Using Systems Engineering Practices for Distributed Control and Building Performance Simulation

Azzedine Yahiaoui*, Abd-El-Kader Sahraoui
*Center for buildings and Systems, Eindhoven University of Technology (TU/e)
P.O. Box 513, 5600MB Eindhoven, the Netherlands
e-mail: a.yahiaoui[at]tue[dot]nl
LAAS-CNRS, 7 avenue du Colonel Roche, F-31077 Toulouse, France &
Université de Toulouse: UPS, INSA, INP, ISAE, LAAS; F-31077 Toulouse, France
e-mail: sahraoui[at]laas[dot]fr

***Abstract*— A distributed simulation between control systems and building performance applications is becoming more and more an invaluable tool in the analysis of Automated Buildings (ABs) for better operation and design. In order to simultaneously fulfill the occupants' needs while reducing energy consumption and greenhouse gas emissions - instead of costly and time-consuming experiments - distributed simulations are required to investigate the impact of advanced control systems on building performance applications. This paper describes the development and implementation of a framework for distributed simulations involving different software tools at the same time over a network. The main role of this framework is to run-time couple one or more building performance simulation tool(s) with a control systems environment over a network, as organized and sorted by similarity to Building Automation and Control Systems (BACS) architecture. Finally, the paper ends with some conclusions and perspectives for future work.**

Keywords- systems engineering; distributed dynamic simulation; building performance applications; control systems; automated building; building automation and control systems

I. INTRODUCTION

Design studies of Automated Buildings (ABs) are becoming more complex, usually because of more stringent needs of the occupants and the environment (i.e., effective control of building indoor processes such as heating, humidifying and lighting while at the same time minimizing the energy consumption and contributing to a reduction of greenhouse gas emissions). A distributed simulation between control systems and building performance applications is growing rapidly towards the preferred means for supporting such design studies. Along with the development of the required diversity of control functions, the fulfillment of the occupants' needs, and the reduction of energy consumption and greenhouse gas emissions, it is necessary to consider several practical aspects such as economical factors. According to [44] and [45], Distributed Control Systems (DCSs) are usually referenced as Networked Control Systems (NCSs). Accordingly, DCSs or NCSs are control systems where control units, actuators, sensors and other components communicate over a network. This type of control systems has several advantages over traditional control systems, such as greater flexibility, reduced wiring, lower installation costs, effective to fault diagnoses and maintenance procedures, control and detection in distribution, easy to expend, and so on. However, the use of communication networks within a control loop makes the analysis and design of a DCS complex as it inevitably brings new problems such as communication-induced time delays, potential loss of information, communication congestion, timing disorder, precise synchronization among nodes, etc. In general, these problems degrade both the performance and stability of the controlled systems. For this reason, a distributed simulation mechanism is developed, in this study, to simultaneously simulate both physical and communication network dynamics. The use of experiments for testing and analyzing new control systems in buildings is still an option, but they are time consuming and cost-prohibitive. For example, an evaluation of a designed control system or calibrating the internal parameters of a control system designed for a building indoor process requires at least 24 hours to obtain the results. In contrast, a simulation takes just a few minutes or less than an hour. Therefore, more and more, distributed simulations between control systems and building performance applications is becoming an invaluable means for supporting design studies of ABs. It also makes some analyses of ABs for better design and operation that are not possible otherwise.

There have been a number of research studies~~,~~ in which different control systems/strategies were applied to building heating, ventilation, air-conditioning, and refrigeration (HVAC&R) equipment and lighting components. For example, Levermore [24], Kalogirou and Bojic [20], Ben-Nakhi et al. [1], Kolokotsa et al. [21], Mathews et al. [29], and Gonzalez et al. [10] presented works related to building energy management strategies through cost reduction and energy consumption savings, while Lute and Paassen [27], Kummert et al. [22], Calvino et al. [3], Yahiaoui et al. [38] Liang and Du [23], and Chen et al. [4] presented works with a

focus on fuzzy, neural network, optimal and predictive control systems of thermal conditions in buildings. In addition, Sharples et al. [32], Kamphuis et al. [18], Jelsma et al. [17], and Kamphuis et al. [19] introduced the use of Multi-Agent Systems (MASs) for managing energy and thermal comfort in buildings. However, most studies are often devoted to the particular control application and specific use of a given building. Moreover, the studies related to the application of MASs were only introduced with ~~a~~ field experiments that were set up in several offices. For this reason, the present paper introduces the concepts and core issues of a structured and rigorous approach, based on SE practice, to integrate advanced control systems in building performance applications. This approach distributes one or more building performance simulation software tools and control systems modeling environment over a network. The overall goal of this approach is to provide practical solutions for enabling the application of multi-variable control systems to building HVAC&R equipment and lighting components, and particularly to improve distributed control applications such as MASs in ABs.

ABs are a class of buildings~~,~~ which are able to accrue economic and environmental benefits by the use of computer-based monitoring to coordinate, organize, manage, adapt and optimize building HVAC&R equipment, lighting components, and facilities related to the maintenance of fire safety and elevator function, among other functions [37], [40]. Among the many other names used to refer to ABs, the most common are building automation (BA), smart buildings (SBs), and intelligent buildings (IBs). The term ABs is used in this paper because it best describes the importance of integrating automatic control systems and intelligent control technologies into a building's environmental performance.

Specifically, ABs are composed of numerous sensors, actuators, and control systems interconnected in such a way to facilitate and adapt a suitable control strategy and/or optimum control reference (or set-point) from the central computer-based monitor system. These basic activities of ABs have been the subject of BACS since the last century. Generally, modern comprehensive BACS use the all-encompassing term building automation systems (BAS) when referring specifically to their control designs, although the terms energy management systems (EMS), building energy management systems (BEMS), building management systems (BMS), and intelligent building management systems (IBMS) are still used, sometimes intentionally to refer to specific functional aspects, but more often by habit [19]. All these names refer to BACS, which greatly increase the interaction of plants systems within buildings, improve occupant comfort, reduce energy use, and allow for distribution of building operations over a network. The relevant international standard uses the term BACS as an umbrella term [13].

Another dimension of BACS architecture is the application of advanced protocol for data communication and information exchange between a central computer and building HVAC&R equipment and lighting components. Other main functions of BACS architecture are effective and efficient management facilities to promote greater occupant satisfaction and productivity, as well as advanced structural design and innovative materials. As described by researchers, e.g. [29], BACS architecture can also integrate systems to improve the response of a building to earthquakes. Accordingly, several communication protocols such as BACnet, LonWorks, and Modbus have been developed for high performance networks used in BACS architecture [13], [12]. Recent approaches have improved the ability of BACS architecture to adjust building performance applications by providing it with the ability to detect climactic changes and occupant behavior [32], [40].

Because HVAC&R equipment and lighting components become important when addressing energy consumption and environmental comfort aspects, the development of their appropriate control systems requires a multidisciplinary approach in order to provide healthy and comfortable conditions for building occupants. In response to this consideration, a systems engineering (SE) approach has been found effective in coordinating multi-disciplinary applications to enable the realization of successful control systems within buildings. Rather than viewing a project as a collection of separate sets of functions and entities, SE concepts take a holistic view of all the aspects of the project as a complete system, aiming for aggregation of an end (or final) product to achieve a given purpose or solve a particular problem. Therefore, the final product is a constituent part of a system that performs operational functions while continuously fulfilling user requirements (i.e., occupants' needs or references).

The remainder of this paper is organized as follows: Challenges to distributed control and building performance simulation are introduced in Section 2, followed by development and implementation in Section 3, and then example of application in Section 4. The conclusion and perspectives for future work are presented in Section 5.

II. CHALLENGES TO DISTRIBUTED CONTROL AND BUILDING PERFORMANCE SIMULATION

Computer simulation of building performance; thermal, visual, energy consumption, and acoustic conditions; and control systems is of particular importance for the modeling of building performance applications, as simulation practices are complex coupled tools in which all of these aspects interact dynamically. This simulation must take into account various technical aspects, such as comfort, safety and occupants needs or preferences. Integrating such a technology is not "yet another add-on artifact", but a balanced approach that preserves invariant properties with the additional constraint of cost reduction. The traditional slogan "faster, better, and cheaper" applies here.

The current situation is that, on the one hand, there exists a domain based control systems modeling environment very advanced in the analysis and design of control systems but still limited in building performance simulation concepts (e.g., Matlab/Simulink).

On the other hand, domain-specific building performance simulation software (e.g., ESP-r) is usually relatively basic in terms of control modeling and simulation capabilities. Marrying the two approaches by run-time coupling building performance simulation software and control systems modeling environment could enable integrated building performance assessment by predicting the overall effect of innovative control strategies in a building indoor environment [37], [41]. By extending this potential in distributing one or more building performance simulation software tools and control systems modeling environment over a network, this would result in a typical pattern of distributed simulation between control systems and building performance applications as qualified by similarity to BACS architecture [41].

Distributing different applications (simulation tools) on the same environment provides the facility the ability to exchange data and events in a distributed and co-operative way. Usually, one application controls the overall simulation procedure at run-time and requests the other application(s) when necessary. Previous and ongoing work by others, particularly for the purpose for building control applications, include coupling between lighting and building energy simulation [16], between computational fluid dynamics programs and building energy simulation [38], and between systems and building energy simulation [5]. However, all the approaches addressed up until now are limited to a particular application, and are often based simply on the coupling of two simulation tools running on the same machine. Moreover, all of these approaches cannot be used for the representation of distributed control applications such as MASs in simulation. For this reason, a distributed dynamic simulation mechanism between one or more building performance simulation tool(s) and control systems environment by run-time coupling over a network is developed in this paper with the purpose to ensure that this approach has a more general and wider applicability.

III. DEVELOPMENT AND IMPLEMENTATION

A. Overview of integrated building control systems

Fig. 1 shows a schematic example of an integrated building control system, where control systems and building zone and/or plant systems are integrated over a communication network and form together a distributed system.

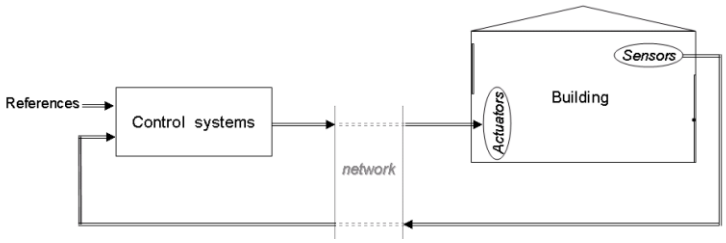

Figure 1. Integrated building control system

B. Description of BACS architecture

ABs are buildings that are controlled by BACS, and often referred as integrated building control systems. BACS is an example of a DCS because it uses a computer-based control system to automatically monitor and control a range of building performance applications including heating, ventilation, air-conditioning, lighting and other tasks such as access control, energy management, and fault diagnoses in a building or a group of buildings via a network. While this has several advantages, it also inevitably brings problems due to the network. Fig. 2 shows a complete BACS architecture that can be described at four main levels [12], [13], [37]:

- The *management level* consists of a central computer used for managing, storing, and analyzing data, communicating with external systems, and operating building equipment and components.
- The *network level* consists of an open protocol connected to the internet through routers used to exchange data between the central computer and substations.
- The *automation level* consists of one or more substations used for interfacing building HAVC&R equipment and lighting components to the network.
- The *field level* represents the low level where building HVAC&R equipment and lighting components (sensors and actuators) and final users are located.

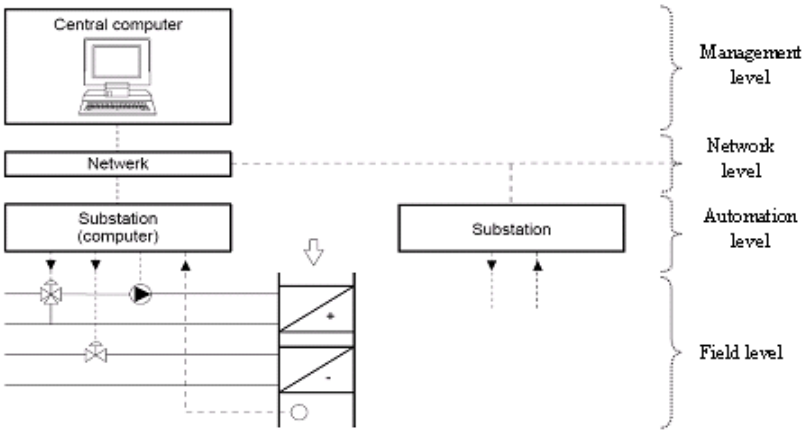

Figure 2. BACS architecture

Because BACS uses a network for data exchange between the central computer and substations (or terminals), this can degrade both the performance and the stability of building HVAC&R equipment and lighting components. The most straightforward way to evaluate such problems without a full-scale implementation of BACS is through a modeling and simulation approach. Therefore, successful design and implementation of BACS require the existence of a simulation tool that allows evaluation and verification of different control and network algorithms. As a consequence, a distributed simulation mechanism is developed and implemented specifically for building control applications to simultaneously simulate both physical and communication networks dynamics.

C. Systems Engineering Practice

SE practice is not new, but the discipline is. It started with large-scale programs in USA, mainly in aeronautics [28], space [32] and particularly in defense [6], [13]. Furthermore, it is becoming popular in countries with well-established aeronautic and military industries; since the late nineties it has been deployed in manufacturing, automotive [24] and recently adopted by the Society of Manufacturing Engineers (SME) [31]. Simply defined, SE is an interdisciplinary approach encompassing the entire technical effort to evolve and verify an integrated and life-cycle balanced set of system, people, product, and process solutions that satisfy customer needs. SE encompasses:

- the technical efforts related to the development, manufacturing, verification, deployment, operations, support, disposal of, and user training for system products and processes;
- the management of the system configuration;
- the translation of system definitions into work break-down structures, and
- information for management decision making.

D. Basic Systems Engineering Process

Without a flexible, yet structured and rigorous approach to solving complex problems involving advanced control systems and building performance applications, practical aspects - including funds and time - can be wasted either by solving the wrong problem, developing an incomplete solution, or over-developing an appropriate (or good) solution. Because the factors affecting the problem definition are often dynamic in the real world, they require a process that is adaptable to changing parameters, yet structured in a way that minimizes lost effort. The SE concept uses the following-steps [2], [33], [11], [39]:

1) Determine the requirements or needs that the solution should fulfill
   - ~~*a)*~~ Define top-level global end-user requirements (or occupant needs)
   - *b)* Perform functional analysis to divide top-level global requirements into low-level local requirements and determine alternate means of achieving the top-level requirements.
   - *c)* Define the inter-relationship between the top-level and low-level requirements, if applicable.

2) Develop concept design(s) that will satisfy all the requirements.

3) Evaluate the proposed concept(s) and decide on most promising approach(s).
   - *a)* Perform trade studies to identify weaknesses and risks and choose the best solution.
   - *b)* Evaluate and optimize to eliminate and minimize weaknesses and risks.
   - *c)* Quantify compliance of concept design(s) relative to top-level global requirements

4) Fully develop the concept design(s) chosen in the previous step.
5) Verify that the system or program meets the top-level global requirements.

Fig. 3 shows a typical example of the V (or Vee) diagram where steps 1) and 3) are interactive. In this paper, the V diagram is used with the SE approach, although other common diagrams such as waterfall and spiral can also be considered. In general, their utilization depends on the application and its usage including user requirements and operating conditions.

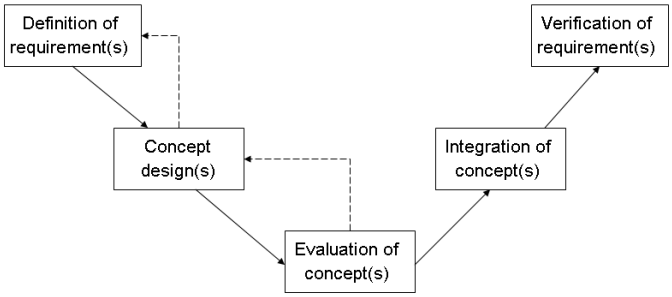

Figure 3. A typical example of the SE approach

The above SE process, or a modified version of it, is often used by organizations, such as [7], for developing new products or solving day-to-day problems because it is natural to follow. However, it is sometimes lacking as a disciplined and systematic framework for quantifying and documenting the various steps, as it can be a less structured process that allows the results to be influenced by chance, limited or irrelevant knowledge and experience, intuition, or other factors [42].

E. Define Systems of Systems

The deployment of SE practice can be carried out in a comprehensive manner by separating the final product (i.e., building itself, its HVAC&R equipment and lighting components, etc.) from the enabling product (i.e., control systems, automation, etc.) and development product (simulation tools, etc.) this can be best illustrated by Fig 4.

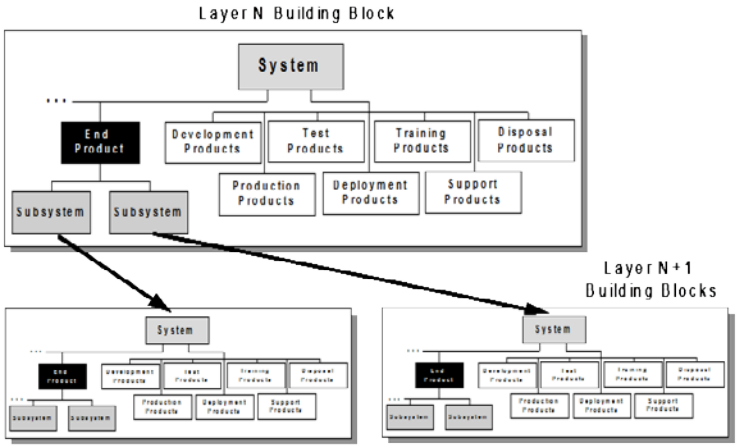

Figure 4. Hierarchy of building blocks

A single block will define the complete solution a complex problem more typical of the design project. When an end product sub-system requires further development it will have its own subordinate building block. Once the descriptions of the end product of the initial building block are completed, and preliminary descriptions of the end product subsystems are defined, the development of the next lower layer of building block can be initiated.

F. Context and Application

The context in this study is related to the development and design of advanced control systems in buildings, especially in building environmental performance. The integration of building science engineering, architecture, construction management and risk assessment for new construction projects and existing buildings becomes a must. Building control and performance applications require a lifecycle of development as shown in Fig 5.

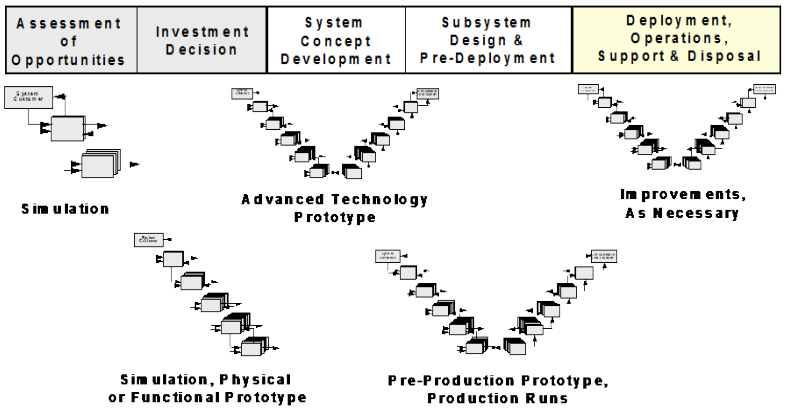

Figure 5. Enterprise-based life cycle phases

In this context, this study is focused on the systems development (including subsystems design and deployment operation) to the deployment operations concerning the development and implementation of the initial systems as a final system (or a prototype) for distributed control and building performance simulations.

G. SE Structured Approach to Developing and Implementing Distributed Control and Building Performance Simulation

Fig. 6 shows a structured approach to a conceptual design of distributed simulation between control systems and building performance simulation [41].

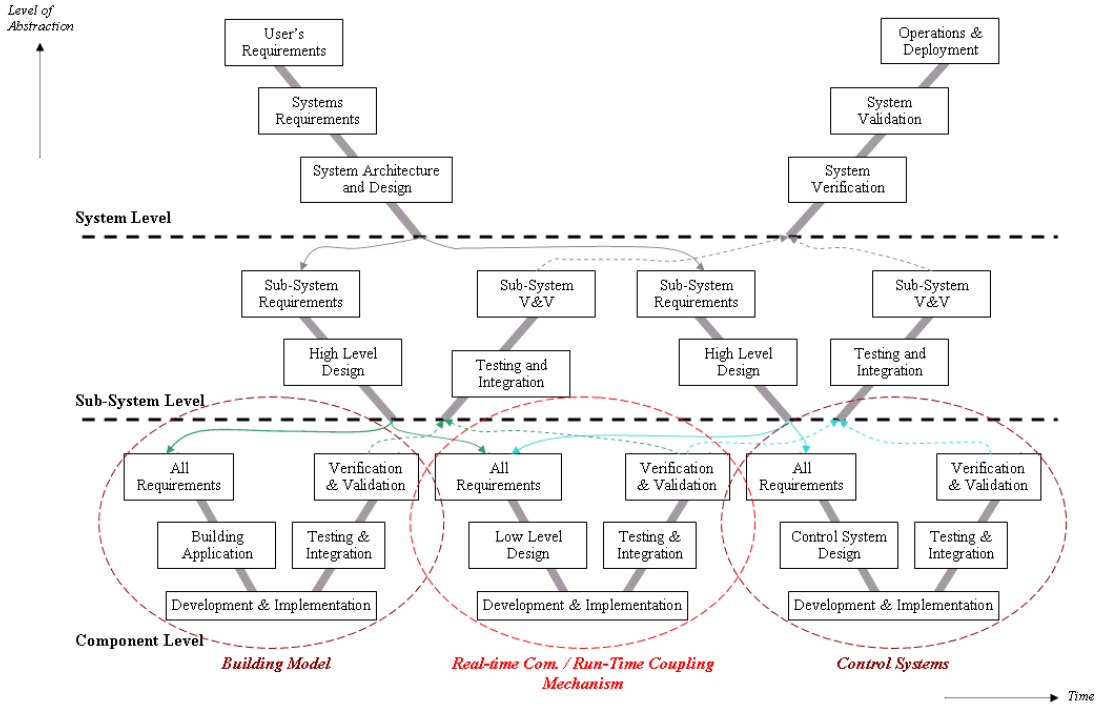

Figure 6. A hierarchical approach to the systematic characterization of distributed control and building performance simulation

In this approach, the main applications of BACS architecture are characterized from the functional viewpoint at different levels of abstraction and organized hierarchically into three levels to allow for easy understanding of the nature of their differences. In accordance with the System of Systems (SoS) concept, one or more V diagrams are developed for each of the interrelated applications that describe a number of phases at each level. With the aid of a practical technique, such as a taxonomy method, adequate methods and/or tools for use at each phase can be identified for managing the complexity of distributed system by decomposing it into subsystems and then components. As shown in Fig. 6, the single V lifecycle diagram can be divided horizontally to distinguish a system level, sub-system level, and component level, respectively.

From an end-user perspective, i.e. at a top-level of abstraction, a building model and its control system is seen as one space (or zone) containing several devices necessary to regulate its indoor environment processes according to certain references set by the occupants. According to SE thinking, this space can be likened to an integrated set of two sub-systems - a building model and its control system - with the latter being the so-called 'building control application'. At the mid-level of abstraction, this integrated set is represented as the two independent sub-systems of a building model and its control system functioning within a cooperative environment such that the control system achieves a desired reference state, as set by the occupants according to the state of the space and its environment. At the bottom level of abstraction, this cooperative environment is perceived as a system component that ensures the exchange of data between these two different sub-systems, i.e. the building model and its control system. As a result, this represents what has been termed as a distributed control system as it particularly refers the application of control systems in buildings indoor environment, as shown in Fig. 1.

H. Development and Implementation of Run-Time Coupling

In [35], [36], [37], it has been demonstrated that using internet sockets is the best means of implementing run-time coupling between Matlab/Simulink and one or more ESP-r(s) because it supports distributed simulation over a network, allowing data exchange between a building model and its external (or remote) control system in different communication modes including synchronous, asynchronous, and partially asynchronous, as shown in Fig.1. The main advantage of using this Inter-Process Communication (IPC) mechanism lies in the fact that although the building model and its external control systems are built separately and can be located on different machines running different Operating Systems (OS) such as Unix-variant and MS-Windows, they work together by exchanging data in a common format including ASCII, binary and Extensible Markup Language (XML) through a network. Fig. 7 illustrates the proposed approach to run-time coupling between Matlab/Simulink and ESP-r.

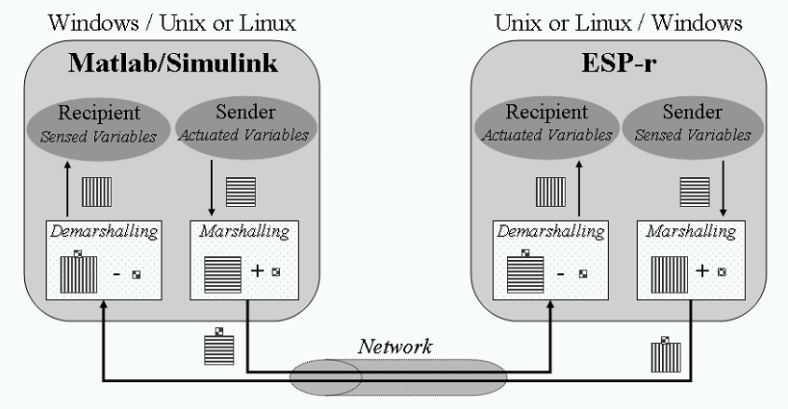

Figure 7. Run-time coupling between Matlab/Simulink and ESP-r

In effect, run-time coupling is implemented with internet sockets in order to facilitate data exchange between Matlab/ Simulink and ESP-r when they are concurrently operating either on the same machine or to increase the speed of simulations, on separate machines connected by a network. In addition, when Matlab/Simulink and ESP-r are located on different machines that run on different OSs and/or use different data formats by initiating protocols, such as Bacnet and LonWorks, run-time coupling can be designed in a way to support portability and distributed dynamic simulations over a heterogeneous network (i.e. on different machines with different OSs and/or different data format protocols). For this reason, in this work different methods for marshalling and demarshalling (or unmarshalling) data over a network were implemented within run-time coupling to convert data (i.e. sensed or actuated variables) into a form of external network representation (e.g. a byte-stream) and then back to their native format prior to access by a building model and/or its control systems, respectively.

The major advantages of this run-time coupling mechanism are that it requires any simulation of a building model and its control systems be built separately using ESP-r and Matlab/ Simulink, respectively, and that it provides the preferred means to complete interoperability tasks in a fashion with no or mirror user interferences. Therefore, it requires only modeling a building model on ESP-r and its control systems on Matlab/ Simulink, and then indicating their interfaces by specifying the port-numbers, modes of exchange, or variables that they will use to import or export data to or from each other.

1) Interfacing Client Socket to ESP-r

Because ESP-r, e.g. [9], is almost completely written in Fortran programming language and socket Application Programming Interface (APIs) can only be implemented in programming languages such as C/C++, mixed-language programming using Fortran and C++ must be used to interface between Fortran and C/C++ programs [8]. Therefore, mixed-language programming is used to develop and implement an approach combining a Fortran common block with global C/C++ extern data structures (or *extern*

structs) of the same name in order to enable the addition of new variables that need to be exchanged with Matlab/Simulink without making large modifications in the existing programming codes.

ESP-r was modified and extended to enable users to obtain data on sensed and actuated variables in the external control systems of building zones, plant components, and/or mass-flow networks, and to choose settings (including server IP address, port number, current process number, network protocol, communication mode, and data-exchange format) for run-time coupling. The added Fortran subroutines that exchange data with Matlab/Simulink and functions indicate when initiating and ending simulations are combined together with socket APIs of the C/C++ client code separately. The C/C++ client code was developed in a hierarchical way in order to support all possible combinations of exchanged variables and settings that a user could choose in run-time coupling with Matlab/ Simulink. Compiling the modified and extended ESP-r code together with the socket APIs of the C/C++ client code generates executable ESP-r, respectively, and allows ESP-r to run as a client process.

2) Interfacing Server Socket to Matlab/Simulink

Matlab/Simulink, [30], has a built-in utility called Matlab EXcutable (MEX) that is often used to convert Fortran, Java or C/C++ programs to a MEX format. The original sense of the Matlab/Simulink word represents two different environments, which are a high-level technical programming language and a graphical block-diagram interface. Depending on which environment is interfaced, two main approaches can be used to link external programs written in C/C++:

- For Matlab, MEX-files are used with dynamically linked programs that, when compiled, can be called from within Matlab in the same way as M-files or built-in functions. In case we need to deal with Simulink, the links can be performed between each other by just using "*sim*" functions.
- Practically the same procedure is adopted by Simulink, although MEX S-functions are used with dynamically linked programs that, when compiled, can be called from within a Simulink block diagram. However, when there is a need to deal with Matlab, the link should be done via M-file S-functions that are more complicated than using a straightforward "*sim*" function.

The first approach is preferable not only because it is less complex than the second approach but also because it offers more advantages, such as 1) the capability to manage a high number of exchanging variables simultaneously, 2) the versatility needed to meet the requirements of run-time coupling, and 3) the ability to implement functionalities that are not accessible to M-file S-functions. Although the MEX-files were originally designed to allow the inclusion of external routines written mainly in C/C++, they are also capable of integrating external shared libraries, such as socket APIs, into Matlab. For these reasons, a MEX-file was used for the development and implementation of the *matespexge* toolbox.

By combining MEX-file functions and socket APIs, access from ESP-r to Matlab and Simulink functionalities, especially to the application toolboxes for advanced control systems, is realized by invoking the name "*matespexge*" from the Matlab prompt. Once the *matespexge* toolbox has been executed, a graphical user interface including icons and menus will display and provide the dialogue for users to create M-files to remotely control a building zone, plant, and/or flow model as built on ESP-r accordingly. Further access from these M-files to Simulink can be obtained by using "*sim*" functions, although access from Simulink to Stateflow should be obtained by incorporating a Stateflow block in the Simulink block diagram. Moreover, these M-files include Matlab functions that contain the left- and right-hand arguments with which the MEX-file is invoked. Therefore, the *matespexge* toolbox was designed with the use of MEX-files that include facilities for enabling run-time coupling between Matlab/Simulink and one or multiple ESP-r(s). After compiling the *matespexge* toolbox, a dynamic executable file is generated with an extension corresponding to the OS over which Matlab/Simulink is running.

Within the implementation of the *matespexge* toolbox, the external routines that specifically exchange data with subroutines for building zone, plant, and flow network modules of ESP-r are encapsulated into a single MEX-file. A global identifier is also integrated to determine which building zone, plant, and/or flow network model will exchange data with the created control file. Due to this fact, the user must provide valid information (i.e. as stated in ESP-r) on the input interface. In addition, as Matlab is an interactive tool, the handling callbacks from ESP-r are ensured by default in order to access Matlab/Simulink as a computational engine. For these reasons, the matespexge toolbox is designed in such a way to let Matlab/Simulink operate as a server process when its created control files are invoked by one of the three ESP-r modules.

Because Stateflow can be used together with Simulink for the simulation of MASs, the use of the *matespexge* toolbox becomes essential for enabling the integration of advanced control systems, such as hybrid systems and MASs in building performance simulation. It enables a user to interactively build, test, and simulate distributed applications between ESP-r and Matlab/Simulink, even when both software tools are running on separate and different OSs. Therefore, it is a key solution in enabling multi-variable control systems of building performance applications that had previously not been feasible.

I. A Practical Approach to Representing BACS Technology in Simulation

The BACS technology, as shown in Fig. 2, consists of several substations or terminals that communicate with a central computer over a network to coordinate the control actions and processes in ABs. Although representing BACS technology in a simulation remains complex, the development of run-time coupling between Matlab/Simulink and multiple ESP-r(s) can allow for identification of practical solutions for enabling the integration of advanced control systems in building performance simulation, and improving distributed control applications in ABs for better operation and design. As one of the constraints of BACS is the network-induced time delays, distributed simulations are required to analyze and simulate both the performance and stability of controlled building HVAC&R equipment and lighting components in ABs. The necessity for distributed simulations originates from the fact that BACS is an interdisciplinary field that requires the study of both control theory and communication networks in design.

If it is assumed that Matlab/Simulink represents a central computer and ESP-r represents a terminal in a manner analogous to BACS architecture, the IPC mechanism used to run-time couple them is designed to support cooperative applications through interoperable middleware that provides a layer facilitating the interface of any building model with its control system. In effect, the use of an IPC mechanism with the middleware not only simplifies data management and distribution over a network, but also provides for the independence and transparency of data exchange between building models and their control systems. This IPC mechanism also allows web-services to be highly portable in distributed simulation while remaining similar to BACS architecture. For these reasons, this work has enhanced the traditional conceptual approach of distributed control and building performance simulation in order to run-time couple Matlab/Simulink with multiple ESP-r(s) via a network, as shown in Fig. 8.

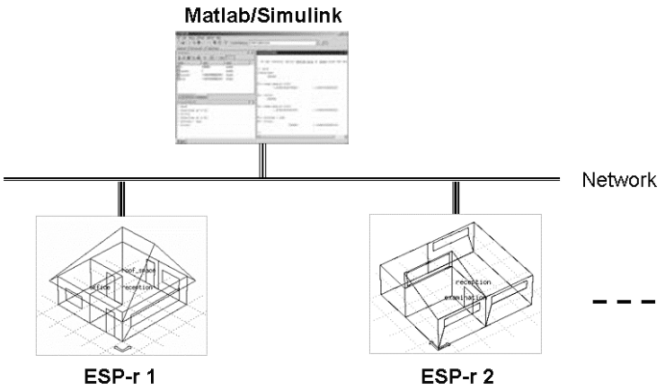

Figure 8. A practical approach to run-time coupling between Matlab/Simulink and multiple ESP-r(s)

This work thus proposes a novel approach to creating and using multiple ESP-r(s) in distributed and parallel simulations with Matlab/Simulink over an open data communication protocol, as occurs when using BACS technology. The practical framework shown in Fig. 8 was developed and implemented using a multi-threads method, for which an equivalent application to BACS architecture can concurrently be executed across multiple machines distributed over a wide-area network. In a similar application to BACS architecture, each ESP-r in the framework is used to model either parts of or an entire building, while Matlab/Simulink is used to model all control systems remotely. The number of connecting ESP-r clients to Matlab/Simulink is currently limited to a maximum of 9 due to the extreme difficulty of managing all exchanged variables while controlling the simulation. Nevertheless, the work developed here can be easily extended to support more ESP-r clients, as is the case with certain applications of real ABs.

J. System-Level Design of Run-Time Coupling

An SE approach to the system-level design of run-time coupling between Matlab/Simulink and ESP-r, i.e. multiple ESP-r(s), was developed and implemented in such a way to perform rapid simulations between building models and their remote control systems, even when both ESP-r and Matlab/Simulink are running on a heterogeneous network. This approach is based on the hierarchical decomposition concept shown in Fig. 8, which implements run-time coupling by taking into account different levels of abstraction, defining different operations on each level of run-time coupling, and proposing model refinements that will translate requirements and specifications to enable cycle-accurate implementation.

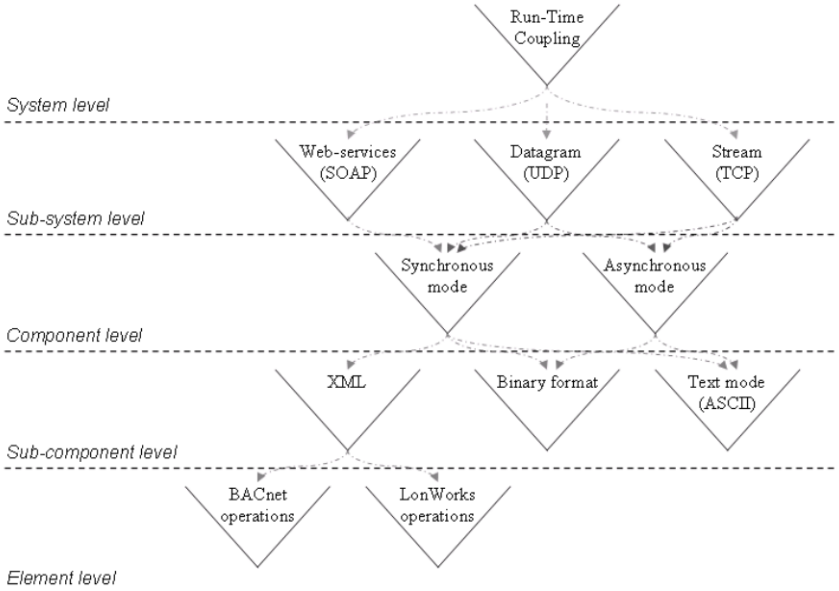

Figure 9. System-level design of run-time coupling between ESP-r and Matlab/Simulink

Fig. 9 illustrates the system-level design as a means of run-time coupling between Matlab/Simulink and ESP-r in which properties such as functionality, connectivity, and mode of exchange are represented on different levels of abstraction and each function is a part of the previous one. For this reason, the system-level design (or design concept) for run-time coupling is based on the SE concept that embeds the V life-cycle model (or process) at all levels of abstraction. The underlying objective of applying the SE concept is to maximize the value of simulation and ensure the translation of the initial (especially functional) requirements into operational functions in the design of run-time coupling and its integrated applications, such as interoperability. Moreover, the use of an SE concept as a design methodology for the development and implementation of run-time coupling between ESP-r and Matlab/Simulink provides a simple and flexible means of interfacing Matlab/Simulink with ESP-r over a heterogeneous network.

K. Extension of Run-Time Coupling to Represent BACS Technology in Simulation

Of the many possible ways to run-time couple more than one ESP-r with Matlab/Simulink at the same time, the Portable Operating System Interface (POSIX) standard for threads has been the most widely adopted [15]. The use of POSIX threads is very advantageous because of its standardization, flexibility, and portability, as well as fact that POSIX threads provide a standardized programming interface for the dynamic creation and destruction of threads (i.e. sub-threads). It also enables use of the same port and a single shared address space to make Matlab/Simulink accessible to all ESP-r(s) connections that are handled on the network. By using a single address space abstraction, it is possible to avoid the overhead inherent to data exchange and provide better support for concurrency, parallelism, and consistency of data exchange in run-time coupling between Matlab/Simulink and multiple ESP-r(s) with substantial ease.

To represent BACS architecture in simulation, the approach shown in Fig. 7 was extended to permit the option of run-time coupling with more than one ESP-r with Matlab/Simulink. This option was developed by using multi-threading in conjunction with C++ codes to support parallel and distributed control and building performance applications between multiple ESP-r(s) and Matlab/Simulink in the same simulation environment. Within this option, all ESP-r(s) should share the same address space of the Matlab/Simulink location and be able to run on either the same machine as Matlab/Simulink or a separate machine connected to a network. Each time a new ESP-r is connected with Matlab/Simulink, its specific thread is created by the *matespexge* toolbox in order to avoid conflicts and data inconsistencies with other concurrent ESP-r(s) participating in the same simulation environment. As all participating (or connected) ESP-r(s) exchange data with the same Matlab/Simulink, any ESP-r can access all the global variables exchanged by Matlab/Simulink through its specific sub-thread. Fig. 10 illustrates an example of how run-time coupling between Matlab/Simulink and multiple ESP-r(s) is implemented using POSIX threads.

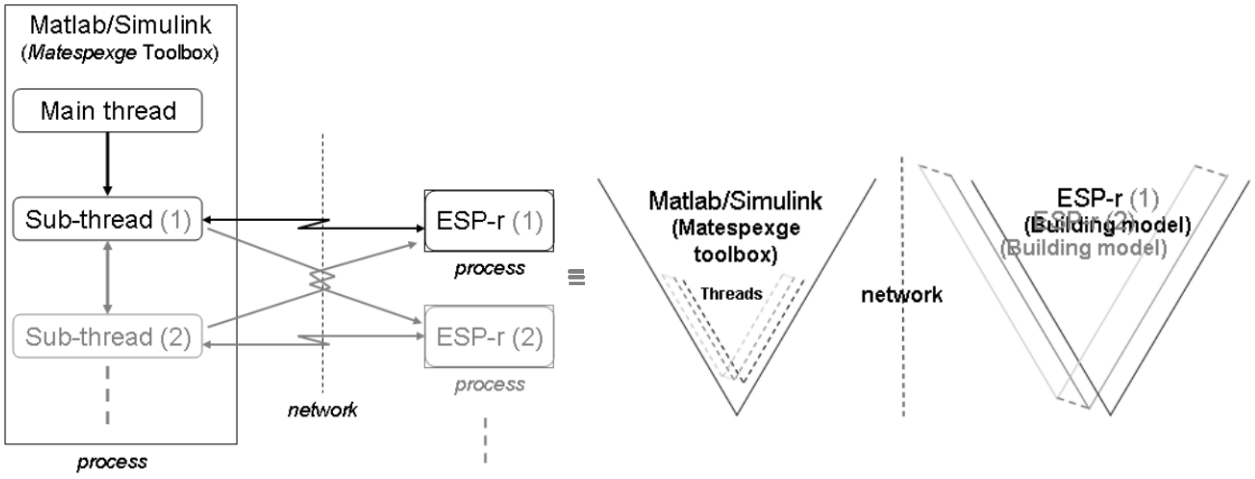

Figure 10. Conceptual view of how matespexge toolbox is multi-threaded with multiple ESP-r(s): representation in a conventional way (left) and its equivalence in the V lifecycle model (right)

As shown in Fig. 10, the *matespexge* toolbox is implemented in such a way that one or more ESP-r(s) can connect and interact with Matlab/Simulink concurrently. The number of ESP-r(s) to run-time couple to Matlab/Simulink depends on the application, varying from one (1) to nine (9) ESP-r(s) simultaneously. This implementation is fairly complex, requiring that the main thread of the *matespexge* toolbox accept incoming connections and create one ESP-r sub-thread for each ESP-r connection that is handled. These ESP-r sub-threads are a part of the *matespexge* toolbox used by shared data structures to communicate with their parallel (or with all) connected ESP-r(s). Because the *matespexge* toolbox can run-time couple with multiple ESP-r(s),

- each data exchange to/from ESP-r is handled by the corresponding ESP-r sub-thread on the *matespexge* toolbox side;
- each ESP-r sub-thread can send data to other connected ESP-r(s) by accessing the shared data structure that contains their references; and
- the sockets connecting the *matespexge* toolbox to each ESP-r can be retrieved through this shared data structure.

Consequently, any interaction between ESP-r(s) can occur via the *matespexge* toolbox, where it is handled by a particular ESP-r sub-thread. In addition, this toolbox is implemented with callback methods to allow remote control systems (i.e. control systems modelled on Matlab/Simulink) to be invoked as they receive data from their corresponding building models built on one or more ESP-r(s). Because building models built on multiple ESP-r(s) can interact with each other via the *matespexge* toolbox, their corresponding remote control systems can also interact with each other on the Matlab/Simulink side. The main objectives of using this approach are to represent the BACS architecture in simulation and enable unrelated remote control systems, particularly advanced control systems such as MASs, to communicate with each other when their corresponding building models are built on diverse ESP-r(s).

In effect, permitting control systems - particularly MASs - to communicate with each other while remotely regulating building zone, plant, and mass-flow models built on diverse ESP-r(s) connected to a network results in the development of advanced building control applications that had previously not been feasible, such as:

- the use of coordinated and interconnected control systems, especially MASs, to better operate and regulate building HVAC&R equipment and lighting components in ABs;
- the use of self-adapting control systems to react to climate changes, the addition or removal of equipment in a building, or building plant variations; and
- the use of self-upgrading control systems to meet occupant needs when damping effects or changes are critical factors in the functioning of the systems

IV. Building Control Application

In [37] has highlighted the importance of run-time coupling between Matlab/Simulink and ESP-r over standalone simulations in the integration of advanced control systems in building performance simulation for improving all quality aspects of building indoor environments, as well as in the simulation of building control applications requiring multivariable control systems. In order to

demonstrate the development and implementation of run-time coupling between Matlab/Simulink and ESP-r, a building model built on ESP-r was used in closed loop with an external Proportional Integral (PI) control modeled on Matlab/Simulink. Fig. 11 shows this building model that is actually the test office in the TNO (Netherlands Organisation for Applied Scientific Research) building located in Delft (the Netherlands) used to investigate the phenomena that influence the indoor climate of buildings. The constructions used in this building were all internally insulated cavity walls except for the wall with window, which was external. The window was single glazed and almost south faced (31^{o} E azimuth).

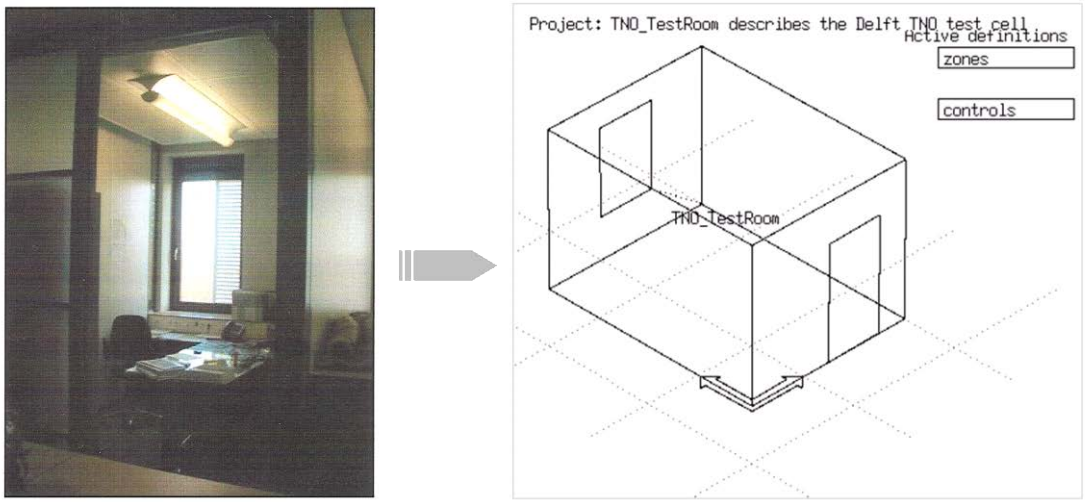

Figure 11. TNO test office facility concept (left) and its model built on ESP-r (right)

The external (or remote) PI control system was used simply to regulate the air temperature in a building zone by supplying the required heating flux capacity to it (maximum is 3000 W). Fig. 12 illustrates a building model built on ESP-r (left) in combination with a continuous PI control system modelled in Matlab/Simulink (right). The simulations were performed by run-time coupling between Matlab/Simulink and ESP-r in a synchronous mode and the data were exchanged between a building model and its external PI control system during the simulation via a network.

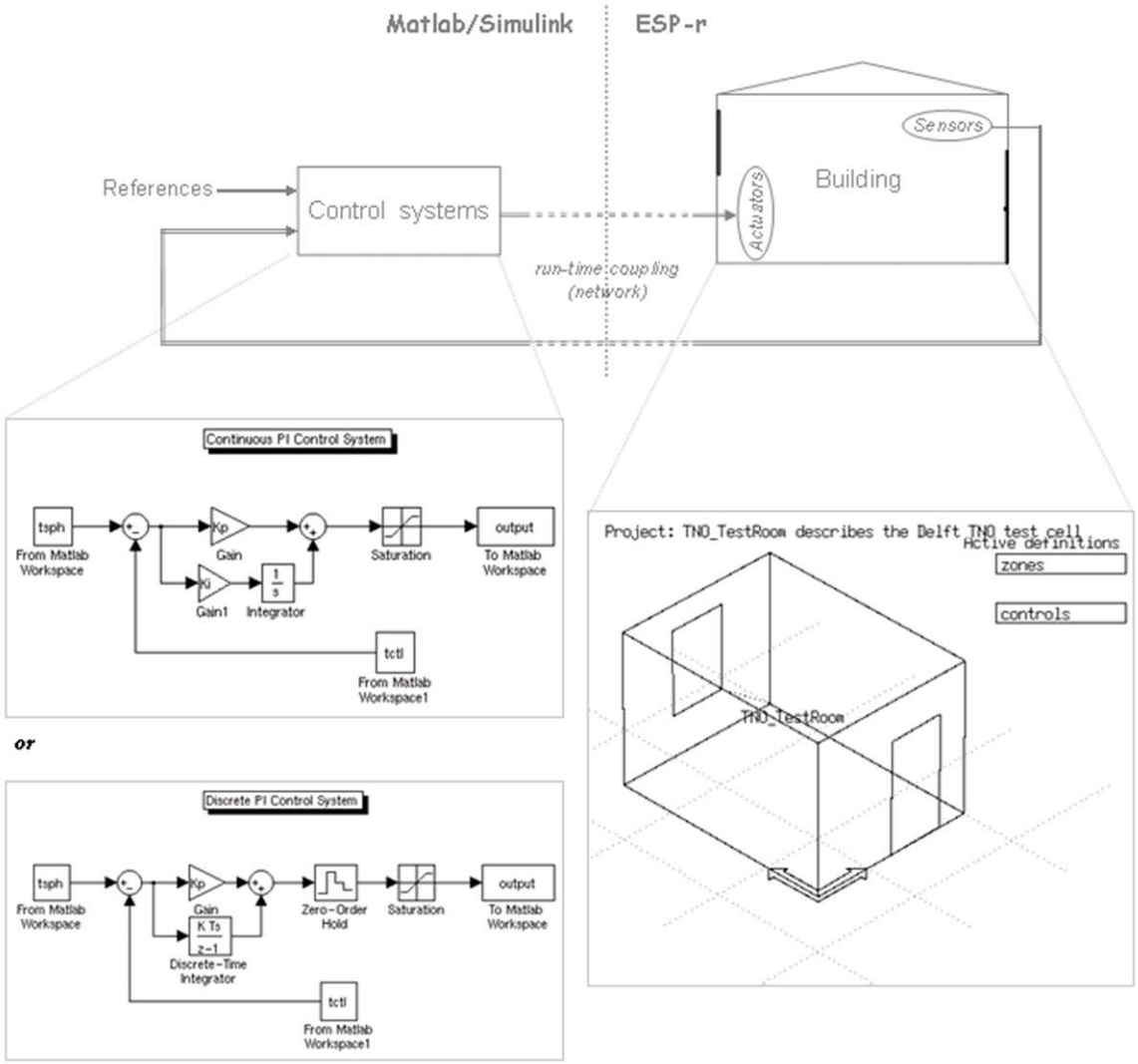

Figure 12. A simple building model built on ESP-r (right) with an external PI control system implemented on Matlab/Simulink (left)

In this application example, both continuous and discrete PI control system are set to maintain the indoor air-temperature at 22ºC between 07:00 and 18:00 o'clock for different numbers of simulation time-steps per hour. Consequently, the input to the PI control system implemented in Matlab/Simulink is the error signal created by subtracting the sensed air-temperature of the building model built on ESP-r from the air-temperature set-point. The output of this PI control system, a weighted sum of the error signal and its integral gain, is the actuated heating flux to the building model built on ESP-r. The weighted gains are the same used for both continuous-time (or analogue) PI control systems and discrete-time (or digital) PI control systems. Hence, the same values for proportional and integral gains are used both continuous-time and discrete-time PI control systems. Fig. 13 illustrates the simulation results obtained with the continuous-time PI control system.

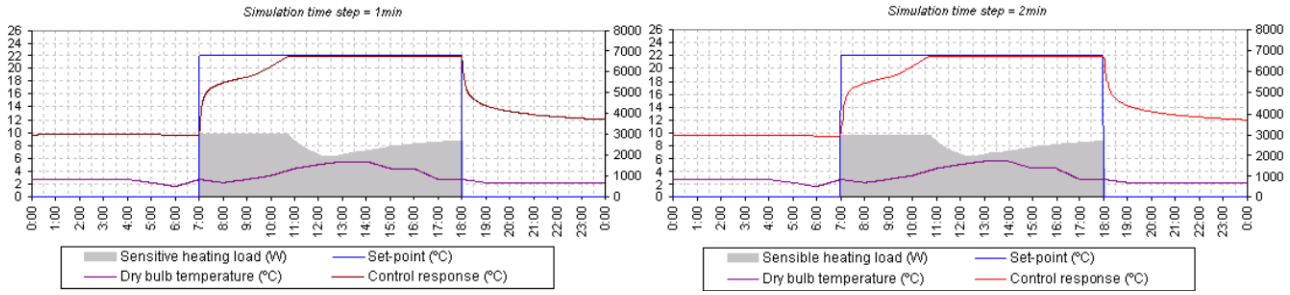

Figure 13. Simulation results obtained with the continuous-time PI control system

For the discrete-time PI control system, the sampling period T_s was 0.1s. Fig. 14 illustrates the simulated results obtained with the discrete-time PI control system.

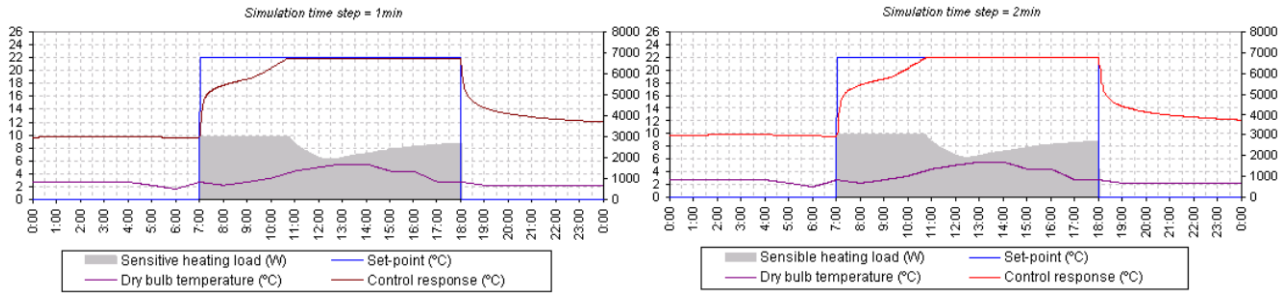

Figure 14. Simulation results obtained with the discrete-time PI control system

Comparison of the simulation results in Fig. 13 and 14 indicate that they are precisely identical despite being obtained by different types of a PI control system (i.e. continuous-time and discrete-time control systems). In addition, it appears that the simulation results obtained with the simulation time-step of 1 min. are similar to those obtained with the simulation time-step of 2 min. Further, once the control response (e.g. the air temperature in a building zone) reaches the set-point, the response becomes stable and is maintained continuously at that level until the end of the occupied period (i.e., between 07:00 and 18:00 o'clock). In addition, it can be noticed from Fig. 13 and 14 that the simulation result of the sensible heat load is optimized, as after reaching the set-point the control system supplies the heat flux into a building zone with the necessary energy.

V. CONCLUSION AND PERSPECTIVES

This paper has attempted to show how a SE methodology can help to develop and implement a distributed simulation mechanism for BACS technology by run-time coupling Matlab/Simulink and one or multiple ESP-r(s) that can be used to provide practical solutions for improving distributed control and building performance applications in ABs in the quest to satisfy occupants requirements while reducing energy use and greenhouse gas emissions. The objective of this approach is to facilitate the development of such complex systems by taking into account most of the design phases, ranging from the user and system requirements phase to the system operations and disposal phase, as previously shown in Fig. 3.

It must be stressed that the SE methodology provides tools that will allow reasonable requirements to be defined in the most effective manner. Because designing a dynamic distributed simulation mechanism for BACS is complex, the use of the SE methodology is needed to define all occupant requirements and required functionalities in the development, implementation, validation, and operation of the functioning processes early in the System Development Life-Cycle (SDLC), i.e., within the V diagram.

This work has shown that the computation speed can be significantly increased by running different applications on parallel computers, building control applications based on optimization and supervision, previously not feasible, can now be integrated by run-time coupling between Matlab/Simulink and ESP-r. The investigation of a simple application example has identified the efficiency of run-time coupling between Matlab/Simulink and ESP-r as essential for the performance of simulations due to the influence that computing capacity. Future work will envisage to analyze and simulate control building applications involving the utilization of multiple of ESP-r(s) by run-time coupling to Matlab/Simulink and to perform a more detailed analysis of the performance of a distributed simulation mechanism using different communication modes such as synchronous and asynchronous.

REFERENCES

[1] Ben-Nakhi, A.E., Mahmoud, M.A., Energy conservation in buildings through efficient A/C control using neural networks. Applied Energy; 73: 5–23, 2001.

[2] Blanchard, B. S., System Engineering Management, John Wiley & Sons, Inc., 1991.

[3] Calvino, F., Gennusca, M.L., Rizzo, G. and Scaccianoce, G., The control of indoor thermal comfort conditions: introducing a fuzzy adaptive controller. Energy and Buildings; 36: 97–102, 2004.

[4] Chen, K., Jiao, Y. and Lee, E.S., Fuzzy adaptive networks in thermal comfort. Applied Mathematics Letters; 19(5): 420–6, 2006.

[5] CSTB. Type 155—A new TRNSYS type for coupling TRNSYS and Matlab, Centre Scienfigique et Technique du Bâtiment's website available on http://software.cstb.fr/articles/18.ppt. Accessed December 2005.

[6] DSMC, Systems Engineering Management Guide, Proof Copy by Defense Systems Man-agement College, Fort Belvoir, VA, edition 1990.

[7] (2012) Energy Efficiency and Renewable Energy (EERE's website), available on http://www.eere.energy.gov/buildings/building_america/

[8] Einarsson, B., Mixed Language Programming, Part 4, Mixing ANSI-C with Fortran 77 or Fortran 90, Proceedings of International Workshop on Current Directions in Numerical Software and High Performance Computing, Kyoto, Japan, 1995.

[9] ESRU, The ESP-r System for Building Energy Simulation. User Guide Version 10 Series, ESRU Manual U02/1, University of Strathclyde, Scotland, 2002.

[10] González, P.A. and Zamarreño, J.M., Prediction of hourly energy consumption in buildings based on a feedback artificial neural network. Energy and Buildings; 37: 595-601, 2005.

[11] (2012) International Council on Systems Engineering (INCOSE's website), available on http://www.incose.org/

[12] ISO, Building Automation and Control Systems (BACS)—Part 2: Hardware, ISO Std. 16 484-2, 2004.

[13] ISO, Building Automation and Control Systems (BACS)—Part 5: Data Communication Protocol, ISO Std. 16 484-5, 2003.

[14] Hoang, N., Jenkins, M., Karangelen, N., Data Integration for Military Systems Engineering, Proceedings of IEEE Symposium & Workshop on Engineering of Computer Based Sys-tems, USA, 1996

[15] Hughes, C. and Hughes, T., Parallel and Distributed Programming using C++, Addison-Wesley, Boston, USA, 2003.

[16] Janak, M., Coupling building energy and lighting simulation. Proceedings of 5th International IBPSA Conference, Prague, CZ 2:307–12, 1997.

[17] Jelsma J., Kamphuis R. and Zeiler W., Learning about smart systems for comfort management and energy use in office buildings, Proceedings ECEEE Summer, 2003

[18] Kamphuis I.G., Warmer C.J., Zeiler W., Wortel W., Akkermans J.M. and Jelsma J., SMART: Experiences with e-services for Smart Buildings, ISPLC Conference, 27-29, Athens, 2002,

[19] Kamphuis I.G., Warmer C.J., Jong M.J.M. and Wortel W., IIGO: Intelligent Internet mediated control in the built environment: Description of a large-scale experiment in a utility building setting, ECN rapport, ECN-C—05-084, 2005.

[20] Kalogirou, S.A., Bojic, M., Artificial neural networks for the prediction of the energy consumption of a passive solar building; Energy; 25: 479-491, 2000.

[21] Kastner, W., Neugschwandtner, G., Soucek, S. and Newman, H. M., Communication Systems for Building Automation and Control; IEEE transactions, Vol. 93, n. 6, pp. 1178-1203, 2005.

[22] Kolokotsa, D., Niachou, K., Geros, V., Kalaitzakis, K., Stavrakakis. G.S. and Santamouris, M., Implementation of an integrated indoor environment and energy management system. Energy and Buildings; 37: 93–9, 2005.

[23] Kummert, M., Andre, P.and Nicolas, J., Optimal heating control in a passive solar commercial building. Solar Energy; 69(1–6): 103–16, 2001.

[24] Liang, J. and Du, R., Thermal comfort control based on neural network for HVAC application. Control applications 2005, CCA 2005, proceedings of IEEE conference; 819–824, 2005.

[25] Levermore, G.J., Building energy management systems: an application to heating and control. London: E & FN SPON, 1992.

[26] Loureiro, G., Leaney, P.G. and Hodgson M., A systems engineering environment for inte-grated automotive powertrain development, Society for Design and Process Science, Vol. 3, 4(41), USA , 1999

[27] Lute, P. and Van Passen, D., Optimal Indoor Temperature Control using a Predictor, IEEE Control System Journal, pp.0272-1708, 1995.

[28] Mathers, G. and Simpson, K. J., Framework for the Application of Systems Engineering in the Commercial Aircraft Domain, Report Version 1.2a, American Institute for Aeronautics and Astronautics, USA, 2000.

[29] Mathews, E.H., Arndt, D.C., Piani, C.B. and Heerden, E., Developing cost efficient control strategies to ensure optimal energy use and sufficient indoor comfort. Applied Energy; 66: 135–59, 2000.

[30] (2012) Matlab/Simulink Documentation (MathWorks's website), available on http://www.mathworks.com/

[31] Sahraoui, A.E.K., Buede, D., Sage, A., Systems engineering research a roadmap, Journal of Systems Science and Systems Engineering, Vol.17, N 3, pp.319-333, 2008.

[32] Sharples, S. Callaghan, V. and Clarke, G., A Multi-Agent Architecture for Intelligent Building Sensing and Control, Intern. Sensor Review Journal l , 1999.

[33] Shishko, R, NASA Systems Engineering Handbook, Proof Copy by National Aeronautics and Space Administration, USA

[34] Snoorian, D., 2003. Smart buildings, IEEE spectrum, Vol. 40, n. 8, pp 18-23, 1995.

[35] Yahiaoui, A., Hensen J.L.M. and Soethout L.L., Integration of control and building performance simulation software by run-time coupling, Proceedings of IBPSA Conference and Exhibition, Vol. 3, pp. 1435-1441, Netherlands, 2003.

[36] Yahiaoui, A., Hensen, J., and Soethout, L., Developing CORBA-based distributed control and building performance environments by run-time coupling, Proceedings of 10th ICCCBE, Germany, 2004.

[37] Yahiaoui, A., Hensen J.L.M., Soethout L.L. and Van Paassen, D., Interfacing of control and building performance simulation software with sockets, Proceedings of IBPSA Conference and Exhibition Montreal, Canada, 2005.

[38] Yahiaoui, A., Hensen, J. , Soethout, L., and Paassen, D., Model based Optimal Control for Integrated Building Systems, Proceedings of the 6th International Postgraduate Research Conference in the Built and Human Environment, Netherlands, 2006.

[39] Yahiaoui, A., Sahraoui, A. E. K., Hensen, J. and Brouwer, P., A Systems Engineering Environment for Integrated Building Design, UK Chapter of INCOSE proceedings, European Systems Engineering Conference, Edinburgh, Scotland, 2006.

[40] Yahiaoui, A., Hensen, J., Soethout, L., and Paassen, D., Simulation based design environment for multi-agent systems in buildings, Proceedings of 7th Inter. Conference on System Simulation in Buildings, Belgium, 2006.

[41] Yahiaoui, A, A Systems Engineering Approach to Embedded Control System Implementation in Buildings", Proceedings of 18th annual International Symposium of INCOSE, Netherlands, 2008.

[42] Wiese, P. and John, P., Engineering Design in the Multi-Discipline Era: A Systems Approach, John Wiley & Sons, Inc, 2002..

[43] Zhai, Z. Developing an integrated building design tool by coupling building energy simulation and computational fluid dynamics programs. PhD thesis, Massachusetts Institute of Technology, USA, 2003

[44] D'Andrea R. and Dullerud, G. E. "Distributed Control Design for Spatially Interconnected Systems", IEEE Transactions on Automatic Control, vol. 48, no. 9, pp 1478-1495, 2003.

[45] Walsh, G.C., Ye, H. and Bushnell, L.,.Stability Analysis of Networked Control Systems, Proceedings of the American Control Conference, pp. 2876-2880, Chicago, USA, 1999

ABBREVIATIONS

ABs: Automated Building
API: Application Programming Interface
BA: Building Automation
BAS: Building Automation System
BACS: Building Automation and Control System
BEMS: Building Energy Management System
BMS: Building Management System
DCS: Distributed Control System
EMS: Energy Management System
HVAC&R: Heating, Ventilation, Air-Conditioning, and Refrigeration
IB: Intelligent Building
IBMS: Intelligent Building Management System
IPC: Inter-Process Communication
MAS: Multi-Agent System
NCS: Networked Control System
OS: Operating System
SE: Systems Engineering
SB: Smart Building
SME: Society of Manufacturing Engineer
SoS: System of System
XML: Extensible Markup Language