DrawNET Xe: GUI and Formalism Definition Language

A. Baravalle*, G. Franceschinis†, M. Gribaudo*, V.Lanfranchi*, M.Iacono‡, N.Mazzocca‡ and V.Vittorini§
* Dip. di Informatica, Univ. di Torino, Torino, Italy,
Email: [andres,marcog,vitaveska]@di.unito.it
† Dip. di Informatica, Univ. del Piemonte Orientale, Alessandria, Italy,
Email: giuliana@mfn.unipmn.it
‡ Dip. di Ing. dell'Informazione, Seconda Univ. di Napoli, Aversa (NA), Italy
Email: [mauro.iacono,nicola.mazzocca]@unina2.it
§ Dip. di Informatica e Sistemistica, Univ. di Napoli *Federico II*, Napoli, Italy,
Email: vittorin@unina.it

***Abstract*— DrawNET Xe is a configurable GUI for graph-based formalism, supporting hierarchical model construction. An important feature of DrawNET Xe is that it allows multi-formalism model composition, and implements the ideas proposed in the OsMoSyS methodology, based on the definition of *metaformalism*, *model class* and *model object* concepts. In this paper the evolution of the tool, from its ancestor DrawNET, is briefly summarized, and its main new features are presented.**

I. Genesis of DrawNET Xe

DrawNET Xe is a configurable GUI for graph-based formalisms also used to experiment multi-formalism modeling of complex systems. This section briefly introduces its evolution from the initial version, DrawNET, passing through an intermediate extension called DrawNET++.

The first impulse to the development of DrawNET stemmed from the research pursued at the *Dip. di Informatica* of the Univ. di Torino [4]. It was initially designed to provide a graphical front-end to extended versions of Petri Nets (PN) such as Fluid Petri Nets, but it was soon recognized that it could have been easily extended to experiment with any kind of graph based formalism. The basis for the tool configurability was the first version of a *Formalism Definition Language* (FDL) used to define the elements of any formalism a user may want to use. Constraints could be defined in the FDL description to express syntactical properties of the graph structure, e.g. they could be used to ensure that in PNs arcs do not connect two places or two transitions. The FDL language is based on XML, moreover the FDL description of a specific formalism implicitly defines an XML format for a model which is the output format of DrawNET. All graphical aspects concerning the elements of formalisms are dealt with separately. An inheritance mechanism allows to derive a new formalism by extending an existing one (e.g. Colored PNs can be derived from PNs by addition of some annotations). Inheritance can be applied also to single formalism elements: for example timed and immediate transitions in the Generalized Stochastic Petri Net formalism could be derived by extending the transition element definition of PNs.

At the same time, a collaboration between two research groups at the *Univ. di Napoli* and the *Dip. di Informatica* of the *Univ. del Piemonte Orientale* was starting on the themes of compositional model construction with sub-model reuse and multi-formalism modeling. DrawNET was adopted to support multi-formalism and compositional models. At this aim, the tool was enhanced by implementing the possibility to manage sub-models to some extent. The new version was named DrawNET++ and first presented and demonstrated in [1]. The tool does not embed any knowledge on the semantics of the models it handles: the semantics is left to external solvers, that can be invoked from the DrawNET++ GUI. Nevertheless, much work has been done by the two research groups to define formalisms integration and composition strategies and develop the external solvers. In particular, ongoing research is aimed at applying workflow principles to cope with the problem of multi-solution when analyzing/simulating multi-formalism models.

Further development of DrawNET++ was stimulated by the formalization of a new approach to multi-formalism modeling (called OsMoSyS), based on meta-modeling and some concept from object orientation [6]. OsMoSys introduces a modeling methodology based on the definition of *metaformalism*, *model class* and *model object*, partially adopted in [2] and [3].

The implementation of the OsMoSys methodology requires the full implementation of hierarchical modeling and of the sub-models instantiation mechanism, the ability to hide part of the sub-model content and to define interface sub-model elements to which the environment can connect. Moreover, for allowing sub-models reuse, the possibility of defining parametric sub-models organized into libraries and of composing them to obtain more complex sub-models is needed. Finally, to allow visualization of solution results on the GUI, as well as to enable multi-solution approaches, it is necessary to include a mechanism to handle the results: these needs, together with the possibility of a more general definition for the FDL, suggested to redesign the tool and a to implement the new version from scratch.

II. DRAWNET XE

At first, the FDL has been extended to better support the requirements. The new version of the FDL (named FDL-2) has been designed to allow the user to specify generic formalisms, in addition to graph formalisms. FDL-2 enables the hierarchical composition of submodels described in different formalisms; in addition, multiple inheritance has been introduced to increase the expression power of the language. Since FDL-2 is no more limited to graph based formalisms, a "GraphBased" formalism has been defined from whom any formalism should be inherited in order to be handled by DrawNET Xe. Support to parametric compositional modeling is similarly obtained by explicit declarations of interface elements and parameter elements. In order to easily add to a formalism all the features needed to deal with parametric submodel composition, the necessary elements have been encapsulated into an "Instantiable" formalism: composable (graph based) formalisms definitions can be easily obtained by deriving from both "Instantiable" and "GraphBased" by multiple inheritance.

Fig. II shows the Graphical User Interface of DrawNET Xe and a Petri Net composed model representing a very simple RAID controller. The model is built by using two submodels, Disk1 and Disk2, which are two model objects, i.e. different instances of the model class "Disk" stored in a PN models library.

The model class "Disk" is a PN model itself (but it could be a model expressed by a different formalism) whose transitions In and Out are defined to be interface elements, since they must be used to link the class instances to the external environment. In addition some parameters are defined (the buffer size and the number of the disks) that have been set to a default value in the class specification (5 and 3 respectively) but may be redefined when instantiating the model objects Disk1 and Disk2. In Fig. II both the composed model and the model class $Disk$ are graphically shown. In the upper right corner the $Model$ elements page shows the available language elements to be inserted into the current model. On its left, the $Object/Class$ window shows that the current library offers two classes, CPU and $Disk$. The same window is used to define the parameters for the $Disk$ class (see the lower right corner of the figure) through the $Import$ and $Export$ windows. The $Element$ window is used: a) to define the element properties; b) to define the results to be evaluated on that element; c) to instantiate an object, as shown in the left side of the figure: in this case, the $Parameters$ folder is shown in place of the "Properties" folder. In this window, the parameters are given to the $Disk2$ object model.

The XML description of the model automatically generated by DrawNET Xe is reported in Fig. 2: a Model Definition Language (MDL) document is composed by two sections.

The first section contains the specification of the classes used to build the model in case it is a composed model. In this case it describes the structure of the $Disk$ model class. It has two interfaces to be linked to other object when instantiated and exports the parameters "DiskInArray" and "BufferSize" with default value 3 and 5, as previously shown. These parameters stay for the "token" property of the "Disks" place and for the "token" property of the "Buffer" place respectively. These places are part of the model, as shown in the rest of the document.

The second section defines the model itself, as the mdl tag states at the beginning of the document. The $RAID$ model contains (use tags) two instances of the $Disk$ class, namely "Disk1" and "Disk2", as shown in Fig. II. Here the $Disk$ class is instantiated by specifying the desired values for parameters. The resulting objects are treated as any other element in the model (see arcs "A2", "A3", "A4" and "A5"). Notice that the class is stored in the model only once and that the $interface$, $parameter$ and use tags are supported by the fact that $Instantiable$, in which they are defined, is an ancestor of the PN formalism used to draw this model.

III. DRAWNET XE DEMONSTRATION

DrawNET Xe is currently under development: the appearance and main components of the tool can be seen in Fig. II, but at the state a subset of its functionalities has been implemented.

The FDL-2 is fully defined and the description of the desired formalism can be read by the GUI, together with the graphical appearance description file, and the elements of the loaded formalism are displayed in the proper palette. The user can draw his models using layers, grids and other standard GUI features. A complete history of the actions taken by the user, plus the ability to undoing/redoing actions is already implemented, together with ability to add graphical elements that do not belong to the model definition (such has box, lines and texts) to describe the systems being drawn. Also the tree-structure of the various components included in a model can be visualized, together with a navigation panel that simplifies the design of large models. Samples of FDL-2 and MDL model descriptions shall be presented during the demonstration, to illustrates their features and possible application in different contexts. Moreover, the DrawNET Xe interface will be compared with the previous one and it will be possible to appreciate the possibility of easily dealing with multi-formalism approaches [5].

ACKNOWLEDGEMENTS

The work of G. Franceschinis and M. Gribaudo has been partially supported by the Italian MIUR under the FIRB project Perf (n. RBNE019N8N). The work of V. Vittorini has been partially supported by Consorzio Interuniversitario Nazionale per l'Informatica (CINI) and by the National Research Council (CNR).

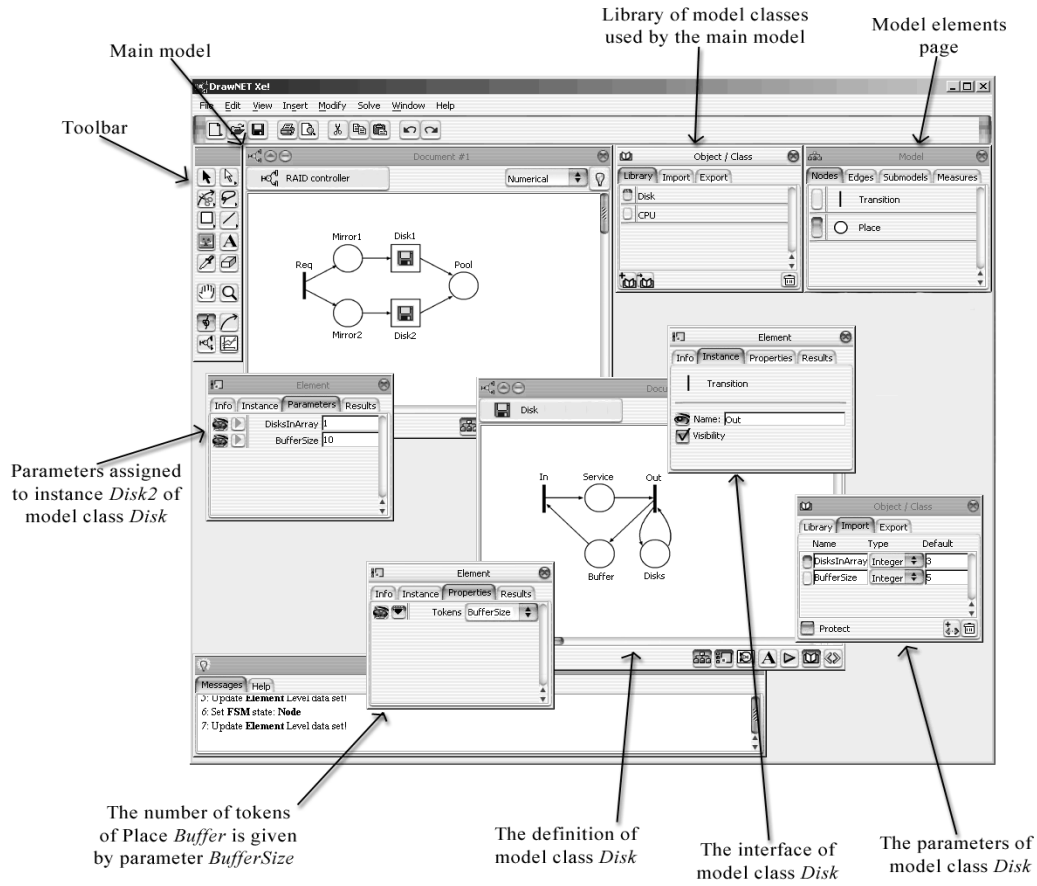

Fig. 1. The GUI of DrawNET Xe

```
<mdl fdl="PN.fdl" main="RAID">

    specification of the "Disk" Model Class

<PN name="Disk">

    Interface of the "Disk" Model Class

        <interface add="In"/>
        <interface add="Out"/>

    Parameters to be set when instantiating a "Disk" object

        <parameter name="DiskInArray" default="3">
         <assign obj="Disks" property="token"/>
        </parameter>
        <parameter name="BufferSize" default="5">
         <assign obj="Buffer" property="token"/>
        </parameter>

        <Transition name="In" priority="1"/>
        <Transition name="Out" priority="1"/>
        <Place name="Buffer"/>
        <Place name="Disks"/>
        <Place name="Service" tokens="0"/>
        <Arc name="A0" from="In" to="Service" weight="1">
        <Arc name="A1" from="Service" to=Out"" weight="1">
        <Arc name="A2" from="Disks" to="Out" weight="1">
        <Arc name="A3" from="Out" to="Disks" weight="1">
        <Arc name="A4" from="Out" to="Buffer" weight="1">
        <Arc name="A5" from="Buffer" to="In" weight="1">
    </PN>
```

```
    specification of the "Raid" Model

<PN name="RAID">
        <Transition name="Req" priority="1"/>
        <Place name="Mirror1" tokens="0"/>
        <Place name="Mirror2" tokens="0"/>
        <Place name="Pool" tokens="0"/>

        the "Raid" Model contains two instances of the "Disk" class

<use class="#Disk" name="Disk1">
            <parameter name="DisksInArray" value="1"/>
            <parameter name="bufferSize" value="10"/>
        </use>
        <use class="#Disk" name="Disk2">
            <parameter name="DisksInArray" value="1"/>
            <parameter name="bufferSize" value="10"/>
        </use>
       <Arc name="A0" from="Req" to="Mirror1" weight="1">
       <Arc name="A1" from="Req" to="Mirror2" weight="1">
       <Arc name="A2" from="Mirror1" to="Disk1.In" weight="1">
       <Arc name="A3" from="Mirror2" to="Disk2.In" weight="1">
       <Arc name="A4" from="Disk1.Out" to="Pool" weight="1">
       <Arc name="A5" from="Disk2.Out" to="Pool" weight="1">
   </PN>

</mdl>
```

Fig. 2. The XML description of a simple composed model

REFERENCES

[1] Franceschinis, G., Gribaudo, M., Iacono, M., Mazzocca, N., Vittorini, V.: DrawNET++: Model Objects to Support Performance Analysis and Simulation of Complex Systems. Proc. 12th International Conference on Modelling Tools and Techniques for Computer and Communication System Performance Evaluation (TOOLS 2002), London, UK, April 2002, number 2324 in Lecture Notes in Computer Science (LNCS), Springer-Verlag, (2002) 233–238

[2] Franceschinis, G., Gribaudo, M., Iacono, M., Mazzocca, N., Vittorini, V.: Towards an Object Based Multi-Formalism Multi-Solution Modeling Approach. Proc. of the Second Workshop on Modelling of Objects, Components and Agents Aarhus (MOCA02), Denmark, **26-27** (August 2002) 47–65.

[3] Franceschinis, G., Marrone, S., Mazzocca, N., Vittorini, V.: SWN Client-Server Composition Operators in the OsMoSys framework, Proc. of 10th Int. Workshop on Petri Nets and Performance Models, PNPM2003, IL, USA, Sept. 2003. IEEE Soc. Press.

[4] Gribaudo, M., Valente, A.: Framework for Graph-based Formalisms. Proc. of the first International Conference on Software Engineering Applied to Networking and Parallel Distributed Computing (SNPD'00), Reims, France, (May 2000) 233–236.

[5] Gilmore S. and Gribaudo , M.: Graphical Modelling of Process Algebras with DrawNET. Tools presentation at the Multiconference on Measurement, Modelling and Evaluation of Computer-Communication Systems, Sept. 2003, IL, USA.

[6] Vittorini, V., Franceschinis, G., Iacono, M., Mazzocca, N., OsMoSys: a new approach to multi-formalism modeling of systems. Submitted to Journal on Software and System Modeling, Springer.