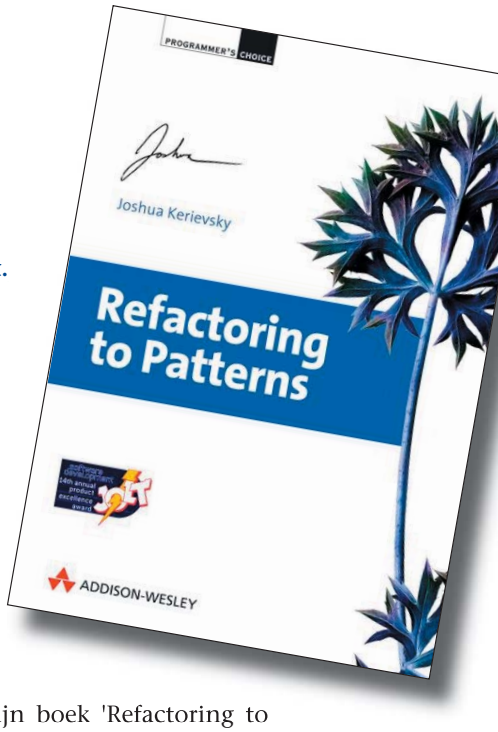
bericht



"Programmeer jij nog?" vraagt een verbaasde cursist me wanneer ik op mijn laptop Visual Studio opstart. Ik knik bevestigend. "Goh, dat hadden we niet verwacht. Iemand in jouw functie die nog steeds programmeert." Nog immer wordt programmeren beschouwd als een geestdodende en inferieure bezigheid waaraan je je hooguit aan het begin van je carrière bezoldigt, met als enig doel zo snel mogelijk door te stomen naar hogere kasten van onze samenleving, zoals die van informatie-analisten of architecten.

De missing link

Het is natuurlijk niet voor iedereen weggelegd, maar uiteindelijk kun je als programmeur zelfs opklimmen tot projectmanager, alhoewel je dan natuurlijk wel weer heel wat jaren verder bent.

Inderdaad. Ik programmeer nog. Misschien zelfs wel met meer plezier. Laat ik me hier beperken tot twee aspecten die mijn leven als ontwikkelaar een stuk aangenamer hebben gemaakt: design patterns en refactoring – en natuurlijk niet de dagelijkse verzuchtingen van versiebeheer, unit testing of daily builds.

Op dit moment werk ik samen met een collega met veel plezier aan een gesophisticeerd stuk gereedschap dat de Tobago MDA Generator heet en dat in staat is een model in te lezen (vormgegeven in UML, geëxporteerd in XMI) en vervolgens aan de hand van voorgedefinieerde patronen en templates allerlei interessante dingen kan genereren, zoals code voor uiteenlopende situaties, talen en frameworks, maar ook Excel spreadsheets met schattingen, en zelfs projectplannen in Word.

Telkens wanneer we aan ons gereedschap werken, bedenken we nieuwe functionaliteit. Afgezien van het feit dat het bedenken van nieuwe functionaliteit gaandeweg een project bij de gemiddelde projectmanager oorzaak nummer één is van grijze haren, burn-outs en vervroegde pensionering, heeft dit ook buitengewoon interessante technische consequenties. Langzaam, beetje bij beetje verschuift de structuur van de code onder de motorkap. Een methode wordt verplaatst, nieuwe eigenschappen en methoden worden toegevoegd aan bestaande klassen, constructors veranderen, factory's worden toegevoegd. En dan, plots, van het ene moment op het andere, zeker als je met meerdere mensen aan de code werkt, concludeer je dat de structuur van de code onduidelijk is geworden, of erger nog, dat je je eigen code niet meer begrijpt. Dat moment is misschien wel een goed moment voor het refactoren van je code.

In 2000 schreef Martin Fowler zijn standaardwerk 'Refactoring'. Een boek vol minipatronen waarmee je de kwaliteit van je code kunt vergroten, zonder er nieuwe functionaliteit aan toe te voegen. Ondanks de onmiskenbare impact van dit boek – zie Eclipse, zie Visual Studio – zijn refactorings weliswaar buitengewoon nuttige, maar vooral ook buitengewoon kleine techniekjes. Mijn te ontrafelen code is simpelweg complexer. Het wachten was derhalve op literatuur waarin aaneenschakelingen van refactorings worden aangewend om grotere problemen aan te pakken.

Vanuit die optiek schreef Joshua Kerievsky zijn boek 'Refactoring to Patterns', dat is verschenen in de Signature Book reeks van Addison Wesley. In dit boek laat de auteur, een vriend van Fowler, zien hoe het gecombineerde gebruik van design patterns en refactorings de kwaliteit van code verbetert. Het materiaal wordt aangeboden in 27 nieuwe, grote refactorings, gekoppeld aan bekende patronen zoals Adapter, Factory Method en Singleton. Een goed voorbeeld is Move Embellishment to Decorator, dat toont hoe code die later aan een klasse is toegevoegd en de kernfunctionaliteit van zo'n klasse vertroebeld kan worden aangepakt. Kerievsky laat met zijn refactorings overigens niet alleen zien hoe je code naar patronen toe beweegt, maar in een aantal gevallen ook juist hoe je er weer vanaf komt. Dit boek is de missing link tussen design patterns en refactoring. Dat is wat je noemt een aangenaam kerstcadeau!

Sander Hoogendoorn (www.sanderhoogendoorn.com)

Waardering ★★★½☆

Joshua Kerievsky
Refactoring to Patterns
Addison Wesley