Bericht uit Java

In mijn vorige column heb ik vier natuurwetten voor Java projecten geponeerd en beloofde ik terug te komen op de invulling van de architectenrol in Java projecten. Daar heb ik mij weer mooi in de nesten gewerkt.

Architecten en natuurwetten

Tja, dan toch maar een poging. Voor de volledigheid, hierbij de eerder genoemde natuurwetten, de kernproblemen van elk Java project:

- Als voor een project al duidelijk is wat de doelstellingen zijn, dan veranderen die doelstellingen toch wel.
- Projecten hebben nooit genoeg tijd, nooit genoeg mensen.
- Programmeurs worden stelselmatig aangetrokken door onzinnige programmeeractiviteiten. (Zwaartekracht?)
- Java is nog erg nieuw, projecten worden altijd bemand door mensen die nog niet bekend zijn met de te gebruiken techniek.

Ik hoor u denken: "Maar dit heeft toch niks met architectuur te maken? De kwaliteit van een architectuur heeft toch niks met projectdoelstellingen, met kwaliteit van programmeurs of met planning te maken?". En dan heeft u het mis!

Architectuur is niet alleen bepalend voor de performance, schaalbaarheid beheerbaarheid, hergebruik of andere technische kwaliteiten van software. Architectuur kan in grote mate bepalend zijn voor de (on-) mogelijkheden van planning, het toepassen van methodieken, het omgaan met programmeurs die nog een heleboel, al dan niet over Java, moeten leren. En laten we eerlijk zijn, er gaan heel veel meer projecten mis omdat zij hun planning niet halen dan omdat het hergebruik of de performance een beetje tegenvalt. Wat hebben we aan een ideale architectuur wanneer het project voortijdig afgeblazen wordt omdat er maar geen voortgang getoond kan worden?

Allereerst moet een architectuur het mogelijk maken om alle functionele aspecten van een systeem zo ongecompliceerd mogelijk te implementeren. Die functionele aspecten zijn de kern van de zaak, die moeten we kunnen implementeren zonder alles te weten van J2EE, XML parsing, webservices en OR-mappings. De functionele aspecten moeten dus met gewone Java objecten gemaakt kunnen worden, geen enterprise beans, geen SQL! Gewone Java objecten, dat kan iedere Java programmeur snappen, daarvoor is een simpele cursus genoeg.

Natuurlijk hebben we ook SQL (of JDBC) nodig en natuurlijk kunnen moderne systemen niet meer zonder XML. Natuurlijk levert J2EE een heleboel productiviteitswinst op. Maar dit soort technologieën zijn complex, moeilijk onder de knie te krijgen. Laten we er in ieder geval voor zorgen dat als we die technologieën gebruiken, ze ons niet weerhouden van de kern van de zaak: de business objecten en de functionaliteit die daarmee geïmplementeerd wordt. We moeten dus een manier zoeken om de business objecten in te bedden in een wereld van complexe technologieën zonder dat die technologieën onderdeel worden van de business objecten. Om dat te bereiken zijn er een paar hele eenvoudige en effectieven patterns: adapter-, proxy- en/of het observer-pattern. Hoe die te gebruiken? Lees Craig Larman, lees Martin Fowler (www.martinfowler.com), lees de *Gang of Four*.

Je zult er achter komen dat het implementeren van de functionaliteit (de business objecten) maar een heel klein deel van systeemontwikkeling is. De meeste tijd gaat zitten in persistentie, in integratie met andere systemen, in user interfaces, in transactiemanagement. Maar je weet in ieder geval dat je met de juiste functionaliteit bezig bent. De meeste architecturen – hoe technisch hoogstaand ook – zijn vooral erg goed in het verdoezelen van de functionaliteit. En we hebben allemaal wel eens meegemaakt waar dat toe leidt.

J. Meermans
is Java-kenner en te bereiken via
meermans@cibit.nl