Issues of Hierarchical Heterogeneous Modeling in Component Reusability

Mokhoo Mbobi
Mokhoo.Mbobi@supelec.fr

Frédéric Boulanger
Frederic.Boulanger@supelec.fr

Mohamed Feredj
Mohamed.Feredj@supelec.fr

Supélec - Computer Science Department
Plateau de Moulon, 3 rue Joliot-Curie
91192 Gif-sur-Yvette cedex, France

Abstract

Heterogeneous systems are systems that obey different functioning laws. For instance, during the design of embedded systems, it is generally necessary to study both the controller and the environment that it controls, these two subsystems being clearly different in nature. Moreover, data processing applications are also increasingly heterogeneous, mixing different technical domains such as telecommunications, man-machine interface, analog and digital electronic, signal processing algorithms. To combine these different technology domains, modeling languages and platforms generally use a hierarchical approach.

This paper highlights how the hierarchy of the model and the changes of model of computation are coupled and why this coupling forbids the use of components that have inputs or outputs that obey different models of computation. In addition, this paper shows that what happens when data crosses the boundary between two domains depends on the modeling environment and it gives some means of managing component in the same level of the hierarchy.

1 Introduction

1.1 Economic and scientific modularity stakes

The latest studies [19] [2] had highlighted the increase and segmentation of embedded systems market, and the related mentioned reasons are durable. So, new kinds of embedded systems will appear, existing systems will change, the services around them will develop and the number of familiar objects containing an embedded processor will increase in a continuous way in the future.

To master this unceasingly growing market with a short increasingly time-to-market, researchers and equipment suppliers are required to respectively deal with technical and economic constraints.

According to the economic constraints, economic choices must be judiciously made in order to maintain the balance between the average price of the embedded components that is increasingly weak and their performances that increase exponentially. For example, today, a cellular telephone able to modify its ringing by downloading a polyphonic musical sequence, to check the schedule of plane, to film video sequences and to send them by electronic mail costs less than one tenth of the price of its ancestor in the first GSM generation whose the functioning was limited to the simple communication. Therefore, the great economic challenge for equipment suppliers will remain the mastery of the ambivalence "increase in performance and drop in prices".

On looking at both the economic constraints and the market trends of embedded systems the "design-development-production" cycle of components must be shortened. That imposes a major change towards the distributed and industrial production modes, promoting more and more modular applications containing reusable components and that are easily maintainable. Thus, in [20], the author affirms that the development of embedded systems follows a process shared by several actors, implementing a co-operation between equipment suppliers and manufacturers and that the concept of components reuse is a strong argument to reduce the study and factory costs. Bringing another economic argument, in [12], the author confirms that the use of modular methodology involves an investment that is amortized only on the long term, after reuse.

According to the technical constraints, they are intrinsic with the embedded system and are taken into account in its different design phases. These constraints are at the functional and operational level and are closely related to the essence of the embedded system. Their taking into account allows the embedded system to guarantee its operational functioning in continuous reactivity with its immediate environment in a sure and safe made way. Such as the pacemaker whose electrodes are well introduce inside a human heart with that it interacts to control its beats in contin-

uous reactivity and in a sure and safe made way. The design of such a systems requires a multidisciplinary of scientific knowledge, from where the need of several specialists in different domains. Moreover, since the cycle of life of some systems is very long compared to the components used for their manufacturing, the impact of degradations and obsolescence in these systems must be limited. Finally, this double statement implies to take into account the solutions that use modular architectural techniques. According to the hierarchical heterogeneous approach, it couples the change of hierarchical level with the changes of model of computation on the modeling. This corrupts the modularity, reduce the reusability of the components and reduce also the maintainability of the system.

1.2 Modularity, Reusability, Maintainability

Systems must be designed in a modular way. They must be elementary or composite components assembled according to a well-defined communication diagram that implements a communication and interfaces syntheses. In [5], the author supports this concept by specifying that currently, the multi partner character of industrial projects requires the modular development capacities, in particularly to be able to separately and independently compile each application component in the form of executable processes, software or hardware, then to integrate them within a final architecture. In the same way, in [3], the author specifies that the reuse is intrinsically a phenomenon arising at the construction phase. It implies the modification of data and algorithms structures that are in the components, and that remain fixed during the execution. Relaying the above ideas, in his development presented in [11], the author shows that the models as their building blocks must be reusable and as autonomous as possible . Then, it specifies that an approach must not only allow expressing the essential properties of a system, but must also guarantee an easy maintainability of the models. Moreover, the evolution of the models must follow that of the embedded systems without calling in question all of what already exists. Consequently, a design approach must provide effective mechanisms of reusability that allows to add, to remove or to specify the model elements without calling into question all the description of the system. Whether the stakes are economic or scientist, the modularity in the embedded systems supports the reusability and improves the maintainability. Moreover the increase of the reusability of the components implies that of the productivity of the designer. This is why, modularity and reusability are currently perceived like a strong arguments in the design of the components. So, embedded system must be designed in a modular way. It consist of the interconnection of several different modules where arise embedded systems that contain other embedded systems.

2. Heterogeneous Modeling

In [4], modeling is defined as a formal representation of a given concept, system or subset whereas the design generally implies the implementation of several successive models, each one being a refinement of the precedent. The first model can be seen as the formal system specification, and the last model can be seen as its implementation. So, the goal of the modeling is the exploration of the models for a final design when the goal of the design remains the implementation [18]. Design and modeling are obviously closely interdependent.

On an abstract level, a model of a system can be regarded as a combination of different technical domains that have different methods of modeling and design. These different domains have different methods of modeling and design that consider their components and their relationship in different ways. Consequently, in each domain, the interactions between components are governed by a specific set of physical law called "Model of Computation-(MoC)". A comparative and detailed study of models of computation is presented in [13].

Currently, the modeling and design of complex systems calls naturally for the use of several models of computation that correspond to the different technical implementations. To illustrate this heterogeneity, we consider the simple example of a third generation multi-media cellular telephone shown in the figure 1. It uses several technical domains of that algorithms of image and signal processing, software, physical interfaces, micro-waves and radio features, opto-electronic, electronic and electricity techniques, networks, etc...

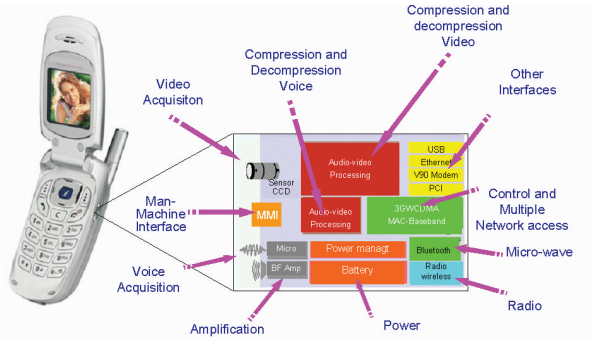

Figure 1. An example of heterogeneity

The organization of such a system and the interactions between its different subsystems imply a connection of the subsystems that are not already using the same model of computation. Such a system that uses different models of computation is called *"Heterogeneous System"*.

Heterogeneous modeling is simply a modeling by using a number of MoCs. Since most systems are heterogeneous in nature, heterogeneous modeling provides more natural and more complete models. For instance, being able to use both state machines and synchronous data-flows allows to describing explicitly the control in the model of a digital signal processing system. If we were limited to the synchronous data flow MoC, control and data processing would have to be coded together and the model would be less expressive and much more difficult to maintain [6].

Current heterogeneous modeling tools allow to take into account and to structure diversity of applications domains. In order to mix different MoCs, each of existing modeling tools can use either an amorphous or a hierarchical heterogeneity approach.

Although being largely used, the hierarchical approach presents some disadvantages we present in this paper.

3. Heterogeneous Modeling Approaches

3.1 Amorphous Approach

Many modeling and design environments support heterogeneous modeling, but they generally focus on a fixed set of MoCs that are generally continuous and discrete signals for electrical engineering or state machines and differential equations for hybrid systems. Since they use few MoCs that are known beforehand, they can define the union of these MoCs, and the total knowledge of the interactions between these MoCs allows to compute the behavior of a heterogeneous model. This approach is called *"Amorphous Approach"*. A digital to analogical signal converter with digital inputs and analog output is an example of this system. SIMULINK and VHDL-AMS are the examples of this modeling and design environments.

3.2 Hierarchical Approach

Modeling and design tools that support an open set of MoCs cannot build the union of these MoCs because of their high number that moreover are not known beforehand. These tools require that each component obeys only one MoC. Since components that are connected obey the same MoC, all the components that are interconnected must obey the same MoC. However, the hierarchical abstraction makes it possible to use a MoC to model a component that is different from the outer MoC in which the component is used. Therefore changes of MoC can only occur at the boundary of a component: this leads to the *"hierarchical heterogeneous modeling"* paradigm used by several modeling and design environments such as el Greco [7], PTOLEMY II [4] etc... .

This hierarchical approach is obviously an efficient way of managing the complexity of a system [14] [15] [10] [18] [6]. This complexity can come from the structure or the behavior of the system; from where the structural hierarchy called also architectural hierarchy and the behavioral hierarchy [1].

Structural hierarchy is used to identify self-contained sub-systems and to define their interface to the other subsystems [16]. This interface may consist of a set of signals. At a higher abstraction level, it may consist of the communication channels that encapsulate more complex communication protocols. This hierarchy allows to design each part of a system in a separate way once the interfaces have been specified.

Behavioral hierarchy is a way of considering a complex process as the result of sequential or concurrent simpler processes. The composition of processes makes use of communication and synchronization primitives such as termination detection, process activation or suspension, rendezvous or exceptions. Exception handling may be considered as behavioral hierarchy since it can be seen as the termination of a process and the activation of the exception handling process [8]. There is also another type of hierarchy called "synchronization hierarchy" that can be considered as a special case of behavioral hierarchy.

The fundamental question is not this hierarchy in itself, it is rather the impact of its coupling with the changes of MoC on the modeling, the design and the maintenance of the applications. This corrupts the modularity, reduce the reusability of the components and reduce the maintainability of the system. We rather think that the hierarchy in a heterogeneous model should not depend on the modeling tools. It should rather represent the compositional structure of a system following its functional decomposability.

In the next section, we present the different issues of this hierarchical approach.

4 Issues of hierarchical approach

Hierarchy manages the complexity of a system by hiding internal details that are not pertinent at a given level of modeling [16] [17] [6].

When you look inside a component, you may either see a low level description of the behavior of the component when the component is atomic or primitive, or you may see a model of this component in the same modeling environment when the component is composite. In both cases, the interface of the component hides its hierarchical level to the internal details of its behavior. So the inner and the outer MoCs can be different. It is still necessary to define how the inner and the outer MoC interact and how data is transformed when crossing a domain boundary [18] [6].

Currently, different modeling tools largely used the hierarchical approach in order to make easier the interactions between two heterogeneous components. But, this approach has some drawbacks which are in the root of the issues that we present bellow:

- the hierarchy of the model is coupled with the changes of MoC and may not reflect the effective structure of the system;
- components that have inputs or outputs that obey different MoCs cannot be used;
- what happens when data crosses the boundary between two domains depends on the modeling environment

4.1 First issue

From the coupling between the hierarchy and the change of MoC arise ad hoc constructions at the boundary between two models of computation. This issue can be solved either by preserving the semantics properties or by changing the semantic properties across models of computation.

4.1.1 Preserving semantic properties across MoCs

Here, only terminals that obey the same MoC can be connected, but a component may obey several models of computation. This way preserves the semantic properties along connection between terminals.

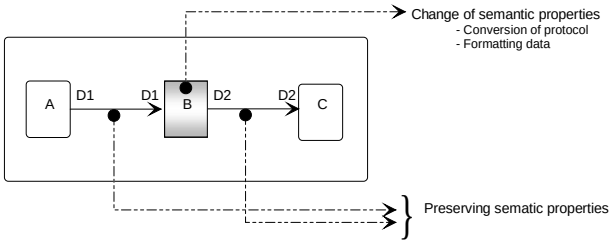

Figure 2. Component obey several MoCs

Since component can obey several models of computation, we can use a third component dedicated to the change of semantics between two heterogeneous components in the same hierarchical level as shown in figure 2. These components are called "Heterogeneous Interface Components, (HIC)". Then, the heterogeneous behavior of the system can occur inside those components as part of their behavior and is therefore an explicit part of the model of the system.

4.1.2 Change of semantic properties across MoCs

Here, components obey only one MoC, but we can make connections between terminals across MoCs as shown in figure 3.

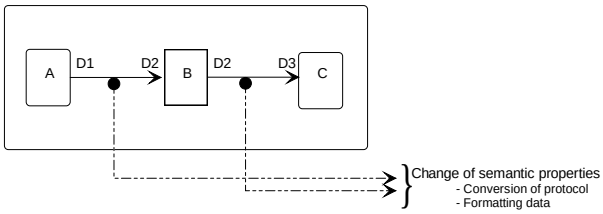

Figure 3. Component obey only one MoC

To achieve this way, the change of semantic properties mechanism between MoCs must be implement either in the core or in the ends of the connection. But to implement change of semantic properties mechanism between MoCs in the core of the connection requires from this connection to become active, so that it must be able to provide both the conversion of communication protocol and the data formatting mechanisms as shown in figure 3.

The challenge is how to do so that from one model of computation to another, the connection is able to achieve the three conditions below, presented in [21] :

1. to translate the common semantic property
2. to ignore the semantic properties in the first MoC that are not present in the target MoC
3. to create the semantic properties in the target MoC that are not present in the first MoC

In the same way, to implement change of semantic properties mechanism between MoCs in the end of the connection requires from this terminal to become active.

To achieve this, some approaches use the mechanism of Remote Procedure Call (RPC) in that a component can call a communication primitive of a connection by reading from or by writing on the interface port. As shown in figure 4, the

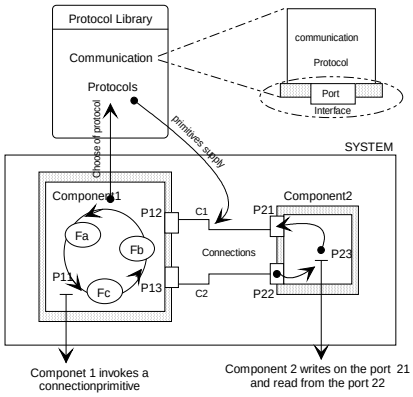

Figure 4. Using RPCs

component M1 choices the communication protocol in the protocols library. Since it can have several implementations

in this library, the synthesis process choices the appropriate protocol to the MoC used by the components M1 and M2 and provides the necessary primitive for the connection.

In object-oriented philosophy, this concept of protocols library is often replaced by the power of the *polymorphism* technique. Note that in PTOLEMY II for instance, this concept is replaced by the dual concept of *polymorphism and hierarchical abstraction* but on different hierarchical levels.

4.2 Second issue

The second issue of this approach forbids the use of components that have terminals that obey different MoCs in the same hierarchical level. For instance, an analogical to digital converter could be modeled in a continuous time domain, the digital outputs being considered as continuous signals with sharp changes. If such a model is close to the reality, it is not at the right abstraction level when you want to consider the outputs as discrete sequences of values. On the contrary, the analogical to digital converter could be modeled using a discrete domain where the continuous inputs would be considered as sequences of discrete samples, turning the device into a resampler or a no-op. To solve this problem, a Heterogeneous Interface Component must be used.

4.3 Third issue

The third issue hides the transformations that occur at the boundary of two domains inside the edge of the components. These transformations depend on the modeling tool and are therefore not explicitly stated in the model of a system. And the designer of the system has neither the clear comprehensibility nor the complete control of what happens when data crosses the boundary between two domains. So, he has not the complete control of the heterogeneous behavior of its system. To solve this problem, two approaches may be used :

- The first approach advocates to allow the designer to edit the edge of the components to specify how data is transformed when it goes through it. If the heterogeneous mode is in the same level of the hierarchy, these edge looks like to the “HIC” presented in section 4.1.1
- The second approach advocates to move these transformations from the edge to the core of the components. This makes the internal specification of the component depend on the domain in that it is used, what impairs modularity and reusability.

4.4 Example

Consider the simple example given in [6]Mbobi4 and shown in figure 5. A signal rectifier illustrates the issue of the hierarchical heterogeneous models. The top level uses flows of data samples, and the behavior of the detector is modeled using discrete events. When the flow of samples enters the detector, it is converted to a sequence of valued events. When an event is produced at the output, its value is used to build a data sample in the outer domain. This is only an example of what may happen at the boundary of a component, and the important point is that these transformations depend on the modeling tool and are not specified in the model of the system. Since the data flow MoC in that the detector is used requires that a sample of data be produced on the output each time a sample is consumed on the input, the discrete event behavior of the detector must respect this condition. So even if the input signal does not change its sign and no event has to be produced, the detector must produce something on its output to obey the semantics of the outer domain. Here, we have put a sampler that uses the value of the last emitted event to produce an output each time an input sample is consumed. We have to put this sampler in the internal model of the detector because of the semantics of the external MoC, so the implementation of the detector depends on the context in that it is used, what impairs modularity and reuse.

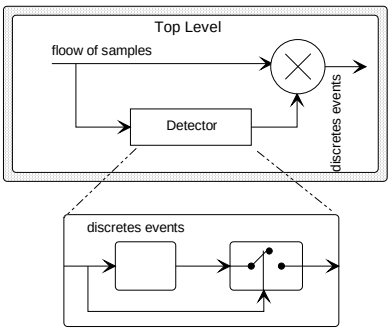

Figure 5. Example of implicit transformation

5. Conclusion

To support heterogeneity arising from the use of different technical domains, modeling hierarchically tools nest different models of computation that characterize these different technical domains.

In this paper, we have discussed the issues that we identify in hierarchical heterogeneous approach . These approach lead to the coupling between hierarchy of the model and the changes of model of computation. So that the components that have inputs or outputs that obey different models of computation cannot be used and what happens when data crosses the boundary between two domains cannot be explicitly specified. In addition, we gave some strategies of managing component in the same level of the hierarchy.

References

[1] R. Alur, T. Dang, J. Esposito, R. Fierro, Y. Hur, F. Ivancic, V. Kumar, I. Lee, P. Mishra, G. Pappas, and O. Sokolsky, *"Hierarchical Hybrid Modeling of Embedded Systems"*, University of Pennsylvania, Available online at http://www.seas.upenn.edu/hybrid/

[2] M. Auguin, *"Systèmes sur Puce : vers l'exploration en milieu complexe"*, Ecole Thématique sur les Systèmes Enfouis, I3S, Université de Nice Sophia Antipolis, CNRS, INRIA, novembre 2003

[3] P.G. Basset, *"The theory of practice of adaptive reusse"*, In SSR, page 2-9, 1997.

[4] S.S. Bhattacharyya et al, *"Heterogeneous Concurrent Modeling and Design in java, Volume I to III"*, Memorandum UCB/ERL M01/12 eecs, University of California at Berkeley, March, 2001

[5] F. Boniol, *"Une approche synchrone multi-formalismes pour la conception de systèmes temps-réel distribués"*, Technique et science informatiques, Hermès, volume 17, n9/1998, pages 1099-1128.

[6] F. Boulanger, M. Mbobi and M. Feredj, *"Flat Heterogeneous Modeling"*, Proceedings of the conference of Internet Processing Systems Interdisciplinaries, Venice, Italy, November 10 to 15, 2004

[7] J. Buck and R. Vaidyanathan. *"Heterogenous modeling and simulation of embedded systems in el Greco"*, Proceedings of the 8th international workshop on Hardware/software codesign, San Diego, California, USA, Pages: 142-146, 2000, ISBN:1-58113-268-9.

[8] J.M. Daveau, *"Spécification système et synthèse de la communication pour le Co-Design Logiciel/Matériel"*, Ph.D. Thesis INPG, TIMA Laboratory, Decembre, 1997

[9] F. Vahid and D. Gajski, "Specification Partitioning For System Design", Proceedings of the IEEE Design Automation Conf., pages 219-224, June, 1992.

[10] A. Girault, B. Lee, and E. A. Lee, Fellow, IEEE, *"Hierarchical Finite State Machines with Multiple Concurrency Models"*, Proceedings of the DATE99 conference, pp.382-383, March99.

[11] R. Hamouche, *"Modélisation des systèmes embarqués base de composants et d'aspects"*, Ph.D. Thesis, Laboratoire des Méthodes Informatiques, Université d'Evry Val d'essone, Juin, 2004.

[12] L. Lagadec, *"Abstraction, modélisation et outils de CAO pour les architectures réconfigurables"*, Ph.D. Thesis, Université Renne 1, Décembre 2000.

[13] E.A. Lee and A. Sangiovanni-Vincentelli, *"A Framework for Comparing Models of Computation"*, IEEE Transactions on computer-aided design of integrated circuits and systems, Vol. 17, no. 12, December 1998.

[14] B. Lee and E.A. Lee, *"Hierarchical Concurrent Finite State Machines in Ptolemy"*, University of California at Berkeley, Proceeding of International Conference on Application of Concurrency to System Design, p. 34-40, Fukushima, Japan, March 1998.

[15] B. Lee and E. A. Lee, *"Interaction of Finite State Machines and Concurrency Models"*, University of California at Berkeley, Proceeding of Thirty Second Annual Asilomar Conference on Signals, Systems, and Computers, Pacific Grove, California, November 1998.

[16] M. Mbobi, F. Boulanger and M. Feredj, *"Non-hierarchical heterogeneity"*, International CCCT, IEEE Computer Society July-August, 2003, Orlando, FlorideUSA,. International IIS, Volume III, ISBN 980-6560-05-01, pp. 430-435.

[17] M. Mbobi, F. Boulanger and M. Feredj, *"Execution Model for Non-Hierarchical Heterogeneous Modeling"*, Proceedings of The 2004 IEEE International Conference on Information Reuse and Integration, November 8-10, 2004, Las Vegas, USA, IEEE, ISBN 2004113902, pages 139 144

[18] M. Mbobi, *"Modélisation Hétérogène Non-Hiérarchique"*, Ph.D. Thesis, Supélec, Université Paris XI (Orsay), Décembre, 2004.

[19] RNTL, Rapport du groupe de travail "système embarqué et temps réel, co-développement", 2001, Availlale on line at http://www.telecom.gouv.fr/rtnl

[20] F. Simonot-Lion, J.P. Elloy, Y. Trinquet. *"AIL_Transport : Un langage de description d'architecture électronique embarquée dans l'automobile"*, Veille Technologique, 45, juin, 2002, p. 34-36.

[21] W-T. Chang, S. Ha, and E.A. Lee, *"Heterogenous simulation - mixing discrete-event models with dataflow"*, Journal of VLSI Signal Processing 15, 127-144, Kluwer Academic Publishers, 1997.