Betreft	**Service Oriented Architecture voor de UT, een verkenning t.b.v. de definitiefase van het CBUS project.**
Van	Eelco Laagland
Voor	MT ITBE en projectleden
Datum	27 juli 2006

Service Oriented Architecture voor de nieuwe Digitale Leer- en Werkomgeving van de Universiteit Twente

Inhoud

Management samenvatting

In dit rapport wordt het concept Service Oriented Architecture (SOA) nader onderzocht op haar relevantie voor de Universiteit Twente. Deze SOA is een van de mogelijkheden om de bedrijfsarchitectuur van de UT, weerspiegeld in de bedrijfsprocessen, af te beelden op een technische ICT infrastructuur. De bedrijfsarchitectuur zelf van de UT komt in dit rapport niet aan de orde, we beperken ons tot de informatiearchitectuur in de vorm van een SOA. We onderzoeken waar dit met name voor het onderwijs domein van de UT, voor zover dat door ICT voorzieningen wordt ondersteund, relevant is. Maar ook zullen, gezien de gesignaleerde wensen om te komen tot een meer geïntegreerde informatievoorziening, de andere domeinen meegenomen worden in de analyse. En hoewel onder services ook handmatige implementaties vallen, beperken we ons tot automatische webservices.

In de rapporten die opgeleverd zijn voor de ELO Adviescommissie is de nadruk gelegd op de begrippen interoperabiliteit door middel van specificaties en standaarden. De conclusie was dat voor het realiseren van een belangrijk deel van de wensen van de gebruikers (studenten en docenten) een Service Oriented Architecture een belangrijke randvoorwaarde is om de ambities van de UT te realiseren. Integratie van de verschillende systemen, flexibel in kunnen spelen op in aard en omvang veranderende doelgroepen en omstandigheden alsmede de mogelijkheid om met best-of-breed componenten ICT systemen samen te stellen zijn de belangrijkste overwegingen hierbij. In dit rapport wordt een nadere uitwerking gegeven van dit begrip en de contouren geschetst van een mogelijke implementatie hiervan in de ICT infrastructuur van de Universiteit Twente.

Inleiding

Het voorliggende rapport is opgesteld in het kader van het project Campus Blend using Sakai (CBUS).Dit rapport heeft als doel het management team van ITBE en het College van Bestuur van de Universiteit Twente van informatie te voorzien ten behoeve van een nadere beleidsbepaling voor de toekomstige digitale leer- en werkomgeving van de Universiteit Twente. Verschillende trendanalyses en onderzoeken geven aan dat (a) de ‘klantengroep’ (de studenten) van een onderwijsinstelling steeds heterogener gaat worden en (b) mede als gevolg hiervan de onderwijsprocessen in de nabije toekomst sneller zullen veranderen dan in het verleden. Hoe deze processen zullen veranderen is nog niet altijd helder. Maar dat ze zullen veranderen en dat steeds sneller zullen doen, is wel algemeen geaccepteerd. Snel en adequaat deze zich snel wijzigende businessprocessen kunnen ondersteunen met ICT voorzieningen wordt gezien als een van de belangrijke uitdagingen voor de komende jaren, en Service Oriented Architecture is een belangrijk kernondereel van de oplossing. Een instelling die haar aanbod niet kan differentiëren in lijn met de diversiteit van de studentenpopulatie zal uiteindelijk een marginaal bestaan gaan leiden.

Service Oriented Architecture, een hype?

SOA of Service Oriented Architecture lijkt na XML het laatste acroniem waarmee de ICT sector ons wil overtuigen van de ultieme oplossing voor onze automatiseringsproblemen, die nu sneller en goedkoper en natuurlijk beter zullen worden opgelost dan voorheen. Een hype? Het begrip SOA krijgt de laatste tijd in ieder geval bovenmatige aandacht in de media, en niet alleen in

gespecialiseerde IT media. Hierin ligt de suggestie besloten dat het belangrijk is. Of en waarom SOA al deze aandacht verdient en of het voor de Universiteit Twente een relevante ontwikkeling is proberen we in dit rapport te onderzoeken. Een aantal indicatoren:

1. Het rapport 'Kiezen en delen", een advies voor het SURF meerjarenplan 2007-2010 van de Wetenschappelijk Technische Raad van SURF kent aan SOA een sleutelrol toe.. Hiermee is SOA in de perceptie van de WTR een belangrijke facilitator voor het uitwisselen van informatie tussen applicaties binnen een instelling. Hierbij kan gedacht worden aan b.v. de leeromgeving TeleTOP en het Vak Informatie Systeem Twente (VIST). Er kan ook nadrukkelijk gedacht worden aan het uitwisselen van informatie tussen instellingen, bijvoorbeeld in 3TU verband tussen de verschillende leeromgevingen TeleTOP/Sakai, Studyweb en Blackboard.
2. In de zakelijke markt is de ontwikkeling richting service orientatie al langer aan de gang en wordt daar in ruime mate voorzien van tools. Die tendens is inmiddels ook ruimschoots aanwezig in het hoger onderwijs. Het IMS Global Learning Consortium[1], een organisatie die zich bezighoudt met de ontwikkeling van standaarden en specificaties voor het HO, heeft in 2003 al een eerste schets opgeleverd van een op service orientatie gebaseerde architectuur in de vorm van het IMS Abstract Framework, primair bedoeld als conceptueel kader voor te onwikkelen e-learning specificaties en instellingsarchitecturen. Ook OASIS (Organization for the Advancement of Structured Information Standards)[2] een internationaal consortium voor de ontwikkeling van e-business specificaties en standaarden houdt zich bezig met SOA en is de eigenaar van vele belangrijke specificaties en standaarden die de bouwstenen vormen van een SOA.
3. Het IMS Abstract framework heeft als basis gediend voor het JISC e-Learning Framework, waarin konkreet aan implementatie gewerkt wordt en dat ook nadrukkelijk service orientatie als leidend ontwerp principe heeft. Al snel werd duidelijk dat voor het hoger onderwijs de beperking tot alleen het domein e-learning te smal was en is het concept uitgebreid tot e-Research en e-Administration. Het overkoepelende (onderzoeks- en ontwikkel)programma draagt nu de naam e-Framework[3] en is inmiddels een samenwerkingsverband tussen JISC en het Australische ministerie van onderwijs en wetenschap DEST. Ook SURF heeft in 2005 een samenwerkingovereenkomst met JISC getekend om deel te nemen aan het e-Framework. Aan deze samenwerking wordt op dit moment (medio 2006) concreet vorm gegeven onder andere op basis van resultaten van een DU project ELO Groei-en Verandermanagement[4], waar de UT ook bij betrokken is. SURF heeft tevens onlangs een coördinerende functie gecreëerd, waarmee nadrukkelijk het e-framework op de agenda in het Nederlandse Hoger Onderwijs zal worden geplaatst.
4. Ook de door de ELO Adviescommissie aanbevolen opvolger van de huidige leeromgeving TeleTOP, de leeromgeving Sakai, heeft service oriëntatie als leidend paradigma.

Al met al voldoende signalen om te veronderstellen dat SOA waarschijnlijk geen hype is maar voldoende substantie heeft om serieus overwogen te worden voor invoering bij de Universiteit Twente. Maar waar gaat het nu eigenlijk om?

[1] IMS, Global Learning Consortium, http://www.imsproject.org/
[2] Zie: http://www.oasis-open.org
[3] e-Framework, zie: http://www.e-framework.org/
[4] Zie de project Wiki: http://wiki.du.nl/ELO_GVM

Wat is een Service Oriented Architecture?

Als we op zoek gaan naar een definitie van SOA zien we dat de term SOA erg vaak gebruikt wordt en doorgaans zonder een heldere eenduidige definitie. Het W3C gebruikt de volgende definitie:

‘ A set of components which can be invoked, and whose interface descriptions can be published and discovered’[5]

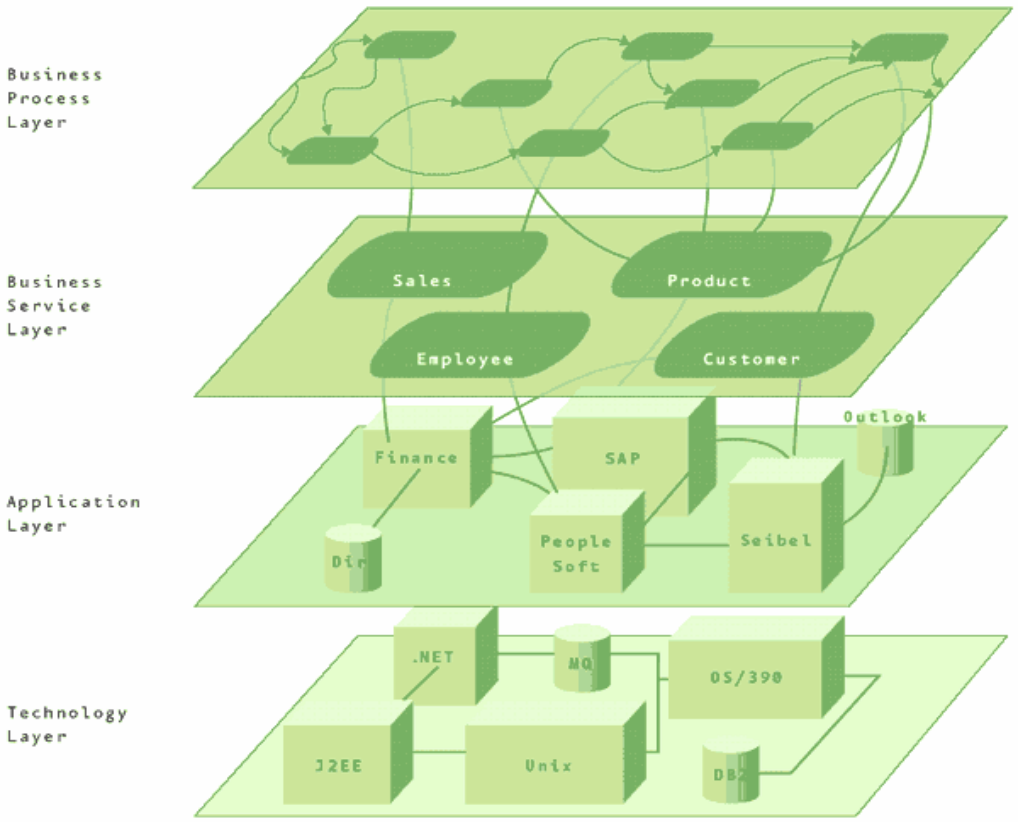
Het accent ligt in deze definitie erg op de het kunnen publiceren van de servicebeschrijving in een (soms openbaar) informatiesysteem, een soort Gouden Gids voor services (UDDI[6]).

Iets uitgebreider geformuleerd: SOA gaat uit van het beschrijven van een informatie infrastructuur als een verzameling van autonome services die met elkaar communiceren. Deze communicatie kan een eenvoudige producent-consument relatie zijn tussen twee services of een meer complexe samenwerking tussen meerdere services. Deze services zijn gemodelleerd volgens de bedrijfsprocessen die ze moeten implementeren, en zijn zoveel mogelijk generiek ontworpen zodat hergebruik mogelijk is in verschillende bedrijfsprocessen. Een belangrijk kenmerk van een service georienteerde architectuur is dat de koppeling tussen de services losjes is en niet hard in de code van de services is ingebouwd. Op dit verschil zullen we verderop in dit rapport iets dieper ingaan. Bij Service Orientatie worden traditionele applicaties opgebroken in discrete onafhankelijke componenten of ‘services’ die op een gedistribueerde manier aangeroepen kunnen worden.

In deze beschrijving is een relatie met bedrijfsprocessen gelegd. Schematisch kan dit weergegeven worden zoals in figuur 1, waarbij bovenop de Applicatie laag een laag gedefinieerd is waar zich de services bevinden die eenvoudige, herbuikbare functionaliteit leveren die aangesproken wordt in de bedrijfsprocessen die zich een de bovenste laag bevinden, de bedrijfsproces laag.

In de verschillende lagen kunnen koppelingen aangebracht worden ten behoeve van integratie. In de technologielaag gaat het bijvoorbeeld om koppelingen tussen de .NET en J2EE platforms. Hoe gebruik je bijvoorbeeld een Microsoft ActiveX Control in een Java bean. In de applicatielaag gaat het om koppelingen bijvoorbeeld ten behoeve van workflow tussen applicaties. Hiervoor wordt doorgaans Enterprise Application Integration (EAI) technologie toegepast met adaptors die vaak erg specifiek zijn voor de functionaliteit van een applicatie. Er is behoefte aan een scheiding van een door een service geleverde functionaliteit van de implementerende applicatie en daarmee van de onderliggende technologie in de vorm van een bedrijfs service laag.

[5] Zie: http://www.w3.org/TR/ws-gloss/
[6] Zie: http://www.uddi.org/



Figuur 1 : Gelaagd model van een Service Oriented Architecture (bron MSDN[7])

Bedrijfsprocessen

In de vorige paragraaf is een relatie gelegd met de bedrijfsprocessen die met behulp van een Service Oriented Architectuur geimplementeerd kunnen worden. Deze bedrijfsprocessen zijn beschreven en afgeleid uit de bedrijfsarchitectuur. Deze bedrijfsarchitectuur is uitdrukkelijk niet het onderwerp van dit rapport, maar om de context te schetsen zullen we er toch kort op ingaan. In het SURF eindrapport van de werkgroep Architectuur, 28 maart 2005[8], bestaande uit een groot aantal vertegenwoordigers van Nederlandse HO instellingen, is een poging gedaan om te komen tot een beschrijving van de bedrijfsarchitectuur van een fictieve HO instelling, de 'referentie HO instelling'. Hierbij is als uitgangspunt gehanteerd dat deze architectuur gericht moet zijn op de toekomst en minder een weerspiegeling zou hoeven te zijn van de huidige bedrijfsarchitecturen.

Daarbij zijn een aantal architectuurpricipes gehanteerd:

- Vraaggestuurd onderwijs;
- Individueel maatwerk kunnen leveren;
- Competentiegericht;

[7] Service-Oriented Archtecture: Considerations for Agile Systems
[8] Zie: http://www.surf.nl/download/050401WGarchDEF.pdf

- Plaats- en tijdonafhankelijk;
- Flexibiliteit ten aanzien van onderwijs:
    - Organisatorisch (creditsysteem ipv jaarsysteem);
    - Inhoudelijk (eigen keuzes, keuzeprogramma's);
    - Didactisch (mix van functionele didactische leervormen).

En de belangrijkste ontwikkelingen in het Hoger Onderwijs zijn in kaart gebracht:

- Het Hoger Onderwijs ontwikkelt zich van een aanbodgerichte naar vraaggestuurde organisatie;
- Verscheidenheid in aan te bieden producten: voltijd/duaal/deeltijd/'losse' cursussen, waarbij een tendens duidelijk aanwezig is naar meer maatwerk/individualisering;
- Studenten en de onderwijsinstelling hebben een gezamenlijke verantwoordelijkheid ten aanzien van het leerproces van de student;
- Toepassing van meerdere en nieuwe vormen van leren:
    - Competentiegericht leren;
    - Niet alleen de nadruk op formeel leren, maar ook op informeel leren
- Innovatieve inzet van ICT, niet alleen ter ondersteuning van het aanbieden van leermateriaal maar onder andere ook voor portfoliomanagement, het beheren van het studiecontract en contentmanagement.
- Plaats- en tijdonafhankelijk werken en leren;
- Internationalisering/regionalisering en nieuwe vormen van samenwerkingsverbanden.

Een belangrijke stelling wordt in het rapport ingenomen:

"Voor een IT-architectuur geldt dat flexibiliteit, openheid, beschikbaarheid en integratie van (externe) applicaties mogelijk moet zijn."

In het SURF architectuur rapport worden 150 elementaire bedrijfsprocessen beschreven voor de genoemde 'referentie HO instelling', die alle domeinen van de HO instelling afdekken. Deze processen zullen niet allemaal even sterk beinvloed worden door de geschetste ontwikkelingen, maar een groot aantal waarschijnlijk wel. Hoewel de voorbeelden in het voorliggende SOA rapport beperkt zijn tot het domain e-Learning, is de SOA benadering dus hiertoe nadrukkelijk niet beperkt. De verschillende afzonderlijke domeinen overlappen elkaar steeds meer, en hierin heeft JISC ook de aanleiding gezien haar e-Learning Framework (ELF) te verbreden met de domeinen e-Research en e-Administration.

In het DU project ELO Groei-en Verandermanagement is voortgeborduurd op o.a. het werk het de SURF architectuurproject, en zijn een aantal bedrijfsprocessen beschreven, met name voor het onderwijsdomein, waarvan de verwachting is dat ze in de nabije toekomst zullen veranderen en ook zullen blijven veranderen. Hierbij is uitgegaan van een inschatting van het HO domein over vijf jaar. Ook in deze studie wordt een trend beschreven van veranderende bedrijfsprocessen, die een flexibele, open, en geintegreerde informatie architectuur wenselijk maken.

Het is deze flexibiliteit, openheid, beschikbaarheid en integratiemogelijkheid met (externe) systemen, die we verwachten te kunnen leveren met een SOA informatie archtectuur. Ook de Universiteit Twente zal op de genoemde uitdagingen een antwoord moeten formuleren. De interne integratieproblematiek speelt nu al, studenten verwachten dat onderwijssystemen geintegreerd zijn (TeleTOP, VIST, Portal, Mail, Roosterinformatie), dit is een expliciete wens die in het voorgaande ELO Keuzeproject geinventariseerd is. De externe integratieproblematiek komt op korte termijn aan de orde in 3TU verband, als bedrijfsoverstijgende processen

geimplementeerd moeten worden. Te denken valt aan een gemeenschappelijk Vakinformatie Systeem (VIST), toegang tot elkaars electronische leer- en projectomgevingen, (bibliotheek)resources e.d. De flexibiliteit zal nodig zijn om de in kaart gebrachte ontwikkelingen in het Hoger Onderwijs te kunnen faciliteren.

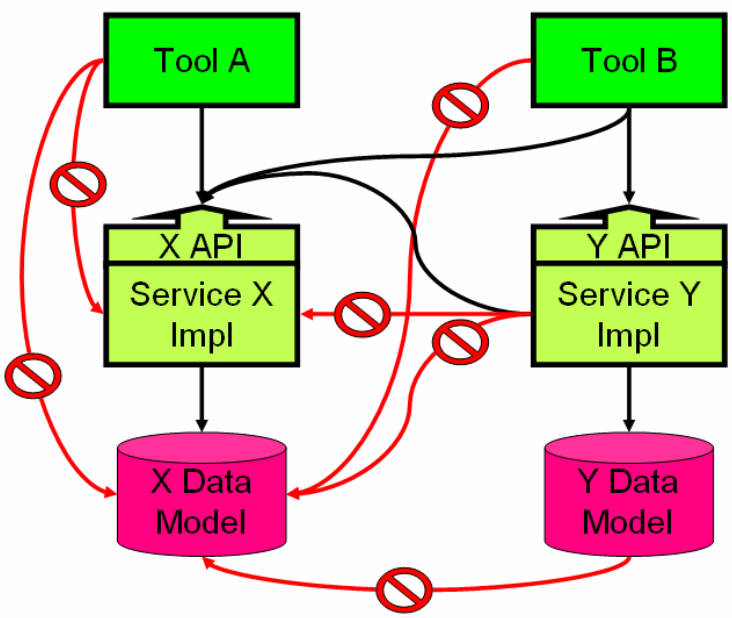
Maar ook om een andere reden is het invoeren van een SOA voor de Universiteit Twente te overwegen. Een groot deel van de huidige bedrijfsprocessen zijn nu geimplementeerd in eigen specifieke applicaties. Daarbij is het bijna onvermijdelijk dat een groot deel van de functionaliteit in meerdere applicatie aanwezig is. Dat levert een groot beheers probleem op en is een bron van inconsistenties. Er is beslist winst te behalen als deze gemeenschappelijke functionaliteit (code) geidentificeerd wordt en waar mogelijk slechts op een enkele plaats aanwezig is en beheerd hoeft te worden. Bij een goed ontwerp van deze gemeenschappelijk functionaliteit is hergebruik in meerdere applicaties (bedrijfsprocessen) mogelijk. Waar functionaliteit op meerdere plaatsen aanwezig is kan gekozen worden voor de beste oplossing, en is een ‘best of breed’ aanpak mogelijk, een op componenten of services gebaseerde aanpak dus.

Wat zijn services?

Een service is een functionele eenheid die duidelijk gedefinieerd is in termen van datamodellen, interfaces en dynamisch gedrag en kan gezien worden als een bouwsteen waarmee grotere functionele eenheden (applicaties) kunnen worden samengesteld.

Voor het ontwerp van een service is de mogelijkheid tot hergebruik in verschillende bedrijfsprocessen belangrijk. Initiële extra investeringen in een goed ontwerp, waarbij de kenmerken van de onderliggende applicatie (implementatie) en de gebruikte technologie zo veel mogelijk geabstraheerd worden, zullen bij hergebruik later hun vruchten afwerpen. Gegevens, gegevenstypen en dynamisch gedrag van de service die alleen voor de (interne) implementatie van de service en niet voor het afnemende bedrijfsproces van belang zijn worden verborgen en zijn niet zichtbaar in de openbare servicebeschrijving. Services bieden zelf geen presentatie faciliteit, kunnen dus bijvoorbeeld geen HTML genereren en kunnen alleen van andere services functionaliteit gebruiken via het gedokumenteerde service interface (Service Adaptor).

Ter illustratie kan het het gebruik van services in Sakai dienen. Zie onderstaande figuur 2.

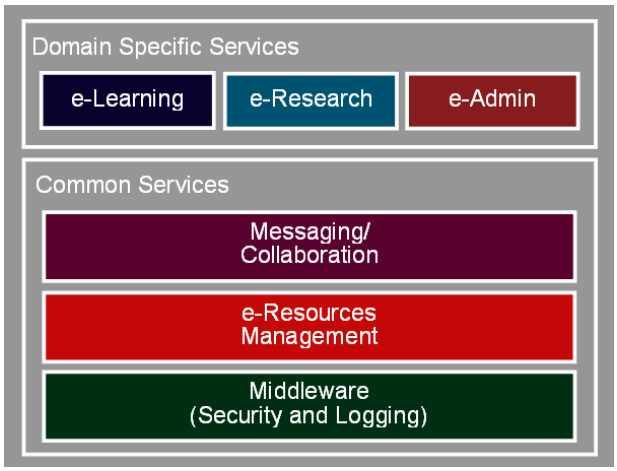


Figuur 2: Afspraken voor het gebruik van services, overgenomen uit 'Sakai Technical Overview", Dec 8, 2005

Tool A, bijvoorbeeld een voorziening om gebruikersvoorkeuren (taal, presentatievoorkeuren voor accessibility) vast te leggen, kan bijvoorbeeld helemaal geïmplementeerd worden met de services die door service X geboden worden. Elke andere tool mag deze alleen aanspreken via het gedokumenteerde service interface API X en niet door rechtstreekse aanroepen van funkties uit de implementaie van service X. Ook toegang tot het onderliggende datamodel is verboden. Dezelfde regels gelden voor Tool B, bijvoorbeeld een cursuscalender, die voor implementatie afhankelijk is van de services van service X en service Y. Ook hier mag tool B alleen via het gedokumenteerde service interface de services van service X en Y benaderen om cursusinformatie volgens persoonlijke voorkeuren weer te geven. Ook de implementatie van service Y mag gebruik maken van de door service X aangeboden services, maar ook hier alleen via het service interface. Op deze manier kunnen de service implementaties en de onderliggende datamodellen gewijzigd of vervangen worden zonder de afnemende services en tools te verstoren.

Domain Services en Common services

In de eerder genoemde frameworks, het IMS Abstract Framework en het JISC e-Framework, worden services onderscheiden in Domain Services en Common Services. Hierbij zijn de services die door de Domain Services geboden worden specifiek voor een bepaald domein of toepassingsgebied, bijvoorbeeld e-Learning, e-Research en e-Admin, zie onderstaande figuur:



Figuur 3: Service indeling in het JISC e-Framework

De Common Services zijn in principe domein onafhankelijk en bieden doorgaans toegang tot de laag met de technische infrastructuur. Voorbeelden van Common Services zijn Authorisation en Authentication.

Niveau's van services, service aggregaties

Er kunnen meerdere niveau's van services onderscheiden worden, waarbij nieuwe services gekonstrueerd kunnen worden met behulp van meerdere eenvoudige services. Deze aggregaties kunnen zelf weer voorzien worden van een service interface. Voor deze aggregaties gelden de eerder genoemde regels natuurlijk ook weer.

Services worden vaak geassocieerd met Web services. Op zich zijn dit separate begrippen. Een service kan ook door middel van een handmatig proces geimplementeerd worden. Ik dit rapport bedoelen we met services voornamelijk de automatische implementatie in de vorm van webservices, zie de definitie in Wikipedia:

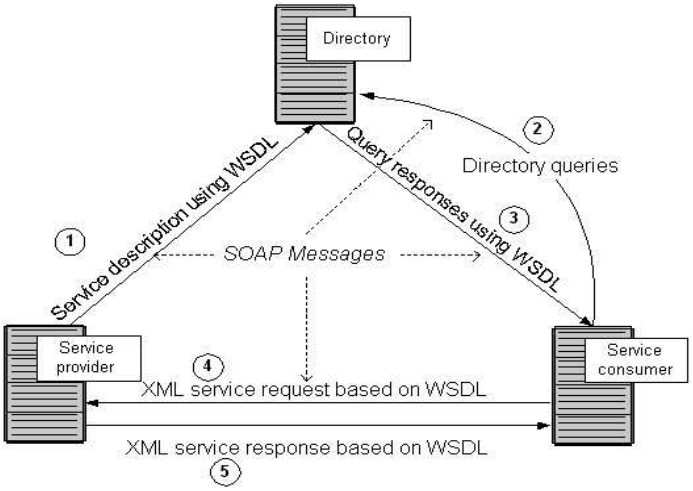
"Een **webservice** kan omschreven worden als een applicatiecomponent die toegankelijk is via standaard webprotocollen. Een webservice maakt het mogelijk om vanop afstand (meestal over het Internet) vanaf een client-computer een dienst op te vragen aan een server, bijvoorbeeld het maken van een berekening, het leveren van gegevens of het uitvoeren van een taak."

Services en bestaande systemen

Services hoeven niet noodzakelijkerwijs opnieuw ontworpen of aangeschat te worden. De meeste bedrijven en instellingen beschikken natuurlijk al over een grote hoeveelheid aan systemen die hun waarde bewezen hebben en soms al langere tijd in gebruik zijn. Deze zogenaamde legacy systemen hebben hun waarde in de organisatie bewezen en hoeven ook niet van de ene dag op de andere vervangen te worden bij invoering van een SOA. Een analyse van de geboden functionaliteit van deze systemen kan duidelijk maken dat een deel van deze functionaliteit geschikt is om in de vorm van een service beschikbaar gemaakt te worden. Er kan dan overwogen worden om deze functionaliteit door middel van een zogenaamde service adaptor toegankelijk te maken.

Wat is er zo nieuw aan een Service Oriented Architecture?

Er is al een lange traditie in de IT van pogingen om de ontwikkelkosten van IT voorzieningen beter beheersbaar te maken en te verlagen. In de loop der tijd hebben verschillende oplossingen het licht gezien, te beginnen met de invoeren van bibliotheken van herbruikbare code in de vorm van linkbare **modules** begin jaren 80 in talen als Pascal en Algol. Een eerste vorm van wat lijkt op service orientatie was C++, waarbij code en de data gezamenlijk 'verpakt' was in de vorm van **objecten**. Maar ook hier was nog steeds sprake van een technlogische afhankelijkheid. De herbuikbare verzamelingen van objecten zijn nog steeds specifiek voor een bepaalde taal (C++ bijvoorbeeld). Een volgende stap in de evolutie van softwareontwikkeling en een poging tot gedistribueerde systemen en hergebruik van functionaliteit zijn **componenten** met als belangrijkste specificaties Distributed Component Object Model (DCOM) van Microsoft en Common Object Request Broker Architecture (CORBA) van o.a. SUN en IBM ontwikkeld als reaktie op DCOM.
Maar ook hier heeft het feit dat deze technologieen platformspecifiek waren brede verspreiding in de weg gestaan. Pas het tot wasdom komen van platformonafhankelijke specificaties voor **services** in de vorm van Simple Object Access Protocol (SOAP) en Web Service Description Language (WSDL) en de mogelijkheid van publicatie van web services in gemeenschappelijke registratiediensten met behulp van Universal Description, Discovery, and Integration (UDDI) heeft een belangrijke impuls gegeven aan het concept Service Oriented Architecturen. Met behulp van deze service directories kunnen services providers bijvoorbeeld verplaatst of vervangen worden zonder dat de afnemende services er hinder van ondervinden. De door service providers geleverde services worden in de vorm van WSDL documenten in UDDI directories opgeslagen die vervolgens door consumerende services/applicaties bevraagd kunnen worden. Zie figuur 4. De Service provider plaatst na installatie zijn lokatie (URL) en zijn te leveren functionaliteit, bijvoorbeeld een Vak Informatie Systeem, beschreven in een WSDL document in een UDDI Directory. Stap 1. Een service die gebruikt wenst te maken van deze aangeboden service bevraagd deze Service Directory (stap 2) en kan op basis van de gevonden lokatie en het WSDL document (stap 3) de communicatie met de Service Provider aangaan (stap 4).



Figuur 4: Publicatie van aangeboden services in een UDDI directory[9]

[9] Overgenomen uit: http://www.service-architecture.com/web-services/articles/web_services_explained.html

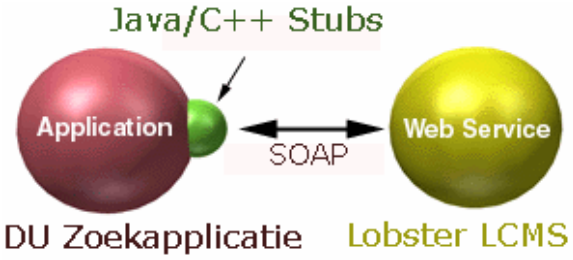
Het nieuwe aspect van SOA is dus dat we eindelijk losgekomen zijn van onderliggende technische specificaties en platformen en we ons bij de definitie en het ontwerp van de services kunnen laten leiden door de bedrijfsprocessen die er mee geïmplementeerd moeten worden.

Implementatie van een Service Oriented Architecture

OK, we hebben het concept van SOA begrepen. Na modules, objecten en componenten nu services. En liefst geen implementatie van technische low-level funkties als zip/unzip, OpenFile/CloseFile, maar implementatie van bedrijfsentiteiten als Klant, Student, Cursus, Assessment, Bestelling. We moeten dus services ontwikkelen of service adaptors kopen of bouwen op onze bestaande applicaties en klaar is Kees? Onze ontwikkelaars naar een WSDL/SOAP cursus sturen?

Nee dus. Zo eenvoudig is het niet. Web services zijn weliswaar relatief eenvoudig te maken, je bouwt er zo tien op een middag. Maar wat er dan bereikt is, wordt een ander soort SOA: Spaghetti Oriented Architecture[10].

Met moderne software ontwikkeltools als Oracle Jdevelopper bijvoorbeeld kan met behulp van ingebouwde wizard op basis van een service specificatie in de vorm van een WSDL snel Java code gegenereerd worden (Java stubs), de de Java calls automatisch omzet in de benodigde SOAP messages die via HTTP naar de service gestuurd worden. Ook het afhandelen van de terugestuurde respons mesages wordt afgehandeld door automatisch gegenereerde Java code. Voor de ontwikkelaar is de service als een Java klasse benaderbaar, hoewel de service fysiek op iedere willekeurige server op het internet gehost kan zijn. Een voorbeeld van dergelijk gebruik is het door de DU gebruikte Learning Content Management Systeem (LCMS), dat in de vorm van een Web Service interface toegankelijk is. Een desktop client applicatie (geschreven in C++) voor het produceren van e-learning content, de LearneXact Packager, benadert dit LCMS (genaamd de Lobster) als service. Maar een zogenaamde Learning Object Repository, een applicatie op maat gemaakt voor de DU, is geschreven in Java en maakt ook gebruikt van deze zelfde LCMS service via Java stubs die door een ontwikkeltool zijn gegenereerd.



Figuur 5: Illustratie van een harde point-to-point koppeling tussen en applicatie en een service met automatisch gegenereerde stubs

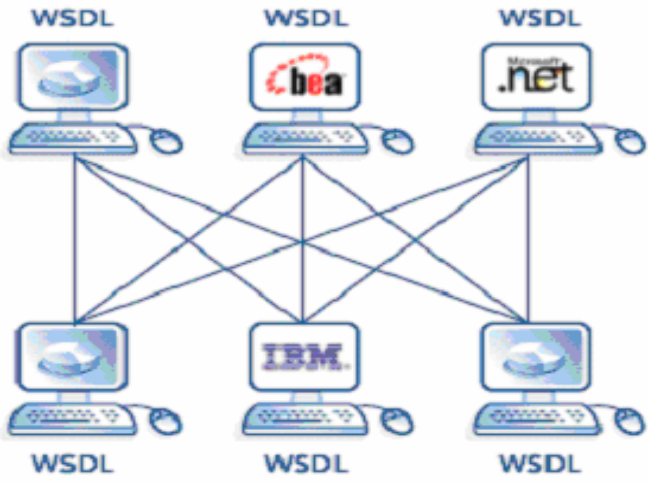
Een soortgelijk point-to-point koppeling is ontwikkeld voor de plagiaaatdetectie webservice die door Ephorus wordt geboden. In TeleTOP is ‘harde’ code op basis van de Ephorus WSDL toegevoegd waarmee de communicatie tussen TeleTOP en Ephorus geregeld wordt.

[10] Zie: ‘Five SOA myths, forever shattered’ van Joe McKendrick

Dit soort harde koppelingen, waarbij verbinding met de afgenomen service vast gekodeerd is in bijvoorbeeld Java of C++ code worden point-to-point verbindingen genoemd.

Bij koppelingen van meerdere service consumenten, zeg N stuks, met meerdere service producenten, zeg M stuks, ontstaat een illustratie van het zogenaamde N*M point-to-point probleem. Er zijn N*M harde koppelingen geproduceerd met behulp van (Java/C++) codestubs bijvoorbeeld. Ter illustratie de case waarbij drie verschillende leeromgevingen, laten we zeggen Sakai, Blackboard en Studyweb, services afnemen van drie vak informatie systemen die voor drie onderwijsinstellingen de cursuskatalogus bevatten. We hebben nu wel het voordeel dat we vakinformatie van drie instelling kunnen integreren in de drie leeromgevingen. Ook hebben we het voordeel dat we dit kunnen zonder kennis van onderliggende applicaties of technologie. Maar de hoeveelheid extra code dreigt al snel onbeheersbaar te worden als er bijvoorbeeld een partner aan het netwerk toegevoegd worden, bijvoorbeeld een regionale HBO instelling waarvan we de vakinformatie ook willen integreren.

Stel we hebben in 3TU verband te maken met drie leer- en werkomgevingen, Sakai, Blackboard en StudyWeb. En we hebben per instellen een eigen vak informatiesysteem. Als we vanuit Twente vakinformatie uit de 3 vakinformatiesystemen willen presenteren zouden er vanuit Sakai 3 koppelingen tot stand gebracht moeten worden. Idem voor het Delftse Blackboard en StudyWeb van Eindhoven. Er zouden dan 9 point-to-point koppelingen ontwikkeld en beheerd moeten worden. Willen we ook nog koppelingen met roosterinformatiesystemen en studentinformatiesystemen realiseren dan wordt de hoeveelheid verbindingen al snel onbeheersbaar.



Figuur 6: Illustratie van het NxM koppelingsprobleeem, overgenomen uit “Service Mediation” van TIBCO[11]

Services op deze manier inzetten vergt weinig bijzondere inzet of opleiding. Zoals al gezegd leveren de meeste ontwikkeltools voorzieningen voor automatische generatie van code om de services aan te spreken. Er is ook geen bijzondere infrastructurele voorziening nodig, de gebruikelijk applicatie servers kunnen gebruikt worden om de services te hosten.

[11] Zie: http://www.tibco.com/resources/solutions/soa/esb_for_soa.pdf

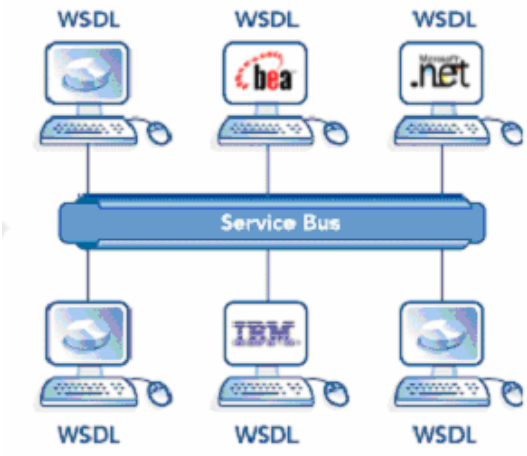
SOA ICT infrastructuur beheers laag

Als we dit point-to-point scenario blijven volgen voor de inzet van services hebben we al snel het paard achter de wagen gespannen. Alle winst die we hoopten te boeken door hergebruik van stabiele code in de vorm van services gaat verloren door de beheersproblematiek van de N*M verbindingen. En de bedrijfsprocessen zijn nog steeds hard gecodeerd in de applicaties, en bij te voorziene steeds vaker voorkomende wijzigingen van deze processen zal er voortdurend een beroep op schaarse ontwikkelaars gedaan moeten worden. Een uitbreiding van de harde koppelingen tussen services met een UDDI service directory zoals eerder beschreven levert een eerste stap in de richting van een SOA infrastructuur beheerslaag en maakt het mogelik dynamisch services te vervangen of te verplaatsen. Maar nog steeds zijn de bedrijfsprocessen hard in de code ingebouwd. Ook zouden we eigenlijk graag zaken als security, service monitoring en toegang tot de services afhankelijk van de consumerende service flexibel willen beheren.

Een oplossing voor dit probleem is het inzetten van een zogenaamde SOA beheerslaag in de vorm van een Enterprise Service Bus met service discovery, security e.d. waarmee flexibel businessprocessen gemodelleeerd en gewijzigd kunnen worden en ook de monitoring en accessmanagement uit de applicatie's getrokken wordt.

De Enterprise Service Bus

SOAP, Web Services Description Language (WSDL) en HTTP hebben in belangrijke mate bijgedragen aan een component specificatie die platform onafhankelijk is, en dus op de twee belangrijkste platforms, J2EE en .NET, alsmede door verschillende grote leveranciers van applicaties ondersteund wordt. Maar implementatie deze technologie alleen is niet voldoende om van een SOA te spreken. Een sleutelconcept bij implementatie van een bedrijfsbrede SOA is een Enterprise Service Bus, middleware waarmee o.a. de schaalbaarheid van de vele point-to-point verbindingen tussen services beheersbaar wordt gemaakt en ook het beheer en de beveiliging van de services geregeld kan worden.

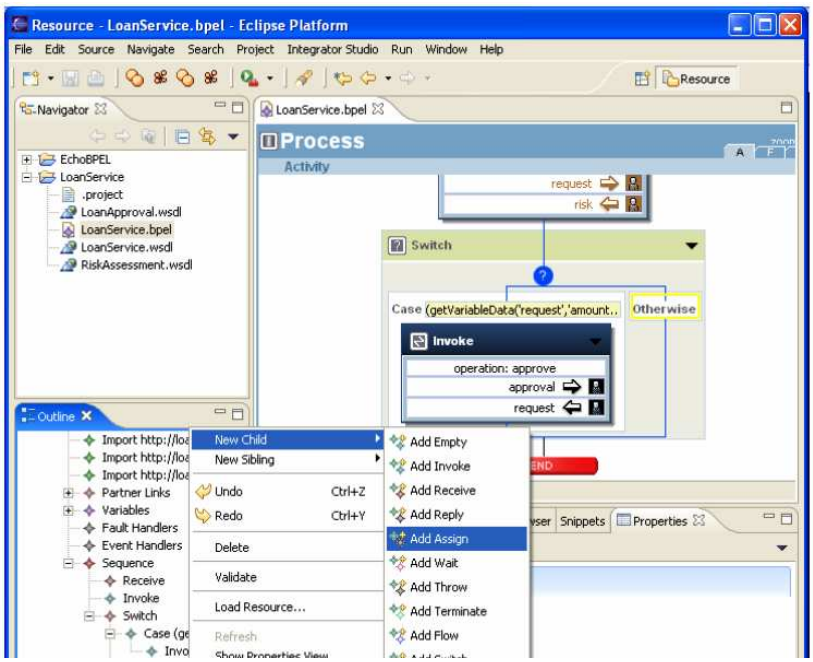


Figuur 7: Service Bus bij het NxM koppelingsprobleeem, overgenomen uit "Service Mediation" van TIBCO[12]

[12] Zie: http://www.tibco.com/resources/solutions/soa/esb_for_soa.pdf

Er bestaat nog geen algemeen geaccepteerde definitie van ESB. Het concept bestaat al een aantal jaren, is in 2001 door Sonic Software geintroduceerd en inmiddels heeft iedere grotere leverancier wel een ESB in het produktenpakket. IBM heeft zelfs twee versie, een Lite versie voor de beginners en een full-blown versie voor ondernemingsbrede inzet.
In het persbericht 'Sonic Software lanceert nieuwe SOA-Infrastructuur voor zakelijke gebruikers' van Progress Software, 10 mrt 2005 staat een redelijk heldere omschrijving, die hier integraal overgenomen wordt:

Een sleutelkomponent van een Enterprise Service Bus is een zogenaamde Business Process Execution Language (BPEL4WS of BPEL) processor[13]. Dit is een voorziening waarmee nieuwe bedrijfsprocessen ontworpen en uitgevoerd kunnen worden op basis van bestaande services zonder harde codering van de logica in bijvoorbeeld Java code voor ieder koppeling. De WSDL definities van de services kunnen bijvoorbeeld uit een UDDI directory gelezen worden in het BPEL ontwerp gereedschap en met meestal visuele editors en wizzard kunnen de bedrijfsprocessen gemodelleerd worden. De gemodelleerde bedrijfsprocesen of applicaties worden in de vorm van BPEL XML script bestanden vastgelegd die vervolgens door BPEL processor worden uitgevoerd.



Figuur 8: Voorbeeld van een BPEL Service Orkestratie modeller (http://www.CapeClear.com)

Naast ondersteuning van BPEL voor het modelleren van bedrijfsprocessen zijn een aantal andere voorziening nodig voor het implementeren van een SOA. Een ESB is verder pas een ESB als er naast BPEL ondersteuning de volgende eigenschappen door geleverd worden:

- gebaseerd op webservice-gerelateerde standaarden (WSDL, UDDI, BPEL etc.)
- document- ofwel XML-georiënteerd (services wisselen XML dokumenten uit)

[13] Zie: http://www.capeclear.com/technology/bpel/index.shtml

- content-based routing en filtering (om b.v. bepaalde berichten voorrang te geven, b.v. grote bestellingen)
- uitgebreide transformatiemogelijkheden (b.v. XSLT transformaties, b.v. om QTI 1.2 om te zetten naar QTI 2.0, een XML spec voor toetsitems)
- messaging faciliteiten (bv. Java Messaging Service (JMS))
- gedistribueerd beheer en verwerking

Waarom en wanneer moet je voor SOA kiezen?

De drie belangrijkste redenen om voor een SOA te kiezen zijn:

1. Integratie van bedrijfsapplicaties intern en extern

   Gebrekkige integratie van verschillende bedrijfsapplicaties, zowel intern als extern, wordt door bedrijven momenteel als het meest urgente probleem ervaren. In bijna elk onderzoek en enquete in de afgelopen jaren op de UT wordt deze problematiek ook steeds weer genoemd. Maar ook in 3TU verband zal de behoefte onstaan applicaties van de drie instellingen op enigerlei wijze te integreren, denk aan de drie leeromgevingen, of elkaars student informatiessystemen.

2. Best-of-breed componenten concept

   Door functionaliteit die meervoudig aanwezig is in applicaties uit die applicaties te halen of vanuit een enkele applicatie toegankelijk te maken als component of service kan de beste implementatie van die functionaliteit ingezet worden en verminderen de onderhouds en beheerslasten.

3. Steeds sneller wijzigende bedrijfsprocessen

   Steeds sneller wijzigende bedrijfsprocessen is een fenomeen waar ook instellingen in het Hoger Onderwijs steeds meer mee geconfronteerd zullen worden. Dit is een van de trends die in het DU project ELO Groei-en Verandermanagement nadrukkelijk naar voren is gekomen.

Het wanneer is moeilijker te beantwoorden. Er zal eerst zorgvuldig nagedacht moeten worden om de scope van de transformatie vast te stellen. Wat is de belangrijkste reden voor de UT om een SOA überhaupt in te voeren? Zien we echt de voordelen van integratie en betere ondersteuning van snel wijzigende bedrijfsprocessen ook op langere termijn? Kunnen we eerst alleen beginnen of zijn we afhankelijk van andere partners? Hoeveel en welke expertise hebben we in huis en wat ontbreekt er nog. Op welke termijn kunnen we die ontbrekende expertise in huis halen. Willen we het zelf of kunnen we het outsourcen? Zijn de applicaties die we hebben en die hun waarde hebben bewezen te voorzien van service adaptors? Willen we zelf nieuwe services kunnen bouwen of kopen we die?

Waar moet je beginnen?

Er worden (door IBM) vier niveau's van SOA adoptie onderscheiden[14]:

1- Implementeer individuele Web-services

Dit eerste niveau van SOA adoptie bestaat uit het analyseren van bestaande of nieuw aan te schaffen of te bouwen applicaties op herbruikbare functionaliteit die beschikbaar gemaakt kan worden voor inzet in een SOA door het ontwikkelen van een zogenaamde service adaptor beschreven in een WSDL dokument. Voor Sakai is een zo'n proces gaande waarbij de Sakai tools worden voorzien van service adaptor. Maar ook applicatie als VIST, TAST, ISIS e.d. zouden kandidaat kunnen zijn om voorzien te worden wat een SOA service adaptor. Er wordt hierbij zo veel mogelijk gebruik gemaakt wat bestaande code die zijn waarde bewezen heeft. Hiermee worden de ontwikkel- en beheerkosten beperkt gehouden omdat we niet van de grond af opnieuw hoeven te beginnen.

2- Service Oriented Integratie van Business Functies

Op het tweede niveau worden de eeste stappen gezet richting Service Oriented Architecture en integratie. Een aantal services die ontwikkeld zijn in de voorgaande stap worden door middel van verschillende integratie mechanismes samengevoegd ter ondersteuning van een eenvoudig bedrijfsproces binnen een bedrijfsdomein, bijvoorbeeld het onderwijsdomein. Bij voorkeur wordt ook een externe service gebruikt die gehost is bij een andere instelling. Er kunnen meerdere integratietypen getest worden, o.a.

- applicatie integratie m.b.v. point-to-point integratie en gegenereeerde code stubs
- procesintegratie door middel van de inzet van (open source) BPEL engines
- processintegratie door middel van inzet van een Lite versie van een ESB

3- Bedrijfsbrede IT transformatie

Hier wordt de SOA implementatie uitgebreid tot alle domeinen binnen de instelling, vanuit het onderwijsdomein richting het administratieve domein, bijvoorbeeld door koppeling met student informatie systemen en het e-Research domein.

4- On Demand Bedrijfstransformatie

Hetr laatste niveau van adoptie van een Service Oriented Arcitecture wordt gekenmerkt door een strategische beslissing om alle bestaande en nieuwe bedrijfsprocessen instellingsbreed met behulp van SOA te transformeren.

[14] Zie: SOA Adoptie

HRM Eisen. Welke kennis moet ik in huis hebben?

Welke vaardigheden moeten we in huis hebben voor de verschillende niveau's van SOA adoptie? Dat hangt natuurlijk voornamelijk af van het ambitieniveau dat we ons stellen en welk niveau van SOA adoptie we als UT/ITBE willen nastreven en of er voldoende draagvlak binnen de organisatie is om de overgang naar een SOA te maken.

Alleen kennis van b.v. de Java pogrammeer taal in onvoldoende om zelf de eerste stap te zetten. Weliswaar kunnen met modere ontwikkeltools redelijk snel WSDL wrappers gegenereerd worden die (bestaande) Java beans voorzien van een Web Service interface (service adaptor), maar analytische vaardigheden en kennis de WSDL specificatie, XML/XSLT, opertionele kennis van modellerings gereedschappen als UML (Universal Modelling Language) en de IMS General Webservices Specification zijn wenselijk on kwalitatief hoogwaardige en herbruikbare services te ontwikkelen.

In tweede instantie zal kennis van UDDI directories om lokatieonafhankelijkheid en uitwisselbaarheid van services mogelijk te maken en kennis van Xpath/Xquery om bijvoorbeeld uit te wisselen XML documenten te kunnen transformeren en relevante informatie uit binnenkomende XML documenten te kunnen extraheren.

Bij SOA adoptieniveau 2, waarin een eenvoudig bedrijsproces met behulp van enkele services wordt geimplementeerd zal kennis van BPEL en een eenvoudige Lite versie van een ESB nodig zijn.

Voor fase 3 van het adopteren van een SOA komt invoering van een instellingsbrede SOA infrastructuursbeheers voorziening in de vorm van een professionele Enterprise Service Bus aan de orde. Het invoeren hiervan en de kennis voor het beheer ervan zullen zowel bij de afdeling I&A als bij T&S een extra professionaliseringslag eisen. In overleg met leveranciers van in gebruik zijnde applicaties zullen deze waar nodig voorzien moeten worden van service adaptors.

Het merendeel van de bestaande en nieuwe bedrijfsprocessen op een service oriented wijze implementeren in de laatste fase van SOA adoptie betekent dat een groot deel van de organsatie (ITBE) de SOA manier van ontwerpen geinternaliseerd moet hebben en hergebruik van bestaande services als leidend principe zal moeten hanteren. Het kan wenselijk zijn om binnen het ITBE een SOA Unit met daarin vertegenwoordigers van I&A en T&S onder leiding van een IT architect te vormen.

Omgevings- en randfactoren voor de UT

Bij de keuze van de implementatie van een SOA heeft de UT met een aantal omgevings factoren te maken. Voor het onderwijsdomein is dat natuurlijk de in het ELO Keuzetraject aanbevolen Open Source leeromgeving Sakai, die service orientatie en standaarden als belangrijk ontwerpparadigma heeft gekozen. Hoewel hier in de praktijk pragmatisch mee wordt omgespringen. Een aantal Sakai tools is gebaseerd op al bestaande code van in het Sakai programma meewerkende instellingen en die code is doorgaans nog niet voorzien van service adapters. Maar nu veel code herschreven is worden langzamerhand begonnen de tools voorzien van service adaptors. Zo heeft het Assessment Tool SAMIGO een eigen specifieke API, maar is

met de volgende release voorzien van een service adaptor. Is het lopende CBUS project worden van alle met Sakai meegeleverde tools de service adaptors in kaart gebracht.

Met het ten einde lopen van de Digitale Universiteit per 1 januari 2007, zal de UT voor de co-financiering van vele projecten een beroep doen op SURF. En SURF heeft een jaar geleden besloten partner te worden in het JISC e-Framework. In dit e-Framework worden op projectbasis services ontwikkeld met name voor HO instellingen die optimaal ingezet kunnen worden in een SOA. In het eeerder geneoemde DU project ELO Groei-en Verandermanagement is een inventarisatie gemaak van JISC projecten warin aan service ontwikkeling worden gewerk. Een vijftigtal services zijn kort beschreven in de delverables van dit project. Het is te voorzien dat ook SURF service orientatie als belangrijke technische randvoorwaarde zal meegeven aan nieuwe project.

Conclusies

Service Oriented Architecture is zeker geen hype en het is voor de UT aan te bevelen hier serieus praktijkervaring mee op te doen alvoren het instellingsbreed in te voeren.

SOA beperkt zich niet het domein Student en Onderwijs, lijkt dus bij uitstek een verantwoordelijkheid van de aanstaande informatiemanager.

In het project ELO Keuze is geadviseerd om Sakai te onderzoeken als mogelijke opvolger van TeleTOP. Een belangijk kriterium hierbij was de beloofde interoperabiliteit van Sakai. In de definitiefase van CBUS is Sakai alleen out-of-the-bix geinstalleerd. Pas in de pilotfase zal hier iets konkreters over gezegd kunnen worden aan de hand van de trials.

Aanbevelingen voor de vervolgaktiviteiten

Een algemene opmerkingen over de invoering van SOA. Er zullen eerst investeringen gepleegd moeten worden voordat de voordelen ervaren kunnen worden. Niet alleen investeringen in software e.d. maar met name ook in de personele sfeer om de benodigde competenties in huis te halen. Ingeschat wordt dat ook personeel aangenomen zal moeten worden met een ander profiel dan momenteel beschikbaaar is. Te denken valt aan een bedrijfsinformaticus bijvoorbeeld. Verder wordt ingeschat dat UT brede invoering en realisatie van een informatie architectuur gebaseerd op SOA een proces van 4 tot 6 jaar zal zijn. Verder zal in de pilotfase van het CBUS project de belofte van Sakai t.a.v. interoperabiliteit nog gevalideerd moeten worden. Daartoe worden verderop een aantal trials voorgesteld.

Konkrete aanbevelingen voor de korte termijn:

- Ontwikkel een bedrijfsarchitectuur voor de UT. Neem hierbij als uitgangspunt het SURF Architectuur rapport. Gezien de reikwijdte en de veelomvattendheid van deze acitiviteit wordt ten sterkste aanbevolen om hiervoor een apart project te entameren en het niet als werkpakket onder te brengen in een (mogelijk) vervolg van het CBUS project. Het wordt aanbevolen dit project aan te laten sturen door de nieuw aan te stellen informatiemanager.

- Pas als deze bedrijfsarchitectuur formeel beschreven is en de benodigde bedrijfsentiteiten bekend zijn kunnen de (web)services ontwikeld of aangeschaft worden waarmee de op SOA gebaseerde informatiearchitectuur gerealiseerd zal worden. Ook de

uitvoering van deze aanbeveling zou bij voorkeur via de eerder genoemde informatiemanager aaangestuurd moeten worden.

- Ter validatie van bovengenoemde aanbevelingen wordt geadviseerd in de pilotfase van het CBUS project een aantal trials op waarbij met SOA adoptieniveau's 1 en 2 op kleine schaal ervaring wordt opgedaan:

    - Identificatie van een project met een UT scope, bijvoorbeeld met de beoogde opvolger van VIST. Voorzie deze van een service adaptor en breng een point-to-point koppeling tot stand met een Sakai tool. Doelstelling: ervaring opdoen met ontwerpen en bouwen van Service Adaptors op bestaande (legacy) applicaties.

    - Identificatie van b.v. een 3TU businessprocess of een intern UT proces, waarin bExchange ingezet wordt en dat met SOA geimplementeerd zou kunnen worden. Als voorbeeld het weergeven van Exchange kalenderinformatie in een Sakai roosterkomponent. Doelstelling: Ervaring opdoen met een BPEL engine en bijvoorbeeld de Oracle Enterprise Service Bus. Deze trial kan tevens dienen om informatie te verschaffen over interoperailiteit van Exchange (en Sakai).

- Bepaal en realiseer een optimale organisatie bijvoorbeeld in de vorm van een instellingsbreed (of misschien wel 3TU) WebService/SOA Team, Quality Assurance Team, waarin de verschillende disciplines verenigd zijn, van de bedrijfsinformaticus,
- via de (BPEL) Process Engineer to de J2EE service ontwikkelaar.

- Beleg de monitoring van interoperabiliteit d.m.v. toepassing van specificaties en standaarden op voldoende hoog niveau in de organisatie. Dwing dit af bij elk toekomstig project en/of aankoopbeslissing (toekomstige informatiemanager?). Werk ook richtlijnen uit in bijvoorbeeld een (update) van de Bouwverordening.