A Standardized Co-Simulation Backbone

Braulio Adriano de Mello[1,2], Flávio Rech Wagner[1]
{braulio, flavio}@inf.ufrgs.br

[1]Universidade Federal do Rio Grande do Sul - UFRS - Instituto de Informática
Cx. Postal 15064 - Av. Bento Gonçalves, 9500 - Campus do Vale
Bairro Agronomia - Porto Alegre - RS - Brasil - 91501-970

[2]Universidade Regional Integrada do Alto Uruguai e das Missões - URI
Departamento de Engenharias e Ciência da Computação
Rua Universidade das Missões, 464 - Sto. Ângelo - RS - Brazil - 98802-470

Abstract

In the field of co-simulation, the construction of a bridge between different simulators and the solution of problems like synchronization and data translation are some of the main challenges. This paper discusses the advantages of the HLA (High Level Architecture) standard to solve these problems and presents a generic architecture to support environments for geographically distributed co-simulation, called Distributed Co-simulation Backbone (DCB), which is based on the HLA.

1 Introduction

Co-simulation is used to make experiments and get information on the behaviour of heterogeneous systems aiming at the validation of their design or at the evaluation of their performance. Heterogeneous systems are characterised by a combination of hardware and software parts or by descriptions in different languages and/or at different abstraction levels [HES 99]. In the development of these systems, distinct parts are usually developed and validated in separate design processes. However, it is necessary to verify if those parts interact correctly taking distributed [BOR 00] communication and cooperation aspects into account. This validation task may be extremely complex because of the system heterogeneity. Therefore, one of the major challenges in co-simulation is the construction of a mechanism for a consistent cooperation among those parts, together with the construction of an efficient bridge among heterogeneous simulators.

Current techniques, environments, and tools for co-simulation have shown that one of the main bottlenecks is the communication interface. The large variety of technologies and their continuous evolution make it difficult to conceive an adaptive mechanism, which would promote the cooperation among heterogeneous simulators, without imposing restrictions on the simulators, regarding their data formats or behaviour.

Some examples of those approaches are: WESE [DHA 00]; MCI [HES 99]; and HILS [STO 98]. In general, existing approaches and tools are adequate for solving a set of predefined problems. But, frequently, unexpected problems must be handled. So, proprietary solutions may impose an excessive burden, in terms of redesign.

This paper presents an architecture for the DCB-Distributed Co-simulation Backbone which is based on the HLA [KUH 00]. HLA has its roots on defense programs and been recognized as a standard by the IEEE in 2000. It proposes rules and mechanisms for the interoperability of distributed, heterogeneous simulators. In particular, the RTI-Run Time Infrastructure, which is part of the HLA standard, offers facilities that give an important contribution for the construction of a generic mechanism supporting distributed co-simulation environments.

The fundamental idea behind the architecture of DCB is to remove the restrictions that are imposed upon independent simulators, considering aspects of interface, cooperation, and synchronization. The DCB architecture presents three definite advantages over other co-simulation environments: it is based on a public standard; it is far more flexible than current approaches; and it allows the integration of existing simulators without imposing modifications on their internal structures, without loosing fidelity and precision.

This paper is organized as follows. The RIT/HLA standard is briefly introduced in Section 2. Section 3 presents DCB. Conclusions and future work in Section 4.

2 Using RTI/HLA to Support Co-Simulation

HLA was initially conceived within the community of “distributed interactive simulation”, considering special needs that were observed in military training [KUH 00]. HLA offers a common architecture for the cooperative and distributed execution of individual simulations where different types of sub-systems, represented by simulators called federates, can interact. The collection of federates, based on a set of rules, is known as a federation [KUH 00]. HLA is specified by three main parts: the interface specification that includes the RTI whose task is to offer services for cooperation among federates (ambassador’s paradigm may be used for this [STR 98]); the Object Model Template (OMT), which defines a common and structured format for federates (SOM-Simulation Object Model) and federations (FOM-Federate Object Model); and a set of rules for the definition and management of the behaviour of federation and federates. The RTI specifies services can be grouped into three categories: federation management, data management, and time management. More details can be seen in [KUH 00]. These services must be also offered by mechanisms supporting distributed co-simulation environments, even if specified or implemented with alternative approaches. However, the RTI presents some restrictions due the heterogeneity that must be handled.

3 Distributed Co-Simulation Backbone

The DCB may be seen as a simulation-specific coordination layer for supporting distributed co-simulation. Its main goal is to offer a generic mechanism supporting communication and cooperation services among heterogeneous federates (according to the HLA). The DCB has been defined in the scope of the SIMOO project to support a transparent cooperation between models generated by it. SIMOO is a general-purpose, object-oriented simulator for discrete systems (mainly embedded systems [WAG 00]). A non-distributed co-simulation tool, integrating SIMOO and VHDL, has been already implemented [OYA 00]. An adaptation of SIMOO to the HLA standard is now underway [WIL 01].

Strongly based on the HLA standard, the building of the DCB extends three main strategies of the RTI removing some of its features that would inhibit flexibility. They are: interface specification, data exchanges and synchronization management. The DCB uses the concept of gateway [STR 98] to handle the complexity of the interface specification. Gateways translate data formats, according to the destination of the data sent through the DCB. Gateways are implemented as part of ambassadors [KUH 00]. In the transfers input/output data, the ambassadors check the availability of data that are controlled by DCB.

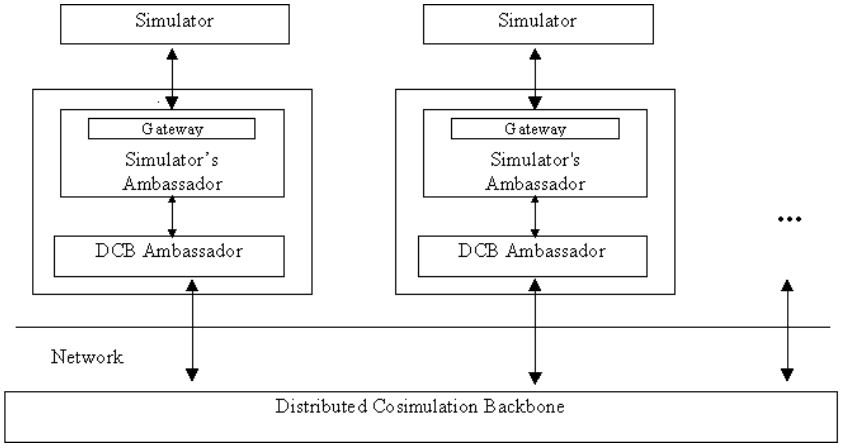

Figure 1 - Architecture of the DCB.

Because of this architecture (Fig.1), two ambassadors must be developed when a new simulator is integrated into a co-simulation environment: the simulator and the DCB ambassadors. Although ambassadors may imply some implementation effort to integrate new federates into a federation due particulars application model, these modifications are less costly if compared to the modifications in the simulators or in the DCB kernel. Also, the DCB infrastructure is general-purpose and is not affected by integration of new elements into a federation. However, the integration of a new simulator implies certain requirements that must be met by simulators in terms of interface specification, data management, and synchronization.

In the interface specification the attributes that will be used for data exchanging and for synchronization must be declared. In the attribute specification, the designer must include the following attributes: the simulator execution mode; TSL-Time of the Last State saved by the federate (in case of asynchronous simulators); LVT–Local Virtual Time of the federate; and a set of output/output variables of the federate. Of course, to add a simulator to an environment, the designer must know the characteristics of the simulators such as: execution mode, internal behaviour, among others.

In the data management the ownership management gives owns of attributes to simulators in according to the synchronization rules. Only a federate that owns a given attribute may update its value, and the ownership of the attribute may vary along the time. It is thus possible for the DCB to implement a mutual exclusion policy upon the use of the attributes by the federates, so avoiding undue attribute value updates, performed either by the local federate or by a remote federate. Ownership management is essential for synchronization purposes. At synchronization, the DCB support a hybrid mode of time advance. In the synchronous mode, the DCB uses the ownership management service

together with the LVT of the federates in order to guarantee that only safe events are executed. The same ownership management mechanism is used for time advancement in the asynchronous mode, where unsafe events and recovery saved state mechanism are supported.

4 Conclusions and Future Work

The DCB to support distributed environments consisting of heterogeneous simulators presents flexible strategies for the management of data interactions and synchronization between the simulators. Although new simulators can be easily (because the generality and flexibility of DCB) introduced into an heterogeneous and cooperative environment, some fundamental features discussed in the paper must be considered. Also, the generality of the DCB can be limited by performance and synchronization requirements of different domains. For example, hardware-in-the-loop in comparison with human-in-the-loop are distinct situations. For the validation of the DCB architecture, a particular federation for the co-simulation of electronic embedded systems is being implemented. In parallel, a supporting environment for the DCB development methodology is being built. It will offer services and resources for the configuration of interfaces of federates and for the semi-automatic generation of the ambassadors. The object-oriented features of the SIMOO [WAG 00] will be basic for this supporting environment.

5 References

[HES 99] HESSEL, F. et. al. “MCI: Multilanguage Distributed Cosimulation Tool”, F.J.Rammig (ed.), *Distributed and Parallel Embedded Systems***,** Kluwer Academic Publishers, 1999.

[BOR 00] BORRIELLO, G. and Want, R. “Embedded Computation Meets the World Wide Web”, *Communications of ACM*, Vl. 43, n. 5, May 2000.

[KUH 00] KUHL, F.; Weatherly, R. and Dahmann, J. *Creating Computer Simulation Systems: An Introcuction to the High Level Architecture*. Prentice Hall, MITRE Corporation, 2000.

[DHA 00] DHANANJAI, R.; Chernyakhovsky, V. and Wilsey, P. A. “WESE: A Web-based Environment for Systems Engineering”. *WEBSIM 2000*. January 2000.

[STO 98] STOLPE, R. and Zanella, M.C. “A Distributed Hardware-in-the-Loop Simulation Environment in Use on a Testbed of a Series Hybrid Drive”, In: *Proceedings…* Manchester, UK, June 1998.

[STR 98] STRASSBURGER, S.; Schulze, T.; Klein, U. and Henriksen, J. “Internet-based Simulation Using Off-the-shelf Simulation Tools and HLA*”*. *Proceedings of WSC*. Washington, USA, December, 1998.

[WAG00] WAGNER, F.R. et. al. “Object-Oriented Modeling and Co-simulation of Embedded Electronic Systems”. In: L.M.Silveira, S.Devadas, and R.Reis (eds.), *VLSI: Systems on a Chip*. Kluwer Academic Publishers, 2000.

[OYA 00] OYAMADA, M. and Wagner, F.R. “Co-simulation of Embedded Electronic Systems”. In: *Proc. of the 12nd European Simulation Symposium*. Hamburg, Germany, Oct. 2000.

[WIL 01] WILDT, D. and Wagner, F.R. “Adapting Simulation Environments to HLA: Discussion and Case Study”. In: *Proceedings…*, Prague, Czech Republic, June 2001.