A SysML extension for Bond Graphs support

Skander Turki
LISMMA (EA 2336) Supmeca Toulon
Maison des technologies
Toulon 83000, France
skander.turki@supmeca.fr

Thierry Soriano
LISMMA (EA 2336) Supmeca Toulon
Maison des technologies
Toulon 83000, France
thierry.soriano@supmeca.fr

***Abstract* – In this paper we present a contribution to SysML for energy interchange depiction. This contribution consists of an extension to the activity diagram. We mapped it to the Bond Graph formalism which is often used for the design of mechatronic systems. We use the extension mechanisms that SysML inherits from UML2.0 such as stereotypes and tagged values to establish mapping rules from activities to bond graphs. The paper is concluded by an example to illustrate this proposal.**

Index Terms – Mechatronics, Bond Graphs, SysML, UML 2.0, Activity diagram.

I. INTRODUCTION

The system approach is essential to deal with complex systems such as mechatronic systems. It is a global multidisciplinary methodological approach that aims at master the practices of systems engineering. SysML [1] is an essay to provide systems engineers with a standard language that covers the specification, analysis, design, verification and validation phases. Bond Graphs are used to depict the energetic transfers between subsystems of different natures (mechanical, electrical, hydraulic…). It is today used in numerous important projects inside companies such as PSA, Renault, General Motors…[5]

The SysML language as it is defined in its current version doesn't enable users to use this powerful tool of analysis, simulation and dimensioning that is the Bond Graph formalism.

In this work we are presenting one possibility to integrate Bond Graphs into SysML using SysML constructs or in the worst case UML 2.0 constructs [3].

In the second section we will introduce SysML. Then we will present the Bond Graph formalism. In the fourth is a detailed presentation of our contribution. Then we give an example. Finally we give some conclusions and perspectives of this ongoing work.

II. SYSML PRESENTATION

SysML is the Systems Modeling Language built as a response to the OMG's (Object Management Group) UML for systems engineering request for proposal [2]. This proposal claims that systems engineers need a standard language easy to integrate both in the engineering teams and in the existing tools. It asserts that the language must help heterogeneous teams (software, electronics, mechanics…) to work together and understand each other. This is why SysML is based on the minimal subset of UML2.0 that satisfies systems engineers needs. It is intended to be minimal to be easily accepted by the systems engineering community. In addition to that SysML is an extension to UML2.0 and tries to bring the minimum change to the UML metamodel to facilitate its implementation to tool vendors. It benefits from the UML extension mechanisms (stereotypes, tagged values). In the "UML 2.0 infrastructure specification" [4] a stereotype "defines how an existing metaclass (or stereotype) may be extended, and enables the use of platform or domain specific terminology" [UML infrastructure page 174]. These mechanisms will be used to specialise SysML to specific domains such as aeronautics, automobile etc. SysML is also aligned with other standards such as ISO AP-233 [8] for data exchange to support tool interoperability. SysML also inherits the XMI (XML model interchange) from UML2.0.

But this minimal SysML or lets say this basic SysML will need to bring to these systems engineers their usual tools by creating extension packages that can be added or removed from their design environment just like the practices of the software engineers with the profiles of JAVA, C++, CORBA and much others. This is why we think that we need to build these extensions for the basic SysML. The activity diagram is already used to express EFFBD's (Enhanced Functional Flow Block Diagrams). These EFFBD's are usually used by systems engineers. Bond Graphs are also usually used by systems engineers. In [11] W. Borutzky establishes a relationship between Bond Graphs and object-oriented modelling. In this paper, we are presenting their application using SysML. The Bond Graph formalism is quickly presented in the following.

III. BOND GRAPHS SUMMARY

Modelling a mechatronic system often needs the description of the energetic transfers between subsystems. In addition a homogenisation of such a description must be accomplished to visualize on the same diagram a phenomenological analysis of the system. This is done by the Bond Graph formalism that allows in the same time to discover the system's equations, which are used for the simulation and the dimensioning.

There are three levels of Bond Graphs representation [6]. The first level is the word-Bond Graph that is used to have a first approach in describing the energy map of the system and its composition. The second level is the Acausal Bond Graphs used to show the energetic transfers by representing each subsystem by its energetic function assuming that the decomposition is advanced enough so that we can replace each subsystem by an elementary component. These elementary components will be defined later. The third level is that of the assignment of the causality which helps finding the system's equations; it is the causal Bond Graph. Causality is represented by a vertical line on the energetic arrows. This concept has nothing to do with the causality concept used in systems theory.

The set of the language constructs of the third level contains all the necessary constructs to build any of the three Bond Graph types.

A subsystem is represented by a closed line with a name. This line represents the frontiers of the subsystem. For each energy interchange of the system with its environment we associate to it an energetic port of a defined type (mechanical energy, electrical energy, etc…). A unidirectional semi headed arrow shows the energy interchange throw this port and carries the data relative to the power transported (e: effort and f: flow). These two variables are necessary and sufficient to describe the energetic transfers inside the system. They correspond to a couple of variables in each energetic domain. (tab 1)

TABLE 1
THE EFFORT AND THE FLOW IN DIFFERENT ENERGETIC DOMAINS.

Energetic domain	**Effort e**	**Flow f**
Translational mechanics	Force	Velocity
Rotational mechanics	Torque	Angular velocity
Electricity	Voltage	Current
Hydraulic	Pressure	Volume flow rate
Thermal	Temperature	Entropy change rate
	Pressure	Volume change rate
Magnetic	Magneto-motive force	Magnetic flow

A. Elementary components or nodes

The elementary components/subsystems are classified (tab 2) by their energetic behaviour (energy dissipation, energy storage…) and their function inside the system (flow sensor..).

In this proposal we use the Bond Graphs notation described in [9] where only TF and GY are extended to modulated elements (MTf and MGy). We don't map mixed elements such as IC, IR or RC. De and Df are used to represent sensors that do not consume power (ideal sensors).

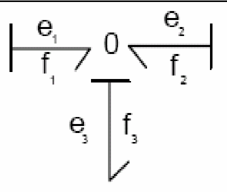
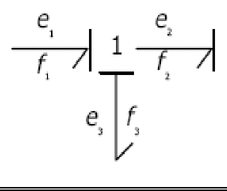
TABLE 2
BOND GRAPH ELEMENTARY COMPONENTS

Active elements	$S_e \rightharpoonup$	Effort generation.
	$S_f \rightharpoonup$	Flow generation.
Passive elements	$\rightharpoonup R$	Energie dissipation node.
	$\rightharpoonup I$	Effort storage node
	$\rightharpoonup C$	Flow storage node.
Sensors	$\rightarrow D_f$	Flow sensor.
	$\rightarrow D_e$	Effort sensor.
Conversion elements	$\overset{1}{\rightharpoonup} \overset{m}{TF} \overset{2}{\rightharpoonup}$	Energy transformation implying : $e_1 = m\,e_2$; $f_2 = m\,f_1$
	$\overset{1}{\rightharpoonup} \overset{r}{GY} \overset{2}{\rightharpoonup}$	Energy transformation implying $e_1 = r\,f_2$; $e_2 = r\,f_1$

B. The junctions

Junctions (tab 3) are used to associate those elementary components. They transmit the energy instantaneously. They must co nnect a number of arrows higher than 1.

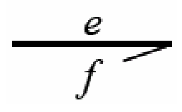
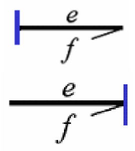
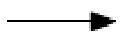
TABLE 3
THE TWO KINDS OF JUNCTIONS

Jonction 0 : all efforts are equal ***Ex:*** Parallel connection in electrics.		$f_1 + f_2 - f_3 = 0$ $e_2 = e_1 = e_3$
Jonction 1 : all flows are equal ***Ex :*** Series connection in electrics.		$f_1 = f_2 = f_3$ $e_1 - e_2 - e_3 = 0$

C. The arrows or bonds

In Bond Graphs there are two types of bonds (Tab 4). The first shows an informational transfer and the second shows an energetic transfer. The first one is represented by a full headed unidirectional arrow. The second, by a semi-headed unidirectional arrow that are assigned a number for identification. In addition to this, in the case of a causal Bond Graph, a vertical line is added on one of the extremities of the arrow. This causal stroke indicates the direction in which the effort bond is directed which is the opposite of the direction of the flow ([9] pg 25).

TABLE 4
BOND GRAPH BONDS/ARROWS

Energetic transfer		Informational transfer
Without causality	With causality	
		

IV. MAPPING TO SYSML/UML2.0 CONSTRUCTS

A. *Candidate diagrams*

It is important to choose a diagram that will be easily recognisable as a Bond Graph diagram to limit the learning effort of systems engineers. In the other hand, it is essential to respect the semantics of the UML diagrams. For example, a final state in a UML statechart is defined as a state that cannot have a state activity (no DoActivity attribute [UML superstructure page 462]). This constraint must be respected for any extension added to the original diagram.

In SysML the diagrams used to describe behaviour are: Use cases, interaction diagrams, parametric constraints diagram, statecharts and activity diagrams.

Use cases cannot express control nodes and are used to express top level system requirements. They cannot express energy or information transfer. Interactions are to be avoided because of the life line representation of objects which is too much far from Bond Graphs. Parametric constraints are used to associate objects properties to express mathematical relations between physical variables which may be useful when we'll need to extract the system's equations. It is not useful at this stage. Statecharts can only represent control flow and not object flow.

Activity diagrams are the most appropriate view because they use constructs that express object transfers and control between actions. They combine system's composition (in term of actions executed) with communication and sequencing between these actions.

It can be useful to compare this proposal with a representation of Bond Graphs that uses diagram assemblies. They may also be considered as an alternative. In fact, they depict a system as a collection of components with specific roles. They also show connections between subsystems inside the hole system ([1] page 49).

It seems that the assembly diagram is closer to Bond Graphs than activity diagrams. In fact assemblies are used to depict the system's composition in a static way. This is the opposite of activity diagrams that are intended to depict a sequence of actions. Of course we can "just not take into account" this actions sequencing, but there is still a risk of misunderstanding of Bond Graphs when expressed by activity diagrams. It is clearly not natural for a systems engineer to use activities one time for sequencing and the other time for a static representation. In the other hand the assembly diagram is composed of very general constructs, it also doesn't have control nodes which is useful to depict junctions.

B. *Mapping Bond Graphs to activity diagrams*

1) Elementary components or nodes: We use the « Action » construct to represent nodes that have an energetic function.

In Bond Graph formalism, systems are decomposed until we obtain subsystems that can be assimilated to an elementary-energetic phenomenon represented by one of the nodes described in the Bond Graph introduction paragraph. We can then assimilate an elementary subsystem to an action as it is defined in the UML 2.0 specification: It is the fundamental element of behaviour specification. It accepts a set of inputs and converts it into a set of output."[UML Spec: Def ACTIONS page 203]

An *Action* is defined as an abstract class. Then we have to inherit from it. We must remember that one of the advantages of SysML is that it will benefit from UML tools. This is why we cannot introduce contradictions between the SysML meta-model and the meta-model of UML2.0. Fig 1.

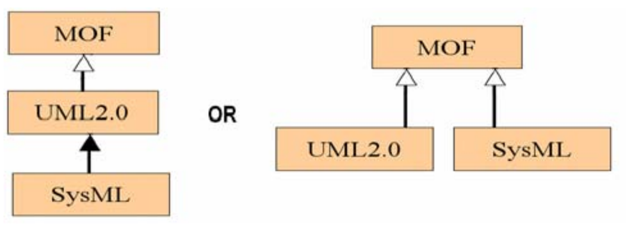


Fig1.SysML : Profile of UML2.0 or new meta-model.

In order to respect this constraint we can use two different solutions. First, we can add another child class to *Action* (Fig 2) and call it BondGraphAction. Second we can use one of the existing child actions and map it to a Bond Graph node.

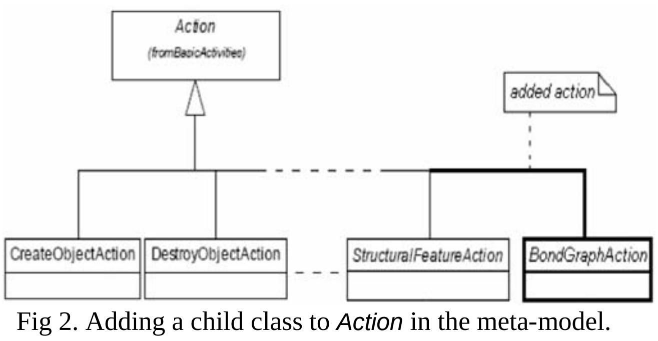


Fig 2. Adding a child class to *Action* in the meta-model.

The UML meta-model is usually not open source. The second solution seems then to be more easily realisable. This is why we use the ApplyFunctionAction meta-class to represent a Bond Graph action. ApplyFunctionAction is composed of a set of inputs, a set of outputs, and is associated to a function. Fig 3. (UML superstructure specification [3] page 207).

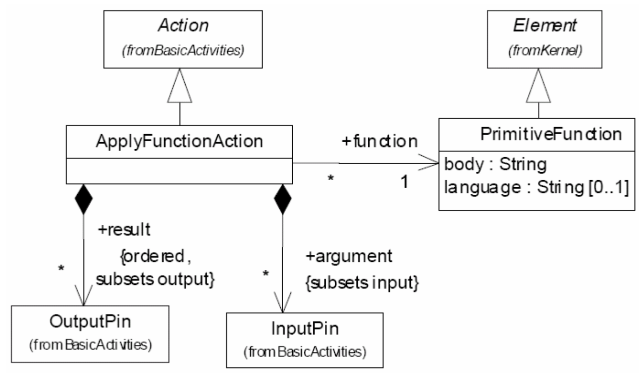


Fig3. ApplyFunctionAction's meta-model.

An ApplyFunctionAction is defined [UML specification page 222] as an action that invokes a predefined function that uses inputs to compute outputs.

To use ApplyFunctionAction we can use the inherited attribute "effect" [UML specification page 280] that is defined as being a description of the function executed by the action. But as in our case the number of possible functions is limited, the most appropriate solution is to specialize ApplyFunctionAction for each function. But we do not use an inheritance of classes, we use stereotypes. Stereotypes are used to add semantics to a metaclass. Its advantage is that it is accessible in the UML tools. In this Fig 4 we show the stereotypes created for the Bond Graph elements. The inheritance arrow used is full-headed to express extension on the meta-model, see [UML infrastructure page 167].

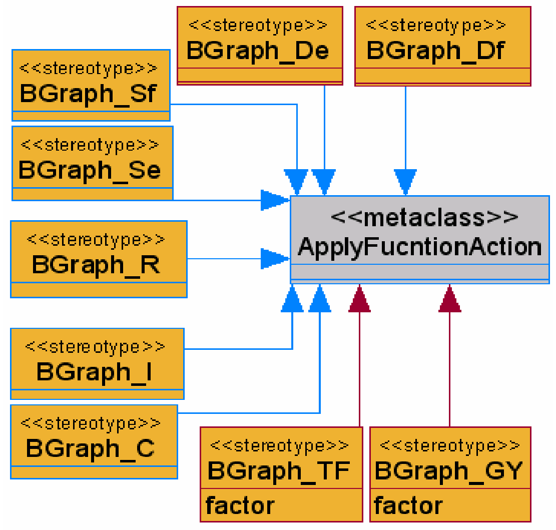


Fig 4 .ApplyFunctionAction stereotypes.

This way we can express on the same diagram, the three levels of Bond Graphs for better understanding. We can also use the PrimitiveFunction object Fig 3, to express the mathematic formula that goes with the element. This formula can be used later for system's equations generation.

In Fig 5 we show an example of application.

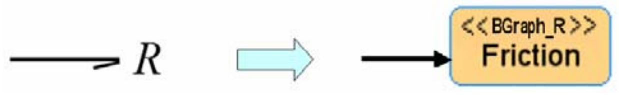


Fig 5. Mapping solution adopted using stereotypes.

2) Junctions: The usage of the Bond Graph junctions correspond to the control nodes in the activity diagram. In the meta-model, control nodes inherit from the abstract class *ControlNode.* To represent the two junctions of BGs that are junction 0 and junction 1, we have to inherit two new nodes of control Fig 6, 7.

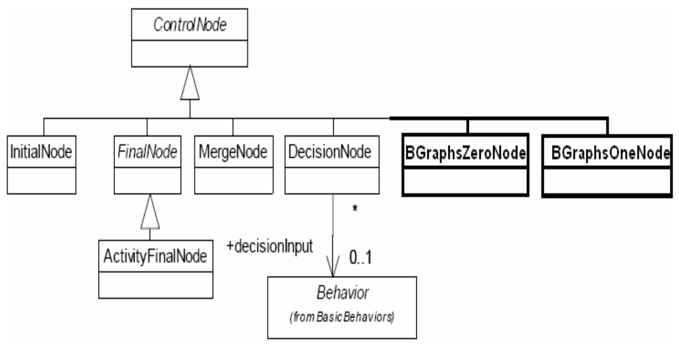


Fig 6. ControlNode hierachy

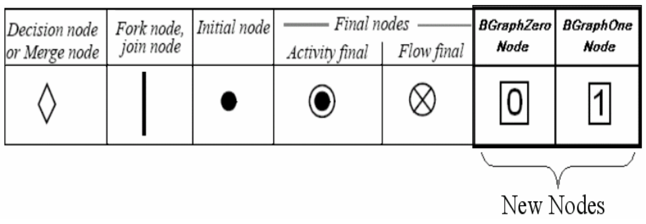


Fig 7. ControlNode notations

We can also use together the decision node and the merge node represented by a diamond for both junction 0 and junction 1. This can be done by stereotyping those nodes Fig 8. In fact the merge node and decision node can be used in the same diagram element[UML superstructure specification page 344]. This is the better solution because it prevents us from accessing the metamodel, we just use the extension mechanisms.

In Fig 8, we use a Bgraph_junction abstract class to inherit from both MergeNode and DecisionNode. Then we stereotype this node to BG_0 and BG_1. Both resulting nodes will be drawn using the diamond.

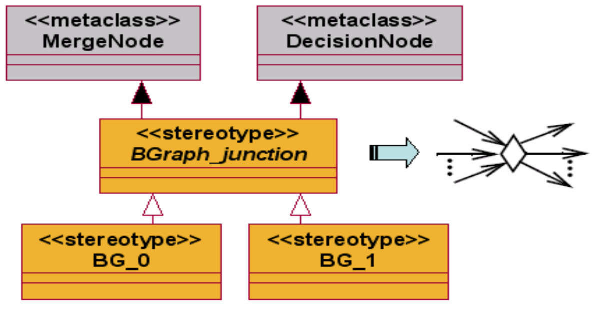


Fig 8. Bond Graph nodes hierarchy.

3) The edges or bonds: Two different edges are defined. We use the stereotypes/tagged values mechanism to define these child classes of the UML 2.0 ObjectFlow class. The black arrow in Fig 9 says it is an extension to the metaclass ObjectFlow. The attributes of the new defined classes are also called tagged values. The causality attribute/tagged value is of type CausalityType which is an enumeration (start, end, nonCausal).

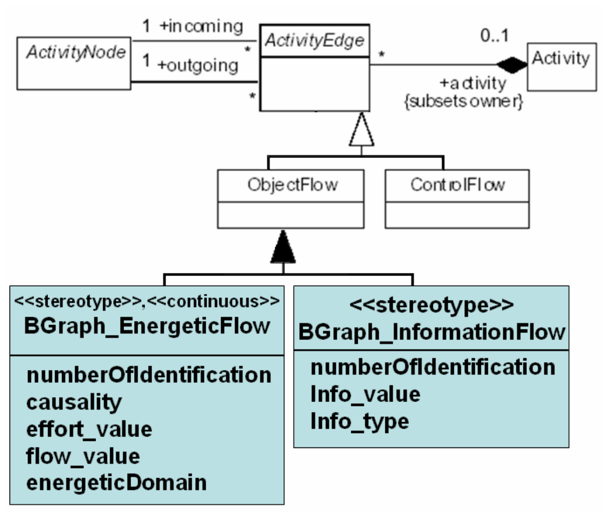


Fig 9. ObjectFlow stereotypes for Bond Graphs.

The energetic transfer is done through an energetic port in BGs. They are represented by pins Fig 10

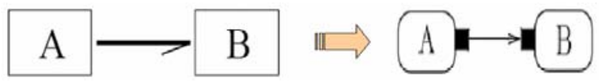


Fig 10. Energy bond representation.

Object flow of activity diagrams in UML 2.0 cannot associate two actions directly. This is why we use pins. Pin is an abstract class, so we use OutputPin or InputPin. In our case they are shown as black little squares to express that it is a streaming port. This means that the isStream attribute is set to True and consequently, the isException attribute is set to False Fig 11.

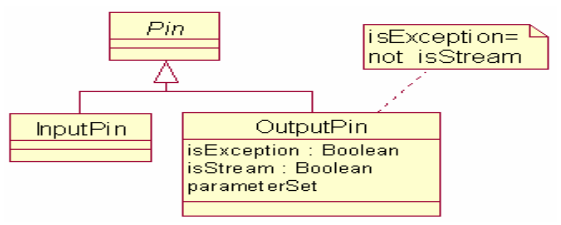


Fig 11. Pins hierarchy.

In the "UML 2.0 superstructure specification" [1] page 352 we can read "Parameters are extended in complete activities to add support for streaming, exceptions, and parameter sets". A specific ObjectNode called ActivityParameterNode is defined to use this parameter (Fig 12) so that an ObjectNode can support streaminf, exceptions and parameter sets. We can also read in page 355 that OutputPins are used with an annotation text {steam} to show streaming (or the little black square Fig 10) and has attributes that can express exceptions and parameter sets. So there are two ways to express streaming for object nodes. This is probably a redundancy in UML2.0.

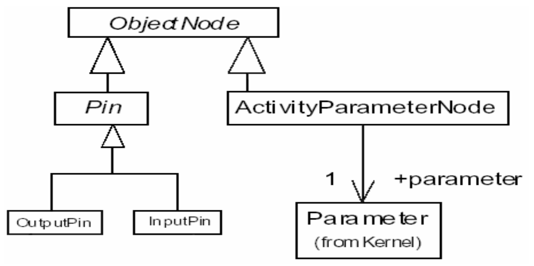


Fig 12. ActivityParameterNode use of the Parameter construct

4) Constraints on the defined elements: For each element a list of constraints need to be defined so that we can express correct bond graphs with this extension. As an example we present here the constraints that come from the causality concept used in bond graphs:

-Junction 0: Only one causality sign close to the junction. (sum(causality=end of entering flows),(causality=start for leaving flows))=1.

-Junction 1: Only one EnergeticFlow whitout a causality signs next to the junction (sum(causality=end of entering flows),(causality=start for leaving flows))=number of flows-1.

-De,Df : No causality (ideal sensors) De can only be connected to 0-junction. Df can only be connected to 1-junction.

-EnergeticFlow from an Se element: causality=end.

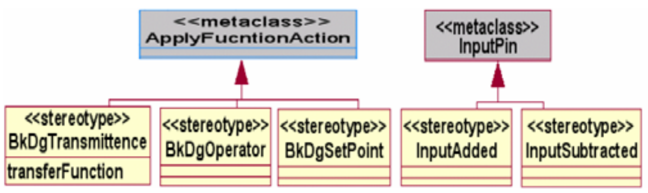
-EnergeticFlow from an Sf element: causality=start.

-Tf element: Both EnergeticFlows with causality=start Or both EnergeticFlows with causality=End.

-Gy element: One EnergeticFlow with causality=end, the other causality=start.

5) Block diagram extension

In order to use this BG representation, we need to add the elements that correspond to the Block Diagram elements. In fact, block diagram is used conjointly with BGs to depict the control part of the system.


<<metaclass>> ApplyFucntionAction
<<metaclass>> InputPin
<<stereotype>> BkDgTransmittence
transferFunction
<<stereotype>> BkDgOperator
<<stereotype>> BkDgSetPoint
<<stereotype>> InputAdded
<<stereotype>> InputSubtracted
Fig 13. Block diagram extensions.

V. Formalism Application

In this section we are showing an example of usage of our activity-bond graph. We describe this simple servo system (Fig 14) with causal bond graphs then with the SysML activity bond graphs.

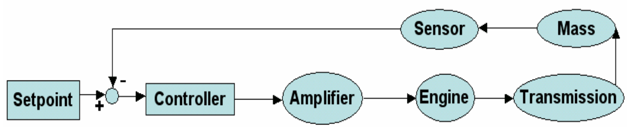


Fig 14. Word-bond graph/Block diagram of the system used as example.

The resulting causal bond graph representation is shown in the following Fig 15. The control sub-system is a block diagram, we only map the physical sub-system with the activity diagram.

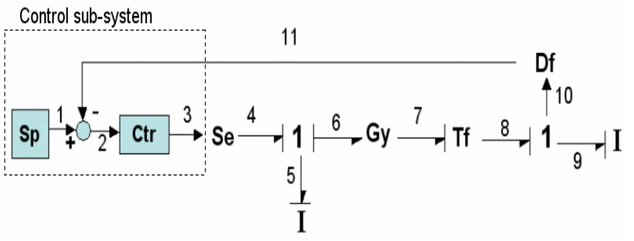
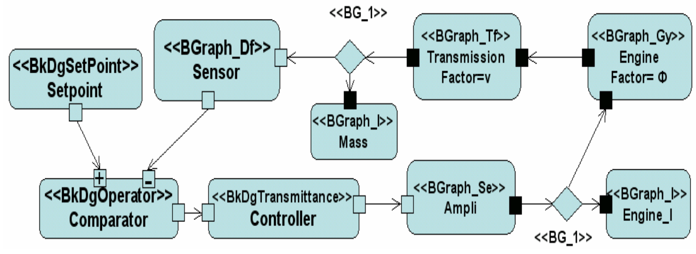


Fig 15. Associated Bond Graph.

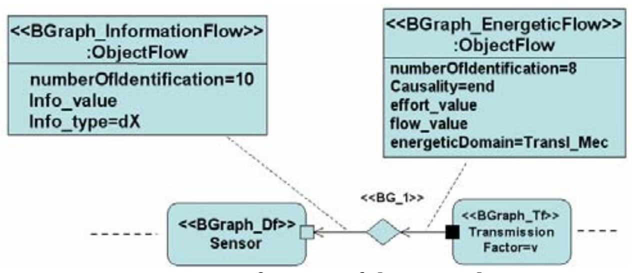
This causal bond graph will be represented in SysML by Fig 16. The flow number 4 is described as an energetic flow stereotyped Bgraph_EnergeticFlow. Its causality is set to the end position and its energetic domain is set to electricity which is one of the values of energeticDomainType enumeration.



<<BkDgSetPoint>> Setpoint
<<BGraph_Df>> Sensor
<<BG_1>>
<<BGraph_Tf>> Transmission Factor=v
<<BGraph_Gy>> Engine Factor= Φ
<<BGraph_I>> Mass
<<BkDgOperator>> Comparator
<<BkDgTransmittance>> Controller
<<BGraph_Se>> Ampli
<<BGraph_I>> Engine_I
<<BG_1>>

Fig 16. Activity representation of the system.

We depict in the following diagram Fig 17 the specification of two bonds. Bond number 1 and bond number 10. The first one is an energeticFlow and the second is an informationalFlow.


<<BGraph_InformationFlow>>
:ObjectFlow
numberOfIdentification=10
Info_value
Info_type=dX

<<BGraph_EnergeticFlow>>
:ObjectFlow
numberOfIdentification=8
Causality=end
effort_value
flow_value
energeticDomain=Transl_Mec

Fig 17. Specification of diagram elements.

VI. CONCLUSION

The activity diagrams are a good formalism to express Bond Graphs. The extensions that are necessary for activities to comply with Bond Graphs are not heavy.

We only used classical extension mechanisms such as stereotypes, tagged values and constraints to define this activities extension. This is why SysML users can easily integrate Bond Graphs to their design environment be it rational rose, objecteering, rhapsody, Poseidon or any other tool that supports usual extension mechanisms which is a considered as a basic feature of CASE tools (Computer-Aided Software Engineering Environments).

VII. PERSPECTIVES

The Bond Graphs/Block diagrams composition can combine the command part with the physical part of the system. This can be an important extension to SysML that may be an argument to convince systems engineers to use this language that wants to be the de facto standard.

One of the advantages of bond graphs is that it can be used to extract the system equations to be simulated. We could generate these equations from our activity diagram and describe them using the parametric diagram included in SysML.

We can also use other simulation tools. One possibility is the generation of Modelica code. Modelica is an object-oriented modelling language with a textual definition to describe physical systems. A Bond Graph library was written in Modelica. In addition to that, Modelica code was generated from Bond Graphs [7, 10].

We need to continuously keep this proposal up to date as both UML 2.0 and SysML specifications are still evolving. We are expecting a new version of SysML to be released by the end of 2005.

REFERENCES

[1] SysML specification V0.9. 10 january 2005.
[2] UML for systems engineering RFP, OMG Document: ad/03-03-41.
[3] UML 2.0 Superstructure Draft Adopted Specification. OMG Adopted Specification ptc/03-08-02.
[4] UML 2.0 infrastructure specification, OMG Adopted Specification ptc/03-09-15.
[5] G.Gandanegara, « Méthodologie de conception systémique en Génie Electrique à l'aide de l'outil Bond Graph Application à une chaîne de traction ferroviaire », PhD report for « Institut National Polytechnique de Toulouse», page 5, 2003.
[6] J. Zaytoon, "Systèmes dynamiques Hybrides", page 94, 2001.
[7] J.F. Broenink,, "Object-oriented modelling with bond graphs and modelica". *International conference on bond graph modelling ICBGM'99, Simulation series Vol 31 no 1,SCS, page 163-168.*
[8] I. Bailey, F. Dandashi, H. Ang, D. Hardy, "Using Systems Engineering Standards In an Architecture Framework", white paper eurostep company.
[9] D.C. Karnopp, D.L.Margolis, R.C.Rosenberg, "System dynamics – Modeling and Simulation of Mechatronic Systems", third edition 2000.
[10] W.Borutzky, "Object-oriented Modeling of Mechatronic Systems Using Bond Graphs", proceedings of the IASTED international conference in Applied Modelling and Simulation, September 1999.
[11] W.Borutzky, Relations between graph based and object-oriented physical systems modelling, ICBGM'99 International Conference on Bond Graph Modelling and Simulation, San Fransisco, CA, Jan. 17-20, 1999, pp.11-17.