Verder kijken dan je neus lang is

Vroeger, héél vroeger was het nog mogelijk om IT-er te zijn en een overzicht te hebben over het hele vakgebied. Met de groei van de IT is dat tegenwoordig een bijna onmogelijke taak. We zien dan ook steeds meer specialisaties.

Eén van de nadelen van deze noodzakelijke opsplitsing van ons vakgebied is dat steeds meer IT-ers een groot deel van de verzamelde IT-kennis missen. Ikzelf ben actief in de wereld van taaldefinities, zowel voor algemene talen als UML en OCL, maar ook voor allerlei domeinspecifieke talen, al dan niet met een XML-syntax. Enige jaren geleden kwam een discussie op gang waarbij er voorgesteld werd om de business eindgebruiker voortaan niet meer

```
A + B
```

te laten schrijven. Nee, vanuit IT perspectief was het natuurlijk veel handiger om met XML te werken. Er werd voorgesteld om voortaan

```
<plus>
        <var name="A"/>
        <var name="B"/>
</plus>
```

te gaan schrijven. Dat was tenminste een structuur die veel beter te bewerken was door de achterliggende applicaties. Op mijn voorstel om de businessgebruiker gewoon A + B te laten schrijven en er zelf de XML-variant uit af te leiden om die intern te gebruiken werd gereageerd met: "Dat is veel te moeilijk." Het bleek dat betreffende IT-expert wel veel kennis had van XML, maar geen idee had wat een parser-generator was. Het omzetten van A + B naar XML is al tientallen jaren gesneden koek, maar de XML-specialist wist dat domweg niet.

Een gevolg van deze onwetendheid is dat er veel wielen telkens opnieuw worden uitgevonden. In plaats van te leren van bestaande kennis en evaring wordt een nieuw gebied, meestal voorzien van nieuwe naamgeving, te vaak gezien als iets dat geheel anders is dan al het voorgaande.

Ik ben al jaren bezig met modelleertalen. Meestal wordt een modelleertaal gezien als iets heel anders dan source-code. Als modellen echter volledig voor code-generatie gebruikt worden, in plaats van als ontwerpplaatje, dan lijken ze wel erg veel op source-code.

De grootste doorbraak in ons denken kwam dan ook toen we deze overeenkomst in het oog kregen. Wij behandelen een modelfile sindsdien op dezelfde wijze als een source-code file. Verbazingwekkend genoeg blijken hierdoor veel van de typische problemen met het maken en beheren van modellen, zoals multi-user problematiek en versiebeheer, als sneeuw voor de zon te verdwijnen.

Een collega van mij doet onderzoek naar manieren om modellen te transformeren naar andere modellen of naar code. Zij zoekt hiervoor bewust inspiratie in de wereld van compilerbouw. Hoewel niet alles *as-is* bruikbaar is, blijken veel concepten wel een prima basis te zijn om modeltransformaties te verbeteren.

Een ander mooi voorbeeld van hergebruik van bestaande kennis is LINQ. SQL is sterk in het werken met verzamelingen, het maken van selecties uit verzamelingen et cetera. Men heeft dit SQL-wiel genomen en de concepten ingepast in C# 3.0. Hierdoor wordt het gemak van het werken met verzamelingen ineens ook binnen C# toegankelijk. Een *select* kan op tabellen in een database gedaan worden, maar ook over willekeurige C# collecties. Iets nieuws uitgevonden? Nee dus, maar wel een prachtig voorbeeld van synthese tussen twee werelden.

Ik ben er dan ook een groot voorstander van om de rest van de mensheid te volgen. Zij hergebruiken het wiel al vijfduizend jaar. Laten we ook in de IT-wereld eens wat meer naar onze buren kijken om te zien of daar geen interessante ideeën voor het oprapen liggen.

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