Bericht uit 

U heeft mij er al vaker de nadruk op horen leggen: Java is een object georiënteerde programmeertaal. Dat is heel fijn voor de OO'er. Je kunt er zonder omwegen al die fijne OO-concepten in kwijt en je kunt het OO/UML model redelijk naadloos transporteren naar Java code. Nu is er iets merkwaardigs aan dat OO. Al dertig jaar wordt door een aantal software goeroes gepredikt dat OO de enige juiste manier van ontwikkelen is. Toch wil het maar niet lukken.

Erg Jammer Beans

Het grootste deel van de systemen die nu ontwikkeld worden is van een "traditioneel" model. Zelfs al die systemen die met moderne OO ontwikkeltools en -talen gemaakt worden zijn in feite gewoon traditionele systemen.

Een opmerking die wij recent opvingen lijkt daar mee te maken te hebben: "Het is toch gek met dat OO, je snapt het - of je snapt het niet, en dan zul je het ook nooit snappen". Wanneer ik in een treurige bui ben na weer een eindeloze wel-niet-OO discussie, wil ik inderdaad nog wel eens geloven dat negentig procent van de IT'ers OO niet snapt en ook nooit zal snappen. Het ligt niet aan de tools en talen, het ligt aan de mensen.

De afgelopen drie jaar is OO in de verdrukking gekomen door een nieuwe stroming: component based development (CBD). OO zou gefaald hebben, zou in de praktijk niet leiden tot meer hergebruik en betere systemen. Met CBD zouden deze doelstellingen wel bereikt worden.

En dus kwam Sun alweer een tijdje terug met een Java component model: JavaBeans en later ook Enterprise JavaBeans (EJB). De doelstelling van de EJB architectuur is dat de componentontwikkelaar zich richt op het implementeren van de "pure" business logica (het leuke werk) en dat anderen zich richten op het bij elkaar brengen van componenten en het inbedden van de componenten in de systeemomgeving (koppelingen met databases, beveiligingssystemen). Deze EJB componentarchitectuur werd gepromoot onder het mom van schaalvergroting: Door EJB's te gebruiken kunnen Java systemen groter, beter, meer gedistribueerd en beter beheersbaar worden! Bovendien zou hergebruik eenvoudiger zijn en meer voor de hand liggen met componenten dan met objecten.

Schaalbaarheid blijkt inderdaad een probleem te zijn geweest voor het oorspronkelijke Java. Niet de schaalbaarheid van de Java Virtual Machine, het Java platform, maar van het aantal ontwikkelaars dat de OO concepten achter Java snapt. Schaalbaarheid is niet vergroot door het platform groter te maken maar vooral door een architectuur neer te leggen die uitnodigt tot traditionele systeemmodellen. Door een opsplitsing te maken tussen procescomponenten (session beans) en datacomponenten (entity beans) past Java ineens in het wereldbeeld van de traditionele ontwikkelaar: Ik snap Java! Java is simpel! De Java goeroe van twee jaar terug, een hartstochtelijke OO-techneut die in drie jaar tijd zelf alle uithoeken van Java ontdekt had, wordt nu met groot gemak opzij geduwd door een nieuw soort Java goeroe: De traditionele ontwikkelaar met al 20 jaar ervaring in de IT. EJB: Cobol in een nieuw jasje! De komende jaren gaan weer een heleboel legacy opleveren!

Ik ben altijd van mening geweest dat OO en CBD hand in hand gaan en dat CBD zonder kennis van OO geen haalbare zaak is. In het beste geval zijn componenten gewoon groot uitgevallen objecten of groepjes bij elkaar gepakte objecten die inderdaad logisch gezien bij elkaar horen. Die mening moet ik herzien. CBD (in ieder geval EJB) zit OO flink in de weg; een veldslag in de OO-oorlog. EJB is gevallen voor de traditionelen. Rug recht! We houden vol! Gewoon nog dertig jaar!

Theo Koster
is werkzaam bij Conclusion communication
(e-mail: tkoster@conclusion.nl)