Advanced Topics in Software Engineering: Sibilla: a framework for simulation and analysis stochastic systems

Prof. Michele Loreti

Advanced Topics in Software Engineering
Corso di Laurea in Informatica (L31)
Scuola di Scienze e Tecnologie

Sibilla: a framework for simulation and analysis of stochastic systems

Sibilla is a Java framework designed for supporting analysis of stochastic systems.

Sibilla: a framework for simulation and analysis of stochastic systems

Sibilla is a Java framework designed for supporting analysis of stochastic systems.

It is a container where the tools supporting specification and analysis of concurrent and distributed large scaled systems can be integrated.

Sibilla: a framework for simulation and analysis of stochastic systems

Sibilla is a Java framework designed for supporting analysis of stochastic systems.

It is a container where the tools supporting specification and analysis of concurrent and distributed large scaled systems can be integrated.

Current implementation of Sibilla includes:

- a set of API for system simulation;
- a set of API for modelling population models.

Sibilla: a framework for simulation and analysis of stochastic systems

- a usable GUI to simplify design and analysis;
- tools for statistical model checking;
- runtime monitoring;
- support to CSL (Carma Specification Language) for the specification and analysis of Collective Adaptive Systems.

Sibilla: Simulation Classes

Sibilla system simulator is based on two concepts:

- Model<S>: this is an interface describing a system to simulate

Sibilla: Simulation Classes

Sibilla system simulator is based on two concepts:

- Model<S>: this is an interface describing a system to simulate
  - ... the type parameter S is the datatype representing system configurations.

Sibilla: Simulation Classes

Sibilla system simulator is based on two concepts:

- Model<S>: this is an interface describing a system to simulate
  - . . . the type parameter S is the datatype representing system configurations.
- SimulationEnvironment<M extends Model<S>,S>: this is the class the implement the simulation framework for a state S of a model M.

Sibilla: Model<S>

```
public interface Model<S> {

  public WeightedStructure<StepFunction<S>> getActivities(
    RandomGenerator r , S s );

  public S initialState();

}
```

Sibilla: Model<S>

```
public interface Model<S> {

  public WeightedStructure<StepFunction<S>> getActivities(
    RandomGenerator r , S s );

  public S initialState();

}
```

where:

- WeightedStructure is a data structure specifically thought to pick elements according to a weight associated to them;
- StepFunction<S> indicates a possible evolution of a system.

Sibilla: SimulationEnvironment<M,S>

```
public class SimulationEnvironment<M extends Model<S>,S> {
  ...
  public Trajectory<S> sampleTrajectory( double deadline ) {
    ...
  }

  public Trajectory<S> sampleTrajectory( double deadline ,
    Predicate<S> reachPredicate ) {
    ...
  }

  public Trajectory<S> sampleTrajectory( double deadline ,
    Predicate<S> transientPredicate , Predicate<S>
    reachPredicate ) {
    ...
  }
}
```

Sibilla: SimulationEnvironment<M,S>
Trajectory<S>

Class Trajectory<S> is used to describe the realisation of a computation...

Sibilla: SimulationEnvironment<M,S> Trajectory<S>

Class Trajectory<S> is used to describe the realisation of a computation...

Data can be retrieve from a Trajectory<S> via the method...

```
public void sample(SamplingFunction<S> f)
```

Sibilla: SimulationEnvironment<M,S> Trajectory <S>

Class Trajectory <S> is used to describe the realisation of a computation...

Data can be retrieve from a Trajectory <S> via the method...

```
public void sample(SamplingFunction<S> f)
```

A SamplingFunction<S> is used to collect data e compute statistics of the data retrieve by a trajectory.

Sibilla: SimulationEnvironment<M,S> SimulationTask<S>

A SimulationTask<S> is responsible for the simulation of a single trajectory.

Sibilla: SimulationEnvironment<M,S> SimulationTask<S>

A SimulationTask<S> is responsible for the simulation of a single trajectory.

Note that multiple tasks can be executed in parallel to improve efficiency.

This is the end!