OBJECT-ORIENTED FRAMEWORK FOR THE DESIGN OF HOME/BUILDING AUTOMATION SYSTEMS

Jair J. Araujo[1], Carlos E. Pereira[2]

Informatics Institute, UFRGS / Federal Center of Technological Education of Pelotas[1]
Robotics, Automation and Control Group, Electric Engineer Department, UFRGS[2]
E-mails: jonko@inf.ufrgs.br/jonko@cefetrs.tche.br[1], cpereira@eletro.ufrgs.br[2]

Abstract: The increasing demand on functionalities in new home and building automation systems leads to a considerable complexity increase both in the development as well as in the operation and maintenance of these systems. The major challenge is how obtain systems entirely integrated from isolated devices and subsystems. It is believed that an important stage to assess the actual need of this integration is to project an automating system without a specific technological focus, which does not occur nowadays. Due to the lack of supporting tools for the project, the stages of specification and project are generally focused on the available technology for implementation. This article aims to fill in this gap by specifying an object-oriented framework for the development of applications in the building automation, enabling the modelling of the systems regardless of the technology it will use, leaving this mapping for the last stage of the project.

Keywords: building automation, home automation, object-oriented, framework.

1. INTRODUCTION

Recent advances in electronics, microprocessors, and software, have considerably influenced the development of both industrial as well as home and building automation systems (HBAS) over the past years. While previous automation systems architectures were mostly centralized and not very flexible, modern industrial and home/building automation systems are highly decentralized and consist of autonomous, microprocessor-based devices, which are able to locally process information and make decisions. Such flexible, adaptive and even considered "intelligent" automation systems rely heavily on a distributed computer-based infrastructure, where smart sensors and actuators, and other automation devices can interact and communicate with each other using industrial protocols.

The use of this technology in modern home and building automation systems has positive benefits in terms of overall system's efficiency, reliability, and adaptive behaviour. However it has also considerably increased the complexity of design, operation, and maintenance activities. Moreover, the ability of smoothly integrating appliances from different vendors in order to achieve a real interoperability is still a dream, despite the attempts done by different organizations and many communication models and protocols for HBAS have been proposed over the last years. Examples are BACNET (SWAN, 2005), Lonworks (ECHELON, 1998), EIB (EIBA, 1999), HomePlugandPlay (CIC, 1998), just to name the most known.

Many authors have proposed distinct strategies to solve the interoperability problem by using runtime infrastructures, such as middleware (CHO, 2002; MOON, 2002). Others studies focus on the human-machine interfaces and try to incorporate "intelligence" in home appliances, with the aim that through a direct interaction among devices users goals can be achieved. Examples of such strategies are the European projects EMBASSI (EMBASSI, 2005) and Dynamite (DYNAMITE, 2005). These

projects aim at enhancing the intuitive interaction with tecnological systems by providing intelligent assistance, multimodal interaction, and anthropomorphic user interfaces within a unified framework.

Despite of the several advances in the area of HBAS, there is a clear lack on tools and methods that allow engineers to conceive HBAS applications with focus on systems' functionalities rather than on low level implementation and configuration details.

This work presents an object-oriented framework for the design of HBAS, which aims to fill this gap. It includes a class library and application profiles for typical activities in the area of building automation, such as HVAC control, energy management, etc. It also takes into account the different viewpoints to be considered when developing a HBAS, establishing a clear separation between logical functionalities, the appliances on which these functionalities can be executed, the physical location of the appliances inside a building and finally the communication protocols and technologies with which the system will be implemented.

The term "framework" is adopted with the same semantic as proposed by Gamma (GAMMA et. al, 2000) a framework is a collaborative class set which enables the re-use of the project to a specific software's class. It defines previously the architecture of application and the responsibilities of classes and objects in order to reduce the projects' decisions."

This paper is divided as follows: section 2 gives an overview on the proposed framework; section 3 describes the proposed design workflow and in section 4 conclusions are drawn and future work activities are signalled.

2. THE PROPOSED FRAMEWORK

This approach outlines the building automation system design in three stages:

- First, the objects and subsystems that model a HBAS are defined. These objects aim to represent the most common functionalities present in HBAS and they are called logical devices;
- In the sequence, logical design is carried on by (re-)using classes and application profiles available in the framework;
- Then, physical devices, corresponding to commercially available home appliances are selected. Logical devices previously selected are then mapped to these physical appliances which will constitute a distributed runtime infra-structure to execute the designed HBAS.

The proposed framework uses a foundation the *Reference Model for Open Distributed Processing* - RM-ODP as proposed in (ISO/IEC, 1995). The RM-ODP is the result from a group work of the ISO/ITU to establish a framework for development of multiplataforma large-scale heterogeneous distributed systems. It defines five viewpoints for representation of an open distributed system:

- enterprise view: whose objective is to specify the objects and the restrictions of a specified system;
- information view: this view has the concepts to allow the specification of information meaning. The information represents the data that needs to be stored and processed in the system;
- computational view: it describes the system as a set of objects which interacts by interfaces;
- engineering view: whose objective is to define the computational structure which supports the transparencies in order to allow the distribution.
- Technology view: it is associated with the choice of hardware and software technology that is needed to make the system.

The starting point for defining the Framework proposed in this paper was a thorough analysis of several open standards for HBAS. The goal was to identify common objects in different protocols in order to specify a minimum object set, which can be mapped for different technologies. Table 1 shows common objects as proposed in some of the cited protocols. The last column to the right depicts some of classes that have been defined in this work.

Table 1 - Equivalence among the object model by some protocols and the classes of the Framework proposed

EIB-Type_Id (dec) **EIB Object types**	**BACnet ID(dec)** **BACnet Object types**	**HomePnP ID(hex)** **HomePnP Object types**	**Model** **(proposed)**
0 - Device Objec	8 - Device Object	01 - Node Control	**PhysicalObject**
11 - File	10 -File	16 - Data Memory	**Persistent**
100 - Analogue-Input	0 - Analogue-Input	08 - Analog Sensor	**AnalogSensor**
101 - Analogue-Output	1 - Analogue-Output	07 - Analog Control	**AnalogActuator**
102 - Analogue-Value	2 - Analogue-Value		**Scene**
103 - Binary-Input	3 - Binary-Input	06 - Binary Sensor	**BinarySensor**
104 - Binary-Output	4 - Binary-Output	05 - Binary Switch	**BinaryActuator**
106 - Counter		1C - Counter/Timer	**Controller**
107 - Loop	12 - Loop	0A - Matrix Switch	**Controller**
108 - Multistate-Input	13 - Multistate-Input	10 - Multi-position Sensor	**MultistateSensor**
109 - Multistate-Output	14 - Multistate-Output	09 - Multi-position Switch	**MultistateActuator**
	6 - Calendar		**Calendar**
	17 - Schedule		**Scene**
		15 - List Memory	**Persistent**
		10 - Display	**UserInterface**
		11 - Medium Transport	**Channel**
		14 - Keypad	**UserInterface**
		1D - Clock	**Clock**

Based on this comparative analysis, a class hierarchy, which correspond the information view at RM-ODP, was proposed.
Other common characteristic among several protocols is the specification of application profiles in order to standardize a common structure that can be used in the devices of different vendors to guarantee interoperability among products of different suppliers. This corresponds to the "profiles" concept in LonWorks and EIB protocol, the "context" in the HomePnP and the concept of "groups" in UPnP protocol (MICROSOFT, 2000).

3. FUNCTIONALITIES DEFINITION

Based on these concepts, subsystems were identified and incorporated to the proposed framework. These subsystems include typical functionalities in HBAS, such as cooling and heating, luminosity control, access control, and so on. Similar functionalities are grouped together and form subsystems.

Subsystems deal with different variables, corresponding to important information for the HBAS, which are related to environment under control (temperature, luminosity, occupation, energy consume, etc.), timing information (date, timetable, intervals, delays, etc.) or that can be related to users (login, notification, configuration, etc.).

These variables have specific attributes in conformity with their type (resolution, value band, etc.) and they can be manipulated differently according to functionalities associated with each variable and also according to control strategy adopted for the environment.

The necessary subsystems to implement a HBAS will depend on the functionalities to be controlled. In the present framework the following subsystems are defined:

- Supervisory: handles all activities related to user interacting (login, notification, configuration, etc) and also to data storage and recovery (configuration, events, alarms);
- Lighting control: allows the environmental lighting control for different use conditions;
- Intrusion control: responsible for identifying unauthorized access to locations and appliances;
- HVAC control: allows the control of temperature, humidity and ventilation in a environment, according to predefined conditions;
- Access control: responsible for users identification and authentication in order to allow access to restrict areas;
- Blind control: responsible for controlling of curtains, shutters, gates, etc;
- Fire controls: it allows control the environmental conditions relatively to fire control (high temperature, smoke, etc.) in order to prevent people and installations to be put at risk.

UML Use Case diagrams were created for all subsystems with their functionalities.

4. METAMODEL DEFINITION

The next modelling stage corresponds to specifying the objects which will compose each subsystem and how they relate to each other. It is necessary to define the concepts and structures which will be used to represent and store relevant information to a HBAS. This stage corresponds to information view in the RM-ODP.

The main concept adopted to represent the information of HBAS is logical device. A logical device corresponds to a device that represents a logical functionality. It has typical attributes according to this functionality, regardless of how it will be implemented afterwards.

A class hierarchy was constructed from the identification of usual objects among different studied protocols (table 1) and the analysis of typical building and home variables (functionalities).

The root of the model, according to figure 1, are the classes LogicalDevice, Subsystem and SimpleDevice. They implement the pattern Composite [9]. This allows to represent a system with any number of subsystems and also to model a subsystem with any number of devices.

The type definition proposed in this framework is done according to data types defined by XML (W3C, 2005). This standard is independent from the programming language. The choice of methods and attributes was done according to the definition of the attributes and methods of open protocols, previously mentioned.

A SimpleDevice may have one or more logical connections. Different types of connections, according to the definitions of the RM-ODP, can be represented with the specialization of the class Channel.

The SimpleDevice class was specialized in order to allow the modelling of different functionalities that represent the HBAS. This class was specialized in the classes ProcessInterface, Controller, UserInterface, ComunicationInterface and Persistent, each one representing a set of typical functionalities of the application.

The ProcessInterface class has the goal of modelling entities related to the physical process. It was specialized in the class Sensor, that is responsible for representing the process input variables and Actuator to represent the process output variables. The Sensor class was specialized in BinarySensor, AnalogSensor e MultistateSensor and the class Actuator in BinaryActuator, AnalogActuator and MultistateActuator.

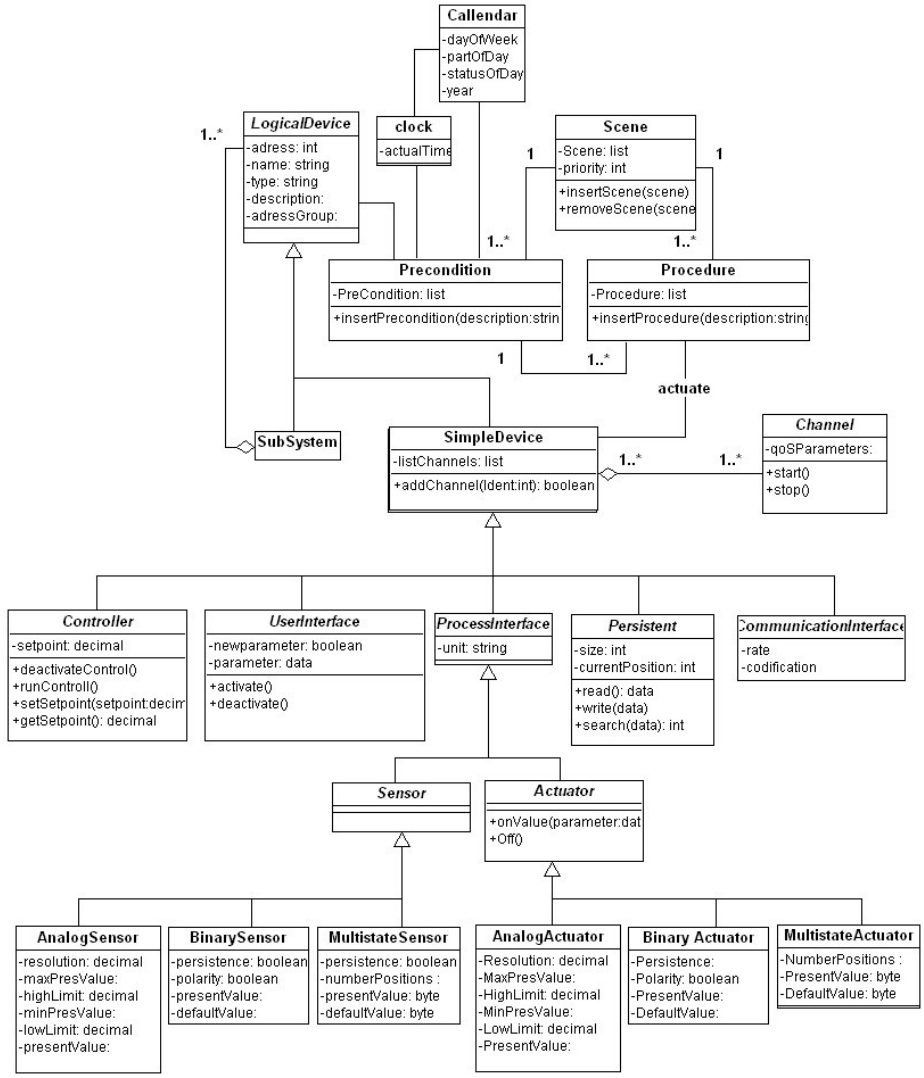

Fig 1 – General view of the Framework's classes.

The class Controller allows the modelling of process control functions. The class UserInterface represents different man-machine interfaces, which can be a data input, (such as a keyboard, a mouse or a microphone) or a data output (such as a display, a speaker or a video).

The class Persistent has the goal of modelling persistent data, such as need for user authentication (login, password, etc), music, videos, etc.

Finally, the last class CommunicationInterface has the goal of representing network components. Specializations of this class are used to model the network as a subsystem, using the same concepts which were used for device modelling.

Other typical characteristic of HBAS is that they may execute pre-determined procedures according to a set of preconditions previously established. In order to handle these functionalities, the concept of scene was adopted and the classes Scene, Precondition e Procedure besides the classes Calendar and Clock were specified. Scenery represented by the class Scene, associates preconditions that must be valid to activate the scene and procedures that will be activated when the scene is active. In order to solve the possible conflicting problem among different active scenarios, priority layers for different systems have been defined, using a similar concept to the adopted by BACnet.

5. THE DEFINITION OF LOGICAL DEVICES AND THEIR INTERACTION

In the next view of the system the different types of objects which compose each subsystem and their interaction should be specified. This stage is defined as computation viewpoint in RM-ODP. In this framework, the computational objects are called logical devices.

Figure 2 presents the logical devices, which were defined for HVAC subsystem. These devices were chosen according to functionalities, which need to be represented, and they correspond to instances of the classes defined in the previous stage.

The multiplicity adopted in the composition allows representing the systems with different complexity. In this step, the type and the direction of the information flow among the computational objects should also be defined. UML Collaboration diagrams were defined for all subsystems in order to represent the relationship among the logical devices specified for each subsystem.

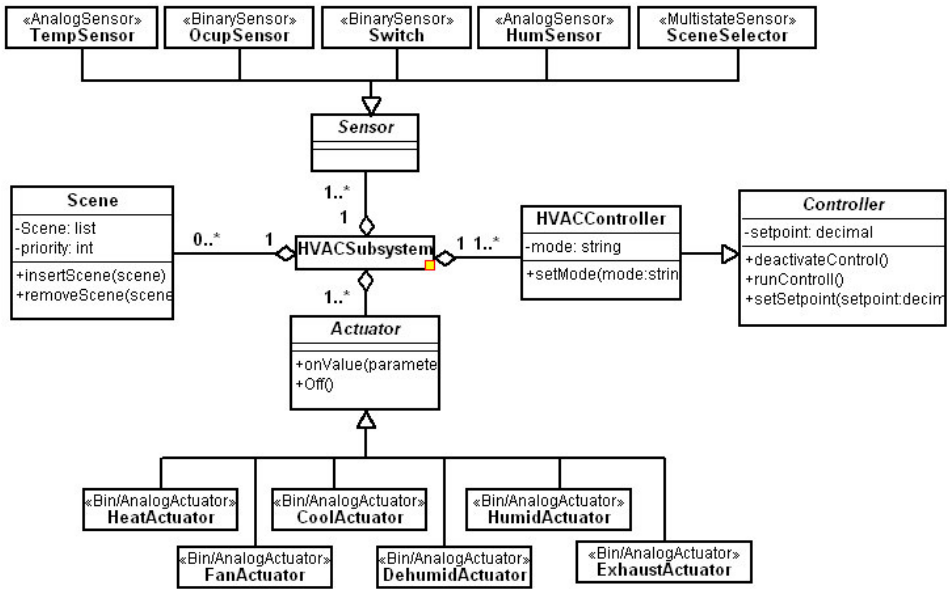

Fig 2 – Logical Devices of the HVAC

6. TECHNOLOGICAL MAPPING

In the next view of RM-ODP, the computational objects are mapped to engineering objects and the mechanisms which will provide transparency are defined. In this work a new technology will not be specified, so the modelling of concepts of this view is unnecessary because the target technology will provide this support.

Then at fourth and last modelling stage, each of the logical devices, defined in the previous stage, will be mapped to physical devices for the target technology, available for implementing the HBAS. Each physical device may contain one or more logical devices, according to functionalities supported by the chosen hardware and it will receive a physical and a network location.

In order to enable this mapping, physical devices of different suppliers should have their functionalities mapped, according to the concepts proposed in this framework, to compose a device library. Based on this library the device that better matches the desired functionality should be selected.

The logical connection among logical devices will correspond to an internal connection when the objects are inside a same node, or it will correspond to a network connection when it is made among different physical devices.

The final result will be the physical model of a HBAS. The configuration tool will then have to import this specification and to automatically configure the final system. In order to ease this task, a XML files describing appliances information are adopted.

7 DESIGN METHODOLOGY SUPPORTED BY PROPOSED FRAMEWORK

The use of the framework previously specified divides the design of a HBAS into three main steps.

The first step corresponds to problem domain analysis and requirements specification. In this stage, the desired functionalities and system's behaviour are specified based on user requirements. It is necessary to concentrate on "*what*" the system should execute and not on "*how*" this should be done.

The second step deals with logical design. The functionalities and the behavior previously specified are mapped to functionalities of the subsystem specified in the framework and the necessary logical devices to represent these subsystems are selected. Figure 3 depicts this stage.

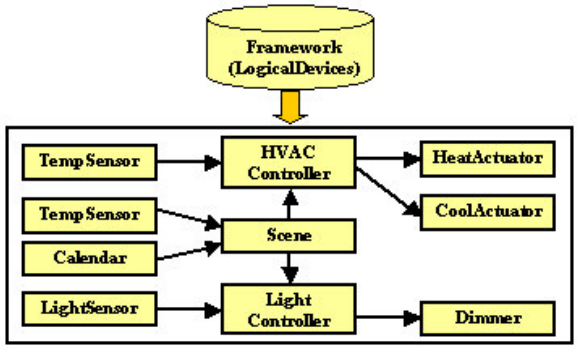

Fig. 3 – Logical Design: selection of logical devices.

In the third step logical devices are mapped to physical devices which support the functionalities specified and logical connections among devices are mapped to physical connections according to specifics technologies available for implementation. The physical devices must be located in the network and the physical environment. Figures 4 and 5 represent this stage.

8. CONCLUSIONS AND FUTURE WORKS

When specifying the framework proposed in this study, the main goal was to get the definition of a architecture on which objects were representative of the necessary functionalities to HBAS, independently of the technology that would be used in the implementation of the HBAS.

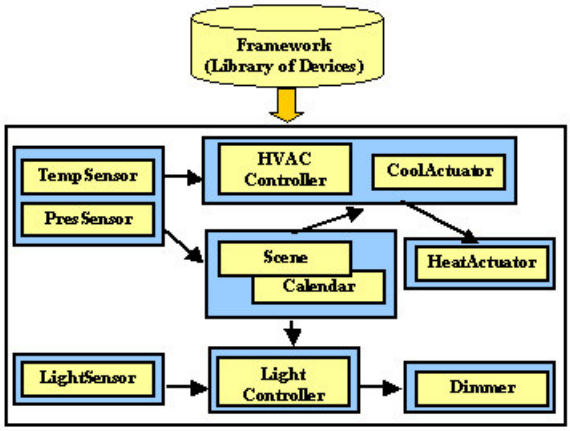

Fig. 4 – Grouping Logical Devices

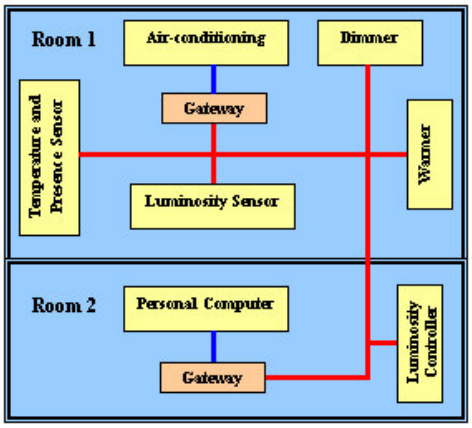

Fig. 5 – Physical and network Localization

Different case studies were developed in order to validate the proposed methodology. One of the case studies deals with the integration of sub-systems from different vendors: (i) a HBAS controller with light and temperature control modules produced by the company Homeystems (HOMESYSTEMS, 2005), (ii) a HVAC sub-system manufactured by Springer-Carrier and (iii) an image processing system for automatic recognition of car licenses to perform access control. All these sub-systems are based on distinct communication and embedded hardware/software architectures and the framework should minimize integration problems due to low level implementation details by using the proposed object models.

So far, the proposed class library has proven to be very useful to represent (at an abstract while yet well defined level) the functionalities and structure of the distinct sub-systems. Preliminary results have been quite encouraging, since the proposed class hierarchy allowed an easy mapping of concepts handled by the different tools, easing the identification of which modules should be modified in order to allow integration. Moreover, by using the proposed class library, missing concepts in the HomeSystems were identified and were incorporated by system designers, which recognize the usefulness of these concepts for HBAS. A software tool to support the framework proposed in this paper is under implementation. In the first stage it may allow the representation of logical design in XML files. Afterwards it will implement filters to transform this file to a format that can be imported by configuration tools of a target technology.

REFERENCES

SWAN, Bill. *The Language of BACnet®-Objects, Properties and Services*. Alerton Technologies Inc.13p.<http://www.gopolar.com/BACnet/ articles.html>

EIBA. *Introduction to the System* (1999) European Installation Bus Association.. 36p <http://www.georg-luber.onlinehome.de/data/ introduc.pdf >

ECHELON CORPORATION. ***Introduction to the LONWORKS System***. (1999). <http://www.echelon.com/support/documentation>

CIC. *HomePnP™ Specification*. (1998) CEBus Industry Council.. 369p. <http://www.cebus.org/Files/ hpnp10.zip>

CHO, Yean Song. *Framework for the Composition of the Home Appliances based on the Heterogeneous Middleware in Residential Networks*. (2002) In: IEEE on Consumer Electronics, 3, pp. 484-489.

MOON, Kyeong-Deok; LEE, Young-Hee; SON, Young-Sung. et. all. *Universal Home Network Middleware Guaranteeing Seamless Interoperability among the Heterogeneous Home Network Middleware*. (2002) In: IEEE on Consumer Electronics, 3,. pp. 546-553.

EMBASSI. *The EMBASSI-project.* <http://www.embassi.de/ewas/ewas_frame.html>

DYNAMITE. *The dynAMITE project.* <http://www.igd.fhg.de/igd-a1/amiatini/ projects/self_organization/dynamite.html>.

GAMMA, E.; HELM, R.; JOHNSON, R. ett all. *Design Patterns.* (2000) Addison-Wesley. 364 p.

ISO/IEC 10746. *Open Distributed Processing - Reference Model.* (1995) International Standard..

MICROSOFT. *Universal Plug and Play Device Architecture.* (2000) Redmond. <http://www.upnp.org/resources/documents.asp>

W3C. *XML Schema Part 2: Datatypes*. (2001) The World Wide Web Consortium Recommendation. 137p. <http://www.w3.org/TR/xmlschema-2/>.

HOMESYSTEMS .<http://www.homsystems.com.br.