

AGEDIS

AGEDIS TESTING METHODOLOGY

OVERVIEW

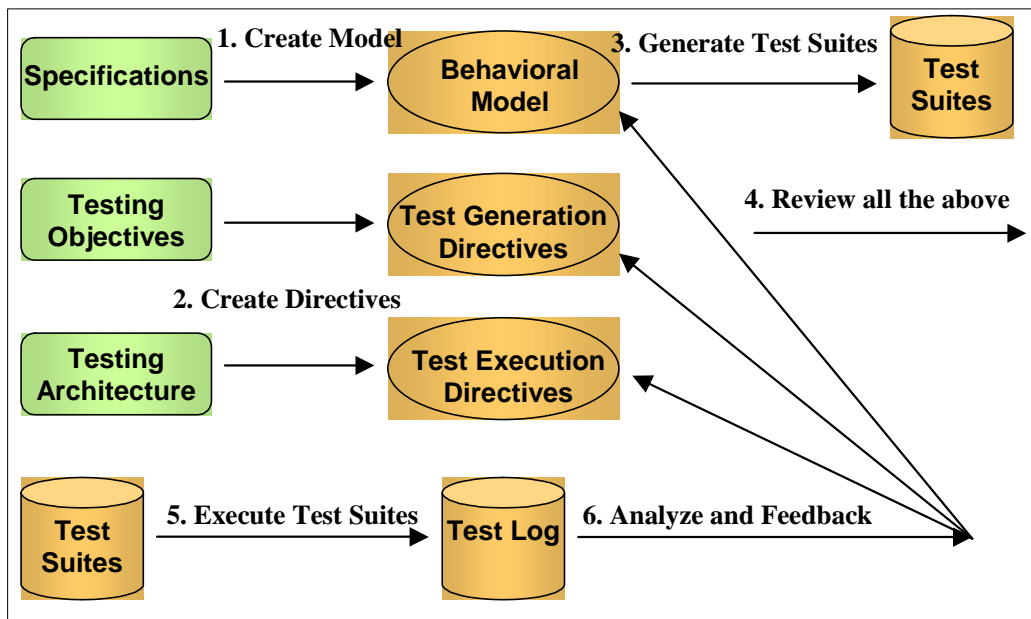


Figure 1. Testing Methodology

The AGEDIS testing methodology is based on an iterative process with the following six key steps:

1. Build a behavioral model of the system under test (SUT).
2. Annotate the model with testing information:
 - Describe coverage criteria, specific test purposes, and testing constraints.
 - Describe the testing interface between the model and the SUT, including points of control and observation.
3. Generate, automatically, a test suite.
4. Review the model, test information, and test suite with developers and customers.
5. Execute the test suite automatically, and log the results.
6. Review the results and repeat steps 2 – 5 until the coverage and quality goals of the test are achieved.

AGEDIS provides the following tools to assist with the implementation of this testing methodology:

- Model Compiler – takes a model written in the AGEDIS Modeling Language (AML), and produces a standard intermediate format that the test suite generator can read. The modeling language includes both the behavioral model and the test generation directives.
- Test Suite Generator – reads the intermediate format and produces a test suite written in the Abstract Test Suite (ATS) format.
- Test Execution Engine – reads the abstract test suite and the test execution directives, runs the test suite against the application under test, and logs the test results.
- AML Profile – enables the Objecteering UML modeling tool to produce models in AML.
- Abstract test suite browser.

The following tools are still under development:

- An integrated platform to control and coordinate the entire AGEDIS tool set.
- A test log browser.
- Automated components for the analysis and feedback step of the methodology.

The relationship between the tools and the data transfer formats is illustrated below.

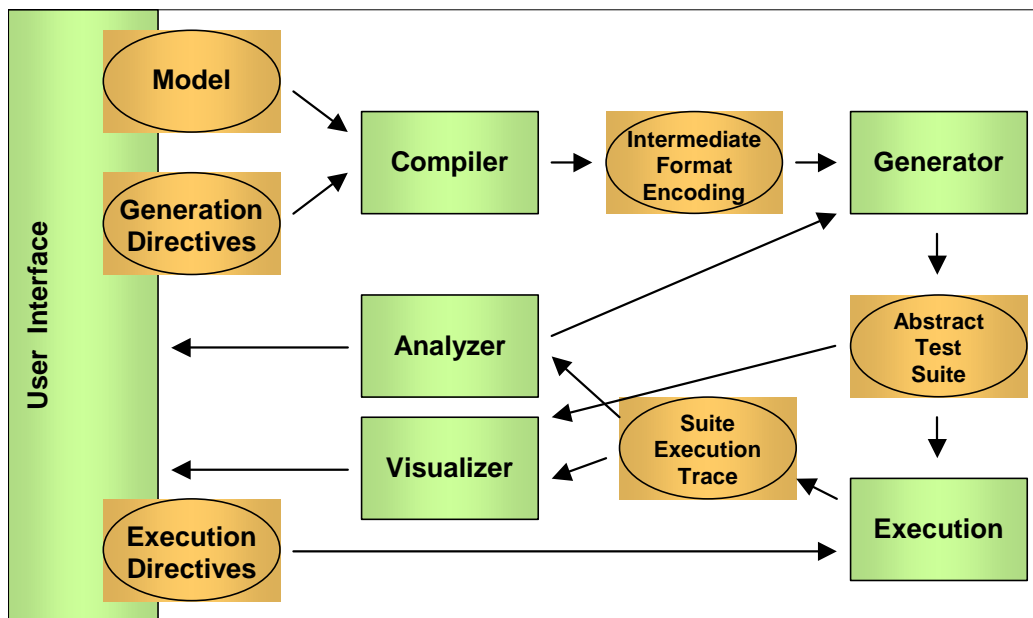


Figure 2. AGEDIS tool architecture.

The modeling language specification and data transfer formats are publicly available from the consortium [website](http://www.agedis.de/downloads.shtml) (www.agedis.de/downloads.shtml).

MODELING

Modeling is the process of creating a test adequate abstraction of the functionality of the application under test. If the application is being developed with a model-based approach, many of the artifacts used in the design models may be reused for testing. The testing model comprises class, object, and state diagrams.

Modeling the application under test, and the creation of test generation directives are currently only supported using the Objecteering UML editor together with the AML profile for that tool. We intend to support other modeling tools and modeling languages in the future.

TEST GENERATION

Test suites are automatically generated from the behavioral model. The abstract test suite includes sequences of stimuli and the expected behavior of the system in response to those stimuli. The expected behavior is derived from the model. The test generation directives control the size and other aspects of the test suite. Test generation directives include:

- Generic coverage criteria
- Specific test purposes
- Restrictions or constraints on the test cases and the test suite to be generated.

TEST EXECUTION

The test execution engine reads the abstract test suite and the test execution directives. It then runs the test cases and logs the results.

The test execution directives include:

- **Constants** – global constants to be used by the execution engine throughout the test execution.
- **Definitions** – describe the SUT (host machines, processes, objects and classes) and give details of its distribution, synchronization, and multiplication (cloning).
- **Initialization** – instructions on how to initialize all objects, processes, and hosts.
- **Extensions** – actions to extend the ATS (setup, cleanup, additional evaluations).
- **Options** – various global execution options.
- **Mappings** – ATS to SUT mapping directives for types, constants, controls and observable features of the SUT.

The test execution engine has three main components:

- **Test Suite Driver** – controls all other components.
- **Host Manager** – represents the Test Suite Driver on each host machine in the execution environment.
- **Process Manager** – maintains SUT Objects.

REFERENCES

- [1] AGEDIS Consortium, Automated Generation and Execution of Test Suites for Distributed Component-based Software, <http://www.agedis.de>.