Bericht uit 

“Hetgeen er geweest is, hetzelve zal er zijn, en hetgeen er gedaan is, hetzelve zal er gedaan worden; zodat er niets nieuws is onder de zon. Is er enig ding, waarvan men zou kunnen zeggen: Ziet dat, het is nieuw? Het is alreeds geweest in de eeuwen, die voor ons geweest zijn.” (uit Prediker 1:9 en 1:10).

De geschiedenis herhaalt zich

Al vanaf het eerste begin mag Java zich in een warme belangstelling verheugen. Logisch, het Java platform wordt immers gekenschetst door alle moderne “buzz words” en in onze branche is men altijd genegen het oude te verketteren en het nieuwe onvoorwaardelijk te omarmen. Met de kracht van het eigen gelijk stonden er onmiddelijk een groot aantal profeten op om de Java-religie onder de ongelovigen te verspreiden. Java was het nieuwe wereldwonder wat alle oude problemen op zou lossen. Hoog werd er opgegeven over de verschillende nieuwe en unieke eigenschappen van Java. In een van de eerste whitepapers van Sun over Java stond deze informele beschrijving van Java: “Een simpele, object georiënteerde, gedistribueerde, geinterpreteerde, robuuste, veilige, architectuurneutrale, portable, high-performance, multithreaded en dynamische programmeertaal”. Deze beschrijving deed een van mijn collega’s de uitspraak ontlokken dat Java volledige “hype compliant” is. Als je echter even door de marketing hype heenkijkt is het natuurlijk duidelijk dat de door Sun genoemde kenmerken van Java niet noodzakelijkerwijs uniek, nieuw of zelfs maar waar zijn. Mijn eerste opmerking betreft ook meteen het eerste steekwoord wat Sun noemt: “Simpel”. Laat ik bij deze wederom een poging doen om dit uit de wereld te helpen: Java is geen simpele programmeertaal! Ik heb de eer gehad om flink wat Java programmeercursussen te geven, en mensen zonder echte programmeerervaring hebben niet noemenswaardig minder problemen met het leren van Java als met het leren van een andere programmeertaal. Wel is het voor programmeurs die al een 3GL-programmeertaal beheersen relatief eenvoudig om naar Java over te stappen. Dat zegt echter meer iets over de aansluiting tussen Java en de bestaande programmeertalen als over Java zelf. Met betrekking tot portabiliteit verwijs ik de lezer volgaarne naar het UCSD P-code systeem (herinnert u het zich nog?). Ongeveer 17 jaar geleden maakte ik op mijn middelbare school kennis met UCSD Pascal: een Pascal-programmeeromgeving gebaseerd op een architectuur neutrale tussencode (pseudo code, ook wel p-code genoemd). De USCD Pascal compiler genereerde die P-code en kon daardoor zonder problemen draaien op iedere hardware waarvoor een P-code Virtual Machine beschikbaar was. Naast een Pascal compiler bestonden er ook een Fortran en naar verluidt zelfs een Cobol compiler voor het USCD P-code systeem. Western Digital heeft nog een tijdje microprocessoren verkocht die de P-code rechtstreeks konden uitvoeren! Wat kunnen we hieruit concluderen: (a) ik zat op een vooruitstrevende middelbare school, en (b) de geschiedenis herhaalt zich! Nu we toch Cobol al even genoemd hebben: Een klant van ons haalde ongeveer 6000 Cobolprogramma’s over van een Bull mainframe naar een Unix-systeem. Na compilatie en testen bleek dat 6 van die 6000 programma’s het niet ‘in een keer deden’. Hoezo portable? Ook op andere gebieden kan Java niet echt pochen op iets daadwerkelijk nieuws: Occam, Smalltalk, Lisp, Ada en andere programmeertalen zijn moeiteloos beter en completer in de implementatie van alle of sommige van de genoemde features. Waarom is Java dan toch zo populair? Mijn verklaring hiervoor is dat Java “fris” is. Het is gegeven de huidige stand van de techniek de meest ideale verzameling programmeertaal-features in een nieuw jasje en opnieuw geïmplementeerd. Zonder ballast uit het verleden en voorzien van een modern en goedogend buitenkantje. Geen nieuwe features onder de motorkap, maar met de nieuwste styling. We zien op de Auto-RAI bijna niet anders.

Jos Visser
is werkzaam als techno-profeet bij
Open Solution Providers.
Hij is bereikbaar via josv@osp.nl