Bericht uit 

Alweer geruime tijd geleden werden we in het vakgebied verblijdt met het begrip 'softwarecrisis'. De informatietechnologie bleek na enig navelstaren ontzettend goed in het te laat en tegen te hoge kosten opleveren van niet-werkende programmatuur, en vervolgens ook nog uitstekend in staat om die programmatuur slecht te onderhouden en gaande weg nog meer fouten te introduceren. Wat voor rol speelt Java in deze crisis?

De menselijke factor

De existentiële crisis waarin de software-industrie zich bevindt, begint langzamerhand aanzienlijke maatschappelijke gevolgen te krijgen. Marslanders raken zoek, uitkeringen worden verkeerd uitgerekend, overledenen krijgen een oproep voor een medische keuring en de nieuwste versie van een bekend besturingssysteem wordt op de markt gebracht terwijl er nog zo'n 63.000 bekende fouten in zitten. Het is duidelijk tijd voor een andere aanpak. Betere programma's schrijven kan in principe op twee manieren: met slimmere programmeurs of met een betere aanpak en betere hulpmiddelen. Nu heb ik persoonlijk de hoop dat de mensheid op afzienbare tijd slimmer gaat worden wel zo ongeveer opgegeven. Wellicht dat als we het menselijk genoom geheel hebben gekraakt dat we het IQ-gen in ongeborenen kunnen gaan manipuleren, maar dan zal je net zien dat dit gen is gekoppeld met het gen voor asociaal gedrag, en ik ben toch liever lid van een lieve maar domme soort. Dus moet het uit de aanpak en de hulpmiddelen komen. Qua aanpak lijkt de objectoriëntatie het beste wat we tot dusver hebben verzonnen. Ik geef toe, het is niet het ultieme antwoord op alle problemen, maar het draagt bij aan betere, stabielere programmatuur die eenvoudiger is te onderhouden en aan te passen. Echter, dom toegepast is ook OO niet in staat om lachwekkende fouten te verhinderen. Toen het Australische leger voor een helicoptersimulator wat kangaroes nodig had pakten ze wat klassen die ze al hadden liggen en pasten het uiterlijk en de bewegingspatronen aan. In de eerste test-run hopten de kangaroes voor de helicopter uit, over een heuvel, pakten hun mobiele SAM-lanceerinrichting en begonnen rakketten op de helicopter af te vuren. Een typisch voorbeeld van een goed idee slecht uitvoeren. Een ander belangrijk element is de beschikbaarheid van betere hulpmiddelen die de programmeur assisteren in het kiezen van een goede architectuur en hem/haar zo compleet mogelijk ondersteunt bij het voorkomen en signaleren van fouten en problemen. Traditioneel bestaat daartegen bij programmeurs nogal wat weerstand, want hoe slimmer het hulpmiddel is, hoe meer de programmeur wordt beperkt in het uitleven van zijn/haar eigen slimme ingevingen. En daarbij gaan dit soort dingen vaak ten koste van de prestaties van de programmatuur tijdens de uitvoering. In een tijd van 1Ghz-processoren op de desktop en genoeg geheugen om de complete bevolkingsgegevens van de mensheid 'in core' op te slaan wordt dit argument echter te vaak onjuist te berde gebracht. De makers van Java hebben getracht een omgeving te scheppen waarbij een aantal essentiële problemen, die aan de basis liggen van de softwarecrisis, in principe kunnen worden opgelost. Ondersteuning voor objectoriëntatie, allerlei controles tijdens uitvoering en het de programmeur verplichten mogelijke foutsituaties af te vangen zijn daar voorbeelden van. Daarnaast worden de ontwerper en de programmeur allerlei mogelijkheden geboden om applicaties goed te structureren en algoritmes elegant en efficiënt te implementeren. Maar, en nu komt de crux, Java staat ook toe om alles wat we de afgelopen tien jaar hebben geleerd in de wind te slaan en al vrolijk prutsend een krakkemikkige en bij puur toeval werkende applicatie in elkaar te knutselen. En dus zijn we per saldo niet verder dan voorheen: de menselijke factor blijft allesbepalend.

Jos Visser
is volhouder bij Open Solution Providers en bereikbaar via jos@osp.nl.