eclipse CON 2005

Integration of Fokus TTCN-3 Tools into Hyades

Diana Vega, Bernard Stepien, George Din

Project aim

- **Fokus –** main contributor to standardization of TTCN-3 language and its execution environment (ETSI)
- **TTCN-3 Testing Tools – TTworkbench product**
  - TTCN-3 Core Language Editor – Eclipse Plug-in
  - TTCN-3 Compiler – Eclipse Plug-in
  - Execution environment
- **Coupling with Hyades**
  - Integration of the tools in Hyades project
  - Combine the high-level language capabilities of the TTCN-3 test language with the test management capabilities of Hyades

What is TTCN-3 ?

- **high-level language** to describe test suites formally
- **powerful matching mechanism** to compare an oracle to response data
- powerful high-level language to **specify test logic as trees**
- separation of concerns between the **abstract test specification** and the **concrete test implementation** using an adapter and coder/decoder (codec)
- powerful test data specification feature -- **data templates** -- that allows unlimited structuring and reusability of test data and specification of matching rules.
- **message** and **procedure** oriented.
- **Parallel test** components and configurations.
- Modularity. Code can be distributed into modules and imported.
- TTCN-3 comes in three versions: core text, graphical and tabular.

TTCN-3 Test module example

```
module PingPong {
        type port PingPongPortType message {
                out charstring;
                in charstring;
        }

        type component MTCType {
                var charstring theWrongResponse;
                port PingPongPortType ping_pong_port;
        }

        type component SystemType {
                port PingPongPortType system_ping_pong_port;
        }

        testcase PlayPingPong() runs on MTCType system SystemType {
                map (mtc: ping_pong_port, system: system_ping_pong_port);

                ping_pong_port.send ("ping");
                alt {
                        [] ping_pong_port.receive ("pong") {
                                log ("received a pong");
                                setverdict (pass)
                        }
                        [] ping_pong_port.receive {
                                log ("did not receive a pong");
                                setverdict (fail)
                        }
                };
                stop
        }

        control {
                execute (PlayPingPong());
        }
```

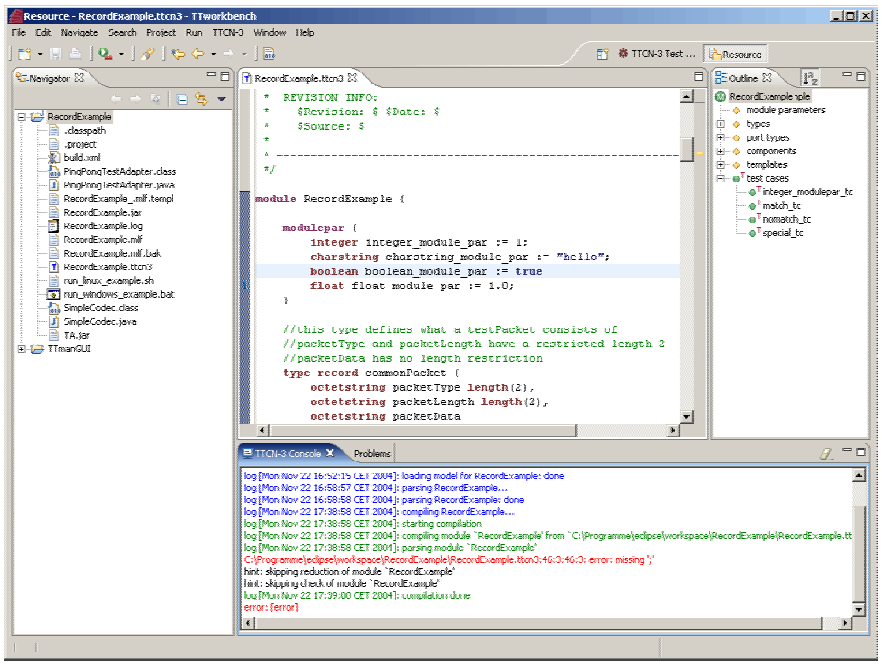

 |

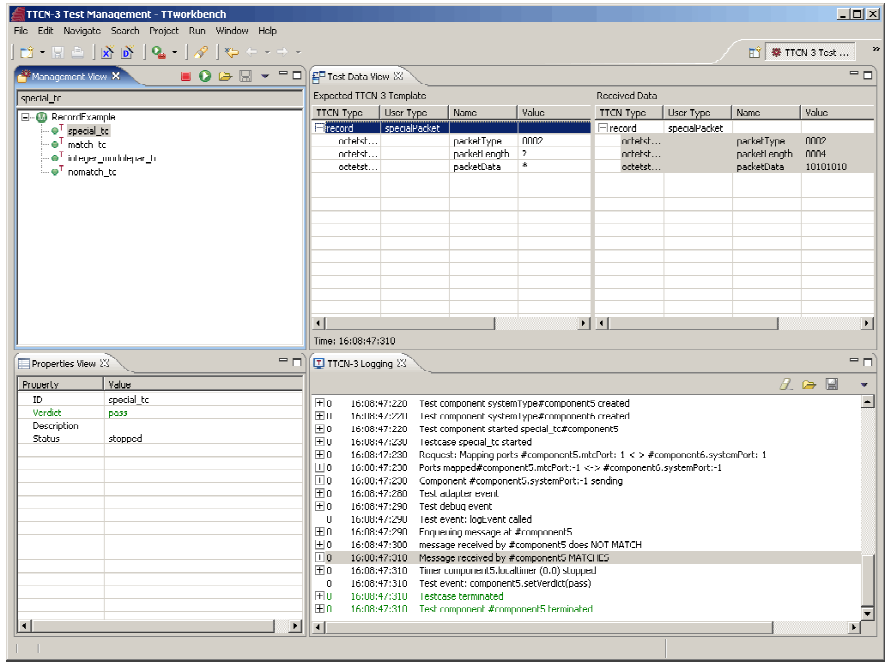
What does the integration mean

- **Hyades - open source project**
  - But it allows vendor testing tools to be integrated in its structure
- **Benefit / Use Hyades tiers**
  - User Interface
    - provides testing, monitoring, profiling, logging tools
  - Data Models
    - models for storing the persisting data
  - Execution Framework
    - an extensible mechanism for executing and controlling tests

 |

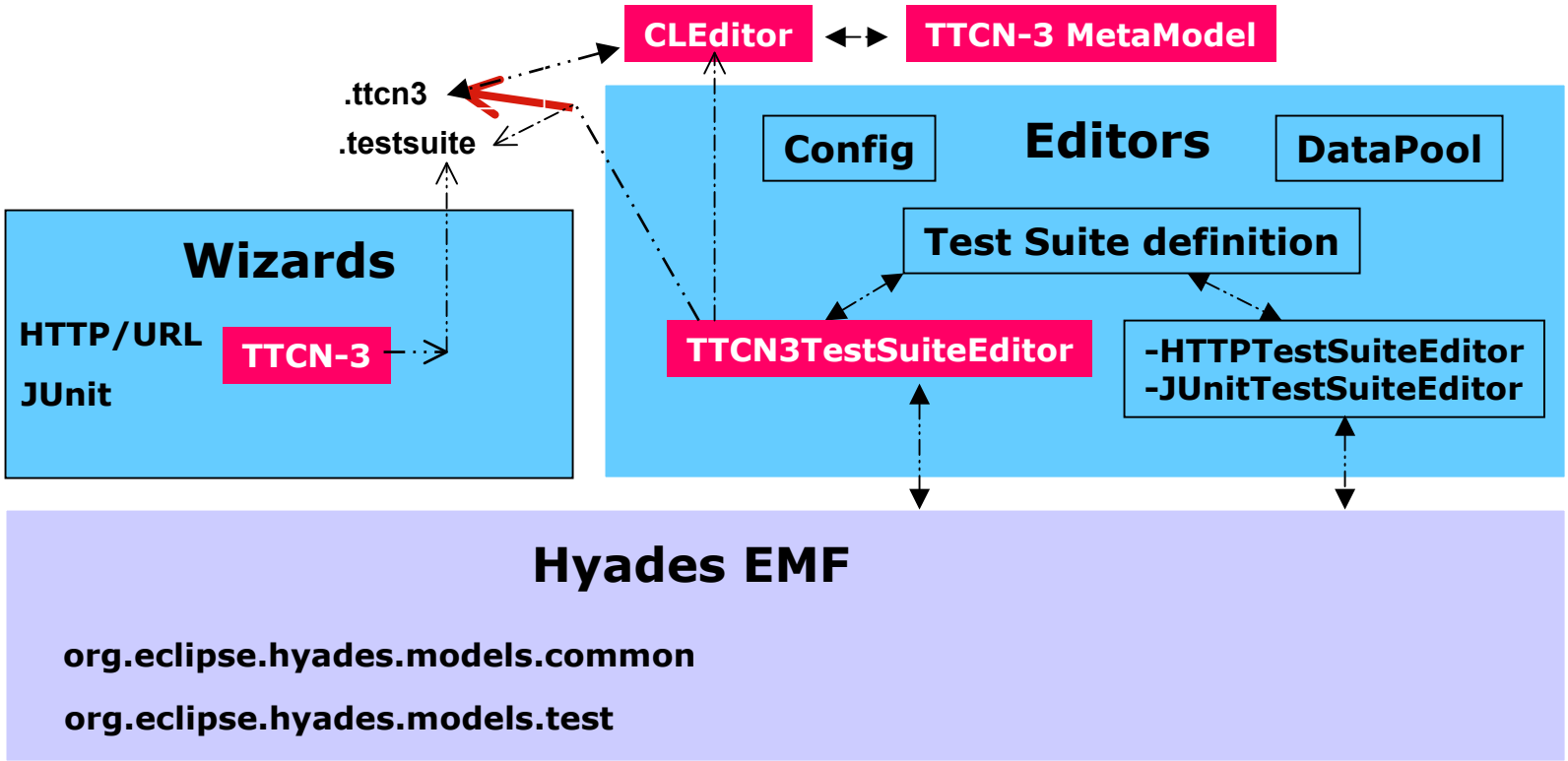
Separate TTCN-3 Eclipse based products





 |

Hyades integration approach



→ use Hyades ***generic wizards*** and ***generic test suite editor***

→ create two kinds of resources

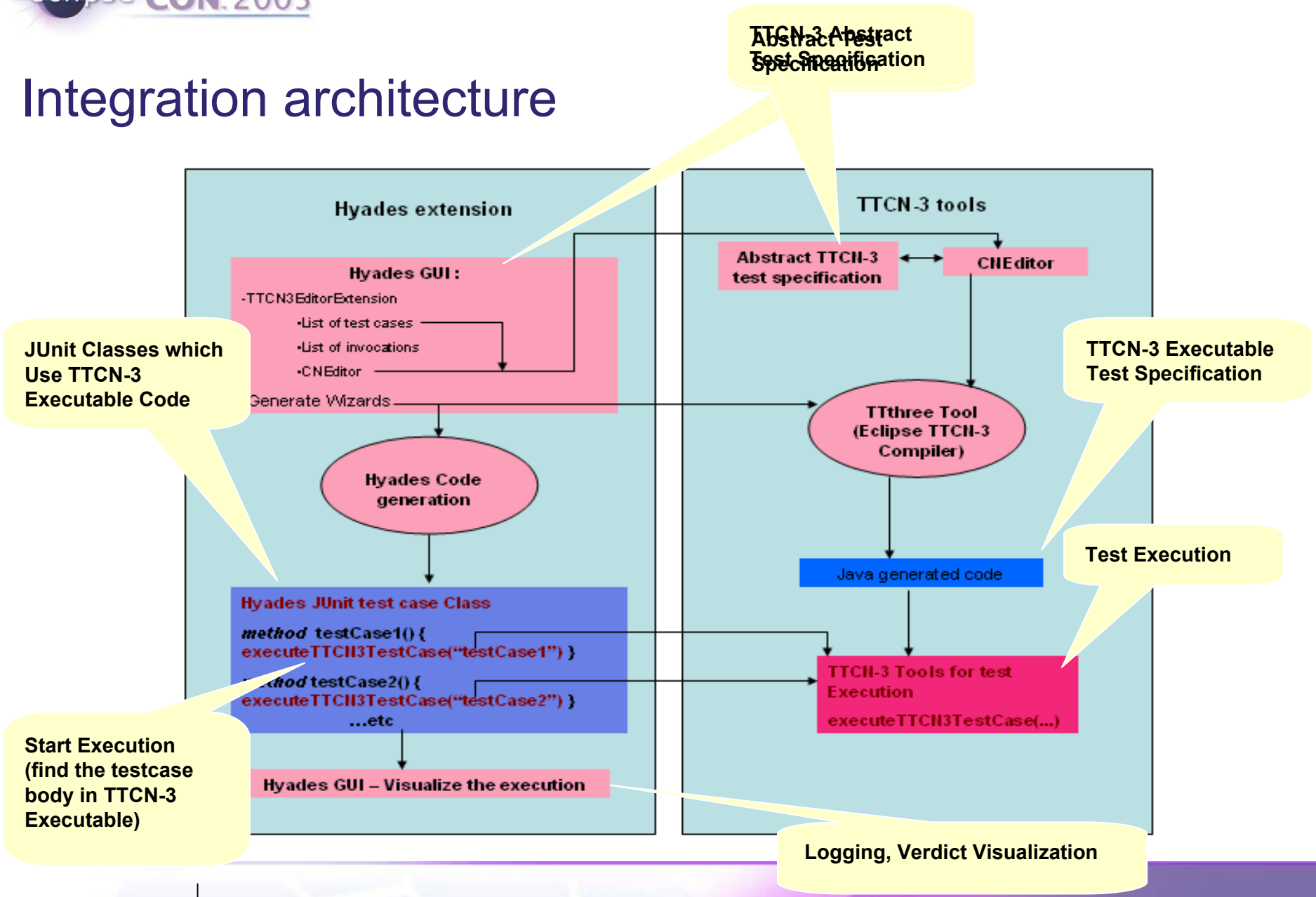
- one based on TTCN-3 metamodel
- one based on Hyades metamodel

 |

Steps for the integration/Implementation

- **Wizard support**
  - New ->Test Suite->TTCN-3 Test Suite
  - Preference pages
  - Pop-up action/options
- **Integration of the Editor**
  - **CLEditor –**abstract TTCN-3 test specification editor for Eclipse
  - **TTCN3TestSuite Editor**
    - based on Hyades TestSuiteEditor
    - includes CLEditor
- **Integration of the Compiler**
  - GUI support via *Generation Wizards*
  - a new wizard page is available with compilation options
  - invoke TTCN-3 Compiler plug-in activity when wizard finishes
- **Execution**
  - start the Hyades execution engine
  - call methods from TTCN-3 tools for test execution (the library TTthreeRuntime)
  - visualize the execution: test verdicts via Hyades generic *execution editor*

Integration architecture



 |

Hyades Extension points usage

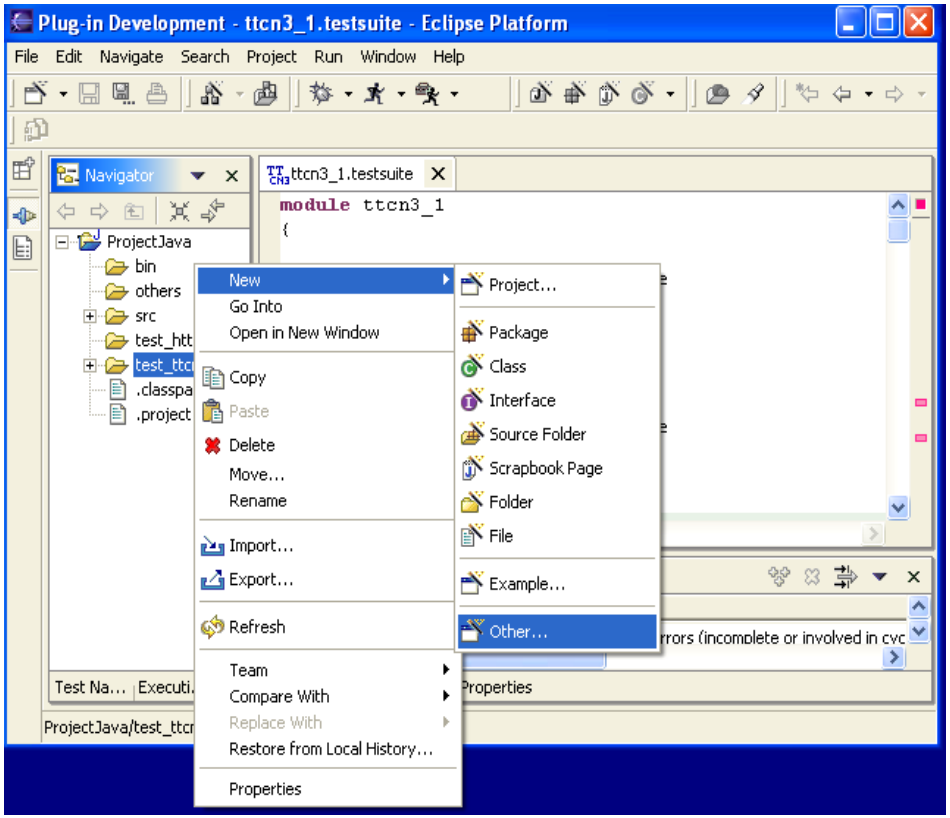
Java code Generation – extension point definition

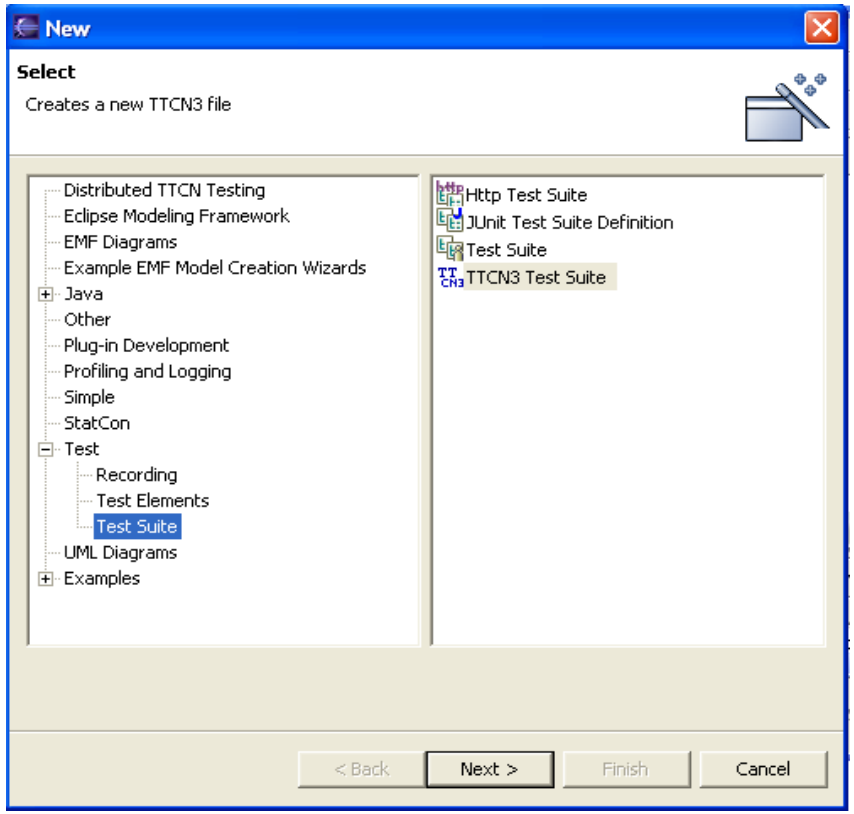
```
<extension
      point="org.eclipse.hyades.test.ui.generateWizards">
    <generateWizard
        name="%GEN_WTITLE"
        type="org.eclipse.hyades.test.ttcn3.junit.testSuite"
        icon="icons/full/ctool16/generate_wiz.gif"
        description="%GEN_WDESC"
        class="org.eclipse.hyades.test.ttcn3.internal.junit.wizard.GenerateWizard"
        extension="testsuite">
    </generateWizard>
  </extension>
```

TTCN-3 TestSuite New Wizard – extension point definition

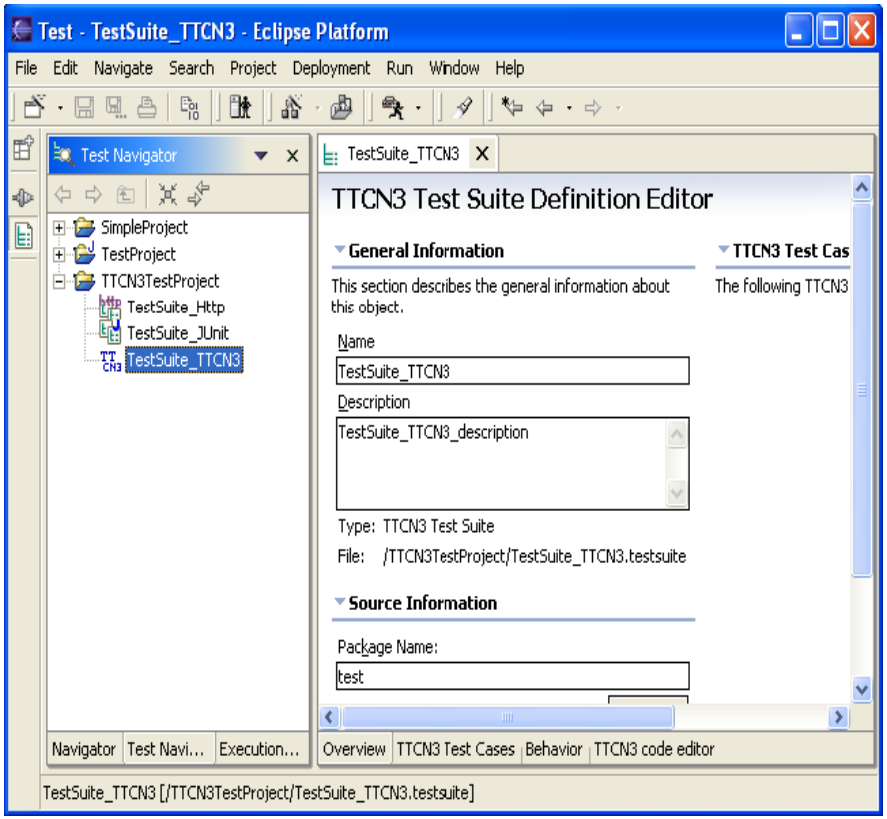
```
<extension point="org.eclipse.ui.newWizards">
    <wizard name="%WIZ_TST_TTCN3_SUITE_TTL"
      icon="icons/full/obj16/ttcn3.gif"
      category="org.eclipse.hyades.test.ui.wizards.new/testSuite"
      class="org.eclipse.hyades.test.ttcn3.internal.junit.wizard.TTCN3TestSuiteNewWizard"
      id="org.eclipse.hyades.test.ttcn3.internal.junit.wizard.TTCN3TestSuiteNewWizard">
       <description>
         %WIZ_NEW_TTCN3_TST_SUITE_DSC
       </description>
    </wizard>
  </extension>
```

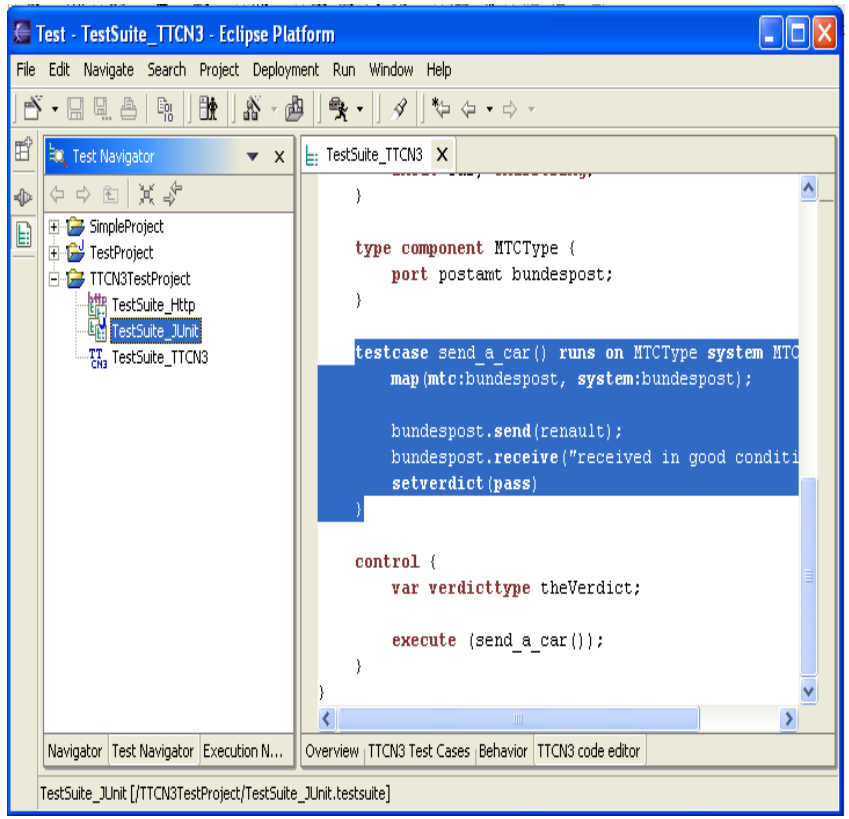
Usage – New Wizards – TTCN-3 Test Suite





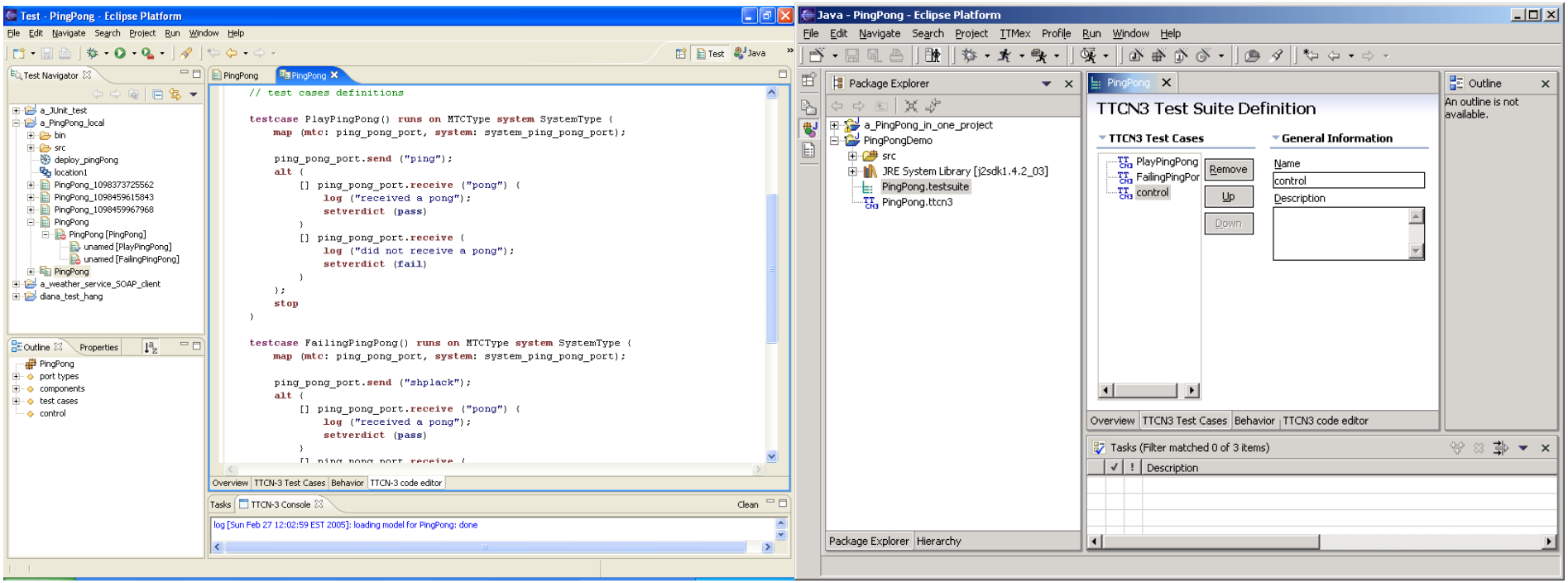
Usage – Edit the resource in the TTCN-3 MultiPage Editor



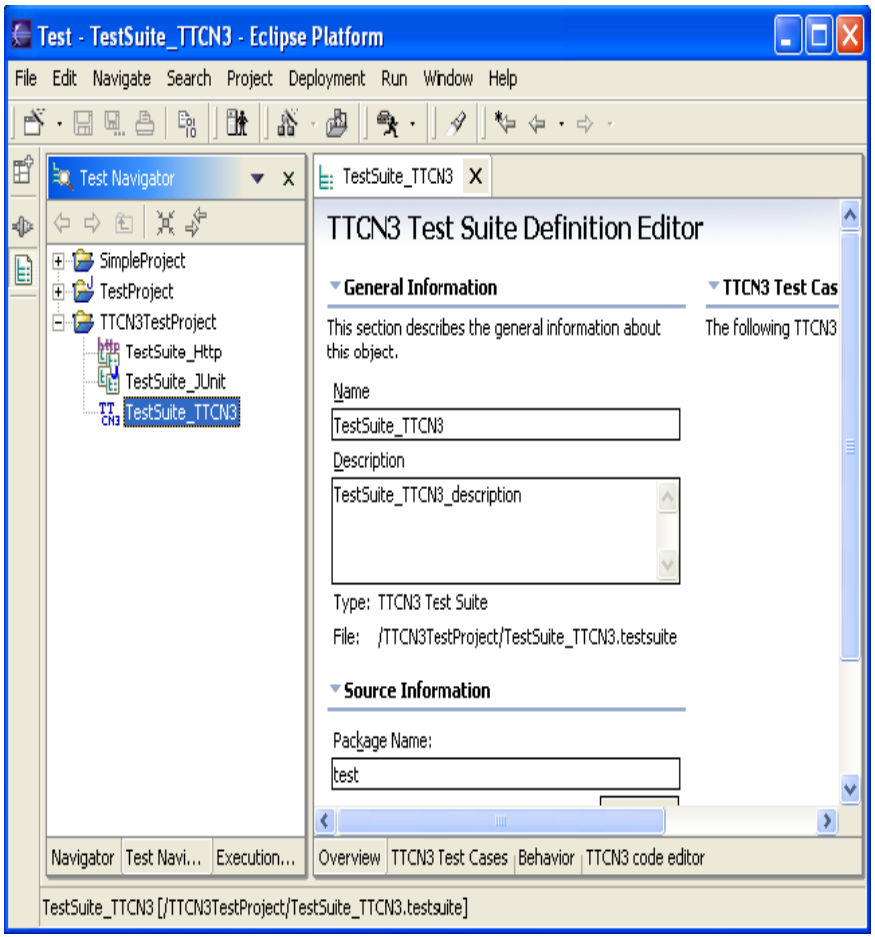


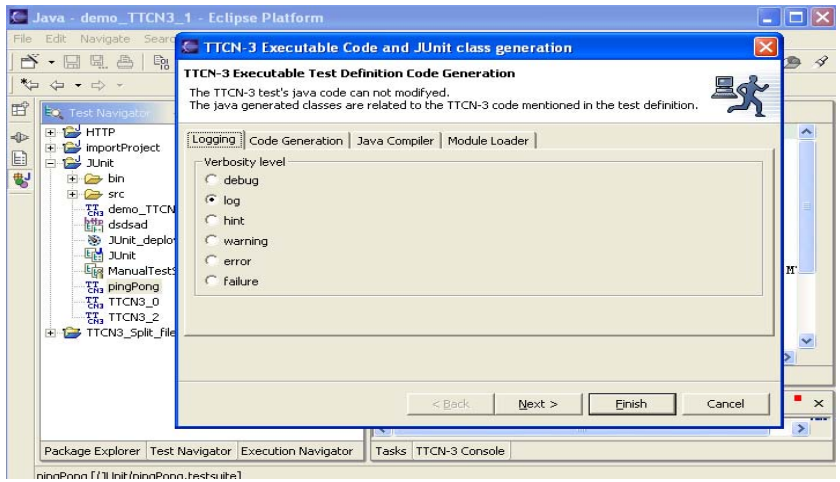
 |

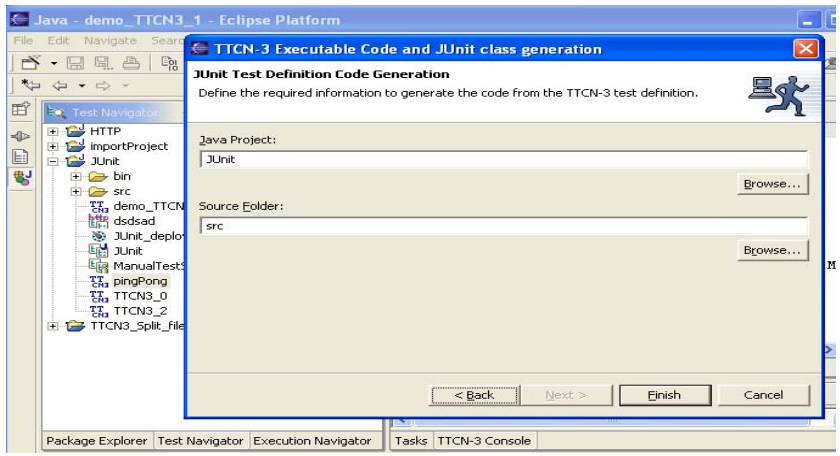
Test suite editor integration



Usage–Generation of the Java code among wizards







 |

Execution integration

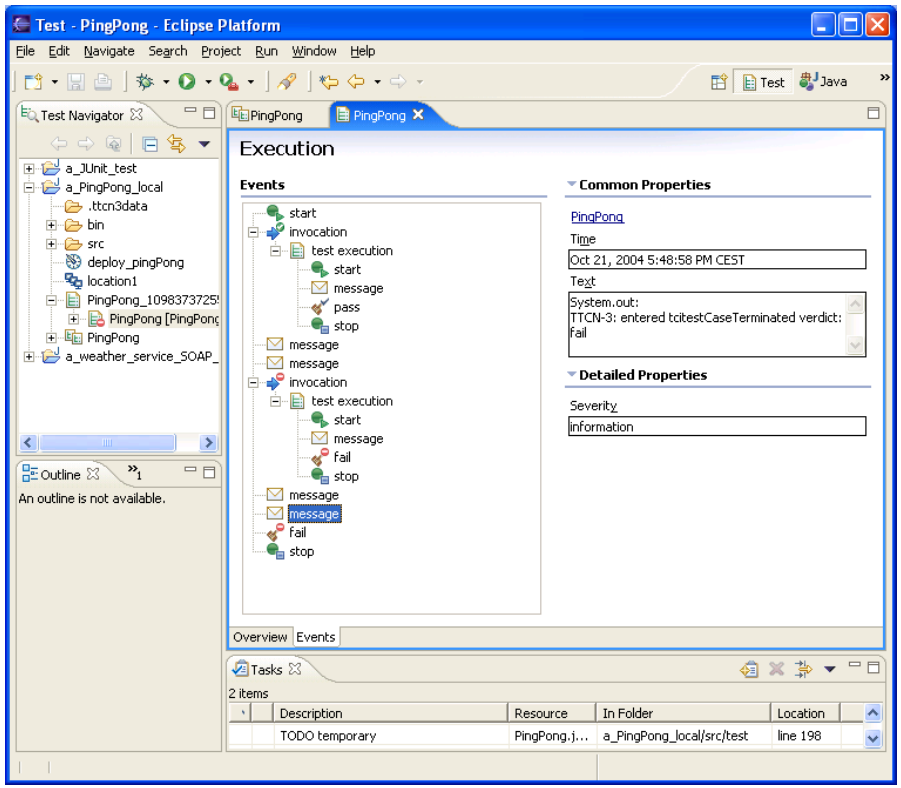
- Generated Hyades execution code consists in invoking a TTCN-3 test case from within a Hyades test case method's code.
- TTCN-3 test verdict is passed to Hyades via the JUnit fail() function for failed test cases and implicitly ignored for passed test cases.

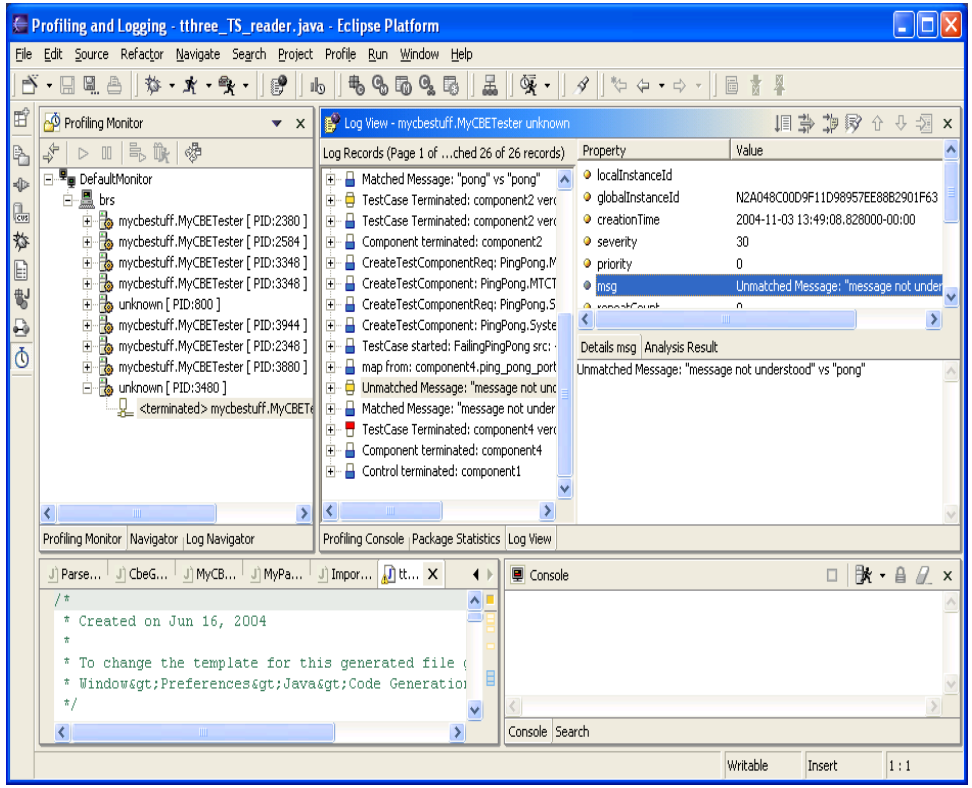
```
public void playpingpong() throws Exception {
// this code has been generated for Hyades/TTCN-3 -  DO NOT MODIFY

                String theTCVerdict = executeTTCN3_TestCase("PlayPingPong");

                if (theTCVerdict.equalsIgnoreCase("fail")) {
                                fail();
                } else if (theTCVerdict.equalsIgnoreCase("error")) {
                                fail(); // this is a work around, errors are caught by TCN-3 tool
                } else if (theTCVerdict.equalsIgnoreCase("none")) {
                                fail();
                } else
                    //do nothing! silent pass in JUnit
                    System.out.println("Test Case PlayPingPong returned from TTCN-3 execution tool with verdict pass");
}
```

TTCN-3/Hyades execution & logging





 |

Hyades/TTCN-3 Integrations issues

- Mapping of the TTCN-3 control part into Hyades behavior concept is very limited. Hyades does not have conditional test case invocations capabilities like TTCN-3.
- TTCN-3 parametric test cases concept can not be mapped to any Hyades concepts.
- TTCN-3 test steps can not be mapped into Hyades concepts.
- Mapping TTCN-3 data templates can not be mapped to Hyades datapool concept mostly because TTCN-3 semantic requirements does not allow data to be external to TTCN-3.
- Logging concept differences: Hyades is reporting only pass or fail while TTCN-3 reports on evaluation of alternatives and could use more diversified color coding.
- No mapping between the TTCN-3 modularity and Hyades (no imports).

 |

Conclusions – Further work – TPTP Integration

- TTCN-3 integration uses the least common denominator principle which implies that many powerful TTCN-3 concepts must be left out.
- TPTP Platform sub-project integration
- Choreography Component
  - Testability Interface

 |