Domain-Specific Modeling

Tim Schneider

January 19, 2017

Contents

Important Terms

- **Model**
  - formal representation (Abstraction)
  - certain correspondence (homomorphism)
  - purpose (pragmatics)
- **Domain**
  - common knowledge of the requirements, concepts and functionality in a field of study
- **Domain-Specific Modeling Languages**
  - textual or graphical representation of concepts, entities and relationships (only those relvant for the domain)

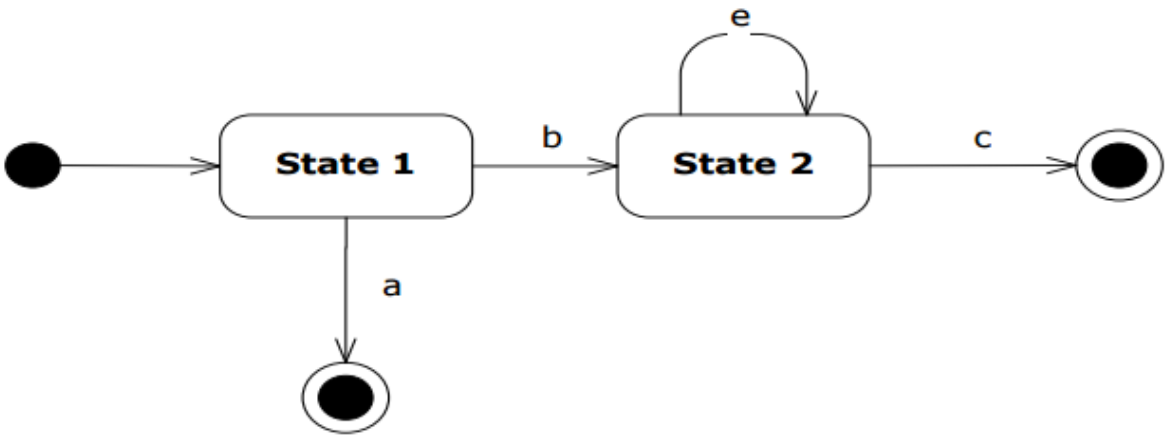
Motivation

- Domain-Specific Modeling Languages in General:
  - support novices and experts in creating models easily
  - learn new ideas and skills in the process

- This Presentation:
  - Focus: domain-specific modeling languages for novices
  - How are novices supported in existing approaches/tools?
  - How can modeling languages created easily?

Domain-Specific Modeling Languages

Graphical



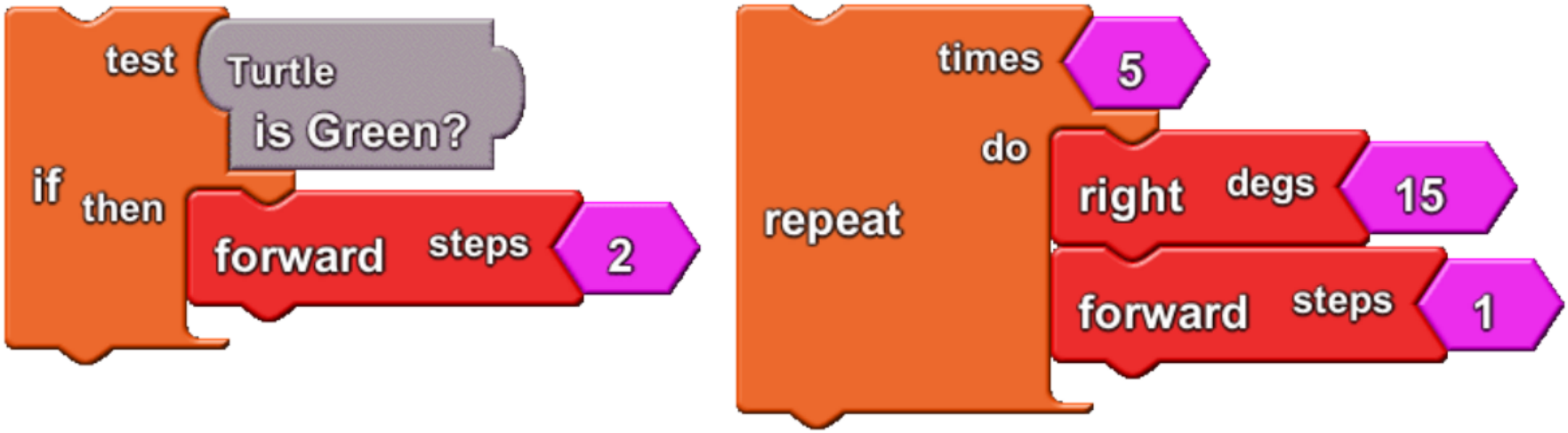
Textual

```
STATES
   State 1, State 2, Start(start), Stop 1(stop), Stop 2(stop)
TRANSITIONS
   Start->State 1, State 1 -b-> State 2, State 2 -e-> State 2,
   State 2 -c-> Stop 1, State 1 -a-> Stop 2
```

Graphical Modeling Languages

StarLogo TNG

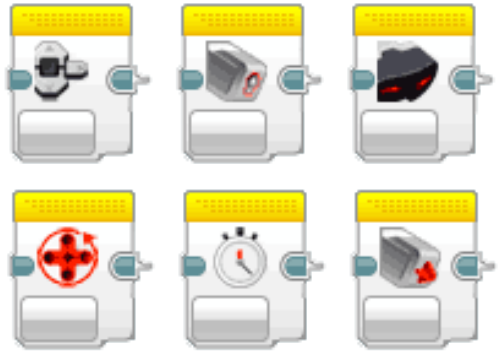
- simulation of complex systems without programming skills
- puzzle-piece blocks:
  shapes only allow syntactically correct constructs
- color based on function



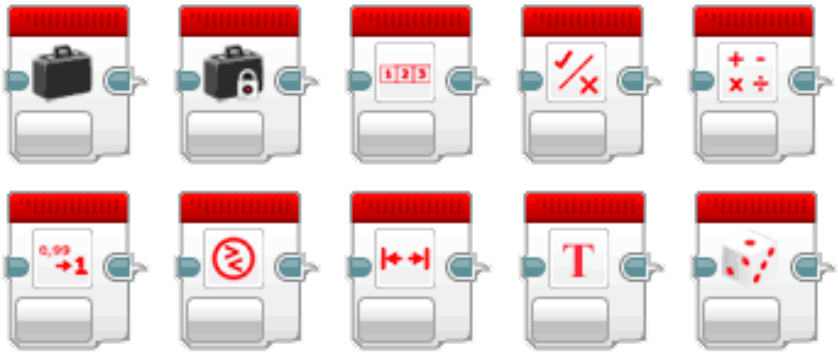
Graphical Modeling Languages

LEGO Mindstorms EV3

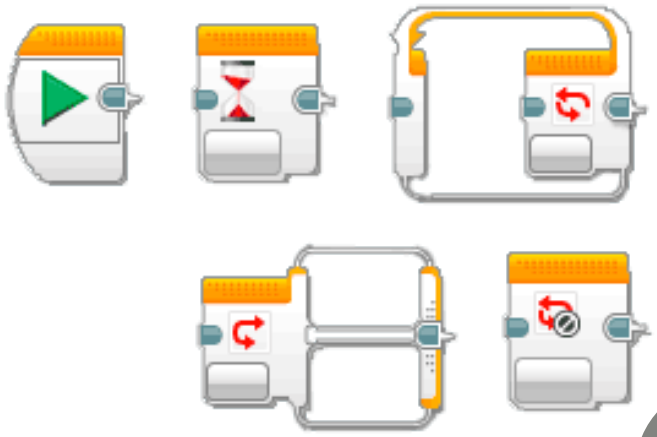
(a) action blocks



(b) sensor blocks



(c) operation blocks



(d) flow blocks

Textual Modeling Languages

PhyDSL

- create models for the game development domain
- fast prototyping of physics-based games
- text editor (syntax highlighting ; text completion)

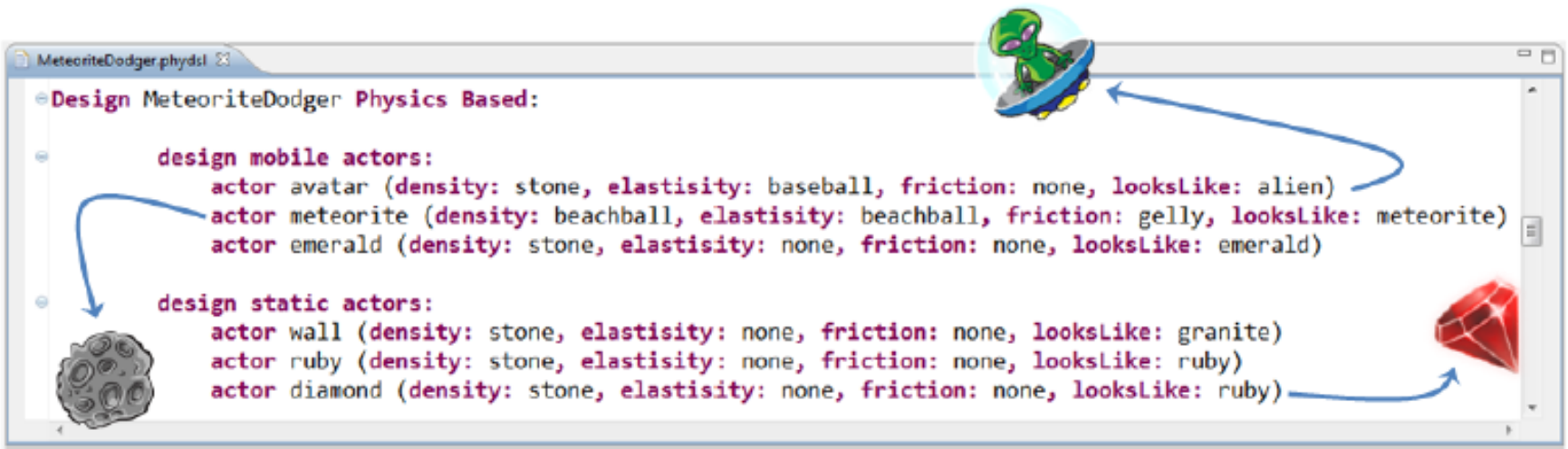


Figure: PhyDSL: Static Actor Definition

Creating Modeling Languages

Xtext

- Grammar Rules (similar to EBNF)
- generates Text-Editor Plugin for Eclipse
- features Syntax-Highlighting; Autocompletion

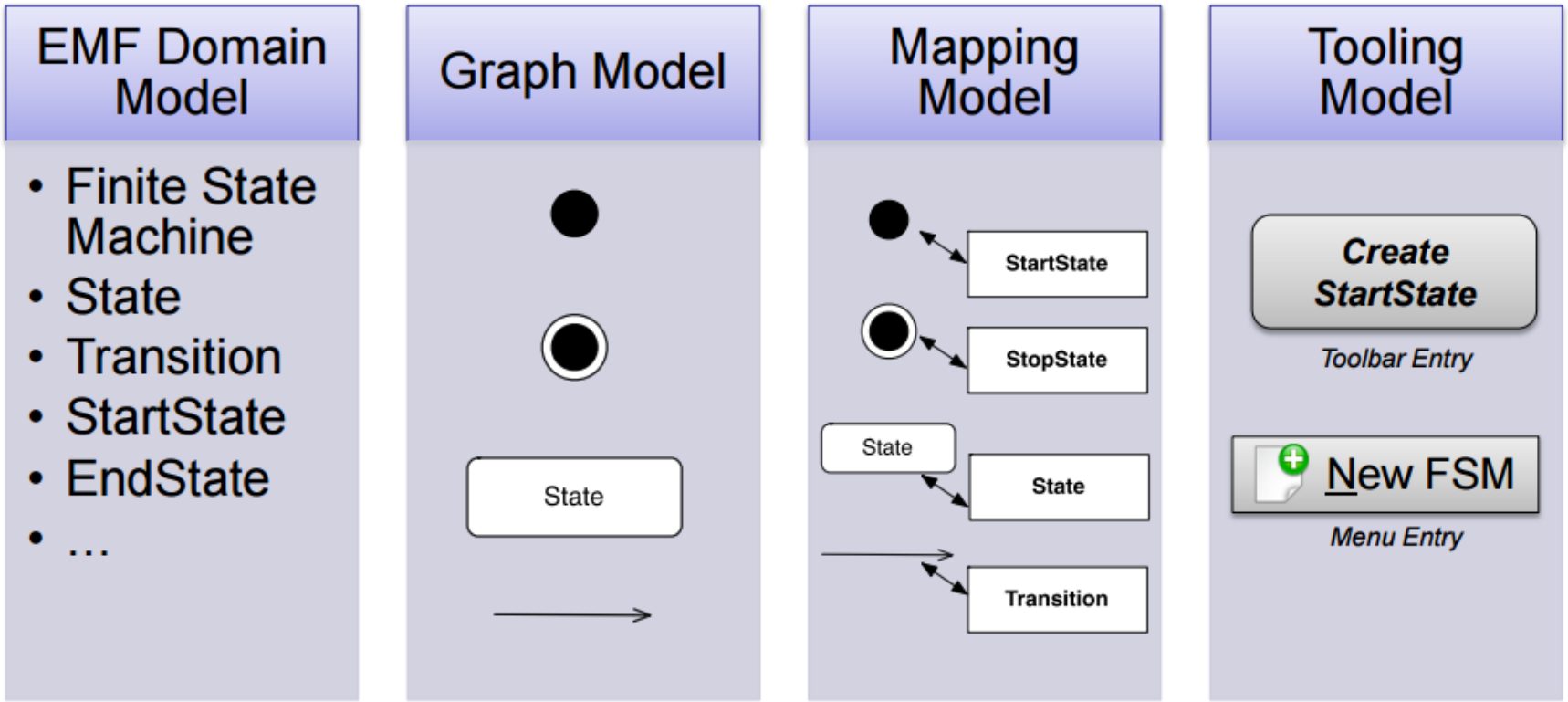
```
FSM: "States:" (states+=State (",")?)*
"Transitions:" (transitions+=Transition (",")?)*;
State: id=ID isStart?="(start)" isStop?="(stop)";
Transition: fromState=[State] "-" (input)? "->"
    toState=[State];
```

Figure: Xtext: Grammar for Modeling Finite State Machines

Creating Modeling Languages

Graphical Modeling Framework (GMF)

- User defines Mapping: Graphical Shapes → Model-Elements
- GMF generates Graphical-Editor Plugin for Eclipse
- features Drag & Drop; Tooling (add/delete Elements via Menus)



Summary

- Support Novices and Experts in Creating Models easily
- Overview ofver some Modeling Languages for Novices
- Textual $\leftrightarrow$ Graphical Modeling Languages
- Example Tools for Creating Modeling Languages (Xtext & GMF)

Questions?