Bericht uit 

Afgelopen week klaagde ik mijn nood bij een collega over de barre kwaliteit van de programmatuur waar ik bij een klant mee was geconfronteerd. Nu is het wel zo dat als ik iets niet zelf heb ontwikkeld, ik al snel vind dat het beter kan. Maar deze keer was het weer eens ouderwets "huilen met de pet op". De ethiek verhindert mij helaas om op de details in te gaan, maar laten we volstaan met de constatering dat ik op de eerstkomende feestjes weer aardig wat anecdotes uit mijn mouw kan schudden...

Waar blijft het Java-ambachtsgilde!?

Terug naar die collega. Als reactie op mijn klagen stelde hij vast dat er wel delijk was geprogrammeerd (en nog slecht ook) maar waar geen softwareontwikkeling bij te pas kwam. Op het eerste gezicht lijkt dit een tegenstrijdige bewering, maar bij nadere bestudering schuilt er een diepzinnige waarheid achter. Toen ik later op de dag nog eens op die opmerking liep te herkauwen drong het tot mij door dat Java bij uitstek een taal is waar dit onderscheid tussen programmeren en software ontwikkelen levensgroot is. Programmeren is eigenlijk het timmeren en metselen van de informatica: het ligt aan de basis van alles wat wij doen, en de kwaliteit van dit ambachtelijke werk bepaalt voor een groot gedeelte de kwaliteit van een informatiesysteem. Zoals we echter geen huis kunnen bouwen door te beginnen met timmeren, kunnen we ook geen informatiesysteem bouwen door te beginnen met programmeren. Achter dit alles moet een plan liggen: een architectuur, een ontwerp en een bouwplan: in principe uitgekauwde koek natuurlijk. In de praktijk blijkt het echter maar al te vaak dat er, met name bij kleinere informatiesystemen, maar gewoon begonnen wordt. Opzich hoeft dit niet slecht te zijn, mits de ambachtsman(of -vrouw) in kwestie zich maar houdt aan de principes van goede software engineering. Zonder ontwerp is dit niet triviaal en met de juiste instelling en ervaring is dit best te doen. Een ambachtelijke programmeur structureert haast automatisch zijn of haar software op een beetje doorzichtige wijze, en maakt bijna onbewust goed gebruik van de features van de programmeertaal en -omgeving. De slechte of onervaren programmeur doet dit niet, en broddelt net zolang door totdat het werkt. Het resultaat is dan een slecht product, wat met wat geluk functioneel nog wel min of meer klopt, maar wat meestal instabiel, ondoorzichtig en moeilijk onderhoudbaar is. Dit onderscheidt de meester van de leerling. Java is een taal waar het onderscheid tussen de meester en de leerling zeer duidelijk wordt. De taal zit boordevol met eigenschappen die, mits juist gebruikt, leiden tot fraai gestructureerde, stabiele, doorzichtige en (daardoor) makkelijk onderhoudbare applicaties. Ik denk hierbij aan dingen als objectoriëntatie, interfaces, multi threading, packages, exceptions, objectserialisatie, dynamische class loaders, introspection, enz.. In de handen van de meester zijn deze zaken de ideale toolset om eens lekker uit te halen. Aan de slecht oplettende leerling is al dit fraais echter niet besteed. Deze programmeert Java of het Visual Basic is, en maakt geen of verkeerd gebruik van de mogelijkheden. Eeuwig zonde, maar onder de dwingende begeleiding van een goede leermeester wordt het wellicht nog wel eens wat met zo iemand. De markt werkt hier echter niet aan mee, want de eerste de beste prutsende beginner die weet dat Java een programmeertaal is (en die wellicht weleens een applet heeft overgetypt) kan al 150 gulden per uur vangen. Het instellen van een programmeursgilde met een verplicht leerlingstelsel is dan ook het overwegen meer dan waard. De Middeleeuwen waren zo gek nog niet!

Jos Visser

is werkzaam als erkend talent bij Open Solution Providers en bereikbaar via josv@osp.nl.