A Visual Editor for Reconfigurable Object Nets based on the ECLIPSE Graphical Editor Framework

Enrico Biermann, Claudia Ermel, Frank Hermann and Tony Modica
Technische Universität Berlin, Germany

{enrico,lieske,frank,modica}@cs.tu-berlin.de

Abstract

The main idea behind *Reconfigurable Object Nets (RONs)* is the integration of transition firing and rule-based net structure transformation of place/transition nets during system simulation. RONs are high-level nets with two types of tokens: object nets (place/transition nets) and net transformation rules (a dedicated type of graph transformation rules). Firing of high-level transitions may involve firing of object net transitions, transporting object net tokens through the high-level net, and applying net transformation rules to object nets. Net transformations include net modifications such as merging or splitting of object nets, and net refinement. This approach increases the expressiveness of Petri nets and is especially suited to model mobile distributed processes. The paper presents a visual editor for RONs which has been developed in a student project at TU Berlin in summer 2007. The visual editor itself has been realized as a plug-in for ECLIPSE using the ECLIPSE Modeling Framework (EMF) and Graphical Editor Framework (GEF) plug-ins.

Keywords: Petri nets, net transformation, graph transformation, visual editor, reconfigurable object nets, Eclipse, GEF

1 Introduction

Modelling the adaption of a system to a changing environment has become a significant topic in recent years. Application areas cover e.g. computer supported cooperative work, multi agent systems, dynamic process mining or mobile ad-hoc networks (MANETs). Especially in the context of our project *Formal modeling and analysis of flexible processes in mobile ad-hoc networks* [PEH07, For06] we aim to develop a formal technique which on the one hand enables the modeling of flexible processes in MANETs and on the other hand supports changes of the network topology and the transformation of processes. This can be achieved by an appropriate integration of graph transformation, nets and processes in high-level net classes.

The main idea behind *Reconfigurable Object Nets (RONs)* is the integration of transition firing and rule-based net structure transformation of place/transition nets during system simulation. RONs are high-level nets with two types of tokens: object nets (place/transition nets) and net transformation rules (a dedicated type of graph transformation rules). Thus, on the one hand, RONs follow the paradigm "nets as tokens", introduced by Valk in [Val98], and, on the other hand, extend this paradigm to "nets and rules as tokens" in order to allow for modelling net structure changes (reconfigurations) of object nets. Firing of RON-transitions may involve firing of object net transitions, transporting object net tokens through the high-level net, and applying net transformation rules to object nets. Net transformation rules model net modifications such as merging or splitting of object nets, and net refinements.

A rule r consists of a left-hand side L related to a right-hand side R and describes the local replacement of L by R. Similar to the concept of graph transformations [EEPT06], each application of a rule $r = (L \to R)$ to a source net N_1 leads to a net transformation step $N_1 \stackrel{r}{\Longrightarrow} N_2$, where in the source net N_1 the subnet corresponding

to the left-hand side L is replaced by the subnet corresponding to the right-hand side R, yielding the target net N_2. Rule-based Petri net transformations have been treated in depth in e.g. [EP04, PU03].

The formal basis for RONs is given in [HME05], where high-level nets with nets and rules as tokens are defined algebraically, based on algebraic high-level nets [PER95]. Here we present a visual editor for RONs which has been developed in a student project at the TU Berlin in summer 2007. The visual editor itself has been realized as a plug-in for ECLIPSE using the ECLIPSE Modeling Framework (EMF) [EMF06] and Graphical Editor Framework (GEF) [GEF06] plug-ins. In RONs, as presented in this paper, the algebraic operations defined for rule applications and transition firing are modeled as special RON-transition types which have a fixed firing semantics. This facilitates the implementation of the RON editor and simulator since no parser for algebraic specifications is needed. It turned out that the four RON-transition types are adequate to model various interesting examples. More information on RONs, case studies and downloads of the RON tools are available on our RON homepage [RON07].

Section 2 introduces RONs along the running example of distributed producers/consumers. After a brief overview of the ECLIPSE Graphical Editor Framework, the milestones of the students' project and the different parts of the RON tool are explained in Section 3. Section 4 gives an outlook on future extensions of the tool.

2 Reconfigurable Object Nets: An Example

In our example RONs are applied to model a distributed system of producers and consumers where several producers and consumers may interact with each other. In the initial state of the sample RON in Fig. 1 potential producers and consumers are distributed on different Net places as independent object nets without interaction. Producer

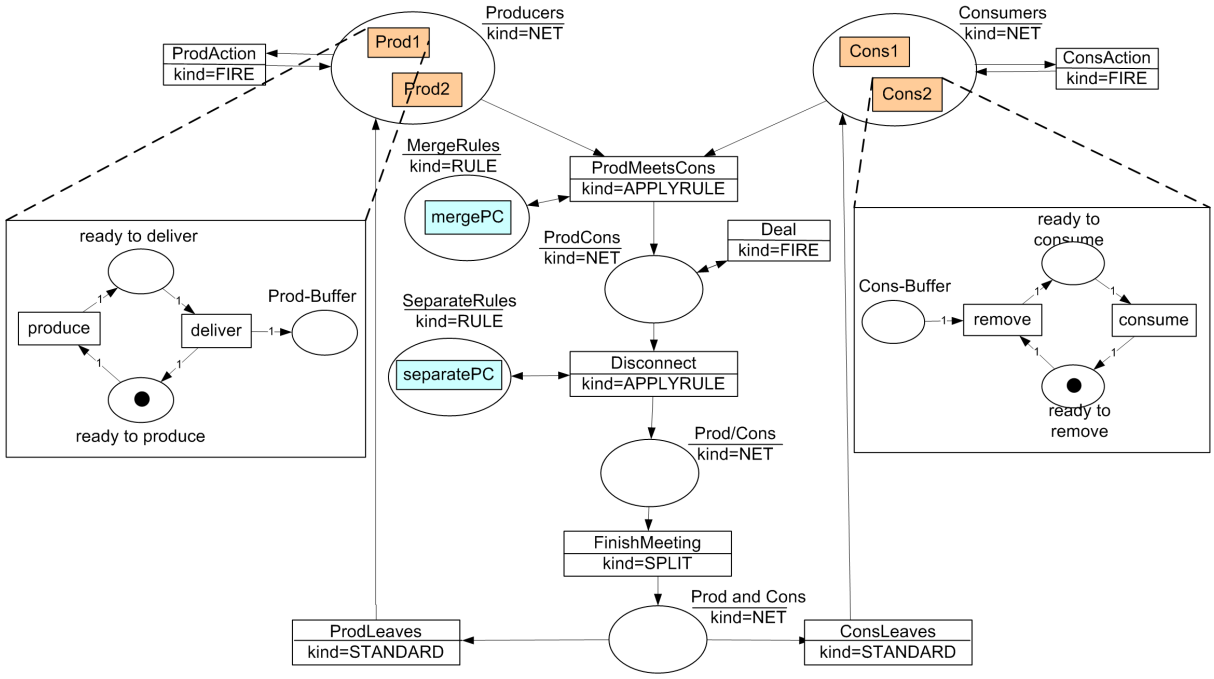

Figure 1: Distributed Producers/Consumers modelled as RON

nets may fire, e.g. they can produce items and place them on the buffer place. Firing in object nets is triggered by firing a RON transition of type FIRE, which takes one object net with marking M from the Net place in its pre-domain and puts the same object net, now marked by one of the possible successor markings of M, into all of

its post-domain places. For producer-consumer interaction, a producer net can be merged with a consumer net by firing the RON transition *ProdMeetsCons* of type APPLYRULE. A transition of this type takes an object net from each of the pre-domain Net places, a rule from the pre-domain Rule place, applies this rule to the disjoint union of all the taken object nets and puts the resulting net to all post-domain Net places. The rule *merge-PC*, depicted in Fig. 2, glues a producer object net and a consumer object net by inserting a *connect* transition between both buffers. A so-called negative application condition (NAC) forbids the application of the rule if there already exists a *connect* transition. Note that the transition *ProdMeetsCons* controls which producer interacts with which consumer.

By firing the FIRE transition *Deal*, in the glued net the consumer now can consume items produced by the producer as long as there are tokens on the place *Prod-Buffer*. Moreover, the producer may also produce more items and put them to the buffer. After the deal has been finished, the nets are separated again by firing the APPLYRULE transition *Disconnect*. This applies rule *separate-PC* in Fig. 2 to the glued net which deletes the *connect* transition from the net.

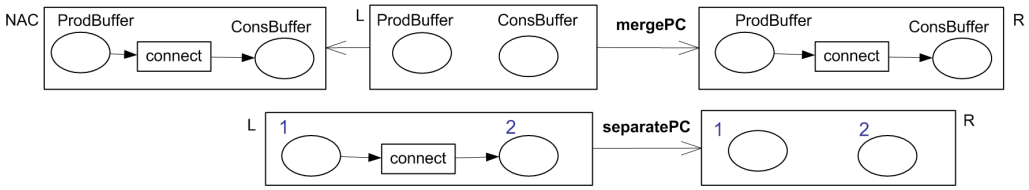

Figure 2: Rules for gluing and for separating a producer and a consumer net

Note that the resulting net, which is put on place *Prod/Cons*, is still one single object net which consists of two unconnected components. In order to split these components into two object nets, a transition of type SPLIT has to be fired. Firing RON transition FinishMeeting results in two separate object nets on place *Prod and Cons*. In a last step, the now separated producer and consumer nets are returned to their initial places by firing the STANDARD transitions *ProdLeaves* and *ConsLeaves*. STANDARD transitions simply remove a net token from each pre-domain place and add the disjoint union of all removed object nets to each of the post-domain Net places. In our example, this means that the STANDARD transitions may also put the object nets back to the "wrong" initial place. It is possible to avoid such behaviour by using APPLYRULE transitions instead where rules check the object nets for the occurrence of a *producer* or *consumer* place, respectively.

3 An Editor for RONs

Conceptually, the visual editor for RONs is divided into four main components which were scheduled to be implemented in four milestones, building on each other, each resulting in software which could be run and tested. Hence, the project consisted of short development cycles leading to fast results and error recognition.

Similar student projects dealing with the development of visual editors as ECLIPSE plug-ins have been carried out by our research group for some years now. It emerged that after a short period of getting used to GEF the students could concentrate on devising and implementing advanced domain specific editing features.

In addition to the usual advantages like Java's platform independence and ECLIPSE being a powerful IDE and available as open source we could make use of ECLIPSE's highly extensible plug-in architecture. We employ the ECLIPSE Modeling Framework (EMF) to automatically generate data model code for the editor from class diagrams and to handle persistence operations.

The Graphical Editor Framework (GEF) provides means of implementing a visual editor based on the Model-View-Controller pattern relying on ECLIPSE's Standard Widget Toolkit (SWT). It supports many operations and features common to most graphical editors like zooming, various layouting of figures, support for drag & drop etc.

To prevent the students from struggling with ECLIPSE's internal details especially at the beginning of the project we provided an abstract framework of an editor demonstrating several concepts and techniques ECLIPSE and GEF offer, e.g. graphical views showing multiple models for rules.

In the following, we present the results of the four milestones, i.e. the RON tree view based on the EMF model for RONs, and the visual editors for object nets, for transformation rules and for high-level nets with the four transition types FIRE, APPLYRULE, SPLIT and STANDARD. Fig. 3 shows a screenshot of the RON editor showing all views and editors.

RON Tree View. View 1 in Fig. 3 shows the main editor component, a tree view for the complete RON model from which the graphical views can be opened by double-click.

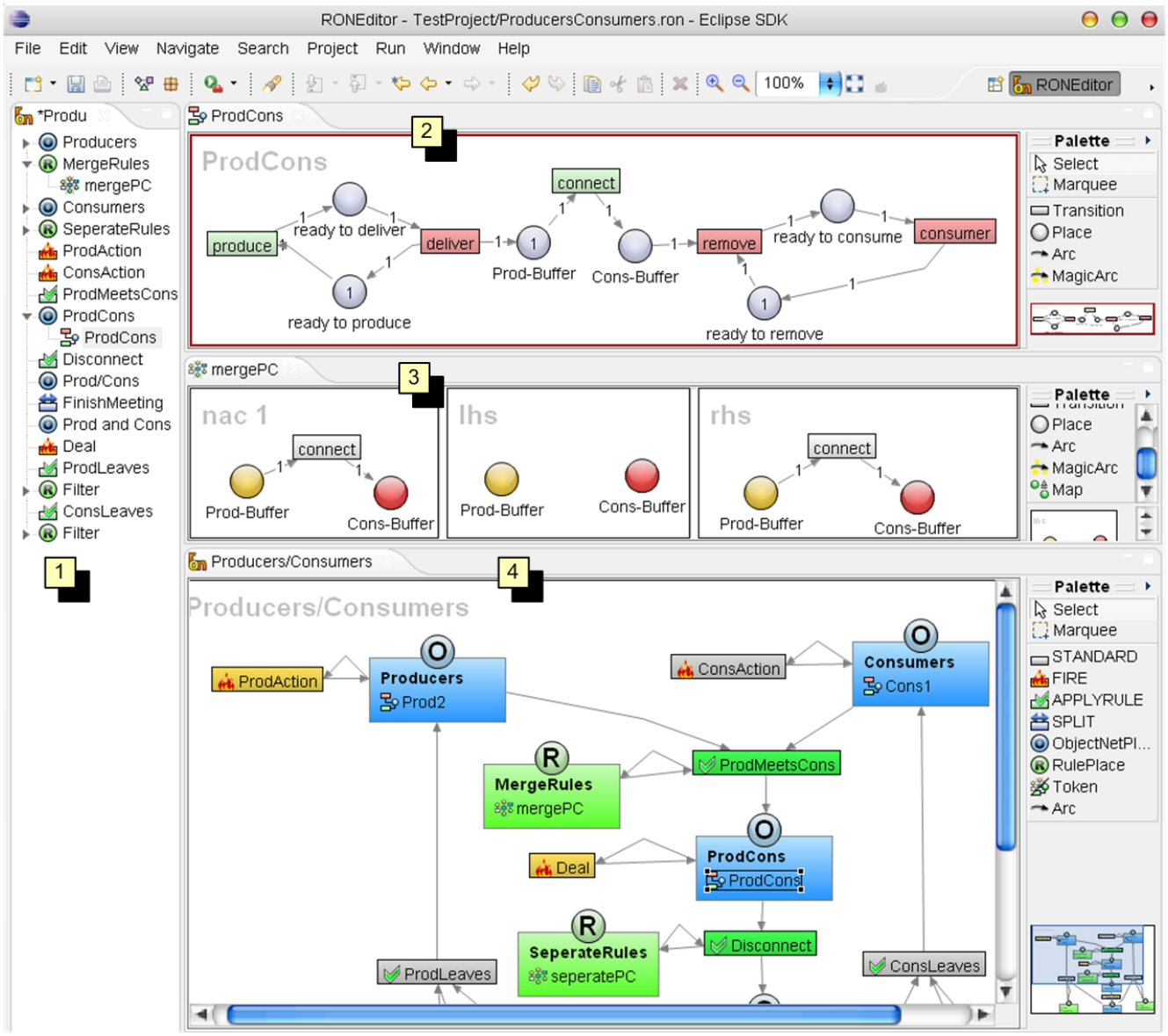

Figure 3: The RON Environment for Editing and Simulating Reconfigurable Object Nets

Object Net Editor. The first component to be implemented was a graphical editor for object nets, i.e. the net tokens on the RON's net-typed places. This component is actually a place/transition net editor, allowing the simulation of firing transitions. Its implementation could be reused for the rule editor and the RON editor as well. An object editor panel is shown in Fig. 3, View 2, holding the object net *ProdCons*, which models producer-consumer interaction.

Transformation Rule Editor. In the second milestone the RON framework was extended by an editor for transformation rules. It mainly consists of three editor panels, one for the left-hand side (LHS), one for the right-hand side(RHS) and one for a negative application condition(NAC) (see view 3 in Fig. 3). Each editor panel is basically an object net editor itself, but with the additional possibility to relate the object nets by defining mappings on places and transitions. Mappings are realized by the *mapping tool* of the rule editor that allows the matching of LHS objects to RHS or NAC objects to indicate which objects are preserved by the rule. In the editor, mappings are indicated by corresponding object colouring. In order to ensure that the mapping specified by the mapping tool is also a valid Petri net morphism, it is checked for each mapped transition that all places in its pre- (post-)domain in the LHS are mapped to the corresponding places in the pre- (post-)domain of in the RHS. Another restriction is that all rules are injective, so different LHS objects must be mapped to different RHS objects. Note that the object net *ProdCons* in Fig. 3, View 2, is the result from applying rule *mergePC* to two object nets *Prod1* and *Cons2* from the places *Producers* and *Consumers*, respectively. (For the situation before the rule is applied, see Fig. 1).

High-Level Net Editor. The third extension of the RON editor involved editing of high-level nets which control object net behaviour and rule applications to object nets. Such a high-level net is drawn in the RON editor panel, shown in Fig. 3, View 4. Here, NET places carrying object net tokens are blue containers marked by an "O" for *Object Nets*. RULE places carrying transformation rules are green containers marked by an "R" for *Rules*. Each transition type has a special graphical icon as visualization: for FIRE, for APPLYRULE, for SPLIT, and for STANDARD. Enabled RON-transitions are coloured, disabled ones are gray.

Simulation of RONs. A RON transition is fired when double-clicked. The simulation of firing transitions of kinds STANDARD, FIRE, and SPLIT has been implemented directly in the editor. In order to simulate firing of APPLYRULE transitions, internally the RON editor was extended by a converter to AGG, an engine to perform and analyze algebraic graph transformations [AGG]. If the user gives the command to fire an APPLYRULE transition he has to select the rule and the object net token(s) in the pre-domain the rule should be applied to. This is realized in the user interface shown in Fig. 3, View 4, by ordering the tokens in the corresponding NET and RULE containers in a way that the uppermost tokens are the ones considered by the rule application. Furthermore, the user is asked for a match defining the occurrence of the rule's left-hand side in the selected object net. Optionally, AGG can find or complete partial matches and propose them to the user in the RON editor. With the selected rule, match, and object net AGG computes the result of the transformation which is put on the post-domain places according to the firing semantics explained in Section 2.

4 Conclusion and Ongoing Work

Modelling mobile and distributed systems requires a modelling language which covers both, change and coordination. RONs meet these conditions and we have shown its power on a reduced but instructive example modelling interaction between producers and consumers. The abstract high-level net controls the flow, selection and use of object tokens being object nets but also rules for reconfiguration. Thus the details on the object level are hidden in these tokens such that they can be modelled separately keeping the view on the high-level net concise.

Currently, RONs use specialized transitions in the high-level view, but we intend to extend the presented editor to cover full algebraic high-level nets, such that a high-level transition is controlled by firing conditions and arc inscriptions consisting of terms and attributes. Furthermore, rule specification shall be extended to cope also with P/T net morphisms which are *not* injective. This would allow net transformations like gluing and cloning while RONs are restricted to connecting object nets via new P/T net transitions. Additionally, we plan to allow tokens in rules, such that a rule application will depend on the presence of tokens, i.e. on the state of an object net. Finally, the editor for RONs shall be extended by analysis features like a check of conflicts between object net transition firing and rule application according to [EHP+07].

References

[AGG] AGG Homepage. http://tfs.cs.tu-berlin.de/agg.

[EEPT06] H. Ehrig, K. Ehrig, U. Prange, and G. Taentzer. *Fundamentals of Algebraic Graph Transformation*. EATCS Monographs in Theoretical Computer Science. Springer Verlag, 2006.

[EHP+07] H. Ehrig, K. Hoffmann, J. Padberg, U. Prange, and C. Ermel. Independence of net transformations and token firing in reconfigurable place/transition systems. In *Proc. Conf. on Application and Theory of Petri Nets and Other Models of Concurrency*, volume 4546 of *LNCS*, pages 104–123. Springer, 2007.

[EMF06] Eclipse Consortium. *Eclipse Modeling Framework (EMF) – Version 2.2.0*, 2006. http://www.eclipse.org/emf.

[EP04] H. Ehrig and J. Padberg. Graph grammars and Petri net transformations. In *Lectures on Concurrency and Petri Nets*, volume 3098 of *LNCS*, pages 496–536. Springer, 2004.

[For06] ForMA$_l$NET. DFG Project, Technical University of Berlin. *Formal Modeling and Analysis of Flexible Processes in Mobile Ad-hoc Networks*, 2006. http://www.tfs.cs.tu-berlin.de/formalnet.

[GEF06] Eclipse Consortium. *Eclipse Graphical Editing Framework (GEF) – Version 3.2*, 2006. http://www.eclipse.org/gef.

[HME05] K. Hoffmann, T. Mossakowski, and H. Ehrig. High-Level Nets with Nets and Rules as Tokens. In *Proc. Conf. on Application and Theory of Petri Nets and other Models of Concurrency*, volume 3536 of *LNCS*, pages 268–288. Springer, 2005.

[PEH07] J. Padberg, H. Ehrig, and K. Hoffmann. Formal modeling and analysis of flexible processes in mobile ad-hoc networks. *EATCS Bulletin*, 91:128–132, 2007.

[PER95] J. Padberg, H. Ehrig, and L. Ribeiro. Algebraic high-level net transformation systems. *Mathematical Structures in Computer Science*, 5:217–256, 1995.

[PU03] J. Padberg and M. Urbášek. Rule-Based Refinement of Petri Nets: A Survey. In Ehrig et al. *Advances in Petri Nets: Petri Net Technology for Communication Based Systems*, volume 2472 of *LNCS*, pages 161–196. Springer, 2003.

[RON07] Student's Project. TFS, Technical University of Berlin. *Reconfigurable Object Nets*, 2007. http://www.tfs.cs.tu-berlin.de/roneditor.

[Val98] R. Valk. Petri Nets as Token Objects: An Introduction to Elementary Object Nets. In *Proc. Conf. on Application and Theory of Petri Nets*, volume 2987 of *LNCS*, pages 1–25. Springer, 1998.