.net column

SOA in een .NET wereld

Zoals dat in ons vakgebied gebruikelijk is, zijn er van tijd tot tijd nieuwe ontwikkelingen, die plotseling leiden tot een hype. Dit geldt ook zeker voor het thema Service Oriented Architecture (SOA), dat op dit moment erg in opkomst is. Wat is SOA nu precies? Hoe sluit de Microsoft .NET wereld aan bij het "SOA gedachtegoed"?

Een mogelijke definitie van SOA is de volgende: "Service oriented architecture is een benadering van losgekoppelde, protocol onafhankelijke, op standaards gebaseerde gedistribueerde systemen, waarbij de via het netwerk beschikbare software-resources beschouwd worden als Services."

Verder gelden er voor SOA vier belangrijke aspecten namelijk: grenzen van services zijn expliciet, services zijn autonoom, services delen een schema en contract en de servicecompatibiliteit is gebaseerd op een policy.

Door de jaren heen is er veel discussie geweest over de componentenbenadering voor software ontwikkeling. Met de komst van SOA, hebben grote spelers consensus bereikt over standaards die veel meer gericht zijn op het oplossen van interoperabiliteit op businessniveau, in tegenstelling tot de componentenbenadering, die vooral technisch gericht is. Het 'implementeren' van een SOA is een multidisciplinaire aangelegenheid, waarbij het aandeel van de business, dat van de IT ruim overstijgt (zei tabel).

IT - 10%	SOAP, XML, UDDI, Messaging, Workflow, Connectivity
Business - 40 %	Standaardisatie van data, eigenaarschap van data, definiëren van processen
Communication - 50 %	Hoe kunnen we binnen het bedrijf standaardiseren, buiten bedrijven, over landen heen

Nu zal het waarschijnlijk geen verassing zijn dat webservices een grote rol spelen binnen een service oriented architecture. Microsoft maakt het vanuit de ontwikkelomgeving, Visual Studio .NET, zeer eenvoudig om zowel een web service te bouwen alsmede een web service te consumeren. Je zou dus al snel de conclusie kunnen trekken dat vanuit een technisch perspectief, .NET helemaal klaar is voor SOA.

Alle types die gebruikt worden in de interface van een service, worden vastgelegd in een schema, dat vaak 'canonical schema' wordt genoemd. Het idee hiervan is, dat alle partijen die gebruikmaken van de service een eenduidige definitie hebben van de gebruikte types. Nu werken we vanuit .NET vaak met 'typed datasets', die ook eenvoudig vanuit een web service te gebruiken zijn.

Vanuit het WSDL contract wordt een xsd-schema geïmporteerd, waarin netjes de structuur van de typed dataset wordt beschreven. Het probleem dat zich echter voordoet, is dat Microsoft de typed dataset in het 'Diffgramm' formaat serialiseert, waarbij de structuur van de data niet overeenkomt met de definitie vanuit het geïmporteerde xsd-schema. De Microsoft client-tools kunnen hier prima mee overweg, maar helaas geldt dat niet voor de tools op andere platforms. Een eenvoudige oplossing zou zijn om geen datasets te gebruiken, maar dan gaat ook een groot deel van de productiviteit in .NET verloren. Een mogelijke oplossing van dit probleem is het gebruik maken van de xsd-tool om vanuit de typed-dataset (xsd) een object-model te genereren. Dit object-model kan door de service worden teruggegeven. Het nadeel van deze oplossing is, dat er voor de omzetting van typed datasets naar het object-model en andersom, extra inspanningen moeten worden verricht vanuit code.

Service oriented architecture is sterk in opkomst en heeft als belangrijkste doel om flexibiliteit op business niveau te realiseren. Op dit moment is de Microsoft tooling (nog) niet klaar om goed gebruikt te worden vanuit een SOA gedachte, maar de verwachting is dat dit beter wordt.

Xander Buffart,
IT Architect bij Info Support