

Eenvoud is een complexe zaak

Architectuur en de beheersing van complexiteit

Serge Bouwens

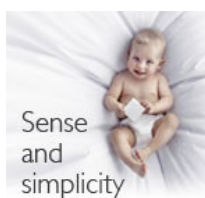
Dit artikel is bedoeld voor architecten. Het gaat in op het voor organisaties steeds groter wordende knelpunt van complexiteit, en de rol die de architect kan spelen bij het managen van complexiteit. Daartoe worden enkele praktisch toepasbare instrumenten aangereikt.

Complexiteit als kritieke factor in organisaties

De afgelopen jaren is het besef gegroeid dat organisaties in problemen raken als de complexiteit ze ergens boven het hoofd groeit. Hetzij bestuurlijk, hetzij in de informatie-huishouding of processen; zogauw de menselijke maat wordt overschreden functioneert het systeem niet meer. Een vluchtige blik op Google leert dat dit besef niet uit de lucht komt vallen: de zoekterm "complexity reduction" geeft rond de 100.000 hits. Zoomen we verder in op IT en organisaties, dan blijkt dat het reduceren van complexiteit in veel organisaties top prioriteit is, en dat dit inmiddels optelt tot macro-economische proporties: "De beheerlast als gevolg van complexiteit is in 15 jaar verschoven van 25 naar 80% van het IT budget." [Economist, 2004]. En Kearney komt in [AT Kearney] tot de conclusie dat complexiteit management geen probleem is van productie, marketing, architectuur of financiën: het is een business probleem.

Er is alle reden op aan te nemen dat de toenemende complexiteit een blijvend probleem is. De kwantitatieve kant van de zaak is in veel opzichten dé complicerende factor. Meer klanten (globalisering), meer relaties (netwerkmaatschappij), meer regels, meer uitzonderingen, meer keuzes, meer specialismen; meer meer meer. Al met al lijkt het een natuurwet dat de complexiteit in een (eco-) systeem altijd toeneemt. Daarmee zitten we in een soort wapenwedloop met het systeem dat we met zijn allen creëren. Een wedloop die we dreigen te verliezen.

Niet alleen voor bedrijven is complexiteit een issue geworden. Ook veel consumenten kunnen het nog nauwelijks behappen. Veelzeggend is de nadruk op eenvoud die een aantal organisaties in hun communicatie benadrukken: "Sense & Simplicity" (Philips), "Interpolis Glashelder", "Leuker kunnen we het niet maken, wel gemakkelijker" (Belastingdienst). Of kijk naar de Google zoekpagina: die blinkt uit in eenvoud en functionaliteit – alle opties zijn verstopt onder de motorkap of onder Geavanceerd Zoeken.



Bij het streven om de zaken eenvoudig te houden speelt het probleem van de communicerende vaten in hoge mate mee. Elke knop die we aan de gebruikerskant weg halen, betekent extra complexiteit in de back office. Managen van complexiteit vraagt dan ook een brede kijk en het stellen van prioriteiten.

Complexiteit en architectuur

Complexiteit heeft vele gedaantes. Spaghetti-infrastructuur, processen met honderden uitzonderingssituaties, globalisering, (Europese) wetgeving, of domweg de enorme volumes. Iets wordt pas complex gevonden als het overzicht ontbreekt, als er effecten optreden waarvan de oorzaak aan onze aandacht is ontsnapt; kortom als we het niet meer beheersen.

Voor architecten is het van belang een gevoel te hebben voor de ernst van de situatie. Waar gaan we het nog redden met optimalisatie, en waar is een fundamentele discussie nodig? In dat laatste geval duurt het meestal enige tijd voordat de analyse gemaakt is. Conklin reikt ons in [Conklin, 2003] een aantal socio-technische inzichten aan die duidelijk maken waarom deze analyse lastig is. Hij onderscheidt twee soorten problemen: 'Gemene' (Wicked) problemen vs 'Tamme' (Tame) problemen. Gemene problemen hebben kenmerken als: je weet pas achteraf of een oplossing echt werkt; elk gemeen probleem vraagt om een unieke oplossing; en: je kan geen alternatieven uitproberen. U hoort het al: IT projecten betreffen meestal gemene problemen. Je kan van buiten niet zien hoe gemeen een probleem is. We zijn geneigd problemen als tam te zien, en dat leidt tot problemen: bij gemene problemen helpt de aanpak van bestuderen-begrijpen-temmen niet. Conklin maakt een heldere analyse en komt tot de conclusie:

"Complexiteit wordt veroorzaakt door drie dingen: 1. de aard van het probleem, 2. de aard van de oplossing en 3. de aard van de organisatie die er mee kampt."

Daarmee wordt een probleem oplossen een sociaal proces. "Because of social complexity, solving a wicked problem is fundamentally a social process. Having a few brilliant people or the latest project management technology is no longer sufficient". Om op een effectieve manier met (organisatie- of architectuur-)problemen om te gaan is het zaak om te kunnen analyseren waar de complexiteit vandaan komt. Er is derhalve behoefte aan tools en technieken om deze analyse te ondersteunen. Dit artikel wil hier een bijdrage aan leveren.

Een duidelijke indicatie dat complexiteit een probleem is, is als in een inleidend gesprek de vraag "Wie weet hier alles van, wie heeft het overzicht?" niet beantwoord wordt, en er bij nader inzien een hele rits (druk bezette) mensen bij elkaar moet worden gebracht. En inmiddels is er ook geen betere openingszin om een oplossing te presenteren dan: "Er is gelukkig een simpele oplossing voor dit probleem". Dat krijgt meer handen op elkaar dan goede of goedkope oplossingen.

De rol van de architect

Welke bijdrage kan de architect leveren aan het managen van complexiteit? Mits hij goed staat opgesteld en zich bij inrichtingsvraagstukken rekenschap geeft van de effecten die architectuur heeft op de complexiteit, kan de architect een sleutelpositie innemen in het complexiteitsmanagement.

Architecten zijn van nature structuurdenkers. Ze zijn goed in de ontwerpbenadering, en het denken in structuren en abstracties gaat hen beter af dan de meeste mensen om hen heen. Maar in deze kernkwaliteit zit ook een valkuil verscholen. Abstractie leidt tot het onzichtbaar maken van de complexiteit van de onderliggende diversiteit en aantallen. Waar de architect de samenhang nog wel ziet, is de rest van het publiek allang afgehaakt. Praktische problemen worden nogal eens over het hoofd gezien of weggewuifd met een luchtig "dat is een kwestie van implementatie/techniek". De organisatie verdrinkt in complexiteit en architectuur draagt daar graag aan bij... Het lijkt mij dat er een relatie is tussen deze valkuil en (complexiteit veroorzakende) producten van architecten.

Zaak is dus om de kracht te behouden die in de kerncompetentie zit, en tegelijk de valkuil te vermijden. En dat kan, bijvoorbeeld door slim gebruik te maken van architectuurprincipes, en het werken met patroonanalyse. Hieronder volgen een tweetal instrumenten die kunnen helpen bij het grip krijgen op complexiteit.

Architectuurprincipes

Architectuurprincipes zijn een krachtig instrument voor de architect. Het is een middel om vorm te geven aan de bedrijfsvisie zonder dat het nodig is om die geheel in modellen uit te werken. Volgens het boekje worden architectuurprincipes afgeleid uit de bedrijfsdoelstellingen, en staan ze dus – eenmaal vastgesteld – nauwelijks ter discussie. Toch vraagt de toepassing om enig gezond verstand. Geen enkel principe moet worden volgehouden tot de dood er op volgt, maar het is maar al te gemakkelijk om dit soort regels op de autopilot toe te passen.

Verschillende architectuurprincipes zijn in de kern ook bedoeld om zaken simpel te houden. Praktijkvoorbeelden die meestal ook écht zo uitwerken zijn:

- Productinnovatie is waar mogelijk gebaseerd op assemblage en parametrisering van elementaire producten tot een samengesteld product.
- Pakketten worden zonder aanpassingen ingezet; het proces wordt indien nodig aangepast.
- Marktstandaarden worden zoveel mogelijk gevolgd.

Ter illustratie dat – op zich verstandige – principes ook complexiteit in de hand kunnen werken, enkele voorbeelden:

Voorbeeld 1: Ontwerp vanuit ketenperspectief

Organisaties zijn onderdeel van een logistieke of administratieve keten. De tijd dat organisaties als eilanden konden worden bestuurd, ligt achter ons. We denken in ketens en ecosystemen. De praktijk leert evenwel dat de omslag naar ketenoriëntatie een lastige is. De belofte van betere processen, informatiehuishouding en dienstverlening wordt vooraf gegaan door nieuwe regelgeving, discussies over taken en bevoegdheden, en eigendoms- en privacy perikelen. Vanuit het perspectief van de individuele actoren wordt het in eerste instantie alleen maar complexer. Het knelpunt is vaak dat veranderingen vooruit lopen op bestuurlijke herinrichting. Het principe 'Ontwerp uit ketenperspectief' werkt vaak uit als: 'ken de hele keten', in plaats van 'ken uw plaats in de keten'. Vanuit dit laatste is het aanzienlijk makkelijker om het aantal betrokken partijen te beperken, zowel in besluitvorming als in ontwerp en uitvoering.

Voorbeeld 2: Gebruik standaardpakketten waar mogelijk

Naar mate de ICT wereld professionaliseert wordt het voor organisaties aantrekkelijker om delen van hun ICT uit te besteden. Dat betekent o.a. dat je minder ontwikkelkennis in huis hoeft te hebben en dat je pakketten als een kant-en-klare black box in je enterprise architectuur in kunt passen. Op andere fronten wordt het wel moeilijker: er bestaat opeens een noodzaak om de uit te besteden component écht te beheersen, alsmede je IT governance (vendor management) ingericht te hebben. De architecten moeten kennis van externe ICT producten hebben, hoe deze in te passen in de bestaande architectuur, en hoe deze te combineren met maatwerk.

Voorbeeld 3: Doublures in functionaliteit voorkomen.

Sturen op kosten en efficiency is meestal verstandig. Door het samenvoegen van twee applicaties die functioneel hetzelfde doen worden absoluut voordelen gehaald in termen van beheer. Maar de vraag is of kosten het kritieke punt zijn. De complexiteit die je introduceert doordat er bij veranderingen meer moet worden afgestemd heeft gevolgen voor de veranderbaarheid en slagvaardigheid van de organisatie. Dat kan voor één project best meevallen, maar als het jarenlang zo doorgaat – en dat lijkt mij eerder de regel dan de uitzondering – wordt het allemaal wel erg ingewikkeld. Niet voor niks hoor je in veel organisaties iedereen kreunen en steunen over de grote aantallen mensen die betrokken moeten worden bij op zich kleine wijzigingen. Waar marktkansen om snelheid vragen is dit dan ook de belangrijkste reden om nieuwe business in een aparte unit onder te brengen: de winst doordat je de complexiteit van het moederbedrijf in procedures en IT infrastructuur achter je

kunt laten is vele malen belangrijker dan de efficiency verliezen doordat zo'n nieuwe unit in het begin nogal chaotisch functioneert en zelf het wiel uitvindt.

Bovenstaande voorbeelden geven aan dat bij het toepassen van architectuurprincipes altijd in een context moet gebeuren. Een principe is natuurlijk gedefinieerd om *altijd* geldig te zijn, maar vaak valt de invulling met amendementen goed bij te sturen. En ook staan principes onderling nogal eens op gespannen voet. Ik zie principes dan ook eerder als het begin van een discussie, dan als het einde ervan. De aanbeveling is dus: Houd oog voor de consequenties voor de complexiteit in de verschillende disciplines (niet alleen IT, maar ook besturing, processen, wetgeving, ...).

Patronen

Het gebruik van (de juiste) ontwerpprincipes is kan de eenvoud ten goede komen. In de praktijk is het zelden zo dat ontwerpprincipes a-priori bepaald worden. Ze komen gaandeweg bovendrijven, of er is een brede set waar een relevante subset uit gekozen moet worden. De kunst is om ze zo vroeg mogelijk als essentieel te herkennen. Daarvoor is een abstractere analyse nodig, minder toegespitst op de specifieke situatie, en meer gericht op het herkennen van patronen hoe men met complexiteit omgaat, of zou kunnen omgaan. Een instrument om dat te doen is aan de hand van een typering van patronen in termen van hun werking. Het helpt om zaken beter te begrijpen, en het bakent de oplossingsruimte verder af. Eerst wat theorie, daarna de toepassing aan de hand van een voorbeeld.

Basispatronen

Wie ervaring met iets heeft – of het procesontwerp is of het vangen van criminelen – heeft de prettige ontdekking gedaan dat zaken die eerder volstrekt los van elkaar leken te staan, bij nader inzien grote mate van overeenkomst blijken te hebben. Dat wil niet zeggen dat je er altijd receptmatig mee om kan gaan, maar de herkenning van het onderliggende patroon helpt wel. In de software engineering is het gebruik van patronen inmiddels een basiscompetentie geworden. Het gebruik van het begrip patronen hier is nog een stap abstracter.

Ik onderken verschillende basispatronen – veranderstrategieën zo je wilt - om met complexiteit om te gaan:

(I) Meer

De essentie is dat we structuur toevoegen om grip te krijgen, zonder dat we het begrippenkader fundamenteel wijzigen:

1. Uniformeren. De eenvoudigste vorm – en bijzonder effectief als het toepasbaar is – is uniformeren. We doen het hier net zoals daar. Dat is gemakkelijk, we hoeven het maar één keer te bedenken, en dat scheelt dus tijd en geld. Standaardisatie – lees ook: hergebruik en naäpen - is met name in de IT een geliefde methode. Uniformeren is helaas niet altijd mogelijk; soms is de variatie te groot.
2. Ordenen. Ordenen laat meer variatie toe dan uniformeren. In organisaties is het aanbrengen van een hiërarchie een bekend voorbeeld. Het segmenteren van de markt of het toepassen van een functiebouwwerk zijn eveneens middelen waarbij een grote en diverse populatie in behapbare groepen wordt ingedeeld. Met name in de (beta-) wetenschap is het toevoegen van structuur zo goed als core business. Ook architecten hebben in hun bagage zaken als architecturen, views en nog 101 andere modellen om de wereld te ordenen.

Het toevoegen van iets aan een situatie die problematisch is vanwege zijn complexiteit is op het eerste gezicht paradoxaal. Het vraagt dan ook een bepaald kennisniveau van de betrokkenen, iets dat we vaak onderschatten omdat we als vakman ingevoerd zijn in de materie en de manier van denken. Pas als we het aan buitenstaanders moeten uitleggen krijgen we daar enig gevoel bij. Met andere woorden: een deel van de complexiteit zit verstopt in het intellectuele kapitaal van de organisatie.

(II) Minder

Eenvoudiger dan iets toevoegen lijkt dan ook om zaken weg te laten. We betalen dan een prijs omdat we dingen missen, maar het wordt in ieder geval op het oog (!) eenvoudiger. De uitdaging is natuurlijk om de juiste zaken weg te laten: "In der Beschränkung zeigt sich der Meister".

1. Abstraheren. Het met betrekking tot een bepaald gezichtspunt weglaten van niet-relevante informatie. De kern van het werken met patronen, en misschien ook wel van het werken met architecturen.
2. Delegeren. Zeer effectieve bestuurlijke maatregel. De complexiteit is er nog wel, maar *ik* zie het niet meer. Automatiseren kan je zien als een voorbeeld van delegeren (van mens naar machine). Iets breder getrokken kan je hier ook de functionele organisatie, en het uitbesteden c.q. inkopen onder laten vallen.
3. Afbakenen. Geliefd bij projectleiders die bewust stukjes werkelijkheid negeren omdat de verandercapaciteit van het project maar al te eindig is. Het beperken van de scope is in feite ook wat we doen als we één specifieke view of architectuur maken. Een integrale 'COPAFIJT-brede' benadering mag dan volgens het boekje zijn, het is ook heel complex. En ook niet altijd nodig. In het modelleren zien we afbakenen ook terug in de vorm van "Information hiding".
4. Faseren. Eigenlijk Afbakenen in de tijd. Iets dat we allemaal intuïtief doen. Niet alles tegelijk. Met name grote projecten hebben nog al eens de neiging om alles tegelijk te adresseren. Dat voegt complexiteit toe in termen van veel mensen en dus afstemming, wat ook weer kan leiden tot onduidelijke prioriteiten en een agenda-infarct bij de projectleiding. In ieder geval voegt het in de lucht houden van alle ballen weer complexiteit toe in je hoofd en je agenda.
5. Stoppen. Heerlijk simpele manier om de complexiteit terug te dringen. Het gewoon niet meer doen. Hoewel, 'gewoon'? Je beslist niet zó maar om een markt niet meer te bedienen, of om een product te discontinueren. En hoe moeilijk is het niet om een applicatie daadwerkelijk uit te faseren, of om een procedure de nek om te draaien. Er is altijd wel een uitzonderlijk geval dat dat weer problematisch maakt. Het toestaan van uitzonderingen is een grote veroorzaker van complexiteit.

(III) Anders

Of complexiteit aanwijsbaar is – zoals entropie in de thermodynamica - is voor mij een open vraag. Het is ook iets subjectiefs. Parafrazerend: "Complexity is in the eye of the beholder". Voor de een kan het bedrijfsprocesmodel een logisch geheel van mensen en middelen zijn, voor de ander is het een ondoorzichtige kluwen van activiteiten.

1. Paradigma veranderen. Fundamentele verandering van hoe we een probleem beschouwen, en (dus) van hoe we het oplossen. Een decennium terug zijn daar mooie boeken over geschreven. Een bekend voorbeeld is bijv. het vervangen van het concept van centrale besturing door decentrale besturing, wat we in min of meer hetzelfde tijdperk terugzien in organisaties (empowerment), ICT architecturen (decentrale IT voorzieningen per business units), het verdwijnen van centraal geleide economieën (val van het communisme), computer architecturen (van CPU naar gedistribueerde systemen), en bestuurskunde (einde van de maakbare samenleving). Steeds weer komt de impuls tot veranderen vanuit het onvermogen om de complexiteit vanuit één punt te regelen. Met een beetje goede wil kan de tegenovergestelde beweging ook gezien worden als gedreven door complexiteitsreductie: decentraliseren t.b.v. effectiviteit (lees: we kunnen het centraal niet meer snel genoeg overzien); centraliseren t.b.v. efficiency (lees: we kunnen de wildgroei niet meer managen).
2. Taak veranderen. Met name als de omstandigheden (nieuwe concurrenten, wetgeving, explosieve marktontwikkelingen) ons dwingen om zaken fundamenteel anders te gaan doen. Als handmatige processen niet meer werken, als concurrenten het tien keer goedkoper doen, of als we onze positie in de keten moeten gaan herdefiniëren.

Bovenstaande indeling is tamelijk willekeurig en gebaseerd op hoe een veranderstrategie bijdraagt aan vereenvoudiging. Bekijk je verandertrajecten integraal, dan is complexiteit in hoge mate een communicerend vat: complexiteit verplaatst vaker dan dat het verdwijnt. Een gekochte applicatie is even complex als een zelf gemaakte, maar het is niet meer óns probleem. Een bedrijfsgegevensmodel is een complex ding maar vermindert de ingewikkelde afstemmingsperikelen. Het is derhalve van belang om goed te begrijpen waar de complexiteit de bedrijfsdoelstellingen het meest frustreert. Waar capaciteit schaars is, is het delegeren van taken naar dat niveau niet de eerste keuze.

Nogmaals ten overvloede: de portee van dit verhaal is niet dat complexiteit te allen tijde vermeden moet worden. Maar wel dat dáár waar we problemen krijgen omdat de menselijke maat wordt overschreden, een oriëntatie op kosten of effectiviteit niet de juiste focus is.

Met deze basispatronen in de hand kan je analyseren of er vanuit het perspectief van complexiteitsmanagement een verandertraject de juiste kant op gaat. Een voorbeeld:

Voorbeeld

In een uitvoeringsorganisatie van de Nederlandse overheid ervaart men problemen in de handhaving. Er is politieke druk om efficiënter (minder mensen) en effectiever (minder fraude) te werken. De situatie is complex omdat er veel gegevens zijn, veel relaties, veel regels en (dus) ook veel interpretaties, maar dat is op zich niet uitzonderlijk. Een nadere analyse van hoe deze situatie zich in de komende 10-15 jaar dreigt te ontwikkelen toont aan dat de huidige complexiteit – nu al een factor die verandering tegenhoudt – enorm gaat toenemen: per jaar 20% meer regels, meer klanten en dus volume, meer veranderingen, meer taken en specialismen, grotere piekbelasting, meer vragen van burgers en bedrijven, en tegelijk de roep om simpele regels en een zero tolerance beleid.

Direct duidelijk is dat deze situatie niet met één maatregel kan worden opgelost. Oplossingen worden o.a. gezocht in (1) aanpassingen in wetgeving, (2) verdergaande automatisering in het volume proces, (3) experimenteren met specialistische teams, en (4) in samenwerking met marktpartijen.

Schematisch zou je de factor Complexiteit als volgt in beeld kunnen brengen:

Maatregel	Patroon	Effect op complexiteit in ¹							Toelichting op effect	
		C	O	P	A	F	I	J		T
(1)	Abstractie (raamwetgeving i.p.v. specifieke regelgeving); Stoppen (verplichtingen i.p.v. keuze mogelijkheden)	↓		↓↑	↓		↓	↓		Verwachting: veel discussie, klachten, incidentwetgeving
(2)	Uniformeren (van papier naar elektronisch); Delegeren (fraudedetectie door expertsystemen)			↓	↓		↓↑		↑	Informatiehuishouding wordt complexer door toename informatiebronnen
(3)	Ordenen (doelgroepbenadering); Afbakenen (specialisatie)		↑	↓↑			↑	↑	↑	Effectiviteit neemt mogelijk toe, maar de complexiteit

¹ Op de horizontale as zijn de COPAFIJT aspecten weergegeven. COPAFIJT is een veelgebruikt acroniem dat staat voor een integrale benadering. Hier staan de letters voor: Context en bedrijfsdoelstelling, Organisatie, Processen, Administratie, Financiën, Informatiehuishouding, Juridische aspecten, en Techniek.

Maatregel	Patroon	Effect op complexiteit in ¹							Toelichting op effect	
		C	O	P	A	F	I	J		T
	per regio)									ook.
(4)	Paradigma/Taak veranderen (controles i.s.m. commerciële dienstverleners); Delegeren (certificering)	↓	↑↓	↑↓	↑↓	↑↓	↓	↑↓	↑	Grote impact en veel effecten: in ieder geval is het verandertraject complex

Legenda:

De toelichting in de tabel is summier en heeft betrekking op de grijze cellen in dezelfde rij. Het is hier bedoeld als een voorbeeld. Bij elke pijl (↓: complexiteit neemt af; ↑: complexiteit neemt toe) hoort een analyse. Zo zal het inschakelen van de private sector bij de handhavingstaak (maatregel (4), kolom Organisatie) enerzijds leiden tot: minder personeel in de uitvoering en minder noodzaak om alle expertise binnen de eigen organisatie te hebben (vandaar de ↓). Anderzijds vraagt het om een stelsel van afspraken over taken en verantwoordelijkheden, toezicht op de handhavers, etc.: vandaar de ↑.

Merk op dat je de horizontale as (COPAFIJT) naar behoefte vervangen door processtappen of een andere indeling die je goed uitkomt. Er is sprake van een breed palet van complexiteitsreducerende maatregelen; toch kan het nuttig zijn om de lijst van patronen te gebruiken als een checklist om per COPAFIJT-aspect verdere maatregelen te bedenken. Maatregelen die nagenoeg geen vereenvoudigend effect hebben (3) zijn verdacht, en als er sprake is van abstractie, is er waarschijnlijk een onderliggend niveau waar die complexiteit belegd wordt.

Conclusies die je in dit voorbeeld zou kunnen trekken zijn:

- In het algemeen wordt de organisatie gespecialiseerder en kennisintensiever.
- In de technologie-hoek wordt het alleen nog maar complexer; daar zijn aanvullende maatregelen gewenst (zeker als je over het hele projectenportfolio hetzelfde patroon ziet ontstaan).
- Maatregel (4) – samenwerken met marktpartijen – is een paradigma verandering: de consequenties zijn talrijk, en het doorgronden daarvan is op zichzelf al complex. Het is onduidelijk hoe deze maatregel uiteindelijk gaat uitwerken in termen van complexiteit.

Conclusie

Complexiteit is een probleem dat naar verwachting steeds prominenter op de bestuurlijke agenda komt te staan. In de besluitvorming zal naast vragen over tijd en geld de vraag: "Kunnen we het allemaal wel behappen?" steeds vaker gesteld worden.

Architecten zijn in principe goed opgesteld om de beheersbaarheid van de complexiteit te bewaken en om innovaties zodanig vorm te geven dat organisaties niet vastlopen in spaghetti infrastructuren en onbegrijpelijke procedure handboeken. Door architectuurprincipes en maatregelen te toetsen op het effect op de complexiteit kunnen betere oplossingen worden bereikt. Het verdient daarom aanbeveling om complexiteitsreductie expliciet mee te nemen in business plannen of referentie-architectuur.

Serge Bouwens

Sogeti

serge.bouwens@sogeti.nl

Referenties

- [Economist, 2004] the Economist: "Make it simple: a survey of information technology", october 2004.
- [AT Kearney, 2004] AT Kearney: "A Survey on Complexity management across the supply chain", http://www.atkearney.com/shared_res/pdf/Complexity_Management_S.pdf
- [Conklin, 2003] J. Conklin: "Wicked problems and social complexity", White paper, CogNexus Institute, 2003. <http://cognexus.org>.