Interview met Bran Selic

‘Je bent niet zomaar een effectieve

UML-er'

Bran Selic is als senior methodologist jaren nauw betrokken bij de ontwikkelingen rond realtime UML. Bij ObjecTime was hij de grondlegger van de *real-time object oriented modeling language* (Room), een methode die na de fusie met het bedrijf Rational is geïntegreerd in de *unified modeling language* (Uml). Selic is ervan overtuigd dat Case de verwachtingen nu eindelijk waar begint te maken.

HANS VAN THIEL

PT Embedded Systems: Momenteel houdt U zich vooral bezig met de nieuwe Real Time Unified Modeling Language (RT UML) standaard die onlangs als voorstel van de OMG (Object Management Group) is gepubliceerd. Daar wil ik straks graag op terugkomen, maar uw hele carrière staat in het teken van het ontwerp van realtime software. Bij ObjecTime was u de grondlegger van ROOM (real-time object oriented modeling language) en die methode is na de fusie met Rational geïntegreerd in UML. Hoewel UML 1.3 een open standaard is van de OMG waar door verschillende bedrijven aan is bijgedragen, werken enkele oorspronkelijke architecten, de drie amigo's: Rumbaugh, Booch en Jacobsen, bij Rational. Hoe bent u de vierde amigo geworden?

Bran Selic: (*glimlachend*) 'Nou, zo zou ik het niet willen stellen. Maar de samenwerking met Rational was een logische stap om de technische markten beter te kunnen bedienen. Vooral vanuit de telecom en de vliegtuigindustrie komt steeds meer vraag naar realtime modelering. Jammer genoeg is er sinds de jaren zeventig, toen ik mijn loopbaan begon, niet veel veranderd in de praktijk van het realtime ontwerpen. Het gebeurt nog vaak op dezelfde manier als toen, hoewel de systemen intussen veel ingewikkelder zijn geworden.'

PT Embedded Systems: In een artikel uit 1999 geeft u een inleidend overzicht van realtime modeleren. Een realtime klok kan als een UML-class worden beschreven maar reactief gedrag (event driven) past beter in een toestandsdiagram (state chart). Wat overheerst in de praktijk, reactief of periodiek gedrag?

Bran Selic: 'Meestal is het een combinatie van beiden, maar de echte wereld is *event driven*. Realtime systemen zijn dus reactief, tenminste, als ze eenmaal draaien. Het is echter dikwijls moeilijk om ze in de operationele toestand te krijgen. Dat aspect wordt nogal eens vergeten, maar tachtig procent van alle problemen met realtime systemen ontstaat in de opstartfase.'

PT Embedded Systems: Welke functie kan een RTOS vervullen?

Bran Selic: 'Een besturingssysteem levert mechanismen, maar applicaties bepalen het beleid. Het is te vergelijken met autorijden. De bestuurder kan het rijden niet delegeren aan de auto.'

PT Embedded Systems: Het ontwerp en de realisatie van gedistribueerde software valt nauwelijks te vergelijken met dat van gecentraliseerde systemen, schrijft u in een ander artikel.

Bran Selic: 'Tijd op zich is niet de kritieke factor, het gaat om de interacties. Die zorgen ervoor dat systemen onvoorspelbaar worden. De communicatie tussen onafhankelijke processor-nodes en multitasking horen daar logisch gezien ook bij, maakt het geheel totaal verschillend. Jammer genoeg kennen veel ontwerpers de theorie van gedistribueerde systemen niet voldoende en zoeken ze naar algoritmen voor problemen die theoretisch onoplosbaar zijn. Bij jullie in Europa is het beter gesteld maar in ons onderwijs ligt de nadruk vooral op programmeren. Bekwaamheid in programmeren is natuurlijk belangrijk maar theoretische informatica zou daarmee geïntegreerd moeten zijn. Eigenlijk gaat het erom *concurrency* te modelleren. Daarvoor zijn betere modellen nodig en moet er meer gebruik worden gemaakt van computersimulatie.'

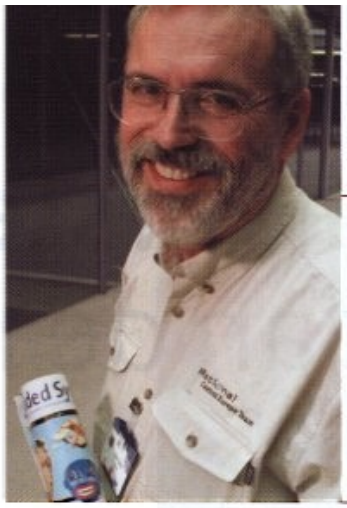
PT Embedded Systems: U hebt een *design pattern* ontwikkeld voor embedded software, het recursive control pattern. Dit is gebaseerd op een scheiding van control en functie en een scheiding van control policy en control mechanisme.

Bran Selic: 'Patterns zijn bewezen algemene oplossingen voor algemene problemen die gebruikt kunnen worden als specifieke oplossing voor specifieke problemen. In die zin zijn er altijd al patterns geweest maar het voordeel is dat er een heleboel nu een eigen naam hebben gekregen. Je kunt er dus naar verwijzen en er over praten, maar (lachend) patterns zijn er al een half miljoen jaar.'

PT Embedded Systems: U omschrijft modelleren als een vorm van abstractie die zowel het begrip van een probleem als de oplossing ervan vergemakkelijkt. Wat is dan een modelleringstool?

Bran Selic: 'In het begin waren dat weinig meer dan tekengereedschappen om diagrammen mee te kunnen maken. Modeleringstools werden vooral gebruikt voor documentatiedoeleinden. Tien jaar geleden verkeerde Case (computer aided software engineering) in een soortgelijke positie als AI (artificial intelligence). Door alle grandioze claims die nooit werden waargemaakt geloofde niemand er meer in. Dat is inmiddels wel veranderd. Je hebt nu echt executeerbare modellen. Je hebt nu niet alleen maar graphics maar een semantiek die ook door de tools wordt ondersteund. Case begint nu echt de verwachtingen waar te maken.
UML is een semi-formele taal die te formaliseren is en modellering en implementatie zal in de toekomst nog verder integreren. Het verschil tussen beiden is ook niet essentieel, het is een kwestie van niveaus van de talen. Een programmeertaal als C of C++ is heel geschikt om details in uit te drukken maar minder geschikt voor hogere abstractieniveaus zoals een state machine.'

PT Embedded Systems: De semantiek van UML is moeilijk. ›



> Jammer genoeg is er sinds de jaren zeventig, toen ik mijn loopbaan begon, niet veel veranderd in de praktijk van het realtime ontwerpen.

Bran Selic: 'Voordat UML er was had je helemaal geen diepere semantiek. Zeker, het is moeilijk om echte implementaties te leren begrijpen, omdat die behoorlijk complex kunnen zijn. Je bent niet zomaar een effectieve UML-gebruiker. Maar met een programmeertaal gaat dat net zo. C++, bijvoorbeeld, is een heel moeilijke taal. Objectoriëntatie is op zich ook heel moeilijk om te begrijpen. Maar wat is het nut van een taal zonder semantiek? Dan zijn modellen niet executeerbaar en moet je alles in je hoofd uitvoeren.
Eén van de doelstellingen van de toekomstige UML 2.0 standaardisatie is om een formele definitie van UML te geven.'

PT Embedded Systems: ROOM is geïntegreerd met UML in wat uw bedrijf RT UML noemt (niet te verwarren met de OMG-standaard, zie verder, red.). Verschillende concepten uit ROOM zijn direct afgebeeld op nieuwe concepten in UML. Hoe groot zijn de verschillen?

Bran Selic: 'In zekere zin is er geen sprake van verandering, er zijn geen conflicten. UML is een uitstekend voertuig om ideeën tot uitdrukking te brengen. We hebben ROOM geëvalueerd en met name Rumbaugh en ik zijn tot een uitwisseling van architectuur gekomen. Het concept van een class in UML is ruim genoeg om bijvoorbeeld *actors* en *capsules* te kunnen beschrijven.'

PT Embedded Systems: 'De officiële UML 1.3 standaard is er nog niet zo lang maar verschillende grondleggers van UML spreken op congressen nu al over allerlei uitbreidingen en veranderingen. Vindt u niet dat de taal een paar jaar de tijd zou moeten krijgen om te stabiliseren?

Bran Selic: 'Versie 2.0 van de UML-standaard is al in voorbereiding en de planning is om in februari 2002 een voorstel bij de OMG af te hebben. Het is de bedoeling dan een aantal wijdverbreide en algemeen gewaardeerde ideeën in de hoofdstroom te brengen in plaats van in speciale bibliotheken. UML is feitelijk een familie van talen waaruit je keuzen kunt doen. Het zijn niet zozeer de UML-architecten, hoewel die natuurlijk het beste willen, maar de gebruikers die veranderingen pushen. Grote bedrijven, met name telecom-bedrijven, oefenen een enorme druk uit op UML. De telecom-bedrijven willen de beste concepten en de beste ideeën in UML hebben. Wat u misschien bedoelt is 'language bloat', het steeds toevoegen van nieuwe dingen aan een taal om er werkelijk alles mee te kunnen doen. Dat moeten we inderdaad vermijden, het gaat om een gemeenschappelijke abstracte basis. Maar een van de dingen die we in versie 2.0 zullen verbeteren is de ondersteuning voor *activity diagrams*. De mensen zijn niet gelukkig met de semantische grondslagen daarvan.'

PT Embedded Systems: U hebt de laatste tijd vooral gewerkt aan de OMG concept-standaard voor realtime UML. Zes bedrijven die op het gebied van realtime modellering actief zijn hebben aan dit document van 125 bladzijden gewerkt. Hoe wordt 'tijd' behandeld in RT UML?

Bran Selic: 'Het voorstel is met opzet zeer flexibel gemaakt. Een standaard moet geen dictaat zijn maar een hulpmiddel. We hebben dan ook geen bepaald tijdsmodel opgedrongen maar alleen principiële regels gegeven over wat tijd is. We hebben bijvoorbeeld open gelaten of tijd discreet of continu is. Tools kunnen dus een eigen model ondersteunen. Elke RTOS-producent heeft bijvoorbeeld een eigen interpretatie van wat een semafoor is en met de standaard blijft dat gewoon mogelijk. Bestaande en toekomstige technieken worden zoveel mogelijk ondersteund. Met de nieuwe RT-standaard kunnen nu in *sequence diagrams* kwantitatief tijdsblokken worden aangegeven. Dit kan zowel absoluut als relatief. *Periodic events* worden ook volledig ondersteund door RT UML.
We willen in juni 2001 met de uiteindelijke versie komen en dan moet eind 2001 de officiële OMG standaard afgerond zijn.'

PT Embedded Systems: Kan het huidige voorstel dus nog worden aangepast?

Bran Selic: 'Jazeker, alle suggesties worden zeer serieus genomen. Ontwerpers zijn soms een beetje geïntimideerd door de OMG, in die zin dat ze denken dat ze toch geen invloed hebben. Maar zo werkt de OMG niet. Het standaardisatieproces staat open voor iedereen. Als er onder uw lezers mensen zijn met commentaar of aanbevelingen moeten zij me een email sturen. Misschien dat het even kan duren, maar zij ontvangen zeker een antwoord en hun bijdrage zal alle aandacht krijgen.'

PT Embedded Systems: Wat is het uiteindelijke doel van de inspanningen op het gebied van modelering en modeleringstools?

Bran Selic: 'Wij werken in elk geval aan tools die de RT UML standaard ondersteunen. Over de andere deelnemers kan ik niets zeggen.
Het doel op langere termijn is om voorspellende modellen te kunnen maken. Daarmee kan je dan, zeg maar, drie architecturen ontwerpen die je dan met UML-tools kunt toetsen.
Traditioneel is er in het realtime ontwerpen veel te veel overgelaten aan hoop. Van alle IT-projecten mislukt zestig procent en bij RT-projecten is het waarschijnlijk nog erger. Omgaan met tijdvoorwaarden is niet gemakkelijk en het begrip tijd is niet gemakkelijk te hanteren. Het doel van modellering en de tools is een bepaalde kwaliteit van ontwerpen te kunnen bereiken en om ontwerpen kwantitatief te maken.
Hopelijk zullen we ooit software kunnen ontwerpen zoals we bruggen kunnen bouwen, vanuit een model dat ons laat weten hoe de software zich zal gedragen. Niet eerst het systeem bouwen en dan pas daar achter komen.' ◘

Bran Selic is te bereiken op: bselic@rational.com

Referenties

- Selic, B. 'Turning Clockwise: Using UML in the Real-Time Domain', Communications of the ACM, 42, 10, p. 46-53 (1999)
- Selic, B. 'Distributed Software Design: Challenges and Solutions'. Rational Software, Kanata, Ontario, Canada (2000)
- Selic, B. 'High Level Design Patterns for Complex Embedded Software'. Rational Software, Kanata, Ontario, Canada (2000)
- 'Response to the OMG RFP for Schedulability, Performance, and Time', Document Version 1.0, OMG document ad/2000-08-04, Joint submission by: ARTiSAN Software Tools, I-Logix, Rational Software Corp., Telelogic AB, TimeSys Corporation, Tri-Pacific Software Inc. (Augustus 2000)