Ron Tolido

Boter karnen. Er wordt vaak lacherig over gedaan. Zó makkelijk, afgeven op een ambacht waarvan haast niemand meer iets weet. Ga er maar eens aanstaan, urenlang die verzuurde room stampen in een echte stampkarn. Dan hoor je niemand meer grinniken, als pas na drie kwartier de eerste boterkorrels komen bovendrijven.

Verouderend Ambacht

Hetzelfde geldt voor andere bijna vergeten ambachten, zoals tin gieten, papier scheppen, vlas bewerken en pitrieten manden vlechten. Hoogstens leuk als toeristische attractie om op de zaterdagmarkt een minuut of vijf grijnzend gade te slaan. Ach ja, vroeger. Toen had je nog geen machines en fabrieken, kinderen. Waar is de tijd gebleven. Maar kom, verderop lonkt het standje met Vietnamese loempia's.

Is programmeren ondertussen ook een verouderend ambacht aan het worden?

Alles wijst erop. Het bouwen van de grote, complexe systemen laten we steeds meer over aan de leveranciers van *standaardpakketten*. Vooral veel configureren en misschien wat regeltjes in een 4GL-achtige taal als ABAP erbij schrijven, daar komt het wel zo ongeveer op neer. Voor de rest is het een kwestie van intensief praten met de toekomstige gebruikers en zorgen dat de processen niet teveel wringen met het systeem. Tel daarbij op dat steeds meer standaard *software on demand* via het Internet wordt afgenomen, en het zou zomaar kunnen dat je geen regel code meer tegenkomt in een project.

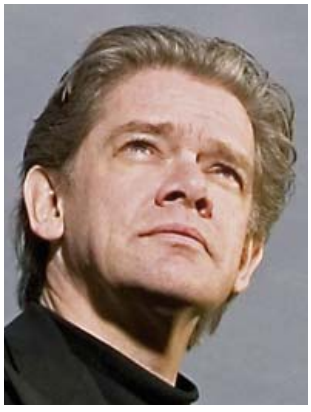
Als we al eens iets strategisch doen rond informatievoorziening, dan produceren we in toenemende mate *Business Intelligence*. Scorecards, dashboards en grafieken: je hoeft er nog geen Hello World voor te kunnen programmeren. Wel moet je er analytisch voor zijn ingesteld, moet je verstand hebben van complexe datastructuren en moet je een diepgaand inzicht hebben in de bedrijfsvoering van de klant. Een schaarse combinatie van eigenschappen en het is daarom niet zo vreemd dat we de grootste tekorten op de arbeidsmarkt in deze contreien tegenkomen.

Verder komen we een groeiende groep van *procesexperts* tegen: die modelleren processen met behulp van grafische tools, vaak onderdeel van een *Business Process Management* systeem. Daarmee worden processen niet alleen in kaart gebracht maar ook automatisch uitgevoerd en beheerd. De benodigde systemen om die processen te ondersteunen klik je letterlijk bij elkaar; puttend uit een bak van services creëer je zo *composite applications*. Vooruit, misschien nog een beetje BPEL en wat artistieke *If-Then-Elsejes* om de boel aan elkaar te lijmen. Meer coderen zit er niet in. En de complexere bedrijfslogica wordt ondergebracht in declaratieve regels, klaar om verwerkt te worden door een externe *rule engine*.

Als we dan al eigenhandig een écht systeem willen bouwen, dan ligt de nadruk op productiviteit, effectiviteit en kwaliteit. Lees: als je in een hoog tempo goede code wil produceren is er maar één strategie: *zo weinig mogelijk coderen*. Niet zo verwonderlijk dat er veel belangstelling is voor de Domain Specific Language en haar stugge broertje *Model Driven Architecture*. We vinden steeds betere manieren om systemen te specificeren op een hoger niveau en daaruit de code te genereren. Dat gaat sneller en geeft minder fouten, vooral als we kunnen putten uit een arsenaal van bewezen referentiemodellen.

Gebruikersinterfaces dan? Die schilderen we vanaf een paletje *mashup-componenten* zo aan elkaar. Of we laten het de gebruikers zelf doen vanuit hun gepersonifieerde, *webgebaseerde portal*. Complexe integratie? Vroeg of laat spreken alle systemen dezelfde servicegeoriënteerde dialecten. En de *integratiemiddelware* zit ergens diep in het netwerk verborgen. Niet iets om de hele dag mee bezig te zijn.

Het zal wel een kenmerk zijn van een volwassen wordende industrie. Veel handwerk wordt uiteindelijk vervangen door industrialisatie en standaardisatie. Het Oude Ambacht resteert. Misschien zien we het ooit nog eens op de braderie: een echte Java-programmeur die in authentieke klederdracht (inclusief een zwart T-shirt met Duke erop) vanaf de *command line* rauwe code produceert. Ach ja, vroeger.



Ron Tolido
tolido.blogspot.com