

SunONE og MS .NET: Tvillinger i krig

Først kom Java. Så kom .NET. Dernest dukket Web-tjenester opp. Hvordan kan det ha seg at .NET og Java hevdes å primært handle om Web-tjenester? Og hva har Java med SunONE å gjøre?

Veien blir til mens vi går, heter det, og IT-historien er full av eksempler på nettopp dette. Derfor er det verken overraskende eller negativt at Java, .NET og omkringliggende teknologier har forandret seg vesentlig i løpet av sin levetid. Tvert imot er tilpasningsdyktigheten positiv og nødvendig.

Mens vi skal holde fokus på situasjonen i dag og forventningene fremover, er det nyttig å ha kunnskap om hvordan vi er kommet hit – ikke minst for å evaluere om leverandørenes veikart mot fremtiden er pålitelige og behovsriktige. Erfaring forteller at det vi blir fortalt om fremtidsplaner og kommende produkter som regel forandrer seg både én og flere ganger innen fremtiden blir nåtid og produktene skal løse praktiske problemer. Dermed blir evalueringene i dag et spørsmål om tillit på den ene siden og erfaring på den andre. Er for eksempel .NET en Windows-sentrisk Java-klone kokt på kildekode og ideer Microsoft lisensierte fra Sun i 1995?

I begynnelsen var Java

Teknologiutviklingen som står på agendaen, begynte tidlig på 90-tallet – med Java, et laboratorie-eksperiment som ble oppdaget ved en tilfældighet og som traff Internett- og Web-utviklingen på et perfekt tidspunkt.⁵ Ideene harmonerte med voksende behov, alternativene manglet og Internettet sørget for å spre budskapet på rekordtid. Beta-utgaver av Java-verktøy ble lastet ned i titusentall, og en stadig mer nettverksorientert verden ventet på offisielle, kommersielle utgaver av vidunderet.

Mens Java ventet på sin tur i manesjen, oppdaget Microsoft hva som var i ferd med å skje. På samme måte som Internettet og WWW kort tid i forveien hadde tatt selskapet på sengen, fremsto Java plutselig som en trussel på flere områder. For det første var teknologien plattform-uavhengig – og forfektet derigjennom det motsatte av Microsofts Windows-sentriske tankegang. Dernest var Java katalysator for en ny arkitektur som flyttet intelligens fra operativsystemene på klient og tjener til henholdsvis nettleser og Web-tjener – med en åpenbar fleksibilitetsgevinst. Og sist men ikke minst var Java utenfor Microsofts kontroll.

⁵ Mer om Java som teknologi og dens opprinnelse i artikkelen "Java: Mer enn stor ord og kaffe" i Mellvik-Rapporten nr. 37, tilgjengelig i pdf-format fra vår Web-tjeneste, se side 35.

Plattformuavhengigheten kombinert med selve konseptet ga Java potensiale langt utenfor tradisjonelle IT-anvendelser, et forhold Sun som eier av teknologien visste å utnytte, ikke minst i markedsførings-sammenheng. Utviklere og utviklingsmiljøer over hele verden flokket seg rundt Java, som representerte frigjøring og muligheter på samme tid. Porting til eller fra Unix eller Windows kostet store ressurser og representerte verken innovasjon eller teknologisk utfordring for programmerere. Videre virket det sannsynlig at Java ville finne veien inn i alle tenkelige og utenkelige sammenhenger, fra mobiltelefoner til biler og kjøkkenmaskiner, og derigjennom revolusjonere vårt syn på og bruk av Internettet.

Microsoft våkner

Stort verre kunne det ikke bli sett fra Microsofts side. Selskapet hadde imidlertid med betydelig suksess håndtert lignende utfordringer, og responderte umiddelbart da trusselen først var avdekket. Første trinn i motangrepet var velkjent og velprøvd: *Embrace and Extend*. Microsoft lisensierte like godt Java fra Sun, og gjorde Windows og Internet Explorer Java-kompatible. En stund så det virkelig ut til at Java ville bli den suksessen Sun og mange utviklingsmiljøer hadde drømt om.

Allerede i 1998 begynte det imidlertid å skurre. I henhold til avtalen med Sun var Microsoft forpliktet til å ikke foreta endringer i Java eller omkringliggende miljø som kunne føre til kompatibilitetsproblemer. Like fullt dukket slike endringer opp mindre enn et år etter at avtalen var inngått. Microsoft hevdet at endringene var nødvendige, og at avtalene kunne tolkes slik at optimaliseringer var akseptable. Dessuten påpekte Microsoft, uten tvil med betydelig rett, at Sun var unødvendig firkantede og negativt innstilt til videreutvikling og tilpasninger av Java. Det passet rett og slett Sun godt at Java hadde dårligere ytelse under Windows enn på andre plattformer.

Dermed braket de to sammen i retten – i parallell med Microsofts monopol-rettsaker. Microsoft hadde allerede utviklet og startet distribusjonen av sin Windows-tilpassede JVM, *Java Virtual Machine*, den sentrale komponenten for utførelse av Java-programmer. Produktet hadde en rekke Windows-spesifikke utvidelser som mer eller mindre

Java inn i rampelyset

Java flyttet seg fra teori til virkelighet i januar 1996, med introduksjonen av utviklingspakken JDK 1.0 (*Java Development Kit*). Tilfeldighetene sørget for at tidspunktet sammenfalt med at Windows 95 for alvor forserte alle hindringer og rullet over verden med en suksess uten sidestykke. Det planlagte anvendelsesområdet for Java var industrielle anvendelser, et kompakt språk og tilhørende miljø for industrielt utstyr og det vi i dag gjerne kaller 'duppeditter'. Miljøet var tilpasset den nye nettverks-virkeligheten, plattformuavhengig og sikkert. Utviklere så en mulighet for å komme ut av låsingen til plattformer.

I stedet for å havne i nettopp slike omgivelser, fant imidlertid Java veien til Netscapes nettleser, der programmerbarheten på klientsiden ble tatt til et nytt nivå. Selv i dag er det i slike omgivelser Java har størst utbredelse, riktignok i større grad på tjenersiden enn på klientsiden. Tilstedeværelsen i industrielle anvendelser er fortsatt relativt beskjeden, om enn voksende, og mens Java er å finne på millioner av mobiltelefoner, er nytteverdien stort sett begrenset til spill.

bevisst ble benyttet av tallrike utviklingsmiljøer. Den juridiske kvernen maler langsomt, og innen Sun fikk rettens medhold og kunne stoppe Microsofts JVM, var Javas plattformuavhengighet historie.

Den juridiske dragkampen mellom Microsoft og Sun fortsetter den dag i dag, og har påført begge parter både medhold og tap. De faktiske for-

holdene lar seg imidlertid ikke bestride: Sun har vært vanskelige å ha med å gjøre, og Microsoft har ved en rekke anledninger misligholdt sine avtalemessige forpliktelser. Konsekvensen er at Microsoft har lyktes i å bremse utviklingen for Java tilstrekkelig lenge til å ha fått på plass sine egne alternativer. Å fjerne Java-trusselen har de imidlertid ikke klart, og hvordan bildet ser ut i dag, skal vi komme tilbake til nedenfor.

Java mot .NET: Hva er forskjellen?

Java og .NET har to sentrale komponenter hver: Programmeringsspråket, henholdsvis Java og C#, og den virtuelle maskinen der programmene utføres, JVM (*Java Virtual Machine*) og CLR (*Common Language Runtime*). På overflaten har de to påtagelig mange fellestrekk, funksjonelt såvel som konseptuelt. Mens det ikke finnes grunnlag for å hevde at Microsoft har benyttet kildekode fra Java i utviklingen av .NET-produktene, er det et faktum at selskapet – med utgangspunkt i sin tidlige Java-lisens – har utviklet sin egen JVM for Windows. Dermed er det også naivt å tro at ressursene og erfaringene som ble opparbeidet, ikke er benyttet i forbindelse med C# og CLR.

Likhetspunktene og historien kvalifiserer spørsmålet om ikke dette er to sider av samme sak, og at det dermed blir likegyldig hvilken vi velger. Svaret er betinget: Ja, de er to sider av samme sak, men nei, det er ikke likegyldig hvilken som velges. Microsoft har implementert Suns ideer, hvilket vi kan ha meninger om, men som ikke er illegalt på noen måte. Dessuten er det ikke til å komme forbi at forholdet mellom de to kunne ha vært annerledes dersom Sun hadde ønsket det (se teksten). En spesialist på området, Stephen Gilmore, har treffende formulert forskjellen slik: *“The Java Virtual Machine is an object-oriented execution environment for any language so long as it's Java. The .NET Platform is an object-oriented execution environment for any language so long as it isn't Java.”* Sett fra utsiden er dette en glimrende oppsummering. JVM er laget for Java, CLR er i prinsippet språkneutrale. Denne nøytraliteten påvirkes naturligvis av de restriksjonene og kravene CLR introduserer, men det er fortsatt mulig – og demonstrert – at kompilatorer for andre språk enn C# kan tilpasses CLR. Microsoft har selv en håndfull slike verktøy, inklusive den relativt ukjente Java-kompatible klonen J# og den langt mer kjente VisualBasic.NET. Sistnevnte er spesielt interessant fordi den er inkompatibel med tidligere versjoner av VB, hvilket har gjort VB.NET mektig upopulær i viktige VB-miljøer som føler seg forrådt av Microsoft.

På lavere nivå er ulikhetene flere. CLR kompilerer alltid koden før den utføres (eller den kan prekompileres), mens JVM kan gjøre begge deler – interpretere eller compilere etter JIT (*Just In Time*) prinsippet. Den viktigste forskjellen er imidlertid restriksjonene som pålegges koden av utførelsesmiljøet. Mens JVM er meget restriktiv, av veldokumenterte sikkerhetsårsaker, gir CLR store frihetsgrader som på den ene siden forenkler utviklingen av effektiv kode for spesifikke miljøer, men samtidig går kraftig på akkord med både sikkerhet og portabilitet.

Begge disse forholdene er viktige, og gitt selskapets historiske forhold til sikkerhet og programvarekvalitet er Microsofts valg overraskende. Armert med erfaringene fra Java vet vi at det ikke er manglende kunnskap om sikkerhetsmessige forhold som har ligget til grunn for valgene. Dermed står vi igjen med prioritering, og det hersker liten tvil om at Microsoft her har gjort et dårlig valg sett fra markedets side: Sikkerhet er .NET-plattformens største handicap og markedets viktigste innvending. Når vi graver dypere i materialet, kommer imidlertid forklaringen for en dag – se rammen på neste side!

Med hensyn til plattformuavhengighet fikk Java demonstrert sine egenskaper allerede før det ble et kommersielt produkt. På .NET-siden er CLR i utgangspunktet plattformuavhengig, og kan kjøre CLR-kompatible programmer under Windows/Intel og på flere Windows CE og PocketPC plattformer. Videre har Microsoft frigitt kildekode til CLR-moduler for BSD-Unix og Mac OS X. Ximian Inc., som nå eies av Novell, arbeider med en CLR-implementasjon for Linux under navnet 'Mono'. På grunn av den åpne (usikre) arkitekturen, skal det imidlertid mer til enn CLR for å kjøre programmer på en gitt plattform. Utstrakt bruk av systembiblioteker som i mange tilfeller kun er tilgjengelige under Windows og med lisens fra Microsoft, gjør portabiliteten temmelig teoretisk i dag. Samtidig er dette et forhold Microsoft kontrollerer, og som de kan velge å forandre når de måtte ønske det.

.NET tar form

Mens tumultene rundt Java raste, arbeidet Microsoft febrilsk med en strategi som for det første kunne bremse interessen for Java og plattformuavhengighet, og dernest gi Windows en langt større spennvidde. Ved å gjøre Windows tilgjengelig på alle tenkelige plattformer, kunne oppmerksomheten rundt plattformuavhengighet reduseres. Selskapet var allerede i gang med barberte utgaver av Windows for PDAer og enkle maskiner (tynne klienter), men manglet en sammenhengende strategi for konnektivitet og nettverkssentriske løsninger.

Resultatet ble .NET. Etter en lett modifisert Java-modell skulle alle selskapets produkter gjøres nettsentriske. Fremtidige utgaver av Windows skulle stå sentralt i strategien, og en omfattende samling verktøy skulle gi utviklingsmiljøene snarveier til målet. Planene ble 'lekket' til media høsten 1999, og fungerte som ytterligere brems for Java i markedet – til

tross for at ingen var i stand til å konkretisere verken planer eller produkter. Selv et år senere, etter at C# (C-sharp), Microsofts svar på Java var introdusert, var .NET-konseptet i beste fall ullent. Forvirringen ble ikke mindre av at selskapets ledere, med Bill Gates og Steve Ballmer i spissen, stadig forandret sitt budskap.

Web-tjenester: Forvirring og oppklaring

Da Web-tjenester for alvor kom på banen i 2000 og 2001, ble forvirringen rundt .NET i første omgang nærmest total. Etterhvert viste det seg imidlertid at de nye ideene hadde positive effekter på det rådende kaoset. Web-tjenester hadde så åpenbare attraktive sider at ingen i markedet kunne la være å definere en strategi med *Web-services* i sentrum. Og dette var nettopp hva Microsoft trengte for å få en solid knagg å henge .NET på. Faktum er at Microsoft siden 1998 hadde arbeidet intenst for å få frem XML-standarden som grunnlag for Web-applikasjoner og integrasjon, ikke fordi slik standardisering egentlig er i selskapets interesse, men for å komme rundt det solide grepet Sun og deres Java-tilhengere hadde fått om utviklingen. Microsoft tok sjansen på å torpedere sin egen massefart med COM og DCOM-standardene for å kunne ta luften ut av Java RMI og CORBA i samme slengen.

Fornytt entusiasme for plattformuavhengighet og for et mer nyansert syn på hva som skal til for å skape reell integrasjon mellom applikasjoner

(se egen artikkel på side 4), fulgte med på kjøpet, og var en akseptabel kamel å svelge for Microsoft i forhold til alternativene. I løpet av det påfølgende året ble både Java med omkringliggende verktøy og løsninger, og ikke minst .NET, omlaftet rundt XML og de ferske *Web Services* standardene.

I kjølvannet av markedets fokusering på Web-tjenester, fikk Microsoft det de trengte for å konkretisere planer og målsettinger med .NET. Resultatet er et omfattende og imponerende rammeverk og utviklingsmiljø for nettverkssentriske løsninger med *Web Services* standardene og XML som sentrale elementer.

Mye vil ha mer ...

Selv om Microsofts .NET-teknologi fortsatt spiller en relativt ubetydelig rolle i markedet, kan den vanskelig kalles noe annet enn en suksess. Java ble bremsert lenge nok til at Microsoft fikk skrappt sammen alter-

RMI – *Remote Method Invocation*
CORBA – *Common Object Request Broker Architecture*
XML – *eXtensible Markup Language*
COM – *Common Object Model*
DCOM – *Distributed COM*

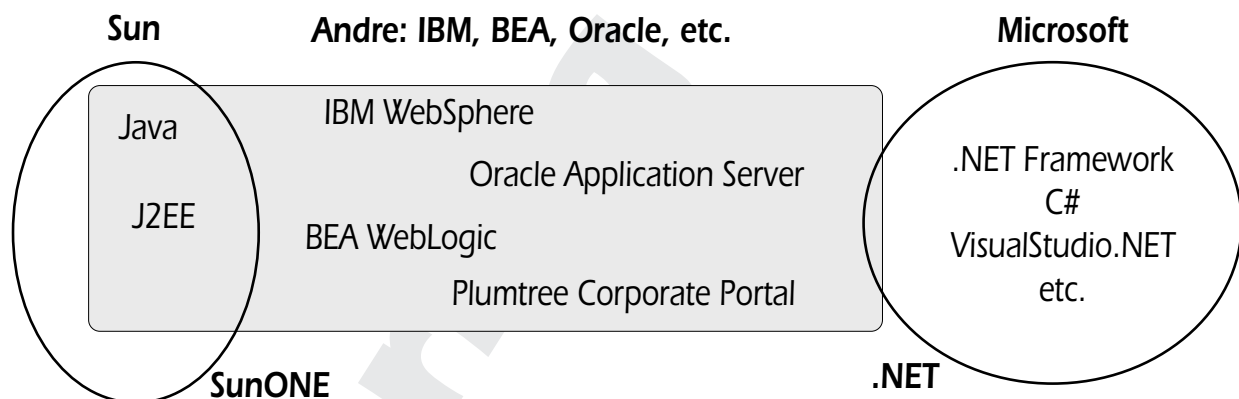
Underlige, men forklarlige valg fra Microsoft

En rekke av disposisjonene Microsoft har foretatt i tilknytning til .NET forekommer unektelig pussige ved første øyekast. Java-klonen J# er ett slikt eksempel: Hvorfor bruker Microsoft ressurser på et 100% Java-kompatibelt språk når deres primære interesse er å få markedet over på C#? Forklaringen er først og fremst realisme. Å ønske noe er så sin sak, men Microsofts teknologer og markedsførere har på ingen måte koblet bort virkeligheten – som forteller dem at Java har for godt grep om markedet til å kunne bli borte, og dessuten har godt fotfeste i skoler og akademiske miljøer. For å kunne utnytte denne kompetansen og samtidig stimulere Windows' posisjon, er veien via J# innlysende. Og når brukerne først har J# under beltet og befinner seg på en Windows-plattform, er veien over til C# kort når fordelene blir åpenbare.

Et annet og enda mer besynderlig forhold er sikkerhet. .NETs CLR er notorisk usikkert i praksis, til tross for at mekanismer for god sikkerhet finnes. Slik arkitekturen implementeres i dag, er det Windows (eller en annen støttet OS-plattform) som ivaretar sikkerheten, ikke .NET-rammeverket. Som vi var inne på på foregående side, er dette i beste fall et betenkelig valg sett fra et sikkerhetsmessig standpunkt. På den andre siden er det den eneste måten å gi .NET-applikasjoner, som per definisjon hører hjemme innenfor nettleserens rammer, tette koblinger med Office og andre regulære Windows-applikasjoner. Den største trusselen for Microsoft siden Web-bølgen startet på 90-tallet har vært nettopp overgangen fra den tradisjonelle vindussystem/OS-kombinasjonen til nettleseren. Det nye rammeverket kompliserer de tette koblingene Microsoft benytter for på den ene siden å forenkle samspeillet mellom ulike verktøy, og på den andre siden å låse brukere til Windows-plattformen. Ved å fjerne sikkerhetskravene til .NET-applikasjoner og flytte sikkerhetsansvaret over til det underliggende operativsystemet, har Microsoft tatt vare på behovet for tett integrasjon og låsing: .NET-applikasjoner kan samarbeide med Office og andre verktøy på samme måten som om de var lokale under Windows. Konsekvensene for sikkerheten er innlysende. Så vil tiden vise om valget var riktig eller galt.

native produkter, programmeringsmiljøet C#, som opprinnelig ble dødsdømt av fagmiljøene, har fått betydelig utbredelse, og Microsoft har etablert en fremskutt plassering i posisjoneringskampen rundt Web-tjenester. Videre har selskapet skaffet seg en betydelig tilstedeværelse på PDA-siden og en beskjeden, men voksende posisjon i mobiltelefonmarkedet. Posisjoneringen i forhold til Web-tjenester og forandringene i deres kjølvann, er generelt bra.

Microsofts største utfordring er å finne på helt andre områder. Selskapet står i voksende grad alene mot resten av IT-verden. Grådigheten kan synes å ha tatt fullstendig overhånd i den forstand at Gates & Co. kaster seg over stadig nye markeder der deres partnere tidligere har hatt forutsigbare forretningsforhold. Gamle venner blir fiender, samtidig med at markedet i sin alminnelighet blir mer skeptisk. Den gryende uro vi ser i markedet i dag, fra offentlige myndigheter i store deler av verden og fra store, internasjonale selskaper, burde få de røde lampene til å gløde i Redmond. Open Source, Linux, Java og andre teknologier får ekstra vind i seilene nettopp på grunn av Microsofts umettelighet. Likeledes er det ikke til å komme forbi at størrelse og markedsdominans slett ikke har vært positive faktorer for programvarekvalitet. Selskapets størrelse og posisjon gjør at dette ikke er noen ulik kamp – snarere tvert imot.



Figur 6 Til tross for enorm innsats fra Microsofts side, er Java og J2EE (JAVA 2 ENTERPRISE EDITION, også kjent som SERVER SIDE JAVA) fortsatt dominant i markedet. Denne situasjonen vil neppe forandre seg i overskuelig fremtid – blant annet fordi programvareleverandører generelt er skeptiske til tett samarbeid med giganten Microsoft. Etter stadige oppkjøp og inntredener i nye markeder er Microsoft en partner få tør å stole på, og som de fleste er redde for.

Markedspolarisering

Vi ser med andre ord en polarisering av programvare-markedet, med Microsoft og deres proprietære teknologier på den ene siden, Sun på den andre, og de øvrige aktørene imellom, men med betydelige bestanddeler av Suns Java-teknologi som sentrale komponenter. Derfor er Java fortsatt dominant i forhold til C# og .NET. Selv om Microsoft ruller ut all verdens verktøy og støtteapparat rundt kommende utgaver av Windows, er det ikke sannsynlig at fordelingen vil forandre seg vesentlig. For altfor mange aktører på hele spekteret fra mobiltelefoner via telekommunikasjon til servere er Microsoft en trussel – et mer enn godt nok incentiv til å velge andre alternativer dersom de fin-

nes. Det gjør de som regel, og alternativene gir dessuten langt større grad av fleksibilitet i forhold til valg av plattformer, verktøy og systemer enn Microsoft. Denne spiralen er vanskelig å stoppe og vil uvegerlig bidra til at polariseringen mellom Microsoft på den ene siden og resten av programvareleverandørene på den andre, fortsetter å utvikle seg.

Hvor blir det av Sun?

Ved siden av etter beste evne å ha markedsført Java og relaterte verktøy, har Sun åpenbart hatt hendene fulle med å forsvare seg mot Microsofts direkte og indirekte angrep siden Java kom på banen i 1995. Vi har allerede diskutert noen av konsekvensene av denne endeløse tvekampen, men Microsoft er på ingen måte den eneste årsaken til at Java ikke har tatt av med den hastighet og styrke mange hadde ventet.

Siden introduksjonen av Java har Sun gjort den ene tabben etter den andre – i forhold til partnere, utviklere, standardiseringsorganisasjoner og markedet generelt. Selskapet har på den ene siden klaget over Microsoft, deres metoder og deres lukkede teknologier, for deretter å snu seg rundt og bruke omtrent samme oppskrift selv. Forsøkene på å få både i pose og sekk gjennom på den ene siden å 'frigi' Java for standardisering, og samtidig holde igjen på rettighetene til lisenser og videreutvikling, har ved flere anledninger satt sinnene i kok i utviklingsmiljøer.

Et annet forhold som ikke nødvendigvis har bremsset utviklingen, men i alle fall har ført til negativ publisitet er Java-prosjekter som først er blåst voldsomt opp, for deretter å feile. Suns HotJava nettleser – plattformuavhengig og i sin helhet skrevet i Java – ble droppet etter en krangel med Netscape, som ikke ønsket konkurranse fra den kanten. For JavaOS og Java-prosessorer ble mye skrik til lite ull, mens Java-baserte SmartCards, som det er produsert mer enn 100 millioner av, har gått forbausende upåaktet hen.

SunONE – Open Net Environment: Hvorfor og hvordan

I ettertid kan det synes underlig at Sun ikke har gjort det for lenge siden, men det er lite produktivt å lete bakover etter hva som skulle ha vært gjort når. Det Sun har gjort med SunONE er å lytte til markedet og erkjenne fundamentale markedsføringsmessige forhold. Ingen av produktene som inngår i rammeverket er nye, de har eksistert på egen hånd eller i grupper med varierende levetid.

Det markedet ønsker er forenkling og forutsigbarhet, og Sun har gjennom SunONE sørget for nettopp dette. Produktene synkroniseres utviklingsmessig, integrasjon på tvers blir enklere, nye versjoner kommer til samme tid og med samme fokus, og tilpasningen til underliggende operativsystemer blir lik. Samtidig har Sun laget en forenklet prismodell som tar hensyn til det faktum at kundene aldri kjøper enkeltprodukter, men funksjonelle kombinasjoner eller grupper.

Sun mener at prisingen blir nøkkelen til fremgang for SunONE, sammen med fokuseringen på integrasjon – med XML som grunnlag. *JavaXML bus* har mye med Web-tjenester å gjøre, og er en sentral komponent i bildet, men brukes underlig nok ikke i markedsføringen fra Suns side.

Ved siden av markedsføring i sin alminnelighet, har Sun minimal kompetanse på salg av volumprodukter. Begge deler er vesentlige svakheter, som ikke minst kommer til uttrykk når motparten er Microsoft.

Det er en alminnelig oppfatning i markedet at Javas suksess så langt i større grad har kommet til tross for Sun, enn på grunn av Sun. Likeledes er det et faktum at selskapets partnere – med IBM i spissen – har spilt en særdeles viktig rolle i legitimeringen av Java som utviklingsplattform. Sist, men ikke minst fremstår det som besynderlig at Sun, etter å ha lyktes i å standardisere Java gjennom ISO uten å slippe kontrollen over teknolo-



Figur 7 Bedre sent enn aldri: Sun har omsider oppdaget at det er en god idé å konsolidere individuelle produkter til pakker – som gjør produktene lettere å forstå, enklere å vedlikeholde og lettere å selge.

giens fremtid, 'glemte' å utnytte standardiseringen til noe som helst. I dag, flere år etterpå, er det få som er klar over at Java er noe annet enn en leverandørstandard.

Når dette er sagt, er det også et faktum at Sun, takket være både Java-teknologien og sin historie som leverandør av operativsystemer og verktøy, er den eneste aktøren i IT-markedet som kan måle seg med Microsoft når det gjelder bredde. Selskapet var gjennom sin tunge tilstedeværelse i akademiske miljøer midt i smørøyet da Internett- og Web-bølgene tok av for alvor, og har gjennom en kombinasjon av egen utvikling og innkjøpte teknologier skaffet seg en bredde på programvaresiden som er bemerkelsesverdig lik tilsvarende hos Microsoft. Overtagelsen av den tyske Office-klonen StarOffice for 3 år siden tet-

tet et viktig hull i porteføljen, og bidro samtidig til økt visibilitet for Sun i sluttbrukermarkedet. Sist, men ikke minst har Sun et helt annet ry med hensyn til programvarekvalitet enn erkekonkurrenten.

Disse forholdene er årsaken til at vi i vår analyse av situasjonen for Sun Microsystems på side 28, konkluderer med at selskapets fremtid er lysere på programvaresiden enn på systemsiden. Denne virkeligheten vegrer Sun seg naturlig nok sterkt fra å akseptere. Tett kobling mellom programvare og systemer har brakt selskapet dit det er i dag, og hardware/systemer står fortsatt for brorparten av selskapets inntekter.

I forhold til Microsoft er Sun på mange måter for en amatør å regne når programvare skal markedsføres, og SunONE er et glimrende eksempel i så henseende. Mens Microsoft lenge før .NET hadde noen som helst substans kunne presentere en samling på flere dusin produkter som skulle inngå i konseptet, oppdaget Sun først i fjor at det er en god idé å samle beslektede programvareprodukter under én paraply. Resultatet ble SunONE – en overmoden konsolidering (se ramme på forrige side). Mindre påtagelig blir det ikke når Sun velger å konsolidere sine utviklingsverktøy innenfor rammen av SunONE under navnet 'SunONE Studio'. Mon tro hvor ideen til det navnet kommer fra?

For begge leverandørers vedkommende er det i første omgang snakk om konsolidering og nye navn på gamle produkter – *same shit, new wrapping*. Poenget er imidlertid ikke å lage noe nytt, men å evaluere hva markedet trenger og deretter sette sammen en historie som er enkel å forstå og tilsvarende lett å selge.

Dermed er vi på sett og vis tilbake til utgangspunktet: Vi konstaterte innledningsvis at .NET og SunONE har mye til felles, men hvor like er

**SunONE vs.
.NET**

de og hva er deres sterke og svake sider? Underliggende teknologi sammen med fordeler og ulemper har vi allerede diskutert, mens innholdet fortsatt er ullent. For Microsofts vedkommende kan vi lite gjøre med denne tåkeskyen. Verken direkte spørsmål, dokumenter, presentasjoner eller andre informasjonskilder gir klare svar på hva som inngår i .NET og hva som ikke gjør det. Den generelle tilbakemeldingen er at alt inngår, og det som ikke inngår i dag, vil inngå i morgen. På et overordnet nivå er dette vel og bra, men det forteller lite om hvordan de enkelte produktene og produktgruppene skal spille sammen – utover at limet dem imellom er Web-tjenester. Om og hvordan åpne standarder skal brukes i den forbindelse, forblir ubesvarte spørsmål.

Tabellen nedenfor gir en oppstilling av funksjonelle ingredienser, og må ikke oppfattes bokstavelig. Oversikten forteller mer om leverandørenes vinkling mot markedet enn mot hva konseptene er i stand til. For begge finnes det et tosifret antall 3. parts leverandører som uten særlig skreddersøm lar seg integrere i rammeverket. Dessuten er det først og fremst rammeverk vi snakker om. Komponentene som kan leveres ferdige fra leverandørene, er viktige nok, men det er synkronisering av rammeverk og behov, og ikke minst hva vi tror om fremtiden som avgjør om den ene plattformen er mer eller mindre attraktiv i forhold til den andre.

Microsoft .NET bestanddeler

- Web-tjenester som sentralt element på alle nivåer
- Utviklingsverktøy: VisualStudio.NET (C#, C++), VisualBasic.NET, .NET Framework (omgivelsene programmer skal utføres i, for ulike versjoner av Windows)
- Samtlige tjeneroperativsystemer og back office verktøy
- Klientprogramvare, inklusive kommende utgaver av Office og andre brukerverktøy

SunONE bestanddeler

- Katalogtjenester og tilhørende sikkerhets-infrastruktur (proxy-tjenester)
- Meta-katalogtjenester – for aggregering av informasjon fra ulike databaser og andre kilder
- Autentiseringstjenester
- Portal-tjenester, Web-tjenester og Web-proxy
- Meldings- (IM, epost etc.) og kalender-tjenester
- Tradisjonelle integrasjons-tjenester (EAI) og B2B integrasjons-tjenester
- Utviklingsverktøy og tilhørende tjenester
- Applikasjons-tjenester – for Java-applikasjoner med grensesnitt bakover mot tradisjonelle applikasjoner og forover mot portal- og Web-tjener

Flere forhold påkaller spesiell interesse i en slik oppstilling. Mens Microsoft i presentasjons- og markedsføringsmateriale stadig kommer tilbake til Web-tjenester, XML og standarder, er disse begrepene vanskelige å finne hos Sun. Dersom vi kommer til disse presentasjonene uten historisk ballast, vil vi sitte igjen med inntrykk av at Microsoft går for standarder mens Sun satser helt og holdent på proprietære produkter og mekanismer. Realiteten er en annen, men forskjellen mellom de to er mindre enn historien skulle tilsi. Dessuten er det et faktum at Microsoft ligger minst et hestehode foran Sun nettopp innen praktisk anvendelse av Web-tjenester med tilhørende standarder og verktøy. Hvor viktig dette er i praksis kan diskuteres, men at Microsoft dermed har en bedre historie å fortelle er hevet over tvil.

På generell basis har Sun et langt lerret å bleke før de kommer i nærheten av Microsoft med hensyn til markedsføring og markedskommu-

nikasjon. Slik situasjonen er i dag, trengs det ekspertise med praktisk erfaring fra Suns produkter for å finne ut hva de egentlig er verdt og hva de står for i forhold til standarder.

Konklusjon

Det viktige valget vi står overfor, er med andre ord ikke mellom SunONE og .NET, men mellom .NET og Java. Figur 6 på side 17 illustrerer hvordan forholdene ser ut i dagens marked. Som vi har sett er det lite som tyder på at de store blokkene vil forandre seg vesentlig i overskuelig fremtid. Microsoft står i voksende grad alene som en Goliat på banen, og kjemper ikke mot én, men mot en betydelig mengde 'Davider' som ikke er verken små eller svake.

I dag har Java et formidabelt overtak i markedet – både med hensyn til antall aktive utviklere og i markedsandel. Som vi har sett eksempler på tidligere når Microsoft har kommet sent ut i starten og blitt tvunget til å spille *catch up*, tar det forbausende kort tid før kontakten med tettefeltet er etablert. Om .NET fortsatt er aldri så ullent i kantene, finnes det nok av praktiske eksempler på at Microsoft allerede har et funksjonelt attraktivt alternativ til Java. Likeledes har Microsoft ved å komme sent på banen hatt anledning til å lære av feilene Sun har gjort. "Om vi ser bort fra sikkerheten, er .NET hva Java skulle ha vært", påpekte en utvikler for en stund siden.

Microsofts sterke fokus på standarder (*Web services*) er besnærende, og bidrar i stor grad til å legitimere selskapets løsninger. Det ville imidlertid være både historieløst og naivt å ikke løfte blikket tilstrekkelig til å se at det er integrasjon med egne plattformer, løsninger og verktøy som er den egentlige målsettingen for Microsoft. Forsakelsen av sikkerhet for å oppnå den nødvendige integrasjonen er et viktig moment å ha med i vurderingen.

Valget står i første omgang om .NET eller ikke .NET. Velger vi å satse på Microsoft som den viktigste og kanskje eneste teknologiske hest i årene fremover, er .NET et naturlig valg. Legger vi vekt på valgmuligheter, sikkerhet og å spre risikoen, bli Java det opplagte fundament – og vi har en lang rekke leverandører å velge mellom. Flere av disse har en bedre historie å fortelle om Web-tjenester enn Sun Microsystems – i dag.

Om .NET og SunONE er tvillinger? Uten tvil. Avkledd er bestanddelene påtagelig like, og om foreldrene har ulike ressurser, interesser og fokusering, og derfor kler dem opp forskjellig, er de rettet inn mot de samme problemstillingene: Å løse dagens og spesielt morgendagens IT-utfordringer på en måte som på den ene siden tar vare på markedets behov, og på den andre siden sikrer dem selv en trygg og voksende posisjon i fremtiden.

Tøff konkurranse har brakt oss et kvantesprang fremover på relativt kort tid. Den situasjonen vil – og bør – vedvare. I løpet rundt *Web services* er det så langt ingen vinnere eller tapere, men aktive deltagere. ■