



Automatisch
testen via de user
interface,
een nieuwe
aanpak

Tanja E.J. Vos

25 november 2017, Amsterdam

Test*



UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA

Software is overal

In short, software is eating the world

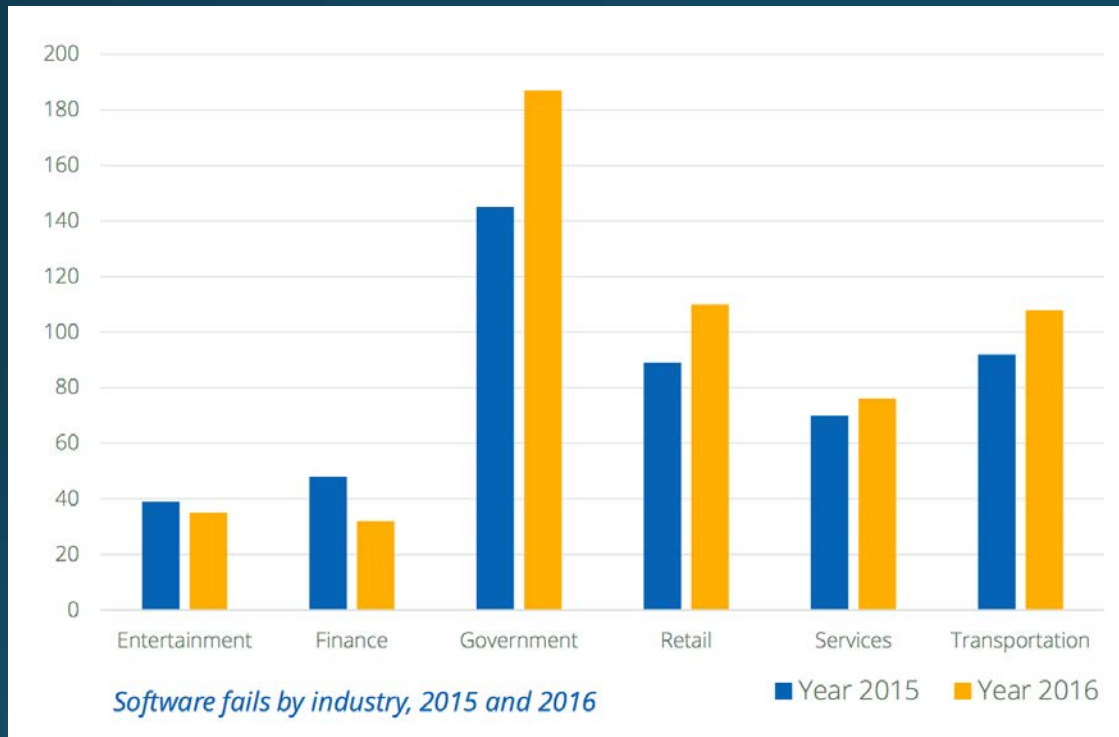
— Marc Andreessen —

[Home](#) > [Nieuws ICT](#) > Elk bedrijf wordt een softwarebedrijf in 2017

Elk bedrijf wordt een softwarebedrijf in 2017

Het ziet ernaar uit dat 2017 een dynamisch jaar gaat worden. Robots, IoT, blockchain, artificial intelligence... het zijn zomaar wat trends die door vakgenoten worden gesignaleerd. De vraag is natuurlijk: gaan deze ook doorzetten en krijg ik er in de praktijk veel mee te maken? Ik heb daarom eens de vijf ontwikkelingen op een rij gezet die volgens mij komend jaar in ieder geval voor een IT-dienstverlener van belang gaan zijn.

Falende software 2015 en 2016



- 4.4 miljard mensen
- > 1.1 biljoen \$



Software is onbetrouwbaar en de gevolgen van falende software worden steeds maar erger...



The Coming Software Apocalypse



Onderwijs

Cultuur van kwaliteitsdenken

Onderzoek

Test automatisering **Test**^{*}ar.org

TESTOMAT
PROJECT
The Next Level of Test Automation

90 %



Testen in het onderwijs

- Eerder!
- Overal! Als rode draad
- Geen test? Geen voldoende!
- Leuk maken!
- Testen is niet saai..... testen is uitdagend!



Onderzoek

Test automatisering

Test^{*}ar.org

Automatisch testen
op Grafische User
Interface niveau

TESTOMAT
PROJECT
The Next Level of Test Automation

Wat is **T**est* ?

- 
- Automatisch testen op user interface niveau
 - UI is waar alles samenkomt
 - Testen vanuit het perspectief van de user
 - Bijna alle software heeft een UI

Waar komt **T**est* vandaan?



Wat is **T**est* niet ?

- Niet capture/replay
- Niet visual testing
- Geen scripts



Welcome! [FAQ](#) [Log In or Sign Up](#) [Join My shopping cart](#) [Join My shopping list](#) [Join My shopping list](#)

[Offers](#) [New](#) [IDEAS](#) [Living room](#) [Bedroom](#) [Bath](#) [Kitchens](#) [Dining](#) [Textiles](#) [Business](#) [Baby & Kids](#) [Outdoor](#) [Summer](#) [Back to college](#) [All](#)

[Home](#) / [Living room](#) / [Bookcases](#)

[View more images](#)




BILLY
Bookcase, white
\$29.99

The price reflects selected options

Article Number: 802.638.32

★★★★★ 5.0 (1) [Write a review](#)

⚠ WARNING: TIP-OVER RISK! This furniture must be permanently fixed to the wall. [Read more here.](#)

Size
15 3/4x11x41 3/4"

Color
white

Why we like it:
It is estimated that every five seconds, one BILLY bookcase is sold somewhere in the world. Pretty impressive considering we launched BILLY in 1979. It's the booklovers choice that never goes out of style.
[Everything about this product](#)

Coordinating Products

+ 

[View all coordinating products](#)

Check stock at your local store
Choose

Store selection may vary and prices may differ from those online.

[Share](#) [Pin it](#) [G+1](#) [Like 27](#) [Share](#)

Color



More Models



[Show all \(17 more\)](#)

Begin "Search for a Billy bookcase"

StartWeb "http://www.ikea.com/us/en"

Click

Write "billy[ENTER]"

End

Begin "View the details"

Click



End

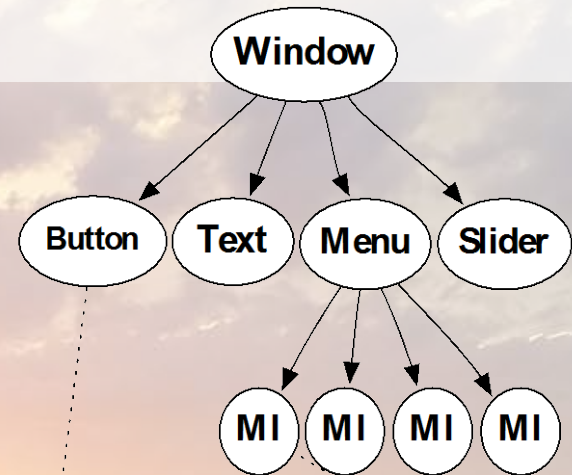
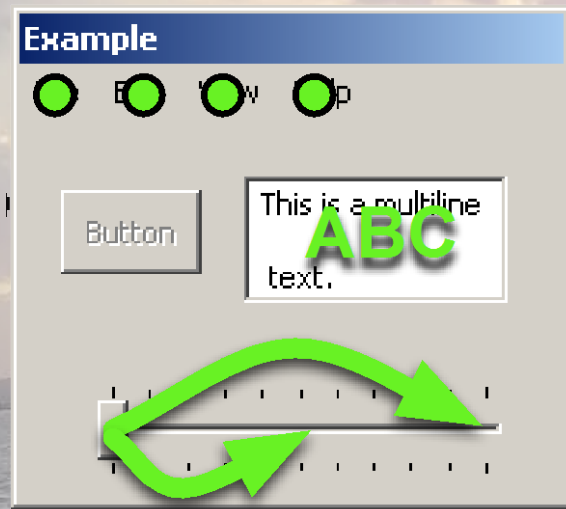
Begin "Check the price"

Check



End

T*est is scriptless

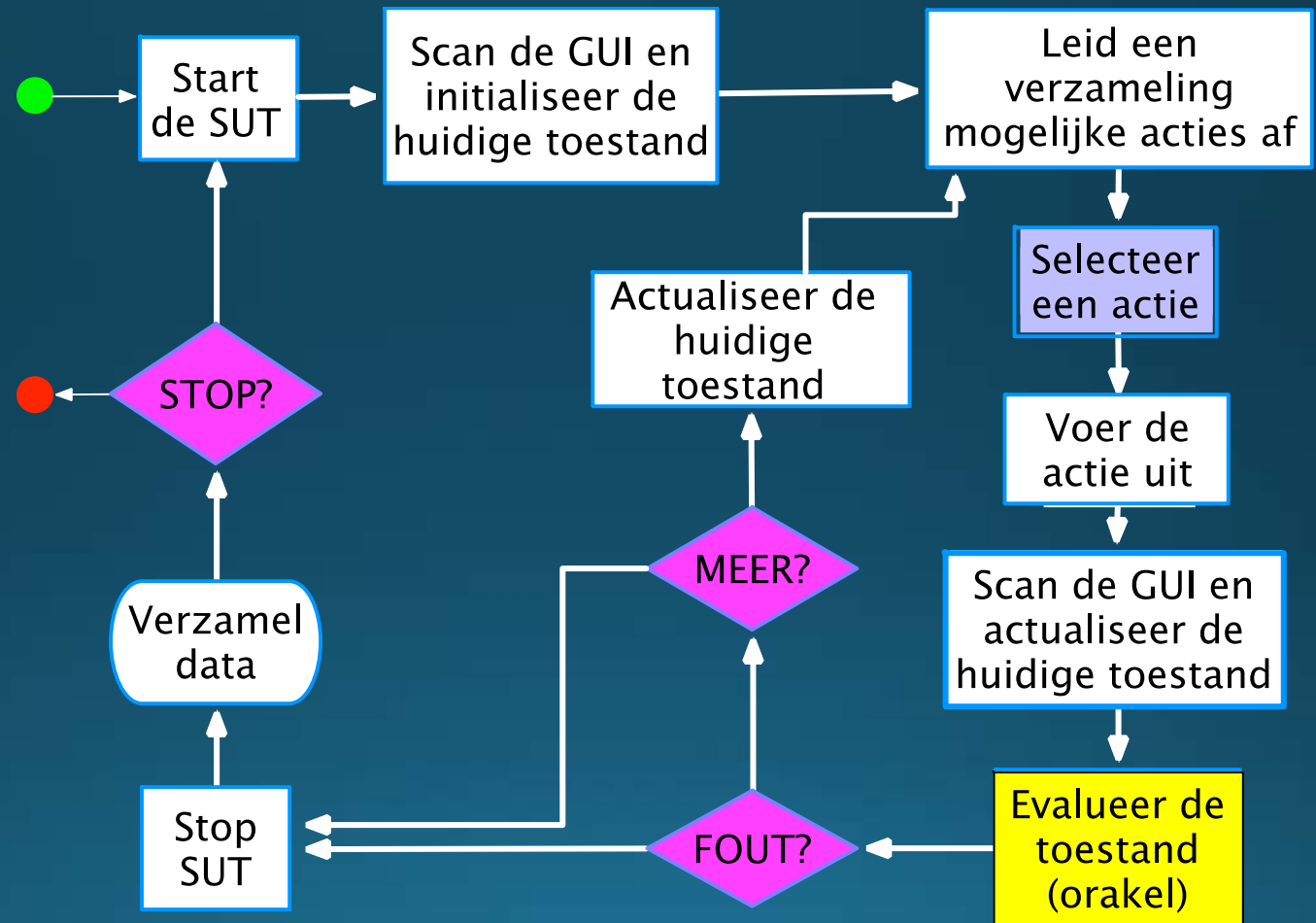


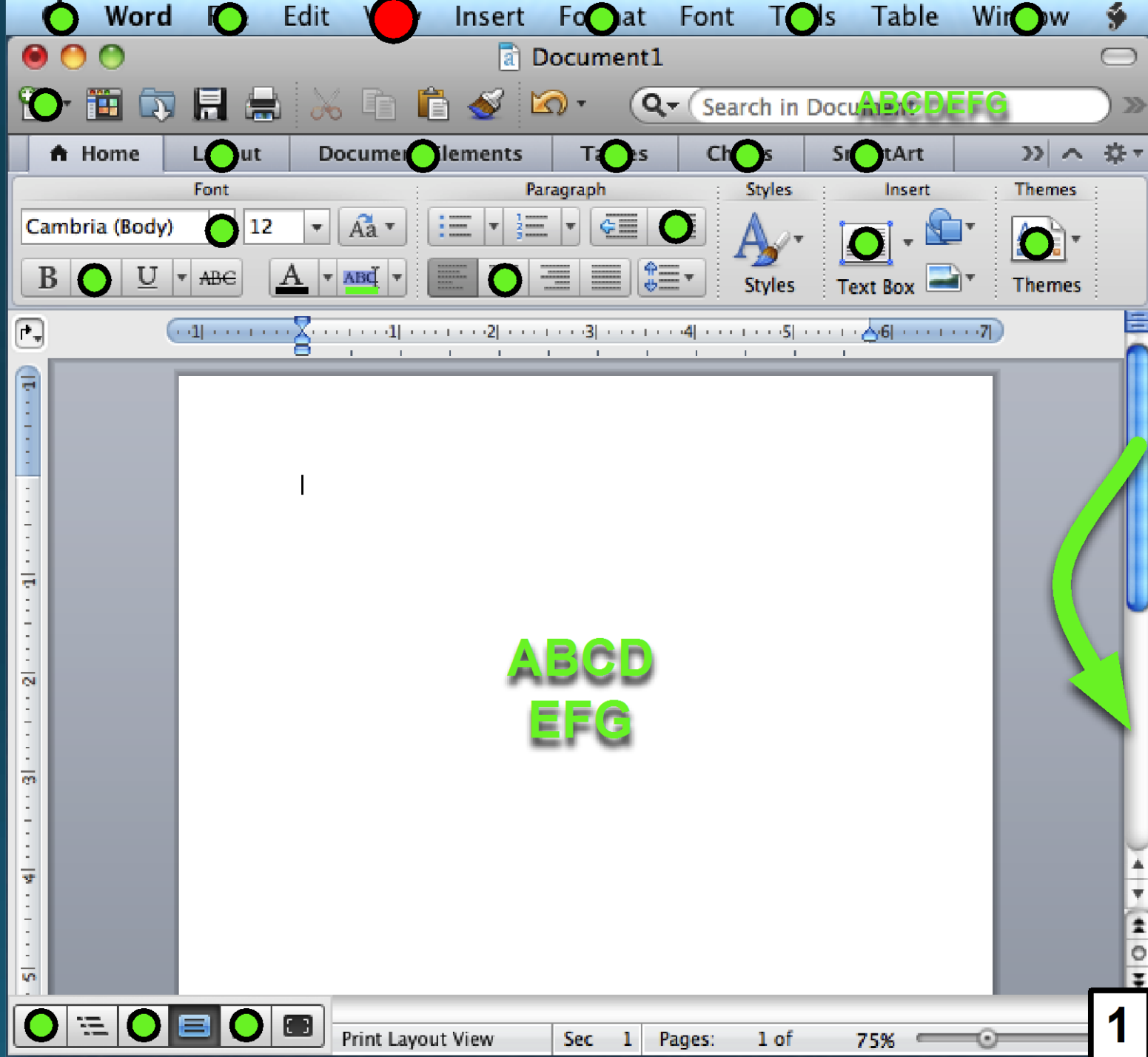
type: TButton
title: "Button"
enabled: false
hasFocus: true
rect: [15, 25, 65, 55]
...

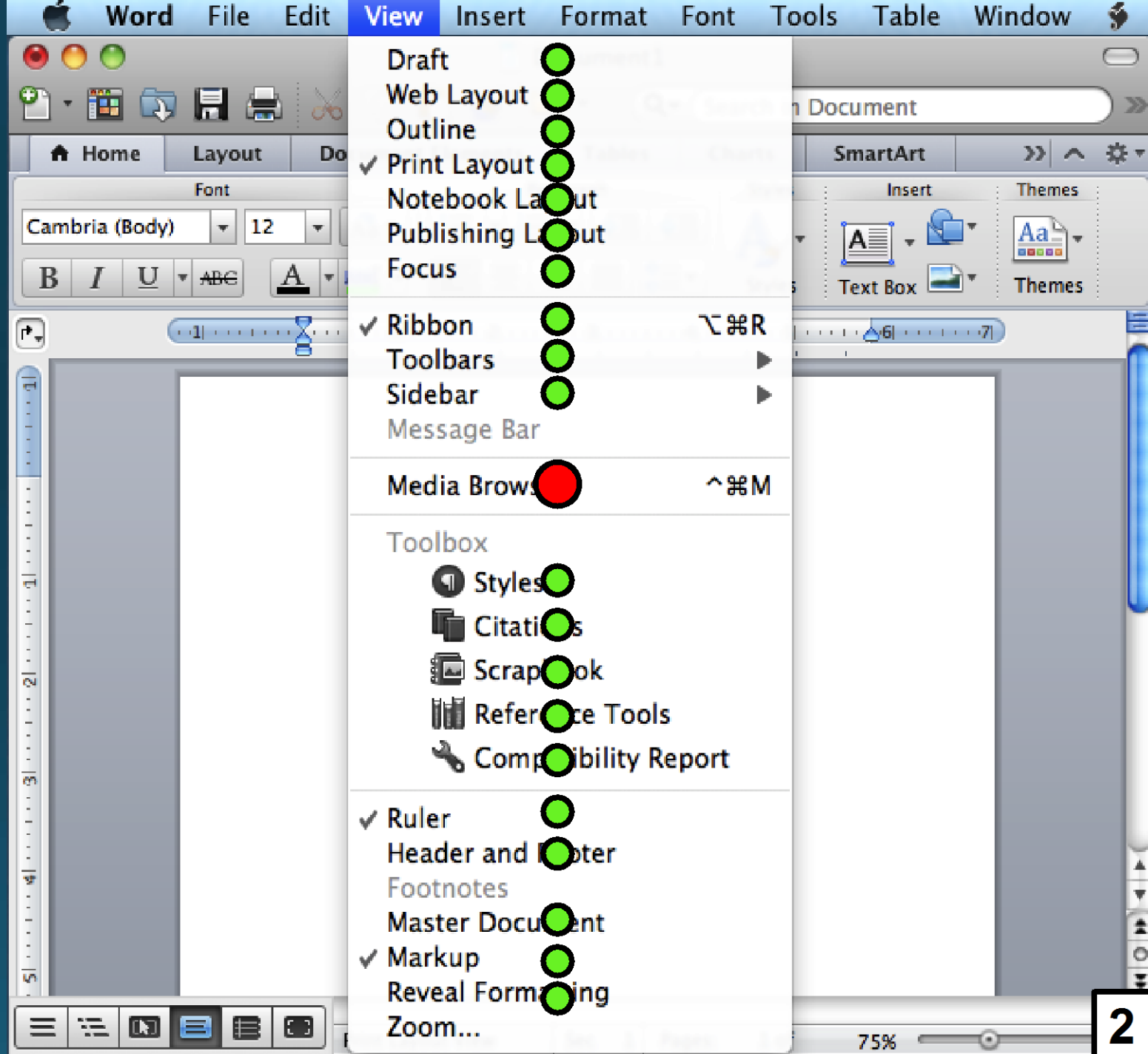
type: TMenuItem
title: "File"
...

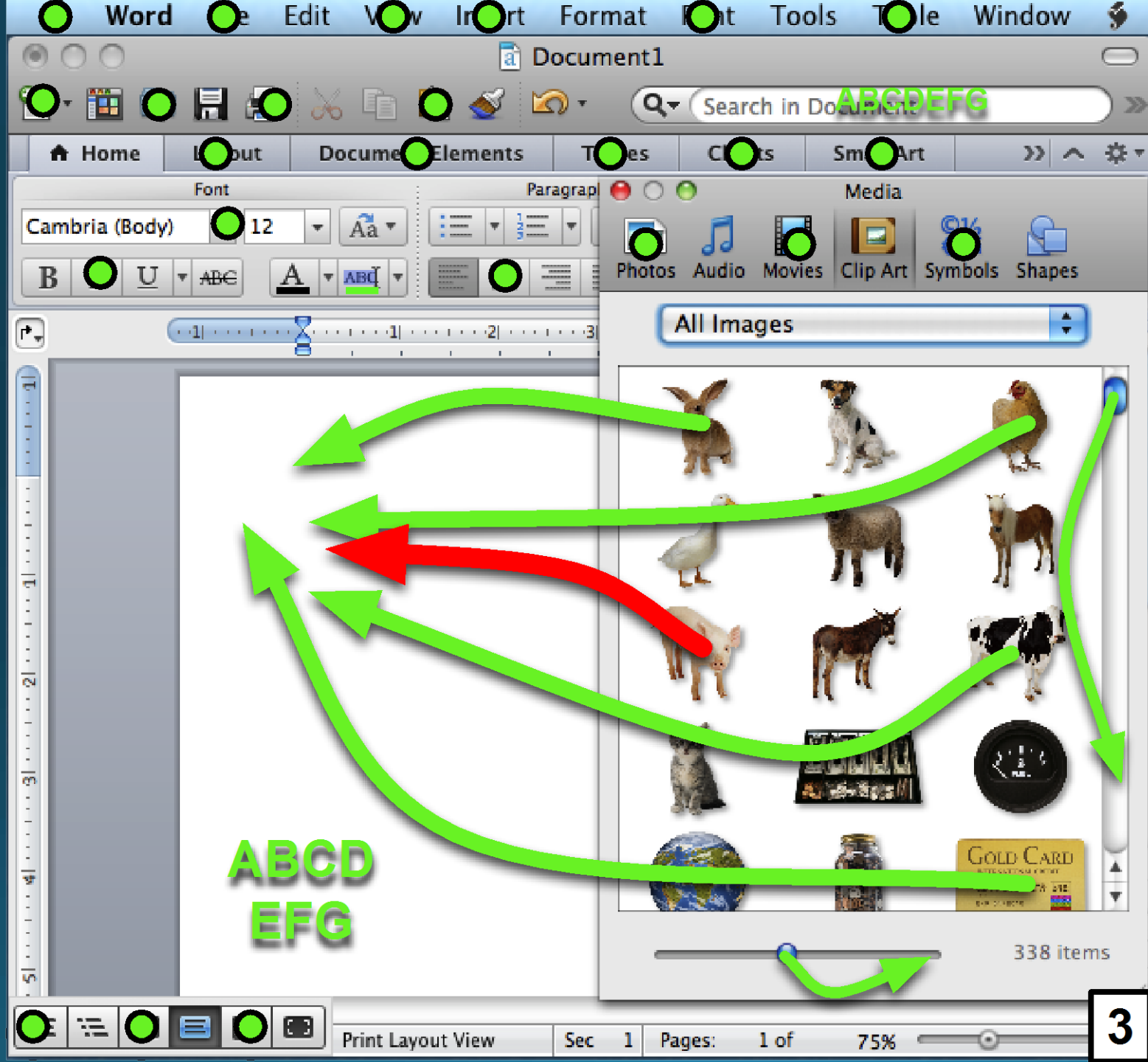
- Genereer sequences of test
- On the fly
- Gebruik een afgeleid model van de GUI

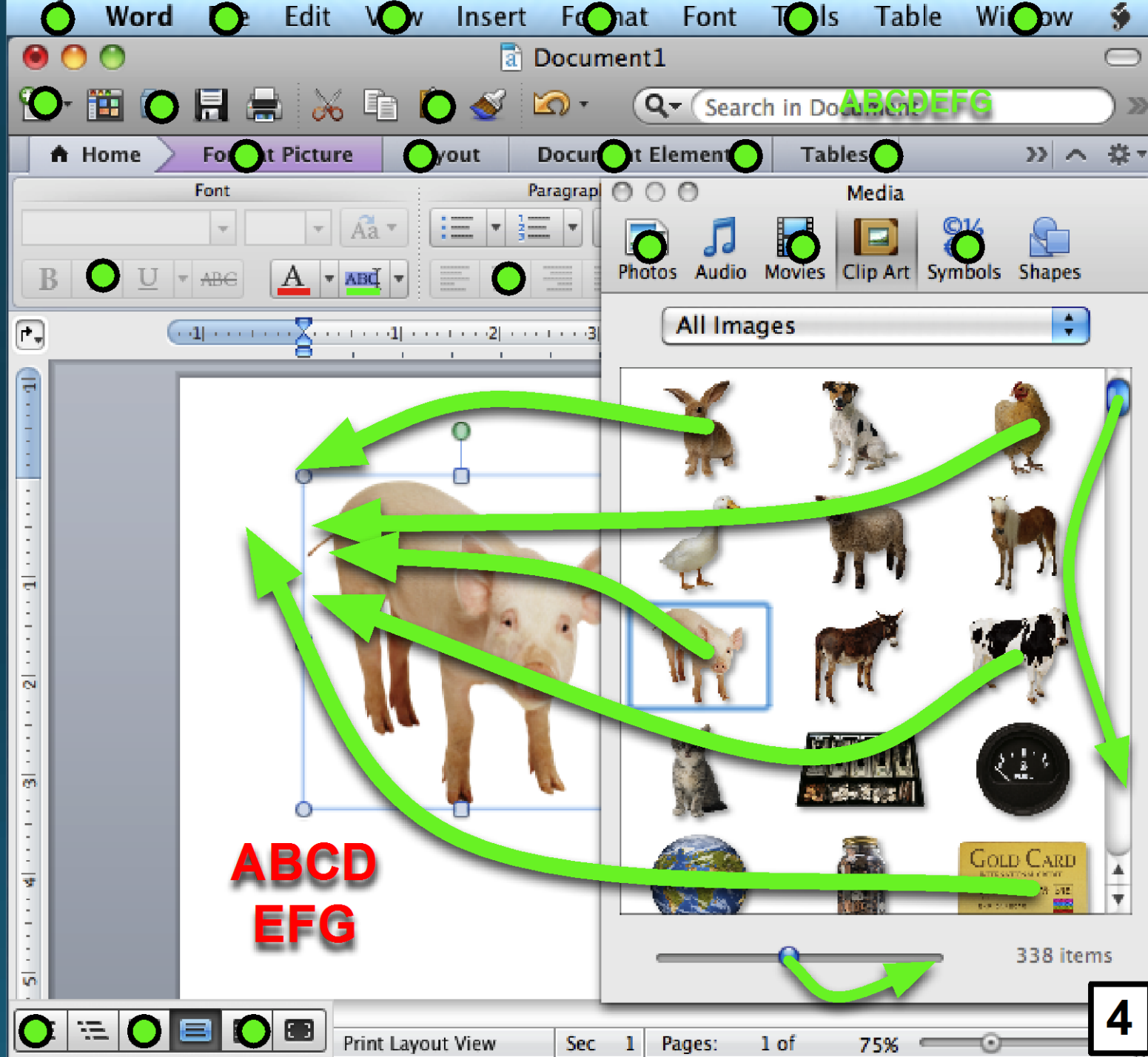
Hoe werkt TESTAR?















Microsoft Error Reporting



Microsoft Word has encountered a problem and needs to close. We are sorry for the inconvenience.

If you were in the middle of something, the information you were working on might be lost. Microsoft Word will attempt to recover your work.

☒ Recover my work and restart Microsoft Word

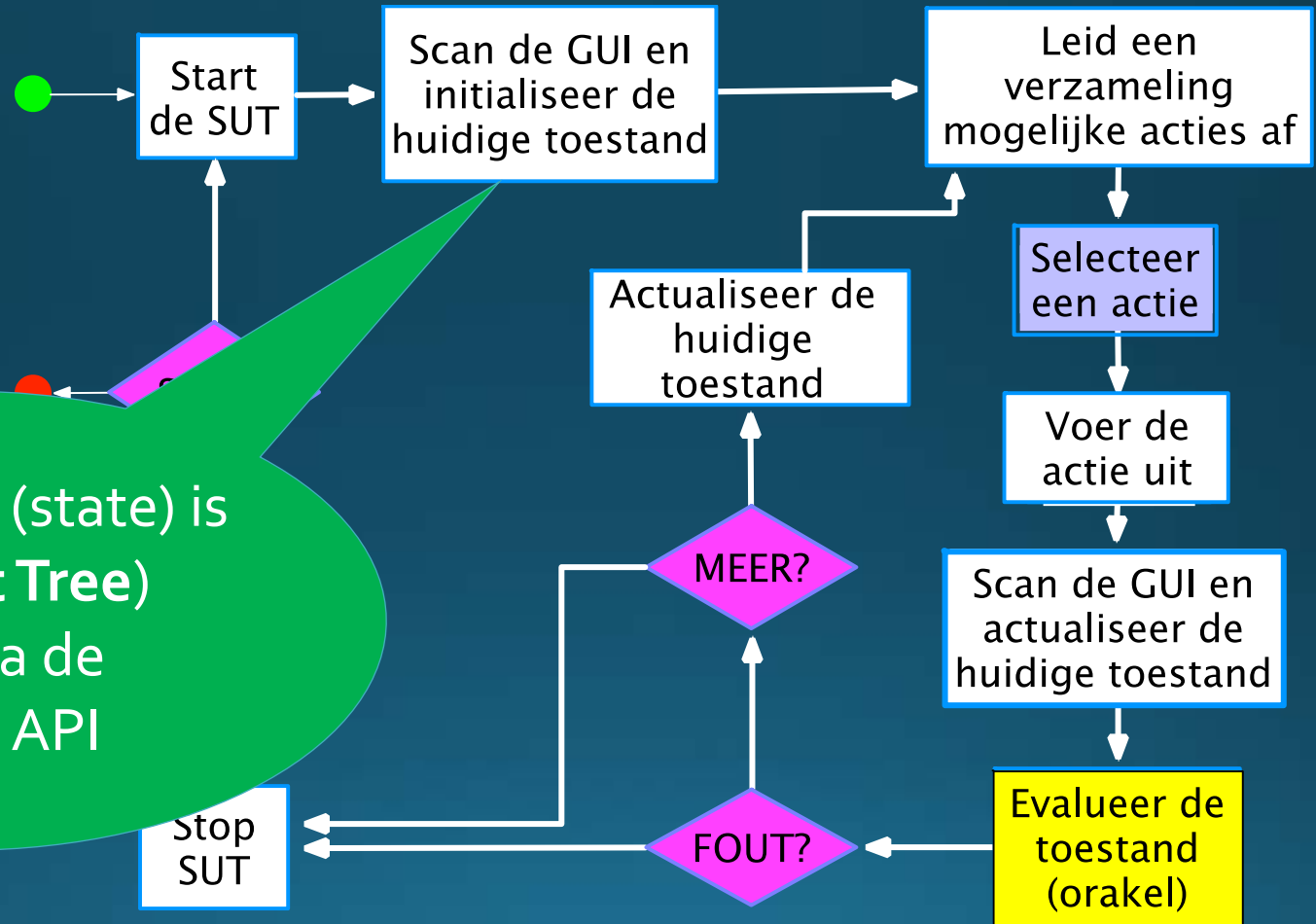
Please tell Microsoft about this problem.

Microsoft Word created an error report that you can send to help us improve Microsoft Word. We will treat this report as confidential and anonymous.

[More Information](#)

[Send](#)

[Don't Send](#)



De toestand (state) is een (**Widget Tree**) verkregen via de Accessibility API

Widget trees

Example

File Edit View Help

Button

This is a multiline
text.

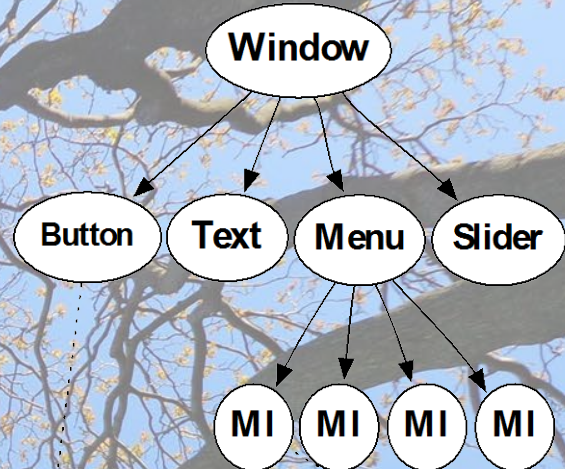


Example

File Edit View Help

Button

This is a multiline
text. **ABC**



type: TButton
title: "Button"
enabled: false
hasFocus: true
rect: [15, 25, 65, 55]
...

type: TMenuItem
title: "File"
...

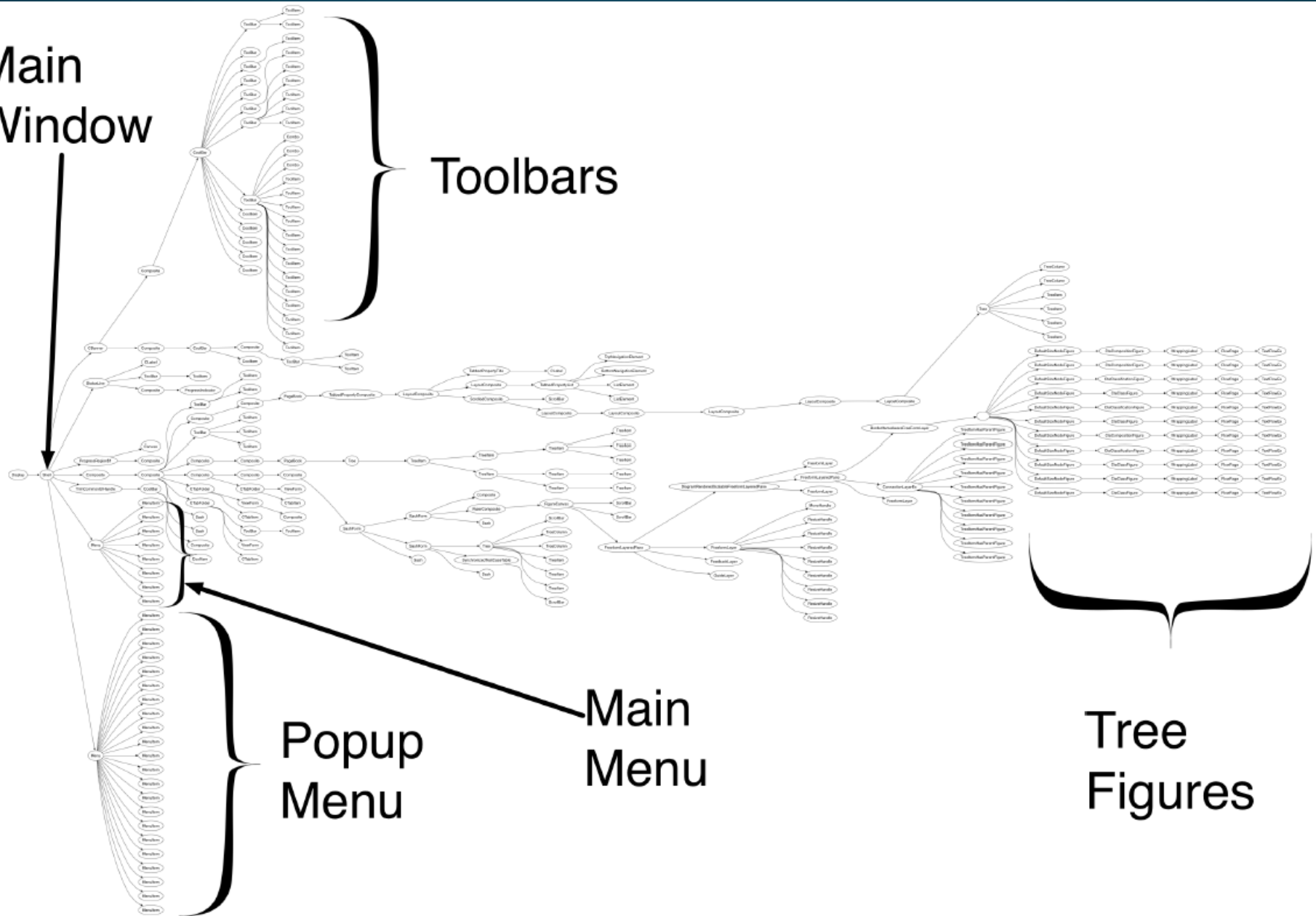
Main Window

Toolbars

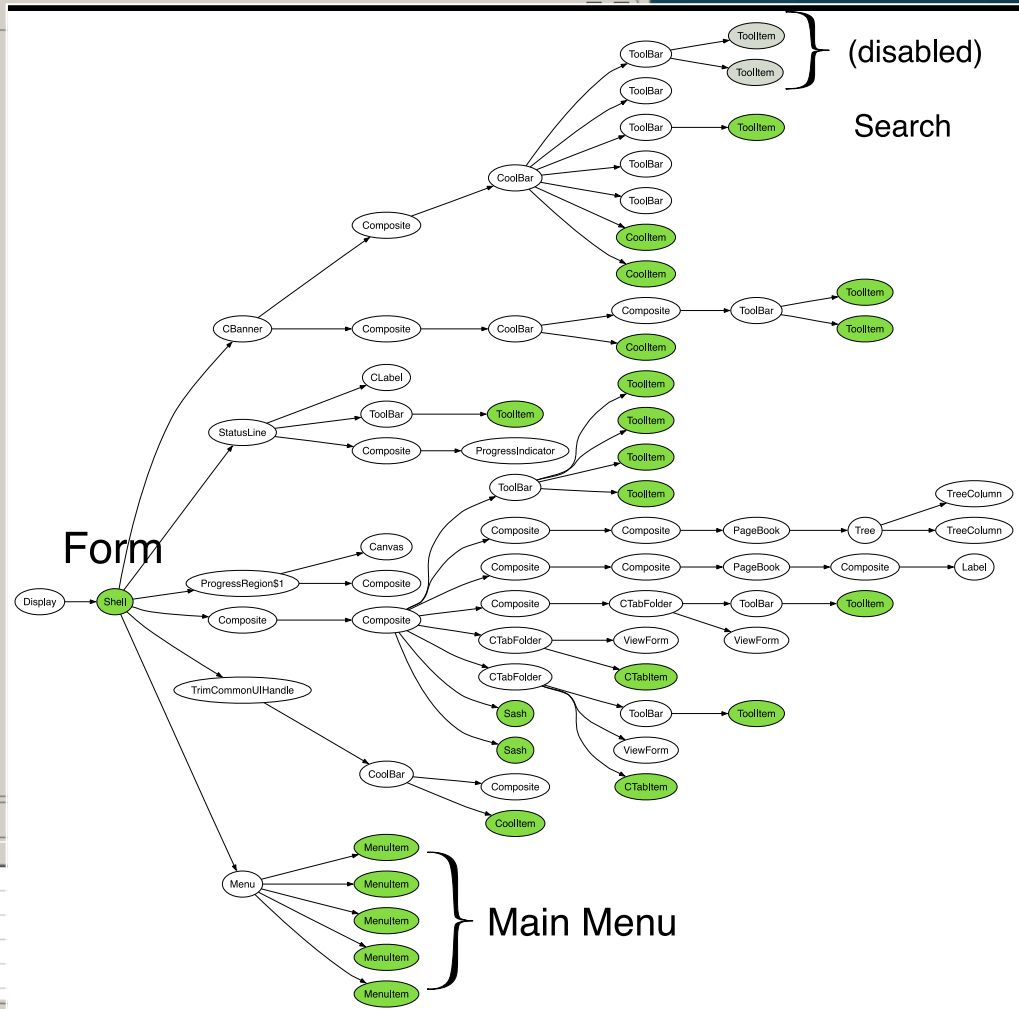
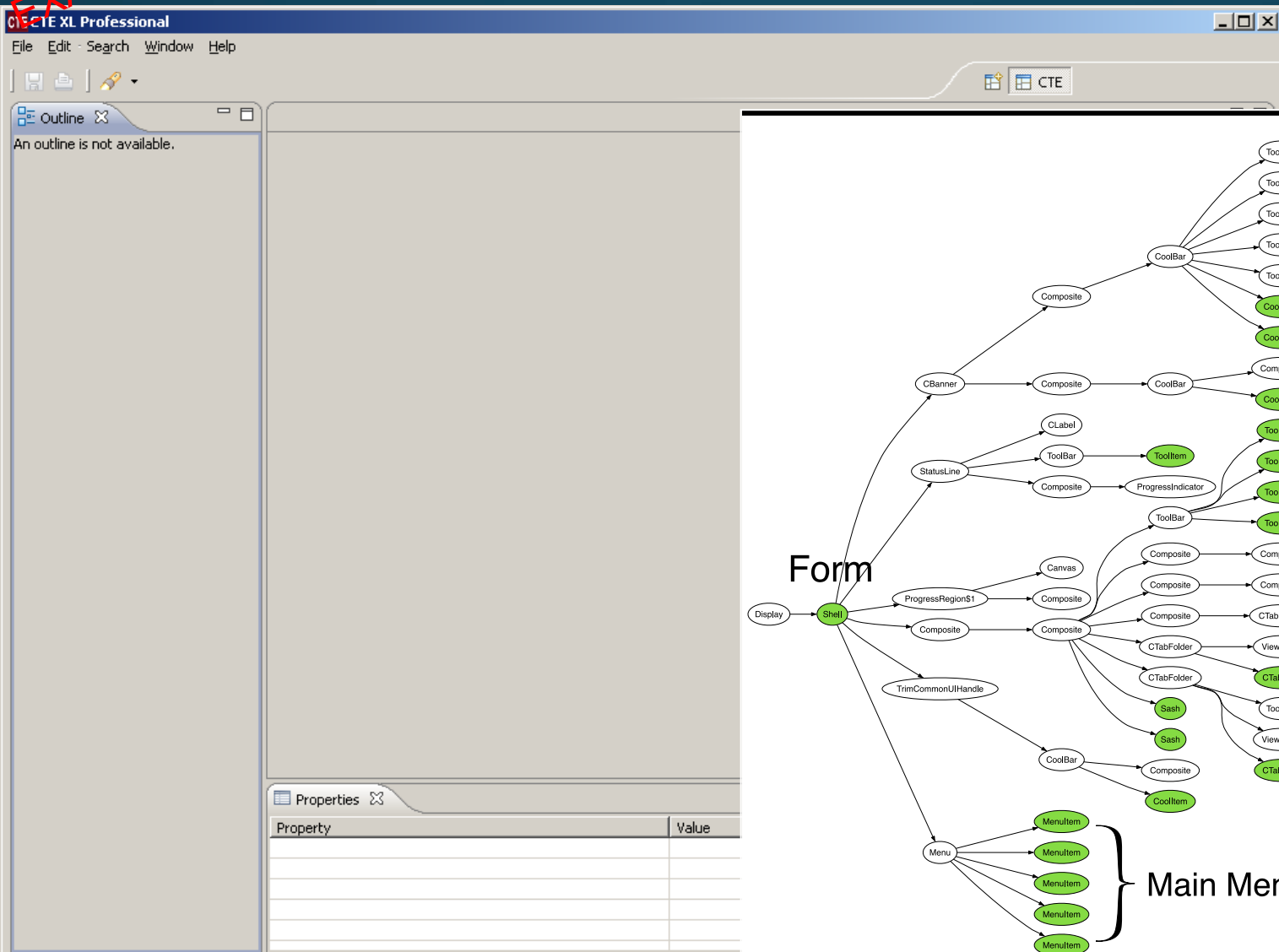
Popup Menu

Main Menu

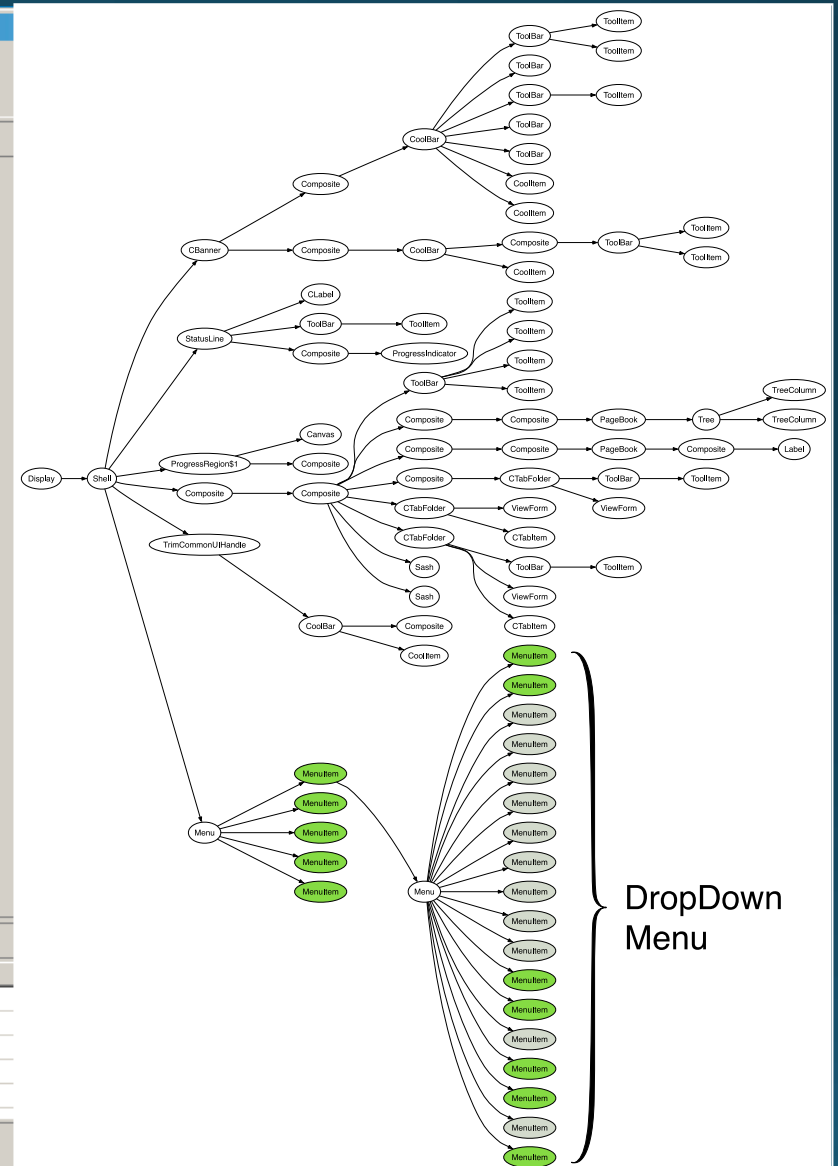
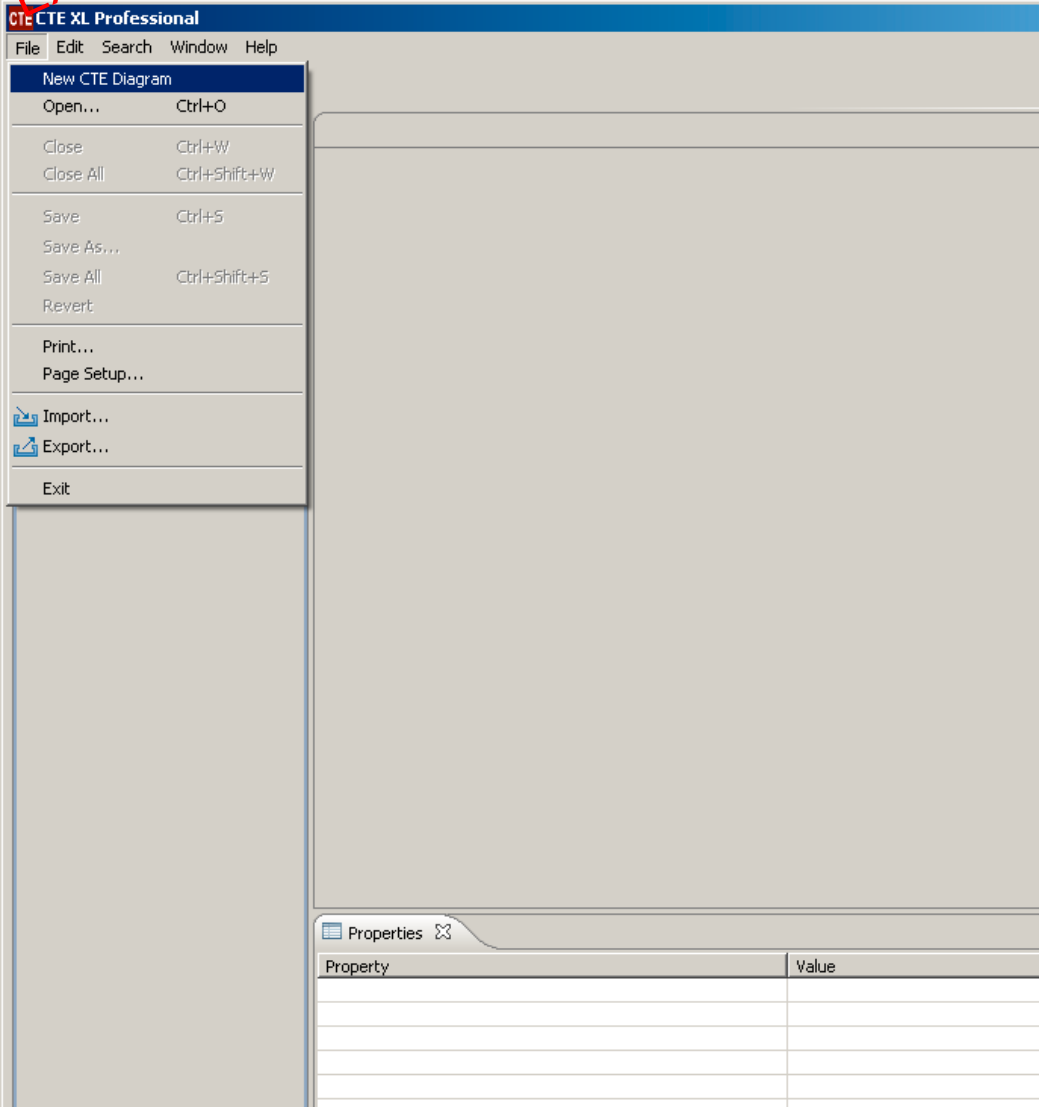
Tree Figures



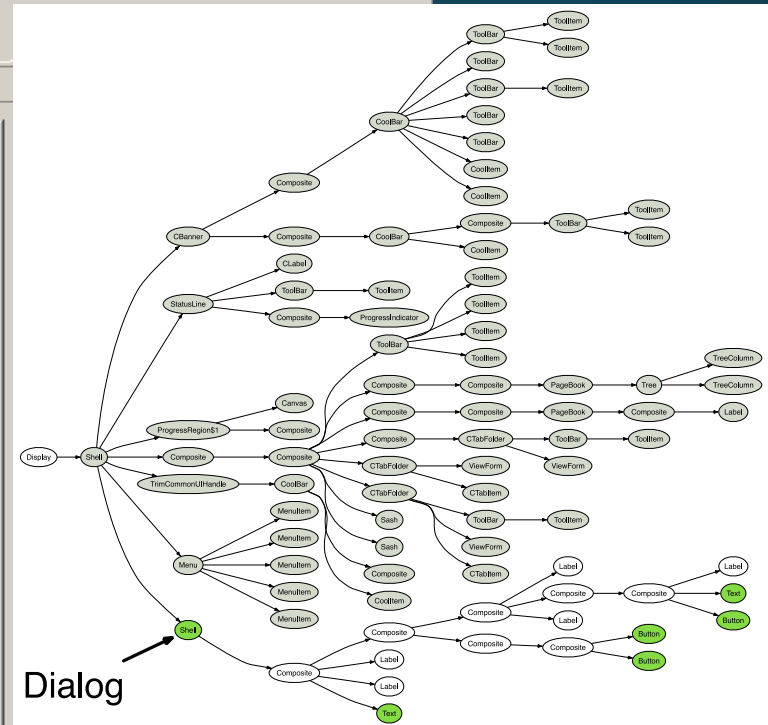
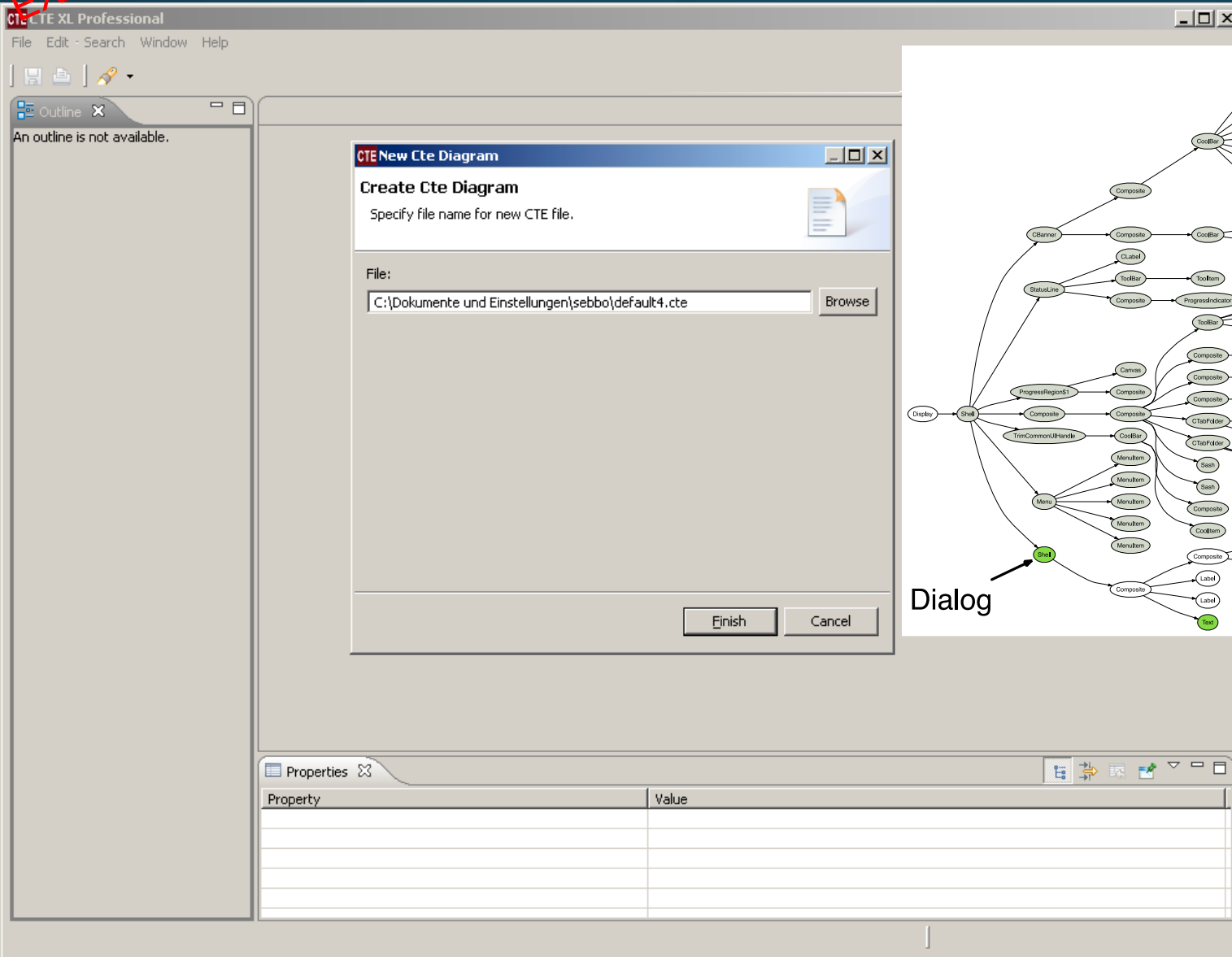
EXAMPLE



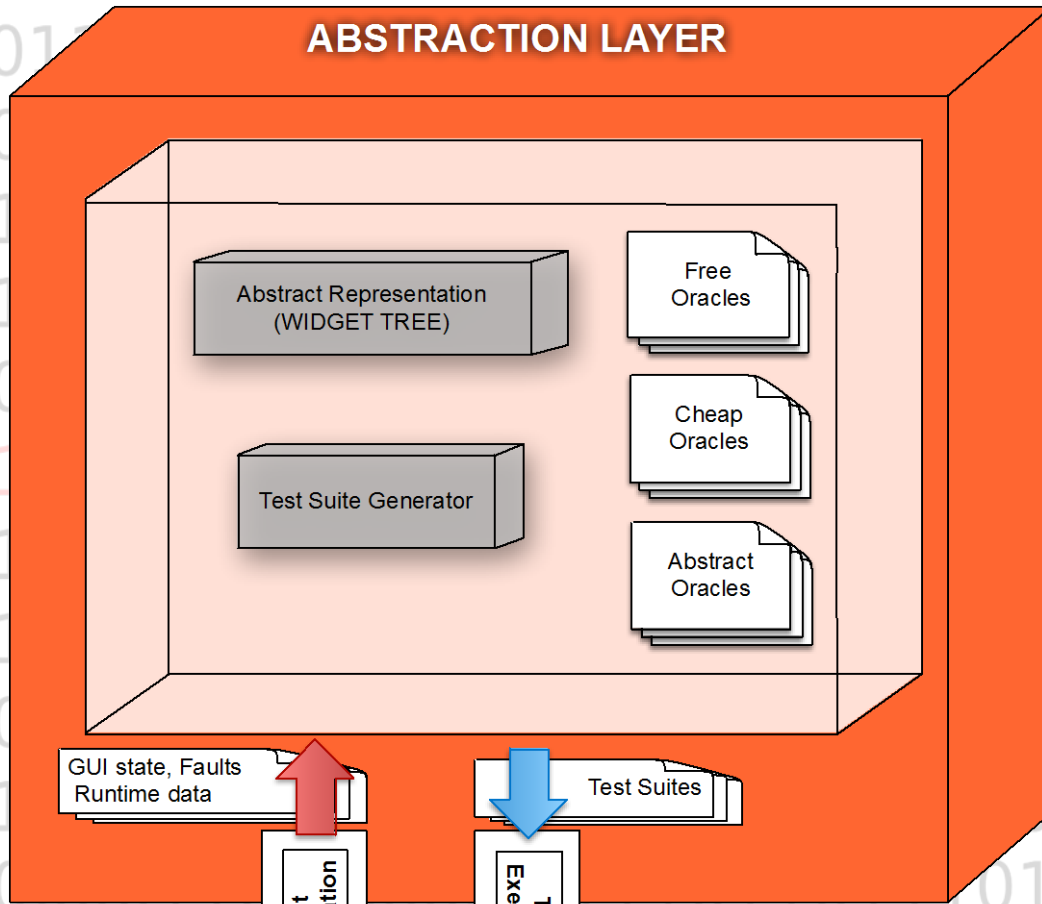
EXAMPLE



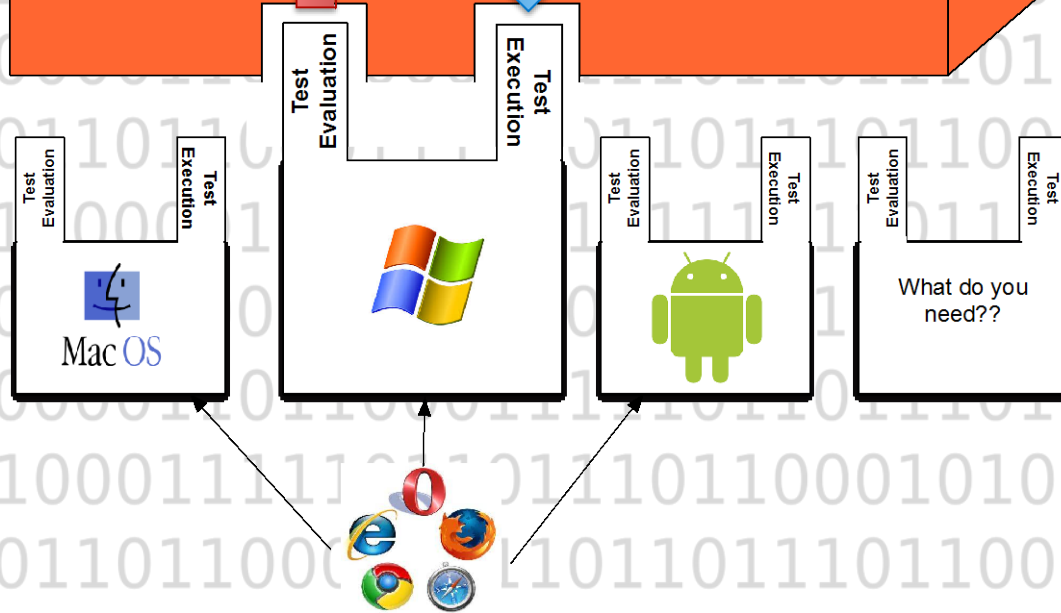
EXAMPLE



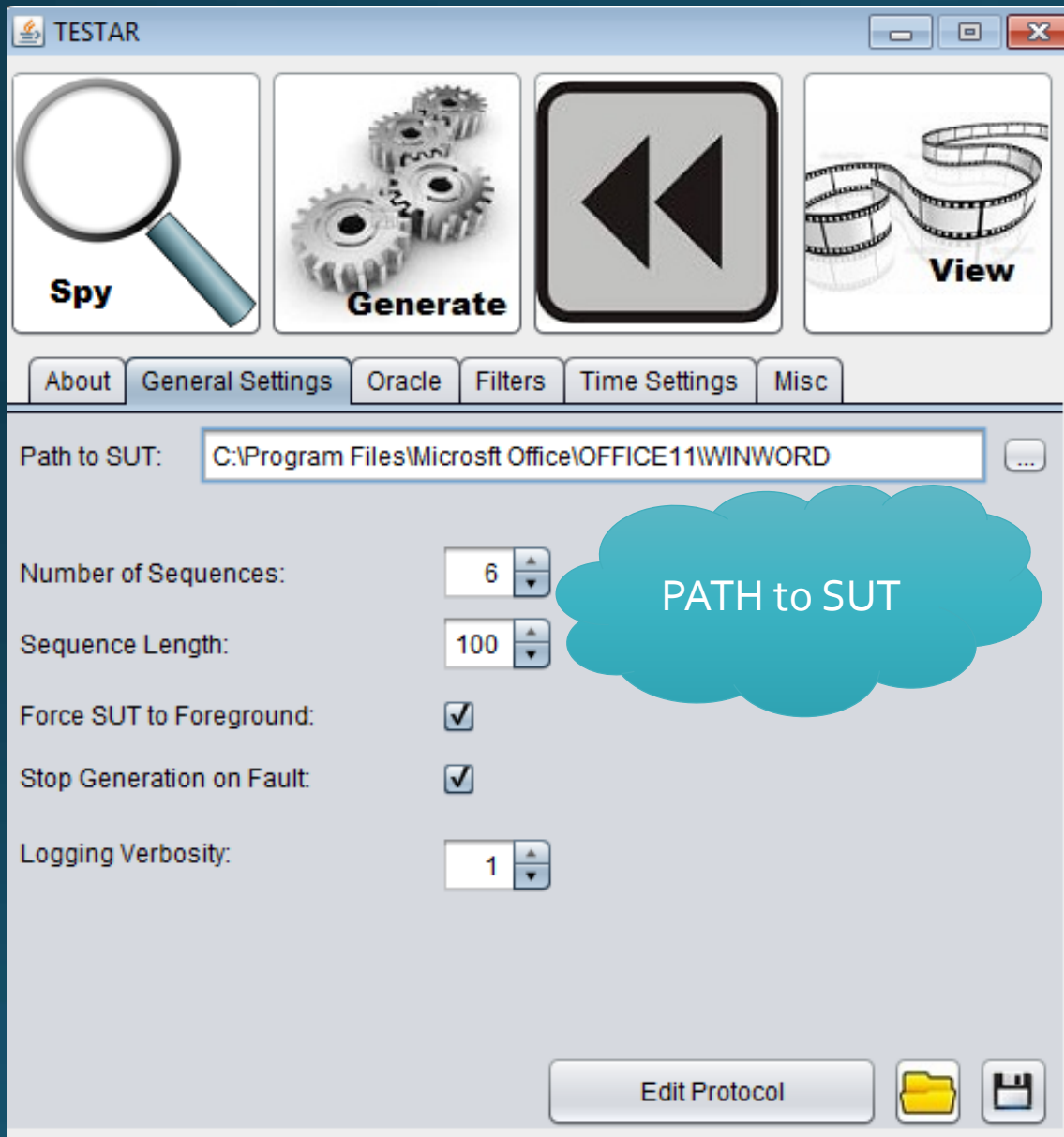
ABSTRACTION LAYER



Test* werkt (in theorie) voor alle soorten software met GUI

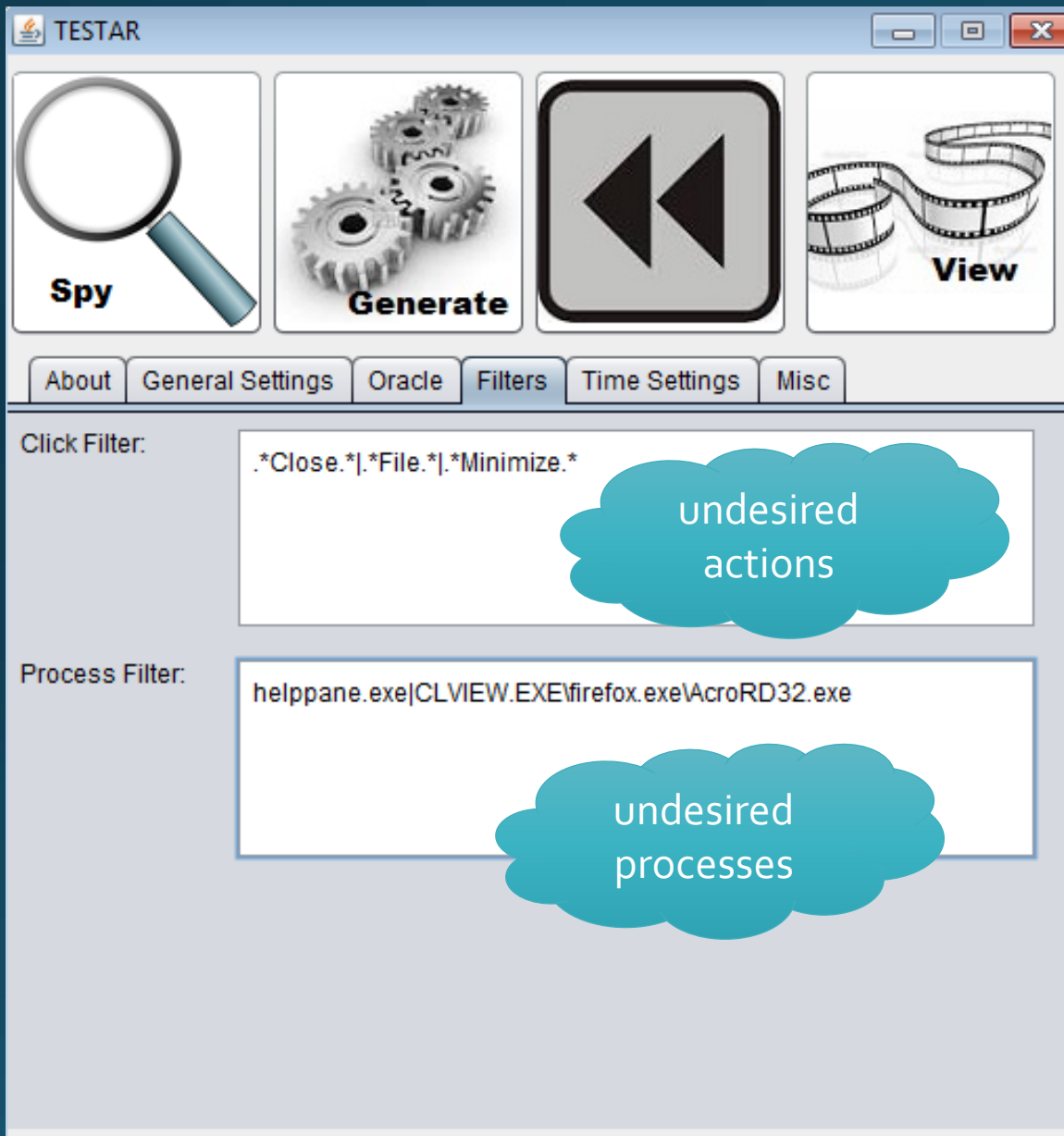


READY



Test^{*}ar.org

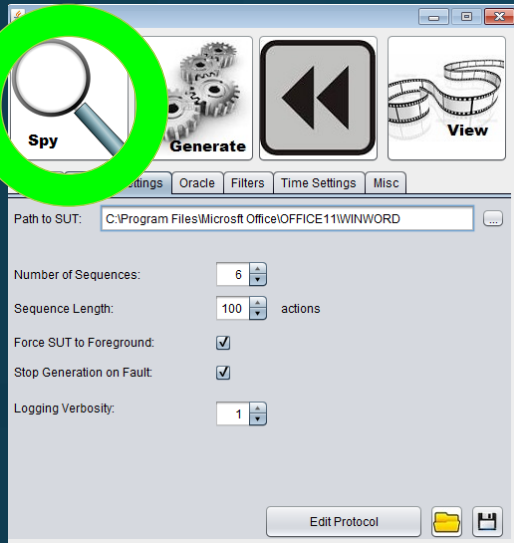
SET



GO!

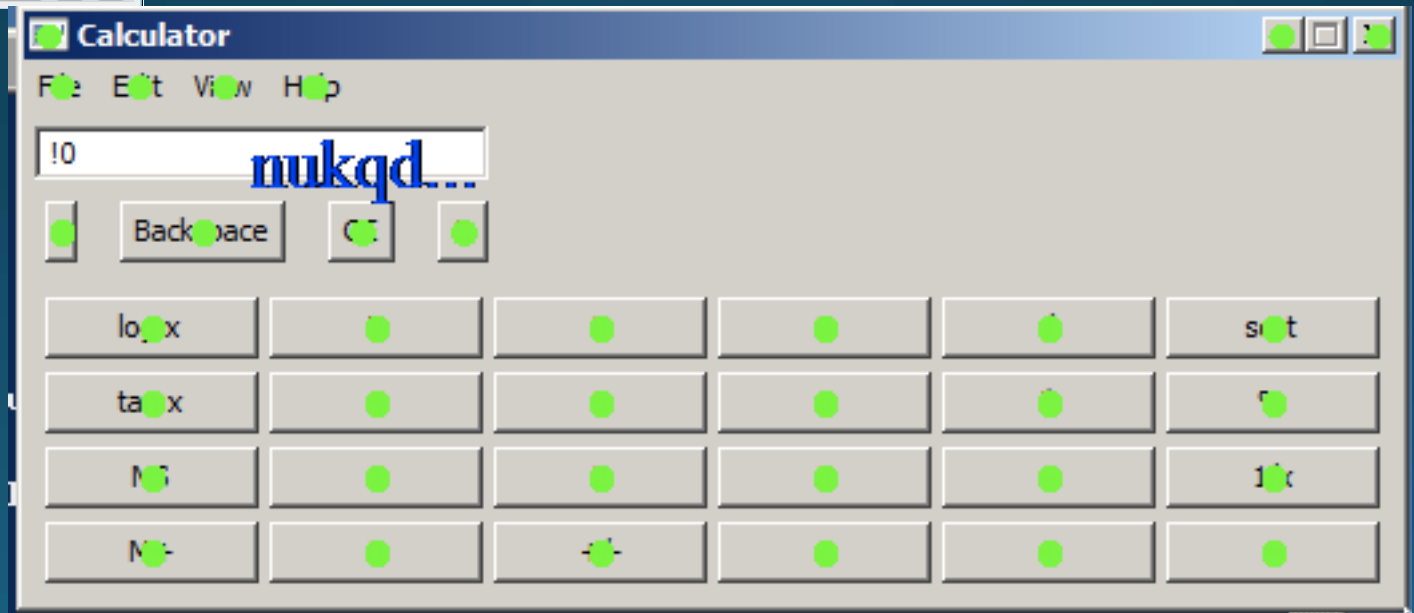


TESTAR SPY mode

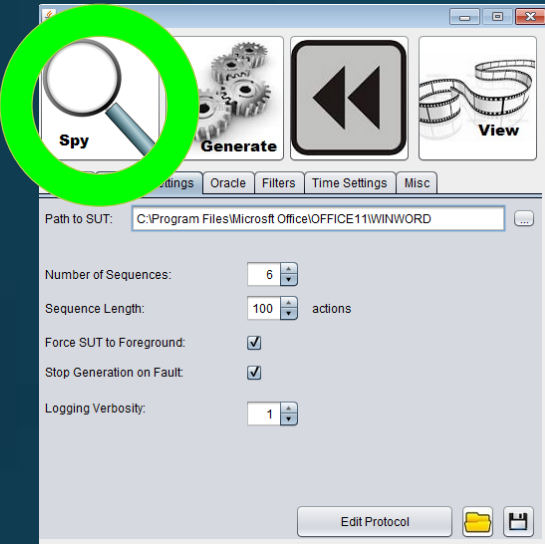


Green Dots

Acties die uitgevoerd kunnen worden in de huidige GUI state

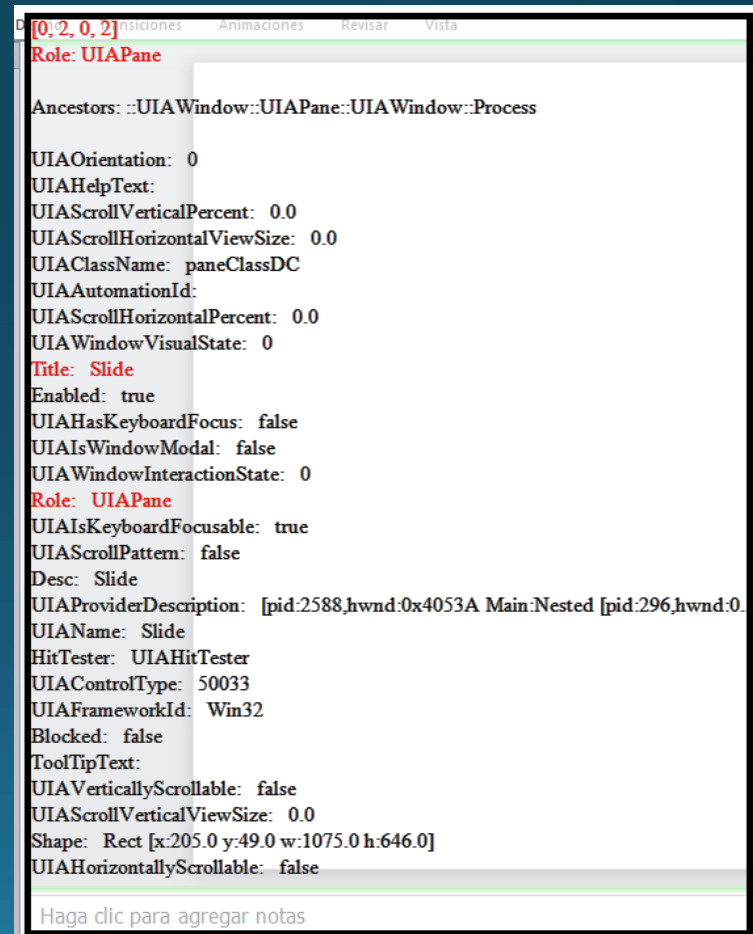


TESTAR SPY mode

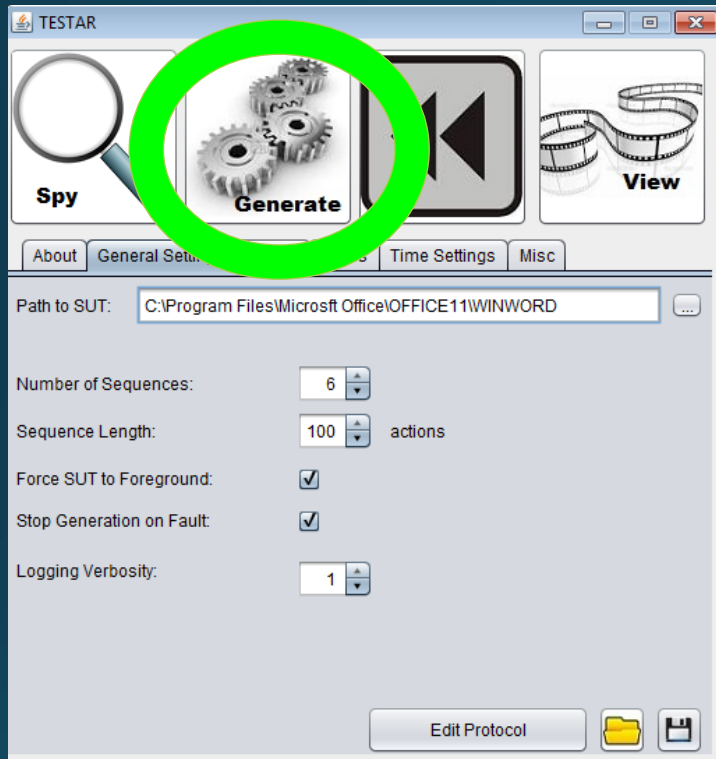


SHIFT + 3

Spy de eigenschappen van een widget



TESTAR **GENERATE** mode

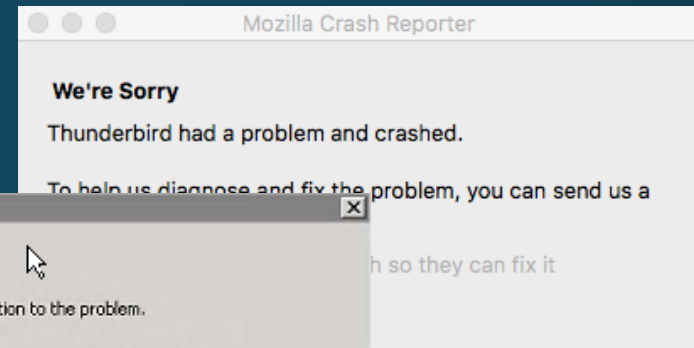
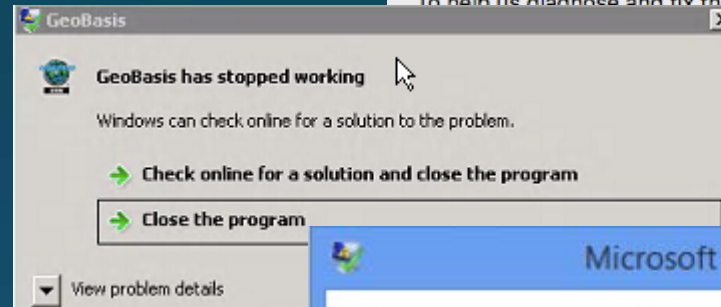
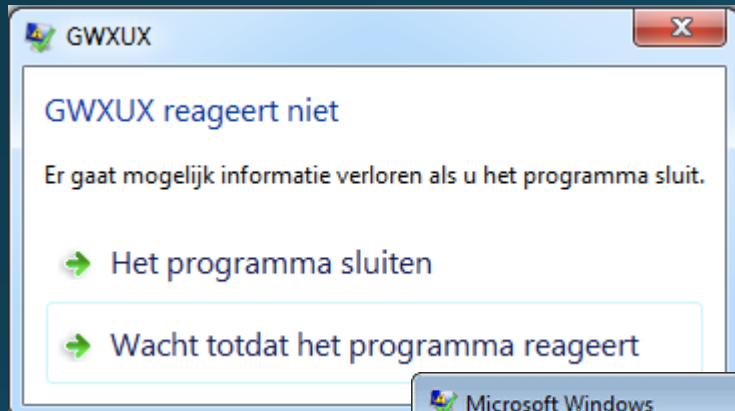


Genereer en voer testen uit

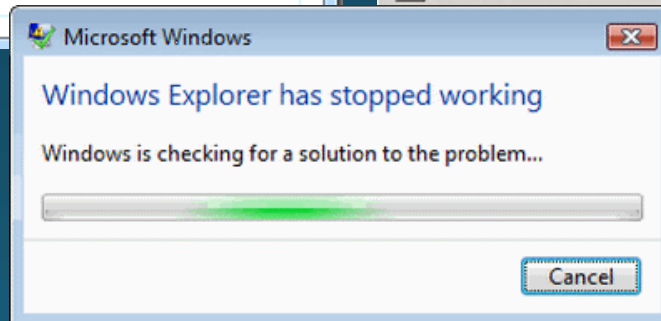
Hoe herkennen we de fouten?

Wat kun je automatisch herkennen zonder domeinkennis?

gecrasht?



geen reactie?



Impliciete oracles

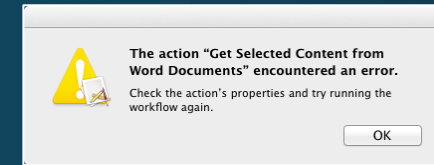
```
Verdict oracle_Crash (State state){  
    if(!state.get(IsRunning,false))  
        return new Verdict("System crashed!");  
}
```

```
Verdict oracle_Responsiveness (State state){  
    if(state.get(NotResponding, true))  
        return new Verdict("System not responding!");  
}
```

- Crashes
- Hangs

Hoe herkennen we de fouten?

Wat kun je automatisch herkennen zonder domeinkennis?



Ongewenste
woorden

ERROR

PROBLEM

FOUT

NULLPOINTEREXCEPTION

```
[java] 0 Click (BUTTON1) at IndexFinder: [0, 7] (origin: (0.5, 0.5), offset: (0.0, 0.0)) Click (BUTTON1) at AXMenuItem (origin:  
[java] 2012-04-01 19:35:24.927 Microsoft Word[9937:9c03] Error [1001] getting window resolution  
[java] 2012-04-01 19:35:24.927 Microsoft Word[9937:9c03] Error [1001] setting resolution to 1  
[java] 2012-04-01 19:35:51.360 Microsoft Word[9937:9c03] Error [1001] getting window resolution  
[java] 2012-04-01 19:35:51.360 Microsoft Word[9937:9c03] Error [1001] setting resolution to 1
```

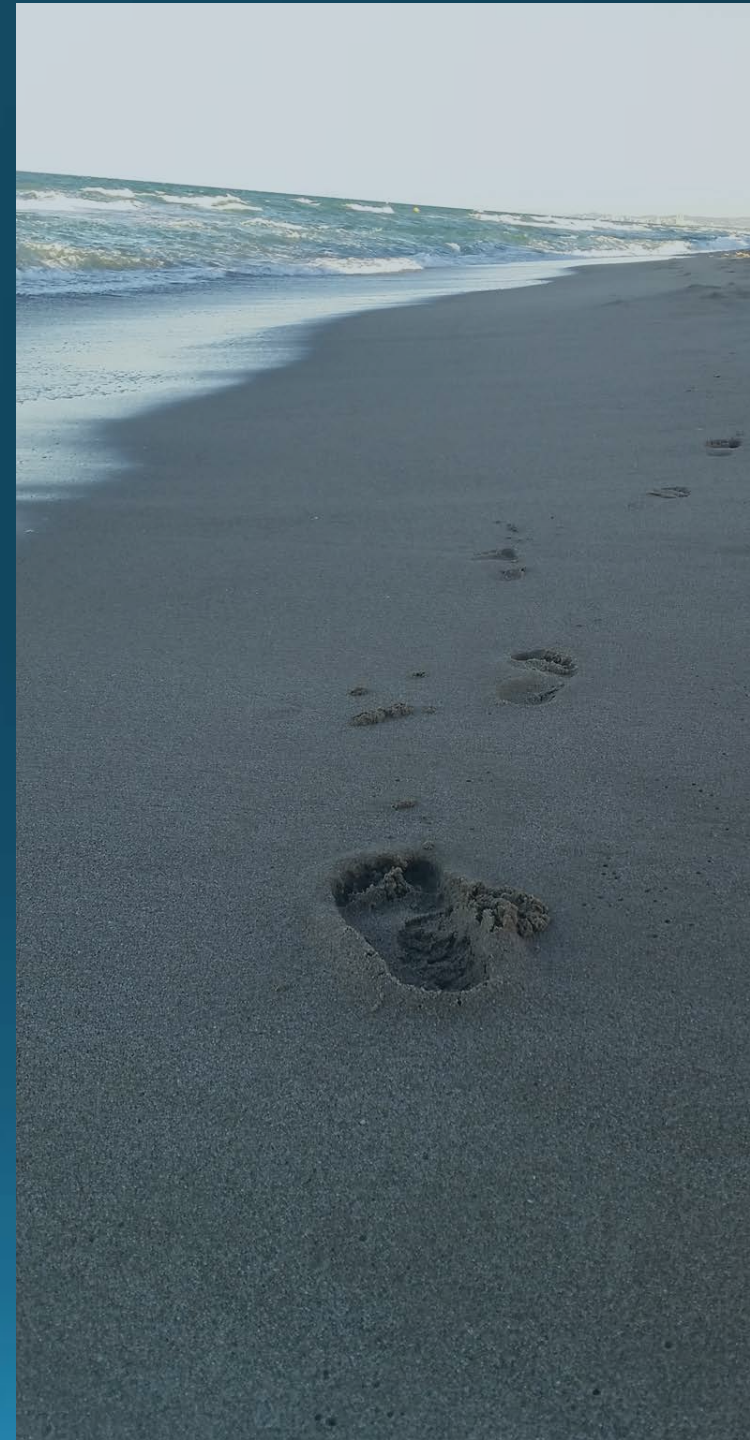
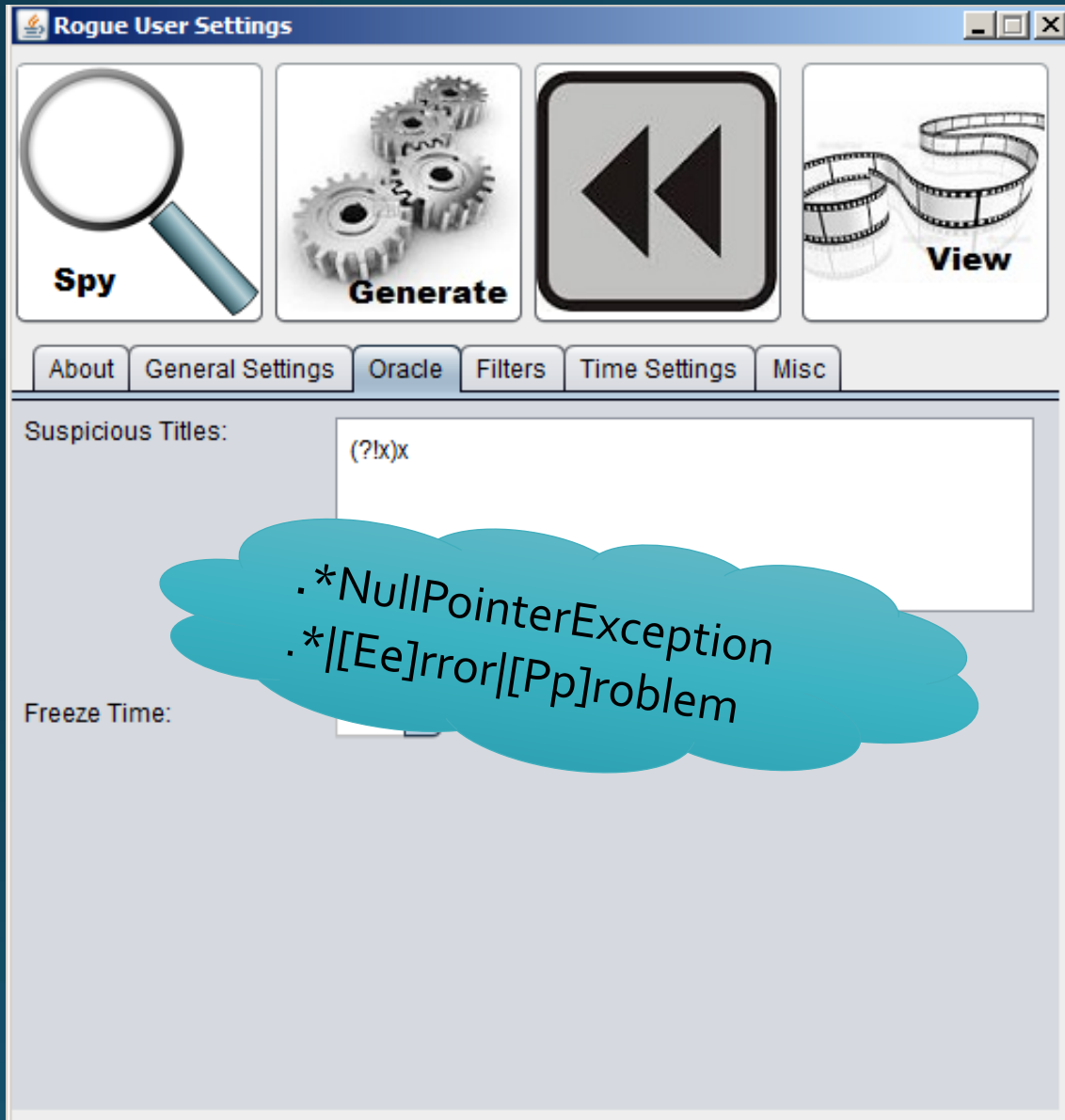


OK

More Information

OK

Oracle – Suspicious titles




```
Verdicts oracle_SuspiciousTitles(State state){  
    verdicts = new Verdicts();  
    String regEx = settings().get(SuspiciousTitles);  
  
    // search all widgets for suspicious titles  
    for(Widget w : state){  
        String title = w.getTitle("");  
        if(title.matches(regEx)){  
            verdicts.add(new Verdict("suspicious title.."));  
        }  
    }  
    return verdicts;  
}
```

Oracle – Suspicious titles (under the hood)

Domein specifieke orakels



Die moeten we programmeren.....

Onder het maken van orakels komen we niet uit

COMPLEXITEIT

VS

EFFECTIVITEIT

ONDERZOEK:

- Automatiseer zoveel mogelijk de rest
- Orakels: maak het de tester makkelijker!


```
Verdicts oracle_ImagesWAI(State state) {  
    verdicts = new Verdicts():  
    for(Widget w : state){  
        Role role = w.get(Tags.Role);  
        if (role.equals("UIImage") && title.isEmpty())  
            verdicts.add(new Verdict("Not all images have an  
                                    alternate textual description");  
    }  
    return verdicts;  
}
```

Add any oracle by additional programming

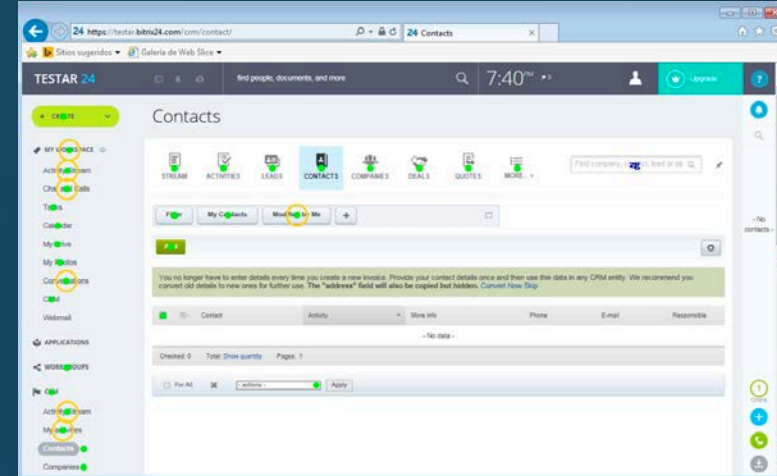
Test*

Hoe weten we of we goed getest hebben?

- Toepassen
- Meten
 - Aantal fouten
 - Ernstigheid van die fouten
- Meten (surrogate measures)
 - Dekking
 - Tijd
 - Vergelijken met handmatig
 - Lengte van de test sequences
 - Aantal verschillende toestanden
 - etc....

ClaveiCon

- Voorbereiden: **26 uur**
- Automatisch testen : **91 uur**
- Na het testen : **1,5 uur**
 - Inspectie van log files,
 - Reproduceren en begrijpen van de fouten
- **10** kritische fouten gevonden die onbekend waren



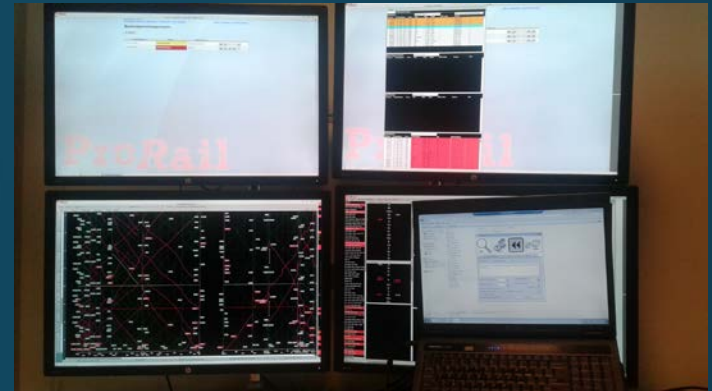
SOFTEAM

- Konden fouten her-injecteren

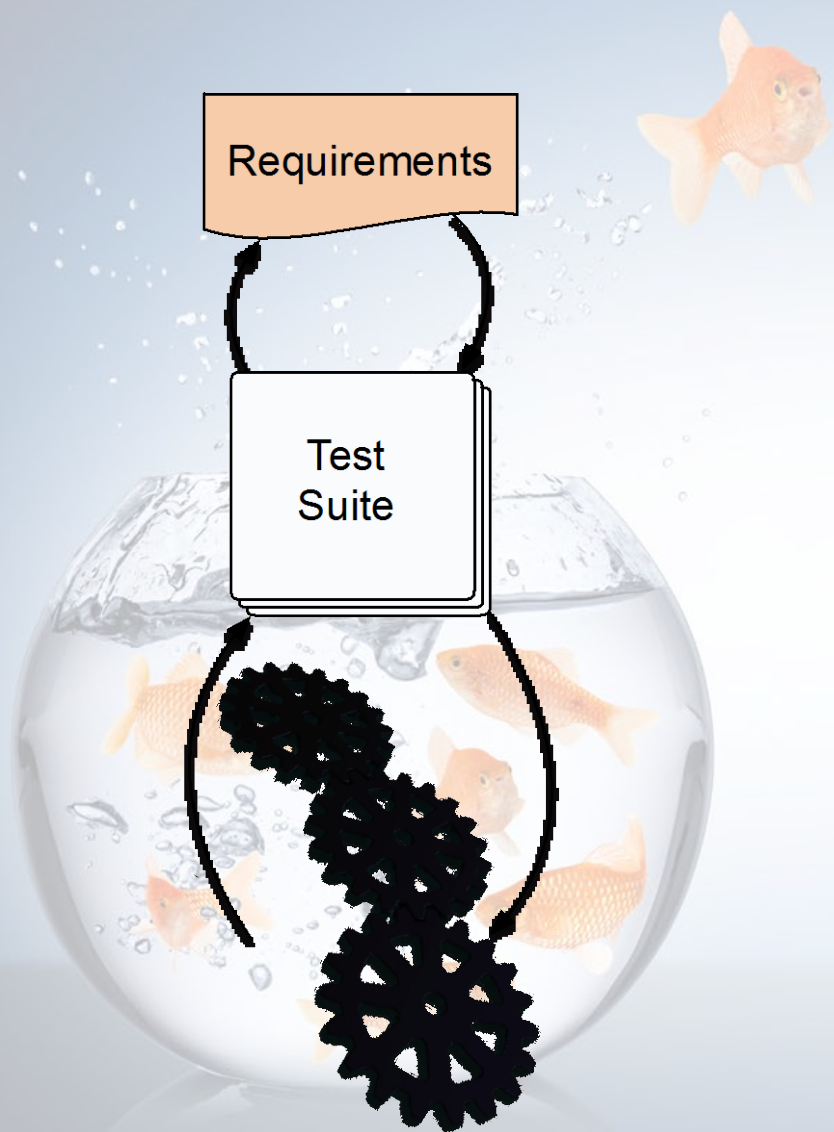
$$\text{FDR} = \frac{\text{Aantal gevonden fouten}}{\text{Aantal geïnjecteerde fouten}} \times 100\%$$

	TESTAR	Handmatig
Voorbereiden	40 uur	36uur
Testen	77 uur	1 uur
Na het testen	3,5 uur	2 uur
FDR	61%	83%
Code dekking	70%	86%

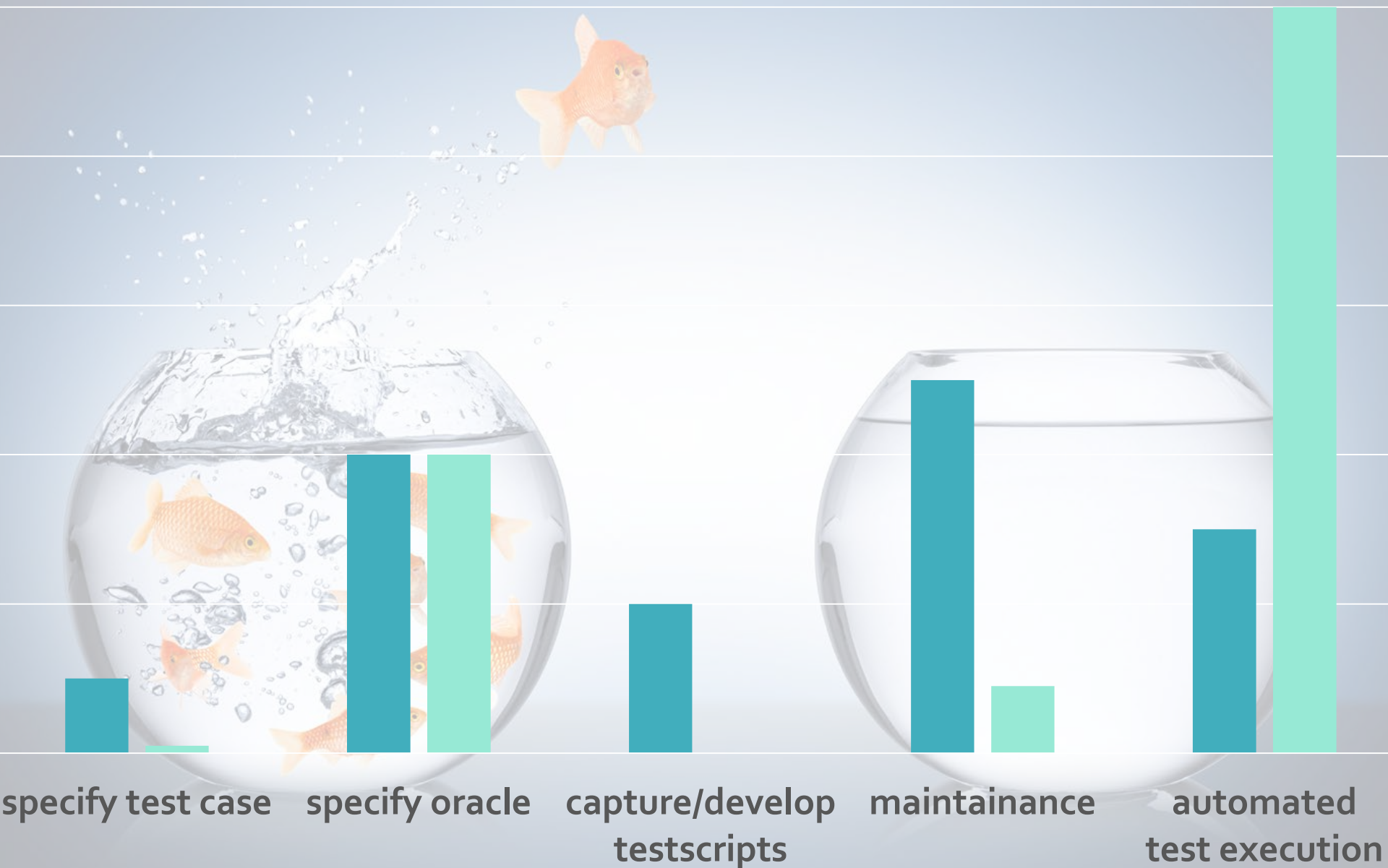
Cap Gemini/ProRail



	TESTAR	Handmatig
Voorbereiden	44 uur	43 uur
Testen	51 uur	6 uur
Na het testen	5 uur	2 uur
Ernstige fouten	4	0
Functionele dekking	80%	73%



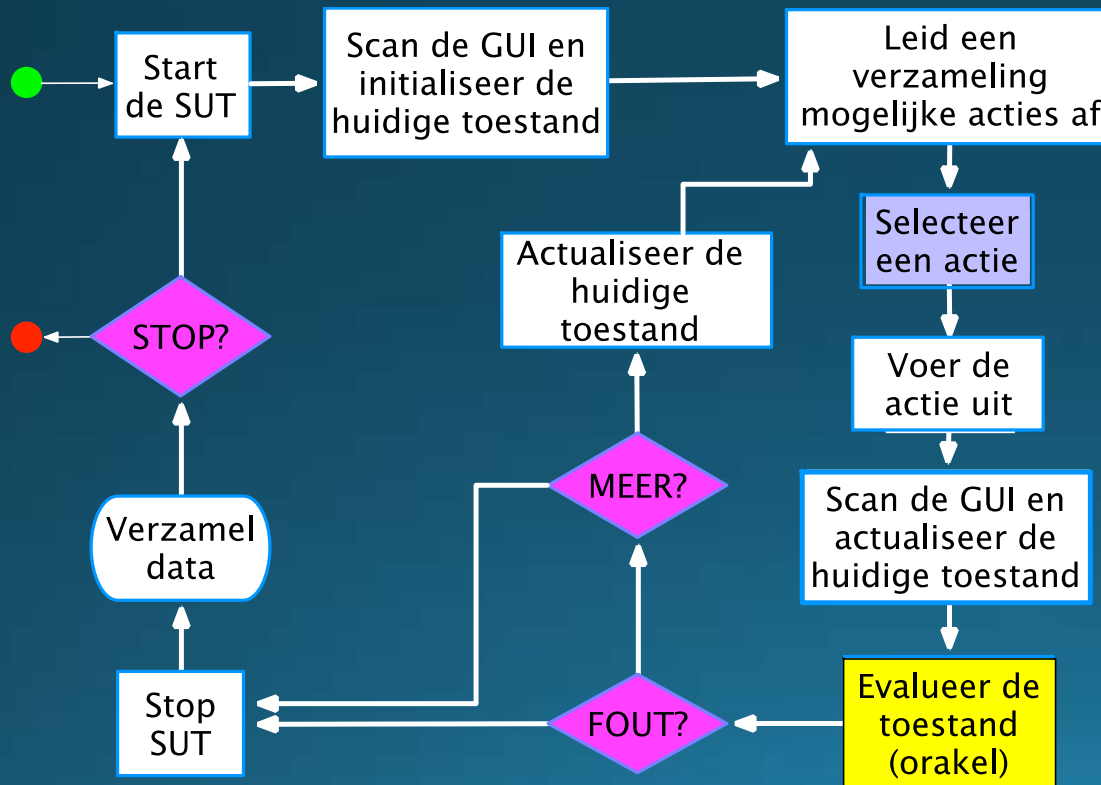
Hoe verandert **Test*** het testen?



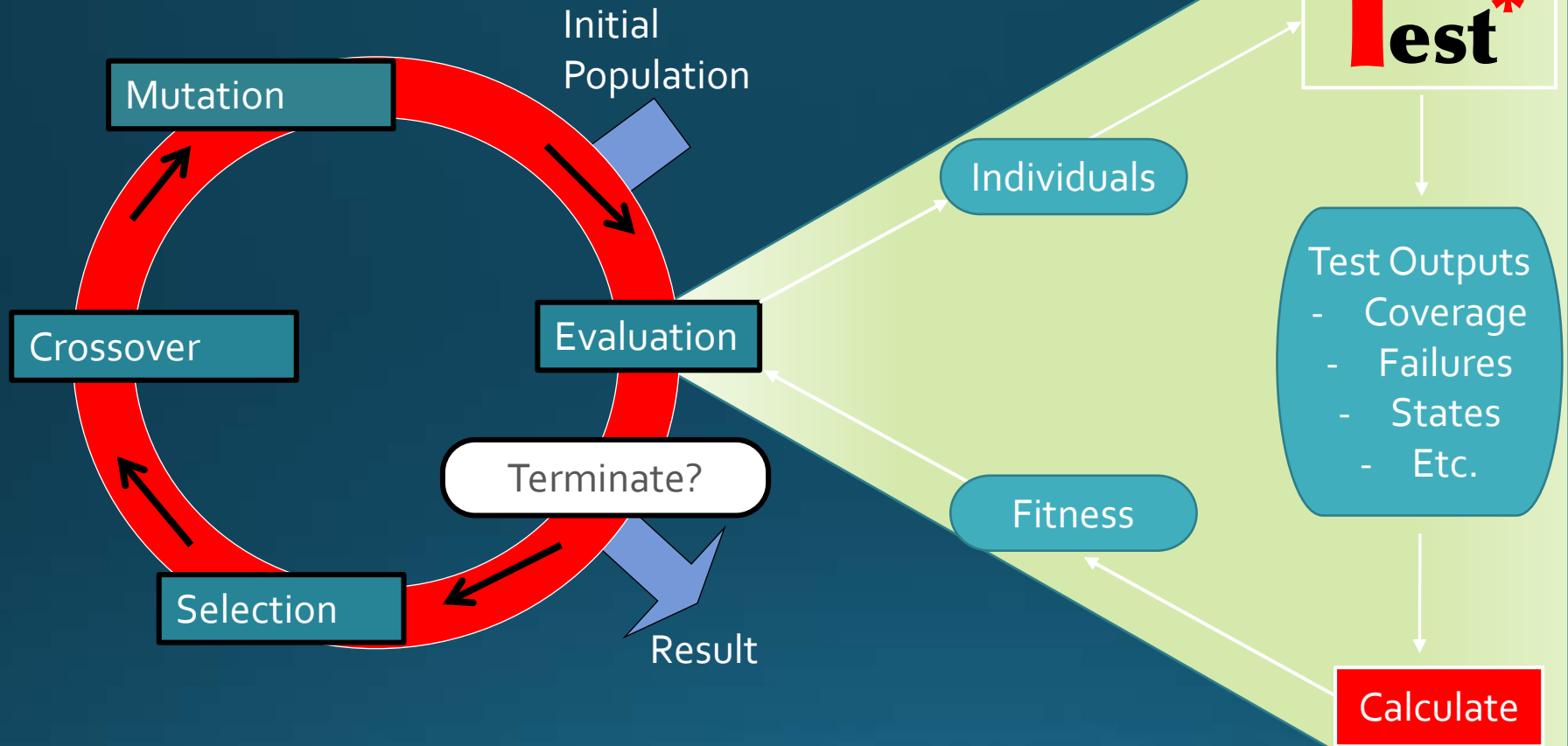
TESTAR towards 2025

Laat TESTAR de beste test strategie leren!

- Genetic programming
- Machine learning algorithms



TESTAR towards 2025



First results: #failures

SUT	Algorithm	Errors	Freezes	False positives
Odoo	GP	4	0	2
	Random	0	0	4
	Q-learning	1	1	6
PowerPoint	GP	1	0	5
	Random	0	1	2
	Q-learning	0	1	5
Testona	GP	2	2	3
	Random	0	3	6
	Q-learning	1	1	3

TESTAR towards 2025

Leer van wat de tool test!

- Meet meer surrogate measures voor test kwaliteit
 - Coverage
 - Profiling
 - Diversity
 - Novelty
- Werken de surrogate measures ?
- Kunnen we ze relateren met type van failures

TESTAR towards 2025

- Specificatie talen voor oracles en actie selectie
 - Maak het de tester makkelijk
- Mocking van externe applicatie via TESTAR
- Meer formele testing theorie nodig
 - Zodat we test methoden kunnen vergelijken
 - Zodat we weten of we het goed hebben gedaan



UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA



TESTOMAT PROJECT

The Next Level of Test Automation

Test^{*}ar.org

- **email:** info@testar.org
- **web:** <http://www.testar.org/>
- <https://www.testomatproject.eu/>
- **telephone:** +34 690 917 971

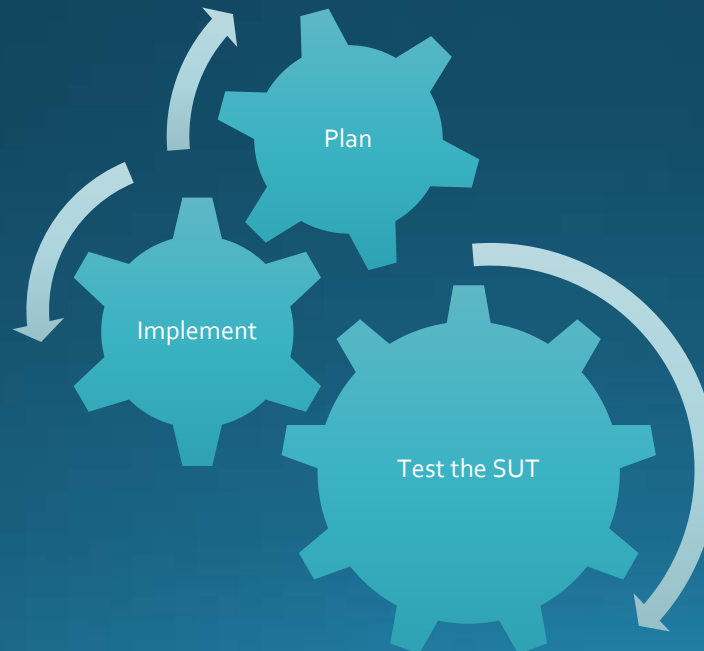
Back-up slides

How to set it up for a specific SUT

(at a client)

1) Planning Phase:

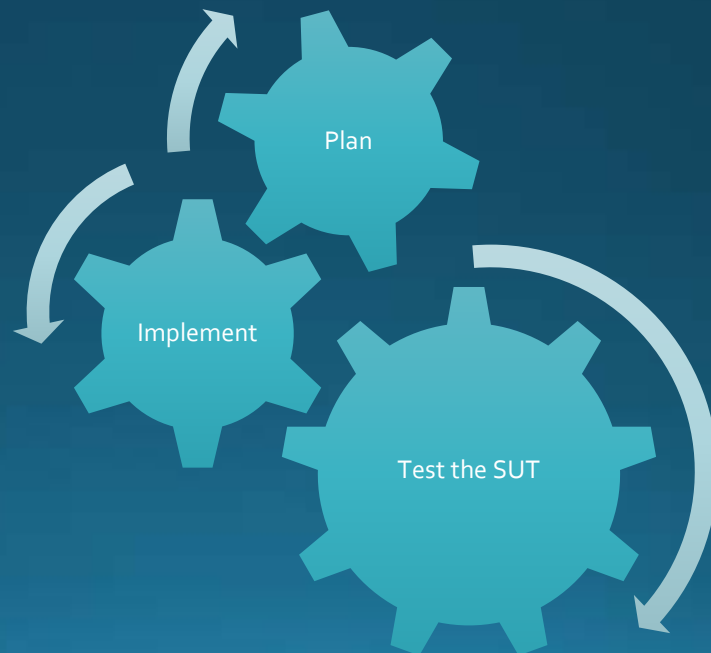
- a) Implementation of Test Environment: Recognize all the UI elements
- b) Anticipate and identify potential fault patterns: what errors do we want to find:
 - Detect crashes is for free and automatically
 - Suspicious output or messages is easy (regular expression)
 - Functionality as defined in the specifications (need oracle programming)



How to set it up for a specific SUT (at a client)

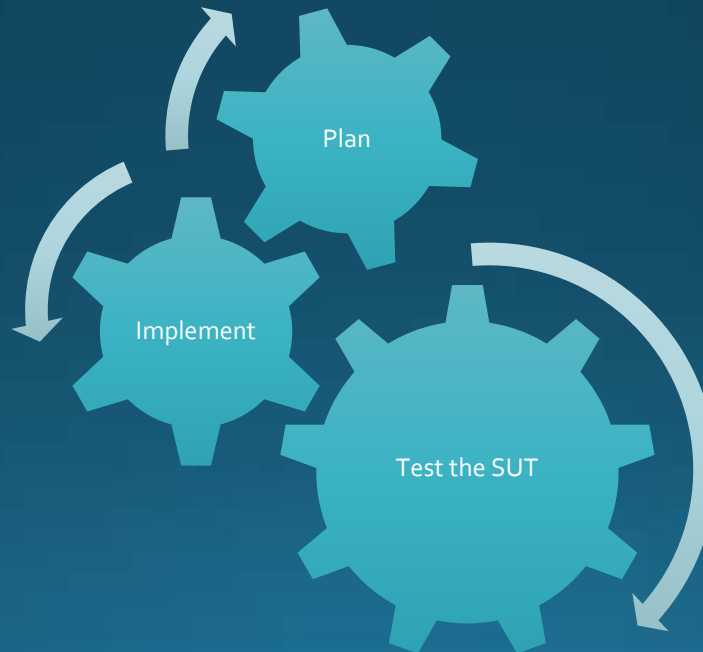
2) Implementation Phase:

- a) Oracle Implementation
 - Implement the detection of the errors defined in the previous step.
- b) Action Definition Implementation
- c) Implementation of stopping criteria



How to set it up for a specific SUT (at a client)

3) Testing Phase: run the test to find errors in the SUT



Where can we find faults?

- Surrogate measures
- We cannot measure %of faults found
- We measure something we believe, hope or have shown to be correlated to that attribute.
- Coverage
- Diversity
- Novelty

Surrogate measures

- Try to execute as many **different actions** as possible?
- Try to make **large call trees**?
- Try to visit as **many different states** as possible?
- Try to make **long sequences**?
- Try to find **novel states**?
- We need to investigate many more

Testing as a Markov Decision Process

- An MDP is a 4-tuple: $\langle S, A, T, R \rangle$
- sets S of possible states
- sets A of possible actions
- description T of the effect of action in a state
 - $T: S \times A \rightarrow S$
 - If the environment is in state s then we select an available action from $a \in A$ that causes a transition to a next state s'
- reward function $R: S \times A \rightarrow \mathbb{R}$

MDP problem: find a policy π which maximizes the reward by selecting an appropriate action in each state

Rewards

- Set S of possible states the SUT can be in
- For all $s \in S$, we have sets $A_s \subseteq A$ of actions
- We can focus is on exploration of the GUI
- Then we reward actions a with low execution count ec

$$\forall s \in S, a \in A_s: R(s, a) = \begin{cases} R_{max} & , \quad ec(a) = 0 \\ \frac{1}{ec(a)} & , \quad \textbf{otherwise} \end{cases}$$

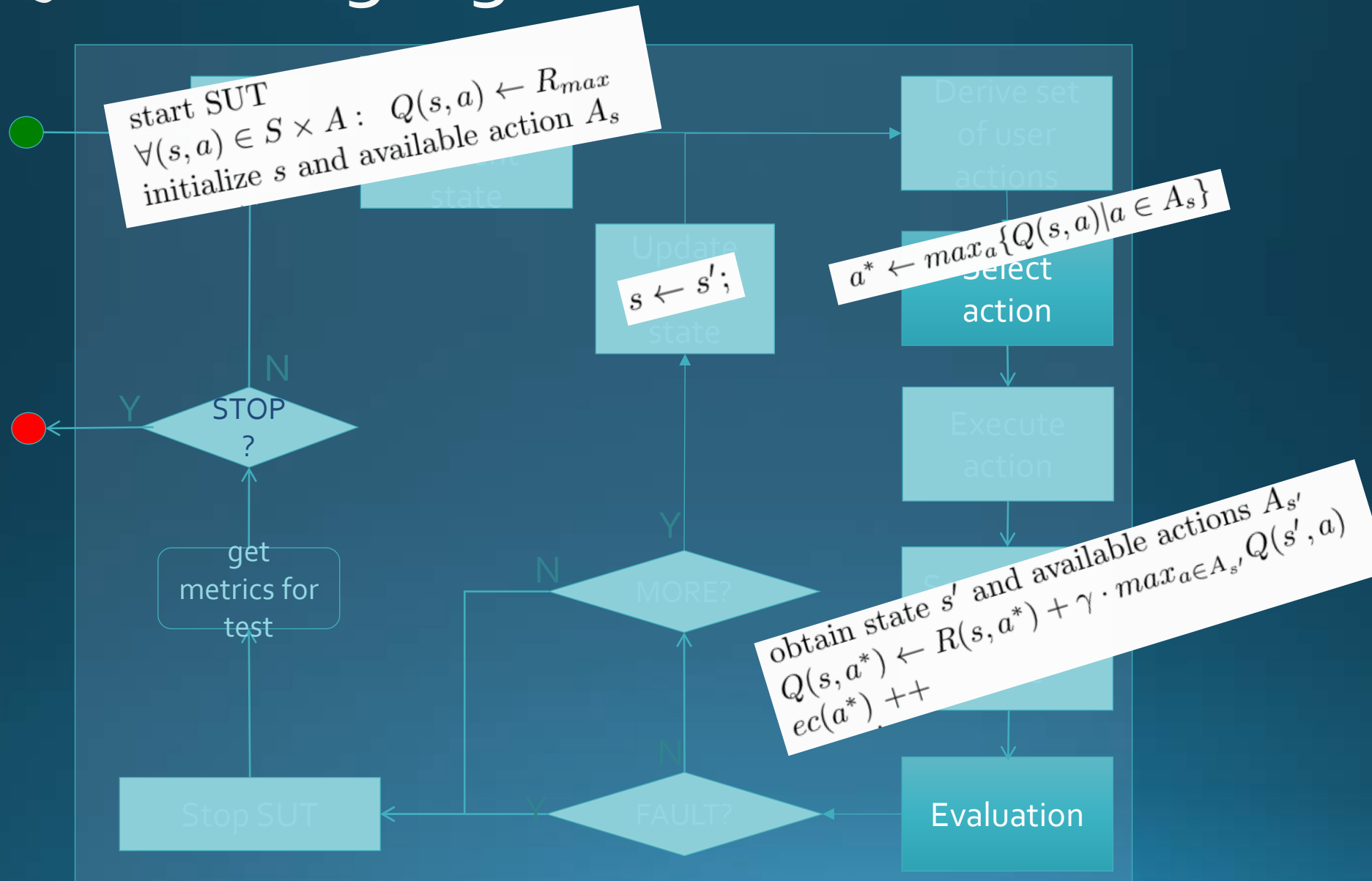
Q-learning algorithm

```
Require:  $R_{max} > 0$            /* reward for unexecuted actions */
Require:  $0 < \gamma < 1$        /* discount factor */
1: begin
2:   start SUT
3:    $\forall (s, a) \in S \times A : Q(s, a) \leftarrow R_{max}$ 
4:   initialize  $s$  and available action  $A_s$ 
5:   repeat
6:      $a^* \leftarrow \max_a \{Q(s, a) | a \in A_s\}$ 
7:     execute  $a^*$ 
8:     obtain state  $s'$  and available actions  $A_{s'}$ 
9:      $Q(s, a^*) \leftarrow R(s, a^*) + \gamma \cdot \max_{a \in A_{s'}} Q(s', a)$ 
10:     $ec(a^*) ++$ 
11:     $s \leftarrow s'$ 
12:  until stopping criteria met
13:  stop SUT
14: end
```

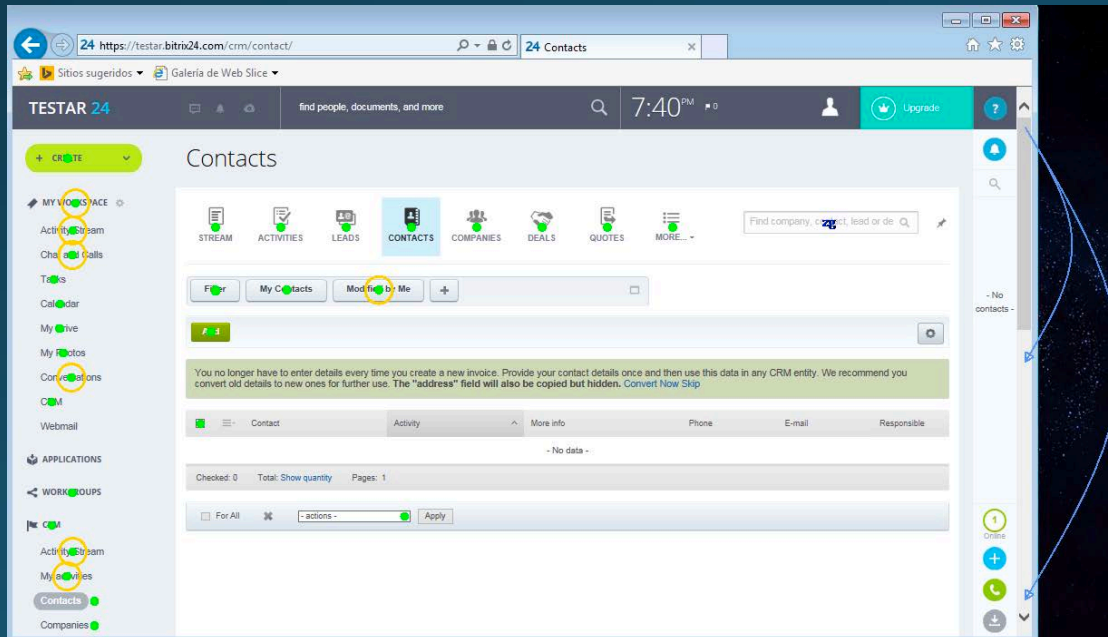
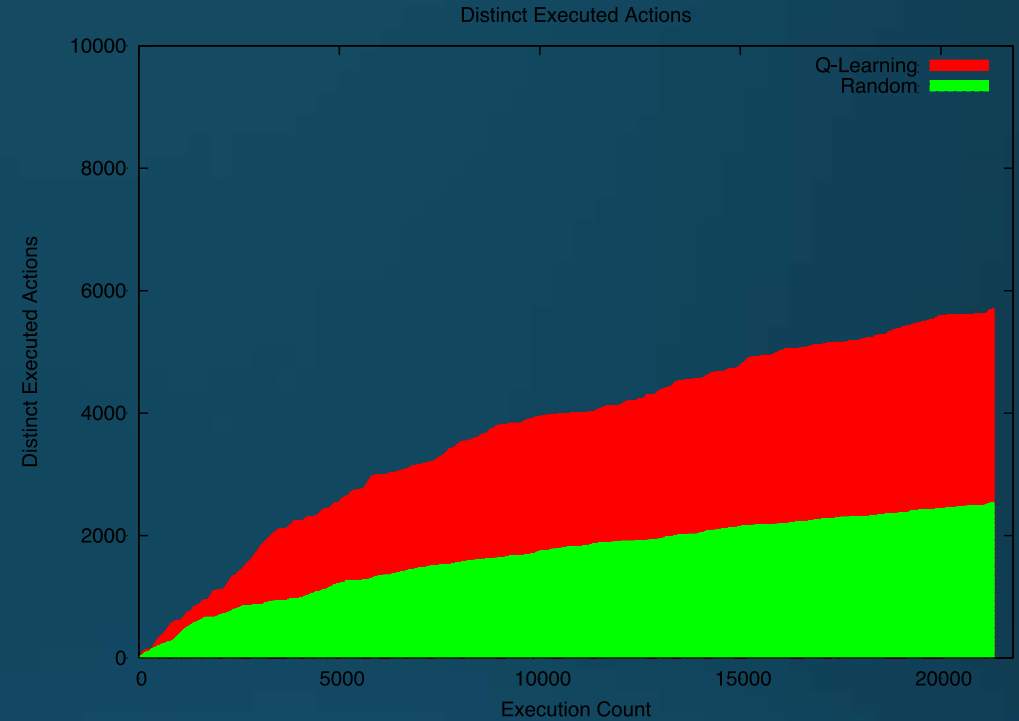
Learn Q

Use Q for

Q-learning algorithm



Tried it out – ClaveiCon



Did we find more faults?

Surrogate measures

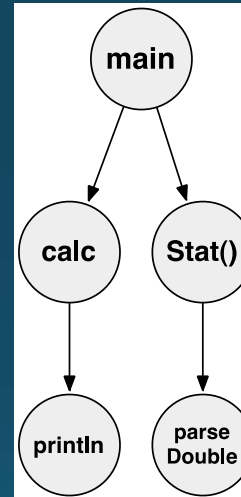
- Try to execute as many **different actions** as possible?
- Try to make **large call trees**?
- Try to visit as **many different states** as possible?
- Try to make **long sequences**?
- Try to find **novel states**?
- We need to investigate many more

Call trees

- the larger the call tree, the more aspects of the SUT are tested (McMaster and Memon)
- Bytecode instrumentation of the SUT to obtain the call tree, so no source code needed

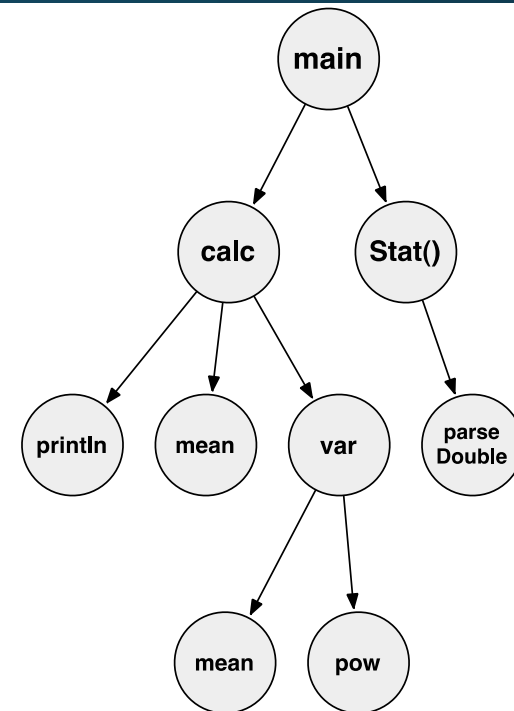
```
1 public class Stat{
2     double[] data;
3     public static void main(String[] args){
4         Stat s = new Stat(args);
5         s.calc();
6     }
7     public Stat(String[] args){
8         data = new double[args.length];
9         for(int i = 0; i < args.length; i++)
10             data[i] = Double.parseDouble(args[i]);
11     }
12     public void calc(){
13         if(data.length > 1){
14             System.out.println(mean() + ", " + var());
15         }else{
16             System.out.println(data[0] + ", " + 0.0);
17         }
18     }
19     double mean(){
20         double ret = 0.0;
21         for(double d : data) ret += d;
22         return ret / data.length;
23     }
24     double var(){
25         double ret = 0.0;
26         for(double d : data) ret += Math.pow(d - mean(), 2);
27         return ret / data.length;
28     }
29 }
```

java Stat 1.0



Number of
leaves = 2

java Stat 1.0 2.0 3.0

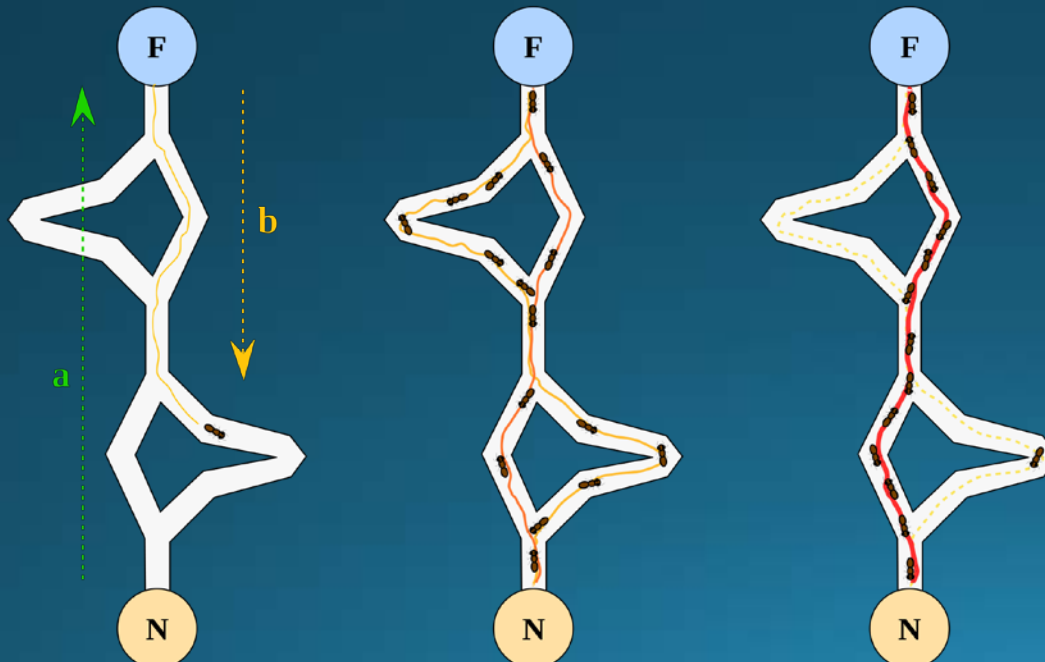
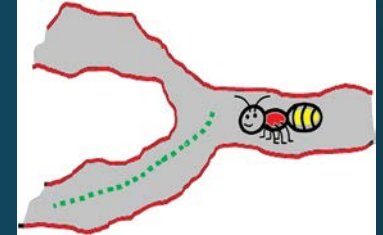


Number of
leaves = 5

Select actions such that they maximize the probability of a large call tree.

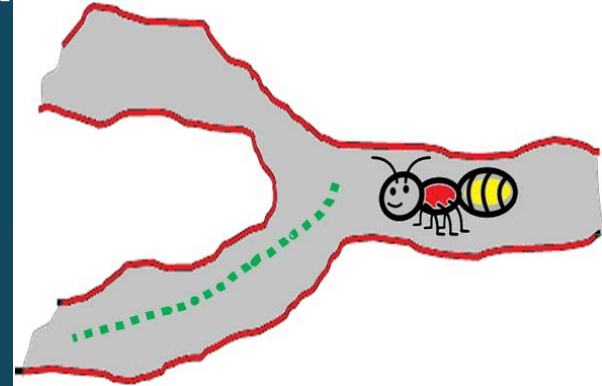
Ant Colony Optimization

- Collectively ants can solve complex tasks
- Ants communicate using pheromones
 - They lay this on their path
 - Pheromone trail strength accumulates when multiple ants use a path
 - Other ants go where there is good pheromone strength

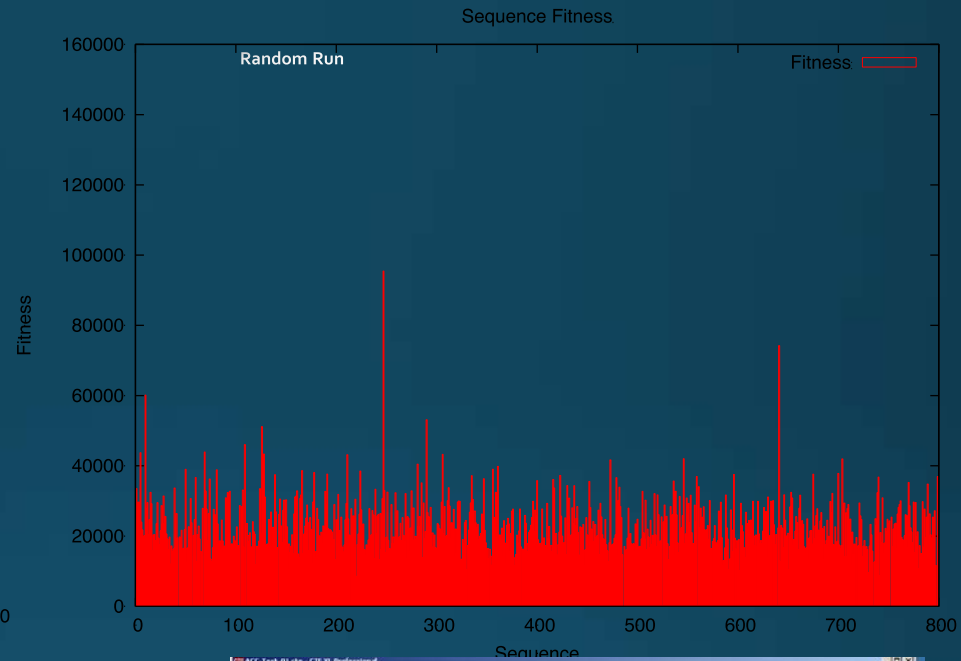
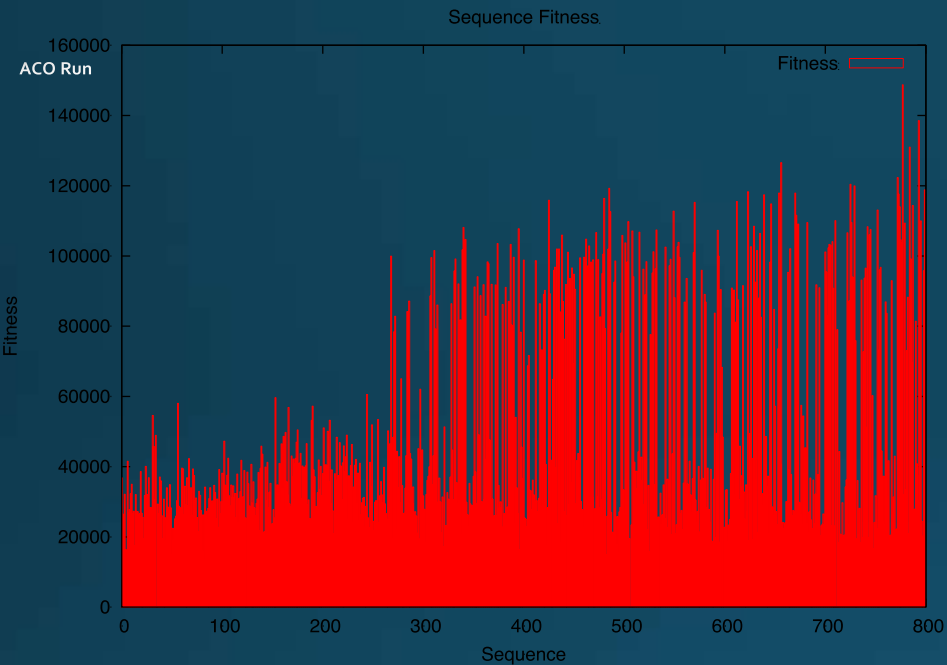


Ant Colony Optimization

- We have a population of ants
- Set of choices $\mathcal{C}$ (= actions)
- The ants generate trails (= test sequences)
- by choosing c_i according to pheromone values p_i (= selection criteria)
- Choices (= actions) that appear in “good” trails (= max call tree) accumulate pheromones

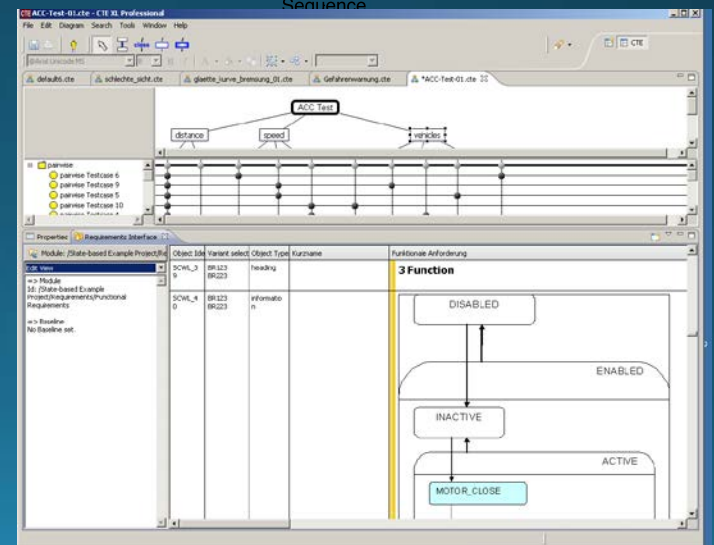


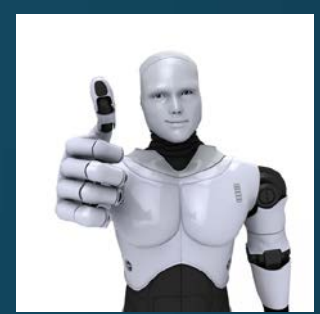
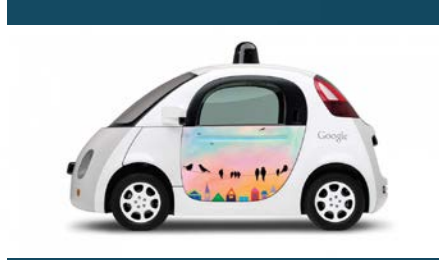
Results



Did we find more faults?

Berner & Mattner
TESTONA



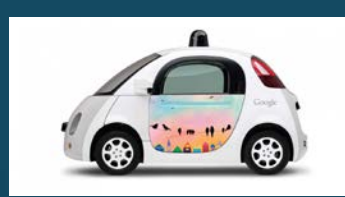


- We do not want to make an instance for each action selection strategy!
- The tools should learn what the best action selection strategy is!

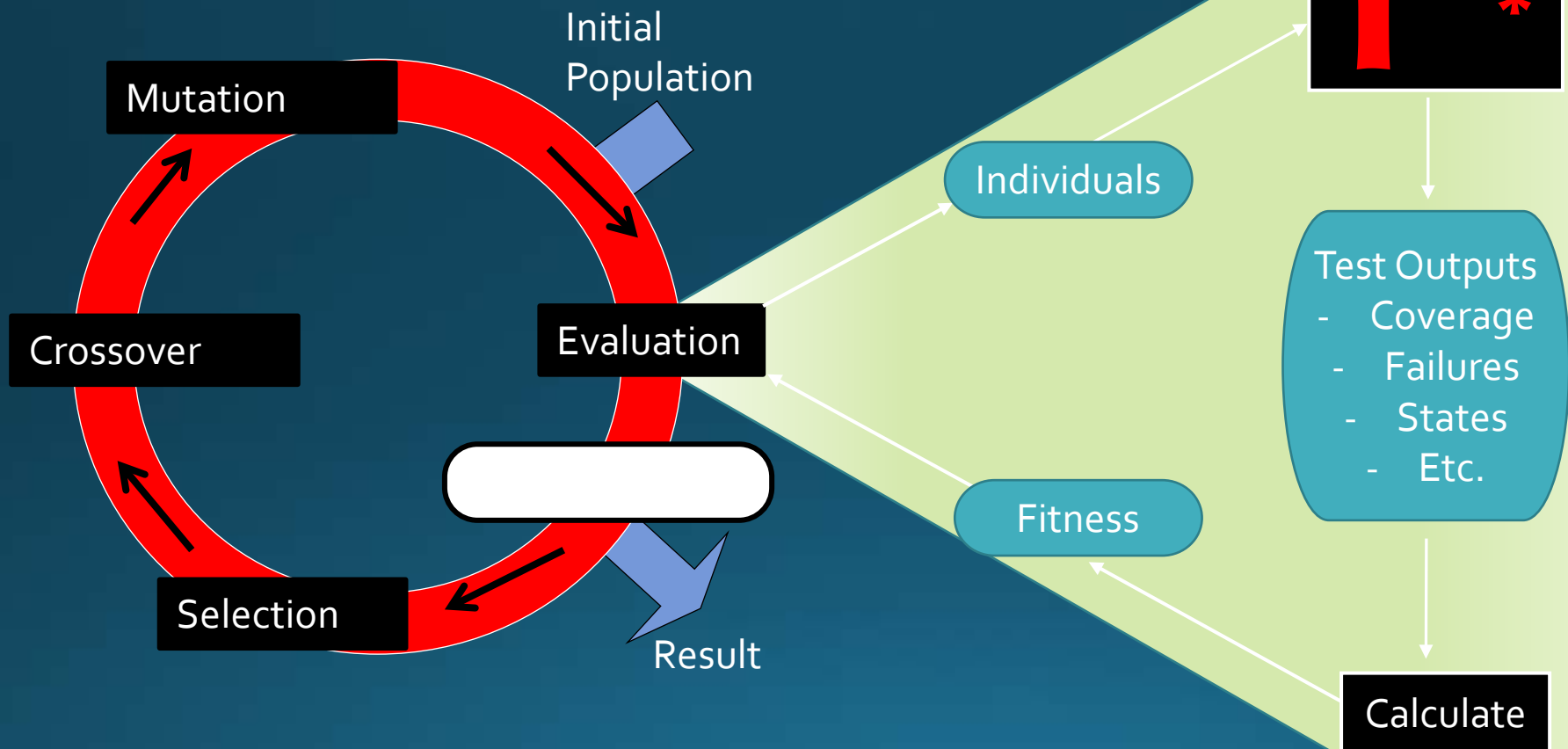


Test Outputs

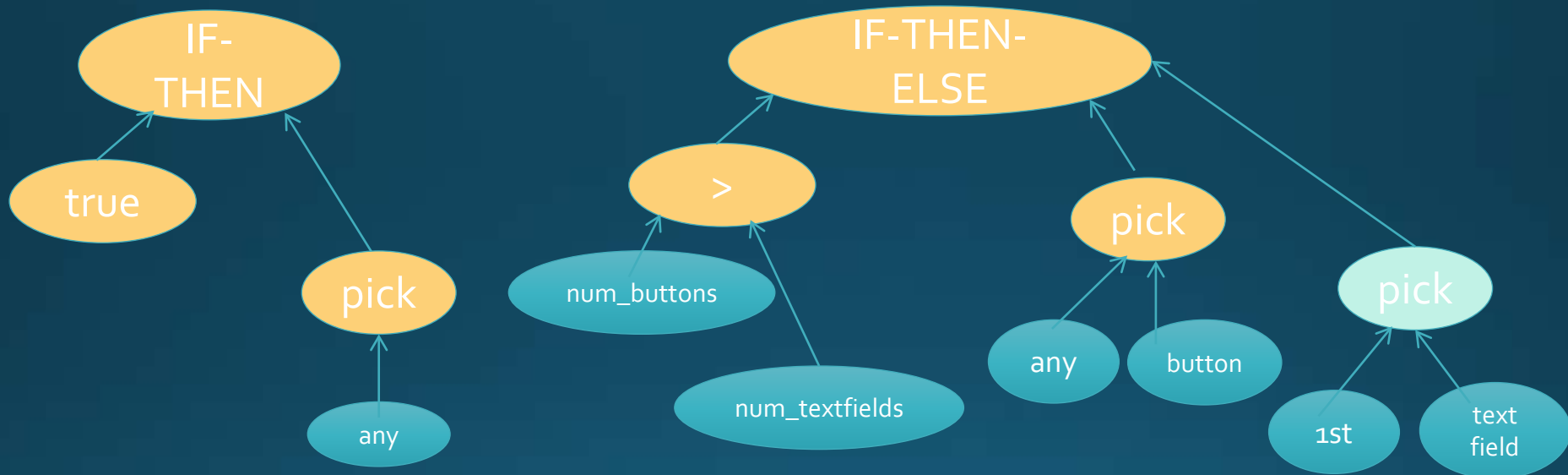
- Coverage
- Failures
- States
- Etc.



Evolve action selection rules



Action selection rules



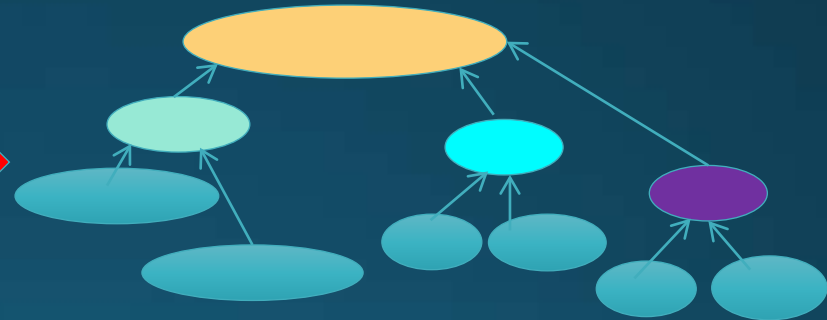
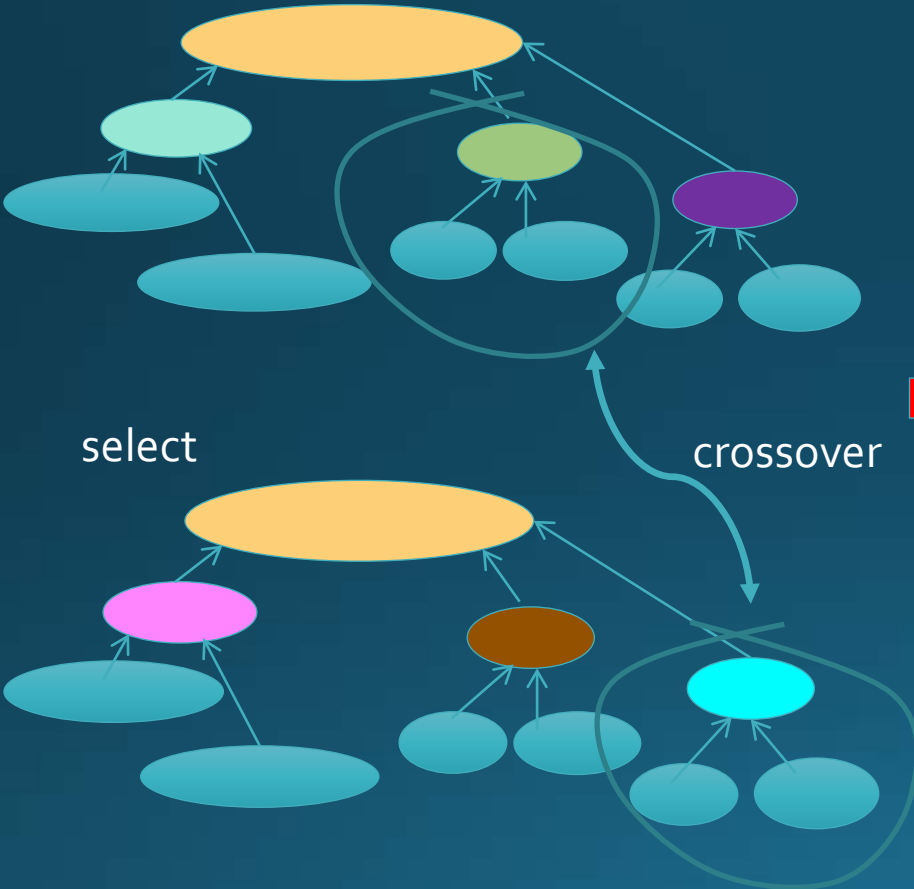
Crossover



select

select

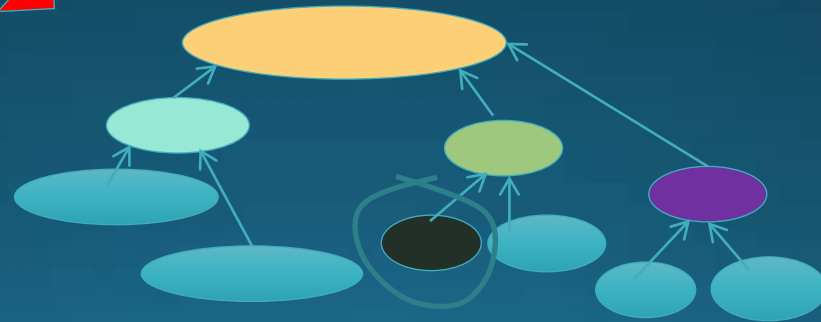
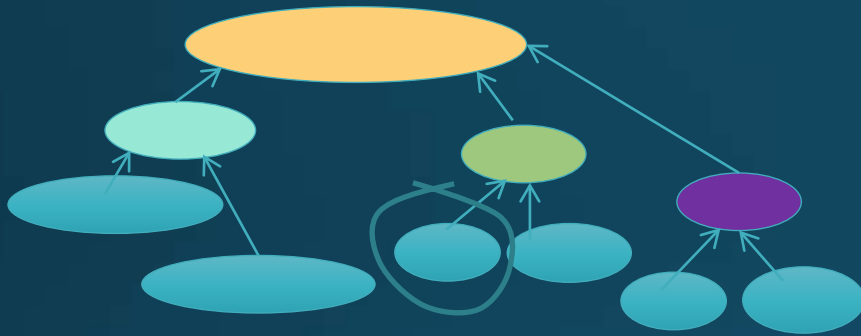
crossover

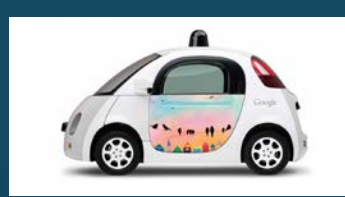


Mutation

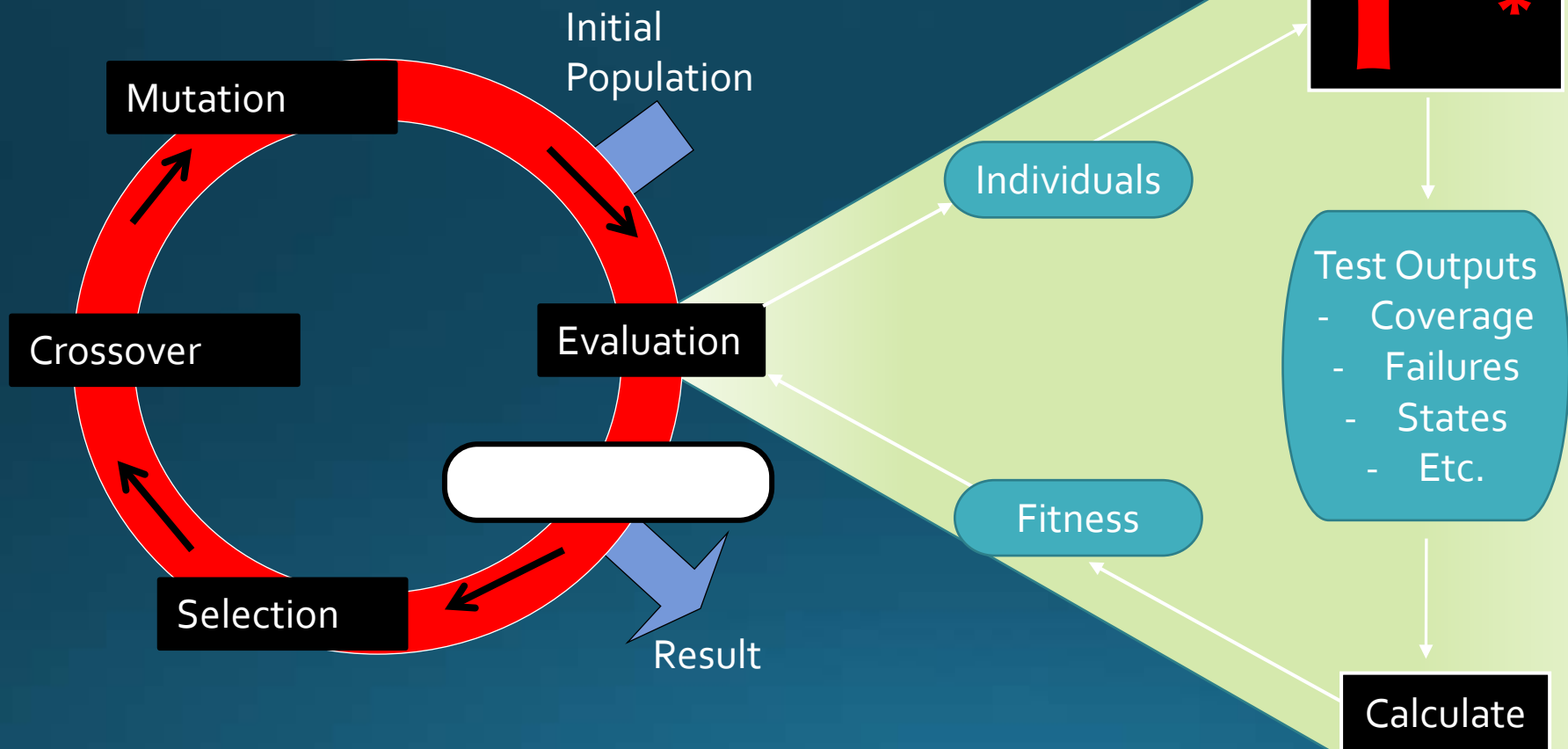


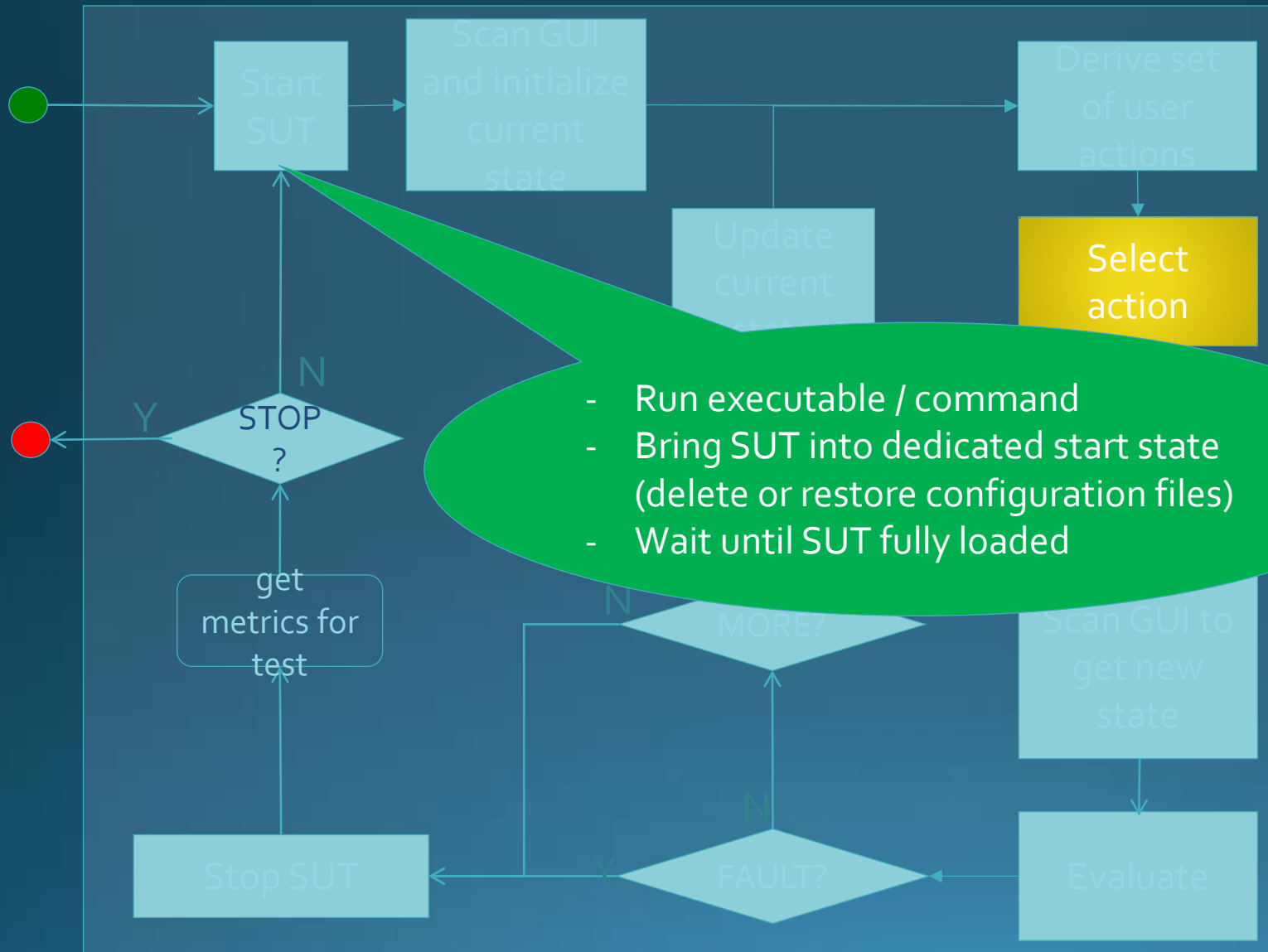
select

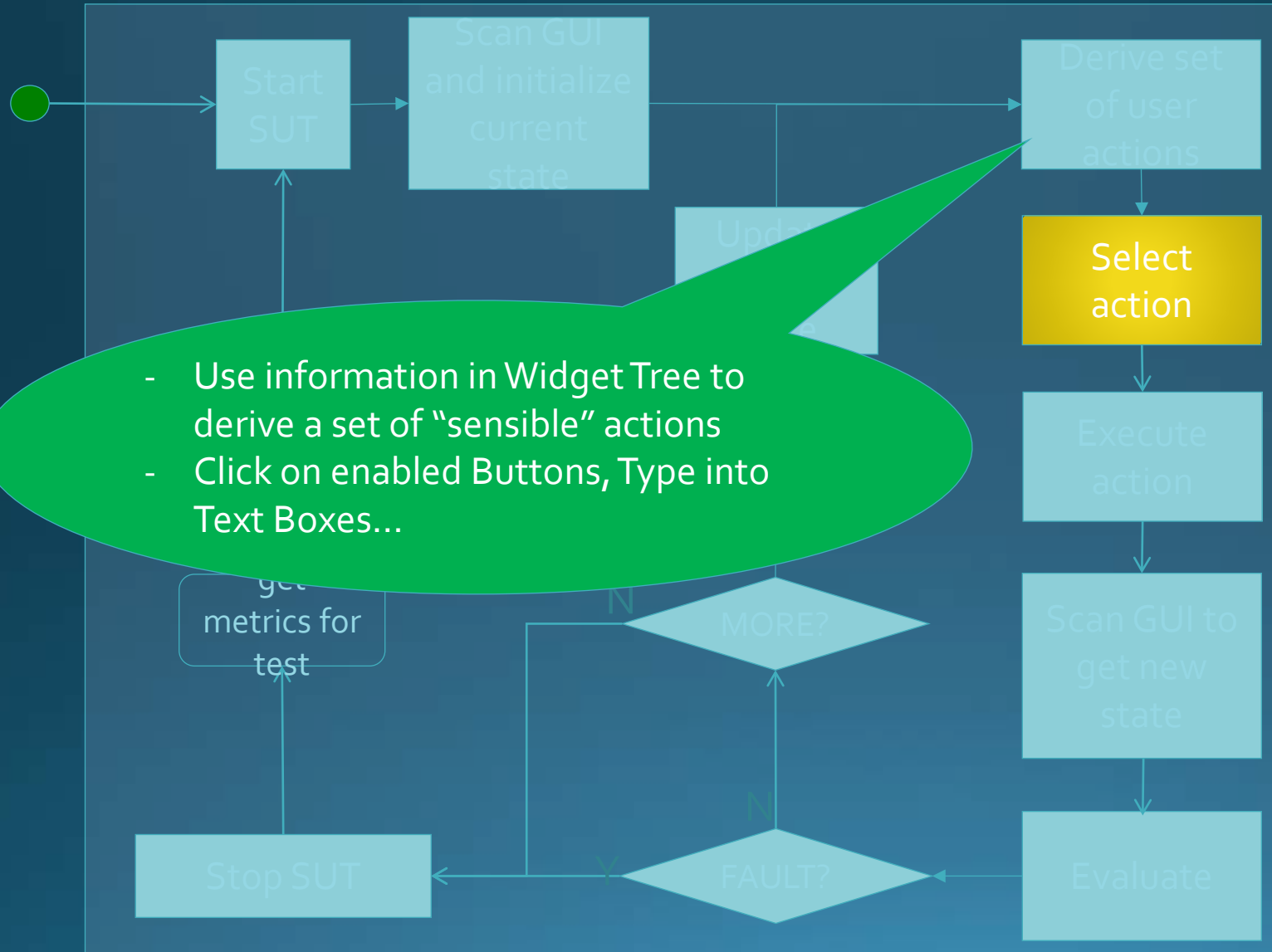


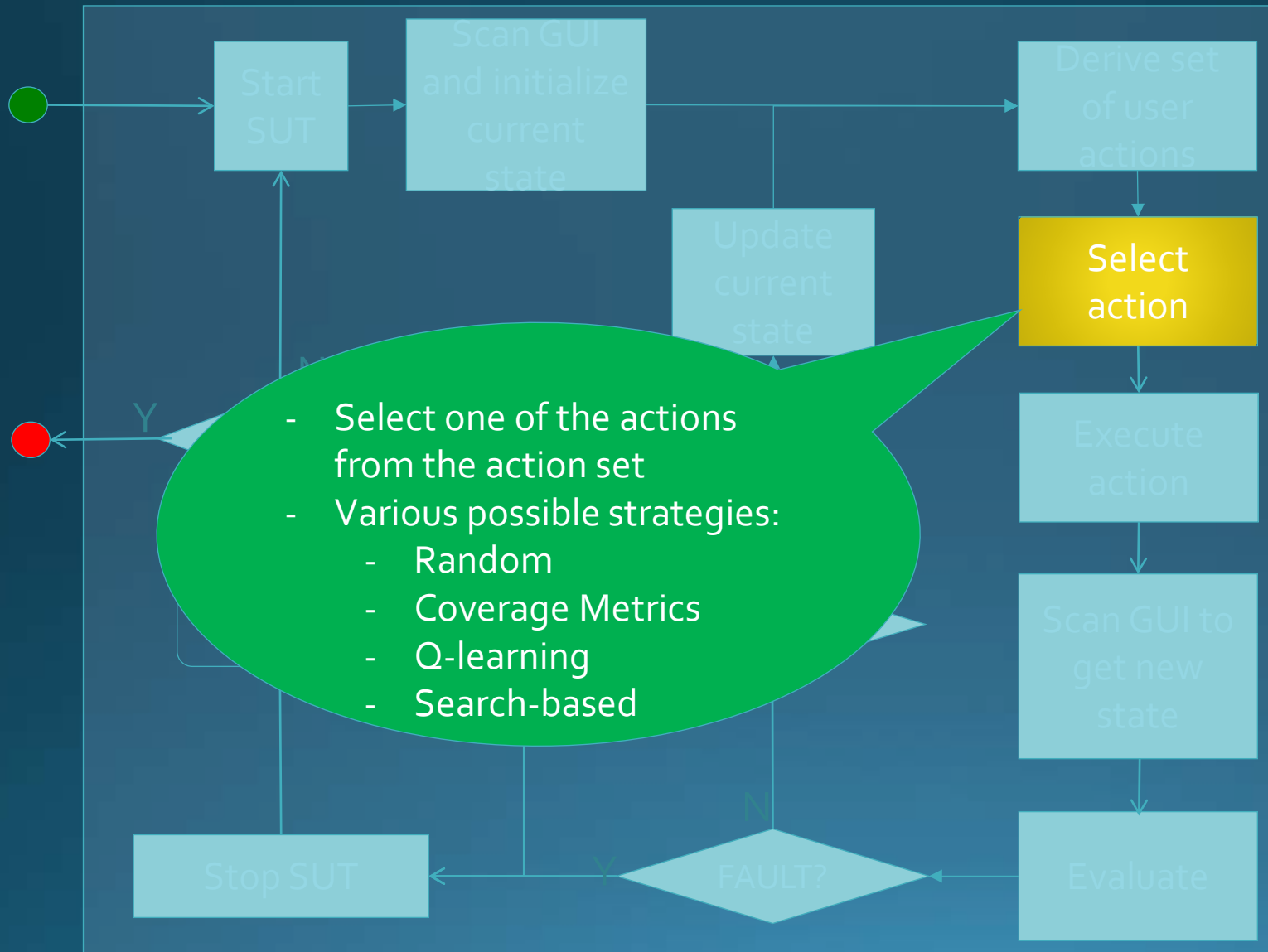


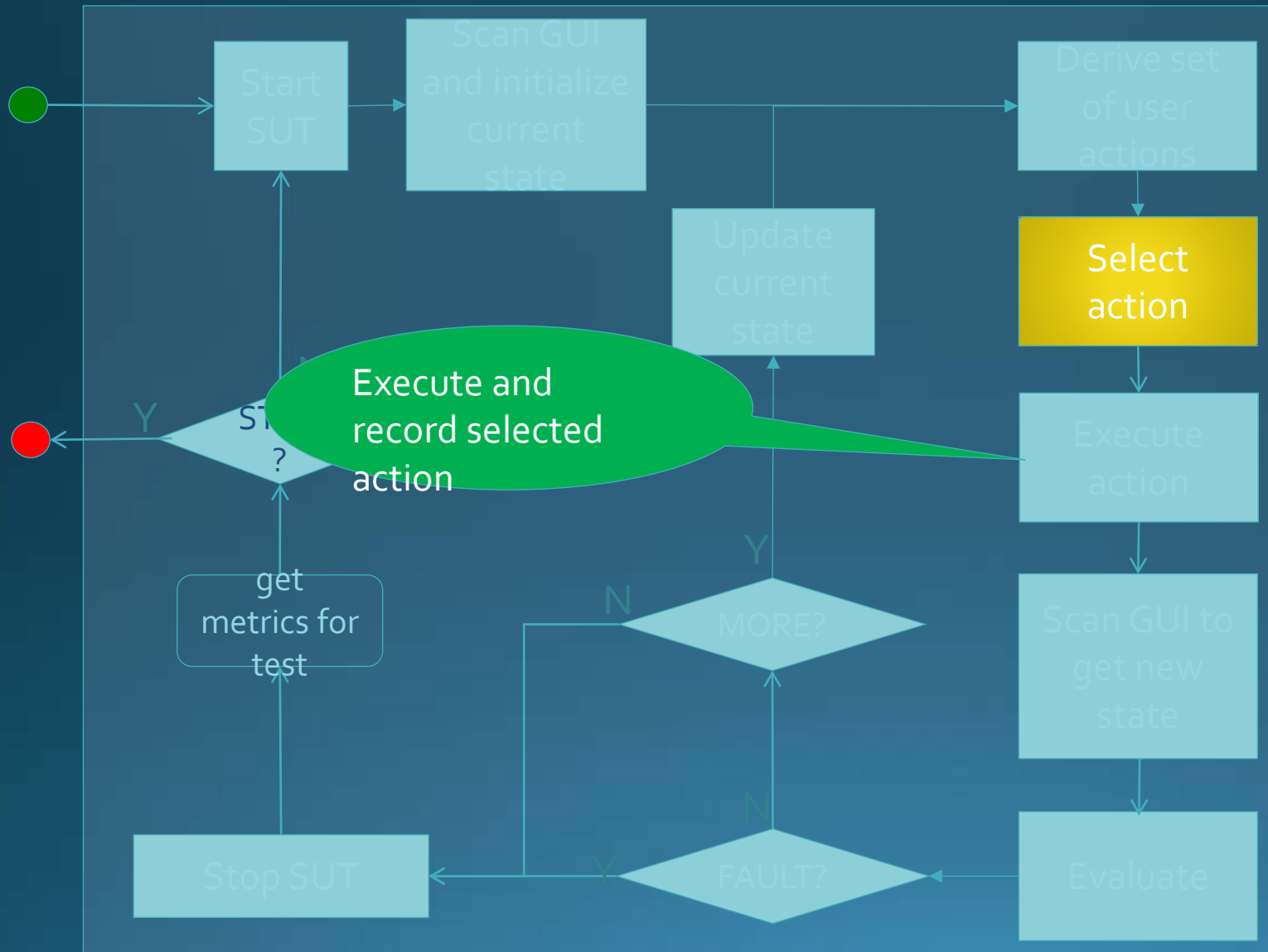
Evolve action selection rules

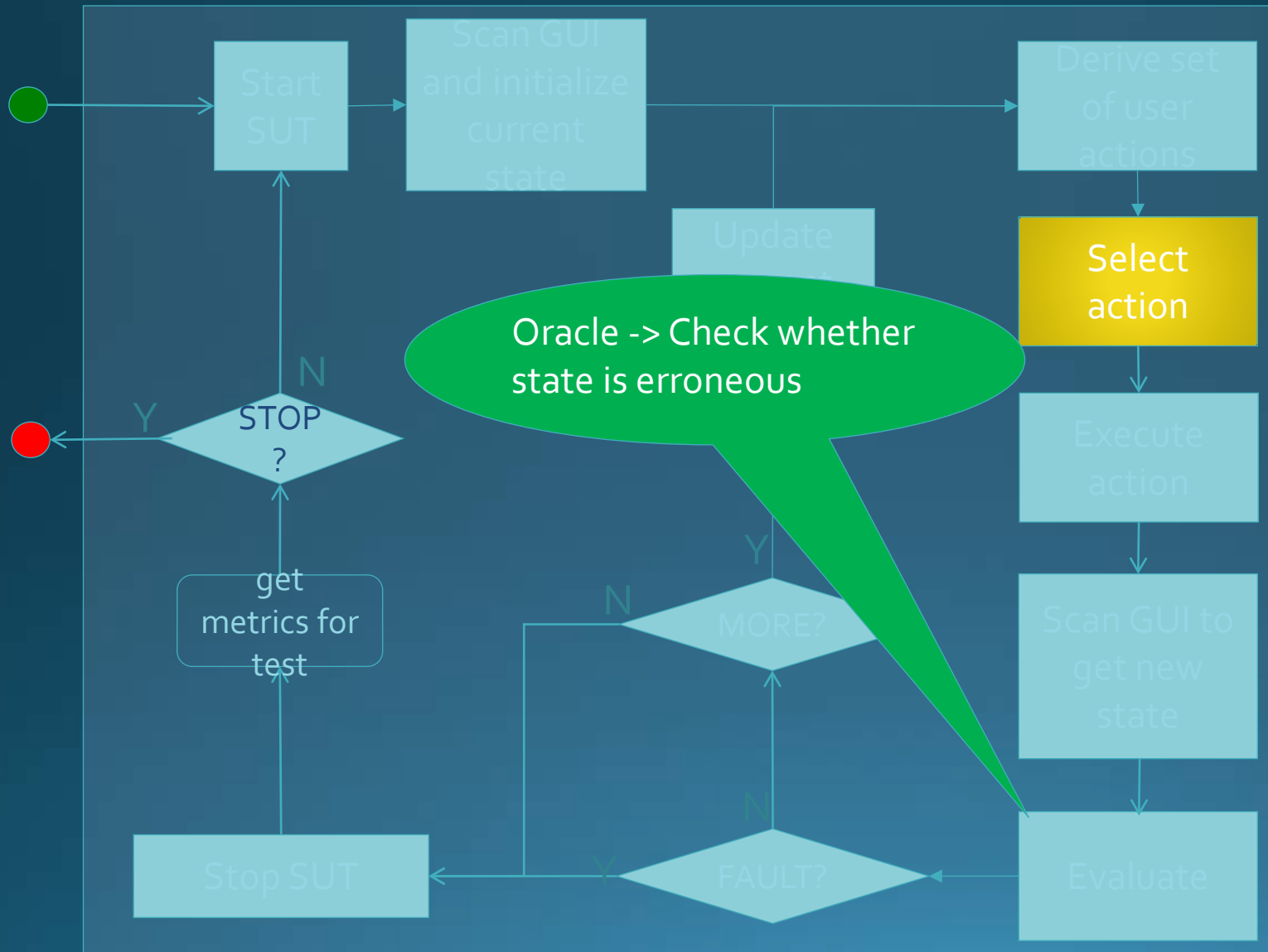


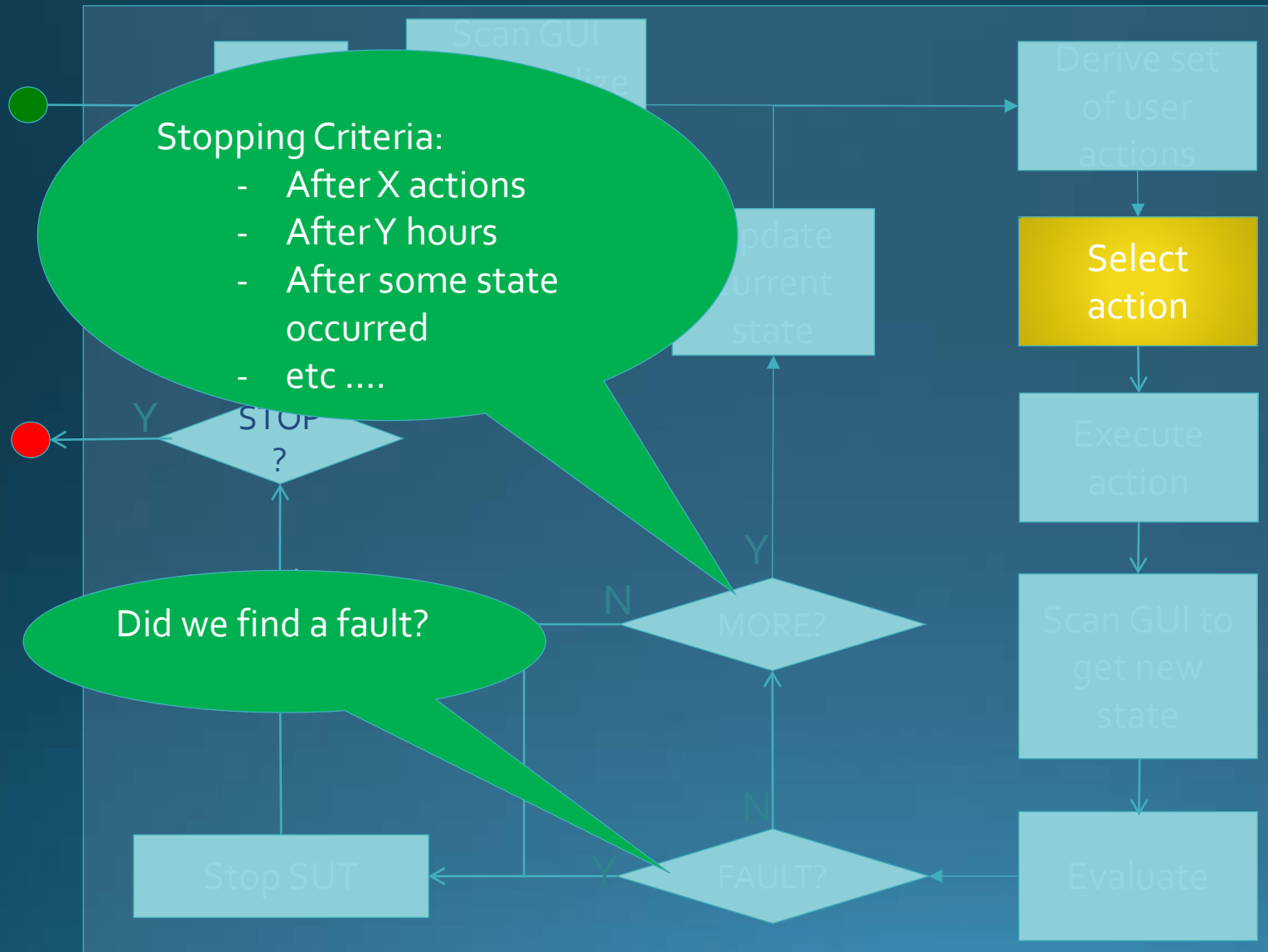


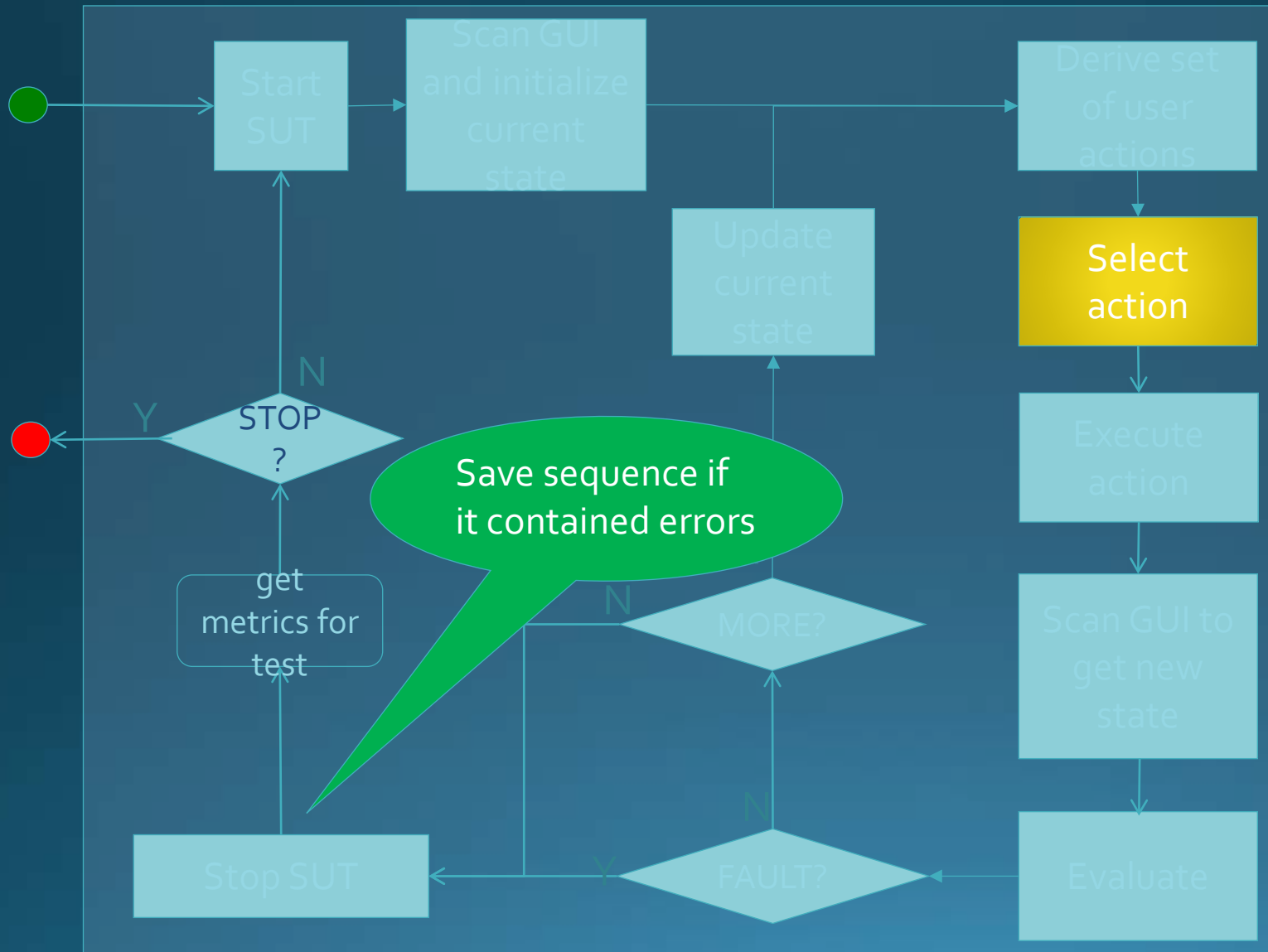


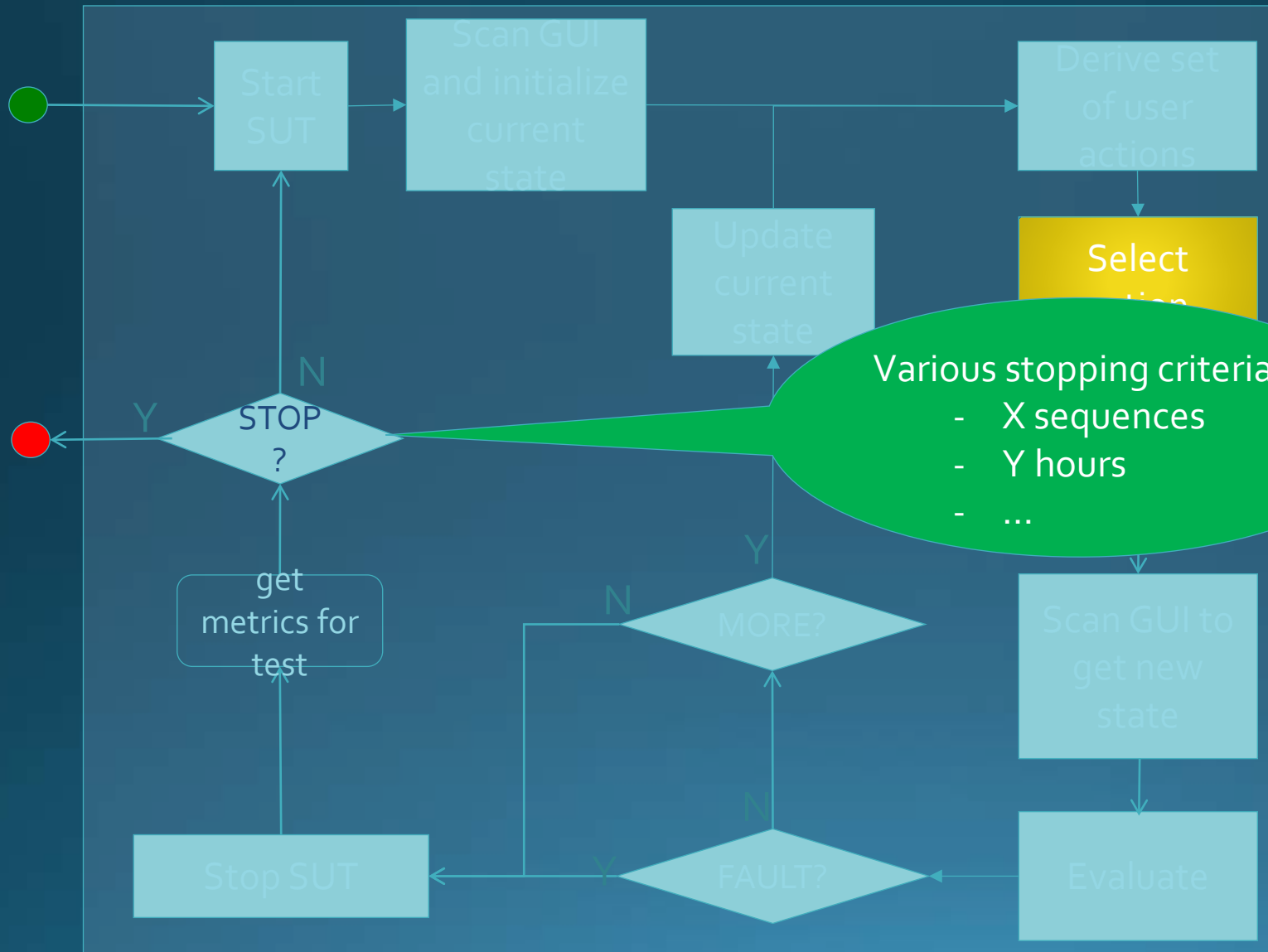












Various stopping criteria:

- X sequences
- Y hours
- ...