Execution Model for Non-Hierarchical Heterogeneous Modeling

Aimé Mokhoo Mbobi, Frédéric Boulanger, Mohamed Feredj

Execution Model for Non-Hierarchical Heterogeneous Modeling

Mokhoo Mbobi
Mokhoo.Mbobi@supelec.fr

Frédéric Boulanger
Frederic.Boulanger@supelec.fr

Mohamed Feredj
Mohamed.Feredj@supelec.fr

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

Abstract

To combine different technology domains, modeling languages and platforms generally use a hierarchical approach. This approach avoids the combinatorial explosion of the number of interfaces between models of computation (MoC), but it forbids the use of components which have inputs or outputs that obey different MoCs. This affects the modularity and the reuse of the components. Moreover, the communication between two MoCs is implicitly defined in the modelling tool, and the designer does not have control over the semantics of what happens at the border between MoCs. In [7], we introduced a new approach that allows non-hierarchical heterogeneity based on heterogenous-interface components (HIC). In this paper, we present the architecture and the execution model we have designed to support HICs in Ptolemy II.

1. Introduction

The design of embedded systems is a complex task that makes use of numerous subsystems which belong to different technical domains. Those domains have different ways of modeling systems, so embedded systems are heterogenous in nature. As an example, consider the simple example on Figure 1 where a detector analyses a radio signal to produce an event when a mobile communication device should change from a base station to another. This detector is at the border between the analogous signal domain of its input and the discrete event domain of its output.

Heterogenous modeling is supported by many languages and modeling tools such as SpecC [11], SystemC [12], ROSETTA [10], POLIS [8], el Greco [4], PTOLEMY II [3] [5], etc. . . . However, these platforms either impose that a given level in the hierarchy uses a single MoC, or allow only a built-in set of MoCs. For instance one for analogous and the other for discrete signals.

These limitations avoid the combinatorial explosion of the number of interfaces between MoCs, but they do not allow the use of components that have inputs or outputs that obey different models. However, such components appear naturally in systems, as shown in our first example. Moreover, coupling heterogeneity and hierarchy impairs the modularity of the system model. Our goal is not to ban hierarchy from system modeling but to dissociate hierarchy and computation models. So, we think that the hierarchy in a heterogeneous model should not depend on the modeling tools, but, it should rather represent the structure of a system following its functional decomposability.

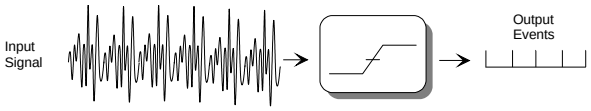

Figure 1. A simple flat heterogeneous system

The approach that we presented in [7] is based on heterogenous interface components which have inputs and outputs that obey different models and therefore allow the connection of sub-systems that behave according to different MoCs or domains. This approach allows the use of flat heterogenous models which have two main advantages: better modularity and facility of maintainability, which are important for distributed simulations for instance; and the possibility to define explicitly what happens at the boundary of several MoCs. The simulation of such flat heterogenous systems requires an execution model that is able to delegate the computation of the behavior of different components to their respective MoC, and to handle Heterogenous-Interface Components (HICs).

This paper presents such an execution model that we designed so that it does not need to know anything about the semantics of the underlying MoCs. We have implemented this execution model in Ptolemy II, and it supports all existing Ptolemy domains. It should also support any future domain because it is independent from their semantics.

2. Non-Hierarchical Architecture

Non-hierarchical heterogeneity allows the use of actors [1] [2] [6] that obey different MoCs at the same level in the hierarchy. This implies that a model can contain actors that communicate according to different semantics. In [7] it has been proposed an non-hierarchal heterogeneity approach which is based on two main components: the Heterogeneous Interface Component which manages the data flows and an heterogeneous execution model which handles the control. It has been that supporting HICs is the key to non-hierarchical heterogeneous modeling.

A non-hierarchical heterogeneous model may contain several domains at the same level of the hierarchy, each domain being in charge of the actors which use its MoC. Since a MoC defines the interactions and the communication between actors [9], a HIC will be handled by several domains which must be coordinated to take into account the fact that the reaction of a HIC according to a MoC may have consequences on its behavior according to other MoCs.

3. Heterogeneous Interface Components

An heterogeneous interface component is in charge of the management of the data flows between different MoCs. By designing the behavior of HICs, we are able to define the semantics of the communication between actors that obey different MoCs.

Since HICs are at the boundary of several MoCs, they must obey the semantics of several domains. However, to be usable with existing MoCs, they must behave exactly as actors of a MoC from the point of view of the corresponding domain. Everything that does not belong to a MoC must be masked when a HIC is considered as an actor of this MoC. This is what we call “projecting a HIC on a MoC”. A HIC

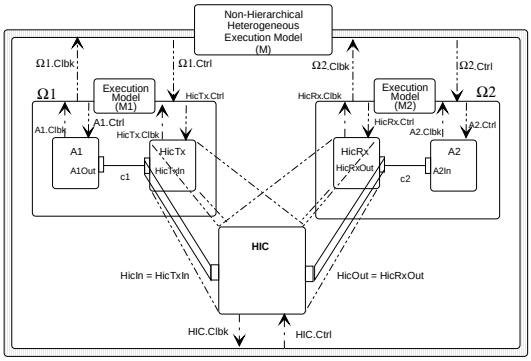

Figure 2. A HIC projected on two domains

is controlled by several execution models that correspond to different MoCs, and by the heterogeneous execution model. The HIC is projected on the various domains and there, each projection obeys the local semantics. The homogeneous behavior and the communication tasks are managed by the projections of the HIC while the heterogeneous behavior is managed by the HIC itself.

The HIC can be activated either in a homogeneous (as a projection) or in a heterogeneous context. In the homogeneous context, the HIC has input and outputs that all obey the same MoC and it behaves like an actor of the MoC. In the heterogeneous context, the HIC coordinates its behaviors in the different domains, according to its global heterogeneous behavior.

4. Non-Hierarchical Execution Model

The simulation of such flat heterogenous systems requires an execution model that is able to delegate the computation of the behavior of different components to their respective MoC, and to handle Heterogenous Interface Components. We designed the following execution model that

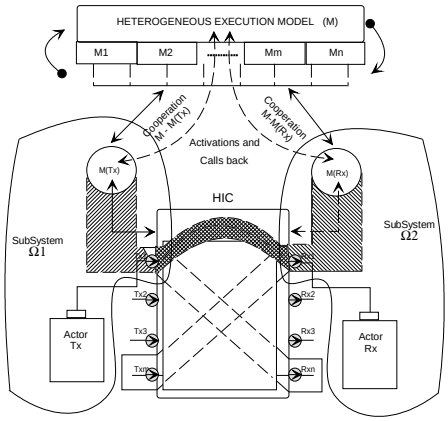

Figure 3. HIC - Execution Model Cooperation

operates in three phases : Partitioning of the system into subsystems, Scheduling of the subsystems and Execution.

In the initialization phase, the execution model divides the system at the border of the domains, and thus creates homogenous subsystems.

The HICs, which are at the border of several domains, are projected onto each subsystem to which some of their ports belong, and the other actors are transferred to their associated subsystems. The execution model copies the connections from the original system to the subsystems and generates virtual dependencies between the different projections of a HIC on the subsystems. Then, it schedules the activation of the subsystems and delegates their internal scheduling to their regular MoC. Finally it executes the system in accordance to the scheduling.

4.1. Structure of the System

The original system contains regular actors, Heterogenous Interface Components and communication channels

that link the ports of actors and HICs.

When there is a path between two HICs, the output of the producer HIC, the input of the consumer HIC and all the actors along the path belong to the same MoC (because regular actors belong to only one MoC) and will therefore be put into the same subsystem. The two HICs at both extremities of the path will be projected on this subsystem which will appear as self-contained from the point of view of its domain because the inputs of the producer HIC and the outputs of the consumer HIC will be masked by the projection. The original communication channels between heterogenous actors disappear with the projection of the HICs since they cannot be handled by the native homogenous execution models. We call such communication channels "heterogenous abstract channels", and they will be handled by the heterogenous execution model. We call "segment" of

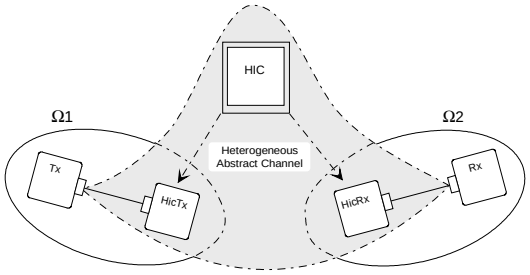

Figure 4. Heterogeneous abstract channel

the system a path between two HICs or between an initial or an end vertex of the graph and a HIC. Segments will be mapped to subsystems by the partitioning algorithm.
It may happen that a communication channel is lost when the system is divided along its segments, as shown on Figure 5. The A_i are regular actors and the H_j are HICs. A_1

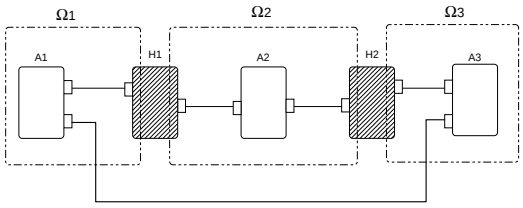

Figure 5. Lost communication channel

belongs to segment Ω_1 with the left port of H_1, A_2 belongs to segment Ω_2 with the right port of H_1 and the left port of H_2, and A_3 belongs to segment Ω_3 with the right port of H_2. However, in this partitioning of the system, the channel between A_1 and A_3 is lost because it belongs to none of the subsystems. We represent such lost communication channels with "homogenous abstract channels". These channels are homogeneous since their ends use the same MoC, but they cannot be represented by regular communication channels in a subsystem because the actors they connect do not belong to the same subsystem. The effective communication along those channels is done through "relay" actors that store data produced by the source actor and provide it to the target actor when its subsystem is activated as on Figure 6 The homogeneous abstract channels are used by the heterogeneous scheduler so that the subsystem that contains a source relay is activated before the subsystem that contains the matching target relay.

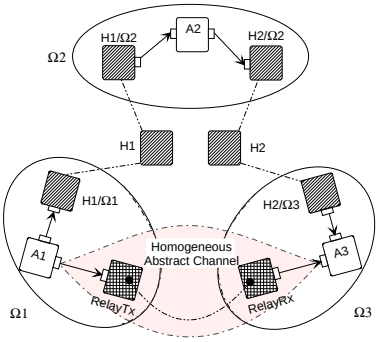

Figure 6. Homogeneous abstract channel

4.2. Partitioning the system

The partitioning algorithm minimizes the use of abstract homogeneous channels by first performing a topological sort on the actors of the system. Then, for each actor, in an order which is compatible with the topological sort, it looks for a subsystem that uses the MoC of the actor. If such a subsystem exists, it puts the actor there if the dependencies allow it, else a new subsystem is created to host the actor. The condition on the dependencies must be respected so that the subsystems that result from the partitioning can be scheduled.

To put an actor A_i in a subsystem S_j, the following conditions must hold:

- A_i must use the same MoC than S_j
- There is no path from A_i to any actor of S_j that goes through a HIC.

These conditions ensure that there won't be cross dependencies between subsystems. For instance, on Figure 7 (a), if we put A and C in the same Ω_1 subsystem (they both belong to domain d_1, and B and D in a Ω_2 subsystem, we cannot schedule the two subsystems because the projection of H in Ω_1 must be activated after B which is in Ω_2, and the projection of H in Ω_2 must be activated after A which is in Ω_1, so there is no possible schedule of Ω_1 and Ω_2.
In this case, the algorithm will build four subsystems, each one containing an actor and a projection of the HIC. It is possible to do a little better since the cross-dependencies

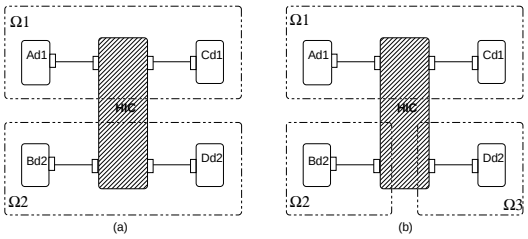

Figure 7. Constraints of scheduling

can be broken using only three subsystems as on Figure 7 (b).

The above conditions may hold for more than one subsystem for a given actor, in which case we choose to put the actor in the subsystem that already contains a projection of the HIC which belongs to the same segment as the actor if any, or we will put it in the most recently created compatible subsystem.

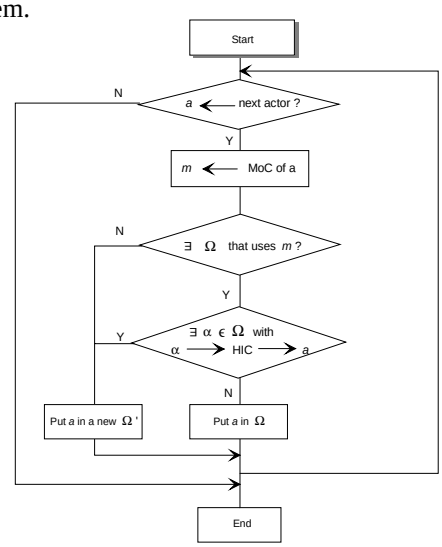

Figure 8. Flow chart of partitioning

4.3. Port disconnection and reconnection

When an actor is put in a subsystem, it is removed from the original model and put in a subsystem of the new model, but the connections to its ports are left untouched. For the new model to behave like the initial one, all connections are cancelled and actors are then connected inside each subsystem according to the communication channels of the initial model before the system is run.

4.4. Ad hoc dependencies, virtual ports

A connected system may be divided into subsystems that are not connected. Several systems can ve divided into several subsystems, and some not be connected because the connection between the two HICs is lost when they are projected on this subsystem. This should not be a problem, but some domains do not support unconnected systems, so we have to make the subsystem connected without changing its behavior.

To enforce the necessary ad hoc dependency between the projections of the HICs, we connect them through ports that are ignored by the HICs but that make the domain see a connected subsystem. We call the functionally unused ports "virtual ports".

4.5. Scheduling of the domains

Our flat heterogeneous execution model relies on the schedulers of the domains of the subsystems to schedule the actors inside the subsystems, so that its does not depend on the semantics of their model of computation. However, it must schedule the activation of the subsystems that were created during the partitioning of the system.

The precedence between subsystems is induced by the abstract homogenous (between actor relays) and the abstract heterogenous (between HIC projections) channels.

A subsystem Ω_1 precedes another subsystem Ω_2, and we note $\Omega_1 \triangleleft \Omega_2$ if either:

- Ω_1 contains an output relay and Ω_2 contains the matching input relay;
- Ω_1 contains a projection of a HIC which has inputs and Ω_2 contains a projection which has outputs.

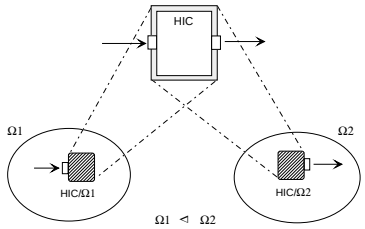

Figure 9. Dependency from HIC causality

Figure 9 shows how the causality relation between the inputs and the outputs of a HIC induces the precedence between subsystems: to be able to produce the right data, the projection of the HIC in Ω_2 must know the result of the reaction of the HIC to its input. The HIC receives data on its input in the Ω_1 subsystem, so this subsystem must be activated before Ω_2.

The scheduling of the subsystems should be done according to the precedence induced by the HICs and the precedence induced by the relays used to preserve homogenous communication channels across subsystems. However, because of the reasons that lead to the creation of relays, the precedence induced by relays on subsystems is always also induced by HICs. Therefore, it is sufficient to

take only the precedence induced by HICs into account for scheduling the subsystems.

After partitioning the system, we build a skeleton of the partitioned system that contains only the projections of the HICs and their dependencies. A topological sort of this skeleton is then used to determine the precedence relation on subsystems, and any order which is compatible with this relation of precedence yields a possible scheduling of the subsystems.

5. Example

Lets consider the flat heterogeneous system of Figure 10 which contains ten actors and uses two domains D_1 and D_2.

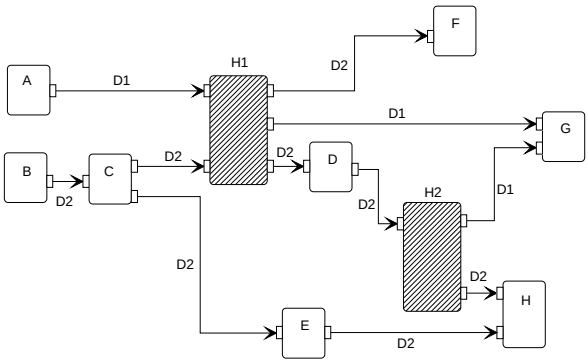

Figure 10. Flat Heterogeneous system

A cannot be put in the same subsystem as any other actor since actors that are on the same side of H_1 do not belong to the same domain, and actors in the same domain are reached by crossing a HIC, so A will be alone with the projection of H_1 in its subsystem.

B, C and E are in domain D_2 and will be put in the same subsystem. However, D and H cannot be grouped with them because there is a path from C to D through H_1, and there is a path from C to H through H_1 and H_2.

F can be put with D and E because they belong to the same domain and do not communicate through a HIC. G is the only actor that uses D_1 to the right of H_1, so it will be in its own subsystem.

Since E and H are connected but are not placed in the same subsystem, there will be two relay actors T_x and R_x that handle communications along this abstract homogenous channel. The skeleton for this partitioning is shown in the Figure 12 And yields the following precedence relations:

$$\begin{array}{lll} \Omega_1 \lhd \Omega_3 & \Omega_2 \lhd \Omega_3 & \Omega_3 \lhd \Omega_4 \\ \Omega_1 \lhd \Omega_4 & \Omega_2 \lhd \Omega_4 & \Omega_3 \lhd \Omega_5 \end{array}$$

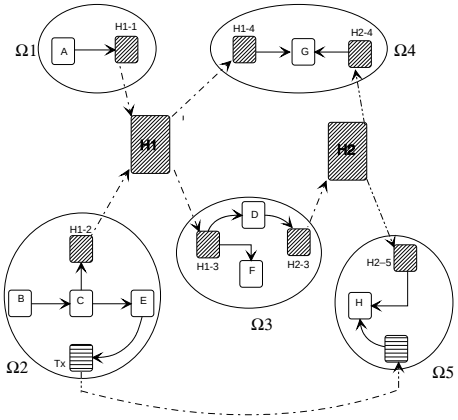

Figure 11. Partitioning of the example system

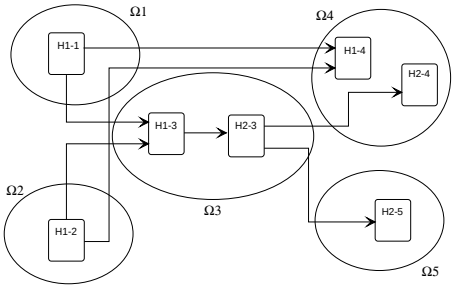

Figure 12. Skeleton of the example system

which allows the following schedules:

$$\begin{array}{lllll} \Omega_1 & \Omega_2 & \Omega_3 & \Omega_4 & \Omega_5 \\ \Omega_1 & \Omega_2 & \Omega_3 & \Omega_5 & \Omega_4 \\ \Omega_2 & \Omega_1 & \Omega_3 & \Omega_4 & \Omega_5 \\ \Omega_2 & \Omega_1 & \Omega_3 & \Omega_5 & \Omega_4 \end{array}$$

6. Simulation

The non-hierarchical heterogenous execution model is implemented in Ptolemy II. As an example, we used it to model a very simple system which use three MoC, SDF, DE and DT in the same level, and in which a sign-change detector drives an amplifier to rectify a sinusoid input signal. The sign-change detector and the amplifiers are HICs: the sign-change detector produces a Discrete event each time the sign of the value of its SDF input changes; and the amplifier changes the sign of its gain each time it receives an event on its DE input.

We cannot give a screen shot of the graphical aspect of the system in Ptolemy II since the graphical interface does not support flat heterogeneous systems yet. The system was built by assembling actors using the Java API of Ptolemy II. Figure 14 shows the result of the simulation of this heterogeneous system in Ptolemy II. The upper plot is the original

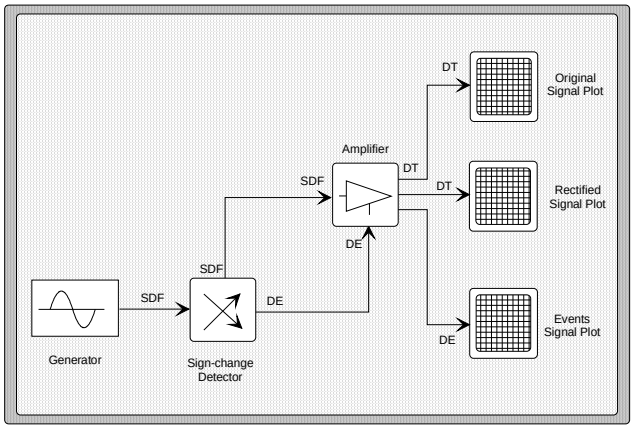

Figure 13. Simulated System

sinusod signal, the middle one is the amplified signal and the lower one is the events that begin the rectification.

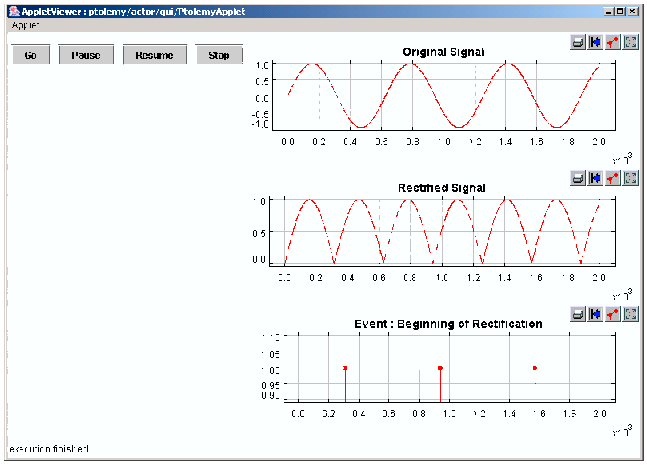

Figure 14. Result of the simulation

7. Conclusion

We proposed a new approach that allows to model heterogeneous systems without introducing artificial hierarchy levels. This approach is based on Heterogeneous Interface Components that have inputs and outputs that use different MoCs. So they can be used as bridges between homogenous "islands" in the modeling of heterogeneous systems.

We designed an execution model that supports HICs and can be implemented on various modeling platforms because it does not require the modification of the existing execution models. It has been implemented in the Ptolemy II plateform which supports many MoCs. This heterogeneous execution model divides the system into homogenous subsystems and maps HICs to homogenous components that can be handled by the native execution models of the subsystems. The scheduling of the subsystems is done without taking into account the semantics of their MoC.

References

[1] G. Agha et al, *"Abstraction and modularity mechanisms for concurrent computing"*, IEEE Parallel and Distributed Technology: Systems and Applications, 1(2):3-14, May 1993.

[2] G. Agha, I. A. Mason, S. F.Smith, and C. L. Talcott, *"A foundation for actor computation"*, Journal of Functional Programming, 7(1):1-72, 1997.

[3] 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

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

[5] C. Hylands, E. Lee, J. Liu, X. Liu, S. Neuendorffer, Y. Xiong, Y. Zhao, H. Zheng, *"Overview of the Ptolemy project"*, July 2, 2003, Technical Memorandum UCB/ERL M03/25.

[6] J. Liu et all, *"Actor-Oriented Control System Design"*, IEEE Transaction on Control System Technology, special issue on Computer Automated Multi-Paradigm Modeling. March 2003

[7] M. Mbobi, F. Boulanger and M. Feredj, *"Non-hierarchical heterogeneity"*, International Conference on Computer, Communication and Control Technologies, July-August, 2003, Orlando, Floride. International Institute of Information and Systemics, Volume III, ISBN 980-6560-05-01, pp. 430-435.

[8] Polis Homepage, Availlable on line at http://www-cad.eecs.berkeley.edu/ polis/

[9] H.J Reekie and E.A. Lee, *"Lightweight Component Models for Embedded Systems"*, Technical Memorandum UCB ERL M02/30, University of California at Berkeley, October 2002.

[10] Rosetta, 2004, Availlable on line at http://www.sldl.org/

[11] Specc, Availlable at http://www.specc.gr.jp/eng/index.htm

[12] SystemC Transaction Level Modeling Working Group Charter.