Prototyping SDL Extensions

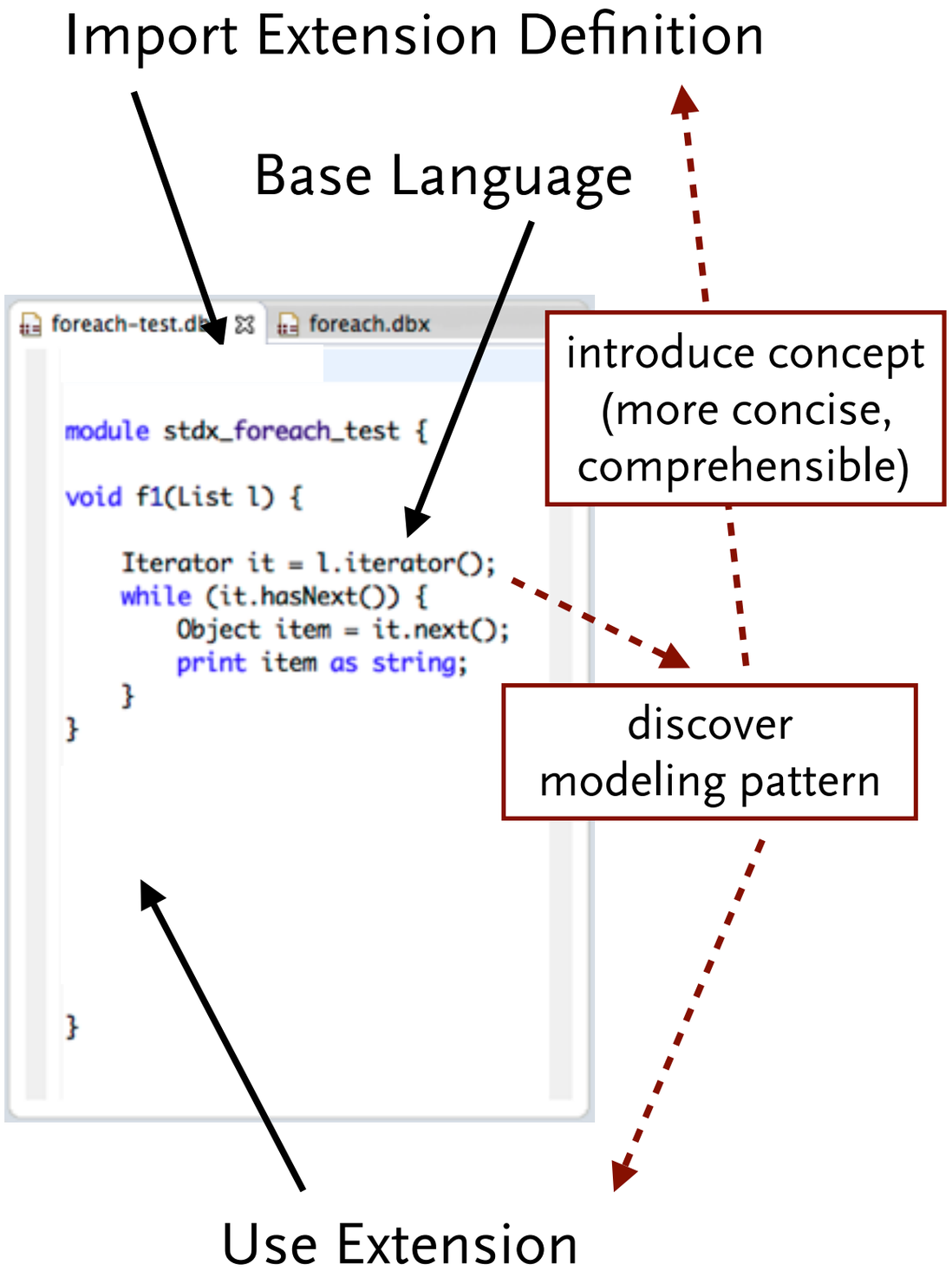
Andreas Blunk and Joachim Fischer

Department of Computer Science,
Humboldt University Berlin, Germany

Outline

- Approach Overview
- Contribution
- Approach Details
- Application to SDL
- Summary

Overview



- An approach for extending a base language by more specific concepts
- Supports iterative development
- Automatically provides tools at each stage
  - Textual editor (modeling)
  - Runtime efficient next-event simulator (model analysis)
- Allows to evaluate design and suitability of a new concept
  - can be directly used in models
  - evaluate the performance of a system modeled
- Simple example: pattern for iterating over list data structure
- Prototyping useable for small concepts (for-each), aim for Domain-specific Languages (DSLs)

Contribution

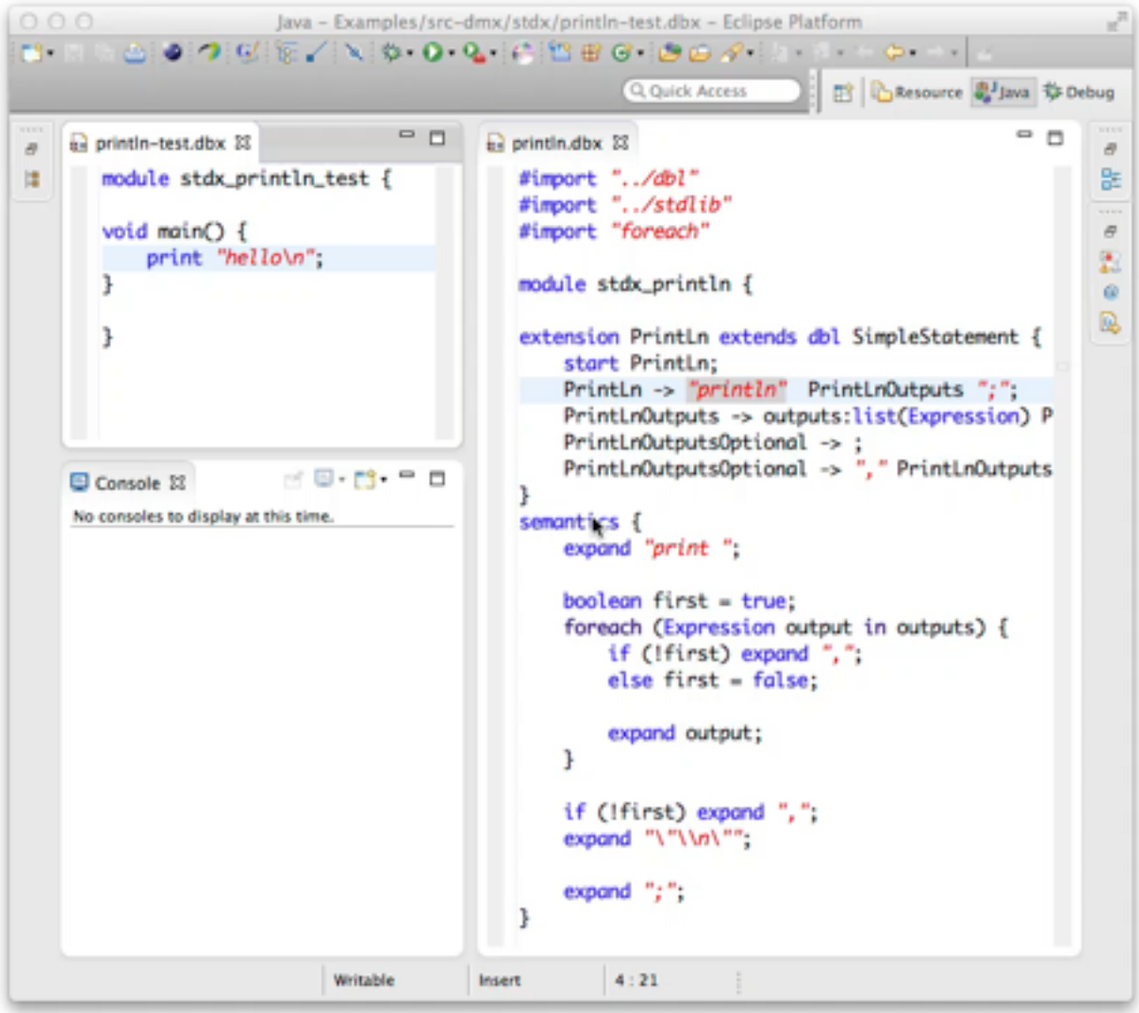
- We understand SDL as a DSL
  - Specific concepts for modeling structural and functional aspects of communication systems
  - General concepts regarding the domain itself
  - If domain gets more specific, e.g. real-time systems, more specific concepts may be needed
  - Examples of proposed SDL extensions: **SDL-RT Semaphores**, Process Priorities, Real-Time Tasks (without integrated tool support)
- Goal
  - Apply the approach to SDL as an archetype
  - Get confidence for possible successful applications to other DSLs

Approach

- Approach is targeted towards
  - DSLs which are used for model analysis by next-event simulation (simulation languages for certain domains)
- Discrete-Event Base Language (DBL)
  - OO language + process-oriented event specification primitives (ESP)
    - ESP: active/passive objects, consumption of model time, blocking wait & reactivate as part of an active object life cycle
  - Execution: DBL —(map-to)—> DBL Core (C++ & Simulation library)
  - DBL Core: novel context switch approach in C++ with low execution time [SpringSim14], close to Assembler
  - Runtime efficiency: important requirement for simulation studies
- Implementation: **DMX** - Discrete-Event Modeling Framework with Extensibility

Approach

- Implemented Parts
  - Textual syntax [SAM Innsbruck]:
    BNF-like Object grammar
  - Execution Semantics [Forum Montréal]:
    Mapping to DBL generates code as text
  - Nested extensions
    - Extension in extension
    - Extension in extension definition
  - Executable DBL to Java/Sim mapping
- Open Parts
  - Technical: executable DBL to C++ DBL Core mapping
  - Conceptual challenges ...



Application to SDL

- Subset SDL_o
  - Definitions of system, process, signal, variable, timer, simple states and transitions (signal, timer, none), tasks output, set/reset timer
  - DBL concepts reused: variable, statement (task), expression (values, timers)
  - Minor issues (details in paper)
- (SDL-RT) Semaphores
  - Semaphore Definition + Take & Give Actions
  - Issue: Concepts cannot be defined modular
    - SDL_o + Semaphores defined in one big extension
    - Actually, Semaphores are an extension of certain concepts of the SDL_o extension
      - Take/Give extend SDL Task
      - Semaphore Definition extends Entity Definition
      - Requires expression means for further extensibility of extensions
      - Syntax is simple, but semantics are difficult

```
system T;

    semaphore SEM, kind=BINARY, policy=FIFO, initial=FULL;

    process ST;
        dcl int i2=0;

        start;
            take SEM with NO_WAIT;
            take SEM with NO_WAIT,
                on OK {
                    task { trace("take SEM OK"); }
                    give SEM;
                },
                on ERROR {
                    task { trace("take SEM ERROR"); }
                    give SEM;
                }
                ;
            take SEM with timeout=10;
            take SEM FOREVER;
            give SEM;
        stop;
    endprocess ST;
endsystem;
```

Application to SDL-RT Semaphores

- Benefits of the extension-based definition
  - Modeling assistance for added concepts
  - Semantics can be defined by using event specification primitives
  - Runtime efficient next-event simulations
- Issue regarding another important application of SDL
  - Code generation to platform-specific concepts,
    e.g. real time operating systems
  - SDL/Semaphores —(map-to)—> DBL —(map-to)—> C++/**Sim**
  - Parallel processes
    —(executed)—> Sequentially as pseudo-parallel processes
  - DBL —(map-to)—> C++/Threads may be feasible

Summary

- Approach for prototyping new language concepts
- Extension-basis allows to reuse general concepts
- Application to SDL subset and SDL-RT Semaphores
- Supports the initial design phase by providing low cost tools
  - Concepts can be directly used in models
  - Evaluate the performance of a system modeled