Service Oriented Development of Building Energy Management Systems
An Architecture Blueprint

Rodolfo Santos[1] and Paulo Carreira[2]
[1]*Department of Computer Science and Engineering, Technical University of Lisbon, Lisbon, Portugal*
[2]*INESC-ID, Lisbon, Portugal*

Keywords: Service-oriented Architecture, Building Automation, Energy Management.

Abstract: Building automation and energy management systems increase the value of buildings by making them more comfortable, productive, and energy efficient. However, the hardware and software technologies underlying actual systems are largely monolithic and incompatible with each other hampering extensibility and favouring consumer lock-in. In order to overcome this problem we propose to create an middleware architecture, that abstract the heterogeneity of automation systems and provide sophisticated functionality to applications for intelligent control and energy management. Our middleware, will expose services that allows to abstract the numerous protocols and technical requirements associated with each manufacturer to applications, allowing developers to apply all their effort designing algorithms and innovative techniques that maximize the energy efficiency, without worrying about the technical details of each existing system. The concept will be applied in a real-world setting with the implementation of a service-oriented architecture prototype for energy management and intelligent control of Instituto Superior Técnico (IST) office rooms.

1 INTRODUCTION

This document addresses the topic of creating an appropriate architecture for intelligent control and energy management of building offices. Given the current economic and social situation, it is of utmost importance to focus on systems and technologies that enable energy efficiency, particularly in large buildings. In Europe, buildings account for 40% of total energy use and 36% of total CO_2 emission (European Parliament and Council, 2010), and these values are expected to grow in the coming years. This aspect result in a major concern for more energy conservation. Large buildings require technologies with sophisticated applications, such as automatic control of lighting and temperature set-points, as well the need for the system to learn with the activity of users inside the building. In addition, these environments must be energy efficient and must simultaneously ensure the comfort and well being of the occupants in order to enable them to become more productive. The main hindrance in developing this vision are the lack of interoperability that automation systems offer. Each manufacturer keeps developing their proprietary specifications and has little motivation to evolve their systems to the emerging trends of communication standards (Litvinov and Vuorimaa, 2011). As well, there is still no solution for the development of applications for intelligent control and energy management in a modular way, based on Service-Oriented Architecture (SOA), where it is possible to reuse and compose services in order to implement new functionality. The current solutions of Building Automation (BA) and Energy Management (EM) are vertically integrated and do not allow communication with devices in a standard way, and poorly support their heterogeneity. Re-designing and re-programming BA and EM systems, in order to change and extend features to support new devices and functionalities are still too expensive and to slow to achieve. Web Services (WS) and SOA has allowed it, more precisely in Information Technology (IT) systems (Lee et al., 2010), due to several business needs, but not embarked on the automation and energy management domain yet. What really happens is that both services and the programmer has to know the details and characteristics of the Building Automation System (BAS), which causes a massive consumer lock in. Increasingly it is necessary that the IT system and BAS are integrated in the architecture that supports the business, since the optimization of the energy and the business processes is something that has to be done in an integrated way.

Santos R. and Carreira P..
Service Oriented Development of Building Energy Management Systems - An Architecture Blueprint.
DOI: 10.5220/0004974903150323
In *Proceedings of the 3rd International Conference on Smart Grids and Green IT Systems* (IEEHSC-2014), pages 315-323
ISBN: 978-989-758-025-3

This work will analyse the various existing functionalities in Building Energy Management System (BEMS) and propose a reference architecture with a set of services collections required in order to create modular intelligent energy management and automation control applications on top of these services.

1.1 Motivation

To illustrate the addressed issue take into account the following scenarios:

Scenario 1 consider a daylight-harvesting system, which aims to use daylight to control the amount of electric illuminance needed to properly light a space, in order to reduce energy consumption. These systems uses a set of sensors and actuators inside the room to measure luminance and adjust the lighting of the lamps, respectively. In this scenario it is necessary to deal with interoperability of diverse devices, their functionality and technical features, since many are from different manufacturers and communication protocols.

Scenario 2 consider a scheduling system, which execute some tasks periodically. This tasks are triggered after an event or set of events have occurred. Events can be a time value or a specific event from the user's activity. In this scenario it is necessary to understand how the scheduling system of each device technology works and map the logic of scheduling with the operation logic of each device.

Existing solutions to address the requirements of scenarios 1 and 2 consist of monolithic applications that do not provide modular services, and also do not allow interoperability with other automation technologies used. In addition, most existing solutions are proprietary and not expose their insights, which don't allow the reusing and extension of its functionality. This situation is illustrated in Figure 1, which compares a monolithic system with a layered architecture (Bastide et al., 2006).

SOA is a paradigm that defines services as key elements of software design. The main goal of SOA approach is to reduce the dependencies of software parts, where each part is typically a remote piece of functionality accessed by clients. By reducing such dependencies, each element can evolve separately and the application becomes more flexible than monolithic applications. The advantages of SOA also apply to automation applications: loose coupling, run time service binding, and location transparency. Indeed, services abstract implementation to provide the desired functionality; they provide faster prototyping based on creation of new services from existing (service composition); they enable handling the diversity of devices and applications, because they are easier to integrate and evolve; they enable integrate BAS functionality with the IT systems underlying the occupant business. Technically, SOA provides abstraction: upon a call to the specific service, the implementation is invoked regardless of whether it is deployed in a device, or in another software module. This latter paradigm motivates the need of a SOA-based reference architecture for Smart Building (SB) systems.

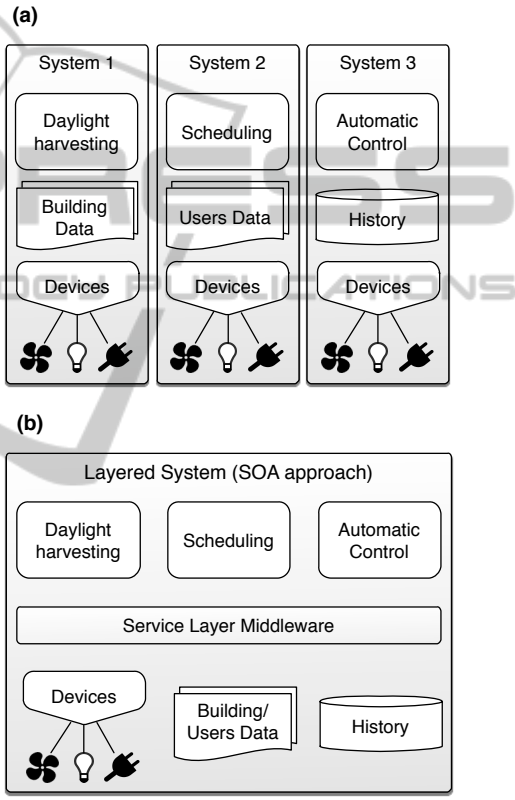

Figure 1: Actual systems vs. SOA approach. (a) Monolithic architecture where high level applications are highly dependent on the original software design. (b) Layered horizontal solution that enables technology abstraction, interoperability, and extensibility.

1.2 Problem Definition

Solutions to this problem are quite complex as is necessary to realize the system's architecture and services that the middleware service layer has to offer. Typically, architectures for these systems include services for energy consumption and a services for commanding devices. Indeed, these service components are needed. It's necessary to integrate all context information (e.g. retrieve the state of electrical devices,

the energy consumption data, and the user presence data), to derive control decisions that meet the user preferences and needs while ensuring the best values of energy consumption in the building.

Solutions, as detailed in *Related Work*, are trying to solve this issue using service-oriented architectures to provide some interoperability between different systems. However, they only expose devices as service calls along with the idiosyncrasies associated with each device. There is no semantics on top of these services, such as logical services to reuse and extend. The characteristics of different devices, the different protocols and different drivers continue to exist, but now at the service level, maintaining the lack of interoperability.

Herein, we propose a way to build BA applications atop of a modular service layer middleware that also abstracts the heterogeneity of BAS. The middleware abstract all systems and low level devices to BA applications, making less complex to develop these software. This novel approach enables taking advantage of systems that already exist, orchestrating and reusing the services they offer to implement new features that take into account the needs of the business. Conceivably in the near future, BAS will enable adding new features without having to redesign the original architecture of the system.

1.3 Contributions

The contributions of this document are as follows:

- An review of the existing solutions concerning service-oriented architectures of BEMS,
- A architecture blueprint for a BEMS, that enable the easy extension and fast development of new BA and EM applications.
- A architecture case study in a real-world setting. Our architecture prototype will be used in IST building, particularly in offices rooms, in order to validate the proposed architecture to this scientific problem.

2 BACKGROUND

In this section, it is given a brief description of some of the key terms and systems addressed, in order to provide a better reader understanding.

2.1 Building Energy Management System

A Building Energy Management System (BEMS) is part of a Building Management System (BMS) and is a computer-controlled system, which when installed in a building, controls it's power systems, including lighting, heating, ventilating and air-conditioning. The basic functions of a BEMS are: monitoring, controlling, and optimizing (Levermore, 2002). The functional aim of BEMS is to manage the environment within a building, so that energy consumption perfectly balances with the way of building utilization and its normal activities. BEMS functionalities addressed in ISO 16484-3 and EN 15232 standards (International Standard, 2007; European Standard, 2006) are: grouping/zoning; event notification; alarm notification; historical data access; scheduling; and scenarios. Also according to ISO 16484-7 standard (International Standard, 2013), a BEMS shall provide the following control features to enable energy performance in buildings: heating and cooling control; ventilation and air conditioning control; lighting control; and blind control.

BEMS are dependent on the other two systems: Building Automation System (BAS) and Energy Management System (EMS) (Bloom and Gohn, 2013).

2.2 Service-oriented Architecture

The idea of a service-oriented architecture (SOA) refers to a paradigm for the construction and integration of software systems where the primary structuring element is the service. In SOA, any subsystem is described by the services it offers (Open Group Standard, 2011). Web Services (WS) are the most common implementation of SOAs. WS consist of modular and independent software applications that can be executed over the web. These applications usually use XML and SOAP over standard network protocols, to implement the communication channels. Organizations that focus their development effort around the creation of services, will realize many benefits, such as, code mobility and reusability, scalability and availability, and better testing.

2.3 OSGi

Open Services Gateway initiative (OSGi) is a component framework specification that aims at bringing modularity to the Java platform. It enables the creation of highly modular and dynamic systems. These

systems are highly modular in the sense that OSGi-based applications are composed by multiple independent modules or components, and the development, testing, deployment, updating, and management on each module have minimal or no impact on the overall system. This means that bundles can be safely added and removed from the framework without restarting the application process (Hall et al., 2010).

3 RELATED WORK

Smart buildings require integration of various building automation systems that are usually made by different manufacturers (International Standard, 2013). Most of the manufacturers tend to focus on a specific domain, while others try to comprise all domains in one application. The need to integrate several network technologies and communication protocols into applications has forced developers and researchers to create service-oriented frameworks or use existing to accomplish that objective.

An bottom-up analysis is presented below, starting from the need of *heterogeneity abstraction and interoperability* of several field devices technologies, where it's required that this systems provides *services exposition and composition* in order to compose all various systems in a *technology independent integration* way. After being achieved the integration of various technologies, it is possible to provide several *energy management and automation functionalities* that will deliver application-level services to the end-user applications.

3.1 Field-bus Heterogeneity Abstraction

Given that there are several automation devices from different manufacturers, they are incompatible and can't communicate with each other. A popular method to achieve integration is defining a common protocol between centralized front-end and individual gateways to proprietary systems.

Building Automation and Control Network (BACnet) (Bushby, 1997) protocol has been widely accepted and used in BAS industry. BACnet is a communication protocol, developed by the American Society of Heating, Refrigerating and Air-conditioning Engineers (ASHRAE) and its concept is to replace the communication portion of each device with a common, standard set of communication rules, so that all devices can exchange information using only one language.

LonWorks (Corporation, 1999) is another completely different solution and a great competitor. LonWorks is actually a family of products and at the core of this technology lies a communication protocol called LonTalk.

Despite the fact they seem good solutions as it is, both require Web Services interfaces to accommodate functionalities integration (Szucsich, 2010), as you will see in the next sections.

3.2 Services Exposition

The primary technologies used in SOA domains are Web Services since they have broad industry acceptance. In the BA, there is an increasing need and opportunity to create interfaces between the physical devices and software applications. BAS developers, inspired by Web Services technologies, began researching for a way to extend the SOA paradigm into the device space, that is, implementing a high-level communications infrastructure based on Web Services protocols at the device level (Jammes and Smit, 2005).

Devices Profile for Web Services (DPWS) (Mensch, 2009) is the extension of the Web Services protocol that surfaced. With DPWS, all messaging, whether related to discovery, control or event notification, is based on the use of SOAP. In that context, a target service is a device. Once discovered, a device exposes the services. This protocol is a service enabler for devices, i.e. converts binary communication to XML via SOAP, and the technical features of the devices still exist and it is necessary to deal with them at the higher level.

Service-Oriented Device Architecture (SODA) (Deugd et al., 2006) is an adaptation of SOA which aims to integrate a wide range of devices into an enterprise system. Once again, devices like sensors and actuators are dealt as services. A SODA implementation adds a new layer to the SOA model, wherein device interfaces and protocol adapters are provided, enabling communication between proprietary and standard device interfaces, and other SOA services over the enterprise network. This protocol is also a service enabler, but offers some integration of devices and provides services extension and composition to the software layers above.

3.3 Technology Independent Integration

System high level protocols are used to connect these automation systems with each other and also to integrate them with management software components.

BACnet/WS (Szucsich, 2010) is a standard designed to be a Web-Service based communication

protocol in the industrial and home automation which can be applied to any fieldbus system. BACnet/WS is independent from the underlying BAS and allows heterogeneity abstraction and exposes their data model and attributes through a web service interface. Although this technology allows to expose services, these are fairly primitive and generic, as only provides basic operations of read, write, and collect historic data of nodes.

OPC Unified Architecture (OPC UA) (Leitner and Mahnke, 2006), is an architecture designed by the *OPC Foundation* to connect industrial devices to control and supervision applications (Hadlich, 2006). The focus of OPC is on getting access to large amounts of real-time data while ensuring performance constraints without disrupting the normal operation of the devices.

Open Building Information Exchange group (oBIX) (OASIS Open, 2013) is a specification and aims to integrate enterprise functions in several building control systems. This specification is designed to provide access to the embedded software which sense and control the environment around us. Current version of oBIX defines two protocol bindings designed to leverage existing Web Service infrastructure: an HTTP REST binding and a SOAP binding.

Home SOA (Bottaro and France, 2008), is an open architecture for home service deployment. It is based on a service platform concept, where distribution and protocol heterogeneity are managed by service-oriented drivers. Home SOA allows developers to model networked applications as a local composition of uniform service interfaces, which become bounded at runtime. This architecture is implemented regarding the OSGi standard (Dobrev et al., 2002), where its specification defines how to deal with device's access: how to share and isolate resources between modules. It proposes a modular architecture that can be opportunistically adapted on various topologies, protocol sets and platform technologies, as it integrates various home protocol drivers in an architecture where functions are registered in a uniform programming language API. This model was applied in multimedia, home automation and device management domains.

DomoNet (Miori et al., 2006) is a service-oriented solution framework designed for home environments. This prototype implements an architecture which makes the connection between lightning devices of two different middlewares: UPnP and KNX. This solution is based on a SOA and uses web services to reach the interoperability between different home computer middlewares. The aim of this solution is to prove how an approach that relies on XML, Web Services and Internet protocols, can provide a uniform architecture to home automation.

LinkSmart middleware, formerly named Hydra middleware (Eisenhauer et al., 2011), is part of a European project for the development of service-oriented software to expose heterogeneous device functions in a universal way and target domains such as home automation, among others. The main idea is to generate the software components and web services in order to integrate within the supported device instances, based on the semantic model meta-data. Decoupling the application development from the device programming brings advantages with regard to modularity, reusability and extensibility.

Service-Oriented Cross-layer Infrastructure for Distributed Smart Embedded devices (SOCRADES) (Cannata et al., 2008) is a Framework for developing intelligent systems in manufacturing and relies on European research project addressing SOA-based manufacturing paradigm. It's primary objective is to develop a design, execution and management platform for next-generation industrial automation systems, exploiting the SOA paradigm both at the device and at the application level. SOCRADES Framework uses a improved DPWS specification in order to perform a Web Service-based network architecture. This defines a set of built-in services that allow service orchestration, service management, semantic web services, and service gateway. At application level, SOCRADES allow a unified enterprise integration via a cross-layered web service infrastructure (Karnouskos et al., 2007).

3.4 Energy Management and Automation Functionalities

Some solutions focuses on energy management and building automation functionalities to provide best comfort while ensuring efficient energy consumption. This aspect is the most important functional requirement in a BEMS.

PoliSave (Chiaraviglio et al., 2010) is a software designed to reduce power consumption in computer networks. It manages the scheduling of power on and off of building computers in order to save energy. The process is carried out via a graphical web-application interface. A similar solution is what we aim to develop in this project regarding the computers, displays and datashows present in the building.

MEDUSA (Davidyuk et al., 2009) is a middleware based on Activity-oriented computing (AOC) paradigm. This paradigm promotes run-time applications by composing ubiquitous systems based on abstract specifications of user activities. It is an interest-

ing approach if we think about the SB context when we define different possible scenarios to be achieved in terms of a lecturer wishes to give a class, an exam or do a presentation.

aWESoME is a Web Service Middleware for Ambient Intelligence environments (Stavropoulos et al., 2013). aWESoME aims to implement a large-scale building system, regarding energy consumption savings. The middleware is the middle layer that establishes the connection between the hardware layer and the various applications layer. The main goal of this solution is to fulfil functional and non-functional requirements of the system and to ensure universal, homogeneous access to the system's services.

Smart Energy Efficient Middleware for Public Spaces (SEEMPubS) (Osello et al., 2013) is an European project that aims exploiting ICT monitoring and control services to reduce energy usage and CO_2 footprint in existing buildings. This approach focuses on easily integration of existing BMS with new sensors and actuator networks. The main goals of this project are: a integrated Building Automation and Control System (BACS); a middleware for energy-efficient buildings domain; and a multi-dimensional building information modelling-based visualization. SEEMPubS layered architecture are composed by three layers: LinkSmart Proxy Layer, SEEMPubS Middleware Layer, and SEEMPubS Application Layer. This solution lacks in providing application-level services to BEMS.

3.5 Discussion

Previous section was addressed several technologies that focus on trying to solve different domain issues, such as automation field-bus heterogeneity abstraction, BA services exposition, BAS technology independent integration, and energy management and automation functionalities providing.

3.5.1 Non-Functional Requirements

Most solutions are supported by SOA, thus providing greater extensibility and interoperability with other technologies in an easier way. Systems that stand out for a higher extensibility and interoperability are: OPC UA, HomeSOA, LinkSmart, SOCRADES, aWESoMe and SEEMPubS. These solutions solve some issues of the problem, because they can help development of BEMS supporting BA technology integration and abstraction. As they are based on SOA it is possible to implement EM and BA functional application-level services over this systems, using a modular service composition, being that the focus of our architecture. The aWESoME and SEEMPubS middleware are the most similar approaches regarding the architecture objectives, as it integrates different layers: hardware, services, integration between services and devices layer in order to achieve their interoperability, and application layer to provide the users of the system a form to manage the devices and the energy consumption of the building rooms within a user-friendly interface.

3.5.2 Functional Requirements

Despite all solutions support BA and EM functionality, majority have a lack support of BEMS functionality, which causes a need to implement a new system with all the standard features. Application-level services providing is an important feature, that enables extension and development of BAS applications easily. No solution fulfils all the characteristics surveyed. However, they all complement each other, increasingly motivating the creation of a new reference architecture. One aspect that lacks enough, are the system ability to provide BA and EM application-level services in order to help the developers to build easily and quickly BEMS sophisticated applications.

4 ARCHITECTURE BLUEPRINT

The main goal is to develop a new middleware architecture in order do overcome the identified issues, such as interoperability and extensibility of BEMS. It is presented a middleware architecture vision and its software layers, and a collection of different services that the middleware must provide and also is presented and described what is the purpose of each service collection.

To achieve the middleware goals, existing building automation and energy management software standards must be refactored into a modular architecture. OSGi framework can be used to implement such architecture, since it has mechanisms that promote modularity and also can afford all requirements of BEMS. The first step to modularize the application, is to identify its essential structure. This will help to understand the dependencies and identify interactions. From there its possible to define the building blocks that constitute the middleware. Finally, with a fully modular and functional middleware it's possible to implement a sophisticated BEMS fully based on SOA, being also highly functional and extensible system.

4.1 Architecture Overview

This section proposes an architecture for the service layer middleware, aiming at flexibility and scalability, enabling easy development of new applications in a modular way.

As opposed to other solutions, this architecture is based on the BA and EM standards (International Standard, 2007; European Standard, 2006), regarding also to the latter paradigms of software architectures. This platform will centralize all the building's control functions, in order to reduce the installation, commissioning, maintenance and hardware costs. The proposed layered architecture is composed by several extensible blocks with well-defined interfaces that abstract it's implementations. The implementation of each block can be a software module or even can be in other proprietary software solution module.

This novel architecture will be very useful, to the extent that at the end we will take advantage of several benefits:

- *Abstraction* - modular service components will allow multiple levels of abstraction;
- *Fast prototyping* - creation and utilization of existing services;
- *Heterogeneity handling* - services will enable integration of devices diversity;
- *IT integration* - will be possible to create services that integrate automation systems with the IT systems present in the building;
- *Lower operational and maintenance costs* - there is no more maintenance dependence with suppliers and manufacturers of equipment and systems.

As depicted in Figure 2, this architecture is able to manage and interoperate with several fieldbus technologies and their devices, thus solving the problem of heterogeneity. The system also implements energy management and intelligent control functionalities on service layer middleware instead on expensive hardware controllers. High-level services, will provide all the necessary functionality to the fast development of BEMS applications. The various services presented are divided into four layers, corresponding to different types of features. Each layer is a different level of abstraction, where the lower layer offers services to upper layer, which simultaneously uses the services of layer below. Service Bus, will enable interoperability between service layers, promoting communication between service providers and service consumers.

4.2 Service Collections

In Figure 3 are shown the set of services that will provide sophisticated functionality to Applications layer. The services were divided into multiple logical levels, the service collections, taking into account the type of BA and EM functionality and the energy-efficient control techniques according to standards (International Standard, 2007; European Standard, 2006).

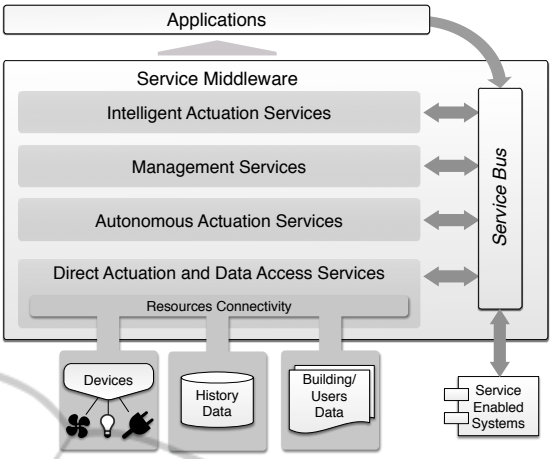

Figure 2: Proposed Service Architecture. The Middleware consists of several service modules which deal and abstract the specifications of each different BAS technology and provide functional services for high-level applications, starting from the field level, then building automation, energy management, and intelligent control.

The *Direct Actuation and Data Access Services Collection* intends to deal with BAS devices communication and it's abstraction. It is also responsible for the devices history data storage, data querying functions, and data fusion and analytics techniques. The *Autonomous Actuation Services Collection* aims to provide some basic functionality actually present in BAS, such as alarms, events, scheduling and scenarios. The *Management Services Collection* aims to achieve a better way to manage the building and the BEMS functions, based on available data models of information. Finally, *Intelligent Actuation Services Collection* provides energy-efficiency control techniques robustly, given the higher service abstraction. These techniques are occupancy-based control, daylight-harvesting, automated blinds control, Heating, Ventilation, and Air Conditioning system (HVAC) control, and some learning techniques in order to take into consideration the user's activities and behaviour.

5 ARCHITECTURE CASE STUDY

The addressed service layer middleware will be applied in a prototype implementation and it's integra-

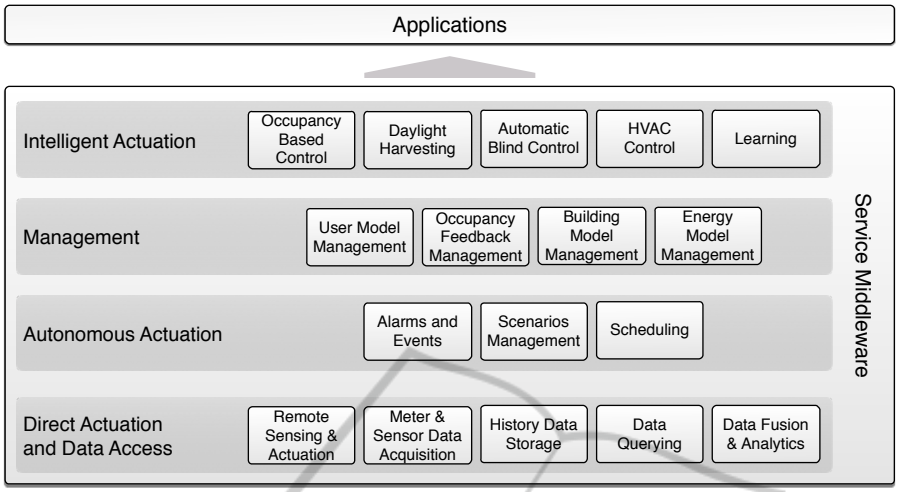

Figure 3: Service collections.

tion with existing BAS technologies already installed in IST building. Automation devices will be installed in office rooms, featuring a number of sensors and actuators, which will be used to implement software modules and services owned by developed middleware platform. This middleware will sustain intelligent automation and energy management applications that will drive the control of various electrical devices inside the office, such as heating, HVAC, and lighting.

In this study, we will not concern with whole functional requirements since they are intrinsic, i.e., the application must operate as any other BEMS solution and the automation devices and services must be fully functional and further developed in OSGi bundles. However, in order to validate the solution presented in this document, non-functional requirements must be validated. Next, we present some of the non-functional requirements that were identified:

- The development of new interfaces and software modules must be simple and not a time-consuming task (empirically measured).
- Deployment of new software modules must not stop the application and they must be available to use immediately.
- Middleware testability and debugging must be ensured.

The solution will be also evaluated according to the code reusability, level of coupling and the relative number of lines of code for the implementation.

There are several proven methods for evaluating an software architecture (Roy and Graham, 2008), such as early and late evaluation methods. Regarding early evaluation, a Scenario-based evaluation method must be applied in order to make sure that the architecture exactly meets all requirements before it's implementation. After architecture implementation, a Metrics-based later evaluation method must be applied in order to proof of compliance with the initial architecture requirements. Presented solutions in related work, were evaluated in similar ways, and we must also use some simulations in order to benchmark and evaluate the performance of final system.

In order to evaluate the user satisfaction, the prototype shall offer a Graphical User Interface (GUI) to the end-user from a handsome display available on the room wall. The display should provide a visualization of consumption curves related with office's HVAC and lighting systems, and even the access and modification of some user profile settings.

6 CONCLUSIONS

Developing software for Building Automation is not an easy task. With the proliferation of Building Automation System (BAS) became almost impossible to integrate several different systems made by different manufacturers. In the document we made a review of existing Building Energy Management System (BEMS) technologies and their main functions. We discovered some lacks in existing systems and we found that the best solution to this problem was to provide a modular layered middleware architecture in order to enable the ability to integrate and extend new Building Automation (BA) and Energy Management (EM) functionality easily. Thus we defined a set of services collections that we believe to be common in all building automation domains, according to existent standards. With this study, we expect to ascertain the best way to define a software architecture for BEMS, in order to the development of their function-

alities is made easier. Chosen technology for the prototype development was OSGi since it provides mechanisms that enforce modular design.

REFERENCES

Bastide, G., Seriai, A., and Oussalah, M. (2006). Adaptation of Monolithic Software Components by Their Transformation into Composite Configurations Based on Refactoring Example of Experimentation : A Shared-Diary System. pages 368–375.

Bloom, E. and Gohn, B. (2013). Building Energy Management Systems IT-Based Monitoring and Control Systems for Smart Buildings : Global Market Analysis and Forecasts.

Bottaro, A. and France, M. (2008). Home SOA – Facing Protocol Heterogeneity in Pervasive Application General Terms :. pages 73–80.

Bushby, S. T. (1997). BACnet TM : a standard communication infrastructure for intelligent buildings. 6:529–540.

Cannata, a., Gerosa, M., and Taisch, M. (2008). SOCRADES: A framework for developing intelligent systems in manufacturing. *2008 IEEE International Conference on Industrial Engineering and Engineering Management*, pages 1904–1908.

Chiaraviglio, L., Mellia, M., and Torino, P. (2010). PoliSave : Efficient Power Management of Campus PCs.

Corporation, E. (1999). Introduction to the LONWORKS System - Version 1.0.

Davidyuk, O., Georgantas, N., Issarny, V., and Riekki, J. (2009). MEDUSA : Middleware for End-User Composition of Ubiquitous Applications.

Deugd, S. D., Carroll, R., Kelly, K. E., Millett, B., and Ricker, J. (2006). Standards & Emerging Technologies SODA : Service-Oriented Device Architecture SERVICES.

Dobrev, P., Famolari, D., Kurzke, C., and Miller, B. A. (2002). Device and service discovery in home networks with OSGi. *Comm. Mag.*, 40(8):86–92.

Eisenhauer, M., Pramudianto, F., Researcher, M. S., and Kostelnik, P. (2011). Towards a generic middleware for developing ambient intelligence applications. pages 26–28.

European Parliament and Council (2010). Directive 2010/31/EU of the European Parlia- ment and of the Council of 19 May 2010 on the energy performance of buildings. *Official Journal of the European Union*, L153:13–35.

European Standard (2006). Energy performance of buildings - Impact of Building Automation Control and Building Management - EN 15232. 00247046:1–63.

Hadlich, T. (2006). Providing device integration with OPC UA. *2006 IEEE International Conference on Industrial Informatics*, pages 263–268.

Hall, R., Pauls, K., and McCulloch, S. (2010). *Osgi in Action: Creating Modular Applications in Java*. Manning Pubs Co Series. Manning Publications.

International Standard (2007). Building automation and control systems (BACS) - Part 3, ISO 16484-3.

International Standard (2013). Building automation and control systems - Part 7: BACS contribution on the energy efficiency of buildings.

Jammes, F. and Smit, H. (2005). Service-oriented paradigms in industrial automation. *IEEE Transactions on Industrial Informatics*, 1(1):62–70.

Karnouskos, S., Baecker, O., de Souza, L. M. S., and Spiess, P. (2007). Integration of SOA-ready networked embedded devices in enterprise systems via a cross-layered web service infrastructure. *2007 IEEE Conference on Emerging Technologies & Factory Automation (EFTA 2007)*, pages 293–300.

Lee, J. H., Shim, H.-J., and Kim, K. K. (2010). Critical Success Factors in SOA Implementation: An Exploratory Study. *Information Systems Management*, 27(2):123–145.

Leitner, S.-h. and Mahnke, W. (2006). OPC UA – Service-oriented Architecture for Industrial Applications.

Levermore, G. (2002). *Building Energy Management Systems: An Application to Heating, Natural Ventilation, Lighting and Occupant Satisfaction*. Taylor & Francis.

Litvinov, A. and Vuorimaa, P. (2011). Integration Platform for Home and Building Automation Systems. (PerNets):292–296.

Mensch, A. (2009). Devices Profile for Web Services Version 1 . 1. (July):1–43.

Miori, V., Tarrini, L., Manca, M., and Tolomei, G. (2006). An open standard solution for domotic interoperability. *IEEE Transactions on Consumer Electronics*, 52(1):97–103.

OASIS Open (2013). oBIX Version 1.1. pages 1–63.

Open Group Standard (2011). SOA Reference Architecture.

Osello, A., Acquaviva, A., Aghemo, C., Blaso, L., Dalmasso, D., Erba, D., Fracastoro, G., Gondre, D., Jahn, M., Macii, E., Patti, E., Pellegrino, A., Piumatti, P., Pramudianto, F., Savoyat, J., Spirito, M., Tomasi, R., and Virgone, J. (2013). Energy saving in existing buildings by an intelligent use of interoperable ICTs. *Energy Efficiency*, 6(4):707–723.

Roy, B. and Graham, T. C. N. (2008). Methods for Evaluating Software Architecture : A Survey.

Stavropoulos, T. G., Gottis, K., Vrakas, D., and Vlahavas, I. (2013). aWESoME: A web service middleware for ambient intelligence. *Expert Systems with Applications*, 40(11):4380–4392.

Szucsich, S. (2010). Web Services in Building Automation with focus on BACnet / WS. pages 1–25.