Building Simulation 2009

Eleventh International IBPSA Conference
Glasgow, Scotland
July 27-30, 2009

BUILDING ENERGY SIMULATION AND OBJECT-ORIENTED MODELLING: REVIEW AND REFLECTIONS UPON ACHIEVED RESULTS AND FURTHER DEVELOPMENTS.

Livio Mazzarella[1], Martina Pasini[2]
[1]Department of Energy, Politecnico di Milano - Milan, Italy
[2]Department of Building Environment Sciences and Technology, Politecnico di Milano - Milan, Italy

ABSTRACT

Over the past 30 years numerous Building Simulation Codes (BSC) have been developed. Nevertheless, none of them has yet become a "standard". Focusing the attention on the use of advanced Object-Oriented Modelling, a review of the most used BSC is here carried out. First, new requirements have been investigated. Actually, the evolving nature of Building Systems and of society demand for tools designed to evolve at the hands of the users. Modularity seemed to be the answer to these needs. However, in the BPS (Building Performance Simulation) field, modularity has been interpreted in different ways leading to different approaches to Modelling and Simulation (M&S). A first purpose of this work is to redefine what modularity is needed in BPS tools (an "enriched" one) and subsequently to investigate if and how actual tools address that. Focusing on Building Fabric's M&S, we have investigated the different approaches taken by Esp-r, IDA ICE and PsiGene. The goal was to understand if they could allow: simpler distributed calculus, distributed responsibilities in code's developing, an exponential evolution of algebraic routines and an increased productivity and widespread use of these tools, thanks to design and process automation. The finding was that almost always when one of these requirements was addressed another one was unsatisfied. It could be concluded that such BPS tool (characterized by an enriched modularity) is not yet ready and that a *technology driven research* should be pursued to develop it. Indeed, different concepts derived by Information Technology should be further analyzed in order to understand if similar formalization might lead to a renewed modelling, simulating and validating methodology more focused and oriented towards BPS tools' evolutionary growth.

INTRODUCTION

Energy savings in buildings is today mandatory in developed countries, so it is imperative to perform, during the design stage, an accurate estimation of the energy used by buildings to assure different kinds of comfort. To do that, we need first to describe the object of the design, as a collection of relationships among parts, i.e. a system (the conceptual model), and then to "emulate" its behaviour (through simulation) with an *assured* level of accuracy. The object of the design is the Building System or BS, which comprises Building's Fabric, Building's Envelope and Service Systems. The need to use simulation stems from its nature, i.e. from its complexity and time dependent performances. Its behaviour depends, in a complicate way, from different aspects, causing problems when simultaneously addressing all the interrelated performances of each subsystem. Simulation is the only way to allow the designer to explore the complex relationships between environment and building's form, fabric, services and control. Besides, innovative technologies and knowledge progresses impose the development of new models and powerful modelling and simulation techniques in different research fields and application domains. These common issues, along with strong differences, claim the importance of a shared resolution approach, able to promote the work done in different fields and to allow reuse by modification. Modularity claimed good features for answering these requirements. However, in the BPS (Building Performance Simulation) field, modularity has been interpreted in different ways leading to different implications. The reason for these differences is certainly the issue's complexity, but it is also the lack of exploitation of progresses achieved in other research fields. Hence, the aim of this paper is to understand:

- why a *fully developed suite of high quality software applications* is still not available;
- how *modularity* has been used till now and which kind of problems has encountered;
- which new *logical structures* might help to define a new modelling, simulating and validating methodology.

To do that, we will try to understand:

- which *new needs* BPS tools should address and *how old BSC might cope* with those,
- what are *meanings* and *aspects of modelling and simulation (M&S)* and why different development process approaches took place;
- which are the *major barriers* in BPS tools' *development process* (validation, numerical solution, etc);

- which kind of *tool have been developed* till now, with which *major differences*.

NEW NEEDS & EXISTING CODES

BSs have changed in the last years, as society has and as the possibilities offered by other technologies have. Many of the BPS tools on the market were born years ago and tried to address past requirements with the help of past knowledge and Information Technologies (ITs). Today, advanced knowledge and tools are available and requirements are changed. Globalization demand for economic convenience in using BPS tools; the law is struggling to emanate general, usable, and reliable regulations to address the environmental problem, while the evolution of BS and society demand an evolutionary growth of such tools. In the following, some of those new needs that more influence the BPS tool's development process will be analysed to understand which issues should be addressed and solved by today's BPS tools.

BS's evolution

Some of the past requirements were dictated by the nature of past BSs. Traditional constructions allow specific assumptions that do not cope with new BS typologies. Large highly-glazed spaces are usually not well represented; room's irregular geometry is usually not well taken into account when calculating internal longwave energy exchange; Indoor Air Quality impositions require accurate determinations of air flow inside and between different zones; the effect on building performances of shading devices linked with internal room geometry cannot be easily assessed; innovative building component or materials are not appropriately represented, etc. To make an example, the Double Skin Façade, an "innovative" technology for the envelop component, is sometimes treated like a whole zone. However, some of its features are not automatically represented by the *common behaviour* of a zone. The air, entering the inlet openings, is heated by some degrees; this effect, along with the presence of the shading device, might affect its natural ventilation; also the determination of inside superficial convective coefficients has more influence in this case than in that of a generic zone.

Today, BSs are thus becoming more and more heterogeneous with components from many engineering domains. They are complicated systems, governed by control laws, filled with innovative responsive elements (plant system, new material, *strategic* building features, etc.). Therefore, BPS tools must include these new complexities and be able to solve simultaneously coupled problems, as already claimed by the need of integrated simulation (Clarke, 2001b; Kelly and Strachan, 2001).

Due to this evolving and inherent interdisciplinary nature, collaboration among model developers should be certainly enabled and encouraged.

Coping with evolution

As biology confirms, complex, natural systems are not created all at once but must evolve over time (Dawkins, 1987). How this evolution takes place is a crucial issue, as described by Fischer's model of evolution (Fischer, *et al* 1994): the Seeding, Evolutionary growth, and Reseeding model (SER). Accordingly to this model, to actualize tools like BPS ones, "*activities of unplanned evolution and periods of deliberate (re)structuring and enhancement must continually alternate each other"*.

Consequently, BPS tools must be designed to evolve with the user's experience, since they cannot be *a priori* designed able to fulfil any user requirements. This evolution will lead to the creation of different "species" grown in different contests to face different problems, i.e. to a growing information space (a mixture of annotations, partial designs, and discussions), which must be structured, generalized, and formalized periodically to incorporate such "natural" evolution into a consistent updated frame. This phase of reseeding, brings IT developers back in to collaborate with application domain experts and might benefit from new languages, frameworks, enhanced functionalities, etc.

User & Developers' role and related consequences

To allow evolutionary growth, advanced features should be introduced or linked to BPS tools. Such features might be: model development facilities, tools for databases compilation and query, for the personalization of user interface, for optimization activities, for the management or presentation of input and output, etc. Some of these facilities might be acquired by interoperability among software; others would benefit a higher level of coherence and therefore should be more deeply integrated.

Project that follow the SER model are based on the belief that human-computer interaction will evolve from *easy to use* (even though not yet completely achieved in the case of BPS) to *easy to develop* (Fischer et al., 2004). Examples of End User Development (EUD) are already present in today's tools, such as recording or writing Visual Basic macros in spreadsheets, using lisp or GC scripting for parameterize CAD drawing with Autocad or with GenerativeComponents, etc.

There will be different levels of development to allow user's customization (simple, advanced and expert development mode). For each level, the right tools should be detected and implemented. Users should not become IT experts. According with the level of interaction they aim to reach, they might be asked to gain some more insight in IT. What might be helpful in such user's customization process is a clear model structure view, which may be achieved by the use of *functional block diagrams.* The navigation through "similar class diagrams", explicitly developed for users, for instance, can give with a picture a lot of information also at model

level. This is crucial for concept's communication, problem's detection and existing code's comprehension, since avoids burying domain knowledge in an unreadable code, where model's assumptions and equations are spread all over the program.

The Open source approach

One important aspect in the evolutionary growing process is the enrichment of the simulation components library. This enrichment might consist in a *compiling activity* (insertion of data, describing equipment, materials, etc., into a specific computer code, e.g. via web pages) or in a *modelling activity* (development of new models or modification of old ones) and may be achieved effectively, following the Open Source Scheme's modelling activity approach.

Having an enlarged pool of potential model developers, the number of implemented models might increase exponentially. A structured web-based information repository is surely needed to assure models diffusion and maintenance. The definition of development's rules, the clear presentation of achieved results and the management and documentation of developed code would allow several developers (separated in space and/or time) to collaborate. A cross-validation might occur as a first kind of validation.

The reseeding phase should follow periodically to ensure the implementation of a stable release of the source code (placed in a specific directory).

However, to guarantee the reseeding phase with a correct schedule, some higher organization should be charged of that. A public-figure might have the right and the interests to fulfil this role, since today's energy conservation laws are asking for more and more complex energy performance standard.

DEVELOPMENT PROCESS' ISSUE

To handle efficiently BS's complexity, many issue had to be faced at different levels, starting from mathematical models joining, to stability issues. A coherent and integrated approach is required. In the following, we will examine some issues concerning the development process of a multi-domain, intelligible, stable, robust and extendible BS's model.

Modelling vs. Simulation

Some model's development approaches tend to separate the activities of modelling from that of simulating. ***Modelling*** is the act of describing a system, by extracting, organizing, and representing in some unambiguous way the knowledge gained upon the System Under Investigation (SUI), i.e. by building a *conceptual model* (CM). There are different system description levels and different languages or *meta-languages* to do that. Undoubtely the features of the CM will affect quite all the aspects of a simulation study: the data requirements, the developing model speed, the model validity, the experimentation speed and the confidence in model's results. However, the CM's notion is still vague and not well defined. Someone separates the CM from the communicative model; others identify a domain-oriented and a design-oriented CM. What seems to be agreed is that the CM refers to the early stages of a simulation study and that it *"is a non-software specific description of the simulation model that is to be developed, describing its objectives, inputs, outputs, content, assumptions and simplifications* (Robinson, 2004).

Simulating is instead the act of performing experiments on the model to make predictions about how the real system would behave; that is how the real SUI would react when subjected to such stimulating conditions. Even in this case some differences exist between a *simulation model* and a *simulation program* (Birta & Arbez, 2007).

A *simulation model* (SM) is the piece of computer code that embodies the SUI; a *simulation program* (SP) is the stand-alone executable code employing such SM and all the other needed facilities (I/O, etc.).

Any of the programming languages today available may be used to build a SM and, later on, a SP, but, typically in the model design phase, model oriented languages, designed expressly to support simulation studies, like Modelica, CSIM, and SIMPLE++, may be more helpful. In these cases, it is possible to develop BPS tools in an advanced "framework" expressly designed for the development of SP.

Verification and Validation

The purpose of a SM is to provide an adequate emulation of some SUI performance. However, as Karl Popper pointed out: "*theories are not verifiable, but they can be corroborated*". Consequently, to assess the SM suitability in respect to the project goals, different activities of Verification and Validation (V&V) should be performed[1].

However, the way the CM evolves towards the SP and the SM's architecture might have an influence on BPS tools' attitude towards V&V.

For example, the possibility to "isolate" part of the model to validate it might be valuable. Nevertheless, in some cases, it will be mandatory to verify or validate each module used in combination with others, instead of validate each of them separately. This has led to the problematic definition of an empirical or analytical whole model V&V methodology (Jensen, 1995; Xiao *et al*., 2002).

Besides, to take under control model's evolution, a specific structure used to encapsulate tests which "*allow developers to ensure that recent code modifications have not resulted in unforeseen impacts on program predictions"* has also been developed (Ben-Nakhi & Aasem, 2002). Still the

[1] for much insight into V&V's activities and meanings refer to Oberkampf & Trucano, 2002

evolutionary nature of BPS tools has led also to the necessity of diffuse V&V activities (Jensen, 1995).

These are some of the aspects that should be handled when defining model's structure and tool's development process.

Modularity's concepts

To allow a development at a community level, the concept of software modularity might help. Once interface's roles have been defined, software modularity should allow each expert to easily develop functionalities he is specialized in. However, modularity has different goal and implication at different levels of the tool's implementation.

At a common user level, modularity provides design functional elements, structured in a way to allow the mixing and recombination of an arbitrary number of these functional components in a specific layout. Consequently, ***functional layout modularity*** and configuration's flexibility are the main aspects at this level. If we look at hospital design, a functional layout modular structure of the model implemented into a BPS tool will allow the designer to choose among modules (intensive care unit, reception area, operating room) or easily add a new one. Such modules may define a "template" for hospital design. An important aspect for modules composition is connection rules' definition, either imposed directly by the user, or automatically generated. Some rules might be mandatory (a bathroom ***must*** be divided from the living room), other not (the kitchen ***should*** be near to the dining room). Generalising, this is the BS's *representational level* of the BS's model. At this level, the user deals with components or subsystems that are real entities and "should only connect the pipe with the radiator". Of course, module's granularity levels should be consistent with the specific design process' goal.

Along with the *representational level*, there are also a *mathematical level*, a *numerical level* and a *code level*, which affect the definition of tools' modularity.

At a *mathematical level*, modularity leads to a structure where combining new subsystems' mathematical models is easy and does not require modifying the entire system model. One fundamental paradigms of Object Oriented (OO) programming, i.e. encapsulation, exemplifies this concept. Actually, it aims at hiding the internal mechanisms and data structures of a software component, e.g. a subsystem's mathematical model, behind a defined interface. In such a way, the other subsystems' mathematical models only need to know what information they have to exchange through that interface. This approach will allow easy ways to replace or improve mathematical models, and may be identified as ***mathematical models modularity***.

To support the evolutionary growth phase, a modular mathematical model, which does not assure extendibility by default, has to comply with some constrains: the definition and use of *standardised module interfaces*. This will allow developing any new mathematical model as a real stand-alone module, which will be later just "plugged in" into the program. This kind of standardization will also improve flexibility and may be identified as ***standardised mathematical models modularity.***

However, problems may arise from the development of mathematical modularity. The numerical solution of the simulated system imposes a choice among several numerical schemes and among ways to handle interactions between mathematical modules. The way mathematical modules are combined with each other could lead to a sequential (TRNSYS) or simultaneous (Esp-r, IDA ICE) solution. The designed inter-module interaction procedure might affect the accuracy and stability of the overall code. The inter-module interaction procedure may allow or deny numerical solution's parallelization. Furthermore, the coupling through co-simulation[2] of extremely different equations could lead to conditionally stable system; hence, much attention should be paid when combining different modules, as showed by Wang & Chen, 2007. These aspects, with specific reference to performance, V&V and accuracy, should be properly handled when developing modular mathematical models. These are the modularity implications at the *numerical level.*

At the end, this standardized mathematical modularity has to be converted into code modularity. **C*ode's modularity*** aims at facilitating code's reuse and maintenance or improvement activities by encapsulation and responsibilities assignment. These requirements are clearly addressed by some of the OO design rules, such that of low coupling[3] and high cohesion[4] between different code's parts.

Hence, modularity may exist at different levels and might affect V&V processes, accuracy prevision, stability, etc. Consequently, we should be careful when speaking about it and we should try to set some constraints at each of the above-mentioned levels to reduce its influences on the tool. Nevertheless, to comply with the evolutionary growth, at any level the modularity has to assure the following goals:

- to allow users to better understand and split the simulated system in a coherent way;
- to ease modules' selection and modification, according with simulation needs;
- to allow easy ways to extend the model;
- to allow modules reuse and exchange via a web-based repository.

Now we can introduce the concept of *enriched modularity*, referring to the aforementioned features.

[2] for an analysis of the implications of different approaches to run-time interoperability among tools, refer to Trcka *et al*., 2007
[3] dependency of a software component from others is low
[4] functional component homogeneity is high and various components work together to reach higher complexity levels

TOOLS COMPARISON

Focusing the attention on the development of building fabric's simulation model and not on that of the whole BS, the main differences among BPS tools will be analyzed. Those differences concern implemented mathematical models, internal structure and applicability domains of BPS tools.

The differences concerning the implemented mathematical models are related to: how conduction heat transfer is calculated (numerical or analytical based methods); how zone air is treated (mixing, nodal, zonal or CFD model); how convective coefficients are calculated; how longwave radiative heat transfer is distributed within zones (with a fictitious T*, with or without view factor calculation); how windows' transmission of direct shortwave or diffuse radiation is calculated and distributed within zones; how 3D effect are taken into account; which kind of controls are allowed; how shading devices are treated; which sky model has been implemented, etc.[5]

Numerical problems, computational time's increase, sensitivity to input data's uncertainty, difficulties concerning algorithm implementation, etc., have led to the implementation of one model over the others. However, even if the differences are so wide-ranging, in many cases, BPS tools have been classified according to the way conduction heat transfer through fabric was treated. Actually, heat exchange by conduction was the main concern of BPS tools once. Along that, the system composed by building enclosures has been usually represented in a "compact" way, by a large sparse matrix of equation with predefined topological structure. Probably, the governing domain theory of combined problems of thermal diffusion and aero-dynamic system, has led to this monolithic representation (Tang, 1997).

However, precisely for solving the difficulties involved in the maintenance, further development or adaptation to non-standard problem of these monolithic tools, in the mid '80s, the need for general-purpose and modular tools, led to the birth of many projects (e.g. EKS[6], SPARK and IDA ICE[7]). Nevertheless, despite the high enthusiasm and expectations for their success, we will see that their supremacy has not yet been clearly stated.

A general-purpose simulation program (IDA ICE, visualSPARK[8], Dymola[9]) treats mathematical models as input data, being characterized by great extendibility and flexibility. The other kind of tools, the Special-purpose ones, (ESP-r[10], EnergyPlus[11]), takes advantages from the structure of a class of problems to reach high execution speed. This approach leads also to robustness and limits the risk of generating insoluble problems, as far as input data are reasonable. Both special-purpose and general-purpose tools commonly lead to the construction of a global matrix for the BS. However, in the first case, the structure of this matrix is pre-defined, and it might be difficult to add other equations. While, in the second case, the tool has been designed to allow the addition of new equations by handling automatic symbolical manipulation of this new system to create a *simpler-to-solve*, or *lower index* matrix.

However, other approaches to modularity exist, thus, in the following, we will go much into the details of the features of the most relevant ones.

Special-purpose tools

Probably the most reliable and documented methodology for the implementation of building's simulation model is that used by Esp-r (Clarke, 2001a). This tool solves conduction with a finite difference approach and bases the process of the simulation model's implementation upon the automatic generation of a single sparse matrix of algebraic equations. This matrix is built thanks to the definition of some primitive parts and their characteristic heat balance equations. This approach discretizes the building in control volumes, each represented by a node. To simplify the concept, a node could be a capacity/insulation system, a surface, or a fluid volume. In all the cases, the most general configuration is studied and self-coupling and cross-coupling coefficients between state variables are defined. A routine builds up a sparse matrix of equations, where these coefficients are opportunely positioned. The solution of this overall sparse matrix is achieved through its partitioning into sub-matrices: "component matrixes" and a "coupling matrix". Each component matrix can be processed using customized solvers applied to each domain equation-set, at any frequency (to manage stiff problems' complexity). At the end, thanks to the coupling matrix, the global solution is calculated. This numerical method ensures accuracy by preserving spatial and temporal integrity of real energy systems, since it solves simultaneously at each time step a whole system of PDE-sets.

In this case, realty has been "modularized", however, to include a new equation in such a structure, it is necessary to know the global matrix's implementation routine and to automatically calculate those self and cross coupling coefficients to be added to the general matrix.

Different functionalities are offered by TRNSYS, where the user can encapsulate new models, as FORTRAN routines with standardized arguments, in so-called TYPEs. These TYPEs are represented through icons that can be dragged and dropped in a

[5] for a more complete list of mathematical models implemented inside BSC refer to Crawley et al., 2005
[6] http://www.esru.strath.ac.uk/Programs/EKS.htm
[7] http://www.equa.se/eng.ice.html
[8] http://gundog.lbl.gov/VS/spark.html
[9] http://www.dynasim.se/index.htm
[10] http://www.esru.strath.ac.uk/Programs/ESP-r.htm
[11] http://apps1.eere.energy.gov/buildings/energyplus/

Graphical User Interface[12]. However, this possibility has not been extended to building's components, limiting the level of standard mathematical model modularity achieved by this special-purpose tool.

General-purpose tools

Since the majority of the physical systems, can be characterized by algebraic and differential equations, ordinary (ODE) or partial (PDE), generic tools for their resolution have been developed. Special-purpose tools require that PDEs are turned into a system of Algebraic Equations. On the contrary, general-purpose tools try to solve, by symbolic manipulation, a "general" system of differential-algebraic equations (DAEs), composed of algebraic and ODEs (PDEs should be discretized in space). One attractive feature of general-purpose simulation tools is that they follow the "divide and conquer" rule, building successively larger component model's libraries.

Among general-purpose tools, there are Dymola, OpenModelica[13] and IDA ICE. Dymola is the most used commercial front-ends for Modelica, like OpenModelica is the free one. Modelica is an Object-Oriented, declarative, multi-domain modelling language for component-oriented modelling of complex systems. The Neutral Model Format (NMF), introduced since the late 1980s by Per Sahlin (Sahlin & Sowell, 1989) and upon which IDA ICE has been developed, is a predecessor of Modelica.

The main objectives of such language are to allow anyone:

- to easily learn modelling, by knowing the system's equations;
- to disregard numerical solution's problems or coding ones and to focus on modelling;
- to easily and quickly implement new models, even by inheriting by old ones;
- to easily understand models developed by others, by identifying each equation.

The goals of this approach are to divide modelling from simulation and to assure extendibility, flexibility of use and code's reuse. The equations written in Modelica describe equality, having no pre-defined causality and are used to define connections and modules' behaviour. The *simulation environment* manipulates these equations symbolically to determine which term are inputs and outputs and which execution order and numerical routine should be preferable to solve the entire system.

Recently, a hygrothemal wall model (Nytsch-Geusen et al., 2005) and a Multizone Airflow Model (Wetter, 2006) have been developed using Modelica. The model describing thermal processes between building components is decomposed into simple thermal processes (heat convection, long-wave radiation, etc.) connected in such a way that they share the temperature of the body and that the sum of all heat flows in such connection point is set to zero. The use of OO programming typical concepts, help in aggregating common subsystems modules into macro-components to achieve more user-friendliness. However, some *drawbacks* in this approach are: poor runtime efficiency, limitations in the solution of system with impulses as inputs, or PDEs and difficulties in understanding simulation error messages, due to the strong symbolic manipulation applied to model's equations.

Besides, even if it is rather easy and natural to achieve reusability using object-oriented modeling tools when models are described by DAE, some problems might arise when model variables are expressed with complicated algebraic algorithms. In the case of highly geometrical problems, as radiation flows' distribution among internal surfaces, object-oriented modelling is less effective and reusability is not gained implicitly by using Modelica. Additional efforts should be made to develop fully reusable components, as shown in (Sodja & Zupančič, 2008).

However, even if some advantage and disadvantage of these tools have been reported, the supremacy of one approach over the other has not been yet proved. Actually a practical and politically acceptable framework for fair comparisons of numerical performance, that takes into account at least the physical phenomena modelled, the level of ambition of the physical models, the level of numerical accuracy obtained and the time resolution obtained, does not exist (Shalin *et al*., 2004).

To give a first quantitative impression of the differences among these tools Shalin *et al*., 2004, made some comparisons between IDA ICE and EnergyPlus. An experiment where EnergyPlus was run with six timesteps per hour, led to 17,712 steps and an execution time of 250 s for a 4 month summer simulation. Running IDA ICE with a maximum timestep of 1,5 h and an adjusted tolerance of 0,015 gave 17,755 steps and an execution time of 127 s. Conversely, comparisons without natural ventilation have shown that EnergyPlus was faster.

Other comparisons between multizone building models developed with Modelica and TRNSYS, in term of development and computational times, have been presented by Wetter & Haugstetter, 2006. The authors compare the model development time for their Modelica model with the multizone thermal building model BuildOpt (giving a time estimate for the TRNSYS building model was quite impossible). They have concluded that using Modelica lead to a five to ten times reduction in development time compared to using C/C++ languages and in a four times smaller code's size. However, TRNSYS was faster than Dymola and some convergence problems have been noticed with Dymola and Simulink.

[12] for a description of TYPEs implementation and dynamic link libraries refer to McDowell et al., 2004

[13] http://www.ida.liu.se/projects/OpenModelica/

Nevertheless, since authors did not explore ways to improve Modelica model's numerical performance, they concluded that a longer computation time might not be an inherent feature of equation-based simulation environments.

The Matlab/Simulink environment

The Matlab/Simulink environment has been recently used to implement BPS (El Khoury *et al.*, 2005; Kalagasidis *et al.*, 2007). As a matter of fact, it allows to define models whose hypothesis are well known by the user, to use the drag and drop functionality offered by Simulink and to be able to use the huge library of numerical and statistical methods implemented in Matlab. However, some drawbacks of such approach have been recently highlighted by some publications (Zupančič & Sodja, 2008). Simulink: 1) imposes the development of procedural models; 2) assumes that a system can be decomposed into block diagram structures with causal interactions, leading to significant analytical transformation's efforts to prepare the problem in this form; 3) is a "signal-oriented" environment, that often lead to algebraic loops whose numerical resolution might be risky; 4) imposes a modelling structure that might forbid the subdivision of models in modules with physical meanings, causing the spread of parameters of singular components in mixed model expressions. In addition, developing something in this context implies the cost of the tool.

The "remote procedure" approach

Going back to the physics of buildings, another tool named PsiGene[14], models building's components (wall, windows, air volumes, radiators, etc.) as autonomous objects that have topological and aggregation relations and that interact asynchronously by messages, exchanging value (surface temperatures, radiation, etc.), only when necessary (Zimmermann, 2001). Within autonomous objects, physical effects are modelled in classical ways by solving the physical equations numerically. No global system of equations is solved at run-time. For example, a room object collects all the heat-flows from its neighboring objects (walls, heating installations, sun, etc.) and uses these values together with the elapsed time interval to compute its air's temperature. It doesn't have to know how those heat flows are calculated. To allow this approach, the interfaces of the objects and the data they exchange are declared by predefined design patterns which provide the "glue" between objects or between partial models (Schütze *et al.*, 1999).

A similar approach is that of agent oriented engineering. This paradigm aims at reaching an object-aware rather than object-oriented model, by using agents, i.e. "*encapsulated computer system situated in some environment and capable of flexible, autonomous action in that environment in order to meet its design objectives*" (Wooldridge, 1997). Further work should help to understand if and how such construct might help BS model's development.

However, even if the PsiGene's approach closely resembles reality, it might lead to numerical problems. In this case, the implementation of airflow models in free spaces and in HVAC systems seems to be still an open problem (Zimmermann, 2001).

CONCLUSION

BSs and society's evolutionary nature imposes that the development process of BPS tools follows the approach described by the SER model. Furthermore, since BPS is an interdisciplinary and heterogeneous subject, a collaborative environment should be enabled and encouraged. However, we have derived that different modelling and simulating activity generates different model's levels, thus it is necessary to analyse the nature and the implication of each of them to set the right recommendations.

All these considerations lead us to underline the importance of attaining an "*enriched*" *modularity* during the model development process, as previously defined. Indeed, without this characterization, modularity has been interpreted in different ways leading to different approaches to M&S. For example, to fulfil flexibility's requirements, autonomous objects (AO) that interact when necessary (PsiGene) have been developed. To free developers from numerical problems, tools that accept differential-algebraic equations (DAE) as input for the simulation (Dymola), have been implemented. To allow new interactions, maintaining the robust and efficient previous structure (Esp-r, EnergyPlus), co-simulation (CS) has been pursued. These different approaches lead to different implications. The first approach (AO) might result in an extensive "atomization" of the calculation, leading to difficulties in reaching a convergent and reliable solution. The second approach (DAE) might result in low numerical resolution efficiency and in debugging difficulties. The third approach (CS) would be difficult to maintain and extend. Each of such approaches, while trying to reach some specific goals, does not succeed in addressing also the others.

The above-mentioned considerations aim at showing that a full-enriched modular BPS tool is not yet ready and that a *technology driven research* should be pursued to develop it. Starting from perceived problems and ambitions and seeking appropriate technological means, different concepts derived by Information Technology should be further analyzed to understand if similar formalization might lead to a renewed modelling, simulating and validating methodology. It can be concluded that, thanks to current IT facilities and to new knowledge formalization derived from other fields, a reseeding phase might lead to a more functional approach to model development and BPS tools use, more focused and oriented towards their evolutionary growth.

[14] http://wwwagz.informatik.uni-kl.de/projects/psigene.html

REFERENCES

Ben-Nakhi A, Aasem EO., 2002, Development and integration of a userfriendly validation module within whole building dynamic simulation, Energy Conversion and Management, vol. 44, 53-64.

Birta, LG. and Arbez, G., 2007, Modelling and Simulation, Exploring Dynamic System Behaviour, Springer, London, UK.

Clarke JA. 2001a. Energy simulation in building design. Second Edition, Butterworth-Heinemann, Oxford, UK.

Clarke, J.A. 2001b. Integrated Building Performance Simulation, ESRU, University of Strathclyde, Glasgow, Scotland, UK.

Crawley, D.B., Hand, J.W., Kummert, M. and Griffith, B.T. 2005. Contrasting the capabilities of building energy performance simulation programs, US DOE, Washington, DC, USA.

Dawkins, R. 1987. The Blind Watchmaker, W.W. Norton, New York, USA.

El Khoury, Z., Riederer, P., Couillaud, N., Simon, J., Raguin, M. 2005. A multizone building model for matlab/simulink environment, Proc. of IBPSA ’05, Montréal, Canada, 525-532.

Fischer, G., Giaccardi, E., Ye, Y., Sutcliffe, A. G., Mehandjiev, N. 2004. Meta-Design: A Manifesto for End-User Development, Communications of the ACM, vol. 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, p.292-298.

Jensen, S.O. 1995, Validation of building energy simulation programs: a methodology, Energy and Buildings, vol. 22, 133-144.

Kalagasidis, A.S., Weitzmann, P., Nielsen, T.R., Peuhkuri, R., Hagentoft, C.E., Rode, C. 2007, The International Building Physics Toolbox in Simulink, Energy and Buildings, vol. 39(6), 665-674.

Kelly, N. and Strachan, P. 2001. Multi-Domain Modelling Using the ESP-r System, Proc. of eSim '01, Ottawa, 253-260.

McDowell, T.P., Bradley, D.E., Thornton, J.W. and Kummert, M. 2004. Simulation synergy: expanding TRNSYS capabilities and usability, Proc. of SimBuild 2004, Boulder, USA.

Nytsch-Geusen, C., Nouidui, T., Holm, A., Haupt, W. 2005. A hygrothermal building model based on the object-oriented modeling language Modelica, Proc. of IBPSA ’05, Montréal, Canada, 867-874.

Oberkampf, W.L. & Trucano, T.G.2002, Verification and validation in computational fluids dynamics, Progress in Aerospace Sciences, vol 38, 209-272.

Robinson S. 2004, Simulation: The Practice of Model Development and Use, John Wiley & Sons, UK.

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

Schütze, M., Riegel, J.P., Zimmermann, G. 1999, PSiGene: A Pattern-Based Component Generator for Building Simulation, Theory and practice of object systems, vol. 5(2), 83-95.

Sodja, A., Zupančič, B. 2008, Some aspects of thermal and radiation flows modelling in buildings using Modelica, Proc. of 10th International Conference on Computer M&S, Cambridge, UK., 637-642.

Tang, D. 1997, Object Technology in Building Environmental Modelling, Building and Environment, vol. 32, 45-50.

Trcka, M., Wetter, M., Hensen, J. 2007. Comparison of co-simulation approaches for Building and HVAC/R System Simulation, Proc. of the Building Simulation ‘07, Beijing, China, 1418-1425.

Wang, L. & Chen, Q. 2007. Theoretical and numerical studies of coupling multizone and CFD models for building air distribution simulations, Indoor Air, vol. 17(5), 348-361.

Wetter, M.& Haugstetter, C. 2006. Modelica versus TRNSYS - A Comparison Between an Equation-Based and a Procedural Modeling Language for Building Energy Simulation, Proc. of the 2nd SimBuild Conference, Cambridge, USA.

Wetter, M. 2006. Multizone Airflow Model in Modelica, Proc. of the 5th International Modelica Conference, Vienna, Austria.

Wooldridge, M. 1997, Agent-based software engineering, IEE Proc. on Software Engineering, 144 (1), 26-37.

Xiao, D., Spitler, J.D., Rees, S.J. 2002, An Analytical Verification Test Suite for Multizone Building Fabric and Control Models in Whole Building Energy Simulation Programs, Proc. of the eSim 2002 Conference, Montreal, Canada, 260-267.

Zimmermann, G., 2001. A New Approach to Building Simulation Based on Communicating Objects, Proc. of IBPSA ‘01 Conference.

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