Java Enterprise

Innehåll

1 Snabbfakta

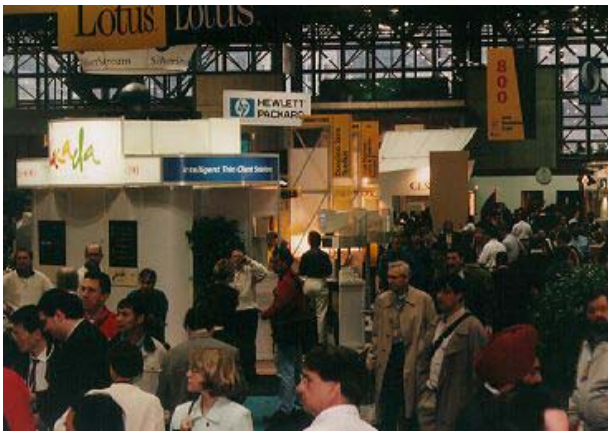
Denna konferens genomfördes för första gången förra sommaren (1997). Stor framgång. De ca 15.000 som då kom var betydligt fler än man hoppats på. I år skulle denna siffra slås med bred marginal, var det tänkt. Av detta blev dock intet. Svårt att uppskatta, men inte kom det mer än 10.000 i år. Varför kan man undra. Java är populärare än någonsin. Affärsgenomgripande arkitekturer, global samverkan, avancerade infrastrukturer. JavaOne samlar årligen en stor skara entusiaster. Skillnaden ligger kanske i att dessa är programmerare och att JavaOne vänder sig precis till dem. Kontakt mellan utbud/förväntan och presumtivt intresserad är upprättad.

Java Business Expo vänder sig knappast till Java-programmeraren, samtidigt som budskapet är för abstrakt för beslutsfattaren, strategen, den som förhoppningsvis är målgrupp här. Visserligen finns det en allt större skara intresserade men budskapet måste rikta sig speciellt till dem annars placeras inte konferensen tillräckligt högt i kön. Denna senare kategori vill veta att den spenderade tiden känns nyttig, ger lösningar på de egna aktuella problemen snarare än en intressant diskussion kring dem.

Kanske är en konferens mitt i julhandeln ingen bra kombination. New York lockar, men av andra saker än IT!

I alla händelser var det en seriös konferens med ett antal parallella sessioner. Där fanns alltid något intressant för varje deltagare vid varje tidpunkt.

I utställningshallen trängdes ca 250 utställare, alla med någon koppling till Java. En vild flora av services och produkter erbjöds t ex application servers, middleware, databaser, utvecklingsverktyg, ramverk, klassbibliotek, böcker, tidskriftsprenumerationer, kurser, inkapsling av legacy system, …..



Lite av varje

Uppsnappad slogan: *”Know more, no less”*.

Travestering på ”Write once, run anywhere”: *”Rewrite constantly, debug everywhere”* (uttryckt av besviken programmerare).

SUN kontrar med: *”Java technology – the way things work”*.

Begreppet att röra sig med i år: *Application Server*.

Ofta hört: *Mission critical, n-tier*.

Förlegat: Open systems. Istället: *Open services.*

2 Keynote-presentationer

Fyra så kallade keynote-presentationer levererades av höga representanter för företag med nära Java-koppling. De representerar knappast en neutral position inom de teman de tar upp, vilket inte heller är meningen. De ska stå för entusiasm, seriositet, vision. De ska bibringa deltagaren känslan av att befinna sig mitt i ett alldeles självklart framgångsrikt kraftfält. Nå, vad hade de då att säga?

2.1 Zander, SUN

”Java scares the hell out of me”, lär Bill Gates ha sagt noterade Zander med inte bara en gnutta förnöjelse och fortsatte: 1980-talet var ”desktop centric computing”, med PCn som fanbärare. 1990-talet kom att kretsa kring SUNs slogan ”the network is the computer” med klient/server och 3-skiktsarkitektur som varumärken. 2000-talets första årtionde kommer att kretsa kring det SUN nu i intensiv TV-reklam för fram som ”.COM” eller uttalat ”what can SUN dot com for you”. Det är dags att skapa mervärde för var och en, för hemmet, för skolan, för företaget. För att understryka det senare plockade han fram en annan av SUNs slogans ”the network is the business” för att avrunda med ”the network is everywhere”. På ett alldeles påtagligt sätt stämmer tankegångarna överens med vad som i stora bokstäver fanns att skåda i SUNs monter på utställningsgolvet, nämligen ”anyone, anytime, anywhere, any device”. Beviset för denna trend ser Zander i vilka företag som är hetast idag. Det är inte längre de konventionella hård/mjukvaruföretagen utan de som förstått att utnyttja teknik för att generera mervärde – tjänster, alltså vad som går under beteckningen ”service providers”. Dit hör rese-, finans-, konsumenthandels-, underhållnings-, sök- och många andra tjänster. Nya tjänsteidéer kommer att springa upp som svampar ur jorden, inte minst när bandvidden ökar och när de mjukvarumässiga infrastrukturerna är på plats. Plus när Java används överallt förstås.

Inte utan stolthet radade sedan Zander upp en imponerande uppsättning Java-statistik:

- Det finns nästan 1 miljon Java-programmerare idag (källa Business Week)
- Man passerar C++ i användning ca år 2000 enligt IDC
- 195 företag har hittills licensierat Java

- 1300 Java-implementerade tillämpningar säljs kommersiellt
- 4 miljoner nerladdningar av JDK 1.1 har genomförts
- 1000 universitet undervisar i Java (tänk när deras studerande sprids i industrin tillade Zander med en drömmande blick)
- 1224 böcker med Java-teman finns att köpa på en uppsjö språk.

Även om man tar statistiken med en nypa salt är den onekligen imponerande, inte minst med tanke på att det hela skett på mindre än fyra år.

Enligt Zander finns det idag ca 200 miljoner persondatorer som potentiella Java-nyttjare. Men än viktigare är konstaterandet att det om ett par tre år kommer att finnas ca 6 miljarder konsumentprodukter med sådan inbyggd datorkraft som lämpar sig för Java Virtual Machine (JVM). Vi kan knappast ana de tjänster som kommer att springa fram ur dessa förutsättningar. Smart Cards bedöms ha en lysande framtid, för övrigt ett område där Europa har en hel del att lära USA. I Europa har vi sedan länge haft telefonkort, olika typer av bankkort, identifieringskort, kontantkort, mm något som först på senare tid anammats på bred front i USA. SUN tillämpar internt själva principen ”Smart Card is my computer” på så sätt att man med hjälp av sitt eget kort kan gå till vilken dator som helst och där ladda ner det egna ”skrivbordet”. I princip räcker det med att datorn har en bläddrare med en JVM. Även service kan underlättas genom att den som har problem loggar in på SUNs hemsida och talar om vad som är bekymret. SUN diagnosticerar därefter genom att ta direktkontakt med datorn.

Varpå Zander avslutade med att annonsera två nyheter; Java 2 och SUN Community Source licensing. Se separata avsnitt nedan.

Vid en efterföljande presskonferens uttryckte Zander en förhoppning om att Microsoft anpassar sig till den nya licensieringsprincipen. Alla övriga 194 licenstagare tycks beredda att göra det. Den nya öppnare principen kring källkodshanteringen innebär en risk eftersom koden ”sprids för vinden” och man inte vet i vilka händer den hamnar. Dock är man övertygad om att de aktuella licenstagarna svarar upp mot erforderliga kompatibilitetskrav (klarade tester). Man ser gärna att licenstagare med nischintressen samlas för att utveckla egna APIer för dessa, speciellt inom områden där SUN inte själva har egen expertis. På så vis skapas en högre utväxling. Dock förutsatt att de svarar upp mot den stipulerade process SUN definierat.

Man kan dock undra hur SUN ska kunna hävda sin rätt på en totalt internationell marknad med inte alltid nogräknade kommersiella aktörer, dessutom agerande i länder vars högsta prioritet inte är att tillgodose SUNs intressen. Att överföra Java-plattformen till ett standardiseringsorgan tror inte SUN på och då inte primärt av kommersiellt egenintresse. Standardiseringsorgan arbetar för långsamt, ofta heller inte med erforderligt fokus. Alltför mycket kompromissande skapar lätt urvattnade resultat ingen egentligen vill ha. Snabbhet är a och o för Javas framgång. Det gäller att både hålla och vidareutveckla det momentum som nu finns.

2.2 Sueltz, IBM

Pat Sueltz är chef för IBMs område för Java-mjukvaror – en stab om ca 3000 personer. En stor del av presentationen kom också att handla om Java-plattformen och dess nu mycket stabila egenskaper. Plattformen är redo för avancerade tillämpningar. Java ”has moved from client to server or to the middle of the business”. Inte minst dess basering på öppen teknologi gör plattformen mycket lämpad för t ex e-commerce och global business systems, d v s det IBM nu för fram under begreppet ”e-business” uttolkat som ”web-enabling core business”.

För att riktigt poängtera dess breda acceptans och vitala vikt för framtida system konstaterade Sueltz att ”Java technology is bigger than IBM, it´s bigger than SUN, it´s bigger than Microsoft, it’s bigger than any of us”. Vilket samtidigt är något av en förutsättning för dess fortsatta framgång. IBM ser med tillförsikt an Javas fortsatta expansion och konstaterar att ”Java technology has gone mission-critical in a big way”. Ett bevis är IBMs intensiva investering inom Java-området. Ett annat den tunga satsning man i Japan gör på Java i konsumentprodukter.

IBM har att svara upp mot en mycket stor existerande kundkrets. Dessa förlitar sig för närvarande (och sannolikt under överskådlig framtid) i stor utsträckning på existerande system. De har utrustats under lång tid med ett brett spektrum av IBM-teknik, olika plattformar, mm. De kräver en realiseringsmiljö som kan införliva dessa, ”wrap them”, accepterar deras existens, samtidigt som den klarar av de utökade integrerings- och eller samverkanskraven med andra system som verksamheter alltmer intensivt kräver. Kraven gäller både mellan system internt och externt. Java-plattformen med sin vision om en öppen, samverkande, teknikneutral miljö är i det perspektivet som ”klippt och skuret” för IBM att erbjuda sina kunder. Enleverantörslösningar är oacceptabla och kommer att så bli i än större utsträckning i framtiden.

Som ett tillämpningsexempel presenterades en video över ett helt Java-baserat försäljnings-leverans- och servicesystem för SAAB, USA kallat IRIS. SAAB passade i sammanhanget på att förevisa både system och en häftig SAAB Cabriolet-bil i utställningshallen. Den senare lottades för övrigt ut sista dagen till en säkerligen lycklig mässbesökare. Andrapris var en resa till Sverige.

Sueltz pekade även på XML som en mycket vital komponent i framtida systembyggande. ”XML makes online information smarter”. ”XML is doing for data what Java is doing for code”. Det står alldeles klart att IBM satsar hårt på produkter och tjänster baserade på en kombination av Java och XML, en kombination av process och data med betoning på öppenhet, standard, samverkan och mobilitet. Kanske något att tänka på och ta efter även för andra företag!?

Sueltz avslutade med att gratulera SUN till releasen av Java 2. Den har varit efterlängtad länge. Samtidigt är releasen säkerligen en (tillfällig?) pust av lättnad för det hårt Java-satsande IBM och en uppmuntran för dess kunder.

2.3 Mc Nealy, SUN

I vanlig ordning inledde SUNs President Scott McNealy med ett antal lustigheter. Med anledning av den närstående julen blev det bland annat ett tiotal julklappar som man inte skulle vilja ha under granen. Ett par av dem:

> Ett paket som det står 'Love Monica' på (med syftning på de då aktuella turerna kring president Clinton)
>
> En bok av Bill Gates med titeln 'The e-mails i don't remember' (med syftning på då aktuell hearing)

McNealy noterade med välbehag det nyligen ingångna samarbetet mellan AOL, Netscape och SUN. Här uppstår en integration mellan innehåll, dess hantering, dess distribution och erforderlig avancerad teknik – med fördel levererad av SUN. Under ytan svävar Java som möjliggörare.

När det gäller Java konstaterade McNealy att nu endast HotSpot (se vidare under avsnitt Java2, prestanda) återstår för att Java-plattformen ska kännas fullödig.

Nästa stora lyft blir JINI, en produkt som snart kommer att releasas. Den är baserad på en idéskiss av Bill Joy, en av grundarna till SUN och nu en fritänkande vice president hos SUN. Syftet med JINI, som givetvis är Java-baserad, är att förenkla kommunikation och dela på service mellan allt från avancerade system till enkla konsumentprodukter. Man pluggar in utrustningen på nätet och får direkt tillgång till dess användargränssnitt, dess i varje läge tillgängliga funktionalitet. Se vidare under Jini-avsnittet nedan. Att det hela fungerade framgick av en demo. Man utgick från ett nätverk med endast en uppkopplad dator. Datorns bläddrare visade information om nättillstånd. Därefter pluggades en kamera till nätet varpå dess existens genast registrerades i bläddrarens sida. Ett klick på dess symbol och kamerans tillgängliga användargränssnitt presenterades. Så klick på dess "ta kort"-symbol. Kortet tas. En skrivare pluggas till nätet. Med klick på "skriv ut"-symbolen initieras kontakt mellan kameran och skrivaren varefter bilden skrivs ut på skrivaren. Man pluggade till sist in en fläkt och kunde enkelt via datorn slå på och av den. Givetvis är varje enhet bestyckad med en JVM och en JINI-agent.

McNealy konstaterade förnöjsamt att alla bekymmer med installation av utrustning som vi idag brottas med snart är ett minne blott. JINI förväntas få en mycket snabb och bred acceptans.

Överhuvudtaget kommer vi att se en explosion av processer i alla upptänkliga produkter, inte minst i konsumentvaror. 1997 fanns det redan 2.9 miljarder processorer men bara ca 70 miljoner PC-datorer. Dessa processorer vill kunna samverka. År 2002 räknar t ex IDC med att över 50% av alla Internet-accesser sker från annan utrustning än PC.

För övrigt önskade sig McNealy att i stort sett allting framöver skulle ha en smartkortläsare. En läsare kostar idag under en dollar. Tänk vilket smidigt sätt att identifiera, behörighets-kontrollera, m m för tusen och en olika ändamål. Även för att ladda ner sitt eget ”skrivbord” på vilken dator som helst – under förutsättning att läsare finns på den och att den är Java-bestyckad förstås. Samma vision har McNealy framfört vid ett antal tidigare framträdanden, d v s att kortet i princip är datorn. Som även Zander nämnde tillämpar SUN själva filosofin internt. Den anställde blir oberoende av tid och rum. Finns bara någon dator i närheten, t ex till låns på hotellrummet, är det bara att sätta igång och jobba.

McNealy konstaterade att det just nu exploderar av nya web-företag i spåren av e-commerce. ”Economy-of-scale” kommer snart att pressa fram utslagningar och samman-slagningar. Därtill kommer helt nya, öppna konkurrensförutsättningar. Kunden får ökad makt, ökad valfrihet. Vinnare blir större nätbaserade service providers. På sikt kommer dessa genom att vara konkurrensutsatta att erbjuda den service, den funktionalitet vi behöver. Makten över till användaren. En direkt effekt blir att företagen i betydligt mindre utsträckning köper egna datorer eller utvecklar egna tillämpningar. En förutsättning är förstås nätkapacitet, Quality of Service och upparbetad tillit.

2.4 Russell, HP

William V. (Bill) Russell är vice president på Hewlett-Packard och general manager för Computer Organization's Enterprise Systems Group. Russell diskuterade bland annat visioner i anslutning till SAABs IRIS-tillämpning. Vi kommer snart att ha bilar som löpande skickar statusinformation till någon tjänst som bedömer behov av service samt lagrar informationen i en databas för senare behov. Behövs servicen kontaktas bilen med budskapet att service behövs och var lämpligaste kapacitet i anslutning till bilen normala färdrutter finns till hands. Ligger felet i bilens datorer eller integrerade nät kan förmodligen korrektionen utföras direkt över det trådlösa nätet. Vid krock ringer bilen direkt upp larmtjänst med uppgift om omfattning och förmodade skador hos resenärer samt givetvis aktuell koordinatangivelse och vilka resenärerna är. (Noterade när de satte sig i bilen i kommunikation mellan bilens system och respektive resenärs individsystem bestående av ett antal samverkande processer för olika behov. Självfallet kontrollerar bilens körkortskontroll samtidigt om föraren har körkort och är nykter samt om så inte är fallet meddelar tändsystemet att inte reagera.) För säkerhets skull tar larmtjänsten kontakt med respektive resenärs vidhängande (”inbyggda”??) hälsomonitor för att kontrollera hjärtfrekvens, blodflöde mm samt om något inte är som det ska meddelar sjukhuset som på basis av erhållen information kan förbereda mottagande alternativt initiera distansinsatser, t ex i form av goda råd till respektive resenär samt vid behov insatser som andra resenärer kan utföra mot någon medpassagerare. Larmtjänsten är för övrigt en typisk service provider.

Dock kan man undra över hur lång tid det tar innan vi människor tillåter denna automatik att utföras delvis ”över våra huvuden”. När och i vilken grad vågar vi känna tillit?

Det är ingen hejd på de tjänster som kan skapas baserade på verkliga eller främmande behov. Tekniken kommer knappast att vara intressant länge till. Jmf elektricitet. Inte intressant som teknik utan i kombination med behovet av glödlampor. Vi kommer att förutsätta och kräva att den underliggande tekniken fungerar. Vilket den också gör i andra sammanhang. Det är inte ofta vi får el- eller teleavbrott. Availability, Quality of Service, Security och Scalability blir nyckelord. Att ha tillgång till och kontroll över datorutrustning

och programvara kommer att bli betydligt mindre intressant. Vad vi önskar och kommer att få är tillgång till (konkurrerande) högkvalitativ service över webben. Vi hyr, lånar den kapacitet vi behöver för den tid vi behöver den, alternativt betalar per gång vi använder oss av servicen. Hela idékonceptet går från en egendomsmodell till en nyttjandemodell. Jämför med det ”normala” samhället. Det är inte många som har sin egen vattenbrunn och avloppsanläggning eller sin egen elgenerator. Vi abonnerar, betalar för graden av nyttjande.

En intressant följdeffekt blir givetvis ett minskat behov av att utveckla egna tillämpningar. Man betalar en service provider för den erforderliga funktionaliteten/servicen istället. Som ett exempel nämnde för övrigt McNealy företaget Granger, en service provider som specialiserat sig på att hålla uppdaterad information för varje abonnerande företag över de produkter och leverantörer som företaget godkänt för köp från. De ombesörjer sedan erforderlig service för att reglera alla steg i ett köp/hyra. Företaget behöver varken skriva motsvarande tillämpning, hyra en köpagent, lagra produkten eller ombesörja leverans till alla som behöver den inom företaget.

3 Nya licensregler

SUN har uppenbarligen tagit intryck av den kritik som riktats mot företaget när det gäller hanteringen av rättigheter, kontroll och ekonomi kring Java. Man har av den anledningen formulerat en betydligt mjukare, öppnare licensieringsprincip under beteckningen ”SUN Community Source License”. Översiktligt gäller följande nya regler:

- Vem som helst kan ladda ner Java 2 i källkod. Därmed blir man i princip licenstagare med automatik.
- Man får modifiera koden bäst man vill samt skicka den vidare till vem man vill.
- Man behöver inte som förr skicka in sina ändringar till SUN. Detta var tidigare en skarp kritik från tunga licenstagare som menade att ändringarna är en typ av företagshemlighet man inte vill och inte heller borde behöva yppa för SUN. Det kan ju gälla tillämpnings-områden av Java som SUN inte behärskar och därför inte borde ha något intresse i annat än av ren nyfikenhet eller för att hålla full kontroll över Javas öde. Än värre, det kan ju vara områden där företaget befinner sig i en intensiv konkurrenssituation med SUN.
- Licenstagare får packa SUNs klassbibliotek med JVMs från andra licenstagare.
- De som önskar ta fram nya APIer för specifika behov för senare införande som Standard Extensions må göra detta efter eget huvud (men efter SUNs initiala godkännande). Dock under förutsättning att de tillämpar den av SUN definierade öppna framtagningsprocessen. Processen från idé till färdig specifikation är väl definierad. För att ge processen en extra dos seriositet och för att få den att upplevas som ”nästan” en officiell standardiseringsprocess, har SUN anlitat Price-Waterhouse som övervakare av att den i varje enskilt fall genomlöps enligt stipulerade krav. Bland aktuella extensions kan nämnas Java 3D, Java Naming and Directory Interface (JNDI), Java Servlet, JavaMail och Java Media Framework (JMF).

  Implementeringar måste därutöver givetvis klara den nu mycket omfattande testsviten (över 10.000 testmoment) som en kontroll av att kompatibilitetskraven är uppfyllda – en förutsättning för garanterad flyttbarhet av koden. Därefter får den rykande kaffe-koppen placeras på produktens binärkod. Försäljningen alternativt spridningen i den egna organisationen kan starta. Först nu börjar SUNs royalty att ”ticka in”. Royaltyns storlek har ungefär samma uppbyggnad som tidigare, d v s regleras mellan respektive licenstagare och SUN med hänsyn till ett antal faktorer i det enskilda fallet.
- SUN ser gärna att även andra än licenstagare medverkar i framtagningsprocessen av ny Java-teknologi. Man hoppas kunna intressera industriexperter, forskare, standardiserings-organ, m fl för att på så vis skapa förutsättningar för än bättre kvalitet i resulterande specifikationer – för att ge processen en dignitet högre seriositet. Givetvis önskar man att detta ska bana vägen för en ytterligare acceptans och spridning av Java-teknologin.

Kritik har riktats mot vad man anser vara SUNs alltför stora inflytande över ett område som i dagsläget drivs fram av en stor mängd tunga aktörer, knappast bara av SUN. Flera av dessa har dessutom lagt hela sin affärsidé, satsat hela sin framtid på Javas framgång. Att låta den fortsatta Java-utvecklingen bedrivas i en mer neutral standardiseringsinstans kan i det läget bedömas som välmotiverat. SUN replikerar att sådana instanser ytterst sällan är tillräckligt effektiva och tillräckligt snabba. Java skulle snabbt tappa sitt momentum, till förfång för alla. SUN anser sig ha gått kritiken till mötes så långt det är rimligt möjligt i och med de nya förutsättningarna – utan att sätta Javas framtid på spel.

Genom denna öppnare filosofi hoppas SUN att licenstagare ska känna sig stimulerade att utveckla Standard Extensions med stor intensitet, allt för att bredda och fördjupa Javas applicerbarhet till varje hörn av IT-området.

4 Java-plattformen

4.1 Java2

I anslutning till konferensen annonserade SUN under stor pompa sin release av **"Java2 platform"**. Den har som beta release gått under beteckningen JDK 1.2. Genom att ge releasen nummer 2 vill man understryka att det nu är fråga om en komplett plattform för att bygga fullödiga, distribuerade, verksamhetskritiska tillämpningar. Releasen är väntad och efterlängtad eftersom den rensar i den snårskog av olika, delvis inkompatibla versioner Java-utvecklare hittills har haft att brottas med. SUN kommer, enligt egen utsago, att hädanefter presentera nya releaser och versioner i ett betydligt lugnare tempo. Java 2 har därutöver kompletterats med en plug in-facilitet, ”Java Extensions Framework” (se under nåbarhet nedan), som tillåter enkel uppgradering med nya komponenter. Att releasen dröjt längre än förutspått beror enligt SUN inte på att man önskat inkludera nya saker i ”elfte timmen” utan på att man inte ville kompromissa med kvaliteten – Java 2 skulle vara ”ready for prime time” innan det släpptes.

Java 2 är en som Zander uttryckte det ”complete rewrite” av tidigare versioner. I princip är det JDK 1.2 med prioritering på prestanda, säkerhet, fullt plattformsoberoende och stabilitet. Det enda som ännu saknas är den tidigare annonserade Hotspot JVM som ännu bara finns i en betaversion.

Java 2 Plattformen är ett ca 20 megabyte stort paket som kan laddas ner på webben. Till detta kommer ytterligare en mängd megabytes för all dokumentation. I huvudsak innehåller paketet följande tre delar:

- Java 2 Runtime Environment (JRE) som består av en bytekodsverifierare som utför en integritets- och säkerhetskoll av koden, en ”class loader” som dynamiskt laddar in klasser under exekvering och Java Virtual Machine (JVM) som tolkar och exekverar koden i samklang med aktuellt operativsystem. Det är med andra ord JRE som kapslar in variationerna i de olika underliggande operativsystemen så att utvecklaren kan realisera sin funktionalitet helt hårdvaru/OS-plattformsoberoende. Än så länge finns JRE för Solaris, Windows NT Server version 4.0 och Windows 95/98. För de flesta andra operativsystem pågår arbete med anpassning.

- Diverse verktyg som editor, kompilator, debugger och dokumentationsstöd.

- En omfattande uppsättning Java bibliotek och så kallade Java Foundation Classes (JFC) där JVC omfattar den funktionalitet som bedömts vara angelägen för de allra flesta tillämpningar och därmed gjorts obligatorisk av kompatibilitetsskäl.

Java 2 plattformen är primärt avsett för generella dator/OS-miljöer, inte minst den allt intressantare server-sidan. För speciella behov finns PersonalJava, EmbeddedJava och Java Card. Se vidare under http://java.sun.com och http://www.sun.com/java.

Annonseringen av Java 2 plattformen skedde med dunder och brak mitt på Manhattan i en av skyskraporna. Först presentation i en stor aula under jord. Därefter hisstransport till 50:e våningens festavdelning med utsikt över hela stan, vickning, levande musik, mm.

Presentationen gavs av Dr Alan Baratz, chefen för Java Software hos SUN. Framförallt noterade han följande:

A Komplett och homogent

- Inga olika versioner, olika utvecklingssteg för APIer. Bara en homogen version.

- Utökad JVC (Java Foundation Classes):

    - Java 2D API, för avancerad 2-D hantering. Innehåller också utskriftsfaciliteter. Detta API har länge varit starkt efterfrågat.

    - Accessability API. Ett API för användare med olika typer av handikapp, t ex talande skärmläsare, taligenkänning, skärmförstoring .

    - Swing set. Består av en uppsättning JavaBeans för drag-and-drop, tool bars, menyer, m.m, funktionalitet som tidigare saknats eller realiserats på olika sätt, anpassat för varje plattform.

    - Ett Java look-and-feel gränssnitt som kan anpassas.

    - Samverkan med icke-Javasystem kan nu genomföras med hjälp av ett CORBA-gränssnitt, något som kraftigt utökar Java-plattformens kontaktyta mot verksamheters totala tillämpningssfär. Genom CORBA API och en inbyggd Java Object Request Broker kan CORBA-objekt anropas direkt från Javakod. Dessa objekt kan vara implementerade i valfritt programmeringsspråk.

- Globaliserirad anpassning. Java 2 stödjer inmatning av japanska, kinesiska och koreanska symboler. Med hjälp av Java 2D API stöds presentation av symboler i ett flertal språk, t ex japanska, arabiska och hebreiska.

- Java 2 är kompatibelt med tidigare versioner vilket underlättar införande och anpassning till Java 2.

B Stabilitet

Java 2 är stabilt eller som Baratz uttryckte det ”rock solid”. Det har stressats i alla upptänkliga tester. Releasen har låtit vänta på sig. Dock har man haft som princip att inte släppa den förrän den känns övertygande stabil och komplett.

C Säkerhet

Den tidigare ”sandbox”-principen har ersatts med en betydligt flexiblare. Man har övergivit principen om full och automatisk säkerhet. Denna fungerade utmärkt så länge man bara kände behov av applets i form av enklare algoritmer, lustiga figurer som rörde sig på skärmen och annat liknande som aldrig behövde referera utanför sin ”sandlåda”. Med den utökade ambitionen att göra Java till en miljö för affärskritiska tillämpningar behövdes betydligt ökad flexibilitet för de utökade roller applets kunde förväntas komma att få. Vissa kan behöva ta kontakt med klientens filsystem, andra kunna operera i den lokala databasen eller initiera utskrifter på den lokala skrivaren, ytterligare andra ta kontakt med andra distribuerade applets utan att behöva gå genom webservern, o s v. Givetvis kan inte denna nya fria värld tillåtas utan kontroll. Ansvaret läggs över på utvecklaren och på systemadministratören. Dessa kan nu reglera *vad* som ska tillåtas och inte tillåtas genom angivande av policies, permissions och access control per tillämpning samt för *vilka* det ska gälla med referens till users, groups, program, OS services, objects och components. Med andra ord allt ifrån full tillgänglighet till ren sandbox-restriktion.

D Prestanda

Prestanda är ett ständigt problem, inte minst vid interpreterande exekvering. SUN har också bedömt prestandahöjning vara högprioriterat. Man har gjort en genomgripande översyn och enligt egen utsago trimmat varje hörn av systemet. Framförallt har garbage collection, minnestilldelning och Just-in-time (JIT) kompilering effektiviserats rejält. Denna JVM går under beteckningen Classic för att skilja den från HotSpot, den ”turbo” JVM som det länge tisslats och tasslats kring. HotSpot är ännu inte inkluderad i Java 2. Den är i dagsläget (december 1998) färdig för beta-release. Officiell release är planerad till april 1999. HotSpot förutspås erbjuda ytterligare prestandaförbättringar.

Trots alla markanta förbättringar kommer med största sannolikhet prestanda att alltid vara ett bekymmer för vissa kategorier tillämpningar. Vi får inte glömma vad som historiskt alltid gällt inom IT: Varje tekniskt framsteg har ”slukats” av nya behov, nya typer av tillämpningar, ökad användning. Trenden är alltmer komplexa tillämpningar, intensivt ökad och komplex samverkan mellan tillämpningars komponenter, o s v. Vårt sätt att hantera denna komplexitet är att bygga abstraktionsnivåer som efter bästa förmåga döljer komplexitet och renodlar de modeller vi har att operera med på varje nivå. Självfallet tar varje nivå kraft av systemet. Middleware-nivån är inte gratis (med nödvändig). Enterprise JavaBeans t ex är som abstraktion (se vidare nedan) inte heller gratis men den är generell och erbjuder attraktiv service. Vi har knappast nått den sista abstraktionsnivån. Håll dock i minnet att detta är ett generellt bekymmer, knappast ett specifikt Java-problem.

E Nåbarhet

Både källkod och binärkod finns nu fritt tillgängliga.

Java 2 är säkerligen mycket efterlängtad och välkommen av alla hårt luttrade Java-utvecklare som tidigare kontinuerligt haft att anpassa sig efter en ström av nya eller modifierade APIer, icke-kompatibla implementeringar, osv. Initialt hade JDK 1995 ca 200 APIer. Java 2 sägs ha ca 1600 specificerade APIer. Java har med andra ord varit ett i högsta grad ”rörligt mål” under sin knappt fyraåriga levnad. 15 klassbibliotek ingår nu i ”core”, den minimi-uppsättning som kan förväntas finnas på varje Java-plattform. Dessa är: applet, awt, beans, io, lang, math, net, rmi, security, sql, text, util, accessability, swing, corba.

Framtida anpassningar och uppdateringar underlättas genom ”Java Extensions Framework” som formaliserar hur nya/ändrade faciliteter dynamiskt kan laddas ner till varje Java 2-plattform och där omedelbart integreras med det övriga. På så vis kan en uppdatering spridas till alla platser momentant. Ingen väntan på uppdateringar efter respektive plattformsleverantörs tidtabell, ingen risk för inkompatibla versioner. Kärnan kan göras minimal och kompletteras efter behov med exakt bara det som behövs snarare än att levereras med en full uppsättning funktionalitet av vilken kanske bara en liten del blir använd.

Java Plug-in, tidigare under beteckningen Project Java Activator, är en ny ”plug-in virtual machine architecture” som dynamiskt uppdaterar en bläddrares Java Runtime Environment (SUNs) så att den klarar av att exekvera alla web-sidor oavsett vilken release web-sidans applets utvecklats under. På så vis kommer exempelvis HotSpot, när det väl releasas, att enkelt kunna pluggas in samtidigt i alla miljöer som önskar tillgång till den. Java Plug-in är en i grunden ganska enkel idé som mycket finurligt råder bot på inkompatibilitetsproblem mellan olika bläddrare. Java Plug-in erbjuder ett steg närmare ”write once, run anywhere”

Sammantaget innebär Java 2-plattformen en milstolpe i Javas ca 3.5 års korta liv. Man har tagit ett imponerande steg från enkla klientmiljöer till fullödig utvecklings- och exekveringsplattform för alla typer av tillämpningar. Alla i dagsläget 195 Java-licenstagare beräknas snabbt anpassa sina produkter till Java 2, möjligtvis med Microsoft undantaget.

4.2 Enterprise JavaBeans m m

4.2.1 Bakgrund

Portaler förutspås en lysande framtid. I alla fall de som lyckas. De kommer att utvecklas till något betydligt häftigare än de sökmaskiner de är idag. Kanske kommer det mesta av en verksamhets och även privatpersoners operationer att utföras via portaler. De kommer inom en möjlig utvecklingsgren att bli bläddrare ”i kubik”, i en annan att svara uppemot fullödig service för någon vertikal industri eller tillämpningsområde.

De generella portalerna kommer att kompletteras med sådana riktade till speciella grupper, t ex det egna företagets anställda eller kunder, beröra specifika ämnesområden och erbjuda den typ av service som gruppen behöver. Inom den senare kategorin kan en hel del av den funktionalitet som idag finns att tillgå hos företagets webbaserade tillämpningar inkluderas. Just i en portals egenskap att kunna knyta kontakt med en verksamhets tillämpningsvärld ligger kontaktytan med så kallade application servers.

Application servers är redan ett missbrukat begrepp. Det tycks stå för allt från en enda specifik till en komplett uppsättning services som tillämpningar efterfrågar under exekvering. Man brukar tala om att tillämpningar eller, i ett modernare perspektiv, komponenter (som i samverkan erbjuder någon form av service) var och en exekveras inom en container som en application server ställer till förfogande. Containern representerar, ansvarar för och utför den service komponenten har tillgång till för att utföra sitt ”jobb”, d v s hanterar exekveringen av komponenten. I denna roll hanterar containern alla dom resurser komponenten behöver och ombesörjer all eventuellt nödvändig kommunikation med externa system. En typ av komponent kanske bara behöver en viss specifik service medan en annan behöver en fullödig uppsättning services. En databashanterare (DBMS) är en application server som inriktar sig på att primärt erbjuda databasanrop för komponentens räkning. En TP monitor erbjuder transaktionsservice, resursallokering, mm. En CORBA-baserad Object Request Broker erbjuder en av OMG-definierad uppsättning CORBA Services. Enterprise JavaBeans specifikationen definierar en fix uppsättning services (APIer) för Java-komponenter, o s v. Med andra ord en brokig samling application servers, var och en med sin specialitet, sitt syfte.

Komponenter är flyttbara mellan plattformar under förutsättning att de opererar under samma leverantörs application server, men knappast annars. EJB representerar ambitionen att utgöra en specifikationen för en standardiserad uppsättning services för komponenter i Java-miljön, med förhoppningen att den kommer att anammas av alla leverantörer. Hur funktionaliteten realiseras är respektive leverantörs ensak så länge som de svarar upp mot specifikationen. Om så är fallet kan ju en komponent placeras hos och exekveras av valfri EJB application server (eller kortare: EJB Server).

Det står alldeles klart att SUN konstaterat den kommersiella potentialen av application servers. Bland annat köpet av NetDynamics (Application Server leverantör) visar på detta. Därmed inkluderar man ett nytt, mer tillämpningsnära abstraktionslager till de tidigare två: en full täckning av all upptänklig hårdvaru/OS-utrustning och ett enhetligt integrationslager i form av Java-plattformen. Kanske ser man en stark marknadsposition inom Application Servers som ett sätt att tjäna mer renodlade pengar på Java-satsningen. NetDynamics’

application server releasas under första kvartalet 1999 tillsammans med dess Studio, som är ett visuellt utvecklingsverktyg. Då tillkommer bland annat även Java WorkShop, ett utvecklingsverktyg för att bygga JavaBeans, applets, servlets och tillämpningar, och Java Blend, ett funktionalitet som transformerar data lagrade i relationsdatabaser och till Java-objekt och vice versa.

4.2.2 Diskussion

Enterprise JavaBeans (EJB) är just nu ett synnerligen hett tema för journalister och teknikbevakare. Många av konferensens teman kom också att kretsa kring EJB vilket är naturligt med tanke på konferensens inriktning mot serverbaserade tillämpningar. En utförlig genomgång av EJB-specifikationen ligger utanför syftet med denna rapport.

Tyvärr tycks EJB närmast omöjligt att förklara på ett redigt sätt, vilket är olyckligt eftersom det enligt allas bedömningar kommer att vara en central facilitet vid byggande och exekvering av framtidens komplexa, distribuerade tillämpningar. Visserligen finns specifikationen, men den kan knappast kallas lättläst, i alla händelser inte för den som inte kan Java-språket. Är faciliteten för avancerad eller komplex för att låta sig förklaras för icke-specialisten? Leder benämningen tankarna för mycket i riktning mot JavaBeans-modellen? Kan surrigheten bero på oklar modell, oklart syfte eller helt enkelt på att det är en teknikerstyrd fix utan klara principer? Var finns t ex komponentmodellen att titta på? Eller ska man se EJB som en uppsättning services tillgängliga över definierade APIer? APIerna behövs självfallet som en exakt specifikation för att kunna åstadkomma enhetliga realiseringar men ger ur användarperspektivet knappast någon ”aha”-upplevelse.

Är kanske EJB helt enkelt sällsynt bra, men ännu lite otydligt beskrivet? Är EJB bara att se som en naturlig fortsättning på succen med Java Beans? Eller en allmänt ’good timing’ i spåren av Javas framfart? Är framgången ett resultat av att EJB fyller ett uppenbart behov? Ska elogen gå till de skickliga marknadsförarna?

EJB-specifikationen och annan matnyttig information återfinns under http://java.sun.com/ejb. SUN har kompletterat specifikationen med en ”roadmap” för dess fortsatta utveckling. I en första fas, med arbetsnamnet *Moscone*, bearbetar man specifikationen för entydighet. Som ett led i denna ansträngning gör man samtidigt en referensimplementering som en ytterligare kontroll eller ”proof-of-concept”. Parallellt tar man fram tester som ska användas för att garantera att alla certifierade EJB-implementeringar har enhetlig funktionalitet. Någon gång vid slutet av 1999 beräknas denna fas vara genomförd. Därefter genomförs fasen *Javits* under något år följt av *Milano*. Mer om dessa vid någon kommande Java-rapportering.

Ingen tvekan råder om att de flesta redan ser EJB som de factostandard för ’enterprise components’ i Java-sammanhang. EJB utgör dessutom basen för den generella komponentmiljö som OMG håller på och specificerar. Effekten blir att det kommer att krävas ett helt nytt mind set, helt nya kompetenser i samband med utveckling av verksamhetbaserade IT-stöd. Maktförskjutning från tekniker till verksamhetsnära personer kommer att ske. Antagligen inte helt smärtfritt. Gedigen och beprövad kunskap/erfarenhet saknas inom området av naturliga skäl. De stödjande verktygen befinner sig ännu i sin linda. Experter kommer att vara mycket efterfrågade under överskådlig tid. Tala om utmaningar. Den klokaste, mest företagsamma och förutseende vinner.

De flesta anser att det kommer att dröja ca 3-5 år innan komponenter finns rikt företrädda i väldefinierade komponent arkitekturer.

Kommer EJBs relevans att minska/försvinna om/när OMGs CORBA Component Model (CCM) blir standard? Kommer Java då att successivt återgå till sin ursprungsroll, d v s att vara ett effektivt programmeringsspråk? Kommer istället EJBs försprång att placera det

som en de factostandard som knappast kan rubbas trots eventuellt senare öppnare lösningar? Kommer istället Microsofts motsvarande lösningar (bland annat MTS) att dominera? Ingen har svaret. Ännu.

4.3 JINI

Alla har säkert drabbats av diverse bekymmer i samband med installation av olika perifera produkter, såsom PC, skrivare, externa diskar, mm på ett nät. Inte lättare är det att få kommunikationen mellan dem att fungera. Uppgraderingar med nya versioner av de mjukvaror de nyttjar ska vi bara inte tala om. Tålmodigt har vi accepterat omständigheterna och förlitat oss på hjälp av personer med expertkunskap. Frågan är varför det ska behöva vara så komplicerat. Samma fråga ställde sig för några år sedan Bill Joy, en av grundarna av SUN, numer med titeln vice president, i praktiken i rollen som visionär och fritänkare. En roll han förmodligen upplever som mer produktiv i den friska luften och de storslagna vyerna hemma i skidorten Aspen i Colorado än i det intensiva Silicon Valley.

JINI är en teknologi som erbjuder utrustning och tillämpningar (element) att registrera sig i ett nätverk och där erbjuda service. När så skett kan andra element i nätverket nyttja den erbjudna servicen. Det första steget, att tala om att man finns, kallas *Discovery*. Man kopplar helt enkelt in sig på nätet vid en port och skickar ett 512 bits informationspaket om sig själv in i nätet. Bland annat innehåller paketet referens till sig själv. Funktionen *Lookup* lyssnar efter sådana paket, noterar dess existens och skickar information om sitt eget gränssnitt (interface) tillbaka till elementet. Med användande av gränssnittet berättar så elementet om sig själv genom att föra över information om de olika typer av service som den erbjuder plus övrig intressant karakteristik till *Lookup*. Detta steg kallas *Join* – elementet blir en ”network citizen”. Obs, även den kod, alternativt pekare till den kod, som andra element behöver för att ta servicen i anspråk skickas med.

Lookup är hjärtat i nätverket – dels ett slags anslagstavla över all service som finns att tillgå inom nätet inklusive diverse intressanta kompletterande uppgifter om servicen, dels en leverantör av erforderlig kod i och för anspråkstagande av en service. Först när ett element begär att få ta ett annat elements service i anspråk, laddas erforderlig kod ner från *Lookup* till klienten. Ingen risk för inkompatibilitet! Dock kan man ana en ganska intensiv trafik över nätet, med andra ord nätet måste vara snabbt.

Eftersom Jini är Java-baserat kan det hantera varje utrustning som innehåller en JVM, från en liten kamera till en stordator. Se vidare under McNealy ovan.

Microsoft har säkert med oro iakttagit det intresse Jini genererat. Följdriktigt har man nyligen annonserat sitt Universal Plug and Play-koncept som enligt bedömare har alldeles påtagliga likheter med Jini. En kommande tvekamp är knappast en alltför vildsint gissning.

En annan aspekt: Om erforderlig kod för att operera på olika slags utrustning under olika optioner och service finns tillgänglig på nätet i en alltid aktuell version, behöver inte längre operativsystemen bli allt svulstigare för att täcka in alla upptänkliga behov. Kanske ligger i detta på sikt en mindre dominerande roll för operativsystemen?

Ytterligare en fundering; är inte CORBA och DCOM avsedda just för detta ändamål? Den tidigare begränsningen med enbart object-by-reference som accepterad parameter i CORBAs message-kommunikation, har ju försvunnit i och med att objects-by-value (t ex Java-kod) nu är tillåten. Är det så att Jini genom sin Java-basering är betydligt enklare och därmed attraktivare – i alla händelser i Java-miljö?

SUN är duktiga på effektiva slogans. Man anser förmodligen att inte bara en bild utan även en slogan säger mer än tusen ord. Följdriktigt har även en Jini-slogan formulerats: *connect anything at anytime, anywhere.*

5 Övrigt

5.1 JP Edwards

JP Edwards står för ett företag och en produkt av typ ERP (Enterprise Resource Planning). Till denna kategori kan höra det mesta av vad som normalt förr gick under beteckningen administrativa datasystem. Finessen är att de olika komponenterna är sammanvävda till en helhet. Nödvändig eller önskad samverkan finns etablerad. I botten ligger normalt en integrerad databas och ett repository för specifiktion av alla beroendeförhållanden, komplexa strukturer, mm. Bland övriga stora aktörer inom detta segment finns SAP, Baan, IDS och i vissa mån IBMs San Francisco Framwork.

JP Edwards valde att hålla en presskonferens kring sin nya produkt vid denna konferens eftersom den till 100% är implementerad i Java, en förutsättning för att kunna föra ut som centrala budskap *anpassbarhet* och *flexibilitet* (”The only constant is change”). Dessa egenskaper har tidigare knappast varit ERP-produkternas främsta kännemärken. Kunden har tvingats anpassa sig till systemet och inte tvärtom. Dessutom har man fått köpa helhet snarare än den funktionalitet man i realiteten behöver. Att sätta i drift och att underhålla kräver en massa kodning, massa specialkunskaper som i sin tur genererat behov av dyra konsultinsatser. Kunder (ofta stora företag) har i ökad utsträckning känt irritation inför denna stelbenthet och de våldsamma kostnader som ofta är förknippade med dess användning. JP Edwards har fångat upp budskapet och vänt det till en egen affärsidé under slogan ”Idea to action”.

Det gäller att skapa mervärde för kunden. Alltför många Business Reengineering-projekt misslyckas av olika skäl. (”Man kan inte klippa ull om man inte föder fåret”). Man ska inte söka ”the best business practice” utan ”the next business practice”. Med andra ord gäller det inte längre att bara hänga med för att överleva utan att ta och ha initiativet. Är man underleverantör är läget lika prekärt. ”How do you dance with a 300 pound gorilla? Svar: Anyway they like. Med andra ord det är huvudleverantören som dikterar villkoren och underleverantören som har att snabbt anpassa sig till varje nytt eller ändrat krav. Något som endast flexibla mjukvaruprodukter kan erbjuda.

Det nya systemet releasas i april 1999.

Vi kan konstatera att förutsättningarna för ERP-system står inför snabba skift. SAP lär för övrigt arbeta med en komponentbaserad lösning. IBMs San Francisco-ramverk är redan från början utvecklat helt i Java. Konkurrensen hårdnar. Kanske hamnar vi snart i ett läge där kunden dikterar villkoren - köper endast det den vill ha. Erbjuder produkterna en komponentbaserad uppbyggnad enligt en enhetlig komponentmodell och enhetliga gränssnitt kanske inte längre de omgivande konsultinsatserna blir leverantörernas ”mjölkkossa”.

6 Några avslutande reflexioner

Kring Java

- IT-branschen florerar av megahype. Många tenderar bli immuna mot prat och ord. Man vill se lösningar. Java har frapperande snabbt gått från vision till realitet, från prat till lösning. Redan byggs tunga, så kallade kritiska tillämpningar i Java. Företag förlitar sig helt enkelt på plattformens bärkraft och överlevnad. Därmed hjälper man också till att etablera den och sprida den – givetvis under förutsättning att genomförda förehavanden lyckats. Därmed etableras en god cirkel understödd av entusiasm, energi, visioner. Plattformen håller. De nyligen etablerade, välgenomtänkta principerna för dess vidareutveckling i kombination med aktivt stöd från i stort sett alla IT-leverantörer borgar för Java-plattformens framtid. Ju mer företagen investerar i Java-teknologi, desto hårdare binder de upp sig, desto mer satsar de på marknadsgenomslag, alltsammans gynnande plattformens stabilitet och spridning. För ett par år sedan ansåg många att Java var en i och för sig frisk fläkt, men ändå en dagslända. Ideologin var god men skulle knappast kunna stå emot marknadens tuffa realiteter. Java skulle heller aldrig kunna svara upp mot de krav system i reell produktionsdrift har att efterleva. Vad fel man hade.

- ”Write Once, Run anywhere” har varit en slagkraftig slogan som fått genomlida ett antal travesteringar som en följd av att visionen inte fann sin motsvarighet i verkligheten. Förväntan böts i besvikelse när de tidigare versionerna av JDK inte kunde svara upp mot förhoppningarna. Historieperspektivet är förstås ett bekymmer man hoppas Java2 nu kommer att släta ut. Alla kval Java-programmerare tvingats genomlida under de senaste tre åren p.g.a. alla bugs och alla ickekompatibla releaser, mm ligger kvar som en gnagande oro för framtiden. En tidig entusiasm förbyttes efterhand till desillusion. Java 2 plattformen erbjuder äntligen en miljö som med betydligt större trovärdighet kan ställas bakom en fortfarande levande slogan. Java 2 måste helt enkelt lyckas. Några reservalternativ finns inte.

  Någon på konferensen invände att de initiala problemen knappast är unika för Java utan gäller för branschen som helhet. Dessutom måste vi hålla i minnet att allting skett på en synnerligen kort tid. SUN har tvingats jobba snabbt för att hålla momentum och trovärdighet. Detta har sedan balanserats på slak lina mot de brister i utvecklings-miljöerna som blivit följden under det turbulenta stadiet. Java har t ex haft att kämpa mot begränsningar på klientsidan när det gäller faciliteter för användargränssnittet, bland annat avsaknad av 2-D-grafik. Serversidan har känt av prestandaproblem och inskränkningar i kontaktytan med den icke Java-baserade tillämpningssfären. Java 2 har förhoppningsvis i mångt och mycket rått bot på problemen. En förlängning av beta-releasetiden skulle ha upplevts som oseriöst.

- Java 2 har blivit en mycket avancerad plattform inkluderande det mesta av vad man kan önska sig. Baksidan av detta ligger på utbildningsplanet. Java 2 är inget man bara installerar och kör. Det gäller att i detalj förstå all funktionalitet och hur den hänger ihop. Därefter gäller det att kunna se och matcha dess potential och begränsningar mot den egna verksamhetens förutsättningar. Till detta kommer kravet på integration med alla existerande system som antagligen knappast kommer att ersättas i närtid. Att ta Java 2 plattformen i anspråk innebär ett långsiktigt strategiskt ställningstagande av dignitet. Det gäller att se konsekvenserna.

- Det är nu dags att flytta intresset från Java till de lösningar, tjänster som kan realiseras med hjälp av Java. Java bryter starkt in på server-sidan. En alldeles påtaglig glidning från lattjo-lajban till seriösa tillämpningar kan märkas. Många existerande tillämpningar kommer successivt att transformeras till Javamiljön. Man kan bara ana den uppfinningsrikedom som kan följa i Javas spår när det gäller nya tillämpningar. Java i kombination med CORBA och IIOP kommer att öppna för lösningar som endast den mest fantasifulle kan peka ut. Men självfallet kommer det alltid att finnas tillämpningar som ställer sådana specifika krav på prestanda och/eller funktionalitet att de bättre realiseras på en specifik plattform.

- Många pekar på att Java trots allt har en inbyggd tröskel att ta sig över. Det krävs tålamod för att bli en duktig Java-nyttjare. Speciellt nu när det växt ut till en avancerad plattform. Den duktige kan åstadkomma de mest fantastiska saker medan den mindre duktige kan skapa lösningar som blir helt fel. Därför är det viktigt att vara med i ett tidigt skede av framväxten av en ny teknologi. Att bygga upp en kunskapsbas i företaget genom att studera, pröva, lyckas och misslyckas. Denna bas bör spridas över många för att undvika sårbarhet. Visst är det ett risktagande om den nya teknologin inte tar fart, men förmodligen är förlusterna betydligt större genom att vara sent ute med inkorporering av ny teknik än upplärningsförlusterna. De senare är ringa. Kunskapen kan säkerligen appliceras på något annat. Observera att en ny generell teknologi sällan stryker på foten. Möjligtvis tar den tid att mogna och realiseras i ett antal olika versioner. Den generella kunskapen kan nyttjas oavsett version.

  Det är med andra ord knappast vettigt att avstå från att åtminstone pröva eller titta på Java. Konkurrenterna gör det. Att stå utanför är att ta en stor risk. Att investera i kunskap och att pröva är en beskedlig form av gardering.

- Vid en diskussion om framtiden konstaterades att de nya abstraktionsnivåerna t ex JavaBeans och Enterprise Javabeans bidrar till innovationer både över och under nivån eftersom man isolerar både problemställning och beroende. Överhuvudtaget ser man en snabb trend mot än mer kraftfulla, lättanvända verktyg som äntligen gör det möjligt för de verksamhetskunniga att själva kunna skapa de stödsystem de behöver med minimalt programmerarstöd och hela tiden med koncentration på problemlösning, inte på intrikata teknikanpassningar.

Allmänt

- E-commerce fick sitt givna utrymme på denna konferens liksom på alla andra. Man kan mycket klart notera att begreppet börjat mista sitt inslag av budskapskraft och övergått till ett slitet begrepp. Från att från början fått representera köp och sälj med hjälp av datorbaserade system i samverkan expanderade det snabbt till att omfatta vad vi i Sverige brukar kalla elektronisk affärsverksamhet, d v s det mesta som direkt eller indirekt har koppling till elektronisk handel, till att så småningom innefatta i stort sett allt som företag utför med hjälp av system i samverkan både internt men kanske framförallt externt med andra företag och/eller individer. Genom denna rockad införlivar man, om och när man så önskar, inom sitt problemområde det mesta av vad som traditionellt hänförts till datorteknik, datateknik, numerisk analys, informationsteknik, företagsekonomi, sociologi, psykologi, ……. Till nytta för ingen. Trenden tycks därför vara att återföra problemställningarna till sina traditionella discipliner och låta e-commerce stå för ett intressant tillämpningsområde, om än med oklara gränsytor.

- XML tilldrar sig ett enormt intresse. Flera presentationer hölls, alla i överfulla salar. Sueltz, IBM pekade också ut XML som en mycket viktig standard i denna nya, öppna värld.

- NCs kommer tillbaka när network services blir tillräckligt rikhaltigt och stabilt och när nätkapaciteten byggts ut. De flesta tycks dock överens om att NCs inte kommer att realiseras i en standardform utan i många varierande former och med varierande kapacitet beroende på avsett syfte.

Vad man än har för preferenser för övrigt kring plattformar, m m kan inte någon undvika att imponeras av vad som kreerats under Javas ca fyraåriga historia. Och det gensvar som resulterat. Vi har återigen, vid sidan av Internet, kunnat följa ett unikt stycke IT-historia som nyss nått ett viktigt delmål i form av Java2 Platform.