Hoe te slagen met Model-Based Testing?

Door: E. Hartog, mei 2020

Rond het jaar 2010 was Model-Based Testing hot. Het werd gezien als een must-have. De methode zou relatief simpel ingevoerd kunnen worden met beperkte investeringen. Toch is het de afgelopen tien jaar niet gelukt het op grote schaal toe te passen. In dit artikel ga ik in op wat Model-Based Testing (MBT) zo aantrekkelijk maakt, maar toch vaak faalt. Op basis van die aspecten bespreek ik hoe je MBT wel succesvol kunt toepassen in echte projecten.

Wat is Model-Based Testing?

Model-Based Testing gebruikt modellen van het systeem om testen te creëren en eventueel uit te voeren. De modellen worden gemaakt op basis van de requirements of modellen worden hergebruikt vanuit het systeemontwerp.
Het grote voordeel zou zijn dat de testen eenvoudiger te onderhouden zijn. Tenslotte hoef je bij een wijziging alleen het model aan te passen; de testen rollen er vanzelf uit. Bovendien kun je die gegenereerde testen direct uitvoeren als ze geautomatiseerd zijn, waardoor je veel tijd kunt besparen. In dit artikel ga ik er van uit dat al enigszins bekend is wat Model-Based Testing is. Anders kun je Wikipedia [4] er nog op na lezen.

In dit artikel wordt Model-Based Testing (MBT) beschouwd waarbij de testgevallen direct door de tooling uitgevoerd kunnen worden. Dit is een beperking ten opzichte van de definitie (zie [1]) waarbij het gebruik van modellen en het afleiden van testgevallen met of zonder tooling ook al gezien wordt als MBT. In mijn ogen is dit echter het “normale” gebruik van testontwerptechnieken. Het automatisch genereren van testgevallen uit die modellen, levert uiteraard winst op. De echte winst zit echter in het automatisch uitvoeren van de testgevallen. Hierop zijn ook de grote verwachtingen uit het verleden gebaseerd.

Waarom faalt MBT?

Dit klinkt allemaal mooi, maar waarom is het gebruik in de praktijk dan toch zo minimaal? Het afgelopen decennium is er uitgebreid geëxperimenteerd met MBT, maar op weinig plaatsen is het echt succesvol toegepast. Tien jaar geleden waren er veel artikelen over en nu wordt er slechts sporadisch over bericht. Hoe kan dat nou? Er zijn verschillende redenen aan te wijzen waarom het niet lukt. Daarvan zijn de belangrijkste, de volgende:

a) Onvoldoend volwassen tooling
b) Te veel doelen stellen met MBT
c) Modellen niet op juiste abstractieniveau
d) MBT toepassen op gebieden waarop weinig winst is te halen

Onvoldoend volwassen tooling

Een belangrijke reden dat MBT niet goed wordt toegepast is het feit dat de tooling nog niet volwassen genoeg is. Er is veel tooling beschikbaar voor gebruik met verschillende modelleer-technieken, alleen is die nog steeds erg beperkt. Enkele zaken waar tools nog niet goed genoeg mee omgaan, zijn de volgende:

- Sommige tools bieden alleen een tekstinterface. Voor het onderhouden van een model is het vrijwel onmisbaar om ook een grafische wijze van modelleren beschikbaar te hebben, zeker in het geval van realistische c.q. complexe modellen.
- Om iets complexere modellen te kunnen gebruiken, is het nodig dat de tools ook een manier hebben om daarmee om te gaan. Een manier om dat te doen is het gebruik van hiërarchieën van modellen. Slechts enkele tools ondersteunen dat.
- Het varieert enorm hoe zeer de tools het testen automatiseren. Sommige genereren alleen testgevallen, terwijl andere direct aangesloten kunnen worden op de System Under Test

(SUT). Andere tools kunnen aangesloten worden op een SUT, maar alleen via een ander tool. Zoals eerder aangegeven is de echte winst te halen uit het automatisch uitvoeren van de testgevallen en de tool dient dat dus te ondersteunen.

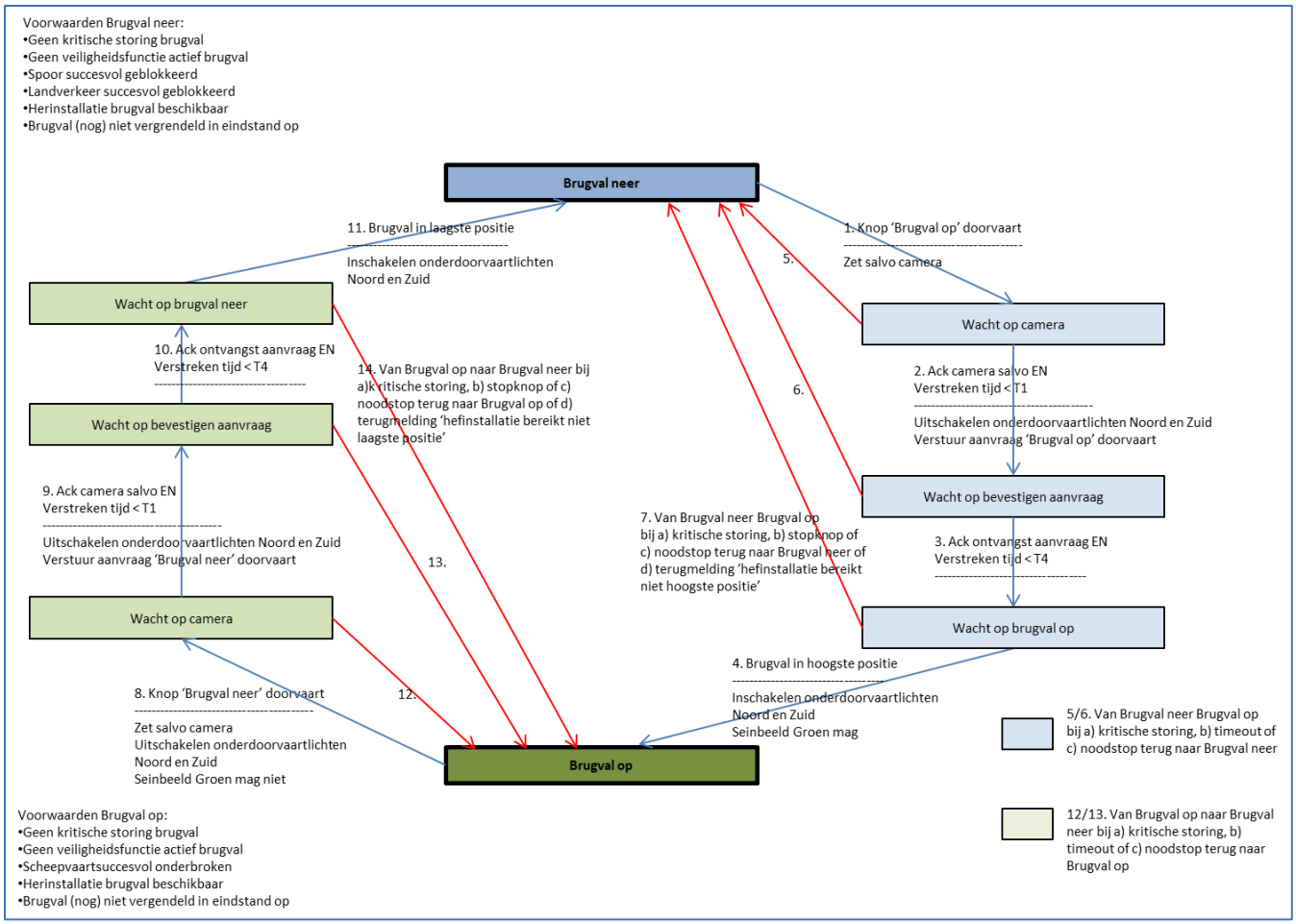
- De rapportagemogelijkheden van de tools blijven vaak beperkt tot het aangeven dat een testgeval niet goed doorlopen is. In andere gevallen komt er een getalletje uit dat aangeeft wat de coverage is (bijv. path coverage). Naast het feit dat dit veel werk geeft om uit te zoeken waar het probleem zit (SUT of model), is het ook geen rapportage die gebruikt kan worden voor de aansturing van de testen.
- Tools bieden maar één modelleertechniek aan en niet meerdere. Aangezien je bij het beperken van productrisico's vaak meerdere technieken gebruikt of zelfs technieken combineert (data met gedrag), kun je in dat geval niet volstaan met 1 techniek. Juist in deze gevallen verwacht je dat tools je kunnen helpen.

Te veel doelen stellen met MBT

Een belangrijke reden waarom MBT vaak faalt, is dat er verschillende verwachtingen zijn [1]. De tester verwacht een betere kwaliteit te krijgen door het inzetten van MBT. De projectmanager verwacht echter dat het testen korter gaat duren. De testanalist wil de modellen vooral gebruiken om onduidelijkheden op te lossen door het model te gebruiken als communicatiemiddel met de gebruikers of businessanalisten.
Zoals ook betoogd wordt in [1] lukt MBT niet als er geen duidelijke doel is waarvoor MBT ingezet wordt. Bijvoorbeeld: Als je betere kwaliteit wilt bereiken, dien je de modellen gedetailleerder te maken dan de bestaande testcases. Het maken van de modellen kost tijd, waardoor het testen dus niet korter wordt; in ieder geval niet in de eerste iteratie. In dat geval zal de projectmanager al snel teleurgesteld afhaken. Pilot mislukt.

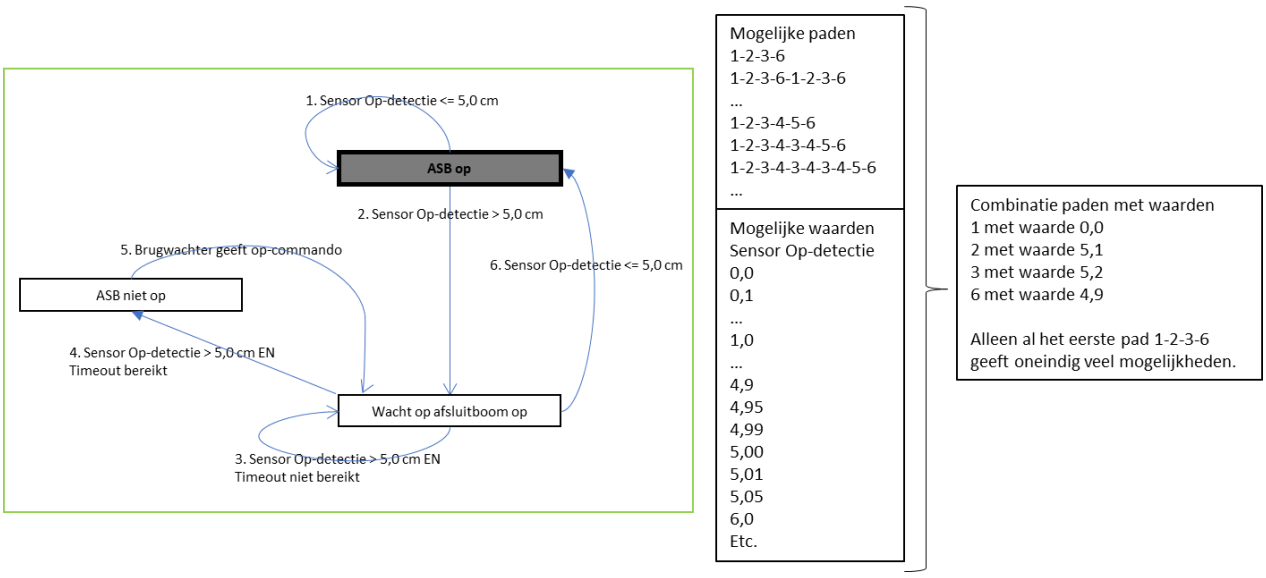
Modellen niet op juiste abstractieniveau

Veel mensen vinden modelleren moeilijk. Dit heeft onder andere te maken doordat het veel kennis en ervaring vereist om modellen op het juiste abstractieniveau te creëren. Helaas zijn slechts weinig testers opgeleid om goede modellen te maken. De modellen die gemaakt worden zijn prima bruikbaar voor het handmatig afleiden van testcases (als ze al gemaakt worden sic). Dit zijn dan ook de modellen die gebruikt worden in de onderzoeken die gedaan zijn naar de bruikbaarheid van MBT in de praktijk. Toepassing van MBT met deze modellen biedt in de praktijk weinig toegevoegde waarde. De complexiteit van deze modellen is simpelweg niet representatief genoeg voor de complexiteit van een systeem in de praktijk. Zie bijvoorbeeld figuur 1 (de blauwe toestand "brugval neer" is de begintoestand). Deze figuur komt uit een project voor het bedienen van een brugdek als onderdeel van de brugbesturing. Door de vele condities en mogelijke paden al best ingewikkeld, maar een echt systeem is vele malen ingewikkelder. In dit project is het model verder uitgewerkt, maar zelfs dan was het niet de moeite waard om de testcases te laten genereren aangezien dit nog prima handmatig kon.



Figuur 1. Model van de besturing van een brugval

In figuur 2 is een ander aspect te zien van modellen, namelijk het feit dat loops in modellen er voor zorgen dat er theoretisch oneindig veel testcases gemaakt kunnen worden. Als dit ook nog gecombineerd wordt met oneindig veel inputdata, wordt dit wel een testcase explosie genoemd. Het is niet zomaar van het model en draaien maar. Je dient hier bij het instellen van de tool rekening mee te houden.

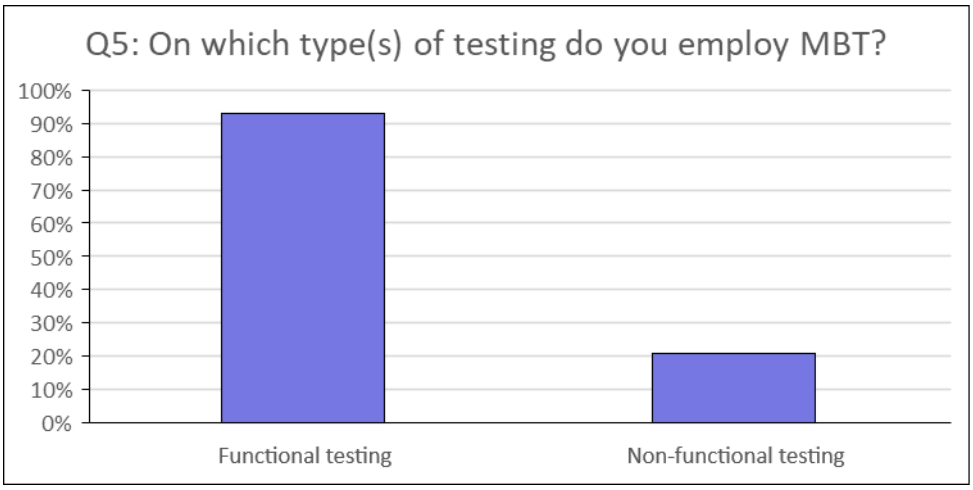


Figuur 2. Model van op-detectie van een afsluitboom

Bovendien blijkt het maken van deze complexe modellen in de praktijk zeer veel tijd te kosten (denk aan enkele maanden). Naast het feit dat de modellen qua structuur complex zijn, dienen in dit soort modellen ook zeer veel gedetailleerde gegevens toegevoegd worden om ze bruikbaar te maken voor toepassing in MBT. Het maken van het model en de kosten die dat met zich meebrengt, wegen dan niet meer op tegen de baten.

MBT toepassen op gebieden waarop weinig winst is te halen

Uit onderzoek blijkt dat MBT het meeste wordt toegepast voor functionele testen en weinig ingezet wordt voor niet-functionele testen (zie figuur 3, zie [2]). Voor functionele testen wordt met redelijk succes automatisering toegepast, waarbij handmatig gemaakte testcases in een testtool worden geïmporteerd om automatisch uitgevoerd te worden. Het testen van non-functionals blijkt in de praktijk niet veel te gebeuren. Een reden daarvoor is dat voor het testen van veel non-functionals vrijwel altijd ondersteuning van tooling nodig, zoals voor het testen van performance, parallelliteit en timing.



Figuur 3. Toepassing van MBT voor functionele en niet-functionele testen [2]

Hoe kan MBT dan wel gebruikt worden?

Om MBT goed te kunnen gebruiken in projecten, dienen we dus manieren te vinden om om te gaan met de redenen voor het falen ervan. De volgende paragrafen bespreken dit.

Duidelijke doelen voor gebruik van MBT

Het is belangrijk vooraf na te denken over het doel waarvoor MBT wordt ingezet. Volgens [1] bestaan de volgende doelen voor Model-Based Testing:

- Verbeteren van kwaliteit door het verhogen van de dekking;
- Verbeteren van de efficiëntie, d.i. sneller uitvoeren van test cases of meer test cases uitvoeren in dezelfde tijd;
- Beheersen van complexe modellen die handmatig niet te beheersen zouden zijn;
- Verbeteren van de communicatie, doordat modellen en diagrammen functionaliteit inzichtelijker weergeven, zodat gebruikers en business dit beter begrijpen en kunnen reviewen;
- Zo vroeg mogelijk in het project testontwerpen te maken om zo vroeg mogelijk fouten of problemen op te sporen.

In de praktijk ben ik meerdere voorbeelden tegengekomen waar MBT op een uitstekende wijze is toegepast, zolang van tevoren het doel gekozen is.
In een project werd een model gemaakt van een communicatieprotocol. Het bleek dat de beschrijving van het protocol onvolledig en inconsistent was. Het maken van het model heeft ervoor gezorgd dat dit protocol op een volledige en gestructureerde wijze beschreven is. Dit was uiteindelijk

ook het hoofddoel van het opstellen van het model. Dit heeft ervoor gezorgd dat zo vroeg mogelijk fouten en problemen opgespoord konden worden. Nadat het model gemaakt is, is het geschikt om testen uit te voeren op de gebouwde interfaces.
In een ander project was het model al grotendeels aanwezig vanuit financiële regelgeving. Eerder werden de regels van het model met de hand doorgerekend. Door dit model in te voeren in een MBT tool, konden snel efficiëntievoordelen behaald worden.
Een ander voorbeeld is van een project wat tot doel had om de kwaliteit te verhogen. In dit geval is er een model gemaakt om 2 parallelle applicaties na te bootsen. Met het model in het MBT-tool kan dan timing gesimuleerd worden, waardoor deadlock en dergelijke opgespoord kunnen worden.

Modellen op juiste abstractieniveau

Zoals eerder aangegeven worden de modellen vaak op een te hoog of een te laag abstractieniveau gemaakt, waardoor het toepassen van MBT geen toegevoegde waarde heeft of het teveel kost om het model te maken. Het is dus zaak om het model op het juiste abstractieniveau te maken.

Belangrijk bij het opstellen van modellen is ook dat het model een duidelijke focus heeft. Als dit ontbreekt wordt er veel detail in het model toegevoegd wat er voor zorgt dat het model te complex wordt. Door meerdere modellen te maken met elk zijn eigen focus, blijft elk model leesbaar en bruikbaar.

Goed modelleren is helaas erg lastig. Bovendien gebeurt het vooral door ontwerpers in de projecten en niet door testers. Toch zou modelleren ook voor testers een basisvaardigheid moeten zijn. Tenslotte worden er veel modellen gebruikt in de vorm van testontwerptechnieken. Bovendien hebben modellen voor testen andere aandachtspunten dan modellen voor ontwerp. Helaas zijn er desondanks weinig testers die de vaardigheid echt onder de knie hebben. Hou dus bij het bepalen van het abstractieniveau van het model rekening met de beschikbare kennis en ervaring van de testers met modelleren.

MBT toepassen op specifieke gebieden of aspecten

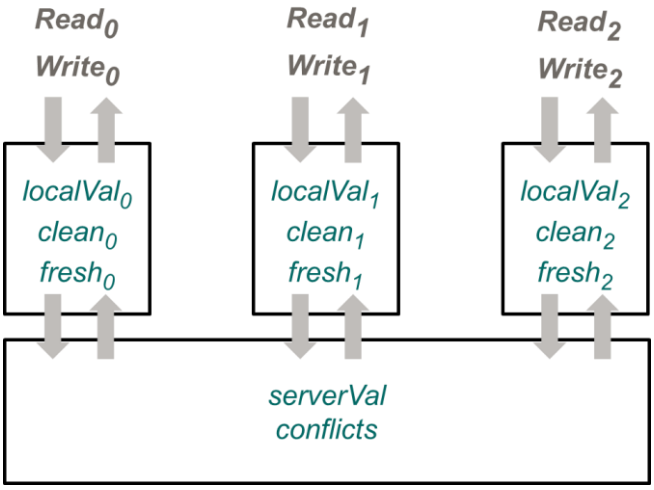
In projecten wordt vaak getracht MBT op alle onderdelen van het systeem te gebruiken. Het blijkt echter dat het teveel inspanning kost om voor alle onderdelen van het systeem modellen te ontwikkelen. De kosten worden te hoog en de baten blijven te laag. Door slim de delen van je systeem te kiezen waar MBT wel goed toegepast kan worden, blijven de kosten binnen de perken en de baten hoog.
De volgende mogelijkheden bestaan om specifieke delen te selecteren waar MBT prima geschikt voor is.

Hoge productrisico's
Als een risico hoog is (vanwege een grote faalkans, hoge kosten bij falen, of beide), kun je ervoor kiezen MBT in te zetten. Bij lage of gemiddelde risico's is het inzetten van MBT simpelweg als schieten op een mug met een kanon.
Dit betekent dus dat je niet in het gehele project MBT toepast. Je past MBT alleen toe op die gebieden waar een verdieping nodig is in verband met het hoge risico.

Specifieke non-functionals met name prestatie-efficiëntie, betrouwbaarheid en security
Sommige MBT-tools bieden randomisatie aan, waardoor het mogelijk is vele paden te testen die met de hand niet of zeer lastig getest kunnen worden.
Denk hierbij aan toepassing in het geval van **parallelliteit**. In dat geval kan de MBT tool prima gebruikt worden om verschillende paden langs te gaan over de verschillende processen om hiermee te testen of er sprake kan zijn van vastlopen (dead lock)

Ook in het geval van **timing issues** kan dit toegepast worden. Timing issues zijn lastig te testen, maar met de juiste MBT tool kunnen verschillende timings gebruikt worden om te zien of dit een probleem geeft. In dit geval wordt de randomisatie gebruikt voor de timing.

Een voorbeeld waar MBT is toegepast om te testen op parallelliteit en timing is beschreven in [3]. Figuur 4 toont het grafische model, zoals dat gebruikt is. Het model toont de wijze waarop DropBox werkt in het geval er vanaf meerdere computers bestanden gelezen en geschreven wordt. In dit geval is er een specifiek MBT-tool gebruikt (TorXakis) dat zelf in staat is via randomisering om verschillende paden uit te voeren over meerdere processen.



Figuur 4. Model voor Dropbox [3]

Tooling

Voor wat betreft de tooling is het lastig om hier veel aan te veranderen. De afgelopen jaren zijn er nieuwe tools gekomen, maar die lossen de eerdere problemen niet op. Eventueel zou je kunnen overwegen zelf tooling te ontwikkelen. Wat wel mogelijk is, is een manier te verzinnen waarbij we op een constructieve manier om kunnen gaan met de beperkingen van de tools. Dat bespreken we in de volgende paragraaf.

Conclusie

In dit artikel heb ik beschreven waarom MBT vaak faalt in de praktijk. Bij goede toepassing kan MBT echter in veel situaties een goed hulpmiddel zijn. Goede toepassing richt zich op:

- Zorg dat **toolkennis** beschikbaar is; huur deze eventueel in;
- Kies een duidelijk **gefocust doel**, zoals het verbeteren van kwaliteit of de efficiëntie;
- Bepaal de benodigde **abstractie** in het model; neem één aspect als doel van het model en let op de beschikbare modelleerkennis en -ervaring;
- Hou de **scope beperkt** en richt je op specifieke delen of aspecten van het systeem; Denk hier aan delen met een hoog productrisico of non-functionals.

Hopelijk geeft dit artikel weer een zetje in de goede richting van Model-Based Testing. Succes met de toepassing ervan.

Referenties

[1] 2016 Model-based Testing Essentials, Wiley, Anne Kramer, Bruno Legeard
[2] 2019 Model-based Testing User Survey: Results, Anne Kramer, Bruno Legeard
[3] Model-Based Testing with TorXakis – The Mysteries of Dropbox Revisited, Jan Tretmans en Piërre van de Laar (https://repository.ubn.ru.nl/bitstream/handle/2066/214659/214659.pdf)
[4] Model-Based Testing, https://en.wikipedia.org/wiki/Model-based_testing

Werkgroep Model-Based Testing

Binnen TestNet bestaat de werkgroep Model-Based Testing. Enige tijd geleden ben ik daar lid van geworden, omdat Model-Based Testing een interessante ontwikkeling is binnen ons testvak. In de werkgroep zijn we tijdens meerdere sessies bezig geweest met het opstellen van modellen en de toepassing daarvan met verschillende tools. Deze sessies gaven mij ideeën en de inspiratie om tot dit artikel te komen. Hiervoor wil ik de deelnemers van de werkgroep bedanken: Rachid, Jan, Jan, Ed, Axel, Leo en René.

Over de auteur

Eduard Hartog: Na 16 jaar als ontwikkelaar en projectleider gewerkt te hebben, heeft Eduard uiteindelijk gekozen voor het kwaliteitsvak. Inmiddels heeft Eduard meer dan 15 jaar ervaring met zowel testen als kwaliteitsmanagement in het technische domein. Hij werkt veel in civiele projecten met een installatie-technische component, zoals tunnels, bruggen en spoor. Daarnaast werkt hij in de communicatie- en offshoresector. Binnen TestNet is hij actief binnen de werkgroep Model-Based Testing.

Contactgegevens: eduard.hartog@improveqs.nl