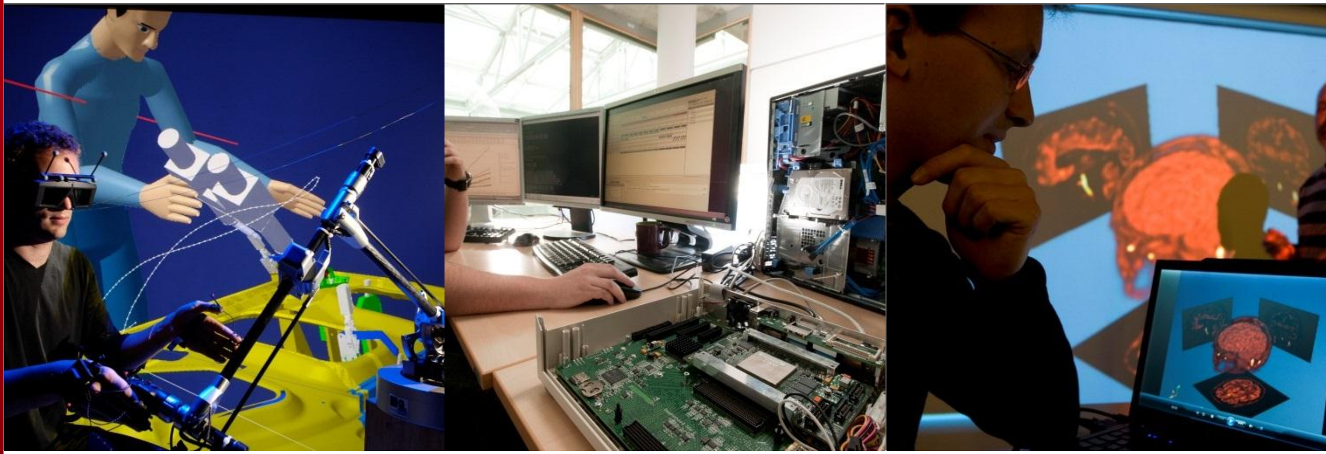
# [PSSM] – TEST SUITE

Jérémie TATIBOUET (CEA LIST)

Arnaud CUCCURU (CEA LIST)



# OUTLINE

A. Objective

B. New architecture
   - What is provided by UTP?
   - Architecture base classes

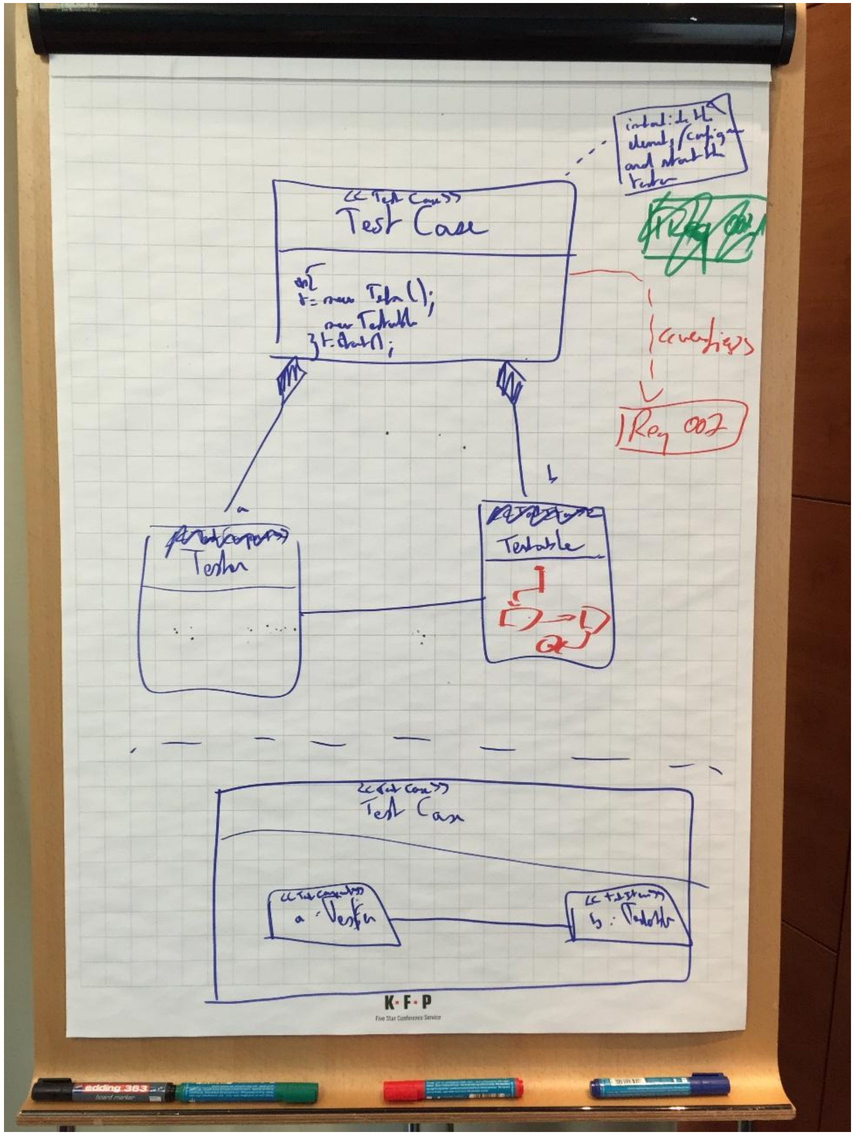
C. Using the new architecture over an example

D. Demonstration

E. Some statistics about the current coverage

# PSSM TEST SUITE

## INITIAL PICTURE



### Berlin discussions

- Structure the test suite
  - Make test suite construction faster
  - Make the architecture easy to reuse
- Establish relationship with UTP
  - Make sure the test suite architecture make sense for test experts

# NEW ARCHITECTURE OF THE TEST SUITE

# WHAT IS PROVIDED BY UTP?

## CURRENTLY BASED ON UTP 1.2

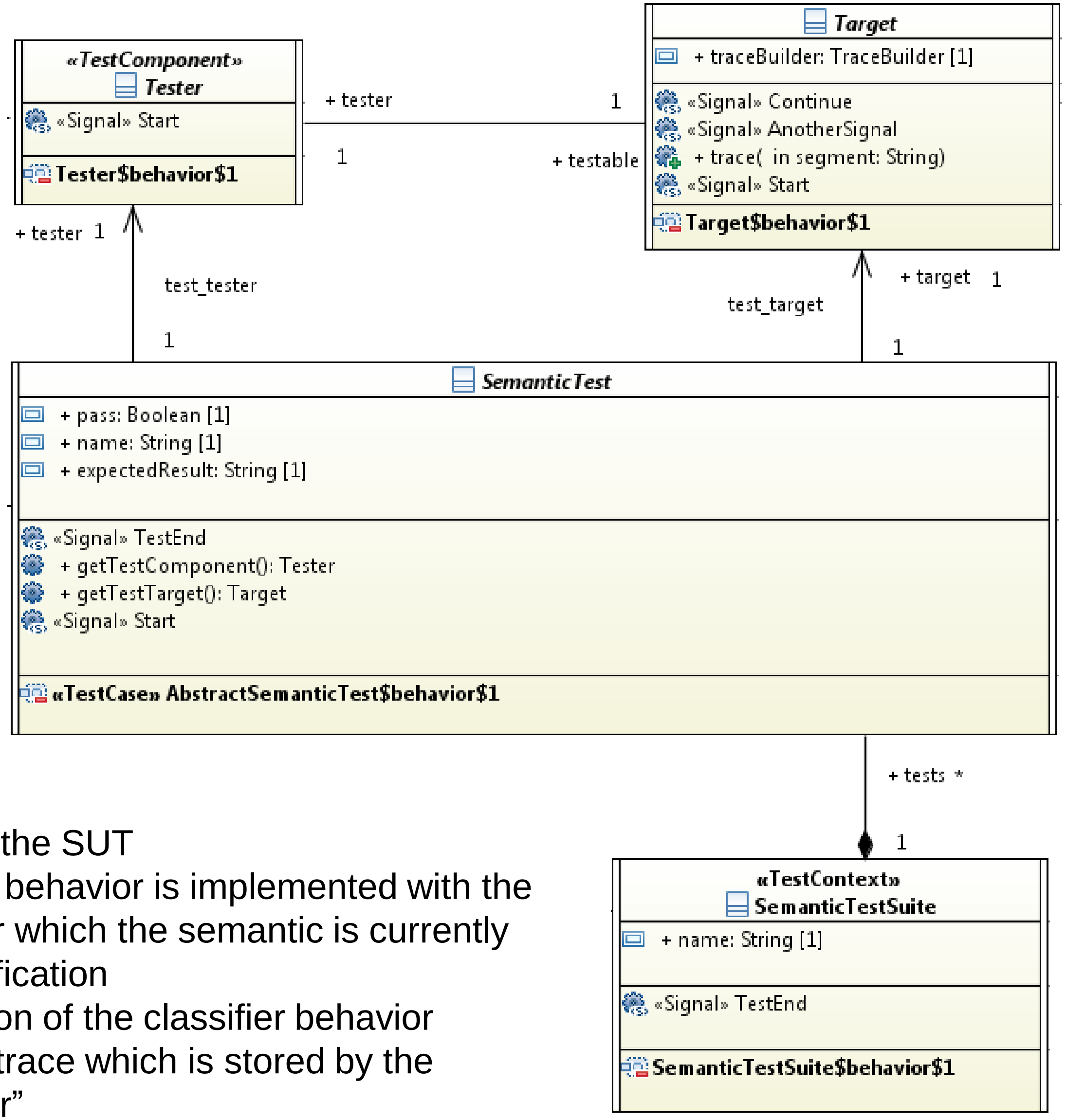
A. TestComponent (p. 33)
   - Communicate with the system under test (SUT).
   - Drive test case by stimulating the SUT.

B. TestCase (p. 42)
   - A test case specifies how test components interact
   - Test cases are owned by test contexts
   - A test case always return a verdict

C. TestContext (p. 34)
   - A test context acts a grouping mechanism for a set of test cases
   - Control logic of test execution can be specified through the classifier behavior of the test context.

# ARCHITECTURE BASE CLASSES

## DERIVED FROM UTP - TARGET



«TestComponent»
Tester
«Signal» Start
Tester$behavior$1

Target
+ traceBuilder: TraceBuilder [1]
«Signal» Continue
«Signal» AnotherSignal
+ trace( in segment: String)
«Signal» Start
Target$behavior$1

+ tester 1
1 + testable

+ tester 1
test_tester
1

test_target
+ target 1
1

SemanticTest
+ pass: Boolean [1]
+ name: String [1]
+ expectedResult: String [1]
«Signal» TestEnd
+ getTestComponent(): Tester
+ getTestTarget(): Target
«Signal» Start
«TestCase» AbstractSemanticTest$behavior$1

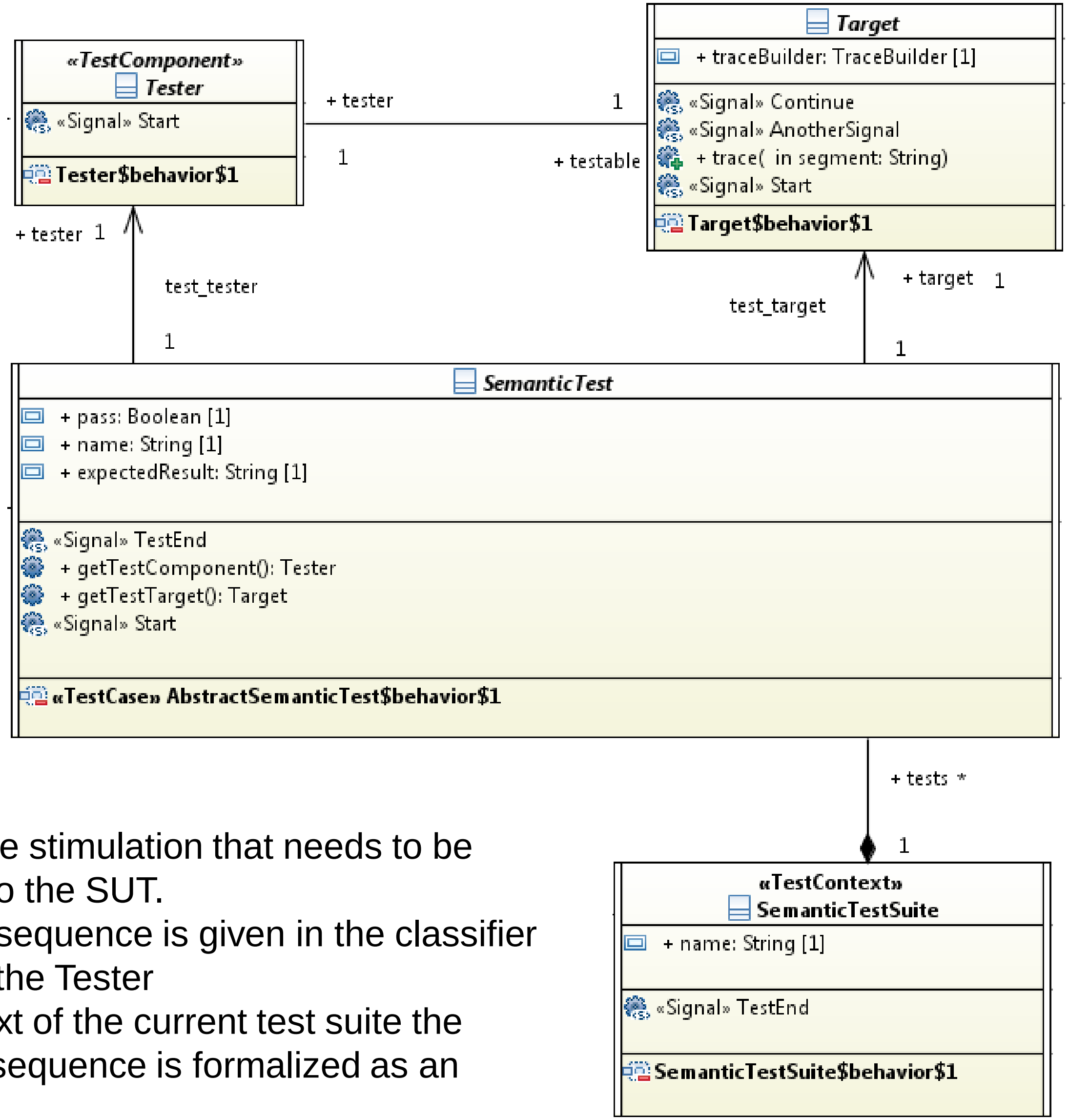
+ tests *
1

«TestContext»
SemanticTestSuite
+ name: String [1]
«Signal» TestEnd
SemanticTestSuite$behavior$1

- Target
  - Represents the SUT
  - Its classifier behavior is implemented with the elements for which the semantic is currently under specification
  - The execution of the classifier behavior produces a trace which is stored by the "traceBuilder"

# ARCHITECTURE BASE CLASSES

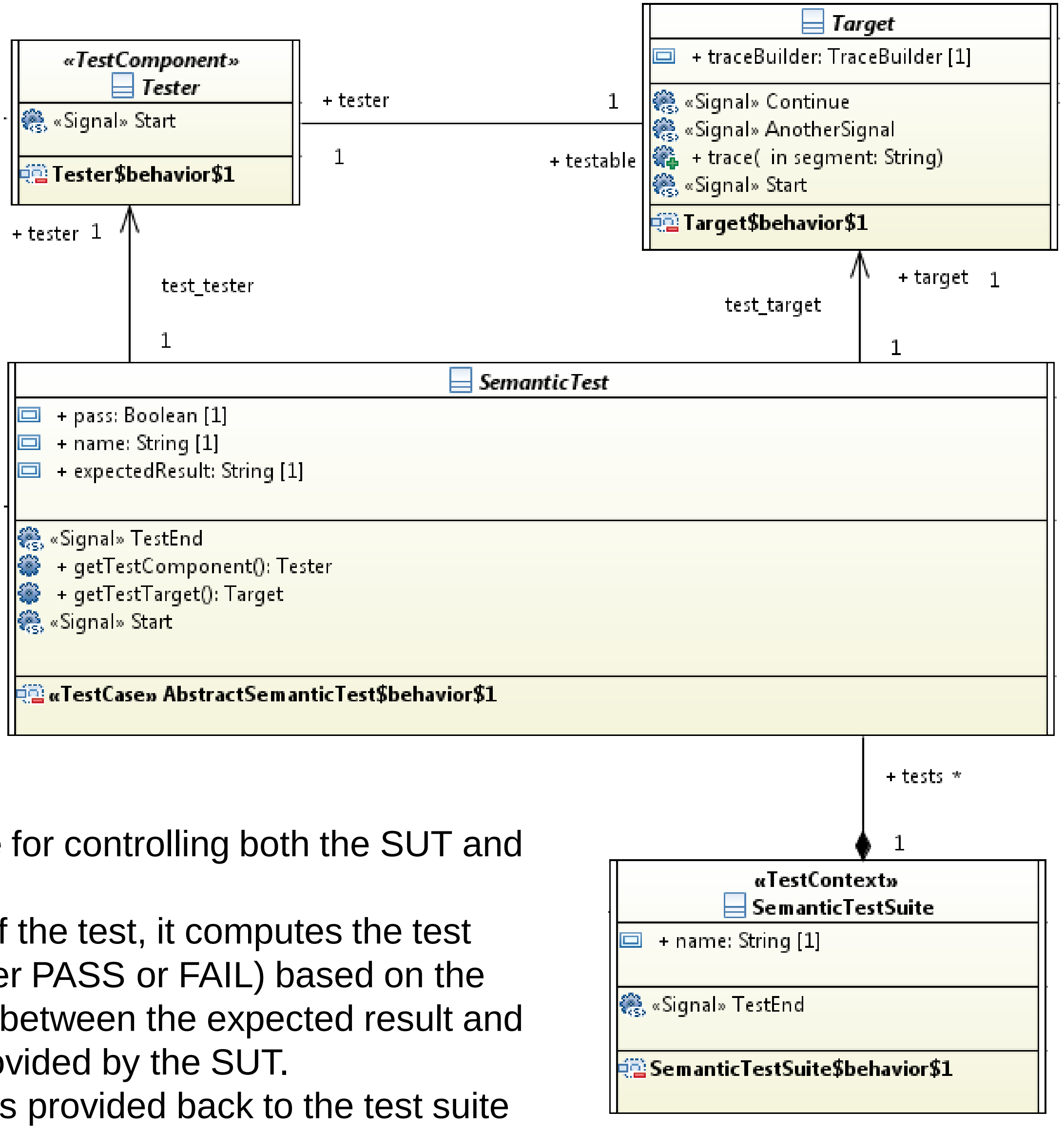
## DERIVED FROM UTP - TESTER



- Tester
  - Produces the stimulation that needs to be addressed to the SUT.
  - Stimulation sequence is given in the classifier behavior of the Tester
  - In the context of the current test suite the stimulation sequence is formalized as an activity
  - It is linked at runtime with the SUT (i.e. Target)

# ARCHITECTURE BASE CLASSES

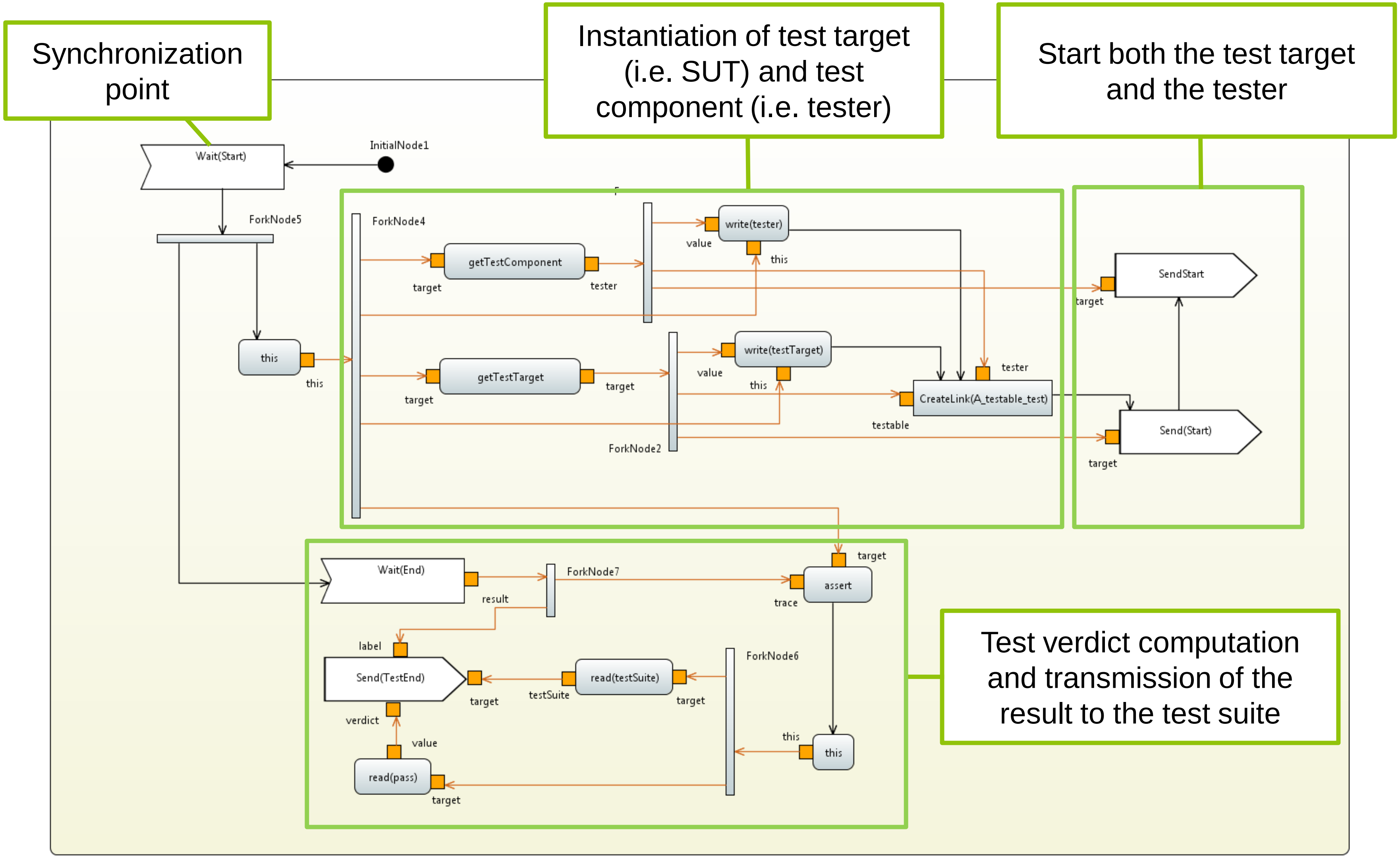
## DERIVED FROM UTP – SEMANTIC TEST



- SemanticTest
  - Responsible for controlling both the SUT and the tester.
  - At the end of the test, it computes the test verdict (either PASS or FAIL) based on the comparison between the expected result and the trace provided by the SUT.
  - The verdict is provided back to the test suite owning the semantic test.

# ARCHITECTURE BASE CLASSES

## SEMANTIC TEST – CLASSIFIER BEHAVIOR



Synchronization point

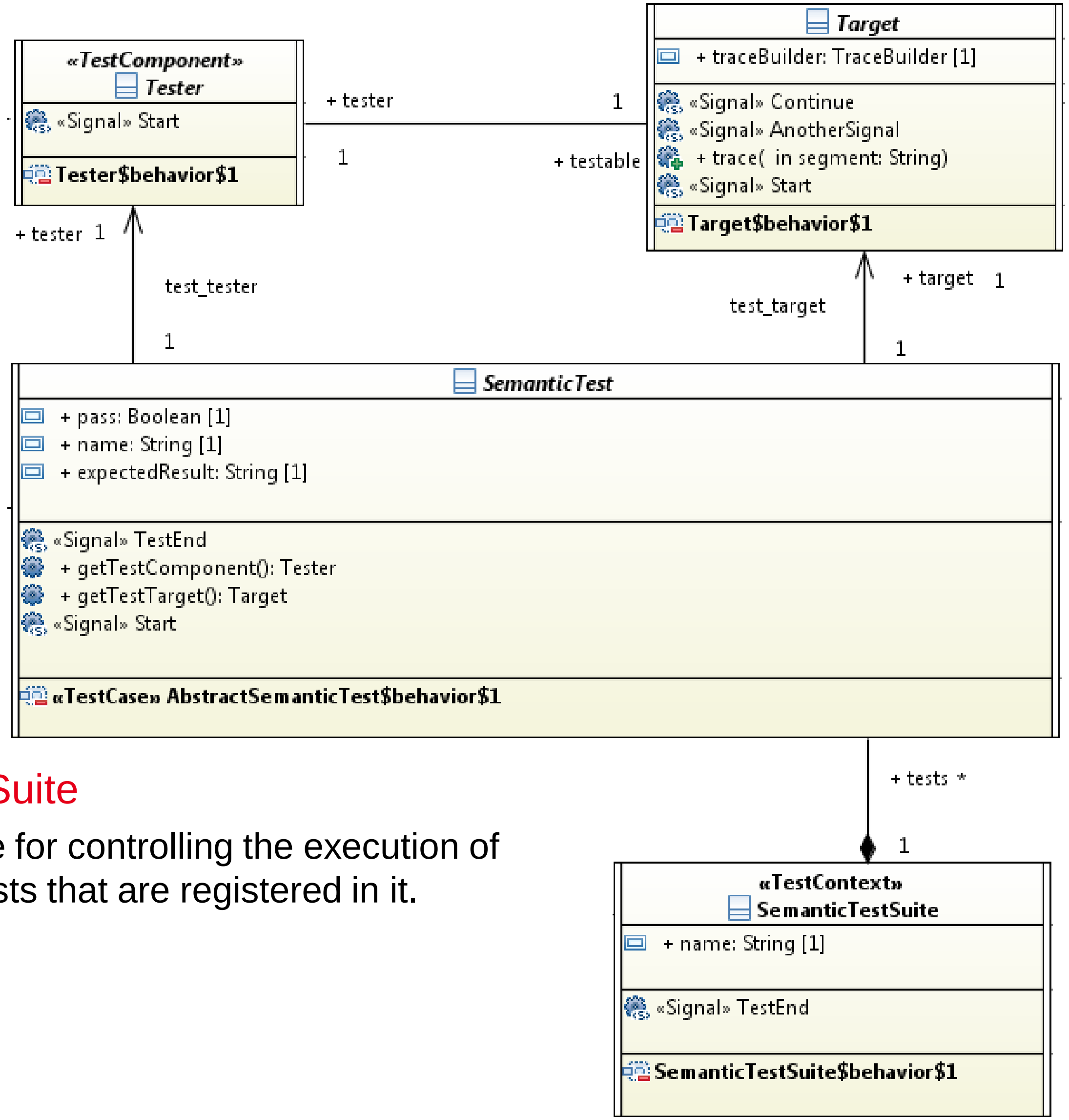
Instantiation of test target (i.e. SUT) and test component (i.e. tester)

Start both the test target and the tester

Test verdict computation and transmission of the result to the test suite

# ARCHITECTURE BASE CLASSES

## DERIVED FROM UTP – SEMANTIC TEST SUITE

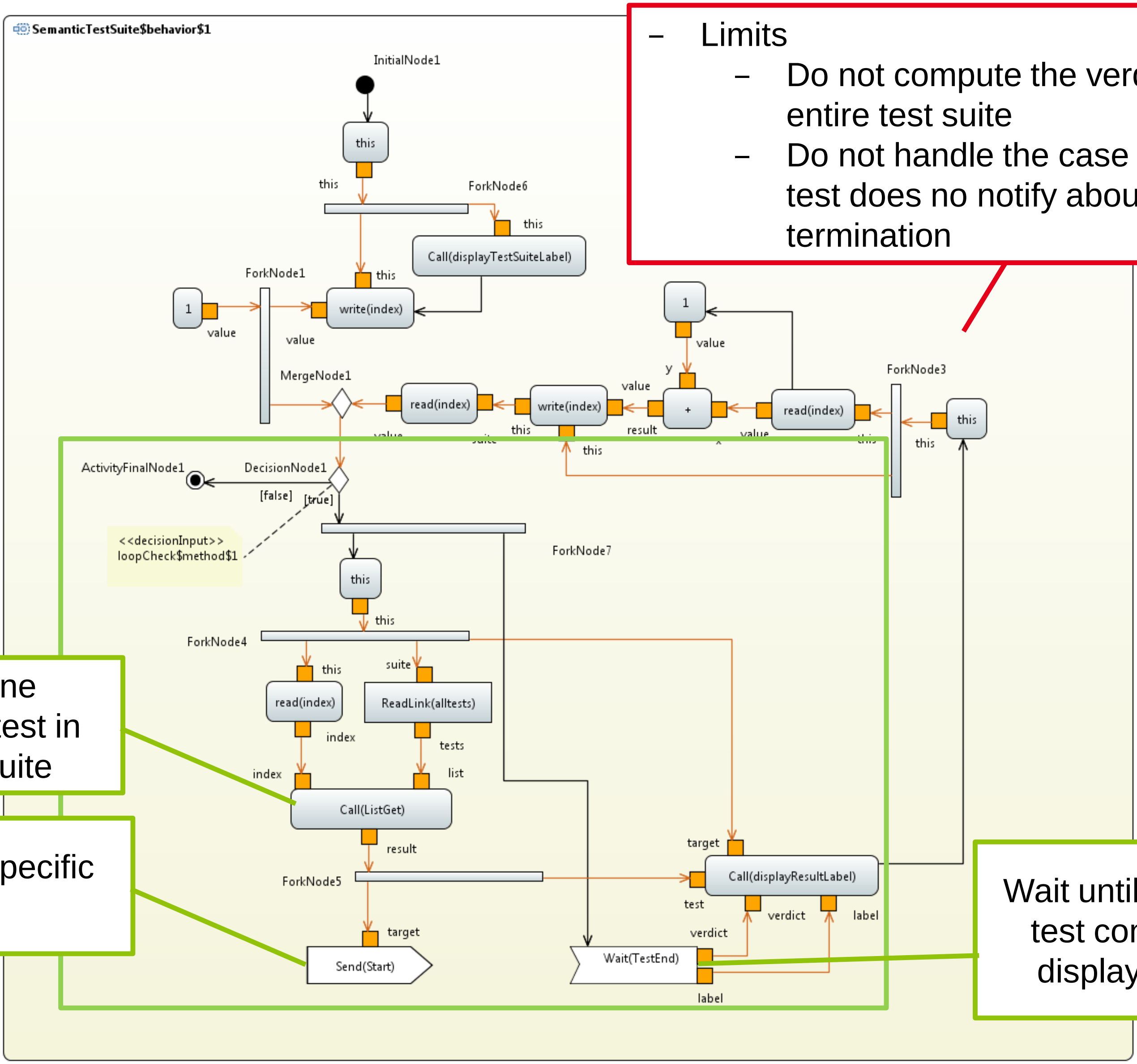


- SemanticTestSuite
  - Responsible for controlling the execution of semantic tests that are registered in it.

# ARCHITECTURE BASE CLASSES

## SEMANTIC TEST SUITE – CLASSIFIER BEHAVIOR

- Limits
  - Do not compute the verdict for the entire test suite
  - Do not handle the case in which a test does no notify about its termination



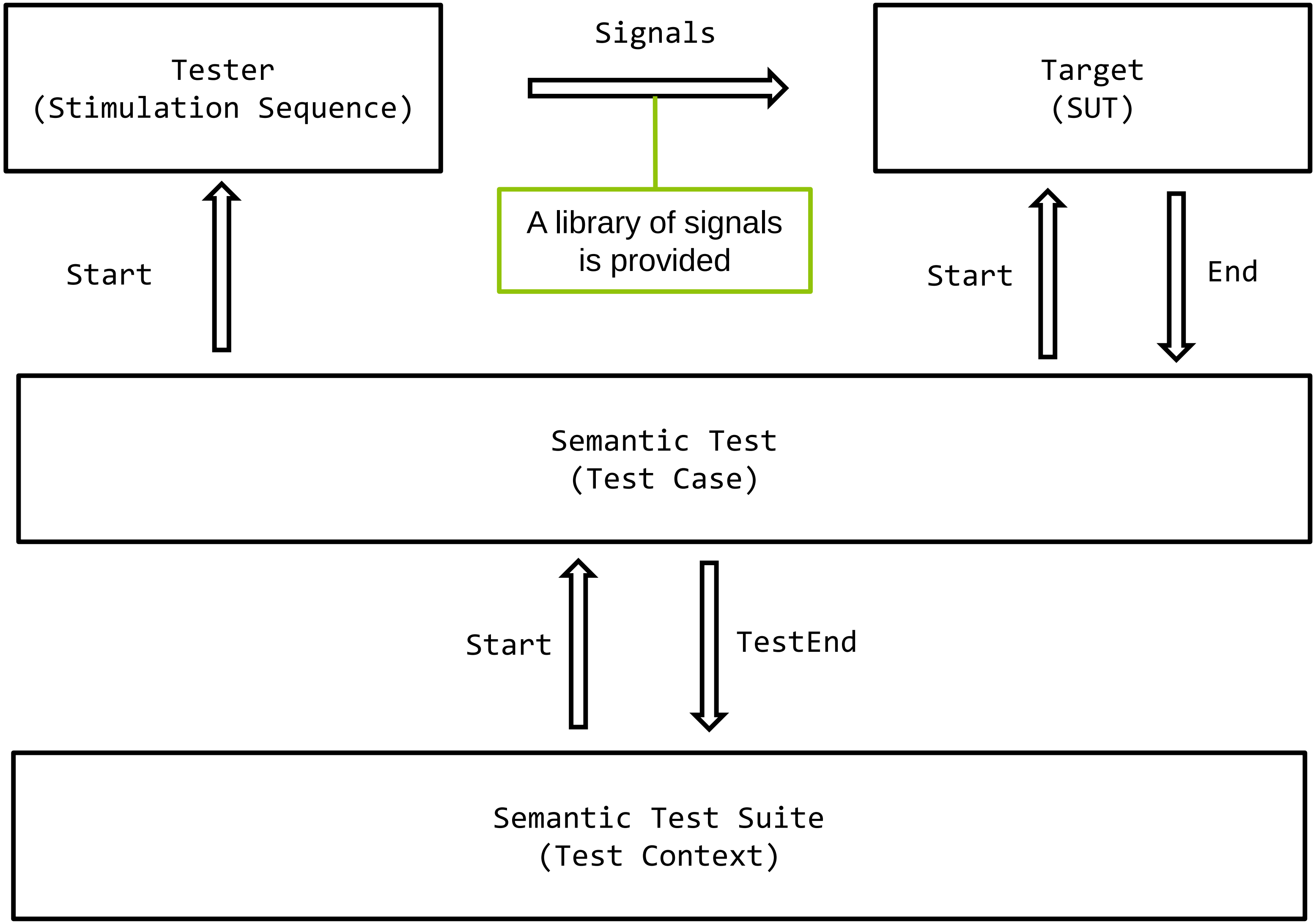
Select one registered test in the test suite

Start this specific test

Wait until the semantic test completes and display the results

# ARCHITECTURE BASE CLASSES

## GLOBAL TESTING DYNAMYCS



A library of signals is provided

# USING THE ARCHITECTURE OVER AN EXAMPLE

## What needs to be extended ?

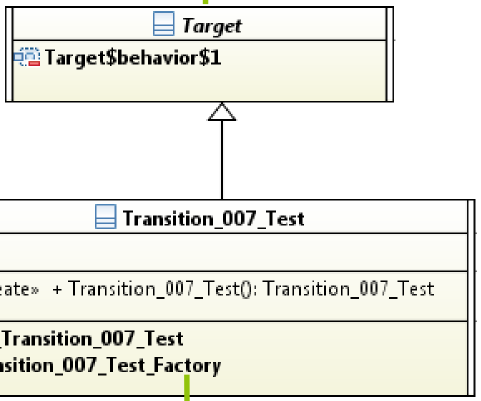
- « Tester » class
  - Needs to provide a specific classifier behavior
  - Note: only extended if stimulus are required – If it is not the case then a dummy tester can be used
- « Target » class
  - Always need to be extended
  - Provides the behavior to be tested
- « SemanticTest » class
  - Operation “getTestTarget” needs to be overridden
  - Operation “getTestComponent” needs too be overridden

# USING THE ARCHITECTURE OVER AN EXAMPLE

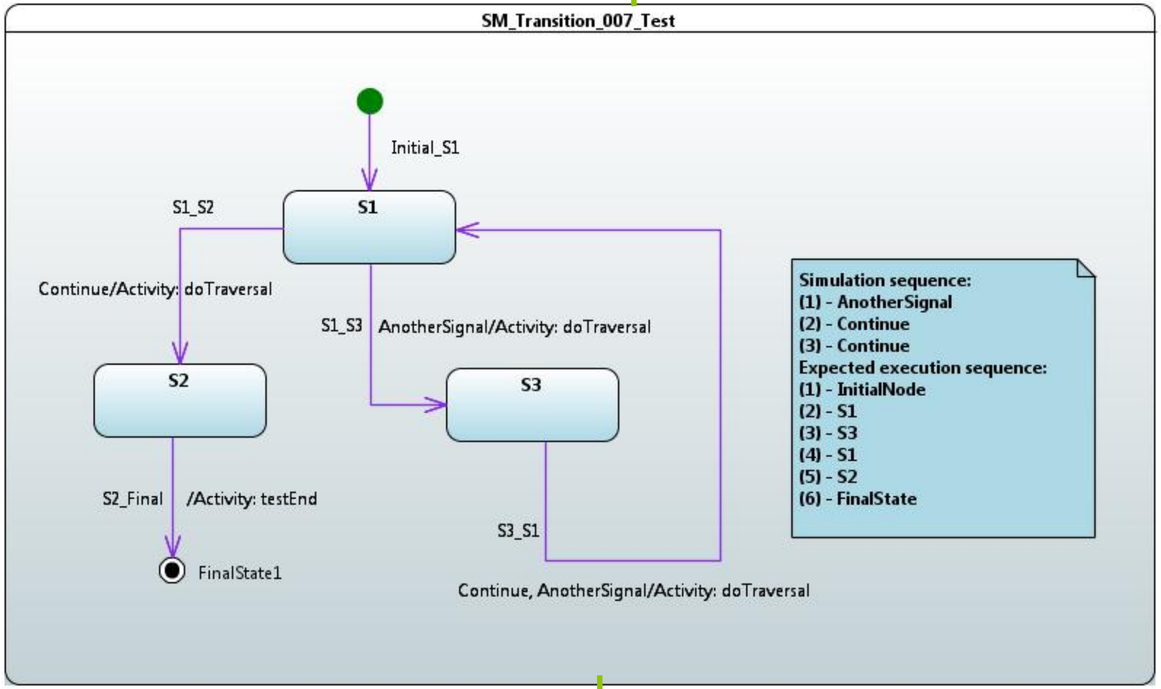
## TEST TARGET SPECIALIZATION

Note: here we do not to have a synchronization point (i.e. waiting for Start).

Base test target class



Target
Target$behavior$1
Transition_007_Test
«Create» + Transition_007_Test(): Transition_007_Test
SM_Transition_007_Test
Transition_007_Test_Factory



SM_Transition_007_Test
Initial_S1
S1
S1_S2
Continue/Activity: doTraversal
S1_S3
AnotherSignal/Activity: doTraversal
S2
S3
S2_Final
/Activity: testEnd
S3_S1
FinalState1
Continue, AnotherSignal/Activity: doTraversal

Simulation sequence:
(1) - AnotherSignal
(2) - Continue
(3) - Continue
Expected execution sequence:
(1) - InitialNode
(2) - S1
(3) - S3
(4) - S1
(5) - S2
(6) - FinalState

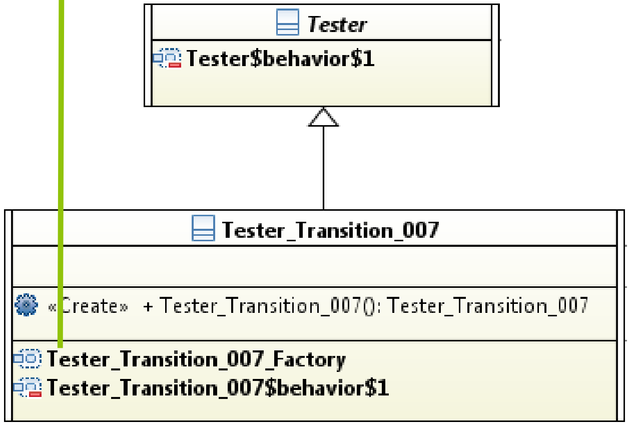
Factory to build and start classifier behaviors of this test target

Classifier behavior of the test target

# USING THE ARCHITECTURE OVER AN EXAMPLE

## TESTER SPECIALIZATION

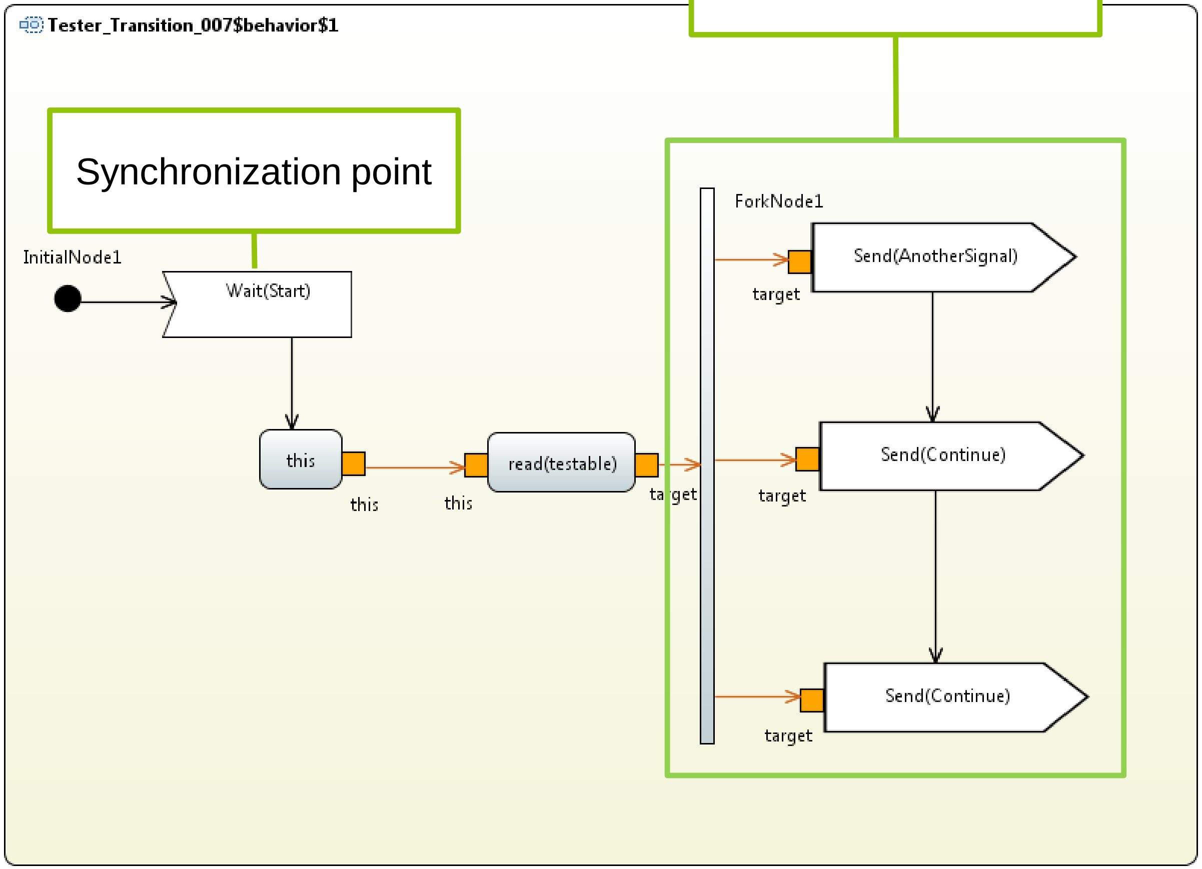
Factory that enables the instantiation of the tester as well as the starting of its classifiers behaviors



Tester
Tester$behavior$1
Tester_Transition_007
«Create» + Tester_Transition_007(): Tester_Transition_007
Tester_Transition_007_Factory
Tester_Transition_007$behavior$1

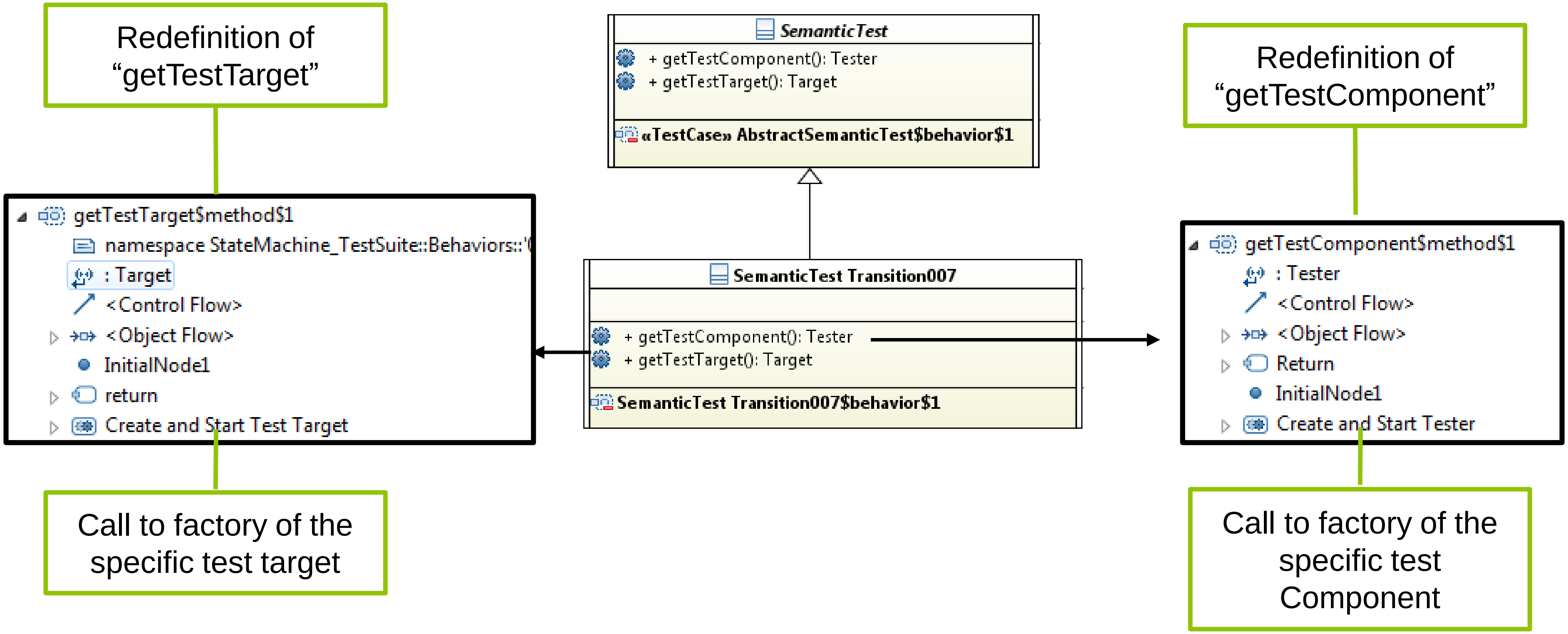
Stimulation sequence

Synchronization point



# USING THE ARCHITECTURE OVER AN EXAMPLE

## SEMANTIC TEST SPECIALIZATION



Redefinition of “getTestTarget”

getTestTarget$method$1
- namespace StateMachine_TestSuite::Behaviors::'
- : Target
- <Control Flow>
- <Object Flow>
- InitialNode1
- return
- Create and Start Test Target

Call to factory of the specific test target

*SemanticTest*
- + getTestComponent(): Tester
- + getTestTarget(): Target
- «TestCase» AbstractSemanticTest$behavior$1

SemanticTest Transition007
- + getTestComponent(): Tester
- + getTestTarget(): Target
- SemanticTest Transition007$behavior$1

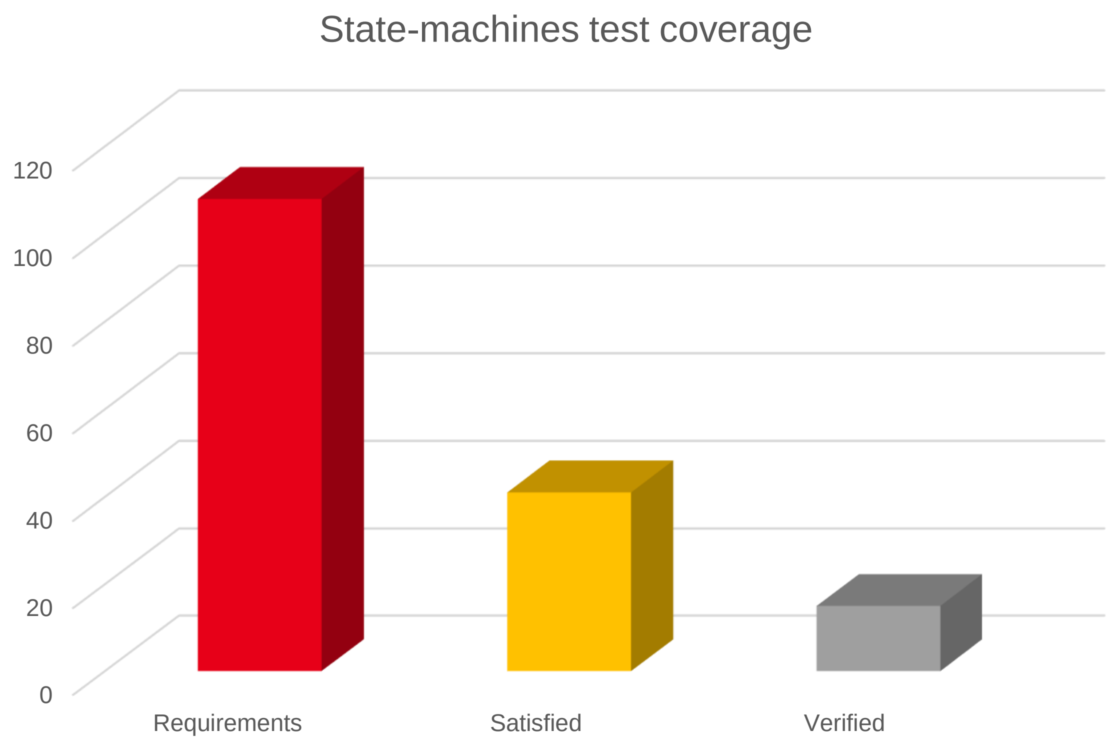
Redefinition of “getTestComponent”

getTestComponent$method$1
- : Tester
- <Control Flow>
- <Object Flow>
- Return
- InitialNode1
- Create and Start Tester

Call to factory of the specific test Component

# COVERAGE STATISTICS

## WHAT IS ACTUALLY SATISIFIED AND VERIFIED?



What is the current coverage ?

- Requirements: 107
  - Note: 6 target none mandatory requirements
- Satisfied (i.e. covered by the prototype): 41 (~38%)
- Verified (i.e. effectively tested): 15 (~14%)