overdruk informatie juni'00

Rekenen met taal

Stef Joosten

Rekenen met taal

Een conceptuele techniek voor beheersbare samenhang

Stef Joosten

E-business, multichanneling, e-finance: infrastructurele investeringen in grote ICT-intensieve organisaties vragen om samenhangende kennis, inzicht en overzicht. Maar hoe krijgen ICT-architecten meer grip op de samenhang? Hoe maken zij hun architecturen concreet? Hoe beoordeelt een organisatie of een nieuw ontwerp past bij de bestaande infrastructuur? In dit artikel bespreekt Stef Joosten een techniek met de naam CC (Calculate with Concepts), waarmee ICT-architecten de samenhang kunnen beschrijven, bestuderen en toetsen op consistentie. Hij illustreert hoe de CC-techniek in de praktijk wordt toegepast en biedt handvaten voor praktisch informatiebeleid dat interne ICT-organisaties slanker, sneller en slagvaardiger kan maken.

In ontwerpdisciplines is kunstenaarschap uitgesloten zonder vakmanschap. De beheersing van methoden, technieken en gereedschappen geeft ruimte voor stijl. Als de architect Berlage zijn vak niet had beheerst, dan had zijn Beurs in Amsterdam ongetwijfeld niet de status van stijlmonument verworven. De CC-techniek, die hier wordt besproken, hoort thuis in de gereedschapskist van ICT-architecten. De essentie van deze techniek is het 'rekenen' met concepten uit de taal waarin de probleemstelling wordt weergeven. De techniek helpt architecten om scherpe definities te maken en daaruit stukken specificatie te genereren. Dit helpt om terminologiediscussies kort en zakelijk te houden en functionele specificaties te concretiseren en te versnellen. Voor ICT-intensieve organisaties betekent dit een stap op weg naar beheersing van ICT en meer consistentie met de business. Daarmee levert CC een bijdrage aan verdere professionalisering van het vak en schept het ruimte voor creativiteit.

In dit artikel wordt uitgegaan van een situatie waarin één of meer architecten in opdracht een gegeven vraagstuk oplossen. Bijvoorbeeld: een bank wil een infrastructuur voor real-time internationale betalingen over openbare e-mailkanalen inrichten ten behoeve van internetcommercianten. Of: een verzekeraar wil zijn bestaande infrastructuur geschikt maken voor het grootschalig en flexibel besturen van bedrijfsprocessen om de kosten te drukken en de time-to-market te vergroten. Uitgangspunt is een typische ICT-architect die in teamverband opereert en die gegeven, niet door hemzelf gekozen businessdoelstellingen ondersteunt. Hij brengt het terrein in kaart, maakt schetsen voor de opdrachtgever, bepaalt de blauwdrukken en bestekken, bewaakt het bouwtraject en begeleidt ten slotte de opdrachtgever in het acceptatietraject. Een ander uitgangspunt is onafhankelijkheid: als het even 'moeilijk' wordt tussen opdrachtgever en aannemer, dan staat de architect zijn opdrachtgever (en niet de aannemer) met raad en daad terzijde.

Aan welke eisen moet een conceptuele techniek voldoen? De ICT-architect in het veld ziet zich geplaatst voor vraagstukken rond infrastructuur, procesbesturing, e-commerce, legacyproblematiek en grootschalige ICT-projecten (Cook 1996 en Inmon 1997). Een gemeenschappelijk kenmerk van deze vraagstukken is de behoefte aan inzicht en overzicht over de veelheid aan details, die in veel situaties ontbreken. Het probleem zit vaak in de omvang en aantallen, en soms zelfs in de complexiteit van de materie. De CC-techniek leidt tot *heldere definities*, automatische *toetsing van consistentie* en *eenduidige specificaties*.

Heldere definities vergen altijd enige discussie. In de praktijk leidt dit soms tot onbestuurbare en tijdrovende definitiestudies ('een Poolse landdag'). De CC-techniek helpt een ICT-architect deze discussie in goede banen te leiden. Hij leidt definities af uit conceptuele modellen en structureert de discussie omdat hij de verbanden kent die hij door formalisatie uit deze modellen heeft afgeleid. Voor conceptuele analyse kennen we van oudsher modelleringstechnieken als ER-modellen, NIAM[1] en objectgeoriënteerde modellen, met bijbehorende notatiewijzen zoals de 'kraaienpootnotatie' of UML. Deze technieken worden voornamelijk gebruikt om gegevensstructuren te ontwerpen. De CC-techniek modelleert wel concepten, maar is verder ongeschikt voor datamodellering. CC representeert de relevante delen van de taal van het probleemgebied, die een architect op grond van documentanalyse en/of een workshop kan achterhalen. Door dit model in verband te brengen met andere relevante talen, kan een architect een consistente verzameling definities afleiden. De 'Poolse landdag' kan dan worden overgeslagen. In de praktijk zijn discussies waar een half jaar voor was gepland, in enkele weken gevoerd.

Het *toetsen van consistentie* ontmaskert conceptuele fouten. Als Jan bijvoorbeeld geautoriseerd wordt om een kredietbeoordeling te doen, maar hij is daartoe niet gekwalificeerd, dan wordt een regel overtreden. De CC-techniek leidt dit soort regels af uit taalmodellen, die zelf ook op interne consistentie worden getoetst. Alles kan met de hand of door een computer worden uitgevoerd.
Conceptuele consistentie heeft consequenties voor de praktijk. Het is immers niet onredelijk dat een organisatie eist dat een ontwerp ongeschonden door de toets komt alvorens hem inhoudelijk te beoordelen en vrij te geven voor realisatie. Op deze manier leidt de CC-techniek tot een algemeen hoger kwaliteitsniveau van ontwerpen. Wanneer deze toets gepaard gaat met registratie in een ConcernArchitectuur Repository (CAR), krijgt de onderneming tevens een instrument waarmee het concerninformatiebeleid op een hoger plan kan worden getild.

Scherpere specificaties levert CC door eisen te vertalen als uitdrukkingen in predikaatlogica. Deze uitdrukkingen worden door de computer gegenereerd. Door de eisen voor verschillende systemen (bijvoorbeeld een klant- of polisregistratiesysteem en schaderegistraties) uit hetzelfde model te genereren, is een verbeterde consistentie tussen systemen eenvoudig te garanderen. De specificatie die zo ontstaat is een accuraat communicatiemiddel voor architect en bouwers.

Voor de architect bestaat de toegevoegde waarde van de techniek uit concreetheid en snelheid in definitie en ontwerp. Een opdrachtgever krijgt niet veel van de techniek te zien; het is immers gereedschap voor de architect. Desondanks zijn de consequenties juist voor opdrachtgevers relevant: architectuur wordt concreter en dus toegankelijk voor beleid. Door ontwerpen formeel te toetsen, heeft elk ontwerp immers een gegarandeerd niveau aan concreetheid en consistentie. Ten slotte kunnen de gegenereerde eisen het bouwproces versnellen doordat programmeurs van start gaan met vooraf bepaalde formele eisen.

De techniek

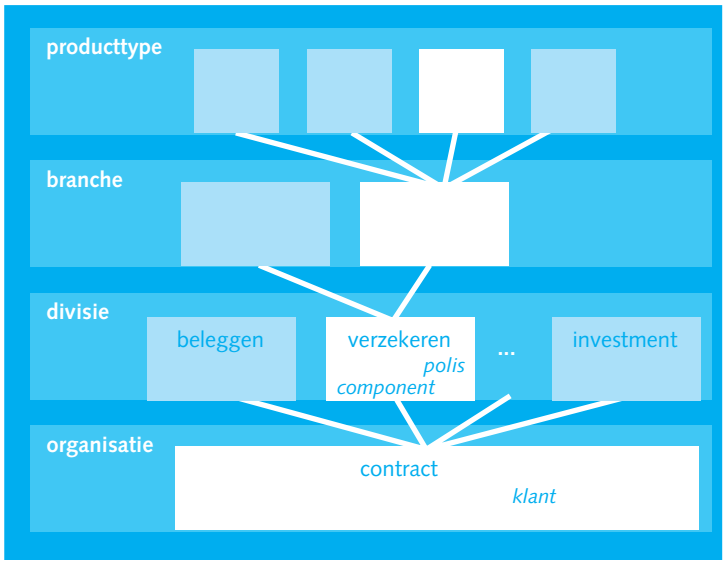
CC formaliseert een beperkt aantal begrippen die relevant en specifiek zijn voor de situatie. Zo kan het begrip 'inslag' specifiek zijn voor autoruitschades, maar de minder specifieke term 'schade' komt terecht in de algemenere verzekeringstaal. Afspraken over begrippen als klant, contract, verplichting en dergelijke kunnen op het niveau van de organisatie worden belegd. Een grote gemeenschappelijke taal bestaat niet in CC. Wel bestaat een gelaagde structuur waarin specifieke talen in de context van algemenere talen zijn geplaatst. De keuze van lagen daarin is voor elke organisatie anders.
Figuur 1 geeft een voorbeeld waarbij de begrippen zijn belegd volgens de criteria 'producttype', 'branche', 'divisie' en 'organisatie'. Bij de ontwikkeling van een nieuw verzekeringsproduct beschrijft de architect slechts de begrippen die nieuw zijn. De computer helpt bij het overzien van de consequenties in de context. Deze gelaagdheid in talen wordt in CC consequent doorgezet. Zo kunnen begrippen omtrent de afhandeling van de particuliere autoruitschade worden gedefinieerd in de context van autoschades, die zich weer afspeelt in de context van de afdeling particulieren van het verzekeringsbedrijf.
In de CC-techniek houdt de architect zich uitsluitend bezig met begrippen uit het probleemveld (zoals 'afhandeling particuliere autoruitschade'). De begrippen, eigenschappen en regels uit de context worden door overerving (automatisch dus) meegenomen. Het principe van specialisatie doet het werk. Elke polis is immers een speciale vorm van een contract, dus moet een polis alle regels ten aanzien van contracten respecteren. Op vergelijkbare wijze kan het begrip polis op divisieniveau verder worden verfijnd tot brandpolis, transportpolis, levenpolis of varia.
Deze aanpak staat toe dat eigenschappen en definities in een verschillende context van elkaar

S.M.M. Joosten is hoogleraar Bedrijfsprocessen en IT aan de Open Universiteit en is directeur van Anaxagoras Procesarchitecten.
E-mail: stef.joosten@ou.nl.

verschillen. Zolang de definities niet strijdig zijn met de onderliggende contextlagen, is alles toegestaan. Het begrip 'begunstigde' in de context van levenpolissen verschilt bijvoorbeeld van de begunstigde in een brandpolis: in een levenpolis moet de begunstigde in de polis genoemd staan, terwijl de begunstigde bij sommige brandpolissen pas bij een claim kan worden bepaald. De definities en eigenschappen mogen zelfs strijdig zijn, zolang de een niet de context vormt waarbinnen de ander is gedefinieerd. CC eist dus wel dat elke taal consistent is met de onderliggende talen. Dat betekent dat een leven-klant en een brand-klant allebei voldoen aan de algemene eisen die voor elke klant in acht worden genomen.

De CC-techniek hanteert een gelaagde aanpak, waarin taalmodellen 'gestapeld' worden. Boven op de 'wereld' komt bijvoorbeeld de 'organisatietaal', daar bovenop de 'verzekeringstaal' en daar bovenop de 'brand-taal' en de 'leven-taal'. Als de CC-techniek correct wordt toegepast, komt elke laag overeen met een autorisatieniveau in de organisatie. Door bijvoorbeeld het brand-taalmodel te autoriseren, zegt een afdeling brandverzekeringen feitelijk dat de begrippen in de context van brand vastliggen. Dat doet zij uiteraard alleen als het model geen inconsistenties bevat, als het goed integreert in de context van onderliggende modellen en als de begrippen ten aanzien van brand voldoende zijn afgedekt. Hierdoor ontstaat een hiërarchische structuur van taalmodellen, waarin elk organisatieonderdeel de vrijheid krijgt om de eigen definities te hanteren zolang ze niet strijdig zijn met die van de omgeving.

Figuur 1



Figuur 2

Het hier gebruikte verzekeringen-voorbeeld[2] bevat vier lagen (figuur 2). De onderste laag bevat definities ten aanzien van objectgeoriënteerdheid. Daar bovenop zijn begrippen als 'business goal' en 'commitment' gedefinieerd die in de omgeving (de wereld) van toepassing zijn. In de derde laag zitten organisatiedefinities als 'contract' en 'klant'. De vierde laag bevat definities die bij het verzekeringsbedrijf in het algemeen horen. De figuur houdt daar op, maar er kan natuurlijk worden doorgestapeld om specifieke concepten van branches (brand, leven enzovoort) toe te voegen, om zo verder te kunnen gaan naar telkens specifiekere definities.

Voorbeeld

Het voorbeeld hieronder is afkomstig uit de praktijk van een verzekeringsbedrijf. Het illustreert dat gegenereerde specificaties relevante eigenschappen weergeven die betekenisvol zijn.[3]

Om de verschillende verzekeringsproducten snel te kunnen aanpassen, heeft een verzekeraar verzekeringscomponenten gedefinieerd van waaruit flexibel een polis (contract) wordt samengesteld. De verzekeraar wil zo een componentenbank inrichten, die consistent is met de bestaande polisregistratiesystemen en schadebehandelingssystemen. Vanuit één architectuurmodel is voor elk systeem een aantal eisen afgeleid (gegenereerd). Voor de verzekeringscomponenten tonen we er vijf:

1. $\forall$ r::Risico; p::Product; g::Gebeurtenis; c::Component:
   r = type(g) $\wedge$ p = type(claim(g)) $\Rightarrow$ c dekt r $\wedge$ c in p
   Deze eis spreekt over een gebeurtenis g (bijvoorbeeld een brand op de Handelskade nr. 4) die erkend is als een risico r (in dit voorbeeld dus een brandrisico). Daarnaast heeft de maatschappij de claim erkend op de polis, die van het type p is (bijvoorbeeld een integrale bedrijfsschadepolis). Dit impliceert dat het genoemde risico gedekt is in één van de componenten van dit product.
2. $\forall$ r::Risico; o::Objectsoort; g::Gebeurtenis; o'::Object; $\exists$c::Component:
   r = type(g) $\wedge$ g betreft o' $\wedge$ o = type(o') $\Rightarrow$ c dekt r $\wedge$ c beschrijft o
   De tweede eis gaat ook over een gebeurtenis g die erkend is als een risico r. Laten we aannemen dat het dezelfde claim betreft van de brand op de Handelskade nr. 4. Deze eis veronderstelt dat het pand Handelskade 4 van type o is (bijvoorbeeld een loods) en vereist dat er een component is die beschrijft dat loodsen gedekt zijn in een component van het brandrisico.
3. $\forall$ c::Component; p::Product:
   deel(c) in p $\wedge$ c in p
   Deze eis beschrijft dat als c een deelcomponent is van een component uit product p, dan is c zelf ook een component van product p. Dit betekent dat componenten kunnen worden opgebouwd uit (kleinere) componenten.
4. $\forall$ p::Product; o::Objectsoort; p'::Polis; o'::Object; (c::Component:
   p = type(p') $\wedge$ p' verzekert o' $\wedge$ o = type(o') $\Rightarrow$ c in p $\wedge$ c beschrijft o
   De vierde eis beschrijft dat als polis p het object o' (het pand Handelskade 4) verzekert, de polis een component moet bevatten die beschrijft dat loodsen gedekt zijn tegen brand.
5. $\forall$ v::Voorwaarde; g::Gebeurtenis:
   in(v) dekt type(g) $\Rightarrow$ g voldoet v
   Als er een voorwaarde is in de component die het risico dekt van de geaccepteerde gebeurtenis g, dan voldoet deze gebeurtenis aan voorwaarde v.

De genoemde eisen geven verbanden weer tussen verschillende systemen. In dit geval gaat het om de relaties tussen de componentenbank, de polisregistraties en de claimregistraties. Dit voorbeeld toont eisen die betrekking hebben op de componenten die gebruikt worden in de functionele specificatie van de componentenbank. Eisen met betrekking tot polissen, verzekerden, verzekerde objecten, en in principe voor elk relevant concept worden vanuit hetzelfde model gegenereerd. Dit waarborgt de onderlinge consistentie van de verschillende eisenpakketten.

Toepassing

De CC-techniek is zoals gezegd conceptueel gereedschap voor architecten. Voor opdrachtgevers is vooral het resultaat van belang: consistente definities, eenvoudige concrete architectuur en een geautomatiseerde toetsbaarheid. De techniek zelf is niet voor de gebruikersorganisatie bedoeld.

Een belangrijke toepassing van CC is het beheersbaar maken van ICT op concernniveau. Op dit punt heeft de CC-techniek nog niet de kans gehad zich in de praktijk te bewijzen, maar ze is er wel voor ontworpen. Neem bijvoorbeeld een situatie waarin elke infrastructuur, elk nieuw systeem, elk nieuw product en elke nieuwe dienst een formeel fiat moet krijgen voordat de vernieuwing in productie mag. Er is niet één autoriteit die alles fiatteert: een internationaal

betaalnetwerk vergt een ander fiat dan het instellen van een maandelijkse ondernemersadviesmiddag voor Twentse middenstanders. Laten we ook aannemen dat een organisatie kennis en documentatie wil structureren rond alles wat ontworpen wordt. De CC-techniek biedt een structuur voor deze kennisopslag, die elke autoriteit in het bezit laat van haar eigen kennis. De ConcernArchitectuur Repository, die de autorisatiestructuur van de organisatie volgt, is zonder reorganisatie in te passen in het lopende beleid. Elke autoriteit die van een nieuw ontwerp een CAR-toets eist, waarborgt dat deze nieuwe kennis is opgeslagen en kan garanderen dat er geen inconsistenties in het ontwerp zitten die door CAR ontdekt konden zijn.[4]

CC is in de praktijk al op verschillende manieren ingezet. Een aantal voorbeelden illustreert het gebruik:

- **'ontmoeilijken' van discussies rond workflowmanagement (bij grote bank, provinciale overheid). Resultaat: vastgelopen projecten vlotgetrokken.**
- **advies over het omzetten van businessmodellen naar workflowmodellen (bij pensioenfonds). Resultaat: foutieve generatie van workflowmodellen aangetoond.**
- **vereenvoudiging procesmodellering (bij grote bank). Resultaat: simpelere modellen.**
- **definitie van de taal WorkPAD (voor intern onderzoek). Resultaat: scherpe afbakeningsregels in het eigen vakgebied.**

Achtergronden

De CC-techniek benut ideeën uit de kennistheorie, informatiesysteemmodellering en formele technieken. In de toepassing doet CC denken aan het ideaal van het corporate datamodel uit de jaren tachtig. De CC-techniek verklaart waarom dat model niet kon werken: begrippen zijn lokaal gebonden aan een plaats, een groep mensen of aan een probleemveld. Als iedereen meepraat over een basaal begrip als 'contract', 'activiteit' of 'inslag', dan breekt een langdurige en weinig zinvolle discussie los. Elk begrip hoort thuis in een beperkte context, en binnen die context voer je een beheerste discussie over een beperkt aantal begrippen. De CC-techniek staat zoals gezegd toe dat definities in verschillende afdelingen, verschillende werkgroepen en verschillende systemen onderling kunnen verschillen, en zelfs strijdig kunnen zijn.

Parnas (1994) onderscheidt drie redenen waarom software veroudert: veranderende behoeftes, uitputting van resources en afwijking van oorspronkelijke concepten en specificaties door voortdurende aanpassingen. Parnas geeft aan dat de laatste het grootste probleem vormt. Juist hier helpt de CC-techniek door het blootleggen van conceptuele afwijkingen.

De CC-techniek lijkt ook op werk uit de kennistheorie (Gruber, 1993 en Siau, 1996) waar kennis in kaart wordt gebracht in structuren die men 'ontologie' noemt. Een kennistechnoloog zou de CC-techniek beschrijven als een boomstructuur waarin ontologieën zitten. Het toetsen van consistentie en het genereren van specificaties zijn functies die in kennismanagement van nut zijn. In die zin is de CC-techniek een instrument voor kennismanagement.

Filosofisch sluit CC aan op de kennisleer. Het idee dat een concept wordt gedefinieerd door zijn eigenschappen (het kwaakt als een eend, het zwemt als een eend en het waggelt als een eend, dus is het een eend) is terug te voeren op Aristoteles. Later is dit idee ook toegepast in NIAM, waar datamodellen worden ontwikkeld vanuit zinnen in de natuurlijke taal. De huidige manier van objectgeoriënteerd modelleren, bijvoorbeeld in een taal als UML, voegt daar het idee van classificatie van begrippen en overerving aan toe. Deze bestaande modelleringstechnieken zijn uiteindelijk gericht op het bouwen van gegevensstructuren. De CC-techniek verwerpt deze gedachte, omdat de focus op het bouwen leidt tot een instrumentele denkwijze. CC-modellen zijn juist gericht op een samenhangend geheel, dat concreet wordt door te toetsen op consistentie en het genereren van specificatieonderdelen.

Conclusie

Een slankere, lenigere en behendigere ICT-organisatie vraagt om beheersbare samenhang. Verkoop via internet, betalen met de GSM, WAP-diensten; elke nieuwe mogelijkheid vraagt om snellere actie en foutloos werkende nieuwe technologie. Grote organisaties die hun producten, hun klanten, hun processen en applicaties los van elkaar blijven beheren, staan machteloos tegenover de snelheid en kwaliteit van nieuwe (en bestaande) concurrenten (Hammer, 1993). Zicht op de samenhang biedt zekerheden die bijvoorbeeld nodig zijn om verouderde systemen te durven ontmantelen. Zonder dit inzicht zullen nieuwe systemen veelal naast de oude systemen blijven functioneren, waardoor het aantal systemen alleen maar verder groeit en de beheersbaarheid daalt.

De CC-techniek biedt een goed vooruitzicht op beheersbaarheid doordat zij structureert op manieren die haalbaar zijn in de alledaagse praktijk. Niemand hoeft immers de hele organisatie in kaart te brengen en elke stap heeft directe toegevoegde waarde. Ook hoeft een organisatie haar bestaande autorisatiestructuren niet af te breken om de ICT beheersbaar te maken. Door documentatie te ontsluiten via CAR ontstaat bovendien een goede, gestructureerde architectuur-repository. Zo biedt de CC-techniek een geloofwaardige stap in het verder beheersen van grootschalige ICT-infrastructuren.

Referenties

Cook, Melissa: *Building Enterprise Information Architectures*, Prentice Hall, 1996.

Gruber, Thomas: *A translation approach to portable ontology specifications. Knowledge Acquisition, 5(2)*, p. 199 – 220, 1993. ftp://ftp-ksl.stanford.edu/pub/KSL_Reports/KSL-92-71.ps

Hammer, Michael en James Champy: *Reengineering the Corporation*, Harper 1993.

Inmon, W.H., John Zachman, en Jonathan Geiger: *Data Stores, Data Warehousing, and the Zachman Framework*, McGraw-Hill 1997.

Siau, Keng, Yair Wand en Izak Benbasat: 'When Parents Need Not Have Children – Cognitive Biases in Information Modeling', Lecture Notes in: *Computer Science – Advanced Information Systems Engineering*, Vol. 1080, P. Constantopoulos, J. Mylopoulos, and Y. Vassiliou (eds.), 1996, Springer-Verlag, pp. 402-420.

Noten

1. Later: 'Nijssens universele informatiekunde'.
2. Een uitgebreide beschrijving is te vinden op www.anaxagoras.com
3. Dit is slechts een klein gedeelte. Het complete voorbeeld is honderd procent gegenereerd vanuit taalmodellen en is te zien op de weblocatie die genoemd is in noot 2.
4. Om de (bewezen) CC-techniek toe te passen voor een (nog onbewezen) CAR-concept is de auteur op zoek naar grote financiële instellingen die hun architectuurprocessen beheersbaar willen maken door het structureren van architectuurkennis.
5. Deze voorbeelden gaan alle over processen, omdat Anaxagoras zich specifiek op procesarchitectuur toelegt. De techniek is ook daarbuiten toepasbaar, omdat samenhang immers altijd vereist dat taalkwesties zijn opgelost.