Erno de Weerd
is trainer bij het Kenniscentrum van Info Support (http://www.infosupport.com) en verzorgt daar cursussen op het gebied van systeemontwikkeling. Hij is bereikbaar via ernow@infosupport.com

Migreren van Visual Basic 6.0 naar Visual Basic .NET

Ontwerp, bouw en onderhoud van applicaties verbeterd

Al sinds Visual Basic .NET aangekondigd werd, leven verschillende vragen over het migreren van Visual Basic 6 naar Visual Basic .NET Dit artikel zet de belangrijkste veranderingen, mogelijkheden en onmogelijkheden op een rij. Naast Visual Basic .NET is er niet alleen een nieuwe programmeertaal maar ook een nieuw programmeerplatform bijgekomen. Wanneer er wordt gekozen voor Visual Basic .NET wordt impliciet ook gekozen voor het Microsoft .NET-platform. Wat is de meerwaarde van Visual Basic .NET ten opzichte van Visual Basic 6.0? Wat zijn de mogelijkheden en gevolgen van het .NET platform? De bestaande Visual Basic 6.0-applicaties kunnen niet met Visual Studio .NET onderhouden worden en maken geen gebruik van het .NET. Wat betekent dit voor deze applicaties? Welke overwegingen zijn er met betrekking tot migratie?

Visual Basic .NET zou er zonder het .NET-platform niet zijn. Het platform voorziet niet alleen in de ontsluiting van de functionaliteit van het operating-systeem, wat deze versie van Visual Basic de krachtigste tot nu toe maakt, maar levert ook een veilige 'run-time'-omgeving aan de applicatie. Een greep uit de extra's:

- Een nieuwe versie van ADO om optimaal data te lezen, te bewerken en in te voeren.
- .NET ondersteunt XML, een platform onafhankelijke standaard voor de overdracht van data.
- Foutafhandeling is gelijk in alle programmeertalen waardoor een applicatie geschreven in de ene taal een run-time-fout kan afvangen die ontstaan is in een andere taal.
- Code Access Security geeft een applicatie de mogelijkheid te controleren welke rechten de applicatie run-time heeft.
- .NET is type-safe. Dit betekent dat run-time bekent is bij het .NET wat het werkelijke type van een variabele of class is. Hierdoor is het platform bijvoorbeeld instaat om verkeerde conversies run-time te detecteren.
- Alle applicaties kunnen meerdere threads starten waardoor het mogelijk is (op een multi-processor-machine) werkelijk meerdere dingen tegelijkertijd uit te laten voeren.

De taal

De taal 'Visual Basic 6.0' is grondig veranderd. De reden hiervoor is dat Visual Basic .NET niet op zichzelf staat, zoals Visual Basic 6.0 dat deed, maar nu, net als ander .NET-talen, gebruik maakt van het .NET-platform. Naast de mogelijkheden die .NET zoals eerder beschreven biedt, stelt het strenge eisen aan de applicaties die er gebruik van willen maken. Dit betekent bijvoorbeeld dat het gebruik van Variants niet meer is toegestaan omdat dit de kans op fouten, die pas 'run-time' zichtbaar worden, erg groot maakt. Ook is 'objectoriëntatie' in Visual Basic .NET opgenomen, om gebruik te kunnen maken van de functionaliteit van het .NET-platform. Overerving, polymorfisme en encapsulatie zijn, dankzij .NET, niet alleen mogelijk binnen de taal maar ook over talen heen. Overerving is niet meer zoals in Visual Basic 6.0 slechts het implementeren van een interface maar werkelijk functionele overerving. De functionaliteit die in de superclass is gecodeerd, is aanwezig in de subclass. Of deze class nu geschreven is in Visual Basic .NET of in een andere .NET-taal; in Visual Basic .NET kan er gebruik worden gemaakt van deze class door er van te erven.

Een paar opvallende veranderingen in de taal:

- Door gebruik te maken van 'Option Strict On' dwingt de compiler af dat impliciete conversies waarbij fouten kunnen

optreden expliciet moeten worden aangegeven. Dit zorgt er voor dat de compiler al bij compilatie waarschuwt voor mogelijke fouten. Zo kan een String niet zomaar daar gebruikt worden waar een Integer verwacht wordt wanneer 'Option Strict' aanstaat. Ook wordt met 'Option Strict On' 'late-binding' niet meer ondersteund. De compiler wil elke functie-aanroep controleren op syntax.

- De String is niet meer Visual Basic-specifiek maar gedefinieerd op het .NET-platform waardoor het vertalen van een String type van de ene naar de andere taal niet meer nodig is.

Migratie

Met Visual Basic 6.0 zijn veel verschillende soorten applicaties gebouwd. Zo zijn er ActiveX DLL's en ActiveX EXE's die beiden COM gebruiken en ondersteunen. Daarnaast zijn er ASP-pagina's, ActiveX documents, DHTML-applicaties en Webclasses voor web-omgevingen gemaakt. En natuurlijk zijn er de 'gewone' Windows-applicaties. Voor al deze typen moeten overwegingen worden gemaakt wanneer over wordt gegaan naar Visual Basic .NET

Stijlen

Wanneer een applicatie gemigreerd wordt verdient het de voorkeur om de applicatie te migreren zonder deze functioneel aan te passen. Het voordeel van deze werkwijze is dat de aandacht volledig gericht kan zijn op de vertaling. Nadeel is natuurlijk dat het direct zichtbare rendement laag is; de applicatie voegt op het eerste oog functioneel niets toe aan de oude. Schijn bedriegt: de applicatie maakt op deze manier al gebruik van de nieuwere (stabielere) omgeving en is op deze manier al klaar om uitgebreid te worden. Uitbreidingen waarbij gebruik kan worden gemaakt van de nieuwe technologie.

Bij het migreren zonder het toevoegen van nieuwe functionaliteit kan goed gebruik worden gemaakt van de documentatie van het oude systeem. Zeker het functionele ontwerp, maar ook grote delen van het technisch ontwerp kunnen hier hun diensten bewijzen. De praktijk leert echter dat deze niet altijd up-to-date of zelfs niet aanwezig zijn. Doordat applicaties vaak 'on-the-fly' zijn aangepast en daarbij de documentatie niet meer is aangepast kan er behoefte zijn aan herontwerp. De kennis van technieken als Refactoring

Migratie hoeft niet in één keer. Sterker nog: soms hoeft het helemaal niet. Niet migreren is altijd een optie. Waar code geschreven wordt, worden fouten gemaakt. Een nieuwe versie introduceert waarschijnlijk nieuwe bugs. Als het risico te groot is en de opbrengst te laag, zou een migratietraject niet in gang gezet moeten worden. Het lastige hierbij is dat een aantal onderdelen van de opbrengst niet of slecht te meten is. Hoe meet je de waarde van de opgedane ervaring van de ontwikkelaar? Als wordt gekozen voor migratie zal de migratie altijd in stappen plaats vinden. Of de tussenproducten nu in productie gebracht worden of niet, het is eenvoudiger om sneller fouten uit de nieuw geschreven code te halen door deel voor deel te migreren, dan wanneer alles in één keer wordt vervangen. Hier zal zeker behoefte aan zijn wanneer meerdere applicaties gebruikmaken van dezelfde componenten. In een typische Microsoft DNA-architectuur zal het één voor één vervangen van de componenten in COM+ makkelijker te controleren zijn dan het in één keer vervangen van de complete middle-tier. Dit is tenslotte ook de manier waarop nu de componenten worden onderhouden. Het grote voordeel van deze manier is dat aan bestaande applicaties snel .NET-functionaliteit kan worden toegevoegd. Het mengen van twee werelden levert echter ook een nieuw probleem op: er moeten nu twee verschillende omgevingen worden ondersteund. Zowel beheer als onderhoud krijgen te maken met een extra complicatie: kennis moeten hebben van twee typen applicaties.

Er zal behoefte zijn aan migratie en het toevoegen van nieuwe functionaliteit. Aangezien het eenvoudiger is opnieuw te beginnen dan het bestaande aan te passen, zal het volledig vervangen van een applicatie veel voorkomen. Het grote voordeel van deze methode is dat het hele ontwerp kan worden aangepast. Het functionele deel kan onder de loep worden genomen en uitgebreid worden met nieuwe eisen. Het technisch ontwerp kan gebruikmaken van de nieuwe mogelijkheden die door .NET geboden worden.

Migratie hoeft niet in één keer. Sterker nog: soms hoeft het helemaal niet. Niet migreren is altijd een optie.

Code

In veel opzichten is de in Visual Basic 6 geschreven code opnieuw te gebruiken.

De Migration Wizard kan code vertalen naar Visual Basic .NET en COM-componenten kunnen door .NET-applicaties worden gebruikt. Het is zelfs mogelijk om .NET-componenten een COM-laagje te geven waardoor in een bestaande applicatie de COM DLL's vervangen kunnen worden door .NET DLL's.

De Migration Wizard kan veel in Visual Basic 6.0 geschreven code vertalen maar dat gaat regel voor regel. De Wizard herkent niet het doel van de code, en kiest dus geen betere .NET-oplossing, maar vertaalt elke regel apart. Zo zullen resize-events van Forms met code om controls op de juiste plaats te houden door de Wizard worden vertaald terwijl .NET in staat is om met een paar eenvoudige properties van de .NET controls hetzelfde effect te bereiken. Helaas moet alle code die betrekking heeft op COM-componenten met de hand worden herschreven; dit kan de Wizard niet. Het meest in het oog springend daarbij is ADO-code. ADO was niet alleen een COM-component, maar de nieuwe versie voor .NET is functioneel zo ingrijpend aangepast dat in veel gevallen opnieuw moet wor-

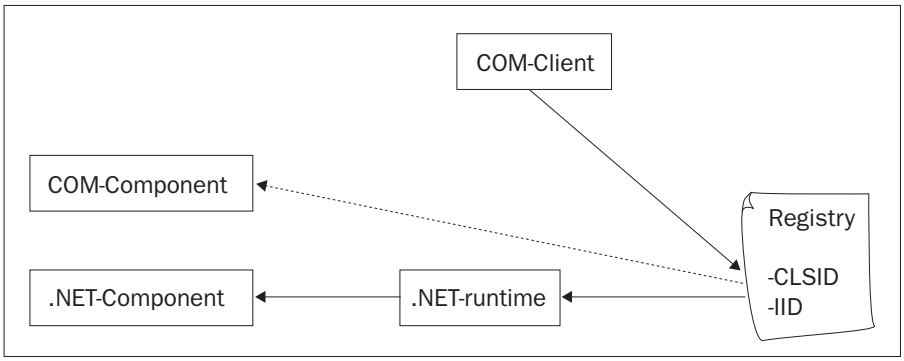
den gekeken naar de code om optimaal gebruik te kunnen maken van de nieuwe mogelijkheden. Met ADO .NET is gekozen voor disconnected recordsets en Forward-only cursoren. Dit betekent dat, afhankelijk van de eerder gekozen data-access strategieën, code soms ingrijpend moet veranderen.

COM

ActiveX controls, gebouwd in Visual Basic 6.0, zijn nog steeds te gebruiken in de Forms-applicaties in .NET. Wel moet extra worden gecontroleerd of de vertaling van data types en de volgorde van events klopt. Mooier is het om in .NET een UserControl te bouwen die alle standaardfunctionaliteiten van de .NET controls direct ondersteunt.

Het benutten van bestaande COM-componenten vanuit .NET is erg eenvoudig; net als in Visual Basic 6.0 kent Visual Studio .NET de mogelijkheid om een referentie om te zetten naar een COM component. In Visual Basic 6.0 had dit tot gevolg dat het type informatie (type library) bekend werd bij de compiler en de intellisense van de editor. Visual Studio .NET zal door het zetten van de referentie een zogenaamde COM-wrapper genereren, die voor de rest van de .NET-applicatie zich voordoet als het COM-component maar niets anders is dan een .NET class die alle aanroepen van functies doorspeelt naar het COM-component.

Het gebruik van .NET-componenten in bestaande COM-applicaties wordt mogelijk gemaakt doordat er in .NET speciale eigenschappen van interfaces, classes, properties en componenten te definiëren zijn. Deze eigenschappen, attributen genaamd, worden door de compiler in de betreffende component opgeslagen en kunnen door de omgeving worden uitgelezen. Op deze manier is het mogelijk om bijvoorbeeld een GUID toe te kennen aan een interface of aan een class, zodat applicaties om een interface ID of class ID kunnen vragen zoals dat bij COM gebruikelijk is. Hierdoor kan een .NET-component zich voordoen als een 'oud' COM-component en deze zo vervangen. De clients die gebruik maakten van het COM-component zullen dan, wanneer ze het COM-component benaderen, uitkomen bij het .NET-component zonder dat ze zelf opnieuw gecompileerd hoeven te worden.



Afbeelding 1.

In afbeelding 1 is te zien dat een bestaande COM-Client uitkomt bij het .NET-Component. Dit werkt doordat de client alleen kennis heeft van de IID's en CLSID's van het COM-Component. Met behulp van de registry werd de client doorverwezen naar het COM-Component maar na juiste registratie van het .NET-Component zal de client bij de .NET-runtime uitkomen. Deze zal op zijn beurt het de functie-aanroepen van de client doorgeven aan het .NET-Component. In code voorbeeld 1 is te zien hoe de .NET class geprogrammeerd moet worden de registratie voor elkaar te krijgen.

ActiveX executables werden tot nu toe om verschillende redenen gebouwd. Soms om een vorm van multi-threading te maken en soms om met DCOM gedistribueerde applicaties te bouwen. Multi-threading is nu in alle .NET-talen, dus ook in Visual Basic .NET, beschikbaar. Gedistribueerde applicaties worden in .NET gemaakt door van Remoting (.NET-specifiek) of van Webservices (platformonafhankelijk) gebruik te maken.

Webapplicaties

Webapplicaties zullen in veel gevallen de eerste applicaties zijn die gemigreerd worden naar .NET. De reden hiervoor is dat alleen de webserver .NET moet ondersteunen en er geen speciale software- of hardware-eisen aan de clients worden gesteld. Bovendien is het bouwen van webapplicaties in Visual Studio .NET zoveel beter te doen dan in Visual Studio 6.0, waardoor het puur vanuit het oogpunt van de bouw al de moeite waard kan zijn om over te gaan. Alle faciliteiten die zo gewoon zijn voor een programmeur van Windows-applicaties zijn nu ook beschikbaar voor de webontwikkelaar. DHTML-applicaties en webclasses zijn niet automatisch te migreren. Nieuwbouw is hier meestal de enige oplossing waarbij het natuurlijk wel mogelijk is om oude en nieuwe webpagina's te combineren.

ASP.NET-pagina's lijken, conceptueel, in veel opzichten op de webclasses uit Visual Basic 6.0. Op de Internet Information Server worden objecten gecreëerd die de uiteindelijke pagina genereren die naar de client verstuurd wordt. Met .NET betekent dit dat het programmeren van de classes gebeurt in een .NET-taal zoals Visual Basic .NET. Deze code zal dan kunnen profiteren van alle voordelen die door .NET geboden worden.

Strategie

Wat en wanneer te migreren? Tot 2008 zal Visual Basic 6.0 door Microsoft worden ondersteund. Veel programmeurs zullen graag willen overstappen naar Visual Studio .NET vanwege de prettig werkende omgeving die alle mogelijkheden van .NET beschikbaar stelt. Vaak zal het migreren van bestaande applicaties veel meer werk opleveren dan het simpel draaien van de Migration Wizard. Als gevolg hiervan zal het vaak beter zijn om opnieuw te kijken naar de applicatie en de beste

oplossing in .NET te bouwen. De .NET-omgeving biedt in veel gevallen een mooiere en betere oplossing dan het regel voor regel vertalen van de applicatie.

Wanneer een applicatie goed functioneert hoeft er geen dwingende reden te zijn om deze nu te migreren naar Visual Basic .NET. Maar wanneer de applicatie, in de toekomst, moet worden aangepast en er functionele eisen moeten worden toegevoegd, kan het verstandig zijn om nu al te bekijken of de nieuwe mogelijkheden van .NET een oplossing kunnen bieden. Op het moment van onderhoud kan dan de complete applicatie worden vervangen.

Technische mogelijkheden van een applicatie zouden nooit de aanleiding moeten zijn om deze te bouwen. Het oplossen van een probleem is een juiste reden om een applicatie te ontwerpen en te maken. Een webservice maken puur en alleen omdat het kan, is nutteloos. Een webservice bouwen omdat de functionaliteit zo op een eenvoudige manier ontsloten kan worden voor een reeks van diverse afnemers is wel zinnig. Natuurlijk is het ook altijd goed om nieuwe technische mogelijkheden uit te proberen met als doel te ontdekken of deze eventuele nieuwe oplossingen mogelijk maken.

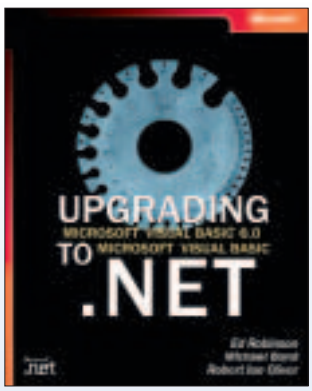
Misschien wel de belangrijkste migratie die plaats moet vinden is het migreren van de mensen die de applicaties gaan ontwerpen en bouwen. De mogelijkheden die door .NET worden geboden zijn enorm krachtig. Bij verkeerd gebruik zal dit leiden tot meer problemen. Ontwikkelaars die al doende de Visual Basic 6.0 hebben geleerd zullen vaak niet weten hoe om te gaan met de nieuwe opties. Opleidingen in programmeren, ontwerpen en objectoriëntatie zullen vaak geen overbodige luxe zijn, maar zelfs een noodzaak om op een juiste manier met .NET om te leren gaan.

Mengen van.NET- en COM

Zoals in het voorgaande al duidelijk blijkt: migratie naar Visual Basic .NET raakt heel veel kanten van het bouwen van applicaties. Voor een succesvolle migratie zal veel kennis nodig zijn van de .NET-omgeving. Er is geen antwoord dat alle vragen oplost. Voor iedere situatie en elke applicatie moet worden nagedacht wat de beste manier is om de applicatie te migreren. Technisch is veel mogelijk, maar het mengen van de .NET- en de COM-wereld zou kunnen leiden tot een applicatie die niet meer te onderhouden is. En niet in de laatste plaats, doordat van beide werelden gedegen kennis nodig blijft om de applicatie te kunnen blijven ondersteunen.

Het .NET-platform en Visual Basic .NET bieden technische mogelijkheden die ontwerp, bouw en onderhoud van de applicaties beter kunnen maken. Het juist toepassen van deze technieken is de uitdaging waar we voor staan.

Microsoft Press



Titel: Upgrading Microsoft Visual Basic 6.0 to Microsoft Visual Basic .NET
ISBN:
Auteur: Ed Robinson, Michael Bond, Robert Ian Oliver

Nuttige internetadressen

- http://msdn.microsoft.com/vstudio/techinfo/articles/upgrade/vbupgrade.asp
- http://msdn.microsoft.com/vbasic/techinfo/articles/upgrade/default.asp
- http://msdn.microsoft.com/vbasic/techinfo/articles/upgrade/vbupgrade.asp
- http://msdn.microsoft.com/vbasic/productinfo/topten/upgrade.asp
- http://www.aivosto.com/vbtips/vbnetmigration.html

Colofon

Microsoft .net magazine

In dit tijdschrift vindt u alle informatie over .NET, het .NET Framework, Visual Studio.NET, ondersteunde talen, technologieën, MSDN, informatieve artikelen van bekende auteurs en interviews met ontwikkelaars en trendwatchers. Microsoft .NET Magazine biedt u technische informatie en toont u de toekomstvisie van Microsoft.

Microsoft .NET Magazine wordt u aangeboden door Microsoft B.V.

Microsoft .NET Magazine is een uitgave van
Microsoft B.V.
Boeing Avenue 30
1119 PE Schiphol-Rijk
Telefoon: (020) 5001500
Fax: (020) 5001004
Internet: www.microsoft.nl

Uitgever
Microsoft B.V.
Derk Wagelaar - Robert Fransen

Projectredactie
Custom Copy + Communicatie, Rotterdam

Redactie en medewerkers
Edward Bakker (CMG), Reinhard Brongers (Big5), Karnoe Dorenbos (Big5), Anko Duizer (Class-A), Teun Duynstee (Macaw), Adrie Geelhoed (CMG), Gerke Geurts (CMG), Jeffrey van Gogh (Microsoft), Robert Fransen (Microsoft), Jeroen Huitink (CGE&Y), Ewoud Jansen (Microsoft), Gijs de Jong (Microsoft), Birgit van de Laar (Microsoft), Dion Olsthoorn (Macaw), Lex Oskam (Microsoft), Michiel van Otegem (ASPNL), Ernst Peter Tamminga (Xcess), Erik S.C. van de Ven (Microsoft), Hans Verbeeck (Microsoft), Marcel de Vries (Info Support), Derk Wagelaar (Microsoft)

Vormgeving
René Peereboom, design@cetera.nl BV, Rijswijk (ZH)

Drukwerk
Wagenaar Communicatie B.V., Epse

Advertentie-index:

Naam bedrijf	Pagina
Borland	72
Cap Gemini/Jaggle	5
Class-A	III
CMG	8
HP/Compaq	II
Info Support	60
Microsoft	42-43
Microsoft	IV
Omnext	66
Sequint	18
Xcess	50