Bericht uit 

Hoe bedenk je een pattern of een anti-pattern? De echte goeie patterns, waaronder de Gang of Four patterns, zijn zo elegant en mooi dat je je afvraagt hoe iemand het heeft kunnen bedenken. Aan de andere kant zijn ze meestal zo simpel dat je je afvraagt hoe je ze zelf over het hoofd hebt kunnen zien.

Anemie

Soms denk ik dat ik er eentje gevonden heb. Een heus pattern of anti-pattern. Is het er ook echt een? Mag ik een heus pattern op mijn naam schrijven? Ja én nee. Als ik dat pattern goed kan gebruiken of als ik anderen kan helpen om mooie systemen te bouwen, dan mag ik best wel zeggen dat ik een pattern ontdekt heb. Maar ja, een pattern is pas een echt pattern als iedereen het pattern kent. Er luisteren gewoon niet genoeg mensen naar mij om mijn patterns de moeite waard te maken. Ach, ik kan er wel mee leven, zo briljant waren die patterns van mij natuurlijk ook weer niet.

De mooiste en belangrijkste patterns zijn natuurlijk al lang bedacht. Het begon met MVC en de Gang of Four en daarna werden alle opvolgende patterns eigenlijk alleen maar minder belangrijk. Er zullen nog wel wat aardige patterns bedacht worden, maar het zal het niet halen bij de eerste. Met anti-patterns gaat dat anders. Een tijdje nadat patterns bedacht waren kwam iemand (wie?) met het idee van het anti-pattern. Als het gaat om anti-patterns, denk ik dat de mooiste nog bedacht moeten worden. Ik denk dat we een heleboel domme dingen doen en maar blijven herhalen. Die domme dingen willen we maar niet onder ogen zien. We moeten het de komende tijd van de anti-patterns hebben.

Een goed voorbeeld van zo'n anti-pattern komt (weer) van Martin Fowler: "anemic domain models". Martin Fowler is tenslotte iemand waar naar geluisterd wordt. Van "anemic domain models" hebben we de afgelopen jaren heel veel last gehad maar we hebben het maar niet willen onderkennen. Heel veel mensen hebben geprobeerd om mensen te wijzen op "anemic domain models" maar dat was tevergeefs. Ik heb er zelfs al meerdere columns aan besteed. Het had geen zin, want het was nog geen anti-pattern.

J2EE EntityBeans zijn een typisch voorbeeld van "anemic domain models". Je maakt Java objecten die niks meer zijn dan domme objecten met alleen getters en setters. Die getters en setters worden bovendien gebruikt door het framework waar de objecten in leven (bijvoorbeeld om persistentie te implementeren) waardoor je geen enkele echte business logica kunt implementeren in deze domeinobjecten. Je had net zo goed in het stenen tijdperk kunnen blijven zitten en COBOL kunnen blijven programmeren.

Tegenover deze "anemic domain models" hoort een ander anti-pattern, maar dat wil Martin Fowler maar niet benoemen. Aan de andere kant van de domme domein objecten horen de agents, controllers en session beans. Die zijn net zo kwalijk als de domme domein objecten. Maar ja, agents, controllers en session beans staan voor het implementeren van processen en we zijn er nog niet rijp voor om te erkennen dat (business) processen ook een anti-pattern zijn in de softwarebouw. In de echte wereld, de wereld buiten de softwarebouw, hebben we al lang door dat business processen erg ongrijpbaar en eindeloos flexibel zijn en dat ze, als het even kan, omzeild worden, ook al willen managers dat meestal niet toegeven. Zodra we software gaan maken of gebruiken denken we ineens dat processen heilig zijn. Zoals ik al zei, de mooiste anti-patterns moeten nog gaan komen.

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