Intelligent Building Management: A Service-Oriented Approach

Catalin Chera, Laurentiu Bucur, and Serban Petrescu

Computer Science Department, "Politehnica" University of Bucharest, 060042 Romania
catalin.chera@cs.pub.ro, {laur.bucur,bspetrescu}@gmail.com

Abstract. This paper introduces the current design of an intelligent building management framework. This framework consists of a service-oriented device network, a simple controller that can interoperate with a variety of devices, platforms, and networks, and a set of applications that allows users to design and compose control policies and services. This framework allows users to contribute information about policies, devices, and control services, and allows end users to compose control and policy services to be executed.

Keywords: service oriented, intelligent building management, policy based computting.

1 Introduction

This paper presents the overall architecture for the FCINT project [1] (Ontology-based Service Composition Framework for Syndicating Building Intelligence). This project aims to provide a service-oriented approach to control and manage building facilities via intelligent controllers.

Recently, numerous smart home projects have been initiated around the world, and they addressed various aspects of home applications and control. Specifically, Tiresias.org lists hundreds of smart home projects ranging from security, solar and wind energy efficiency, home appliances, smart workplace, voice-activated control, RFID, to elderly care and comfortable living for the physically impaired, hearing impaired, and visually impaired people. In the U.S., Arizona State University [2],[3], Duke University [4], Iowa State University [5] and Washington State University [6], among many other universities, have their smart home projects. In Europe, the SOFIA project [7] created its own version of smart home that outlines several generations of smart home. The DEHEMS project (www.dehems.org) and the FCINT project have also advanced in that direction. The Hydra project [8] just ended involved many countries and proposed a service-oriented network to support smart home applications. The SENSEI project connects devices with ontology for cyber-physical applications.

I. Dumitrache (Ed.): Adv. in Intelligent Control Systems & Computer Science, AISC 187, pp. 323–337.
springerlink.com

Organizations such as Cisco, Home Depot, ETRI, IBM, Intel (Home Energy Management) [9], Microsoft (Microsoft Future Home) [10], Philips, Siemens [11], and Toyota [12] have their own visions and projects. For example, the Microsoft Future Home uses voice recognition, cellular phones, cloud computing, visual and interactive wallpaper, and intelligent human-computer interaction to enhance daily living. The Siemens Smart Home Solution [11] is based on cellular phones and addresses comfort, security, energy (HVAC and lighting), healthcare, communication, and entertainment.

These projects have different focus and objectives, and involve different technologies. For example, Washington State University focuses on data mining and learning from data to support various activities and consumption; the HYDRA project proposed a tiered architecture and initiated a service-oriented approach for facility management; the SOFIA project aimed at providing great life experience supported by knowledge engineering; the Intel project provides computing processors to support smart home applications; the Siemens project leverages its large pool of devices and know-how to support a wide range of home activities. One may classify these projects as related to hardware (such as Intel), to devices, to data mining and optimization, to energy (solar panels and efficient buildings), to knowledge engineering (ontology, reasoning, and learning), and to user interface and experience (such as interactive wallpapers).

This project aimed to develop an infrastructure for people to share their building control services using a service-oriented platform. It will be an open platform to allow different people to participate and contribute: end users may contribute their profiling and preference data, device companies can supply device information including functional descriptions and input/output, software developers can contribute their control and policy services. In this way, both users and developers can compose new services by mashup for their application.

This project will provide an infrastructure for controlling building facilities with the following features:

A. The infrastructure should minimize the change to the existing infrastructure or installation;
B. The infrastructure should allow different people to contribute different parts of the system including information related to control devices, control policies, buildings, weather, and energy consumptions. It should allow changes to be made while keeping track of these changes;
C. End users can be able to understand, discovery, compose, or select appropriate control or policy services from the server for their buildings based on their own preferences.

This paper is structured as follows: Section 2 reviews the related work on device control. Section 3 presented the roadmap for the FCINT project. Section 4 illustrates the software architecture of the FCINT project. Section 5 concludes this paper.

2 Related Work on Device Interoperability

Recently, significant progress has been made in control devices with respect to device interoperability, e.g., how different devices from different companies can communicate and cooperate to perform missions. Numerous research organizations have proposed service-oriented infrastructure to support device interoperability. Most notable approaches include SODA [13], DPWS, and earlier UPnP and OSGi (www.osgi.org). A common thread of these approaches is to wrap the functions provided by devices as services that can be published, discovered, composed, executed, and monitored, just like software services in enterprise computing.

For example, SODA, supported by SODA Alliance with 27 partners from 6 countries, is an OSGi-based service-oriented architecture for connecting sensors and actuators. The SODA Stack consists of four layers from presentation, ESB, adapter, and device from top to bottom. The Stack provides a logical model for interfacing devices into the enterprise. In this way, the control devices will act like software services in enterprise computing without any software installation. Any new devices wrapped according to the SODA specification can be added to the network, and cooperate with numerous devices already in the network without any modification to the device. The adapter layer, through the SOA binding framework, bridges the gap between the device drivers and the communication to an Enterprise Service Bus (ESB) [14]. Its responsibilities are handling the session-level communication and device registration on the ESB. The Eclipse Foundation [15] has developed a Device Kit for SODA-compliant medical devices. The ESB layer can be any ESB suite installed in the enterprise. The Situation Layer includes all applications that respond and interact with the lower levels.

DPWS, a new international standard based on SODA, is supported by industrial giants such as Microsoft, Nortel, Red Hat, and Schneider Electric. It is an open-source specification on top of SODA. The DPWS uses WS: Addressing [16], WS: Discovery [17], WS: Eventing [18], SOAP [19], and XML [19]. In the EU SIRENA project, Schneider Electric produced early DPWS-compliant embedded devices [20]. Furthermore, SOA4D (Service-Oriented Architecture for Devices) [21] has established a website to host open-source solutions to support SODA, DPWS, and related technologies. As of February 2011, it has listed only 8 projects in two application areas.

WS4D is another organization that provides information related to DPWS. It uses WS-Discovery, WS-Eventing, WS-MetaDataExchange [22], WS-Transfer [23], WS-Security [24], WS-Policy [25], WS-Addressing, SOAP-over-UDP [26], SOAP, WSDL, XML, and XML Schema [19] as its stack of technology. The main applications of WS4D include cellular phones, wireless sensor nodes, and automation devices. UPnP shares many objectives with DPWS, and DPWS may eventually replace UPnP. UPnP has many standard SOA features such as media and device independence, user interface control, operating system and programming language independence, programmatic control, extensibility, and it

supports device addressing (based on IP), discovery, description, control, event notification, and presentation. The standard provides a set of specifications for secure messaging, device discovery, description and eventing from a device exposed as a web service. In DPWS a device is identified by a Hosting Service. Essentially, DPWS brings most SOA features from enterprise computing to the device level.

Many existing standards are available to connect devices in a network for building facility management such as LonWorks [27] (ISO/IEC 14908.1 standard), KNX [28] (ISO/IEC 14543-3 standard), and BACnet, and BACnet is a standard developed by the American Society of Heating, Refrigerating and Air-Conditioning Engineers (ASHRAE) supported by approximately 190 device vendors.

3 Roadmap of Control Devices

Smart home controllers can be divided into the following generations:

1. *Networked*: Devices are connected either via wired or wireless networks, and they can be remotely controlled via standard-based protocols such as BACnet standards. This is the current state-of-practice as many modern buildings are in this state, including the Palace of Parliament in Romania. Most devices in the Palace are connected, and many of them have sensors to detect movement. For example, the lights will automatically turn on once the sensors detect human movement in the room in the Palace today.

2. *Service-oriented networks for devices*: Devices are not only connected, but their services can also be published, discovered, composed, executed, and monitored in a service-oriented manner using standard-based protocols such as SOAP, WS-eventing, and WS-addressing. This is the current state-of-the-art, and DPWS is a representative standard. However, the design, implementation, and experience of using this approach are just beginning to appear in the literature. Applying existing standards in SOA leverage significant technologies that have been developed, however, most these protocols and standards have been developed for enterprise computing, the applicability of these to building facility control needs to be validated by further research and experimentation. In this generation, user applications need to be map to device services that have been published.

3. *Policy-based computing with simulation and ontology*: Devices are not only connected on a service-oriented network, but also user control services including policies can be published, discovered, composed, executed (or enforced). This is further supported by simulation so that users can simulate the behavior before deploying policies to be enforced. Currently, there is no standardized solution for device networks at this time. While policy enforcement for enterprise computing has been studied [29],[30],[31],[32], and policy enforcement has standards such as WS-Policy, specific policy enforcement for device networks has not been addressed. Instead, WS-policy has been proposed in the DPWS stack. In WS-policy, policies are enforced at the beginning of a service call in enterprise computing, but policies in a device network may need to be enforced at all times

(e.g., a temperature detector needs to enforce a policy of low temperature for a room at all times). Thus, policy standards for enterprise computing can be used at a higher layer, but at a lower layer, a new kind of policy mechanisms including policy specification, discovery, and enforcement for a device network will be needed. The low-level policy may be device-specific or application-specific, but will often focus on specific functions of the concerned device.

4. *Intelligent control:* In this phase, massive data concerning intelligent buildings such as devices, weather, BIM, and personal preferences will be available for data analysis, and computational intelligence can be applied to data. While numerous data mining and computational intelligence algorithms are available such as those done by the CASAS project, it is necessary to experiment these algorithms on buildings with devices connected in a service-oriented device network.

4 Current Project Components

In figure 1 the overall project architecture is illustrated. Central to the architecture is the client side Smart Building Controller. The client will host a service-oriented infrastructure that allows various services and policies to be installed and configured for execution and device control. Once a new service is loaded, the client will have a different behavior. The client will have an event-driven architecture (EDA) [33] and runs in real time to handle various events.

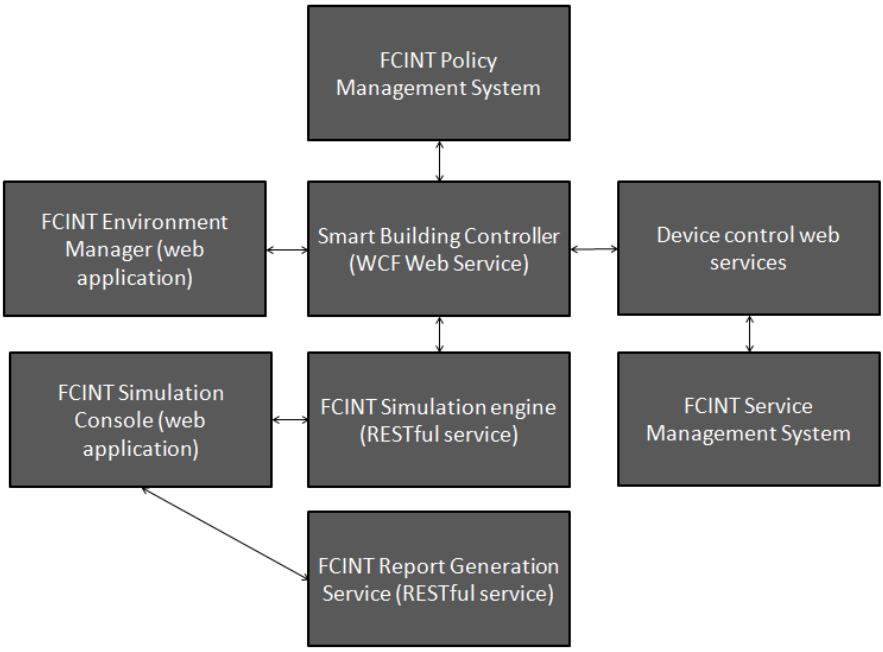

Fig. 1. The FCINT software architecture

4.1 The FCINT Smart Building Controller

The FCINT Smart Building Controller [34] is a service-oriented application that runs in the premises of a building environment to control. It's to control the devices and installations in the building by means of policy-based web service policy orchestration.

Fig. 2. The FCINT Smart Building Controller console

A simple scenario for demonstrating the policy-based approach of the SBC operation is a policy of the form:

If the windows are open, stop the air conditioner.

In this scenario there are two web services orchestrated by the SBC:

-Air_Condition_1
-WindowService

Air_Condition_1 receives commands from the SBC as web service calls and controls the air conditioning units. The *WindowService* receives status requested from the SBC as web service calls and returns the state of the windows. This policy can be expressed as a rule of the form:

on (WindowService.window_opened_event) *if*
(Air_Condition_1.IsOpened()=true) *then* Air_Condition_1.Close() *do* log_error

The *WindowService* registers the *window_opened_event* with the Smart Building Controller. The event triggers the policy that checks the status of the air conditioner by invoking the *Air_Condition_1* service instance. If the air conditioner is running then the policy enforcement invokes the *Close()* method. If the call to the service results in an error, the compensation action *log_error* is performed. The result of the policy enforcement is translated into the appropriate call to the air conditioning control web service that shuts down the air conditioning devices.

4.2 The FCINT Simulation Framework

The FCINT Simulation Console (figure 3) and Simulation Engine (figure 5) constitute a service oriented web application that allows:

a) The estimation of operating costs (consumption) of devices and facilities in a building, without their mandatory physical installation on site. Optionally, however, the simulator can estimate the energy consumption of real devices and installations controlled by a Smart Building Controller. This is achieved by the service composition mechanism between the FCINT Simulation Engine and the SBC Web service.

b) Testing the response of a Smart Building Controller to external events. This is achieved by creating various simulation scenarios (figure 4) in the simulation console, which contain events that are sent for execution to the Smart Building Controller during the execution of the scenario. The types of events are: Start/Stop commands for real devices and custom events that can originate from a real device. Upon receiving a custom event, the Smart Building Controller executes one or more event-triggered policies that specify the SBC behavior in the event of their occurrence, using a rule-based policy approach.

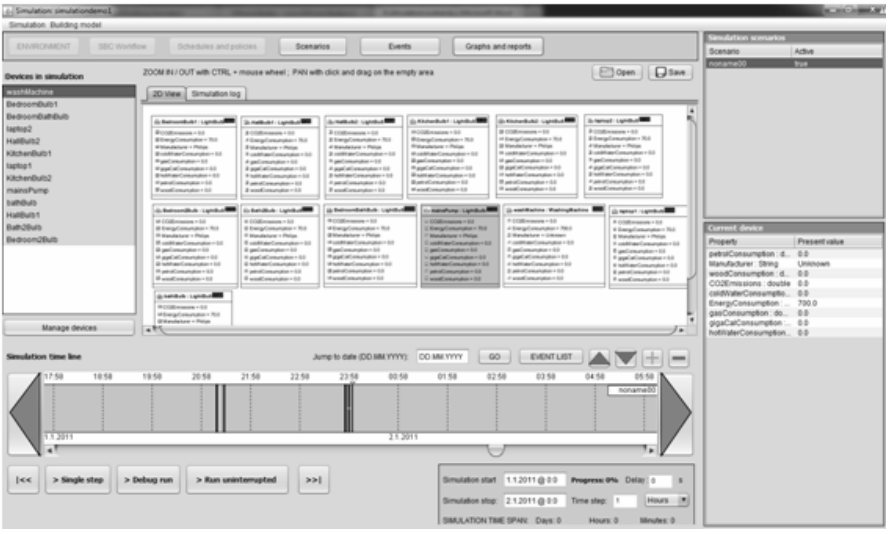

Fig. 3. The FCINT Simulation Console

The FCINT Report Generation service (figure 6) allows the Simulation Console to create a PDF report of a simulation. The report (figure 7) summarizes the energy, water, gigacallory and fossil fuel consumption and estimated costs for the duration of a simulation. The generated reports are PDF files that are an estimate of the total utility bill for the operation of a building environment.

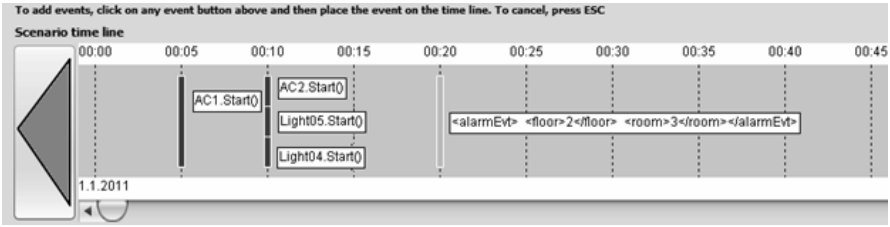

Fig. 4. A simulation scenario containing command and custom events

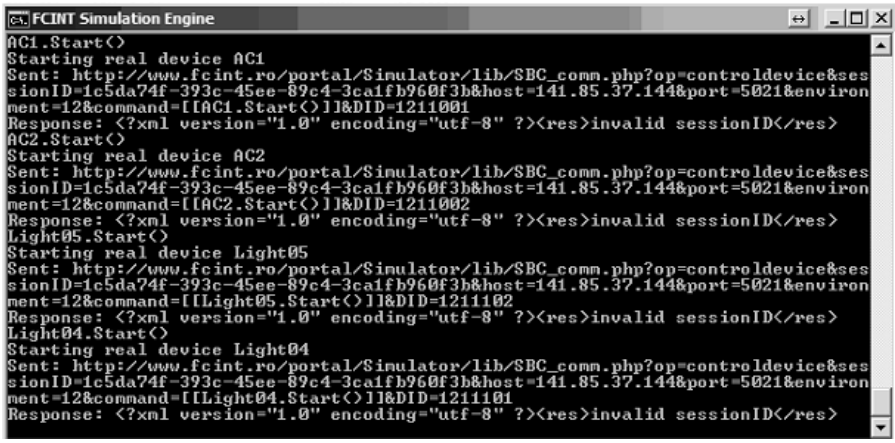

Fig. 5. The FCINT Simulation Engine

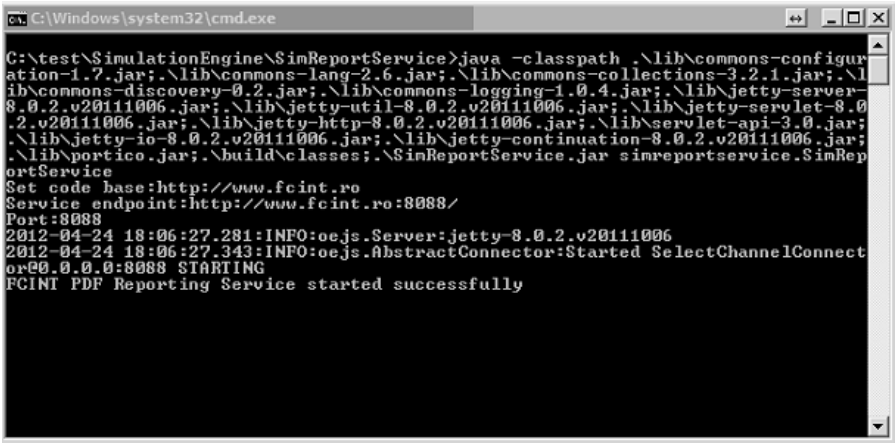

Fig. 6. The FCINT Report Generation Service

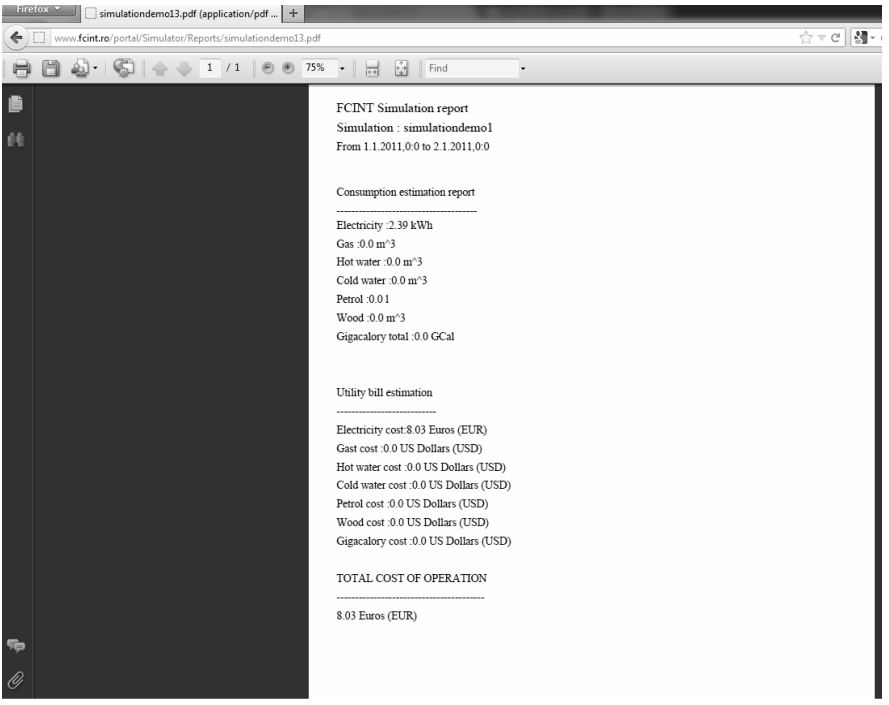

Fig. 7. Sample FCINT PDF simulation report

4.3 The FCINT Policy Management System

In the FCINT Smart Building Controller, users can specify control policies for their buildings according to their preferences. A policy is a set of rules that specifies constraints on the execution of the workflow. In this case, policies apply to the Smart Building Controller's orchestration workflow. Policies are loaded and enforced at runtime by the SBC using dynamic code generation. The FCINT Smart Building Controller uses PSML-P policy specification language [35] for policy enforcement.

A PSML-P Policy is a set of rules of the form:

$$on\ (\mathrm{E})\ if\ (\mathrm{C})\ then\ \mathrm{P}\ do\ \mathrm{A}. \tag{1}$$

where:

- E is an event;
- C is a condition which triggers the execution of a policy;
- P is a property the system must hold;
- A is a compensation action which is executed if P is not satisfied;

The types of events supported by the proposed architecture are:

a) The beginning and ending of a service call, denoted as S+/S-. These rules can be used to allow or deny a certain service call.
b) Data access, denoted by a small capital letter followed by a + for the writing operation and the d- for the reading operation. Example: d+/d-.
c) An arbitrary event type registered on the ESB by any of the following:

- Services on the Service Layer;
- Portal COM module;
- User Interface Module.

Based on the policy chain, the Smart Building Controller generates a sequential workflow that is executed when the event occurs. A special type of policy rule is the continuous rule that is enforced at each step of the orchestration workflow.

The FCINT Policy Management System is an application that allows the user to create, edit and manage the policies installed in a Smart Building Controller. The user can define both continuous operation policies and event-triggered policies by editing the rules using the user interface shown in figure 8. The user can also enable, disable and delete policies (figure 9).

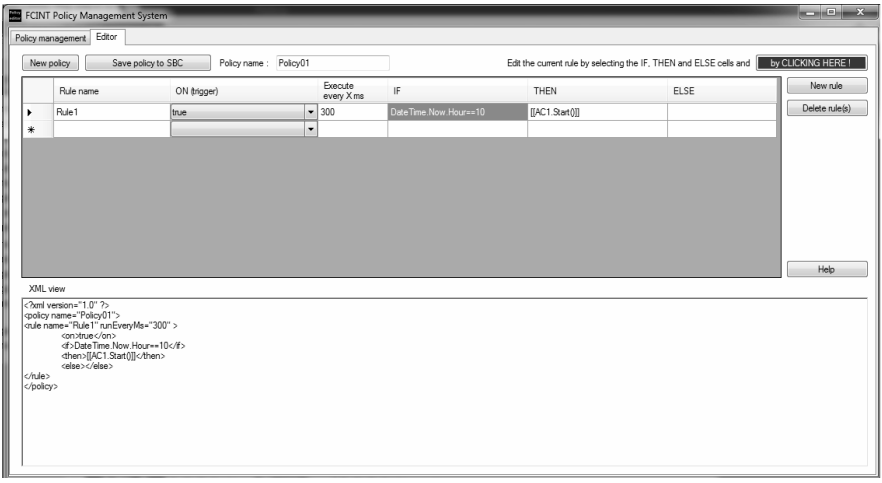

Fig. 8. The FCINT Policy Management System – Policy editing features

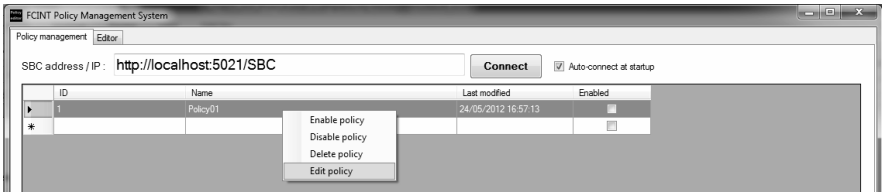

Fig. 9. The FCINT Policy Management System –Management features

4.4 The Device Control Web Services

The Device Control Web Services are web services running in the building that control physical devices and expose their functionality via the WSDL API. The Smart Building Controller orchestrates the execution of the device control Web Services based on user commands and on the set of policies active at any given time. The device control web services are installed, configured and finally tested from the FCINT Environment Manager.

4.5 The FCINT Service Management System

The FCINT Service Management System is a web-based repository where service developers can publish device control web services (figure 10) and where Smart Building Controller administrators can search for services (figure 11) that can be downloaded, installed and configured as device control web services.

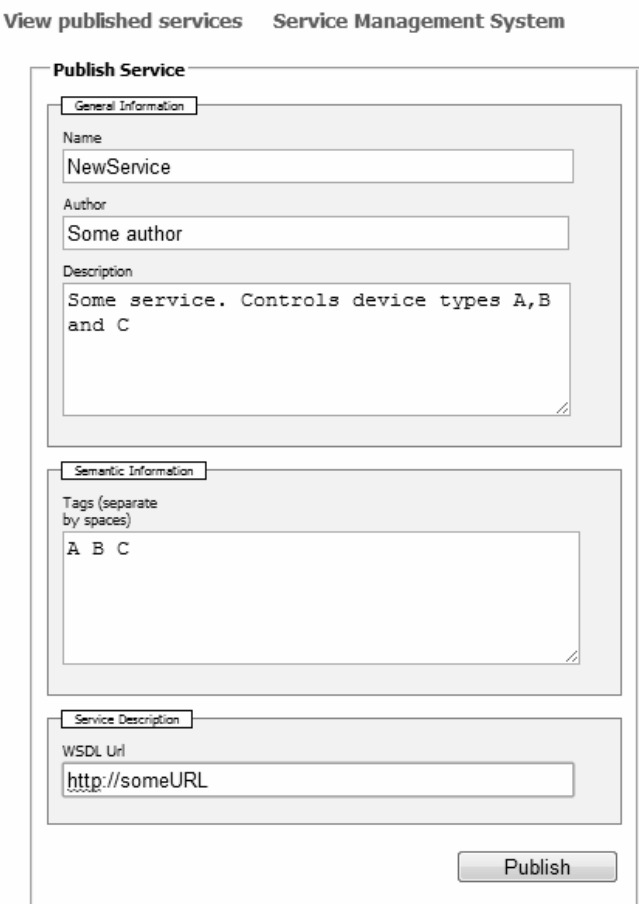

Fig. 10. The FCINT Service Management System service publishing feature

Fig. 11. The FCINT Service Management System service discovery features

4.6 The FCINT Environment Manager

The FCINT SBC can be accessed externally by a web application – the FCINT Environment Manager (figure 12). The building web control application connects to a Smart Building Controller instance to list the devices and device groups installed. It allows the user to create and configure the calendars and operating schedules of the devices and to visualize the scheduled actions and SBC activity logs for a given time interval. It also allows the visualization of SBC device status and their direct control.

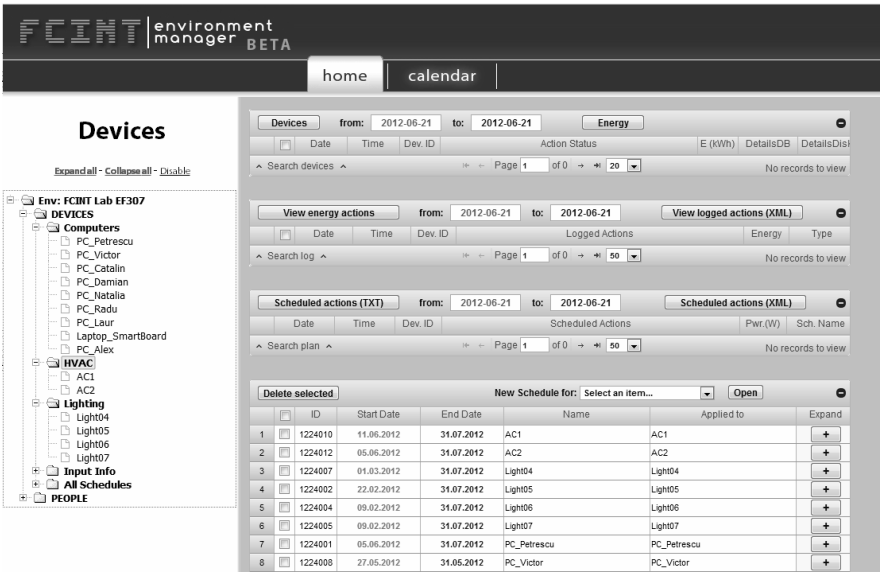

Fig. 12. The FCINT Environment Manager

5 Conclusions

This paper presents the overall architecture of the FCINT project, and it includes a small but smart controller, a set of web applications that allow users to publish, discover, compose, simulate control and policy services and a set of tools that allow the creation, management and simulation-based testing of smart building control policies.

Acknowledgments. The research presented in this paper was supported by the EU POS-CCE project FCINT No. 181/18.06.2010.

References

1. FCINT project, http://www.fcint.ro
2. Xu, J., Lee, Y.H., Tsai, W.T., Li, W., Son, Y.-S., Park, J.-H., Moon, K.-D.: Ontology-Based Smart Home Solution and Service Composition. In: ICESS 2009 (2009)
3. Lee, Y.H., Li, W., Tsai, W.T., et al.: A Code Generation and Execution Environment for Service-Oriented Smart Home solutions. In: Proc. of IEEE SOCA (October 2009)
4. Duke University, Smart Home project, http://smarthome.duke.edu/projects/list, http://www.tiresias.org/research/guidelines/smart_home.htm
5. Iowa State University, Smart Home project, http://smarthome.cs.iastate.edu/index.html
6. Washington State University Smart Home project, http://ailab.wsu.edu/casas/
7. Katasonov, A., Palviainen, M.: Towards Ontology-driven Development of Applications for Smart Environments. In: Proc. Intl. Workshop on the Web of Things at PerCom 2010, Mannheim, Germany, March 29-April 2 (2010)
8. The EU HYDRA project, http://www.hydramiddleware.eu/news.php
9. Intel Home Energy Management, http://edc.intel.com/Applications/Energy-Solutions/Home-Energy-Management/
10. Microsoft Future Home, http://www.youtube.com/watch?v=ODpReoKQVXM
11. Siemens, http://www.siemens.com/innovation/en/publikationen/publications_pof/pof_fall_2008/gebaeude/vernetzung.htm
12. Toyota Dream House, http://tronweb.super-nova.co.jp/toyotadreamhousepapi.html
13. The SODA Alliance, http://www.sensorplatform.org/soda/
14. Enterprise Service Bus (ESB), http://en.wikipedia.org/wiki/Enterprise_service_bus
15. Eclipse, http://www.eclipse.org
16. WS-Addressing, http://www.w3.org/Submission/ws-addressing/
17. OASIS Standard. Web Services Dynamic Discovery (WS-Discovery) (July 1, 2009), http://docs.oasis-140open.org/ws-dd/discovery/1.1/os/wsdd-discovery-1.1-spec-os.docx
18. Box, D., et al.: Web Services Eventing (WS-Eventing) (March 15, 2006), http://www.w3.org/Submission/2006/SUBM-149WS-Eventing-20060315/
19. Chen, Y., Tsai, W.T.: Service-Oriented Computing and Web Data Management, From Principles to Development, 2nd edn. Kendall Hunt Publishing Company (2010) ISBN 978-0-7575-7747-5
20. Jammes, F., Mensch, A., Smit, H.: Service-Oriented Device Communications using the Device Profile for Web Services. In: MPAC 2005, pp. 1–8 (2005)
21. SODA Alliance, http://www.sensorplatform.org/soda
22. Web Services Metadata Exchange (WS-MetadataExchange), http://schemas.xmlsoap.org/ws/2004/09/mex/
23. WS-Transfer, http://www.w3.org/Submission/WS-Transfer/

24. WS-Security, http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss
25. WS-Policy, http://www.w3.org/Submission/WS-Policy/
26. SOAPoverUDP, http://schemas.xmlsoap.org/ws/2004/09/soap-over-udp/
27. The LonWorks 2.0 Platform, http://www.echelon.com/products/lonworks_platform.htm
28. The KNX Standard specifications, http://www.knx.org/knx-standard/knx-specifications
29. Tsai, W.T., Chen, Y., Paul, R., Zhou, X., Fan, C.: Simulation Verification and Validation by Dynamic Policy Specification and Enforcement. Simulation, Transactions of the Society for Modeling and Simulation International 82(5), 295–310 (2006)
30. Tsai, W.T., Chen, Y., Paul, R., Chung, J.-Y.: Data Provenance in SOA: Security, Reliability, and Integrity. Service Oriented Computing and Applications Journal (2007)
31. Tsai, W.T., Huang, Q., Xiao, B., Chen, Y., Zhou, Z.: Collaboration Policy Generation in Dynamic Collaborative SOA. IEEE Computer Society Press (2007)
32. Tsai, W.-T., Zhou, X., Wei, X.: A Policy Enforcement Framework for Verification and Control of Service Collaboration. Information Systems and E-Business Management 6, 83–107 (2008)
33. Event-Driven Architecture, http://en.wikipedia.org/wiki/Event_driven_architecture
34. Bucur, L., Tsai, W.T., Petrescu, S., Chera, C., Moldoveanu, F.: A Service-oriented Controller for Intelligent Building Management. In: Proceedings of the 18th International Conference on Control Systems and Computer Science, Bucharest, Romania (2011)
35. Xu, J.: PSML-O: Ontology Specification Language and its Applications, http://gradworks.umi.com/33/61/3361858.html