Jos Warmer

Iedereen heeft gelijk

Sinds enige tijd verdiep ik me in domein specifieke talen. Binnen de modelleerwereld (UML, OCL, DSL, MDA, etc.) is het modelleren van bepaalde zaken zoals structuur gemeengoed geworden. Voor het aspect business logica kennen de meeste modelleertalen veel te weinig mogelijkheden. Ook vanuit de DSL-wereld blijkt dit geen eenvoudig op te lossen probleem te zijn. Vandaar dat ik maar eens om me heen ben gaan kijken. Zo kwam ik uit op het gebied van de business rules. Hoe zouden de DSL en de business rules-aanpak elkaar kunnen completeren?
Slecht idee! Het eerste artikel dat ik vond ging over DSL versus Business Rules. Wat er beter was! Geschreven door iemand uit de business rules wereld. De onvermijdelijke conclusie was dat je beter alles met business rules kunt doen. Zinvol artikel? Wat mij betreft niet. Iedere technologie heeft zijn sterke en zijn zwakke kanten. Helaas zie je veel te veel mensen die zo enthousiast zijn over een technologie dat ze volledig doordraven.
Zo zie je discussies over .NET versus J2EE, over UML versus DSL, over XML databases versus SQL databases, over declaratief versus procedureel versus functioneel programmeren en nog veel meer.
Op technisch gebied is bootstrapping een mooi voorbeeld. Een technologie moet met zichzelf ontwikkeld worden. Soms past dat, soms helemaal niet. De mensen die het voor elkaar krijgen zijn wel altijd heel trots op hun kunstje.
Naast het ophemelen van de "eigen" technologie door negatief te zijn over andere technologieën wordt de eigen technologie vaak als de silver bullet gezien. Alles valt ermee op te lossen en de reikwijdte van het toepassingsgebied wordt groter en groter.
In de modelleerwereld heeft men bijvoorbeeld geprobeerd om met visuele modellen het programmeren te vervangen. Het kan, maar de kracht van visuele modellen ligt op een ander gebied. Visueel programmeren is uiteindelijk nooit doorgebroken.
Een ander voorbeeld: leveranciers van een ESB hebben vaak ook ontwikkelomgevingen en applicatieservers voor het geval je er nog even iets met de hand bij moet bouwen. Er zijn natuurlijk genoeg andere, en over het algemeen veel betere, oplossingen, maar alles moet per se in de eigen technologie.
Vanuit het idee achter domain specifieke talen zou je juist zeggen dat iedere technologie zijn eigen thuisdomein heeft waar de technologie optimaal tot zijn recht komt. In andere domeinen kan een technologie best bruikbaar te maken zijn, maar veelal zijn er betere oplossingen mogelijk.

Het probleem van het gebruik van meerdere technologieën door elkaar is dat ze niet altijd op elkaar aansluiten. Een technologie waarmee je alles binnen één omgeving kunt lijkt dan wel ideaal. Pas hier mee op. Wanneer alles binnen één technologie gebeurd, dan is de kans groot dat de verschillende aspecten van de oplossing zo naadloos in elkaar overgaan dat je feitelijk een monolithische oplossing creëert. En we weten welke onderhoud- en legacy-problematiek dit met zich meebrengt.
Door het aansluitingsprobleem van verschillend technologieën worden we gedwongen om na te denken over de interfaces tussen de onderdelen van de oplossing. We moeten ze identificeren en vervolgens deze interface in de oplossing expliciet maken. Even een snelle shortcut is niet meer mogelijk. Door het gebruik van verschillende technologieën worden we dus gedwongen om een veel beter ontwerp te maken. Lastig, maat o zo nuttig. Zo heeft elk nadeel zijn voordeel.
Technisch gezien is er al veel mogelijk. Met behulp van XML en SOA kunnen veel zaken technisch al gecombineerd worden. Nu nog doen: beoordeel iedere technologie op zijn kracht en gebruik hem daar waar hij het beste to zijn recht komt. Dan heeft iedereen uiteindelijk een beetje gelijk.

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