# PLUMES: TOWARDS A UNIFIED APPROACH TO BUILDING PHYSICAL MODELING

Sylvain Robert[1], Benoît Delinchant[2], Bruno Hilaire[3], and Yann Tanguy[1]
[1] CEA, LIST, 91191 Gif sur Yvette, France
[2] Grenoble University, G2ELab, 38402 St Martin d'Hères, France
[3]CSTB, TIDS/MOD-EVE, Sophia Antipolis, France

## ABSTRACT

In this paper, we give an overview of an on-going research work aiming at assessing the benefits that could be drawn from applying advanced software engineering techniques – namely model-driven architecture, component-based approaches and model-based system engineering – to support building life cycle tasks (especially design ones) which entail making use of digital models.

## INTRODUCTION

The construction industry is rapidly evolving, propelled forward by growing ecological concerns, increased competition, globalized markets, and always more ambitious and stringent regulatory frameworks. This has a strong impact, both on workflows and on tools. Practices focus – more than ever – on efficiency, flexibility and integration, while buildings life cycle stakeholders tend to make use of more advanced tools, especially of those relying extensively on Information and Communication Technology (ICT) (Rezgui et al, 2011). For instance, it is now common practice to rely on ICT-intensive Building Management Systems (BMS) to support operation phase activities – e.g. relating to energy monitoring and management or to facilitate maintenance tasks. The same way, building design tasks heavily rely on software tools – e.g. Computer-Aided Design (CAD) tools and physical modeling and simulation tools – to properly define, predict and optimize building behavior in all its dimensions (structural, energy, etc).

From this steady and strong (r)evolution, new needs and requirements have emerged, which call for cross-expertise in the considered business domain (the construction / building sector) and in cutting-edge software-intensive systems engineering approaches. Actually, the rush towards ICT/software support has led to a significant but somehow disordered growth of the related tools offer. As an illustration, a repository from the US Department of Energy (DoE) includes 393 software tools (as of January 2013) (DoE, 2013) dedicated to evaluating energy efficiency, renewable energy and sustainability in buildings. The issue there is that this massive offer comes with what could be termed colloquially a massive mess: from one tool to another, not only functionalities vary but also provider status and license kind, openness, physical domains targeted, software architectures, underlying languages and frameworks, mathematical modeling paradigms, data formats, etc. On the other hand, buildings are way more complex than they used to be: highly multi-physical systems - sometimes even coined *cybernetic* given the importance of ICT in nowadays buildings (Wetter, 2011), requiring accounting for various and intermixed physical phenomena (lighting, airflow, energy, mechanical, control-command,…) in an integrated way. The challenge is therefore to enable exhaustive and integrated buildings simulations, while relying on disparate and somehow natively incompatible tool: a challenge which is the research focus of the PLUMES project, funded by the French National Research Agency.

The intent of this paper is to give an overview of the objectives of this research project and to share its first outcomes. The project is ongoing (one year to go) and a lot of work is required to implement and assess the first results. Still, it is our belief that the research issues dealt with and the paths taken to address them are worth sharing with the Building Simulation research community.

The paper is structured as follows: section 2 gives an overview of the project's research scope and highlights our motivation. Section 3 and 4 introduce two major research topic addressed in PLUMES, respectively model-driven architecture applied to building simulation, and a (still prospective) research thread aiming at devising an approach for model-driven engineering of executable buildings modeling components. At last, section 5 gives some conclusions and perspectives.

## AN OVERVIEW OF THE PLUMES PROJECT

When it comes to "integrate the incompatible" (see the introduction), the keyword is usually *interoperability*, i.e. "the ability of two or more systems or components to exchange information and to use the information that has been exchanged" (IEEE, 1990). Indeed, the main issue in PLUMES is to devise an approach that could enable, in a systematic and reproducible way, information exchange between most (if not all) buildings modeling and simulation tools. In fact, in a broader sense, the aim is to contribute to improve building modeling and simulation support along the following dimensions: reusability, modularity, evolutivity, portability and, sustainability. These keywords are quite well-known in the software engineering area, where they are part of the motivation for several long-term research trends: software design and

development paradigms - e.g. component-based approaches (Szyperski, 2002) or model-based engineering (Schmidt, 2006), execution platforms - e.g. for distributed computing (OMG, 2012), methodological guidelines and design practices, standardization - e.g. of data and interfaces, or modeling language (OMG, 2011). Hence the base assumption of PLUMES, which is that, related to the aforementioned objective, consequent benefits can be gained by applying some key assets from the software engineering area to the one of building modeling and simulation. By the way, this is already somehow true: a quick tour of practices and ongoing research in the field of ICT for building design show that, like Mr Jourdain had been talking prose and never known it[1], this community tackles well-known software engineering research challenges without explicitly naming them (and sometimes without fully benefiting from the rich state of the art pertaining to them).

The interoperability issue quoted at the beginning of the section, is actually twofold - see (Howie et al, 1997) for instance: (i) *data-focused interop*erability is related to how data may be exchanged between several systems / components; (ii) *execution-focused interoperability* concerns combined execution of two or more executable pieces of software, each potentially relying on different and non-straightforwardly compatible technologies. The feedback we got in the scope of the project from field interviews shows that the current support for these two kinds of interoperability in the buildings modeling domain is still poor - this is also shown by the literature, see e.g. (Steel et al, 2012), (Radosevic et al, 2005), (Plume et al, 2007). This requires the stakeholders to implement ad-hoc strategies – dedicated data translators, one-to-one tools coupling – which tend to be effort and time-consuming, error-prone, and which by no way end up in the long-term, scalable approaches and tools that are needed.

When it comes to data-focused interoperability, one main issue is the connection between the so-called Building Information model (BIM) and the buildings physical modeling / simulation tools. The BIM covers an extensive range of assets (Succar, 2009), among which technological ones are prevailing – the main being the (still theoretical) possibility to rely on a single logical, consistent source for all information associated with the building (Howell, 2005): the BIM is basically the repository of all digital information pertaining to the building. It is now widely recognized as a cornerstone of future tools and practices in the construction industry, and many works aim at improving combined support for BIM-based collaborative design work and building energy optimization (Crosbie, 2011). Here, combining actually means connecting: the intent is to be able to use the data contained in the BIM (building 3D geometry, physical parameters like e.g. walls thermal characteristics) to seamlessly feed downstream simulation tools. As shown in the subsequent section, the issue is not as straightforward as it seems and model-driven architectures (MDA)-like approaches (OMG, 2003) could be an elegant solution to deal with it.

As mentioned in the introduction, properly modeling the breadth of physical phenomena that occur within the building usually requires relying on several tools. The issue here is therefore the one of execution-focused interoperability, which may be tackled in two main ways. The first (and most widely implemented) is the runtime coupling one: two or more environments interact at runtime and exchange data in order to simulate jointly various physical phenomena (e.g. combined air flows and energy simulation). Different coupling strategies may be differentiated (e.g. strong vs weak) (Trčka et al, 2010), and some implementations make use of a dedicated coupling middleware (Wetter et al, 2008). The second approach is the inter-environments porting one: the aim here is to enable reusing executable modeling components issued from one environment in another (non natively compatible) environment. In this matter, the state of the art focuses on simulation tasks and still lacks a generalist approach that would be enable not only to model the intermix of physical phenomena, but also to reuse and capitalize in a systematic and generic way modeling components across different application domains. As shown subsequently, PLUMES proposes a novel approach to the issue, relying on the component-based software paradigm (Szyperski, 2002). Combined with principles and tools stemming from Model-Based System Engineering (MBSE), this approach clears the path for an intensive use of physical modeling components in the whole building life cycle.

## A MDA APPROACH TO BUILDING SIMULATION

From a theoretical point of view, using the BIM to perform energy simulation is not a high-end research challenge, since it eventually boils down to mere data processing (data-focused interoperability). However, the vitality of the related research area tends on the contrary to show that the issue is far from being straightforward and remains open (Hitchcock et al, 2011).

In order to combine BIM and simulation, one can distinguish two possible approaches. The first advocates full integration of tools and data models, and is mainly implemented by software vendors which tools span a large part of building design phases (and even life cycle). The aim is to rely on a single building data model (BIM), used as a sole reference in all design tasks, including simulation. BIM authoring and simulation tools are fully integrated at the data level, but often also at the user

[1] *Le bourgeois gentilhomme*, Molière, 1670

interface level. This approach therefore treats simulation tools as *BIM-aware* tools, using the terminology introduced by A. Watson in (Watson, 2011). This strategy lacks openness and is demanding: lacks openness, because only the simulation tool(s) already embedded may be used; demanding, because any extension, e.g. to add simulation capabilities, requires a significant implementation effort. The second approach to BIM/simulation interoperability advocates "light" integration and, relies on data translation in order to generate the building model required by the simulation tool from a building model conforming to a given BIM data model. This approach is the most frequent and several works have already attempted - and to some extent, managed – to perform such connections (Cormier, 2011) (Bazjanac, 2011). Implementing such a linkage basically requires the three following assets: (i) a base data format, in which architectural building models will be expressed; (ii) software tools to process the architectural model data format and to generate corresponding input files for simulation; (iii) a simulation tool.

The choice of the data format to be used for architectural models (i) has already been discussed extensively. Most authors tend to prefer relying on an open, standard, public format than to rely on proprietary formats, and the Industry Foundation Classes (IFC) (Building Smart, 2007) appears clearly as a reference. The reason for fending off proprietary formats is quite obvious and stems from the necessity to ensure interoperability between tools from several – if not all – software vendors. The IFC is currently the only format implemented (as export / import functionalities) in most of the CAD tools, and the only open generalist format standardized by an international consortium. For more details about the IFC and its place in the BIM interoperability landscape, the reader may refer to the study performed by J. Steel and al in (Steel et al, 2012).

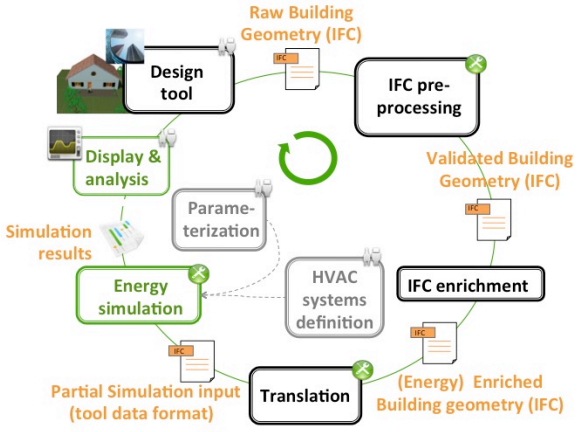


*Figure 1 Usual IFC to energy simulation process*

At this point, several difficulties arise. If we have a look at the usual BIM to simulation process sketched in Figure 1, we see that it starts with generating an IFC file from a BIM authoring tool (Design tool). This IFC file generally contains all information pertaining to the building geometry. However, the first issue is that this geometry requires some pre-processing to be performed. This pre-processing consists first in a model-checking phase to identify errors and inaccuracies in modeling – e.g. meeting and intersection of objects – and to enrich building geometry with so-called upper levels space boundaries (Bazjanac, 2010) (those are necessary to obtain a valid geometry before translation to the simulation tool and, IFC-compliant BIM authoring tools do not generate them). The result of pre-processing is a clean (containing no errors or inaccuracies) and complete (with an enriched geometry) IFC architectural file. This file has then to be enriched with additional properties pertaining to the building, that are required for energy simulation (e.g. wall thermal properties). After enrichment, the IFC file is translated to the input format of the simulation tool, and this is the point at which another difficulty is encountered: the file that is obtained from the IFC processing chain only includes information pertaining to the building, but not to the energy systems to be deployed (Heating Ventilation and Air Conditioning, HVAC). Actually, this stems from limitations of the IFC itself, since the data format does not include – at least in its current version - the necessary modeling elements, and from the poor IFC export capabilities of the CAD tools as well (Robert, 2012). This therefore requires these systems to be described in a dedicated separate format – generally the one of the targeted simulation tool.

Now, what about Model-Driven Architecture?

Well, looking at Figure 1, a reader familiar with MDE (Model-Driven Engineering) would probably have had the feeling that it reminds of a standard MDA (Model-Driven Architecture)-process (OMG, 2003): several modeling steps, together with an iterative enrichment, and in between automated data translation. The process actually starts with what could be called an application-independent IFC model that is transformed and enriched to become an energy-simulation-focused IFC model, and then again enriched and transformed to the targeted simulation tool data format. And indeed it is the assumption of the authors that such a process is an MDA-one or at least, could be one provided some improvements are achieved: upgrading the IFC, in order to enable all information (building and HVAC systems) to be contained in IFC files, improving CAD tools export facilities, implementing the required data / model transformation support tools and, providing the adequate methodological guidelines. The resulting process we are aiming at is the one described in Figure 2: the information flow is

entirely based on IFC models that are iteratively refined: a first step is to generate the IFC (both building geometry & HVAC systems) from CAD tools – with, as in the usual way, a pre-processing phase for the geometrical part; then, the IFC model undergoes a two-step enrichment: first, to issue an energy-simulation-focused IFC file (dedicated to energy simulation but independent from any specific tool); then, to issue a tool-specific IFC file that is then translated to the targeted tool input data format. The interest here is to capitalize the information contained in the simulation-focused IFC file that may potentially be reused to target other energy-simulation tool than the one initially targeted (note also that such a process may be generalized to target any kind of simulation).

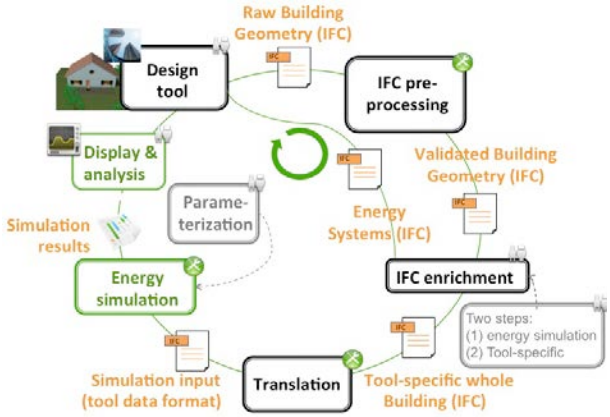


*Figure 2 MDA IFC to energy simulation process*

As a first step to put this vision into practice, our work - detailed in (Robert, 2012) - first entailed evaluating the capabilities and assessing the limitations of IFC with respect to HVAC systems description and of CAD tools with respect to IFC export. As far as IFC is concerned, the second step was to issue some propositions of extension of the standard. The rationale of the study was not to target completeness, but more to follow an iterative process: (i) choosing a set of widely used energy simulation languages and environments (namely Modelica (Fritzson, 1998), TRNSYS and EnergyPlus ); (ii) select a set of building energy systems to model (in our case: ventilation system, water heater, central heating system and electricity production); (iii) model those systems in the targeted environments; (iv) model those systems with IFC; (v) confront the models and analyze the outcomes; (vi) at last, when required, issue propositions of IFC enhancements.

A first conclusion that can be drawn is that the current support, whether it concerns IFC or CAD tools IFC export facilities, is poor. Regarding the first point, we have defined a set of extensions to the IFC that enable modeling the classes of energy systems we have analyzed. These extensions are expressed as IFC property sets. These are native IFC mechanisms that allow adding properties to existing IFC meta-elements. These mechanisms are not as powerful as those of MDA languages, like the UML (Unified Modeling Language) profiling technique (OMG, 2011), but this is the easiest and most flexible way to extend the IFC specification. As far as the latter point is concerned, our observation is that IFC export capabilities are highly variable: in particular, some tools are more powerful for geometry aspects, while others show more compliance to the IFC standard for energy systems IFC export. Our prototyping plans therefore entail relying on different CAD tools, depending on the aspect being considered. The main aim of this prototyping phase (to be achieved by mid-2013) is to be able to reify the process described in Figure 2, and to demonstrate the relevance of a clear and formalized separation of concerns between platform-independent models and platform-specific ones. This process will be assessed with two building models, one comprising only an energy system, the other a basic building coupled to an energy system. Two main energy simulation platforms will be used: TRNSYS and EnergyPlus, with the aim to demonstrate that the same energy-simulation focused IFC model may be used to target both tools.

## MODEL-BASED ENGINEERING OF EXECUTABLE MODELING COMPONENTS IN THE BUILDING DOMAIN

A wide range of activities performed within building life cycle makes use of modeling software components. These are mainly used to simulate and predict resulting building behavior to assess design choice but there is a steady tendency to enlarge their scope of use - e.g. at operation time (Pang, 2012). It is foreseeable that these will be somehow pervasive to the whole building life cycle in a near future. One issue is however that – as mentioned in the introduction – the breadth of physical modeling environments and languages translates into a massive but highly heterogeneous offer. Related to this concern, one objective in PLUMES is to devise innovative ways to engineer executable modeling components, while making the most of legacy ones. We therefore advocates an engineering process that rely on both following approaches (Delinchant, 2004): (i) the “white-box” approach, which consists in explicitly defining the intended behavior of the components thanks to a dedicated language (e.g. Modelica); (ii) The “black-box” approach, which advocates relying on legacy executable components. The components’ behavior is not explicit, only their interfaces are.

The main added value of our work lies in the definition of a software component specification (MUSE) allowing to making the most of MBSE advantages, independently from any target application and able to tackle heterogeneous

environments. As shown thereafter, in this approach, a component can be viewed as a physical component, owning physical port, independently from the targeted application. More over, to facilitate model reuse, these components may be expressed as descriptive (white-box) model and / or as executable legacy (black-box) models.

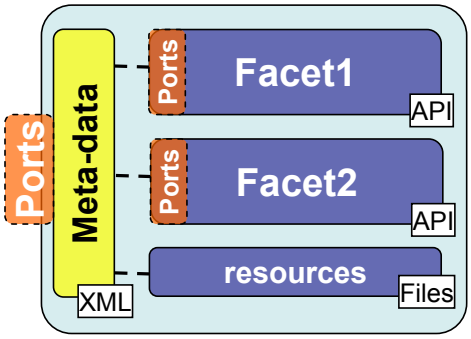


*Figure 3: Structural view of a multi-facets MUSE component*

**MUSE framework**

The MUSE[2] framework includes both conceptual and software assets. From the conceptual point of view, MUSE is a specification of a multi-application modeling component standard for the building domain. In the scope of the PLUMES projects, two reports have clarified this specification, which may be summed up as follows: (i) a MUSE component is a technology and application-agnostic software component which features a set of well-defined interfaces; (ii) To ensure a better separation of concerns, MUSE components feature layered interfaces (Figure 3). MUSE components own physical ports (the ports related to the actual physical inputs / outputs of the actual system/equipment represented by the component, e.g. a flow) and general-purpose interfaces, e.g. to query components' metadata. Then, a MUSE component may also own a set of independent, application-specific (e.g. dedicated to simulation) interfaces; (iii) in the latter case, the interface is called a *facet*. A facet gathers a consistent set of services (methods) dedicated to one particular application. A MUSE component may own several facets at the same time, and in its current state, three kinds of application have been defined: simulation, optimal sizing, and optimal control (PLUMES, 2012). By way of illustration, Figure 4 shows the methods offered by the *ODE dynamic simulation* facet. These methods enable managing the underlying model in the scope of a simulation (notifying incoming events, asking for state variables derivatives to be computed, etc); (iv) MUSE supports several underlying mathematical modeling paradigms, including continuous, discrete and hybrid systems, ordinary differential equations (ODE), differential algebraic equations (DAE); (v) MUSE components may be atomic or composite. In the latter case, the capabilities (available facets) of the composite component depend on those of the internal components.

The MUSE framework also includes a set of software modules dedicated to support MUSE components generation and import, in order to cope with the multiplicity of existing tools. MUSE plug-ins are add-ons to existing environments (e.g. TRNSYS) enabling importing MUSE components, while MUSE plug-outs are add-ons dedicated to import and execution of MUSE components within the considered environment.

To date, the MUSE approach has been assessed in the following frames: (i) To couple physical models from different simulation tools: a MUSE component was generated from an energy simulation environment (COMFIE Pleiades[3]) and then reused in other simulation environments (the Modelica environment Dymola and Matlab/Simulink) (Gaaloul et al, 2011); (ii) To reuse MUSE components generated from a simulation tool (Modelica/Dymola) in an optimal sizing tool (Verdière et al, 2012).

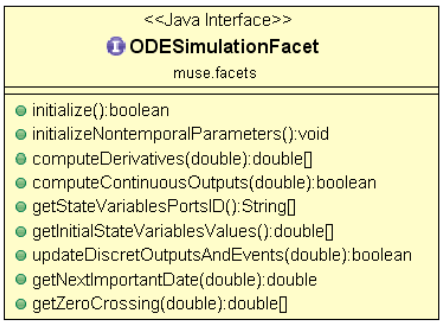


*Figure 4: MUSE facet for dynamic simulation*

In the scope of the PLUMES project, this tool support is being further enlarged and enriched, notably by providing a MUSE plug-out for the TRNSYS simulation tool and a MUSE plug-in for an energy management tool.

There are several inherent benefits to rely on a framework like MUSE. One of the main is that it enforces a clear separation of concerns between the functional view of the components (i.e. its physical ports) and the various non-functional views (the application-specific facets). Therefore, it allows to model and define compositions iteratively: at first, at the physical level without taking care about target applications, and in a second time, taking care of any needed application-specific assembly. Also, as emphasized in the second section, the ability to reuse modeling components across applications (e.g. from simulation to optimal sizing) is very beneficial. Actually, provided the framework is assessed and adopted, it could encourage the development of novel usages of modeling components as well as novel

[2] MUSE is the French acronym for « building energy systems unified models »

[3] http://www.izuba.fr/logiciel/pleiadescomfie

business models. This is actually what is intended in PLUMES: a large-scale modeling components market, including large components web libraries and a breadth of software tools supporting components customizing and assembling. This latter point, related to the tool support for components customization and assembly, is addressed in the scope of the project as a complement to the MBSE approach, as described in the next section.

**Towards a model-driven executable modeling component factory**

While MBSE (Model-Based System Engineering) is widely applied in several industrial areas (for instance in the aerospace or automotive industry)[4], it still has not – surprisingly enough – made its breakthrough in the building domain and, even the research literature includes few references. We may quote the work of P. Geyer in (Geyer, 2011), which focuses on applying the SysML[5] (a general-purpose modeling language dedicated to systems modeling, based on the UML, which includes nine diagrams covering three design concerns: requirements, structure and behavior) to building design for design optimization purposes and which shows quite nicely the resulting benefits of such an approach.. Also relevant to the considered issue, some works aim at combining the SysML with various simulation environments and languages, like e.g. Modelica (Paredis, 2010). Modelica is a widely used and renowned physical modeling language, which features object-oriented acausal systems description. It complements well SysML, by providing the necessary modeling constructs for behavioral physical systems modeling, where SysML is used to describe the structural aspects. Therefore, our choice in PLUMES is to heavily rely on these assets for all purposes related to "white-box" modeling (see the introduction of this section).

More precisely, we propose to implement an approach similar to the one described by Kerzhner and al in (Kerzhner, 2011) but tailored for the building domain. The rationale is to define a two-step modeling process, which entails (manually) defining SysML design models, and then to generate SysML models specifically dedicated to some kind of analysis. The automated translation is configured thanks to a (manually defined) model which describes how design-level components and interfaces map to analysis-level ones. We plan to target SysML/Modelica models at the analysis level. A specificity of our application scope is that it will require to define a specific library of SysML components and interfaces for the building domain, to be used at the design level (this work is on-going). As far as the modeling tool support is concerned, we rely on the Papyrus UML tool[6].

But the main novelty of this research work is that we combine this white-box SysML / Modelica Model-driven approach along with black-box executable components assembly facilities based on the MUSE framework. The aim is basically to be able to define and express MUSE component assemblies in SysML, whether they include only white-box Modelica components, black-box executable components, or a mix of both. The (idealized) process that is to be implemented is the following: having to model a complex building system (a combination of various energy systems and building components), the user first sketches a first model of the assembly relying on the dedicated design-level SysML library. Then, depending on applicable requirements, the user may switch to the SysML/Modelica analysis-level view to define from scratch the components and their behavior relying on the Modelica language. The user may also queries MUSE component databases and check for the availability of off-the-shelves components that could fit his needs. If so, the characteristics of the selected MUSE components are exhibited in the SysML modeling tool. The user is then able to connect the components, regardless from their origin, provided they offer the required facets. The resulting component assembly may then be used to generate a new, composite MUSE component. Note that this assembly process will be, in an analogy to the layered MUSE architecture, an iterative one: assemblies will be performed first at the physical, application-independent, level. Then, each specific application (simulation, optimal sizing,…) targeted will be dealt with in a separate assembly. This is particularly relevant in the scope of MBSE, which generally advocates multi-view / multi-level modeling to ensure a clear separation of concerns.

## CONCLUSION

In this paper, we have described some on-going research work performed in the scope of the PLUMES French collaborative research project. This work aims at applying several advanced approaches stemming from the software engineering area to the building area. The first outcomes suggest that some limitations of the available tool support in the building domain could be overcome this way. The paper illustrates this by showing how MDA can be the base of novel and more effective connections between the Building Information Model and simulation tools, and how model-driven engineering and component-based approaches combined could contribute to the emergence of novel usages of executable modeling components. The short-term perspectives of the project are mainly related to

[4] A sceptical reader may simply try an internet search engine query to check this assertion.

[5] OMG Systems Modeling Language, Object Management Group, http://www.omgsysml.org/, (accessed July 2012)

[6] http://www.eclipse.org/modeling/mdt/papyrus/ (accessed July 2012)

implementation and assessment of the solutions, with an end of 2013 horizon.

## ACKNOWLEDGEMENT

This work presented in this paper is performed in the scope of the PLUMES project, funded by the French National Research Agency.

## REFERENCES

Rezgui Y. and Miles J., 2011. Harvesting and managing knowledge in construction: From Theoretical Foundations to Business Applications, Spon Press, ISBN 978-0-415-54596-9.

US Department of Energy, 2013. Building Energy Software Tools Directory, http://apps1.eere.energy.gov/buildings/tools_directory/, accessed January 2013.

Wetter M., 2011. A view on future building system modeling and simulation, in "Building Performance Simulation for Design and Operation", 2011, Jan L. M. Hensen and Roberto Lamberts, Routledge, UK, ISBN: 978-0-415-47414-6

IEEE, 1990. IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries, New York, NY.

Szyperski C., 2002. Component Software: Beyond Object-Oriented Programming, 2nd ed., Addison-Wesley Professional, Boston ISBN 0-201-74572-0.

Schmidt, D.C., 2006. Model-Driven Engineering, February 2006, IEEE Computer 39 (2).

OMG, 2012. Common Object Request Broker Architecture (CORBA) 3.3, Object Management Consortium, http://www.corba.org/, accessed January 2013.

Howie C. T., Kunz J., and Law K. H., 1997. Software Interoperability, CIFE Technical report#117.

Steel J., Drogemuller R., Toth B., 2012. Model interoperability in building information modelling, Software System Modelling 11:99–109

Radosevic M., Hensen J., and Wijsman A., 2005. Implementation strategies for distributed modeling and simulation of building systems, 9th International IBPSA conference, Montréal, Canada.

Plume J., Mitchell J., 2007. Collaborative design using shared IFC building model - learning from experience, Automation in construction 16, 28-36.

Succar B. 2009. Building information modelling framework: A research and delivery foundation for industry stakeholders, Automation in Construction 18 357–375, Elsevier.

Howell I. and Batcheler B., 2005. Building Information Modeling Two Years Later –Huge Potential, Some Success and Several Limitations, www.laiserin.com/features/bim/newforma_bim.pdf, accessed January 2013.

Crosbie T., Dawood N., Dawood S., 2011. Improving the energy performance of the built environment: The potential of virtual collaborative life cycle tools, Automation in Construction 20 205–216.

Trčka M., Hensen J. L. M., and Wetter M., 2010. Co-simulation for performance prediction of integrated building and HVAC systems - an analysis of solution characteristics using a two-body system, Simulation Modelling Practice and Theory, Volume 18, Issue 7, p.957-970.

Wetter M. and Haves P., 2008. A modular building controls virtual test bed for the integration of heterogeneous systems, in proceedings of the 3rd National Conference of IBPSA-USA, Berkeley, California.

OMG, 2012. MDA Guide Version 1.0.1, Object Management Group, http://www.omg.org/cgi-bin/doc?omg/03-06-01 (accessed July 2012)

OMG, 2011. Unified Modelling Language (UML), Object Management Group, http://www.uml.org/, accessed January 2013.

Hitchcock R. J. and Wong J., 2011. Transforming IFC architectural view BIMS for energy simulation, Proceedings of Building Simulation 2011.

Watson A., 2011. Digital buildings - Challenges and opportunity, Advanced engineering informatics 25 573-581.

Cormier A., Robert S., Roger P., 2011. Towards a BIM-based service oriented platform: application to building energy performance simulation, Building Simulation 2011, Sydney, Australia.

Bazjanac V., Maile T., O'Donnell J. T., C. M. Rose M., Mrazovic N., 2011. Data environments and processing in semi-automated simulation with energyPlus, CIB W078-W102, Sophia Antipolis, France.

Building Smart, 2007. IFC2x Edition 3 Technical Corrigendum 1, Building Smart International Alliance for Interoperability.

Bazjanac, V., 2010. Space boundary requirements for modeling of building geometry for energy and other performance simulation, Proceedings of CIB W078 conference, Cairo (Egypt).

Robert S., Hilaire B., Sette P. and Soubra S., 2012. Paving the way for exhaustive and seamless BIM-based building energy simulation, proceedings of CIB W078 conference, Beirut, Lebanon.

Fritzson P. and Engelson V, 1998. Modelica - A unified object-oriented language for system modeling and simulation, in proceedings of European Conference on Object-Oriented Programming (ECOOP).

Pang X., Wetter M., Bhattacharya P., Haves P., 2012. A framework for simulation-based real-time whole building performance, Building and Environment 54 100-108

Delinchant B., Wurtz F., Magot D. and Gerbaud L., 2004. A component-based framework for the composition of simulation software modeling

electrical systems, Journal of Simulation, Society for Modeling and Simulation International, Special Issue: Component-Based Modeling and Simulation. Jul 2004; vol. 80: pp 347 – 356

PLUMES 2012. Interfaces du composant MUSE, internal PLUMES project's report, June 2012.

Gaaloul S., Delinchant B., Wurtz F. and Verdière F., 2011. Software components for dynamic building simulation, Proceedings of Building Simulation 2011 conference.

Verdière F. and al, 2012. Modelica models translation into java components for optimization and DAE solving using automatic differentiation, , Computer Modelling and Simulation (UKSim).

Geyer P., 2011. Systems modeling for building design: a method based on the Systems Modeling Language, Proceedings of the 2011 eg-ice Workshop, University of Twente, The Netherlands.

Paredis C. J. J., Bernard Y., Burckhart R. M., De Koning H.P., Fridenthal S., Fritzson P., Rouquette N. F., Schamai W., 2010. An overview of the SysML-Modelica Transformation Specification, in Proceedings of the 2010 INCOSE International Symposium, Chicago, IL.

Kerzhner AA, Jobe JM, Paredis CJJ, 2011. A formal framework for capturing knowledge to transform structural models into analysis models, Journal of Simulation 5, 202-216.