IST SmartOffice

Rodolfo André Henriques dos Santos
Instituto Superior Técnico
rodolfo.santos@tecnico.ulisboa.pt

November 2014

Keywords: Service Oriented Architecture, Building Automation, Energy Management, Modular Programming

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 favoring consumer lock-in. In order to overcome this problem, we propose to create a middleware that abstract the heterogeneity of automation systems and provide sophisticated functionality to applications for intelligent control and energy management. Our solution also exposes 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 and user comfort, without worrying about the technical details of each existing system. The concept is validated with the implementation of a service-oriented architecture prototype that enables energy management and intelligent control of 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 results 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 with the aim of 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 too 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.

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 users 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).

Solutions to this problem are quite complex as is necessary to realize the systems architecture and services that the middleware service layer has to offer. Typically, architectures for these systems include services for energy consumption and services for commanding devices. Indeed, these service components are needed. It is 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.

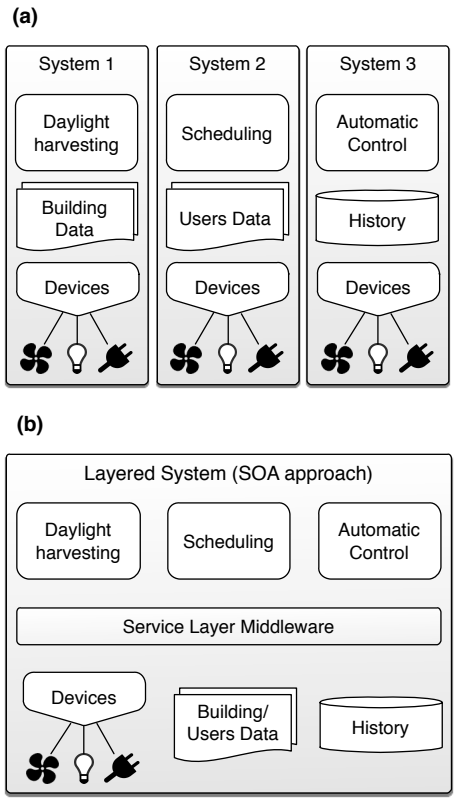

Figure 1: Actual systems vs. SOA approach. (a) Monolithic architecture where high level applications are highly dependent on the original software design. (b) Layered horizontal solution that enables technology abstraction, interoperability, and extensibility.

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.

This work analyses 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. At the end, IST SmartOffice prototype is integrated with the existing BAS available in IST Taguspark Building.

2 BACKGROUND

This section gives a brief description of some of the key terms and technologies addressed, in order to provide a better reader understanding.

2.1 Building Energy Management System

A BEMS is a computer-controlled system, which when installed in a building, controls its power systems, including lighting, heating, ventilating and air-conditioning. The basic functions of a BEMS are: monitoring, controlling, and optimizing (Levermore, 2002) all electrical devices in building. 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: 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.

Our literature review 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.

Most of 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 (Leitner and Mahnke, 2006), HomeSOA (Bottaro and France, 2008), LinkSmart (Eisenhauer et al., 2011), SOCRADES (Cannata et al., 2008), aWESoMe (Stavropoulos et al., 2013) and SEEMPubS (Osello, 2013). These solutions solve some issues, 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 de-

vices 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 an user-friendly interface.

Despite all solutions support BA and EM functionality, majority have a lack to supporting BEMS functionality, which causes a need to implement a new system that integrates all the standard features. Application-level services providing is an important feature, that enables extension and development of BAS applications easily. No solution fulfills 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 architecture main goal is to develop a new middleware architecture with the purpose of 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.

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 is 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 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 its implementations.

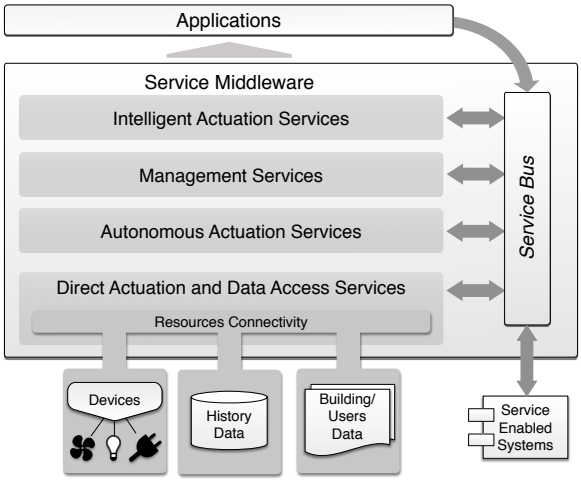

Figure 2: Proposed Service Architecture. The Middleware consists of several service modules which deal and abstract the specifications of each different BAS technology and provide functional services for high-level applications, starting from the field level, then building automation, energy management,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

This novel architecture will be very useful, to the extent that at the end we will take advantage of several key 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 Resources Connectivity

Resources connectivity is the service layer responsible for manage and communicate with the various BAS protocols. It also allows the access multiple data models manipulation, such as history databases, building models and users data preferences. In order to internal model interact with different protocols and BAS technologies, it is necessary to define a common API (Application User Interface) and create adapters that cope with various and heterogeneous protocol implementations and adapt them to a specific common protocol - the Datapoint Connectivity API, as depicted in Figure 3.

The Resources Connectivity Layer is essential to decouple the internal service API from specific BAS protocols and specifications (like KNX, X10, iLight, Modbus, etc.). The BEMS services contact to a specific BAS technology within the Resources Connectivity layer. The clearly defined interface between our internal services and the device driver services allows control of different devices via different adapters. Adapters will take care for translation and presentation of specific protocol details to Connectivity API service. Protocol Integration can access the different adapters implementations via OSGi services that guarantees a clear datapoint abstraction. Then, the decoupling and generic access to various devices via OSGi service interfaces is provided.

Datapoint API defines a general design rule to minimize implementation effort in Resources Connectivity Layer to be able to flexibly provide and support new adapters on customer/vendor request without (or with minimal) changes in internal services.

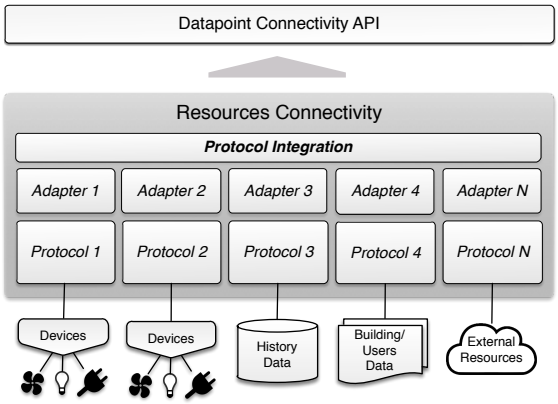

Figure 3: Resources Connectivity Layer. Figures above show the basic separation of different protocol implementations and the common and unified protocol – the *Datapoint Connectivity API*. The different adapters provide a clear interface of devices from different protocols, while Protocol Integration layer is intended to join the various adapters in a common communication protocol.

4.3 Service Collections

In Figure 4 are shown the set of services that will provide sophisticated functionality to Applications layer. These services were divided into multiple logical levels, 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).

The Direct Actuation and Data Access Services Collection intends to deal with BAS devices communication and its 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 users behavior and activities.

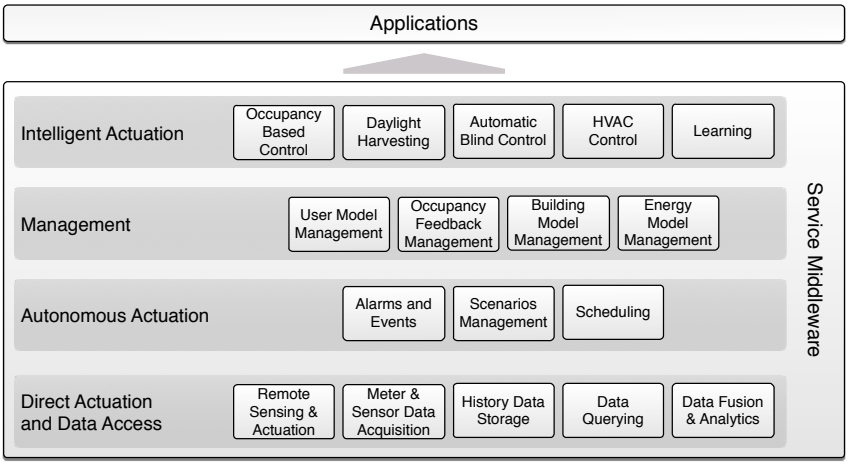

Figure 4: Service collections. The services were divided into multiple logical levels, according to the level of abstraction and also taking into account the type of BA and EM functionality.

5 IMPLEMENTATON

During the middleware implementation phases we developed an ecosystem of building automation and energy management services contained in the OSGi Framework. Then we specified the Datapoint Connectivity API, that enabled an unified way of communication between different services implementations, and also an abstraction layer to communicate with the heterogeneous environment of automation technologies. Later, we implemented techniques for services orchestration and management inside OSGi, and also the OSGi services exposition to the Web via Service Wrappers.

To give an overview of the implemented software components, the Figure 5 shows some of the developed bundles inside the OSGi platform, which corresponds to the different services and functionalities developed. The software components are grouped into 3 types:

1. **Core Bundles** corresponds to essential software that allows the proper operation mode of the global system
2. **Adapter Bundles** are software components that adds the communication support with new protocols and technologies
3. **Service Bundles** allow the addition of new functionality over services extensibility and composition

The implemented architecture was conceived having in mind that all the services are exposed using the same interface, the Datapoint Connectivity API. Lower level components are service providers and of higher level components that are service consumers. The end-user applications are only service consumers using the Datapoint Connectivity REST API.

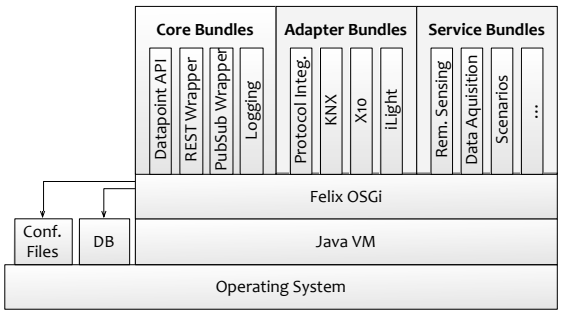

Figure 5: OSGi Level SW Components Overview. SW components follows the concept of modularity, where each piece of software is contained in an OSGi bundle.

6 EVALUATION

In this work, we designed and developed a service-oriented architecture for BEMS. Since there is no reference model of a SOA for BEMS, it is not possible to compare the results of our solution with another one. Therefore, our evaluation is based on an early software architecture evaluation that consists in a Scenario-based Software Architecture Evaluation complemented by two case-studies:

(i) Case-study 1: An empirical evaluation of the implemented prototype: the IST SmartOffice

(ii) Case-study 2: The development and evaluation of energy consumption prediction Services for IST Taguspark building

6.1 Scenario-based Software Architecture Evaluation

The analyses of ATAM (Architecture-based Trade-off Analysis Method) (Kazman et al., 2000) architectural approaches by applying scenarios and the ensuing discussions, surfaced a set of risks, sensitivities, and trade-offs:

Collected Risks

1. We are dependent on a component framework (OSGi). Exists the possibility of the framework stop being supported and a lot of functionality become legacy given the new trends and market needs
2. Many manufacturers of BAS may not accept this new architecture, and may difficult the provision and support of new features
3. Some specific technologies may have some limitations, which can not completely validate the requirements. E.g. the X10 protocol does not guarantee delivery of messages, which the state of device may be inconsistent with the state of the system
4. Perform the analysis of historical datam in order to detect malfunction situations can generate a high rate of false positives.

Collected Sensitivities

1. The service performance and scalability depends on BAS hardware performance
2. We are depending on the quality of sensors data, because may be there are associated rates of errors and failures.

Collected Trade-offs

1. Using the same Datapoint Connectivity API to implement all system services can simplify the usability of the overall system, but also a greater effort is needed to all services meets all the API specification
2. Using more than one API for the different services will simplify the implementation of each service, but it will reduce drastically the usability of the system. The user has to know all the APIs specification to interact with the available services.

6.2 Case-study 1: IST SmartOffice Prototype Evaluation

In order to evaluate the solution extensibility and heterogeneity handling, it was required to develop several fieldbus adapters to promote different technologies integration in a heterogeneity environment. This protocol integration is supported by the middleware, through the Protocol Integration Adapter. The BAS equipment installed in Taguspark Building enabled the development of device drivers that interact with major automation devices of the building. The middleware usability and functionality towards the device drivers implementation could be assessed through the development of such drivers.

In Figure 6, the infrastructure diagram of IST building describes all fieldbus technologies evolved, such as KNX, iLight, X10, LIFX, and INOV Meters.

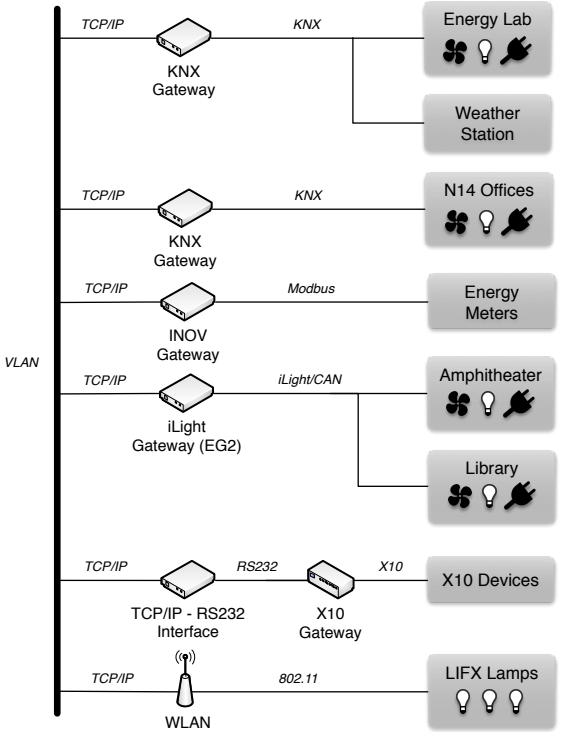

Figure 6: Fieldbus Infrastructure diagram.

Protocol	*KNX*	*iLight*	*INOV*	*X10*	*LIFX*
Functionality Coverage	100%	67%	100%	67%	67%
Complexity	High	High	Medium	Medium	Low
Notifications	Yes	No	No	No	No
Integration Dificulty	Higher	Medium	Lower	Lower	Lower

Table 1: Summary of integrated technologies results. The table shows the functionality coverage in the final solution after the integration process, the technology complexity over the developed adapter, the support of notifications and the difficulty encountered in the integration process.

From the analyses of the functionality coverage with the various fieldbus technologies integration with existing services and we can conclude that our

solution successfully addresses the SOA design principles of modularity, extensibility and interoperability. As depicted in Table 1, the majority of the functionalities were supported technologies after the integration process, even when some of them have limitations and do not support some functionalities by default. The technology complexity and the integration difficulty are also compared in order to evaluate the technology ability to integrate with others.

Finally, we can prove that existent technologies and standards needs to be improved in order to cope with SOA requirements by default, easing the integration process with other existing technologies.

6.3 Case-study 2: Energy Consumption Prediction Service

This case-study describes the construction and evaluation of energy consumption prediction services for intelligent buildings based on their heterogeneous occupancy. The IST Taguspark facility is used, where building occupation based on WiFi network connections are compared with the aim of better predict energy consumption of the building. We developed three services using these prediction models: linear regressions, fuzzy systems and neural networks. With these new prediction services, becomes possible to perform sensor data queries for future periods.

Prediction models are developed from a history dataset acquired from INOV Energy Meters readings, and also from occupancy estimation data stored in our History Data Storage Service.

Wifi usage information is available from SNMP MIB Agents of the 54 Access Points across the building. For that, we had to develop an additional adapter to integrate SNMP Protocol, using the SNMP4J Java Library, which acquires data periodically from all building network equipment via SNMP protocol. The Figure 7 shows the relation between Wifi connections (occupancy) and the overall energy consumption of the building.

The prediction models and services were developed using MATLAB Software from MathWorks. After the development, we integrated Matlab models into Java using the MATLAB Library Compiler. For *linear regressions* we used the Multiple Linear Regression. The *fuzzy system* was developed using Sugeno-type Fuzzy Inference System (FIS) with Subtractive Clustering (SC) algorithms. Regarding *neural networks*, a feedforward backpropagation model was applied.

Then, we presents the models simulations and respective results, using the following prediction models: *linear regressions*, *fuzzy systems* and *neural networks*. We used 80% train and 20% test data, the equivalent to 676 samples of 15-min test data, corresponding to ≈ 7 days (week-long) energy consumption prediction.

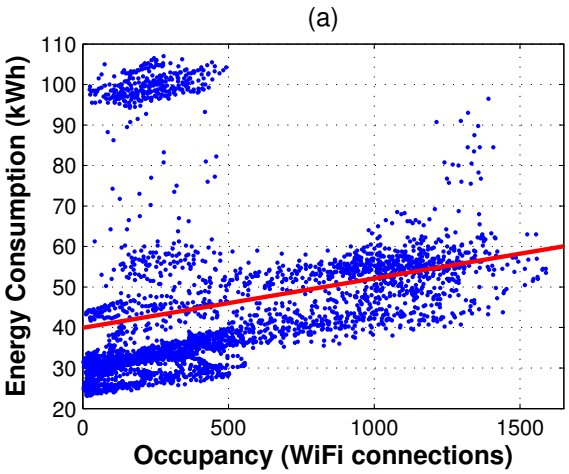

Figure 7: Scattered plots between building occupancy indicators and energy consumption, showing the WiFi connections and energy consumption relation.

The performance of all models is assessed for precision and recall by computing the Variance Accounted For (VAF) and Mean Absolute Error (MAE) indexes over a testing set. The VAF measures the percentage of variance between the two signals and can be expressed as:

$$VAF_i = \left(1 - \frac{var(y_i - \hat{y}_i)}{var(y_i)}\right) \times 100\% \quad (1)$$

The MAE measures the mean of absolute error values, and are expressed as:

$$MAE = \frac{1}{n}\sum_{i=1}^{n} |\hat{y}_i - y_i| \quad (2)$$

Figure 8 shows the VAF and MAE values for each type of model, comparing the best performance and the lowest error. The model that stands out for the best VAF and lower MAE index is the *Fuzzy System*. The *Neural Network* model has identical performance and error values.

In summary, fuzzy systems and neural networks, shows the most suitable models in order to easily predict a week-long energy consumption of an intelligent building based on occupancy variables, given their performance indexes.

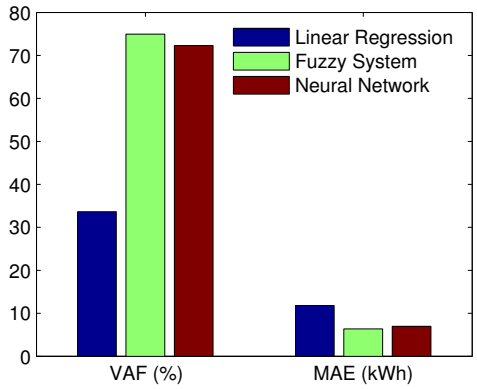

Figure 8: VAF comparison (higher is better) and MAE (lower is better) values for distinct used methods, summarizes the final performance of each energy consumption prediction service.

7 Discussion

During the evaluation phase, we defined some risks and trade-offs of this architecture, that helped to redefine some requirements in order to deploy an improved system architecture.

Then, during the Case-Study 1 phase that consisted in an integration process, the flexibility has been evaluated through the implementation of several technology adapters through a common protocol interface, the Datapoint Connectivity API. Our architecture proved to be quite capable to dealing with heterogeneous environment of automation devices and technologies present in the building. Nevertheless, not all features have been integrated, given there are some limitations of the technologies, being a risk factor that can make all the requirements is not fulfilled, as indicated in ATAM results.

Finally, in Case-Study 2, the energy prediction services development allowed us to prove the extensibility of the global system, where the capacity of integration with different sources of information, can deliver intelligent behavior functionalities using some data-mining techniques, such as the capacity of predict future data based on the analysis of indirect factors obtained from different data sources.

At the end, we can conclude that our solution successfully addresses SOA for BEMS requirements and principles of flexibility, usability and extensibility.

8 CONCLUSIONS

Developing software for Building Automation is not an easy task. With the proliferation of BAS became almost impossible to integrate several different systems made by different manufacturers. In the document we made a review of existing 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 architecture middleware, with the aim of enable the ability to integrate and extend new BA and 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 functionalities is made easier. The chosen technology for solution development was OSGi since it provides mechanisms that enforce modular design.

The results achieved in the evaluation of the implemented solution proves that IST SmartOffice prototype can fulfill all the defined requirements of SOA for BEMS. The interoperability goals were achieved and different protocols and services have been integrated in a software solution.

In the future, we hope that IST SmartOffice system will be extended to more ambient intelligent capabilities, introducing more energy-related benefits to the large buildings and their environment conditions, such as: comfort, productivity, energy-efficient. Comfort, by allowing users to adjust their personal needs. This is achieved by intelligent control of devices from a set of sophisticated applications built on top of intelligent services. Productivity, by optimizing the work environment for whatever tasks at hand. The building will deliver automatic and transparent behavior to the users in order to bring a better user satisfaction, since his comfort will be optimum, which allows workers to concentrate better for a longer periods of time, resulting in a higher level of productivity. Energy efficient, by reducing the overall energy consumption, not only from services with intelligent control which keep some devices off when they are not needed, but also by User Behaviour Transformation (UBT) services, that advise the occupants to meets the best practices related to the usage of electrical devices.

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.

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.

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 Parliament 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.

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.

Kazman, R., Klein, M., and Clements, P. (2000). ATAM : Method for Architecture Evaluation. (August).

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.

Open Group Standard (2011). SOA Reference Architecture.

Osello, Anna Acquaviva, A. A. (2013). Energy saving in existing buildings by an intelligent use of interoperable icts. *Energy Efficiency*, 6(4):707–723.

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.