A system of systems view on integrating control for energy efficient building assessment

Azzedine Yahiaoui

Center for Building and Systems, Technische Universiteit Eindhoven (TU/e), P.O. Box 513, 5600MB Eindhoven, The Netherlands
Email: azzedine.y@outlook.com

Abstract: The ability to simulate combined control and building performance applications in a distributed manner is increasingly becoming an invaluable tool in the assessment of automated buildings (ABs) for better design and operation. To make such assessments without the need for costly and time consuming experiments, distributed simulations can be used so as to simultaneously fulfil the occupants’ needs while minimising energy use and potentially reducing greenhouse gas emissions. Therefore, distributed simulations are necessary to assess the impact of control systems on building performance applications (i.e., building indoor environments) over a network. This paper describes the development and implementation of a unified framework based on a system of systems (SoS) concept for distributed control and building performance simulations involving two or more different software tools via a network. The main objective of this framework is to couple one or multiple building performance simulation tool(s) with a control systems modelling environment by run-time over a network in order to similarly represent building automation and control systems (BACS) architecture in a simulation. Finally, the paper ends with some conclusions and perspectives for future work.

Keywords: systems engineering; distributed dynamic simulation; building performance applications; control systems; automated buildings; building automation and control systems; BACS.


Biographical notes: Azzedine Yahiaoui is a Researcher at TU/e in the Netherlands. He holds a BSc degree in Electrical, Electronics and Communications Engineering, MSc in Electronics Engineering specialised Control Systems, Master of Advanced Studies (MAS) degree in Automatic Systems, and PhD in Distributed Computing and Control Systems. He has extensive experience in modelling and simulation, development of distributed simulation systems, application of systems engineering concepts, and design of control systems for different engineering domains. His current research interests include system of systems engineering, hybrid dynamical systems, multi-agent control systems, verification and validation of complex systems, and buildings automation and control systems.
1 Introduction

Feasibility design studies of automated buildings (ABs) are becoming more complex as stringent needs of the occupants and the environmental quality require more effective control of building indoor processes such as heating, air-conditioning, ventilation, and lighting while at the same time minimising energy consumption and reducing greenhouse gas emissions. A distributed simulation between control systems and building performance applications can be the preferred means for supporting such design studies. Along with the development of the required diversity of control functions and algorithms, the fulfilment 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.

Because building heating, ventilation, air-conditioning and refrigeration (HVAC&R) systems 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. A systems engineering (SE) approach has been found effective in coordinating the multi-disciplinary applications to enable the realisation 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 and references). Other requirements for representing such applications in simulation must also be considered at early stage of development. The requirements, such as what frequency they operate over, what type of control to apply and what form of data exchange to use, are crucial to ensure that all parts of the application function properly. In fact, the application objectives are identified from stakeholders’ value requirements. The main stakeholders are the users, who determine the requirements and the associated priorities. With respect to these objectives, key performance indicators (KPIs) are then useful as a tool to investigate and manage changes in application data and deployment. With the rapidly occurring changes within buildings, the development of a distributed simulation mechanism for building automation and control systems (BACS) is then in part based on a modern approach, which is known as agility, to respond to the speed of operations and to changing requirements within an application. Therefore, this emphasis was partially on agility in SE as it focused on flexibility and speed in the process of distributed modelling and simulation. However, the application of an agile SE in this study is still limited, and this has promoted an important consideration in improving the design of a distributed simulation mechanism.

The remainder of this paper is organised as follows: background, challenges to distributed control and building performance simulation are introduced in Section 3, followed by development and implementation in Section 4, and then example of application in Section 5. The conclusion and perspectives for future work are presented in Section 6.
2 Background

ABs are a class of buildings, which are able to accrue economic and environmental benefits by the utilisation of computer-based monitoring to coordinate, organise, manage, adapt and optimise building HVAC&R equipment and lighting components, and also facilities related to the maintenance of fire safety and elevator function, among other functions (Yahiaoui et al., 2005, 2006b). With 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 a wide number of sensors, actuators, and control units interconnected in such a way to facilitate and adapt a suitable control strategy and/or an optimum control reference (or set-point) from the central computer-based monitor system. These basic activities of ABs have been the subject of BACS technology since the last century. Traditional and modern comprehensive BACS use in general 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 (Kamphuis et al., 2005). 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 (ISO, 2003).

Another dimension of BACS architecture is the application of a standard or open 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., (Mathews et al., 2000), 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 (ISO, 2003; 2004). 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 behaviour (Sharples et al., 1999; Yahiaoui et al., 2006b).

According to D’Andrea and Dullerud (2003) and Walsh et al. (1999), distributed control systems (DCSs) used to implement these architectures are usually referred to as networked control systems (NCSs). 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 fault diagnoses and maintenance procedures, control and detection in distribution, easy to expand, 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 synchronisation 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 analysing 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 HVAC&R equipment and lighting components. For example, Ben-Nakhi and Mahmoud (2001), González and Zamarreño (2005), Kalogirou and Bojic (2000), Kolokotsa et al. (2005), Levermore (1992), Mathews et al. (2000), and Shaikh et al. (2014) presented works related to building energy management strategies through cost reduction and energy consumption savings, while Calvino et al. (2004), Chen et al. (2006), Gruber et al. (2014), Kummert et al. (2001), Liang and Du (2005), Lute and van Paassen (1995), and Yahiaoui et al. (2006a) presented works with a focus on fuzzy, neural network, optimal and predictive control systems of thermal conditions in buildings. In addition, Jelsma et al. (2003), Kamphuis et al. (2002, 2005), Sharples et al. (1999) and Yang and Wang (2013) introduced the use of multi-agent systems (MASs) for comfort and energy management 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 modelling 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.

3 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 modelling 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 artefact’, 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 modelling 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 modelling and simulation capabilities. Marrying the two approaches by run-time coupling building performance simulation software and control systems modelling environment could enable integrated building performance assessment by predicting the overall effect of innovative control strategies in a building indoor environment (Yahiaoui et al., 2005; Yahiaoui, 2014). By extending this potential in distributing one or more building performance simulation software tools and control systems modelling environment over a network, would result in a typical pattern of distributed simulation between control systems and building performance applications as qualified by similarity to BACS architecture (Yahiaoui, 2014).

Distributing different applications (simulation tools) on the same environment provides the facility the ability to exchange data and events in a distributed and cooperative 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 (Janak, 1997), between computational fluid dynamics programs and building energy simulation (Zhai, 2003), and between systems and building energy simulation (CSTB, 2005). However, these approaches are limited to a particular application, and are often based simply on the coupling of two simulation tools running on the same machine. Other recent approaches based on co-simulate tools or shared libraries like building controls virtual test bed (BCVTB) are also used to couple different simulation programs for co-simulation. However, these approaches are also limited for the representation of distributed control applications using asynchronous or partially synchronised communication mode 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 was here developed with the main purpose to ensure that this will have a more general and wider applicability.

4 Development and implementation

4.1 Overview of a distributed building control system

Figure 1 shows an illustrated example of a distributed building control system, where control units (or systems) and building sensors and actuators are integrated over a communication network and then form a real-world DCS.
A system of systems view on integrating control

Figure 1  A distributed building control system

4.2 Description of BACS architecture

ABs are buildings that are controlled by BACS, and often referred as distributed 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. Figure 2 shows a complete BACS architecture that can be described at four main levels (ISO, 2004; Hoang et al., 1996; Yahiaoui et al., 2005):

- the **management level** consists of a central computer used for managing, storing, and analysing 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 HVAC&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.

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 real-world implementation of BACS is through a modelling and simulation approach. Therefore, it is required to have a simulation tool that supports analysis and evaluation of any building control application over a network, and enables integration of different control strategies and algorithms in building performance simulation. Consequently, a distributed dynamic simulation mechanism was developed and implemented mainly for BACS to simultaneously simulate and analyse building control applications in the presence of communication network dynamics.
SE practice is not new, but the discipline is. It started with large-scale programs in USA, principally in aeronautics (Mathers and Simpson, 2000), space (Wiese and John, 2002), and particularly in defence (DSMC, 1990). 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 (Loureiro et al., 1999) and recently adopted by the society of manufacturing engineers (SME), see e.g., (Sahraoui et al., 2008). 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 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

- information for management decision making.

Although many SE standards exist, the ANSI/EIA-632 standard is used in this study because it provides an integrated set of fundamental processes that aid in engineering or reengineering a system. This is also a high-level standard, which is applied to the engineering of any kind of system. For this reason, the EIA-632 standard is here used to guide the development of a distributed simulation mechanism for BACS.
4.4 Basic SE 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 minimises lost effort. The SE concept uses the following steps (Blanchard, 1991; Shishko, 1995; International Council on Systems Engineering, 2016; Yahiaoui, 2013):

1. determine the requirements or needs that the solution should fulfil:
   a. stakeholder analysis
   b. define top-level global end-user requirements (or occupant needs)
   c. perform functional analysis to divide top-level global requirements into low-level local requirements and determine alternate means of achieving the top-level requirements
   d. 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 optimise to eliminate and minimise 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.

Figure 3 A simplified V-model of the SE approach

Figure 3 shows a simplified example of the V (or Vee) diagram where steps 1 and 3 are interactive. In this paper, the V diagram (or model) is used with the SE approach as a tool
for the application of SE concepts to manage the development and implementation of a distributed dynamic simulation mechanism for BACS architecture.

The need for applying a V-model of the SE approach to the development of a distributed simulation mechanism stems from the challenges of transforming defined requirements into an operational, life-cycle-optimised, mechanism, and representing different simulation tools as a system of systems (SoS) integration in the same simulation environment. Within this framework, a comprehensive and structured methodology is used to develop a distributed simulation mechanism across a heterogeneous network with a capacity of using different network protocols and data exchange formats. Therefore, the use of the V-model approach to design of a distributed simulation mechanism will help save time and cost in the event of any possible changes to the developed mechanism, and test specifications.

4.5 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 Figure 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.
4.6 Context and application

This study will mainly focus on the development and design of advanced control systems in buildings, to analyse and enhance building environmental performance. The integration of building science engineering, architecture, construction management and risk assessment is required for new construction projects and existing buildings as well. Building control and performance applications require then a lifecycle of development as shown in Figure 5.

![Figure 5](image)

This performance simulation supports the systems development (including subsystems and components design and deployment operation), deployment operations from the development and implementation of the initial systems as a final system (or a prototype) for distributed control and building performance simulations. It is also useful for analysing performance problems and sustainability of the building.

4.7 SE structured approach to developing and implementing distributed control and building performance simulation

Figure 6 shows a structured approach to a conceptual design of distributed simulation between control systems and building performance simulation (Yahiaoui, 2014).

The main applications of BACS architecture are characterised from the functional viewpoint at different levels of abstraction and organised hierarchically into three levels to allow for easy understanding of the nature of their differences. Following the 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 Figure 6, the single V 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 environmental processes according to certain references set by the occupants. 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 DCS as it refers the application of control systems in building indoor environment, as shown in Figure 1.

4.8 Development and implementation of run-time coupling

The design of run-time coupling between Matlab/Simulink and one or more ESP-r(s) begins with the definition requirements, and proceeds eventually to conceptual design of the run-time coupling by means of trade-off analysis, and then to detailed design of every part of the run-time coupling being developed.
4.8.1 Design requirements

The main objective of run-time coupling of Matlab/Simulink with one or more ESP-r(s) is to support distributed control and building performance simulations with sufficient capabilities in order to enhance principally the flexibility and scalability in integrating any control system modelled in Matlab/Simulink within any building model built in ESP-r. For the achievement of this objective, a set of requirements were first identified and set forth as the basis for the development and design of the run-time coupling. However, these requirements must then be taken into consideration at the early-stage of development. Among the most important of these requirements are (Hughes and Hughes, 2003; Janak, 1997):

- the ability for run-time coupling between Matlab/Simulink and one or more ESP-r(s) to run on a heterogeneous network as on Windows and Unix Operating Systems (OS)
- the ability for run-time coupling between Matlab/Simulink and one or more ESP-r(s) to support data exchange over a network in either unidirectional or bidirectional
- the ability for run-time coupling between Matlab/Simulink and one or more ESP-r(s) to support different data exchange formats including ASCII, binary and Extensible Markup Language (XML)
- the ability for run-time coupling between Matlab/Simulink and one or more ESP-r(s) to support different communication modes including synchronous, asynchronous, partially synchronous (or asynchronous)
- update cycle (batch, event driven, threshold driven, etc.)
- the possibility for run-time coupling between Matlab/Simulink and ESP-r to enable simulations with either a real building (e.g., building emulator) or a control test-rig (e.g., hardware in the loop testing), in which the inter-process communication (IPC) must then be platform independent.

4.8.2 Trade-off analysis

After evaluating and selecting the most suitable solution among a number of possible options, using network (or internet) sockets has been chosen and approved by Yahiaoui et al. (2003, 2004, 2005) as the best means of implementing run-time coupling between Matlab/Simulink and one or more ESP-r(s) because they meet all the requirements of the run-time coupling, including those described above, and they can also be used to represent in simulations real-time building control implementations over a network, as shown in Figure 1.

4.8.3 Conceptual design

Network sockets are an IPC mechanism that is used for run-time coupling between ESP-r and Matlab/Simulink to support modelling of a building model and its external control systems separately. Both the building model and its control systems can be located on different machines running different OS such as Unix and MS-Windows. They also work together by exchanging data in a common format including ASCII, binary and extensible
markup language (XML) through a network, and by supporting communication modes such as synchronous, asynchronous, and partially synchronous. Figure 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

The initial advantage of developed run-time coupling with network sockets is the capability of integrating and extending entities 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 main advantages of this developed run-time coupling are that it permits any simulation of a building model and its control systems to be built separately using ESP-r and Matlab/Simulink, respectively, and that it provides the preferred means to handle interoperability tasks, especially cross interdisciplinary data integration and exchange between ESP-r and Matlab/Simulink with no or minor user interferences. Therefore, it requires only modelling 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.

### 4.8.4 Interfacing client socket to ESP-r

Because ESP-r, e.g., (ESRU, 2002)), 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 (Einarsson, 1995). 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 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.

4.8.5 Interfacing server socket to Matlab/Simulink

In Matlab/Simulink Documentation (2016) has a built-in utility called Matlab EXcutable (MEX) that is often used to convert C or 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++ code:

- 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
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 `matespexe` 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 realised by just 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 the analysis of multi-variable control systems of building performance applications that had previously not been feasible.

4.9 A practical approach to representing BACS technology in simulation

The BACS technology, as shown in Figure 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 assessment, 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 analyse both the performance and the stability of controlled building HVAC&R equipment and lighting components in ABs. The necessity for distributed simulations originates from the fact that BACS is a technology 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 manner
similar to BACS architecture, the IPC mechanism used to run-time couple them is then
designed to support cooperative applications through interoperable middleware that
provides a layer facilitating the interface of any building model with its control system.
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 Figure 8.

Figure 8  A practical approach to run-time coupling 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 Figure 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 nine 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.

4.10 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 Figure 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

![Diagram](image)

Figure 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 objective of applying the SE concept is to maximise 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.

**4.11 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 (Hughes and Hughes, 2003). The
use of POSIX threads is very advantageous because of its standardisation, flexibility, and portability, as well as the fact that POSIX threads provide a standardised 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 Figure 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 separated machines connected to a network. Each time a new ESP-r is connected to Matlab/Simulink, its specific thread is created by the matespexge toolbox so as 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. Figure 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 multi-threading matespexge toolbox with multiple ESP-r(s), (a) representation in a conventional way, (b) equivalence in the V lifecycle model.

As shown in Figure 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 to nine 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
• 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 call-back 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
• 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.

5 Building control application

Yahiaoui et al. (2005) highlights 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 modelled on Matlab/Simulink. Figure 11 illustrates 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 construction methods 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°E azimuth).

Figure 11  (a) TNO test office facility concept (b) model built on ESP-r (see online version for colours)

Figure 12  (a) A simple building model built on two instance of ESP-r with (b) an external PI control system implemented on Matlab/Simulink
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 3,000 W). Figure 12 shows the same building model built on two separate instances of ESP-r (left) in closed-loop with its PI control system (either continuous or discrete) modelled in Matlab/Simulink (right). The simulations were performed by run-time coupling between Matlab/Simulink and two instances of ESP-r in a synchronous mode and the data were exchanged between the building model and its external PI control during the simulation via a network.

In this application, both the continuous and discrete PI control system are set to maintain the indoor air-temperature at 22°C 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 (or reference). The output of this PI control system, a weighted sum of the error signal and its integral gain, is the actuated heating flux capacity supplied to the building model built on ESP-r. The weighted gains were the same used for both PI control systems, either continuous-time (or analogue) or discrete-time (or digital). Hence, the same values for proportional and integral gains were used both continuous-time and discrete-time PI control systems. Figure 13 illustrates the simulation results obtained with the continuous-time PI control system.

By comparing the simulation results obtained by ESP-r1 with those obtained by ESP-r2 at the simulation time step of 1 and 2 min, it can easily be observed that these obtained results are similar and comparable to each other; even though they were obtained in different simulation time steps. For the discrete-time PI control system, the sampling period $T_s$ was 0.1s. Figure 14 illustrates the simulated results obtained with the discrete-time PI control system.
As mentioned above, it can also easily be observed that by comparing the simulation results obtained by ESP-r1 with those obtained by ESP-r2 at the simulation time step of 1 and 2 min, these results are also similar and comparable to each other; even though they were obtained in two instances of ESP-r while using different simulation time steps. The simulation results were obtained in separate instances of ESP-r by run-time coupling to Matlab/Simulink in synchronised communication mode, i.e., when both ESP-r and Matlab/Simulink waits for each other in computation while exchanging data over a network.

Comparison of the simulation results obtained in Figure 13 and 14 indicate that they are precisely identical despite being obtained by means of using 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 in two instances of ESP-r. Further, once the control response (e.g., the air temperature in a building zone) reaches the set-point, the control response becomes stable and is maintained continuously at that level until the end of the occupied period, which is valid between 07:00 and 18:00 o’clock. In addition, it can be noticed from Figure 13 and 14 that the simulation result of the sensible heat load is optimised, as after reaching the set-point the control system supplies the heat flux into a building zone with the necessary capacity (or energy).

6 Conclusions

This paper showed how SE best practices can help 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 mainly 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 Figure 3. The SE methodology provides tools that 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 all required functionalities (i.e., functional requirements) 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 optimisation and supervision, previously not feasible, can now be integrated by run-time coupling between Matlab/Simulink and one or more ESP-r(s). 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 analyse and simulate control building applications involving the utilisation of multiple of ESP-r(s) by run-time coupling to Matlab/Simulink in different communication modes such as synchronous and asynchronous and to perform a more detailed analysis of the performance of distributed simulations over these different communication modes.

Acknowledgements

The authors would like to thank Abdelkader Sahraoui, Prof. at LAAS-CNRS of Toulouse for his significant input to the application of SE best practices to the development and design of a middleware for distributed control and building performance simulations, and Jan Hensen, Prof. at Eindhoven University of Technology (TU/e) for his critical help in this research, as well as the anonymous reviewers for their valuable feedback to the manuscript.

References

A system of systems view on integrating control


Hughes, C. and Hughes, T. (2003) Parallel and Distributed Programming using C++, Addison-Wesley, Boston, USA.


### Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABs</td>
<td>automated building</td>
</tr>
<tr>
<td>API</td>
<td>application programming interface</td>
</tr>
<tr>
<td>BA</td>
<td>building automation</td>
</tr>
<tr>
<td>BAS</td>
<td>building automation system</td>
</tr>
<tr>
<td>BACS</td>
<td>building automation and control system</td>
</tr>
<tr>
<td>BEMS</td>
<td>building energy management system</td>
</tr>
<tr>
<td>BMS</td>
<td>building management system</td>
</tr>
<tr>
<td>DCS</td>
<td>distributed control system</td>
</tr>
<tr>
<td>EMS</td>
<td>energy management system</td>
</tr>
<tr>
<td>HVAC&amp;R</td>
<td>heating, ventilation, air-conditioning, and refrigeration</td>
</tr>
<tr>
<td>IB</td>
<td>intelligent building</td>
</tr>
<tr>
<td>IBMS</td>
<td>intelligent building management system</td>
</tr>
<tr>
<td>IPC</td>
<td>inter-process communication</td>
</tr>
<tr>
<td>MAS</td>
<td>multi-agent system</td>
</tr>
<tr>
<td>NCS</td>
<td>networked control system</td>
</tr>
<tr>
<td>OS</td>
<td>operating system</td>
</tr>
<tr>
<td>SE</td>
<td>systems engineering</td>
</tr>
<tr>
<td>SB</td>
<td>smart building</td>
</tr>
<tr>
<td>SME</td>
<td>society of manufacturing engineer</td>
</tr>
<tr>
<td>SoS</td>
<td>system of system</td>
</tr>
<tr>
<td>XML</td>
<td>extensible markup language</td>
</tr>
</tbody>
</table>