Anko Duizer is trainer/coach bij Class-A.

In de ban van Indigo

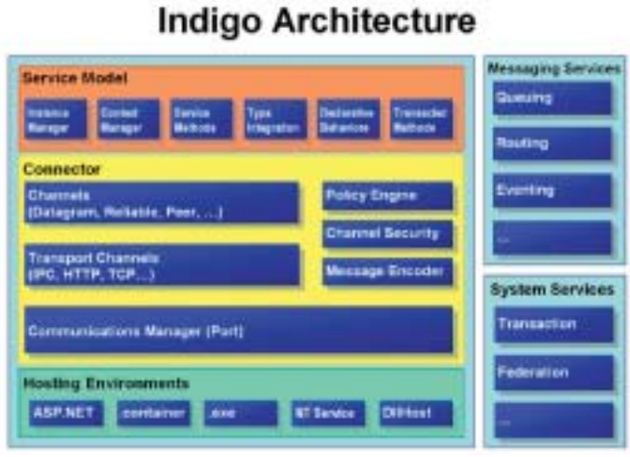
HET NIEUWE MODEL VOOR SERVICEGEORIËNTEERD PROGRAMMEREN

Wat is Indigo? Het is een eenduidig programmeermodel en een communicatie-infrastructuur voor het ontwikkelen van 'connected'-systemen. Een technologie die onderdeel is van de volgende versie van Microsoft Windows, bekend onder de codenaam: 'Longhorn'. Dit artikel bevat een overzicht van wat Indigo inhoudt.

Indigo brengt de slogan 'Make the connection!' in de praktijk. Het is de infrastructuur voor het bouwen van bruggen tussen computersystemen. Indigo maakt verkeer over deze bruggen mogelijk. Dit wordt servicegeoriënteerde programmering genoemd. Om het belang van Indigo duidelijk te maken wordt eerst ingegaan op de problemen met de huidige technologie. Vervolgens wordt de servicegedachte beschreven als antwoord op deze problemen. De rest van het artikel beschrijft hoe Indigo invulling geeft aan deze servicegedachte.

Het software-ecosysteem

Al jarenlang zijn we in de software-industrie op zoek naar hergebruik van code. Zo is hergebruik een basisgedachte van de objectoriëntatie. Inmiddels zijn er veel programmeertalen en technologieën die steunen op de principes van objectoriëntatie. Dit heeft echter niet de mate van herbruikbaarheid gegeven waar we naar op zoek zijn in de software-industrie. Er zijn verschillende redenen waarom objecten nauwelijks herbruikbaar zijn. Ondanks dat heeft objectoriëntatie z'n waarde bewezen op diverse andergebieden. Om de herbruikbaarheid van software te verhogen is er een nieuwe abstractie gekomen boven op objecten: componenten. Maar ook de componenten hebben tot dusver deze belofte niet volledig ingelost.
Zowel objecten als componenten zijn elementaire onderdelen van het huidige .NET Framework. Toch blijft de vraag naar herbruikbaarheid groot. Inmiddels zoeken we herbruikbaarheid op het niveau van computersystemen. Wanneer we bruggen weten te slaan tussen de diverse eilanden van applicaties verbeteren we de herbruikbaarheid enorm. Deze bruggen steunen op het idee van services. In basis gaat de servicegedachte er vanuit dat computersystemen gebouwd worden rondom een set autonome services. Indigo is de set aan technologieën die



Afbeelding 1. Indigo-architectuur

Principe	Uitleg
Grenzen zijn expliciet	In tegenstelling tot objectoriëntatie ziet de servicegedachte het sturen van 'messages' wel als een expliciet model. Het versturen van een 'message' naar een service moet worden gezien als het oversteken van een grens. In werkelijkheid kan dit ook echt zo zijn omdat de andere service op een andere machine of zelfs letterlijk in een ander land draait. Het oversteken van grenzen is duur, dit moet dan ook zo min mogelijk gebeuren. Dit betekent dat de 'messages' zo ontworpen moeten worden dat een eenmalige oversteek voldoende is om het verzoek volledig in te willigen.
Services zijn autonoom	Services zijn verantwoordelijk voor een eigen deployment en versiemodel. Wanneer gebruik gemaakt wordt van services moet men rekening houden met het gegeven dat de andere service het verzoek niet uitvoert, om wat voor reden dan ook. Transacties en queues kunnen goed worden gebruikt bij deze manier van ontwikkeling. Er is niet één grotere macht die alles controleert. De service beslist zelf!
Deel het schema, niet de class	Er kan geen enkele aanname worden gedaan over de technologie waarmee een service is geïmplementeerd. Het is dus niet zinvol om een class, die implementatiegebonden is, met andere services te delen. Tussen services moeten schema's worden gedeeld waarin staat beschreven wat de inhoud is van de 'messages' die worden uitgewisseld. Daarnaast is er een contract nodig waarin services 'afspreken' wat het gedrag is dat verwacht mag worden. Anders gezegd: "Wat voor soort antwoord krijg ik bij welke vraag?". Een voor de hand liggende techniek om te gebruiken is XML in combinatie met XML-schema's.
Compatibiliteit gebaseerd op policies	Expliciete statements met betrekking tot mogelijkheden en eisen van een service legt men vast in een zogenaamde 'policy'. Andere services kunnen op basis hiervan nagaan of het vertoonde gedrag normaal is en wat de eisen zijn waaraan voldaan moet worden om gebruik te maken van de service.

Tabel 1. Basisprincipes van service-oriëntatie

Indigo onderdeel	Servicemodel
Servicemodel	Dit is het primaire onderdeel waar ontwikkelaars gebruik van maken wanneer ze Indigo gaan programmeren. Het servicemodel biedt een declaratief programmeermodel. Dit is gebaseerd op attributen, zoals bijvoorbeeld transacties die worden gedeclareerd in het .NET Enterprise services-model. Het servicemodel is geïmplementeerd als “managed api” en dus te gebruiken in iedere CLR-programmeertaal. Het servicemodel ondersteunt twee contracten: *datacontract* en *servicecontract*. Een datacontract is te vergelijken met een XML Schema. Het datacontract brengt structuur aan aan informatie en beschrijft hoe een CLR-type extern gebruikt kan worden. Het servicecontract beschrijft het patroon van boodschappen welke services met elkaar uitwissen. Het betreft hier gedrag. Uiteindelijk geeft het servicemodel een programmeermodel om inkomende verzoeken in de vorm van een message te koppelen aan CLR-code. Dit is vergelijkbaar met het huidige ASMX-programmeermodel. Het servicemodel is verantwoordelijk voor instantie en sessiemanagement. Daarnaast biedt het model attributen om de beveiliging van de service te specificeren en te vragen om frameworkdiensten. Het is niet noodzakelijk om assemblies met de consument van een service te delen. Het model lijkt wat dat betreft opnieuw sterk op ASMX. Het is dan ook eenvoudig om bestaande ASMX gebaseerde webservices om te zetten naar de Indigo-technologie.
Connector	De connector is het kloppende hart van de Indigo-technologie. Ook de connector is geïmplementeerd als een set van managed classes. De doelstelling van de connector is het verzorgen van veilige en betrouwbare “message” gebaseerde connectiviteit. De connector verzorgt I/O op basis van het SOAP-model. Indigo maakt het mogelijk om messagegebaseerde applicaties te bouwen onafhankelijk van de manier van transport en het doelplatform. De connector is ontworpen om een kleine “footprint” te hebben en een uitstekende “throughput” zodat de connector uitermate geschikt is voor performancegevoelige systemen. Firewalls, proxies en gateways zijn geen hindernis voor de connector. Het uitgangspunt is dat een dergelijk element altijd aanwezig is in de infrastructuur. Dit in tegenstelling tot de huidige technologie waarmee connecties verwezenlijkt kunnen worden. Uiteindelijk is de connector de implementatie van Microsoft van een aantal WS-*specificaties zoals deze worden opgesteld door het W3C. Dit zijn onder andere de specificaties WS-ReliableMessaging, WS-Security, WS-Trust en WS-Federation.
Hosting-omgevingen	Met Indigo moet het mogelijk zijn om alle CLR-code aan te bieden via een servicegeoriënteerd model. Daarom wordt er een diversiteit aan mogelijke hosting-omgevingen geïmplementeerd. Om op runtime gebruik te maken van het servicemodel en de connector moet de code ergens draaien. Dit gebeurt in de zogenaamde hosting-omgeving. Wanneer er gebruik wordt gemaakt van ASMX is ASP.NET bijvoorbeeld de hosting-omgeving. Overigens is ASP.NET ook een mogelijke hosting-omgeving voor Indigo naast een exe, dllhost en een NT Service. Met de komst van Windows Longhorn wordt er een nieuwe hosting-omgeving geïntroduceerd als onderdeel van Avalon (de code naam voor het presentatie-subsysteem van Windows Longhorn, namelijk container. De verwachting is dat de meeste Indigo services gebruik gaan maken van ASP.NET als hosting-omgeving. Dit heeft als voordeel dat er een eenduidig beheer en configuratiemodel ontstaat op de “middle-tier”
Messaging en System services	Indigo maakt zelf ook gebruik van de service-gedachte. Als onderdeel van de technologie biedt het namelijk services aan die ieder een kan gebruiken in een applicatie. De meest in het oog springende service op dit moment is de transactie-functionaliteit. De transactie-functionaliteit gaat schuil onder de naam System.Transactions en implementeert onder andere het WS-AtomicTransaction-protocol. System.Transaction heeft als doelstelling om het gebruik van transacties eenvoudig en eenduidig te maken over het gehele platform. Technologieën zoals Microsoft SQL Server, ADO.NET, MSMQ en de DTC worden ondersteund. Er wordt zowel een expliciet als een impliciet transactiemodel ondersteund. Wanneer het noodzakelijk is, kan er vanuit de code volledige controle worden verkregen over een transactie. In de meeste gevallen zal het impliciete model echter uitstekend voldoen.

Tabel 2. De verschillende onderdelen van Indigo

vanuit Microsoft invulling gaat geven aan de servicegedachte. Het moet op een eenvoudige manier mogelijk zijn om applicaties te voorzien van services die eenduidig met elkaar kunnen communiceren.

Wat zijn de basisprincipes van service-oriëntatie?

Een service is een programma dat interactie maakt door het uitwisselen van boodschappen (messages). In Indigo vormt een set services gezamenlijk een systeem. Services worden gebouwd voor een lange tijd; de beschikbaarheid en stabiliteit zijn kritische succesfactoren. Het systeem moet om kunnen gaan met nieuwe toekomstige services.

Om dit te bereiken wordt voor servicegeoriënteerde ontwikkeling binnen Microsoft uitgegaan van de volgende principes.

In afbeelding 1 is de architectuur schematisch weergegeven. Binnen de verschillende onderdelen zijn de componenten weergegeven, die in de praktijk worden geïmplementeerd door één of meer managed classes. De verschillende onderdelen zijn beschreven in tabel 2.

Het programmeren van Indigo

De programmeur gaat in de toekomst het meeste gebruik maken van het servicemodel. Dit deel van Indigo is ontworpen om gebruikt te worden vanuit managed code. Juist dit servicemodel is op dit moment nog in ontwikkeling. De interfaces en classes die nu worden aangeboden in de huidige Windows Longhorn build gaan veranderen in de nabije toekomst.

Zoals in voorbeeldcode 1 duidelijk wordt, zorgt het servicemodel er voornamelijk voor dat het mogelijk is om via attributen:

* aan te geven wat het datacontract is (Dit is de representatie van CLR-types naar de buitenwereld).
* hoe de service aangesproken kan worden, en
* wat het verwachte resultaat is.

Op hoofdlijnen gaat het programmeermodel er op deze manier uitzien, maar de attribuutnamen en de details kunnen nog verschillen. Het connectormodel is op dit moment in een stabiele toestand. Er worden geen grote veranderingen meer verwacht. Dit deel van Indigo kan nu worden bekeken en gebruikt om ervaring op te doen. Het is juist dit onderdeel van Indigo dat een goed inzicht verstrekt in de interne werking van het Indigo connectie-mechanisme. Ook in de toekomst kunnen developers dit onderdeel gebruiken wanneer ze maximale controle willen hebben.

In afbeelding 2 wordt de werking weergegeven van de connector. De connector is gebaseerd op vier concepten: messages, ports, channels en services. De *message* is data in een SOAP-formaat.

Wat is indigo?

Indigo is een .NET Framework-technologie voor het bouwen en hosten van servicegeoriënteerde systemen. Indigo is een technologie die voortkomt uit het rijtje: COM, COM+, MSMQ, .NET Remoting, ASMX en .NET Enterprise services. Indigo is een evolutie van al deze technologieën maar dan gebaseerd op een nieuw uniform programmeermodel. Het programmeren van Indigo-applicaties doe je op basis van een "managed api". Het is dus mogelijk om met iedere willekeurige .NET-programmeertaal Indigo-applicaties te ontwikkelen.

De doelstelling van Indigo is om het op een eenduidige manier mogelijk te maken om zowel interne als externe communicatie te verwezenlijken. Het moet mogelijk zijn om systemen over het internet met elkaar te koppelen, maar ook om een RPC-verzoek tussen twee processen op één machine via Indigo plaats te laten vinden.

Omdat systemen aan elkaar gekoppeld kunnen worden doet Indigo geen enkele aanname over de implementatietechnologie van het systeem "aan de andere kant". Een belangrijke doelstelling is interoperabiliteit.

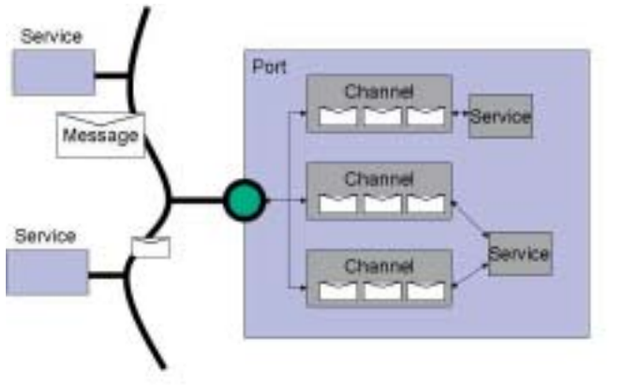
Zolang de message voldoet aan de SOAP-standaard is het mogelijk om ieder willekeurig verzoek dan wel antwoord via een message te versturen. Vervolgens is er een *port* die dient als endpoint voor services. Een message wordt naar een port verstuurd en vanuit de port kan een message worden verstuurd. De port is onafhankelijk van het transportprotocol. Messages bewegen tussen ports.

Achter een port bevindt zich een *service*. De service is een willekeurig stuk code. Indigo denkt over services als een stuk code dat kan worden uitgevoerd op een willekeurige manier. Een policy geeft de mogelijkheden van de service aan en een contract beschrijft de mogelijke message-uitwisselingen. Tussen de port en een service is er een *channel*. Voor één service kunnen er meer channels worden gedefinieerd, die op hun beurt weer ieder verschillende eigenschappen kunnen hebben. De channel kan bijvoorbeeld gebruik maken van speciale services om iets te doen met beveiliging. Op basis van bovenstaande concepten is het mogelijk om connectiviteit toe te voegen aan .NET-applicaties op een eenduidige manier. Het is een eenvoudig model, maar met zeer krachtige mogelijkheden. De grootste kracht is waarschijnlijk dat het onderliggende transportprotocol en achterliggende implementatie van de service geen invloed hebben op het communicatie-programmeermodel zoals Indigo dat voorstaat.

```
Service]
public class ServiceClass
{
   [DataContract]
   public class DataClass
   {
      [DataMember]
      private int DataMember1;
      private string DataMember2;

      [DataMember]
      public string DataMember3 { get { } set { } }
   }
   [ServiceMethod]
   public void ServiceMethod()
   {}
}
```

Listing 1.
Pseudo-code servicedeclaratie



Afbeelding 2. Indigo-programmeermodel

Indigo is unificatie

Op dit moment zijn er op het Microsoft-platform meer programmeermodellen voorhanden die allemaal min of meer hetzelfde voor ogen hebben. Deze zijn in de loop der jaren ontstaan. Zelfs het .NET-platform beschikt over diverse mogelijkheden om hetzelfde te bereiken. Denk hierbij bijvoorbeeld aan .NET Remoting en ASMX webservices.

Met Indigo komt er een einde aan de diversiteit van modellen. Zoals weergegeven in afbeelding 3 worden in ieder geval de programmeermodellen van ASMX, .NET Remoting en .NET Enterprise services samengevoegd in het nieuwe programmeermodel van Indigo. Dit betekent overigens niet dat we de sterke eigenschappen van een bepaald model moeten missen. Indigo biedt ruimte aan alle functionaliteiten uit de verschillende technologieën en beschikt over een eenduidig programmeermodel. Alle functionaliteit kan op een zelfde manier vanuit code CLR-code worden toegepast.

Hoe kan ik me voorbereiden op Indigo?

Is het zinvol om vandaag al bezig te zijn met Indigo en kan de developer zich erop voorbereiden? Het is zeker zinvol om vandaag bezig te zijn met Indigo. Het weerspiegelt de gedachtegang hoe servicegeoriënteerde software door Microsoft geïmplementeerd gaat worden. Dit geeft belangrijke inzichten waar we vandaag de dag al rekening mee kunnen houden bij het opzetten van onze eigen architecturen of ontwerpen.

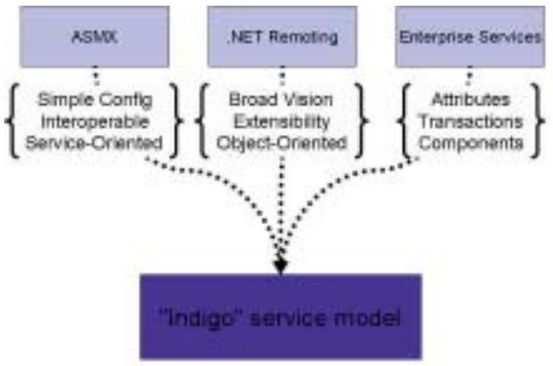
De servicegedachte is verder voor een groot deel al te implementeren met de tools die het .NET Framework nu biedt. De belangrijkste is de ASMX-technologie. Een goede voorbereiding op Indigo is het ontwerpen en gebruiken van ASMX webservices.

Microsoft heeft inmiddels een preview uitgegeven van Windows Longhorn met daarin de bits van Indigo. Deze versie staat bekend als de "PDC-versie". Het connectormodel is in deze versie prima bruikbaar.

Nieuw programmeermodel

Met de komst van Indigo wordt er een totaal nieuw programmeermodel geïntroduceerd. In de basis is Indigo bedoeld om systemen op een eenduidige manier met elkaar te laten communiceren. Het is de Microsoft-implementatie van de servicegedachte. Indigo is een belangrijk onderdeel van Windows Longhorn.

Services zijn de volgende belofte op het gebied van herbruikbaarheid van de bestaande code. Het blijft afwachten of dit wel het juiste abstractieniveau is om succesvol te zijn. Inmiddels zijn er steeds meer bedrijven, waaronder Microsoft, die overtuigd zijn van het servicegeoriënteerde model en er veel energie in steken. Services zijn nog nieuw, maar hebben het in zich om voorlopig een



Afbeelding 3. Indigo: unificatie van programmeermodellen

belangrijk onderdeel te zijn van onze software-ontwikkeling. We kunnen op basis van de bestaande technologieën aan de slag, maar Indigo is de werkelijke implementatie van de Microsoft-visie op servicegeoriënteerd programmeren.

Referenties

http://msdn.microsoft.com/longhorn
http://longhorn.msdn.microsoft.com/
http://msdn.microsoft.com/architecture

Anko Duizer is trainer/coach bij Class-A werkzaam als trainer/coach bij Class-A en is te bereiken via anko.duizer@class-a.nl. Daarvoor heeft hij 5 jaar gewerkt bij Microsoft als consultant. Onder zijn klantenkring bevinden zich voornamelijk Top100 bedrijven in Nederland. Sinds begin 2001 is hij bezig met .NET. www.class-a.nl