Bericht uit Java

Voor programmeurs, designers en softwarearchitecten is het een goed gebruik om domeinobjecten te maken. Domeinlogica wordt geïsoleerd in deze objecten. Hier wordt uitgeprogrammeerd hoe verzekeringspolissen verkocht worden, hoe patiënten ontslagen moeten worden uit het ziekenhuis en hoe orders geaccordeerd worden. Het zijn objecten waarin de kern van de applicatie uitgeprogrammeerd wordt.

Uiterlijkheden

Met allerlei designpatterns en frameworks hoort een programmeur een scheiding aan te brengen tussen deze domeinlogica en alle benodigde technologie daaromheen. Domeinobjecten zouden niets moeten weten van SQL, van logging of van front-end technologie als Servlets, STRUTS of JSF. Sommige ontwikkelaars zijn daar goed in, andere wat minder.

De reden om deze domeinobjecten te isoleren van de omliggende technologie is eenvoudig. Technologie raakt verouderd; over twee jaar hebben we een nieuwe Oracle-versie, komend jaar moeten we van onze gewone webinterface naar een AJAX-interface en ons loggingmechanisme moet vervangen worden vanwege compliancy-issues. Terwijl al die technologie veroudert en vervangen wordt, blijft domeinlogica lange tijd houdbaar. Daar waar domeinlogica wél zou veranderen, leidt die verandering tot veranderingen in de technologielagen eromheen. Een systeemverandering naar aanleiding van een verandering in 'de business' is beter te verkopen dan een verbouwing naar aanleiding van een technische verandering. Het lijkt duidelijk: domeinlogica en technologie moet gescheiden worden.

Er is één zeer populair en succesvol soort applicatie waar domeinlogica sterk geïntegreerd wordt met de user interface: Games. Waar wij, bouwers van nette business-applicaties, er nooit over zouden piekeren om domeinobjecten en user interface technologie te integreren, zijn gamebouwers niet anders gewend. Hoe games van binnen gebouwd worden is niet makkelijk te achterhalen. Eén ding is in ieder geval zeker: de 'voorkant' van een game is het meest complexe en in het oog springende onderdeel van een game. Wat we óók weten van gamebouw, is dat er heel veel meer geld mee verdiend wordt dan met onze business software. Verder weten we dat er in de gamebouw werkelijk aan hergebruik gedaan wordt. Wij business software-bouwers zeggen wel dat we zouden willen hergebruiken inde praktijk komt het er niet van.

Zou daar de oplossing liggen voor onze hergebruikproblemen? Zouden we maar niet willen hergebruiken omdat onze te hergebruiken objecten geen 'smoel' hebben? Omdat we ze niet kunnen herkennen? Waar een gamebouwer een gebouw-object hergebruikt vanuit het ene spel naar het andere spel, kan hij of zij dat gebouw ook werkelijk zien. Hij kan zien dat het een mooi gebouw is, met leuke details, dat er met veel aandacht aan geprogrammeerd is. Het gebouw ziet er goed uit en nodigt dus uit tot hergebruik. Hoe anders is het bij onze domeinobjecten. Dat zijn alleen maar saaie stukken Java-code. Misschien ziet de potentiële hergebruiker wel dat de code netjes opgemaakt is, maar op geen enkele manier straalt van de code uit dat er met oog voor detail en kwaliteit gewerkt is. De code is niet te vertrouwen. Met (user interface) JavaBeans hebben we een tijdje geprobeerd om op basis van de user interface hergebruik te organiseren. JavaBeans (knoppen, stukken scherm, en andere widgets) werden gebouwd en in een klik-en-sleep manier hergebruikt. Dat is natuurlijk niet genoeg. We moeten verder gaan. Ons klantobject moet er niet uitzien als een scherm met invoervelden, maar als Lara Croft. Ik durf te wedden dat we dan gaan hergebruiken. Hergebruik is misschien wel belangrijker dan ontkoppeling.

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