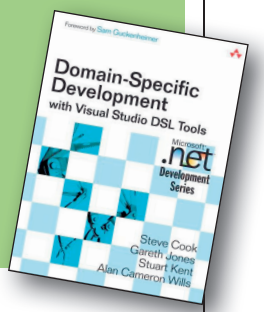
Sander Hoogendoorn

Lichtjaren geleden, toen ik voor mijn boek over UML nadacht over de verschillende modelleertechnieken die ik in een systeemontwikkelproject nodig vond, viel het me al op. Niet alle gezichtspunten in een project laten zich gemakkelijk vangen in een bestaande modelleertechniek. Hoe leg je een datamodel vast in UML? Wat doe ik met hiërarchische bedrijfsprocessen? Of nog voor de hand liggender: hoe modelleer ik de user interface?



Handleiding

Omdat ik vond dat ook die gezichtspunten een visuele weergave verdienden, beschreef ik in mijn boek hoe een datamodel is vast te leggen in een klassendiagram. Voor het vastleggen van de processen vond ik het business hierarchy diagram. En voor het modelleren van de user interface bedacht ik, na veel uitproberen in projecten, zelf een modelleertechniek, die ik het user interface diagram doopte.

En om een lang verhaal kort te maken, zo'n eigen modelleertechniek noemen we tegenwoordig een domain specific language (DSL). Hoewel een pionier als het Finse Metacase al jaren over een volwaardige tool beschikt voor het maken van DSL's, is het idee inmiddels ook door Microsoft opgepakt, getuige de Teletubbie designers in Visual Studio 2005. Maar Microsoft heeft grote plannen. Eindelijk beschikt het bedrijf over een technologie en bijbehorende toolkit (de DSL Tools) die het mogelijk maakt analyse en ontwerp van een project te integreren in Visual Studio – voorzover . NET-projecten overigens hierop zijn te betrappen.

En dus verscheen recent *Domain-Specific Development with Visual Studio DSL Tools* van de hand van Steve Cook, Gareth Jones, Stuart Kent en oudgediende Alan Cameron Wills. Laat ik meteen met de deur in huis vallen: als je wilt weten wat een DSL eigenlijk is, en waarom je er een wilt maken is *Domain-Specific Development with Visual Studio DSL Tools* niet jouw boek. Deze titel gaat niet over DSL's, maar over Microsoft's DSL Tools. Dat is een belangrijk onderscheid. Het boek is vooral nuttig wanneer je al weet wat voor DSL je zelf wilt ontwerpen, zoals een user interface diagram. De auteurs gaan grondig in op het definieren van een DSL, hoe het domeinmodel van zo'n DSL tot stand komt, hoe de uiteindelijke diagrammen eruit gaan zien, tot aan alle manieren om een connector aan een modelelement te verbinden toe, hoe aanvullend gedrag voor een DSL is te programmeren, hoe er code mee te genereren is en dan is er nog de deployment. In de 576 pagina's van het boek leggen de auteurs alle details van de DSL Tools bloot, soms tot op het punt dat je denkt: ok, *I'll take it from here*.

Oneerbiedig gezegd is *Domain-Specific Development with Visual Studio DSL Tools* een handleiding. Een goede weliswaar, maar het blijft een handleiding. Zeg nu zelf: lezen wij techneuten wel eens een handleiding? Toch pas nadat we zelf hebben zitten knutselen en er niet uit komen? En dat is nu net hèt moment om dit lijvige boek open te slaan.

Mijn belangrijkste conclusie na het lezen van het boek was overigens dat er voor het helemaal uitwerken van een DSL, met alle puntjes op de i, zoals het veranderen van de cursor als je over een plek in het diagram sleept waar je een nieuw modelelement mag toevoegen, heel wat water onder de brug door moet stromen. Zou het niet handig zijn niet meteen heel UML te verketteren, maar deze standaard te gebruiken waar mogelijk? Een use case blijft altijd een use case, een aardige manier om requirements vorm te geven. Dan hoeven we niet alles dat we modelleren opnieuw te definiëren, maar alleen in uitzonderingsgevallen veel werk maken van een DSL. Bijvoorbeeld voor het modelleren van de user interface. Dat scheelt een hoop tijd, en die hadden we nu juist heel hard nodig in onze projecten.

Waardering ★★★½☆

Auteur:	Steve Cook, Gareth Jones,Stuart Kent, Alan Cameron Wills
Titel:	*Domain-Specific Development with Visual Studio DSL Tools*
Uitg.:	Addison Wesley

Sander Hoogendoorn
www.sanderhoogendoorn.com