Programmeren is te moeilijk

Enige tijd geleden viel mijn oog op een artikel dat stelde dat we niet meer hulpmiddelen voor programmeurs nodig hebben, maar vooral veel betere programmeurs. De essentie van het verhaal was dat programmeurs die veelvuldig gebruik maken van bepaalde hulpmiddelen niet meer goed kunnen programmeren, omdat ze te beperkt bezig zijn. In plaats van de hulpmiddelen te gebruiken moeten ze vooral veel zelf programmeren.

Op het eerste gezicht lijkt hier wel wat in te zitten, maar nader bekeken is het natuurlijk onzin. Als iedereen een superkei in zijn vak zou zijn dan gaat het altijd wel goed, maar zoals in ieder vak kom je meestal wel uit op een normaal-verdeling. Tien procent is topper, tien procent bakt er niets van en de rest zit ergens tussenin. Niets mis mee, de een heeft nu eenmaal andere talenten dan de ander. Maar de oplossing van het probleem is natuurlijk niet om alle hulpmiddelen die het leven eenvoudiger maken af te schaffen. Alsof we het onszelf zo moeilijk mogelijk zouden moeten maken zodat we tenminste alles kunnen als dat nodig mocht zijn. Wanneer je deze redenering volgt, zouden we allemaal nog fijn in Assembler programmeren. Dan kunnen we tenminste echt programmeren!

Misschien is het beter om na te denken hoe het komt dat die negentig procent zo'n moeite heeft om productief te zijn en kwaliteit te leveren. Het zijn misschien geen toppers, maar grotendeels wel degelijk capabele mensen.

Laten we eens kijken wat een ontwikkelaar allemaal tegenkomt bij een moderne applicatie. Neem bijvoorbeeld een gemiddelde Java-applicatie. Daarin wordt gebruik gemaakt van onder meer: de Java-programmeertaal, een aantal Java-API's, meestal wel wat XML, XDS en XSLT, JSP, Struts, Session Beans, Entity Beans of wellicht Hybernate of Spring, SQL, UML, webservices, en tegenwoordig ook nog Ajax. Deze lijst is niet compleet, maar geeft wel een aardig beeld van het aantal technische details dat een ontwikkelaar dient te weten.

Hoeveel mensen begrijpen dit in detail? Hoeveel mensen zijn nog in staat om alle onderlinge verbanden en afhankelijkheden te doorzien? Gezien de complexiteit zullen dat er maar weinig zijn, zelfs de toppers hebben er al moeite mee.

Als één van de auteurs van de UML-standaard ken ik UML beter dan de meeste mensen. Kijkend naar UML 2.0 moet ik bekennen dat ook ik niet meer alle details begrijp, laat staan alle onderlinge verbanden. Sterker nog, mijn inschatting is dat niemand ter wereld dit nog kan. En dan hebben we het nog maar over één van de gebruikte technieken. Het is toch geen wonder dat de meeste ontwikkelaars overweldigd worden, en fouten maken waarvan ze zich niet bewust zijn? Kunnen we ooit verwachten dat iedere ontwikkelaar dit allemaal begrijpt? Nee dus!

De enige oplossing is om de veelheid aan complexe details te verminderen. Dit betekent dat software-ontwikkeling naar een hoger abstractieniveau getild zal moeten worden. Dit is de reden dat Model Driven software-ontwikkeling, of het nu is met MDA of met DSL's, wel succesvol móet zijn. Het verhoogt het abstractieniveau en verbergt onnodige complexiteit. Flexibele code-generatoren kunnen het grootste deel van het programmeerwerk uit handen nemen. Ze leveren meer productiviteit en betere kwaliteit dan we handmatig ooit zullen bereiken. We moeten dus juist minder programmeren, en veel meer modelleren.

Voor het realiseren van de juiste abstracties is veel kennis en inzicht nodig. Dit is het gebied waar de top-ontwikkelaars de meeste toegevoegde waarde hebben. Hun taak is niet zozeer het laten zien van individuele complexe hoogstandjes, maar veel meer het helpen productief maken van de grote groep iets minder getalenteerde collega's.

Jos Warmer,
Partner Ordina SI&D.
E-mail:
jos.warmer@ordina.nl.