

# Automatische modelgebaseerde testgeneratie en -uitvoering

Alan Hartman en Steven Van Proeyen

## Samenvatting

De testdiscipline zou kunnen profiteren van een nauwere samenwerking tussen de academische wereld en de bedrijfs wereld op zoek naar praktisch toepasbare methoden met behulp van industrie-sterke schaalbare tools, dit alles vertrekkende vanuit kennis omtrent huidige testtechnieken, testtools en testprocessen. Het AGEDIS project dat tracht op dit vlak een stap in de goede richting te zetten, wordt voorgesteld. Onderwerp van dit onderzoeksproject is test automatisering, meer concreet automatische modelgebaseerde testgeneratie en –uitvoering. De nadruk ligt bij test generatie, een testfase die indien zij op een gepaste manier wordt geautomatiseerd de kost-effectiviteit van het testen opmerkelijk kan verbeteren.

## Inleiding

Uit het inleidend artikel kon reeds verstaan worden dat onderzoek in software testen gevoelig achterloopt op andere software disciplines. Toch representeert deze test discipline een belangrijke schakel in het tijdig, binnen het budget naar de markt brengen van kwalitatieve software. Om verandering in deze situatie te brengen is het niet genoeg dat academici zoeken naar nieuwe testtechnieken zonder rekening te houden met behoeften en ontwikkelingen in de bedrijfs wereld, of dat de bedrijfs wereld experimenteert met nieuwe testprocessen zonder gebruik te maken van reeds interessante bestaande technieken, noch is het voldoende dat de bedrijfs wereld en de academische wereld samenwerken aan specifieke testtechnieken, maar deze technieken niet ondersteunen door praktisch toepasbare methoden en bruikbare tools. De testdiscipline zou kunnen profiteren van een nauwere samenwerking tussen de academische wereld en de bedrijfs wereld op zoek naar praktisch toepasbare methoden met behulp van industrie-sterke schaalbare tools, dit alles vertrekkende vanuit kennis omtrent huidige testtechnieken, testtools en testprocessen. Het AGEDIS project dat tracht op dit vlak een stap in de goede richting te zetten, wordt voorgesteld. Onderwerp van dit onderzoeksproject is test automatisering, meer concreet automatische modelgebaseerde testgeneratie en –uitvoering.

## Modelgebaseerd testen<sup>1</sup>

In principe kan elke vorm van software testen als modelgebaseerd bestempeld worden. De tester gaat steeds een mentaal model vormen van het te testen systeem alvorens taken aan te vatten zoals het ontwerpen van test cases. Wanneer men deze modellen persistent gaat documenteren en vervolgens gebruiken voor het genereren van test cases of het evalueren van testresultaten gebruikt men veelal de term modelgebaseerd testen. Modellen kunnen vele vormen aannemen: diagrammen (bijvoorbeeld UML toestandsdiagrammen), grammatica's, tabellen (bijvoorbeeld beslissingstabellen), control flow grafieken, ... Afhankelijk van de specifieke situatie zullen bepaalde modellen meer gepast zijn dan andere modelvormen<sup>2</sup>.

---

<sup>1</sup> Een interessante collectie artikels over modelgebaseerd testen kan online bekeken worden op [http://www.geocities.com/model\\_based\\_testing/](http://www.geocities.com/model_based_testing/)

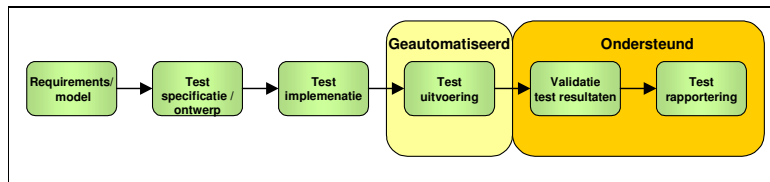
<sup>2</sup> Niettegenstaande een controle flow grafiek die de interne structuur van de source code weerspiegelt ook als een model kan aanzien worden, wordt de term modelgebaseerd testen vooral gebruikt in de

Modelgebaseerd testen wordt interessant wanneer tests automatisch kunnen worden gegenereerd met daarvoor ontwikkelde tools. Het proces van modelgebaseerde testgeneratie kan echter ook manueel worden uitgevoerd. Voor een goed voorbeeld hiervan verwijzen we naar een artikel van Microsoft's modelgebaseerde test goeroe Harry Robinson.<sup>3</sup>

Niettegenstaande we de laatste jaren een enorme groei hebben kunnen zien van modelgebaseerde ontwikkelingsactiviteiten, voornamelijk UML gerelateerd, is lang niet iedereen het eens over het belang en de positie van modellering in het gehele ontwikkelingsproces. Systemen worden nog vaak gemodelleerd na het implementeren van het systeem (of delen ervan). In vele gevallen fungeren de modellen als basis voor code generatie van een deel van het systeem of wordt het gebruikt in functie van documentatiedoeleinden. Een belangrijk deel van de kosten gepaard gaande met debug- en testactiviteiten is dermate hoog door het gebrek aan een sterke link tussen het ontwerp enerzijds en het creëren en uitvoeren van tests anderzijds<sup>4</sup>. Een betere integratie tussen test- en ontwikkelactiviteiten zou hier enig soelaas kunnen bieden. Indien een modelgebaseerde testmethode aanwezig zou zijn, die de manuele testactiviteit voor een deel kan vervangen en de onderhoudbaarheid van de testware aanzienlijk kan vereenvoudigen, zouden de testkosten aanzienlijk gedrukt kunnen worden. Deze redenering is wellicht een oorzaak van de recente opgang in UML gebaseerde test research en vormt tevens één van de drijfveren achter het AGEDIS project.

### Automatische testgeneratie en –uitvoering

Het woord automatisering in een test context kan vele dingen betekenen. In figuur 1 worden de verschillende testactiviteiten voorgesteld met een indicatie waar de automatisering zich momenteel in de meeste testautomatiseringsprojecten bevindt, namelijk de testuitvoeringsfase. Hiervoor wordt dikwijls gebruik gemaakt van een 'test automation framework'. Typische voorbeelden zijn Mercury's WinRunner en Rational Robot. De andere testtaken worden geheel of gedeeltelijk manueel afgehandeld.



Figuur 1: Conventionele aanpak testautomatisering

Bij het toepassen van automatische testgeneratie gaat men ook streven naar het automatiseren van het ontwerp en de implementatie van test cases d.m.v. een test generator. Er kan een onderscheid gemaakt worden tussen test generatoren en modelgebaseerde input generatoren. Deze input generatoren genereren niet de verwachte output (en dus geen volwaardige testcases) waardoor de gebruiker een orakel nodig heeft dat de verwachte output gaat voortbrengen naast de gegenereerde

---

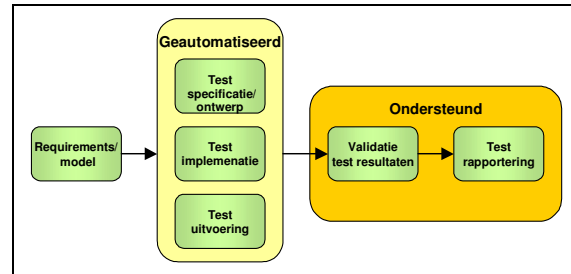
context van black-box modellering van software. Model-based Software Testing (El-Far I.K. en J.A. Whittaker) ([http://testingresearch.com/Ibrahim/papers\\_html/encyclopedia.htm](http://testingresearch.com/Ibrahim/papers_html/encyclopedia.htm))

<sup>3</sup> Robinson H. (1999), Finite State Model-Based Testing on a Shoestring. [http://www.geocities.com/model\\_based\\_testing/online\\_papers.htm](http://www.geocities.com/model_based_testing/online_papers.htm)

<sup>4</sup> Hailpern B. en P. Santhanam, Software debugging, testing and verification, IBM Systems Journal, Vol 41, N 1, 2002. <http://www.research.ibm.com/journal/sj>

input sequenties. Figuur 2 geeft de toename van automatisering weer bij het toepassen van automatische testgeneratie en –uitvoering. Er wordt benadrukt dat dergelijke test aanpak geen algemene gang van zaken is en voor een groot deel nog onderzoeksgebied betreft.

Wanneer de testgeneratie (of een deel) automatisch voortvloeit uit modelspecificaties van het te testen systeem en test cases automatisch worden uitgevoerd door een ‘test automation framework’ spreekt men van modelgebaseerde testgeneratie en –uitvoering.



Figuur 2: Testautomatisering AGEDIS

## AGEDIS<sup>5</sup>

Test automatisering is niet nieuw. Reeds in de jaren 70 van de vorige eeuw werd test automatisering onderzocht<sup>6</sup>. Een groot aantal test automatisering tools hebben het daglicht gezien. Toch werd terecht recentelijk opgemerkt dat bij het creëren van test automatisering tools ongeveer 80 procent van de inzet gericht is naar het opzetten van test infrastructuur en het mogelijk maken van testactiviteiten en slechts 20 procent zichtbare functionaliteit en toegevoegde waarde betreft<sup>7</sup>. Een voornaam objectief van het AGEDIS project is het verminderen van deze 80 procent voor nieuwe initiatieven door het zo toegankelijk mogelijk maken van de tool architectuur. Positieve ervaringen met automatische modelgebaseerde testgeneratie en –uitvoering intern bij enkele partners zijn een belangrijke drijfveer voor het AGEDIS project<sup>8</sup>.

Bij het concretiseren van figuur 2 in een test methodologie en tool architectuur zijn vele pistes mogelijk. Voor een model kunnen verschillende modelleertalen gebruikt worden zoals SDL of UML, test specificaties kunnen voorgesteld worden in diverse formaten, enz. De AGEDIS test methodologie en tool architectuur worden beknopt beschreven. Vele eigenschappen van AGEDIS worden bevestigd in het artikel ‘een ideale architectuur voor modelgebaseerde test- en verificatiesystemen’ van testexpert David Gelperin<sup>9</sup>.

## Test methodologie

De AGEDIS test methodologie is gebaseerd op een iteratief proces en kan onderverdeeld worden in 6 voorname stappen (zie figuur 3). Allereerst wordt een gedragsmodel van het te testen systeem gebouwd en worden testgeneratie directieven opgesteld. Het model is voor een groot deel gebaseerd op toestandsdiagrammen die

<sup>5</sup> Automated Generation and Execution of test suites for DIstributed component-based Software.

<sup>6</sup> Jessop W.H. e.a. (1976), ATLAS – an Automated Software Testing System, 2<sup>nd</sup> International Conference on Software Engineering.

<sup>7</sup> Guckenheimer S. (2002), The Revolution in Software Testing.

[http://www.therationaledge.com/content/dec\\_02/f\\_revolutionSoftwareTesting\\_sg.jsp](http://www.therationaledge.com/content/dec_02/f_revolutionSoftwareTesting_sg.jsp)

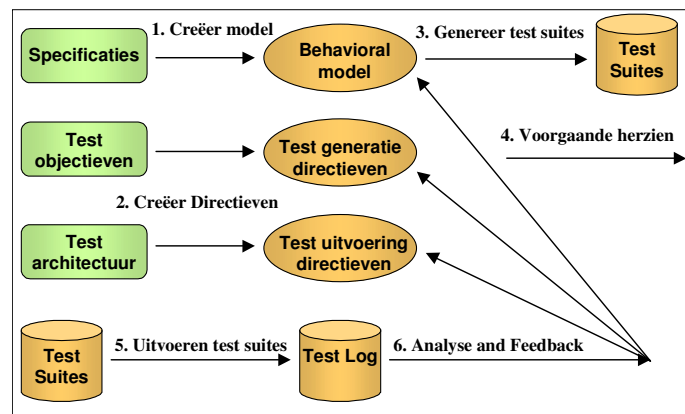
<sup>8</sup> Gronau I. e.a. (2000), A Methodology and Architecture for Automated Software Testing.

<http://www.haifa.il.ibm.com/projects/verification/gtcb/publications.html>

<sup>9</sup> Gelperin D. (2001), An ideal architecture for model-based verification & test systems.

<http://www.stickyminds.com>

vlug onderhevig zijn aan een explosie van toestanden bij het modelleren van een niet triviale applicatie. Vandaar dat gebruik wordt gemaakt van test directieven die een selectie van paden zullen maken uit het geheel aan mogelijke paden.



Figuur 3: test methodologie

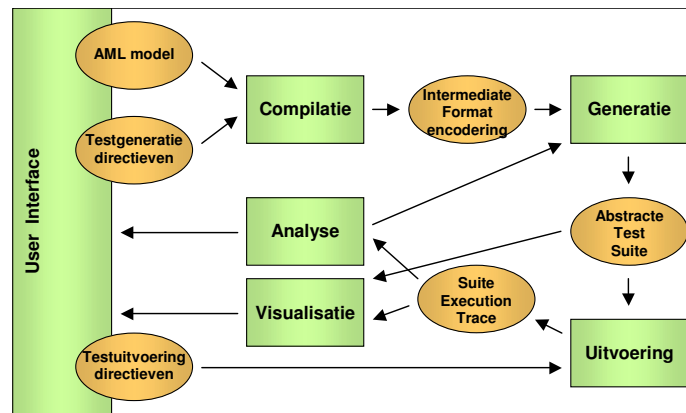
Tevens wordt de test interface tussen het model en het te testen systeem beschreven, zoals o.a. controle- en observatiepunten. Vervolgens wordt automatisch een test suite gegenereerd en het model, de test informatie en de test suites worden herzien met ontwikkelaars (en klanten). De test suite wordt automatisch uitgevoerd en de resultaten worden gelogd. Stap 2 tot en met 5 worden vervolgens herhaald tot de gewenste coverage en kwaliteitsobjectieven behaald zijn.

### Tool architectuur

De tool architectuur (zie figuur 4) bestaat uit een software modelleertaal, een compiler voor deze taal, een coverage-gedreven test generator, een component-gebaseerde test execution engine, en productiviteit en feedback tools. Een belangrijk objectief van AGEDIS is de tool architectuur open te houden. Zo wordt als modelleertaal geen proprietary taal gebruikt en kunnen ook andere modelleertalen ontwikkeld worden die passen in de architectuur. Met het oog op een breed gamma van toepassingen en het voorkomen van een te steile leercurve werd gekozen voor een op UML-gebaseerde modelleertaal. De abstracte test suites (ATS) en de suite execution trace (SET) zijn XML-gebaseerd en kunnen dus eenvoudig (bijv. met een xml editor) bekeken en bewerkt worden of gebruikt worden als basis voor geavanceerde analyse- en visualisatietools. Door het hanteren van abstracte test suites wordt er een onafhankelijkheid gecreëerd t.o.v. de test execution engine (uitvoering).

AML (AGEDIS Modeling Language) is gebaseerd op een beperkte subset van de UML 1.4 standaard en incorporeert klasse-, object- en toestandsdiagrammen. De vereisten voor de taal vloeien voort uit studies in het kader van het project naar andere talen (o.a. SDL, Murphi, UML, CSP, Z) en aanbevelingen aangebracht uit case studies met voorlopers van de AGEDIS tools. Door het uitbreiden van UML a.h.v. stereotypes en 'tagged values' wordt het mogelijk compileerbare AML modellen te construeren. De AML functionaliteit wordt eenvoudig aangeboden door het gebruik van een AML profile dat geïnstalleerd kan worden in een UML modelleertool. De output van de compilatie is in het IF 2.0 (Intermediate Format) formaat waarvan de

specificatie vrij beschikbaar is<sup>10</sup>. Gebaseerd op het model construeert AGEDIS een toestandsmachine die het gedrag van de software simuleert. Elk uitvoeringspad van het model is een mogelijke test case. Het mogelijk aantal uitvoeringspaden is meestal zeer hoog en wordt beperkt door het toepassen van testgeneratie directieven.



Figuur 4: tool architectuur

In verschillende testgeneratie tools zoals T-VEC en UniTesK zijn deze test directieven impliciet ingebouwd in de tools en kunnen dus niet aangepast worden aan verschillende noden<sup>11</sup>. De kost van deze flexibiliteit is enige toename in complexiteit. Test directieven kunnen bestaan uit coverage criteria, test objectieven en test constraints. De output van de testgenerator is een abstracte test suite, een verzameling test cases die kunnen worden uitgevoerd op de te testen applicatie, in een formaat dat onafhankelijk is van de test execution engine<sup>12</sup>. Ook manueel gecreëerde tests kunnen aan de test suite worden toegevoegd. De output van de test uitvoering is het XML-gebaseerde SET (Suite Execution Trace). Testuitvoering directieven sturen de uitvoering d.m.v. bijvoorbeeld het aangeven van mappings tussen methodes in het ATS formaat en concrete methodes die interageren met het te testen systeem. De SET bestanden kunnen als basis dienen voor geavanceerde analyse- en visualisatietools, inclusief feedback tools die het mogelijk maken de test suites beter aan te passen aan de specifieke noden.

## Conclusie

Het installeren van een testproces gefundeerd op automatische modelgebaseerde testgeneratie en –uitvoering kan het automatiseringsniveau drastisch verhogen. De verschuiving van test cases naar software model, als cruciale testware heeft groot potentieel tot het verlagen van de testkosten. Een grondige bestudering van de haalbaarheid van de aanpak en een economische afweging in verschillende contexten is daarom aangewezen. Het ideale moment lijkt gekomen om deze haalbaarheid te onderzoeken. Een kritische massa heeft zich ontwikkeld met enige ervaring in UML

<sup>10</sup> <http://www.agedis.de/downloads.shtml>

<sup>11</sup> Hartman A., Model based test generation tools. <http://www.agedis.de/downloads.shtml>

<sup>12</sup> De testgenerator is gebaseerd op technologie van een aantal partners, met name GOTCHA (Generation of Test Cases for Hardware Architectures) en TGV (Test Generation with Verification) en profiteert op deze manier van deze expertise en ervaring.  
<http://www.haifa.il.ibm.com/projects/verification/gtcb/publications.html>,  
<http://www.irisa.fr/pampa/VALIDATION/TGV/TGV.html>

modellering, modelgebaseerde ontwikkeling kent momenteel een hoogtepunt en de modelleertools en uitwisselingsformaten lijken volwassen te worden.

AGEDIS onderscheidt zich vooral door de praktische UML-gebaseerde aanpak, de scheiding van het opstellen van test directieven enerzijds en het systeemmodel anderzijds, en de open architectuur op basis van standaarden. Het vertrekt vanuit state-of-the-art testtechnologie en know-how aangebracht door de academische en commerciële partners. Bovendien onderzoekt het project de mogelijke complementariteit met bestaande testtechnieken en ontwikkel- en testprocessen en heeft het als belangrijke objectieven de aanpak toe te passen op realistische case studies en het aanleveren van industrie-sterke tools ter ondersteuning van de test methodologie. De open architectuur nodigt uit tot complementaire onderzoeksinitiatieven waarbij reeds bestaande elementen kunnen hergebruikt worden. De vooruitgang van het AGEDIS project kan gevolgd worden op de officiële website <http://www.agedis.de>.

#### Andere referenties

- Automated Generation and Execution of Test Suites, AGEDIS White Paper, weldra beschikbaar
- Fewster M. en Graham D. (1999), Software Test Automation, Addison-Wesley
- Dustin E. e.a. (1999), Automated Software Testing: Introduction, Management and Performance, Addison-Wesley