to be extended with the capacity to make persistent objects of its atomic types and records. The concept of using a database to manage the file system is definitely not new. An object-relational database would be able to store reading and writing rights, authorship, etc. It would also be able to include which applications require a given file. Presently, removing applications from Windows 98 is a real-world version of a boolean process. All too often, either too little or too much is removed.

Subsequent to SIGAda99, Tom Moran posted the SmplSrvr demo at www.adapower.com. This program demonstrated that: 1) a JAVA design could be ported to Ada to produce a true executable; 2) how easy it was to use the RRsoftware CLAW components (www.rrsoftware.com), and 3) Ada real-time functionality to make a small, simple server that is able to work with either networked or local HTML pages. Particularly, SmplSrvr is able to obtain the output of the HTML FORM GET and

POST operations and translate this output into strings. After the strings were parsed, the FORM data could then be utilized as inputs to a local Ada program. Thus, Ada software that was created to serve as a thick Windows binding is capable of being ultimately used to implement an HTML-XML browser-based GUI for both embedded and distributed applications.

3.3.4 XML-XHTML GUI: The era of the present Microsoft Windows graphical user interface, GUI, software standard is ending. It will be replaced by an Internet based standard. Microsoft is quite correct that the operating system for the PC and the Web should be based on a common set of software components. This is nothing more than employing the princip-les of object oriented design. The real question is which Internet standard will succeed? The present consensus is Extensible Markup Language, XML. XML is controlled by the World Wide Web consortium, W3C (www.w3.org). This group has the great virtue of not being controlled by either Sun or Microsoft. It also has the advantage of including some very good software engineers and producing public standards. Many companies are supporting XML. These include: IBM (http://www.ibm.com/developer/xml), Sun (http://java.sun.com/xml), and Microsoft (http://msdn.microsoft.com/xml/general/whyxml.asp).

The use of XML-HTML is an effective way to break the Microsoft monopoly. XML- XHTML could serve as the basis of a new portable Graphic User Interface. The browser serves as front end of the operating system. Printers and graphics boards could be programmed to directly execute XML. Direct interpretation of XML by the printer and graphics board eliminates the main underpinning of the Microsoft Windows monopoly, its huge collection of device drivers. The only real problem is to create totally Client based programs with XML. Tom Moran's SmplSrvr has already shown that this is feasible.

3.3.5. Scripting Language: The obvious choice for the macro language is an Ada J code compiler or the equivalent. End-users of commercial software often must create macros, batch files, and formulas. These all can be represented as an Ada parameter-less procedure or classical main program. If nec-essary, an implicit instead of an explicit "with" and "use" statement can be employed for present conventual usages. Much of this functionality was present in the Alsys AdaWorld programming environ-ment.

A fourth generation programming language environment based on Ada should be created to provide a semi-automated, user-friendly approach for the creation of commercial end-user programs [9]. This should be based on copying and transforming Ada sub-programs to display the formal parameters and to employ a text entry box or list box for the specification of the actual parameters. ASIS functionality can be employed to permit type-checking at the time of filling these boxes instead of waiting for a conventional compilation.

3.3.6. License Conflict: Although much of the technology necessary to create an Ada based operating system exists, commingling of standard commercial licensed and The GNU Lesser General Public License[10] could provide a significant problem. The legal question is which would dominate for a product when parts were commingled based entirely on engineering considerations? Please note, that it is assumed that the sources for both the GNU Lesser General Public License and the standard commercial licensed software will be available to the end user. However, the commercial licensed software will require a fee for a user's license. Unfortunately, the amount of useful Ada'95 software is limited. Thus, division of it into two mutually incompatible parts could provide an insurmountable obstacle to the creation of commercial Ada products. One possible solution would be to do the licensing on a package rather than a total system basis.

The GNU Lesser General Public Lic-

ense and the Ada Developers Cooperative License [11] have a major difference in how they treat, "Commercial Executable Products". These are products sold in a for-mat that results in money being transferred. At this point, I have strongly argued [12] that the developers should receive a decent percentage of this money. The Ada community does not need to make a free contribution to the investors in a future Red Hat.

Ada technology permits the evolution of Ada commercialization beyond the current GNU model. The inability to objectively evaluate each developer's contribution to a large product results in the simplest of solutions. In the case of GNU Public License[1] software, everyone works for free. Fortunately, this no longer needs to be the case for Ada. Steve Blake's description of ASIS technology established that it is feasible to enumerate all of the components of the Ada source that are linked. The addition of one variable, vendor number, to each package should permit a reproducible, consistent means of dividing the fruits of the developers' labor. Admittedly, this will require the assignment by humans of weighting factors for each Ada construct. However, once these somewhat subjective factors are agreed upon, the relative contributions of developers can be automatically computed. This is the ultimate in recursion. We use software engineering to divide up the royalties in a reasonable manner; and this technology motivates the use of software engineering.

3.4. The grass really is not greener on the other side

The usual Ada fatal mistake is to believe that the grass really is greener on the other side. A major problem in achieving commercial success is that the Ada developers keep making bindings to the rest of the world instead of using Ada to produce an improved product or extend an existing product by employing Ada technology. The best ways to persuade an engineer to learn new technology are to demonstrate that 1) the use of this technology really reduces the effort needed to complete a project and 2) knowledge of this technology will look good on his/her resume. Ada buys a developer very little, if all it provides are thin bindings to lousy software. The Microsoft design for a windowing system is not the best solution. Each window could be a task and the screen memory could be a protected type. If the system is in sufficient part web based, use of HTML etc. for the interfaces to the operating and other systems makes sense. The Ada community certainly should borrow from existing standards like CORBA or COM and produce products that combine them with the Distribution Annex. In fact, Ada COM interfaces should evolve to work on both Windows and POSIX based operating systems. Ada needs innovative commercial product, such as the Top Layer Networks' Layer switch, described by Mike Kamrad in his talk.

4. Conclusions

1) The SIGAda '99 Commercializing Ada Workshop has demonstrated the feasibility of creating a very advanced, yet simple Ada based operating system.

2) Groups and individuals that attempt to build or upgrade products in Ada should obtain protection of their intellectual property that will permit them to share in future commercial successes.

3) The spectacular rise of JAVA has proven that it was and is possible to replace the dominant computer language.

4) The complexity, inefficiency, cost, and the lack of reliability of today's commercial software provides a great business opportunity.

5) ASIS provides Ada with a unique commercial advantage. Royalties can be equitably divided between the developers.

6) If Ada were launched as a brand-new programming language which facilitates and expedites the creation

of reliable, efficient, portable software that worked on clients, servers, and the Web; it would be a hot new issue on today's (March, 2000) stock exchange.

7) Much talk about virtue (software engineering) has had little if any effect in increasing the use of Ada.

8) Money does talk! Since virtue failed, let's try greed.

5. Acknowledgements

I wish to thank the speakers for their excellent presentations, the Workshop attendees for their questions and suggestions. Stephanie H. Leif and Tom Moran have made suggestions which have significantly improved this paper. This paper and the Workshop would not have been possible without the efforts of the Workshop organizers and the organizers of SIGAda '99. This project was supported primarily by Newport Instruments internal development funds and in part by Small Business Technology Transfer grant number 1R41CA73089 from the National Cancer Institute. The NCI support was used for the test-ing of the concept of using web pages as the GUI for controlling an instrument. The opinions stated are solely those of the author.

6. References and further reading

1. GNU General Public License, Version 2, June 1991, Copyright (C) 1989, 1991 Free Software Foundation, Inc. 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA (http://www.gnu.org/copyleft/gpl.html) (1991).
2. Richard Stallman, "The GNU Manifesto", Copyright (C) 1985, 1993 Free Software Foundation, Inc. (http://www.gnu.org/gnu/manifesto.html).
3. George Orwell, "Animal Farm: A Fairy Story", 50th anniversary edition New American Library; ISBN:0451526341 (1996).
4. N. Wirth & J. Gutknecht, "Project Oberon, The Design of an Operating System and Compiler," Addison-Wesley, ACM Press, ISBN 0-201-54428-8, 1992 (http://www.oberon.ethz.ch/).
5. H. Shen and T. P. Baker, "A Linux Kernel Module Implementation of Restricted Ada Tasking", Ada letters XIX, No 2 pp. 96..103 (1999).
6. Red-Hat Form: 10-Q Filing Date: 1/14/2000, (http://www.corporate-ir.net/ireye/ir_isite.zhtml?ticker=RHAT&script=800&layout=10)
7. M. P. Card, "FIRM: An Ada Binding to ODMG-93 1.2", Track: 6 (Object-Oriented Development), Soft-ware Technology Conference 96 (1996).
8. M. P. Card and P. G. Springer, "The Best of Both Worlds: Distributing Objects with CORBA and an ODBMS," Slides from the Software Technology Conference 98 (1998).
9. R. C. Leif, "Commercializing Ada". ACM Ada Letters 16 pp. 44-45 (1996).
10. GNU LESSER GENERAL PUBLIC LICENSE, Version 2.1, February 1999, Copyright (C) 1991, 1999 Free Software Foundation, Inc. 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA (1999).
11. R. C. Leif, "Ada Developers Cooperative License (Draft) Version 0.3", Ada letters XIX, No 1 pp. 97..107 (1999) (http://www.acm.org/sigada/wg/cauwg/cauwg.html).
12. R. C. Leif, "SIGAda '98, Workshop: How do We Expedite the Commercial Use of Ada?." Ada letters XIX, No 1 pp. 28-39 (1999) (http://www.acm.org/sigada/wg/cauwg/cauwg.html)

Formella metoder, mer eller mindre bra

Referat från ”Summer school” i Skövde, presentation av professor Michael G Hinchley, med lite egna funderingar i ämnet av Ingmar Ögren (iog@toolforsystems.com).

Alla programvarukonferenser med självaktning brukar ha en föredragshållare som talar om hur bra och viktigt det är med formella metoder för programvaruarbete. Så också sommarskolan i Skövde där den här rollen spelades av professorn Michael G. Hinchley från universitetet i Queensland i Australien. För mig blev detta extra intressant eftersom han tog upp ett projekt, som jag tyckte mig ha hört om vid en annan konferens och då med helt annorlunda slutsatser.

Hinchleys inledning

Professor Hinchley började med att dra ett antal tänkvärda citat från kända storheter i programvarubranschen:

- Det är ett allvarligt misstag att tro att programmerarens produkt är programmen han skriver; programmeraren måste producera pålitliga lösningar och han måste producera och presentera dem med övertygande argument. (Dijkstra)
- Jag tror att den svåra delen av att bygga programvara är att specificera, konstruera och prova en konceptuell konstruktion, inte arbetet att representera den och att visa att representationen stämmer med konstruktionen. (Frederick P. Brooks Jr.)
- Framsteg inom programmering är möjliga enbart om vi är villiga att temporärt helt negligera sambandet mellan våra program (i textuell form) och deras implementering i exekverbar kod. Helt enkelt: för effektiv förståelse av våra program måste vi bortse från datorernas existens. (Dijkstra)
- Inuti varje stort program finns ett litet program, som försöker komma ut. (Hoare)

Nästa steg efter den tankeväckande inledningen var att definiera formella metoder:

- En formell metod är en uppsättning verktyg, notationer och tekniker för att utveckla system, som är färdiga i tid, inom budget, som tillfredsställer användarens krav och som är bevisbart korrekta. Dessutom:
  - o en specifikation, som skrivs med formell semantik
  - o en härledningsapparat (med logik eller bevissystem)
  - o tillhörande verktyg ”bevisassistenter”.

Här börjar det bli intressant, för den som i likhet med referenten pysslat ett tag med systemarbete: Har vi verkligen hållit på och bekymrat oss i så många år över svårigheten att få budgetar och tidsplaner att stämma med kundkrav när lösningen hela tiden fanns till hands i form av formella metoder?

Formalisering av flygbiljettbokning

Som exempel på hur man kan arbeta formellt, drogs ett exempel med flygbiljettbokning där ett antal begrepp som ”city” och ”passenger” etablerades och olika samband skrevs upp i formell matematisk notation.

Själv hade jag lite svårt för detta eftersom jag hängde upp mig redan på grundbegreppen:

- Det som omtalades som ”city” tyckte jag uppenbart borde kallas ”airport”, som ju både kan finnas flera i en stad och delas mellan flera städer.
- Det som omtalas som ”passagerare” var uppenbart ”biljettkund” som kanske är en juridisk person, som inte ens är kompetent att vara passagerare på ett flygplan.

Inte blev det bättre när jag tog upp dessa funderingar med professor Hinchley: Eftersom flygbolagen talar om ”city” och ”passenger”, så måste vi acceptera konventionen och inte hänga upp oss på att logiken blir oklar!

Slutsatsen för min del blev här snarast att jag började ana varför det är svårt att bygga system för flygbiljettbokning.

Kritik av Railtrack-projektet

Härnäst började professor Hinchley tala om järnvägssignalering i England. Det handlade tydligen om ett projekt där man inte insett nyttan av formella metoder eftersom det mesta uppenbart gjorts fel. Allra värst vara att man inte hade utgått från en ordentlig systemlösning, utan tagit en lösning från en tillverkare av leksakståg - som naturligtvis inte fungerade eftersom leksakstågen bara gick i inomhusmiljö!

Presentationen var roande men gav misstanken att professor Hinchley, som förespråkare av formella metoder, hade speciell anledning att tycka illa om brittiska banverket, Railtrack.

En alternativ syn på samma projekt

Detta kom mig att tänka på en kravhanteringskonferens i London 1998. En av föredragshållarna var Eddie Edwards, projektledare för Railtracks ”Network Management Centres”.

Han berättade om Railtracks erfarenheter av att försöka arbeta med en formell specifikation:

1996 uppdrog Railtrack åt ett konsultföretag att genomföra en kravfångning för projektet (Nytt nationellt signalsystem för järnvägen i Storbrittanien) och att ta fram en användarkravspecifikation. De vanliga stegen gicks igenom: intervjuer med användare, arbetsgruppsmöten, återmatning, granskningar/genomgångar och slutligen användarnas godkännande av specifikationen. Konsulten betalades och vi planerade fasen där vi skulle översätta användarkraven till en specifikation med systemkrav, som vi kunde använda för att begära in anbud från industrin.

Det var då vi upptäckte att ingen inom företaget egentligen förstod användarkraven så bra att de kunde användas för att definiera en uppsättning systemkrav. Konsulten hade samlat på sig en mängd information, mycket nödvändigtvis ”anekdotiskt”, analyserat informationen, delat upp det hela i ett antal akademiskt rena och oigenkännliga processer, samt skrivit ner det hela i en notation som var komplett obegriplig för dem som var ensamma om kompetensen att validera det hela. Sex månader av projekttiden hade gått åt till denna verksamhet vilket lade en del begränsningar på de efterföljande faserna.

Vad kunden behöver från en konsult, som skall samla in krav, speciellt från slutanvändare är en demonstation av hur enkelt det kan vara, inte hur märkvärdigt och invecklat det är. Utan den här målsättningen, kommer konsulterna att fortsätta att inte riktigt möta sina kunders behov och att gradvis reducera sin kundbas. Samtidigt kommer kunderna att utveckla egen expertis på verksamhetsprocesser och systemintegration.

Avslutande funderingar

Oberoende av om Hinchley och Edwards verkligen talade om samma projekt eller inte så belyser de bägge redovisningarna en del av motsättningarna runt formella metoder:

Formalisten: Det matematiska uttryckssättet är enklast och elegantast samt ger den säkraste logiken med möjlighet att bevisa påståenden.
Praktikern: Det jag inte kan begripa kan jag inte heller bedöma mot verklighetens krav - värdelöst!

Formalisten: Den matematiska notationen är enkel, det behövs bara en kort kurs för att sätta sig in i och förstå skönheten med min notation.
Praktikern: Jag har jobb över öronen. Varför skall jag gå en ”kort kurs” för att formalisterna envisas med att rita matematiska krumelurer i stället för att skriva ut i klartext vad de menar?

Formalisten: Formella samband och bevis är viktiga föra att projektet skall bli klart i tid, inom budget och i enlighet med slutanvändarens krav.
Praktikern: Vi måste nog börja med att klarlägga våra grundbegrepp och gå vidare stegvis från vad vi vet nu.

Formalisten: Ett problem i det här projektet är att användarna är så okunniga och lata att de inte vill besvära sig med att granska och läsa en ordentlig specifikation.

Praktikern: Ett problem med formalisterna är att de har svårt att begripa vad systemet egentligen behöver göra och att de envisas att uttrycka sig i en obegriplig syntax.

Om man antar att Hinchley och Edwards faktiskt talar om samma projekt så är det här synnerligen intressant och visar på ett av våra största problem när det går snett i komplexa systemarbeten och vad som tenderar att hända: Två parter i systemarbetet (här kunden och konsulten) får svårigheter att tala med varandra och börjar i stället tala om varandra. Visserligen kan detta vara roande för oss som lyssnar, men det hjälper knappast projektet.

Slutsatsen är att visst är det bra med mera formalism än vad som erbjuds av naturligt språk, men syftet måste hela tiden vara att etablera och upprätthålla kontakt inom projektet så att folk talar med varandra och inte om varandra!

Ingmar Ögren

Teknikbevakningsprojektet

Vid rådsmötet i samband med höstsemniariet, godkändes det förslag till teknikbevakningsprojekt, som vi redogjorde för i förra numret av Rendezvous. Beslutet innebär en planeringsram på 75.000 kr, som alltså skall användas till att få fram refererat av viktigare konferenser och artiklar i de större programvaruorienterade tidskrifterna, redogörelser för annat teknikintressant etc. Publiceringen avses ske i Rendezvous eller på hemsidan.

Ett första resultat av rådets beslut kan avläsas i form av konferensreferaten i detta nummer.

Efterlysning av konferensreferat

Teknikbevakningsprojekt är intresserat av att få fram referat bl a från nedan uppräknade konferenser under första halvåret 2001 (urval ur hemsidans konferenslista).

För ca 2-sidiga utnyttjade referat utgår en ersättning om 5000 kr. I första hand hoppas vi på referat från personer som får sina rese- och konferenskostnader betalada av arbetsgivaren (eller på annat sätt). Projektet har inte möjlighet annat än i undantagsfall att bidra till att täcka del av sådana kostnader (men det kan alltså förekomma).

2001-01-17—19 First OMG Workshop On Embedded Object-based Systems, San Jose, CA, USA

2001-03-05—06 SREIS'01: Symposium on Requirements Engineering for Information Security, West Lafayette, IN, USA

2001-03-26—29 OMG DOCsec2001, The Fifth Workshop on Distributed Objects and Components Security, Annapolis, MD, USA

2001-04-29—05-03 STC 2001: 12 th Software Technology Conference Salt Lake City, UT, USA

2001-05-02—04 ISORC 2001, The 4th IEEE International Symposium On Object-oriented Real-time Distributed Computing,Magdeburg

2001-05-12—19 ICSE 2001: IEEE: 23rd International Conference on Software Engineering Toronto, Ontario, Kanada

2001-05-14—18 Ada-Europe'2001: 6th International Conference on Reliable Software Technologies, Leuven nära Bryssel

2001-05-18—20 SSR '01: Symposium on Software Reusability 2001 Toronto, Ontario , Kanada

2001-06-04—08 JavaOne Conference & Exhibition, San Francisco, CA, USA

2001-06-30-07-04 ISCA '01: 28th International Symposium on Computer Architecture, Göteborg
http://www.ee.princeton.edu/~isca2001/

Nya medlemmar

Vid senaste rådsmötet och vid VU sammanträde den 5 december antogs följande nya medlemmar i SESAM, vilka härmed hälsas välkomna även som läsare av och bidragsgivare till Rendezvous:

Calisto Data AB
HiQ Approve AB
IBM Svenska AB
Silicon Graphics AB

Kalendern

13 febr	VU
19 april	VU
19 april	AiS vårseminarium
26 april	Rådsmöte
Maj	Ag Metodik verktygssem
5 juni	VU
23 aug	VU
4 okt	VU
24-25 okt	Höstsem o Rådsmöte
6 dec	VU

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

forts ”Teknikbevakningsprojektet”

Efterlysning av artikelreferat eller originalartiklar

På liknande sätt efterlyser vi personer som vill åta sig att skriva korta sammanställningar eller referat om för SESAM-kretsen intressanta artiklar eller ämnen i de vanligaste Software Engineering och Ada publikationerna. Ex på sådana är Ada Letters, Ada User Journal, ACM Communications, IEEE Computer, IEEE Software, Software Engineering Notes, INCOSE Systems Engineering Journal.

Det behöver väl kanske inte sägas, men givetvis välkomnas också originalartiklar från SESAM-medlemmar (och andra) om intressanta och aktuella tekniska frågor, liksom diskussionsinlägg etc.

Erbjudanden om artiklar etc och intresseanmälningar om bidrag görs till Teknikbevakningsprojektet c/o Ingemar.Carlsson@mbox2.swipnet.se.

Push och pull

För att uttrycka sig i moderna kommunikationstermer, har det varit väldigt mycket ”push” i Rendezvous och väldigt lite ”pull” från läsarna. Detta skulle vi vilja ändra på. D v s vi behöver ”input” från SESAM-medlemmarna om vad Ni egentligen skulle vilja läsa om i Rendezvous och på SESAM hemsida. Fundera över det och hamra tangenterna!

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