Thema

Veel bedrijven proberen een Service Oriented Architectuur (SOA) op te zetten met behulp van Web Services gebaseerd op een set van XML standaarden zoals SOAP, WSDL, UDDI en aanverwante; kortweg de WS-* stack genoemd. Toch is deze aanpak niet probleemloos. Voor een aantal use cases leidt het gebruik van deze standaarden en gerelateerde toolsets tot een grote complexiteit. Maar daarnaast is er ook fundamentele kritiek. De WS-* aanpak maakt maar beperkt gebruik van de zeer succesvolle regels en architectuur van het traditionele web. Er is een manier om die wel optimaal te benutten. Deze architectuurstijl wordt betiteld als REST, Representational State Transfer.

REST: optimale abstracte architectuurstijl

Paul Hermans
is consultant op het vlak van XML en semantische web technologieën en fervent aanhanger van Resource Oriented Computing,
paul@proxml.be

Guy Crets
is Business Integration Architect bij Integr8 Consulting (Cronos),
Guy.Crets@cronos.be

Om uit te leggen wat REST en Web Services volgens de REST-stijl betekenen in vergelijking met 'fullblown' SOA, vertrekken we van het web voor documenten en het protocol waarop dit gebaseerd is, namelijk HTTP 1.1. Daarna besteden we aandacht aan de doctoraatsthesis van Roy Fielding waarin REST voor de eerste maal is beschreven. Dan beschrijven we even de werking van WS-* om vervolgens uit te leggen hoe RESTful Web Services daarvan verschillen. We sluiten af met enkele kritische bemerkingen t.o.v. REST en geven ook advies voor welke use cases de RESTful aanpak meer geschikt is.

HTTP 1.1

In juni 1999 is versie 1.1 van het Hypertext Transfer Protocol gepubliceerd met als één van de editors Roy Fielding. Deze specificatie beschrijft een Request-Respons protocol. Kenmerkend is dat er op de server resources ter beschikking staan. Een client kan *requests* doen naar URI identificeerbare *resources* met behulp van een vaste lijst van metho-

Methode	*Semantiek*	*Safe* **Verandert de state van de server niet.**	*Idempotent* **Meerdere identieke requests hebben hetzelfde effect als een enkele request.**
HEAD	Een GET maar zonder de response body.	+	+
GET	Een request voor een representatie van de gespecifieerde resource.	+	+
POST	Een submit van data die door de gespecifieerde resource moeten geprocessed worden.	-	-
PUT	Een upload van een representatie van de gespecifieerde resource.	-	+
DELETE	Delete van de gespecifieerde resource.	-	+
OPTIONS	Geeft de HTTP methods terug die door een server voor een bepaalde URL supporteert.	+	+

Tabel 1: Overzicht van HTTP methoden

Status code serie	Semantiek	Status Code voorbeelden	Semantiek
1xx:Meta	Gebruikt voor negotiatie met de HTTP server		
2xx:Succes	Een indicatie dat de operatie succesvol was	200	OK
		201	Created
		204	No content
3xx:Redirection	Een indicatie aan de client dat hij extra werk moet doen om te krijgen wat hij wil	301	Moved permanently
		304	Not modified
		307	Temporary Redirect
4xx:Client-side error	Iets is fout aan de client kant	400	Bad Request
		401	Unauthorized
		404	Not found
		409	Conflict
5xx:Server-side error	Fout aan de server kant	500	Internal server error
		503	Service unavailable

Tabel 2: Overzicht van response statussen

den. De semantiek en de kenmerken van deze methoden zijn zeer duidelijk vastgelegd (zie Tabel 1). Daarbij worden de resources zelf niet uitgewisseld maar *representaties* ervan (in formaten zoals HTML, XML, PDF).

Daarnaast biedt versie 1.1 een hele lijst van voorgedefinieerde headers waarvan er een groot aantal dienstig zijn bij het optimaliseren van caching (één van de grote verbeteringen t.o.v. versie 1.0) en essentieel kenmerk van een succesvol gedistribueerd hypermedia systeem.

Ook de responses zijn in hoge mate gestandaardiseerd.

Zeer kenmerkend voor het web van documenten is dat de state van de browser verandert door het volgen van hyperlinks of het doorgeven van parameters m.b.v. een form.

Samengevat: HTTP is een request-respons protocol waarbij representaties in welbekende MIME-types van resources worden uitgewisseld met behulp van een vaste lijst van methodes met een zeer wel gedefinieerde semantiek en gebruik makend van een standaard lijst van response codes. De state van de applicatie wordt veranderd m.b.v. hypertext.

En zoals we elke dag nog kunnen vaststellen, werkt dit protocol op een immense schaal.

REST

In 2000 promoveerde Roy Fielding met de thesis 'Architectural Styles and the Design of Network-based Software Architectures'. Op basis van de concrete ervaringen met HTTP beschrijft hij de volgens hem optimale abstracte architectuurstijl voor gedistribueerde systemen en noemt die REST (Representational State Transfer).
Gedistribueerde systemen worden volgens hem het best gediend met een client-server protocol:

- dat stateless is; elke request van de client naar de server moet alle informatie bevatten om aan de request te kunnen voldoen,
- waar caching mogelijk is,
- waar er gebruik wordt gemaakt van een uniforme interface; een vaste lijst van methoden,
- waar er vlot intermediairen zoals proxies en gateways kunnen worden ingebracht; een gelaagd systeem waarbij elke component niet verder moet kijken dan de laag waarmee hij interageert.

Verder moet die uniforme interface aan de volgende regels voldoen:

- resources moeten geïdentificeerd kunnen worden; een unieke identifier hebben,
- manipulatie van deze resources gebeurt door representaties,
- de boodschappen moeten zelfbeschrijvend zijn, zodoende dat intermediairen (proxies) de boodschappen kunnen behandelen zonder de body, de content te moeten parsen.
- hypermedia is de motor van de applicatie state; het volgen van links leidt tot een nieuwe state.

Dit is eerder een abstracte beschrijving. HTTP is volgens Fielding een concretisering van deze principes, maar voldoet in het gebruik niet altijd aan alle regels. Zo is het gebruik van cookies voor het bijhouden van een sessie ID absoluut niet RESTvol, net zoals het meegeven van een gebruikersnaam in de URL.
Er zijn nog andere voorbeelden te vinden van RESTvolle protocols. We denken dan bijvoorbeeld aan WebDAV, een extensie bovenop HTTP, die een aantal methoden toevoegt aan de bestaande HTTP-lijst. WebDAV staat voor Web-based Distributed

We zien elke dag dat het HTTP-protocol op een immense schaal werkt

Authoring and Versioning en is bedoeld om collaboratief files te beheren en te bewerken op een remote webserver. De meeste document/content management systemen ondersteunen WebDAV.
Een ander voorbeeld is de NETKERNEL application server waar alles, ook je code, resources zijn die via een vaste lijst van methods (SOURCE, SINK) en via representaties gemanipuleerd worden.

WS-*

Het gebruik van distributed objects met CORBA en DCOM brak in de jaren negentig niet echt door en deze technologieën werden helemaal niet geadopteerd op het Web. Ondertussen was het concept van een services benadering ontstaan. Opgejaagd door een aantal XML over HTTP-initiatieven, sloegen Microsoft en IBM de handen ineen en kwamen vrij snel met SOAP op de proppen. SOAP definieert enerzijds een eenvoudige XML-enveloppe structuur met focus op de binding met HTTP. Anderzijds was er de uitdaging van XML-binding, de omzetting tussen programmeertaal structuren en XML structuren – waarvoor initieel SOAP encoding naar voren werd geschoven.
Vervolgens werden in een ijltempo een groot aantal nieuwe webservices specificaties gecreëerd voor beschrijving van webservices, adressering, beveiliging, transacties, policies, reliable messaging en zo meer. Dit resulteert uiteindelijk in de WS-* stack.

Op die WS-* stack valt terecht heel wat kritiek te spuien: specificaties kwamen en gingen, conflicterende specificaties werden door concurrerende groeperingen gepubliceerd en uiteindelijk was er een initiatief zoals WS-I nodig om de specificaties verder scherp te stellen.

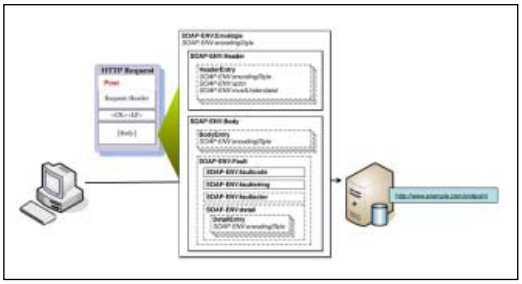
Enkele punten van kritiek:

- WSDL en zijn gebruik van SOAP encoding t.o.v. beschrijving op basis van XML Schema
- Geen eenduidige manier om aan te roepen service te specificeren (SOAP Action, URL, wrapped style en WS-Addressing)
- Globale UDDI servers die werden afgevoerd en waarbij en nog steeds geen alternatief zoals een naming service bestaat.
- Conflicterende standaarden voor non-XML attachments: SOAP with Attachments, DIME en uiteindelijk MTOM
- Nieuwe versies van de SOAP en WSDL specificaties die amper worden geadopteerd, maar waarmee wel rekening dient gehouden in nieuwe specificaties.
- Geen enkel initiatief om de bericht inhouden te standaardiseren of XML bouwstenen te definiëren.
- Een algemene focus op het gebruik van Web Services als synchroon RPC mechanisme.
- De praktische werkwijze zoals wsdl of wrapped style die nergens gespecificeerd wordt.

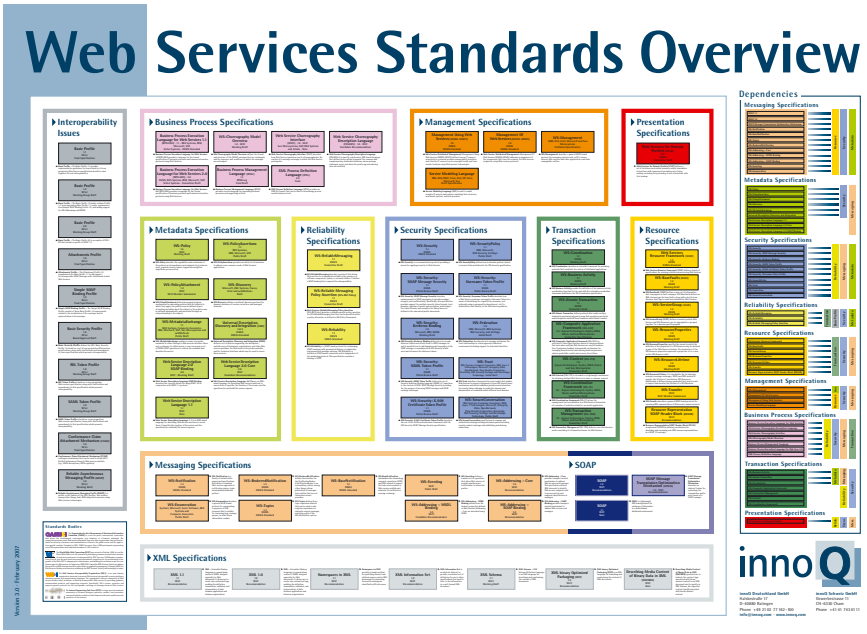
Ondertussen is er heel wat ervaring opgedaan met de ontwikkeling van SOAP Web Services. En in de praktijk worden de specificaties hogerop in de WS-Stack slechts in beperkte mate gebruikt. WS-I compliant webservices op basis van SOAP over HTTP en beschreven met WSDL 1.1 werken prima. De SOAP stacks zoals WCF aan Microsoft-zijde en de JAX-WS compliant stacks aan Java-zijde hebben een goed niveau van stabiliteit, maturiteit en compatibiliteit bereikt. En daarnaast is er ook een ecosysteem ontstaan van hulpmidden, gaande van ontwikkeltools tot Enterprise Service Buses en monitoring hulpmiddelen.

SOAP versus REST

Wat essentieel is in de vergelijking met REST, is dat wanneer SOAP over HTTP gebruikt wordt, die SOAP-message wordt overgebracht als content van de HTTP-body. In het SOAP-bericht staat de service die aangeroepen gaat worden evenals de parameters die de methode verwacht (de scope). De rijkdom van het HTTP protocol wordt amper benut. Het uitsluitend gebruik van POST betekent bijvoorbeeld dat alle caching mogelijkheden van HTTP onbenut blijven.



Een grote versie van deze afbeelding is te vinden op: www.release.nl/site/Hetblad/extra.html



Er kwam al redelijk snel reactie van enkele roependen in de woestijn zoals Mark Baker, Paul Prescod en Steve Vinoski dat de basisconcepten van het Web niet in de WS-* standaarden zaten. Maar wat betekent dit dan concreet?
Laat ons het verschil duiden aan de hand van een webservice die zijn functionaliteit aanbiedt zowel via SOAP als via een REST API. Deze webservice is S3 van Amazon, die online storage aanbiedt. Bij S3 kun je een account krijgen. In die account kun je buckets (mappen) aanmaken en in deze buckets kun je dan files met metadata en bepaalde toegangsrechten plaatsen.

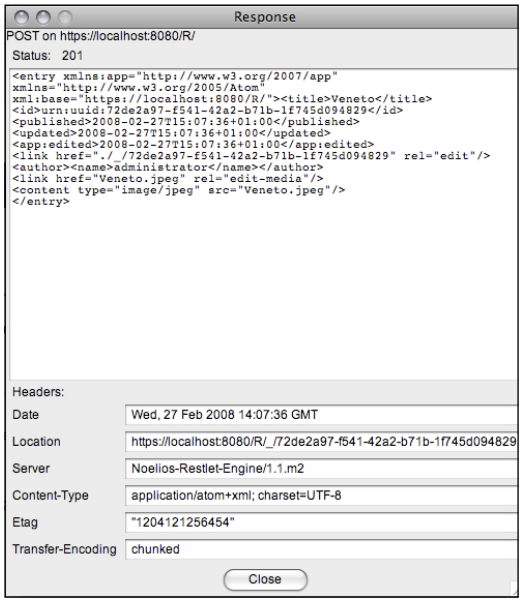
Hieronder een overzicht van beide API's, SOAP en REST.

Object	SOAP endpoint	(SOAP) Method	REST resources	HTTP Method
AWS Account	https://s3.amazonaws.com/soap	ListAllMyBuckets	http://s3.amazonaws.com/	GET
Bucket	https://s3.amazonaws.com/soap	ListBucket <Bucket>STRING</Bucket>	http://s3.amazonaws.com/ProXML	GET
	https://s3.amazonaws.com/soap	CreateBucket <Bucket>STRING</Bucket>	http://s3.amazonaws.com/ProXML	PUT
	https://s3.amazonaws.com/soap	DeleteBucket <Bucket>STRING</Bucket>	http://s3.amazonaws.com/ProXML	DELETE
Object	https://s3.amazonaws.com/soap	GetObject <Bucket>STRING</Bucket> <Key>STRING</Key>	http://s3.amazonaws.com/ProXML/image.jpg	GET
	https://s3.amazonaws.com/soap	PutObject <Bucket>STRING</Bucket> <Key>STRING</Key>	http://s3.amazonaws.com/ProXML/image.jpg	PUT
	https://s3.amazonaws.com/soap	DeleteObject <Bucket>STRING</Bucket> <Key>STRING</Key>	http://s3.amazonaws.com/ProXML/image.jpg	DELETE

Laat ons even de kenmerken van de twee API's vergelijken:
SOAP
- alle request naar één en hetzelfde endpoint
- de methode info zit in de SOAP body
- de scope zit in de SOAP body

REST
- Account, Buckets en Files zijn verschillende resources met een URI, waarin dus de scope zit.
- de aangeboden methoden zijn de standaard HTTP-methoden
- de response statussen zijn de standaard HTTP response statussen
- de representaties zijn XML-files

REST volgt de principes van het Web: stateless, cachebaar, gebruik makend van de uniforme HTTP interface, kunnen gebruik maken van proxies, gateways en zo meer.
Helemaal volgens de principes van REST, op de uitzondering na dat er geen links in de XML-representaties zitten om de applicatie state te kunnen aanpassen, is iets wat we bijvoorbeeld wel zien in een andere REST API, APP. APP staat voor Atom Publishing Protocol, een HTTP-gebaseerd protocol voor het creëren en updaten van web resources die als representatie formaat gebruik maakt van het Atom feed format, een XML taal voor web feeds.
De links zie je bijvoorbeeld in de volgende response body na een gelukte post van een foto. Je ziet een link naar de metadata en een link naar de foto zelf.



Al in 2003 liet Jef Barr van Amazon weten dat indien zij services via verschillende API's aanboden, de REST variante 'by far' de meest populaire was; een stelling die in 2005 onderschreven werd door de toenmalige CEO van Flickr.
Dare Obasanjo (Microsoft) en James Snell (IBM), beiden vroeger betrokken bij het ontwikkelen van webservices specificaties en gerelateerde toolsets en nu beiden werkzaam aan de applicatiekant, kiezen nu pertinent voor REST en meer specifiek voor APP.
Waarom? Omdat dit simpel is en volgens de regels van het web werkt.

Overzicht van beschikbare RESTful WebServices

Voor een mooi overzicht van een groot aantal publieke Web Services verwijzen we naar de website Programmable Web (http://www.programmableweb.com/apis). Uit deze website leren we dat het aantal REST webservices een enorme groei heeft gekend in de laatste jaren.

Ook krijgen applicaties hoe langer hoe meer een RESTful API. In de eerste plaats content manage-

De B2B-standaarden

Naast Web Services is er nog een derde, minder bekende stroming, de B2B-standaarden. Organisaties wisselen al lang gestructureerde data uit met elkaar op basis van standaarden zoals EDIFACT, HL7 in de gezondheidssector of SWIFT in de financiële sector.
Ook in deze B2B wereld worden internet en XML technologie geadopteerd, denk maar aan ebXML. Er zijn initiatieven voor het definiëren van XML bericht structuren, bvb. OAG, GS1 en UBL. En ook voor het betrouwbaar en veilig uitwisselen van gegevens zijn er meerdere initiatieven, bvb. het populaire EDIINT AS2. Zulke B2B initiatieven zijn dikwijls gericht op een welbepaalde sector of regio, bvb. RosettaNet in de wereld van de halfgeleiders.
Opvallend gegeven is dat Web Services technologie in de B2B wereld amper wordt gebruikt. Ontzettend populair zijn XML over HTTPS alsook beveiligde file transfers.

ment systemen zoals Alfresco, Nuxeo, Daisy met zelfs een poging tot standaardisatie in de vorm van CMIS (Content Management Interoperability Services), maar ook toepassingen zoals Mingle voor agile project management tot Microsoft SQL server via ADO.NET Data Services.

Uiteraard kennen de SOAP Web Services ook nog de nodige successen. Fraai voorbeeld hiervan zijn de Enterprise Services zoals ze door SAP worden geïmplementeerd. Uitzonderlijk hierbij is de energie die in de goede modellering en definitie van de services wordt geïnvesteerd.
Het valt toch op dat zowel aan WS-* als aan REST zijde weinig aandacht is om de service definities of resource representaties te standaardiseren. Dit in sterke tegenstelling tot de B2B wereld (zie kader).

Het ontwikkelen van RESTful Web Services

Welke stappen moet ik zetten om een RESTful API te ontwerpen? Joe Gregorio heeft deze nu algemeen aanvaarde procedure uitgewerkt:

- definieer de data set
- split de dataset in verschillende resources, bijvoorbeeld de lijst van onze werknemers, een werknemer enzovoorts.

Definieer dan per resource

- de naam, c.q. URI, b.v. http://www.onzefirma.com/werknemers/ voor de lijst van werknemers, http://www.onzefirma.com/werknemers/P123 voor de werknemer met personeelsnummer P123.
- de toegelaten methoden (GET, PUT, DELETE, POST) van de uniforme interface
- het formaat van de inbound representatie (XML, JSON, ...)
- het formaat van de outbound representatie, b.v. de XML representatie van een werknemerslijst. Let op de inclusie van het link element dat de unieke identifier van de individuele werknemer(s) bevat. Remember 'hypertext as the engine of application state'.
- de response codes (succes, error) bij de verschillende methoden
- de mogelijke linken naar andere resources in de response representaties

Tooling

Zowel in de Java-omgeving als in de .NET omgeving kan men mits annotaties methoden REST-gewijs exposen. Aan de Java-kant hebben we sinds kort JSR 311: Java API for RESTful Web Services, ook wel JAX-RS genoemd. Naast de door SUN ter beschikking gestelde referentie implementatie, Jersey, zijn er andere beschikbaar:

- als onderdeel van het populaire Restlet framework JBOSS's RESTeasy

```
<?xml version="1.0" encoding="UTF-8"?>
<werknemerlijst>
    <werknemer>
        <naam>Jan Janssens</naam>
    <link>http://www.onzefirma.com/werknemers/P123</
link>
    </werknemer>
    <werknemer>
        <naam>Piet Pieters</naam>
    <link>http://www.onzefirma.com/werknemers/P456</
link>
    </werknemer>
</werknemerlijst>
```

- als onderdeel van de Apache CXF web services stack

Het Service Pack 1 voor 'Visual Studio 2008 and .NET Framework 3.5' bevatte ook ADO.NET Data Services Framework v1, het zogenaamde project Astoria. Hiermee krijgt de .NET omgeving een mooie ondersteuning voor REST. En dit zal in de toekomst alleen maar toenemen, *cf.* de recente aankondiging van verbeterde REST interfaces in WCF 4.0 en een REST Starter Kit.
Ook in andere programmeeromgevingen kun je Restvol aan de slag: in Ruby en 'Ruby on Rails', in Python en Django, in Groovy en PHP dank zij 'Project Zero' van IBM.

Conclusie

Betekent dit nu het einde van WS-*? Absoluut niet. WS-* heeft overgewicht, maar een afgeslankte versie is perfect bruikbaar, inclusief compatibiliteit over platformen heen. SOAP is trouwens het minder praktische protocol geworden voor ontwikkelingen binnen een Service Oriented Architecture.
REST is op zijn best voor CRUD operaties over het web, zowel voor content als voor data (Data Services), maar heeft momenteel nog geen simpele en gestandaardiseerde oplossingen voor batch operaties of transacties waar de WS-* stack wel oplossingen voor heeft.

In essentie is het geen keuze, want de use case bepaalt de technologie. Indien het om een simpele request-respons gaat waar het HTTP protocol de context (identity, credentials, gewenst responsformaat) perfect kan beschrijven, ga voor REST.
Gaat het om requests die meerdere partners, meerdere processen omvat en complexere transactionele aspecten heeft, dan spreekt het voor zich dat de context mee in de request wordt gestoken, zoals SOAP dat doet.

Zoals Brian Sletten, terecht stelt: "Use SOAP for invoking behavior, use REST for managing information."

Het terechte punt van de Restafarians (aanhangers van het REST protocol) is dat SOAP een mismatch is voor de simpele use cases voor het request-responsgewijs managen van informatie. «

Referenties

Architectural Styles and the Design of Network-based Software Architectures, Roy Fielding, 2000, University of California. http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm
RESTful Web Services, Leonard Richardson, Sam Ruby, O'Reilly Media, 8 May 2007, ISBN-10: 0596529260
http://www.infoq.com/articles/rest-introduction
http://www.infoq.com/articles/tilkov-rest-doubts
http://www.infoq.com/articles/rest-anti-patterns
http://www.xfront.com/REST-Web-Services.html
http://tomayko.com/writings/rest-to-my-wife
http://www.25hoursaday.com/weblog/2008/08/17/ExplainingRESTToDamienKatz.aspx
http://www.tbray.org/ongoing/When/200x/2008/08/18/On-REST
http://bitworking.org/news/125/REST-and-WS
http://www.ajaxonomy.com/2008/xml/web-services-part-1-soap-vs-rest
http://soundadvice.id.au/blog/2008/07/06/#core-rest-patterns
http://www.coactus.com/blog/2007/01/starting-with-the-web/
http://tssblog.techtarget.com/index.php/interoperability/mini-guide-rest-representational-state-transfer/
http://steve.vinoski.net/blog/internet-computing-columns/

18 maart 2009 • Hotel Lapershoek Hilversum

Agile software development in de praktijk

Seminar met veel praktijkvoorbeelden en een interessante case

met Sander Hoogendoorn

U krijgt antwoord op de volgende vragen:

- Wat is nu precies agile software development?
- Wat kenmerkt agile projecten?
- Welke agile methodieken zijn er eigenlijk en wat zijn hun overeenkomsten en verschillen?
- Welke methodiek past het best bij uw organisatie?
- Welke agile best practices zijn er en welke kunt u al direct toepassen?
- Hoe verkoopt u agile software development aan uw klanten?
- Kunt u agile techieken toepassen in productdevelopment?

Dit seminar met Sander Hoogendoorn, agile thought leader bij Capgemini, laat zien hoe agile software development bijdraagt aan het succesvol uitvoeren van software development projecten. Sander bespreekt de nadelen van traditionele, lineaire methodieken en hij gaat hij in op de karakteristieken van agile software development. Ook komen bekende misvattingen over agile aan bod.
Het seminar geeft de overeenkomsten en verschillen tussen de belangrijkste agile methodieken, zoals Scrum, extreme programming, Smart, Lean en DSDM aan. Natuurlijk passeert ook een groot aantal best practices, tools en technieken uit de alledaagse praktijk de revue. Gastspreker Stefaan van Royen van MediaMine NG toont de “lessons learned” bij de succesvolle invoering van agile software development in zijn bedrijf.
Deelnemers aan dit seminar krijgen een helder inzicht in de positieve bijdrage die agile software development levert aan projecten, en zij krijgen een reeks van concrete en pragmatische handvatten, technieken en best practices voor het implementeren van agile software development in de organisatie.

Bestemd voor ú
De materie en de vele praktijkvoorbeelden in dit seminar hebben tot doel de kwaliteit en productiviteit van uw projecten te vergroten. Het is daarom bedoeld voor iedereen die betrokken is bij softwaredevelopment: opdrachtgevers, IT-managers, projectmanagers, architecten, informatieanalisten, ontwerpers, ontwikkelaars en testers.

DATUM	**18 maart 2009**
LOCATIE	**Hotel Lapershoek Hilversum**
TIJD	**van 09.30 tot 17.00 uur**

Kijk snel op www.arrayseminars.nl voor het complete programma!

Array SEMINARS

in samenwerking met Computable