In gesprek met .NET-ontwerper Anders Hejlsberg

OPNIEUW PLEZIER BIJ HET SCHRIJVEN VAN CODE

Anders Hejlsberg is 'Distinguished Engineer' bij de ontwikkeldivisie van Microsoft Corporation. Bekend als de uitvinder van Borland's Delphi programmeertaal is Hejlsberg tevens de hoofdontwerper van C# en vervult hij een sleutelrol in de ontwikkeling van het .NET Framework.

Webservices zijn de volgende stap in automatisering, steunend op dezelfde technologie en infrastructuur als HTML en het web. Maar terwijl de laatste een persoon vereisen aan de ontvangkant, zijn XML en webservices volledig geautomatiseerd. "Het zou echter niet gedaan kunnen worden met de bestaande technologieën", zegt Hejlsberg. "We weten dat het schrijven van gedistribueerde applicaties met de huidige tools, zoals COM, DCOM, CORBA, RMI, enzovoort veel te gecompliceerd is in het gebruik. In de loop van de tijd hadden we veel te veel technologieën ontwikkeld om die nog goed te kunnen beheren. We beseften dat we ze niet op een geleidelijke manier konden vernieuwen, maar COM alleen hier en daar konden 'tweaken'. We moesten alles weg gooien en een heel nieuw platform creëren."

Het gevolg is de ontwikkeling van een abstractielaag die is afgezonderd van de Windows API's. "In de tijd van Win16 en Win32 schreef je code tegen de WinAPI.h en het was net magie om een en ander op de juiste manier te laten werken", zegt Hejlsberg. Library's als MFC, Visual Basic en Delphi's OWL hielpen wel, maar hadden een vervelend neveneffect. "Jouw keuze van een programmeertaal werd ook jouw keuze voor een programmeermodel, en vaardigheden neem je niet zomaar mee naar andere programmeermodellen." Een VB-programmeur zat vast binnen de grenzen van de taal en als hij Visual C++ wilde leren, betekende dat hij het vanaf de grond af aan moest leren.

CLR

Microsoft doorbrak deze beperking met de Common Language Runtime die alle talen ondersteunt en data en code ziet als objecten. "Welke taal je ook gebruikt, je hebt toch toegang tot alle .NET-features. Met de CLR bevrijd je je van COM-programmering terwijl het alles – data en code – omzet in een object dat kan worden gebruikt of overerfd (inheritance) door andere applicaties. De CLR schakelt de noodzaak van een 'registry' uit doordat elk object zelfbeschrijvend is, en hij maakt het mogelijk applicaties met meerdere versies DLL's naast elkaar te verwerken. Er kunnen meerdere versies van dezelfde DLL op de computer bestaan en ze kunnen naast elkaar draaien in twee verschillende applicaties omdat ze zijn 'ge-sandboxed'." De Windows API zal te zijner tijd met pensioen worden gestuurd, zegt Hejlsberg, omdat programmeurs naar de CLR zullen migreren voor de applicatieontwikkeling.

De veranderingen in Microsoft's algemene strategie zijn verreikend. Zowel Bill Gates als Hejlsberg benadrukken dat toekomstige API's die door Microsoft worden gereleased het .NET Framework zullen aanspreken en niet de C-style API-releases die we in het verleden hebben gezien. Hejlsberg: "Het was de praktijk dat C++ programmeurs als eersten de nieuwe API-functionaliteit ter beschikking kregen, terwijl VB-programmeurs vaak moesten wachten totdat Microsoft of een andere partij de API wrapper voor VB-gebruik maakten. Door het .NET Framework aan te spreken, krijgen alle

.NET-talen direct gelijke toegang tot de nieuwe API's."

C#

"Ik ben ontzettend blij met waar we met C# op uit zijn gekomen", zegt Hejlsberg. "C++ ontwikkelaars krijgen te maken met een korte leercurve als zij de taal oppakken. Ze voelen zich empowered en productief, en zeggen zelfs dat ze opnieuw plezier hebben bij het schrijven van code – wat toch heel belangrijk is. Wat doet het er toe als je productief bent terwijl je geen lol hebt? En dat is toch waar het ontwikkelaars om te doen is."

Hejlsberg zegt dat hij nu "denkt over de toekomst van C#", en "wegen om het abstractieniveau opnieuw te verhogen, zonder de favoriete tools van mensen af te nemen." Tevens zegt hij dat een van zijn huidige taken het toevoegen van generics is, een manier om te voorzien in strongly typed collections, daarbij de noodzaak uitschakelend om specifieke types te casten bij de toegang van objecten die zijn opgeslagen in een .NET collection class. Hij karakteriseert de voorgestelde implementatie als een opgeschoonde versie van C++ templates, maar dan wel één die ook wordt begrepen door de infrastructuur. Hejlsberg vergelijkt C# generics met Java-templates, daarbij opmerkend dat ofschoon de Java-implementatie geen modificatie nodig heeft naar de virtuele machine, het ook geen verwerkingsefficiëntie heeft, en alleen maar werkt voor 'reference types' en niet voor 'value types'. Met andere woorden, de 'type casts' zijn eenvoudigweg verborgen, niet geëlimineerd. Daartegenover staat dat de geplande toevoeging van generics aan .NET van toepassing zou zijn op het framework-niveau — dat het framework op intrinsieke wijze de datatypen zou begrijpen en dat de type casts zouden worden geëlimineerd. Als een belangrijk neveneffect geldt dat de toevoeging van generics op het framework-niveau hen potentieel beschikbaar maakt voor alle .NET-talen, niet alleen C#. Hejlsberg wil echter niet zover gaan om te zeggen dat ze in een toekomstige versie van VB.NET zullen worden geïmplementeerd. Hejlsberg: "Elke taal heeft haar eigen persoonlijkheid. C# heeft een hele sterke C++ afstamming die we willen bewaren. VB.NET heeft een breedsprakiger stijl. Ik denk dat het Microsoft VB-team er zeker van wil zijn dat de VB-klant en tevreden zullen zijn met hun product. Uiteindelijk geeft de VB-klant er bijvoorbeeld minder om dat hun keywords dezelfde zijn als die in C#."