

## Adopting the Hyades Platform

*Milestones Overview*



## Outline

- ➔ Integration Milestones
  - Testing Roadmap
  - Profiling & Logging Roadmap
  - References



## Hyades integration overview

- Adopting Hyades is an incremental process
  - ▶ There are many levels and variants for a tool to use Hyades
  - ▶ This slide deck provides guidelines to integrate existing tools or new tools with Hyades
- Use-case based integration levels:
  - ▶ This deck follows a use case based approach to describe the incremental steps to move along the Hyades adoption curve:
    - Each use case is described with its sequence of steps, along with the integration points for the tool provider and the benefits for the user
    - A few additional considerations are described addressing specific interoperability or compatibility issues
  - ▶ The presentation focuses on testing, and profiling & logging integration
- Note: This presentation assumes a good understanding of the Hyades terminology (see UML 2.0 Testing Profile), and its base architecture



## Testing integration milestones

1. Unmanaged Hyades tests Milestone
  - ▶ Existing tests are imported
  - ▶ Test Suites, Test Cases are not created in the Eclipse shell
  - ▶ Test behaviors, datapools and configurations do not use the Hyades data models
  - ▶ Tests can be executed from the shell and verdicts are logged
2. Managed Hyades tests Milestone
  - ▶ Existing tests might be imported
  - ▶ Test Suites, Test Cases are created and defined in the shell
  - ▶ Test behaviors, datapools and configurations, etc. may use the Hyades data models and associated editors
  - ▶ Tests can be executed locally or distributed, mixed or not with other test types
3. Integrated testing Milestone
  - ▶ Legacy tests can be imported
  - ▶ All or most of the tests assets are defined in the shell
  - ▶ Test behaviors, datapools and configurations use the Hyades data models and extend existing editors (if applicable)
  - ▶ Tests can be executed locally or distributed, mixed or not with other test types, with test logs including all the details about actions and verifications



## Testing integration benefits summary

1. Unmanaged Hyades tests Milestone *(for existing tools)*
  - ▶ Your customers can schedule tests created by your products as part of Hyades Test Suites, run them in batch, get runtime analysis data from their text execution (code coverage, memory/performance data), etc.
2. Managed Hyades tests Milestone
  - ▶ Your customers use only one user interface to get their job done
  - ▶ Your customers benefit from all the tools available in Eclipse and their integration (traceability, configuration management, etc.)
3. Integrated testing Milestone
  - ▶ Your customers can easily migrate from the products of your competitors to yours
  - ▶ Your customers can leverage different types of UIs, scripting languages, etc. to define their test scripts according to their knowledge and expertise



## Profiling & logging integration milestones

1. External data collection Milestone
  - ▶ Runtime data and logs are collected outside of the shell and are imported manually
  - ▶ Hyades views are used to analyze the runtime data or the logs
2. Integrated data collection Milestone
  - ▶ Data collectors and model loaders are used to collect runtime data
  - ▶ Runtime data is stored using all or part of the Hyades data models
  - ▶ Hyades views might be extended or replaced to display collected data
3. Integrated profiling & logging Milestone
  - ▶ Data collectors are integrated with the launch configuration
  - ▶ Runtime data and logs are controlled at runtime
  - ▶ Runtime data is stored using the Hyades data models
  - ▶ Hyades views are preferably extended to display the runtime data



## Profiling & logging integration benefits summary

1. External data collection Milestone
  - ▶ Your customers get access to a wide range of user interfaces to explore the data you are collecting
  - ▶ The data you are collecting can be correlated with additional information
2. Integrated data collection Milestone
  - ▶ Your customers can collect runtime data and logs from multiple types at the same time
  - ▶ The collected data can be correlated with additional Hyades-collected data or other profiling & logging tools
3. Integrated profiling & logging Milestone
  - ▶ Your customers use a common, integrated user interface to configure, collect and analyze their applications

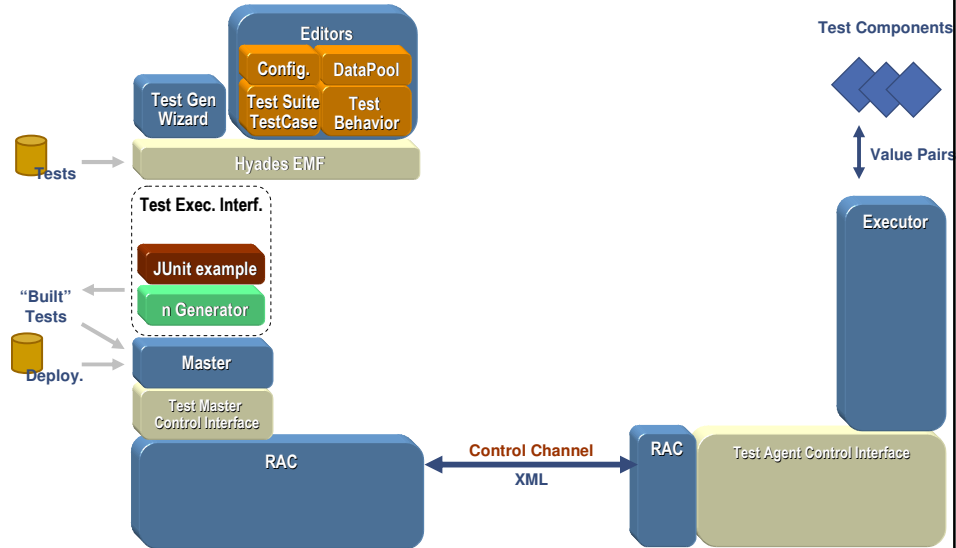


## Outline

- Integration Milestones
- ▶ Testing Roadmap
- Profiling & Logging Roadmap
- References



## Hyades testing: The high-level picture



## Milestone #1: Unmanaged Hyades tests

- This step is the first incremental step for existing tools
  - ▶ If you are starting a new project, you should look at step 2
- Tests are managed outside Hyades
  - ▶ Existing tests are imported
  - ▶ Test Suites, Test Cases, Test Contexts (or equivalent) are not created in the Eclipse shell
  - ▶ Tests behaviors, datapools and configurations do not use the Hyades data models
  - ▶ Tests can be executed from the shell and verdicts are logged



## Use Case 1.1 – Import tests and run

- Use Case
  - ▶ User clicks **File > Import** and selects **myTool** test scripts
  - ▶ A wizard enables user to select which test scripts need to be imported, and where (i.e. projects/folders) they should be imported
  - ▶ User's workspace is populated with a test case for each test script
  - ▶ Each test case references the test script to be invoked
  - ▶ User proceeds to run the test case
    - A separate user interface is brought up for the user to follow the test execution
    - The SUT (System Under Test) has been started with profiling capabilities
- Integration Points
  - ▶ Import Wizard
  - ▶ Test Execution Interfaces (IExecutionEnvironment, IExecutor, IExecutableObject, ExecutionEnvironmentAdapter, ExecutableObjectAdapter)
- Benefits
  - ▶ Tests can be executed from the Eclipse shell (N.B. Their verdict is not yet reported in the log)
  - ▶ Tests can be scheduled to be executed in a test suite along with Hyades tests
  - ▶ Users can leverage the Hyades profiling and logging capabilities provided by Hyades as they are doing their testing



## Use Case 1.2 – Run tests with verdict logging

- Use Case (extends UC1.1)
  - ▶ As the execution finishes, the test log viewer is brought up and provides the test case verdict
- Integration Points (in addition to UC1.1)
  - ▶ Test Execution Interfaces (IRemoteHyadesComponent)
- Benefits
  - ▶ Test Cases' verdicts are reported in the shell
  - ▶ Existing Test Cases can be scheduled with other Hyades Test Cases
  - ▶ Test Suites with multiple types of test cases can report into a single final verdict



## Additional considerations

- Compliance to the UML 2 Testing Profile
  - ▶ Imported test scripts need to be referenced as the behavior of a Test Case in a Test Context
  - ▶ A Test Motivator should be defined for each imported Test Case
  - ▶ The execution of a test case produces a test log with one verdict belonging to the following categories: Pass, Fail, Inconclusive, Error
- User Experience consistency
  - ▶ The execution of the imported test cases is consistent with the rest of Hyades



## Milestone #2: Managed Hyades tests

- Tests are managed from Hyades
  - ▶ Existing tests might be imported
  - ▶ Test Suites, Test Cases, Test Contexts are created and defined in the shell
  - ▶ Test behaviors, datapools and configurations, etc. may use the Hyades data models and associated editors
  - ▶ Tests can be executed locally or distributed, mixed or not mixed with other test types



## Use Case 2.1 – Create Test Case

- Use Case
  - ▶ User clicks **File > New > Test Case**.
  - ▶ A wizard enables user to select which type of Test Case needs to be created.
  - ▶ If existing Test Contexts exist, user is prompted to include the Test Case in one of the Test Contexts or create a new one .
  - ▶ Test type specific pages are displayed to let the user to select additional inputs to generate the test case .
  - ▶ User hits **Finish** .
  - ▶ The editor of the test context in which the test case is generated opens up .
- Integration Points
  - ▶ New Test Case wizard extension points
  - ▶ Test Structure meta-model
- Benefits
  - ▶ Tests can be created from the Hyades environment for the purpose of test planning .
  - ▶ Tests can be organized, versioned using Hyades test navigator and Eclipse integration with configuration management tools.
  - ▶ Test creation user experience is consistent.



## Use Case 2.2 – Edit test meta-data

- Use Case (extends UC2.1)
  - ▶ Alternate: User double-clicks a Test Case or its Test Context in the Test Navigator
  - ▶ User creates new test cases from this editor
  - ▶ User defines properties for each test case, such as test motivations, description, and ad-hoc properties specific to the test case type
- Integration Points
  - ▶ Test Context editor (if needed extensions)
  - ▶ Test Structure meta-model (if needed additional properties)
- Benefits
  - ▶ Test Cases can be linked to external requirements
  - ▶ Most of the Test Cases meta-data is consistent across the different test types users have to deal with



## Use Case 2.3 – Edit Test Suite behavior

- Use Case (extends UC2.1)
  - ▶ Alternate: User clicks the Test Navigator to open a Test Context
  - ▶ User looks at the Test Context behavior
  - ▶ User defines the behavior of the Test Context
    - The Test Context behavior invokes Test Suites and/or Test Cases of several types
- Integration Points
  - ▶ Test Context editor (if needed extensions or replacement of the Test Context behavior editor)
- Benefits
  - ▶ Test Cases of different types can be combined in Test Suites for the purpose of defining different levels of regression tests (smoke test, full regression, base features, advanced features, etc.)



## Use Case 2.4 – Run composite Test Suite

- Use Case (extends UC2.3)
  - ▶ User right clicks a Test Context in the Test Navigator and selects run
  - ▶ If no configuration has been defined for this test context, the user can select one to be used to run the test
  - ▶ The execution starts on the platforms defined in the Test Configuration for the Test Context
  - ▶ The test log summary page is displayed to the user with the final verdict
- Integration Points
  - ▶ Test Execution interfaces
- Benefits
  - ▶ Test Suites with Test Cases of mixed types can be executed
  - ▶ Test results are rolled up automatically for Test Suites executing different types of Test Cases
  - ▶ Configuration testing across different types of Test Cases is supported



## Use Case 2.5 – Analyze test log

- Use Case (extends UC2.4)
  - ▶ User gestures to open the test log detail page
  - ▶ User navigates through it to understand which validations failed
  - ▶ User clicks a validation to verify the difference between the expected value and the actual value
- Integration Points
  - ▶ Test Execution interfaces (extended IRemoteHyadesComponent)
  - ▶ test log viewer
- Benefits
  - ▶ Test execution histories are integrated
    - No need to go back and forth between test log and existing tool or view
  - ▶ Test execution histories can include trace events coming from both the test harness and the SUT
    - Verifications are correlated with events generated by the SUT enabling easier problem determination
  - ▶ Test execution histories are presented in a consistent way across tools



## Use Case 2.6 – Start Test Case behavior editor

- Use Case (extends UC2.3)
  - ▶ User double-clicks a Test Case in the Test Navigator or the Test Context editor to open its behavior
  - ▶ A specific editor provided by the tool is opened displaying the test case logic
- Integration Points
  - Test Context editor
  - Test Navigator
- Benefits
  - ▶ All the test definition actions are done from one central place
  - ▶ Test Case behavior editors might be embedded in the Eclipse shell



## Use Case 2.7 – Associate resource with Test Context

- Use Case
  - 1) User clicks a Test Context to open it
  - 2) They associate a datapool, and a datapool to the Test Context
    - These resources might be Hyades resources or tool specific resources
- Integration Points
  - ▶ Test Context editor
- Benefits
  - ▶ Test assets can be automatically deployed to the execution environment at execution time. i.e. There is no need for the user to manually deploy the assets



## Additional considerations

- User experience consistency
  - ▶ The Test Navigator is extended through its available extension points
    - There is one central point to look at all the test assets
  - ▶ The Test Log viewer is extended through its available extension points
    - There is one auditable log of a test suite execution in a well understood format.
    - Any existing reporting tools for Hyades execution histories can be used
  - ▶ The Test Context editor is extended through its available extension points
    - There is a consistent way to plan tests independently of their type
  - ▶ The Test Execution Interfaces are implemented for the master and the agent side
    - Hyades Test Configurations can be used to configure Test Cases
    - Remote execution of tests is supported
  - ▶ Test assets should comply to the UML 2.0 Testing Profile structure meta-models
    - Improvements to Test Context editors benefit all Hyades-compliant test tools



## Milestone #3: Integrated Testing

- Integrated testing Milestone
  - ▶ Legacy tests can be imported
  - ▶ All or most of the tests assets are defined in the shell
  - ▶ Tests behaviors, datapools and configurations use the Hyades data models and extend existing editors (if applicable)
  - ▶ Tests can be executed locally or distributed, with other test types or standalone, with test logs including all the details about actions and verifications



## Use Case 3.1 – Edit Test Data

- Use Case
  - ▶ User opens a Test Context and clicks a Test Case to open its data
  - ▶ The Hyades datapool editor opens with a grid enabling user to define test parameters, test data values and equivalence classes
  - ▶ User associates the parameters from the datapool with the Test Behavior of the Test Case
- Integration Points
  - ▶ Datapool editor
- Benefits
  - ▶ The Test Data can be reused by other test types
  - ▶ Extensions to the datapool editor benefit other test types



## Use Case 3.2 – Edit Test Configuration

- Use Case
  - ▶ User opens a Test Context and gestures to open one of its test configurations (associated deployments)
  - ▶ The Hyades deployment editor opens enabling user to define the platforms to be used to distribute the test execution
  - ▶ User associates the Test Components to a given machine to be used to run it
- Integration Points
  - ▶ Deployment editor
- Benefits
  - ▶ The test configuration user experience is common



## Use Case 3.3 – Edit Test Case behavior

- Use Case
  - ▶ User opens a Test Context and gestures on a Test Case to open its behavior
  - ▶ The default test behavior editor for the selected Test Case type is opened (source code)
  - ▶ User wants to look at this test case behavior using an higher level of abstraction and gestures on the Test Case to open its behavior, selecting a different behavioral editor provided by another tool vendor
  - ▶ The test behavior is open with this other editor
  - ▶ User modifies the behavior
  - ▶ As they switch to the first editor, the test behavior has been updated with the changes made in the previous step
- Integration Points
  - ▶ Test data model
- Benefits
  - ▶ User can easily migrate from two test tools leveraging the behavioral data model
  - ▶ User can leverage different editors tailored to different levels of abstraction



## Use Case 3.4 – Run tests w/ actions

- Use Case (alternate to UC1.3)
  - ▶ User gestures on a failed verdict to look at the log. The Hyades test log viewer includes all the verifications and actions performed by the test case along with those which failed
  - ▶ User gestures to open a validation action UI to look at the details of the different between the actual value and the expected one
- Integration Points (in addition to UC1.3)
  - ▶ Test Execution Interfaces (extended IRemoteHyadesComponent)
  - ▶ Log Viewer Extension Point
- Benefits
  - ▶ The execution log includes a summary that fully reflects the test
    - Auditability
    - Logs are easier to follow and correlate to tests
  - ▶ Test logs include trace events coming from both the test harness and the SUT
    - Validations and actions performed by the test harness are correlated with events generated by the SUT enabling easier problem determination



## Additional considerations

- Interoperability
  - ▶ Tools should conform to the Hyades test data models (behavior, data)

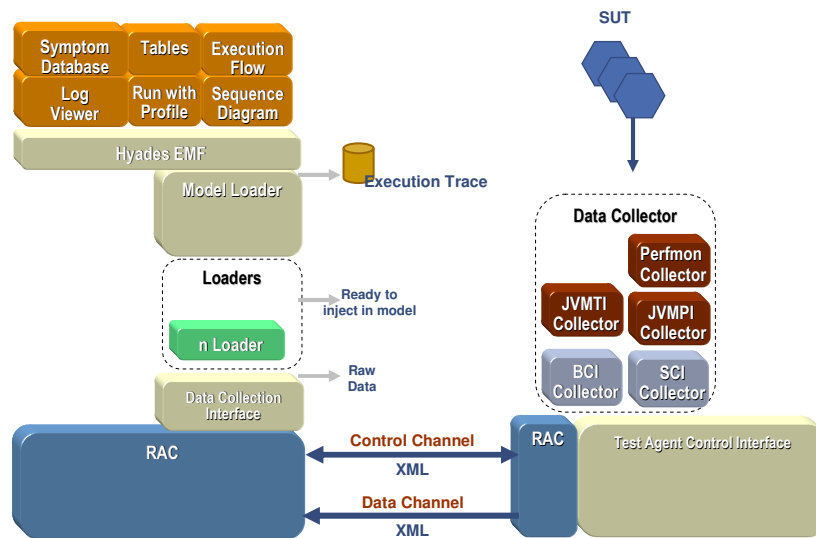


## Outline

- Integration Milestones
- Testing Roadmap
- ➔ Profiling & Logging Roadmap
- References



## Hyades Profiling & Logging: The high-level picture





## External data collection Milestone #1

- This milestone enables existing data collectors, or newly created ones to leverage the Hyades user interfaces
- External data collection Milestone
  - ▶ Runtime data and logs are collected outside of the shell and are imported manually
  - ▶ Hyades views are used to analyze the runtime data or the logs



## Use Case 1.1- Import existing data

- Use Case
  - ▶ User collects runtime data from a system using specific
  - ▶ User clicks **File > Import** and selects a file previously collected
  - ▶ User defines where the data needs to be imported (project, monitor)
  - ▶ The profiling monitor navigator displays the imported data
- Integration Points
  - ▶ Import Wizard
- Benefits
  - ▶ Runtime data can be analyzed through existing user interfaces
  - ▶ Runtime data can be associated with other Eclipse assets
  - ▶ Runtime data can be managed by configuration management tools



## Use Case 1.2 – Analyze collected data

- Use Case
  - ▶ User opens the profiling monitor navigator
  - ▶ User clicks a monitor to open the data using a trace viewer (execution flow, sequence diagram)
  - ▶ If the captured data is a “trace” (capturing historical interactions), the view is populated with the various interactions. Otherwise, the viewer displays that there is no relevant data to be displayed
  - ▶ User gestures on a higher level node in the profiling monitor navigator to look at the combined/correlated information between two captured traces
- Integration Points
  - ▶ Trace model
- Benefits
  - ▶ Existing views can be leveraged to look at specific data collected
  - ▶ Data collected by specific data collectors can be correlated with other data collected by Hyades
  
- Note: The same use case could apply for statistical data or logs,



## Additional considerations

- The collected data should populate all the relevant models properly to enable its use by existing user interfaces and proper correlation



## Integrated data collection Milestone #2

- Integrated data collection Milestone
  - ▶ Data collectors and model loaders are used to collect runtime data
  - ▶ Runtime data is stored using all or part of the Hyades data models
  - ▶ Hyades views might be extended or replaced to display collected data



## Use Case 2.1 – Start application to collect data

- Use Case
  - ▶ User clicks **Run** to start a new application and collect runtime data for it
  - ▶ User goes through the description of a launch configuration to select the nature of the data to be collected as well as its depth
  - ▶ User selects a nature related to a new data collector
  - ▶ User finishes this wizard. The application starts.
  - ▶ The profiling monitor navigator is updated to display the different monitors started on
  - ▶ As the application is running, user snapshots or refreshes the current views with relevant data collected
- Integration Points
  - ▶ Launch configuration
  - ▶ Trace/Logging/Statistical models
- Benefits
  - ▶ User can collect runtime information using several data collectors at the same time
  - ▶ Data collection can be done remotely



## Use Case 2.2 – Hook to application to collect data

- Use Case (*extends PUC2.1*)
  - 1) User configures an additional launch configuration to start collecting runtime data on an application running locally or on a remote system
  - 2) User fills in information about the process to connect to
  - 3) The profiling monitor navigator is updated to display the different monitors available on the running process
  - 4) As the application is running, user snapshots or refreshes the current views with relevant data collected
- Integration Points
  - ▶ Connect to Wizard
  - ▶ Trace/Logging/Statistical models
- Benefits
  - ▶ User can start an application and later start collecting runtime data



## Integrated profiling & logging Milestone #3

- Integrated profiling & logging Milestone
  - ▶ Data collectors are integrated with the launch configuration
  - ▶ Runtime data and logs are controlled at runtime
  - ▶ Runtime data is stored using the Hyades data models
  - ▶ Hyades views are preferably extended to display the runtime data



## Major considerations

- Interoperability
  - ▶ Existing Hyades views are extended through their extension points
    - This enables extensions to views to benefit data collected by other tools
  - ▶ Data collectors should populate all the Hyades data models with relevant data
    - Runtime data collector by a given tool can be displayed by views provided by other tool providers



## Outline

- Integration Milestones
- Testing Roadmap
- Profiling & Logging Roadmap
- ➔ References



## Hyades Resources

- Hyades website
  - ▶ <http://www.eclipse.org/hyades/>
- Join the hyades-dev mailing list
  - ▶ <https://dev.eclipse.org/mailman/listinfo/hyades-dev>
- Read the Hyades newsgroup
  - ▶ <news.eclipse.org:eclipse.tools.hyades>
- Install the current drop (from the main hyades web page)
  - ▶ Installation and usage instructions at:
    - <http://dev.eclipse.org/viewcvs/indextools.cgi/~checkout~/hyades-home/docs/Installing%20Hyades.html>



## Hyades Resources (cont)

- MOF reference
  - ▶ MOF 1.4 <http://cgi.omg.org/docs/formal/02-04-03.pdf>
  - ▶ EMF <http://www.eclipse.org/emf/>
- OMG UML 2.0 Testing Profile
  - ▶ RFP <http://cgi.omg.org/docs/ad/01-07-08.pdf>
  - ▶ Submissions <http://www.fokus.gmd.de/research/cc/tip/projects/u2tp/EMF>