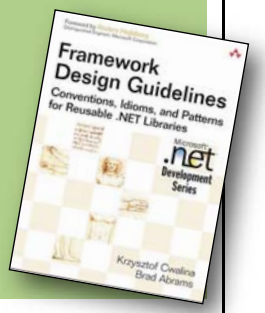
Sander Hoogendoorn

Sander Hoogendoorn
www.sanderhoogendoorn.com

Ik schreef mijn eerste professionele software zo'n achttien jaar geleden. Op dat moment werkte ik aan een serie rekenprogramma's voor Rijkswaterstaat. Op mijn eerste portable computer, toen ook al een IBM. Niet echt draagbaar en met een klein monochroom scherm. Groene letters op een zwart scherm. De goede oude tijd.



Verplichte kost

Er is meer dat in bijna twintig niet is veranderd. Voor Rijkswaterstaat programmeerde ik in Turbo Pascal, tegenwoordig in C#. Het leukste van het schrijven van deze applicaties voor Rijkswaterstaat was voor mij dat ik, door wat tijd te investeren in het opzetten van een klein framework, veel tijd bespaarde bij het schrijven van de programma's. De eerste van de veertien schreef ik in een slordige veertig uur, de laatste in slechts twee uur. Van de opbrengsten kocht ik mijn eerste eigen auto.

Eigenlijk doe ik al twintig jaar hetzelfde. Het liefst schrijf ik frameworks, en inmiddels ook codegeneratoren, die het ontwikkelen van applicaties vergemakkelijken en versnellen. Het pure plezier in mijn vak en veel vakgenoten met mij haal ik uit het ontwikkelen van steeds slimmere code. Je start met een lastige uitdaging: wat is het beste type voor een parameter van een methode die generieke collecties, zoals KeyedCollection, aan moet kunnen, maar daarnaast ook andere types? De eerste oplossing schiet je natuurlijk net te binnen als je s ochtends onder de douche staat of wanneer je het voetbalteam van je zoontje traint. 's Nachts, als het huis stil is, probeer je het uit. Het werkt! Toch blijft het knagen. Binnen de kortste keren heb je diverse mogelijkheden uitgeprobeerd, IList, ICollection<T> of toch maar gewoon object[]. En allemaal voelen ze net niet goed genoeg aan. Programmeren is pure intuïtie.

Op zo'n moment wil je wel eens weten waar de ontwikkelaars van het .NET framework bij Microsoft zelf voor zouden kiezen. Tot nu toe waren we daarvoor aangewezen op fijne tools als Reflector van Lutz Roeder, waarmee je de code van het .NET framework kunt inzien. Verhelderend! Maar de code laat niet zien waarom bepaalde keuzes zijn gemaakt. Welke afwegingen hadden de ontwikkelaars?

Voor die mensen die zich dat soms afvragen, is het ultieme boek verschenen. *Framework Design Guidelines* van Krysztof Cwalina en Brad Abrams neemt de lezer mee langs allerhande vraagstukken rond het ontwikkelen van frameworks en applicaties. Geen abstracte patronen, maar nitty-gritty details. Wanneer wordt een value object beter als een struct en wanneer beter als een class opgezet? Welk type laat ik een methode retourneren als ik een lijst van objecten wil teruggeven? Gebruik ik functionele foutmeldingen of gooi ik toch liever exceptions? Wanneer kiezen de makers van het .NET Framework eigenlijk voor een factory?

Framework Design Guidelines is een heerlijk boek, waar de do's, dont's en considerations letterlijk van de pagina's spatten. Wat het boek nog waardevoller maakt zijn de annotaties van ruim een dozijn toparchitecten van Microsoft, zoals Jeffrey Richter en Anders Hejlsberg, die hun ideeën, maar ook hun voortschrijdend inzicht geven over de totstandkoming van het .NET framework. Ik kan me voorstellen dat deze annotaties in eerste instantie als review-commentaar bij het manuscript zijn neergepend, maar zo waardevol bleken dat ze voor de lezer zijn behouden.

Voor iedereen die frameworks ontwikkelt, is *Framework Design Guidelines* simpelweg verplichte kost. Maar ook voor iedereen die het alleen al een sport vindt om steeds betere en mooiere code te schrijven beveel ik het boek van harte aan. Genoeg gepraat nu. Ik ga snel wat code refactoren.

Waardering ★★★★☆

Auteur: Krysztof Cwalina, Brad Abrams
Titel: *Framework Design Guidelines. Conventions, Idioms, and Patterns for Reusable .NET Libraries*
Uitg.: Addison Wesley