Bericht uit Java

Hoera, er komt weer een nieuwe Java versie aan! J2SE 1.5, ofwel J2SE 5.0 ofwel J5SE. Hoe de nieuwe Java in de volksmond moet gaat heten is nog niet helemaal duidelijk en dus hebben ze bij Sun vooralsnog de naam 'Tiger' naar voren geschoven (in officiële Sun-documenten worden er zelfs meerdere versies en namen door elkaar heen gebruikt!).

Langdradige Threads

Wat levert die nieuwe versie ons op? Misschien heeft u al een blik geworpen op de online documentatie of verhalen gehoord op congressen en symposia. Met veel tromgeroffel worden de verbeteringen gepresenteerd. De meeste verbeteringen vallen onder de categorie 'makkelijker ontwikkelen'. Ook voor beheerders en infrastructuur-mensen is het een en ander verbeterd (meer monitoring mogelijkheden en een aantal schaalbaarheids-issues aangepakt). De ene Java gebruiker zal blij zijn met de veranderingen, de andere zal ze niet nodig hebben of zelfs lelijk vinden. Ik vind ze vooral onbeduidend. Sinds de introductie van Java zijn er misschien nog nooit zo veel nieuwe en grote syntactische aanpassingen doorgevoerd, maar toch is het allemaal meer van hetzelfde.

Java zelf was ooit een grote verbetering. Doordat Java gekomen is zijn we anders gaan ontwikkelen. We zijn meer op de kern van het ontwikkelprobleem gaan focussen en ontwikkelen is gemakkelijker geworden. Verstokte liefhebbers van andere talen (Smalltalk, C++, …) zullen stellen dat Java niets anders is dan het bij elkaar rapen van een aantal oude bekende principes. Toch heeft Java ons werk verbeterd. We hoeven ons nu minder met platformen, bits en pointers bezig te houden. We kunnen nu veel gemakkelijker foutafhandeling gedegen aanpakken. Er zijn tools ontstaan die ons volledig anders laten werken. Kortom, de wereld is er mooier op geworden. Wordt de wereld mooier van de wijzigingen die er nu aan komen? Nee.

Waar zit ik dan wel op te wachten? Ik ben bang dat ik, ondanks mijn liefde voor Java, zit te wachten op aanpassingen die helemaal niet passen in Java. Aanpassingen die de kern van Java raken en die dus nooit doorgevoerd zullen gaan worden. Misschien moet ik wel wachten op een hele nieuwe taal. Wat ik wil, is een nieuw threading model. Wat mij betreft is het huidige model van threads en methods veel te star en traditioneel. Stel je voor, een webwinkel waar je auto's kunt kopen. Je klikt op een auto ("jou wil ik hebben") en je krijgt daarna één scherm met daarop een aantal vragen: "Welke kleur had u gewenst?" en ook "welk financieringsmodel had u gewenst?" en "welke dealer moet de auto leveren?". Dit valt in Java alleen maar op een hele gekunstelde manier te maken. Je zou aan het auto-object willen vragen: "lever jezelf aan die klant", waarna de auto aan de slag gaat met kleuren, financiering en dealers. Het huidige Java kan niet anders dan deze acties één voor één afhandelen. Omdat je deze vragen wel degelijk tegelijk aan een gebruiker aan wilt kunnen bieden, ontkom je er niet aan om deze overkoepelende logica buiten het object om te programmeren. Die logica komt dan terecht in een ouderwetse traditionele presentatielaag.

Stel je voor: een Java-constructie waarin objecten kunnen zeggen: "Ik heb die method zoveel mogelijk uitgevoerd, en heb nog wat vragen uitgezet. Kom later nog maar eens terug, of draai de boel maar terug". Dat zou nog eens vernieuwend zijn. Objecten die op een wat meer "fuzzy" manier omgaan met hun methods. Dan zijn transacties ook echt onderdeel van de kern van de zaak geworden. Momenteel zijn transacties niks anders dan een lastige bijkomstigheid van een vervelende OR mapping. Helaas past het niet in Java. We moeten het doen met meer van hetzelfde.

Daan Kalmeijer is docent consultant bij CIBIT adviseurs | opleiders (e-mail: daan@cibit.nl).