Bericht uit 


**Een tijdje geleden kwam ik op kantoor toen een collega van mij een nieuwe Java interactieve ontwikkelomgeving demonstreerde. De ontwikkeltool zette blijkbaar in op een interactieve, klantgedreven, demonstratie want het eerste wat hij mij vroeg was: "Wat wil je doen?". Ik was in een zeer recalcitrante bui, en antwoordde: "Ik wil graag mijn informatiearchitectuur ingeven". U snapt het al, dit was het einde van de demonstratie.**

# Wat kan een Java-tool?

De nieuwste inzichten op het gebied van het ontwikkelen van informatiesystemen gaan uit van een gedistribueerde object-georiënteerde aanpak waarin functionaliteit en verantwoordelijkheden over objecten worden verdeeld volgens een vierlagen architectuur. In die architectuur onderscheiden we de "view"-laag, de "proces"-laag, de "domein"-laag en de "infrastructuur"-laag. Objecten in een laag kennen wel de structuur van de onderliggende, maar niet van de bovenliggende lagen. Daarnaast worden voor veel voorkomende ontwerpproblemen steeds meer gebruik gemaakt van "patterns": standaard ontwerppatronen die hun waarde in de praktijk reeds ruimschoots hebben bewezen. In feite vormen deze patronen herbruikbaarheid van deelontwerpen; in mijn ogen een veel betere vorm van herbruikbaarheid dan het hergebruik van code. Gegeven het feit dat de gedistribueerde object-georiënteerde aanpak de gewenste is worden we daarin nog maar bijzonder slecht ondersteund door de ontwikkeltools van vandaag. Want zijn de populaire Java-ontwikkeltools eigenlijk niets anders dan veredelde "Visual Basic"-omgevingen waar toevallig een veel betere programmeertaal achter ligt? Er wordt in die tools volledig vanuit het grafische gebruikersinterface geredeneerd en de indruk wordt gewekt dat het ontwikkelen van een Java-programma begint met het via "sleep en plof" in elkaar steken van schermen, om vervolgens achter die schermen nog wat Java-code te hangen. Niets is natuurlijk minder waar!

Genoemde tools zien er flitsend uit en lenen zich dan ook voor flitsende demonstraties waar een presentator voor het toeziend oog van managers en hoofden systeemontwikkeling een kant-en-klare applicatie uit de grond trekt met behulp van een paar wizards en vier-en-een-halve regel Java-code. Maar slaan dergelijke demonstraties ergens op? De haastig in elkaar gestoken applicatie voldoet meestal aan geen enkele eis van performance, concurrency, schaalbaarheid en onderhoudbaarheid. Verder leveren die tools meestal 2-tier applicaties op, en daar willen we nu juist van af! De eerstvolgende leverancier die me nog een visuele "data-aware grid control" met-database-koppeling wil laten zien krijgt dus de wind van voren!

De meeste Java-ontwikkeltools zijn vrij recht-toe-recht-aan ports van de bekende C, C++, Pascal/Delphi en Basic ontwikkeltools. Zij richten zich op de allerlaatste fase van een softwaretraject: de bouw, en volgens mij zit hem daar helemaal niet de essentie van de softwareontwikkeling. De problemen zitten hem juist in de analyse en ontwerpfase! Bouwen en debuggen, alhoewel best belangrijk, horen eigenlijk triviaal te zijn! De dingen die de genoemde tools goed oplossen, namelijk het tekenen van schermen en het debuggen van de code, vormen in mijn ogen eigenlijk geen groot probleem. Sterker nog, het gaat vaak handiger zonder al die tools die de aandacht afleiden van waar het om gaat en vaak tot slechte applicaties leiden. Het zou zo mooi zijn wanneer er Java-ontwikkeltools zouden komen waarmee je je klasse-model kan ingeven, ontwerppatronen kan toepassen, interfaces kan specificeren, de aard van objectcommunicatie kan definiëren (rechtstreeks, CORBA, DCOM), om dan als klap op de vuurpijl vervolgens vanuit dat model de daadwerkelijke Java-code ook nog in te kloppen. En dit alles multiplatform en betaalbaar! Je zou me vooralsnog niet blijer kunnen maken!

*Jos Visser*
is werkzaam als ontwikkelingswerker
bij Open Solution Providers.
Hij kan worden bereikt via josv@osp.nl.