AiCARR Journal

2016, 38, 60-65bis

Research paper

BUILDING PERFORMANCE SIMULATION PROGRAMS: BETWEEN "OPERABILITY" AND "ADEQUACY"

Livio Mazzarella, Martina Pasini
Dipartimento di Energia, Politecnico di Milano

ABSTRACT

Energy efficiency in Buildings, combined with an efficient use of the energy provided by renewable sources, are essential objectives set by the revision of the European Energy Performance of Buildings Directive. To achieve these objectives, an accurate estimate of the behavior of the system to be built/improved must be available during all stages of the design process or energy audit (if existing). While designing or improving energy efficiency, other important and associated goals must be addressed, such as environmental health (hygrothermal, acoustic and luminous), costs, environmental sustainability, etc. Having to choose a dynamic simulation program to inform the design process it is necessary to analyze the possibilities offered by different available software, in terms of accuracy and completeness, while taking into account ease of use and included facilities aimed at supporting the design process itself. Over the past years, numerous Building Performance Simulation tools (BPSts) have been developed with the ambition of removing some shortages of existing BPSts in addressing today's users' requirements, sometimes by underestimating the reasons for those lacks of functionality. A software improvement that is focused only on usability might oversimplifies the complexity of the model used by the tool, or its use, while a focus on rapid prototyping might respond poorly to the requirements of a certain typology of users. A critical review of today's requirements and available tools is here presented, with the aim of informing a better awareness of possibilities offered or denied by current BPSts.

Key words: Simulation tools, Building Energy Performance, co-simulation

1. Introduction

Energy savings in buildings, combined with an efficient use of energy provided by renewable sources, are essential objectives set by the revision of the European Energy Performance of Buildings Directive (European Parliament, 2010).

To achieve these objectives, an accurate estimate of the energy flows, resulting by different design/operation choices, must be available during all stages of the design process or energy audit (with existing buildings). However, designing or improving energy efficiency cannot be conceived apart from the maintenance or achievement of other important and concomitant goals, such as assuring environmental health (hygrothermal, acoustic and luminous), reducing costs, pursuing environmental sustainability, etc. To manage a multi-purpose design process characterized by potentially conflicting objectives, one or more tools are needed to "inform" the designer on the outcomes of his choices. To understand the complex nature of the system under investigation, i.e. the building, which consists of a plurality of interacting elements (building envelope, plant systems, users and services) tools meant to "emulate" its behaviour (through simulation) with an assured level of accuracy are required. These tools, referred to as dynamic simulation programs, are based on an approximate model (mathematical, physical, etc.) of the system under investigation, which must be able to adequately describe its complexity, variability and interconnections. It is understandable that the selection or defini-

tion of a model not appropriate to mimic the main aspects of the system under enquiry can lead to an erroneous or too shallow assessment of the different, investigated, design choices. It is therefore essential to use in a competent and conscious way each available tool by evaluating opportunities and possibilities offered by it. Among the building performance simulation tools existing today, for reasons of historical development, it is difficult to find one that integrates all the functions potentially required by a designer, both horizontally, such as thermal, energetic, lighting and acoustics analysis, and vertically, such as integral equations and CFD analysis of flow fields (Clarke and Hensen, 2015). By contrast, a number of specialized tools have been developed in each particular field to enable different levels of detailed analysis. It follows that, for a truly integrated design, a plethora of different tools should be used, sometimes requiring a redundant data entry for the description of the building or its part in each different tool. Therefore, for the analysis of multi-objective problems and with the increase in the level of detail required to study the different parts of the simulated system, it becomes more and more important the possibility to establish a dialogue between various specialized tools. This dialogue should be as transparent as possible to the user, i.e. the user should do as little as possible to gain this possibility and, certainly, he should not be asked to describe the simulated system more than once.

This urgent request raised, one more time, the question whether there are new requirements which must be met by today's Building Performance Simulation tools (BPSts).

The first element to consider is that today the potential user of such tools is not anymore "unique" as in the past; currently the figures involved in the evolution of these tools can be divided into three categories:

- the "basic" user;
- the "advanced" user;
- the developer.

The major concerns of the basic user are:

- the reduction of the time required for modeling, data entry and analysis of the results gained by multiple design alternatives. To support the decision process, each of these design alternatives should be identified by indicators calculated by the tool, meant to quantify the performance of the system with respect to a sufficient number of economical/functional aspects;
- the possibility of accurately simulate the largest number of existing and/or under-developing systems.

These two desires identify two main requirements for the BPSt: "operability", which answers to the first concern, and "adequacy", which meets the second request by embracing completeness and accuracy of the models implemented in BPSts.

As confirmed by one of the last surveys submitted to the experts of the sector (Attia, et al 2012), these two macro categories of requirements, divided into several and more detailed sub-categories, are deemed as the most important ones by the basic user, even though their relative importance is influenced by the cultural background of each user (architectural or engineering). Luckily, however, the quest for "short learning time" has become, for such instruments, of secondary importance. As a matter of fact, given the importance that building energy efficiency has reach in recent years, the necessity to gain sufficient knowledge for using such tools has been finally and deservedly recognized/accepted.

In addition to the necessities of basic users, those related to advanced users and developers have significant relevance too. The continuous update of these instruments is in fact crucial for their survival. It is therefore fundamental to add additional requirements to those of the basic user, such those of being easily customizable, expandable and maintainable.

In the following paragraphs, we will try to analyze and summarize the conditions which BPSts should meet and the different answers that have been given to these needs by the existing BPSts.

2. ESSENTIAL REQUIREMENTS

The two main requirements that have been identified before were analyzed in more detail in the survey mentioned earlier (Attia, et al, 2012), where the following sub-requirements have been further explained:

1. Usability and Information Management (UIM) of interface;
2. Integration of Intelligent design Knowledge-Base (IIKB);

3. Accuracy of tools and Ability to simulate Detailed and Complex building Components (AADCC);
4. Interoperability of Building Modelling (IBM);
5. Integration with Building Design Process (IBDP).

To these requirements other two can be added, due to the spread of cloud services and the growing need to make calculations that require significant computational processing time (Mazzarella, et al 2014), i.e.:

- the ubiquity and shareability of project files (in the form of BIM, simulation models, final reports, etc.);
- the reduction of the required calculation time, especially in case of multi-objective optimization and/or multi-physical simulations (CFD, ray-tracing, etc.), through the optimization and parallelization of the code.

2.1. Operability vs Adequacy

Operability includes all the sub-requirements listed above except one, i.e. the adequacy, which is fully described by the requirement AADCC (accuracy and completeness). It must be noted that the adequacy of a software does not imply the correctness of its final results, which also depend on the way of application of the specific instrument. In fact, an appropriate tool can be misused if the user introduces an inadequate representative model of the system under enquiry. This latter aspect, dominated by users' expertise, will not be further argued in current discussion.

Even if the two identified families of requirements are considered the most important by the users of BPSts, their relative importance is not equally considered by all of them (Attia, et al 2012). This difference in users' expectations has led to some of the shortcomings showed by current BPSts, as the result of a divergent orientation of the development market of such tools. As a matter of fact, a category of users has shown more interest in the automation of the design process via an IIKB, while the other in the AADC (Attia, et al 2012). This has led some developers to work primarily on the graphical user interface and on simplifying data entry (operability), without devoting too much concern about other aspects, in opposition to other developers who have done little to simplify the input process, but have dedicated their efforts on the quality of calculation algorithms (adequacy). A simulation tool that is responsive to current requirements, however, must be developed respecting both these expectations. This can only happen when the different figures, able to drive the development of BPSts, will share the same aims, as stated in (Attia, et al 2012): "*the typical uni-disciplinary design process where the architect and engineers work in separate islands and with no performance goals cannot achieve the new millennium objectives*".

Therefore, it is important to pursue "operability" requirements, without producing adverse effects in terms of adequacy of the obtained results. The answer to these requirements can be neither to simplify the underlying simulation model regardless of its real complexity, nor to over-simplify the use of a complex model by deceiving its properties. To understand how these complexity reduction mechanisms should be combined with the accuracy of the simulation results, while maintaining a sufficient completeness, it is better to take a step back and ask ourselves what confidence we have with the degree of accuracy of the calculations performed by the different BPSts available on the market.

2.2 Accuracy Assessment

In recent years, a renewed attention has been paid to the assessment of the degree of error reached by the estimates gained through BPSts. In particular, it has been given greater attention to the errors made in the assessment of zone air temperatures, or air nodes temperatures. In the past, the main interest was mainly fixed upon the assessment of integral errors in terms of energy use, while little was said about the estimation of the temperatures reached by the different components of the building system. If we recognize the same importance to energy consumes and thermal comfort requirements, a correct evaluation of the temporal profiles of the temperatures reached at the inner surface of the walls and by air volumes becomes a matter of principal importance.

In this regard, it is central to underline one of the results obtained during the definition of a methodology for the empirical validation of BPSts, developed as part of Annex 58 of the

IEA Project ECB "Reliable Building Energy Performance Characterisation Based on Full Scale Dynamic Measurements" [1]. The results obtained from a first sample application of this methodology (Strachan et al 2015), have, in fact, revealed that none of the most popular BPSts has produced results that excel in terms of accuracy, in all the metrics used, compared to the others.

First, it should be pointed out how important is the definition of a set of different metrics by which to evaluate the accuracy of the calculation and of a set of variables, of different nature, temperatures and flows, to which apply each of the previously defined metrics, to understand the different distribution of the errors made in the evaluation of a variable. The IEA Annex 58 validation methodology has identified two "significant indexes" upon which assessing the accuracy achieved in the estimate of the different variables of interest (Strachan et al 2015), i.e.:

- a "magnitude fit" index;
- a "shape fit" index, or level of correspondence in the shape of the profile assumed by the considered variable.

The first index, of "magnitude fit", was defined as the absolute average difference between measurement and prediction gained for each variable of interest, for each experimental period. The second index, of "shape fit", was given by Spearman's rank correlation coefficient (Kendall and Gibbons 1990) between predictions and measurements of the same variables. These two metrics were applied to air node temperatures and to the power supplied by the heating system.

To these metrics and variables, however, it may be useful to add some more assessments. For example, the set of variables, to which apply the different metrics, could be extended to the inside surface temperature of the walls of the different thermal zones, while the applied metrics may include other assessments practical for understanding the level of matching of the shape of the profile or "shape fit" assumed by each variable. In fact, the coefficient of Spearman is a mean value index, while it might be useful to have information relative to the time lag eventually detected on the occurrence of peak values in predicted and measured variable or relative to the estimation error of these peak values, which was one of the metrics used in some previous methodology, such as the ASHRAE Standard 140-2011 (ASHRAE, 2011).

Even this example of application of the proposed validation methodology showed significant differences in the evaluation of air nodes temperatures obtained from different models and tools and has confirmed the importance of the identification of the metrics and methodology to be used for evaluating the accuracy of currently available BPSts to enable the comparison of the results gained by them. Often, in fact, the different BPSts are tested by following different validations methodologies and/or the obtained results are summarized only partially, making the comparison between the different tools more complex.

2.3. Evolutionary growth

A BPSt, by describing complex and evolving systems, must be subjected to a continuous development/update. The chosen development model could determine its success or disposal, as described by (Fischer, et al 1994). Fischer has identified as the unique model applicable to such systems (highly complex and constantly evolving) the SER model: "Seeding, Evolutionary growth, and Reseeding", showed in Figure 1. This model relies on a perpetual cyclic process where periods of not planned evolutionary activity, in the hands of the users, are followed by scheduled periods of (re)structuring and capitalizing on the work previously done, with the help of IT developers.

BPSts could therefore benefit from a structure that allows them to evolve with their users' experience, enabling them to adapt and respond to unknown requirements during the early stages of development. This development model is in fact based on the belief that human-computer interaction will evolve from "easy to use" (which does not mean that skills related to the understanding of the inherent complexity of the tool are not necessary) to "easy to develop" (Fischer et al., 2004). Examples of End User Development (EUD) are represented by the ability to record or write macros in Visual Basic within a spreadsheet or the ability to use a particular language for the configuration of a CAD drawing. The importance of facilitating such Evolutionary Growth is, as a matter of fact, one of the drivers/catalysts in the use of general-purpose programs. These kinds of tools have

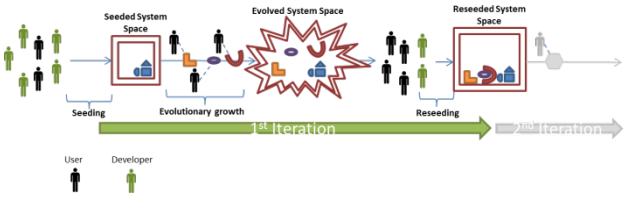

Figure 1 – Seeding, Evolutionary growth, and Reseeding development model for complex and evolving systems.

have been in fact created to facilitate the definition of mathematical models not yet implemented in available BPSts.
It will be necessary, of course, to implement levels of "user-customization" characterized by different development difficulties (simple, advanced and expert) and, for each level, the most suitable tool should be identified and implemented. Therefore, the user should not become an IT "expert", regardless of the level of "personalization" desired, however, if this desire for modification is high, more computer skills may be needed.
An important activity that requires no computer skills could be the enrichment of the database of "components" available for the simulation. This enrichment could be performed by entering data, relative to product available on the market, through web pages, according to the exchange format which is more adequate to be used by dynamic simulation codes.
An activity that would call for greater commitment is that of implementing new models or modifying existing ones. In these cases, to allow a good cooperation between different developers separated in space and/or in time, it would be of fundamental importance to implement a clear code structure, to define useful development rules, to clearly present the results achieved and to continuously manage and document the code developed.
This type of evolution, from time to time, has been considered sufficient to determine the success of a software against another. Such a type of evolution, however, leads to the creation of different parts of code, created by different mindsets, sometimes to tackle contingent problems, which sometimes are not properly coordinated or harmonized or are in partial overlap with each other's. Such an "heterogeneously" evolved system could benefit greatly from a better structuring, generalization and formalization. This Reseeding phase should push IT developers to collaborate again with the experts of the application sector of the software and would allow the re-implementation to benefit from new languages, frameworks and IT technologies previously not available.

3. EXISTING TYPES OF BPSt

In the web page "Building Energy Software Tools" [2], until recently managed by the US Department of Energy and devoted to briefly present currently available BPSts, by selecting the filter "Whole-building Energy Simulation", we will get a list of 42 software, ranging from more or less complex calculation engines (among the best known: EnergyPlus, ESP-r, TRNSYS, IDA/ICE and the Modelica Buildings library) to graphical user interfaces designed for supporting the use of such calculation engines (among the best known: DesignBuilder and Openstudio). In some cases, it is not immediate to understand if you are watching an interface to a calculation engine (and in this case, which calculation engine) or a calculation engine.
Sometimes, even though not always, together with the tool are distributed also the input files used during the validation process of the BPST and the documentation summarizing all or parts of its results. These two aspects are indeed of fundamental importance to compare the different BPSts in term of both "Operability" and "Accuracy".
Certain of those programs, such as IDA ICE and the Buildings Modelica library, are born with the idea of exploiting languages more close to those of the scope of the application of BPSts, along with a general-purpose computing engine, to make the evolutionary growth phase more practical by a user non expert in information technology and/or numerical analysis.
Currently available tools can be therefore grouped in two main families:
- special-purpose programs;
- general-purpose programs.

3.1. Special-purpose programs

"Special-purpose programs" are programs written in a computer language, like "C" or "FORTRAN", with their relative evolutions, for

solving a well-defined problem. The development of this kind of tools starts with the definition of a specific problem, characterized by a certain structure and nature. For this specific problem, an efficient numerical/computational solution is sought and implemented. This approach is eligible to achieve a good robustness of the implemented code and limits the risk of generating ill-defined problems, as long as the input data are reasonable.
The specific nature of these programs do not preclude a modular structure, which is more or less favorable to the inclusion of new pieces of code. TRNSYS, for example, allow to easily include new implementations which are relative to technical-systems parts, while it is not the same for changing/expanding models which are relative to the building. This greater or lesser ease in the management/enrichment of the code is sometimes linked to commercial issues, although more often it is due to the structure with which the computational model has been implemented for a particular system, such that of the building. In some cases, in fact, such as ESP-r, the computational model that implements the mathematical model for the exchange of mass and energy within a building is implemented in a single matrix, which is then partitioned in order to reduce the calculation time. This matrix, in fact, can reach significant size, leading to long calculation times for its inversion. Numerical analysis, suggests methods for partitioning sparse matrices aimed at reducing required computation time, but to add or change parts of the mathematical model represented computationally by this matrix, it is necessary to understand its structure and it might be needed to consider new partition routines, if the latter ones had not been implemented following an appropriate “a priori” logic. Another approach used to reduce problems of handling such large matrixes, had been to introduce “generic” addenda into the equations of the mathematical models implemented inside these matrixes, in order to allow, through their use, to introduce aspects not previously considered.

3.2. General-purpose programs

General-purpose programs are created to solve any kind of problem which the user might decide to describe in it. The user of such a type of software, to simulate a given system must, in general, define a physical model, then its mathematical model, and then introduce it in the program, through its specific programming language. For the solution of this model, the program exploits the numerical libraries included in it, but which may not always be adequate to the needs. For example, if the mathematical model is described by a differential equation of partial derivatives and numerical libraries available to the program, as in the case of Modelica and IDA ICE, are only capable of solving ordinary differential equations, the user must change the mathematical model which he intends to simulate in order to revert to what is resolvable by the general-purpose program that he is using. As well as forcing the user to build a mathematical model of the system to be simulated, which involves specific skills not owned by all potential users, such programs often require the knowledge of a proprietary programming language, which is functional to the formal description of equations involved in the system, more that to the implementation of numerical algorithms.
The purpose of this type of programs, characterized by high extensibility and flexibility within the domain of operation, is to limit the responsibilities assigned to the developer to the model description, by delegating their numerical solution to the libraries implemented in the tool.
Among the general-purpose programs must be cited Dymola, OpenModelica and IDA ICE. Dymola is the most used commercial front-end for Modelica, while OpenModelica is its free alternative. Modelica is a declarative language for object-oriented modeling. The Neutral Model Format (NMF) is the language, introduced in the late ‘80s by Per Sahlin (Sahlin and Sowell, 1989), upon which was developed IDA ICE and which was the predecessor of the Modelica language. While IDA-ICE is born for the simulation of building performance and therefore is natively equipped with a library of specific component models recalled transparently by the user through a graphical interface (which therefore can avoid building the various mathematical models through programming language), Modelica, which maintains its general purpose feature, supports this objective through the "Buildings Modelica library", an open source library developed for this purpose. However, from the developer's point of view (and not from that of BPSt users) there are some cases, such as the calculation of the form factors between com-

plex geometries, for which the biggest problem does not lie in writing of one or more mathematical equations, but in the optimized management of the numerical calculation process. In these cases, the proprietary programming language, aimed at the formal description of equations, is less efficient and functional than a classic programming language, such as C ++, C #, Fortran, etc.

Some disadvantages of the approach used in these type of programs are: lack of computational efficiency, limitations in the solution of mathematical systems whose numerical translation is not already included in the numerical libraries available to the program and difficulties in understanding error messages generated by the program during the simulation, because of the symbolic manipulation applied to the system of equations.

Another example of general-purpose programs includes the use of Matlab, coupled to Simulink for the definition of blocks meant to simulate the building system.

In this case Simulink offers its drag and drop functionality while Matlab an extensive library of numerical and statistical methods. However, some drawbacks of this approach have been pointed out by (Zupančič and Sodja, 2008), such as:

- It imposes the development of procedural models;
- It assumes that a system can be decomposed into block diagram structures with causal interactions;
- It is a "signal-oriented" environment, that often leads to algebraic loops whose numerical resolution might be risky;
- Developing something in this context implies the cost of the Matlab-Simulink license.

3.2. Co-simulation

The possibility to couple special-purpose codes between each other can be a solution when the dimension of the tool, in term of lines of code, poses difficulties in the maintenance and growth of the tool.

An example of the implementation of this coupling is represented by the development of a new group of elements within EnergyPlus, such as the "ExternalInterface". To implement such coupling, whose management is not yet completely transparent to the user, changes in the EnergyPlus code have been required and, in some cases, a manager is required for the exchange of information between involved agents. This manager, an implementation of which is the Building Controls Virtual Test Bed (BCVTB), can be avoided when using the Functional Mock-up Units Import (FMU) option in EnergyPlus. In both cases, this coupling is meant to assign externally calculated values to those variables of a monolithic software, for which the external interface has been expressly developed. At the moment, the use of this opportunity has been mainly related to "loosely coupled" problems, i.e. problems for which it was not "strictly necessary" to perform an iterative interaction between the various actors. In particular, in EnergyPlus the external interface has been defined, up to now, for three types of input, namely:

- ExternalInterface:Schedule, or ExternalInterface:FunctionalMockupUnitImpot:To:Schedule
- ExternalInterface:Actuator or ExternalInterface:FunctionalMockupUnitImport:To:Actuator
- ExternalInterface:Variable or ExternalInterface:FunctionalMockupUnitImport:To:Variable.

Other external interfaces have been defined to expose to the other actors the output of EnergyPlus calculations, such as the:

- Output:Variable;
- EnergyManagementSystem: OutputVariable; ExternalInterface:FunctionalMockupUnitImport:From:Variable

While the function of the first two input interfaces is easy to understand and their nature of "controller-actuator" can be recognized, the last might seem more general, but it actually has a feature similar to the EnergyManagementSystem:GlobalVariable (an internal object of EnergyPlus for plant management), i.e. to pass values to variables of a specific EnergyPlus internal modeling language for the definition of control and management systems, called Energy Runtime language (Erl). The difference between this approach and that managed internally by EnergyPlus, is that in this case, the numerical value passed from the external interface to EnergyPlus at the beginning of each time step for a thermal zone, remains constant within this time step, not allowing the change due to iterations performed within the same time step. A strategy of this kind could be inefficient, in some cases, both in computational and implementative terms. The development of

a special-purpose software that has been implemented according to an "enhanced modularity" (Mazzarella and Pasini, 2009) might be more efficient than such a coupling, thanks to some facilities offered by the structure of object-oriented programming. Thanks to object-oriented paradigms, the developer can easily locate the piece of code to be replaced and/or extended and modify it, by maintaining its "relative position" with respect to other calculation procedures, by inheriting already implemented functionalities that might be useful for the new implementation.

Another concern, when coupling different tools, regards the risk of an inappropriate use of models born with specific assumptions, perhaps forgotten over the years. Often, in fact, some of these programs have evolved over many years and contains solutions considered sufficiently satisfactory for a given historical moment or which represented the only possibility for the time. For example, today EnergyPlus requires a time step of calculation of maximum 3 minutes when the finite difference method is selected for the computation of heat exchange by conduction within the walls. This short time step does not blend well with models that do not consider the response time of the simulated systems and, more generally, with system-related stationary models. A stationary model for the simulation of HVAC systems was in fact well suited until the time step of discretization was of the order of one hour. To avoid such coupling problems, the tool should generate error messages when the time step is not sufficiently bigger than the characteristic time of those models simulated with steady state solutions. The coupling of different tools must therefore be pursued only after a careful analysis of the tools to be joined has been performed.

The Reseeding phase would be functional, in such cases, to rationalize what has been developed over a long period, by verifying appropriateness and consistency of the different parts of the code and by combining those parts in the most efficient way.

CONCLUSION

When choosing among currently available BPSts to design buildings and/or larger systems, to be as efficient as possible and to be able to proficiently exploit energies from renewable sources, it is necessary to consider the possibilities offered by the different BPSts in terms of accuracy and completeness, while taking into account ease of use and support facilities for the design process, eventually included in the tool.

Unfortunately, the assessment of the accuracy of the different tools is still a difficult process. As a matter of fact, currently available BPSts do not give the same results, especially when estimating variables considered of secondary importance in the past. Besides, not always, all the supports useful for comparing their results accuracy are given together with the tool, such as the input files for the software, which implement the models outlined in the validation procedure and a complete report containing all the indexes defined in the validation methodology, applied to all the identified variables. Certainly, the difference between simulation and operation is partially due by the randomness of some of the drivers of the simulation, which cannot be represented by a single scenario, and the knowledge of the accuracy achieved in each scenario is the starting point to be able to consider many of them, in order to evaluate the performance of the designed system in different circumstances.

Given the difficulty of pursuing simultaneously objectives potentially conflictual, such as "operability" and "adequacy", the efforts spent in developing BPSts have been torn apart, resulting in the distribution on the market of a number of tools, with very different features.

New general-purpose programs have tried to give an answer to the need of rapid model development and/or coupling of existing models developed for the simulation of parts of the system under enquiry, while many of the new implementations of special-purpose software have been focused on incrementing ease of use, sometimes by reducing the complexity of the simulated system. Co-simulation has tried to address the difficulties involved in modifying the oldest special-purpose programs. However, not always what can be coupled, should be coupled or is efficiently coupled from a computational and implementative point of view, therefore caution is advised in using such solutions.

In conclusion it must be put in evidence that for

a tool to be easily maintainable and extensible, the choice is not compulsory among general-purpose or special-purpose solutions, but lies in the implementation of an architecture for the code, which is characterized by enough modularity. It is important also to remember, that no BPSt can compensate for a lack of user expertise in creating a proper representative model of the system at enquiry.

REFERENCES

ASHRAE Standard 140-2011. 2011. Standard Method of Test for the Evaluation of Building Energy Analysis Computer Programs. ASHRAE, Atlanta, GA.

Attia S., Hensen J.L.M., Bertran L., De Herde A. 2012. Selection criteria for building performance simulation tools: contrasting architects' and engineers' needs. Journal of Building Performance Simulation, 5(3), 155-169.

Clarke J.A., Hensen J.L.M. 2015. Integrated building performance simulation: Progress, prospects and requirements. Building and Environment, 91, 294-306.

European Parliament, 2010. Directive 2010/31/EU of the European Parliament and of the Council of 19 May 2010 on the energy performance of buildings (recast). Official Journal of the European Union, 18(06).

Fischer G., Giaccardi E., Ye Y., Sutcliffe A. G., Mehandjiev N. 2004. Meta-Design: A Manifesto for End-User Development. Communications of the ACM, 47(9), 33-37.

Fischer G., McCall R., Ostwald J., Reeves B., Shipman F. 1994. Seeding, evolutionary growth and reseeding. Proc. of the SIGCHI conference on Human factors in computing systems, Boston, United States, 292-298.

IEA Annex 58 2015. Reliable Building Energy Performance Characterisation Based on Full Scale Dynamic Measurements. http://www.kuleuven.be/bwf/projects/annex58.

Kendall M., Gibbons J.D. 1990. Rank CorrelationMethods. 5th ed., 69–77. London: Edward Arnold.

Mazzarella L., Pasini, M. 2009. Building energy simulation and object-oriented modelling: review and reflections upon achieved results and further developments. Proceedings of Eleventh International IBPSA Conference, Glasgow, July 27-30, 638-645.

Mazzarella L., Pasini M., Hoonejani Shahmandi N. 2014. Challenges, Limitations, and Success of Cloud Computing for Parallel Simulation of Multiple Scenario and Co-Simulation. 2014 ASHRAE/IBPSA-USA Building Simulation Conference Atlanta, GA September 10-12.

Sahlin P., Sowell E.F. 1989. A neutral format for building simulation models, Proc. of IBPSA '89 Conference, Vancouver, Canada, 147-154.

Strachan P., Svehla K., Heusler I., Kersken M. 2015. Whole model empirical validation on a full-scale building. Journal of Building Performance Simulation, 1-20.

Zupančič B., Sodja A. 2008. Object oriented modelling of variable envelope properties in buildings. WSEAS Transactions on Systems and Control, 3(12), 1046-1056.