



Werkgroep Model Based Testen

..Vakgenoten leren van vakgenoten.....

Even voorstellen



Rachid Kherrazi

Rachid Kherrazi CTO AKKA Netherlands
20 jaar ervaring in ICT:



ASML

altran

AKKA
PASSION FOR
TECHNOLOGIES

Actief lid TestNet community
Trekker MBT werkgroep
Ervaring met Security Testing, Model Based
Testing, Artificial Intelligence en Test
automatisering



Eduard Hartog

Eduard Hartog
Senior Consultant Improve Quality Services BV
30 jaar ervaring in ICT



Top Systems
POWERED BY STEFANINI

ICT+
Improve
QUALITY SERVICES

Actief lid MBT werkgroep
Van ontwikkelaar t/m project manager
Laatste 10 jaar in multidisciplinaire projecten
Test Management

Agenda

- Intro
 - Wie zijn wij?
 - Wat is MBT?
- Wat hebben we afgelopen tijd gedaan?
- Lessons learned vanuit pilots en praktijk
- Demo – SpecExplorer
- Hoe nu verder?
- Q&A

TEST NET

Inloggen Zoeken...

Startpagina Evenementen Bibliotheek **Wergroepen** Contact

Actieve werkgroepen

Afgeronde werkgroepen

Hoofdsponsors 2020

bartosz CGI Capgemini
CloseSure OELAN
Improve PARASOFT
politeqa identfy
sogeti SQUERIST
SUPPORT BOOK TALENT TESTERSHUTE
ISQATE qppocconsult

Wilt u TestNet ook sponsoren?
Kijk hier en kijk naar de mogelijkheden!

Werkgroep Model Based Testen

Trekker: **Rachid Kherrazi**

Zoekt nog leden: Ja

Inleiding:
Model Based Testen (MBT) is een nieuwe en veelbelovende manier om de kwaliteit van ICT te verbeteren. Bij MBT wordt een abstract model gemaakt dat het gewenste gedrag van het systeem op een precieze en inzichtelijke manier weergeeft, doorgaans grafisch. Met behulp van een MBT tool kunnen uit dit model automatisch testen gegenereerd worden. MBT is de volgende logische stap in testautomatisering: naast automatische executie van scripted of keyword-based test cases kunnen test cases nu ook automatisch gegenereerd worden. Daarnaast levert het model veel inzicht op, en worden vele fouten en inconsistenties al tijdens het maken van het model gevonden, zonder dat er ook maar een test is uitgevoerd.



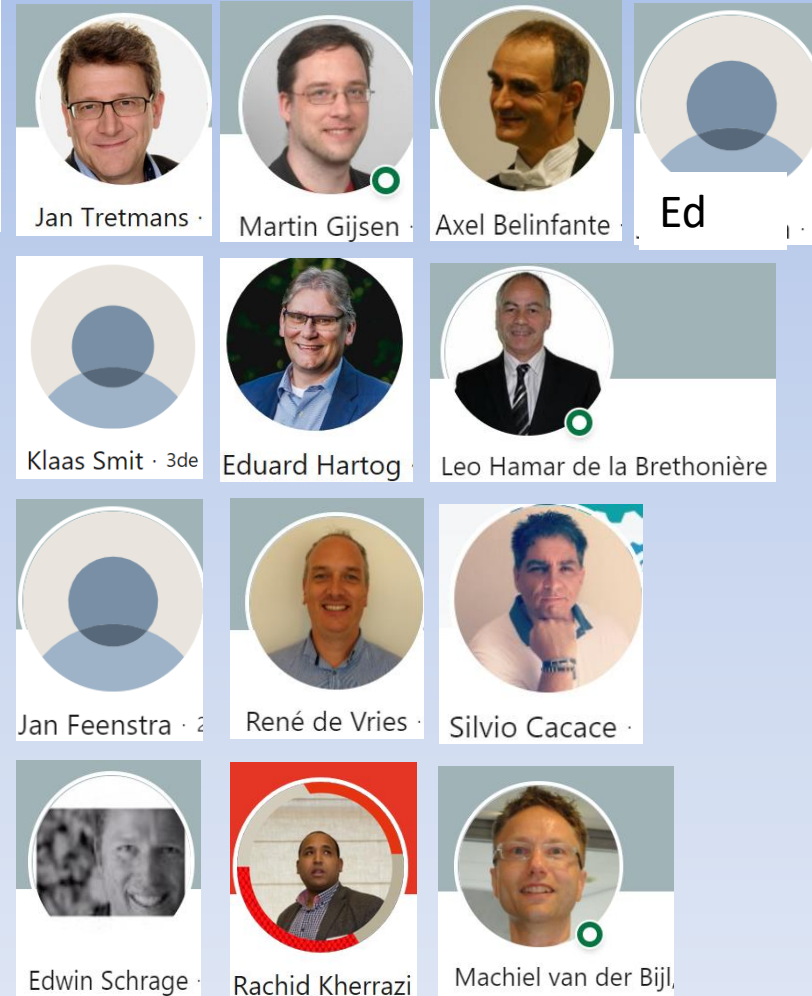

Omdat MBT veel meer dan traditionele technieken afhankelijk is van tooling, is het belangrijk onderzoek te doen naar de bruikbaarheid en schaalbaarheid van deze tools.

Daarom heeft de werkgroep zich de afgelopen jaren sterk gefocust op het verkennen van het toollandschap voor MBT.

In deze presentatie willen we de resultaten van ons onderzoek met jullie delen en een kijkje geven in onze toekomstige plannen waar we samenwerking zoeken met de Artificiële Intelligentie en Testen werkgroep.

Agenda

- Wat was ons plan
- Wat is er bereikt en
- Wat zijn de vervolg stappen



Jan Tretmans · Martin Gijsen · Axel Belinfante · Ed

Klaas Smit · 3de · Eduard Hartog · Leo Hamar de la Brethonière

Jan Feenstra · 2 · René de Vries · Silvio Cacace ·

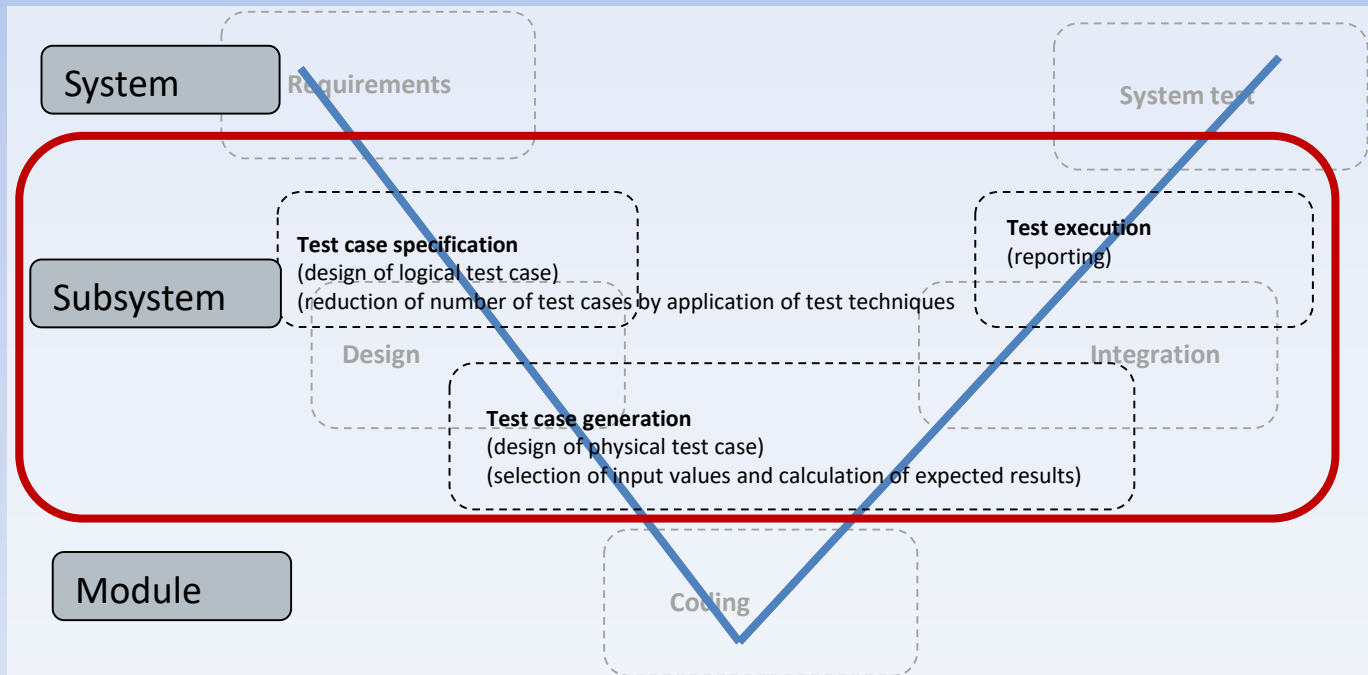
Edwin Schrage · Rachid Kherrazi · Machiel van der Bijl

Wat is Model Based Testen?

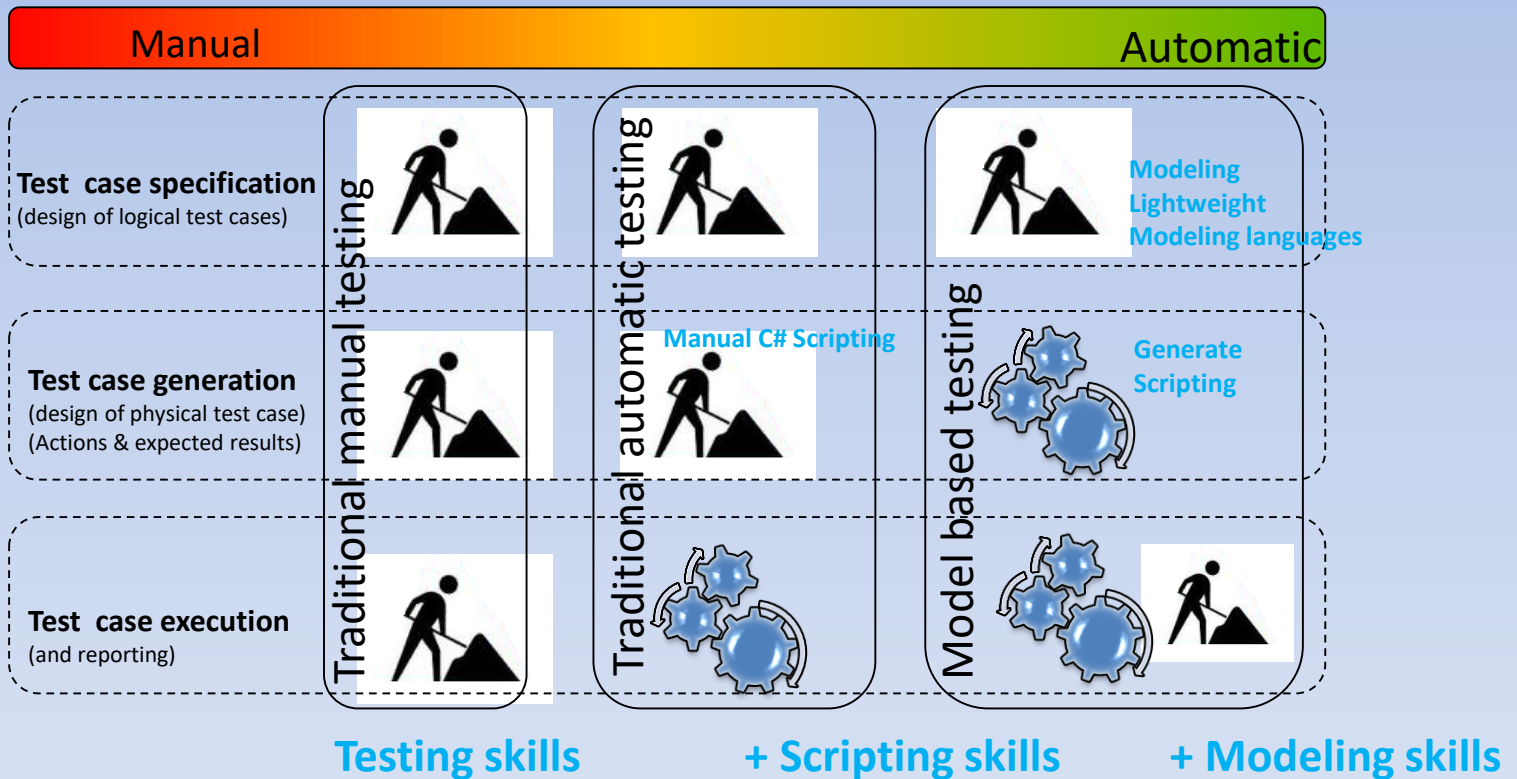
- Model Based Testen (MBT) is een bewezen en veelbelovende manier om de kwaliteit van ICT te verbeteren.
- Bij MBT wordt een abstract model gemaakt dat het gewenste gedrag van het systeem op een precieze en inzichtelijke manier weergeeft, doorgaans grafisch.
- Met behulp van een MBT tool kunnen uit dit model automatisch testen gegenereerd worden.

Testproces

3 main steps in test process



MBT is the automation of test case generation



Even voorstellen



Rachid Kherrazi

Rachid Kherrazi CTO AKKA Netherlands
20 jaar ervaring in ICT:



ASML

ALTRAN

AKKA
PASSION FOR
TECHNOLOGIES

Actief lid TestNet community
Trekker MBT werkgroep
Ervaring met Security Testing, Model Based
Testing, Artificial Intelligence en Test
automatisering



Eduard Hartog

Eduard Hartog
Senior Consultant Improve Quality Services BV
30 jaar ervaring in ICT



Top Systems
POWERED BY STEFANINI

ICT+
Improve
QUALITY SERVICES

Actief lid MBT werkgroep
Van ontwikkelaar t/m project manager
Laatste 10 jaar in multidisciplinaire projecten
Test Management

Wat hebben we gedaan?

- Tools gekozen uit long-list
- Pilots uitgevoerd met tools
- Lessons learned vanuit pilots en praktijk

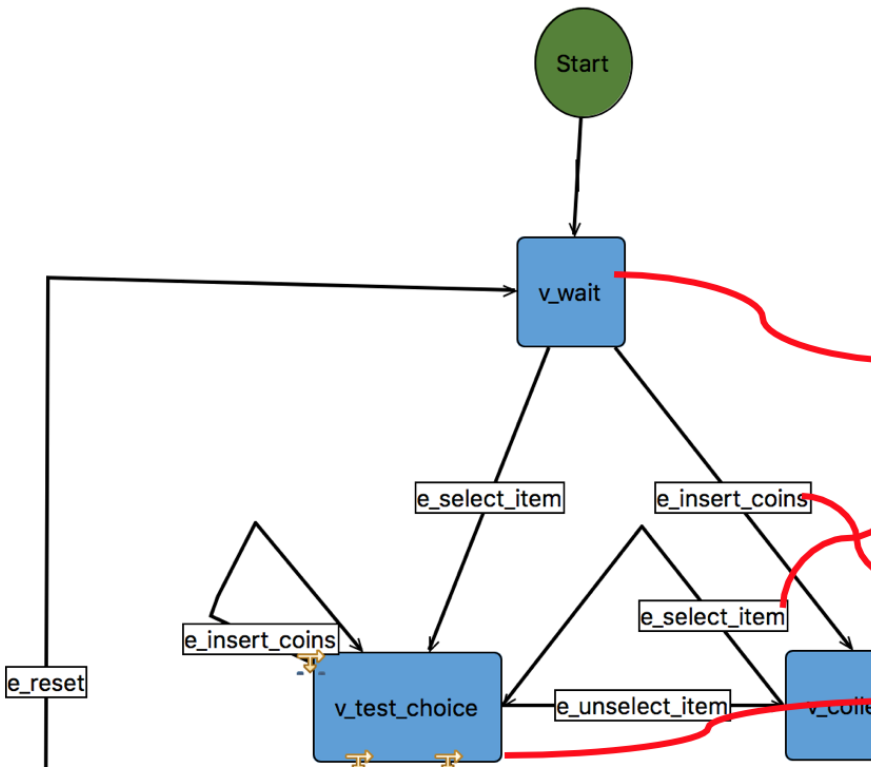
Onderzochte tools

Tools die we onderzocht hebben:

- GraphWalker
- TestCompass
- TorXakis
- SpecExplorer

GraphWalker

- Maakt code op basis van model



```

package com.company;

import java.nio.file.Path;

@Generated(value = "src/test/resources/com/company/DrinkDispenser.json")
@GraphWalker(value = "random(edge_coverage(100))", start = "Start", groups = { "default" })
public class DrinkDispenserImpl extends ExecutionContext implements DrinkDispenser {

    public final static Path MODEL_PATH = Paths.get("com/company/DrinkDispenser.json");

    @Override
    public void v_wait() {
        System.out.println("Executing:v_wait");
    }
    @Override
    public void e_select_item() {
        System.out.println("Executing:e_select_item");
    }
    @Override
    public void e_insert_coins() {
        System.out.println("Executing:e_insert_coins");
    }
    @Override
    public void v_test_choice() {
        System.out.println("Executing:v_test_choice");
    }
}
  
```

TestCompass

- Eenvoudig model maken
- Gebruik voor communicatie en verificatie

TESTCompass
2d. Login_0.3

✉
📖
?
☰

Project
Model
Test

Model Name

Description

Created At: 2020/10/23 9:57:01 AM
Last Modified: 2020/10/23 11:06:59 AM

Export Model

Export Model as PNG

Save as New Version

```

    graph TD
      Start((Start)) --> Login[Login on the website]
      Login --> U1{Fill in Username?}
      U1 --> U2{Fill in correct Username?}
      U1 --> U3{Fill in Password?}
      U2 --> U4{Fill in Password?}
      U2 --> U5{Fill in correct Password?}
      U3 --> U4
      U3 --> U5
      U4 --> E1[<Enter>]
      U4 --> E2[<Enter>]
      U5 --> E3[<Enter>]
      U5 --> E4[<Enter>]
      E1 --> M1[Message]
      E2 --> M2[Message]
      E3 --> M3[Message]
      E4 --> M4[Message]
      E3 --> E5[<Enter>]
      E5 --> L[Logged in]
      L --> End((End))
      M1 --> End1((End))
      M2 --> End2((End))
      M3 --> End3((End))
      M4 --> End4((End))
  
```

Zie ook: <https://youtu.be/NQ8lkqTU2gs>

Terms & Conditions | Privacy Policy | Contact UsCopyright © TestCompass 2020

TorXakis

- Tekstueel model – research
- Sterke methoden generatie:
 - Randomisatie
 - Parallelliteit

```

|--
TorXakis - Model Based Testing
Copyright (c) 2015-2017 TNO and Radboud University
See LICENSE at root directory of this repository.
-|)

TYPEDEF ProduceReport ::= ProduceReport { id, value :: Int } ENDDF
TYPEDEF MeasureReport ::= MeasureReport { id, value :: Int } ENDDF
TYPEDEF Correction ::= Correction { id, value :: Int } ENDDF

FUNCDEF sgn ( x :: Int ) :: Int ::=
  IF x > 0
  THEN 1
  ELSE IF x < 0
  THEN -1
  ELSE 0
  FI
FI
ENDDF

PROCDEF produce [ In_Material      :: Int
                  ; In_Correction   :: Correction
                  ; Out_Material    :: Int
                  ; Out_ProduceReport :: ProduceReport
                  ]( ) ::=
  In_Material ? id
  >>> produceLoop [In_Material, In_Correction, Out_Material, Out_ProduceReport] (id)
ENDDF

PROCDEF produceLoop [ In_Material      :: Int
                      ; In_Correction   :: Correction
                      ; Out_Material    :: Int
                      ; Out_ProduceReport :: ProduceReport
                      ]( id :: Int ) ::=
  (
    Out_Material ! id
    >>> In_Material ? nextId
    >>> EXIT ! nextId
  )
  |[ Out_Material ]|
  (
    ( Out_Material ! id >>> EXIT )
    ||| ( Out_ProduceReport ? pr [[ id(pr) == id ]] >>> EXIT )
  ) >>>
  ( In_Correction ? correction [[ id (correction) == id ]] >>> EXIT ? nextId :: Int )
  )
  >>> ACCEPT ? nextId :: Int IN
  produceLoop [In_Material, In_Correction, Out_Material, Out_ProduceReport] (nextId)
NI
ENDDF

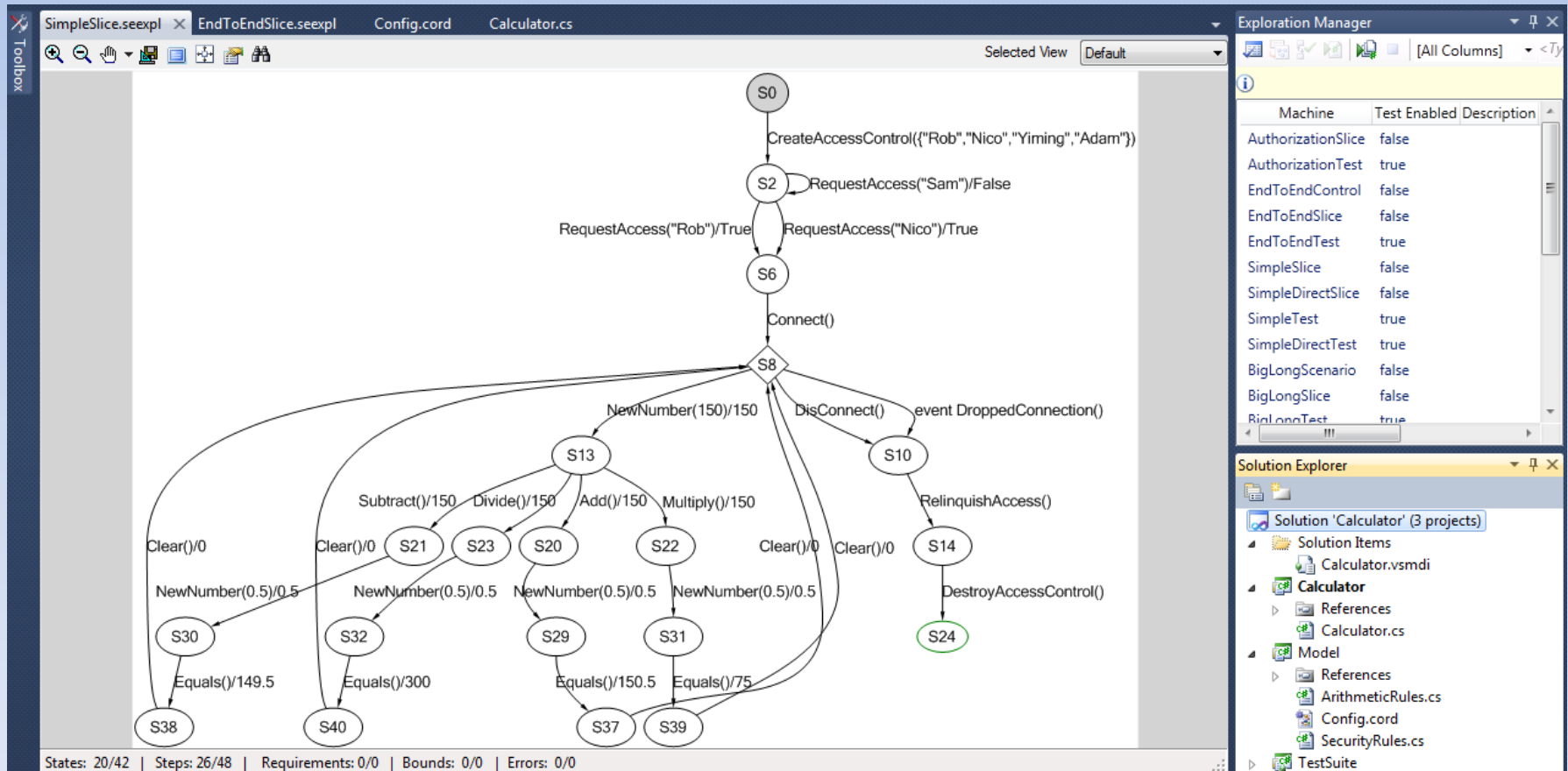
PROCDEF measure [ In_Material  :: Int
                  ; Out_Report  :: MeasureReport
                  ; Out_Material :: Int
                  ]( ) ::=
  In_Material ? id
  >>>>>> ControlLoopModel.txs Top (1,0) Git-feat/test-selection-replay (TorXakis mode company Helm yas Projectile[TorXakis] FlyC-)

```

SpecExplorer



- Model op basis van toestandsdiagram



Lessons learned vanuit pilots en praktijk



- Veel voordelen MBT
- Toch wordt MBT mondjesmaat gebruikt
- Wat zijn daarvan de redenen?
- Hoe kun je MBT goed gebruiken in de praktijk?

Voordelen MBT

- Tijd- en kostenbesparing
 - Een wijziging hoeft alleen in het model aangepast te worden
 - Testgevallen worden gegenereerd vanuit model
 - Genereren en evt. automatisch uitvoeren van testgevallen
- Kwaliteitsverbetering
 - Verhogen testdekking
 - Beheersen complexiteit
 - Vroeg fouten opsporen met model
- Verbetering van de communicatie
 - Iedereen gebruikt hetzelfde model
 - Model is voor iedereen begrijpelijk



Waarom wordt MBT weinig gebruikt?

- Te veel doelen willen bereiken met MBT
- Modellen worden niet op het juiste abstractieniveau toegepast
- MBT wordt toegepast op gebieden waarop weinig winst is te halen
- Nog stappen te maken met tooling

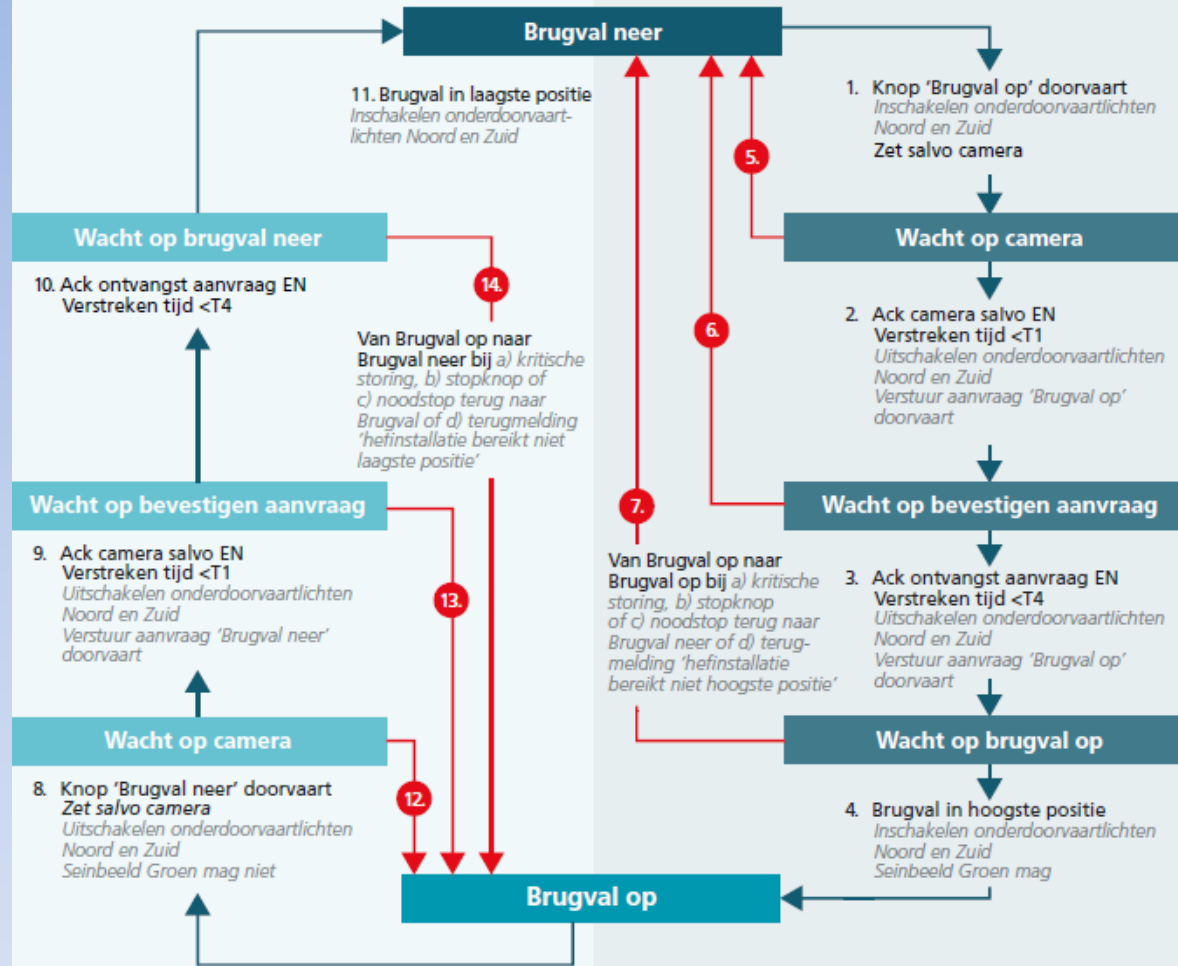
Te veel doelen willen bereiken



Niet het juiste abstractieniveau

- Eenvoudige modellen voegen weinig toe
- Meerdere doelen worden beoogd met model
- Complexiteit van werkelijkheid moeilijk in model te vatten - Modelleren is moeilijk

- Voorwaarden Brugval neer:**
- Geen kritische storing brugval
 - Geen veiligheidsfunctie actief brugval
 - Spoor succesvol geblokkeerd
 - Landverkeer succesvol geblokkeerd
 - Herinstallatie brugval beschikbaar
 - Brugval (nog) niet vergrendeld in eindstand op



- Voorwaarden Brugval op:**
- Geen kritische storing brugval
 - Geen veiligheidsfunctie actief brugval
 - Scheepvaart succesvol onderbroken
 - Herinstallatie brugval beschikbaar
 - Brugval (nog) niet vergrendeld in eindstand op

UC22: Brugbeweging sluiten doorvaart

- 56. Van Brugval neer naar Brugval op
 a) kritische storing, b) timeout of
 c) noodstop terug naar Brugval neer
- 12/13 Van Brugval op naar Brugval neer
 a) kritische storing, b) timeout of
 c) noodstop terug naar Brugval op

UC21: Brugbeweging openen doorvaart

Maak modellen op het juiste abstractieniveau



- Geef het model een duidelijke focus
- Bepaal juiste abstractieniveau om winst te bereiken
- Houd rekening met modelleerkennis van testers; train evt. testers in modelleren

Kies juiste toepassingsgebied

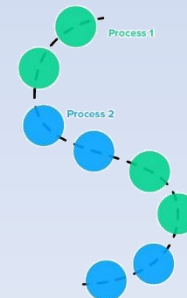
- MBT wordt toegepast waar weinig winst is te behalen
- Kies onderdelen waar MBT waarde toevoegt

Bijvoorbeeld:

- Hoge productrisico's
- Specifieke aspecten die lastig met “normale” testgevallen te vatten zijn, zoals:
 - Parallelliteit
 - Timing
 - Reliability

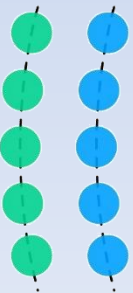
Risk

Concurrency



Parallelism

vs



Tooling

- Niet altijd grafisch interface om te modelleren
- Niet altijd complexe modellen mogelijk
- (Vaak) maar één modelleertechniek
- Aansluiting MBT tool op System Under Test (SUT) is wisselend
- Beperkte rapportagemogelijkheden

Tooling

- Wachten
- Zelf ontwikkelen
- Hybride vorm: Verschillende tools gebruiken of manueel/automatisch

Adviezen

- Kies een duidelijk **gefocust doel**
- Zorg voor voldoende kennis/ervaring om de **benodigde abstractie** in het model te bepalen
- Hou de **scope beperkt** en richt je op specifieke delen of aspecten van het systeem
- Zorg dat **toolkennis** beschikbaar is

Even voorstellen



Rachid Kherrazi

Rachid Kherrazi CTO AKKA Netherlands
20 jaar ervaring in ICT:



ASML

ALTRAN

AKKA
PASSION FOR
TECHNOLOGIES

Actief lid TestNet community
Trekker MBT werkgroep
Ervaring met Security Testing, Model Based
Testing, Artificial Intelligence en Test
automatisering



Eduard Hartog

Eduard Hartog
Senior Consultant Improve Quality Services BV
30 jaar ervaring in ICT



Top Systems
POWERED BY STEFANINI

ICT+
Improve
QUALITY SERVICES

Actief lid MBT werkgroep
Van ontwikkelaar t/m project manager
Laatste 10 jaar in multidisciplinaire projecten
Test Management

Demo met Spec Explorer Look and Feel

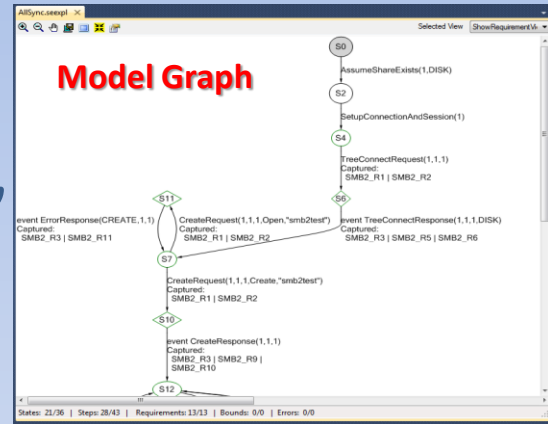
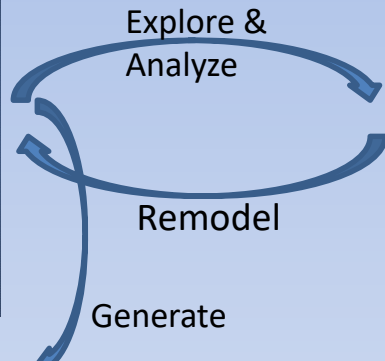


```

#Region Tree Connect
/// <summary>
/// Describes a tree connect request.
/// </summary>
[Action]
static void TreeConnectRequest(int sequenceId, int creditReq, int treeId)
{
    Contracts.Requires(treeIds.Count == treeIdsInFlight + 1);
    CheckRequest(sequenceId, creditReq);
    inflight.Add(new TreeConnectRequest(sequenceId, shareId));
    treeIdsInFlight++;
}

/// <summary>
/// Describes a tree connect response.
/// </summary>
[Action]
static void TreeConnectResponse(int sequenceId, [Domain("CreditDomain")
int treeId, ShareType shareType])
{
    TreeConnectRequest request =
        (TreeConnectRequest)CheckResponse(Command.Values.TREE_CONNECT,
        Capture(5, "tree connect request must be responded"));
    Requires(share.ContainsKey(request.shareId) && share[request.shareId] == 6, "only existing share should have successful share share = share[request.shareId];
}
    
```

**# Model
(or other .Net
Language)**



Test Suite

Configured Test Suite

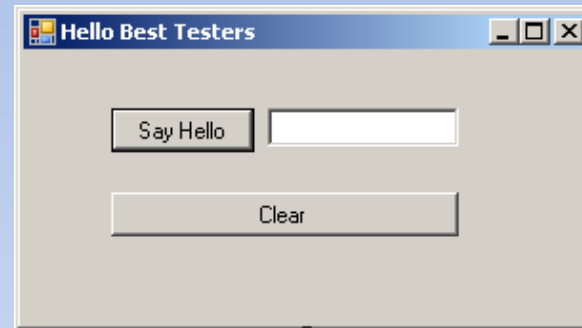
Execute

<http://www.nunit.com/>

Nunit

Passed: 1 Failed: 0 Errors: 0 Inconclusive: 0 Invalid: 0 Ignored: 0 Skipped: 0 Time: 174.07%

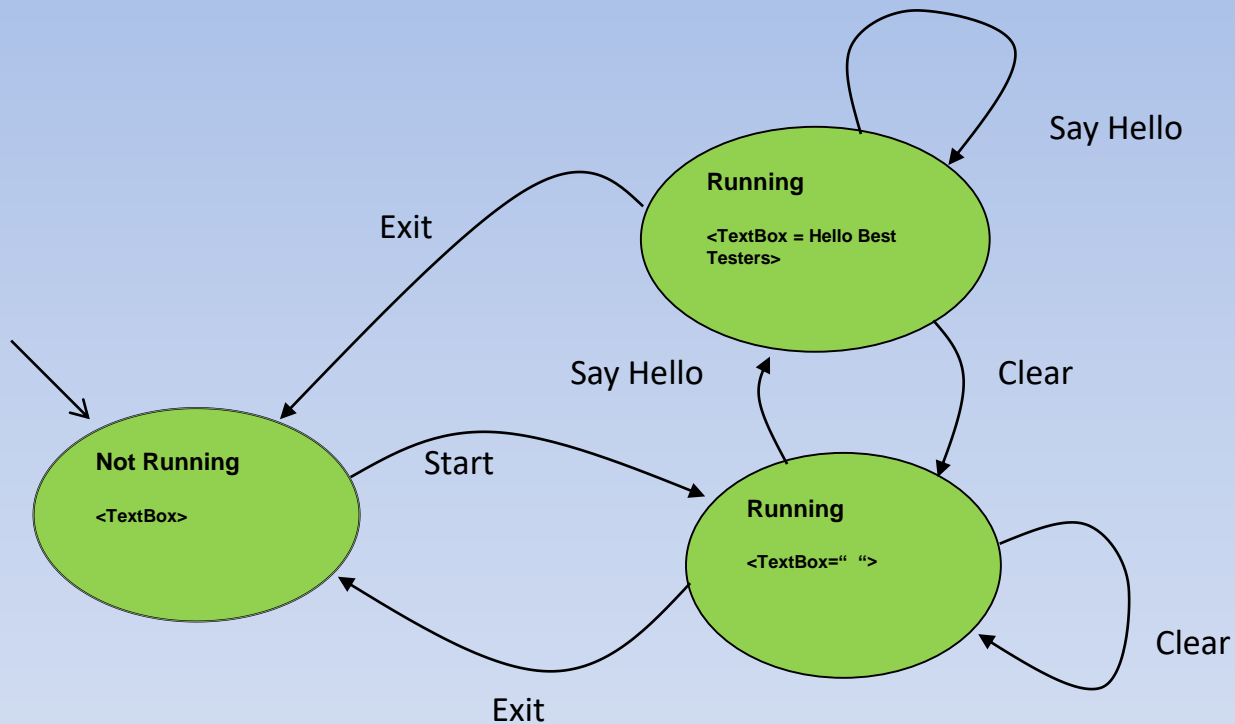
Example: Hello Best Testers



A typical use case:

#	Action	Expected result
1	Start application	App. started
2	Press Say Hello button	Text box = "Hello Best Testers"
3	Press Clear button	Text box = " "
4	Exit application	App. closed

Model: a simple FSM



States, Variables, Actions, pre and post condition

Textual Model: C# Model

```
namespace DemoModel
{
    public enum State
    {
        NotRunning,
        Running
    }

    static class ModelProgram
    {
        static State appStatus = State.NotRunning;
        static string TextBox=" ";

        [Probe]
        public static string Satatus()
        {
            return string.Format("appStatus:{0},TextBox:{1}", appStatus, TextBox);
        }

        [AcceptingStateCondition]
        static bool InitialStatus()
        {
            return appStatus == State.NotRunning ;//&& TextBox == " ";
        }
    }
}
```

States declaration

Initialization

Optional, just for visualization in graph

When to stop in case of test case

Textual Model: C# Model

```
[Rule(Action = "StartApp()/result")]
static Implementation.State StartApp()
{
    Condition.IsTrue(appStatus == State.NotRunning);
    Requirement.Capture("requirement1.1:StartApp");
    appStatus = State.Running;
    return (Implementation.State)appStatus;
}
```

```
[Rule(Action = "StopApp()/result")]
static Implementation.State StopApp()
{
    Condition.IsTrue(appStatus == State.Running);
    Requirement.Capture("requirement1.2:StopApp");
    appStatus = State.NotRunning;
    TextBox = " ";
    return (Implementation.State)appStatus;
}
```


```
}
```

```
}
```

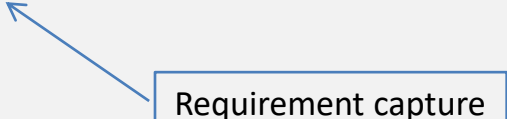
Textual Model: C# Model

```
[Rule(Action = "ClearText()/result")]
static string ClearText()
{
    Condition.IsTrue(appStatus == State.Running);

    TextBox = "";
    Requirement.Capture("requirement1.3:ClearText()/result");
    Console.WriteLine("Clear text");
    return TextBox;
}
```



Action declaration



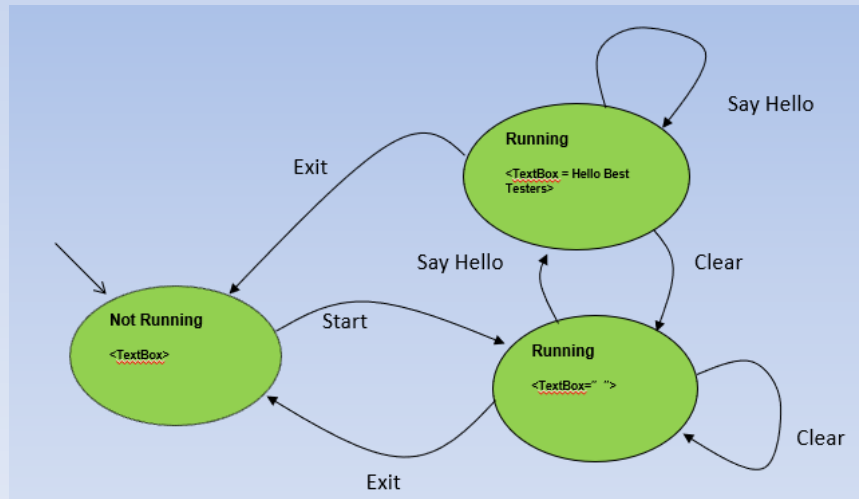
Requirement capture

```
[Rule(Action = "SayHallo(A)/result")]
static string SayHallo(string A)
{
    Condition.IsTrue(appStatus == State.Running);
    Requirement.Capture("requirement1.4:SayHallo");

    TextBox = A;
    return TextBox;
}
```


Live demo

- Live demo spec explorer Say Hello voorbeeld



Wat zijn onze vervolgstappen?

1. Samenwerken met werkgroep AI en Testen
2. Vervolg onderzoek MBT onderwerpen

Hoe kan AI MBT helpen?

- Test cases/test scenario's genereren vanuit een MBT model met behulp van AI
- Test cases/test scenario's clusteren met behulp van AI in geval van veel test cases en tijd restrictie
- Re-engineering vanuit log data of observaties om MBT model te reconstrueren (kan gebruikt worden voor regressie test)
- Test data selectie in model based testen met behulp van AI
- AI testen met AI, Model based observatie testen)
- MBT model vs AI model

Wat zijn onze vervolgstappen?

1. Oproep: iedereen is welkom om mee te helpen
2. Interesse? Neem dan contact via Rachid.kherrazi@outlook.com

Vragen

