Jos Warmer

Toen UML in 1997 een standaard werd bestonden er bijna honderd verschillende modelleertalen. De ontwikkelaars van veel verschillende modelleertalen zijn bij elkaar gekomen en hebben gezamenlijk de UML gedefinieerd. Binnen korte tijd zijn de meeste andere modelleertalen verdwenen en stapte iedereen over naar UML

UML is dood, leve UML!

De afgelopen jaren is UML op steeds grotere schaal gebruikt. In de praktijk stuit men hierbij op steeds meer problemen. UML gaat er vanuit dat een systeem in één monolithisch model beschreven wordt. Weliswaar kan je een UML model via verschillende diagrammen bekijken, maar achter de diagrammen blijft het één model. De IT historie heeft al keer op keer bewezen dat monolithische oplossingen niet schaalbaar zijn, dus laten we eens kijken hoe het UML vergaat.

In de praktijk zijn multi-user access en versiebeheer van UML-modellen terugkerende problemen welke niet adequaat opgelost zijn. Ieder tool kent zijn eigen proprietary oplossing, maar geen van alle werkt in de praktijk goed genoeg.

Nadat UML gestandaardiseerd was, werden ideeën over Model Driven Development ontwikkeld, al dan niet in de MDA-variant van de OMG. Vanzelfsprekend ging men hier UML voor gebruiken. Omdat UML veel te groot en complex is om code uit te genereren gebruikt ieder tool zijn eigen UML-dialect. Deze zogenaamde profiles met bijbehorende stereotypes definiëren een tool-specifiek dialect van UML. De complexiteit van UML maakt het ontwikkelen van profiles en code-generatoren tot een tijdrovende activiteit. De resultaten zijn over het algemeen te star en te beperkt.

Ervaring met codegeneratie en het definiëren van UML profiles leert dat je met UML meestal wel ongeveer kunt modelleren wat je wilt, maar nooit precies. Voor code-generatie is deze precisie wel essentieel.

In de wereld van Model Driven Development en code-generatie worden de beperkingen van UML in brede kring dan ook steeds meer erkend. Maar hoe dan verder?

Alternatieven waren tot voor kort lastig te maken. Het definiëren van een specifieke modelleertaal geschikt voor code-generatie is nog wel te doen, maar het ontwikkelen van de tools, met name voor visuele modelleertalen, koste veel te veel tijd. Sinds kort is hier verandering in gekomen. Speciale workbenches voor het definiëren van domeinspecifieke modelleertalen zijn snel populair aan het worden. Deze workbenches genereren de volledige tooling, inclusief visuele editor, en integreren deze direct in de gebruikte IDE. Zo heb je de Microsoft DSL Tools die volledig integreren met Visual Studio en de GMF + EMF tools welke volledig integreren in de Eclipse-omgeving.

Zelf hiermee een eigen op maat gesneden modelleertaal maken is hiermee economisch haalbaar geworden. Het is in de meeste gevallen zelfs minder werk om een eigen taal te ontwikkelen met behulp van deze workbenches, dan een profile te ontwikkelen voor UML. Bovendien bieden deze domeinspecifieke talen de mogelijkheid om kleinere en flexibelere talen te ontwikkelen.

Voor het gebruik binnen Model Driven Development en grootschalige code-generatie blijken domeinspecifieke talen vele malen meer effectief te zijn dan UML. In deze wereld is de rol van UML dan ook langzamerhand uitgespeeld.

Is UML dan definitief dood? Geenszins! UML zal nog lange tijd de standaardtaal zijn voor menselijke communicatie. Willen we verder, dan zullen we afscheid moeten nemen van UML en zoals een bekend schrijver zegt: “Partir, c’est mourir un peu”. Zo gaat UML toch een klein beetje dood.

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