Domain-Specific Modeling

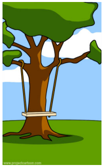
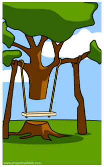
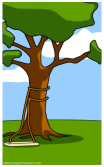
Tim Schneider

10.02.17

Contents

Motivation

Problem:
- error-prone communication between stakeholders
- general purpose languages hard to learn

				
How the domain expert explained it	How the project-leader understood it	How the analyst designed it	How the programmer wrote it	What the user really needed

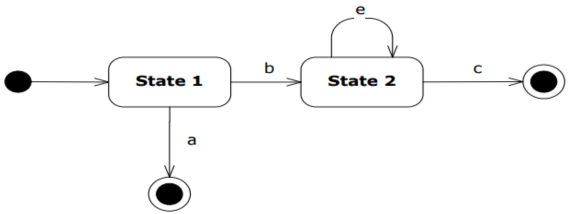
Solution:
- let domain experts create their own “programs”
- use well-understood terms and concepts for representation

Important Terms

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

Domain-Specific Modeling Languages

Graphical



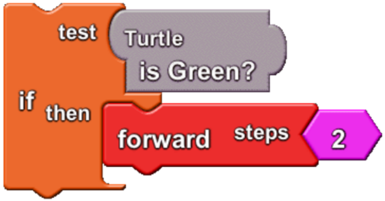
Textual

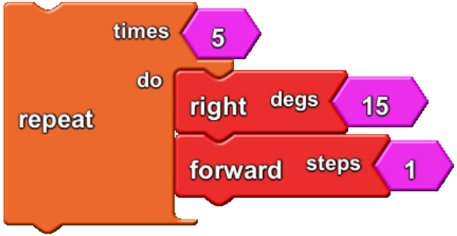
```
STATES
   State 1, State 2, Start(start), Stop 1(stop), Stop 2(stop)
TRANSITIONS
   Start->State 1, State 1 -b-> State 2, State 2 -e-> State 2,
   State 2 -c-> Stop 1, State 1 -a-> Stop 2
```

Graphical Modeling Languages

StarLogo TNG

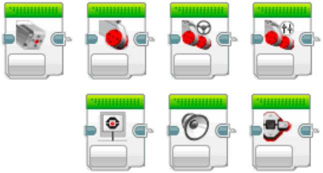
- simulation of complex systems without programming skills
- puzzle-piece blocks:
  shapes only allow syntactically correct constructs
- color based on function



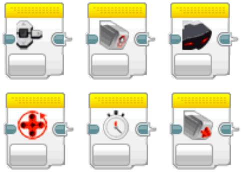


Graphical Modeling Languages

LEGO Mindstorms EV3

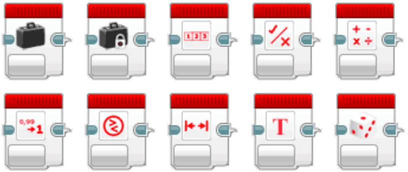


(a) action blocks

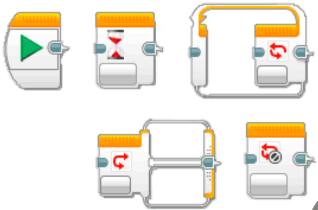


(b) sensor blocks

(c) operation blocks



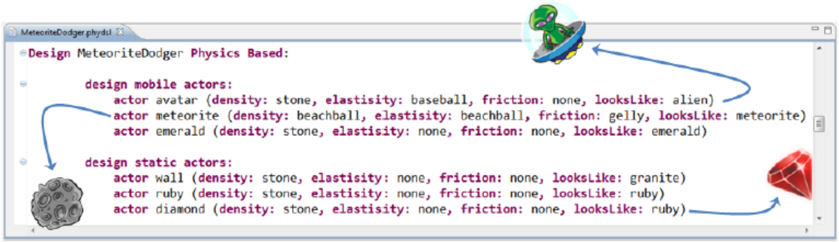
(d) flow blocks



Textual Modeling Languages

PhyDSL

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



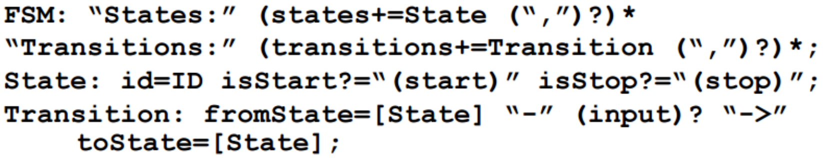
```
Design MeteoriteDodger Physics Based:

        design mobile actors:
            actor avatar (density: stone, elastisity: baseball, friction: none, looksLike: alien)
            actor meteorite (density: beachball, elastisity: beachball, friction: gelly, looksLike: meteorite)
            actor emerald (density: stone, elastisity: none, friction: none, looksLike: emerald)

        design static actors:
            actor wall (density: stone, elastisity: none, friction: none, looksLike: granite)
            actor ruby (density: stone, elastisity: none, friction: none, looksLike: ruby)
            actor diamond (density: stone, elastisity: none, friction: none, looksLike: ruby)
```

Figure: PhyDSL: Static Actor Definition

Creating Modeling Languages

Xtext

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



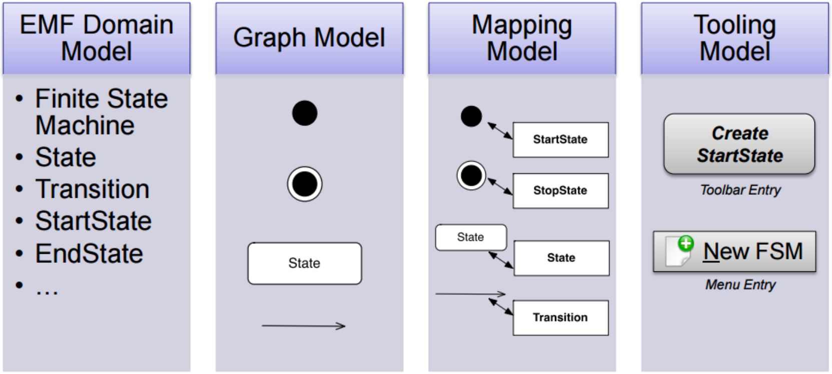
```
FSM: “States:” (states+=State (“,”)?)*
“Transitions:” (transitions+=Transition (“,”)?)*;
State: id=ID isStart?=“(start)” isStop?=“(stop)”;
Transition: fromState=[State] “-” (input)? “->”
    toState=[State];
```

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 Domain Experts in Creating Models easily
- Overview over some Modeling Languages
- Textual $\leftrightarrow$ Graphical Modeling Languages
- Example Tools for Creating Modeling Languages (Xtext & GMF)

Questions?