The case for a Systematic Development of Building Automation Systems

Paulo Carreira
IST and INESC-ID Lisboa
Taguspark Campus
Porto Salvo, Portugal
Email: paulo.carreira@ist.utl.pt

Vasco Amaral
CITI, Departamento de Informática,
Faculdade de Ciencias e Tecnologia
Universidade Nova de Lisboa, Portugal
Email: vasco.amaral@di.fct.unl.pt

Bruno Barroca
CITI, Departamento de Informática,
Faculdade de Ciencias e Tecnologia
Universidade Nova de Lisboa, Portugal
Email: bruno.barroca@di.fct.unl.pt

***Abstract*—It is widely acknowledged that Automated Demand Response (ADR) is a key factor for the success of the Smart Grid. However, for ADR to be realized to its full extent a new generation of Building Automation Systems (BASs) will have to be developed and the existing upgraded to leverage dynamic tariff plans and curtailment calls sent from the grid. It is well known that developing BASs is an involved matter, mostly because application software is costly to develop and maintain. In practice, this state of affairs is slowing the rise of innovative solutions for ADR.**

In recent years, software engineering practices such as Model-Driven Engineering have been proposed as a means to alleviate the problem of developing BASs, which we believe should be presented to the Smart Grid community. This article presents the main challenges of creating software for BASs (from a modelling perspective), overviews the state-of-the-art and presents an outlook on possible model-driven engineering techniques that can be effectively applied to significantly improve the development of BASs in the near future.

***Index Terms*—Model Driven Development; Domain Specific Modeling Language; Smart Grid; Building Automation**

I. Introduction

A Smart Grid (SG) is a computerized power grid that is flexible enough to coordinate consumption with production in real-time while accommodating highly intermittent power sources like renewable energy, micro-generation, as well as responding to highly variable demand posed by the introduction of plug-in electric vehicles [1]. This flexibility however, can be hard to achieve in practice. The solution seems to lie in *Demand Response* (DR), which consists of eliciting a greater participation on the demand side by exposing consumers to varying energy prices dependent both on production costs and consumption needs. The premise is that in face of more precise price signals customers will act with economic rationality [2] to the incentive and engage in *(i)* performing *demand shifting*, transferring their electric loads from high price periods to low price off-peak periods or, otherwise *(ii)* in performing *demand reduction* by cutting their loads or performing self-generation whenever it makes economic sense to do so. However, DR is being hindered by the fact that, apart from a few large industrial consumers, most consumers do not actually react to price variations [3].

A Smart Building (SB) is an automated building that employs a number of technologies aiming at improving user comfort and reduce different types of waste, in particular in reducing energy consumption. The core of a SB is the Building Automation System (BAS), a digital network of electrical devices and appliances that may *(i)* respond more adequately to the requirements of the user in a given instant, for example automatically adjusting the room temperature to the user's desire or, *(ii)* reducing of energy consumption by, e.g. switching loads based on occupancy or by performing set-point relaxation among other techniques.

Automated DR (ADR) systems assist consumers in performing DR by freeing them from the concerns of performing demand shifting and demand reduction by controlling electric loads automatically, and in price-responsive fashion [4], [3], [5]. These systems receive tariff plans and curtailment calls from the grid and send commands to the devices. Currently available solutions for ADR are still very ad-hoc in nature, i.e., they are highly dependent on the building and on its particular equipment, the type of requests sent by the grid. The response of the system and most idiosyncrasies of the automation hardware being controlled are actually hard-coded in the software. The net result are solutions that are expensive to adapt to new buildings and lack the agility to evolve to accommodate new requirements.

In order for an ADR systems to perform optimally, it needs to access detailed information about the status of the devices, information obtained from sensors regarding the status of the building, and updated information regarding consumer requirements, which can be highly unpredictable. Creating such an ADR system presents significant technical challenges require a close cooperation of the Software Engineering, Automation and Smart Grid communities. Only then it will be implemented in a wide scale.

We foresee that the solution for ADR lies in a *Building Automation System* (BAS) that can monitor the building and control electrical loads to provide an optimal and automated trade-off between two conflicting end-user requirements: user comfort and energy prices [6]. In fact, it is expected that in the coming years, BASs will play an important role in the realization of the Smart Grid [7] both in *(i)* lowering the energetic footprint of buildings—by intelligently controlling electric devices based on various factors such as occupancy or weather conditions,—and in *(ii)* promoting a more rational

use of energy by automatically scheduling devices or lower the consumption periods. The outlook on a generalised installation of BASs portrays significant social and economical impact [8], [9]. Even international organisations and governments are endorsing their adoption [10], [9], [11].

However, as we will see further, BASs are still expensive, scaring away many potential adopters and, in practice, hampering their widespread adoption [12]. The main factors that contribute to the high costs of development and maintenance of BAS are *(i)* the heterogeneity of communication protocols and devices characteristics, *(ii)* the lack of a sound tool support to deal with the complexities of the design and implementation of a BAS and *(iii)* lack of tools that empower the non-technical user to reconfigure, in some extent, the system behavior to meet his needs (i.e.. end-user requirements).

Meanwhile, the scientific community in software engineering research, has found several solutions for dealing with the inherent complexity of software systems and their development. Ranging from *agile development* processes to *model-driven development* approaches, these solutions aim to tackle the complexity of heterogeneous software systems (where we can include also BASs) by capturing the recurring patterns in system development for a particular domain in order to systematize and streamline the software's development maintenance and configuration activities. These patterns can be expressed and conceptually captured by means of Domain Specific Languages (DSLs) which have the most appropriate level of abstraction. Using DSLs, not only the system designers can create affordable high quality BASs, but also the end-users can configure their systems in a flexible and usable way.

In this paper we present on overview of the main challenges of creating software for building automation and present an overview of the recent developments in the field of Model-Driven Engineering for creating BASs. Our manuscript is organized as follows: in Section II we present the background of the Home and Building Automation domains and present their main shortcoming and challenges in terms of modeling. Then, in Section III, we overview the related work in terms of MDE for solutions for creating BASs. In Section IV, we describe a set of models and their respective relationships, resulting from a preliminary analysis on the BAS's development domain. Finally, in Section V we present a short summary and main conclusions.

II. Background

A. Heterogeneity

Building Automation (BA) is a domain where many different industry standards have been defined for device communication such as: BACNet, EIB / KNX, LON or Modbus, in addition to many other proprietary solutions. Unfortunately, there has been few consensus on reaching an unique standard on this domain. Also, despite the fact that these platforms display similar functionalities, they are largely incompatible and expensive to interface with each other [13], [14]. Integration of different standards is even discouraged to favor customer lock-in [15] (due to commercial reasons). It turns out that the concept of a Smart Building is only fully realized to the extent of the integration of its devices and subsystems. Integrating devices of different platforms is frequently desirable since a device of one platform can be better suited and cheaper for a particular task. Without appropriate tools, however, this integration calls for a manifold of detailed knowledge of hardware and vendor-specific configuration tools that is not cost effective for the supplier and thus, in practice, integration is either very limited or not performed at all. Many buildings already feature a number of sub-systems that can be automatically controlled, such as HVAC, lighting, safety equipment, among others. Consider the example of using data from the building security system to turn off lights and reduce cooling when occupants are not present. Usually these sub-systems were built and deployed at different points in time during the life of the building, by distinct suppliers and using different standards leading to an heterogeneous BAS. Currently, each of these sub-systems often operate isolated, because each supplier is specialized in one technology, while lacking the technical knowledge, and the commercial motivation to be of assistance on any another technology. Conceivably, if a reusable and technology neutral description of BASs existed, as buildings undergo changes (extensions and reconfigurations), any supplier could (virtually) perform maintenance to the existing BAS or integrate another solution in the existing BAS, leading to an effective cost reduction. However, the current industry practice is still far from this vision mainly because there is still no agreed standard to describe the BAS in a technology-neutral way.

B. Lack of sophisticated behavior

Another obstacle to the development and maintenance of BASs is that implementing new behavior, i.e. new control logic to coordinate BAS's devices, is quite challenging [16], [17]. The development of new application software for embedded devices as long been known to require significant technical expertise. In BAS, the control logic ends up scattered through multiple devices, specified as embeddable code using more than one language, which along with proprietary configuration files, becomes often too complex to manage and understand. As a result of this poor engineering practice, changes are costly and error-prone, and to be economically sound, many BASs' suppliers are forced to clip many essential features in order to cope with their intrinsic complexity. Most automated buildings arguably perform any kind of truly intelligent control beyond set-point maintenance, daylight control and schedule based actuation. Moreover, newly added functionalities often interfere with each other, or display conflicting requirements (e.g. saving energy conflicts with maximizing user comfort or ensuring the correctness of control logic), and hence of the system as whole, becomes challenging. Finally, if we need to integrate existing BASs with other facilities and enterprise management software packages, the problem becomes unmanageable, given the breath and complexity of the behaviour that those BAS are required to display in this situation.

C. Insufficient user empowerment

In BASs, each sensor or actuator contains an electronic interface that is responsible for, respectively, sensing electric signals and sending the appropriate messages into the network or, conversely reading data from the network and generating the appropriate electric signals. Each of these network interfaces is known as a *node*. Nodes send messages into the network that are received by other nodes, thus forming a distributed network of devices running embedded software applications.

In general, BAS do not considerably improve the comfort of the building occupants, or live up to the expectations of increased flexibility. BASs allow few customizations beyond set-point adjustment, scenario configurations and definition of schedules. Once the system is installed and commissioned, it presents a fixed set of functionalities to the occupants of that space. Any customization of both the structure (new devices) and their behaviour, requires the intervention of the BAS supplier to recommission the whole BAS configuration. In practice, both occupants and the building owner are unable to effectively take full control of their BAS.

Moreover, once installed and commissioned, the cost of upgrade of a BAS is often very high. These systems cannot be upgraded and easily connected to one another (at least in the same way as the other modern consumer-level electronic devices). Changing the behavior of the system or upgrading it by adding new devices requires devices to be re-commissioned. Re-commissioning can only be undertaken by a specialist with the aid of tools specific to each fieldbus technology. In current practice, there is no simple solution for the owner of a Smart Home to bring in a new smart electronic device and integrate it with the existing automation system.

III. Current Models and Solutions

Streamlining the development of Building Automation Systems is a complex and pervasive issue [16] that has been targeted by computer scientists as well as industry consortia through standards, development tools and languages. However, as we will try to make clear, these initiatives have targeted specific problems, resulted in a number of mostly isolated solutions. The reasons for this state of affairs are multiple. On the one hand, companies that develop automation hardware see few economic benefits in comming toghether to develop a unified methodology. On the other hand, and perhaps more importantly, creating all-encompassing, principled and sound methodologies for developing BASs have been hampered by the highly heterogeneous nature of the problem, which requires appropriate conceptual tools.

One promising solution to solve this issue lies in the concept of Domain Specific Modeling (DSM), which as been gaining momentum in the Computer Science community [18]. Domain models, or *models*, cater several benefits such as enabling reuse—we can build libraries of models; enabling rapid prototyping—components can be automatically built from models (even from incomplete models); models are at higher level of abstraction and are easier to appropriate by a non-expert user; they are nearer to requirements and thus we can adapt faster to requirements. Models can be analyzed for consistency and the feed-back can be given to the user in a meaningful fashion.

A. Dealing with Software complexity: Models vs. Complexity

In its essence, DSM's goal is to tackle software/information and hardware/physical systems' complexity by modelling everything explicitly, at the most appropriate level(s) of abstraction using the most appropriate formalism(s) and simulating or deriving the target systems automatically from those models. With appropriate and simple (but not simpler) *descriptions* of the intended system, it's complexity can virtually be reduced to a manageable level, hence unquestionably enhancing the domain users' productivity. In order to produce these description models, it is essential to provide precise (but still usable) modelling formalisms so that they can be used to concisely expresse both the problems and the solutions using by using a specialized language, rendering the appropriate terms of the application domain, known as a *Domain Specific Language* (DSLs).

These simple high-level models expressed using these languages can be automatically translated into lower-level models that represent the same original information albeit with more detail. This automatic derivation (or approximation) is again specified as a transformation model. In our case, lower-level models are typically represented using languages that are supported by the target fieldbus implementation platforms.

Recently, the introduction of Language Workbenches has powered the rapid development of these kinds of DSLs by providing a relatively inexpensive way of generating editors and compiler tools for them. The *language workbench* is a term coined in 2005 by Fowler to describe a new class of software development tool, planned to build software through a rich environment of multiple, integrated DSLs [1]. The underpinning idea of language workbenches is to build a software system by identifying the various areas of that system and using (perhaps defining and building) a DSL for each area. Remarkably, language workbenches also apply DSM by providing appropriate languages, called *meta-models*, in order to enable the software language engineers to describe the different facets of their new DSLs, and integrate them together into a coherent whole. According to Fowler, to define DSLs, the workbench supports:

- Defining the schema for a Semantic Model for the language;
- Defining one or more rich editing environments for the language;
- Defining the behavioural semantics for the language, through some mix of interpretation and code generation;

In fact, language workbenches are what make these tools prominent. Developers have been creating DSLs for decades, but without dedicated comprehensive languages and tools, they

[1] http://martinfowler.com/bliki/LanguageWorkbench.html

could not break the inherent complexity of building a new language. Language workbenches aim at taking this further: some language workbenches support editing in regular text, others use projection editors that support structured text that does not need parsing, or graphical diagrams, or both. Moreover, DSM combined with development methodologies driven by models (MDD) has been successfully applied to several industrial level projects, such as IP Telephony and call processing, insurance products, or mobile phones [18]. Reasonably the same concepts can be applied to the development of Building Automation Systems.

B. Applications of Domain Specific Modeling in Building Automation

We were looking for DSM solutions that can be applied to tackle the particular problems identified in BAS's development: heterogeneity, advanced behaviour, and user empowerment. Actually, there is some evidence that the BA and other related domains are gradually putting DSM into practice. For instance, a domain that draws some similarities with Building Automation is Industrial Automation (IA). DSM techniques are used more thoroughly in IA than in BA. In IA, Programmable Logic Controllers are programed using a graphical DSL (called Ladder logic) for expressing manufacturing control logic. Also here, the goal of using DSLs is to isolate engineers from the low level details.

Another example are the languages designed for the Automation Control domain which are quite expressible: they target aspects such as distribution (dSL [19]) and hierarchical abstraction behaviour specifications (Monaco [20]). For instance Monaco, allows the specification of sequences of events, handling of asynchronous events, exceptions and errors, support writing reliable and correct programs that are easy to maintain and understand.

DSLs can provide simple, intuitive descriptions of the system. Several DSLs have been proposed for tackling different aspects of home and building automation. Namely DomoML [21], HomeML for exchanging data about device placement [22], HomeTL for modeling sequences of actions in a home environment [23], HAAIS-DSL for decoupling fiedlbus design from configuration and commissioning for the development of home automation system [24]. DSL approaches have also been used in assisted design, i.e., engineering environments that store recurring design patterns and accelerate the design phase by suggesting the appropriate device configurations to fulfill a particular requirement [25].

Nevertheless, given this state-of-the-art in DSM, we say that BAS are still developed using low level languages and tools, and lacking a principled development methodology (such as MDD) which can enable a technology independent implementation of the systems' requirements. A pertinent example is the mapping of an on/off output port of a switch sensor of some platform to the compatible on/off input port on a lamp actuator of another platform [26], [27], [17]. In this case, it is not even possible to specify a mapping to a push switch sensor which only provides the push event. In our view, this notion of compatibility is too restrictive and oversimplified with respect to real world applications. There should a be a way to create a new adapter that maps push events to on/off events. The logic of the adapter should be (ideally) platform neutral, in order to be applied in multiple platforms (e.g. by translating switch events into on/off events); or composed with other adapters, in order to create richer and more flexible adapters. However, there were recently successfull attempts to introduce MDD in the development of BAS [28], that are able to explore the modeled similarities between different BAS, in order to reuse tests. These promissing attempts clearly demonstrate the potential of both DSM and MDD to reduce BAS development's complexity.

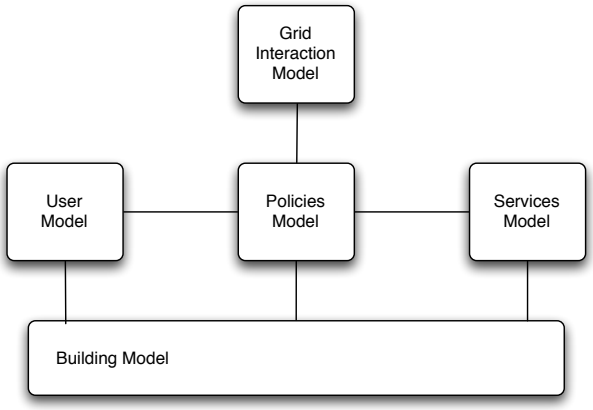

Figure 1. Requirements models uses distinct domain models to capture the high-level behavior of the BAS. At the top, the *services model* describes the control services available; then, the *user model* describes the organization of the occupants, and finally the *grid interaction model*, describes how grid pricing is mapped into energy saving requirements. The *policy model*, at the center, brings together all previous models. At the bottom, a *building model* specifies the aspects related to the building's structure and space characteristics; concepts underlying the upper models.

IV. Requirements and Implementation Models

In this section we overview how the different models can be used in the development of a BAS. In practice, BASs can developed using different models, while using a multi-paradigm approach (also called *viewpoints* or *perspectives*), enabling a collaborative way of development, by promoting separation of concerns. Each paradigm model can be created by different users, with different experts using different tools. And then integrated. The models can be populated by different domain experts. Part of the models can even be synthesized from information provided by third-party systems.

We foresee a twofold approach that uses domain models *(i)* at the *requirements level*, and *(ii)* at the *implementation level* to assist the development and maintenance of a particular BAS. A mapping has to be maintained a between these two levels of models.

This separation of concerns into two levels has several advantages. Firstly, requirement models are clearer by only specifying what is the intended structure and behaviour of the BAS, i.e., we do not specify how the BAS will actually work. As a result, the requirements models are no

longer embedded in the implementation, and therefore they can be used for complex analysis such as completeness, consistency, and behavioural checking independently of the implementation. An implementation model can be further replaced and evolve without the danger of losing important building related knowledge (frequently embedded in the implementation/configuration of the system itself). Secondly, these requirements specifications are developed at a higher level of abstraction, nearer to the BAS's end-users, which means that some models may be developed by an expert who may have significant knowledge of the intended system regarding the interactions of the building and its occupants and services but, nevertheless, has rather little knowledge about the technical details of the BAS. Moreover, the requirements specification can be used as a global model to enable the creation of components that enable applications to interoperate.

As the building evolves and new devices are eventually installed, these models should be updated accordingly. Once formalized in the models that we propose, requirements will display dependency relationships among them together with the implementation models. This aspect is particularly beneficial since it allows to perform impact analysis in an automated fashion, and enables the creation of tools to perform *what-if* analysis. For example, which spaces will be affected, or which occupants will be affected, if a service must be changed; or if a group of occupants has to be merged with another, what preferences have to be revised; or even which spaces would be affected if the system forced a certain energy consumption profile on them. Finally, this model of requirements can be used to extract useful quantitative metrics for planing purposes.

A. Requirements models

Requirements make explicit the policies and preferences of groups and individual occupants concerning the how the services underlying each task can be downgraded to meet a given energy saving target. They provide a statement (or a specification of) what the BAS must do (liveness) and prevent happening (safety). This specification has to be implemented by the BAS. Some of the entities defined on each of the presented requirement models have elements that are glued together in the definition of policies in the policies model (Figure 1). A *task* is defined by the occupancy of a space during a given time interval and requiring a set of services on that space during its duration. The BAS may use knowledge about policies and preferences to optimize the building to cater the services of each task with the lowest amount of energy. In order to define these preferences, another four models must be defined: the building model, that defines the spaces, along with their relationships; the services model which defines what services are available and how these services can be downgraded to use less energy; the user model that defines profiles and relationships of individual and collective occupants of the space and finally, the grid interaction model, which describes how the specific grid pricing policies map to the energy saving requirements. Next, we describe each of these models in greater detail.

1) Building Model: The building model represents the building's spaces, their characteristics and relationships. In practice, this model provides a hierarchical decomposition of the building's physical structure which can be used by several building automation applications that are able to reason about the building. Information regarding physical decomposition can be used for different purposes such as guiding software applications in the aggregation of data according to the physical structure or to predict which spaces can be affected as a consequence of activating a service on a contiguous space. Building models also feature information regarding materials, room orientation and electrical installation. The electrical installation displays the information regarding placement of points of control and devices such as luminaries. Building information is useful in the commissioning process of the fieldbus network, for example, to determine the configuration parameters of the daylight harvesting controller, and also to create sophisticated controls for HVAC that take into account building envelope parameters.

Querying the Building Models for the structure of the building is a basic functionality required by most tools used in building automation such as fieldbus commissioning applications, Centralised Technical Management tools and Energy Management Systems. Yet there is no standard to exchange building modeling information for these tools. **Facilities Management**, as a professional activity, has also been facing the problem of interoperability of different software tools that support their activity such as Computerized Maintenance Management and Computer-Aided Facilities Management tools [29]. The current state of industry practice in this are is to adopt the Building Information Model, which was originally developed for Architecture, Engineering and Construction (**AEC**) as a standard [30].

2) Services Model: The services model specifies the *services* that can be activated in a given space of the building during a possibly long period of time. Under this perspective, services are defined as influencing and being influenced by the *Environment Physical variables*, which can be related to light (luminance, glare), or be related to air (temperature, freshness), etc. This specification is usually maintained by the **System Engineers** during the construction of the BAS.

Another concern expressed in these service models is the notion of *goals* that are to be achieved in a particular BAS by means of *tasks* defined at the building level. Those can be understood as having certain requirements in terms of these services. Variants of the same service are defined that have different quality and energy consumption requirements. The relationship among variants of the same service is used to determine how a service can be downgraded to save energy.

There are several approaches in Requirements Engineering (RE) for obtaining requirements according to some specific decomposition criteria (e.g., viewpoints or goals) that we foresee as potentially interesting to be used as a base for a modeling languages at the Services level. Goal-Oriented Requirements Engineering (GORE) like KAOS[31], I*[32] use goals and tasks to precisely elicit, develop, structure, specify,

analyze and negotiate requirements.

3) User Model: The user model describes the organic of the occupants and their relationships as well as information regarding their preferences and space usage constraints. The organic of the occupants (or user profiles) describes how groups or individuals relate to each other. These relationships should be made explicit because they often entail sharing and inheritance of preferences and constraints. For example, a department may have a set of policies defined as constraints that apply to all individuals belonging to a given user profile. Individuals will inherit the preferences defined for the department. However, certain individuals can be part of a special profile group to which we can override the policies defined in higher profile levels with new policies. Profile specifications will also be composed of a set of task definitions which characterises what are the most common tasks of a given profile, and what are the expected building resources required by them. Finally, from a BAS user model we should be able to extract an occupancy model that details what are the expected occupancy plan for a given room in the building, based not only on recorded statistical information, but also on the users' preferences expressed by means of users' agendas etc. The user model can play an active role in the dynamic activation of energy management policies. The user profile schemas are, of course, dependent on a particular organization, and therefore it should be first defined by either the human resources director or by the Facility Manager. During regular operation of the BAS, each **Occupant** will dynamically change this model according to its needs or preferences.

4) Grid Interaction Model: The grid interaction model specifies how particular tariff plans and response agreements are mapped into energy saving requirements. This model describes a negotiation business process for energy bidding. This model will be changed and adjusted as the Utility operator offers new tariff plans on the market. In generic terms, the output of this process is a schedule of intervals in the form $\langle t_i, t_f, r \rangle$, that associate an energy reduction requirement r with the interval denoted by the initial and final instants t_i and t_f, respectively. The energy reduction requirement can be a discrete value, commonly understood by all actors, such as NONE meaning unconstrained or regular operation, MODERATE for savings with marginal disruption of business or occupant comfort, MAX for a scenario of maximum achievable savings possibly implying business changes or occupant discomfort, and CRITICAL to be issued only on situations of catastrophe that require power usage to be reduced to an absolute minimum to prevent the shutdown of the power network.

5) Policies Model: The policies model integrates the previous four models into describing how services can be downgraded to meet a given energy saving requirement. This model consists of a set of declarative rules that denote degradation constraints defined by the Facility Manager. By degradation constraints we mean the preferences of the users when occupying the space to perform a given task (which requires certain services to be available for that space) during a given time interval (e.g. $PolicyRule = User_{task} \times Room_{id} \times Interval_{time} \times Grid_{Mode} \Longrightarrow Service$) or define what are the services available for each user's task (e.g $PolicyRule = User_{task} \times Service$), thus connecting user's tasks with service's tasks. During the regular operation of the BAS, these policy rules (along with the users' occupancy information) can be used to determine what commands should be sent to each control system. In the background, these constraints can be used to determine what services can be downgraded to meet a given energy saving target on given space during a given period. With these rules we can derive traceability links between the different models in order to perform impact analysis or to verify the consistency between the user's comfort requirements and the building's energy consumption requirements.

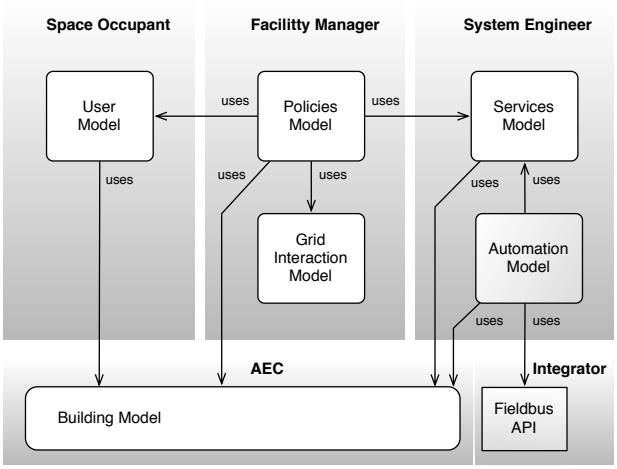

Figure 2. Domain Models required to develop BASs—actors involved with their responsibilities and concept usage relations between models.—Requirements models are rendered in white and implementation models are rendered in gray. The Fieldbus API refers to a standard that is displayed in the diagram for completeness sake.

Moreover, the policies model can be used to perform simulations by applying the same rule on modified space, user models or energy saving requirements; or to predict the services that will be needed for each space. This forecast information can be used to either take advantage of demand-shifting or to explore joint optimization.

B. Implementation Models

Implementation models are those that will assist in the development of the BAS. For better understanding the role of the different models involved and their use in the MDD framework, we envisage three main actors corresponding to three phases in the modeling activities. The first, phase consisting of designing the system objects, control of devices, their composition and orchestration as well definition of the services to offer by the BA and building model (though this last one has crosscutting information that is accessed by means of a specific view to each actor), are all specified the by the System Engineer; the second phase of the BA's control policies and GRID interaction is specified by the Facility Manager; and

finally, the individual's preferences and profiles, declared by the Occupant.

Following the MDD spirit, these models are themselves artifacts that should dynamically facilitate the creation of DSLs to constrain the space and services for both the User and the Facility manager.

1) Automation Model: An automation model describes how specific devices can be orchestrated to achieve the desired behaviors. Also, as some devices already have non-programmed implicit behavior, the automation model will allow to encapsulate but still describe their relation with the world.

Instances of automation models will be used to implement services specified in the Services models. Therefore the Services declared at the Service Model level can be considered abstractions focusing on the goals (the *what*), while the automation will be focused on the objects and their behavior (the *how*). Altough this level is already close to the device objects description, it still is at the modeling level and not at the code implementation level, often created with a *General Purpose* language. Therefore these models keep abstract notations and constraints in terms of the Building automation Domain.

We can find some examples of modeling solutions that make use of appropriate generators, for example, in the field of High Energy Experimental Physics Instrumentation, for control of elements in the detector machines. In CSML [33] we can find a solution based on the definition of specific stereotypes embedded in the UML models.

Specifications created with automation models are largely independent of the fieldbus technology, since they abstract the heterogeneity involved focusing on the common concepts, and thus can be re-deployed without being changed. However, as expected, these automation models will be dependent (or make use) of the Building model.

2) Fieldbus API: As already mentioned, current BAS are built using a wide range of technological solutions based in field bus architectures that are very different in nature (e.g different communication standards/protocols). This raises a rather puzzling technological challenge to come up with a BAS solution standard which is able to effectively deal with the heterogeneity nature of BAS implementations, while overcoming the customer lock-in legacy effect that has been created by the incumbent BAS' providers. Therefore, we foresee the definition of an open standard for these kinds of solutions, materialized in the form of an unified field bus API which is able to interface and connect different BAS architectures and their particular devices. During the BAS' implementation phase, the BASs' integrators should be able to couple together existing solutions from several different BASs' vendors with minimum effort. Following a MDD approach, this API can be the base target of model transformations that will automatically translate specified automation models (by the system engineer) into API calls expressed in a given general purpose language.

We feel that open nature of this devised API is the key for the successfull wide-adoption of BAS (in terms of cost), which in turn have a direct impact on the sucess of the SG. For example, OPC[2] and EPICS[3] are, respectively, industrial and academic projects aiming at providing open infrastructures for accessing a range of different hardware. Although most of this work stems from industrial automation, it is interesting under the perspective of BASs because most underlying ideas can be adapted to create an intermediate Open Standard for building automation to which all integrators (or vendors) should be compliant with.

C. Relations between models

As we show in Figure 2 we present the relations between all the involved models, in the sense that the entities on each source model needs to reference the used target model. The following are some illustrating examples:

- User Model uses Building Model: In order to specify a user profile, we should be able to indicate the expected occupancy pattern according to the building map.
- Automation Model uses Building Model and Service Model: The Automation model will reference the services declared in the Service Model, expressing that a given component or orchestration in the Automation model can implement a declared service in the Service Model, within a given building location.
- Services Model uses Building Model: Each service definition in the Service Model describes on which building location the specified service will be active.
- Policies Model uses Building Model, User Model, Grid Interaction Model and Services Model: The rules expressed in the Policies can ultimately reference all the existing requirement models.

However, having the above described relations, we can foresee the realization of all the uses relations depicted in Figure 2 in many different ways. In a MDD perspective, these relations can ultimately be realized by means of model transformations which can ultimately be bi-directional. In fact, any of the proposed models are artifacts than can be either designed by the respective actors, or automatically generated by means of model transformations. Having a model transformation for specifying the automated generation of models can be useful to optimize and automatize further evolutions of the system in a controlled fashion.

In one direction, transformations can be used to realize the link between both the specified requirement models (i.e it includes the Building Model and configurations from the Policy and Service Models) and the implementation models (i.e.. Automation Model). Automated generation by means of model transformations can also link the Automation model with the underlying API—if we have a metamodel of the general purpose language on which the required API calls are expressed. In the opposite direction, changes in the existing implementation models can (for instance) be automatically transformed into new constraints updating the previous existing requirement models. That is the case when new equipment

[2]OLE for Process Control: http://www.opcfoundation.org

[3]Experimental Physics and Industrial Control System, Argonne National Laboratory: http://www.aps.anl.gov/epics

introduces new behaviour in the underlying automation model, and this behaviour is in turn translated into new services or even new policy rules which then have to be checked for consistency against the existing user's defined policy rules.

V. Conclusion

This paper focuses on Building Automation Systems and, based on the current state-of-the-art, highlights what we believe to be the major reasons for their limited success so far. BAS' application software is costly to develop and maintain. This fact becomes more clear given the heterogeneity of solutions in BAS, the multiplicity of standards and proprietary solutions, and the continuous need for user empowerment during the system's life cycle.

We argue that a possible approach is to bring Software Engineering techniques like MDD and DSM approaches to this domain. These techniques have been successful in other domains to tackle the development of systems with similar complexity.

Finally, we indicated what is the required information to develop BAS solutions while tackling all of the identified problems. Also we gave a deeper insight on the nature of this information in terms of requirement and implementation models, and established a relation between them by means of model transformations. With appropriate modeling and metamodeling tools, these models can be soundly integrated in practice in a future systematic and integrated framework.

VI. Acknowledgments

This work was developed in the context of the following research institutions: INESC-ID and CITI fund PEst-OE/EEI/UI0527/2011 Centro de Informática e Tecnologias da Informação (CITI/FCT/UNL) - 2011-2012

References

[1] A. Ipakchi and F. Albuyeh, "Grid of the future," *IEEE Power and Energy Magazine*, vol. 7, no. 2, pp. 52–62, 2009.

[2] A. K. David and Y. Z. Li, "Consumer rationality assumptions in the real time pricing of electricity," *IEE Proceedings-Generation, Transmission and Distribution*, vol. 139, no. 4, pp. 315–322, 1992.

[3] N. Hopper, C. Goldman, and B. Neenan, "Demand Response from Day-Ahead Hourly Pricing for Large Customers," *The Electricity Journal*, vol. 19, no. 3, pp. 52–63, 2006.

[4] S. Braithwaite, *Residential TOU Price Responce in the Presence on Interactive Communication Equipment*. Springer, 2000, ch. 22, pp. 359–373.

[5] A. Faruqui and S. Sergici, "Household response to dynamic pricing of electricity – a survey of the empirical evidence," SSRN eLibrary Working Paper Series, 2010.

[6] S. Kilicote and M. A. Piette, "Advanced Control Technologies And Strategies Linking Demand Response And Energy Efficiency," in *Procs of Int'l Conf. for Evaluation of Building Operations (ICEBO '05)*, 2005.

[7] A. S. Massoud and B. Wollenberg, "Toward a Smart Grid: Power delivery for the 21st century," *IEEE Power and Energy Magazine*, vol. 3, no. 5, pp. 34–41, 2005.

[8] IEA, "The Power to Choose – Demand Response in Liberalised Electricity Markets," OECD International Energy Agency, Tech. Rep., 2003.

[9] B. Bach, D. Wilhelmer, and P. Palensky, "Smart buildings, smart cities and governing innovation in the new millennium," 2010.

[10] W. B. C. for Sustainable Development, "Transforming the Market: Energy Efficiency in Buildings," WBCSD, Tech. Rep., 2009.

[11] C. Goldman, M. Reid, R. Levy, and A. Silverstein, "Coordination of Energy Efficiency and Demand Response," Berkeley National Laboratory, Tech. Rep., Jan. 2010.

[12] C. Reinisch, M. J. Kofler, F. Iglesias, and W. Kastner, "Thinkhome energy efficiency in future smart homes," *EURASIP Journal on Embedded Systems*, 2011.

[13] W. Kastner, G. Neugschwandtner, S. Soucek, and H. Newmann, "Communication systems for building automation and control," *Proceedings of the IEEE*, vol. 93, no. 6, pp. 1178–1203, Jun. 2005.

[14] K. Wacks, "Home systems standards: achievements and challenges," *IEEE Communications Magazine*, vol. 40, no. 4, pp. 152–159, Apr. 2002.

[15] D. Snoonian, "Smart buildings," *IEEE Spectrum*, vol. 40, no. 8, pp. 18–23, Aug. 2003.

[16] F. Praus, W. Granzer, and W. Kastner, "Enhanced Control Application Development in Building Automation," in *7th IEEE International Conference on Industrial Informatics*, Jun. 2009, pp. 390–395.

[17] M. Jimenez, F. M. Rosique, P. Sanchez, B. Alvarez, and A. Iborra, "Habitation: A Domain-Specific Language for Home Automation," *IEEE Software*, vol. 26, no. 4, pp. 30–38, 2009.

[18] S. Kelly and J.-P. Tolvanen, *Domain-Specific Modeling*. Wiley-IEEE Computer Society Press, March 2008.

[19] B. D. Wachter, T. Massart, and C. Meuter, "dSL: An Environment with Automatic Code Distribution for Industrial Control Systems," in *Principles of Distributed Systems*, ser. LNCS, M. Papatriantafilou and P. Hunel, Eds. Springer-Verlag, 2004, vol. 3144, pp. 226–233.

[20] H. Prähofer, D. Hurnaus, C. Wirth, and H. Mössenböck, "Monaco: A DSL Approach for Programming Automation Systems," in *Procs. of the Software Engineering Conference (SE'2008)*, Feb. 2008, pp. 242–256.

[21] V. Miori, L. Tarrini, M. Manca, and G. Tolomei, "An Open Standard Solution for Domotic Interoperability," *IEEE Transactions on Consumer Electronics*, vol. 52, no. 1, pp. 97–103, Feb. 2006.

[22] C. D. Nugent, D. D. Finlay, R. J. Davies, H. Y. Wang, H. Zheng, J. Hallberg, K. Synnes, and M. D. Mulvenna, "homeML – An Open Standard for the Exchange of Data Within Smart Environments," in *Pervasive Computing for Quality of Life Enhancement*, ser. LNCS, T. Okadome, T. Yamazaki, and M. Makhtari, Eds., vol. 4541. Springer-Verlag, 2008, pp. 121–129.

[23] A. Rugnone, E. Vicario, C. Nugent, M. Donnelly, D. Craig, C. Paggetti, and E. Tamburini, "HomeTL: A visual formalism, based on temporal logic, for the design of home based care," in *IEEE International Conference on Automation Science and Engineering*, 2007, pp. 747–752.

[24] P. J. Clemente, J. M. Conejero, J. Hernández, and L. Sánchez, "HAAIS-DSL: DSL to develop Home Automation and Ambient Intelligence Systems," *European Conference on Computer Systems*, 2009.

[25] H. Dibowski, J. Ploennigs, and K. Kabitzsch, "Automated Design of Building Automation Systems," *IEEE Transactions on Industrial Electronics*, vol. 57, no. 11, pp. 3606–3613, 2010.

[26] C. Reinisch, W. Granzer, F. Praus, and W. Kastner, "Integration of heterogeneous building automation systems using ontologies," in *34th Annual Conf. of IEEE on Industrial Electronics, 2008*, 2008, pp. 2736–2741.

[27] D. Bonino, E. Castellina, F. Corno, and M. Liu, "Technology independent interoperation of domotic devices through rules," in *IEEE 13th International Symposium on Consumer Electronics*, May 2009, pp. 971–975.

[28] J. M. Conejero, P. J. Clemente, R. Rodríguez-Echeverría, J. Hernández, and F. Sánchez-Figueroa, "A model-driven approach for reusing tests in smart home systems," *Personal and Ubiquitous Computing*, vol. 15, no. 4, pp. 317–327, 2011.

[29] A. Lewis and D. Riley, "Defining High Performance Buildings for Operations and Maintenance," *International Journal of Facility Management*, 2010.

[30] C. Eastman, P. Teicholz, R. Sacks, and K. Liston, *BIM Handbook: A Guide to Building Information Modeling for Owners, Managers, Designers, Engineers and Contractors*. John Wiley & Sons, 2008.

[31] A. Dardenne, A. van Lamsweerde, and S. Fickas, "Goal-directed requirements acquisition," in *Science of Computer Programming*, 1993, pp. 3–50.

[32] E. S. K. Yu, "Social modeling and i*," in *Conceptual Modeling: Foundations and Applications*, 2009, pp. 99–121.

[33] K. Zagar, M. Plesko, M. Sekoranja, G. Tkacik, , and A. Vodovnik, "The control system modeling language," in *Procs. of the 8th Int'l Conf. on Accelerators and Large Experimental Physics Control Systems*, 2001.