Microsoft .NET: De toekomst is dichterbij dan u denkt

Het zal velen niet ontgaan zijn dat Microsoft op het Forum 2000 in Redmond een nieuwe visie en strategie heeft geïntroduceerd, Microsoft .NET genaamd. Maar wat is Microsoft .NET eigenlijk? Wat gaat Microsoft .NET betekenen? Hoe ver zijn deze visie en strategie al werkelijkheid geworden? Vragen die wellicht niet voor iedereen te beantwoorden zijn. Dit artikel gaat in op deze en andere Microsoft .NET vragen.

Ronald van Vugt

Microsoft .NET (in het vervolg .NET genoemd) is zo omvangrijk dat het niet in één zin is samen te vatten. .NET is in een drietal gebieden onder te verdelen:

- een *visie* op hoe software gaat evolueren om optimaal gebruik te maken van het internet en de grote variëteit van devices die in het dagelijkse leven gebruikt worden;
- een *platform* dat de ontwikkelaar en de infrastructuurspecialist ondersteunt om de .NET-visie te implementeren;
- een *application hosting service* die diensten over het internet aanbiedt, gehost vanuit Microsoft datacentra om de .NET-visie en het .NET-platform te ondersteunen en een permanente geldstroom voor Microsoft te genereren, naast de traditionele licenties.

Op elk van deze drie gebieden zal hierna dieper worden ingegaan. De rode draad is steeds ingegeven door de volgende uitspraak van Microsoft: 'Empower people through great software, any time, any place, and on any device.' Alles in .NET is hierop gericht.

De Microsoft .NET-visie

In de nieuwe wereld van .NET staat de gebruiker centraal. Deze gebruiker zal altijd en overal op elk device de beschikking hebben over de eigen informatie en informatie van anderen. De gebruiker heeft volledige controle over die informatie en bepaalt welke informatie, wanneer en hoe beschikbaar komt. De gebruiker zal dan ook een volledige 'new user experience' ervaren. Om dit te realiseren zijn 'smart devices' nodig. Hierbij moet niet alleen gedacht worden aan een computer of een PDA, maar ook aan bijvoorbeeld een smart phone of een smart radio. Smart devices hebben de volgende kenmerken:

- *Smart about you:* Weet de voorkeuren van de gebruiker; weet waar de gebruiker fysiek is.
- *Smart about the network:* Past zich aan de bandbreedte aan; offline en online bewust; weet welke services beschikbaar zijn.
- *Smart about other devices:* Ziet andere devices en wisselt informatie uit; weet hoe het services kan bieden aan andere devices.
- *Smart about services:* Presenteert de informatie aangepast aan het device; past de invoermogelijkheden aan het device aan.

Om deze kenmerken te kunnen implementeren, heeft een smart device moderne technieken beschikbaar, zoals handschriftherkenning, vingerafdrukherkenning en spraakherkenning. Microsoft is enkele samenwerkingsverbanden met hardwareleveranciers aangegaan om er zorg voor te dragen dat smart devices beschikbaar komen. Een van de eerste beschikbare smart devices is de tablet pc. Dit is een computer ter grootte van een A-viertje (maar wel dikker), waarbij vrijwel het hele oppervlak in beslag wordt genomen door het beeldscherm. Een toetsenbord

is niet aanwezig. Invoer vindt bijvoorbeeld plaats via handschriftherkenning. De verwachting is dat de tablet pc in 2002 beschikbaar zal komen.

.NET zal dus veel meer integreren in het dagelijkse leven van de mens. Naast de gebruikelijke computers en PDA's, zullen uiteindelijk ook gebruikersapparaten voorzien worden van .NET. Internet speelt hierbij een belangrijke rol als centrale plaats van informatie en diensten. Microsoft spreekt dan ook van de volgende generatie internet. Er zijn veel raakvlakken met SUN's visie Sun One. Alleen gaat de visie van SUN nog een stap verder. SUN spreekt van een 'internet of things'. Hierbij is alles en iedereen verbonden met het internet.

Het Microsoft .NET-platform

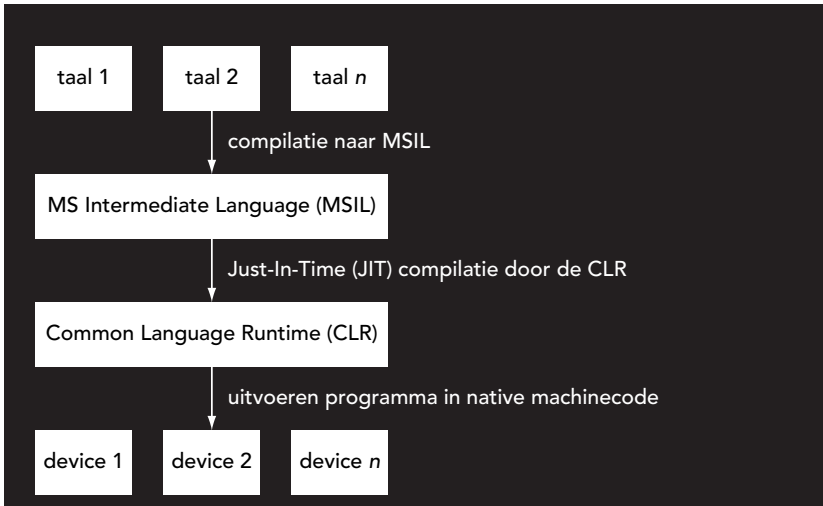
Het nieuwe .NET-platform is fundamenteel anders dan het huidige Microsoft DNA-platform. Het businessprobleem en de voorkeur van de ontwikkelaar vormen de basis voor de programmeertaal die gebruikt gaat worden. Ook moet de ontwikkelaar zich kunnen concentreren op het businessprobleem en niet om allerlei huishoudelijke zaken eromheen, zoals garbage collecting en beveiliging.

In .NET is dan ook gekozen voor een grote keuzevrijheid aan programmeertalen, anders dan het J2EE-platform waar Java de enige programmeertaal is. Op dit moment zijn er ongeveer twintig programmeertalen beschikbaar, variërend van Visual Basic tot Eiffel, maar ook bijvoorbeeld Cobol. Ook is recent J# .NET aangekondigd. Met deze taal kan in de programmeertaal Java op het .NET-platform worden ontwikkeld. Java-programmeurs kunnen hiermee meteen aan de slag met .NET. Overigens zullen programma's gemaakt in J# .NET alleen draaien op het .NET-platform, niet op het J2EE-platform.

In het .NET-platform worden vrijwel alle huishoudelijke taken automatisch afgehandeld, zonder tussenkomst van de ontwikkelaar. Daarnaast hoeft geen rekening meer gehouden te worden met het device waarvoor ontwikkeld wordt, of dit device nu Windows 2000 is (in .NET wordt Windows 2000 ook als een device beschouwd), een PDA of bijvoorbeeld een smart phone. Het 'write-once, run anywhere'-principe is van toepassing, net zoals in Java. Toch is er een belangrijk verschil. In .NET geldt deze uitspraak in het bijzonder voor smart devices en niet voor allerlei platformen, terwijl deze uitspraak voor Java vooral geldt voor allerlei platformen, naast devices die Java-enabled zijn.

Hoe wordt deze keuzevrijheid van programmeertaal en device bereikt? Hiervoor wordt een aantal technieken en componenten gebruikt. Een van de belangrijkste componenten is de Common Language Runtime (CLR). De CLR ontkoppelt het operating system van de programmatuur. De programmatuur maakt gebruik van de CLR, en de CLR communiceert met het operating system. Zodra er een CLR beschikbaar is voor een bepaald device, kan de programmatuur op dit device draaien. Op dit moment is er alleen een CLR beschikbaar voor het Windows-platform. Een CLR voor bijvoorbeeld het LINUX-platform is echter heel goed denkbaar, vooral omdat de Common Language Infrastructure (CLI) bij de ECMA (European Computer Manufacturers Association) als open standaard geregistreerd is. Om er zorg voor te dragen dat elke programmeertaal (die geschikt gemaakt is voor .NET) gebruik kan maken van de CLR, worden alle programmeertalen gecompileerd naar een tussentaal, Microsoft Intermediate Language (MSIL) genoemd. Dus ongeacht welke programmeertaal wordt gebruikt, elk van deze programmeertalen wordt gecompileerd naar MSIL. Deze MSIL lijkt op machinecode, maar is platformonafhankelijk. Op het moment van uitvoeren wordt de MSIL code Just-in-Time gecompileerd door de CLR in native machinecode voor het uitvoerende platform. Dit principe is weergegeven in figuur 1. De gecompileerde code wordt opgeslagen in een cache, zodat bij een volgend gebruik van de code geen compilatieslag meer uitgevoerd hoeft te worden. Op deze wijze wordt device- en programmeertaalonafhankelijkheid bereikt.

Naast de CLR wordt er in .NET ook gebruik gemaakt van gemeenschappelijke base classes. Ongeacht de programmeertaal zal steeds gebruik worden gemaakt van dezelfde base classes. Daarnaast is er het Common Type System (CTS), het gemeenschappelijke typesysteem waaraan variabelen moeten voldoen. Een string in Visual Basic .NET is bijvoorbeeld gelijk aan een string in Cobol .NET. Hierdoor is het mogelijk om op eenvoudige wijze informatie tussen programmeertalen uit te wisselen, zonder dat conversies nodig zijn. Het is overigens ook heel goed mogelijk om verscheidene programmeertalen te gebruiken die gezamenlijk het businessprobleem oplossen. De base classes en het CTS ondersteunen hierbij, maar zeker ook de nieuwe geïntegreerde ontwikkelomgeving, Visual →



Figuur 1. Taal- en device-onafhankelijkheid.

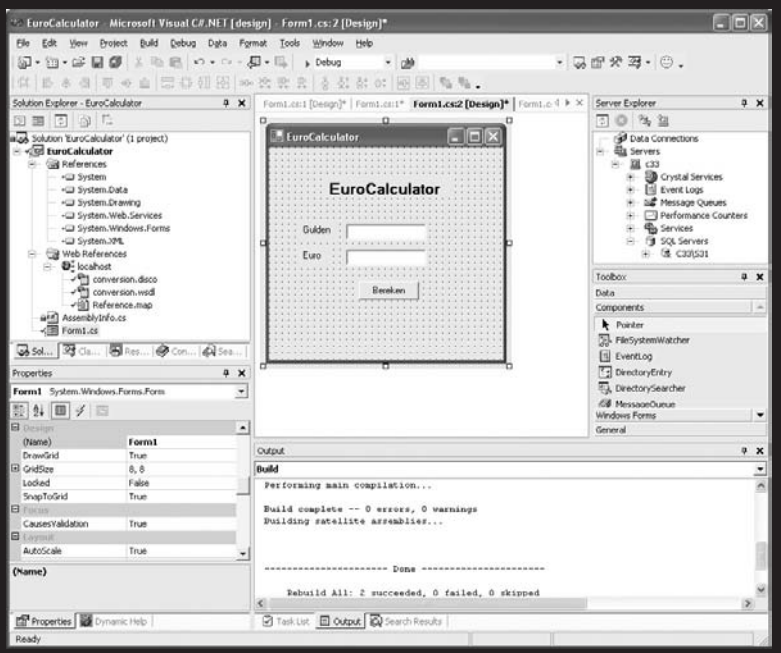
Studio .NET. In deze krachtige ontwikkelomgeving kunnen programmeertalen toegevoegd worden, is debuggen over meerdere talen mogelijk en is deployment van de applicatie mogelijk. De grote kracht is de integratie. Zodra een ontwikkelaar zich de ontwikkelomgeving eigen heeft gemaakt, kunnen alle stappen in het ontwikkelproces vanuit de ontwikkelomgeving uitgevoerd worden. Ook uitbreiding van de ontwikkelomgeving, bijvoorbeeld het toevoegen van een nieuwe programmeertaal, kan naadloos plaatsvinden zonder dat de ontwikkelaar nieuwe onderdelen van de ontwikkelomgeving hoeft te leren. Figuur 2 geeft een indruk van de nieuwe ontwikkelomgeving. Op het eerste gezicht ziet dat er complex uit en er is zeker een leercurve nodig, maar het blijkt een uitstekende en volledige ontwikkelomgeving te zijn. Visual Studio .NET is op dit moment (oktober 2001) alleen beschikbaar in een bèta 2-versie. De verwachting is dat de nieuwe ontwikkelomgeving in februari 2002 gereleased wordt.

Bij de introductie van .NET is ook een nieuwe programmeertaal geïntroduceerd, namelijk C# (spreek uit als C sharp). Voor deze naamgeving is gekozen om de volgende reden. Als goed wordt gekeken naar de sharp (de '#') zijn dit vier plustekens. Eigenlijk staat er dus C++ ++. Een verbeterde versie van C++ dus. Toch is dit maar ten dele waar. C# heeft meer weg van de programmeertaal Java. Hoewel er verschillen zijn, zal de Java-programmeur snel zijn weg kunnen vinden in C#. Overigens zal de C++-programmeur zich C# ook relatief snel eigen kunnen maken. Voor de Visual Basic-programmeur geldt dit in veel mindere mate. Daarvoor zijn de verschillen in beide programmeertalen te groot. Om de overgang voor de Java-ontwikkelaar of het migratietraject van Java naar .NET zo klein mogelijk te maken, heeft Microsoft JUMP to .NET (JUMP staat voor Java User Migration Path) geïntroduceerd. Naast ondersteunende tools wordt ook een automatisch conversieprogramma geleverd dat Java-code omzet naar C#-code. .NET is volledig object georiënteerd. Dit geldt ook voor alle .NET-programmeertalen. Voor ontwikkelaars die hiermee weinig ervaring hebben zal de leercurve naar .NET aanmerkelijk langer zijn dan voor ervaren objectgeoriënteerde ontwikkelaars.

Binnen alle programmeertalen is ook de foutafhandeling aanmerkelijk verbeterd. In het huidige ontwikkelplatform van Microsoft zijn er verschillende methoden voor foutafhandeling. In .NET worden alle fouten afgehandeld met exceptions. Hierdoor is het mogelijk om de code die de foutafhandeling verzorgt, los te koppelen van de code die het businessprobleem oplost. Hierdoor wordt het schrijven, maar vooral ook het onderhouden van code, vereenvoudigd.

Deployment

Op het huidige Windows-platform zal iedereen wel eens problemen hebben gehad met deployment van applicaties (het 'uitrollen' van applicaties). Voordat een applicatie uitgevoerd kan worden, moet er veel gebeuren. Bestanden moeten in allerlei directories geplaatst worden, DLL's moeten geregistreerd worden, er moet voor gezorgd



Figuur 2. De nieuwe ontwikkelomgeving, Visual Studio .NET.

worden dat dezelfde DLL's met verschillende versienummers elkaar niet in de weg zitten (de zogenaamde 'DLL-hell') en zo zijn er nog meer aandachtspunten. Om in het huidige Microsoft-platform ervoor te zorgen dat een applicatie automatisch gedeployed kan worden, wordt er een script gemaakt om dit te realiseren. Vooral in grote organisaties met veel applicaties zijn er dagelijks veel automatiseringsdeskundigen bezig met het maken van deze scripts, vooral tijdens migraties zoals van Windows 98 naar Windows 2000. Voor alle applicaties zal een script gemaakt of aangepast moeten worden.

In .NET is de deployment gereduceerd tot 'XCOPY-deployment'. Dit houdt in dat er een of enkele bestanden zijn die in een directory geplaatst worden. Er zijn geen registraties of andere handelingen nodig. Hierna kan het programma gestart worden. Binnen .NET worden deze bestanden assembly's genoemd. In een assembly is zowel de code (in het Microsoft Intermediate Language-formaat, dus platformonafhankelijk) als meta-informatie opgeslagen. De CLR zal hier gebruik van maken. Door middel van een manifest kan een assembly zichzelf volledig beschrijven. Ook het versiebeheer is met een assembly volledig afgedekt. Verschillende versies van dezelfde assembly kunnen naast elkaar draaien. Automatisch wordt de juiste versie van een assembly gebruikt. Ook biedt een assembly bescherming tegen misbruik. Zo start een applicatie niet op als niet alle benodigde assembly's met de juiste versies aanwezig zijn. Als een assembly is gemodificeerd door een virus, zal de applicatie niet starten. Deployment in .NET is absoluut een grote verbetering ten opzichte van de huidige deploymenttechnieken.

Webservices

Het internet is een heterogene mix van diverse platformen, applicaties en data. Ook Microsoft accepteert met .NET eindelijk deze realiteit. De wereld is heterogeen en zal nooit alleen met het Microsoft-platform draaien. Microsoft zoekt dan ook aansluiting met al bestaande platformen. Ook maakt Microsoft gebruik van standaarden, zit Microsoft zelf in diverse standaardisatiecommissies en worden eigen Microsoft-technieken geregistreerd bij diverse instanties. Dit is een radicale maar noodzakelijke wijziging in het beleid van Microsoft. Microsoft staat immers bekend om de geslotenheid en de eigen 'standaarden'. Ook Microsoft realiseert zich nu dat de volgende generatie internet, en daarmee .NET, alleen kan groeien als alle grote wereldspelers zich aan standaarden houden. De marketingstrategie van Microsoft is er nu op gericht om het Microsoft-platform het beste platform te laten zijn om deze standaarden te implementeren.

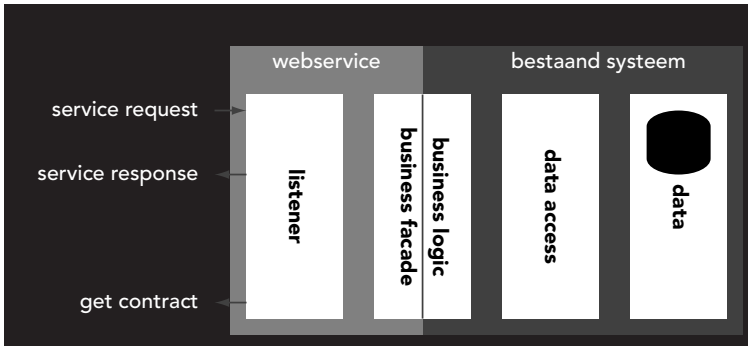
Een van de pijlers van .NET zijn webservices. Webservices vormen een goed voorbeeld van een standaard die ook door Microsoft is omarmd. Webservices gaan steeds belangrijker worden, niet alleen in .NET, maar op alle platformen. Webservices zullen ook de brug slaan tussen de verschillende platformen; integratie tussen verschillende platformen zal vooral met webservices plaatsvinden. Volgens Gartner zal eind 2002 75 procent van de bedrijven met meer dan $100 miljoen omzet gebruik maken van webservices. In 2003 zullen daar de bedrijven met minder omzet bijkomen.

Een webservice biedt op een gestandaardiseerde manier de mogelijkheid om informatie te ontsluiten en te integreren in bestaande bedrijfssystemen, gebruikmakend van XML, de wereldwijde standaard voor data-integratie. Daarnaast kunnen webservices gebruikt worden voor Enterprise Application Integration (EAI). Alle grote wereldspelers (zoals Microsoft, IBM, SUN) en een tweehonderdtal kleinere spelers hebben gezamenlijk webservices als standaard geaccepteerd. Dit is de belangrijkste kracht van webservices, naast het gebruik van XML.

Hoe werkt een webservice? Naast de gebruikelijke code om de businesslogica en dataopslag te implementeren, wordt gebruik gemaakt van een SOAP-listener (Simple Object Access Protocol). De SOAP-listener is de voorkant van de webservice. Dit is in figuur 3 weergegeven. Via SOAP/XML kan vervolgens de webservice worden aangeroepen. Ook kan het contract worden opgevraagd. In dit contract, gerepresenteerd door een XML-document, is de webservice-interface precies beschreven. Hierdoor kan de webservice eenvoudig in een ontwikkelomgeving worden geïntegreerd. Ook in Visual Studio .NET is deze integratie erg eenvoudig. Nadat een webservice is toegevoegd aan Visual Studio .NET, kan de webservice beschouwd worden als een local class, terwijl de webservice aan de andere kant van de wereld geïmplementeerd kan zijn.

In de visie van Microsoft zal de ontwikkeling van nieuwe programmatuur anders verlopen. Steeds meer zullen belangrijke basiscomponenten beschikbaar komen in de vorm van webservices. Die kunnen rechtstreeks als bouwblok in Visual Studio .NET gebruikt worden. De Component Based Development-technieken worden zo in een andere vorm weer gebruikt. De ontwikkelaar zal zich steeds meer bezighouden met het selecteren van de juis-

→



Figuur 3. De opbouw van een webservice.

te webservices en alleen de lijm tussen deze webservices nog zelf bouwen.

Om webservices verder te ondersteunen, zijn er UDDI-diensten (Universal Description Discovery and Integration). Deze UDDI-diensten kunnen beschouwd worden als een gouden gids voor businessapplicaties en diensten. Met UDDI is het mogelijk om een webservice te zoeken die een bepaalde functionaliteit aanbiedt. Meer informatie is te vinden op www.uddi.org. Ook binnen Visual Studio .NET wordt gebruik gemaakt van UDDI. Als een webservice wordt toegevoegd aan de ontwikkelomgeving, kan via UDDI gezocht worden naar de webservice die de juiste functionaliteit aanbiedt.

Het gebruik van webservices biedt verschillende voordelen:

- platformonafhankelijk;
- integratie van een webservice met eigen bedrijfsapplicaties is eenvoudig;
- gestandaardiseerd, gebaseerd op XML en SOAP;
- integratie van content en applicatie van verschillende bedrijven is eenvoudig;
- bestaande (achterliggende) infrastructuur kan grotendeels gebruikt worden, zodat er geen verlies van huidige investeringen is;
- flexibel en schaalbaar concept;
- firewall friendly; omdat een SOAP/XML-call gebruik maakt van HTTP als transportmedium, wordt alleen port 80 gebruikt.

Application hosting service

Een belangrijk onderdeel van de visie van Microsoft is any place. Om dit te bereiken is het belangrijk dat informatie centraal is opgeslagen, en wat is er meer centraal dan het internet? Daarom introduceert Microsoft de .NET My Services (voorheen 'Hailstorm Services' genoemd). Deze .NET My Services zijn SOAP/XML-webservices op het internet die persoonlijke informatie van een gebruiker kunnen opslaan en ophalen, zoals contactpersonen, e-mail, agenda, documenten, favoriete websites, beschikbare devices, portemonnee, profiel en dergelijke. Deze informatie kan gedeeld worden met andere gebruikers, maar ook met bedrijven. Op deze manier kan bij een website automatisch de voorkeur van de gebruiker worden opgevraagd en kunnen formulieren automatisch worden ingevuld. Authenticatie vindt plaats met Microsoft Passport. Iedere gebruiker die de .NET My Services wil gebruiken, moet een Microsoft Passport hebben. Eenmaal ingelogd met Microsoft Passport, zijn alle diensten die zijn aangesloten op Microsoft Passport (dit kunnen ook niet-Microsoftdiensten zijn) toegankelijk zonder nog een keer extra in te loggen.

Naast het zelf opvragen van informatie komt er ook een pushmechanisme beschikbaar, .NET Alerts genoemd. De gebruiker kan hierbij aangeven welke informatie ontvangen moet worden. Deze informatie wordt via e-mail of MSN Messenger verstuurd zodra er iets te melden is wat de gebruiker interessant vindt. Bedrijven die al belangstelling hebben getoond zijn bijvoorbeeld eBay, Nasdaq, de reissite Expedia, de financiële nieuwsdienst CNBC en nog ongeveer twintig, vooralsnog alleen Amerikaanse, bedrijven. Onduidelijk is nog welke Nederlandse bedrijven zullen deelnemen.

Overigens is vooral Microsoft Passport niet onomstreden. Veel organisaties hebben al protest aangetekend wegens het feit dat met Microsoft Passport veel persoonlijke informatie van heel veel gebruikers is opgeslagen en wordt beheerd door één organisatie. Ook heeft SUN, samen met een dertigtal andere bedrijven, een alternatief aangekondigd. Bij dit alternatief wordt wel het belangrijkste voordeel bereikt, gebruikersgemak, maar niet de nadelen. Persoonlijke informatie wordt in het alternatief gedistribueerd opgeslagen, waarbij de gebruiker zelf kan aangeven welke informatie waar opgeslagen moet worden. Microsoft Passport in de huidige vorm zal dan ook waarschijnlijk nog enkele wijzigingen ondergaan. De .NET My Services en .NET Alerts worden in 2002 verwacht.

Conclusie

Microsoft .NET lijkt een revolutie te gaan worden in de automatisering en voor het omgaan met automatisering voor de gebruiker. Zover is het echter nog niet. Een aantal belangrijke componenten is nog niet beschikbaar, zoals Visual Studio .NET, smart devices en .NET My Services. Pas vanaf 2002 zullen deze componenten geleidelijk beschikbaar komen. Verder zal .NET de komende jaren evolueren. De verwachting is dat het zeker nog tot 2004 zal duren voordat de visie van Microsoft voor een belangrijk deel implementeerbaar is. Toch biedt .NET op dit moment al een aantal belangrijke voordelen. Vooral het nieuwe ontwikkelplatform en het .NET-framework zijn zeer krachtig. De ontwikkeltool die nodig is om applicaties voor dit nieuwe platform te ontwikkelen, Visual Studio .NET, zal naar verwachting begin 2002 uitkomen. Daarnaast is er nu al een stabiele bèta 2-versie beschikbaar, zodat kennis en ervaring opgebouwd kan worden.

De toekomst is dichterbij dan u denkt. Gaat u mee?

←