2

009 IEEE/ACM International Conference on Automated Software Engineering

Generation of Simulation Views for Domain Speciﬁc Modeling Languages based on

the Eclipse Modeling Framework

Enrico Biermann, Karsten Ehrig, Claudia Ermel, Jonas Hurrelmann

Institut fu¨r Softwaretechnik und Theoretische Informatik

Technische Universita¨t Berlin, Germany

E-mail: tigerprj@tfs.cs.tu-berlin.de

Abstract—The generation of tools for domain speciﬁc mod-

eling languages (DSMLs) is a key issue in model-driven

development. Various tools already support the generation

of domain-speciﬁc visual editors from models, but tool gen-

eration for visual behavior modeling languages is not yet

supported in a satisfactory way. In this paper we propose

a generic approach to specify DSML environments visually

by models and transformation rules based on the Eclipse

Modeling Framework (EMF). Editing rules deﬁne the behavior

of generated visual editors, whereas simulation rules describe a

model’s operational semantics. From a DSML deﬁnition (model

and transformation rules), an Eclipse plug-in is generated,

implementing a visual DSML environment including an editor

and (possibly multiple) simulators for different simulation

views on the model. We present the basic components of

Tiger2, our EMF-based generation environment, along the

environment generation process for a small DSML modeling

the behavior of ants in an ant hill.

propose a generic approach to specify behavior-modeling en-

vironments by EMF models and EMF model transformation

based on graph transformation [9].

In our modeling environment, a set of EMF transformation

rules called editing rules deﬁne the editing commands of the

generated visual editor, i.e. the model syntax; on the other

hand, a set of simulation rules describe a model’s operational

semantics. For automatic simulation, rule application is con-

trolled by activity diagrams. From a DSML deﬁnition (EMF

model, view deﬁnitions and EMF transformation rules), an

ECLIPSE plug-in is generated, implementing a visual DSML

environment including an editor and (possibly multiple)

simulation views on the model.

After reviewing graph and EMF transformation concepts

in Sect. II, we introduce our running example in Sect. III.

Sect. IV deﬁnes the EMF models used for DSML speci-

ﬁcation, and Sect. V presents TIGER2 [10], our generation

environment based on EMF and EMF model transformation.

Keywords-EMF, EMF transformation, visual environment

generation, graph transformation, simulation;

I. INTRODUCTION

II. EMF TRANSFORMATION BASED ON GRAPH

TRANSFORMATION CONCEPTS

Domain speciﬁc modeling languages (DSML) are of

growing importance for software engineering, and the rapid

development of DSML tools is a key issue in model-

driven development. Meta-tools including MetaEdit+ [1],

DiaGen [2], AToM3 [3], GME [4], Marama [5] and DSL

A domain-speciﬁc visual language (VL) is modeled by a

type graph deﬁning its symbols (node types) and relations

(edge types). Sentences or diagrams of the VL are given by

graphs typed over (i.e. conforming to) the type graph. Such

a VL type graph (which may also contain multiplicities and

inheritance arcs) corresponds closely to a meta-model. Node

types may be attributed by attribute types.

On the basis of a type graph deﬁning a VL, and instance

graphs typed over this type graph representing different

states of a model, step-wise simulation is now described by

graph transformation between these states. The main idea

of graph transformation is the rule-based modiﬁcation of

graphs where each application of a graph transformation

Tools [6] have been developed to support rapid speciﬁcation

and generation of DSML tools. Moreover, the Eclipse Mod-

eling Framework EMF [7] has recently come to be a quasi-

standard for meta-modeling in practice, consisting of an

implementation of core concepts based on MOF. The above

mentioned meta-tools do not yet take the EMF format for

meta-modeling into account, which poses difﬁculties when

existing EMF models serve as basis for tool generation.

A notable exception is the EMF-based editor generator

GMF [8] which is widely used in ECLIPSE projects on

model-driven software development. In contrast to editor

generation, the generation of tools for visual behavior mod-

eling languages based on EMF is not yet supported in a

satisfactory way. Visual behavior models are the basis for

model simulation with the purpose to validate the model

behavior with respect to its requirements. In this paper we

rule leads to a graph transformation step. The core of a

r RHS)

is a pair of graphs

graph transformation rule (LHS →

(LHS, RHS), called left-hand side and right-hand side, and

an injective (partial) graph morphism r : LHS → RHS. A

graph morphism consists of structure-preserving mappings

from nodes in LHS to nodes in RHS, such that for an edge

from node n to node n in LHS which is preserved by

1

2

6122375



Table I

MAPPING EMF NOTIONS TO GRAPH TERMINOLOGY

the rule, we have a corresponding edge from node r(n ) to

r(n ) in RHS. In our approach, all graph morphisms are

2

1

injective, i.e. they do not merge elements. Applying the rule

(LHS →r RHS) means to ﬁnd a match of LHS in the source

graph and to replace this matched part in the source graph

by the corresponding RHS, thus transforming the source

EMF notion

Model

Graph term

Type graph with attribution, inheritance, multi-

plicities. Edges can be marked as containments.

Model instance Typed, attributed graph with containment edges

graph into the target graph. Intuitively, the application of rule

Class

Node in type graph

Node in typed graph

r to graph G via a match (LHS →

m G)

deletes the image

Object

m(LHS) from G and replaces it by a copy of the right-

Association

Edge in type graph (with possible multiplicities

or containment mark)

hand side m∗(RHS), leading to a graph transformation step

r,m

Reference

Edge in typed graph that must not violate certain

multiplicity and containment constraints.

G =⇒ H from G to graph H. Note that a rule may only be

applied if the so-called gluing condition is satisﬁed, i.e. the

deletion step must not leave dangling edges [9]. A rule r

may be extended by a set of negative application conditions

NACs) [9]. A match LHS →

m G

satisﬁes a NAC with the

n NAC

, if there is no graph

ship relation between objects. Thereby, they induce a tree

structure in model instantiations. Containment implies a few

constraints for model instantiations that must be ensured at

run-time. As semantical constraints for containment edges,

the MOF speciﬁcation states the following:

injective NAC morphism LHS →

q

morphism NAC → G with q ◦ n = m.

An example of a rule application is shown in Fig. 1, where

mappings are indicated by corresponding numbers in the

graphs. The LHS matches its Field node to Field node 2 in

G. Hence, the node of type Ant in H, which is created by

the rule, is linked to Field 2 and to node AntWorld.

•

•

”An object may have at most one container.”

”Cyclic containment is invalid.”

EMF provides implementations of instance models which al-

ways ensure these consistency constraints for containments.

In [13], containment constraints of EMF model transfor-

mations are translated to a special kind of graph transforma-

tion rules (called EMF transformation rules from now on)

such that their application leads to consistent transformation

results only. In this paper we will use EMF transformation

rules only and thus can be sure that we always have valid

EMF instance models. Note that EMF transformation rule

applications change an EMF model instance in-place, i.e.

the model instance is modiﬁed directly.

III. RUNNING EXAMPLE: ANT WORLD

Figure 1. Rule application example

The AntWorld simulation was used as case study at the

Graph-Based Tools Workshop GraBaTs 2008 [14]. For the

complete case study, please refer to [15]. Ants are moving

around searching for food. If an ant ﬁnds food, it returns it to

its ant hill in order to grow new ants. On its way home, the

ant drops pheromones marking the path to the food reservoir.

If a searching ant hits a pheromone mark, it follows the

pheromone path to the food. The area in which the ants

move is modeled by a grid of nodes. The area grid looks

like a spider’s web with the ant hill in its center, see Fig. 2.

The AntWorld simulation works in rounds. Within each

round, each ant performs one of the following actions:

The Eclipse Modeling Framework EMF [7] is a modeling

and code generation facility for building tools based on a

structured data model. For a model described by a class

diagram, EMF provides tools and runtime support to produce

a set of Java classes and supports interoperability with other

EMF-based tools, e.g. OCL checkers.

Although EMF provides basic operations for modify-

ing EMF-based models, it is still difﬁcult to deﬁne more

complex operations on these models. Our approach uses a

recently developed ECLIPSE plug-in [11], [12] supporting

modeling and code generation for EMF model transforma-

tions, based on structured data models and graph transforma-

tion concepts. The conceptual differences between modeling

based on typed, attributed graphs and object-based modeling

as performed by EMF are shown in Table I.

• If the ant has no food and is on a food ﬁeld, it takes

one piece of food. It may still move within this round.

• If the ant carries some food, it follows the links towards

the ’inner’ circle. On its way, the ant drops pheromones

to guide other ants to the food.

Usually, EMF models have containment constraints in

addition, which do not occur in plain graph transformation.

Containment relations, i.e. aggregations, deﬁne an owner-

• If an ant with food is on the hill node, it drops the food

and may leave the hill within the same round.

6

122486

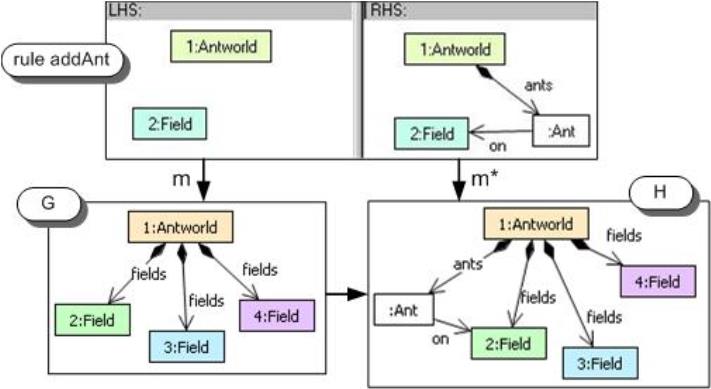




Fig. 4 shows the activity diagrams and rules for food

handling and ant creation. If an ant comes to a food ﬁeld, it

picks up one part of food (rule pickFood). On the hill, an ant

drops the food (rule dropFood) and creates a new ant using

rule createAnt. (See [15] for the complete set of AntWorld

activity diagrams and rules.)

Figure 2. AntWorld example

•

An ant without food checks the neighbor node(s) of

the next outer circle for pheromones. If there are nodes

with pheromones, the ant chooses one of these ﬁelds

randomly. If not, the ant moves to any neighbor ﬁeld

randomly. An ant without food shall not enter the hill.

Initially, the area grid consists only of the hill with eight

ants and the ﬁrst two circles without food (see Fig. 2).

Whenever an ant enters the outmost circle (i.e. the border of

the yet known area), a new circle of nodes is created. During

the creation of this next circle, every 10th node gets 100

parts of food. After each round, the pheromones evaporate,

and the hill consumes the food brought to it and creates one

new ant per delivered food part.

Fig. 3 shows the AntWorld EMF model. All objects are

contained in class AntWorld. We model Ants, Fields, Food

and Pheromones as classes. Fields are further divided into

their speciﬁc roles like Hill, Normal and Exit ﬁelds which

form the main four axis in the grid starting from the Hill.

Figure 4. Food handling and ant creation

IV. MODELING DSML ENVIRONMENTS

In this section, we describe the EMF models underlying

our approach to DSML speciﬁcation and tool generation. As

demonstrated in the last section, we aim at a VL speciﬁcation

(Fig. 5) based on EMF, consisting of the parts

•

Language: An EMF model for the VL, EMF transfor-

mation rules for editing operations and for simulation

steps, and a set of activity diagrams specifying the

application of simulation rules.

•

•

Visualization: Mappings from EMF model elements to

GEF ﬁgures and connections which realize their visual

appearance in the generated environment.

Views: Mappings from EMF model elements to views

for different visualizations.

Figure 3. EMF Model for the Ant World VL

For building the start system for our simulation shown in

Fig. 2, we provide some basic editing rules to draw the grid

and some ants (see e.g. rule addAnt in Fig. 1).

For simulation, we have rule sets for the tasks ant move-

ment (searching or carrying food), for AntWorld management

Moreover, a VLSpec also contains Session information,

used to store the session in the same model instance (ﬁle).

Fig. 6 shows the Language model. We deﬁne the syntax

by an alphabet (the EMF model or EPackage) and editing

rules, and give the semantics in form of operational simu-

lation rules, controlled by activity diagrams. The language

does not hold the EPackage as containment but as associ-

ation and thus the EMF model may be deﬁned externally.

For our AntWorld example, Fig. 3 is showing the EMF

model representing the AntWorld VL’s alphabet.

(

expanding the outer circle, decay pheromones), for handling

food (picking up food or dropping it on the hill) and for

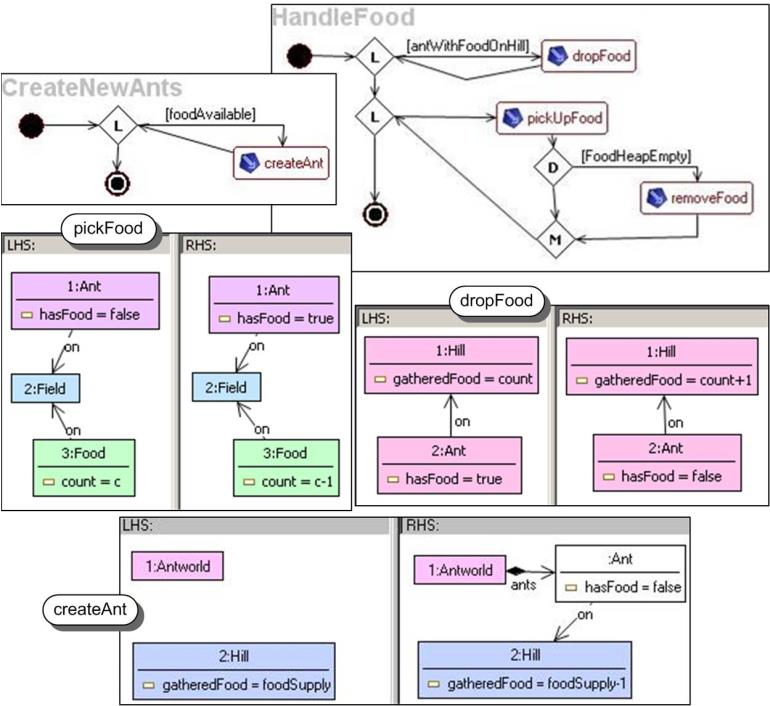
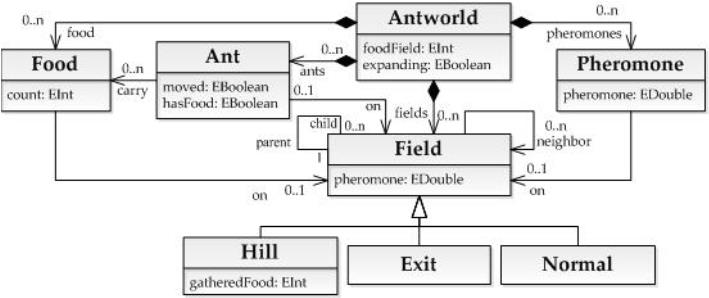
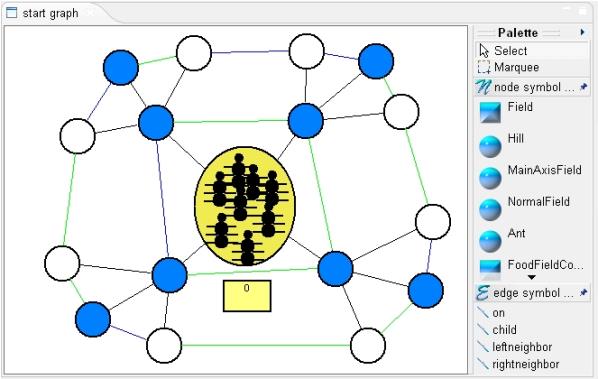
creating new ants. For the rule control ﬂow of simulation

rules, we deﬁne activity diagrams for the different tasks,

where each named activity corresponds to a rule application.

6

122597



is found there as an extension of P.

Figure 5. Core model of a VL speciﬁcation

Figure 8. Simulation deﬁned by activity diagrams

Figure 6. Language model

In order to achieve the visualization of the language

elements, different kinds of shapes can be deﬁned. Figures

are used to represent Nodes in a Graph and Connections

are used to visualize Edges. They directly relate to their

counterparts in DRAW2D, the graphic toolkit of the ECLIPSE

Graphical Editing Framework GEF [17].

Besides the alphabet, the Language contains the Rule

and Graph deﬁnition (see Fig. 7). A Graph consists of

For view deﬁnition, mappings from symbols to visual

representations are deﬁned. Each View provides a distinct

mapping of the model elements, enabling different visual

representations of the same model within a single editor.

For our AntWorld view, model element Ant is mapped to

an (invisible) Rectangle ﬁgure containing three Circles (the

body parts) and three Polylines (the six legs), see Fig. 2.

V. THE GENERATION ENVIRONMENT

A VL speciﬁcation based on an EMF model in com-

bination with a rule-based speciﬁcation of editor com-

mands and simulation behavior is used in our generation

environment TIGER 2 (Transformation-based Generation of

Environments) to generate a corresponding visual modeling

environment for the speciﬁed visual DSML. Whereas our

previous TIGER tool [10] was an editor generator only, in

TIGER 2 also controlled units of simulation rules can be

speciﬁed, and automatic simulation can be visualized in

different views at the same time. The architecture of TIGER

2 consists of the three components Designer (where the

modeler deﬁnes the VLSpec), Generator (translating a VL-

Spec to Java code) and the Generated Tool Environment (an

ECLIPSE plug-in containing a visual editor and simulation

views for the speciﬁed VL).

Figure 7. Rules and Graphs of a Language

Nodes and Edges, which in turn have a type mapping to the

elements of the alphabet. Each Node has a set of incoming

and outgoing Edges to form the graph structure. Each Node

has to be assigned to a corresponding EClass and each Edge

has to be assigned to a corresponding EReference of the

EMF model. Additionally, a start graph for simulation may

be deﬁned. Each Rule consists of an LHS, an RHS and a

number of NACs which are all Graphs. Mappings are used

to deﬁne graph morphisms. As AntWorld start graph serves

the graph shown in Fig. 2 (in concrete syntax). Sample rules

for the AntWorld have been presented in Sect. III.

For rule application control, activity diagrams are used

[

16]. The main idea is to reﬁne a SimpleActivity by a rule

In the current state, the Designer is mainly tree-based,

except for the deﬁnition of rules and activity diagrams,

where visual editors exist (see the screenshots in Sect. III).

The Generator translates EMF models directly to Java

code, using the Java Emitter Templates. The generated code

can be seen as a run-time data model of the structure

deﬁned in the class diagram, i.e. the Java classes contain all

attributes and references. The generated code manages the

that is applied when the activity is executed. Activities are

linked by Next edges (see Fig. 8). To control the ﬂow,

DecisionActivities and LoopActivities with graph constraints

at their outgoing Next edges are used. A step in the activity

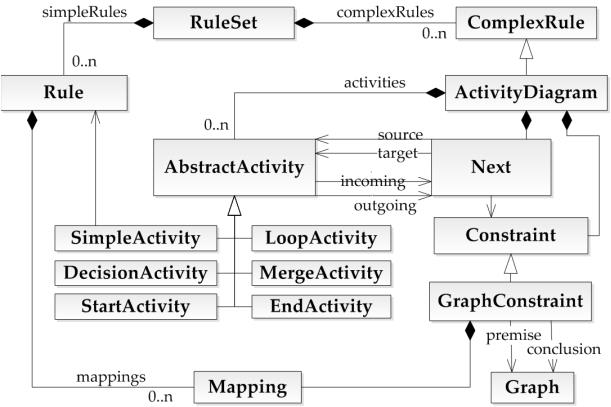
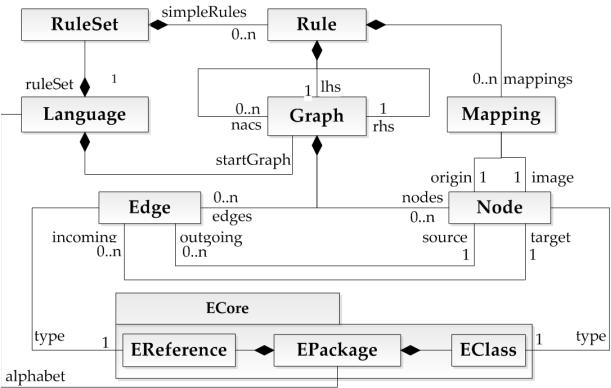
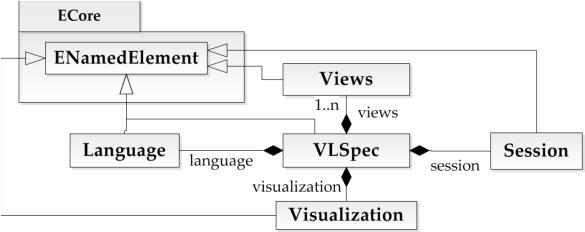
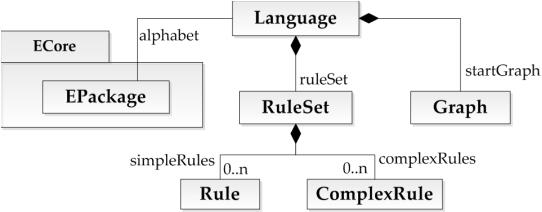
diagram can only be performed if the corresponding graph

constraint P → C is fulﬁlled, i.e. if in the case that premise

P is found in the current graph, then also the conclusion C

6

132608



life cycle of objects (create, delete, set attributes etc.), while

ensuring multiplicity and containment constraints. Further,

a persistence API is provided, implementing load / save

operations for model instances.

REFERENCES

[

1] J. Tolvanen and M. Rossi, “MetaEdit+: Deﬁning and Using

Domain-Speciﬁc Modeling Languages and Code Generators,”

in OOPSLA. ACM Press, 2003, pp. 92–93.

The transformation rules modeled using the TIGER De-

signer are copied to the generated editor project as an XMI

ﬁle. The structure of the XMI ﬁle is similar to that shown

in Fig. 7. The execution of the rules is handled by the EMF

TIGER transformation engine [11].

[

2] M. Minas, DiaGen / DiaMeta – The Diagram Editor Gener-

ator, 2007, http://www.unibw.de/inf2/DiaGen/.

[

3] J. de Lara and H. Vangheluwe, AToM3 – A Tool for Multi-

formalism and Meta-Modelling, 2008, http://msdl.cs.mcgill.

ca/projects/projects/AToM3/.

Activity diagrams deﬁning the control ﬂow of rule ap-

plications are translated to transformation units. Each unit

represents a different kind of control ﬂow. For example a

sequential unit will execute each subunit once in a given

order and is equivalent to a number of sequential activities

in an activity diagram. A decision-merge construct is repre-

sented by a conditional unit. Different units can be nested in

any way with the innermost units representing single rules.

The overall layout of the views is handled via an ECLIPSE

perspective. The controlling class sets up the default position

of the editor and the views within the whole workspace.

We use our GEF-based framework MUVITOR (Multi-View-

Editor) [18], generalizing recurring code fragments for many

editor features. It supports nested editor modes with multiple

graphical viewers and animated simulation of behavior. The

architecture is designed in a way that encapsulates complex

underlying chains of commands in GEF and simpliﬁes the

interaction with the Eclipse workbench.

[

[

4] A. Ledeczi, A. Bakay, M. Maroti, P. Volgyesi, G. Nordstrom,

J. Sprinkle, and G. Karsai, “Composing domain-speciﬁc de-

sign environments,” IEEE Computer, pp. 44–51, 2001.

5] J. Grundy, J. Hosking, J. Huh, and K. Na-Liu Li, “Marama:

An Exclipse Meta-toolset for Generating Multi-view Environ-

ments,” in ICSE. ACM Press, 2008, pp. 819–822.

[6] Microsoft, “Domain speciﬁc language tools,” http://msdn.

microsoft.com/vstudio/DSLTools/.

[7] Eclipse Modeling Framework (EMF) – Version 2.4, Eclipse

Consortium, 2008, http://www.eclipse.org/emf.

[

[

8] Eclipse Graphical Modeling Framework (GMF), Eclipse Con-

sortium, 2007, http://www.eclipse.org/gmf.

9] H. Ehrig, K. Ehrig, U. Prange, and G. Taentzer, Fundamentals

of Algebraic Graph Transformation, ser. EATCS Monographs

in Theor. Comp. Science. Springer, 2006.

[

10] Tiger: Generating Visual Environments in Eclipse, TU Berlin,

2

005, http://www.tfs.cs.tu-berlin.de/tigerprj.

Fig. 9 shows a sample view of the Generated Environment

for AntWorld simulation.

[11] EMF Tiger: A Model Transformation Framework for EMF,

TU Berlin, 2009, http://tfs.cs.tu-berlin.de/emftrans.

[

12] E. Biermann, K. Ehrig, C. Ko¨hler, G. Kuhns, G. Taentzer,

and E. Weiss, “Graphical Deﬁnition of In-Place Transforma-

tions in the Eclipse Modeling Framework,” in MODELS, ser.

LNCS, vol. 4199. Springer, 2006, pp. 425–439.

[

13] E. Biermann, C. Ermel, and G. Taentzer, “Precise Semantics

of EMF Model Transformations by Graph Transformation,” in

MODELS, ser. LNCS, vol. 5301. Springer, 2008, pp. 53–67.

[

[

14] A. Zu¨ndorf, “AntWorld,” University of Kassel, 2008, http://

∼

www.se.eecs.uni-kassel.de/ fujabawiki/index.php/AntWorld.

15] E. Biermann, K. Ehrig, C. Ermel, and J. Hurrelmann, “Gener-

ation of Simulation Views based on EMF: Long Version,” TU

Berlin, Tech. Rep. 2009-17, 2009, http://www.eecs.tu-berlin.

de/menue/forschung/forschungsberichte/2009.

Figure 9. Generated AntWorld simulation view

VI. CONCLUSION AND FUTURE WORK

[

16] S. Jurack, L. Lambers, K. Mehner, and G. Taentzer, “Sufﬁ-

cient Criteria for Consistent Behavior Modeling with Reﬁned

Activity Diagrams,” in MODELS, ser. LNCS, vol. 5301.

Springer, 2008, pp. 341–355.

We have introduced the underlying concepts of an EMF-

based generator of modeling tool environments for visual

DSMLs. Our implementation TIGER2 is an ongoing project.

Future work is the completion of a mainly visual and

intuitive Designer component, followed by a comprehensive

user evaluation to better compare the generation environment

to existing approaches.

[

[

17] Eclipse Graphical Editing Framework (GEF) – Version 3.4,

Eclipse Consortium, 2009, http://www.eclipse.org/gef.

18] T. Modica, E. Biermann, and C. Ermel, “An ECLIPSE

Framework for Rapid Development of Rich-featured GEF

Editors based on EMF Models,” in Methodical Development

of Modelling Tools, 2009.

6

132719

