Test and Benchmark Framework

Tests

Automatic JUnit tests are executed in each continuous integration build in Bamboo. They can also be run locally in your KIELER development environment.

All tests are located in the test folder of the semantics repository. They perform their test based on models loaded from the models repository.
Executing these test requires a local checkout of the repository. The path to the repository must be added to the following environment variable when executing the tests:

```
models_repository=path/to/models/repository
```

it is also possible to specificy multiple repositories, using the following notation:

```
models_repository=[path1, path2]
```

Models Repository

The models repository uses **property files** to detect and configure models used in tests.

The properties associated with a model file are derived from a hierarchy of property files. All files named directory.properties assign properties to the directory they are located and **all subdirectories**. Files named modelA.properties assign properties to all files in the same directory with the same filename, i.e. modelA.sct or modelA.broken.sct.

There are some predefined properties which generally control model detection and categorization but you can add any other property.

Key	ValueType	Default	Combination	Description	Example
ignore	Bool	True	Override	Ignored model files / directories will not be included in the automatic testing process.	`ignore = false`
confidential	Bool	False	Override	If set the test / benchmarks should not publish any information about the content or meta-data of the model.	`confidential = true`
modelFileExtension	Comma separated list of strings	Empty	Override	A list of file name suffixes (file extensions) for identify model files.	`modelFileExtension = sct`
traceFileExtension	Comma separated list of strings	Empty	Override	A list of file name suffixes (file extensions) for identify test trace files.	`traceFileExtension = eso, .trace`
resourceSetID **DEPRECATED**	String	Empty	Not propagated	A globally unique identifier which will cause the associated model files to be loaded into one resource set which allows resolving cross references between models files.	`resourceSetID = my-unique-id`
modelProperties	Comma separated list of strings	Empty	Combined	A list of model specific categories that should be assigned to the model. The categories are handled as a set where is property file in the hierarchy can add or remove (using !) new tags.	`modelProperties = tiny-model, !broken`
Other properties	String	Empty	Override	Any user specific property.	`complexity = 9001`

Note that ignore is true by default, that means you have to set it explicitly to include new files/folders in the automatic testing process.

Benchmarks

WIP

Benchmarks run similar to test. They run the models in a models repository, provided in the same way as mentioned before.

Local Benchmarks

To run the benchmarks locally you first have to provide the models repository (including the environment variable mentioned above). Then you need the appropriate plug-ins. in you runtime configuration.

To activate the local benchmarks set the following environment variable:

```
local_benchmark=project_name
```

Where the project_name specifies the project to save the results into. The benchmarks will create the project if it does not exist and will create a json file containing the benchmark results.