Exploratory modeling with SmallDEVS

Vladimír Janoušek
Elős Kironský

Faculty of Information Technology
Brno University of Technology, Czech Republic

Outline

- Class-based modeling
- Prototype-based modeling
- Exploratory modeling
- SmallDEVS

Context

- Discrete Event Systems Specification
  - Atomic models
  - Coupled models
- Many available implementations:
  - DEVSJava
  - DEVSC++
  - PyDEVS

Class-based modeling

- Atomic components are defined as classes
- Coupled components as well
- Structural changes possible
- No new atomic components can be introduced at run-time
- Poor support of interactive model building & testing

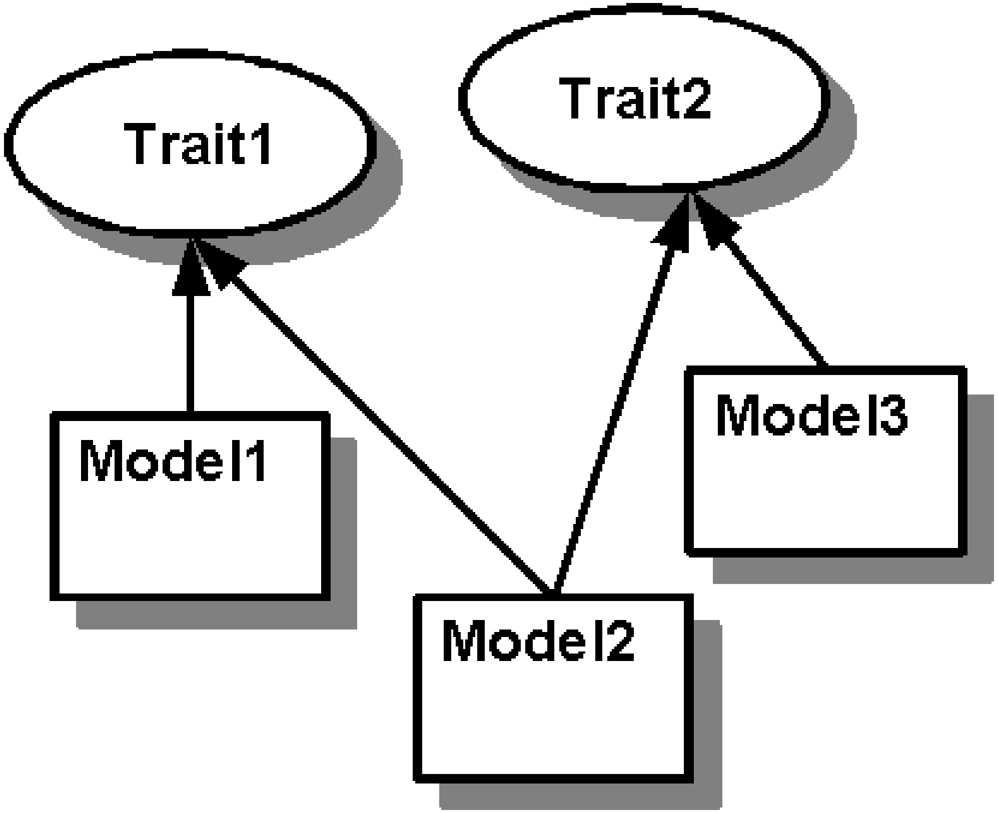
Prototype-based modeling

- Proto-objects
  - Define their own slots & behaviour
  - Self
- No key feature of class-based approach is lost
- More flexibility in object building, reusability, behaviour sharing

Behaviour sharing

- Traits
- Delegation
- Dynamic inheritance
- Attachable to models
- Implemented as proto-objects

Behaviour sharing



Exploratory modeling

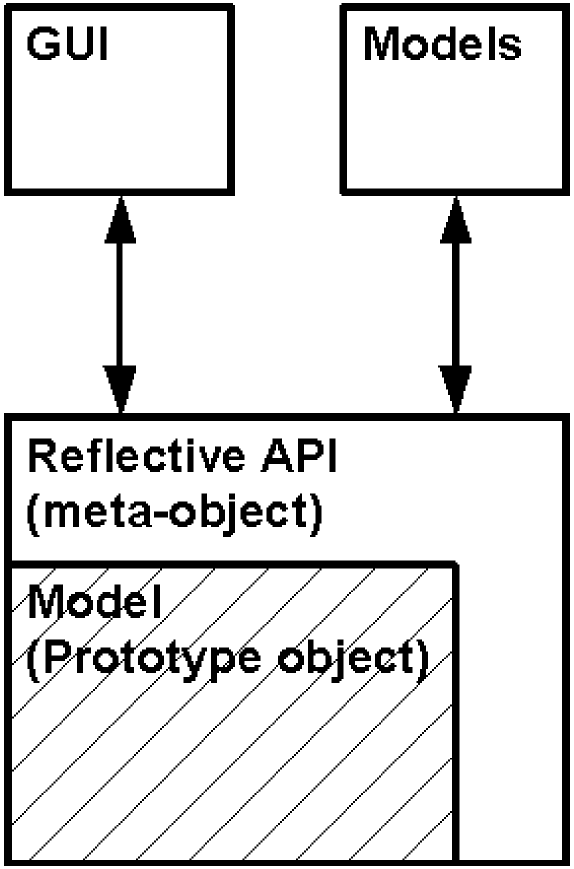
- Analogy of exploratory programming
- Interactive exploring of the state of a running system
- Live objects (models)
- Editing live models at run-time
- Exploring the effects of editing
- Reflectivity

Reflectivity

- Available through methods of a reflective API (meta-objects)
- Both for models and simulations
- Inspect models and simulation state anytime
- Needed for evolvable models
- Models can inspect and edit models

Reflectivity

- Wrapped prototypes
- GUI is inspecting models through interface
- Models can change models



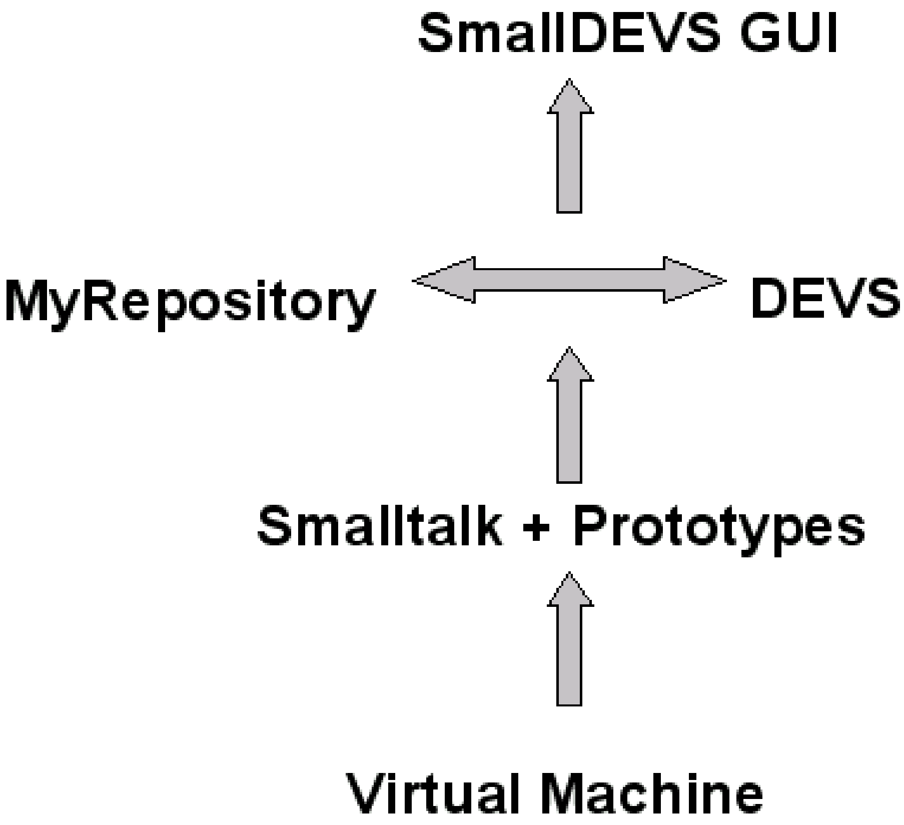
SmallDEVS system

- Implemented in Squeak Smalltalk
- DEVS modeling and simulation environment

SmallDEVS system

- Supports:
  - Class-based modeling
  - Prototype-based modeling (preffered)
- Simulation of atomic and coupled models
- Model management
- Models can change during run-time
- Dynamical, Flexible, Interactive

SmallDEVS architecture


SmallDEVS GUI

MyRepository

DEVS

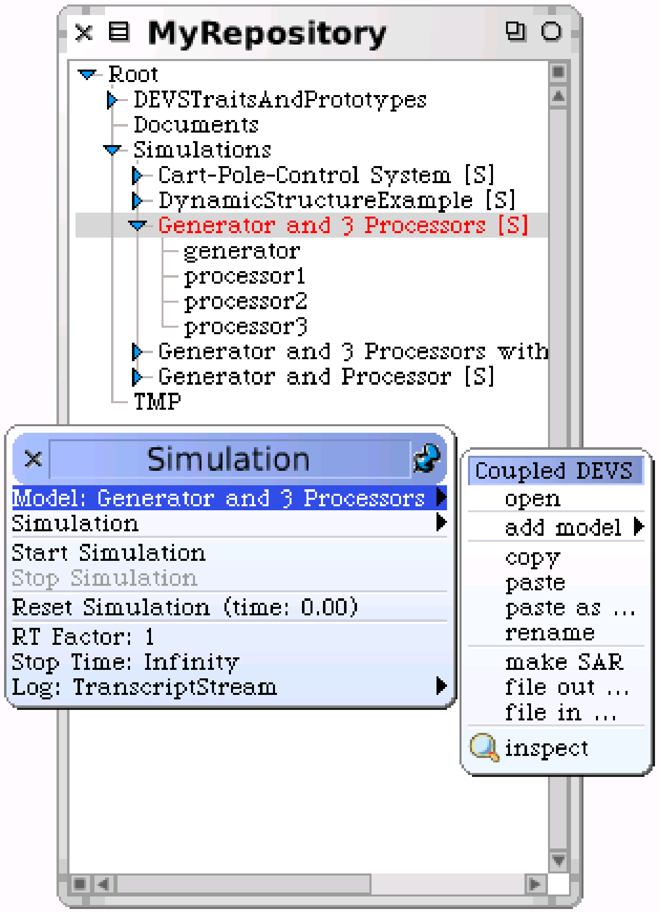
Smalltalk + Prototypes

Virtual Machine

SmallDEVS-MyRepository

- File manager like
- Manages models, simulations and other objects (traits, documents, …)
- Hierarchical structure visualization
- Operation with models (objects)

SmallDEVS-MyRepository



Reset Simulation (time: 0.00)

RT Factor: 1

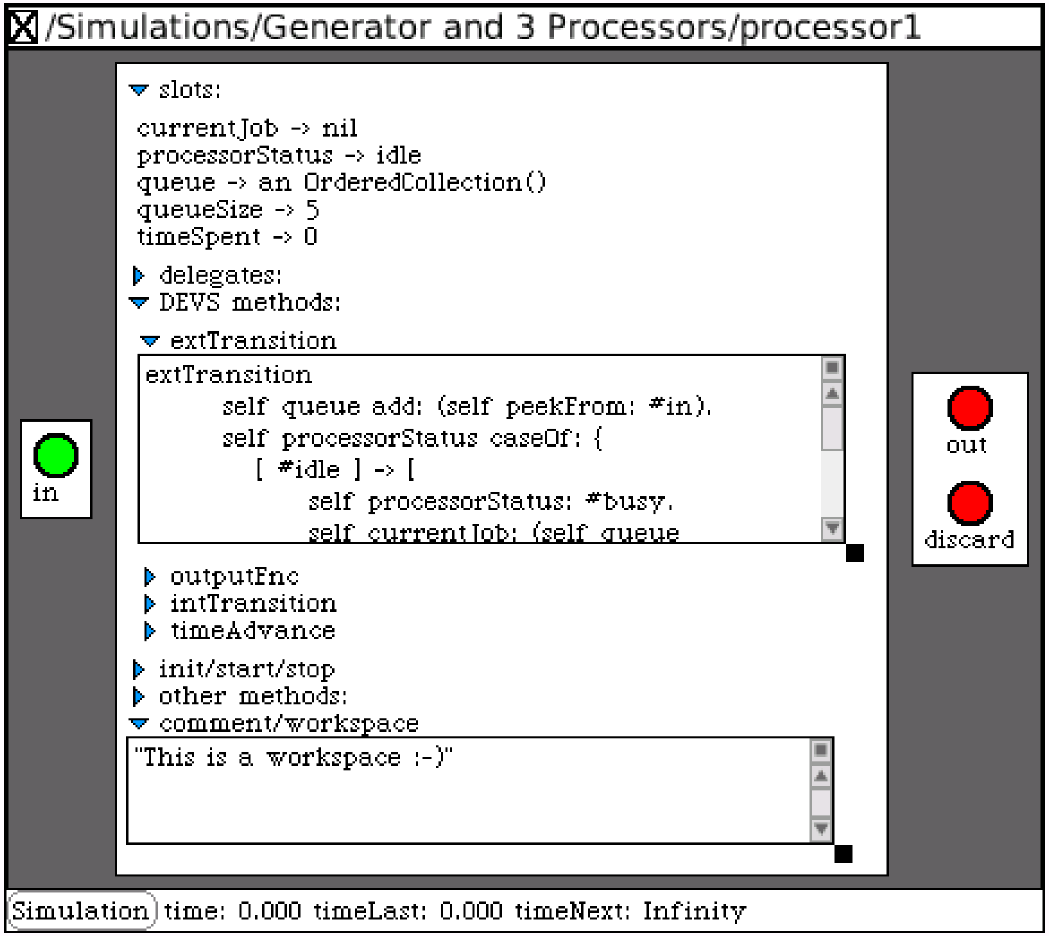
Stop Time: Infinity

Log: TranscriptStream

SmallDEVS-Atomic inspector

- Atomic model inspector
- Atomic model editor
- Inspired by Self’s outliner

SmallDEVS-Atomic inspector



☒ /Simulations/Generator and 3 Processors/processor1

▼ slots:

```
currentJob -> nil
processorStatus -> idle
queue -> an OrderedCollection()
queueSize -> 5
timeSpent -> 0
```

▶ delegates:

▼ DEVS methods:

▼ extTransition

```
extTransition
        self queue add: (self peekFrom: #in).
        self processorStatus caseOf: {
           [ #idle ] -> [
                self processorStatus: #busy.
                self currentJob: (self queue
```

▶ outputFnc

▶ intTransition

▶ timeAdvance

▶ init/start/stop

▶ other methods:

▼ comment/workspace

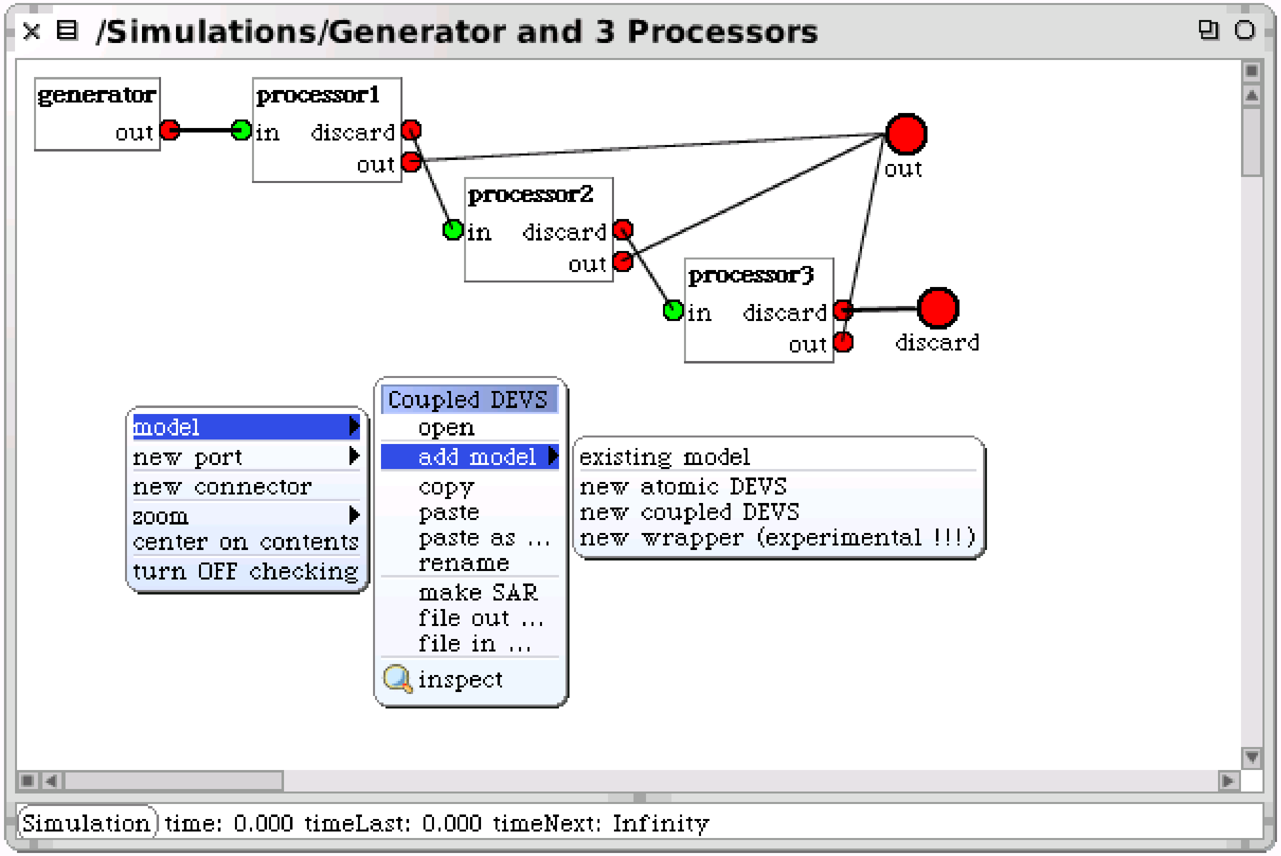
```
"This is a workspace :-)"
```

time: 0.000 timeLast: 0.000 timeNext: Infinity

SmallDEVS-Coupled inspector

- Graphical representation of a coupled model's structure
- Edit structure interactively
- Access to all key elements of SmallDEVS

SmallDEVS-Coupled inspector



time: 0.000 timeLast: 0.000 timeNext: Infinity

Why SmallDEVS?

- Real bottom-up approach, from concrete models to abstractions with shared behaviour
- Understanding by modeling
- No difference between a model and any snapshot of a running simulation
- Automatic evolution of models during simulation

Future research

- Multi-paradigm modeling
- DEVS description meta-language
  - Platform independent
  - Debug in SmallDEVS, simulate in C++
- Model continuity

References

- www.fit.vutbr.cz/~janousek/SmallDEVS