Innehåll

1 Inledning 3

2 Keynotes 5
2.1 Jacobson
2.2 Bernstein

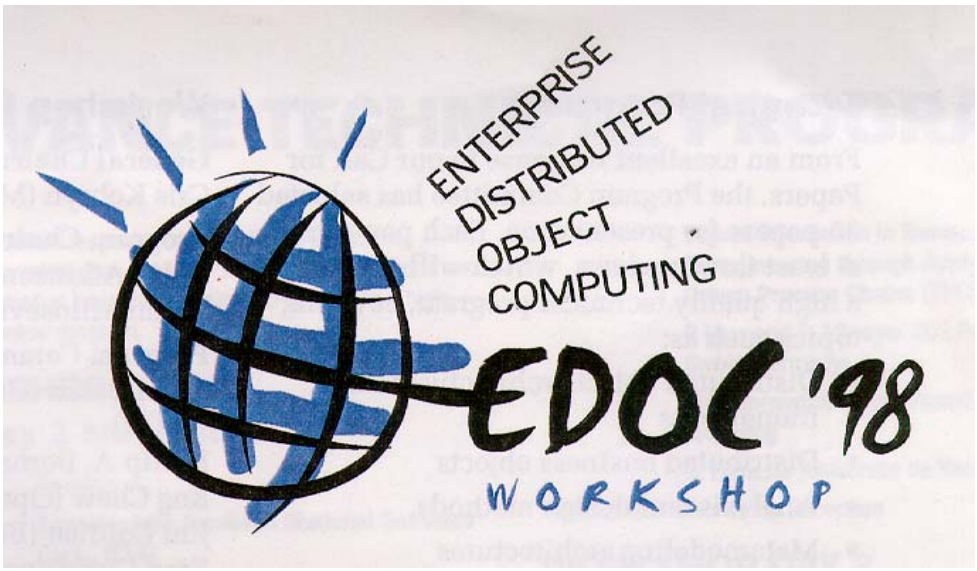
3 Några intressanta teman och papper 9
3.1 RM-ODP
3.2 Boger, et al: Electronic Contracting with COSMOS
3.3 Hoffner, Schade: Co-operation, Contracts, Contractual Match-Making and Binding
3.4 Karsten Riemer: "A process-Driven, Event-Based Business Object Model"

4 Debatt 15

5 Självrannsakan? 17

1 Inledning

Föreliggande rapport innehåller noteringar från konferensen EDOC ’98 som genomfördes i San Diego i början av november 1998. EDOC står för Enterprise Distributed Object Computing. Konferensens hemsida finns förmodligen fortfarande under http://uml.systemhouse.mci.com/events/edoc98/home.htm.



Konferensen gavs första gången förra året i Brisbane, Australien med sådan framgång att man beslutade göra den till en årlig konferens. Årets konferens är med andra ord den andra i ordningen. Till skillnad från många andra konferenser kring objekt/komponent-orientering har EDOC en genuint akademisk prägel. Papper skickas in något halvår i förväg, genomläses av en programpanel som väljer ut de bästa för presentation och inkludering i konferensdokumentationen (proceeding). Mer än 100 papper hade skickats in varav 36 valdes ut. Enligt programkommitténs ordförande Colin Atkinson var standarden genomgående hög på insända papper.

Tyvärr tycks även de utvalda papperen bli lästa av endast ett fåtal eftersom konferensen endast lyckades locka till sig ca 100 deltagare. Förmodligen – motsägelsefullt nog – på grund av den akademiska framtoningen. De mer kommersiella konferenserna under samma tema lockar ju tusentals deltagare. Måste vara en klar besvikelse för arrangörerna. Temat är helt rätt och ”inne”, platsen likaså. Men avsaknaden av ”närhet” till state-of-the-art i form av produkt/arkitekturanalyser, erfarenhetsredovisningar och en rejäl vidhängande utställningsdel begränsar intresset. Folk söker kompletterande information kring det som redan diskuteras/debatteras i fackpressen snarare än diskussioner runt intressanta teoretiska problemställningar eller intrikata lösningar på problem som ännu endast hör hemma i den akademiska världen. Alternativt koncentrerar man sig på att få tips om lösningar på alla egna akuta problem som tornar upp sig hemmavid. För problem av stora mått finns det både i närtid och när det gäller strategier kring arkitekturer, systemutvecklingsmetodik, mm. Vem kan idag med förtröstan fatta strategibeslut när såväl teknik som övriga förutsättningar för verksamhetens bedrivande både ofta och radikalt ställer etablerade villkor, spelregler, sanningar på huvudet?

En annan anledning till det bristande intresset kan vara att man inte haft resurser för marknadsföring. Kanske är inte EDOC ännu inpräntat i allas medvetande på samma sätt som t ex OOPSLA-konferensen. Kanske är man bara trängd av övriga stora konferens-arrangörer som alla ”hoppar på” de nya trenderna med *Enterprise*, *Internet, E-commerce* och *Component* som förtecken. Heder dock till både en snygg hemsida, proffsigt program och heltäckande dokumentation. För att inte tala om genomförande.

Kanske finns det viss anledning till självrannsakan i den akademiska kretsen. Är det inte så att industrin och dess förbundna organ för de facto standardisering står för såväl visionerna som innovationerna inom området för närvarande? Kanske orkar den akademiska världen inte hänga med, utom inom vissa nischområden. Att ha urvals-processer som tar mer än ett halvår från det tankarna sätts på pränt till dess presentationen genomförs är heller knappast lockande, utom i undantagsfall, i denna snabbföränderliga värld. Inte sällan redovisar papper kompletterande aspekter, tankar, lösningar på i andra sammanhang tidigare presenterade papper. Ofta är de framsprungna ur fleråriga projekt. För de redan initierade säkerligen angelägen information, för övriga knappast någon skakande upphetsning.

Överraskande statistik: Av det totala antalet accepterade papper (36 stycken) var endast 14 från USA. Tyskland var starkt representerat med 9 papper, Norge med 2. Inget från Sverige. Se dock följande avsnitt.

2 Keynotes

Varje konferens har inbjudna talare (keynote speakers). Det bör vara kända namn som lockar. Den ene var Ivar Jacobson, knappast okänd i Sverige. Knappast okänd i USA heller som en av de tre ”amigos” bakom UML – Unified Modeling Language. Den andre talaren var Philip Bernstein, en gammal räv inom branschen med förflutet som professor vid MIT. Hans primära intresseområde har under 30 år varit och är fortfarande databas-området.

2.1 Jacobson

Jacobson gav en historisk betraktelse över systemutvecklingsområdet ”sedan begynnelsen”. Flikade med omisskännlig näsa för PR-effekter in sin roll bakom Ericssons framgångar som en följd av en sällsynt välutvecklad och strukturerad utvecklingsmodell. Med start redan på 60-talet, över standardisering inom telekommunikationsvärlden i form av SDL, till något som sedermera kom att bli Objectory-metoden.

Nu vidareutvecklad och förfinad som the Unified Software Development Process, eller kort och gott the Unified Process. Jacobson spenderade en hel del av tiden kring denna processmodell, dels för att den är intressant, dels för att den utgör fundamentet för ett nytt webbaserat stödverktyg som Rational tagit fram under beteckningen ’Rational Unified Process’. Verktyget vägleder utvecklingsteam genom utvecklingsprocessen, förmodligen med referenser till var, när, hur andra stödjande verktyg (Rationals?) kommer in i bilden. Härmed har modelleringsspråket UML fått ett nödvändigt komplement i form av en processbeskrivning. Processens nyckelord är ”Iterative and Incremental”, ”Use Case Driven” och ”Architecture Centric”. Rational, Jacobson och många andra ser förstås gärna att Unified Process etableras som en industristandard. Förmodligen får man rätt, med den gedigna erfarenhetsbas processen baseras på, i kombination med det ”flyt” UML har för närvarande.

Den som inte vill köpa Rationals verktyg kan istället köpa boken *The Rational Unified Process: An Introduction* skriven av Philippe Kruchten, och utgiven på Addison-Wesley. Eller, som Jacobson mycket lämpligt påpekade, går det ju alltid bra att vänta ett litet tag tills boken *The Unified Software Development Process* kommer ut på Addison-Wesley. Den är nämligen skriven av – just det – *Ivar Jacobson, Grady Booch,* och *James Rumbaugh.*

Användningsfall (use cases) fick också sin beskärda del av presentationstiden. Det är för övrigt närmast overkligt vilket intresse användningsfall väcker, inte minst i anslutning till objektorienterad systemutveckling. Idén är ju att på ett rimligt formaliserat sätt definiera vad systemet förväntas ge för typ av service till varje användare (kategori). Det är ju knappast något nytt, knappast så där vansinnigt avancerat heller. Däremot är en sådan kartläggning självklart av avgörande betydelse för att öka chansen för att systemet kommer att bli verkningsfullt. Kanske kan en hel del av förklaringen hämtas i att OO-företrädare i gemen tidigare knappast varit speciellt intresserade av, än mindre kommit i kontakt med, systemutvecklingens tidigare faser. Det har mest varit ”pang på rödbetan”, häftiga realiseringar. Jacobsons budskap är i det perspektivet extra värdefullt. OO-baserad systemutveckling behöver också ske under ordnade former.

Dock, slår det åhöraren; hela utvecklingsprocessen består av ett stort antal steg, vart och ett viktigt för helheten. Med iterationer blir det hela mycket komplext. Visst, säkert kontrollerbart och med ett väldefinierat utfall. Men orkar vi, hinner vi utveckla system på det sättet framöver? Med verksamheter som byter skepnader kontinuerligt, med påföljande krav på systemens anpassbarhet? ”Shortcuts” av olika slag behövs. Unified

Process är i det avseendet flexibelt – är mer som ett ramverk som kan preciseras efter behov och omständigheter. Självfallet ställer det istället rätt ansenliga krav på kompetens kring såväl processen, med alla sina intrikat inflätade steg och iterationer, som det tänkta systemet och dess syfte. Blir det för komplicerat eller för osäkert att ”sy ihop” en egen process kanske man, av trängande skäl, realiserar en egen variant – med än mer osäkra följder. Med andra ord dilemman mellan enkelhet och precision, mellan flexibilitet och formalism.

Helt klart befinner vi oss vid randen av en omfattande förnyelse av systemutvecklings-processen. Kanske går det äntligen att nå fram till en allmänt vedertagen process på samma sätt som det, mot alla odds, när tiden var mogen gick att nå fram till ett standardiserat modelleringsspråk (UML)? Kanske skapar komponentansatsen med sina enklare modeller och teknikabstraktioner nya, enklare förutsättningar? Kanske är det helt enkelt så att systemutvecklingsfasen måste tillåtas ta tid och genomföras på ett väldefinierat sätt, både i ett system- och användarförankringsperspektiv, för att helhetslösningen ska kunna svara upp mot förväntningarna? Kanske behöver vi ännu klipskare utvecklare som både förstår den nya tidens teknologi och förmår utnyttja den flexibelt, samtidigt som vederbörande har närkontakt med verksamhetens behov? (Är vi på väg mot en ytterligare accentuering av produktivitetsskillnader mellan mycket duktiga och helt vanliga utvecklare?) Kanske kommer vi mer sällan att bygga nya system utan snarare och oftare komplettera/anpassa välutvecklade stommar? Tänk om det istället blir så att vi mycket snart kommer att arbeta med teknikförutsättningar och informations-stödsbehov som ställer det mesta av vad vi idag refererar till som kloka arkitekturer och utvecklingsmodeller på huvudet?

En konferensdeltagare reflekterade över om exempelvis inte användningsfalls-ansatsen måste vidareutvecklas till något som svarar mot dessa nya förutsättningar. Användare och system blir allt svårare att hålla isär. Inte minst nya intelligenta agenter, aktiva affärs-objekt mm kommer att göra dem till såväl system, nyttjare av system som, i ett längre perspektiv, producent och konsument av system. En verksamhet kommer att bestå av en dynamisk uppsättning samverkande Business Components (mänskliga och datoriserade) som tillsammans utför en uppsättning uppgifter som representerar verksamhetens syfte och utförs som en följd av de signaler (i vid bemärkelse) verksamheten är satt att reagera på. Så är ju i och för sig fallet redan idag men framtiden kommer att i betydligt större utsträckning göra människor och system flexibelt och dynamiskt utbytbara mot varandra.

Att systemutveckling alltid har varit, fortfarande är och förmodligen framgent kommer att vara ett område för frustration, tunga misslyckanden, våldsamma kostnader verkar vara ett axiom. Med andra ord ett svårlösligt dilemma. Förhoppningen ligger på de nya trenderna med komponentorientering, industriell utvecklingsprofil, återanvändning, komponentmarknader, effektiva programmeringsspråk och utvecklingsverktyg. För, bland all uppgivenhet finns trots allt fler som lyckas och lyckas över förväntan. Vad som dock tycks genomgående för dessa är mindre team av mycket duktiga, målinriktade människor med både verksamhets- och teknikprofil. Tyvärr räcker ju inte dessa efterfrågade individer till alla projekt, överallt. Ständig utbildning/vidareutbildning blir ett allt nödvändigare krav.

Till sist; Ivar Jacobson är en person man på konferenser lyssnar till med all den vördnad som bör ankomma en person med en ”livstid” bakom sig som nyskapare och språkrör i branschen.

2.2 Bernstein

Bernstein blev för ett antal år sedan lockad till Microsoft. Han är inte ensam. Många klipska IT-människor har på senare tid gått samma väg. T ex Jim Gray, som just blivit utvald som mottagare av 'the 1999 ACM Turing Award', dataområdets Nobelpris. Grey var på 70-talet en centralfigur inom IBM vad gäller databashantering, speciellt transaktionshanteringens alla vindlingar. Hamnade sedan på Digital och Tandem för att, som sagt till sist? hamna hos MS.

Åter till Bernstein. Han är nu ansvarig för Microsofts repository-satsning. Repositorier är en produktkategori som länge levt en undanskymd tillvaro men som fler och fler nu börjar få upp ögonen för igen som en följd av de alltmer ökande kraven på översikt och kontroll över företagens informationsresurser. Ta exempelvis Data Warehouse (DW). Ingen DW-produkt klarar sig utan ett samverkande repository. Och vilket företag tittar inte idag på DW? De båda keynote-presentationerna har för övrigt repository som en gemensam nämnare:

Oavsett om man använder the Unified Process eller något annat och oavsett vilket verktygsstöd som används under arbetet behöver det definierade lagras någonstans. Det är just detta ett repository är till för.

Inte många från den tidigare generationen repository-produkter har överlevt. Nu är det huvudsakligen Unisys, Platinum, IBM och Oracle som konkurrerar med Microsoft. Därutöver finns självfallet någon typ av repository inbyggd som komponent i de flesta avancerade utvecklingsverktyg och ERP-system. I det senare fallet är det inget själv-ändamål utan en tvingande nödvändighet i brist på alternativ. Det är bland annat för den marknaden Microsoft (och säkert alla de andra) vill kunna erbjuda ett standardiserat alternativ att "plugga in" istället för det egenutvecklade. Repositoriet tillhör ju sannolikt inte deras "kärnverksamhet" lika lite som t ex en DW-produkt har behov av att binda upp sig mot en viss databashanterare. Ett repository ska med andra ord inte upplevas som en produkt i sig utan som en komponent i ett större sammanhang, låt vara en synnerligen central sådan. Det som bland annat skiljer ett repository från en databashanterare är en fördefinierad, ofta utbyggbar, informationsmodell, ett antal stödjande verktyg som specialutformats för att operera mot informationsmodellen och en funktionalitet för att styra och kontrollera innehållet i såväl informationsmodell som databas. Givetvis finns någon form av databas tillgänglig under detta serviceskikt.

Microsoft anser sig ha en så kallad öppen informationsmodell. Den är dock inte underkastad något fristående standardiseringsorgan utan öppen så tillvida att man talar om hur den ser ut. Helt naturligt strävar man efter konsistens med bland annat UML, COM, SQL, Java och andra vedertagna modeller. Metamodellen är tyvärr inte överensstämmande med OMGs MOF, vilket försvårar standardiserat utbyte av modelldata genom ett enhetligt gränssnitt. Förmodligen hoppas Microsoft som vanligt på en marknadsdominans. Vem bryr sig då om anpassning till de factostandarder andra än de egna? Utbyte av modelleringsdata åstadkoms genom traditionell så kallad brygga, d v s ett överenskommet överförings- och inkodningsformat. CDIF (Common Data Interchange Format) har länge varit en använd standard för ändamålet. Här har man istället valt att generera XML-strukturer i form av DTDer (Data Type Definition). Motsvarande ansats är för övrigt på väg att accepteras som standard inom OMG. Se inlämnat förslag till SMIF (Stream based Model Interchange Format) med beteckningen XMI (XML Metadata Interchange) i OMGs dokumentkatalog (under ad/98-10-07).

Av tradition har repositorier associerat till systemutvecklingsverktyg. Vilket knappast genererat något våldsamt intresse. Bernstein anser att Data Warehousing nu träder fram som en typisk "killer application" för repository på samma sätt som det trätt fram som en skänk från ovan för databasleverantörerna. Den konventionella databasmarknaden växer

endast marginellt. DW exploderar. Förståelse och respekt för metadata kommer som en direkt följd.

Men det räcker inte med en databas och en informationsmodell i ett repository. Om flera verktyg jobbar mot samma, vilket man gärna önskar, gäller det att stämma av, inte bara att de arbetar i enlighet med semantiken hos informationsmodellens begrepp. Även hur och när man läser respektive uppdaterar databasen är av avgörande vikt. Det är visserligen sant för alla typer av databaser men den våldsamt komplexa datastrukturen i ett repository lägger en helt annan ansvarsdimension på de operationer som utförs. Om olika verktyg dessutom används för att tillsammans ge ett modelleringsstöd gäller det att deras samverkan är väl injusterad såväl mot processen som mot repositoriet. En ytterligare faktor att ta hänsyn till är behovet av att hålla kvar det historiska perspektivet på alla uppgifter under systemets livscykel. Till skillnad från de flesta databaser, som bara lagrar aktuella uppgifter, måste i repositoriet en avancerad versionshantering upprätthållas. Det är för att hantera alla dylika omständigheter som ett managementstöd behövs för övervakning, styrning, mm.

Förvånande nog tycktes många av deltagarna få en ”aha”-upplevelse av Bernsteins presentation, som om reporitorier var en helt ny företeelse. UML som möjlig repository-modell, utvecklingsverktygens behov av repositorystöd samt repositoriernas behov av att utväxla information tycktes vara nya reflexioner. Nåväl, en gemensam semantik (UML, mm), ett gemensamt överenskommet sätt att transportera informationen mellan verktyg och gärna ett standardiserat gränssnitt mot repositoriet konstaterades snabbt vara angelägna ingredienser i en repositorymiljö.

Bernstein ansåg avslutningsvis att repository-området kryllar av intressanta utmaningar för forskare, är komplext men samtidigt ’Big Business’, inte är outforskat men väl underutvecklat. Kraftsamling behövs!

Faktum är att Microsofts Repository redan skeppats i över 1 miljon exemplar tillsammans med Visual Basic. Det kommer även att packeteras med SQL Server 7.0. Över 70 mjukvaruleverantörer bygger tillämpningar baserade på produkten.

Kanske kommer Microsofts sena men kraftfullt markerade inträde på repository-arenan att bana väg för en pånyttfödelse i stor skala.

3 Några intressanta teman och papper

Att referera **alla** papper är knappast varken intressant eller produktivt. Tre papper har valts ut för kortare referat dels därför att de är intressanta, dels för att de är rätt typiska för konferensen. Den som vill fördjupa sig i övriga papper måste köpa en proceeding. Vart man vänder sig framgår av konferensens hemsida, se ovan. Den som nöjer sig med de ljusbilder respektive talare använde för att presentera sitt papper finner dem direkt på hemsidan under ”Final Program”. Men först ett tema som på något vis utgjorde ledstjärnan på konferensen och därmed indirekt gav en extra seriositet till konferensen, nämligen RM-ODP.

3.1 RM-ODP

En full tutorial och några av konferenspapperen kom, hedrande nog, att på olika sätt beröra RM-ODP, Reference Model for Open Distributed Processing, som är beteckningen på en grupp standarder som tagits fram av ISO och ITU-T i samverkan (ISO 10746-1→ 10746-4 eller ITU-T X.901→ X.904).

Knappt några inom området distribuerade system tycks känna till standarderna. Övriga gärna refererar till dem, men mest av artighetsskäl. Många anser ointresset vara oförtjänt eftersom mycket av innehållet är synnerligen välgenomtänkt och välbearbetat stoff som resultat av ett mångårigt arbete. Anledningen kan ligga i standarders alltid lika tråkiga layout, stela innehåll och avsaknad av pedagogisk finess. RM-ODP har därutöver ett ganska abstrakt innehåll som inte är så lätt att ta till sig om man inte ”är med på noterna” när det gäller grundidéerna. Nåväl, innehållet är tänkvärt och platsar väl som en referensmodell över distribuerade informationssystem.

Värt att nämna är att OMG ställer krav på att alla förslag som kommer in för bedömning ska referera till hur och var det relaterar till referensmodellen RM-ODP. Dessutom pågår omarbetning av OMGs Object Management Architecture (OMA) för att anpassa den mer till RM-ODPs viewpoint-definitioner (se vidare nedan).

Kort om standarden

Open Distributed Processing (ODP) står för “systems which support distributed processing allowing heterogeneity of components and crossing of organisational boundaries (autonomy)”.

RM-ODP är ett ramverk uttryckt som “framework for distribution, interworking, interoperability and portability”. Syftet är att ge en abstrakt och generell modell (“the big picture”) som beskriver portabilitet trots heterogena miljöer, döljer konsekvenserna av en fysisk distribution samt hanterar samverkan mellan ODP-system så att information kan utbytas meningsfullt och funktionalitet kan nyttjas på ett smidigt sätt. Man bör se RM-ODP som ett tankestöd och/eller designstöd.

För ändamålet har man formulerat fem olika infallsvinklar (viewpoints) på ODP:

– Enterprise Viewpoint
  som beskriver ett systems syfte, omfattning, inriktning, principer, mm

– Information Viewpoint
  som beskriver informationens semantik och informationsbehandlingen

– Computational Viewpoint
som beskriver det funktionella perspektivet, den funktionella nedbrytningen

– Engineering Viewpoint
som beskriver den erforderliga infrastrukturen för att stödja distribution

– Technology Viewpoint
som beskriver val av teknologier för implementeringen

Mest i fokus just nu är Enterprise Viewpoint. Den anses för otydlig i sitt nuvarande skick och har visat sig ge upphov till missuppfattningar av olika slag. Det otydligt sagda är det otydligt tänkta, kanske? En av de aktiva i standardiseringsarbetet nämnde rättframt att man när standarden skrevs hade en allmän känsla för behovet av denna infallsvinkel men helt enkelt inte visste hur den skulle definieras. Ett annat syfte är att väcka intresset för den, eftersom man anser den vara av central vikt för att uppnå ett välfungerande system. Tekniker tycks ha en nonchalant inställning till vyn eller i alla händelser en (alltför) snabb uppfattning om vilka begrepp som behövs för att beskriva den. Samtidigt begär man tolkningsföreträde till begreppens innebörd trots avsaknad av någon som helst unik kompetens inom detta verksamhetsrelaterade perspektiv.

Arbete pågår med andra ord för att expandera och precisera Enterprise Viewpoint. Arbetet utförs inom ISO-JTC1/SC7/WG3. I första hand inriktar man in sig på att ta fram en uppsättning begrepp med vars hjälp det går att beskriva det distribuerade systemets syfte, krav, omfattning, organisatoriska hemvist, aktörer, roller, resurser, regler, … Observera att ett distribuerat system här inte bara representerar en teknisk lösning utan utgör ett helhetsperspektiv inklusive samtliga involverade aspekter kring dess existens. Några aktuella begrepp är Enterprise Object, Community, Role, Contract, Policy, Permission, Obligation, Resource, Activity, Process, Action, Physical och Social Behaviour.

Kanske borde begreppen, när de väl preciserats, föras in som en profile i UML? I dagsläget verkar dock samarbetet mellan UML- och RM-ODP-falangerna vara endast av högst spontan karaktär. Om inte annat borde RM-ODP-gruppen, av rent praktiska skäl och för att nå ut med sina budskap, använda sig av samma modelleringsbegrepp för att formulera sina modeller som UML är definierat i, dvs den som finns definierad i OMGs Meta Object Facility (MOF). Värt att notera är att dessa begrepp överensstämmer med dem UML använder för att definiera objektmodeller.

Till sist några ord om *Transparencies* som ett exempel på hur RP-ODP arbetar med abstraktioner för att dölja komplexitet. En Transparency är en abstraktion eller inkapsling av sådan komplexitet som systemutvecklaren inte ska behöva bekymra sig om, som ska finnas som services i den underliggande stödjande infrastrukturen.

Man har definierat följande initiala uppsättning Transparencies:

- Access transparency; döljer skillnader i datarepresentation och anropsprinciper
- Location transparency; döljer fysiska adresser och uppdelningen lokala/externa referenser
- Relocation transparency; refererad komponent kan flyttas obemärkt utan effekt för den refererande
- Migration transparency; frikopplar en relokering från objektet självt

- Persistence transparency; maskerar en komponents aktiva och passiva faser
- Failure transparency; frikopplar misslyckande och återskapande från objektet självt
- Replication transparency; ser till att eventuella kopior alltid håller konsistens sinsemellan
- Transaction transparency; döljer den koordination som behövs mellan objekt i och för en viss operation de utför tillsammans .

Detta var bara några axplock från RM-ODP-standarden. Den som är intresserad av att veta mer rekommenderas titta in hos DSTC i Australien där man ägnat mycket tid åt distribuerade system och åt RM-ODP:
http://www.dstc.edu.au/AU/research_news/odp/ref_model/ref_model.html.

3.2 Boger, et.al: Electronic Contracting with COSMOS – How to Establish, Negotiate and Execute Electronic Contracts on the Internet

Elektronisk affärsverksamhet eller e-commerce är på allas läppar. Givetvis innehöll även denna konferens några papper inom detta område. Detta papper berör, som framgår av titeln, främst kontraktsskrivning.

Varje kommersiell transaktion omfattar enligt författarna explicit eller implicit tre faser. Den första fasen kallas information phase. Där söker man, jämför, testar, ökar kunskap. Om fasen leder fram till kontakt mellan parter i och för fullgörande av en affärstransaktion vidtar nästa steg, negotiation phase, under vilken det gäller att komma överens om villkoren. Fasen utmynnar i ett underskrivet kontrakt (eller om implicit, ett ”handslag”). Därefter utförs det som överenskommits i execution phase.

Om kontraktet ges en strukturellt fast form med given semantik i alla dess beståndsdelar kan kontraktet förutom att vara ett legalt dokument även fungera som den gemensamma punkten för styrning och kontroll av transaktionen. Bland annat måste ingå uppgifter om respektive part, om vad som ska åstadkommas (överföring av varor, pengar, tjänster, licenser, …), hur det ska ske, legala avsnitt, beroende till andra kontrakt, mm. I den föreslagna lösningen benämnd COSMOS (www.ponton-hamburg.de/cosmos) lagras kontrakten i en objektdatabas i enlighet med en relativt komplex datastruktur. Översättning till XML-format tillämpas för nödvändiga överföringar av kontraktsutkast mellan parter, mm.

I COSMOS sköts förhandlingarna genom en elektronisk mäklare (Broker). Principerna bygger på ODP/CORBA Trader Service. Mäklaren har till sitt förfogande en informationsbas bestående av standarduppgifter om de parter som kan vara potentiella kontraktsskrivare i olika sammanhang, vad som finns till försäljning och av vilka, vad som önskas köpas och av vilka samt en uppsättning standardkontraktsskelett. Den som har något att erbjuda, säljaren, registerar detta som en Service Type hos mäklaren. Köparen definierar på motsvarande sätt sitt intresse för köp. I båda fallen innehåller Service Type ett antal uppgifter som preciserar levererad respektive önskad/krävd Quality of Service (QoS). Dessa QoS-attribut är viktiga ingredienser i och för mäklarens matching av lämplig köpare-säljare-kombination.

Förhandlingsjobbet består till stor del i att jämka på de krav som preciseras i de båda parternas QoS så att man till slut enas om en uppsättning som kan ingå i kontraktet. För kontraktsskrivningen har mäklaren som nämnts tillgång till ett antal mer eller mindre standardiserade kontraktsmallar för olika typsituationer. Vissa delar kan vara ifyllda medan andra ligger på mäklarens ansvar att komplettera med. Saknas mall måste kontraktet kreeras i sin helhet.

Som läsaren säkert anat är både köpare och säljare i COSMOS realiserade som Business Objects (BO), d v s som elektroniska företrädare för någon reellt legal part. Samtliga inblandade komponenter är för övrigt uppbyggda som BOs. Tanken var att de skulle realiseras och fås att samverka över OMGs Business Object Component Architecture (BOCA). I och med att BOCA röstades ner av OMG i somras får man nu söka en annan lösning.

När väl kontraktet är undertecknat återstår att utföra det överenskomna, d v s utföra exekveringsfasen. Det som ska utföras finns beskrivet i kontraktet som en formell workflow-specifikation uttryckt i en Petri-Net-modell. Specifikationen kan, när mänskliga företrädare är inblandade, om så är lämpligt presenteras och uppdateras genom ett grafiskt användargränssnitt. Specifikationen tolkas av en workflow engine i COSMOS. Den tar i sin tur de nödvändiga kontakterna med involverade parters ställföreträdare i exekveringsrummet. Eftersom även dom är realiserade som BOs sker kontakten genom anrop av någon av dess methods. Dessa BOs förutsätts sedan ha de nödvändiga kontakterna med sin ”riktiga” part för att i interaktion med dem se till att begärda krav verkställs i realiteten.

3.3 Hoffner, Schade: Co-operation, Contracts, Contracual Match-Making and Binding

Även detta papper handlar om e-commerce med koncentration på kontraktsöverenskommelsefasen. Man konstaterar, något utförligare än papperet ovan, att livscykeln kring ett samarbete mellan parter för något syfte schematiskt omfattar följande steg:

- Contractual match-making; Parter för ett samarbete söks och identifieras.
- Exchange information, negotiation and signing a contract; Förhandling och acceptans av en överenskommelse.
- Contract translation; De nödvändiga förutsättningarna och den beredskap som krävs inom varje organisation för att utföra det överenskomna jobbet sätts upp.
- Contractual binding; Nödvändig kontaktyta mellan organisationerna realiseras så att samarbetet kan genomföras på ett sätt som svarar mot överenskommen Quality of Service.
- Contract Consummation; Produktion och konsumtion av överenskommen service.
- Co-operation dismantling; Samverkan avslutas i ordnade former.
- Co-operation analysis; Utvärdering inom respektive organisation mot de förväntningar som låg till grund för kontraktsskrivningen.

Papperet koncenteras mot den första och fjärde aspekten. Under hela livscykeln måste dock kontakterna etableras mellan de fristående parterna på ett sätt som svarar upp mot

”trust, accountability, liability and responsibility”. För att underlätta detta förutsätts varje part sätta upp en gateway genom vilken all kommunikation ombesörjs avseende ett kontrakt. En gateway utför kontroller, transformeringar, övervakning av information som flyter in och ut över partens gräns. Jämför det tidigare papperets Business Objects.

Contractual match-making

Papperets förslag är även det en vidareutveckling av ODP/CORBA Trader Service. Denna service bygger som refererats i det tidigare papperet på att producenten lämnar över ett erbjudande i form av en Quality of Service-lista till en mäklare (Trader). Konsumenten formulerar på motsvarande sätt sina önskemål. Finner mäklaren en producent som svarar mot önskemålen är saken klar enligt ODP/CORBA Trader Service. Papperet konstaterar att detta är ett ensidigt sätt att se på saken. Det borde finnas möjligheter även för producenten att ”syna” den tilltänkta konsumenten (d v s dess QoS-lista), något som för övrigt det tidigare papperet tagit för givet. Ett skissat sätt att åstadkomma ”jämvikt” är att komplettera de båda parternas egenskapslistor (QoS-listor) med en uppsättning villkorskonstruktioner som mäklaren utvärderar både för konsumentens och producentens räkning. Som en ytterligare förfining kan både QoS-lista och villkor sändas över till motparten för egna bedömningar som ett komplement till mäklarens.

Contractual binding

Papperet redogör sedan för ett par olika sätt att generera förutsättningarna för utförande av det som överenskommits. Bland annat genereras de gateways som behövs som kontaktyta mellan parterna under ”exekveringen”.

3.4 Karsten Riemer: ”A process-Driven, Event-Based Business Object Model”

Global konkurrens, snabba kast hos marknaden, komplexa affärsberoenden, allt snabbare generationsskiften hos produkter, mm ställer företagen inför tuffa förutsättningar. Nya affärsmöjligheter har ofta mycket begränsat tidsintervall för att lyckas. Eftersom affärsprocessen är det centrala – det är den som skapar affärsmöjligheten – krävs snabb anpassning av systemstöden till denna. En väl avvägd sammansättning av köpta, återanvända och egenutvecklade system behövs. Enbart färdiga paket skapar låsningar, egenutveckling är dyrt. Komponenter framstår mer och mer som utmärkta byggstenar för ändamålet. Dessutom krävs väldefinierade gränssnitt som möjliggör ”plug-and-play”.

Oavsett vald strategi underlättas anpassningar om systemlösningen har en uppbyggnad och ett språkbruk som i möjligaste mån svarar mot affärsprocessen.

Riemer anser att det går att skapa både en affärsmodell och en funktionell systemspecifikation med hjälp av samma uppsättning modelleringsbegrepp eller komponenttyper. Fyra av dessa utgör den standardmässiga basen nämligen Business Object, Business Event, Business Process och Business Rule.

Ett Business Object är en representation av en företeelse som är känd i verksamheten, är identifierbar och har såväl tillstånd (vad som gäller just nu) och beteende i form av en uppsättning methods som arbetar mot tillståndsbeskrivningen.

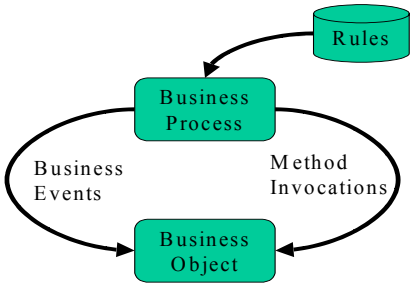
Ett Business Event är något som inträffar när tillståndsbeskrivningen för ett Business Object ändras.

En Business Process består av ett antal arbetssteg som opererar på och ändrar tillståndet för Business Objects..

En Business Rule är en uppsättning villkor som avgör när Business Events ska/kan inträffa i relation till varandra och i relation till omständigheter hos involverade Business Objects.

I normalfallet är Business Objects och Business Events stabilare till sin specifikation än de andra två.

Tillståndet hos ett Business Object inspekteras eller ändras i och med att någon av dess methods utförts tillfullo. När så skett genereras ett Business Event som publiceras. Vilken komponent som helst kan abonnera på valfri uppsättning Business Events. Självfallet avgör verksamhetens dynamik vilka abonnemang som är produktiva.



Modellen är enkel men kraftfull. Åtminstone kan den utgöra en god bas för förfiningar.

Riemer visar sedan hur samma modell i princip kan realiseras utan större förändringar genom nyttjande av existerande teknologier som Notification services, containers som Enterprise JavaBeans, transaktionshantering av varje process-steg, osv.

4 Debatt

Ett par paneldebatter fanns inplacerade i programmet. Här följer några noteringar från debatten ”Enterprise Architecture - Rigor vs Pragmatism”. Ett spännande tema för en debatt. Tyvärr ”tände” den aldrig. Kanske beroende på sammansättningen av debattdeltagare, kanske på upplägget, kanske beroende på att inga alltför motstridiga åsikter fanns att debattera. Varje debattör berättade om sitt ”skötebarn” (repository, RM-ODP, UML, …). Ingen försökte definiera betydelsen av ’Enterprise Architecture’. Däremot var alla ense om att det **inte** var detsamma som ’Infrastructure’. Den förra berör någon verksamhetsrelaterad aspekt medan den senare refererar till den teknologi som används för ett systems realisering.

Däremot hamnade man snabbt i frågeställningen om UML borde kunna beskriva olika aspekter på ’Enterprise Architecture’ eller ej. Något som borde vara svårt att ta ställning till innan man rett ut vad begreppet betyder. Nåväl, alla var rörande ense om UMLs stora värde som enande faktor när det gäller begrepp och notation för systemmodellering vid olika faser av systemutvecklingen. Däremot var man inte ense om var ansvaret började och slutade. Skulle hela systemets livscykel täckas in från första analyssteget till systemets avveckling? Skulle betoningen ligga på systemutvecklingsdelen och där endast på systemdesign, därmed inte innefatta analys och implementering? Kanske analys ändå? Några entydiga besked gavs inte men väl ett antal synpunkter:

Många menade att UML har ett antal luckor, inte minst när det gäller perspektivet Business Modeling. Mot detta ställdes synpunkten att det gäller att skilja på Enterprise Business Architecture och Enterprise System Architecture. Den förra sammanfaller med vad RM-ODP klassar som Enterprise Viewpoint. UML har inget ansvar för att uttrycka denna vy. UML bör entydigt orienteras mot systemperspektivet. Varpå någon kontrade med att påpeka att det knappast går att bygga bra system om man inte dessförinnan klarlagt dess syfte i ett verksamhetsperspektiv, dvs Enterprise Viewpoint. T ex måste ett systems roll och ansvar kunna beskrivas – helst på ett formaliserat sätt. Utan ett välgenomtänkt verksamhetsorienterat processperspektiv som innefattar både väl uttalade mål, beskriver vilka roller som ska stödjas och på vilket sätt samt systemets övergripande plats i organisationen blir utfallet ett lotteri med många nitlotter. Systembyggnaden hamnar i händerna på tekniker, sådana som ofta anser dessa aspekter vara irriterande och försvårande omständigheter. Alltså bör Enterprise Business Architecture perspektivet inkluderas i UML.

Några ansåg att UML generellt behöver byggas på med mer stringent semantik. Varför inte, menade någon, utveckla UML till ett visuellt programmeringverktyg, dvs göra det så detaljerat att i stort sett allt som behövs för full exekverbarhet finns specificerbart. Åtminstone borde UML ge mer stöd för komponentbaserad utveckling, något även Business Object Design Task Force (BODTF) inom OMG anser. (OMG-medlemmar kan hitta mer information i dokumentet ad/98-10-09.pdf på OMGs hemsida.) Andra önskade bättre processbeskrivning och regeldefinitioner. I princip borde vidareutvecklingen av UML vara ett kontinuerligt pågående arbete, vilket det i dagsläget också ser ut att vara i OMG. Någon påpekade att även hanteringen (styrning, kontroll, dimensionering, …) av ett system i drift måste modelleras eftersom det i vissa fall resulterar i ett eget automatiserat system som på olika sätt integreras med grundsystemet.

På något sätt kändes det som att det ursprungliga syftet med UML, att ensa begrepp och notation för modellering av systemdesign, smygande hade flutit över något slags totalsyfte. Man kan förmoda att om detta vinner gehör inom OMG kommer vi snart att övergå från kvalitet till kvantitet, inte både och. UML blir ett slags uppslagsverk att

hämta förklaringar och tips ifrån, inte den stringenta ensning av begreppsapparaten för visst syfte den ännu representerar.

Med fortlöpande tillflöde av begrepp i UML ökar svårigheten att uppnå entydig användning av alla begrepp exponentiellt med tanke på alla relateringar som måste upprättas och vidmakthållas mellan begreppen. Observera att redan idag ett antal av begreppen i den antagna standarden står under het debatt. Den som tittar in på OMGs hemsida under 'OMG Revision Issues' finner att den grupp som arbetar med revidering av UML har en mängd utestående frågeställningar att reda ut. I slutet av november 1998 var det 394 stycken. Och då är det inte fråga om vidareutveckling – bara förtydliganden, felrättningar, mm. Med ytterligare kontinuerlig påspädning kommer inte oklarheterna att bli färre. Detta gäller även om man försöker bryta upp begreppen i olika så kallade profiles.

Nog är det lite överoptimistiskt att tro att UML ska kunna bli all modellerings begrepps-källa. Systemen är för mångfacetterade, liksom infallsvinklarna, liksom syftena, liksom användarna, liksom …. Hur tro att alla dessa behov ska kunna samordnas och tillgodoses av ett standardiseringsorgan, OMG? Världen väntar inte på OMG. Låt UML stå för vad det från början var tänkt för. Komplettera med brister, rätta fel, förfina vid behov. Men var vaksam på att inte glidande expandera syftet. Utveckla ett separat XXX för det nya syftet. Det kommer alltid att finnas fler typer av modeller och flera olika syften för varje modelltyp. Det är inte att begära att en enda uppsättning begrepp ska klara allt detta. Knappast ens önskvärt. Olika modeller och olika syften kan för övrigt kräva helt olika modelleringsspråk. UML är idag baserat på OMGs MOF. Kanske klarar inte MOF att formulera alla typer av begreppsmodeller? Kanske är MOF i andra fall som att skjuta mygg med kanoner?

För övrigt får formaliseringen inte drivas in absurdum. Behövs både precision (exv UMLs Object Constraint Language - OCL) och vanlig engelska i modelleringsarbetet. För mycket precision kräver ett komplext nytt språk som endast några få förmår använda riktigt. Språket blir ett hinder för övriga och får av den anledningen inte något avgörande genomslag/momentum. Dessutom måste man ha klart för sig syftet med språket. Precision är inget självändamål. Värt att betänka är att problemen ofta inte ligger i språkformalism utan i bristande kunskap om den värld som ska modelleras. Något som knappast är en kritik, bara en realitet i de flesta systemutvecklingssammanhang man måste ha förståelse för och ta hänsyn till.

Som en avrundning framhölls det att svåra problem är och förblir svåra. De blir knappast enklare genom modelleringsstöd och UML. Däremot mer hanterbara. I stort sett alla hoppas på att komponentbaserad utveckling ska ge det lyft som aldrig objekt-orienteringen och dess objektmodell fick i systemeringssammanhang. Varning dock; komponentorienteringen är ingen "silver bullet". Har vi otur går den samma väg som OO. Har vi tur, inklusive en hel del skicklighet, kan komponentorienterad systemutveckling komma att bana väg för en ny systemutvecklingsera.

Debatten lär gå vidare!

5 Självrannsakan?

Med tanke på det låga deltagarantalet uppstod dels frågan om det var lönt att satsa på ett tredje EDOC, dels frågan om det i så fall fanns anledning till ändring av profilen. Ett förslag önskade utökning till IEDOC där I skulle stå för ”Inter”, eftersom många system bygger på samverkan över företagsgränserna. Ett annat var att C-ets Computing skulle få innebörden ”and Components” i stället. Komponenter är gångbart som dragplåster medan objekt knappast är det längre. En spontan omröstning gav dock mycket markant övervikt för att ha kvar den nuvarande innebörden. Dock ansågs allmänt att Workshop skulle bytas ut mot Conference för att indikera en ambitionshöjning. Risken att intresserade tror det är en aktivitet för ett mer slutet sällskap försvinner på köpet.

EDOC ’99 kommer att gå av stapeln 27-30 september 1999 i Mannheim, Tyskland.

Till sist; På tal om objekt konstaterade en deltagare att vi har en övertro på att dagens middleware-lösningar är någon slags ultimär slutpunkt. De är för teknikorienterade och i många stycken – trots sina avancerade egenskaper – för begränsade i sina egenskaper. De bygger på en objektmodell som egentligen bara de riktigt inbitna älskar och som bara programmerarna orkar sätta sig in i. Framförallt saknar lösningarna verksamhets-perspektivet i sina modeller. Komponentorientering, som bygger på gamla beprövade utvecklings- och modelleringsprinciper, ligger betydligt närmare den stora skaran utvecklare och är betydligt lättare för icke-tekniker att ta till sig. Får komponentbaserade system genomslag och vi tvingas anpassa dem till dagens middleware-krav riskerar dessa system snabbt bli morgondagens legacy-system. Ges vi däremot möjlighet att modellera våra system i nära överensstämmelse med vår verksamhet och verksamhetens språkbruk så arbetar vi på en abstraktionsnivå som ökar chanserna till smidig följsamhet och till ökat oberoende från alla nya teknikgenerationer. Det finns många tecken som tyder på att sådana integrationsstöd inte bara hör hemma i paneldebatter utan också på ritborden.

Någon klok debattörs slutkläm om att man alltid ska vara försiktig med att påstå att man funnit ”the ultimate solution” får stå som slutkläm också för denna reserapport.