5

A framework for simulating home control networks

Rafael J. Valdivieso-Sarabia, Jorge Azorín-López,
Andrés Fuster-Guilló and Juan M. García-Chamizo
University of Alicante
Spain

1. Introduction

Trasgu[1] is a control networks design environment valid for digital home or other places. The introduction of services provided by information society technologies, especially control networks, is growing at business, buildings, houses… There are a high number of protocols and technologies available in control networks. The set of control technologies that makes the applications viable are diverse and each follows his own rules. For example, there are different standard for control technologies like X10 (Fuster & Azorín, 2005), KNX/EIB (Haenselmann et al., 2007), LonWorks (Ming et al., 2007) , CAN (Jung et al., 2005), Zigbee (Pan & Tseng, 2007), etc and owned technologies like Bticino, Vantage, X2D,... In spite of standardization attempt, the design and implementation of control facilities is complex. Every technology presents a few limitations. We find among them the own configuration tool. It is proprietary software that allows the network design and configuration. Proprietary software, provided by the supplier, is the main design and configuration tool for control networks. Tools used by technologies considered as automation networks standard are: European Installation Bus Tool Software (ETS) for Konnex (KNX), and LonMaker Integration Tool (LonMaker) for Lonworks. Both tools have the same purpose, but they have different design and configuration methodology. A design realized with any tool is not compatible with other one and in many cases they cannot be linked. This has repercussions on increase of time and cost when the design needs several technologies that must coexist to offer a higher service. Technology choice depends of user requirements, because it might not be solved as well with all technologies. Even there might be project whose requirements are unsolved with a single technology, so we will need some technologies integration. In many situations, it turns out complex to integrate them in a common system. In spite of it, we need to communicate them to provide higher services, even if they belong to different networks: control networks, information networks, multimedia networks and security networks. There are residential gateways based on middleware and discovery protocols to make integration task easier.

[1] Trasgu is Asturian (Spanish) mythology goblin. It realizes household chores, but if it gets angry, he will break and hide objects, he will shout, etc.

Residential gateway middleware is connectivity software that allows communication among different technologies, in this case control technologies. Common middleware in ICT are: CORBA, J2EE y .Net. J2EE and .Net are the most used, the first one is based on Java and the second one is based on different Microsoft programming languages: C#, Java#, Visual Basic .Net,...

Discovery protocols facilitate devices connection to networks and services negotiation: Universal Plug and Play (UPnP) (Rhee et al., 2004), Jini Network Technology (Sun, 1999), Home Audio/Video Interoperability (Havi), Open Service Gateway initiative (OSGi) (Kawamura & Maeomichi, 2004), Services and Network for Domotics Applications (SENDA) (Moya & López, 2002)... UPnP is an open architecture distributed on a network, which is independent from technology. The goal is to get an unattended connection among different devices technology. Jini gets infrastructure to federate services in a distributed system. It is based on Java programming language. Jini has not been successful because there are not many devices supporting Jini. Havi is architecture oriented to electrodomestic appliance. The objective is to get a services set to make interoperability and distribuited networks development in home easier. OSGi defines a open and scalable framework to execute services in a safe way. The goal is to join devices in a heterogeneus network at aplication layer in order to compone services. SENDA framework is a device networking that uses CORBA. The aim of SENDA is similar to OSGi philosophy although SENDA is trying to improve it. There are some technologies to integrate control network, but none of them is being clearly succesful in all contexts, although some of them have a little market share.

Independently from integration there is another problem in actual control networks. Once we have designed a control network according to user requiments, we must realize the network installation to validate the correct operation. This fact introduces high temporal and economical costs in the installation, because if designer detects a fault, he should to solve it in the real facilities. This situation can be avoided through a simulation task after designing and before network installing, but control networks owned tools do not allow realizing simulations. They manage to realize validations to low level. For example, ETS realizes validations with physical and group addresses assigned to the devices. There is a tool associated with the ETS called EIB Interworking Test Tool, EITT. It is specialized in analysis of devices, moreover offers the possibility of simulating the network protocols. In spite of the low level validations that these tools realize, none of them manages to realize a simulation of the control network behaviour. A consequence is that designer cannot be able to verify the correct functioning, until control network has been implemented and installed.

Simulation brings advantages in the design of control installation. Simulation as a tool in the development techno-scientist allows detect errors prematurely in the design phase (Denning, 1989). It is able to verify and validate the control network design (Balci, 1998) in order to reduce costs. Simulation has the same place in design phase, that testing at implementation phase, since in both cases checks are made on the work performed. Therefore simulation is seen as a test case (Norton & Suppe, 2001), where checks are made to a higher level of abstraction. Besides the professional tools, in the literature there are some control network simulators like VINT project (Breslau, 2000), DOMOSIM (Bravo et al., 2000), (Bravo et al., 2006), VISIR (González et al. 2001) and (Conte, 2007). The VINT project presents a common simulator containing a large set of networks model. It is composed by a simulator base and a visualization tool. It is focused for simulation of TCP/IP network

protocols for research in TCP behaviour, multicast transport, multimedia, protocols response to topology changes, application level protocols, etc.
DOMOSIM and VISIR are orientated to educational area, so his principal use is teaching methodology of design of facilities. The negative aspect is the disability to join with tools of design that are used in the professional environment. (Conte, 2007) presents a study of home automation systems through simulation/emulation environment. It modelling home automation system like agents, where each agent action is characterized with cost, duration and quality. The agent behaviour is modelled with a set of states and a set of rules. The simulator is a software environment and agents are implemented in LabView and LabWindows CVI and are executed at same time. To run a simulation, user has to define the virtual environment and execute it. This is a powerful simulation environment but is not integrated in designing environment, so we must design the control network in the owned tool and later we must re-design at simulation environment.
Control networks carry a high temporary and economic cost. High cost due to three factors principally: Requirements of integration different technologies, inappropriate designing methodologies and design validation by means of experimentation. The proposal gathered in this chapter is to provide an environment to provide control network architectures in the digital home. The objective is that these architectures can be valid for any technology, paying special attention to network simulation task.

2. Modelling control systems

Modelling systems are based principally in two types of methodologies: bottom-up methodologies and top-down methodologies (Sommerville, 2004).
The technologies and methods used for the modelling of home automated systems are few developed. They are based on the use of very low level technologies (Muñoz et al., 2004). The great diversity of control technologies causes that designing control networks following bottom-up methodologies might turn out a complex task and might generate systems inadequate to requirements.
The top-down methodologies are characterized essentially abstract. They require that ingenuities are conceived before any consideration. Ingenuities are conceived, therefore, free from any condition imposed by the technologies. These methodologies match perfectly with our philosophy of design independent implementation technologies.
Model Driven Architecture (MDA) (Mellor et al., 2004) and Services Oriented Architectures (SOA) (Newcomer & Lomow, 2005) are within independent implementation philosophy. In one hand MDA allows transitions from conceptual models of systems using automatic code generation is possible to obtain fast and efficient middleware solutions. In the other hand SOA provides a methodology and framework for documenting business skills and power to support the consolidation and integration activities. The network nodes make its resources available to other participants in the network as independent services that have access to a standardized way. In contrast to object-oriented architectures, SOAs are formed by application services weakly coupled and highly interoperable. The communication between these services is based on a formal definition platform independent and programming language. The interface definition's encapsulates the peculiarities of an implementation, which makes it independent of the technology, the programming language or developer.
Designing control networks, with this methodology, require answer the following questions:

What functionalities I want to offer? E.g. for following functionalities: safety, comfort and automation, we have to provide the following services: intrusion detection, access control, alarms, lighting and temperature control.
How should behave services? That is, how relate them. In the case of the above examples: we should indicate how want to regulate temperature or the relationships with other devices: intrusion detection, alarms.
What technologies I'm going to use for implement the system? Once we have clear functionalities that network is going to offer and the behaviour, we have to decide among available technologies to choose the best for this situation.
We propose a model based on the three above questions. The model consists in three layers called: functional, structural and technological.
Functional is the most abstraction layer. It describes installation functionalities. It avoids thinking about how to do this and technology implementation. So we got translate user requirements to the functionalities that the installation will provide. In this layer, the control installation, *CI*, is defined by a set of services, *Si*, which are demanded by users:

$$CI = \{S_1, S_2, \ldots, S_n\} \tag{1}$$

Each service, *Si*, needs to satisfy a set of tasks, *ti:*

$$S_i = \{t_{i1}, t_{i2}, \ldots, t_{in}\} \tag{2}$$

The next level of abstraction is called structural. It is focused on the structure and behaviour of the generic devices. Since the structural layer, the control installation, *CI*, is composed of a set of generic resources, *Rs*, and a wide range of connections, *C*, which are established between resources:

$$\begin{aligned} CI &= CI(Rs, C), \\ Rs &= \{Rs_1, Rs_2, \ldots, Rs_n\}, \\ C &= \{C_1, C_2, \ldots, C_m\} \end{aligned} \tag{3}$$

A resource represents the entity that may be physical or logical and provides some tasks, *tij*. At this level, both the resources and connections are independent of technology, because it only represents the installation structure. Resources are represented as a set of tasks, *tij*, offered to other entities:

$$RS_i = RS_i\{t_{i1}, t_{i2}, \ldots, t_{im}\} \tag{4}$$

Connections are represented as associations between two resources. They are formed by an input resource, *Rsi*, and an output resource, *Rso*:

$$C_i = C_i(Rs_i, Rs_o) \tag{5}$$

The lower abstraction layer is technological. It is an instance of structural layer using a specific technology. It takes into account more realistic aspects of the implementation. Resources viewed from the technological layer are defined by the set of tasks, *Ti,* and by his set of physical characteristics, *CAi*, like position, size, weight, etc:

$$\begin{aligned} R_i &= R_i(T_i, CA_i), \\ CA_i &= \{ca_{i1}, ca_{i2}, \ldots, ca_{ip}\} \end{aligned} \tag{6}$$

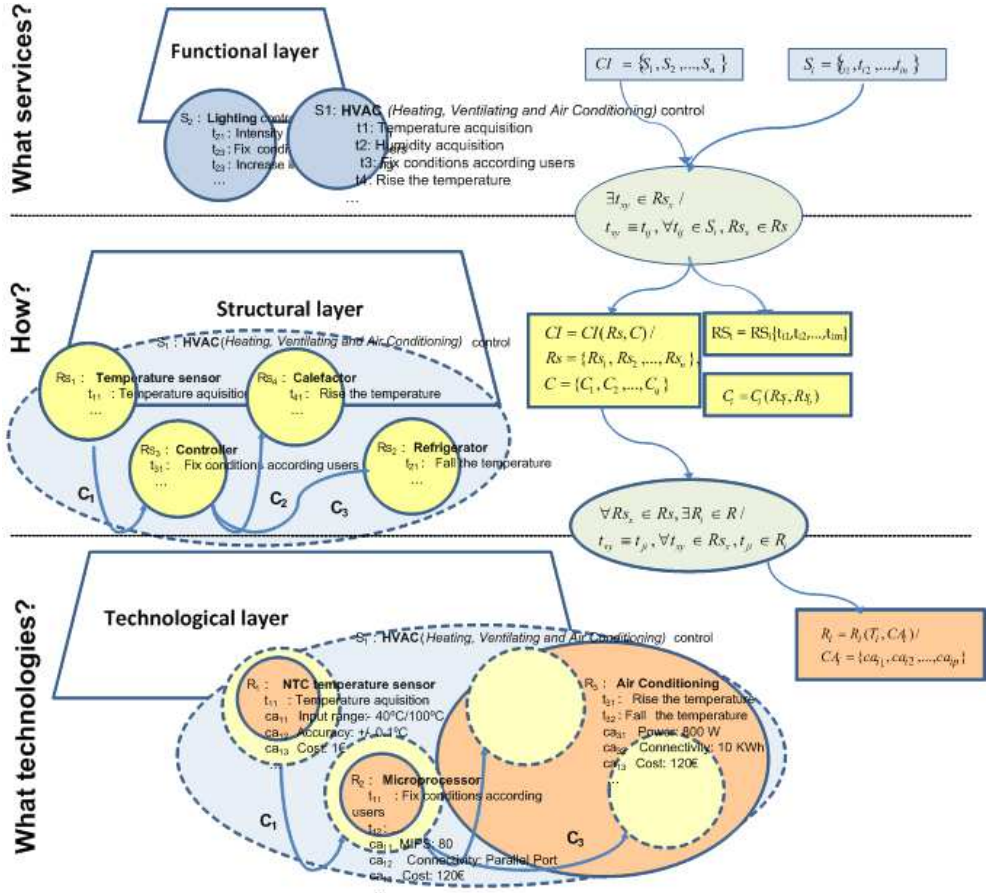

Fig. 1. Correspondence between layers and model equations

At this layer, resources, *Ri*, tasks, *Ti*, and characteristics, *CAi*, are determined by implementation technologies. This representation reflects reality faithfully.

As the level of abstraction is reduced, it is necessary to make transitions from functional to structural and, finally to technological level. To achieve the first transition, services should have all their tasks paired with some of tasks provided by resources of the technological level. Therefore task, *txy*, of any resource, *Rsx*, should be equivalent to task, tij, required by service *Si*. This must be satisfied for all tasks, *tij*, required by the service, *Si*:

$$\exists\ t_{xy} \in Rs_x\ /\ t_{xy} \equiv t_{ij},\ \forall\ t_{ij} \in S_i,\ Rs_x \in Rs \qquad (7)$$

Structural to technological level transition needs to match generic resources *Rs* with technological resources *Ri*. Therefore all tasks, *txy*, at all generic resources, *Rsx*, should be matched with task, *tij*, of technological resources *Ri*. This matching must be between equivalent tasks:

$$\forall\ Rs_x \in Rs,\ \exists R_i \in R\ /\ t_{xy} \equiv t_{ji},\ \forall t_{xy} \in Rs_x,\ t_{ji} \in R_i \qquad (8)$$

When these transitions have been performed successfully, the control network will be defined in three abstraction levels: functional, structural and technological. Figure 1 shows the correspondence between layers and model equations using a Heating Ventilating and Air Conditioning (HVAC) service as example. It defines the functional level a set of tasks like: temperature acquisition, humidity acquisition, fixing conditions according users, rises the temperature, falls the temperature etc. The structural level is defined by resources that implements the previous tasks like: temperature sensor, which implements temperature acquisition, controller, which implements fixing conditions according users, calefactory which rises the temperature and refrigerator which falls the temperature. Finally the technological level defines what technology implements the resources defined at the structural level. In this case a NTC temperature sensor has been chosen as temperature sensor. The temperature acquisition task and can be defined by characteristics like: input range, accuracy and cost. Microprocessor was chosen as controller. It has fixing conditions according user task and it is defined by characteristics like: MIPS, connectivity and cost. Finally, the air-conditioning implements two tasks: rise the temperature and fall the temperature and it is defined by power, connectivity, cost, etc.

3. Trasgu: Design environment

Trasgu is a prototyping environment based on the previously defined model. It can be used from three different abstraction levels: functional, structural and technological. The installation can be defined from a higher level of abstraction and choose specific implementation aspects in the last steps. It provides robustness design and reduction of development time, because transitions from the highest abstraction level to the lowest are progressive and simple. It facilitates finishing all tasks that are needed throughout the life cycle of control network. These tasks will be used by their respective actors who are involved in a control installation: designers, integrators, property developers, installation engineer...

The model has power enough for developing features that the environment offers. Environment provides the following features: designing a control installation independently any technology; perform simulations to verify and validate that our design meets users specifications; create architecture in agreement to the chosen technology for implementation; make a budget with each technology in order to determine the cost of each one; develop different types of reports, including a report showing the wiring installation. Nowadays tasks implemented by trasgu are: designing control network and simulation task. Figure 2 shows possible users for each environment feature, e. g. Prescriber will be in charge of designing the control network and simulate it in order to validate initial decisions, programmers will be in charge of modelling architecture with implementation technology and simulate it. On the other hand, promoters will be in charge of budgets and installer will be in charge of installation reports.

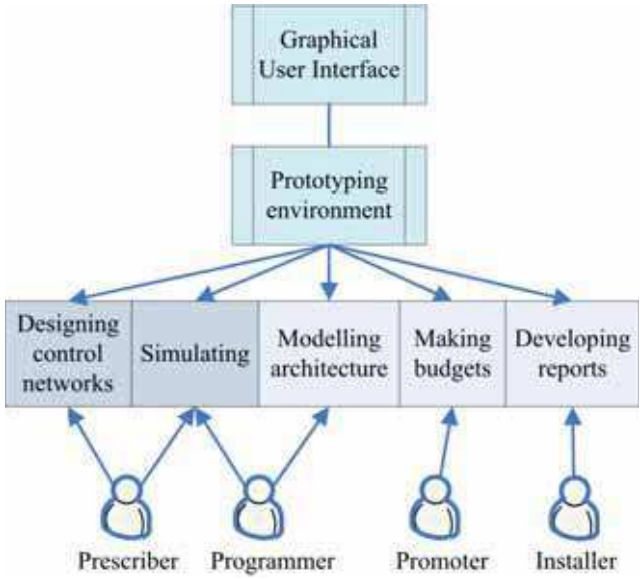

Fig. 2. Environment features and their possible users

The control network design is the first task. This task is characterized by adding some resources into a building plan. Resources have been classified into three types: input, Eq. (4), services, Eq. (2) and output, Eq. (4). First of all, services must have been added to the building floor. Then we have to add the input and output resources and connect with their corresponding services. All resources are connected to one or more services that are controlling their network behaviour. Representation based on services has been chosen to represent control network because this representation reflects that input and output resources can be used by several services. Figure 3 shows an example of three services: security, lighting and gas detection. The presence sensor and binary light are shared with security and lighting services. Gas detection service uses a gas sensor and acoustic alarm. And finally acoustic alarm is shared with security service.

The simulation task is based on design of the control network. Control network can be simulated specifying simulation parameters. This task probes that design is satisfied with specification.

The architecture modelling task allows generating the architecture that implements the design in a real installation. The architecture is middleware based (Valdivieso et al., 2007), (Fuster et al, 2005). This middleware provides communication among different control technologies and protocols.

The budget creation task allows generate budgets from the technological design. It will provide an estimate cost for the costumer. This task requires an automatic communication with the devices dealer for database maintenance.

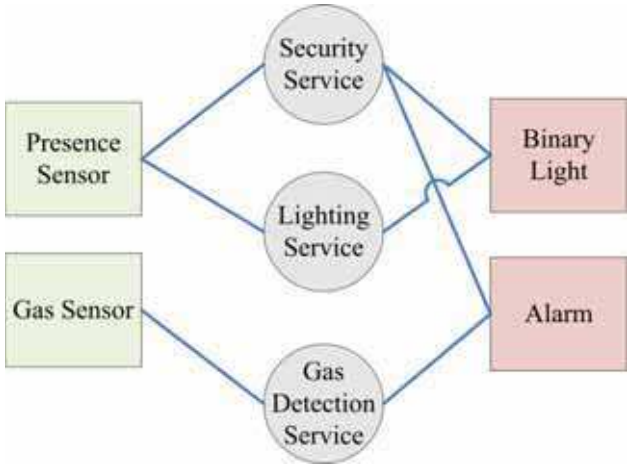

Fig. 3. Representation based on services

The reports development task allows generating some types of reports. Each of them is focused to show an installation feature like: wired connections, devices situation, device configuration, services…
Environment features can be used from different abstraction layers. Figure 4 shows relationships between environment features and abstraction layer, in which are implemented. The installation design could be used from functional, structural and technological layers. Installation simulation can be used from functional, structural and technological layers. Architecture modelling only supports the structural and technological layer. It is technological because it is implemented over real devices and it is structural because some architecture layers could be common. Making budgets is useful only from technological layer because the budgets are for customers. Instead reports can be created from the functional, structural and technological layers, because reports can be useful for different applications. The customer could need a services report, a developer could need a structural report and an installation engineer could need a wired reports.

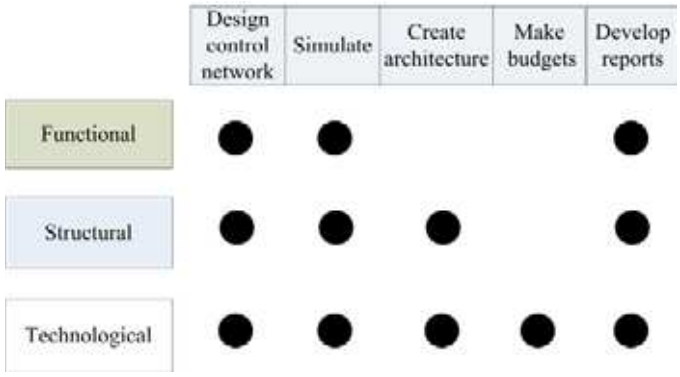

Fig. 4. Abstraction layers supported by each feature

3.1 Resources definition

Resources of functional level are classified according to input and output resources. At functional level resources are classified in order to determine subcategories inside input and output resources.

Input resources are also subdivided to continuous and discrete sensors. Each family sensor is associated to a single magnitude to measure. Continuous sensors are those which measures continuous magnitudes like temperature, humidity, lighting, etc. Discrete sensors are those which detect events like: gas escape, presence sensors, broken windows, etc. These sensors indicate that an event has happened.

Output resources are subdivided into two groups: continuous and discrete actuators. Continuous actuators are characterized because they feedback continuous sensors and discrete do not feedback sensors. The first type is e.g. HVAC device that modifies the magnitude measured previously by a continuous sensor. Second type is e.g. acoustic alarm which does not invoke any event. Continuous actuators interact and modify the magnitude measured by sensors. Continuous actuators are determined by the direct magnitude that they modify directly and indirect magnitude list that might be modifies in any way by the actuator, e.g. HVAC modifies directly temperature and indirectly humidity. Other aspect to be explained is the capacity of some actuators to modify the physical structure where they are installed, e.g. blind engine or window engine that modify the wall structure allowing or prevent lightening and rise or fall the temperature.

3.2 Implementation

The environment has been implemented using Java and uses an information system implemented based on XML. The XML files reflect information of resources definition that can be used in the installation and the information about the current network. This information is reflected from abstraction layers defined at the model. The resources reflected in the XML files represent a subset of the features offered by market. The environment could expand the information system adding new resources in XML files. Adding new resources is easy because we only have to define the XML files that define the new resource and interactions with magnitudes.

The prototyping environment presents four different areas, see figure 5. On the top of the window, we can see the services that can be added to the network. On the left side there are all the generic resources that can be used in the design. In the middle, there is the building floor map. In this area services and generic resources can be added for designing the control network. On the right side, there is hierarchical list of all the resources added in the network classified by resource type.

The control network design starts up opening a map of the building where user can drag and drop resources and services in order to design the network. The next action is adding services, input and output resources into the map, after doing that, user has to connect resources and services in order to establish relationships. Moreover, user has to configure parameters of input and output resources and, finally, user must determine the behaviour of each service through logical operators between input and output resources. It is not possible link inputs and outputs resources directly because it is necessary a service that link them. Figure 5 shows a control network designed at the environment. The control network contains two services: HVAC and technical security. HVAC is formed by a temperature sensor and an air-conditioning and heating. The technical security is composed by a gas

sensor and an acoustic alarm. Although this design does not share any resource, it is possible share resources into some services as we have show at figure 3.

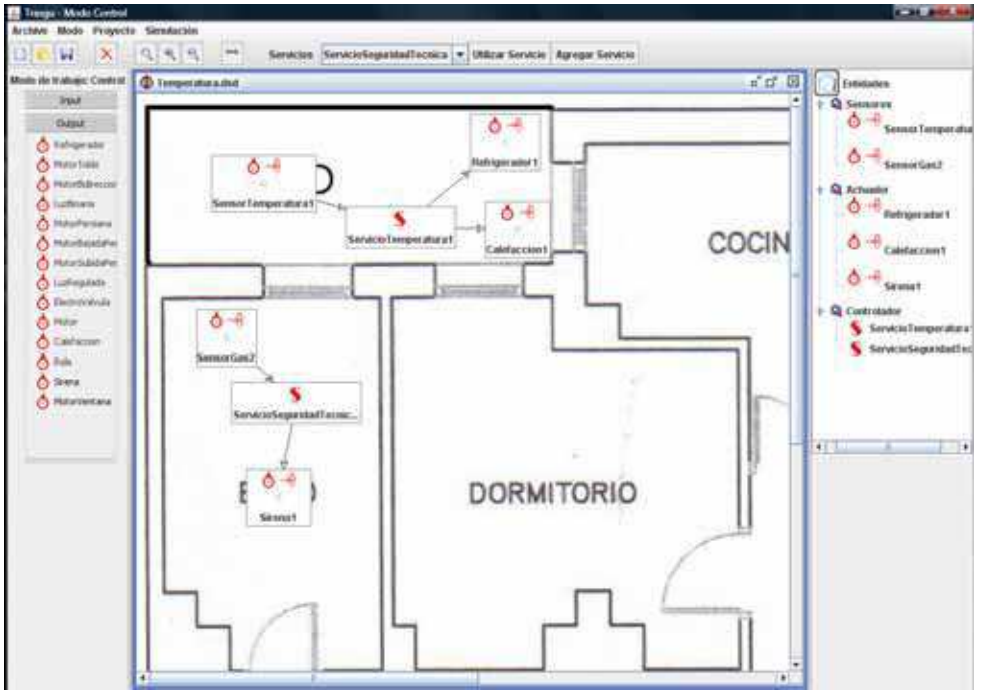

Fig. 5. Environment aspect and control network design. It contains HVAC and technical security services

Each service defines their behaviour based on some functions. These functions receive inputs from input resources and provide outputs to output resources. Functions are composed by logical, arithmetical or comparing operators. This specification contains inputs, and outputs that allows interconnect them using operators. Figure 6 shows services definition window. On the middle of the figure we can see inputs resources on the left and output resources on the right. Operators are situated on the left side of the window and it could be drag and drop into the previous area in order to link inputs and outputs. Operators should be configured in order to define the behaviour of the service. This figure contains the HVAC service behaviour. Refrigerator will be turned on when temperature rises above 25 Celsius degrees and it will be turned off when temperature falls under 25 Celsius degrees. Heating will be turn on when temperature reaches 15 Celsius degrees and it will be turn off when temperature rises above 15 Celsius degrees. According to this parameters temperature will be between 15 and 25 Celsius degrees.

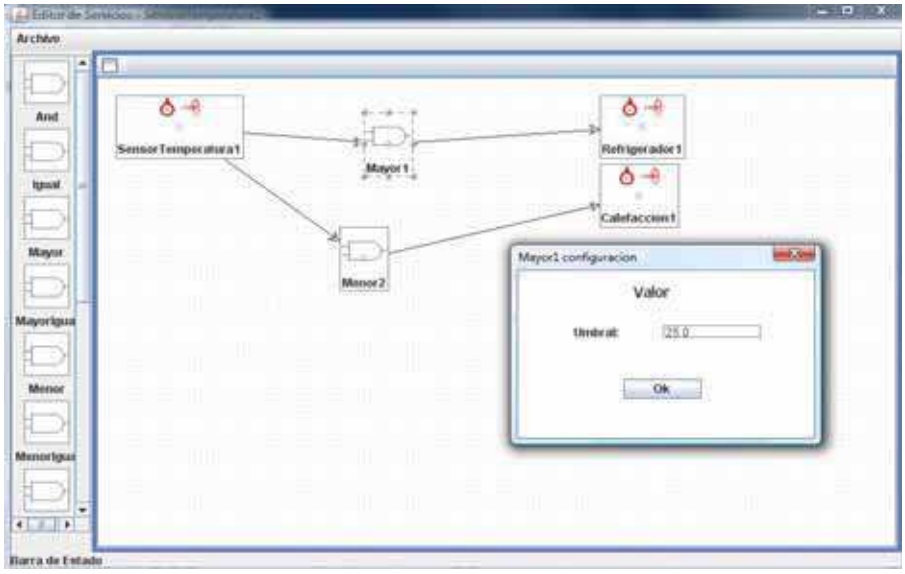

Fig. 6. HVAC performance is defined by logical operators

4. Simulation

The environment is designed to perform simulations from different abstraction levels. Environment simulates from functional layer Eq. (1), Eq. (2), structural layer Eq. (3), Eq. (4) and Eq. (5) and technological layer Eq. (6). The functional simulation is responsible for simulating the behaviour of generic control network. With this validation we can be sure that generic device configuration and connections are correct. The structural simulation includes the functional, but adds new features, like the real position of the resources and installation regulation. The technological simulation determines the real behaviour that the implemented control network is going to provide.

Functional layer simulating it is explained. This simulation is determined by two parameters: simulation time and frequency. The first one represents the interval we want to simulate. The second is the number of values from each resource that we will take in a time unit.

Simulation task needs a flag for continuous and discrete sensors. This action is carried out by event generators. We have designed two event generator types, discrete event generator and continuous event generator. The first type gives up data for discrete sensors, and is based on probability percentage, that is defined for each discrete sensor in the network. The second type represents input data for continuous sensors and is based on medium value and variation value throughout simulation time. Obtaining a sinusoidal wave which represents continuous magnitudes like temperature, humidity, etc...

The results are represented in two views. The first one shows building map, control network and resources values are drawn under their own resource. Resources values are changing at a given frequency in order to view the evolution of the control network in simulation time. This view is showed in figure 7.

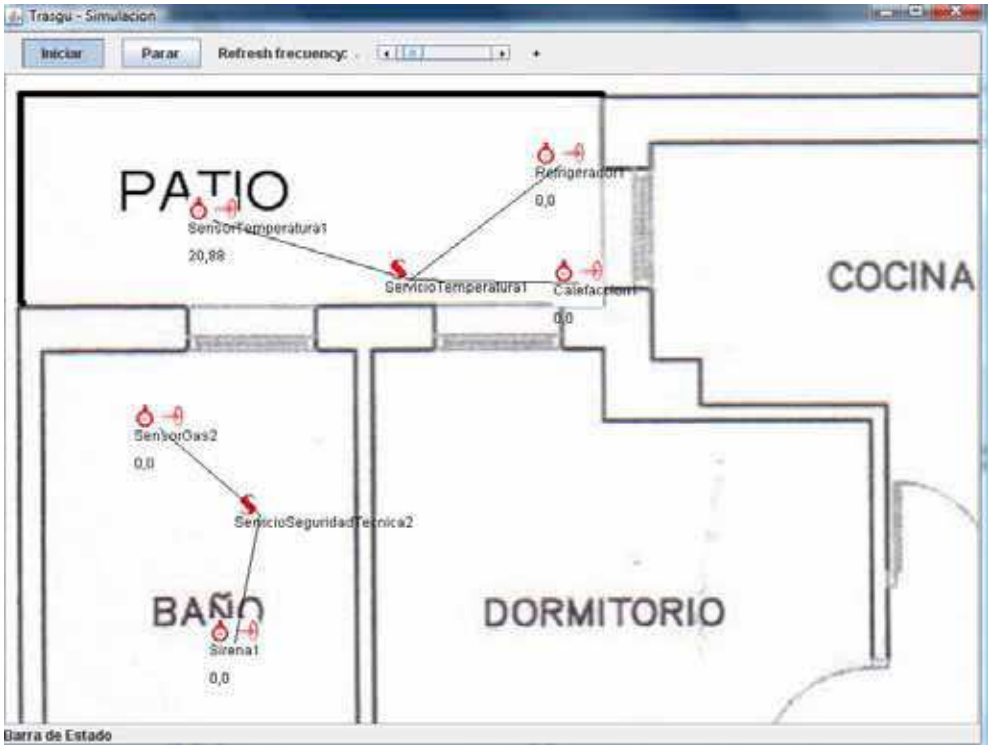

Fig. 7. Simulation results view showing values at the frequency defined at the top of the window

The second view represents a graph. A graph is showed for each resource in the previous view. The graph represents the resource behaviour in simulation time. There are three types of graphs.

The first graph type shows services. This graph shows all signals of the resources linked in the service, representing all the information in one graph, although sometimes data range are different so we have to view graphs individually at each resource. Thinking in our HVAC service example, the graph shows temperature sensor, air-conditioning and heating values. The second graph type represents sensors and shows event generators values and values given by the sensor. In HVAC example, the temperature sensor graph shows external temperature and the temperature sensor measured, which might have a little error. The third graph is reserved for actuators, and shows actuator values drawn as a single signal representing the actuator performance. In HVAC example, the air-conditioning and heating will be show in two graphs, each one shows a signal indicating when the air-conditioning is on or off. Figure 8 shows three graphs, temperature sensor at left, air-conditioning at right on top and heating at right on bottom. Temperature sensor graph shows the external temperature in blue, which is generated as a variation event, and sensor temperature in red. The red signal has peaks because the sensor has been defined with measure error. Moreover the red signal is ever between 25 and 15 Celsius degrees because the air-conditioning starts

when temperature rises and the heating starts when temperature falls. Air-conditioning and heating graph shows when they start to work.

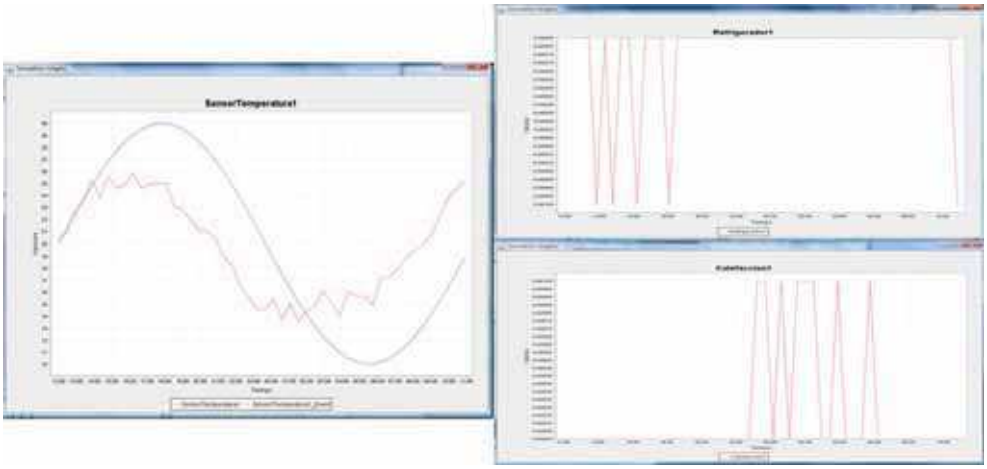

Fig. 8. Simulation results graph of temperature sensor, air-conditioning and heating

4.1 Implementation

Simulation is realized by an external module. It is implemented in external way to leave us introduce future changes in the simulation module esaily, without impact for the environment. It has been developed in Matlab / Simulink because allows high precision and efficiency.

The communication between the simulation module and environment has been done through a Java library called jMatLink (Müller & Waller, 1999). It is responsible for sending commands to Matlab. This action requires knowing Matlab commands perfectly. We have developed a library called jSCA. It is responsible for encapsulating Matlab commands to facilitate the communication. The features of jSCA are implemented as method which allows: create instances of Matlab; create, configure, and remove blocks in Simulink; configure parameters simulation, and obtain the results of the simulation. This library is a level above jMatLink. The environment make calls to jSCA library and this library calls to jMatlink library.

The architecture that achieves the communication between Java application and Simulink is showed at figure 9. The first three layers are inside the environment, but the latter two are external environment. The first layer is the own prototyping environment. The second one is the jSCA library. The third one is jMatlink library. The fourth is Matlab engine. It is a daemon that is listening commands thrown by jMatlink. We use Matlab Engine as middleware between jMatLink and Simulink. So we launch Simulink and throws commands through Matlab Engine.

The interaction between the user and the architecture layers defined previously is showed by a sequence diagram. This diagram shows temporal sequence of main calls. Figure 10 shows the sequence diagram, which corresponds to a specification language: Unified Modelling Language (UML). Temporal sequences of calls begin when user designs the

control network. The next action is to simulate it. If user starts simulation trasgu will call jMatlink layer to createMDL file. This layer starts Matlab server and calls Matlab engine to create the mdl file. Finally, Matlab engine calls Simulink and it creates the mdl file. Once mdl has been created Matlab engine calls Simulink for start simulation. When simulation finish, simulation values are returned layer after layer until trasgu received it and then trasgu visualizes it.

Fig. 9. Architecture that achieves communication among Java application and Simulink

Functional simulation requires a parallel model from the functional layer in Simulink. This model is equivalent to resources XML files defined in the information system used by the environment. Each resource defined in the environment is corresponded by another defined in Simulink. The Simulink model consists on a block set formed by input blocks, controller blocks, output-blocks and input-output blocks. There are also two block types: discrete and continuous. The discrete blocks are dedicated to digital devices, and continuous are for analogical devices.

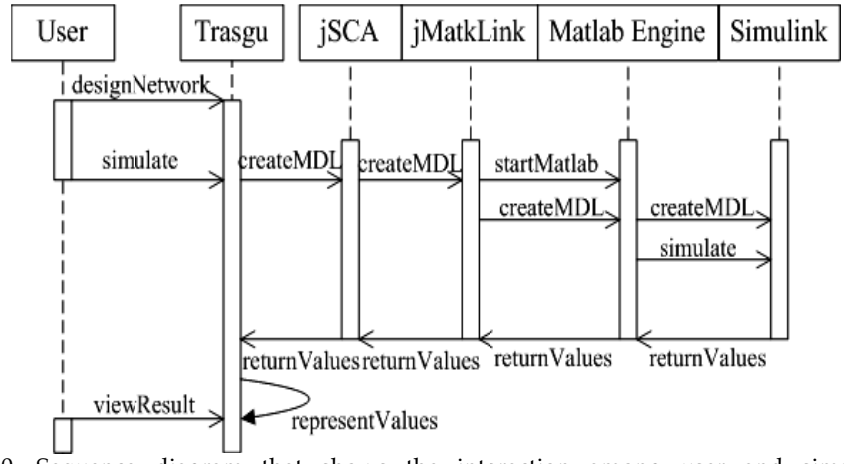

Fig. 10. Sequence diagram that shows the interaction among user and simulation architecture

The simulation starts creating an equivalent control network in Simulink dynamically. The Simulink control network is saved into mdl file. To create the mdl file, we add the corresponding network blocks. Parameters of each block are configured and connections among blocks are created. The next step is to add probability or variation events for each sensor. Through these events the input resources are activated and we can see how the network reacts. The last step is to feedback the signal from actuators to sensors, adding actuators outputs and original event, obtaining real value.
An example of this Simulink network is showed on figure 11. It shows, at section I, events generators and adding block that represent feedback. Each input resource has a correspondent event generator: discrete or continuous. Section II has input resources. Section III shows services which join input and output resources. Services definitions are made recursively by others mdl files. These mdl files are equivalent to the services defined at environment. At section IV we have output resources. Finally, section V shows feedback generated by actuators, which modify measured magnitudes. The feedback is adjusted in function of how we have configured the actuator at the environment. Each section shows a white box connected to each resource. These boxes store a value list obtained from simulation. This list contains discrete values taken at given frequency.

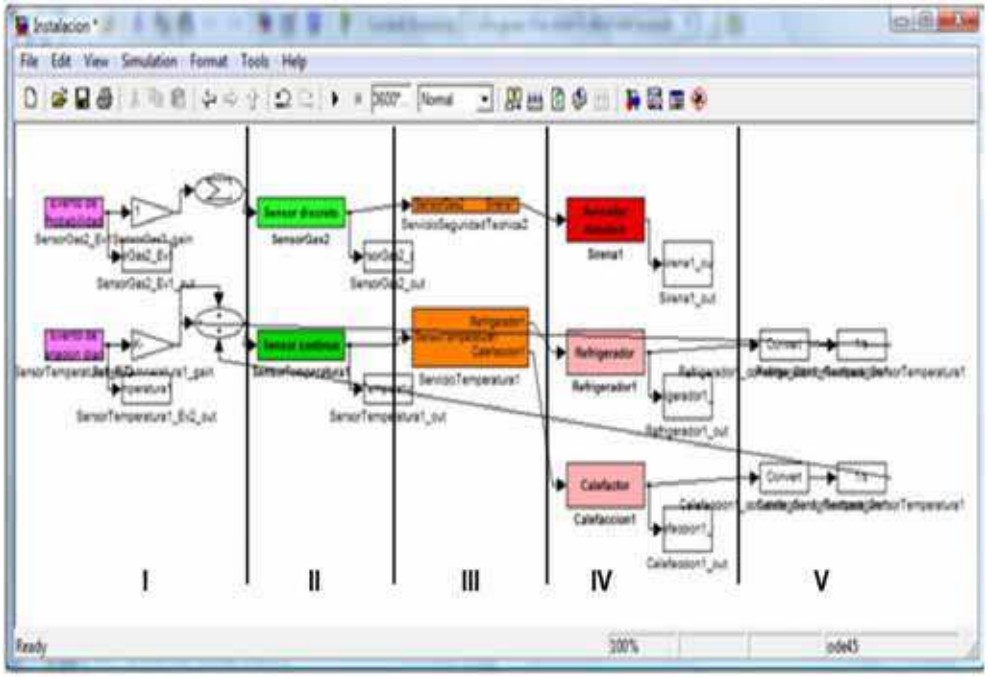

Fig. 11. Simulink control network generated dynamically from a previous design made at the environment

When the mdl file is created, Simulink executes the simulation and the environment reads all variable lists from Simulink and represents them at the given frequency in the view defined previously at environment. This way, the simulation is not interactive, when the

simulation process is launched, the user has to wait until Simulink returns the results of the simulation.

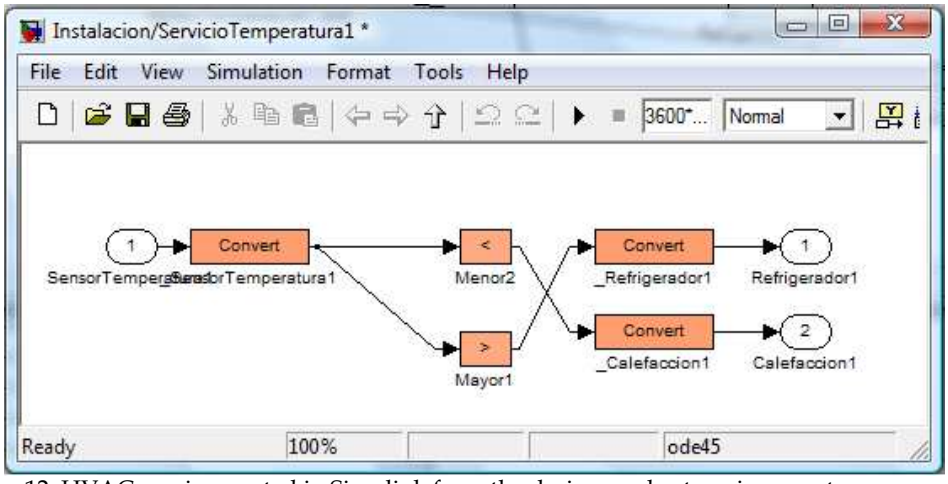

Fig. 12. HVAC service created in Simulink from the design made at environment

Figure 12 shows the mdl file that defines a HVAC service designed previously at environment on figure 6. At left side, input resources connection can be viewed as white circles, in this case temperature sensor. In the middle, we there are conversion data type blocks for avoiding Simulink data type errors and comparison operators that defines when the actuators will start to work. Actuators are drawn as white circle on the right.

5. Conclusions

This chapter presents a prototyping environment for the design and simulation of control networks in some areas: digital home, intelligent building. The objective is to facilitate the tasks of designing and validating control networks. We propose a top-down methodology, where technology implementation choice is left to the last phase of designing process. The environment kernel is the control network model around different abstraction layers: functional, structural and technological.

The model is the basis on which the environment gets applications by particularization: network design, simulation, specifying architectures, developing reports, budgets, and so on.

Simulation is presented as an intermediate phase in methodology of design control network. It reduces costs and helps us to design effective and efficient control networks.

Future work is aimed to provide competitive solutions in the design of control networks. The nature of the problem requires addressing increasingly scientific and technological aspects of the problem. Therefore, in short term, we deepening in aspects of generalization of the results: new models for structural and technological layers, an interactive simulation module and simulation of technological layer… In long term we are going to generalize to others context like industrial buildings, office buildings…

6. References

Balci, O., 1998. Verification, validation, and testing. In The Handbook of Simulation. John Wiley & Sons.

Bravo, C., Redondo, M.A., Bravo, J., and Ortega, M., 2000. DOMOSIMCOL: A Simulation Collaborative Environment for the Learning of Domotic Design. ACM SIGCSE Bulletin, Vol. 32, No. 2, 65-67.

Bravo, C., Redondo, M., Ortega, M.., and Verdejo, M.F., 2006. Collaborative environments for the learning of design: a model and a case study in Domotics. Computers & Education, Vol. 46, No. 2, 152-173.

Breslau, L. Estrin, D. Fall, K. Floyd, S. Heidemann, J. Helmy, A. Huang, P. McCanne, S. Varadhan, K. Ya Xu Haobo Yu. 2000. Advances in network simulation. Computer, Vol. 33, Issue: 5, pp: 59-67. ISSN: 0018-9162.

Conte, G. Scaradozzi, D. Perdon, A. Cesaretti, M. Morganti, G. 2007. A simulation environment for the analysis of home automation systems. Control & Automation, 2007. MED '07. Mediterranean Conference on, pp: 1-8, ISBN: 978-1-4244-1282-2

Denning, P. J., Comer, D. E., Gries, D., Michael Mulder, C., Tucker, A., Turner, A. J., Young, P. R., 1989. Computing as a discipline. Communications of the ACM, Vol.32 No.1, 9-23

Fuster, A. and Azorín, J., 2005. Arquitectura de Sistemas para el hogar digital. Hogar digital. El camino de la domótica a los ambientes inteligentesI Encuentro Interdisciplinar de Domótica 2005, pp 35-44

Fuster, A., de Miguel, G. and Azorín, J., 2005. Tecnología Middleware para pasarelas residenciales. Hogar digital. El camino de la domótica a los ambientes inteligentes.I Encuentro Interdisciplinar de Domótica 2005, 87-102.

González, V. M., Mateos, F., López, A.M., Enguita, J.M., García M.and Olaiz, R., 2001. Visir, a simulation software for domotics installations to improve laboratory training, Frontiers in Education Conference, 31st Annual Vol. 3, F4C-6-11.

Haenselmann, T., King, T., Busse, B., Effelsberg, W. and Markus Fuchs, 2007. Scriptable Sensor Network Based Home-Automation. Emerging Directions in Embedded and Ubiquitous Computing, 579-591.

Jung, J., Park, K. and Cha J., 2005. Implementation of a Network-Based Distributed System Using the CAN Protocol. Knowledge-Based Intelligent Information and Engineering Systems. Vol. 3681.

Kawamura, R. and Maeomichi, 2004. Standardization Activity of OSGi (Open Services Gateway Initiative), NTT Technical Review, Vol. 2, No. 1, 94-97.

Mellor, S., Scott K., Uhl, A. and Weise, D., 2004. MDA Distilled, Principles of Model Driven Architecture, Addison-Wesley Professional.

Min, W., Hong, Z., Guo-ping, L. and Jin-hua, S., 2007. Networked control and supervision system based on LonWorks fieldbus and Intranet/Internet. Journal of Central South University of Technology, Vol. 14, No.2, 260-265.

Moya, F. and López, J.C., 2002 .SENDA: An Alternative to OSGi for Large Scale Domotics. Networks, pp. 165-176.

Müller, S. and Waller, H., 1999. Efficient Integration of Real-Time Hardware and Web Based Services Into MATLAB. 11th European Simulation Symposium and Exhibition, 26-28.

Muñoz, J., Fons, J., Pelechano, V. and Pastor, O., 2003. Hacia el Modelado Conceptual de Sistemas Domóticos, Actas de las VIII Jornadas de Ingeniería del Software y Bases de Datos. Universidad de Alicante, 369-378

Newcomer, E. and Lomow, G., 2005. Understanding SOA with Web Services. Addison Wesley.

Norton S. and Suppe F., 2001. Why atmospheric modeling is good science. MIT Press. p. 88-133.

Pan, M. and Tseng, Y. 2007. ZigBee and Their Applications. Sensor Networks and Configuration. Ed. Springer Berlin Heidelberg, pp:349-368 ISBN:978-3-540-37364-3

Rhee, S., Yang, S., Park, S., Chun, J.and Park, J., 2004. UPnP Home Networking-Based IEEE1394 Digital Home Appliances Control. Advanced Web Technologies and Applications, Vol. 3007, 457-466.

Sommerville, I. 2004. Software Engineering, 7th ed. Boston: Addison-Wesley.

Sun Microsystems Inc, 1999. JINI Architectural Overview, Technical White Paper, 1999

Valdivieso, R.J., Sánchez, J.G., Azorín, J. and Fuster, A., 2007. Entorno para el desarrollo y simulación de arquitecturas de redes de control en el hogar. II Internacional Simposium Ubiquitous Computing and Ambient Intelligence.

Automation Control - Theory and Practice

Edited by A D Rodi

ISBN 978-953-307-039-1

Hard cover, 350 pages

Publisher InTech

Published online 01, December, 2009

Published in print edition December, 2009

The present edited book is a collection of 18 chapters written by internationally recognized experts and well-known professionals of the field. Chapters contribute to diverse facets of automation and control. The volume is organized in four parts according to the main subjects, regarding the recent advances in this field of engineering. The first thematic part of the book is devoted to automation. This includes solving of assembly line balancing problem and design of software architecture for cognitive assembling in production systems. The second part of the book concerns different aspects of modelling and control. This includes a study on modelling pollutant emission of diesel engine, development of a PLC program obtained from DEVS model, control networks for digital home, automatic control of temperature and flow in heat exchanger, and non-linear analysis and design of phase locked loops. The third part addresses issues of parameter estimation and filter design, including methods for parameters estimation, control and design of the wave digital filters. The fourth part presents new results in the intelligent control. This includes building a neural PDF strategy for hydroelectric satation simulator, intelligent network system for process control, neural generalized predictive control for industrial processes, intelligent system for forecasting, diagnosis and decision making based on neural networks and self-organizing maps, development of a smart semantic middleware for the Internet , development of appropriate AI methods in fault-tollerant control, building expert system in rotary railcar dumpers, expert system for plant asset management, and building of a image retrieval system in heterogeneous database. The content of this thematic book admirably reflects the complementary aspects of theory and practice which have taken place in the last years. Certainly, the content of this book will serve as a valuable overview of theoretical and practical methods in control and automation to those who deal with engineering and research in this field of activities.

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following:

Rafael J. Valdivieso-Sarabia, Jorge AzorÂn-Lopez, Andres Fuster-Guillo and Juan M. Garcia-Chamizo (2009). A Framework for Simulating Home Control Networks, Automation Control - Theory and Practice, A D Rodi (Ed.), ISBN: 978-953-307-039-1, InTech, Available from: http://www.intechopen.com/books/automation-control-theory-and-practice/a-framework-for-simulating-home-control-networks

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai

Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821