IT-oplossing moet aansluiten bij organisatie

HANDVAT BIJ KEUZE

In de vele artikelen, rapporten en definities die over BPM gaan wordt steeds meer het standpunt ingenomen dat BPM een 'management discipline' is. Het zou gaan om vraagstukken zoals: hoe krijg ik de organisatie tot een procesdenkende organisatie?

Door André Boonzaaijer, Frits Bussemaker, Linda Terlouw

Vaardigheden zoals met name change management zijn dan van belang. Immers, zoals Forrester onlangs in een rapport heeft aangegeven: in 85 procent van alle bedrijfsprocessen zit nog steeds menselijk handelen. Een IDC-rapport van augustus 2007 gaat zelfs zover om te stellen dat een organisatie BPM kan invoeren zonder BPMS. Met andere woorden, de management-discipline kan ingevoerd worden zonder de specifieke ondersteunende technologie.
In de discussie over BPM moet men echter niet uit het oog verliezen dat in de praktijk nagenoeg altijd de onderliggende ICT-applicaties geïmplementeerd dienen te worden. Steeds meer applicaties zijn standaard en kunnen als softwarepakket gekocht worden. Echter, er blijven nog vele, vaak complexe, applicaties over die als maatwerk gebouwd moeten worden. Wanneer er een keuze gemaakt moet worden voor maatwerk dan zijn op dit moment de twee alternatieven Java EE en .NET. De spanning tussen fervente Java EE en .NET programmeurs kan soms hoog oplopen en lijkt vooral een interessant vraagstuk tijdens een LAN-party onder het genot van een pizza en een ruime hoeveelheid cola. Het is niet gemakkelijk om je in deze vete te mengen zonder dat de technische termen je om de oren vliegen. Laat u hier niet door afschrikken, want technisch gezien verschillen de platformen weinig van elkaar. Het doel van dit artikel is de BPM professional een handvat te geven om de discussie te voeren over de zaak waar het echt om draait: in hoeverre sluit de geboden oplossing aan bij de organisatie? Hiermee kunt u een betere keuze uit deze alternatieven maken als u toch ook maatwerk moet leveren.

Het vraagstuk

Vanuit het opkomende SOA-gedachtegoed luidt de redenatie dat de implementatie van services verborgen is en dat de

Illustratie: Leon van Leeuwen.

Vanaf het moment dat de twee platformen bestaan is er sprake van een continue wedloop

onderliggende technologie nauwelijks meer relevant is. Orkestraties beschrijven in welke volgorde en onder welke condities geautomatiseerde services aangeroepen worden. Of de onderliggende systemen, die de services aanbieden, gebouwd zijn in Java EE, .NET, Cobol of C++ doet niet meer ter zake.
In theorie kan een organisatie dus elk mogelijk ontwikkelplatform in gebruik hebben en de betreffende applicaties met elkaar laten communiceren. Een erg praktische keuze is dit echter niet, omdat voor elke technologie vaardigheden voor de ontwikkeling en het beheer binnen de organisatie nodig zijn, een ontwikkelstraat ingericht moet worden en licentiekosten verschuldigd zijn. Daar bovenop moet continu de afweging worden gemaakt in welke technologie wat gebouwd wordt. Deze nadelen, samen met het toenemende aantal organisaties dat een 'buy-before-make' beleid hanteert, leiden ertoe dat veel organisaties hun maatwerkontwikkeling consolideren tot één platform. De twee meest populaire alternatieven voor de ontwikkeling van nieuwe applicaties zijn Java EE en .NET. Hoewel ook nieuwe technologieën opkomen vanuit de web-applicatiewereld, zoals Ruby-on-Rails, zijn dit (nog) geen serieuze alternatieven voor complete, complexe bedrijfsapplicaties.

Een objectieve vergelijking maken valt niet mee. Veel experts op dit gebied hebben 'gevoelsmatig' een voorkeur en informatie over de verschillen tussen beide platformen is doorgaans technisch van aard. Dit artikel beschrijft slechts in vogelvlucht de verschillen tussen Java EE en .NET, zonder op technische details in te gaan. De ontstaansgeschiedenis van beide platformen wordt kort besproken, de oorzaken van de verschillen komen aan bod en wordt de vergelijking gemaakt. Tenslotte volgen het toekomstbeeld en de conclusies.

Ontstaansgeschiedenis

Halverwege de jaren negentig bracht Sun de programmeertaal Java uit voor het bouwen van client-side applicaties. Dit type applicaties wordt (vrijwel) volledig op de machine van de gebruiker uitgevoerd. De kerngedachte achter Java was dat het type hardware en type besturingssysteem niet meer ter zake doet door gebruik te maken van een virtuele machine. Deze virtuele machine is een abstractielaag tussen de programmeertaal Java en de machinetaal van het specifieke hardware platform die de gedachte van 'compile once, run anywhere' moest realiseren. Hoewel hier in de praktijk wat haken en ogen aan zitten, was de gedachte duidelijk: een strakke scheiding tussen programmeertaal aan de ene kant en hardware/besturingssysteem aan de andere kant.
In dezelfde periode bracht Microsoft haar eerste server-gebaseerde ontwikkelomgeving uit. Een server-side applicatie wordt voor verreweg het grootste deel op een server uitgevoerd en op de machine van de gebruiker bestaat alleen een 'dun schilletje' (meestal in de vorm van een web browser). Waar Microsoft zich voorheen alleen richtte op de client-side omgevingen, betrad zij nu een bredere markt. De technologieën van Microsoft vormden op dit moment nauwelijks een uniform geheel; voor elk deelgebiedje binnen de software-ontwikkeling had Microsoft een aparte oplossing.
Door de opkomst van internet en de toenemende vraag naar server-gebaseerde ontwikkelomgevingen kon Sun niet achterblijven en bracht zij enige jaren later J2EE uit dat in volgende versies hernoemd is naar Java EE. Wanneer Microsoft in 2000 haar verschillende technologieën consolideert en uitbrengt in .NET 1.0, is de basis voor de concurrentiestrijd gelegd. In .NET is het mogelijk met verschillende programmeertalen te werken, waarvan C# de meest gebruikte is. Beide platformen bieden de mogelijkheid zowel client-side als server-side applicaties te bouwen en de overeenkomst in syntax van Java en C# is zelfs voor niet-programmeurs opvallend.
Vanaf het moment dat de twee platformen naast elkaar bestaan is er sprake van een continue wedloop, waarin dan weer Java EE dan weer .NET komt met nieuwe ontwikkelingen – op de voet gevolgd door de ander.

Verschillen

Zoals gesteld zijn beide platformen qua programmeertaal vrijwel identiek. Door het over en weer 'lenen' van technieken, bieden beide platformen vandaag de dag vrijwel dezelfde mogelijkheden. Voor het doorgronden van de echte verschillen is het nodig de technische bril af te zetten. Dan wordt namelijk duidelijk dat de commerciële structuren afwijken en dat de oorsponkelijke kijk op software-ontwikkeling andere 'normen en waarden' meegeeft.
De commerciële structuren zijn in één oogopslag duidelijk: Microsoft is de enige drijvende kracht achter .NET en Java EE kent vele spelers. Na de start van Sun met Java/J2EE zijn IBM, BEA, Oracle en vele andere partijen aangehaakt en ook de open source-wereld levert haar bijdrage. Het verschil van de oorspronkelijke kijk op software-ontwikkeling ligt wat

genuanceerder. Grofweg kan gezegd worden dat Java EE vooral werkt vanuit een 'technology push' en .NET vanuit een 'demand pull'. Java is in de academische wereld vanaf eind jaren negentig zeer sterk omarmd. Dit heeft de laatste tien jaar gezorgd voor een ontzettende vooruitgang in fundamenteel nieuwe technieken. Vooruitgang binnen het .NET platform ontstaat voornamelijk naar aanleiding van klantvraag en vraag uit de bestaande ontwikkelgemeenschap.
De verschillen tussen beide platformen die hierna aan de orde komen hebben een directe oorzaak en gevolg relatie met de bovengenoemde punten.

Tools en frameworks

Als een organisatie kiest voor .NET volgt vrijwel automatisch een set aan tools en standaardcomponenten (ook wel 'frameworks' genoemd) die ingezet moeten worden voor de bouw van applicaties. Bij een keuze voor Java EE ligt dit geheel anders. Op de markt bestaat een grote verscheidenheid aan tools en standaardcomponenten, waarbinnen een organisatie nogmaals een keuze moet maken. Een prijskaartje voor een Java EE oplossing is niet gemakkelijk te bepalen vanwege het verschil in onder andere licentiekosten en supportkosten.
Een voorbeeld hiervan is de Integrated Development Environment (IDE), de ontwikkelomgeving waarin de programmeur zijn applicaties maakt. Voor .NET is de enige keuze Microsoft Visual Studio. Microsoft heeft hier nog een aantal gratis varianten van uitgebracht, maar daarmee houdt de keuze op (er zijn open source alternatieven als Sharpdevelop in ontwikkeling; omdat we deze oplossingen echter nog niet in grootschalige productiesituaties zijn tegengekomen laten we ze hier daarom buiten beschouwing). Voor Java EE is de keuze enorm, variërend van open source oplossing tot zware commerciële oplossingen. Java EE architecten komen dus voor keuzes te staan die ze aan de ene kant meer vrijheid geven dan .NET architecten, maar waarbij ze ook het gevaar lopen op het verkeerde paard te wedden. Niet elk framework of elke tool is namelijk een lang leven beschoren. De keerzijde van een keuze van het 'veilige pad' binnen .NET is de afhankelijkheid van Microsoft voor nieuwe tools en frameworks.

Heterogeen versus Wintel

In de inleiding van dit artikel is duidelijk gemaakt dat de kerngedachte van Java EE de onafhankelijkheid is van de programmeertaal, het onderliggende besturingssysteem en de hardware. De verschillende software vendors hebben dit idee in stand gehouden; de tools worden dus geleverd voor verschillende besturingssystemen en verschillende typen hardware. Hierbij moet wel worden opgemerkt dat niet elke software vendor het volledige spectrum aan besturingssystemen en hardware platformen bedient. Microsoft, daarentegen, levert haar .NET-tools alleen voor Windows op zogenaamde x86 machines. Elementen uit nieuwere versies van .NET worden vaak niet in alle productieversies van Microsoft Windows ondersteund. Microsoft heeft er als dominante partij op besturingssysteemgebied geen belang bij haar software-ontwikkelproducten ook voor andere besturingssystemen aan te bieden. Dit is uiteraard geen probleem voor een echt Microsoft-huis, maar als een organisatie net een investering heeft gedaan in een groot Unix server-park dan zal de balans vrij snel richting Java EE doorslaan.

Java EE	.NET
Grote keuze aan tools en standaardcomponenten.	Beperkte keuze aan tools en standaardcomponenten.
Er kan in taal geprogrammeerd worden, maar programma's werken op meerdere besturingssystemen.	Er kan in meerdere talen geprogrammeerd worden, maar programma's werken alleen op het Windows besturingssysteem.
De omgeving ondersteunt een modelgerichte manier van werken en is daarmee minder toegankelijk voor een beginnend programmeur.	De omgeving ondersteunt een grafisch gerichte manier van werken en is daardoor erg toegankelijk voor een beginnend programmeur.
Vooral geschikt voor heterogene omgevingen.	Sluit goed aan op het Windows besturingssysteem en andere Microsoft-producten.

De commerciële structuren zijn in één oogopslag duidelijk

Woordenlijst

1. Java Enterprise Edition (EE). Een ontwikkelplatform, gebaseerd op de programmeertaal Java, oorspronkelijk afkomstig van Sun en nu door meerdere leveranciers ondersteund.
2. Microsoft .NET. Een ontwikkelplatform van Microsoft, waarbij onder andere de programmeertalen C# en Visual Basic kunnen worden gebruikt.
3. Ruby-on-Rails. Een programmeertaal die zich puur richt op het bouwen van websites (waar Java en .NET zich richten op de gehele geautomatiseerde informatievoorziening binnen een bedrijf), heeft op dit moment een flink 'hype gehalte'.
4. Client-side applicatie. Een applicatie waar het grootste deel van de verwerking aan de kant van de computer van de gebruiker wordt uitgevoerd.
5. Server-side applicatie. Een applicatie waar het grootste deel van de verwerking aan de kant van de computer van de server wordt uitgevoerd en de gebruiker slechts werkt met een 'dunne schil' (Internet browser).
6. Integrated Development Environment (IDE). De tool waarin een programmeur zijn applicaties maakt, zoals bijvoorbeeld Eclipse en Netbeans (open source) en Oracle JDeveloper en IBM WebSphere Studio Application Developer (commercieel).
7. Foundation technologieën. Een verzameling aan nieuwe technologieën die Microsoft heeft uitgebracht voor workflow, applicatie-integratie en gebruikersschermen.
8. Community. De gemeenschap van programmeurs voor een bepaalde technologie.
9. Enterprise Service Bus. Een tool die het mogelijk maakt om applicaties volgens servicegeoriënteerde principes met elkaar te laten communiceren.

Fundamentele modellering of RAD

Tot voor kort werd .NET als het meest geschikt gezien voor kleine, grafisch sterke applicaties en Java voor de grote, complexe server-gebaseerde applicaties met een relatief simpele gebruikersinterface. Hierin is echter een verschuiving zichtbaar. Het afgelopen jaar is Microsoft met een aantal zogenaamde Foundation technologieën gekomen, die juist ook het grootschaligere server gebaseerd ontwikkelen moeten vergemakkelijken.
Binnen de Java community ontstaan steeds vaker frameworks die ook de ontsluiting richting de voorkant makkelijker maken. De nadruk op het sneller kunnen ontwikkelen van complexere gebruikersinterfaces komt binnen de Java EE community steeds sterker naar voren.

De programmeur

IT is mensenwerk. Een applicatie ontwerpen en bouwen is een combinatie van het bedenken van creatieve oplossingen en het nauwkeurig en gestructureerd uitwerken van deze bedachte oplossing. In dit proces kunnen uiteraard meerdere iteraties plaatsvinden. 'De' Java EE programmeur en 'de' .NET programmeur vliegen dit proces allebei op hun eigen wijze aan, althans als we kijken naar de generatie die al wat langer meeloopt. De .NET programmeur heeft meestal geen IT-gerelateerde opleiding genoten en is vaak doorgegroeid vanuit jarenlange ontwikkeling met Microsoft-technologieën als Visual Basic of MS Access, vanwege de toegankelijkheid ook wel 'click-and-it-works' omgevingen genoemd. De Java EE programmeur is meestal meer theoretisch gericht. Hij is langer bezig met ontwerp en zal dus veelal vooraf dingen uitwerken en pas later implementeren, ook wel gesteld als: het verschil tussen vooruit inparkeren (pragmatisch te werk gaan en een snelle oplossing hebben) en achteruit inparkeren (van tevoren goed nadenken over mogelijke problemen in de toekomst). Nu .NET een volwassen omgeving is, kiezen veel jonge programmeurs die theoretisch sterk onderlegd zijn voor een carrière als .NET programmeur. De keuze voor de carrière in Java EE of .NET is in principe willekeurig geworden.

Integratiemogelijkheden

In de inleiding is reeds genoemd dat technologie steeds verder 'onder de motorkap' komt te liggen in service georiënteerde architecturen. Hoewel technologie zeker niet de grootste uitdaging is voor de realisatie van een service georiënteerde architectuur, moet de ontsluiting van applicaties niet als triviale zaak afgedaan worden. Moderne integratiemiddelen (ESB's, ofwel Enterprise Service Bus) zijn helaas nog niet compleet volwassen en de geboden functionaliteit verschilt nogal per variant. Helaas is het nog steeds zo

dat de technologie waarin de ESB zelf gebouwd is, ook nog van belang is voor de aansluiting op de bestaande omgeving. In een echte heterogene omgeving scoren de ESB's op basis van Java EE, zoals bijvoorbeeld IBM WebSphere en Oracle ESB nog steeds beter dan de Microsoft-oplossing BizTalk. Microsoft biedt daarentegen een steeds betere integratie aan voor verschillende typen applicaties op het Windows-besturingssysteem met haar Foundation technologieën. De vraag blijft wel hoe waardevol dit is als er mainframes, AS/400'en en UNIX machines in gebruik zijn binnen de organisatie. En is een dusdanig sterke koppeling tussen ontwikkelplatform en/of integratieplatform en besturingssysteem wel gewenst?

Het maken van een keuze

Het maken van een keuze tussen beide platformen is niet eenvoudig en erg situatieafhankelijk. Verreweg het belangrijkste criterium voor de keuze is de bestaande IT-omgeving. Organisaties zullen hun huidige IT-omgeving nooit in één klap willen vervangen. En met IT-omgeving bedoelen we hier niet alleen de hardware en software, maar ook de mensen. Bij een zogenaamd Microsoft-huis is .NET meestal de logische keuze, terwijl een heterogene omgeving meer baat heeft bij Java EE. De omscholing van mensen is doorgaans geen gemakkelijk proces. Hoe comfortabel zij zich voelen bij een nieuw platform heeft enerzijds te maken met hun programmeerervaring in verschillende omgevingen en anderzijds met hun conceptuele denkvermogen over het programmeren. .NET is toegankelijker voor de zogenaamde 'omgevingsspecifieke' programmeur. Voor programmeurs, die een brede IT-kennis hebben, zal het uiteindelijk niet zo veel uitmaken in welke omgeving ze werken.

Tot vrij kort geleden was het zo dat Java EE vooral geschikt was voor de grote organisaties en .NET voor het MKB. Op dit moment maakt .NET ook een opmars binnen de grote organisaties. Hierbij moet wel vermeld worden dat weinig grote organisaties volledig overgaan op .NET. Het betreft meestal een ontwikkelstraat naast Java EE. Alleen een ontwikkelstraat op basis van .NET voor een grote organisatie is op dit moment dus nog een minder bewezen oplossing dan het alternatief gebaseerd op Java EE.

Toekomstbeeld

De ontwikkelingen in beide platformen lijken momenteel harder te gaan dan ooit. Zowel Microsoft als Sun, BEA, Oracle en andere timmeren aan de weg en de nieuwe versies lijken elkaar in steeds rapper tempo op te volgen. Zoals we op meerdere punten al eerder aangaven lijkt het alsof beide platformen elkaar sneller tot grotere hoogten doen groeien – een sterk staaltje van marktwerking.
Toch is er in de Java EE wereld de afgelopen maanden een ontwikkeling geweest die kan leiden tot een trendbreuk. Zowel IBM, BEA als Oracle hebben na jaren van het braaf volgen van de SUN-standaarden gekozen voor een minder afhankelijke positie. De open source wereld lijkt ook zijn eigen pad te kiezen. Het lijkt er daarom op dat het Java EE platform van één samenhangend geheel uiteen dreigt te vallen in een aantal verschillende stromingen. Gecombineerd met de trend dat het marktaandeel van .NET de laatste jaren een stijgende lijn volgt, kan het wel eens betekenen dat het .NET platform een inhaalslag gaat maken en een dominante positie gaat verwerven op de markt. Met de visie van Microsoft op de ondersteuning van zaken als SOA en BPM, gecombineerd met de ondersteuning van breed gedragen standaarden, zien we dat .NET een volwassen alternatief begint te worden voor de grootschaliger enterprise applicaties waar voorheen Java EE heer en meester was. De grote vraag is dus: weet de Java EE wereld haar positie te behouden?

Conclusies

Wanneer stukken programmacode uit een Java omgeving en een .NET omgeving naast elkaar gezet worden, is er nauwelijks een verschil te ontdekken. Nieuwe ontwikkelingen komen ook vrijwel tegelijkertijd uit. In essentie bestaat er vanuit technisch oogpunt dus geen of weinig verschil tussen beide platformen. Toch is dit geen aanleiding om een willekeurige keuze te maken, zie de vergelijking in de tabel. Omdat deze verschillen ook een organisatorische impact kunnen hebben, is juist de blik van de BPM-professional gewenst bij het maken van een keuze. Dus laat u niet afschrikken, maar kom voor uw belangen op wanneer de discussie toch over maatwerk gaat!

André Boonzaaijer, Frits Bussemaker, Linda Terlouw
Ir. A. Boonzaaijer (aboonzaaijer@sogyo.nl) is werkzaam als Senior Consultant bij Sogyo.
Ir. L. Terlouw (linda.terlouw@ordina.nl) is werkzaam als Solution Architect bij Ordina Integration Solutions. Daarnaast is zij als promovendus verbonden aan de TU Delft.
Ir. F. Bussemaker (frits.bussemaker@ordina.nl) is werkzaam als Senior Management Consultant bij Ordina Consulting. Daarnaast is hij voorzitter van BPM-Forum Nederland.