



eclipse.org

Adopting the Hyades Platform

Milestones Overview



eclipse project
universal tool platform



Outline

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



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 presentation 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
- Note: This presentation assumes a good knowledge 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
 - ▶ Tests Suites, Test Cases are not created from the Eclipse environment
 - ▶ Tests 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
 - ▶ Tests 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 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

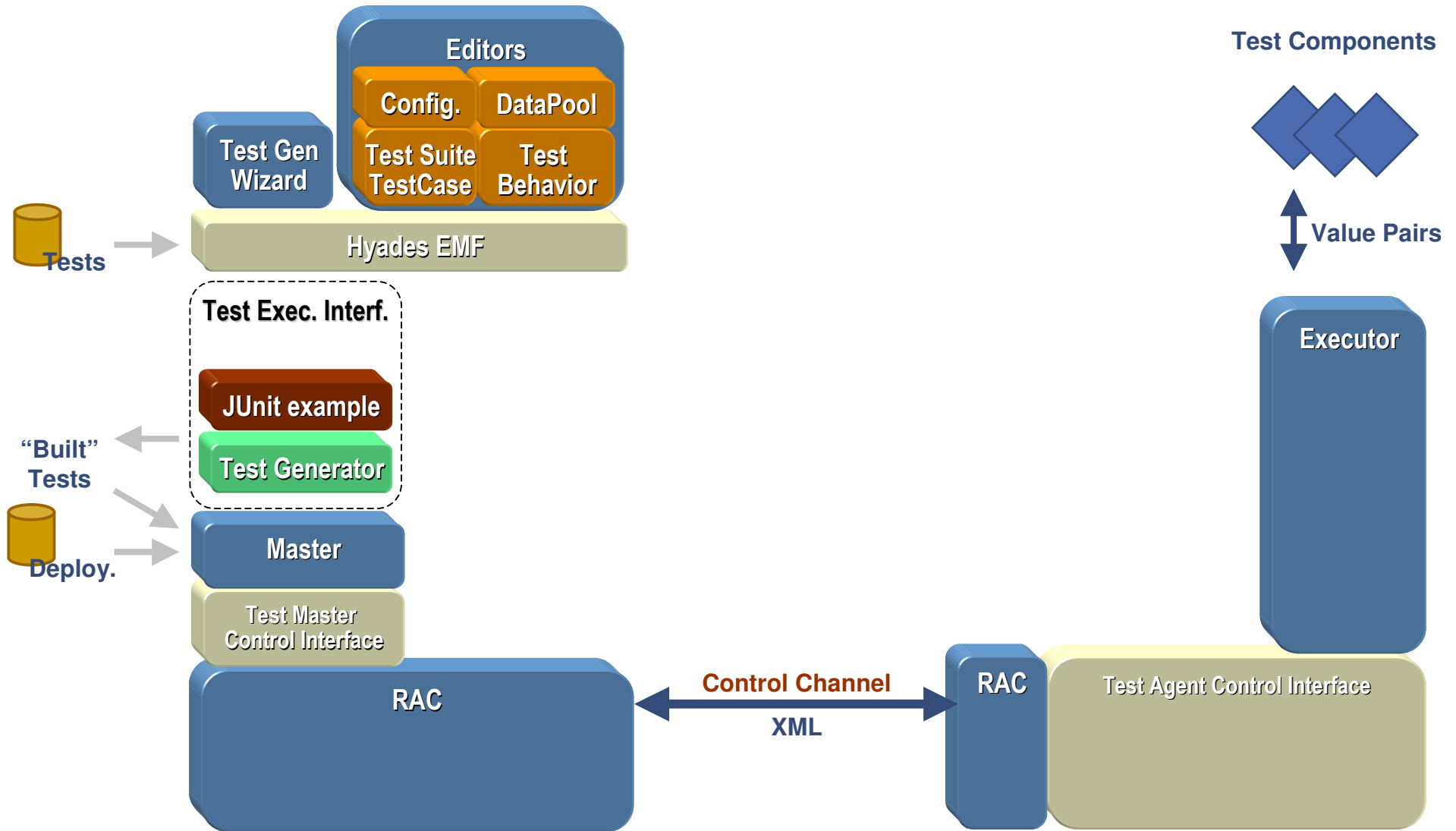


Outline

- 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
 - ▶ Tests Suites, Test Cases, Test Suites (or equivalent) are not created with Eclipse
 - ▶ 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 selects File->Import and chooses “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 runs 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 (Todo: provide example as part of Hyades)
 - ▶ Test Execution Interfaces (IExecutionEnvironment, IExecutor, IExecutableObject, ExecutionEnvironmentAdapter, ExecutableObjectAdapter)
- Benefits
 - ▶ Tests can be executed from Eclipse (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 w/ 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 verdict 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 Suite
 - ▶ 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 Suites 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



Use Case 2.1 – Create Test Case

- Use Case
 - ▶ User selects File->New->Test Case
 - ▶ A wizard enables user to select which type of Test Case needs to be created
 - ▶ If existing Test Suites exist, user is prompted to include the Test Case in one of them 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 Suite in which the Test Case is generated is brought 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 clicks the Test Navigator to open a Test Case or its Test Suite
 - ▶ User clicks to create new Test Cases from this editor
 - ▶ User defines for each Test Case properties such as test motivations, description, and ad-hoc properties specific to the Test Case type
- Integration Points
 - ▶ Test Suite 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 Suite
 - ▶ User opens the Test Suite behavior
 - ▶ User defines the behavior of the Test Suite
 - The Test Suite behavior invokes Test Suites and/or Test Cases of several types
- Integration Points
 - ▶ Test Suite editor (if needed extensions or replacement of the Test Suite 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 Suite in the Test Navigator and selects run
 - ▶ If no configuration has been defined for this Test Suite, 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 Suite
 - ▶ The test log summary page is displayed to the user with the final verdict
- Integration Points
 - ▶ Test Execution interfaces
- Benefits
 - ▶ Test Suites with mixed types of Test Cases 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 clicks to open the test log detail page
 - ▶ User navigates through it to understand which validations failed
 - ▶ User double 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 clicks a Test Case in the Test Navigator or the Test Suite editor to open its behavior
 - ▶ A specific editor provided by the tool is opened displaying the Test Case logic
- Integration Points
 - Test Suite 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 environment



Use Case 2.7 – Associate resource with Test Suite

- Use Case
 - 1) User double-clicks a Test Suite to open it
 - 2) They associate a Datapool to the Test Suite
 - These resources might be Hyades resources or tool specific resources
- Integration Points
 - ▶ Test Suite 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 Suite 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 Suite 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, mixed or not with other test types, with test logs including all the details about actions and verifications



Use Case 3.1 – Edit Test Data

- Use Case
 - ▶ User opens a Test Suite and selects 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 Suite and opens 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 Suite and selects 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 selects the Test Case to open its behavior, selecting a different behavioral editor provided by another tool vendor
 - ▶ The test behavior is opened 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 clicks 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 clicks a validation action 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)

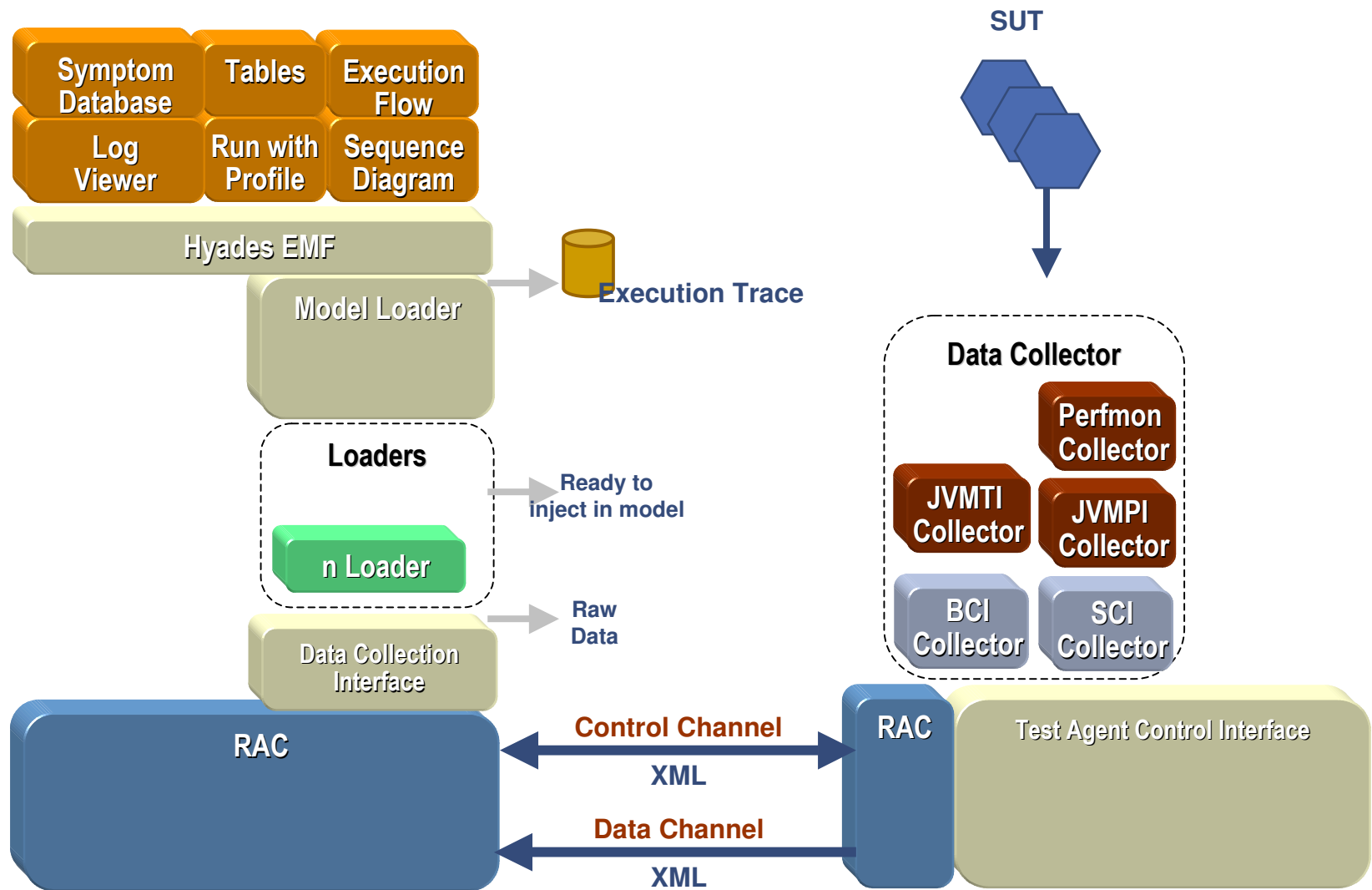


Outline

- 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 a specific data collection mechanism
 - ▶ User clicks File->Import and selects a file previously collected
 - ▶ User defines where the data needs to be imported (project, monitor)
 - ▶ The profiling monitor 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
 - ▶ 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 clicks a higher level node in the profiling monitor 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 selects “Run ” to start a new application and collect runtime data for it
 - ▶ User creates a new launch configuration
 - ▶ User selects a profiling set, along with the limits for the data collection
 - ▶ User clicks “Profile”. The application starts.
 - ▶ The profiling monitor is updated to display the different monitors available
 - ▶ 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 – Attach 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 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 from 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 build (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 (Meta-Object Facility) reference
 - ▶ MOF 1.4 <http://cgi.omg.org/docs/formal/02-04-03.pdf>
 - ▶ EMF (Eclipse Modeling Framework) <http://www.eclipse.org/emf/>
- Object Management Group UML 2.0 Testing Profile
 - ▶ Request For Proposal <http://cgi.omg.org/docs/ad/01-07-08.pdf>
 - ▶ Submissions <http://www.fokus.gmd.de/research/cc/tip/projects/u2tp/EMF>