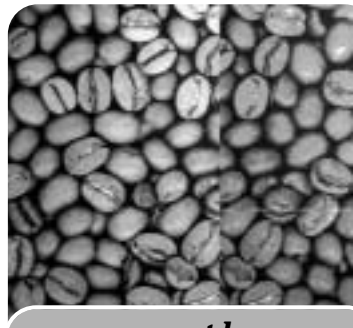
Steeds meer bedrijfsprocessen worden naar het internet getild. Om de gewenste functionaliteit snel ter beschikking te krijgen, is een moderne en solide software development omgeving een vereiste. Momenteel zijn er twee snel opkomende platforms die pretenderen aan dergelijke eisen te kunnen voldoen: Microsoft .Net en Java 2 Enterprise Edition (J2EE). Beiden worden aangeduid als enterprise platforms.



Niet slechts een keuze

Een vergelijk: Microsoft .Net en J2EE

Zoals de titel boven dit artikel al doet vermoeden, is dit geen ordinaire platformvete zoals er al zovelen zijn uitgevochten. In plaats van het voeren van een heilige oorlog kunnen we ons beter realiseren dat het waarschijnlijk slechts een kwestie van tijd is voordat bedrijven zullen beseffen dat ze niet louter een keuze hebben gemaakt, maar dat ze beide platforms binnen hun organisaties hebben ingebed. Dit artikel is een vergelijking van beide platforms en een opsomming van hun belangrijkste overeenkomsten en verschillen.

DEFINITIE Om de vraag te kunnen beantwoorden, wat eigenlijk een enterprise platform is, zullen we een aantal criteria moeten definiëren voor het begrip 'enterprise'. Denk met name aan niet-functionele zaken, zoals bijvoorbeeld: beschikbaarheid, betrouwbaarheid, beveiliging, schaalbaarheid, multi-platform- en multi-device ondersteuning. Ook kan gedacht worden aan het ondersteunen van bepaalde (open) standaarden, het kunnen toepassen van moderne ontwikkelmethodieken en de beschikbaarheid van een breed pakket aan (ontwikkel)tools. Beide platforms worden ontwikkeld met een dergelijke omgeving als doel. Uit Gartner onderzoek (zie figuur 1) is gebleken dat binnen nu en twee tot drie jaar beide platforms zullen uitgroeien tot toonaangevend op dit vlak.

AND THE WINNER IS... Zal er uiteindelijk een platform als overwinnaar uit de strijd komen? Dat lijkt hoogst onwaarschijnlijk. Een voor de hand liggend scenario is dat bedrijven met een beperkte diversiteit aan besturingssystemen eerder een keuze maken ten gunste van .Net, terwijl bedrijven met een grotere diversiteit aan besturingssystemen eerder zullen kiezen voor Java. Grote bedrijven ontkomen waarschijnlijk niet aan een wereld waarin beide platforms een rol zullen spelen. In plaats van ruzie maken over 'of – of' kan die energie beter worden besteed aan het uitdenken van oplossingen voor 'en – en'.

WAT MAAKT HET VERSCHIL? In de definitie van beide platforms zit een interessante tegenstelling: multi-OS versus multi-language. Java predikt sinds jaar en dag 'Write once, run anywhere'. Dat wil zeggen: schrijf je applicatie in Java en diezelfde applicatie zal vervolgens ongewijzigd kunnen draaien op ieder operating system (OS) waarop een Java Virtual Machine beschikbaar is. Dat aantal is in de loop der jaren behoorlijk toegenomen.

Microsoft daarentegen gelooft meer in de keuzevrijheid van programmeertaal. Zo kunnen bestaande kennis en ervaring worden ingezet voor het nieuwe platform; .Net zal draaien op ieder OS zolang de naam maar begint met 'Windows'. Hebben we hiermee dan meteen het grote verschil tussen beide platforms te pakken? Ja, maar met de kanttekening dat er in principe ook andere programmeertalen dan Java zouden kunnen verschijnen, die bij het compileren bytecode zullen opleveren. Ten aanzien van het .Net platform blijven er geruchten bestaan over mogelijke ports van .Net naar een ander

> Het is merkwaardig dat leveranciers specifieke uitbreidingen op standaards uitbrengen die niet portable zijn

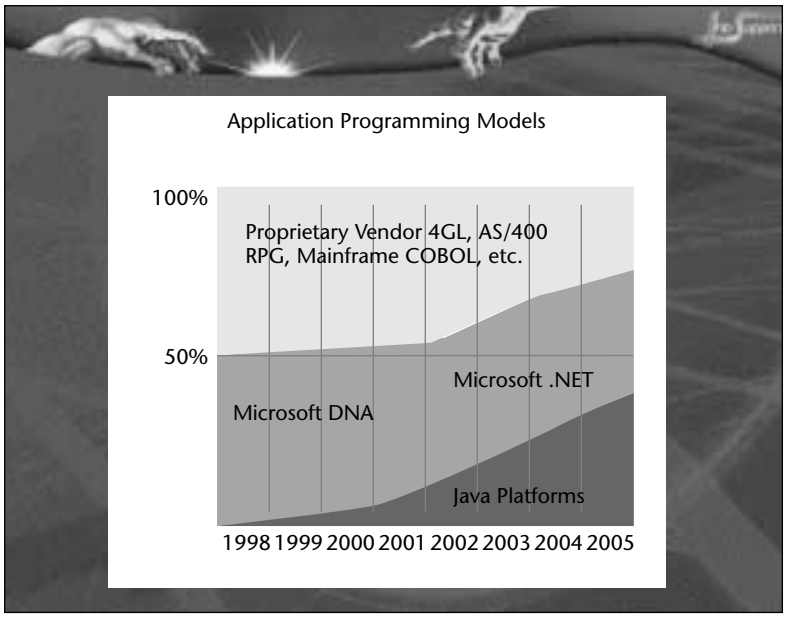
OS. Denk bijvoorbeeld aan Corel Linux. Microsoft is immers voor een groot deel eigenaar van Corel en uit Redmond komen vaker verassingen.

SPEERPUNTEN Hoewel de insteek verschillend is, hebben beide platforms hetzelfde doel: het presenteren van een compleet en robuust development framework, bedoeld voor het bouwen en in productie brengen en houden van betrouwbare en schaalbare enterprise systemen. De overige speerpunten bestaan voor beiden uit: web services, gedistribueerde component modellen en data services.

De basis van het .Net Framework ligt in het Common Type System (CTS). Het CTS is kort gezegd een verzameling van afspraken waaraan de programmeertalen voor .Net zich moeten houden zodat gecompileerde code met elkaar kan samenwerken. De Common Language Runtime (CLR) is de executieomgeving van .Net waarin de gecompileerde code zal draaien. Programmeertalen die zich conformeren aan het CTS leveren bij compilatie zogenaamde Intermediate Language (IL) op en deze wordt in de CLR geladen en geëxecuteerd. Het motto van .Net is dan ook: "ontwikkel het in een willekeurige taal en draai het in .Net". Is dat in de praktijk dan ook zo? Het gaat in ieder geval niet zo makkelijk als het lijkt. Bestaande programmeertalen moeten worden aangepast om IL te kunnen opleveren en programmeurs moeten leren rekening te houden met de regels van het CTS. Vaak zal het er op neerkomen dat er dus toch het nodige zal moeten worden bijgeleerd, dus waarom dan maar niet meteen over naar de .Net taal bij uitstek: C#.

OPEN OF GESLOTEN? Ondanks het feit dat Microsoft pretendeert een open platform aan te bieden is het in de kern juist relatief gesloten. Maar daarin schuilt juist ook de kracht van .Net. Onder de noemer 'open' heeft Microsoft een aantal zaken gedeponeerd bij standaardorganisaties zoals bijvoorbeeld ECMA (www.ecma.ch). Zo zijn de taal C# en de standaard bibliotheken hier ondergebracht. Er dient echter een onderscheid gemaakt te worden tussen de 'openheid' van specificaties en die van het proces om tot die specificaties te komen. Het eerste is bij Microsoft wel het geval; belangrijke onderdelen worden naderhand tot standaard verheven. Voor het proces geldt dat de invloed die je als commerciële- of onafhankelijke partij kunt uitoefenen op deze 'standaards' niet bepaald groot is. Microsoft bepaalt wat er wel en niet wordt overgenomen. Dat levert voor Microsoft een grote mate van controle op. Deze aanpak is vergelijkbaar met de strategie zoals IBM die in het verleden voerde voor AS/400.

Het grootste voordeel van .Net is met name terug te vinden in het feit dat Microsoft niet alleen het framework heeft opgetuigd, maar ook de eigenaar is van een belangrijk deel van de commerciële producten daaromheen. Van voor tot achter kan een volledig geïntegreerd platform worden neergezet dat zijn gelijke in de markt niet kent. De prijs is die klanten betalen is de enorme 'leap-of-faith' die wordt gevraagd: uiteindelijk is alles afkomstig van dezelfde bron en draait het op slechts een enkel leveranciersgebonden besturingssysteem. Een overstap naar een andere leverancier is waarschijnlijk alleen met veel pijn en moeite te maken.



FIGUUR 1. Voorspelling van Gartner (Bron: Gartner)

OPEN OF TE OPEN? Voor Java is niet het besturingssysteem, maar de taal de spin in het web. De Java Language Specification (JLS) vormt dan ook de basis van het platform. Een gecompileerde Java applicatie bestaat uit zogenaamde bytecode. Deze bytecode kan op ieder platform waar een Java Virtual Machine aanwezig is, worden gelezen en geëxecuteerd. Dit gaat echter niet altijd zonder slag of stoot. Voor Java op de server geldt het zo goed als overal, maar met name ontwikkelaars die wel eens cross-platform een user interface hebben gepoogd te ontwikkelen, weten wat ik bedoel.

De evolutie van het Java platform en daarmee J2EE is onderhevig aan het Java Community Process (JCP).

Het JCP is een door Sun opgerichte organisatie die 'spec lead' is voor het verder ontwikkelen van de taal Java en de standaardisatie van zaken daaromheen (www.jcp.org). Waar voorheen alleen Sun de scepter zwaaide is nu een geformaliseerd proces op gang gekomen waarin door vrijwel alle gezaghebbende instanties en bedrijven uit de Java wereld wordt bepaald hoe en waar wijzigingen worden geïmplementeerd. Er is met name aandacht voor de stem van de individuele ontwikkelaar. Die houdt zich tenslotte 'op de werkvloer' bezig met de Java technologie. Op tal van manieren kan de ontwikkelaar zich uitspreken over zaken die door het

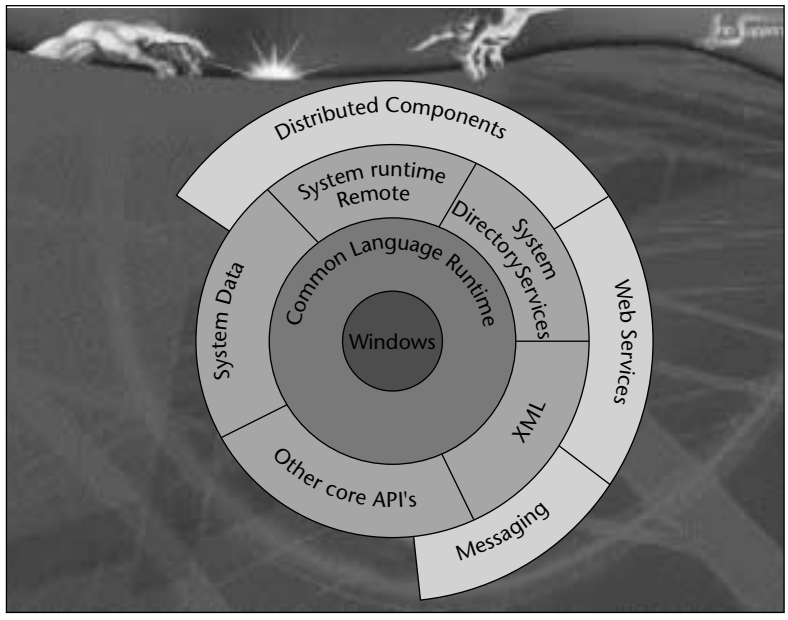
JCP moeten worden aangepakt. Vaak worden uitbreidingen geïnitieerd vanuit Open Source initiatieven zoals bijvoorbeeld Apache of worden door commerciële bedrijven zogenaamde 'best-of-breed' oplossingen aangedragen. Het JCP levert uiteindelijk specificaties af en soms een zogenaamde 'referentie implementatie', maar nooit een commercieel product. De commerciële J2EE leveranciers zullen vervolgens hun best doen om in een bepaalde context hun implementatie van die specificaties zo optimaal mogelijk te laten presteren, om daarmee onderscheidend te zijn naar de markt. Soms gebeurt dit door leverancierspecifieke extensies mee te leveren. Dit laatste levert een paradox op, die ik verderop nog nader zal toelichten. De bottom line voor J2EE is dat aan dit platform de laatste zes à zeven jaar is gebouwd. In tegenstelling tot .Net zijn er voldoende alternatieve producten en leveranciers zijn waar met geringe inspanning naar kan worden uitgeweken. Hoe open kan open zijn?

ARCHITECTUUR Eerder in dit artikel heb ik de speerpunten van beide platforms genoemd, te weten webservices, gedistribueerde componentmodellen en dataservices. We kunnen deze in beeld brengen door een plaatje te schetsen van de architecturen van beide platforms (zie de figuren 2 en 3). Verrassend genoeg zijn de figuren nagenoeg identiek. Het hart van de figuur wordt gevormd door het platform, met daarin voor beide frameworks de bijbehorende mogelijkheden. Bovenop het platform draait een runtime- of executie omgeving. Hoewel verschillend van naam, werken ze in principe identiek. Ze interpreteren of executeren (direct dan wel indirect) de aangeleverde IL- of bytecodes. De ring bovenop de runtime bestaat uit de belangrijkste systeembibliotheken, zoals bijvoorbeeld functionaliteit voor distributie, manipulatie van XML, ontsluiten van data en directory systemen en overige zaken. Deze laag vertegenwoordigt de bouwmaterialen voor de componenten in de buitenste laag. Deze componenten zijn op hun beurt weer de bouwstenen voor applicaties.

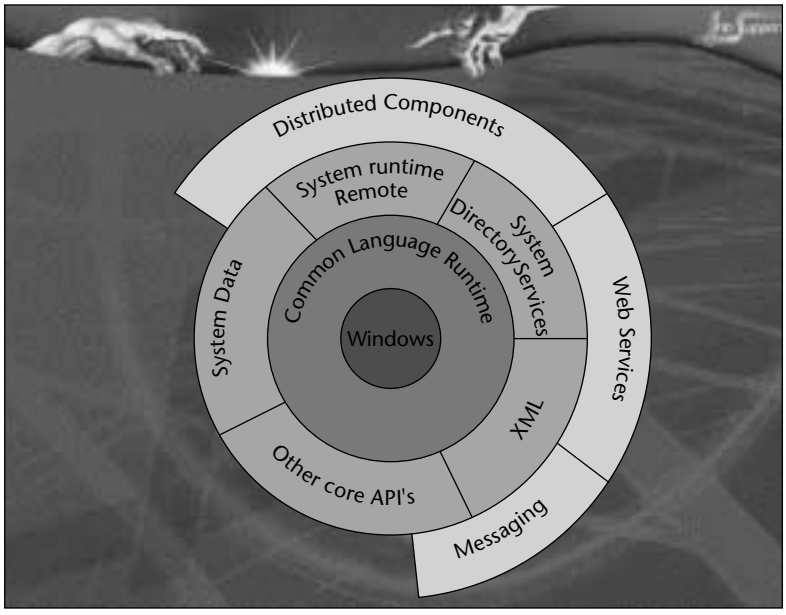
VOOR NIETS GAAT DE ZON OP Uiteindelijk gaat het ook in deze business om geld verdienen. Het is best aardig om daar eens naar te kijken. Voor Microsoft geldt: hoe meer .Net ontwikkeling er plaats vindt, hoe meer Windows en .Net Server licenties er verkocht zullen worden. Dat .Net allerlei programmeertalen ondersteunt die niet van Microsoft zijn, is commercieel gezien geen enkel probleem. Uiteindelijk moeten er toch licenties worden aangeschaft voor het besturingssysteem - en dat is van Microsoft. De taalonafhankelijkheid van het . Net framework biedt weinig meerwaarde, want uiteindelijk ontwikkelt iedereen in C#. Microsoft heeft voorlopig nog geen intentie om .Net multi-platform te maken, maar derden zien hier misschien een gat in de markt.

STERKE CONTROLE Voor J2EE klinkt het allemaal te mooi om waar te zijn: volledig open en keuzevrijheid tot op productniveau. Mocht Java echter onder sterke controle van Sun blijven, dan zal Sun er op de een of andere manier beter van willen worden. De redenering is als volgt: hoe meer J2EE ontwikkeling er plaats vindt, hoe meer leveranciers er zullen komen die vervolgens weer licentiegeld in het laatje brengen. Sun is eigenaar van de rechten van Java en zal daar dus geld voor opstrijken. Daarnaast hoopt Sun te profiteren van de naamsbekendheid, om zodoende commerciële producten, zoals een Application Server of een geïntegreerde ontwikkelomgeving te slijten. De hoop die misschien aanvankelijk heeft geleefd dat iedereen voor zijn Java applicaties ook Sun hardware zou aanschaffen heeft men maar laten varen. Sun streeft nu naar volledige openheid in combinatie met het JCP. De kans dat Sun ooit de rechten van Java zal opgeven is echter bijzonder klein. Zowel Microsoft als Sun bundelen tegenwoordig de applicatieserver technologie gratis mee met het besturingssysteem. Het is echter vanwege Java's openheid ook heel goed mogelijk om te kiezen voor de applicatieserver technologie van een andere partij. In het

> Grote bedrijven ontkomen waarschijnlijk niet aan een wereld waarin beide platforms een rol zullen spelen



FIGUUR 2. Microsoft .Net architectuur



FIGUUR 3. Java 2 Enterprise Edition architectuur

geval van een commerciële implementatie dient dan vaak nog een (flink) bedrag neergeteld te worden. Er zijn ook commerciële leveranciers of Open Source organisaties die de applicatieserver technologie gratis weggeven. Maar dan zullen klanten bijvoorbeeld weer voor technische ondersteuning moeten betalen.

ONTSLUITEN VERSUS NEGEREN Binnen beide frameworks zijn een aantal mogelijkheden voor het integreren van de twee werelden. Van oudsher is er natuurlijk de mogelijkheid om een 'message bus' te creëren waardoor twee verschillende applicaties middels een afgesproken berichtformaat, data kunnen uitwisselen. Een andere mogelijkheid is om gebruik te maken van zogenaamde 'bridges'. Een bridge is een encapsulatielaag om een bestaande component of applicatie heen, waarin de vertaling van het ene framework naar het andere framework wordt verricht. Zo zijn er bijvoorbeeld bridges waarin een Enterprise JavaBean zich voor kan doen als een COM component en andersom. Een modern alternatief wordt gevormd door 'webservices'. Webservices zijn diensten die op het (inter)netwerk worden aangeboden en die met behulp van een implementatie-onafhankelijk protocol (SOAP) communiceren. Zowel .Net als J2EE bieden ondersteuning voor dit fenomeen. Een stap op de goede weg is de oprichting van de Web Services Interoperability Organization (WS-I). De WS-I is een organisatie die zich bezighoudt met het platform-, operating system- en programmeertaalonafhankelijk maken van webservices. Bij dit initiatief hebben zich al meer dan honderd bedrijven aangesloten.

DE KLEINE LETTERTJES 'Ieder voordeel heb z'n nadeel', om met een beroemde voetballer te spreken. Dat geldt ook voor .Net en J2EE. Bij .Net is er een hele sterke afhankelijkheid van een enkele leverancier. Weliswaar sluit Microsoft de deur niet helemaal, toch zal er voor derden slechts een kleine rol weggelegd zijn. Voor Microsoft is het nieuw om niet direct competitie te kunnen voeren tegen een bedrijf, maar tegen een specificatie waarachter tal van bedrijven staan. Het is interessant om die strijd te volgen. Met .Net slaat Microsoft een volledig nieuwe koers in. Er is momenteel geen migratiepad naar een andere omgeving beschikbaar. Wel is er integratie met bestaande legacy systemen, door middel van koppelingen zoals COM-Interop, ODBC, MSMQ bridges en EDI/XML.

De 'openheid' van J2EE is een paradoxaal gegeven. Een open specificatie met veel inspraakmogelijkheden voor zowel de ontwikkelaar als de commerciële partijen creëert een dilemma. Aan de ene kant moeten standaarden waterdicht gedefinieerd zijn, zodat er gemakkelijker ontwikkeld kan worden tussen producten onderling. Tegelijkertijd is het dan voor commerciële partijen niet meer interessant om een product op de markt te brengen, wanneer er van dat product al tien varianten bestaan die weinig meerwaarde ten opzichte van elkaar hebben. Daar zal de industrie uiteindelijk een modus in moeten vinden.

Een tweede punt is dat het voor leveranciers moeilijk wordt om een grote groep ontwikkelaars of klanten aan zich te binden als de applicatie met een enkele muisklik geport kan worden naar een product van een andere leverancier. Het is dus merkwaardig dat leveranciers allerlei specifieke uitbreidingen op standaards uitbrengen die niet portable zijn. Het gebruik ervan is voor eigen risico.

En tenslotte: dat er keuzemogelijkheden en alternatieven zijn in de Java wereld is een goede zaak. Toch zal er altijd ook een vorm van selectie moeten plaatsvinden op het moment dat een bepaald product nodig is in een project. Kortom: veel keuzes, zowel positief als negatief.

CONCLUSIE Concluderend kunnen we stellen dat er inderdaad twee volwaardige enterprise platforms bestaan, die voor een groot gedeelte hetzelfde zijn. Beiden hebben een specifieke eigen stijl. Op een aantal punten zijn radicaal andere keuzes gemaakt, maar op het conceptuele vlak verschillen ze niet eens zoveel. Beide platforms zijn nog relatief jong. De huidige stand van zaken is dat er de komende maanden nog het nodige op stapel staat. Hoe de platforms zich houden in de echte wereld is een kwestie waar we in de loop der tijd achter zullen komen.

Bert Ertman
Ertman is als ICT architect werkzaam bij Info Support B.V.