Bericht uit Java

Tijdens een cursus had ik het over design patterns. Als voorbeeld liet ik het composite pattern zien. Ik vertelde dat een vliegtuigbouwer zijn vliegtuig als composite pattern zou kunnen modelleren. Je zou dan tegen een vliegtuig zeggen 'bestel jezelf' en het vliegtuig geeft diezelfde vraag door naar al zijn onderdelen. Elk samengesteld onderdeel vraagt zijn onderdelen zichzelf te bestellen en elk enkelvoudig onderdeel zet zichzelf op de bestellijst. Alle onderdelen in één keer besteld! Mooier kunnen we het niet maken.

Animistische programmeurs gezocht

Dit keer zei iemand in de zaal: "Ik zie niet goed hoe je dit in de praktijk doet". Ik stond (hopelijk) vriendelijk maar verbaasd te kijken. Ik kan me werkelijk niet anders voorstellen dan dat je een vliegtuigbouwbestelsysteem op deze manier modelleert. "Onderdelen die zichzelf bestellen, dat kán toch helemaal niet!" Ah! Hij snapt objectoriëntatie (nog) niet, concludeerde ik.

Je kunt mij naïef noemen, maar objectoriëntatie is wat mij betreft nog steeds de enige zinvolle manier om software te bouwen. De wereld wil er maar niet aan en heeft de OO-theorieën de laatste jaren alleen maar verwaterd met componenten en services. OO heeft zijn kans gehad en heeft niet opgeleverd wat het op moest leveren. OO zou mislukt zijn. En dus zijn we weer terug bij af. We maken weer componenten en services en over die componenten en services heen implementeren we weer (business-) processen. De traditionele scheiding tussen data en proces is nog steeds onwrikbaar. Waarom zien mensen toch niet in dat je dan één probleem op twee plekken oplost? Al die waardeloze, onbeheersbare, niet te bouwen, niet te plannen software zou toch boekdelen moeten spreken?

Mijn stelling is dat we objectoriëntatie nooit op grote schaal écht geprobeerd hebben. We hebben het zelfs nooit helemaal goed uitgelegd. OO-docenten zeiden: "Objectoriëntatie is een hele logische, natuurlijke manier om de wereld te modelleren". Met OO kun je de echte wereld helemaal niet beter modelleren dan met datamodellen, procesmodellen, agentmodellen of welke andere smaak dan ook! Objectoriëntatie geeft je de mogelijkheid om die modellen te manipuleren, te veranderen, te kneden. Alle andere modelleervormen staan dat nauwelijks toe. Met procesmodellen kun je de wereld modelleren maar je hebt geen idee wat er gebeurt als je het proces aanpast. Datzelfde geldt voor datamodellen. Met objecten manipuleer je, op één plek, de kern van het probleem. Je ziet direct wat de consequenties zijn. Dit is de enige manier om werkelijk complexe systemen te maken.

Waarom wil dat er toch niet in? Waarom moet je in élk project, bij élke cursus, aan élke projectmanager uitleggen dat het ook anders kan. Ik ben al jaren op zoek naar het antwoord op die vraag. Ik heb (relationele) databases de schuld gegeven, ik heb onszelf de schuld gegeven ("wij hebben de business uitgelegd dat zij in processen moeten denken voordat ze aan ICT-projecten beginnen"). Ik heb managers de schuld gegeven (managers willen systeemontwikkeling als fabriekswerk zien, objectoriëntatie is te creatief daarvoor). Ik begin nu de maatschappij en onze geschiedenis de schuld te geven. Ik word zo paranoïde dat ik me zelfs kan voorstellen dat ons geloof schuldig is. Wij Westerse ontwikkelaars komen voort uit een monotheïstische geloofswereld. Er is één god die alles overziet. Wij zijn in zijn evenbeeld gemaakt en ook wij overzien zo onze systemen. Onze godsdienst. Zouden mensen met een animistische achtergrond ook zo'n moeite hebben met OO? Als je gelooft dat in elk dier en in elke boom een god schuilt, dan moet je toch geen moeite hebben met een vliegtuigonderdeel dat zichzelf bestelt? Alle animisten aan de Java!

Daan Kalmeijer is docent consultant bij CIBIT-adviseurs | opleiders (e-mail: daan@cibit.nl).