Navigeren naar kwaliteit

Waarom dagdromen we wel van de nieuwste testosteron-Lamborghini en nooit van een nieuwe Twingo-zonder-turbo? Kennelijk is de eerste leuker of beter. Terwijl beide toch voldoen aan alle eisen die worden gesteld aan een auto. Dit dromen heeft meer te maken met voorkeuren dan met eisen. Ook voor software geldt zoiets. Zo kan opensource altijd beter zijn, of juist niet. Al is het subjectief, schijnbaar is er een maatlat die aangeeft wat beter is. Omdat ik software altijd wil verbeteren, is dat een interessant fenomeen.

In software geldt zo'n informele 'dit-is-beter-maatlat' niet alleen voor producten. Ook voor ontwikkelmanieren gelden sterke voorkeuren. Zo is Agile moderner en klaarblijkelijk daardoor beter. En al jaren geldt dat automatisch testen altijd beter is. Beter dan wat, vraag ik me dan af. De opdrachtgever is vaak duidelijk: het moet sneller en goedkoper, dan is het beter. Mijn vraag is opnieuw: hoe? Hoe maken we software beter, zowel het product als de manier?

Er wordt veel geschreven over Scrum en andere managementmethodieken. Die helpen, vooral om het ontwikkelproces efficiënter te managen. Ze zeggen echter weinig over het hoe. Met de invoering van Scrum lijkt de klassieke aanpak – analyse, design, coderen, testen – verouderd te zijn. Veel (beginnende) ontwikkelaars vragen zich daarom af hoe ze voor kwaliteit zorgen. Voor een ervaren, professionele topontwikkelaar speelt dat minder. Hij doet eerst een beetje van dit, dan een beetje zus en daarna komt er automatisch goede code. Maar omdat ik allergisch ben voor het woord 'automatisch' zoek ik verder. Ook omdat die automaat niet werkt voor aanstormend talent.

We kunnen natuurlijk gemakkelijk denigrerend doen over die jonge, surfende en klikkende, onervaren ontwikkelaars. Ik noem ze wel eens de tomtomgeneratie. Ze kunnen immers niet eens fatsoenlijk kaartlezen. Vroeger, toen moesten wij een autorit van tevoren plannen, continu zelf bijhouden waar we waren. En vooral niet fout rijden. Tegenwoordig stappen ze in, zetten hun apparaatje aan en rijden weg. Op elk kruispunt wordt ze verteld wat de juiste richting is. Ja kijk, zo is er geen kunst meer aan, zo kan iedereen het.

Albert Mietus is R&D-architect bij JVH Gaming en eigenaar van Softwarebetermaken.nl

Wellicht is ook bij softwareontwikkeling behoefte aan een navigatiesysteem. Geen eindeloos, ouderwets plannen en vooruitdenken. Maar een *agile* systeem dat stap voor stap vertelt welke kant je op moet en automatisch corrigeert als je een afslag mist. Die aanpak bestaat sinds kort bij JVH en heb ik Modimodi gedoopt.

Modimodi vertelt je niet eenmalig wat de route is, maar geeft telkens suggesties over de volgende stap om tot een goed product te komen. Ook als er (technische) blokkades zijn of als de specificaties veranderen. Helaas kan Modimodi niet echt praten. De ontwikkelaars zullen een hoop zelf moeten doen, zoals modelleren met objecten en het designen van interfaces (vandaar Modi). Dat moet cyclisch en in een hoog tempo. Het zus en zo van ervaren ontwikkelaars wordt expliciet gemaakt en toegankelijk, door gebruik te maken van eenvoudige bewegwijzering.

Een paar ontwikkelaars hebben deze nieuwe aanpak, dit navigatiesysteem, uitgeprobeerd. En terwijl u dit leest, geef ik workshops, omdat het voor ons werkt. Met kleine stapjes kunnen ze op diverse abstractieniveaus direct aan de slag. Bovendien verschuift hun focus hierdoor naar het zoeken van uitzonderingen en dus het voorkomen van fouten. Ook bij onervaren ontwikkelaars. En iedereen weet continu wat ze vanmorgen, vanmiddag of morgen kunnen doen.

Bovendien blijkt de Modimodi-aanpak prima te combineren met Scrum, Test-Driven Development en andere moderne, kortcyclische methodieken. Dat zijn zaken die jongeren aanspreken en die ze als vanzelfsprekend zien. Moderner is immers beter. Al heeft mijn methodiek (vooralsnog) geen enkele wetenschappelijke onderbouwing, de maatlat is duidelijk. Liever een aanpak die werkt en motiveert dan een ouderwetse, maar beproefde automaat-zonder-turbo.