COMPONENT APPROACH TO DISTRIBUTED MULTISCALE SIMULATIONS

Katarzyna Rycerz
Institute of Computer Science, AGH, al. Mickiewicza 30, 30-059 Kraków, Poland
Academic Computer Centre – CYFRONET, Nawojki 11, 30-950 Kraków, Poland

Marian Bubak
Institute of Computer Science, AGH, al. Mickiewicza 30, 30-059 Kraków, Poland
Faculty of Science, Informatics Institute, University of Amsterdam, Science Park 107, 1098 XG Amsterdam, The Netherlands

Keywords: Multiscale simulations, HLA, components, Distributed simulations, Grid computing.

Abstract: This paper presents a component approach suitable for distributed multiscale simulations. We describe the integration of the High Level Architecture (HLA) distributed simulation standard with modern computing frameworks: component technologies and Grid infrastructures, which enable users working on distributed simulations to easily exchange, the simulation models available to them. We discuss the design challenges involved in developing an HLA component which can be steered by a user during simulation runtime. More specifically, we present a solution which supports concurrent control of two different layers: the Grid (exploited by the user to steer the component). and the HLA. This functionality is presented on a sample multiscale stellar system simulation.

1 INTRODUCTION

Multiscale simulations are of great importance for complex system modeling. Examples of such simulations include e.g. blood flow simulations (Caiazzo et al., 2010), solid tumor models (Hirsch et al., 2009), stellar system simulations (Portegies Zwart et al., 2008) or computational chemistry simulations (Suter et al., 2009) .

The main scientific objective of this work is to determine how to exploit large-scale computing technologies in order to facilitate the creation and execution of complex simulations consisting of modules with different time scales. We focus on the requirements of composability, interoperability and reuse – here the main challenge is to support easy creation and interconnection of simulation modules by multiple users, for instance when one user creates modules while another one connects them in different ways.

Current solutions are usually restricted to specific models or simulation methods. However, partial solutions standardizing complex connections between simulation models already exist. In our work we have chosen to apply the High Level Architecture (HLA)[1] as it enables plugging in and unplugging various simulations from a single simulation system. HLA functionality is useful for multiscale applications, especially with these with different time scale, as it provides ability to connect simulation modules with different time management mechanisms. Additionally, as a means of composability support, HLA introduces a uniform way of describing events and objects being exchanged between modules.

In our previous work (Rycerz et al., 2008) we have proposed a component approach to the HLA standard solutions and integrate it with the Grid technology. The main advantage of this approach over the raw HLA is that it facilitates the development of simulation systems from components prepared by external users. In our solution, the behavior of each simulation module, wrapped as a component, is defined by an external user who has the ability to steer (from outside) the functionality of HLA used by his/her component. In order to enable collaboration and resource sharing, we rely on the Grid technology; more specifically on the H2O environment which is capable of hosting HLA components (Kurzyniec et al., 2003).

In this paper we focus on the design aspects of such components, namely on support for concurrent control of two different layers: the Grid layer

[1] High Level Architecture specification – IEEE 1516

Rycerz K. and Bubak M..
COMPONENT APPROACH TO DISTRIBUTED MULTISCALE SIMULATIONS.
DOI: 10.5220/0003592701220127
In *Proceedings of 1st International Conference on Simulation and Modeling Methodologies, Technologies and Applications* (SIMULTECH-2011), pages 122-127
ISBN: 978-989-8425-78-2

exploited by the user to steer the component and the HLA layer which supports multiscale-specific requirements, such as advanced time management. Basing on the Active Object pattern (Lavender and Schmidt, 1996) we have introduced a scheduler dealing with user's external requests comming from a Grid layer to a component.

This paper is organized as follows: in Section 2 we outline the available Grid and component technologies and related work for multiscale simulations. In Section 3 we describe the HLA component model. Section 4 presents an experiment involving a sample multiscale simulation of a dense stellar system. Conclusions and information on future work can be found in Section 5.

2 RELATED WORK

The requirements mentioned above are addressed by numerous partial solutions, including tools for multiscale simulation development as well as component and Grid approaches supporting composability and reusability.

The Multiscale Multiphysics Scientific Environment (MUSE) (Portegies Zwart et al., 2008) is a software environment for astrophysical applications where different simulation models of stars systems are incorporated into a single framework and a scripting approach is used to couple individual models. The Multiscale Coupling Library and Environment (MUSCLE)(Hegewald et al., 2008) provides a software framework for developing simulations according to the complex automata theory (Hoekstra et al., 2007). It has been applied in coronary artery in-stent restenosis simulations (Caiazzo et al., 2010).

Requirements related to reusability and composability of existing models can be met by applying modern IT solutions such as component and Grid technologies. Examples of the component approach include the Service Component Architecture (SCA)[2] where business functionality is provided as a series of services assembled together to create solutions that serve a particular business need, and the Common Component Architecture (CCA) (Armstrong et al., 2006) (with implementations such as MOCCA (Malawski et al., 2006)), used in high-performance computing where scientific components are directly connected by their uses and provides ports.

The Grid technology (Foster et al., 2002) is oriented towards joining geographically distributed communities of scientists working on similar problems. Currently, one of the most important Grid middleware platforms is the Globus Toolkit (Foster et al., 2002) together with the WS-Resource Framework (WSRF) which repurposes the classical Web Service interface to provide stateful WS-Resources[3]. The Grid also can, for example, be constructed from pure Web services and Grid Services (Foster et al., 2002) or by using the H2O resource sharing platform (Kurzyniec et al., 2003). Examples of production grids include European e-infrastructures such as EGI[4] or DEISA[5].

To the best of our knowledge, none of the existing multiscale computing tools or component models supports advanced time synchronization between simulation modules. Thus, we propose to apply the component approach to HLA simulations. Additionally, as our goal is to support the exchange of simulation modules between scientists regardless of their actual geographical location, we have decided to use the Grid, opting for the H2O framework (Kurzyniec et al., 2003) which is - at the same time - portable (Java-based), secure, scalable and lightweight.

3 HLA COMPONENT MODEL FOR MULTISCALE SIMULATIONS

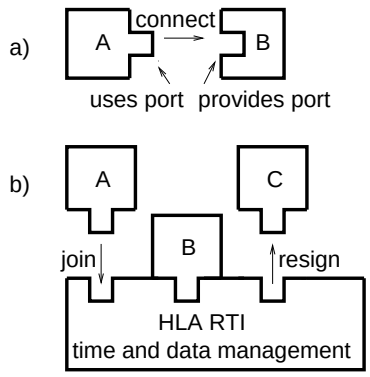

Figure 1: Comparison of two component models: (a) direct connections in CCA by *provides* and *uses* ports (b) components joining/resigning from the tuple space in HLA.

The features of the presented component model correspond to multiscale simulation requirements. Typically, such simulations require a significant amount

[2]G. Barber. Service component architecture home, 2007. http://osoa.org/display/Main/Service+Component+Architecture+Home

[3]WSRF home page: http://www.globus.org/wsrf/

[4]European Grid Infrastructure http://www.egi.eu/

[5]Distributed European Infrastructure for Supercomputing Applications http://www.deisa.eu/

of time to complete, producing partial results during execution. They also require support

for advanced types of connections between components, including time and data management. Additionally, as many simulation kernels exist in the form of legacy code, the component developer should be able to reuse such code in the most convenient way.

To fulfill these requirements we proposed a HLA Component model (Rycerz et al., 2008). In this paper we focus on the design aspects of HLA components, namely on the support for concurrent control useful for steering of a component during the simulation runtime. The steering include joing and risigning from a simulation, changing a time policy (useful for joining modules of different time scale) and altering components' connections for exchanging data. Our previous work simply exploited concurrency control mechanisms provided by the HLA RTI, in this paper we present comparison to alternative solution based on Active Object pattern.

The basic idea of the component model is shown in Fig. 1. Components can use advanced HLA RTI time and data management to which they can plug in via join/resign mechanisms. In comparison, the well-known CCA model (Armstrong et al., 2006) introduces direct connections between components by joining their *provides* and *uses* ports. The two key features which distinguish the proposed approach from existing ones can be described as follows:

- Support for time-consuming simulations: setup and steering of components is possible not only prior to actual execution (as in most component models) but also at runtime.
- The functionality provided by HLA that can be used by a component is available by external interfaces of the component – this facilitates the development of simulation systems from components provided by external users and allows to fully exploit the mechanisms of collaboration and resource sharing.

Supporting advanced steering of components during runtime raises many issues. One of the most important problems is how to effect appropriate transfer of control across many layers: (1) requests from the Grid layer external to the component (joing and risigning from the simulation, changing time and data management) (2) simulation code layer and (3) HLA RTI layer. The component should be able to efficiently process external requests for changing simulation state in the HLA RTI layer and, at the same time, deal with the actual simulation which may communicate with other simulation components via its RTI layer.

The solution presented in (Rycerz et al., 2008) exploited concurrency control mechanisms provided by the HLA RTI. External requests were processed in a synchronous fashion as they arrived, and were synchronized with internal simulation calls to HLA RTI within the HLA RTI itself (Fig. 2a). The advantage of this solution was that, apart from abort requests, external request handling did not require modifications in legacy simulation kernels. However, the solution also required appropriate exception handling in HLA itself and (as already remarked) the simulations exhibited a tendency to behave nondeterministically: we observed deadlocks and starvation of external or internal HLA requests which significantly slowed down execution.

In the current solution, presented in Fig.2b, we have decided to apply ideas derived from the Active Object pattern which separates request invocation from execution. We introduce a scheduler which deals with user external requests comming from the Grid layer and stores them in a queue. Once called from the simulation loop, the scheduler dispatches contents of the queue without interfering with actual simulation execution. This proposed solution requires only small modifications to the legacy simulation kernel (single routine); in return the developer acquires full control over external requests. Our solution exhibits deterministic execution patterns with no deadlock or starvation. The actual overhead of the scheduler is small, as presented in Tab. of Section 4. A detailed comparison of both component design approaches is shown in Tab. 1.

4 EXPERIMENTS INVOLVING SIMULATION OF A DENSE STELLAR SYSTEM

Currently, the implemented prototype of the CompoHLA system includes the CompoHLA components and a Java client that allows the user to manipulate these components. The implementation is based on the H2O technology.

The functionality of the CompoHLA system is presented with the example of a multiscale dense stellar system simulation. We have decided to build components from two MUSE modules: evolution (macro scale) and dynamics (meso scale) that run concurrently. In this experiment we use HLA time management features when one simulation controls the timing of the other. The dynamics module (called *constrained*) needs to receive an update from evolution (called *regulating)* before it passes a predetermined checkpoint. Data containing mass, radius, 3D posi-

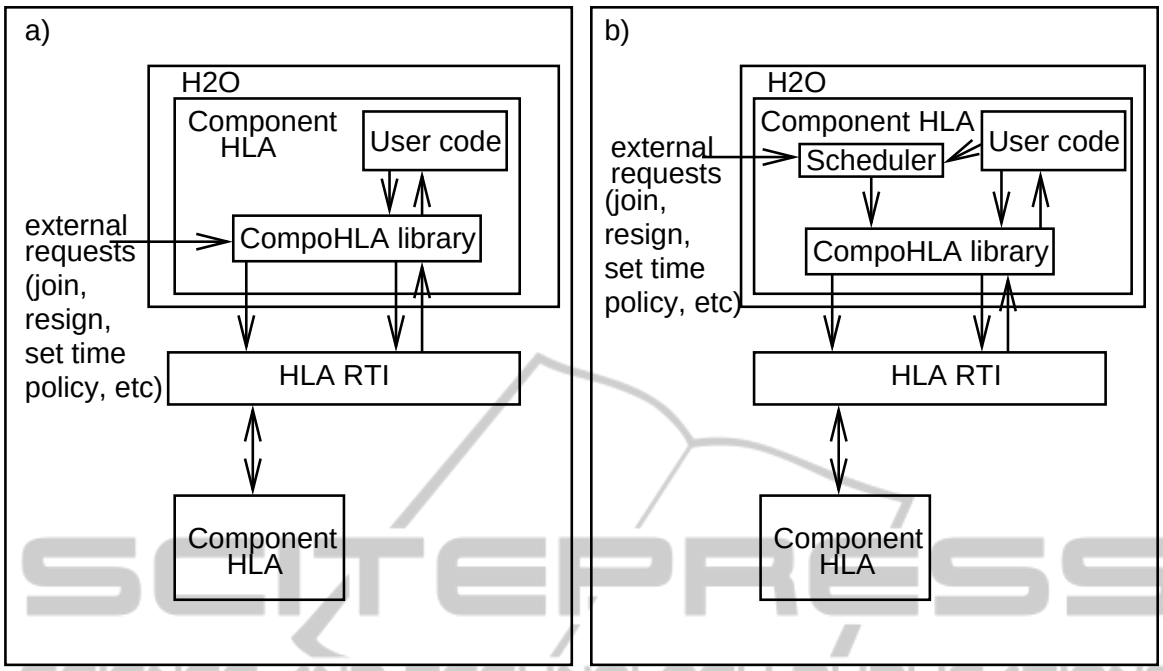

Figure 2: Architectural comparison of HLA components: (a) external and user simulation requests processed concurrently by CompoHLA and the HLA Runtime Infrastructure (RTI) library, (b) external requests queued by a scheduler which processes them following a call from user code.

Table 1: Comparison of HLA component design approaches.

Using mechanisms of HLA RTI concurrent access control	**Explicit scheduler**
transparent to the developer	requires calling a single routine in a simulation loop
synchronous mode - requests processed as soon as possible	asynchronous mode - invocation separated from execution
requests processed at any point in the simulation loop	requests processed when the scheduler is called from the simulation loop
dependent on implementation of concurrency control in HLA RTI	independent of the HLA implementation behavior
concurrency difficult to handle effectively - e.g starvation of requests causing overhead in simulation execution	concurrency easy to handle; low scheduler overhead

tions and velocity of stars is exchanged between evolution and dynamics. The HLA components are asked to join the simulations (both), set the time-regulating policy and publish data (evolution component), set the time-constrained policy and subscribe to data (dynamics component). After the initial 10 simulation steps, both components were asked to unset their time policy, resign from the simulation and halt. Our implementation uses H2O v2.1 and HLA CERTI implementation v3.3.2. Experiments were performed on different grid sites of Dutch Grid DAS3[6]. The actual setup of the HLA Component experiment is shown in Fig.3. The component client was run on the Leiden Grid node, the dynamics component was run on the Delft Grid node, the evolution component was run at UvA, Amsterdam while the HLA control process was run at Vrije Universiteit, Amsterdam. All Grid nodes shared a similar architecture (dual AMD Opteron computing nodes, 2.4 GHz, 4GB RAM). DAS 3 employs a novel internal wide-area interconnect (StarPlane) based on light paths between its Grid sites.

[6]The Distributed ASCI Supercomputer 3 web page http://www.cs.vu.nl/das3

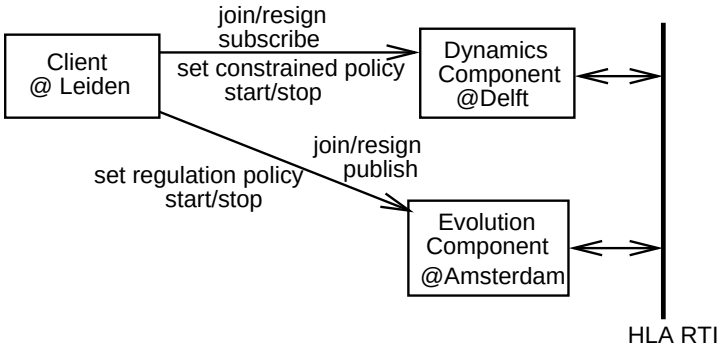

Figure 3: Setup of HLA Components experiment. The client sends requests to two components with differing scales, responsible for simulating the dynamics and evolution of a dense stellar system.

Table 2: Timing of actions in HLA Components.

External requests	**time, millisec**	σ
Join dyn.	10	1
Join evol.	9	0.8
Publish evol	6	0.4
Subscribe dyn	8	0.5
Set time policy evol.	4	0.4
Set time policy dyn.	9	0.1
Start dyn.	7	0.4
Start evol.	4	0.1
Unset time policy dyn.	14	1
Unset time policy evol.	9.5	0.5
Resign evol.	5.6	0.9
Resign dyn.	8	0.1
Stop dyn.	9	0.5
Stop evol.	5	0.1
Actions during simulations	**time, millisec**	σ
dynamics calculations	17.6	0.08
evolution calculations	0.004	0.0001
synch dyn with evol	0.007	0.0001
synch evol with dyn.	0.04	0.001
scheduler (both)	0.6	0.05
total sim time (evol)	1.1	0.05
total sim time (dyn)	18.3	0.08

The simulation model, wrapped by the dynamics component, was more computationally intensive: in our experiment the total execution time of the dynamics simulation was 18.3 seconds while the evolution simulation took 1.1 seconds. Detailed results are presented in Tab.2 which lists the execution time of two different types of actions in the dynamics component. We have calculated average values from 10 runs; σ indicates standard deviation. One group contains the timing of external requests for the HLA component (from the client perspective). As can be seen, this is on the order of several milliseconds. The requests are called asynchronously, so that during the call they are only scheduled in the queue. The other group contains actions taken during the simulation run: dispatching requests from the queue by the scheduler, actual computations and synchronization with the evolution component by means of HLA RTI. The results of the experiment show that time of external requests processing and later dispatching by a scheduler is considerably shorter than calculation time of computationaly intensive (dynamics) component and that the component layer does not introduce significant overhead. Performance tests have also shown that the overhead of HLA–based distribution (especially its repeating part which involves synchronization between multiscale elements) is small and that HLA can be successfully exploited in multiscale simulations.

5 SUMMARY AND FUTURE WORK

In this paper we have presented a new design of HLA component model which enables the user to dynamically compose/decompose distributed simulations from multiscale elements residing on the Grid. The presented approach differs from raw HLA where all decisions about actual interactions are set during implementation of simulation modules. In our solution, the multiscale application can be build from existing components by controlling HLA mechanisms from outside and joining of components is based on HLA functionality. Therefore, the components can communicate by exchanging data with time stamps using HLA (data management facility). To join components of different time scale one can set specyfic time policy of particular component (time management facility).

The additional feature of our solution is that component steering can also be done during actual sim-

ulation runtime. For that, we have used Active Object pattern with a scheduler that separates invocation of external requests from their execution. Therefore, we avoid interference between user request processing and running simulation. The features of that solution have been comparated with the approach of simply using concurrency control mechanisms provided by the HLA RTI.

The functionality of the prototype is presented with the example of a multiscale dense stellar system simulation – the MUSE environment (Portegies Zwart et al., 2008). The results of the experiment compare time execution of (1) actual computation, (2) the user external request processing, (3) dispatching requests by scheduler and (3) modules synchronization using HLA communication. The results show that the introduced component layer does not introduce significant overhead in comparison to computation time.

The HLA Components described in this paper are designed to facilitate composability of simulation models by means of HLA mechanisms accessible and steerable from the Grid layer. However, to fully exploit their composability potential, the use of such components should be part of a larger system which supports development and execution of complex multiscale simulations by applying the presented HLA component model. Therefore, we are working on integration of HLA Components with the GridSpace Virtual Laboratory [7].

ACKNOWLEDGEMENTS

The authors wish to thank Alfons Hoekstra and Jan Hegewald for discussions on MUSCLE, Simon Portegies Zwart for valuable discussions on MUSE and our colleagues from DICE team for input concerning GridSpace. The authors are also very grateful to Piotr Nowakowski for his suggestions. The research presented in this paper was partially supported by the European Union in the EFS PO KL Pr. IV Activity 4.1 Subactivity 4.1.1 project UDA-POKL.04.01.01-00-367/08-00 "Improvement of the Didactic Potential of the AGH University of Science and Technology – Human Assets" and also by MAPPER project – grant agreement no 261507, 7FP UE.

REFERENCES

Armstrong, R., Kumfert, G., McInnes, L. C., et al. (2006). The CCA component model for high-performance scientific computing. *Concurrency and Computation : Practice and Experience*, 18(2):215–229.

Caiazzo, A., Falcone, J.-L., Evans, D., et al. (2010). Complex Automata models for tissue growth in a stented artery. In *Proceedings of European Consortium For Mathematics In Industry, ECMI08*, Mathematics in Industry. Springer. in press.

Foster, I., Kesselman, C., Nick, J., et al. (2002). The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration. *Open Grid Service Infrastructure WG, Global Grid Forum*. http://www.globus.org/alliance/publications/papers.php.

Hegewald, J., Krafczyk, M., Tölke, J., et al. (2008). An Agent-Based Coupling Platform for Complex Automata. In *ICCS '08: Proceedings of the 8th international conference on Computational Science, Part II*, pages 227–233, Berlin, Heidelberg. Springer-Verlag.

Hirsch, S., Szczerba, D., Lloyd, B., et al. (2009). A Mechano-Chemical Model of a Solid Tumor for Therapy Outcome Predictions. In *ICCS '09: Proceedings of the 9th International Conference on Computational Science*, pages 715–724, Berlin, Heidelberg. Springer-Verlag.

Hoekstra, A., Lorenz, E., Falcone, J.-L., et al. (2007). Toward a Complex Automata Formalism for Multi-Scale Modeling. *International Journal for Multiscale Computational Engineering*, 5(6):491–502.

Kurzyniec, D. et al. (2003). Towards Self-Organizing Distributed Computing Frameworks: The H2O Approach. *Parallel Processing Letters*, 13(2):273–290.

Lavender, R. G. and Schmidt, D. C. (1996). Active Object : an Object Behavioral Pattern for Concurrent Programming. In *Pattern Languages of programs design 2*, pages 483–499.

Malawski, M., Bubak, M., Placek, M., et al. (2006). Experiments with distributed component computing across grid boundaries. In *Proceedings of the HPC-GECO / CompFrame workshop in conjunction with HPDC 2006*, Paris, France.

Portegies Zwart, S., Mcmillan, S., Nualláin, B. O., et al. (2008). A Multiphysics and Multiscale Software Environment for Modeling Astrophysical Systems. In *ICCS '08: Proceedings of the 8th international conference on Computational Science, Part II*, pages 207–216, Berlin, Heidelberg. Springer-Verlag.

Rycerz, K., Bubak, M., and Sloot, P. M. (2008). HLA Component Based Environment For Distributed Multiscale Simulations . In Priol, T. and Vanneschi, M., editors, *From Grids to Service and Pervasive Computing*, pages 229–239, Berlin, Heidelberg. Springer-Verlag.

Suter, J. L., Anderson, R. L., Christopher Greenwell, H., et al. (2009). Recent advances in large-scale atomistic and coarse-grained molecular dynamics simulation of clay minerals. *J. Mater. Chem.*, 19:2482–2493.

[7] http://dice.cyfronet.pl