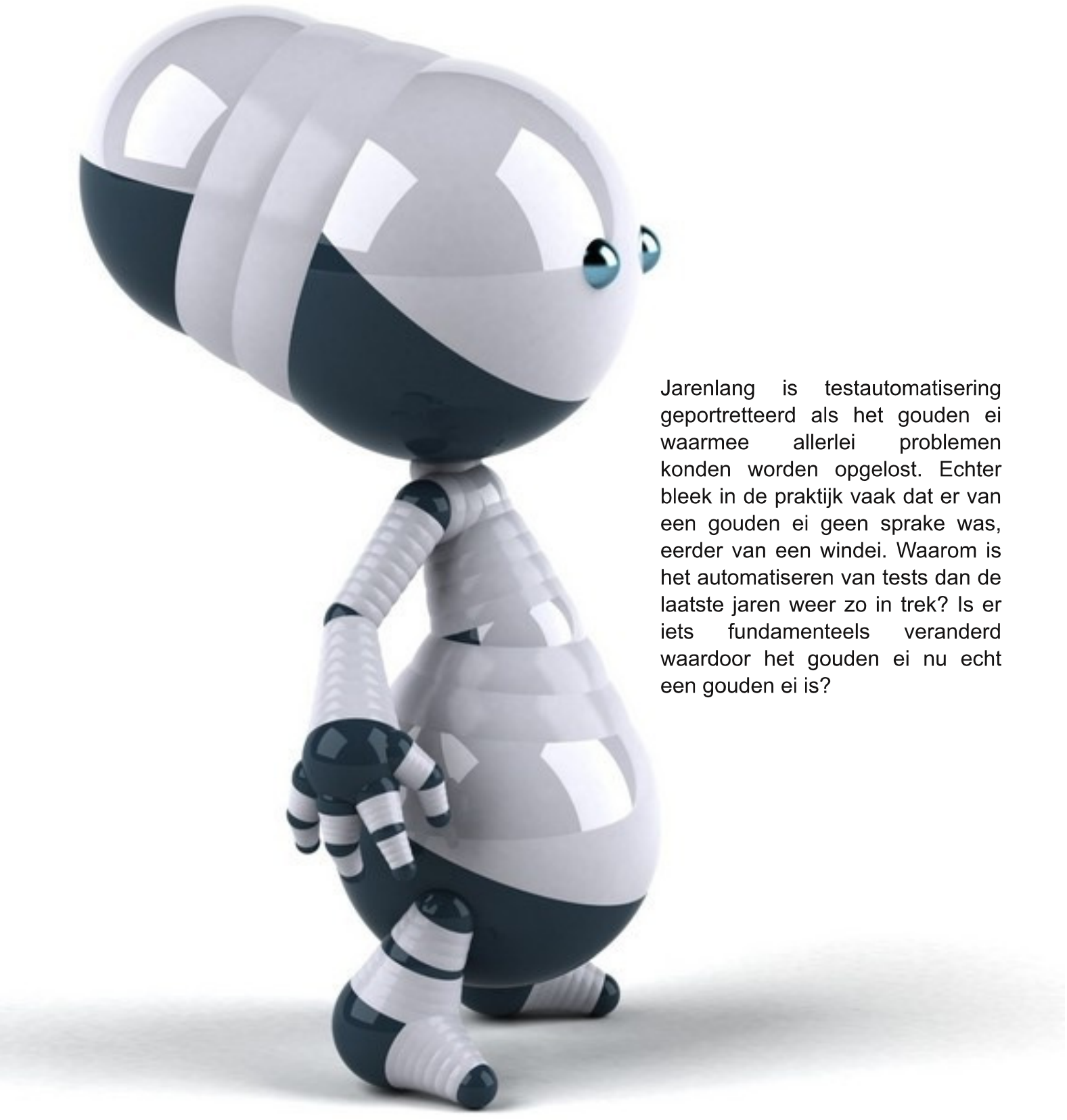
Testautomatisering
Waarom gaat het nu wel werken?

Door Martijn de Vrieze



Jarenlang is testautomatisering geportretteerd als het gouden ei waarmee allerlei problemen konden worden opgelost. Echter bleek in de praktijk vaak dat er van een gouden ei geen sprake was, eerder van een windei. Waarom is het automatiseren van tests dan de laatste jaren weer zo in trek? Is er iets fundamenteels veranderd waardoor het gouden ei nu echt een gouden ei is?

Een stukje geschiedenis

Van oudsher zijn de tools die men gebruikt voor het automatiseren van testen gebaseerd op het opnemen en afspelen (record/ playback) van testscenarios. Het grote voordeel hiervan, zei men, was dat er zonder enige kennis van code snel en eenvoudig tests geautomatiseerd konden worden. In de praktijk bleek echter al snel dat volledig steunen op record/ playback erg kwetsbaar was, aangezien deze opgenomen tests vrij breekbaar waren. Om de tests voor langere tijd bruikbaar te maken moest er toch veel gescript worden, vaak in de zelfbedachte (proprietary) talen van de diverse tools. De tools waren niet bepaald intuïtief in het gebruik waardoor er veel tijd en geld ging zitten in training in het gebruik van de tools en uiteraard in consultants om de tools te implementeren bij klanten.

De lange en vaak pijnlijke trajecten voor het implementeren van geautomatiseerd testen waarbij vaak over budgetten heen gegaan werd en deadlines niet gehaald werden hebben ertoe geleid dat testautomatisering gezien werd als een ‘bodemloze put’, ‘zonde van de tijd en het geld’ en ‘het scheelt uiteindelijk toch niets’.

In de afgelopen jaren is er een duidelijke verschuiving gekomen in het soort tools dat gebruikt wordt voor testautomatisering en daarmee ook in de succesverhalen rond geautomatiseerd testen. De ouderwetse record/playback tools zijn vervangen door veelal kleinere frameworks waar tegenaan geprogrammeerd moet worden, de proprietary talen van de tools zijn uitgefaseerd en vervangen voor volwassen programmeertalen als C#, Java en Python. Record/ playback wordt niet langer gezien als de oplossing, maar slechts gezien als inspiratie voor de daadwerkelijke testscripts. Naast de vele grote namen bij de commerciële tools zijn er steeds meer kleine namen en open source projecten in opkomst om het testen te automatiseren.

Waarom gaat het nu wel werken dan?

De afgelopen jaren is het testvak op vele gebieden volwassener geworden. Zo zijn de testopleidingen breder geworden dan alleen maar gericht op methodologieën, met trainingen in technisch testen, agile testen, testautomatisering, etc. Het technische aspect van testen is nadrukkelijker aanwezig in het werkveld van de tester. Doordat testers steeds meer betrokken raken bij het testen van de technische oplossingen in plaats van alleen de traditionele functionele tests, wordt het voor hen ook een logischere en kleinere stap om testautomatisering toe te passen.

Niet alleen de testtools maar ook de software heeft in de loop van de afgelopen jaren een grote verandering door gemaakt. Door de opkomst van Object Oriented Programming en Service Oriented Architecture (SOA) zijn er nieuwe lagen aangebracht in de software. Deze lagen zijn onafhankelijk van elkaar te testen en dus ook automatisch te testen. Door niet alleen op de graphical user interface (GUI) te automatiseren, maar ook net daaronder, kunnen tests sneller en stabieler geautomatiseerd worden.

De tests worden technischer en de applicaties en omgevingen die getest moeten worden zijn dat ook geworden. Het is niet afdoende bij een SOA omgeving om slechts via de user interface te testen, ook de onderliggende architectuur moet getest worden.

De API's (application programming interface) van de services lenen zich bij uitstek om automatisch te testen. Veel van de API's van tegenwoordig zijn van dermate technisch niveau dat tools altijd benodigd zijn om ze te testen. Al is het maar om de xml bestanden te genereren. Alle communicatie met de API's gaat namelijk door middel van (technische) berichten. Door gebruik te maken van tools om een API te testen wordt het geautomatiseerd testen van deze omgevingen bevorderd en vereenvoudigd.

Naast de omgevingen zijn ook de testautomatiseringstools gegroeid. Het merendeel van deze tools is niet meer record/playback gedreven, maar meer door de code of de modellen van de software under test. Doordat de tools, net als de software under test, technischer zijn geworden is het makkelijker geworden om zonder goede ontwikkelervaring of -kennis toch technische omgevingen solide te automatiseren. Doordat de tooling meer richting code gaat is het mogelijk herbruikbare modules te definiëren en daarmee de testautomatisering ook modulair op te bouwen.

Code gebaseerde testtools forceren de testers om beter na te denken over het automatiseren, het automatiseren te benaderen als code schrijven en dus ook de testscripts als dusdanig te onderhouden (in een versioning systeem, code reviews, herbruikbaar, onderhoudbaar etc.)

Daarnaast zijn de ontwikkelingen in de open source wereld van groot belang geweest voor de herontdekking van test automatisering. Er zijn diverse enorm populaire tools, zoals Selenium WebDriver, Sikuli, Watir en nog vele anderen die hebben bijgedragen aan het stijgende gebruik van testautomatisering en het succes daarvan binnen (vooral) agile omgevingen.

Tenslotte heeft naast tooling en technische omgevingen ook de opkomst en populariteit van Agile development methoden een grote positieve invloed op testautomatisering. Doordat testers en ontwikkelaars nauwer samenwerken in korte iteraties wordt het gebruik van testautomatisering bijna geforceerd. De doorlooptijden zijn dermate korter dan in traditionele ontwikkelmethoden, dat de doorlooptijd die nodig is voor regressie testen simpelweg niet beschikbaar is.

Testers en ontwikkelaars ontwikkelen samen een structurele oplossing voor de regressietest door binnen de iteratie nieuwe functionaliteiten zowel handmatig als geautomatiseerd te testen. Deze geautomatiseerde tests kunnen vervolgens weer opgenomen worden in de regressieset, waardoor deze in de loop van een aantal iteraties (of sprints) uitgroeit tot een volwassen, geautomatiseerde regressietest.
De technische expertise van de ontwikkelaars zorgt voor een hogere kwaliteit van de testautomatiseringscode.

Martijn de Vrieze is als testconsultant werkzaam bij Polteq Test Services. Vanuit zijn functie binnen Polteq ondersteunt Martijn organisaties bij veelal technische test trajecten. Dit door middel van advisering, coördinatie en uitvoering van test automatisering, load & performance testen en automatisering verbeter trajecten. Voor zijn start bij Polteq heeft Martijn bij verschillende werkgevers en opdrachtgevers gewerkt. Door de diversiteit aan opdrachtgevers heeft hij brede (automatisering) kennis opgedaan en toegepast.