Jos Warmer is geestelijk vader van de *Object constraint language* (OCL) die al in versie 1.1 van de UML-standaard is opgenomen. Bij realtime toepassingen bestaat historisch gezien een sterke behoefte aan precies en eenduidig specificeren en OCL is verreweg het meest formele deel van UML. De echte kracht van OCL zit 'm echter in de combinatie met UML-diagrammen.

HANS VAN THIEL

Interview met Jos Warmer, geestelijk vader van de *Object constraint language*

'Diepgang van OCL is

PT Embedded Systems: Al vanaf 1997 ben je betrokken bij de standaardisatie van UML door de OMG, eerst als lid van de IBM/ObjecTime groep, nu met het bedrijf Klasse Objecten. Je bent ook lid van de UML Revision Task Force die de komende UML 2.0 vernieuwing coordineert. Wat gaat er veranderen?

Jos Warmer: 'Op dit moment is UML 1.4 de officiële standaard en er komt eerst nog een versie 1.5. Daarin zal weinig veranderen ten opzichte van nu, maar versie 2.0 die voor 2002 staat gepland zal wel compleet vernieuwd zijn. Er komt bijvoorbeeld een functionele uitbreiding met de zogenoemde *diagram interchange*-faciliteit. Met een visuele taal als UML moet je toch ook plaatjes kunnen uitwisselen tussen verschillende tools en dat kan nu nog niet. De gereviseerde standaard zal gebaseerd zijn op drie zogeheten RfP's (request for proposal) die samen de grondslag voor een nieuw UML gaan vormen. De eerste RfP behandelt de infrastructuur, zeg maar de kern van UML. Voor de gebruiker verandert er weinig maar de specificatie wordt in allerlei opzicht beter omschreven. Dat is vooral van belang voor de ontwikkeling van UML-tools. De superstructuur RfP behandelt uitbreidingen van UML voor speciale doeleinden en in dat deel komen diverse nieuwe elementen om software te modelleren. De derde RfP moet de OCL (object constraint language) verder integreren in de specificatie van UML. OCL krijgt ook een logisch-mathematisch gefundeerde semantiek en er komen mogelijkheden om dynamische aspecten van objecten te modeleren.'

> 'Een nadeel van formele talen is dat het gebruik een logisch-mathematische achtergrond vereist'

PT Embedded Systems: Je bent de geestelijk vader van de *Object constraint language* (OCL) die al in de UML 1.1 standaard is opgenomen. Toch krijgt OCL in de literatuur over UML in het algemeen weinig aandacht. Kun je iets vertellen over de begintijd? Is het toeval dat OCL door de werkgroep met ObjecTime is geïntroduceerd of is het vooral interessant voor embedded systemen?

Jos Warmer: 'OCL is gebaseerd op Syntropy van Steve Cook en John Daniels, maar er hebben een heleboel mensen

te bevatten'

Foto: Jan Bogaerts

aan gewerkt en er zijn meer invloeden geweest. Steve Cook en ik werkten allebei bij IBM en vanuit die positie is OCL ingebracht in de standaardisatie van UML door de OMG in 1997.
OCL is oorspronkelijk bedoeld om *business rules* precies te kunnen formuleren in samenhang met UML-diagrammen. Het begrip 'bedrijfsregels' moet je natuurlijk ruim opvatten; dat kunnen heel goed randvoorwaarden zijn van een embedded realtime systeem. Bij realtime toepassingen bestaat historisch gezien een veel sterkere behoefte aan precies en eenduidig specificeren en OCL is verreweg het meest formele deel van UML. Je kunt bijvoorbeeld invarianten, postcondities en precondities beschrijven op modelniveau en dat zijn constructies die je op implementatieniveau ook vindt in een programmeertaal als Eiffel. Wat in OCL *constraints* zijn worden daar *asserties* genoemd. OCL past heel goed in het *design by contract*-idee dat Bertrand Meyer met Eiffel wilde ondersteunen. Maar de ontwikkeling van OCL is ook beïnvloed door de formele specificatietaal Z.'

PT Embedded Systems: De belangstelling voor OCL is de laatste twee jaar gegroeid.

Jos Warmer: 'Het is een nuttige aanvulling op de UML-diagrammen. Je kunt de analogie nemen van een figuur en een onderschrift daarbij. Een onderschrift staat niet los van de figuur maar verduidelijkt de bedoeling. Met OCL kun je precisie toevoegen en je ziet de notatie dan ook op diverse plaatsen in de specificatie van UML zelf. Met name Grady Booch was een voorstander van formele beschrijvingen in de standaard. Als je met UML software gaat modelleren dan kun je, >

OCL, UML & Jos Warmer

De *Unified modeling language* is zoals wellicht bekend een grafische notatie voor het modelleren van (objectgeoriënteerde) software. De UML-diagrammen en hun elementen zijn met hun betekenissen gestandaardiseerd door de Object Management Group. Deze OMG is met 750 leden de grootste particuliere standaardisatie-organisatie in de wereld en beheert behalve UML onder meer ook Corba (common object request brokerage architecture).
Al vanaf het begin in 1997 is het mogelijk in UML-modellen randvoorwaarden of beperkingen (Eng. constraints) te formuleren voor attributen en waarden van operaties. Hiervoor kent UML een aparte niet grafische notatie: de *Object constraint language* (OCL). Ook condities voor overgangen in object-toestandsdiagrammen kunnen exact worden aangegeven met dit formalisme.
Al dan niet toevallig is OCL indertijd als standaardisatievoorstel ingebracht door onder meer IBM en ObjecTime, beiden bedrijven met interesse in de markt van embedded systemen. De taal is oorspronkelijk bij IBM ontwikkeld door Jos Warmer op basis van de Syntropy-methode van Steve Cook en John Daniels. Eigenlijk was het uitgangspunt externe randvoorwaarden, zoals bedrijfsregels, te kunnen preciseren. Maar OCL is van begin af aan ook gebruikt in de definitie van UML zelf. Juist een industriestandaard, waar toolbouwers zich naar richten, moet immers helder en ondubbelzinnig zijn. OCL is een formele taal die gebaseerd is op eerste orde predikatenlogica. Er kunnen precondities, postcondities, invarianten en guards (randvoorwaarden voor toestandsovergangen) mee worden geformuleerd. Begrenzingen van attribuutwaarden in klassediagrammen zijn in OCL nauwkeurig aan te geven. De taal bevat notaties voor verzamelingen (sets), geordende verzamelingen (sequences) en verzamelingen met gelijke elementen (bags) en >>>>

met het stuk tekst dat OCL is, uitspraken doen over attributen, associaties en condities van operaties binnen je eigen model. Het gebruik van een industriespecificatie in de praktijk staat of valt met de beschikbaarheid van tools en de ondersteuning voor OCL is op dit moment beperkt. Er zijn enkele parsers beschikbaar en OCL wordt ondersteund door ArgoUML, het open source UML-project.
Als in OCL 2.0 ook de semantiek een formele interpretatie krijgt dan wordt het mogelijk UML-modellen met behulp van OCL automatisch te valideren en te verifiëren. Het gaat hierbij niet om implementatie met bijbehorende codegeneratie maar een soort simulatie van je model met automatische controle van alle constraints. Op dit moment valt dat allemaal nog onder de noemer van research.
OCL is in principe geschikt als querytaal en dan moet je denken aan het modelleren van database-applicaties. Het Zweedse bedrijf Boldsoft werkt aan toepassingen op dat gebied. Het OCL 2.0 voorstel dat wij bij de OMG hebben ingediend is zeer positief ontvangen en wordt onder meer onderschreven door Rational, IBM, Telelogic en Iona. Ik verwacht dat de ondersteuning van OCL door tools zeker zal verbeteren in de komende jaren.'

PT Embedded Systems: Is het waar dat TogetherSoft een eigen voorstel heeft ingediend?
Jos Warmer: 'Dat is waar, al is het nog niet zeker in hoeverre dat voorstel gehandhaafd blijft. Ieder lid van de OMG kan standaardisatie voorstellen doen en dat is ook de praktijk. De OMG maakt zelf geen standaarden, zij coördineert alleen de totstandkoming daarvan en beheert ze. Klasse zit met een aantal andere bedrijven en instellingen in een zogenoemd *submission team* en wij hebben dit jaar een voorstel voor OCL 2.0 gemaakt. Het standaardisatieproces houdt in dat alle voorstellen in de OMG worden besproken en dat daar uiteindelijk een consensus uit voortkomt. Dat overleg wil natuurlijk wel eens moeizaam verlopen maar iedere deelnemer beseft wel dat je voor een industriestandaard een brede overeenstemming nodig hebt. Het werkt echt niet als je met 51% van de stemmen je wil er doordrukt, dan blijft zo'n standaard een stuk papier. Je hebt wel zo'n 90% consensus nodig voor een brede implementatie in tools.'

PT Embedded Systems: Je noemde eerder de invloed van de specificatietaal Z. Waarom is er met OCL voor een nieuw formalisme gekozen?
Jos Warmer: 'Een nadeel van formele talen is dat het gebruik een logisch-mathematische achtergrond vereist. De diepgang van OCL is te bevatten – ga je nog dieper dan krijg je al gauw een schrikreactie. De notatie van OCL is bedoeld om in de praktijk te worden gebruikt. Verder kun je OCL geleidelijk invoeren. Je kunt je constraints ook in natuurlijke taal als commentaar bijschrijven en de precisie van OCL gebruiken al naar behoefte en vermogen. Als je Z wilt gebruiken is dat een 'big bang'-overstap – of alles of niets.
De echte kracht van OCL zit echter in de combinatie met UML-diagrammen. OCL is bijvoorbeeld niet uitbreidbaar en het heeft geen uitdrukkingen voor tijd. Maar als je een UML-package maakt met notaties voor tijd kun je al die klassen gebruiken in OCL. Je kunt dan in OCL alles uitdrukken over tijd. De taal is dus zelf niet uitbreidbaar, maar wel via de omweg van UML.'

> 'De diepgang van OCL is te bevatten – ga je nog dieper dan krijg je al gauw een schrikreactie'

PT Embedded Systems: Je bent, samen met Anneke Kleppe, auteur van *The Object Constraint Language: Precise Modeling with UML* uit 1999. Komt daar een nieuwe uitgave van zodra de OCL 2.0 standaard officieel is geworden? Zijn er nog andere activiteiten gepland?
Jos Warmer: 'Ja, we hebben met de uitgever afgesproken dat de invoering van UML 2.0 daar een goed moment voor is. Wat de verdere plannen betreft: wij willen bij Klasse Objecten zelf een tool gaan bouwen dat de mogelijkheden realiseert die wij in OCL zien. Dat wordt dan geen volledige UML-suite maar een module die met andere commerciële tools kan samenwerken. Ook voor dit project is de komende OCL-revisie een goede aanleiding.' ■

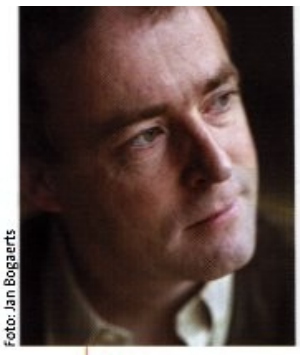
Foto: Jan Bogaerts

>>>> allerlei daarmee samenhangende operaties. Maar ook Integer, Real, String en Boolean zijn basistypen. OCL heeft de *look and feel* van een OO-programmeertaal.
Een van de doelstellingen, aldus Jos Warmer, was dat de formele notatie begrijpelijk moest zijn voor gebruikers zonder specifiek logisch-mathematische training. Het kenmerkende van OCL is echter dat de ontwerper het kan gebruiken om voorwaarden te formuleren voor de klassen, objecten en associaties van zijn of haar eigen model. Sterker nog, elke OCL-constraint moet gebonden zijn aan een bepaald object, dat de constraintcontext wordt genoemd. Vanuit die context kunnen de associaties van dat object gevolgd worden om allerlei samenhangende randvoorwaarden te formuleren. Bij het modeleren wordt OCL dus altijd gebruikt in combinatie met UML-diagrammen en nooit op zichzelf.
Jos Warmer is inmiddels consulent bij *Klasse Objecten* en publiceerde samen met Anneke Kleppe over UML en OCL. Hij is lid van de UML Revision Task Force en voorzitter van de OCL Working Group van de OMG. Als zodanig is hij nauw betrokken bij UML 2.0 met inbegrip van OCL, dat in de loop van 2002 als standaard moet verschijnen. Jos Warmer is ook actief in een zogenoemde *submission team* dat de eigenlijke voorstellen aan de OMG voorlegt.

Literatuur

- Jos Warmer en Anneke Kleppe, *Praktisch UML*, 2e editie, Addison-Wesley, (2001, Nederlandstalig).
- Jos Warmer and Anneke Kleppe, *The Object Constraint Language: Precise Modeling with UML*, Addison-Wesley, (1999).

- www.klasse.nl
- http://neptune.init.fr/Biblio/ocl-publications.shtml
- www.omg.org
- www-3.ibm.com/software/ad/library/standards/ocl.html