Ron Tolido

Ontwikkelen voor Godot

Informatietechnologie. Vaak niet meer dan een absurdistisch toneelstuk. *Op een verlaten landweg voeren twee C#-programmeurs een steeds vager wordend gesprek. Ze zijn in afwachting van meneer Godot, opdrachtgever en hoofd van de gebruikersorganisatie. Godot weet als enige wat het doel en nut is van de systemen die moeten worden ontwikkeld. Jammer daarom dat hij nooit zal komen opdagen. Dat weerhoudt de beide hoofdrolspelers er niet van om diepgaand te discussiëren over hoe ze meneer Godot nog beter van dienst kunnen zijn. Het publiek zakt er eens lekker voor onderuit.*

Het is niet zo moeilijk voor te stellen waar de dialoog over zou gaan. Als toneelschrijver tref je een wereld vol ongerijmde inspiratie aan in de kringen van *software engineering*. Al langer dan iemand zich ook maar kan herinneren wordt er gewerkt aan methoden, technieken en tools om de productiviteit van systeemontwikkeling te verhogen. En het fraaie is dat het om een wereld gaat die zichzelf in stand houdt: zodanig lang dat niemand meer weet wat het oorspronkelijke probleem nou ook alweer was.

Zo heeft zich een complete industrie geworpen op *Enterprise Application Integration*. Er worden platforms ontwikkeld, *integration brokers* en XML-standaards. In dikke boeken vol patterns en best practices kun je lezen hoe je 23 totaal verschillende systemen met elkaar kunt verbinden. Niemand die zich aan de zijlijn dommig afvraagt waarom dat aantal eerst niet eens tot drie wordt teruggebracht.

Voorts ken ik een Britse IT-expert die een zeer succesvolle, internationale carrière heeft opgebouwd als *Java Troubleshooter*. Hij reist dagelijks heel Europa af om projecten uit het slop te trekken die gebukt gaan onder geheimzinnige bugs en vooral performanceproblemen. De grootste veroorzaker van de ellende blijkt hem te zitten in de totale *mismatch* tussen de objectoriëntatie van Java en het relationele model van databases. Ontwikkelaars van het type *Getatoeëerde Snowboarder* maken hippe ontwerpen in UML en bouwen de ene Enterprise Javabean na de ander, bij voorkeur in de vorm van veel te fijnmazige, zevenvoudig gelaagde class-hiërarchieën. Al dat fraais moet dan helemaal aan het einde worden opgeslagen in zo'n platte, ordinaire relationele database. En dat wringt dan vreemd genoeg.

Onze Britse probleemoplosser vaart er wel bij. Hij wordt ijlings ingevlogen om de grootste stommiteiten ongedaan te maken. Doorgaans via het extreem versimpelen van de gekozen oplossingen en het wegvijlen van de ergste object-georiënteerde dikdoenerij.

Bang om zonder werk te komen hoeft hij de komende jaren niet te zijn. De Machine van Godot draait op volle toeren.

De volgende versies van C# en Java introduceren nog meer ingewikkelde, volmaakt zinloze taalelementen. Programmeurs zullen zich *en masse* laten bijscholen, want de kennis moet wel actueel blijven.

Een nieuwe, verfrissend naïeve generatie productleveranciers denkt te weten hoe software uit ontwerpspecificaties kan worden gegenereerd; maar de opkomst van het nog altijd even kansloze MDA (*Model Driven Architecture*) zal hoogstens leiden tot meer slecht geoptimaliseerde, nauwelijks te doorgronden code.

Zelfs SAP helpt een handje mee. Het kortgeleden gelanceerde NetWeaver platform geeft de tot nu toe ongecompliceerd levende SAP-programmeurs de volledige rijkheid van J2EE tot de beschikking. Ik zou maar alvast wat herdrukken van *Java voor Dummies* gaan plannen. Zomaar een ingeving.

Misschien moeten we onze hoop vestigen op webservices. Die zijn voorlopig zo plat en eenvoudig in het gebruik dat daar nauwelijks moeilijke software omheen valt te ontwerpen. Veel meer dan een grafische editor en een scriptingtaaltje om de services aan elkaar te knopen heb je uiteindelijk niet nodig. Koppel dat aan een paar goed geplaatste, ouderwetse *databasetriggers* en je hebt de Zen-achtige leegheid waarmee complete RAD races zijn gewonnen. Je zou er bijna filosofisch van worden. Tijd voor weer zo'n bespiegelende haiku:

Java programmeur
Gestrand in brij van syntax
Dronken van code