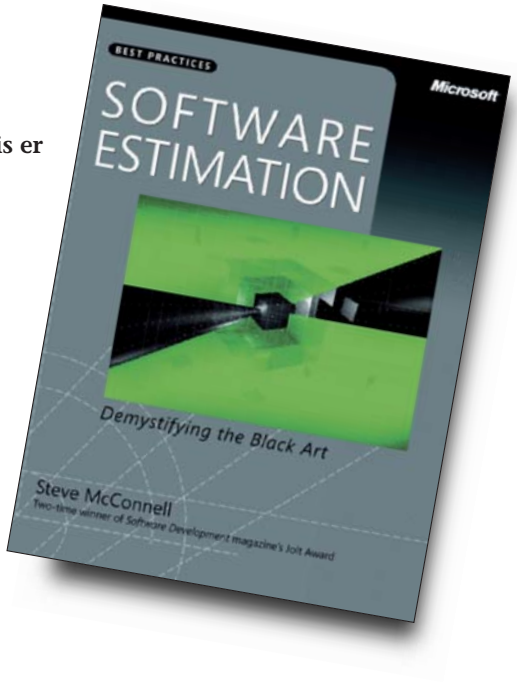
Het schatten van software-ontwikkelprojecten is geen sinecure. Gek genoeg vond ik dat vijftien jaar geleden ook al, en is er in die tijd nauwelijks iets veranderd. Ooit, een aantal werkgevers geleden bedacht ik een uitermate pragmatische techniek die opvallend genoeg best kekke resultaten opleverde.

Je telt het aantal pagina's van het functionele document dat je in je handen gedrukt krijgt door de desbetreffende sales manager, en je vermenigvuldigt dit met het aantal tabellen in het onvermijdelijke datamodel.



T-shirt sizing

Het resultaat geeft een redelijke indruk van het aantal uren dat nodig was om de applicatie te bouwen. Tot zover de cursus schatten in vijf minuten. Natuurlijk moet je tegenwoordig veel serieuzer de wereld in kijken wanneer je schatting maakt. Mooi woord voor galgje trouwens: functiepuntanalyse. Helaas is het nog steeds schering en inslag dat projectmanagers krampachtig blijven proberen de bij de start van het project gemaakte schatting waar te maken, zelfs als de requirements van het project verdubbelen of de productiviteit nog niet half zo hoog blijkt als oorspronkelijk uit de duim gezogen was.

Tot zover de zwarte kunst van het maken van schattingen. Dat bedacht ook Steve McConnell, auteur van het overbekende boek Code Complete. McConnell buigt zich in zijn nieuwe boek *Software Estimation. Demystifying the Black Art* over het fenomeen schatten. Omdat ik al een tijd rondloop met het idee een boek te schrijven over schatten met smart use-cases, begon ik geïnteresseerd aan het boek. Het bleek een heel aardig boek. Of eigenlijk, het is een goede samenvatting van wat er in de afgelopen twintig jaar over schatting is gepubliceerd door de grote namen in dit niche-vakgebied; en dan met name Larry Putnam, Barry Boehm en Capers Jones. Zelfs nu ik het boek zo goed als uit heb, is mij niet helemaal duidelijk wat nu McConnell's eigen bijdrage is.

Niet dat zo'n bloemlezing niet nuttig is – dit is absoluut het geval. Maar het boek mist een beetje een climax. Het kabbelt voort langs een grote verzameling tactiekjes, techniekjes en formules, terwijl ik eigenlijk zat te hopen op die ene alles omvergooiende eindconclusie. Ik kon 'm niet vinden.

Dus inmiddels weet ik dat je geen single point estimates moet maken, dat de wet van de grote getallen helpt, en dat je dus kleine features moet schatten en dat developers standaard 20% tot 30% te optimistisch zijn. En o ja, dat er een leuke, pragmatische techniek bestaat die t-shirt sizing heet, waarbij je zowel de technische complexiteit van een feature als de waarde van deze feature voor de business uitdrukt in S, M, L of XL, en vervolgens je project prioriteert op basis van deze data.

Maar met uitzondering van die laatste is dat eigenlijk geen nieuws. Hooguit een bevestiging van bestaande ideeën over het schatten van software development. Ook die ene techniek die je nu echt verder helpt in je projecten ontbreekt. Misschien dat ik nu toch maar eens moest beginnen aan het schrijven van een boek over het schatten met smart use-cases. De leukste anekdote over schatten in *Software Estimation* is overigens die waarin McConnell beschrijft hoe hij manmoedig en met diverse technieken probeert vast te stellen hoeveel pagina's zijn boek Code Complete zou gaan bevatten, en waarin de wens een boek van 250 tot 300 pagina's te schrijven overschaduwd werd door zijn schattingen die uitkwamen op zo'n 750 pagina's. Zoals bekend bleek de schatting uit te komen, en niet de wens. En voor de rest? Ach, er zijn leugens, grove leugens en statistiek.

Sander Hoogendoorn
(www.sanderhoogendoorn.com)

Waardering ★★★½☆

Auteur: Steve McConnell
Titel: *Software Estimation - Demystifying the Black Art*
Uitg.: Microsoft Press