Bericht uit 

"Echte mannen testen niet". Dat is wat je tussen de regels hoort wanneer je met ontwikkelaars over testen gaat praten. "We hebben toch een hele afdeling met test-specialisten, haal die er maar bij" is wat je de projectleiders hoort zeggen. Waar het om gaat: testen is vervelend. Testen kost altijd meer tijd dan je verwacht en je kunt nooit genoeg testen.

Testen en afwassen

Maar ja, eigenlijk zijn ontwikkelaars altijd een beetje bezig met testen. Alles wat ze doen zullen ze toch een keertje moeten beoordelen. Misschien draaien ze de juist gemaakte code even, zien dat het doet wat ze verwachten, en verklaren ze dat ze klaar zijn. Niemand kan in een keer een serieus systeem maken zonder ook maar één foutje te maken. Daar zijn we gewoon niet slim genoeg voor.

Testen kun je, zoals alle vervelende klusjes, op twee manieren aanpakken. De eerste en meest gangbare manier is: doorwerken, doorwerken, doorwerken en aan het eind van een heleboel werk maar eens gaan testen. De andere manier is: werken-testen, werken-testen, werken-testen. Het is net als met afwassen. De een verzamelt een aanrecht vol voordat hij/zij (meestal hij) gaat afwassen en de ander wast elk bordje af direct nadat hij/zij (meestal zij) het gebruikt heeft. Nu is er een verschil tussen de twee afwasstrategieën en de twee teststrategieën: de verhouding hoeveelheid afwas/afwasmoeite is constant: tien borden afwassen kost tien keer zo veel moeite als één bord afwassen. De verhouding programmeerwerk/testwerk lijkt veel minder constant.

Sommigen zeggen dat het voordelig is om testwerk op te sparen, je weet dan beter wat je moet testen en hoeft waarschijnlijk niet allerlei code te testen die uiteindelijk toch niet gebruikt is. Daar zit wat in. Later testen is minder testen! Waar hun argument misgaat is dat we niet afgerekend worden op de hoeveelheid tests die we doen. We worden afgerekend op de tijd die we besteden aan het totale ontwikkelen (inclusief testen) en op de kwaliteit van het resultaat. Efficiënt werken en goede kwaliteit afleveren is wel degelijk gebaat bij snel en vaak testen, zolang er maar winst is in de tijd die we besteden aan het coderen. Hoe kan (vaak) testen zorgen voor minder codeerwerk? Door te zorgen dat de test bepaalt wat we maken en niet andersom! In de wereld van Extreme Programming (XP) heet dat: Test First. Maak eerst een test, kijk of je test toevallig al werkt (je weet maar nooit) en verander het systeem dan zodanig dat het wel aan de test voldoet. Je gaat nergens aan sleutelen als er niet een test is die aangeeft dat er iets gemaakt of gerepareerd moet worden. Op die manier besteed je nergens meer tijd aan dan nodig en maak je alleen functionaliteit waar werkelijk om gevraagd wordt. Testen is natuurlijk niet eigen aan Java. Testen moet met elke programmeertaal. Wat wel specifiek is aan Java zijn een aantal tools die wij als Java-ontwikkelaars tot onze beschikking hebben. Het vergt even wat configureerwerk en inwerktijd, maar dan kun je je ontwikkelproces ook zodanig inrichten dat testen werkelijk een centraal, en sturend, onderdeel van je werk wordt. Tools als JUnit, Ant en CruiseControl. Het is net alsof je een afwasmachine koopt. Je moet je keuken ervoor uit elkaar halen, je bent wat kastruimte kwijt. Maar dat afwassen, dat doe je voortaan fluitend.

J. Meermans
is Java-kenner en te bereiken via
meermans@cibit.nl