Sander Hoogendoorn

Sander Hoogendoorn
www.sanderhoogendoorn.com

Soms wordt ik door bedrijven of hogescholen gevraagd een mooi verhaal te komen vertellen. Vaak een goede gelegenheid om eens te sparren met developers. Zo sprak ik recent in Arnhem bij Avisi – een klein, zeer enthousiast Java-bedrijf. En terwijl ik vertelde over agile, Smart, smart use cases, patronen en het genereren van code vanuit UML viel me weer op hoe verschillend Java en .Net toch zijn.

Goeroe Kent

Zonder religieus te worden en te roepen dat de een beter is dan de ander, zijn, afgezien van de parallellen in taal en de vraag wie nu wat van wie afkijkt, beide wereld vooral anders. Bij Avisi vliegen de frameworks je om de oren en kunnen de developers moeiteloos alle nieuwe JSR's opsommen – zo leer ik dat JSR-170 content management definieert. Men zich stelt zich vooral de vraag met welke frameworks en tools een nieuw project kan worden aangegaan. Daarbij worden nieuwkomers als Seam en Selenium moeiteloos gecombineerd met gouwe ouwe als Spring en Hibernate.

In de wereld van .Net spelen open source frameworks geen rol. Zelfs Spring en Hibernate krijgen nauwelijks vaste voet aan de grond. Dit wordt vooral veroorzaakt door het enorme tempo waarmee de fabriek van Microsoft nieuwe frameworks produceert. Zelfs als deze nog lang niet af zijn, zoals het Entity Framework, wachten developers liever op de roerselen van Microsoft dan zich te verlaten op open source. Niet eens verwonderlijk gegeven het grote aantal developers dat Microsoft op deze klussen zet.

Het heeft er bovendien alle schijn van dat Java developers gemiddeld beter belezen zijn. Iedereen weet wie Martin Fowler, Eric Evans, en Scott Ambler zijn of kent ze persoonlijk.

Prompt na deze overpeinzingen op de terugweg van een inmiddels aardedonker Arnhems bedrijventerrein, valt thuisgekomen een mail van mijn belezen oud-collega Ron Kersic in mijn inbox. "Er mag wel weer eens een boek review van jou in JavaMagazine," daagt Ron uit. Een link toont me het nieuwste boek van Kent Beck dat *Implementation Patterns* heet. Intrigerende titel.

Nu ben ik altijd wat sceptisch tegenover Kent Beck. Dat komt door Alistair Cockburn. Ooit tijdens een autorit, zwaar onder invloed van de jetlag, verhaalde Alistair van een presentatie van Beck waarin hij uitlegde sommige code nog niet te refactoren "because the code doesn't know yet where it wants to go." Waarop Alistair, vermoeid uit het zijraam turend peinsde: "his feet just left the ground." En inderdaad, *Implementation Patterns* heeft iets zweverigs.

Het begin is weer briljant, als onze esoterische guru strooit met oneliners. Memorabel is "I have seen too much ugly code make too much money to believe that quality of code is either necessary or sufficient for commercial success or widespread use." Soms is Kent scherp. Maar over de hele linie? Met *Implementation Patterns* schrijft Beck geen stijlgids voor Java, en de patronen in het boek zijn kleiner dan design patterns. Ze zijn ruwweg verdeeld in categorieën als class, state, methods en collections en beschrijven beslissingen die developers dagelijks vele malen maken, zoals de keuze tussen een abstracte klasse en een interface, of hoe om te gaan met method overloading. Hierbij gaat Beck uit van het terechte idee dat code zelfdocumenterend moet zijn. Eigenlijk is het boek een verzameling losse essays waarin Kent Beck uitlegt hoe hij graag programmeert. Vaak zijn zijn ideeën bijzonder leesbaar, soms wat basaal en af en toe niet concreet genoeg. Daardoor schommelt het boek – zoals vaker bij Kent Beck – tussen "leuk om te lezen" en "goed dat iemand dit eens heeft opgeschreven." Op zijn beste moment doet *Implementation Patterns* denken aan *Framework Design Guidelines* van Krzysztof Cwalina en Brad Abrams – ook een tip van Ron – maar mist het de diepgang en annotaties van dat standaardwerk.

Waardering ★★★½☆

Auteur: Kent Beck
Titel: Implementation Patterns
Uitg.: Addison Wesley