

Domain Specific Modeling

Tim Schneider
tim-1.schneider@uni-ulm.de

Institute of Databases and Information Systems, Ulm University

Abstract. Nowadays computers and smartphones make it easier for both novices and domain experts to build and explore their own models and learn new scientific ideas in the process. Domain Specific Modeling can support both experts and novices in building models for different domains. There are approaches (e.g. LEGO Mindstorms,Starlogo) which enable novices with few or even no programming skills to implement their own models. Domain Experts on the other hand may use modeling languages (e.g. SimuLink,BPMN) designed for a domain to create models, which allow them to focus on domain-specific problems instead of implementation-specific details like supported language features of a programming language. In this paper we will give an overview over different domain-specific modeling languages with a focus on modeling languages for novices as well as describe features offered by Xtext and GMF (from the Eclipse Modeling framework) supporting the creation of textual and graphical modeling languages.

1 Introduction

For centuries people from da Vinci to Einstein have created models to help them better understand patterns and processes in the world around them. Nowadays computers and smartphones make it easier for both novices and domain experts to build and explore their own models and learn new scientific ideas in the process. Domain Specific Modeling can support both experts and novices in building models for different domains. There are approaches (e.g. LEGO Mindstorms,Starlogo) that enable novices with few or even no programming skills at all to implement their own models. Domain experts on the other hand may use modeling languages (e.g. SimuLink,BPMN) designed for a specific domain to create models, which allows them to focus on domain-specific problems instead of implementation-specific details like supported language features of a general purpose programming language (e.g. C,C++,Java). In this paper we will give an overview about domain specific modeling languages for novices. Additionally we will present the tools Xtext and GMF for creating textual and graphical modeling languages.

1.1 Foundation

Before speaking about domain specific modeling languages we will give some short definitions for the terms *model* and *domain*.

Model When speaking about domain specific modeling languages, we need to define what a model is:

A model is a formal representation of entities and relationships in the real world (abstraction) with a certain correspondence (homomorphism) for a certain purpose (pragmatics) [1].

This definition contains three aspects: abstraction, homomorphism, pragmatics. A model should create a simplified view on the represented entities and relationships and only contain details which are relevant (abstraction). Additionally statements on the model elements should hold for the represented entities and relationships (homomorphism). The question which details are relevant and should be included in the model is determined by the purpose of a model (pragmatics). Creating models is something we do in our everyday lives: When building a house in LEGO a child creates a model which serves as a representation of building in the real world. It is a simplified view on the original as it abstracts from details like used materials, installed cables and water pipes and focusses on the shapes of the building. Another example for creating models is an architect, who creates blueprints for some house, serving as a model for it. Blueprints contain 2D projections of walls and installations in that house from different viewpoints and abstract from the original by omitting details like materials and some 3D information. We can see from this example that these abstraction also brings a representational bias: Electric cable installations and water pipes cannot be modeled in LEGO but they can with blueprints. LEGO models on the other hand allows to model structures directly in 3D while blueprints cannot.

Domain Although novices (e.g. the child) and experts (e.g. the architect) may have different views on the same thing (e.g. the house) they share a common understanding in the concepts of the real world: Both of them know that the concepts “wall” and “roof” are related to parts of a building and that a house needs both to be a valid house. This common knowledge of the requirements, concepts and functionality in a field of study is referred to as a domain (e.g. architecture).

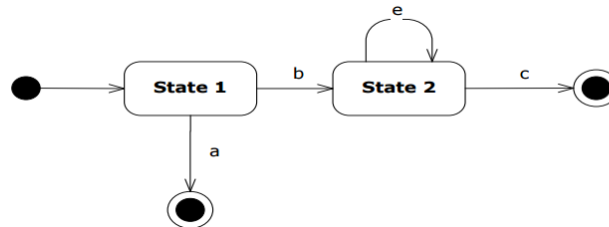
1.2 Domain Specific Modeling Languages

Domain specific languages are modeling languages, designed to create models representing entities and relationships of a certain domain, containing textual or graphical representations of relevant concepts from that domain. The purpose of a domain specific modeling language is to support users in building models from a specific domain.

There are two kinds of modeling languages, which are used in these tools to create models: *Graphical Modeling Languages* use graphical shapes and *Textual Modeling Languages* use text to represent entities and relationships from the real world. The same model may be expressed in both, a graphical or textual modeling language.

A finite state machine for example can be represented as a set of nodes, labeled boxes and arrows, while it can also be represented by a textual description following a grammar to represent the same finite state machine.

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

```

Fig. 1: comparison between textual modeling language and graphical modeling language for the same model

Users have different levels of expertise in a domain: a Software-Engineer has a more detailed knowledge in the computer science domain as a medical doctor. We will therefore distinguish between domain experts and novices and focus on domain specific languages for novices in this paper.

2 Domain Specific Modeling Languages for Novices

As there is a variety of software tools available, which implement different modeling languages, this section will give an overview about modeling languages designed to support novices with few or even no programming languages in creating models and help them better understand patterns and processes in the world around them.

2.1 StarLogo TNG

StarLogo is a client based modeling software, developed specifically to enable students to create models for the simulation of complex systems without extensive programming skills [2]. It provides a graphical modeling language using blocks instead of a textual syntax. These blocks are coloured based on the function they have in the program and their shapes only allow syntactically correct

constructs [2]. This eases the learning of basic programming concepts such as control-flow (e.g. loops, if-then statements), assignments and methods by providing intuitive graphical mappings. Since StarLogo TNG uses these puzzle-piece shapes, understanding control-flow, for example, “requires little more than visually parsing the if or repeat commands Figure 2 to conclude that items placed within the then slot will be performed if the items placed within the test slot are true, or the items placed within the do slot will be subject to the number of repetitions placed within the times slot” [3]. Using puzzle-piece shapes for basic programming concept avoids syntactic errors, which are one of the main difficulties in learning textual programming languages. Figure 2 shows an example for the blocks used by StarLogo TNG : the curved shape of a Boolean variable is easy to distinguish from the pointed shape of a number value.

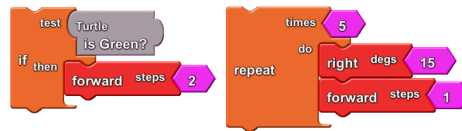


Fig. 2: StarLogo TNGs graphical programming blocks. Example if (left) and repeat (right) blocks are shown. The if block commands a Turtle agent to take two steps forward if it is green; the repeat block commands an agent to repeat five times the sequence of turning right 15 degrees and taking one step forward. (from [3])

2.2 Snatch

- Graphical Modeling Language (similar to StarLogo)
- developers goal: “make it easy for everyone, of all ages, backgrounds, and interests, to program their own interactive stories, games, animations, and simulations, and share their creations with one another”

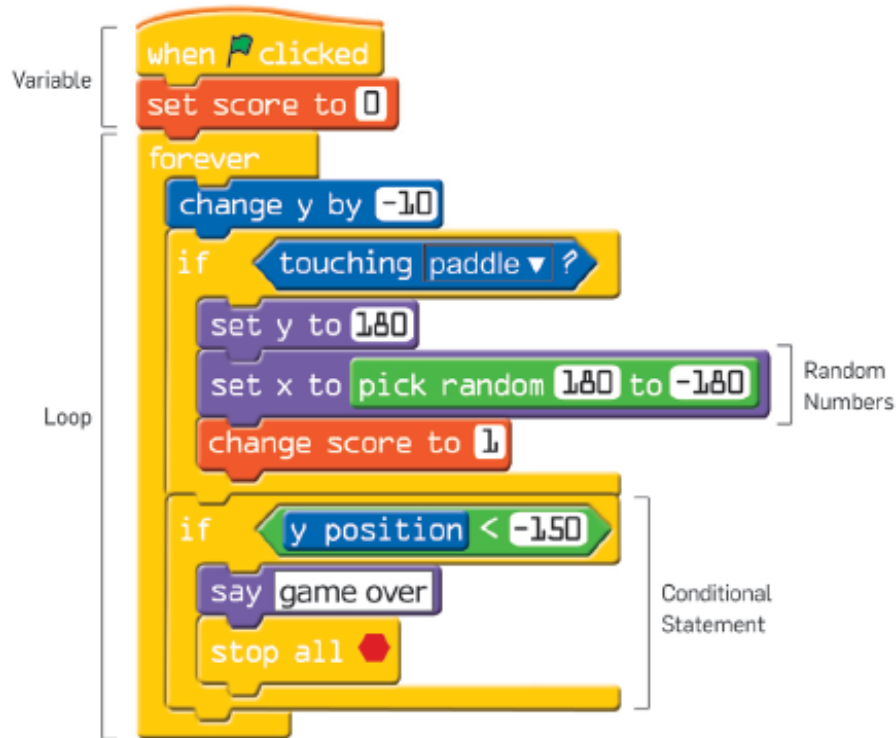


Fig. 3: Sample Scratch script (from Pong-like paddle game) highlighting computational and mathematical concepts

2.3 PhyDSL

Both StarLogo and Snatch use graphical modeling languages for creating models. *PhyDSL* is a tool, designed as an Eclipse[?] plugin that uses a textual modeling language for creating models for the game development domain. It was designed for the purpose of supporting the fast prototyping of physics-based games, including platform, shoot em up, puzzle and maze games [4]. The provided frontend includes a text editor (including syntax highlighting and text completion), which is used to define gameplay in highlevel terms describing the gameplay static, dynamic elements and their behaviors by a domain specific language. It is based on the *Eclipse Modeling Framework* [EMF] and uses codegeneration for deriving C# code supported by Microsofts DirectX API. The modeling language provided by PhyDSL consists of four gameplay definition sections: mobile and static actor definition (Fig. 4), environment and layout definition (Fig. 5), activities definition (Fig. 6), and scoring rules definition (Fig. 7).

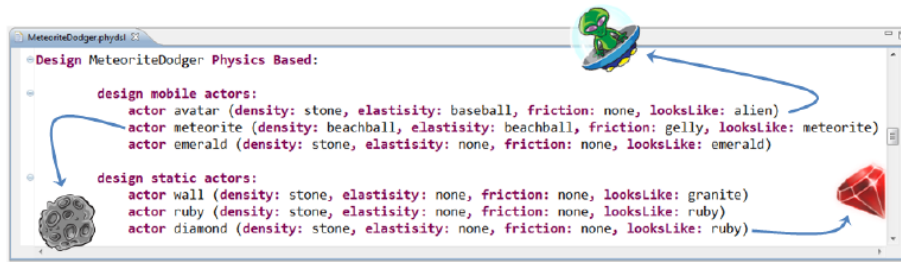


Fig. 4: PhyDSL: static actor definition (from [4])

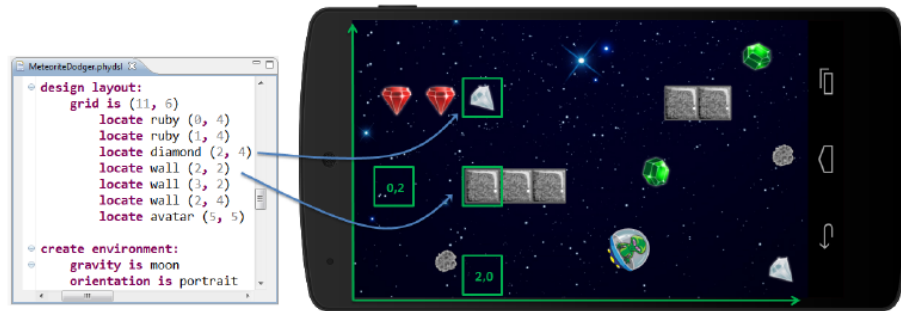


Fig. 5: PhyDSL: environment and layout definition(from [4])

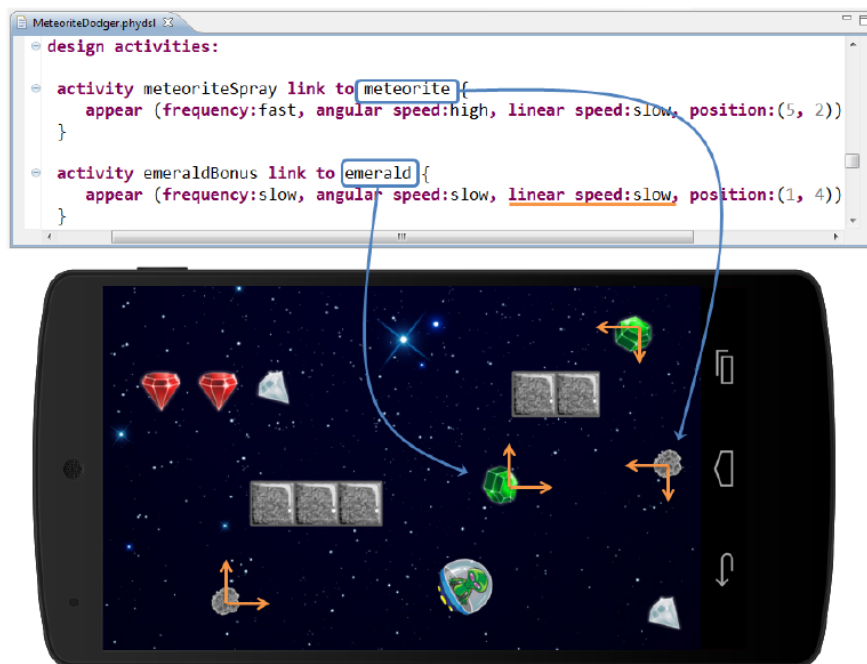


Fig. 6: PhyDSL: activities definition(from [4])

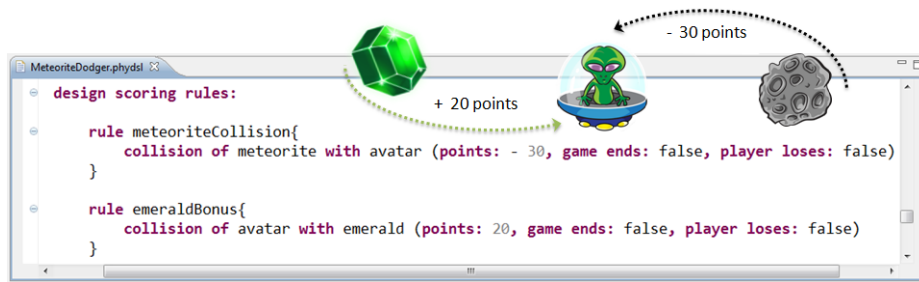


Fig. 7: PhyDSL - Scoring Rule Definition for Collision Rules (from [4])

2.4 LEGO Mindstorms

Another example for a graphical modeling language made for novices is the EV3 Programmer App for LEGO Mindstorms which includes a block-based graphical modeling language for creating programs for the LEGO MindStorm Robots. Programs are loaded and executed on a EV3 P-brick 11, which offers input and output connections for interaction with sensors and actuators. Different colors are used to indicate whether a block triggers some action (green), is used for flow control (orange), reads data from some input (e.g. color sensor) (yellow) or triggers some data operation (e.g. read/write some variable).



Fig. 8: The action blocks control the actions of the program, e.g. motor rotations, image, sound and the light on the EV3 P-brick.

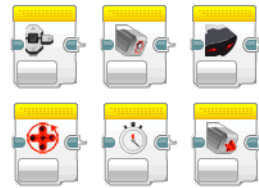


Fig. 9: The sensor blocks allow to read the inputs e.g. from a color sensor, IR sensor, touch sensor.

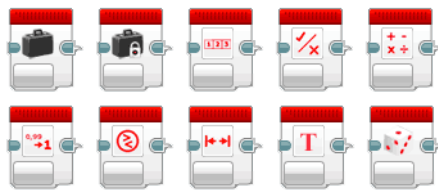


Fig. 10: The data operation blocks let the user write and read variables, compare values for example.

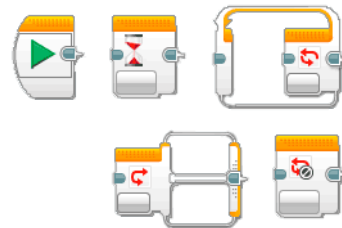


Fig. 11: The Flow blocks control the flow of the program.

2.5 Sensor

- enables people without programming skills to build mobile data collection and management tools for citizen science

3 Creating Domain Specific Modeling Languages

Domain Specific Modeling Languages are implemented in various software tools using textual or graphical modeling languages. In fact, there are approaches to



Fig.12: EV3 Programmer App on a tablet used to create programmes using a graphical modeling language

define these modeling languages as models using some kind of meta-language. Xtext and GMF are implementations from the Eclipse Modeling Framework (EMF), which adopt this approach.

3.1 Ecore

Xtext and GMF use a

- a meta model for describing models and runtime support for the models .

3.2 Xtext

Xtext is a framework for development of programming languages and domain specific modeling languages [5]. It uses a syntax similar to extended backusnaur form (EBNF) to describe custom textual modeling languages. From this modeling language definition Xtext generates a custom Text-Editor for Eclipse whose features include code-completion and syntax-highlighting as well as a parser for parsing models from text files specified in the defined grammar. This makes it easy for experts of the language design domain to create models for custom modeling languages without programming skills or knowledge on how to implement a parser. As the modeling language definitions are stored in the Ecore format, other tools in the EMF framework can process data from the parsed models (e.g. Acceleo[?] for performing codegeneration).

```

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

```

Fig. 13: Xtext grammar of a textual modeling language for finite state machines. Figure 1 contains an textual description of a state machine which follows this grammar definition.

3.3 GMF

- used to create graphical DSLs for models described in Ecore
- connects domain model and a graphical model via mapping modell

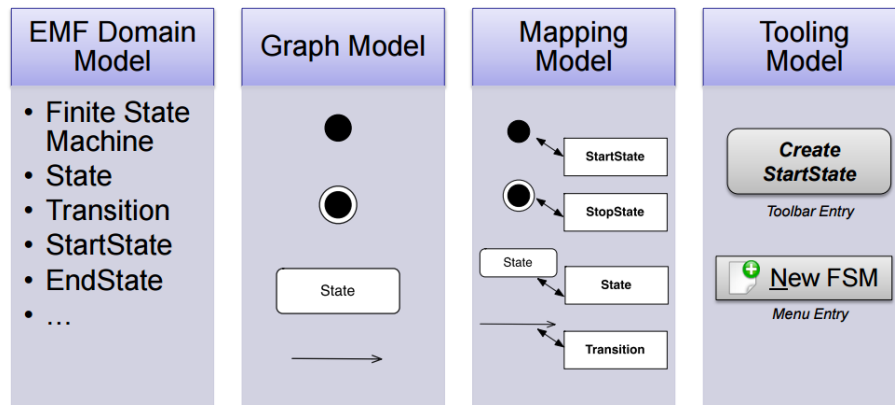


Fig. 14

4 Summary

- DSM allows novices to build and explore their own models and learn new scientific ideas in the process
- domain experts are supported by setting focus on domain specific problems

References

1. Stachowiak, H.: Allgemeine modelltheorie. (1973)

2. Klopfer, E., Scheintaub, H., Huang, W., Wendel, D.: Starlogo tng: Making agent based modeling accessible and appealing to novices in artificial life models in software. (2009)
3. Smith, V.A., Duncan, I.: Biology students building computer simulations using starlogo tng. *Bioscience Education* **18**(1) (2011) 1–9
4. Guana, V., Stroulia, E.: Phydsl: A code-generation environment for 2d physics-based games. In: 2014 IEEE Games, Entertainment, and Media Conference (IEEE GEM). (2014)
5. Eysholdt, M., Behrens, H.: Xtext: implement your language faster than the quick and dirty way. In: *Proceedings of the ACM international conference companion on Object oriented programming systems languages and applications companion*, ACM (2010) 307–309
6. OMG: Business Process Management Notation (BPMN) 2.0 (21.05.2010)
7. Group, L.: Lego mindstorms programming (1999)
8. Amyot, D., Farah, H., Roy, J.F.: Evaluation of development tools for domain-specific modeling languages. In: *International Workshop on System Analysis and Modeling*, Springer (2006) 183–197
9. Kim, S., Mankoff, J., Paulos, E.: Sensr: evaluating a flexible framework for authoring mobile data-collection tools for citizen science. In: *Proceedings of the 2013 conference on Computer supported cooperative work*, ACM (2013) 1453–1462
10. Gronback, R.C.: Eclipse modeling project: a domain-specific language (DSL) toolkit. Pearson Education (2009)
11. France, R., Rumpe, B.: Domain specific modeling. *Software and Systems Modeling* **4**(1) (2005) 1–3
12. Meliones, A., Giannakis, D.: Visual programming of an interactive smart home application using labview. In: 2013 11th IEEE International Conference on Industrial Informatics (INDIN), IEEE (2013) 655–660
- 13.
14. Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B., et al.: Scratch: programming for all. *Communications of the ACM* **52**(11) (2009) 60–67