Bericht uit Java

Ik heb eigenlijk nooit begrepen wat nou precies een IT-architect is. Ik weet dat er verschillende soorten IT-architecten bestaan. Je hebt de infrastructuur-architect, die houdt zich bezig met servers, routers, netwerken... Dat kan ik me nog wel voorstellen. Aan de andere kant heb je Informatie-architecten. Wat die doen is mij een raadsel. Informatie kun je niet bouwen, informatie is er, je kunt het hoogstens ordenen, sorteren, herschikken, opslaan of beschikbaar maken. In de Java wereld hebben we het vooral over de applicatie-architect of technisch architect. Dat zal wel iemand zijn die vertelt dat de servlets aan de voorkant zitten, de beans in het midden en de database aan de achterkant. Big deal!

De natuurwetten van Java

Vrij regelmatig komt er in de wereld van de architecten een schijnbaar onbelangrijke vraag weer bovendrijven: "In hoeverre lijkt een IT-architect op een bouw-architect?". En als vervolg daarop ook de meer wezenlijke kwestie: "Kunnen wij IT-ers iets leren van de manier waarop de wereld van de bouw georganiseerd is?". Ik denk dat wij IT-ers sowieso nog een heleboel te leren hebben en we zullen dus zeker ook wat kunnen leren van de bouw. Het bouwen van gebouwen als metafoor voor het bouwen van software gaat op één plek mank: software heeft geen "last" van natuurwetten, gebouwen wel. Met Java kan ik alle natuurwetten aan mijn laars lappen. Een gebouw van tienduizend verdiepingen hoog? Zo gedaan:

```
Gebouw g = new Gebouw();
for(int i = 0; i < 10000;
i++) {
  g.voegVerdiepingToe(new
Verdieping(i));
```

OK, het voorbeeld is flauw. De consequenties zijn er niet minder om. In termen van natuurwetten kan ik duizenden kilo's software aan een heel dun draadje laten hangen. Kleine objectjes kunnen hele grote applicaties dragen (of laten vallen). We kunnen alles maken wat we willen op elke manier die we kunnen bedenken. We krijgen het (uiteindelijk) altijd wel voor elkaar. Helaas zijn de meeste van die manieren op een of andere manier niet gewenst. We moeten keuzes maken. Voor een technisch architect in de bouw is dat makkelijk. Zodra zijn ontwerp ter discussie gesteld wordt kan hij verwijzen naar de natuurwetten: "Als je het anders doet stort het in, waait het weg, valt het om". Voor een technisch architect in de IT is dat veel moeilijker. Ik zie architecten altijd terugvallen op dezelfde zouteloze argumenten: hergebruik, flexibiliteit en performance. Dat zijn natuurlijk heel belangrijke doelen. Ze zijn alleen niet voldoende om een keuze te kunnen maken tussen al die verschillende manieren om software te maken. Deze drie argumenten zullen altijd leiden tot eeuwige wellesnietes situaties. Het zijn geen natuurwetten. Het is juist de IT-architect die natuurwetten nodig heeft. Ik poneer hierbij vier Java-natuurwetten:

- 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.

Dit zijn de natuurwetten waarop een architect zijn ontwerpen zou moeten richten, dit zijn de problemen waar een project mee zit. Elke architectuur die zich niet met deze natuurwetten bezig houdt, heeft geen recht van bestaan en zal in de praktijk niet volgehouden worden. Hoe het dan wel moet? Wordt vervolgd...

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