Bericht uit Java

Documenteren?

Er ging ooit het verhaal rond dat er bij Microsoft geen documentatie geschreven werd. Geen enkel spoor van documentatie, nooit! Het zou zelfs de officiële stelling van Microsoft zijn dat documentatie toch niet in de pas loopt met de werkelijke code en dat er daarom maar beter van documentatie afgezien kon worden. Ik heb altijd dankbaar gebruik gemaakt van dit verhaal, vooral bij UML-cursussen. Het is ook een fijne stelling om aan cursisten voor te leggen, er komt altijd de voorspelbare reactie "Ah! En daarom zijn die Microsoft systemen zo ...!". Aan de andere kant heb ik zelf de "er is geen documentatie, en die komt er ook niet" stelling altijd wel mooi gevonden, lekker tegendraads en nogal on-Microsoft. Een beetje XP-ig zelfs: niks bouwen of schrijven wat je misschien later nodig zou kunnen gaan hebben, alleen bouwen wat je nú nodig hebt.

Enkele dagen terug hadden we, een aantal collega's onderling, het over deze 'Microsoft-stelling': "Code is de enige correcte documentatie, er is geen andere documentatie dan de code!", riep een collega om ons te prikkelen. Het werd een ingewikkelde discussie en we kwamen met z'n allen niet echt op één goed gedefinieerde conclusie uit. We hebben allemaal wel meegemaakt hoe documentatie 'achter kan lopen' of zelfs volledig incorrect kan zijn. De schade die dat brengt in de beheersfase van systemen kan zeer groot zijn en zelfs het systeem te gronde brengen. Documentatie geeft een schijnzekerheid, managers teken aan "de documentatie is op orde" en gaan met een zucht achterover zitten. Hoera, het systeem is goed want het is gedocumenteerd.

Wat in ieder geval belangrijk is, is dat er weinig documentatie nodig zou moeten zijn. Meer documentatie is niet gelijk aan betere documentatie. In ieder geval zou het systeem zelf zo gemaakt moeten zijn dat er nauwelijks documentatie nodig is. "Ja, maar systemen worden ingewikkeld en ingewikkelde systemen hebben véél documentatie nodig". Nee dus, daar geloof ik niet in. We maken op een slordige manier systemen en slordige systemen hebben veel documentatie nodig. Daar geloof ik wél in! Onze systemen kunnen we niet zomaar minder complex maken maar we kunnen wel minder slordig gaan bouwen. Wat zijn slordige systemen? Dat zijn systemen waarbij voor elk probleem een nieuwe oplossing gezocht wordt, in nieuwe talen, op nieuwe platformen. Systemen waarbij problemen op meerdere plekken tegelijk (of half) opgelost worden of systemen waarbij te veel onderdelen van elkaar afhankelijk zijn. Een goed systeem heeft minder documentatie nodig. Zouden er systemen zijn die géén documentatie nodig hebben?

Documentatie zou je zoveel mogelijk moeten genereren. JavaDoc is een voorbeeld van gegenereerde documentatie. Maar, dan moet je wel commentaar toevoegen aan je Java code. Dat doet niet iedereen en bovendien is dat commentaar vaak verouderd, foutief gekopieerd of anderszins onbruikbaar. Je zou een JavaDoc-generator willen hebben die ook werkelijk je code induikt, die op zoek gaat naar relaties tussen objecten, naar structuren in het systeem en naar de algoritmen die je geprogrammeerd hebt. Het enige wat je toe zou moeten voegen is de reden waarom je het zo geprogrammeerd hebt. Pas dán is documentatie maximaal gegenereerd.

MDA (Model Driven Architecture) zorgt voor een andere kijk op documentatie. Het ideaal van MDA is dat je een (UML) model van de business logica van je systeem bouwt en dan de rest genereert. MDA zegt dat de werkelijke waarde van een systeem in het model ligt en niet in de (platformspecifieke) code. Vandaag genereer ik J2EE code vanuit mijn model en morgen .NET code. De code is waardeloos geworden. Waarom zou ik nog documenteren als ik mijn model heb? Alle kennis zit toch in het model?

Als we écht verwachten dat we over een paar jaar alle code kunnen genereren, dan zouden we nu toch al iets moeten kunnen bouwen wat op basis van code alle nodige documentatie genereert? Waar wachten we op?

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