.net column

# Documentatie

**Het schrijven van een programma bestaat uit verschillende werkzaamheden die niet altijd allemaal even enthousiast worden uitgevoerd. De grote liefde van de meeste programmeurs gaat uit naar het typen van de code en dat is niet raar, want dat is meestal waar de programmeur mee begon en voor gevallen is. Door opleiding, ervaring en opgelegde regels wordt de programmeur geconfronteerd met andere, aanvullende bezigheden, die misschien wel nuttig zijn, maar lang niet zo leuk.**

Eén van die bezigheden is het maken van documentatie. Naast een handleiding voor eindgebruikers, een handleiding voor beheerders, rapporten voor managers of testhandleidingen, moet ook de code van begeleidende tekst worden voorzien zodat de programmeur en zijn collega's, nu en in de toekomst, de code kunnen begrijpen en zo kunnen onderhouden. Veel programmeurs vinden het schrijven van deze documenten, op zijn zachts gezegd, niet leuk.

Waarom is dat niet leuk? Ik denk dat het komt doordat het schrijven van die documentatie voor de programmeur een herhaling van zetten is. Eerder is hij, al coderend, tot een oplossing gekomen en nu moet hij die gedachtegang opnieuw langs om de overwegingen op te nemen in zijn documentatie. Het maakt daarbij niet uit of hij op voorhand al het een en ander had genoteerd; die notities zullen bijgewerkt moeten worden.

Hoe doet een grote club als Microsoft dit? Het .NET-Framework bestaat uit onvoorstelbaar veel regels code die niet alleen door Microsoft-programmeurs onderhouden moeten worden maar ook door henzelf en ons op de juiste manier moeten worden gebruikt. Een, in mijn ogen, cruciale verantwoordelijkheid ligt bij de naamgeving van namespaces, classes, methods, parameters en properties. Het is door de enorme omvang van het .NET-Framework niet meer mogelijk dat een programmeur eerst de volledige documentatie leest en dan met alle kennis aan de slag kan. In de praktijk zal een programmeur meestal een grof idee hebben hoe de class zal heten en typt dan die naam in zijn code of in de help in en kijkt dan wat de Intellisense of de MSDN oplevert. Natuurlijk kan de programmeur het fout hebben en in de verkeerde hoek aan het zoeken zijn; voor wie was het direct logisch dat de Process class in de namespace System.Diagnostics zit?

De leesbaarheid van code is niet altijd optimaal. Dit ontstaat vaak doordat er een kloof is tussen het functionele probleem dat moet worden opgelost; als de klant een spaarkaart heeft worden bonuspunten bijgeschreven en de technische oplossing: `if(k.s == 2) b++;` In een dergelijk geval kan commentaar een oplossing bieden maar daar moet wel nagedacht worden over de leesbaarheid van het commentaar. "Als k.s twee is dan wordt b met één verhoogd" is nutteloos. Iedere programmeur die deze code leest weet dat al. Nuttiger is: "als de klant een spaarkaart heeft worden bonuspunten bijgeschreven". Maar nog beter is het wanneer de code zelf leesbaar is: `if(klant.Status == Status.SpaarkaartHouder) bonuspunten++;`

Naast de beter leesbare code is de kans op logische fouten door de programmeur ook veel kleiner. Mijn paradigma voor ontwerp- en programmeerwerk is dan ook: "Als je het geen goede naam kunt geven, maak het dan niet." Als je de naam niet kunt bedenken, weet je ook niet wat het doet. Als je niet weet wat het doet, hoe kun je het dan maken? Als je het toch maakt, wie kan het dan gebruiken? Misschien dat we door pragmatisch om te gaan met documentatie die eerste verliefdheid kunnen vasthouden.

*Erno de Weerd – Info Support. Over deze column kan verder worden gediscussieerd op http://blogs.infosupport.com/ernow.*