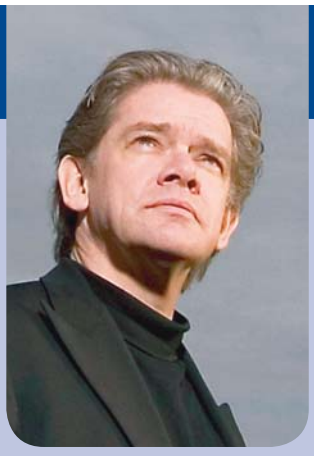
Ron Tolido



Java Mijlpaal

Ik weet niet precies wat ik ervan moet denken. Sun heeft eindelijk de code van Java vrijgegeven aan de Open Source-gemeenschap. Volgens Sun's Executive VP Rich Green gaat het om niets minder dan een 'mijlpaal voor de hele IT-industrie'. Je zou ook kunnen zeggen dat het om een bevestiging gaat van iets dat we al veel langer wisten: er valt eenvoudigweg geen geld te verdienen met het ontwikkelen en onderhouden van een programmeertaal.

Maar goed, een mijlpaal dus. Kijk eens aan. Dat past dan neem ik aan naadloos in de enorme rij van gedenkwaardige momenten die we in de afgelopen tien jaar met Java hebben meegemaakt.

Ik aarzel nog een beetje wat ik tot nu toe de meest indrukwekkende mijlpaal vind.

Misschien is het wel het verschijnen van de eerste *applets*: van die ellendige stukken code die er veel te lang over deden om te laden in je browser en je vervolgens trakteerden op een gekunstelde animatie of een lachwekkende poging om iets op een invulformulier te laten lijken. Met de komst van Java-applets verloor het nog zo jonge principe van de lichtgewicht Internet browser in één klap zijn onschuld. En getuige de bonte kakofonie van niet gestandaardiseerde, maar *best wel artistieke* 'plug-in' gebruikersinterfaces in de browser van nu (hartelijk dank ook aan de enthousiaste navolgers die ons ActiveX en Flash brachten) echoot dit historische Java-moment nog elke dag door.

Of wacht, zal het toch de doorbraak van objectoriëntatie zijn geweest? Met het populair worden van Java is iedereen pas echt goed aan de *inheritance* en het *dynamisch polymorfisme* gegaan, evenals vele andere eh... complexiteitsverlagende concepten. Een van de meest verheffende verworvenheden van objectoriëntatie is wel de nagenoeg volmaakte incompatibiliteit met het principe van relationele databasesystemen: dat heeft geleid tot een reeks van kromme tot zeer briljante oplossingen in de vorm van *object-to-relational mappers*. Oplossingen voor een probleem dat we zelf met droge ogen in ons vakgebied hebben toegelaten. Met veel moeite een pompoen in een te kleine metalen kubus weten te proppen en er dan nog trots op zijn ook: dat kan alleen in de IT.

Maar laten we eerlijk zijn: de meest memorabele mijlpaal is toch wel dat achteraf ondefinieerbare moment geweest waarop we Java (en daarna het Microsoft-zusje C#, laten we Redmond hier vooral niet vergeten) zijn gaan opvatten als de rechtmatige opvolgers van de 4GL's van de jaren '90. Hadden we net de manier gevonden om met simpele, hoogproductieve talen systemen te ontwikkelen die dicht tegen de gebruikersinterface, de database en de bedrijfsvoering aanzaten, wierpen we ons zelf vrijwillig terug naar de Spartaanse taferelen van een generatie eerder. Bijna een 'pré-3GL' wereld waarin moeilijkdoenerij en ambachtelijk coderen vanaf de *command line* de culturele norm waren.

Zou Java ook zo populair zijn geworden als het gewoon 'OAK' was blijven heten? Niet te lang over nadenken.

De eerlijkheid gebiedt ons vast te stellen dat Java een mooi gestructureerde, goed doordachte programmeertaal is die bovendien relatief weinig ruimte laat voor beveiligingslekken en bugs door geklungel met geheugen. Bovendien is er een karrenvracht aan goed geoptimaliseerde applicatieservers die Java-programmatuur snel en platformonafhankelijk kunnen draaien. Nu nog zorgen dat we met Java een hogere productiviteit bereiken. En dat kan maar op één manier: *zo weinig mogelijk Java coderen*. Steek daarom ongemakkelijk veel tijd in het vinden en opzetten van *frameworks* – al dan niet met dank aan Open Source- zodat het bouwen van oplossingen vooral bestaat uit het invullen van wat nog nét niet helemaal af is. En overweeg serieus *modelgedreven ontwikkeling*: als de tekenen uit de praktijk ons niet bedriegen wordt het steeds realistischer om het overgrote deel van Java-code uit (al dan niet grafische) businessmodellen te laten genereren. Aan de horizon lonken bovendien al *Domain Specific Languages*: krachtige talen die speciaal bedoeld zijn om oplossingen voor een bepaald domein in zo weinig mogelijk bewoordingen uit te drukken, waarna helemaal aan het einde – vlak voor de oplevering – mogelijk nog iets aan codegeneratie wordt gedaan.

En dat brengt ons, via een knap verwarrend oponthoud, weer terug naar waar we ons een jaar of tien geleden met 4GL's zo veelbelovend naartoe bewogen.

Is dat pas écht een mijlpaal, als we straks geen regel Java meer coderen?

tolido.blogspot.com