10th UCAAT User Conference on Advanced Automated Testing

ETSI The Standards People

Tests Automated Generator Based On Specifications

Cosmin Vîlceanu

Tudor Tomuța

NOKIA

14/11/2023

Agenda

1. Starting point
2. The need for optimizing test writing
3. Tests Automated Generator Based on Specification
4. Benefits
5. Next steps
6. Q&A

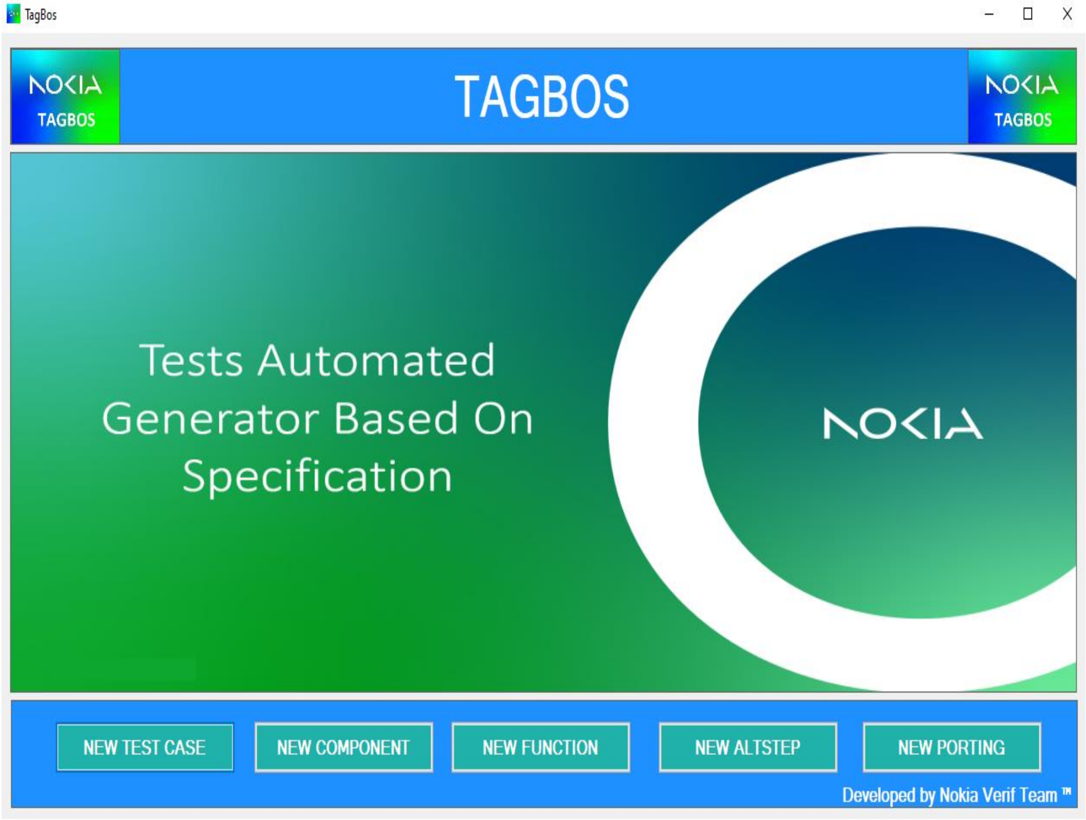
Starting point

- Plane Integration Testing (PIT)
    - We test all applications as part of E2E scenarios
        - All hardware elements are simulated
        - Strong development relationship with software teams
    - Applications are treated as a system, whose external interfaces are being tested (black-box)
- TTCN-3
    - Our testing framework is based on TTCN-3
- Model-based testing (MBT) Technique
    - Models that describes the functional aspects of the system under test
    - Test Cases, Functions, TTCN-3 Components derived from the models
- TAGBOS
    - In house custom interface for test generation

The need for optimizing test writing

- Improve our Testing Plan
  - UML diagrams are used inside our organization, we just find a way to optimize and standardize them
- Reduce time spent on test case writing
- Test earlier and efficiently
  - Everyone relies on the same specification level
- Tool (TAGBOS)
  - User friendly GUI
  - One click away to generate your custom test-cases
  - UML diagram as an input since everyone is familiar with it
  - Generation based on internal Coding Guidelines rules and project structure

Tests Automated Generator Based on Specification

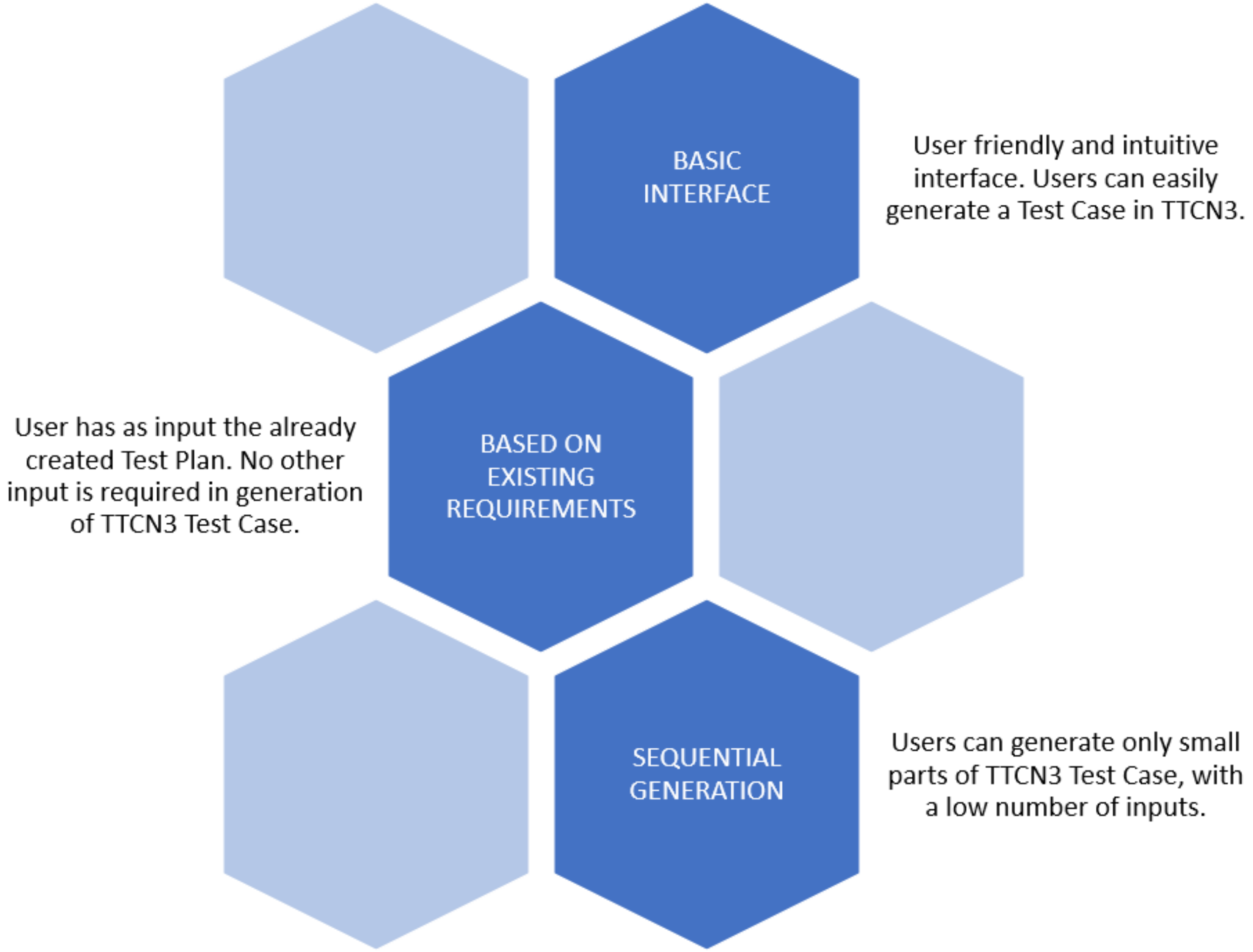


TagBos

TAGBOS

Tests Automated Generator Based On Specification

Developed by Nokia Verif Team ™



BASIC INTERFACE

User friendly and intuitive interface. Users can easily generate a Test Case in TTCN3.

BASED ON EXISTING REQUIREMENTS

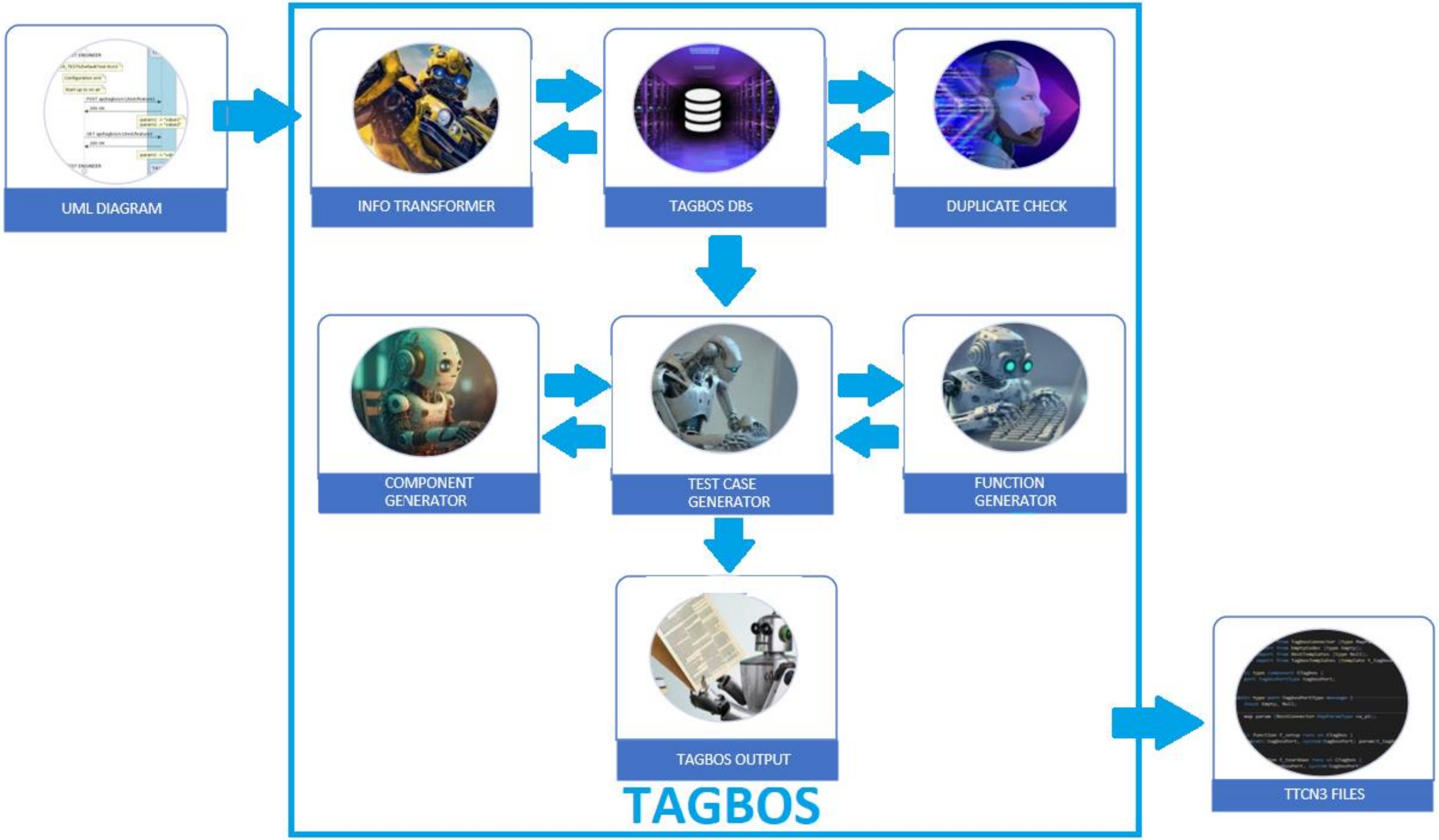
User has as input the already created Test Plan. No other input is required in generation of TTCN3 Test Case.

SEQUENTIAL GENERATION

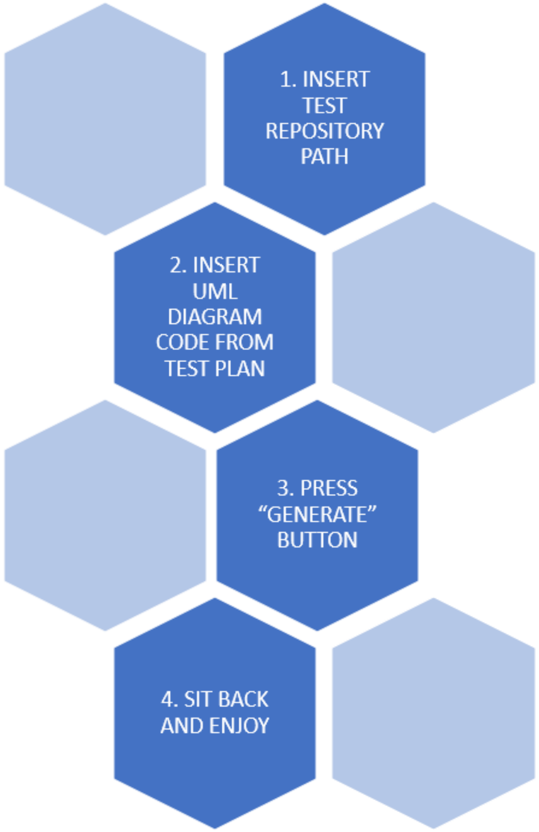
Users can generate only small parts of TTCN3 Test Case, with a low number of inputs.

NOKIA

Structure


UML DIAGRAM
INFO TRANSFORMER
TAGBOS DBs
DUPLICATE CHECK
COMPONENT GENERATOR
TEST CASE GENERATOR
FUNCTION GENERATOR
TAGBOS OUTPUT
TAGBOS
TTCN3 FILES

Test Case Generator

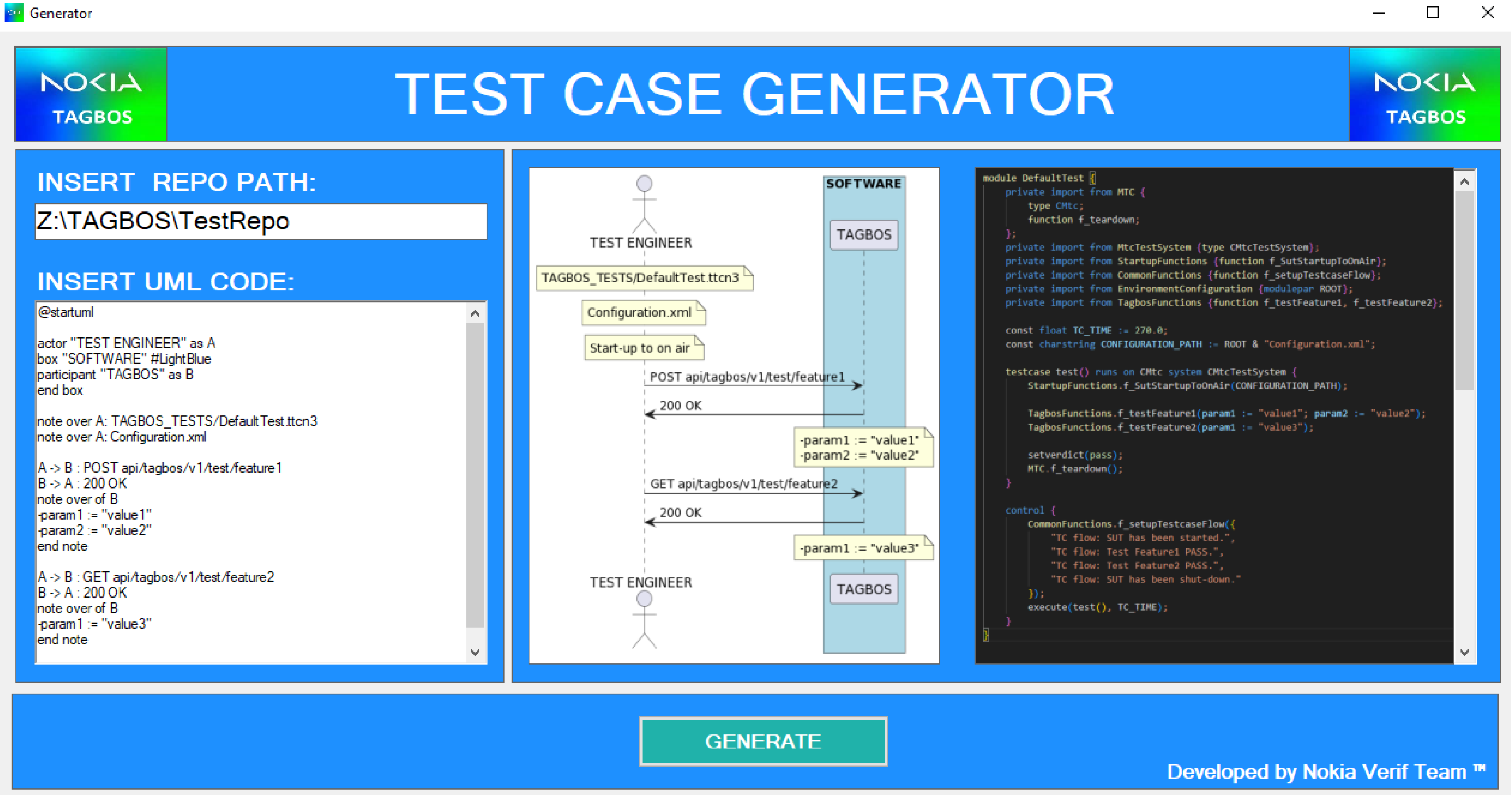


1. INSERT TEST REPOSITORY PATH

2. INSERT UML DIAGRAM CODE FROM TEST PLAN

3. PRESS “GENERATE” BUTTON

4. SIT BACK AND ENJOY



TEST CASE GENERATOR

INSERT REPO PATH:

Z:\TAGBOS\TestRepo

INSERT UML CODE:

```
@startuml

actor "TEST ENGINEER" as A
box "SOFTWARE" #LightBlue
participant "TAGBOS" as B
end box

note over A: TAGBOS_TESTS/DefaultTest.ttcn3
note over A: Configuration.xml

A -> B : POST api/tagbos/v1/test/feature1
B -> A : 200 OK
note over of B
-param1 := "value1"
-param2 := "value2"
end note

A -> B : GET api/tagbos/v1/test/feature2
B -> A : 200 OK
note over of B
-param1 := "value3"
end note
```

```
module DefaultTest {
    private import from MTC {
        type CMtc;
        function f_teardown;
    };
    private import from MtcTestSystem {type CMtcTestSystem};
    private import from StartupFunctions {function f_SutStartupToOnAir};
    private import from CommonFunctions {function f_setupTestcaseFlow};
    private import from EnvironmentConfiguration {modulepar ROOT};
    private import from TagbosFunctions {function f_testFeature1, f_testFeature2};

    const float TC_TIME := 270.0;
    const charstring CONFIGURATION_PATH := ROOT & "Configuration.xml";

    testcase test() runs on CMtc system CMtcTestSystem {
        StartupFunctions.f_SutStartupToOnAir(CONFIGURATION_PATH);

        TagbosFunctions.f_testFeature1(param1 := "value1"; param2 := "value2");
        TagbosFunctions.f_testFeature2(param1 := "value3");

        setverdict(pass);
        MTC.f_teardown();
    }

    control {
        CommonFunctions.f_setupTestcaseFlow({
            "TC flow: SUT has been started.",
            "TC flow: Test Feature1 PASS.",
            "TC flow: Test Feature2 PASS.",
            "TC flow: SUT has been shut-down."
        });
        execute(test(), TC_TIME);
    }
}
```

GENERATE

Developed by Nokia Verif Team ™

Test Case Generator

INFO TRANSFORMER

TAKES UML DIAGRAM CODE AND SPLIT ITS LINES IN REQUIREMENTS. THOSE LINES ARE INSERTED IN SPECIFIC DATABASE

TAGBOS DBs

DATABASES ARE SPLIT IN MORE CATEGORIES AND MAKES THE TRANSFORMATION OF REQUIREMENTS IN MODELS MUCH EASIER

DUPLICATE CHECK

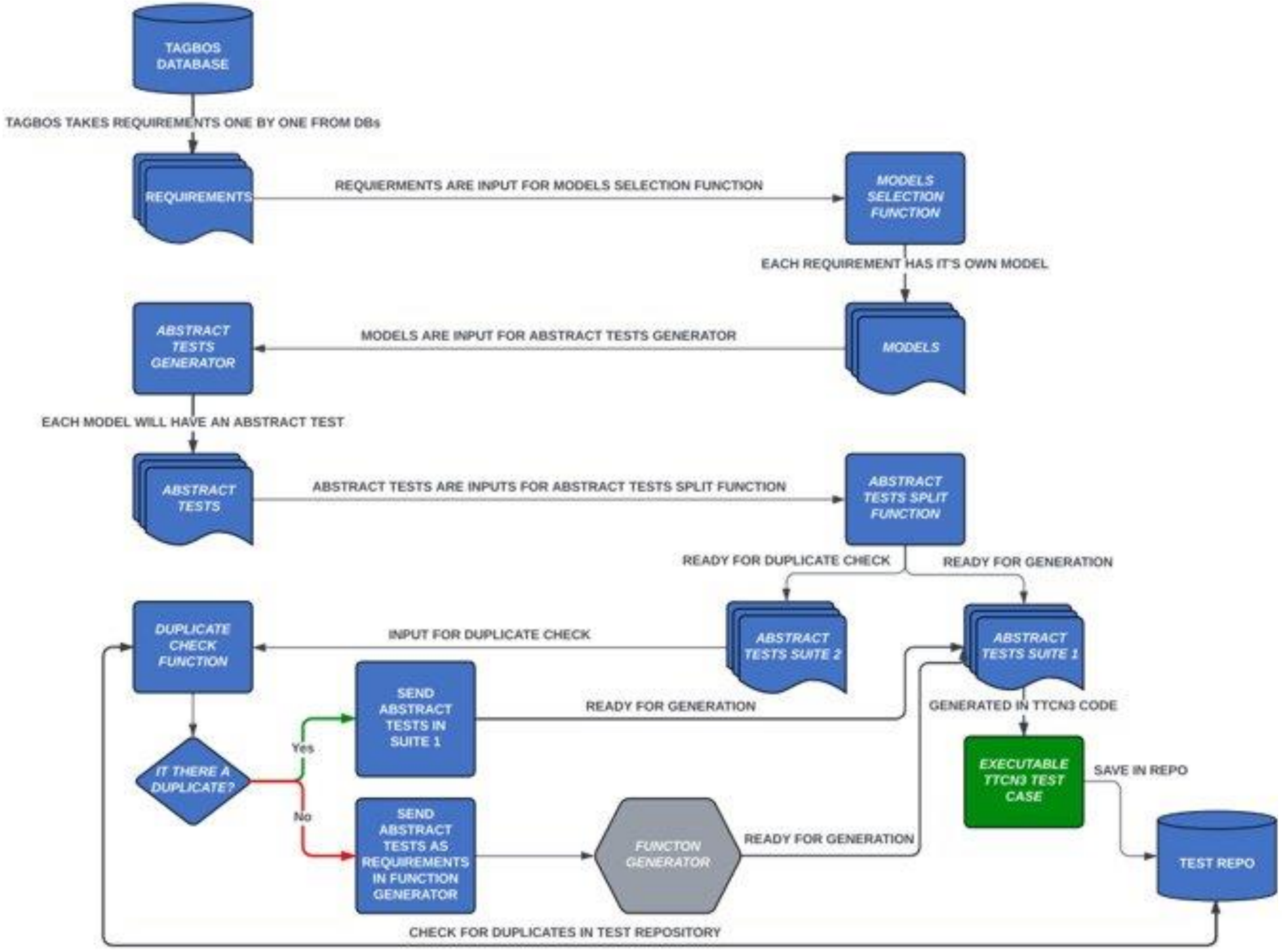
TAKES ALL THE INFO FORM ABSTRACT TESTS AND COMPARE THEM WITH EXISTING CODE IN TEST REPOSITORY

TAGBOS OUTPUT

TAKES ALL THE ABSTRACT TESTS AND TRANFORMS THEM IN TTCN3 CODE

Test Case Generator

TEST CASE GENERATOR


TAGBOS DATABASE
TAGBOS TAKES REQUIREMENTS ONE BY ONE FROM DBs
REQUIREMENTS
REQUIERMENTS ARE INPUT FOR MODELS SELECTION FUNCTION
MODELS SELECTION FUNCTION
EACH REQUIREMENT HAS IT'S OWN MODEL
MODELS
MODELS ARE INPUT FOR ABSTRACT TESTS GENERATOR
ABSTRACT TESTS GENERATOR
EACH MODEL WILL HAVE AN ABSTRACT TEST
ABSTRACT TESTS
ABSTRACT TESTS ARE INPUTS FOR ABSTRACT TESTS SPLIT FUNCTION
ABSTRACT TESTS SPLIT FUNCTION
READY FOR DUPLICATE CHECK
READY FOR GENERATION
ABSTRACT TESTS SUITE 2
ABSTRACT TESTS SUITE 1
INPUT FOR DUPLICATE CHECK
DUPLICATE CHECK FUNCTION
SEND ABSTRACT TESTS IN SUITE 1
READY FOR GENERATION
GENERATED IN TTCN3 CODE
Yes
IT THERE A DUPLICATE?
No
EXECUTABLE TTCN3 TEST CASE
SAVE IN REPO
SEND ABSTRACT TESTS AS REQUIREMENTS IN FUNCTION GENERATOR
FUNCTON GENERATOR
READY FOR GENERATION
TEST REPO
CHECK FOR DUPLICATES IN TEST REPOSITORY

THE UML DIAGRAM IS CHANGED IN REQUIREMENTS BY INFO TRANSFORMER

INPUT:
13 DIFFERENT REQUIREMENTS FROM DATABASES

MODELS:
13 DIFFERENT MODELS, ONE FOR EACH REQUIREMENT

EACH MODEL IS DEVELOPED ACORDING WITH NOKIA CODING GUIDLINES

THE 2 NEW REQUIREMENTS WILL BE INTRODUCED IN THE NEXT 2 GENERATORS AND THERE WILL BE CHANGED IN 2 NEW ABSTRACT SUITES

ABSTRACT OUTPUT:

11 ABSTRACT SUITES

+

2 NEW REQUIREMENTS

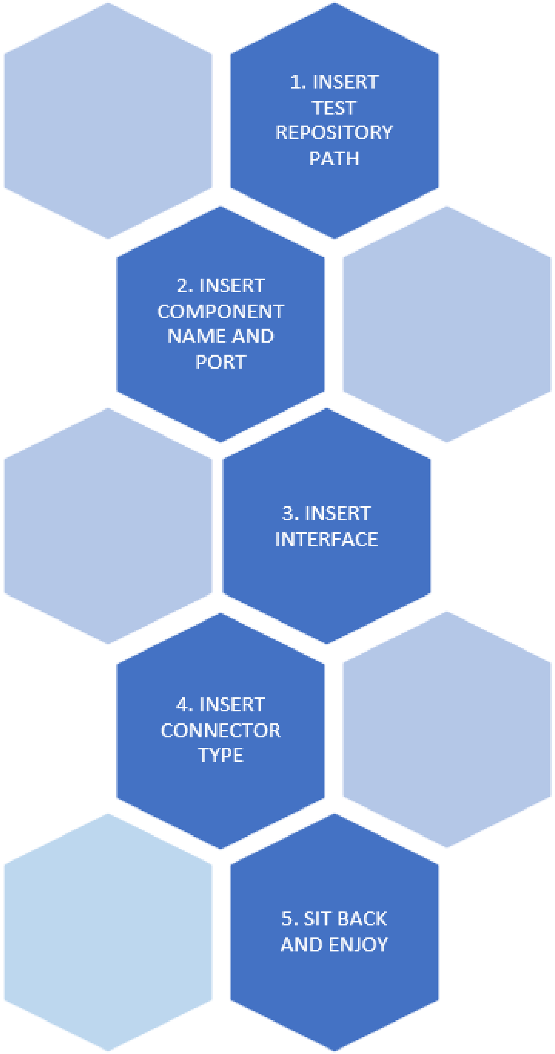
4. EXECUTABLE OUTPUT:

1 TTCN3 TEST CASE DIRECTLY IN TEST REPOSITORY

THE TTCN3 TEST CASE WILL CONTAIN ALL ABSTRACTS SUITES FROM ALL 3 GENERATOS

NOKIA

Component Generator



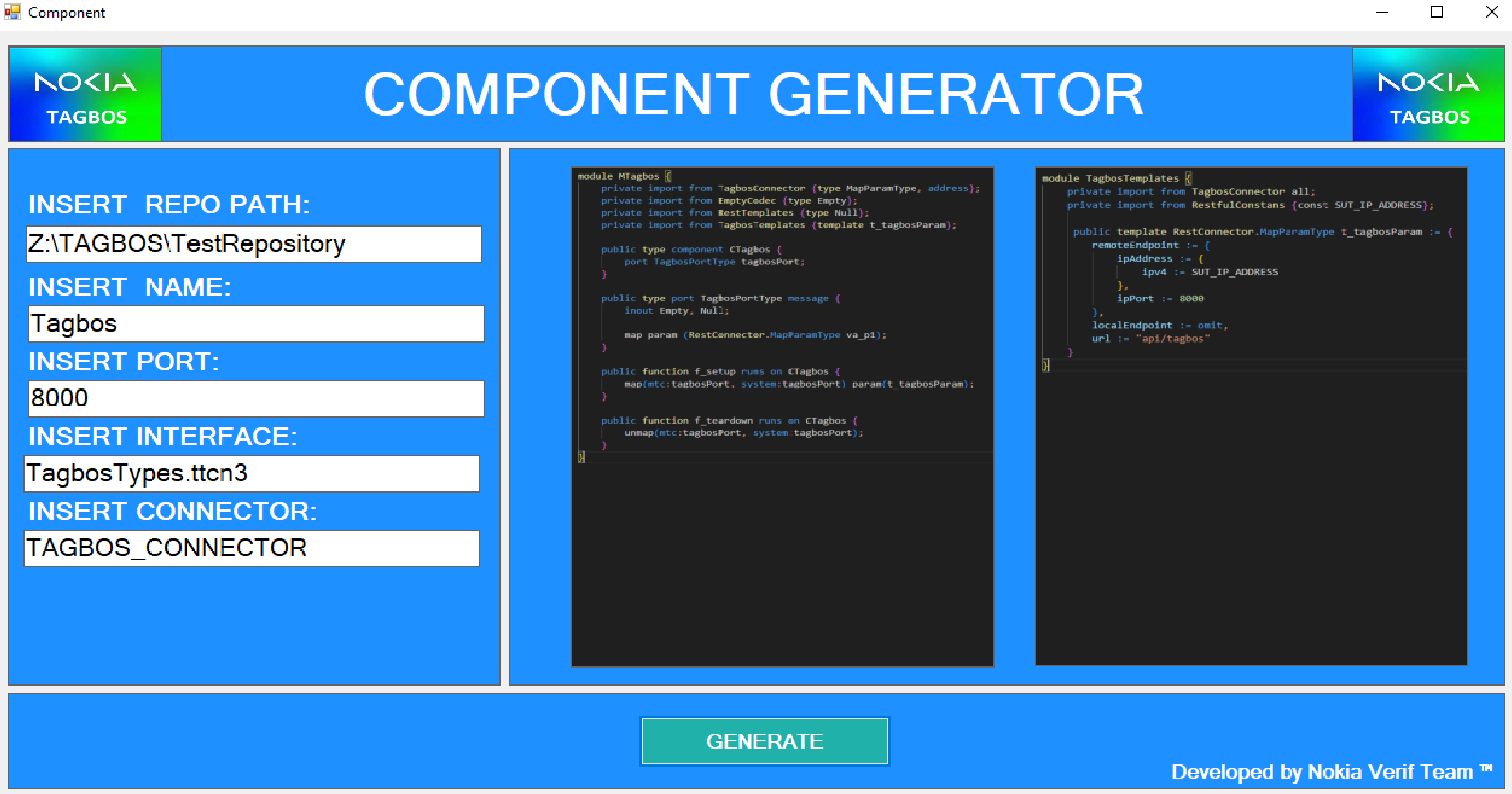
1. INSERT TEST REPOSITORY PATH

2. INSERT COMPONENT NAME AND PORT

3. INSERT INTERFACE

4. INSERT CONNECTOR TYPE

5. SIT BACK AND ENJOY



COMPONENT GENERATOR

INSERT REPO PATH: Z:\TAGBOS\TestRepository

INSERT NAME: Tagbos

INSERT PORT: 8000

INSERT INTERFACE: TagbosTypes.ttcn3

INSERT CONNECTOR: TAGBOS_CONNECTOR

```
module MTagbos {
    private import from TagbosConnector {type MapParamType, address};
    private import from EmptyCodec {type Empty};
    private import from RestTemplates {type Null};
    private import from TagbosTemplates {template t_tagbosParam};

    public type component CTagbos {
        port TagbosPortType tagbosPort;
    }

    public type port TagbosPortType message {
        inout Empty, Null;

        map param (RestConnector.MapParamType va_p1);
    }

    public function f_setup runs on CTagbos {
        map(mtc:tagbosPort, system:tagbosPort) param(t_tagbosParam);
    }

    public function f_teardown runs on CTagbos {
        unmap(mtc:tagbosPort, system:tagbosPort);
    }
}
```

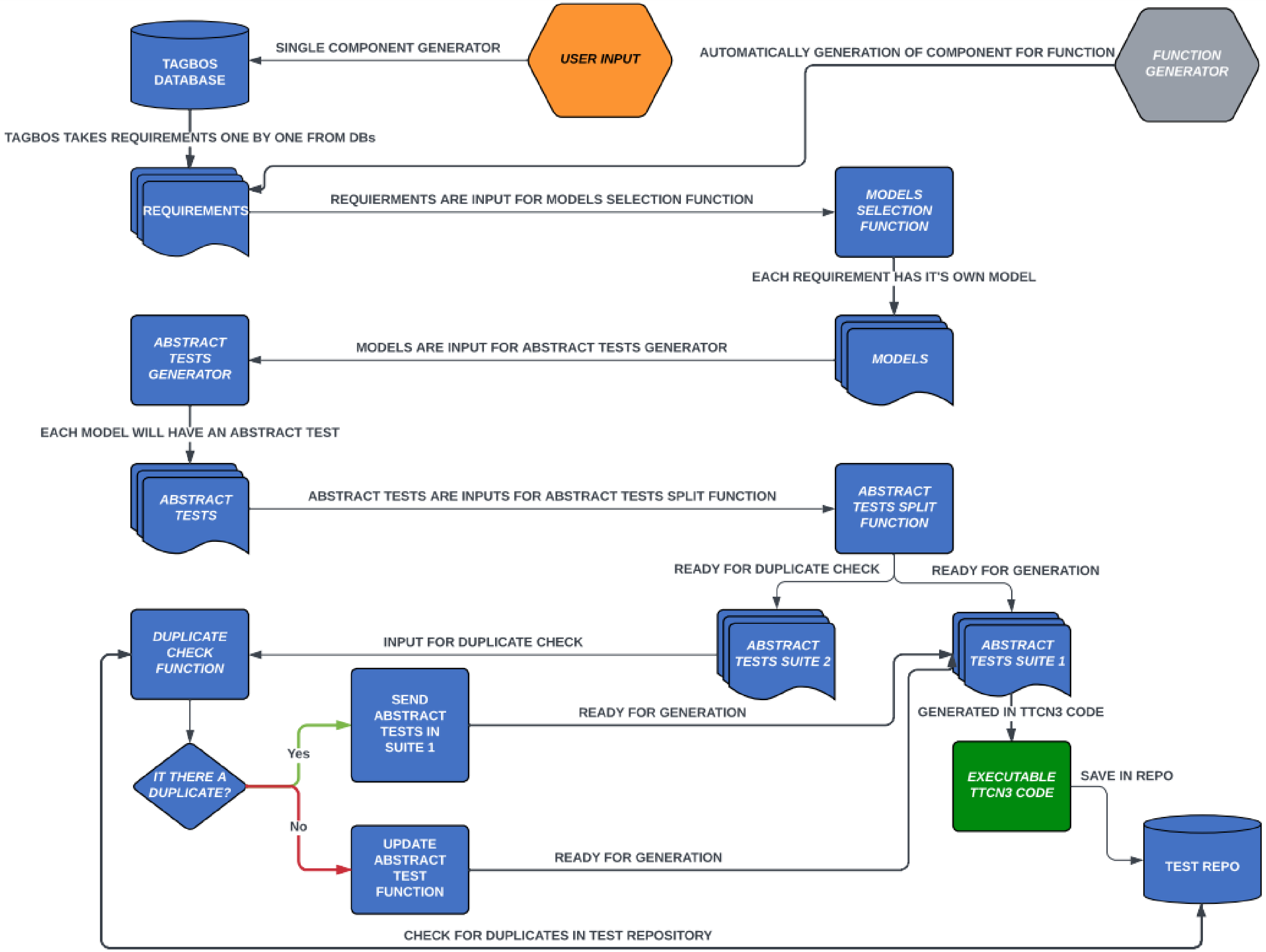
```
module TagbosTemplates {
    private import from TagbosConnector all;
    private import from RestfulConstans {const SUT_IP_ADDRESS};

     public template RestConnector.MapParamType t_tagbosParam := {
        remoteEndpoint := {
            ipAddress := {
                ipv4 := SUT_IP_ADDRESS
            },
            ipPort := 8000
        },
        localEndpoint := omit,
        url := "api/tagbos"
    }
}
```

Developed by Nokia Verif Team ™

Component Generator

COMPONENT GENERATOR


TAGBOS DATABASE
SINGLE COMPONENT GENERATOR
USER INPUT
AUTOMATICALLY GENERATION OF COMPONENT FOR FUNCTION
FUNCTION GENERATOR
TAGBOS TAKES REQUIREMENTS ONE BY ONE FROM DBs
REQUIREMENTS
REQUIERMENTS ARE INPUT FOR MODELS SELECTION FUNCTION
MODELS SELECTION FUNCTION
EACH REQUIREMENT HAS IT'S OWN MODEL
ABSTRACT TESTS GENERATOR
MODELS ARE INPUT FOR ABSTRACT TESTS GENERATOR
MODELS
EACH MODEL WILL HAVE AN ABSTRACT TEST
ABSTRACT TESTS
ABSTRACT TESTS ARE INPUTS FOR ABSTRACT TESTS SPLIT FUNCTION
ABSTRACT TESTS SPLIT FUNCTION
READY FOR DUPLICATE CHECK
READY FOR GENERATION
DUPLICATE CHECK FUNCTION
INPUT FOR DUPLICATE CHECK
ABSTRACT TESTS SUITE 2
ABSTRACT TESTS SUITE 1
SEND ABSTRACT TESTS IN SUITE 1
READY FOR GENERATION
GENERATED IN TTCN3 CODE
Yes
IT THERE A DUPLICATE?
No
EXECUTABLE TTCN3 CODE
SAVE IN REPO
UPDATE ABSTRACT TEST FUNCTION
READY FOR GENERATION
TEST REPO
CHECK FOR DUPLICATES IN TEST REPOSITORY

THE USER INPUTS ARE CHANGED IN REQUIREMENTS BY INFO TRANSFORMER OR REQUIREMENTS ARE COMING DIRECTLY FROM FUNCTION GENERATOR

INPUT:
14 DIFFERENT REQUIREMENTS FROM DATABASES AND FUNCTION GENERATOR

MODELS:
14 DIFFERENT MODELS, ONE FOR EACH REQUIREMENT

EACH MODEL IS DEVELOPED ACORDING WITH NOKIA CODING GUIDLINES

ALL ABSTRACT TESTS ARE CHECK AGAINST CODE DUPLICATION AND CODING GUIDLINES

ABSTRACT OUTPUT:

14 ABSTRACT SUITES

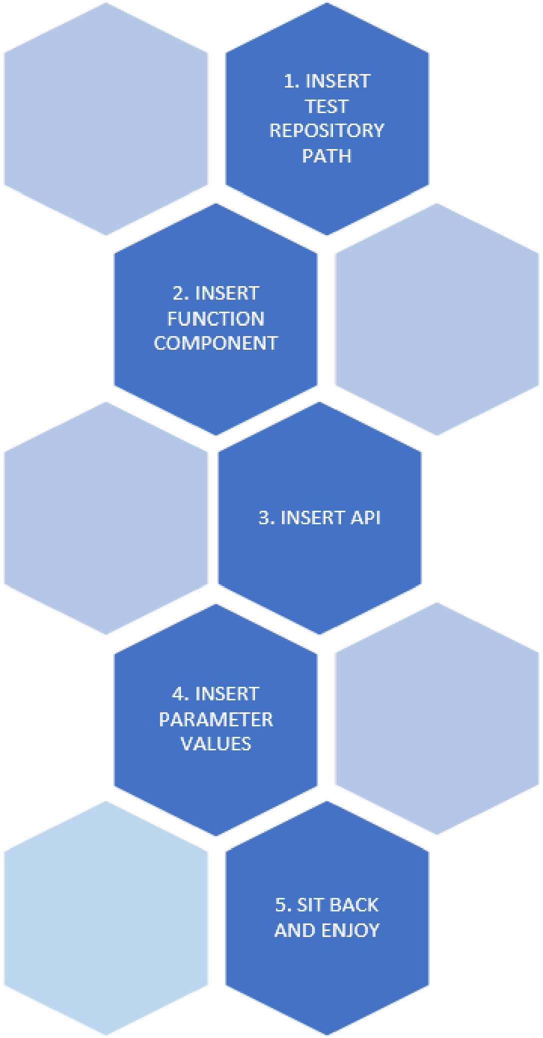
4. EXECUTABLE OUTPUT:

1 TTCN3 COMPONENT IN TEST REPO

IF COMPONENT GENERATOR IS USED STANDALONE WILL CREATE THE COMPONENT ONLY IN TEST REPO, IF IS USED THROUH TEST CASE GENERATOR WILL ADD ALSO THE COMPONENT ON NEWLY ADDED FUNCTIONS

NOKIA

Function Generator



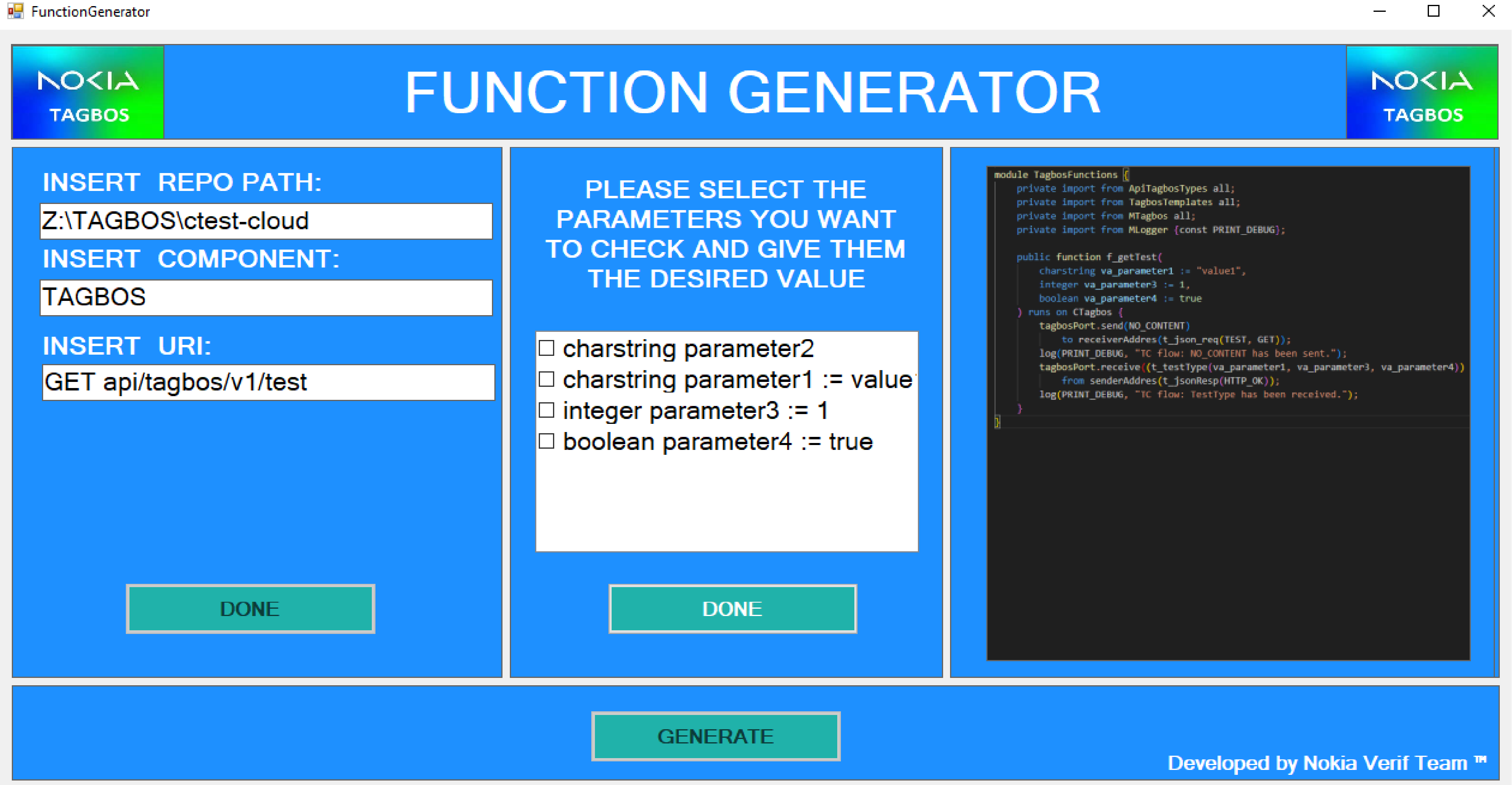
1. INSERT TEST REPOSITORY PATH

2. INSERT FUNCTION COMPONENT

3. INSERT API

4. INSERT PARAMETER VALUES

5. SIT BACK AND ENJOY



INSERT REPO PATH:

Z:\TAGBOS\ctest-cloud

INSERT COMPONENT:

TAGBOS

INSERT URI:

GET api/tagbos/v1/test

PLEASE SELECT THE PARAMETERS YOU WANT TO CHECK AND GIVE THEM THE DESIRED VALUE

- ☐ charstring parameter2
- ☐ charstring parameter1 := value
- ☐ integer parameter3 := 1
- ☐ boolean parameter4 := true

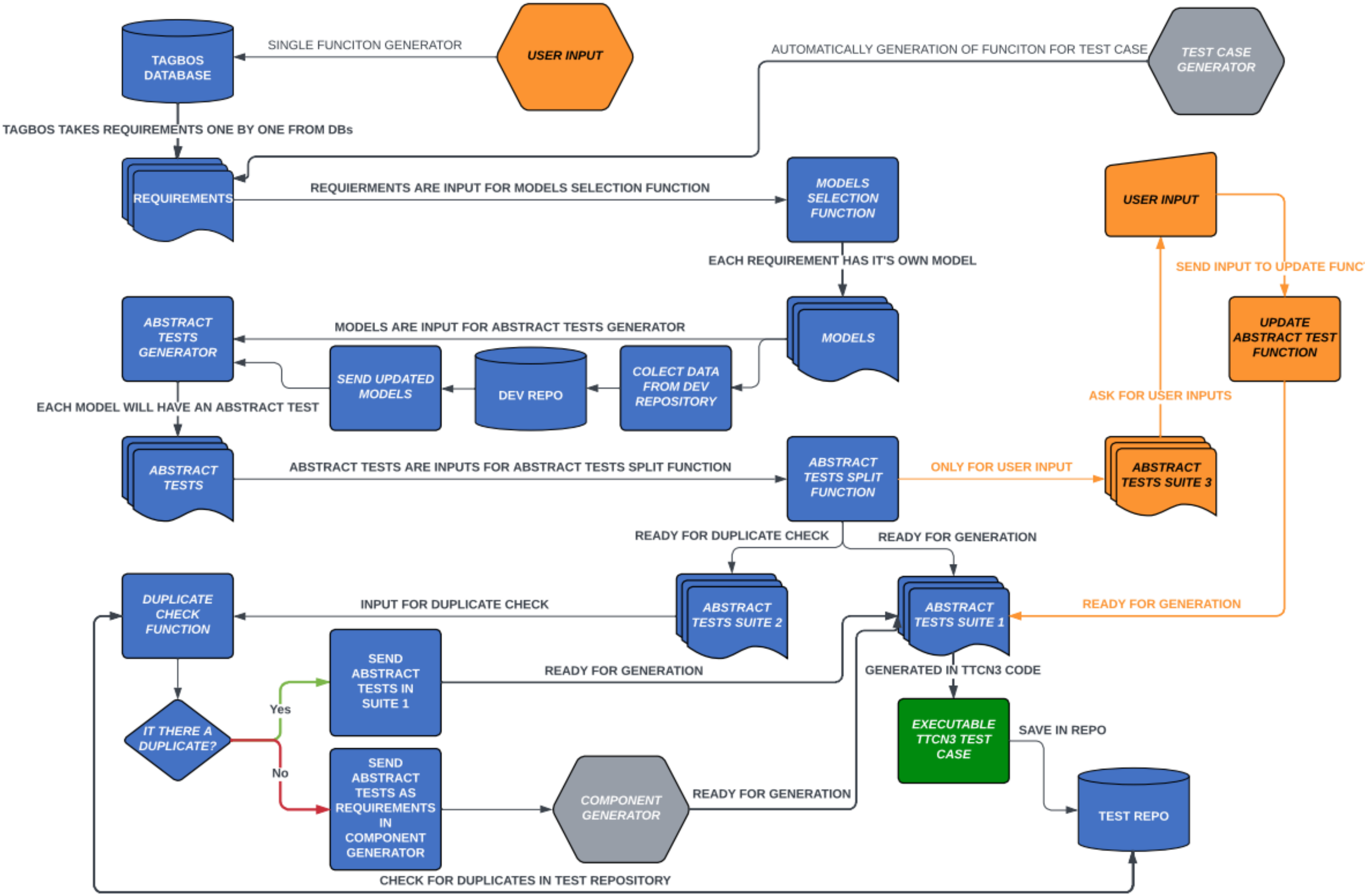
```
module TagbosFunctions {
    private import from ApiTagbosTypes all;
    private import from TagbosTemplates all;
    private import from MTagbos all;
    private import from MLogger {const PRINT_DEBUG};

    public function f_getTest(
        charstring va_parameter1 := "value1",
        integer va_parameter3 := 1,
        boolean va_parameter4 := true
    ) runs on CTagbos {
        tagbosPort.send(NO_CONTENT)
            to receiverAddres(t_json_req(TEST, GET));
        log(PRINT_DEBUG, "TC flow: NO_CONTENT has been sent.");
        tagbosPort.receive((t_testType(va_parameter1, va_parameter3, va_parameter4))
            from senderAddres(t_jsonResp(HTTP_OK));
        log(PRINT_DEBUG, "TC flow: TestType has been received.");
    }
}
```

Developed by Nokia Verif Team ™

Function Generator

FUNCTION GENERATOR



TAGBOS DATABASE

SINGLE FUNCITON GENERATOR

USER INPUT

AUTOMATICALLY GENERATION OF FUNCITON FOR TEST CASE

TEST CASE GENERATOR

TAGBOS TAKES REQUIREMENTS ONE BY ONE FROM DBs

REQUIREMENTS

REQUIERMENTS ARE INPUT FOR MODELS SELECTION FUNCTION

MODELS SELECTION FUNCTION

USER INPUT

SEND INPUT TO UPDATE FUNC

EACH REQUIREMENT HAS IT'S OWN MODEL

ABSTRACT TESTS GENERATOR

MODELS ARE INPUT FOR ABSTRACT TESTS GENERATOR

MODELS

UPDATE ABSTRACT TEST FUNCTION

SEND UPDATED MODELS

DEV REPO

COLECT DATA FROM DEV REPOSITORY

ASK FOR USER INPUTS

EACH MODEL WILL HAVE AN ABSTRACT TEST

ABSTRACT TESTS

ABSTRACT TESTS ARE INPUTS FOR ABSTRACT TESTS SPLIT FUNCTION

ABSTRACT TESTS SPLIT FUNCTION

ONLY FOR USER INPUT

ABSTRACT TESTS SUITE 3

READY FOR DUPLICATE CHECK

READY FOR GENERATION

DUPLICATE CHECK FUNCTION

INPUT FOR DUPLICATE CHECK

ABSTRACT TESTS SUITE 2

ABSTRACT TESTS SUITE 1

READY FOR GENERATION

SEND ABSTRACT TESTS IN SUITE 1

READY FOR GENERATION

GENERATED IN TTCN3 CODE

Yes

IT THERE A DUPLICATE?

No

EXECUTABLE TTCN3 TEST CASE

SAVE IN REPO

SEND ABSTRACT TESTS AS REQUIREMENTS IN COMPONENT GENERATOR

COMPONENT GENERATOR

READY FOR GENERATION

TEST REPO

CHECK FOR DUPLICATES IN TEST REPOSITORY

THE USER INPUTS ARE CHANGED IN REQUIREMENTS BY INFO TRANSFORMER OR REQUIREMENTS ARE COMING DIRECTLY FROM TEST CASE GENERATOR

INPUT:
13 DIFFERENT REQUIREMENTS FROM DATABASES AND TEST CASE GENERATOR

MODELS:
13 DIFFERENT MODELS, ONE FOR EACH REQUIREMENT

EACH MODEL IS DEVELOPED ACORDING WITH NOKIA CODING GUIDLINES

THE NEW REQUIREMENT WILL BE INTORDUCED IN THE COMPONENT GENERATOR AND THERE WILL BE CHANGED IN NEW ABSTRACT TEST

ABSTRACT OUTPUT:

12 ABSTRACT SUITES

+

1 NEW REQUIREMENT

4. EXECUTABLE OUTPUT:

1 TTCN3 FUNCTION IN TEST REPO AND TEST CASE

IF FUNCTION GENERATOR IS USED STANDALONE WILL CREATE THE FUNCTION ONLY IN TEST REPO, IF IS USED THROUH TEST CASE GENERATOR WILL ADD ALSO THE FUNCTION IN TEST CASE

Used Models

TEST CASE GENERATOR

HEADER
MODULE
IMPORTS
CONTANTS
TESTCASE
CONTROL
MTC AND SUT START-UP
MTC AND SUT SHUT-DOWN
EXISTING TTCN3 FUNCTION

ABSTRACT SUITE 1 MODELS

ABSTRACT SUITE 2 MODELS

SCENARIO SPECIFIC FUNCTION

NEW TTCN3 FUNCTION

NEW TTCN3 COMPONENT

EXISTING TTCN3 COMPONENT

COMPONENT GENERATOR

API
NEW API
EXISTING API
HTTP TEMPLATE
METHOD
PARAMETERS

ABSTRACT SUITE 1 MODELS

ABSTRACT SUITE 2 MODELS

EXISTING PARAMETERS
NEW PARAMETERS
EXISTING VALUES
NEW VALUES
NEW COMPONENT
EXISTING COMPONENT

VALUES

ABSTRACT SUITE 3 MODELS

FUNCTION GENERATOR

COMPONENT
MTC INTEGRATION
PORT MAPPING
TEMPLATE
FUNCTIONS
SUT SETUP
SUT CLEAN-UP
MTC SETUP
MTC CLEAN-UP

ABSTRACT SUITE 1 MODELS

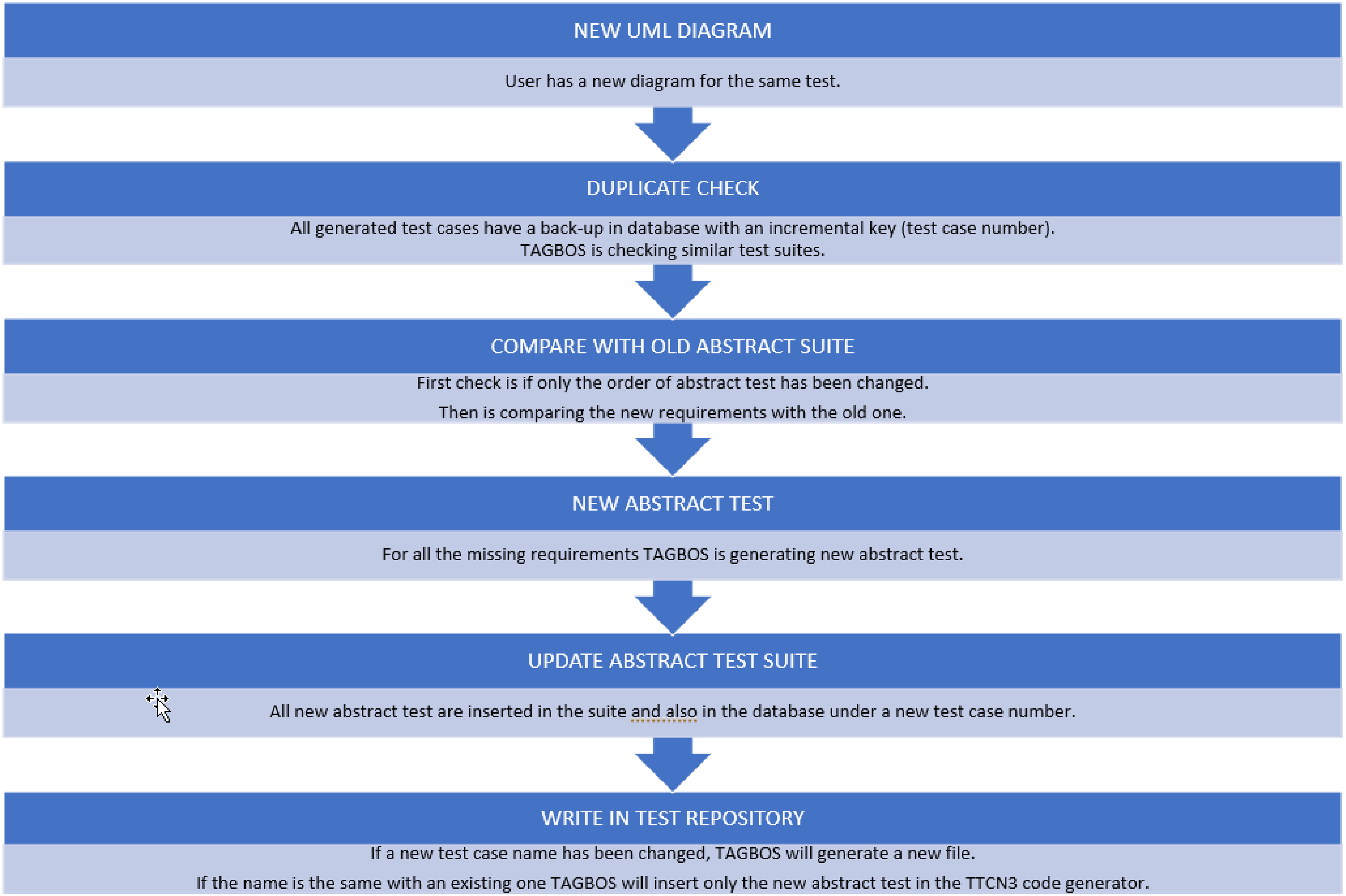
ABSTRACT SUITE 2 MODELS

TEST CONNECTOR
INTERFACE PORT
PORT DEFINITION
IMPORTS
PORT PARAMETERS

Test Case Regeneration

TEST CASE GENERATOR

TAGBOS DBs



NEW UML DIAGRAM

User has a new diagram for the same test.

DUPLICATE CHECK

All generated test cases have a back-up in database with an incremental key (test case number).
TAGBOS is checking similar test suites.

COMPARE WITH OLD ABSTRACT SUITE

First check is if only the order of abstract test has been changed.

Then is comparing the new requirements with the old one.

NEW ABSTRACT TEST

For all the missing requirements TAGBOS is generating new abstract test.

UPDATE ABSTRACT TEST SUITE

All new abstract test are inserted in the suite and also in the database under a new test case number.

WRITE IN TEST REPOSITORY

If a new test case name has been changed, TAGBOS will generate a new file.

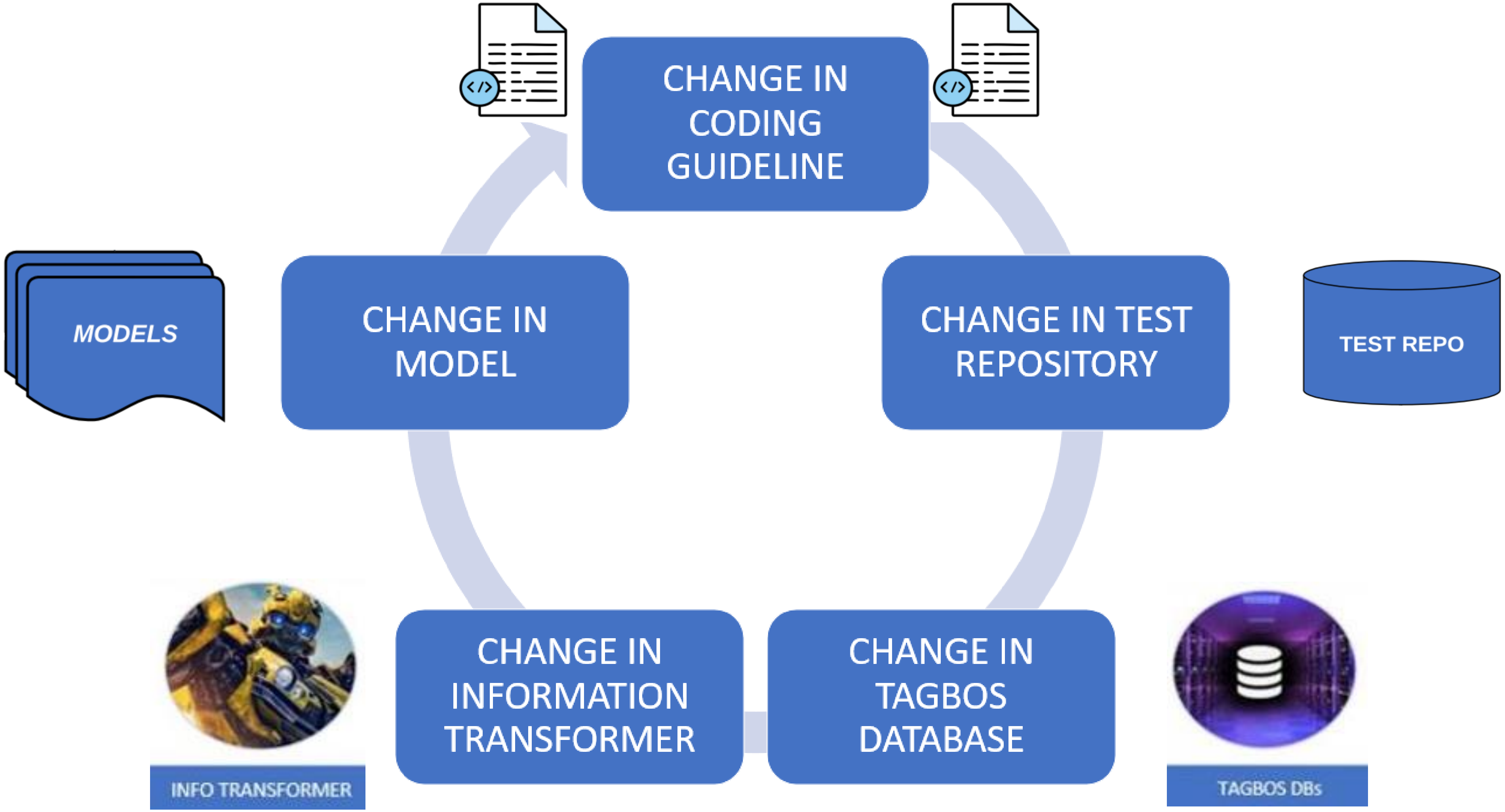
If the name is the same with an existing one TAGBOS will insert only the new abstract test in the TTCN3 code generator.

DUPLICATE CHECK

TAGBOS OUTPUT

NOKIA

Maintenance



CHANGE IN CODING GUIDELINE

CHANGE IN TEST REPOSITORY

CHANGE IN TAGBOS DATABASE

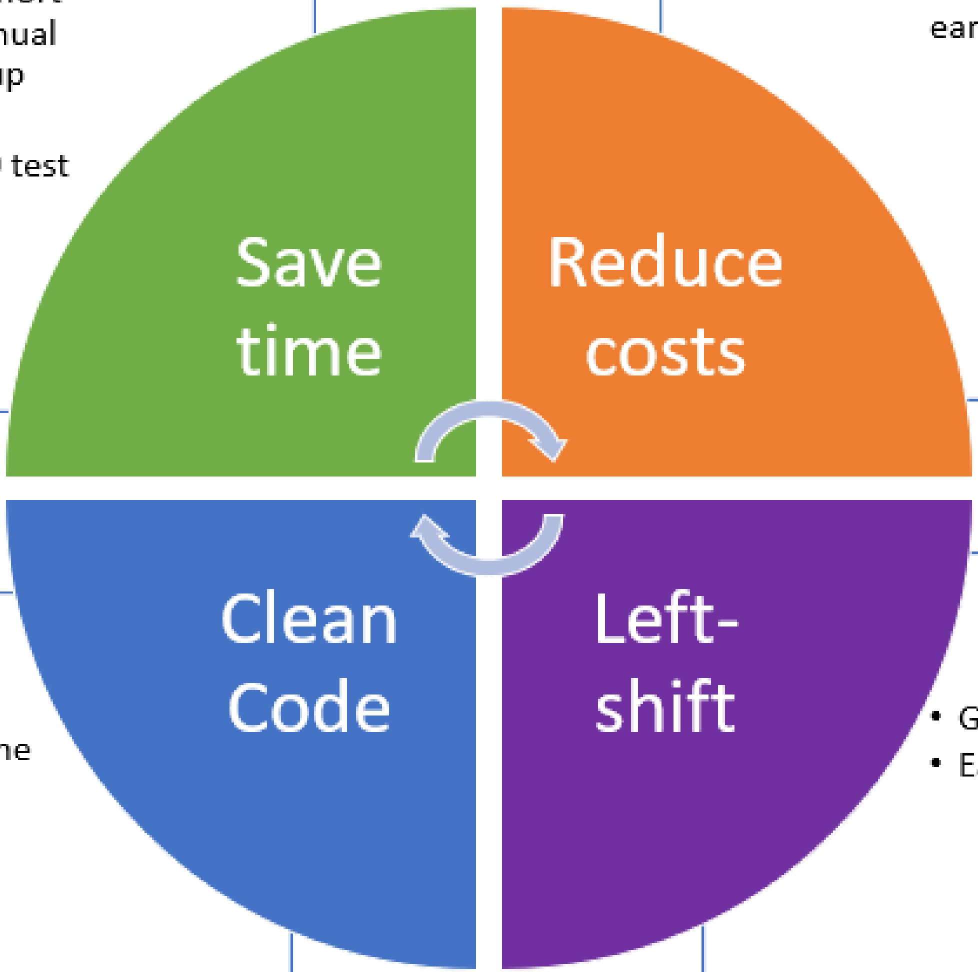
CHANGE IN INFORMATION TRANSFORMER

CHANGE IN MODEL

Benefits

- Optimize testing effort by eliminating manual test case writing: up to 33%/feature
- For more then 100 test engineers

- SW bugs can be found in early stages

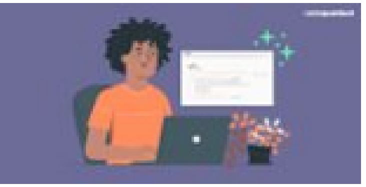


Save time

Reduce costs

Clean Code

Left-shift

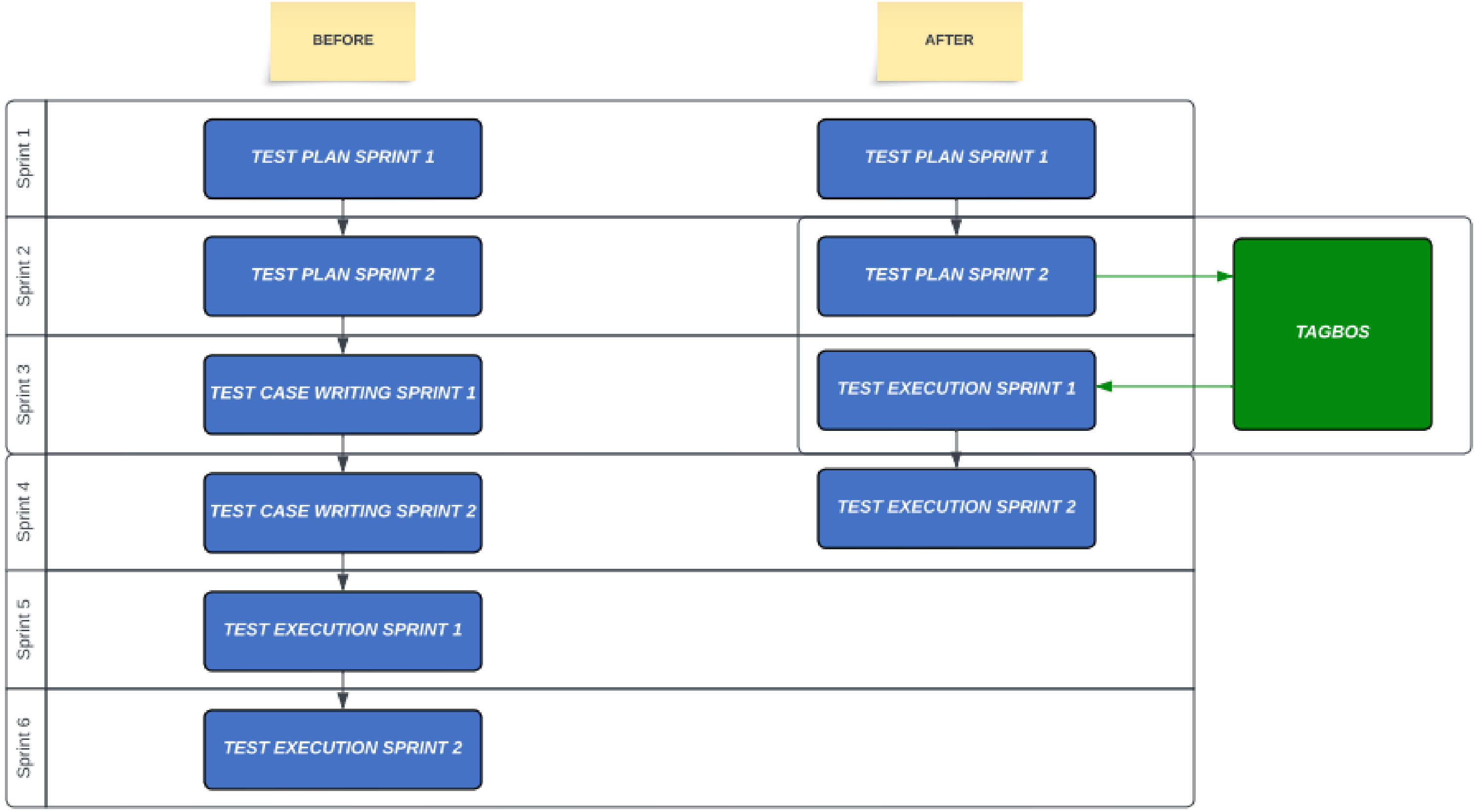


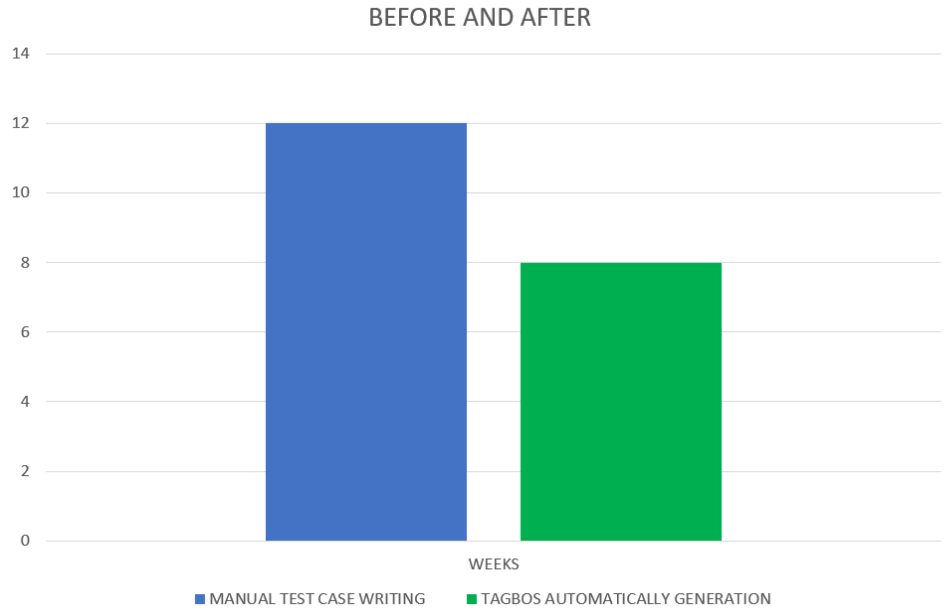
- Reviews can be done 50% more faster

- Gain up to 2 sprints
- Earlier test execution

Benefits

FEATURE TEST CYCLE IN PIT





Next Steps

- Offline model-based testing to Online model-based testing
- Add another interface than REST API
- Generate new altsteps for partial use or test case use
- Add porting functionality –> port from one configuration to another
- Download & Setup automatically the test repository
- Future AI/ML capability to learn and automatically generate new test cases

Q&A

Thank you!

Cosmin Vîlceanu
cosmin.vilceanu@nokia.com

Tudor Tomuța
tudor.tomuta@nokia.com