Distributed simulation of building systems for legacy software reuse

Citation for published version (APA):

Document status and date:
Published: 01/01/2006

Document Version:
Accepted manuscript including changes made at the peer-review stage

Please check the document version of this publication:
• A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher’s website.
• The final author version and the galley proof are versions of the publication after peer review.
• The final published version features the final layout of the paper including the volume, issue and page numbers.

Link to publication

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.
• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain
• You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the “Taverne” license above, please follow below link for the End User Agreement:
www.tue.nl/taverne

Take down policy
If you believe that this document breaches copyright please contact us at:
openaccess@tue.nl
providing details and we will investigate your claim.

Download date: 09. Nov. 2019
DISTRIBUTED SIMULATION OF BUILDING SYSTEMS FOR LEGACY SOFTWARE REUSE

M. Trcka (Radosevic), J.L.M. Hensen
Architecture, Building and Planning - Building Physics and Systems;
Technische Universiteit Eindhoven, 5600MB
E-mail: m.radosevic@bwk.tue.nl

ABSTRACT: The use of integrated building performance simulation can substantially help in improving a building design with regards to comfort levels and fuel consumption, while reducing emission of greenhouse gasses. However, the traditional tools that are closed for inter-communication, limit the modeler to use of components only available within that particular package. This paper gives an overview of distributed simulation approach that can alleviate above limitation. Each program can represent only a part of a building system that is able to model, exchanging the necessary information during the execution and bridging the gaps between the tools. Several important issues closely connected with its implementation, such as synchronization, are pointed out, and the sensitivity of a model on different coupling strategies is studied. The paper concludes with highlighting the gained flexibility in modeling and simulation of building performance that arises from the distributed approach.

Keywords – building performance simulation, distributed simulation, external coupling, model reuse, software interoperability

1. INTRODUCTION

Building performance simulation (BPS) is being integrated into the design process, as benefits from its utilizations are being greatly recognized and valued (look at IBPSA proceedings ’89-’05). Many developments and improvements of tools and the deployment of the tools are a focus of many research groups around the world. Due to the decentralized developments, each BPS tool is restricted to a limited number of systems/components it can represent, which becomes evident when one wants to model and simulate an innovative building and/or systems (HVAC, lighting, shading, vents, operable windows, thermal storage systems, embedded renewable energy systems, etc.). Some components are available in one simulation environment and the some in the other. Still, the majority of tools are legacy codes originated from the mid seventies. They are: domain specific, not reusable, large, complex monoliths that are difficult to maintain, but still useful.

Previously (Hensen 1991, and Hensen and Clarke 2000) it has been argued that in the area of system simulation there is still enormous amount of work to be done. System modeling and simulation capabilities develop very slowly and take up an enormous amount of resources (time wise and financial). An efficient way forward would be to share developments and to reuse existing component models. However, the efforts of components models reuse and interoperability issues are mostly focused on model definition phase of modelling and simulation process, as shown on Figure 1. An overview of available techniques is given in Hensen et al. (2004). For example, data model reuse may be achieved on the product model level, either by sharing (Lockley et al. 1994) or exchanging (Bazjanac and Crawley 1999) information. Even though a common product definition model eases the use of simulation tools it addresses only part of the overall problem. Additional reuse may be achieved on the level of physical process models. This can be realized on source code level or in a more generic way by expressing the models in a neutral format, such as Neutral Model Format - NMF (Bring et al. 1999) that is now integrated in Modelica (Tiller 2001). Both data and process model reuse take place before run (or execution) time as shown in the upper part...
of Figure 1. So far, there was no a general mechanism in building performance simulation that would enable a modeller to model across various simulation environments while exploiting advances of each.

![Figure 1 Reuse in different phases of modeling and simulation process](image)

This paper introduces the concepts, background and core issues of an approach to process and data reuse by run-time exchange of information between the legacy simulation environments. In general terms, this approach in literature is recognized under the term: Distributed Simulation (DS), while the domain existing (legacy) software is referred as Commercial Of The Shelf (COTS) simulation software.

### 3. DISTRIBUTED SIMULATION APPROACH VS. TRADITIONAL SIMULATION APPROACH

One way to perform the simulation is to bring all component models together in a monolithic stand-alone simulation model that runs on an uniprocessor machine. That is usually done in the field of building performance simulation and it is known as program integration, where different domains are represented and simulated within the same program. The main drawback of the program integration approach is that the user is always restricted to the options/features offered by that particular program. An alternative to the traditional monolithic approach is distributed simulation. The driving motivation for distributed simulation is to integrate several separate simulations (federates) into a single simulation (federation).

Compared to the traditional approach, DS generates several advantages (Boer 2005, Ganse 2005, and Fujimoto 2005):

- reusability of already existing (legacy) COST software,
- combination of heterogeneous technologies,
- collaborative model design and development process,
- information hiding,
- scalability and fault tolerance,
- geographically distributed components, and
- reducing model execution time & more available memory.
The distributed simulation breaks boundaries between different simulations and by that introduces the potential to “pool recourses”, i.e., to use the best simulation model available without being limited to those available “locally”.

4. STATE-OF-THE-ART IN DISTRIBUTED SIMULATION

The concept of distributed simulation in particular has its roots in three separate communities: high performance computing community, defense community and internet gaming industry (Fujimoto 2003 and Wilcox et al. 2000).

There are two widely used architectures: client-server and peer-to-peer. In former architecture, simulation is executed on server machines, to which clients can log on from remote sites. The latter architecture does not have servers. The simulation is executed across many machines – peers. In the context of this research, we refer to the distributed simulation not only if the execution involves several computers, but moreover if there are at least two executables (federates) that exchange information in the federation run-time.

The activities enabled by the techniques and mechanisms that facilitate communication and data sharing between processes (applications) are called interprocess communications (IPC). The Figure 2 (partially taken from McGregor (2005)) shows the IPC taxonomy.

![Figure 2 IPC Taxonomy (McGregor 2005)](image)

An overview of most commonly used IPC protocols is given in Yahiaoui et al. (2003), and the reader is referred to that paper for more information. Buss and Jackson (1998) compared three higher-level architectures for distributed computing: HLA (High Level Architecture), CORBA (The Common Object Request Broker Architecture) and RMI (Remote Method Invocation) and distinguished three basic elements of distributed architectures as shown in Table 1.

| Table 1: Elements of distributed architectures (HLA, CORBA, and RMI) |
|-------------------|------------------|------------------|
| **Object interface language** | **Object manager** | **Naming service** |
| HLA | OMT | RTI | Federation execution |
| CORBA | IDL | ORB | DII |
| RMI | Java | UnicastRemoteObject and Naming Java classes | Registry Java class |
RMI uses its native implementing java language for object interface language, while HLA and CORBA define their own separate interface specifications that are distinct from their implementing languages, object model template (OMT) and interface definition language (IDL), respectively. The object manager is a backbone through which objects on all machines communicate, while the naming service is the mechanism by which clients discover the available objects on server during the computation run-time. RMI is language-specific and suitable for use with newly developed applications. Both, CORBA and HLA are concerned with legacy applications, possibly developed in different languages. However, CORBA as well as the majority of the IPC mechanisms represented on the Figure 2 (apart from HLA) are made to facilitate communication between two applications in general and not simulations in particular. As such, CORBA is not sufficient for straightforward use with simulation packages, where additional management of time and data exchange is required.

CORBA has been used in many projects in industry for integration of legacy code into distributed environment. For example, NASA Glenn Research Center (GRC) program, within NASA’s High Performance Computing and Communication (HPCC), has been developing large scale, detailed simulation environment for design analysis of aircraft engines, called Numerical Propulsion System Simulation (NPSS) (Follen et al. 2001, and Sang et al. 2002). The environment focuses on three aspects of modeling capabilities, such as: integrating engine components, coupling of multiple disciplines (aerodynamics, structural mechanics, heat transfer and combustion), and engine component zooming at adequate level of fidelity. In order to use legacy FORTRAN codes developed for majority of scientific and engineering applications GRC has developed a distributed simulation environment based on CORBA. They compile and link re-engineered FORTAN code with CORBA/C++ wrapper. Only a few changes within the FORTRAN codes are necessary. A control logic module of the component is moved to the client side, while the computation module is left on the server side. This gives the client the flexibility of controlling and invoking individual iterations. However, time management issue is not addressed.

On the other side, a great effort is put in developing higher-level architectures for distributed simulation in particular by US Department of Defense (DoD). These efforts resulted in a standardized protocol (Aggregate Level Simulation Protocol – ALSP), a domain specific Distributed Interactive Simulation (DIS) and finally merging these results in HLA that is today considered state-of-the-art in distributed simulation and was in 2000 made IEEE standard for distributed simulation (1516).

The HLA consist of three components: 1) a set of rules that govern certain characteristics of HLA-compliant simulations (for both federates and federation) (US DoD 1998), 2) OMT that describes the information of common interest to a group (called a federation) of cooperating federates, and 3) an interface specification (but not the interface itself) to a Run-Time Infrastructure (RTI) that provides the software environment needed by the federates to exchange information in a coordinated fashion. The rules for time and data management are clearly specified, and that is was distinguishes HLA the most from other distributed architectures.

Today, HLA is still mostly used only within defense community for military training simulators (Li et al. 2005 and Wilcox et al.2000) and in multi-player gaming (Wilcox et al. 2000, Pollini and Innocenti 2000). However, some initial actions have been made in order to adopt the standard in industry (Boer 2005, and Strassburger 2001). A vision of DaimlerChrysler is a “digital factory”. Such a factory model would be able to represent and simulate the entire factory process before going to “brick and mortar” (Boer 2005) by means of consistent data management, simultaneous development of product and production and early consideration of production requirements in developments. Existing models are
narrowly focused on isolated questions and only by coupling those models one can have a model of a detailed digital factory that cover all relevant causal relationships (Boer 2005, Taylor et al. 2002, and Taylor et al. 2004).

Besides DaimlerChrysler, Boeing also started to use distributed simulation (Wilcox et al. 2000). Both groups extensively cooperate with military, but they extended the use of HLA to their other aspects of business.

The broader automotive industry (Wang et al. 2005) uses distributed simulation to model supply chain management. Supply chain refers to the flow of material, information and services from the raw material suppliers through factories and warehouses to the end customer. It includes different organizations and process that all play a part throughout this journey. By using distributed simulation, each organization can develop its own simulation model, which encapsulates the information regarding that particular organization, without giving its know-how, and all independently designed models can be coupled and serve a common goal (Boer 2005, Duggan 2002, and Taylor et al. 2002).

Moreover, HLA was used in Lee (2004) to model satellite cluster management as for event based traffic simulator (Strassburger 2001, 2004).

In parallel to the initial attempts to use the HLA in the civilian domain applications, there is a delicate discussion weather or not the approach is suitable for needs outside defense community. Taylor in Taylor et al. (2002) argues that the HLA complexity that suites defense community requirements, might be in excess of relatively simple data exchange requirements in major industries, and questions the appropriateness of HLA implementations away from its original domain. In addition he raises the issue of data exchange approach standard lacking, as without such a standard there cannot be universal interoperation between COST environments. Wilcox at al. (2000) wonder weather many industrial communities either do not share, or have not yet explored the requirements to combine distributed models. Boer (2005) states that the size of projects in industry is smaller relative to the size of projects in military and that most of projects in industry will not benefit from HLA, considering costs. He argues that industry requires less complex solution. This reflects on the statement made by Paul in (Taylor et al. 2002). He argues that HLA in industry is only a solution that is looking for a “fantasy” problem and not finding one yet.

However, the implementation of either CORBA or HLA for distributed building systems simulation mainly raises difficulties when interfacing the legacy tools. The BPS tools are manly written in Fortran for which no object interface language (IDL, OMT) mappings have been defined. Much time and extra materials are necessary to overcome this difficulty as discussed in Yahiaoui et al. (2004).

By implementing a less complex IPC and formulizing the time management mechanism we believe that distributed simulation in the domain of building performance simulation can push the technology limits, enabling more flexible use of available legacy tools.

4.1. State-of-the-art in building performance distributed simulation

Some software-specific work has been done regarding the distributed simulation (run-time coupling) in the field of building performance simulation in general, i.e. ESP-r and Radiance (Janak 1997), ESP-r and Fluent (Djunaedy et al. 2003) and EnergyPlus and MIT-CFD (Zhai 2003).
In addition to the general run-time coupling developments, there are a few developments regarding distributed simulation in the domain of Heating, Ventilation and Air-Conditioning (HVAC) systems and control. Here we address some of the developments.

TRNSYS developers introduced a new type 155, defined as MATLAB connection. Matlab is launched at every TRNSYS time step as a separate process. The type 155 communicates with the Matlab engine through a Component Object Model (COM) interface. Any Matlab command (including Simulink simulations) can be run within a TRNSYS simulation (CSTB 2003). The similar approach is implemented in TRNSYS coupling with EES. TRNSYS is able to execute EES at each time step to solve a given set of equations.

A link between EnergyPlus and TRNSYS was used before EnergyPlus has obtained its own photovoltaic component model (TESS 2003). EnergyPlus module communicated information found in the EnergyPlus input file concerning photovoltaic arrays to TRNSYS. TRNSYS was then automatically launched during an EnergyPlus simulation to determine the performance of the PV array before returning control back to EnergyPlus. EnergyPlus then waited for TRNSYS to complete, then recuperated the output files that TRNSYS generates during its run and incorporated them into its native output-reporting format. The use of Windows API calls was used. However, the link is not a real distributed simulation application as there is not communication between programs on time step basis.

The assortment of efforts within the BPS domain presented show that there is a need for interoperable simulation environments. However, the research has not yet offered a general standardized framework for building interoperable simulation environments. This project has pioneered mechanisms on which the frameworks could be based.

6. TIME AND DATA MANAGEMENT ISSUES

Maybe the most important issue when discussing distributed simulation is time synchronization. To enable distributed simulation, components need to exchange data at run-time, and to synchronize their local (simulation) clocks. But, the time is in many cases the point of much confusion. There are three different types of time (Fujimoto 1998):

**Physical time** refers to the time in the physical system being modeled. For instance, a physical time may be as long as a year (annual building performance simulation).

**Simulation time** refers to the simulation representation of time. It is correlated to physical time, but the exact duration of one time unit will depend on computational algorithms and software execution speed.

**Wallclock time** refers to real time of simulation execution. For example, an annual (physical time) building performance simulation may be executed in 1 min (depending on software and model complexity) of wallclock time, and on the other side a CFD simulations wallclock time easily exceeds the physical time.

Time management is concerned with the mechanisms used by simulations to advance through simulation time. The discussion of synchronization and time management primarily concerns the simulation time. However, there are cases in praxis where synchronization with wallclock time is required (HIL or MIL (hardware/man in the loop) simulations). Fujimoto (1998) makes distinction of scaled real time simulations and as-fast-as possible simulations. Simulations in the domain of building performance fit the second approach (again apart real time HIL simulations (Xu et al. 2004)). For the latter approach the simulation execution does not have direct relation to wallclock time. HIL and MIL applications are not concern of this study and we will not further address synchronization with wall clock time.
In distributed simulation, federates run concurrently on potentially separate machines, each of them following its own time management scheme. However, the simulations are seamlessly dependent on each other, and the execution of one will influence the execution of the other in the corresponding federation (simulation) time. It is therefore important that the simulation clocks of each federate are synchronized with federation time, i.e. the simulation time of other federates.

Many studies dealing with distributed simulation address the issue of synchronization (Fujimoto 1998, Tacic and Fujimoto 1998, Wang et al. 2004, and Boukerche et al. 2005). They all generally tackle the event driven simulations. The simulations in the BPS field are time-stepped, where each time advance made by a federate is of some fixed duration of simulation time. The advance in federate simulation time (by this fixed stepped fashion) should not be allowed unless the federate received all data relevant for the current time step from other federates in federation. This brings us to couple of important points regarding synchronization realization:

1. each information to be exchanged must have a time stamp,
2. federate must not receive any information with time stamp less than its current simulation time, and
3. federate needs to have mechanism of determining weather all necessary to be exchanged information in the current simulation time step has been received.

We distinguish internal and external time management approaches. The internal time management indicates that the synchronization checking procedure is coded within federates themselves. On the other side the synchronization can be compassed within the inter process communication (IPC) mechanisms, applying blocking mode, for example.

There are two major groups of algorithms for synchronization (Fujimoto 2003):
- Conservative – take precautions to avoid the possibility of processing data out of time stamp order, i.e. execution mechanism avoids synchronization errors
- Optimistic – does not necessary avoid synchronization errors, but rather use a detection mechanism and recovery approach, known as roll-back. Introducing roll-back to an existing simulator requires a major re-engineering effort (Page et al. 1999) to incorporate state saving mechanism.

It is necessary to mention that several very important issues arise when building performance simulation is distributed, which has not been considered in the papers related to distributed simulation in military or automotive industry.

Firstly, we are dealing with legacy software and the major idea with which we approach distributed coupling is that we were not going to make major interventions over the original code. Secondly, distributed models can be very strongly coupled. This means that inputs for one federate are outputs of another federate, which are function of the outputs of the first federate. We call this strong interdependence: coupling with a feedback. Due to the autonomy of legacy software it is not always possible to exchange all necessary information in the current simulation time step (Treka (Radosevic) et al. 2006) and that, results in federates that are delayed in simulation time to other federates. Weather there will be a delay will depend on the applied coupling strategy as well as on the federate system solution technique.

There are two different coupling strategies:
- quasi-dynamic coupling (Zhai 2003), or loose coupling (Struler et al. 2000), or ping-pong coupling (Hensen 1999) and
- fully-dynamic (Zhai 2003), or strong coupling (Struler et al. 2000), or onion coupling (Hensen 1999).
For accuracy constrains, the coupling strategies are closely related to time steps length. Accuracy as well as stability constraints will limit the simulation time step length in case of the first strategy. The second strategy allows longer time steps for the same accuracy, but it requires an iteration procedure to ascertain user defined convergence criteria. Hence, for the same accuracy we can either reduce time step length and apply the first strategy, or employ larger time steps with iterations between federates. Only with full dynamic coupling we can avoid time delays between federates, but that we will have to be compensated with iterations. Advantages and disadvantages of these approaches are addressed elsewhere (Hensen 1999). The difficulty of applying iterations again differs between the coupling mechanisms employed, i.e. discontinuous or continuous (see section 7) and is dependant on the nature of coupled component model (steady state or dynamic) (Treka (Radosevic) et al. 2006).

Also, the external coupling will result in different time step variable exchange depending on which system solution approach is used (Radosevic et al. 2005). In terms of individual component models two main system solution approaches can be distinguished: input-output based (each component is represented by an input/output relationship), and conservation equation based (each component is described with time-averaged discretised heat and/or mass conservation statements which are combined to form a plant system matrix, and which are solved simultaneously for each simulation time step using either an implicit, explicit or mixed numerical scheme). Inter programs time step variable exchange will disturb the original intra time step variable exchange of the base program that uses implicit numerical scheme in conservation equation based approach. However, if explicit numerical scheme in the same approach is used the external coupling will keep the original intra time step data exchange consistent. The same applies for input-output based component modeling approach. Small coupling time steps, which are required by the chosen coupling strategy (ping-pong), will result in neglecting the discrepancy between the inter- and intra- time step variable exchange schemas and ensure the stability and accuracy of the obtained results.

Finally, this brings us to the third issue: implementation of a multi time step approach. Although it was previously mentioned that the advances in federates simulation time are made by fixed time step, some of the software in the domain implement multi time step approach to deal with a huge range of time constants within the simulated system. The federate time step that is assigned at the beginning of the simulation might be reduced during the simulation execution. This fact raises some limitations for synchronization management mechanisms. Other federates can require information to be exchanged on a global federation time step and should not receive the information that the federate employing multi time steps sends on reduced time steps. Or the federate employing multi time steps should not send the data, unless the simulation progressed for by the global time step duration.

7. SENSITIVITY OF COUPLING STRATEGY

Developments associated with this project were published elsewhere (Radosevic et al. 2004, Radosevic et al. 2005, and Treka (Radosevic) et al. 2006). The major goal is to establish mechanisms that deal with all the issues associated with distributed simulation of legacy software from the building performance simulation field. The approach undertaken by the project is to develop components within each environment that will be used to interface other environments. Such components were developed for TRNSYS, ESP-r, EnergyPlus and several smaller stand alone programs (e.g. EARTH), respecting the developer’s style in each program. The project core issues are related to data and time management (synchronization) and coupling strategies between programs relative to the accuracy and stability of the simulation results. For implementation purposes, the developed prototypes are based on peer-
to-peer architecture and use named pipes (UNIX application), shared memory (Windows applications) and sockets (UNIX-Windows applications). We implemented conservative time management, as execution speed does not fall into the main focus of interest.

Two distinct mechanisms are developed, named continuous and discontinuous. In the former federates are executed concurrently and have internally build synchronization algorithm. In the latter, there is so called “base” and “external” federate. The base federate is a master federate and invokes external federate when necessary. Both mechanisms, implement quasi-dynamic coupling strategy, i.e. no inter software iterations are employed. We justify this decision by the following example, showing the results of the coupling strategies sensitivity study.

The influence of the coupling strategy on the simulation results was investigated for different coupling time steps. For the study a simple HVAC network was used as shown in Figure 3. To assess the sensitivity of coupling strategy, the simulation output of a one-program model (i.e. assumed to be fully-dynamic coupled) is compared with the simulation output of a distributed model (quasi-dynamic coupled) of the same overall system. Several simulations were performed changing the length of coupling time step, i.e. the frequency of data exchange among federates.

The results in Figure 4 show that if the exchange frequency is high enough, the results for both coupling strategies are similar. However, if the coupling frequency is reduced, the difference between the strategies increases. For example, if the coupling time step is reduced to ten minutes, oscillation that appears due to the particular combination of control parameters, controlled system and simulation time step are much larger than if the models do not iterate. With further reduction of the coupling frequency, the difference with regard to the oscillating amplitude between the strategies is less apparent, but the phase shift due to the time delay between federates is still present.

From the example, it can be concluded that with an appropriately chosen coupling- as well as simulation time step the differences between strategies are small. The loosely (quasi-dynamic) coupled federates will produce the same quality results as strongly (fully-dynamic) coupled ones. For reasons of simplicity, the quasi-dynamic approach was chosen as the starting point for the prototypes elaborated in (Radosevic et al. 2005, and Trcka (Radosevic) et al. 2006).
9. CONCLUSIONS

With its roots in domains of software engineering and military training simulation, the benefits of distributed simulation environments are also being recognized in some parts of civil industry. With distributed approach each program can represent only a part of a system that is able to model. It may happen that a simulation model potentially residing on a distant computer already provides the particular functionality. The distributed simulation is in this case an alternative to additional effort for rewriting the software locally. Additionally, a varied level of detail of a simulation models can introduce better correlation: fidelity in obtained results vs. simulation goals as well as the improved behavior of the simulation models that can have varied time management schemes. The distributed simulation brakes boundaries between different simulations and by that introduces the potential to “pool recourses”. We recognized the potential of its use in the building performance simulation and explored its benefits and pitfalls.

The time management issue has been addressed in particular. Different approaches have been discussed in relation to the specific types of simulation packages and solution techniques used in the building performance simulation.

To finish, the example that investigated the influence of employed coupling strategy to the simulation results has been presented. The results from the study justified the choice of quasi-dynamic coupling strategy for the developed prototypes.

Figure 4 Influence of coupling strategy on zone temperature calculation with different coupling time steps
10. REFERENCES


International Conference PLEA. Passive and low energy architecture, pp. 1185-1190, Eindhoven University of Technology.


Tiller M.M. 2001. Introduction to physical modeling with modelica, Kluwer Academic Publisher, Norwel, Massachusetts


