Distributed Building Performance Simulation - a Novel Approach to Overcome Legacy Code Limitations

ABSTRACT
This paper describes the development as well as implementation strategies for distributed simulation of building systems by run-time coupling of existing software. The approach differs from the traditional way of developing software, where additional models are added by incorporating new modules in an existing program. This paper focuses on the actual coupling mechanism. A case study is presented which illustrates the need and potential of the approach. The conclusion is that a distributed simulation environment is more flexible, practical, and powerful than the sum of the individual software programs.

INTRODUCTION
The traditional (manual) methods for designing heating, ventilation and air-conditioning (HVAC) systems are being surpassed by simulation tools because:

- buildings become more and more complex in terms of shape, lay-out, functionality and services;
- requirements for flexibility and adaptability increase;
- modern building standards and codes are performance based rather than prescriptive; i.e. addressing questions such as “how many hours per year will the temperature rise above a certain value?” and “what will be the annual energy consumption per square meter floor?”

Advances in hardware and software have resulted in a flood of building simulation tools. However, each tool is applicable only to a subset of the overall problem, and is limited both in scope and resolution. The majority of tools are legacy codes often originating from the seventies. On the whole they are domain specific, not reusable, large, complex monoliths that are difficult to maintain, but still useful.

Previously (Hensen 1991; 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. An overview of how this can be done is given in (Hensen et al. 2004). One way would be on the product model level, either by sharing (Lockley et al. 1994) or exchange (Bazjanac and Crawley 1999) of product information. Even though a common product definition model eases the use of simulation tools, it addresses only part of the overall problem. Reuse can also take place on the level of physical process models. This can be realized on source code level (program integration) 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 follow a traditional approach, where all component models are brought together in a monolithic stand-alone simulation program. The integration takes place before run (or execution) time, as shown in the upper part of Figure 1.
This paper introduces the concepts, background and core issues of process and data reuse by run-time exchange of information between legacy simulation environments. In general terms, this approach is recognized in literature as Distributed Simulation (DS), in which context the domain existing (legacy) software is referred to as Commercial Off-the-Shelf (COTS) simulation software. The aim of the current work is to resolve the communication between various HVAC component or system simulation software packages. The goal is to define the coupling methodology in terms of content and frequency of the data exchange.

**DISTRIBUTED SIMULATION**

Opposed to the driving motivation for parallel simulation, which is decreasing the simulation execution time, the main motivation for distributed simulation is to integrate several separate simulations (federates) into a single simulation (federation). Each sub-system is modeled in appropriate software and simulated, potentially, using different computers (Figure 2), while intermediate results are communicated over the network during execution time. The possibility to model various interdependent aspects over a wide range of applicability and resolution allows much greater flexibility in the use of building energy simulation.
Compared to the traditional approach, DS may offer several advantages (Boer 2005; Ganse 2005; and Fujimoto 2005):

- reusability of already existing (legacy) COTS software;
- combination of heterogeneous technologies;
- collaborative model design and development process;
- information hiding;
- scalability and fault tolerance;
- geographically distributed components;
- reducing model execution time & more available memory.

Distributed simulation breaks boundaries between different simulation environments and by that introduces the potential to “pool resources”, i.e. to use the best simulation model available without being limited to those available “locally”.

**State-of-the-art in general**

There are two widely used architectures for distributed computing: client-server and peer-to-peer. In the 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 our case DS does not necessarily involve more than one computer; it suffices if there are at least two executables (federates) that exchange information in the federation run-time. Distributed simulation requires communication between processes (applications) or interprocess communication (IPC). Figure 3 (partially taken from (McGregor 2005)) shows the IPC taxonomy.

![Figure 3 Interprocess communication (IPC) taxonomy (McGregor 2005)](image)

An overview of most commonly used IPC protocols is given in (Yahiaoui et al. 2003). Buss and Jackson (1998) compare three architectures for distributed computing: HLA (High Level Architecture), CORBA (Common Object Request Broker Architecture) and RMI (Remote Method Invocation) and distinguished three basic elements of distributed architectures as shown in Table 1.

<table>
<thead>
<tr>
<th>Distributed architecture</th>
<th>Object interface language</th>
<th>Object manager</th>
<th>Naming service</th>
</tr>
</thead>
<tbody>
<tr>
<td>HLA</td>
<td>OMT</td>
<td>RTI</td>
<td>Federation execution</td>
</tr>
<tr>
<td>CORBA IDL</td>
<td>ORB</td>
<td>DII</td>
<td></td>
</tr>
<tr>
<td>RMI</td>
<td>Java</td>
<td>UnicastRemoteObject and Naming Java classes</td>
<td>Registry Java class</td>
</tr>
</tbody>
</table>

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 the 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 in Figure 3 except HLA were developed to facilitate communication between applications in general and not between simulations in particular. Therefore CORBA is not ready-made for use with simulation packages, since additional management of time and data exchange is required.

CORBA has been exploited in many projects in industry. For example, NASA Glenn Research Center (GRC) program, within NASA’s High Performance Computing and Communication (HPCC), is developing a large scale, detailed simulation environment for design analysis of aircraft engines, called the Numerical Propulsion System Simulation (NPSS) (Follen et al. 2001; and Sang et al. 2002). Based on CORBA, the NPSS reuses legacy FORTRAN codes for many scientific and engineering applications The NPSS environment focuses on three modeling aspects: integrating engine components, coupling of multiple disciplines (aerodynamics, structural mechanics, heat transfer and combustion), and engine component zooming at adequate level of fidelity.

The US Department of Defense (DoD) puts a lot of effort in developing higher-level architectures for distributed simulation. This resulted in a standardized protocol (Aggregate Level Simulation Protocol – ALSP), and in the High Level Architecture (HLA) which is the current state-of-the-art in distributed simulation. HLA became in 2000 the IEEE (Institute of Electrical and Electronics Engineers) standard for distributed simulation (1516).

Today, HLA is mostly used within the 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 steps have been taken in order to adopt the standard in industry (Boer 2005 and Strassburger 2001). The industry domains that have so far tried to exploit the advantages of HLA, include supply chain simulation, digital factory simulation, traffic simulation, and similar. Although some attempts to use HLA in civil applications have been made, there is also an ongoing discussion whether or not the approach is suitable outside the defense community. Taylor et al. (2002) argues that the HLA complexity that suits 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. Boer (2005) states that projects in industry are much smaller than military projects and that most industry projects will not benefit from HLA, considering the associated costs. Boer (2005) argues that industry requires a less complex solution. Along the same lines, (Taylor et al. 2002) argues that HLA in industry is only a solution that is looking for a “fantasy” problem but did not find one yet.

However, the implementation of either CORBA or HLA for distributed building systems simulation mainly raises difficulties when interfacing the building simulation domain legacy tools. The BPS tools are mainly 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 (Follen et al. 2001; and 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.

Distributed building performance simulation

In the field of building performance simulation, some software-specific DS work has been done in order to integrate high-resolution light simulation, involving ESP-r and Radiance (Janak 1997), or to integrate computational fluid dynamics, involving ESP-r and Fluent (Djunaedy et al. 2003). The coupling between TRNSYS and COMIS (Dorer and Weber 1997), EnergyPlus and COMIS (Huang et al. 1999), EnergyPlus and MIT-CFD (Zhai 2003), EnergyPlus and DeLight etc., are not implemented as external. TRNSYS and EnergyPlus incorporate additional domain tools by converting them into the new programs’ subroutines, i.e. types. This is what is called by internal coupling.

Other work focused on integrating HVAC simulation. TRNSYS developers introduced a new type 155, defined as MATLAB connection. The latter application 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 features) can thus 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 (Keilholz 2002).
A link between EnergyPlus and TRNSYS was used before EnergyPlus obtained its own photovoltaic component model (TESS 2003). EnergyPlus communicated product model data concerning photovoltaic (PV) 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 waited for TRNSYS to complete, and recuperated the output files that TRNSYS generates during its run and incorporated them into its native output-reporting format. Windows API calls were used for the communication. This process doesn’t involve communication on a time step basis, so it does not constitute DS as described in the introduction. Recently, the link between EnergyPlus and Spark has been developed. Individual HVAC components in EnergyPlus can be modeled with SPARK problems allowing the use of SPARK stand-alone HVAC models in the place of the native EnergyPlus models (Curtil 2004).

The above range of efforts indicates the need for integration in the BPS domain. However, until now, the research is inconclusive. There exists no general standardized framework for integration of building performance simulation environments.

**COUPLING DECISION METHODOLOGY**

The use of building simulation should be problem-led rather than tool-led. Due to the increasing requirements in terms of knowledge and skills, as well as increasing computing resources, it is generally advisable to obey Einstein’s principle “a model should be as simple as possible, but no simpler.” In the current context this means that the starting point should be the lowest possible model resolution and complexity level that satisfy required accuracy of performance indicator(s) of interest; e.g. as in Table 2. Choosing the system model for a specific purpose is still more an art than an engineering discipline (Forbus 1996, Moody 2005). However, there are some rational processes that can be applied for model development (Trcka et al. 2006b). The processes can be based on a checklist form or they can include some quantified qualities to validate the chosen modeling abstraction level. The use of a checklist, bounding abstractions and (differential) sensitivity analysis has been recognized (Trcka et al. 2006b) as a potential checking procedure for definition of decision-making criterion of modeling abstraction level.

<table>
<thead>
<tr>
<th>Table 2</th>
<th>Potential questions/problems in design and maintenance of HVAC systems</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ref. number</td>
<td>Design question</td>
</tr>
<tr>
<td>1</td>
<td>Maximum load calculation</td>
</tr>
<tr>
<td>2</td>
<td>Inquire about comfort condition</td>
</tr>
<tr>
<td>3</td>
<td>Energy consumption - gross</td>
</tr>
<tr>
<td>4</td>
<td>Energy consumption - global</td>
</tr>
<tr>
<td>5</td>
<td>Fuel consumption</td>
</tr>
<tr>
<td>6</td>
<td>Component effect on building energy performance</td>
</tr>
<tr>
<td>7</td>
<td>Fault detection and diagnosis</td>
</tr>
<tr>
<td></td>
<td>Optimization of the control</td>
</tr>
</tbody>
</table>

A decision-making procedure is schematically shown in Figure 4, in which the numbers correspond to the questions/problems in Table 2. Each has its own minimum resolution level, which does not necessarily have to be sufficiently accurate in any specific case.
For example, if information about maximum temperature were required (question 2), pure conceptual system modeling would initially be the minimum resolution level. If from the checking procedure (shaded area in Figure 4) turns out that the uncertainty of the model gives a rise to unacceptable inaccuracies of the performance parameters, the level of system modeling should be one higher on the resolution scale, as indicated. The checking procedure can be performed by applying either bounding abstraction test or a differential sensitivity analysis (Treka et al. 2006b).

However, it may happen that no model exists for one or more system components in a particular simulation environment. Going to a higher level of complexity and implementing external coupling is quite demanding. That is why the checking procedure is also important to justify such decision. Again, defining the range of change or bounding of coupled variables, and comparing their influence on the change of the values of performance parameters of interest would facilitate the decision-making process. However, such definition of checking procedure does not represent influence of transient changes of coupling variables, and at its best can bound the qualities of interest. Therefore, a simpler check for necessity is adopted in this paper as follows.

**Justifying external coupling of HVAC component models**

It is obvious that external coupling of HVAC component models is justified only if:

- the model for a required component does not exist in the base program, or
- the existing model is not adequate for the specific study, or
- the component is a real building and/or system.

If justified, there are potentially two distinct ways of running the simulations of the base and the external programs: they can be run-time coupled or sequentially coupled. The sequential solution means that once one simulation is finished its output will be redirected to the input of another program. The run-time coupled approach requires run-time exchange of coupled data between the simulations.

Figure 5 shows possible configurations in terms of necessity for coupling of the subsystems. Only in special cases a sequentially coupled approach will be sufficient. Most cases will require a coupled approach.
In open systems, if the coupling data are constant, or if they vary only as a function of the first subsystem model and its input, then the system can be analyzed decoupled (sequentially coupled). This is not the case if the coupling data changes not only as a function of the first subsystem model and its input, but also by what is going on in the second part of the system (e.g., a controllable fluid flow inducer is placed in the second subsystem and consequently influences the flow in the upstream subsystem).

Further, in closed-loop configurations, there is inherent feedback between subsystems. It doesn’t matter whether the loop is closed by the working fluid or due to control signals. There will be dynamic interactions between the components and thus run-time coupling is required in these cases.

**RUN-TIME COUPLING**

The work reported here pioneers mechanisms on which building performance simulation frameworks could be based.

**Data-to-be exchanged**

The minimum number of variables that defines the thermodynamic state of a working fluid is theoretically known from Gibbs phase rule. These variables together with a variable that quantifies a flow uniquely determine the coupling set of variables among components in a HVAC system. However, in many HVAC component modeling approaches, the mass flow is a known quantity and thus very often there is no need to consider pressure drop.

For moist air as the working fluid, the temperature and first and second phase mass flows are to be exchanged between programs. In case of water as the working fluid, temperature and mass flow will be sufficient. More discussion on this subject including the quality of the coupled variables can be found elsewhere (Trcka et al. 2006a).

In case of control loops where the sensed and the actuated variable are not in the same program, it is also necessary to communicate these variables between the coupled programs.
The internal synchronization procedure requires the knowledge about simulation time progress, and in the case the synchronization is managed in that way (see section on synchronization), a time stamp variable needs to be transferred between the coupled programs.

**Different mechanisms**

The current work is based on prototypes using specific building simulation environments, such as ESP-r, TRNSYS and EnergyPlus, as well as some smaller standalone component model implementations such as EARTH. However, the research outcomes such as exchange mechanisms and associated knowledge should be software and platform independent and are thus more widely applicable.

Depending on the context, transient behavior of HVAC components can be regarded as dynamic, quasi-static or steady-state. This distinction is important for the choice of coupling mechanism. In case of dynamic behavior it is important to keep track of the evolution of the results over time.

Figures 6 and 7 show two different coupling mechanisms. Figure 6 illustrates a coupling mechanism for a discontinuous-running external program (or federate). The base program invokes the external program and waits until that is finished before it continues itself. This mechanism is straightforwardly applicable for steady-state component models. However, the output of a transient simulation model is also a function of the components’ state at the previous time step. Since the discontinuously coupled program is restarted every time step, it does not have the information about component’s state from the previous time step, and thus for consistent dynamic evolution of the simulation results, the components state history needs to be externalized.

Figure 6 External coupling mechanism for a discontinuous running external program
Figure 7 External coupling mechanism for a continuous running external program

Figure 7 illustrates the coupling mechanism for a continuous running external program. Both programs run in parallel and exchange data in a certain user-defined manner. This mechanism is more suitable for transient component models, since the relevant state history is internally managed.

**COUPLING QUALITATIVE ISSUES**

**Synchronization**

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. Building performance simulations are normally in the time domain, where each time advance made by a program is of some fixed duration of simulation time. It is obvious that all information to be exchanged must have a time stamp. Federates should not receive information with a time stamp older than its current simulation time. Federates need to know whether all required information for the current simulation time step has been received. No federate should proceed to the next time step unless it received all data relevant for the current time step from other coupled federates.

There are two main approaches for synchronization (Fujimoto 2003) as follows.

- **Conservative** – in which precautions are taken to avoid processing data out of time stamp order, i.e. execution mechanism avoids synchronization errors.
- **Optimistic** – which does not necessary avoid synchronization errors, but rather use a detection mechanism and recovery approach, known as roll-back. Because it needs a state saving mechanism, enabling roll-back in an existing simulator requires major re-engineering (Page et al. 1999).

We distinguish internal and external time management approaches. Internal time management indicates that the synchronization checking procedure is coded within federates themselves. In that case, the time stamp is recognized as an additional variable to be exchanged. On the other side, the synchronization can be compassed within the inter process communication (IPC) mechanisms, applying blocking mode, for example. This is what is called by external coupling. In the prototypes presented in this paper the conservative approach is externally implemented.
**Coupling strategy**

In the run-time coupling approach each application runs separately, interacting with other applications through its boundaries. There are two different external run-time 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, Hoefliger et al., 2000), or onion coupling (Hensen, 1999).

In the former, distributed models run in sequence, where each model uses the known (from the previous coupling time step calculation) output values of the coupled model. The feedback between the programs is lagged for one coupling time step. Accuracy as well as stability constraints limit the simulation time step length in case of this strategy. Shorter the coupling time step, lesser the influence of the lagging. The latter coupling strategy requires that models iterate within each time step until the error estimate falls within a specified predefined tolerance. This improves the feed back, and as such it allows longer time steps for the same accuracy, but it requires an iteration procedure to ascertain user defined convergence criteria.

For coupling to a steady state component in a discontinuous manner, the employment of iterations does not pose additional issues. However, the situation is different for the continuous mechanism in any case, as well as for discontinuous mechanism applied for coupling to a transient component model (results depend on the simulation time evolution). It is either synchronization management issue, or history data management issue, or both that require a roll-back or time step rewind of the coupled program or the state history, if iterations are engaged, and thus increasing the effort for the code adjustments significantly.

Before the decision on the coupling strategy is made, its influence on the simulation results was investigated for different coupling time steps.

For the study a simple HVAC network was used as shown in Figure 8. Two models of the same systems were constructed. First model is a single monolithic ESP-r model of the whole system, while the other is a distributed model where two parts of the system are modeled in two coupled ESP-r models. To assess the sensitivity of coupling strategy, the simulation output of the monolithic (one-program) model (i.e. assumed to be fully-dynamically coupled) is compared with the simulation output of the distributed model (quasi-dynamically coupled). Several simulations were performed changing the length of coupling time step, i.e. the frequency of data exchange among programs.
Influence of coupling strategy on zone temperature calculation with different coupling time steps

The results in Figure 9 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 increased to ten minutes, oscillation that appears due to the particular combination of simulated system and control parameters 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, one can conclude that with an appropriately chosen coupling as well as simulation time step the differences between strategies are small. The loosely (quasi-dynamic) coupled simulations will produce the same quality results as strongly (fully-dynamic) coupled ones. As the starting point for the prototypes elaborated in this paper, and for reasons of simplicity, the fully dynamic approach is only applied with discontinuous mechanism, used for coupling to steady state component models, while the quasi-dynamic approach is used in general.

Inter vs. intra time step data exchange

In terms of system component modeling two main 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). Advantages and disadvantages of these approaches are addressed elsewhere (Hensen 1996).

External coupling will result in different time step variable exchange depending on which approach is used, as shown in Figures 10, 11 and 12. The circles represent the component state and its output at a specific moment in time. The arrows indicate the information flow, which (in terms of location) follows the fluid flow; i.e. from sending to receiving component. The grey boxes indicate the placement of the external component in the overall model.
When the base federate uses a fully implicit numerical scheme, a plant system matrix is solved at each time step. However, when an external component is interconnected its dynamical and physical behavior is unknown. Therefore, it is not possible to uniquely define its dependencies with other parts of the system and only an explicit determination of temperature, first and second phase mass flows as calculated by the external program can be used.

Further, if the exchange of data takes place before solving the system matrix, the thermodynamic state of the incoming connection for the future time step of the component is still to be solved. This means that the values of the variables that describe the thermodynamic state of the incoming connection, calculated for the last time step, will be forwarded to the external federate. Based on these values the external program performs the calculation and sends the data back to the base program.

Inter-programs time step variable exchange will disturb the original intra-time step variable exchange of the base program that uses an implicit numerical scheme (Figure 10).

However, if an explicit numerical scheme is used (Figure 11) the external coupling will keep the original intra time step data exchange consistent. The same applies for input-output based component modeling approach (Figure 12).

Implement & implementation

In order to prototype the above, additional features had to be developed for legacy domain tools, e.g. for ESP-r, TRNSYS and EnergyPlus. The following discussion is limited to changes within the ESP-r software as an example.

Additional components have been developed within the ESP-r simulation environment in order to implement and test coupling of continuous and discontinuous running external models of either air or water.
systems (Figure 13). Instead of additional components, additional connections (in ESP-r terminology) could have been used as well. From the solution point of view both approaches are identical. However in case of connections, it would be very important to take care of the order in which the connections would be defined.

The mechanism for the discontinuously running external program uses intermediate files. In case of the continuously running external program, data transfer is via so-called named pipes. In both cases it would be possible to use other inter-process communication (IPC) mechanisms as well.

In case of the named pipes IPC, process blocking and synchronization is provided as part of the pipe services. For IPC mechanisms, which do not provide this service, a time-stamp variable could be added to the set of exchanged data for checking and synchronization purposes.

Figure 13 shows the process of variables exchange and places where original code had to be altered to accommodate the requirements for run-time exchange of selected variables. The figure shows main subroutines involved in simulation of an HVAC system model. We will address only the subroutines which code had to be adjusted to accommodate the coupling.

**Figure 13** External coupling implementation in ESP-r within one time step

(MZPADJ, CONTRL, MZPMSU, and MZNASS are ESP-r-specific subroutines)

CONTRL subroutine determines system control status based on most recent available results. To enable the exchange of control data, “fictive” control components and additional control low were constructed. The role of the new control low is simple. It copies relevant information from its source and transfer it to the coupled program. “Fictive” components are only used to enable the standard system/control definitions in ESP-r, and do not have any influence on the solution.

MZPMSU subroutine sets up the system equations in a matrix form. It calls components’ subroutines that generate matrix coefficients and locate them in the system network matrix. Two new component’s subroutines were constructed (for each coupling mechanism), one as an air-loop component and the other as water-loop component. The components interface the coupled program and send/receive data to/from it. ESP-r model does not need to have any knowledge about a coupled subsystem simulation model. If coupling time step differs from (greater then) the simulation time step employed internally, the coupled variables are kept constant, and equal to last transmitted values, during coupling time step calculations.
TEST CASE STUDY

As an illustration, consider the greenhouse with air recirculation through an earth-to-air heat exchanger from (Ghosal, et al. 2004) shown in Figure 14. The objective here is to assess the energy saving potential of the ground-coupled heat exchanger consisting of buried pipes.

The greenhouse itself is modeled in ESP-r, which currently lacks a model for an earth-to-air heat exchanger. EARTH, a program that models and simulates earth-to-air heat exchanger is run-time coupled to overcome this deficiency. ESP-r and EARTH are continuously coupled through named pipes implementing quasi-dynamic coupling strategy and conservative, externally implemented synchronization procedure. The EARTH model takes into account the dynamics of the ground storage. Using the continuous coupling mechanism the history state data do not have to be externalized. New ESP-r component that interfaces external program sends the information about the working fluid state and its flow through the named pipe. On the other side of the named pipe, the external program, i.e. EARTH waits for the information, and when it is received the program performs calculations. i.e. evolves in simulation time for one coupling time step (that does not have to be equal to the simulation time step). The simulation results are then transferred to ESP-r, again through the pipe. As mentioned earlier, named pipes have services that provide synchronization procedure. Read and write operations to a named pipe are blocking, by default. If a process reads from a named pipe and if the pipe does not have data in it, the reading process will be blocked. Similarly if a process tries to write to a named pipe that has no reader, the writing process gets blocked until another process opens the named pipe for reading.

Figure 14 Greenhouse coupled to an earth to air heat exchanger

Some simulation results are presented in Figure 15. These are for three one-day simulations (this is only for demonstration purpose, as the dynamics of the ground heat storage itself are not visible for such short simulation time) using climate data for New Delhi, India: one without coupling the external exchanger, and two coupled simulations with either 350m3/h air volume flow rate and 50m pipe length (design 1) or 700 m3/h air volume flow rate and pipe length 120m (design 2). The programs’ simulation time steps equal to coupling time step and its value is kept to one hour.

As expected, the air temperature in the greenhouse varies less when it is coupled to the earth-to-air heat exchanger. The variations are smaller with design 2. The greenhouse is heated during the night and cooled during the day. For higher volume flow rates the difference in temperatures can be as high as 7 degrees Celsius.
Figure 15 Air temperature profiles: ambient temperature, zone temperature for free floating conditions, zone and outlet temperatures from ground coupled heat exchanger for design 1 and 2

The variations of the outlet temperature of the ground coupled heat exchanger depend on a specific heat exchanger design and variations of the temperature within the zone. The heat exchanger and the greenhouse are strongly coupled and only by coupled simulation these interactions can be taken into account.

The heating/cooling potential of the exchanger is shown in Figure 16. It is calculated from the difference in temperature inside the house and temperature at the outlet of the heat exchanger. As can be seen, it depends on the design and varies over the day. In the early and late hours the greenhouse is heated by the earth to air heat exchanger and in the mid day it is cooled down.
Figure 16 Change in ventilation heat capacity provided by the ground heat exchanger, calculated on the basis of difference in zone and outlet temperature of the exchanger

IN CONCLUSION

The external run-time coupling approach promises to be very flexible in several respects. Current limitations of non-shared developments in HVAC component modeling can be easily overcome as the various parts of a building with systems configuration can be simulated in different environments. 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.

We recognized the potential of distributed simulation use in the building performance simulation and explored its benefits. It may be argued that a simulation environment, able to address all the questions which may come up in the design, operation and maintenance of HVAC systems, is like a giant puzzle. We see the work presented here as a small part of this puzzle, which aims to enable the communication between existing tools and in doing so will enable a more flexible use of simulation.

However, an extra effort is required to allow legacy tools to interface other (legacy) tools. The approach undertaken in this paper was to construct separate "interface" components in each environment. Through the interface components, coupled programs are able to exchange relevant information. Further, the communication is done employing one of the IPC mechanisms. The same IPC mechanism has to be employed by the coupled programs, as well as the sending components has to have knowledge of what data is being received, i.e. mass flow, temperature, control related data, etc. Therefore, a standard protocol is required for external coupling implementation in general. Mostly due to the waiting procedure, employed by the prototypes, the distributed simulation requires longer execution time compared to the monolithic simulation.

This paper presented coupling strategies implemented in a prototype. The advantages and disadvantages of each were considered. The implementation of the working prototype has been demonstrated on an example case study.
REFERENCES


6 Curtil, D. 2004. personal communication


9 Forbus K.D. 1996. “Qualitative reasoning”, The Qualitative Reasoning group, Institute for the Learning Sciences, NorthWestern University, Evanston, IL, USA, DRAFT: Chapter for the CRC handbook of computer science.


LIST OF CAPTIONS FOR FIGURES

Figure 1
External run time coupling

Figure 2
Distributed modeling and simulation

Figure 3
Interprocess communication (IPC) taxonomy (McGregor 2005)

Figure 4
Flow chart of the decision-making methodology

Figure 5
External coupling necessity: “−” – inadequate approach for specific schema; “+” – applicable approach for specific schema; “o” – refers to an approach that is applicable for specific schema, but more complex than its alternative and does not bring extra benefits.

Figure 6
External coupling mechanism for a discontinuous running external program

Figure 7
External coupling mechanism for a continuous running external program

Figure 8
Simple HVAC network modeled in distributed fashion

Figure 9
Influence of coupling strategy on zone temperature calculation with different coupling time steps

Figure 10
Inter and intra time step data exchange with an external component assuming that the base program uses a fully implicit numerical scheme

Figure 11
Inter and intra time step data exchange with an external component assuming that the base program uses a fully explicit numerical scheme

Figure 12
Inter and intra time step data exchange with an external component assuming that the base program uses an input-output based approach

Figure 13
External coupling implementation in ESP-r within one time step
(MZPADI, CONTRL, MZPMSU, and MZNASS are ESP-r-specific subroutines)

Figure 14
Greenhouse coupled to an earth to air heat exchanger

Figure 15
Air temperature profiles: ambient temperature, zone temperature for free floating conditions, zone and outlet temperatures from ground coupled heat exchanger for design 1 and 2

Figure 16
Change in ventilation heat capacity provided by the ground heat exchanger, calculated on the basis of difference in zone and outlet temperature of the exchanger