

OOPSLA '99 med



Bluefish

1-5 November, Colorado
Convention Center av Richard
Glanmark.

Efter mycket om och men an-
lände jag till Denver sent på
eftermiddagen. Otroligt nog så
träffar jag på mina gamla
kolleger ifrån Inspire ute på
gatan och vi gick snabbt till
krogn och åt några otroligt,
fast inte förvånande, stora ham-
burgare.

Förra årets OOPSLA handlade
våldigt mycket om *patterns*,
men i år fanns det inget ämne
som stod ut mer än något
annat. Det skulle möjligen vara
XML i så fall som togs upp i en
del tutorials och paneler.
Annars var det en blandning av
komponenter, *patterns*, Java
och en hel del annat.

Det här var andra gången jag
var på OOPSLA, men tyvärr så
tyckte jag nog att förra årets
upplaga i Vancouver var en
aningens bättre vad gällde ut-
budet, ämnena och kvalitén.

Som vanligt på OOPSLA så
finns det att välja på *tutorials*,
workshops, *demonstrations*,
panels, *technical papers*, *de-
sign fest* och mycket annat. För
övrigt så deltog ca 2 100 pers-
oner, att jämföra med ungefär
2 600 förra året.

Tutorials

Som vanligt varierar kvalitén
våldigt mycket på de tutorials
som gavs. Utav de fem halv-
dags-tutorials jag gick på var

tre bra och de övriga två så
pass dåliga att jag gick innan
föreläsningarna slutade.

Putting Metaclasses to Work

Ira R. Foreman, IBM

Foreman, som har jobbat på
IBMs SOM modell, var mycket
kunnig inom metaklassområdet
och gav en tutorial som var
både pedagogisk och intressant.

En metaklass är alltså ett klass-
objekts klass. Det vill säga, om
man även betraktar en klass
som ett objekt och eftersom
varje objekt måste ha en klass,
så innebär det alltså att klass-
objektet har en klass som kallas
för *metaklassen*. Begreppen
under föreläsningen kunde
minst sagt vara lite förvirrande
och svåra att hålla isär.

Foreman beskrev dels hur en
generell metaklassmodell där
varje klass kan explicit namnge
sin metaklass, kan implemen-
teras samt olika användnings-
områden för metaklasser.



Eric Gamma med Håkan
Klevfors

Ett mycket intressant område
är hur man kan använda meta-
klasser som egenskapsklasser.
Genom att kombinera "nor-
mala" klasser och metaklasser
kan man skapa egenskaper hos
den ursprungliga klassen. Ett
exempel är en metaklass, som
på ett enkelt sätt men inte
nödvändigtvis det mest
effektiva, gör alla klasser som
ärver från den trådsäkra. På så
sätt kan beroendena i ett klass-
bibliotek på ett enkelt sätt bli
linjära istället för exponentiel-
la, enligt Foreman.

Sammanfattningsvis sade Fore-
man att om de funktionella
språken gav oss *verb*, objekten

subjekt, så ger metaklasserna,
oss *adjektiv*. En intressant
tanke!

Software Building-The Road to a Software Architecture Worthy of the Name

Kevlin Henney, QA Training,
Alan O'Callaghan, De
Montfort University

Denna tutorial var kanske den
bästa jag var på. De båda före-
läsarna gav en mycket intres-
sant föreläsning som till en
början kändes mer som en in-
vited speaker istället för en
vanlig tutorial.

Hela deras tema gick ut på att
jämföra *software architecture*
med vanlig byggnadsarkitektur.
Enligt föreläsarna finns det inte
en definition på vad arkitektur
egentligen är. Utan istället är
det en mängd influenser och
känslor inblandat i definition-
en. Henney och O'Callaghan
menar att arkitektur bl a består
av *struktur*, *kommunikation*,
och *mening*.

Vad har en *modell* med arki-
tekturen att göra? D v s, en
skiss, ett UML-diagram i vårt
fall. Föreläsarna poängterade
det viktiga att inte blanda ihop
modellen med det verkliga sy-
stemet. En modell är, som sagt,
bara en modell.

Vad är skillnaden mellan *de-
sign* och arkitektur då? Design
är en medveten och avsiktlig
process medan arkitektur inte
alls behöver vara med flit. Alla
system har en arkitektur, oav-
sett om den var avsiktlig eller
ej.

Henney och O'Callaghan vi-
sade senare, med paralleller till
byggnadsarkitektur, vikten av
att bygga en sund arkitektur
m h a en rad tekniker. De
nämnde bland annat kompo-
nenter och *connectors* (inter-
faces), *patterns*, beroende-
hantering, flera lager o s v. En
intressant synpunkt vad be-



bluefish

träffar lager är att man normalt ser de som abstraktionsnivåer. Ett alternativ är att designa de olika lagerna efter hur lång tid de förändrar sig. Vidare menade de att arkitekturen påverkar även de sociala strukturerna. Beroende på hur man partitionerar system får det olika effekter i hur människor interagerar och arbetar.

Föreläsningen avslutade med att beskriva vikten av att försöka explicit uttrycka den arkitekturella stilen och att bevara den i ett system.

Vad hände på kvällarna?

Denver

I och med att vi bodde i downtown där nästan allt är nybyggt, fick man en känsla av att Denver är en prätigt och ren stad. Men om man tittar lite närmare ser man en stad med en lång gruvgrävarhistoria som börjar redan 1859 då guld hittades.



Denver skyline.

Ifrån hotellet kunde man se de snötäckta Rocky Mountains. Synd att man inte hann åka lite skidor! På kvällarna roade vi (jag och Inspire-människor) oss med sport som basket, amerikansk fotboll och ishockey. Självklart fanns det också en hel del ställen att gå ut på men det blev mest sportbarer och restauranger.

Denver kändes inte som världens klubbställe direkt, men en del av oss hann även med att kontrollera dem aspekterna också.

Speakers

Keynote Address

James Burke

På OOPSLA är keynote-talet en av höjdpunkterna. Oftast har man bjudit in en mycket betydelse- och inflytelsefull person och så även i år. Årets talare, James Burke, har medverkat på BBCTV *Tomorrow's World* och var BBCs chefsreporter för Apollo-uppdragen. Han har påverkat allmänhetens förståelse för teknik och vetenskap och blivit kallad för att tillhöra västvärldens intellektuella elit.

Burkes huvuddrag i denna keynote handlade mycket om innovation och hur vårt samhälle, genom tradition, är väldigt inordnat efter förutbestämda mönster. *Reductionism* var ett ord som Burke ofta använde sig av för att beskriva detta fenomen.

Talet var relativt akademiskt men underhållande och det märktes klart att det var från en mycket intellektuell person. Tyvärr måste jag säga att jag inte följde med hela tiden. Jag tror att det berodde en hel del på att jag antecknade på en Toshiba Libretto, istället för papper och penna. Mitt i talet kraschade dessutom datorn med följd av att jag tappade koncentrationen ett tag. Eller så är jag helt enkelt inte så smart som jag har trott.

I vilket fall som helst, innovation, enligt Burke, är en process som kan liknas med att summera ett och ett och få resultatet tre, d v s $1 + 1 = 3$. För att åstadkomma detta måste man oftast gå utanför de traditionella synsätten och metoderna. Burke berättade humoristiskt om hur diverse innovationer har kommit till, utifrån helt till synes slumpmässiga samband och händelser. Till exempel, att IBM skapades utifrån de problem som fransk silkesvävning hade,

menade Burke, fast kanske inte helt seriöst.

Vidare så påpekade han hur marknaden hela tiden har drivit utvecklingen, d v s behoven hos mänskligheten har skapat innovation. Marknaden kräver helt enkelt att institutionerna, som är mycket konservativa, uppfinnar sig själva om och om igen.

Nu när internet har kommit, så finns en otrolig mängd information tillgänglig för alla. Information som tidigare bara varit tillgängligt för en intellektuell elit. Burke menar att samhället idag kanske har kommit fram till en *post reductionism* där utvecklingen inte längre är begränsad av de traditionella barriärerna. Till sist föreslog James Burke att det kanske vore nyttigt för alla att leka en timme då och då på jobbet för att förbättra kreativiteten.

Invited Talk - Fun With Squeak and Other Smalltalks

Dan Ingalls

Dan Ingalls, en av Smalltalks skapare, gav här en rolig demonstration av Squeak. Squeak är en fri distribuerad (mjukvaru) låda med roliga, underhållande, och användbara verktyg och tekniker. Squeak är skrivit i Smalltalk och är en extremt dynamisk miljö som kan köras på de flesta plattformar.

Det är väldigt svårt att försöka beskriva vad Squeak egentligen är och vad det kan göra. Men tänk dig ett extremt interaktivt system där man mer eller mindre kan inspektera, modifiera och interagera med alla objekt på skärmen;. Ett system där ljud och grafik är lika enkelt att manipulera som att skriva ett textdokument. I och med att Squeak är skrivit i Smalltalk så kan man självklart ta upp den underliggande



koden för varje objekt och leka runt med.

Jag kan verkligen rekommendera att titta närmare på den här supercoolgrejen på <http://www.squeak.org>.

Paneler

Paneler på OOPSLA är ett slags diskussionsforum där intressanta personer är inbjudna att diskutera aktuella ämnen. Publiken får sedan oftast ställa frågor till panelen eller göra egna inlägg. I år gick jag kanske på lite väl många paneler istället för technical papers som är lite mer tekniska.

Ubiquitous Computing – What’s It Good For? Eller PC:n är död!

Denna var i särklass den mest visionära och futuristiska panelen, och även kanske den mest underhållande, som jag var på. Ubiquitous Computing (data överallt) handlar om potentiellt små dataenheter som finns överallt; i sensorer, telefoner, kaffebryggare, dörrar, konferensrum o s v.

Michael Gorlick, från The Aerospace Corporation, var den mest visionära. Han menade att i framtiden kommer det att finnas hundra- eller kanske tusentals små enheter, i väggar, dörrar, i byxbältet, eller som bara flyter omkring i luften och som ger tillgång till rätt information och precis när man behöver den. Bara fantasin sätter stopp! För att klara hantera alla dessa enheter behövs nya grepp som, enligt Gorlick, kommer att komma ifrån spelindustrin med dess förståelse för kognitiv modellering. Vidare menade han att alla dessa små intelligenta enheter kommer att ersätta den då uråldriga och klumpiga PC:n!

Is UML Also an Architectural Description Language

Anledningen varför jag gick på den här tutorialen var nog för att jag ville lyssna på några berömdheter som Grady Booch, Rational Software, David Garlan, CMU, och Desmond D’Souza, Platinum Technology.

Alla, förutom Grady Booch, tyckte att UML inte är ett optimalt språk för att beskriva en arkitektur. Med det menade de att i en ADL vill man kunna lyfta ut och understryka de intressanta och viktiga aspekterna i ett system som t ex connectors, interfaces, architectural styles och patterns. UML kan visst uttrycka dessa saker, men på många olika sätt och kanske inte alltid tydligt. Dock menade Desmond D’Souza att med UML så är vi ett steg närmare och ville gärna se ett UML som är likt XML, d v s utökbart.

Are Components Objects?

En relativt menlös diskussion i den meningen att det idag inte finns någon konsensus om vad en komponent egentligen är för någonting. Men:

- *Clemens Szyperski*, ifrån *Microsoft*, gav en ganska tung men luddig definition som (luktade Microsofts COM-modell) som gick ut på att det är skilda saker, ungefär som klasser och instanser.
- *Alan Wills*, *TriReme International*, hade en mer pragmatisk inställning där varken definitionen av en komponent eller om en komponent är ett objekt är viktigt. Utan han menade att komponenter är på det sätt som vi borde byggt mjukvara från början, d v s med väl-specificerade gränssnitt och utbarhet.

XML and Object Technology

Kommer XML att ersätta objektteknologi var något som panelen försköte besvara. Nej kom de fram till, dock med lite olika infallsvinklar.

XML kommer snarare att komplettera objektteknologi som t ex CORBA, men också överlappa och kanske ersätta tekniker inom andra områden och tillämpningar. Speciellt attraktivt är XML, kom panelen fram till, som protokoll mellan system som från början aldrig var menat att kommunicera med andra system, s k legacy system.

För vissa teknologier, som EDS till exempel, är det till och med troligt att XML kommer att ta över. Däremot kommer nog även XML få brottas med vissa standardisering gällande DTD-er m m.

Intressant var också att höra att medan vissa såg hur man skulle kunna generera XML och kod utifrån objektmodeller i t ex UML, så fanns det vissa som *inte* ville ha en hårdare integration mellan OO och XML.

Hur var det att använda Libretton?

Nja, si sådär. Den största fördelen är att den är så lätt att ta med sig. Men storleken innebär också lite problem med att kunna skriva på den. Den största nackdelen är livslängden på batterierna, ca en och en halv timme per batteri. Summa summarum? Okej då, godkänt!

Vi syns nästa år,
Richard.