A New Systematic Design Approach for Distributed Control and Building Performance Simulation

Azzedine Yahiaoui

Center for Building & Systems, Technische Universiteit Eindhoven (TU/e)

PO Box 513, 5600MB Eindhoven, the Netherlands

a.yahiaoui@bwk.tue.nl

***Abstract*-Demand for distributed simulations between control modeling and building performance applications is rapidly growing as the preferred means for supporting design studies of Automated Buildings (ABs) that are becoming increasingly complex because of urgent needs for developing sophisticated improvements in building indoor environments. In order to overcome problems in meeting the occupants' needs while reducing energy use and greenhouse gas emissions, it requires using distributed simulations to study the impact of advanced control strategies on building performance applications through virtual representations rather than using experiments, which are usually time-consuming and cost-prohibitive. For this reason, this paper mainly describes the development and implementation of a framework for distributed simulations involving different software tools over a network. The main role of this framework is analogous to a cooperative middleware that distributes one or more building performance simulation tool(s) and control modeling environment by run-time coupling over a network as qualified by similarity to Building Automation and Control Systems (BACS) architecture. The paper finishes by giving an outlook on the present work for further developments.**

Keywords- Systems Engineering; Distributed Dynamic Simulation; Building Performance Applications; Control Systems; Automated Building; Building Automation and Control Systems

I. INTRODUCTION

Today, a distributed simulation between control systems and building performance applications is increasingly becoming an important enabler in the analysis of Automated Buildings (ABs) for better design and operation. In this context, a distributed simulation consists of several run-time coupled software tools that communicate over a network to coordinate the control actions and tasks of building performance applications. However, to develop the necessary diversity of control functions for all the plant systems that operate within a building indoor environment, it is required to consider several practical aspects such as economic factors in order to fulfill the occupants' needs (i.e. by maintaining the building indoor processes, such as air-temperature, relative humidity, and light level, at the desired occupants references) while reducing energy consumption and greenhouse gas emissions. Hence, the use of experiments for testing and analyzing new control systems in buildings is an option, but they are time-consuming and cost-prohibitive. For example, when testing a designed control system or calibrating its internal parameters, it requires at least 24 hours to obtain the results. In contrast, when using simulations, it takes only few minutes or one hour at most. For this reason, a distributed simulation between control systems and building performance applications is increasingly becoming an essential enabler in the analysis of ABs for better design and operation.

ABs are a class of buildings, which are able to accrue economic and environmental benefits by the use of computer-based monitoring to coordinate, regulate and optimize building heating, ventilation, air-conditioning, and refrigeration (HVAC&R) equipment, lighting components, and facilities related to the maintenance of fire safety and elevator function, among other functions [20, 22]. 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 environmental performance. In principle, ABs are composed of numerous sensors, actuators, and control units 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 Building Automation and Control Systems (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 [13]. All these names refer to BACS, which greatly increase the interaction of plant 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 [4].

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. [17], 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 [3, 4]. 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 [15, 22]. 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., the occupants need).

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 is given in Section 4. The conclusion is presented in Section 5.

II. CHALLENGES TO DISTRIBUTED CONTROL AND BUILDING PERFORMANCE SIMULATION

Computer simulation of control and building performance and energy consumption is of particular importance to 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, and safety. 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 system 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 system modeling environment could enable integrated building performance assessment by predicting the overall effect of innovative control strategies in a building indoor environment [20, 23]. By extending this potential in distributing one or more building performance simulation software tools and control systems modeling environment over a network, this results in a typical pattern of distributed simulation between control systems and building performance applications as qualified by similarity to BACS architecture [23, 24].

Distributing different applications, especially simulation environments and tools on the network provides the facility 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 some ongoing works by others for the purpose of building performance simulation include for example coupling between lighting and building energy simulation [8], between computational fluid dynamics programs and building energy simulation [27], and between systems and building energy simulation [5]. However, these approaches were limited to a particular application, and often based on coupling only two software tools running on the same machine over the same operating system (OS). Besides these approaches, some libraries such as Building Controls Virtual Test Bed (BCVTB) are also used to couple different simulation tools for performance assessment of integrated building energy and controls systems [11], but still limited in terms of a detailed representation of BACS technology or real-time networked building control applications in simulation. For this reason, a framework distributing one or more building performance simulation tool(s) with control systems environment over a network by run-time coupling was developed and implemented based on SE approach with several different communication options for the objective of a more effective and reliable applicability.

III. DEVELOPMENT AND IMPLEMENTATION

A. Overview of Networked Building Control Systems

Fig. 1 shows an example of a networked control and building application that shows two separate different parts including control and building zone and plant systems, and a network that links each other. In general, this network is a protocol that supports data communication between control systems and building HVAC&R equipment and lighting components.

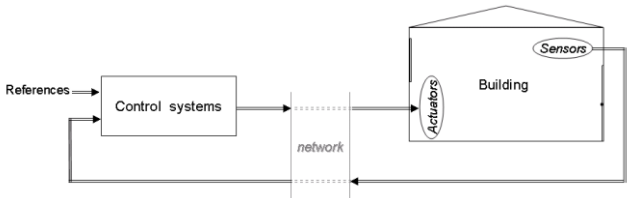

Fig. 1 Network-based building control application

B. Description of BACS Architecture

As outlined above, ABs are buildings that are controlled by BACS, and often referred as network-based building control systems. BACS is an example of a Distributed Control System (DCS) as 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 brings inevitable problems due to the use of the network. Fig. 2 illustrates a complete BACS architecture that can be described at four main levels [3, 4, 20]:

- 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.

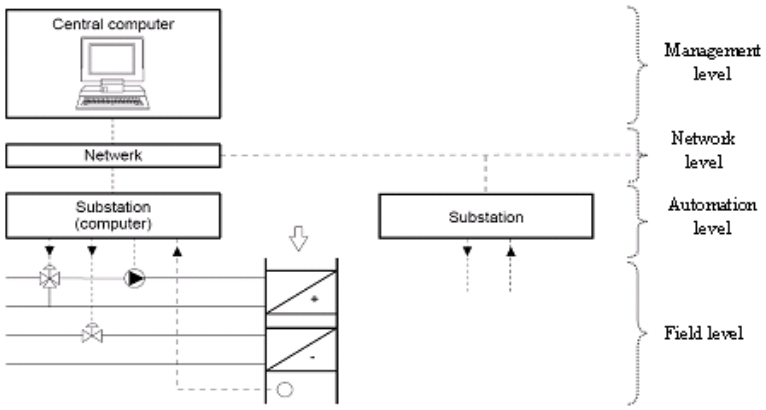

Fig. 2 BACS architecture

C. Systems Engineering Practice

SE is an emerging discipline that has been traditionally applied to complex technical development programs in which a software and/or hardware system was being developed and realized successfully [25, 9]. In literature, there are many definitions for SE. As a simple definition, SE is an interdisciplinary collaborative process (or methodology) used to ensure that a user need(s) or requirement(s) is satisfied throughout its entire life-cycle model such as waterfall, spiral, and V (or Vee) diagram. It concerns itself with effective phases suchKM as analysis of requirements and functions, synthesis, verification and validation of the design solution. Another way of defining SE, in accordance with e.g. [10], is that SE is a generic-solving process that provides mechanisms for defining, describing, and evolving both the product and the process needed to build the final product. This process should be applied throughout the entire system life-cycle model to all activities associated with the product development, verification/test, design, training, operation and use, support, distribution, and disposal. For the purposes of this study, the V diagram (or model) was used as a way of showing the SE process and relating the different phases in the system development life cycle of a distributed dynamic simulation mechanism to one another.

D. Basic Systems Engineering Concept

Effectively without a flexible, but a structured and rigorous approach to solving complex problems concerning 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, this requires a process that is adaptable to changing requirements, yet structured in a way that minimizes lost effort. The SE concept uses the followings [1, 9, 16, 22]:

1) Determine the requirements or needs that the solution should fulfill.

   a) Define end-user requirements (or occupant needs) that are considered as top-level global requirements.
   b) Perform functional analysis to divide top-level global requirements into low-level local requirements and determine an alternate means of achieving the top-level requirements.
   c) Define the interrelationship 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 was 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.

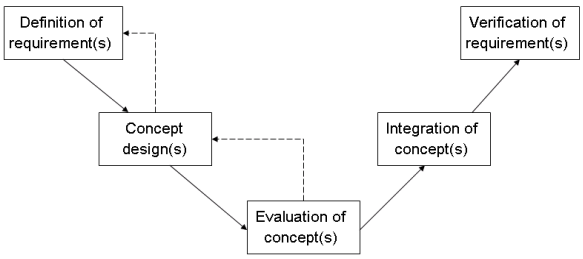

Fig. 3 A typical example of the SE approach

The above SE concept or a modified version of it is often used by organizations such as [2] for developing new products or solving day-to-day problems because it is natural to follow. In addition to this, it is sometimes lacking as a disciplined and systematic framework for quantifying and documenting the various steps, resulting in a less structured process that allows the results to be influenced by chance, limited or irrelevant knowledge and experience, intuition, or other factors [26].

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

Fig. 4 shows a structured approach to a conceptual design of distributed simulation between control systems and building performance simulation [23, 24].

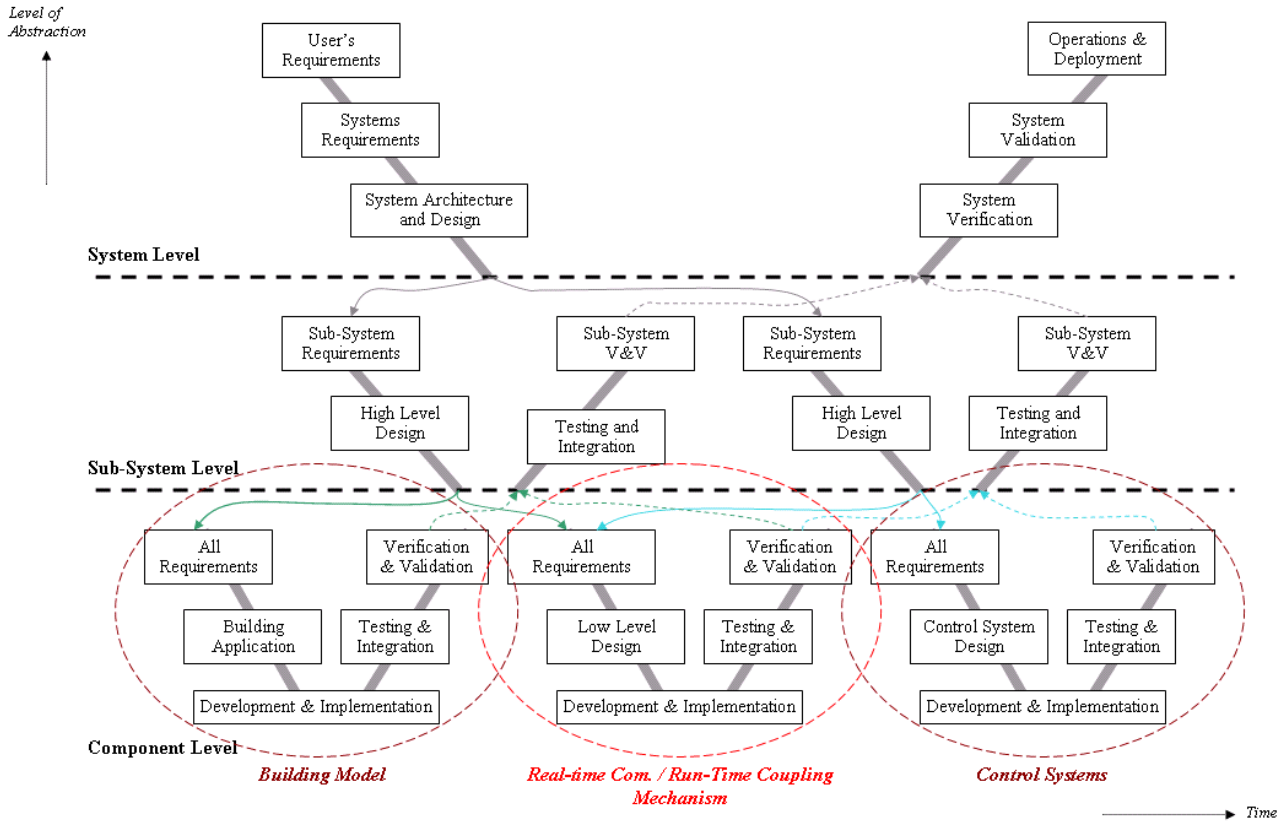

Fig. 4 A hierarchical approach to the systematic characterization of distributed control and building performance simulation

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. 4, 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 plus 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, that are of 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 to the application of control systems in buildings indoor environment, as shown in Fig. 1.

F. Development and Implementation of Run-Time Coupling

The framework for distributed control and building performance simulations must have sufficient capabilities to enhance the flexibility in integrating any control system modeled in Matlab/Simulink to any building model built in ESP-r. In particular, the set of requirements set forth in run-time coupling one or more ESP-r(s) with Matlab/Simulink must be taken into account by this framework. The most important requirements for successful run-time coupling ESP-r with Matlab/Simulink are [15, 16]:

- The ability to run the different software tools, ESP-r and Matlab/Simulink, simultaneously on a heterogeneous network including Unix-variant and Windows;
- The possibility to run-time couple with e.g. a test-rig for a control system (hardware testing in the loop), or even a building emulator, in which the Inter-Process Communication (IPC) must be platform independent;
- The ability that run-time coupling between one or more ESP-r(s) and Matlab/Simulink supports different communication modes including synchronous, asynchronous, partially asynchronous;
- The ability that run-time coupling between one or multiple ESP-r(s) and Matlab/Simulink supports data exchange over a network in either unidirectional or bidirectional; and
- The ability that run-time coupling between one or more ESP-r(s) and Matlab/Simulink supports different data exchange formats such as ASCII, binary and Extensible Markup Language (XML).

In [18, 19, 20], 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 remote control system in different communication modes including synchronous, asynchronous, and partially asynchronous, as shown in Fig. 1. The main advantage of using this IPC format lies in the fact that although the building model and its control systems are built separately and can be located on different kinds of machines, they work together by exchanging data in different formats including ASCII, binary and XML across a network. Fig. 5 illustrates the proposed approach to distributed simulations between ESP-r and Matlab/Simulink by run-time coupling over a network.

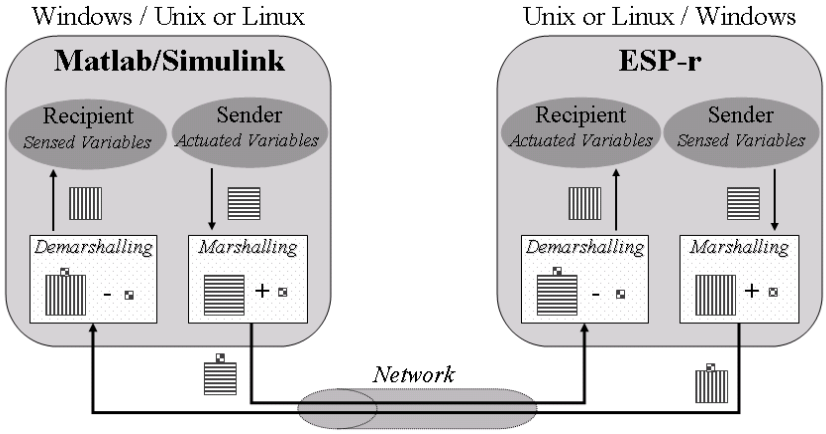

Fig. 5 Run-time coupling between Matlab/Simulink and ESP-r

In effect, run-time coupling is implemented with Internet sockets 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 over 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 the 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 that 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 minor user interferences. Therefore, it requires only modeling a building model on ESP-r and its control systems in 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 [7] 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 [6]. Therefore, mixed-language programming is used to develop and implement an approach combining a Fortran common block with global C/C++ external 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 by run-time coupling to 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 [14] 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 and 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 Simulink needs to be dealt with, 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 and 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 over it, 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.

G. 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 was developed in such a way to perform rapid simulations between building models and their 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, as shown in Fig. 6, 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.

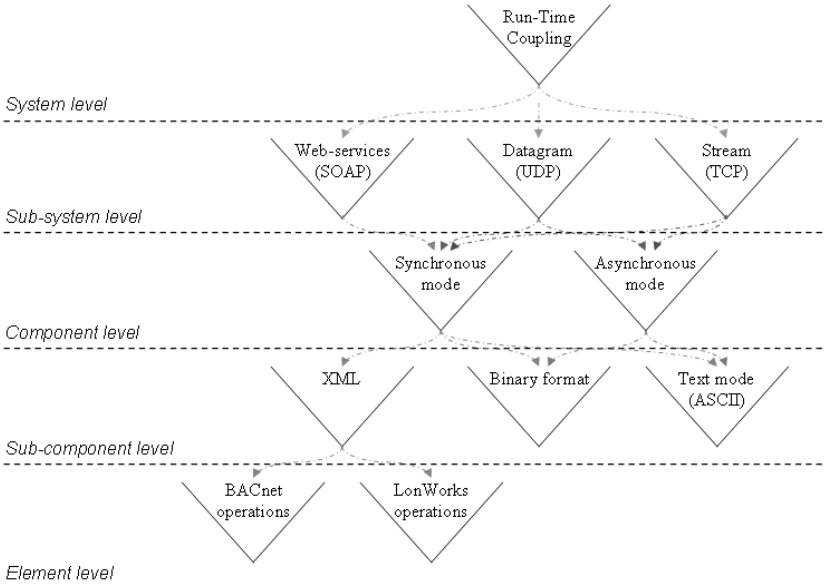

Fig. 6 System-level design of run-time coupling between ESP-r and Matlab/Simulink

Fig. 6 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.

H. 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 [12]. The use of POSIX threads is very advantageous because of its standardization, flexibility, and portability, as well as the 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. 5 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. 7 illustrates an example of how run-time coupling between Matlab/Simulink and multiple ESP-r(s) is implemented using POSIX threads.

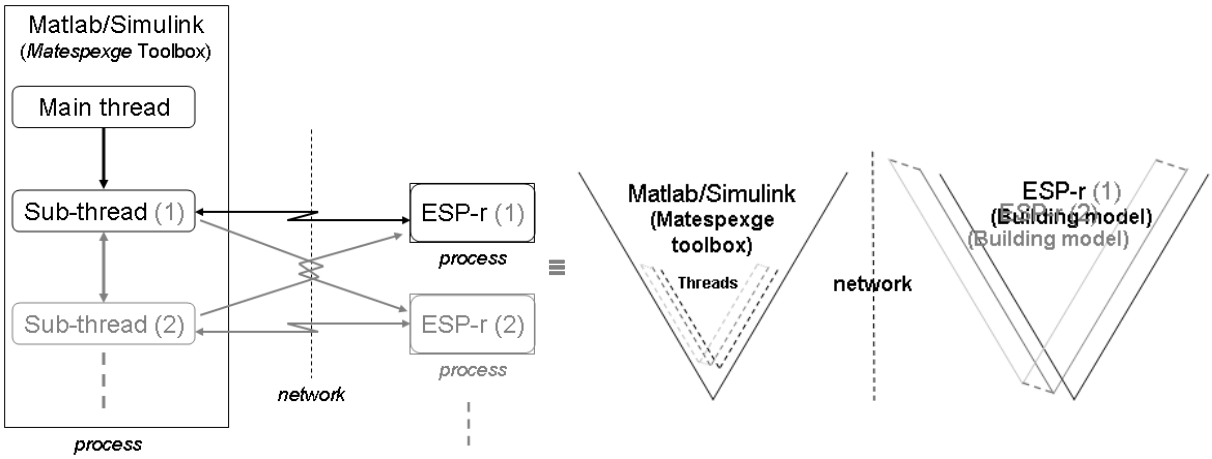

Fig. 7 Conceptual view of how *matespexge* toolbox is multi-threaded with multiple ESP-r(s): representation in a conventional way (left) and its equivalence in the V lifecycle model (right)

As shown in Fig. 7, 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 accepts incoming connections and creates 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, the *matespexge* toolbox is implemented with call-back methods to allow remote control systems (i.e. control systems modeled 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 representing the BACS architecture in simulation and enabling unrelated remote control systems, particularly advanced control systems such as Multi-Agent Systems (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 been infeasible, 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

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. 8 shows this building model that is actually the test office in the TNO building located in Delft (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 (31o E azimuth).

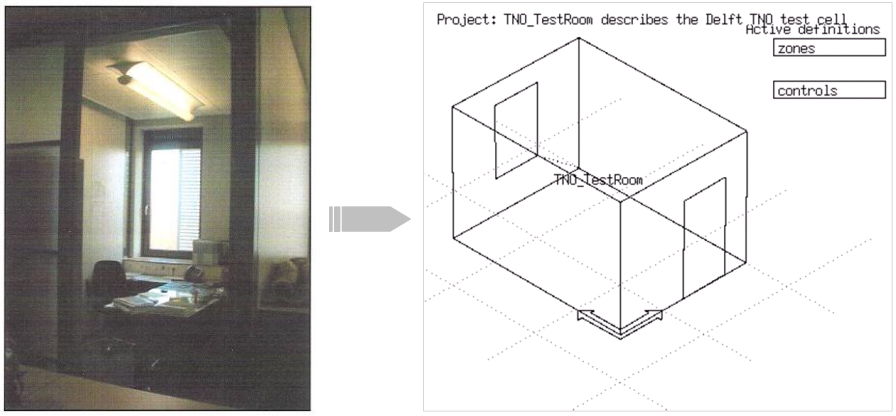

Fig. 8 TNO test office facility concept (left) and its model built on ESP-r (right)

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. 9 illustrates a building model built on ESP-r (left) in combination with a continuous PI control system modeled on 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.

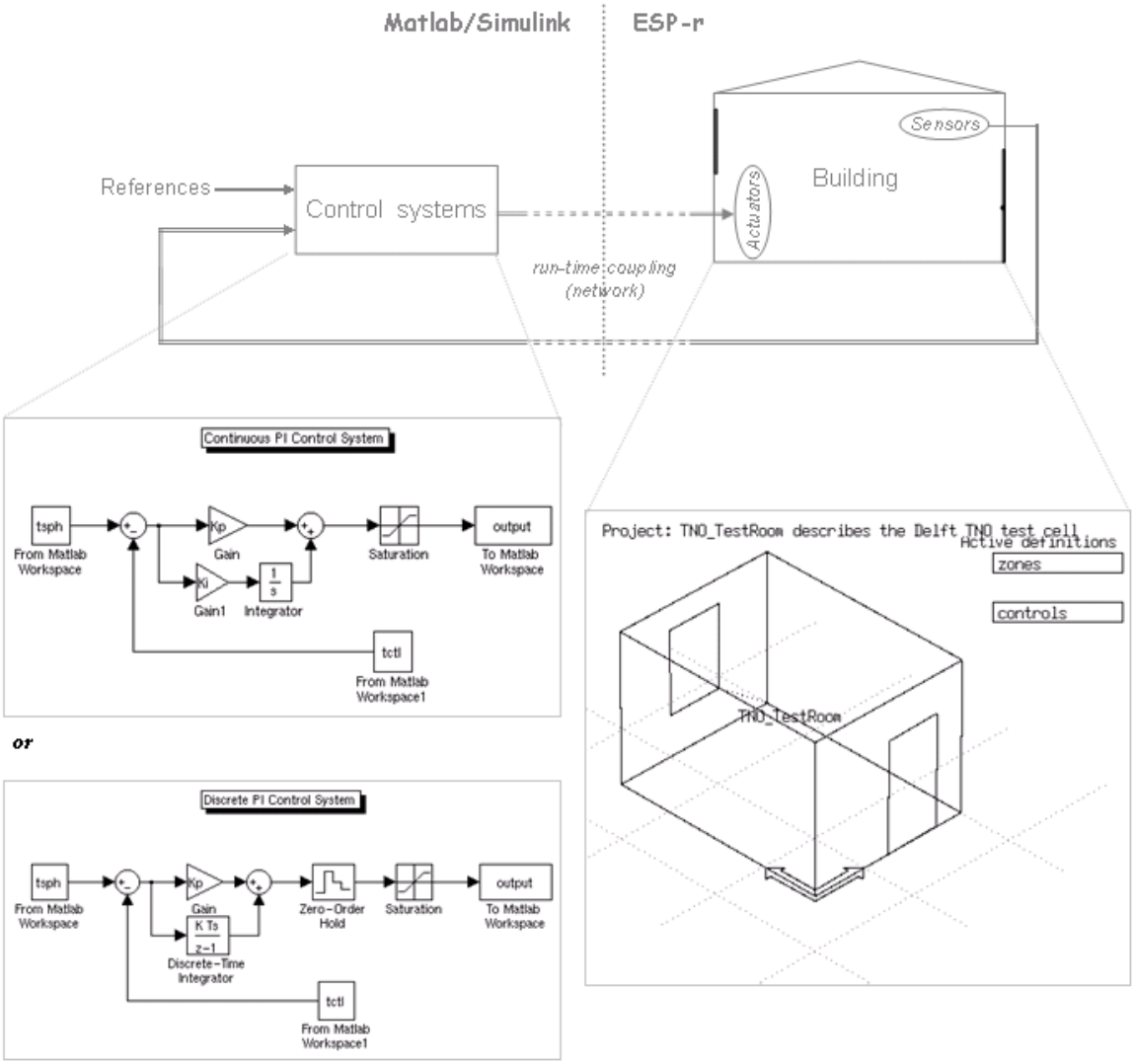

Fig. 9 A simple building model built on ESP-r (right) with an external PI control system implemented in Matlab/Simulink (left)

In this example of application, both continuous and discrete PI control systems were 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 was 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, was the actuated heat flux to the building model built on ESP-r. The weighting gains were 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 were used in both continuous-time and discrete-time PI control systems. Fig. 10 illustrates the simulation results obtained with the continuous-time PI control system.

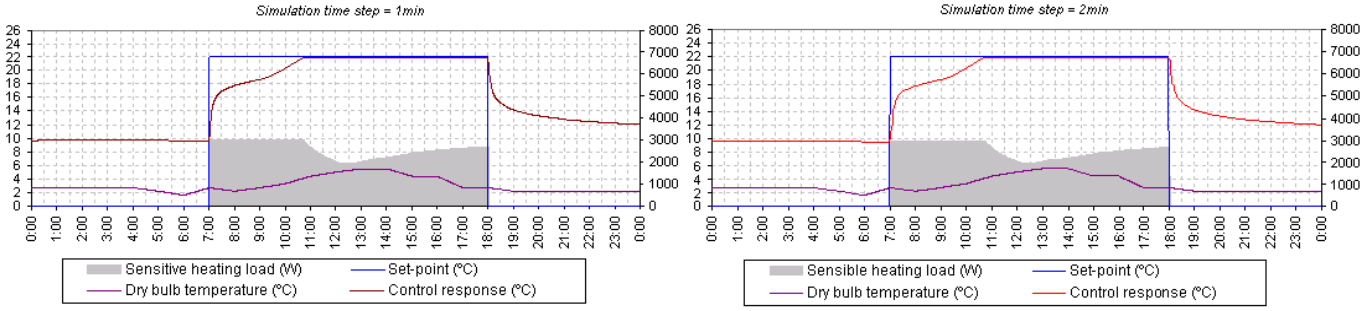

Fig. 10 Simulation results obtained with the continuous-time PI control system

For the discrete-time PI control system, the sampling period was 0.1s. Fig. 11 illustrates the simulated results obtained with the discrete-time PI control system.

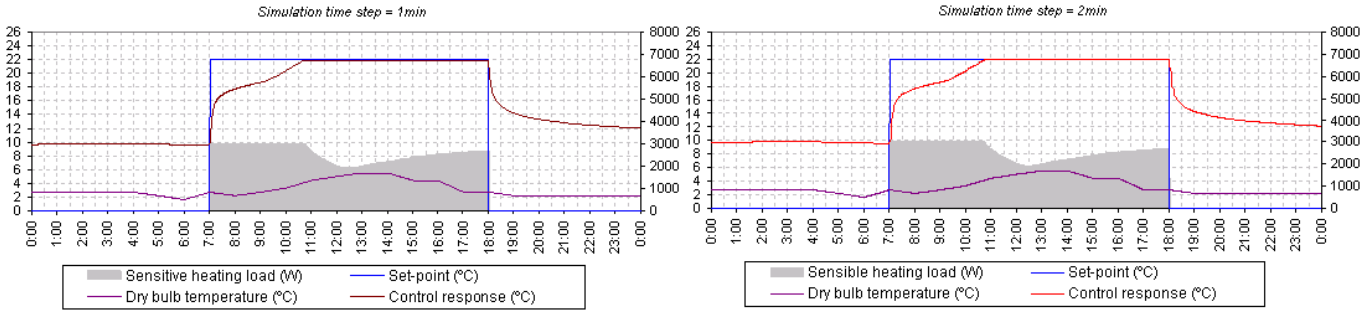

Fig. 11 Simulation results obtained with the discrete-time PI control system

Comparison of the simulation results in Fig. 10 and 11 indicate that they were precisely identical despite being obtained by different types of a PI control system (i.e. continuous and discrete). In addition, it appeared that the simulation results obtained with the simulation time-step of 1 min were similar to those obtained with the simulation time-step of 2 min, as well as that once the control response (e.g., the air temperature in a building zone) reached the set-point, the response became stable and was maintained continuously at the level of the set-point until the end of the occupied period (i.e. between 07:00 and 18:00 o'clock). In addition to this, it can be noticed from Fig. 10 and 11 that the simulation result of the sensible heat load was optimized as after reaching the set-point, the control supplied 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 the approach was 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 phases, as partially 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. Based on the perspective that designing a dynamic distributed simulation mechanism for BACS is a complex system, 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 lifecycle diagram. This work has shown that significant speedup as well as building control applications that were previously not possible can now be achieved with the utilization of distributed control and building performance simulation. The investigation of a simple example of application has identified the efficiency of run-time coupling between Matlab/Simulink and ESP-r as a significant means for the performance of distributed simulations. Among perspectives, future work 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 using different communication modes including synchronous and asynchronous and partially asynchronous.

REFERENCES

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

[2] (2009) Energy Efficiency & Renewable Energy (EERE website), available on http://www.eere.energy.gov/buildings/building_america/.

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

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

[5] CSTB – 2003, 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] Einarsson, B, "Mixed Language Programming, Part 4, Mixing ANSI-C with Fortran 77 or Fortran 90," Proc. of International Workshop on Current Directions in Numerical Software and High Performance Computing, Kyoto, Japan, 1995.

[7] ESRU, the ESP-r System for Building Energy Simulation. User Guide Ver. 10 Series, U02/1, University of Strathclyde, Scotland, 2002.

[8] Janak, M., "Coupling building energy and lighting simulation," Proc. of 5th Internat. IBPSA Conference, Prague, CZ 2:307-12, 1997.

[9] (2013) International Council on Systems Engineering (INCOSE's website), available on http://www.incose.org/.

[10] IEEE-1220, IEEE Standard for the Application and Management of the Systems Engineering Process, IEEE Inc., USA, 1996.

[11] Haves, P., Xu, P., "The building controls virtual test bed – a simulation environment for developing and testing control algorithms, strategies and systems," Proc. of 10th International Building Performance Simulation Association Conference, Beijing, 2007.

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

[13] Kastner, W., Neugschwandtner, G., Soucek, S. and Newman, H. M., "Communication Systems for Building Automation and Control," IEEE journal, vol. 93, no. 6, pp. 1178-1203, 2005.

[14] (2013) Matlab/Simulink Documentation (MathWorks's website), available on http://www.mathworks.com/.

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

[16] Shishko, R., "NASA Systems Engineering Handbook," Proof Copy by National Aeronautics and Space Administration, USA, edit. 1995.

[17] Snoorian, D., "Smart buildings," IEEE spectrum, vol. 40, no. 8, pp. 18-23, 2003.

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

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

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

[21] 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.

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

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

[24] Yahiaoui, A., and Sahraoui, A. E. K., "A Framework for Distributed Control and Building Performance Simulation," IEEE 21st International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE), France, 2012.

[25] Whalen, J., Wray, R., and McKinney, D., "Systems Engineering Handbook, "Version 2.0, INCOSE. Inc., 2000.

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

[27] 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.

ABBREVIATIONS

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