Integrating Statechart Components in Polyglot

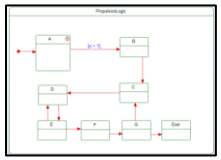
Daniel Balasubramanian, Gabor Karsai
ISIS / Vanderbilt University

Jason Biatek, Michael W. Whalen
University of Minnesota

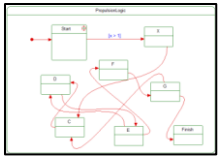
Corina Pasareanu, Thomas Pressburger, Michael Lowry
NASA Ames Research Center

Supported by NASA Cooperative Agreement NNX09AV58A

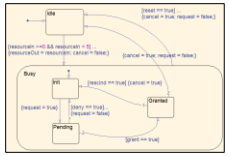
Motivation



UML
Statecharts



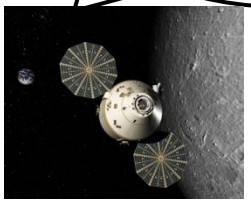
Stateflow
Statecharts



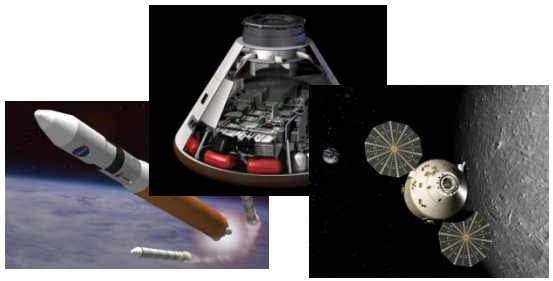
Rhapsody
Statecharts

Ares

Orion



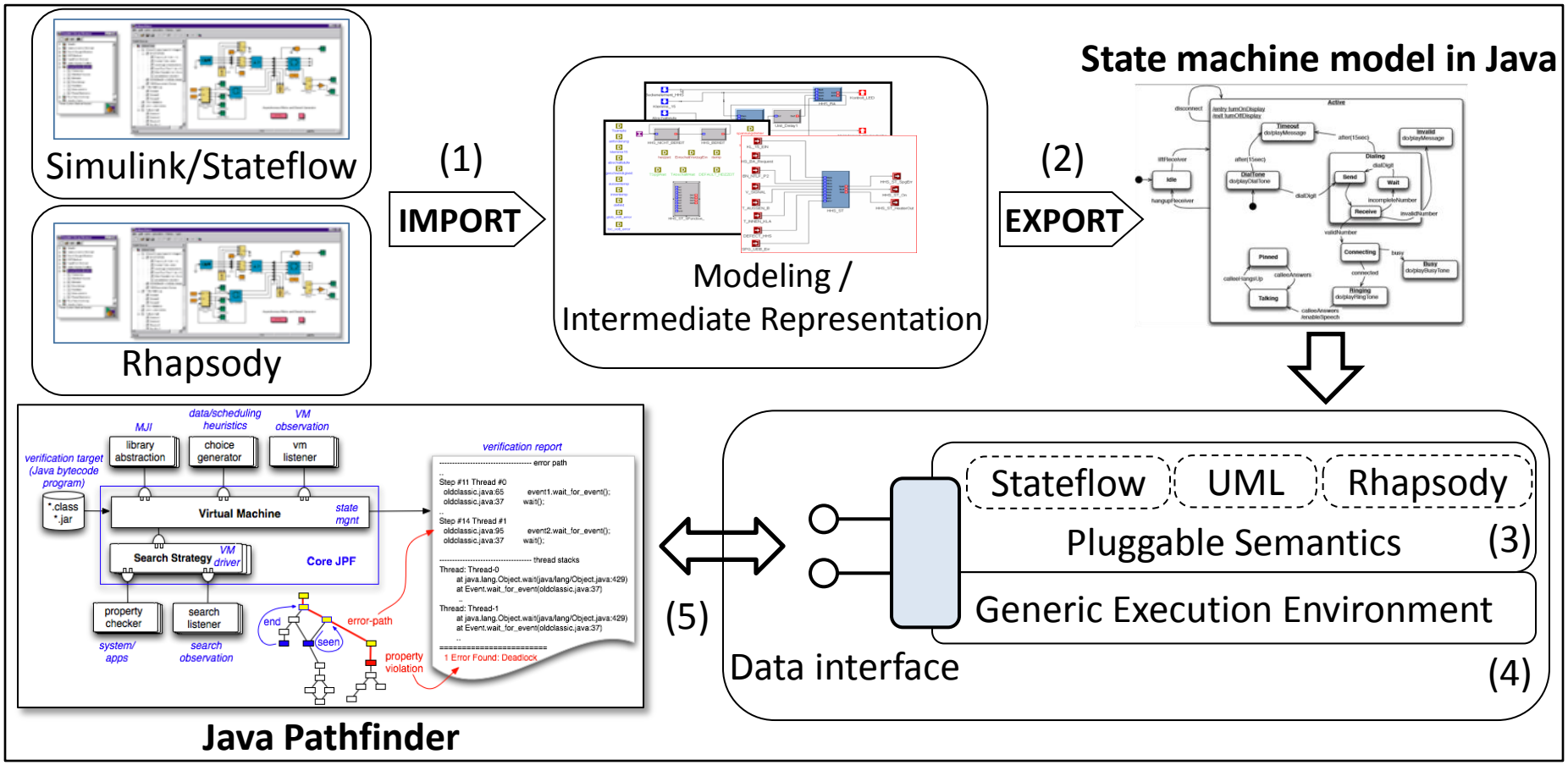
Orion (software framework)



Integrated system

Integrated system can't be analyzed at the model level in any of the original environments…

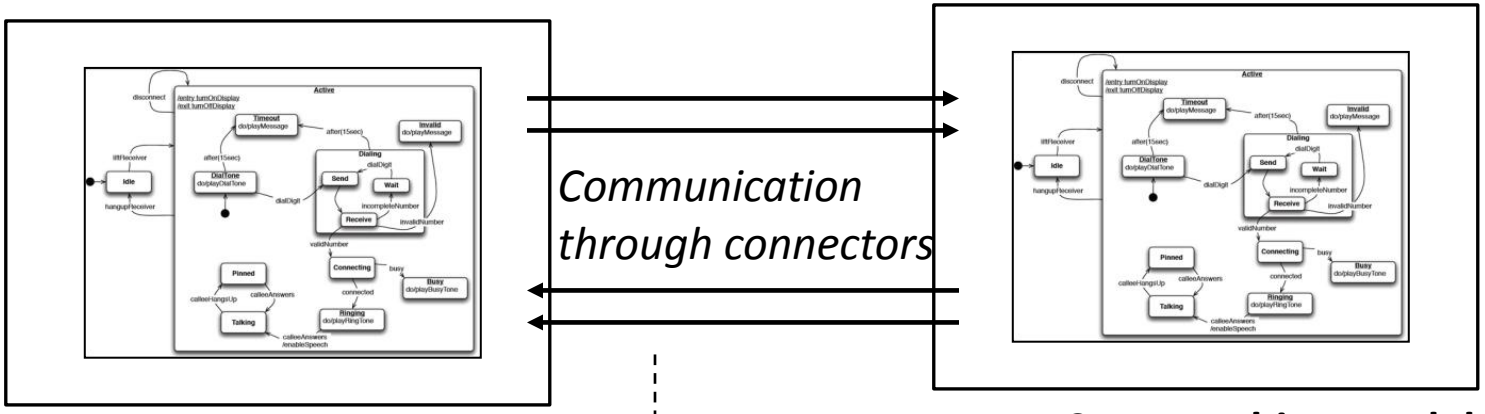
Polyglot: a Statechart Analysis Framework



What about communicating, asynchronous components?

Extension 1: Connectors

- A **connector framework** provides generic communication mechanism.
  - Basic mechanism is FIFO, non-lossy. Library includes lossy and non-FIFO connectors.



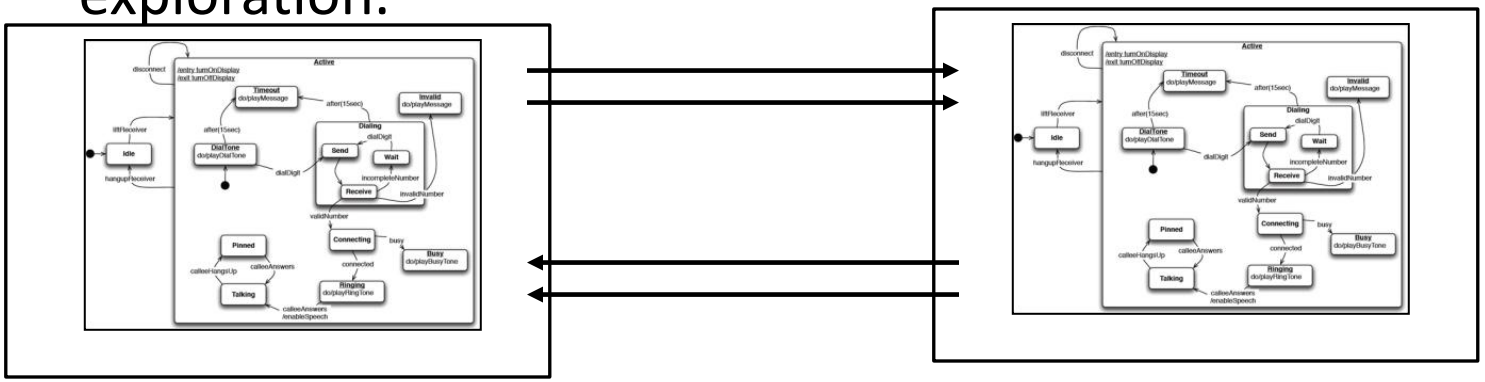
Stateflow model

UML Statemachine model

```
public interface Connector {
  boolean recvFrom(Component to, Component from, Object[]  res);
  boolean sendTo(Component from, Component to, Object arg);
}
```

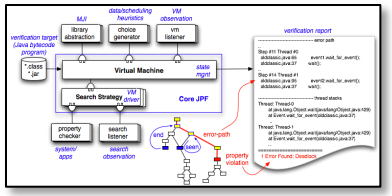
Extension 2: scheduling

- A generic **scheduling framework** sequences component execution and property checking.
  - Basic version is non-deterministic. Integrated with JPF for exploration.



Stateflow model

UML Statemachine model



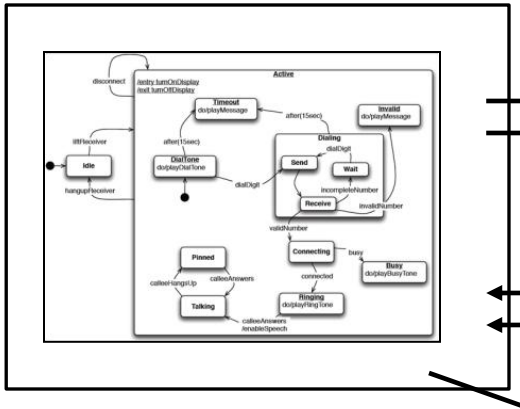
Java Pathfinder

Guides execution

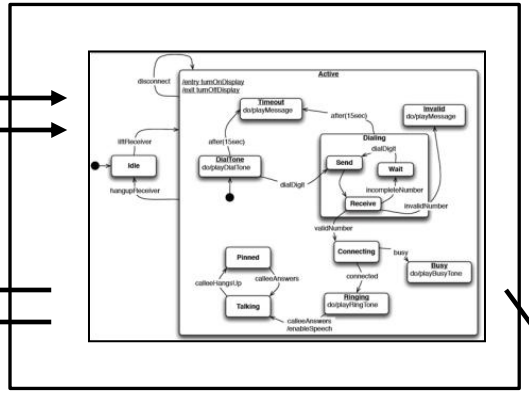
```
class BasicScheduler extends Scheduler {
  public void run {
    // non-deterministically choose component
    // check properties (next)
}
```

Extension 3: properties

- Observer automata used to check properties

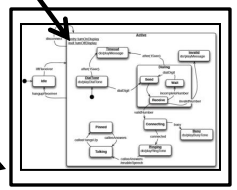


Stateflow model



UML Statemachine model

inputs

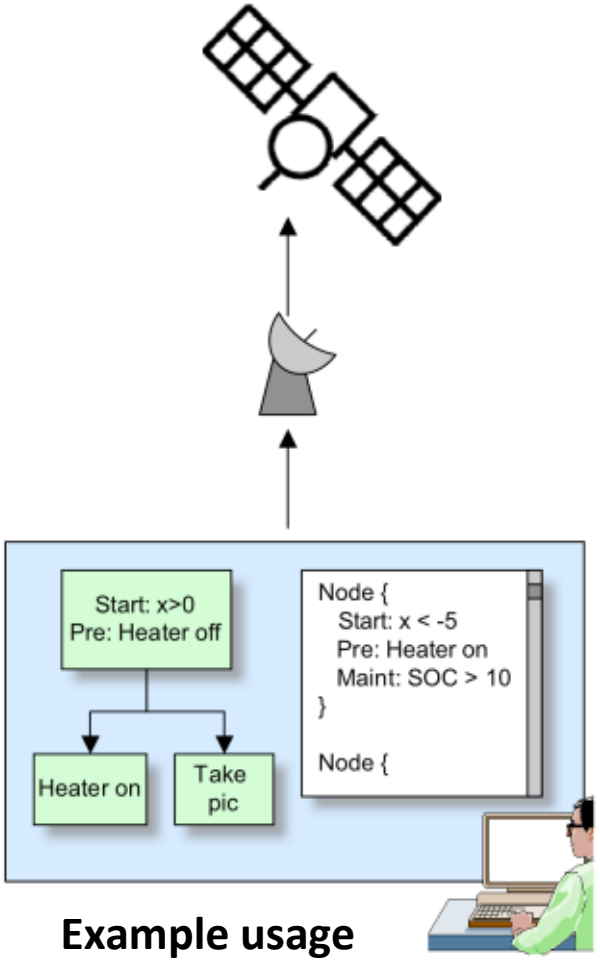


Observer automaton

```
public class Observer {
  void run() {
    // determine state of observed components
    // check property
    // report violation
  } }
```

Plexil integration

- PLan EXecution Interchange Language
- Reactive, synchronous, and deterministic language for commanding and monitoring space systems



Example usage

PLEXIL to Java translation

- Plexil uses implicit state machine nodes and a 3-valued logic (True, False, and Unknown), interacting with the outside world using lookups
- Initially, JPF took over ***13 hours*** to verify a relatively simple plan
  - Choice points for lookups were being created on demand in the middle of a step, causing JPF to capture irrelevant state information
  - By precomputing values between steps, that ***time dropped to just 8 minutes***, with further optimizations possible
- JPF verification uncovered a problem when If(Unknown) occurred, where the Then and Else branches would not start (since the expression was not true or false) causing the plan to become unresponsive
- Regression testing against the reference PLEXIL implementation also revealed a possible infinite loop in a state machine

Ongoing work

- More efficient analysis of Plexil plans
- Efficient symbolic execution of Statecharts in Polyglot
  - Custom symbolic execution engine, multithreading

Questions?