Pechmann Piotr, Czapiewski Piotr
West Pomeranian University of Technology

PAI/IB – Agent-Based Polish Natural Language Interface to the Control System of an Intelligent Building

Abstract: One of the key aspects of building management systems is that different components of the system could be easily and conveniently controlled. Standard solutions available in commercial building management systems use either stationary devices, with the menu of a complex structure (computer terminals), or provide access only to the local parts of the system (remote controls). Neither solution tend to be particularly convenient. An answer to the problem can be an interface based on the user's natural language queries and commands. This papers presents such a model, called PAI/IB, for Polish natural language.
Keywords: natural language interfaces, agent-based systems, building management systems

1. Introduction

Computerized building management systems (BMSs), despite their high cost, are becoming more and more common. The reason for their increasing popularity is not only greater comfort for the user when controlling the devices in the house, but also improved safety and decreasing costs of operation. These goals can be achieved by combining all the devices and systems in a building, including heating, air conditioning, lighting, monitoring and access control, home appliances, etc. into one intelligent, integrated structure.

However, in such an approach a rather complex system arises, which results in certain problems, the communication with the user being a major one. One of the basic premises of the idea of 'an intelligent house' is that everything can be controlled from one place and in an easy, convenient way. To control such a highly complex system, a significant number of options must be included in the interface, which is possible only for software operating on a computer terminal (e.g. PC computer or an LCD control panel). Such solution, however, is not very convenient. First, the interface used by that kind of software would have to have a complex multi level structure, which might be difficult to use for some users, such as the elderly, for example. Second, computer terminals as such are not particularly mobile: LCD panels are attached to the walls, and comfortable operation of even a netbook requires a place to sit.

A good solution to the problem could be using natural language as a communications medium:

- it is known to all the potential users of the system;
- it allows for easily expressing commands providing control of all the parts of an intelligent building;
- it can be used in both a local and a remote way through different, well-known channels such as voice, internet chat, e-mail or text message;
- convenient access to these channels is provided by fully mobile devices such as mobile phones or palmtops.

Although the systems presently available on the market offer a possibility of natural language communication, the range of options is quite limited. Only short and simple voice or text commands are accepted, which limits the ability for full control of all the components of the system.

In spite of the above, till today only few attempts have been made to develop solutions allowing for controlling intelligent building components using natural language commands [1, 2, 3, 9] or such that could be used for that purpose [4, 7, 8]. And those few existing projects were carried out for English language. This paper presents the results of research on such a type of a solution, with Polish natural language as a medium – the PAI/IB model.

2. PAI/IB Model

There are a few essential assumptions that form the basis for the PAI/IB model:

- The interface should constitute an intelligent agent system mediating between the user and the building management system.
- The interface should facilitate communication in Polish natural language. The system architecture, however, should make it possible to adapt the interface for using a different natural language such as English, for example.
- The interface should accept complex sentences as well as simple clauses, including gerund clauses and noun phrases.
- It should be able to recognize that the user's request is not precise enough and to obtain missing information through a dialog with the user.
- It should allow the user two way communication with the system through different channels such as speech, an internet communicator, e-mail or text messaging.
- It should be independent of the standard in which the building management system is implemented, e.g. KNX/EIB, Modbus, BACnet, etc.

The PAI/IB model designed with these assumptions in mind consists of the following three layers (fig. 1):

– subsystem for input/output channels handling (IOCH),
– subsystem for messages handling (MH),
– interface to building management system (IBMS).

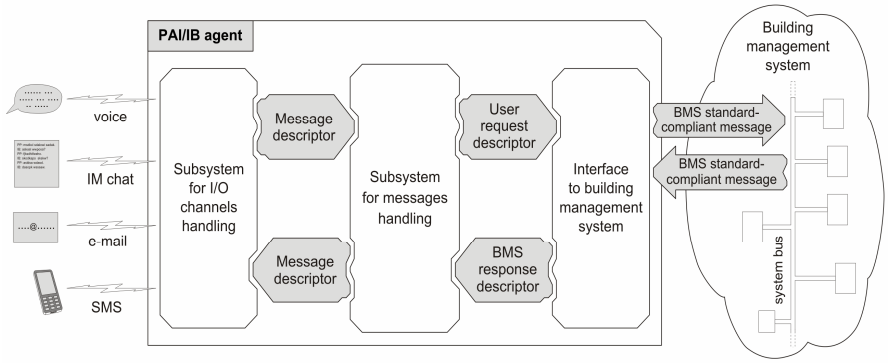

Fig. 1. PAI/IB Model structure

These layers will be discussed further in the paper.

2.1. Subsystem for input/output channels handling

The IOCH subsystem, depicted in figure 2, consists of modules handling communication channels (CCHM) and a proxy module (PM). A particular interface implementation may contain all of CCH modules or only some of them.

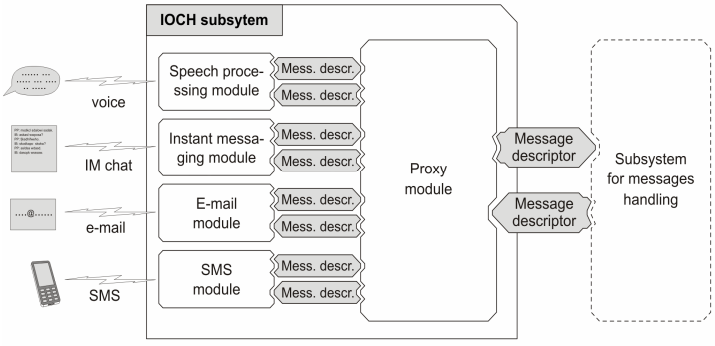

Fig. 2. The diagram of the IOCH subsystem

Each of the CCH modules cooperates with corresponding hardware-software resources in order to receive user's message and send back a response or a query from the system. For that purpose:

- The speech processing module (SPM) consist of two separate parts responsible for recognizing and generating speech. From the hardware point of view, it uses microphones and loudspeakers installed in particular rooms of the building.
- The instant messaging module (IMM) has a form of a client of one of the instant messaging networks such as AIM, Jabber or GG. It uses a standard internet connection.
- The e-mail module (EM) has a form of an appropriate e-mail system client and also uses a standard internet connection.
- The SMS module (SMSM) receives and sends text messages containing messages from the user and the BMS system. It could cooperate with a GSM modem or a cellular phone.

The purpose of all the CCH modules is basically the same:

– they extract a natural language message from the medium typical of a given communication channel;
– they wrap the natural language messages in the same medium.

The CCH modules also provide basic user identification required for controlling the communication process and providing access control to particular systems within the BMS. In the case of the voice channel the identification is based on voice characteristics, while in case of other CCH modules a communication network ID (IMM), an e-mail address (EMM) or a cellular phone number (SMSM) serves as a base for identification.

The process of receiving and sending messages by CCH modules is mediated by the proxy module, which is a kind of a common gate that CCH modules use to communicate with the second layer of PAI/IB system.

When CCH modules transfer the messages to PM they attach a structure called *source identifier* (SI) to it. This identifier is a pair consisting of channel identifier and user identifier. These identifiers are in form of character strings defined in the appropriate configuration files. The following pair:

<user_message_text, source_identifier>

is called the *message descriptor* (MD). The proxy module sends the descriptor to the second layer of the PAI/IB agent.

A corresponding process happens the other way around. The proxy module receives message descriptors form the second layer of the interface formed by a pair:

<system_message_text, source_identifier >

In this case, the source identifier is identical with the SI contained in the descriptor of that user message an answer to which is the system message. This solution enables the PM and CCH module to pass a message to the user using the same communication channel that the user first used to communicate with the system. It should be noted, however, that each of the listed CCH modules might handle a few separate communication channels bound to different e-mail addresses, telephone numbers, IM system numbers and rooms (voice channels).

2.2. Subsystem for messages handling

The MH subsystem is the key component of the entire PAI/IB model. It is also the main novel element of this solution. Its structure is depicted in fig. 3.

This subsystem consists of two main parts. They are: (1) the session handling module (SHM) and (2) a pair consisting of OSA module (OSAM) and ORS interpreter (ORSI).

Session handling module

The session handling module is a kind of 'brain' of the whole agent. Its main task is to manage the processing of particular requests made by the users of the BMS. The process runs as follows:

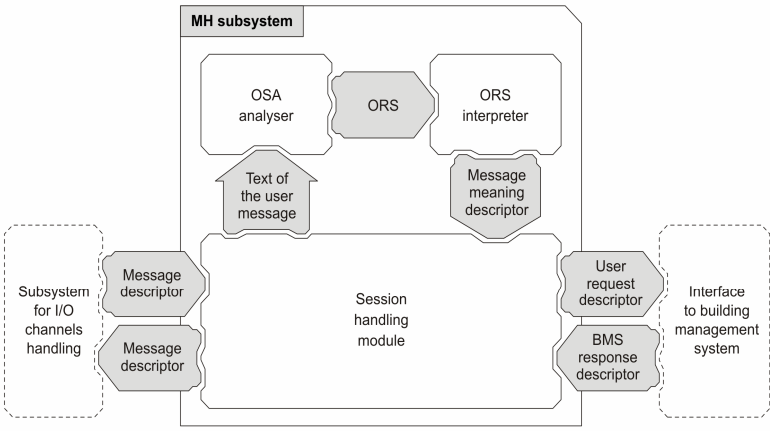

Fig 3. The MH subsystem diagram

a) Receiving the user's request and opening a new session.
b) Obtaining all the information describing the details of the task, and, based on them, creating a request descriptor.
c) Sending the descriptor to the BMS interface.
d) Receiving a response descriptor from the BMS interface.
e) Generating a natural language answer for the user based on the content of the response descriptor.
f) Sending the message descriptor to the proxy module in the first layer.

PAI/IB system has been designed to be multi-access – in other words, to be capable of parallel processing of many simultaneous users' requests sent through any channel. Particular requests are processed by the SH module in separate sessions.

A session starts with receiving the user's request by SHM. The request is passed on by the proxy module of IOCH subsystem in the form of a message descriptor, which has been described above.

Having received the request, the SH module creates recourses necessary for handling the session and labels the session with a unique identifier which consists of the source identifier and a successive request number. Next, the execution of the main task takes place, that is, obtaining all the information pertaining to the details of the task. This information is "extracted" through the semantic analysis of users' messages performed by the OSA module and the ORS interpreter (see the next paragraph). As the result of the process, a *message meaning descriptor* (MMD), containing the information on the subject of the request, is returned to the SH module. The descriptor is a four-tuple of the following form:

< OBJ_ID, ACT_DESC, ACT_START, ACT_STOP >

The meaning of those components is following:

- OBJ_ID is the object identifier, i.e., the identifier of a device or system that the user's request pertains to. It is assumed in the model that objects are identified by a pair: *object location identifier*, *object class identifier*.
- ACT_DESC is a description of an action to be undertaken in relation to the object. This description contains action identifier (e.g. turning a device on/off, checking the parameter values, etc.) as well as a specification of the details of an action, when needed (e.g. the temperature the heating is to be set at).
- ACT_START and ACT_STOP components determine the beginning and ending times of an action if that information was given in a user's request.

For example, for the request:

Turn the floor heating in the bathroom at 6 pm.

the following MM descriptor could be generated:

< OBJ_ID:: roomBATH:devFLOORHEAT;
ACT_DESC:: actTURNON; ACT_START:: 06:00; ACT_STOP:: –; >

The model is equipped with mechanisms that recognize and deal with a situation when a user's request in incomplete. An example of such a situation might be the order:

Turn off.

where it was not specified which device the order pertains to. In such a case the SH module engages the user into a natural language dialog in order to obtain the missing information. The base for this process are:

a) an ontology describing the properties of a particular systems and devices controlled by BMS that was created for the PAI/IB model;
b) algorithms based on that ontology, which allow for determining which information is missing.

One of the components of this ontology are Polish natural language queries related to the particular piece of missing information. Once the SH module determines the piece of missing information, it sends a message containing an appropriate query (obtained from the ontology) to the first layer of the interface.

The answers given by the user are analyzed by the OSAM/ORSI pair in the same way as the message that starts the session, then the analysis result is sent to SHM in a form of a successive MMD. The SH module considers the information contained in the descriptor and once again verifies if the information related to the request is complete. If it is necessary to obtain additional information from the user, the process is repeated.

Having received all the necessary information, the SH module places it in the *user request descriptor* (URD), which it then sends to the third layer of the interface. This descriptor has the same structure as the message meaning descriptor discussed above, however, as a rule, it contains all the necessary information. Based on an URD modules contained in the third layer of the PAI/IB system generate control messages, which are then sent to BMS for execution.

If, as the response to a completed task, BMS sends a return message, the SH module receives a description of that message from the third layer of the PAI/IB agent. The description has a form of *BMS response descriptor* (BMSRD), which is a triple <*object_identifier*, *action_identifier*, *result_identifier*>. Based on the pair *object_identifier*, *action_identifier*, SHM recognizes which request the answer pertains to. The result identifier might be:

- a symbol signifying the task completion or an error code if the request described such an action as *Turn the floor heating in the bathroom*;
- a value identifier if the request had a form of a query pertaining to a parameter value such as *What is the air temperature in Wojtek's room?*

Based on the information contained in the BMSR descriptor, predetermined phrases contained in the configuration files and the appropriate algorithms, the SH module generates a natural language answer for the user. The answer is then placed in the message descriptor and sent to the first layer of the agent, which completes the session handling process.

The OSAM/ORSI pair

The pair consisting of OSA module and ORS interpreter enables the agent 'to understand' Polish sentences and phrases submitted by the user.

OSA module implements the method of the object-oriented semantic analysis (OSA). It is a heuristic method of analysis of Polish sentences, which can serve as commands and queries for computer systems of different classes, for example, databases or agent systems. The analysis preformed by the OSA module results in a description of the meaning of a sentence (or phrase) in the form of an object-oriented representation of semantics (ORS). The OSA method has been described in detail in [5], the description of the ORS model can be found in [6].

The OSA method and the ORS model are ‘domain universal’ in the sense they can be used for analysis of commands and queries to computer systems of different classes and operating in different problem domains. Binding to a particular problem domain, in our case controlling the components of an intelligent building, is provided by the ORS interpreter. This module ‘translates’ the meaning of the user’s message recorded in the ORS form into messages and data structures required by the system the user needs to communicate with. In the case of PAI/IB system, it is the MM descriptor described above.

2.3. Interface to building management system

Interface to building management system is the last layer of the PAI/IB model. This layer contains two modules (fig. 4):

- module for control message generation (CMG),
- module for system response descriptor generation (SRDG).

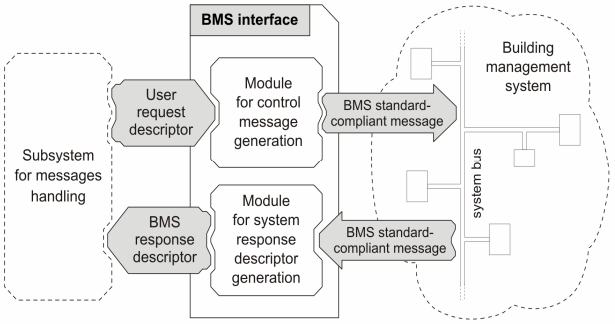

Fig. 4. The diagram of the third layer of the model

The CMG module translates the requests described in the UR descriptor into messages complying to the standard in which the BMS is implemented. An important part of this process is to determine, basing on object identifier (given in the UR descriptor), the address or identifier of a controlled device in a given BMS. A special configuration file is the basis for such a translation. In this file the addresses (identifiers) of the devices are bound to components of the object identifiers. The structure of this file and algorithms built in the CMG module enable proper handling of requests concerning both a single device, when one control message should be generated (one device address is determined), e.g.:

Turn on the floor heating in the bathroom.

as well as many devices, which requires generating many control messages (and determining many separate device addresses), e.g.:

Turn the light in the building off.

The SRDG module performs the reverse task – it generates an BMSR descriptor based on messages sent by the devices after the action defined in the user’s request has been completed. The descriptor generation process is based on the same configuration files as the process of generating control messages complying to chosen BMS standard.

Separating the third layer of the model that performs the above functions make the PAI/IB agent independent of any particular BMS standard. When the standard is specified (KNX, Modbus, BACnet or any other) all its specific requirements, such as content and format of control messages, etc. are reflected only in CMG and SRDG modules. Thus hooking the PAI/IB agent up to the BMS implemented in any standard requires only applying appropriate modules in the third layer of the agent.

3. Interface prototype

The efficiency of the solution described above has been verified through tests with an PAI/IB agent prototype. The prototype was implemented in Java 1.5 language; the tests were

performed on a PC platform with Windows XP operating system.

The main layer of the interface has been fully implemented in the prototype, i.e. the MH subsystem, as well as the selected modules included in the first and the third layers of the model:

– the proxy module of the IOCH subsystem,
– the instant messaging module,
– the interface module for the BMS simulator.

Two simulators were designed to run the correctness tests. The first simulated operation of Jabber communication network and cooperated with the instant messaging module. It made it possible to control the system through the internet chat mode.

The second one simulated an intelligent building and cooperated with BSMI module. This simulator included systems and devices listed further on and performed two functions. First, it created visual simulation of the effects of the user's commands, which made it possible to assess the correctness of analysis and interpretation of the user's commands by the PAI/IB agent. Second, it simulated the process of receiving answers for user's queries regarding the state of controlled systems and devices from the BMS.

The tests showed the effectiveness and functionality of an interface implementing the PAI/IB model. The prototype handled correctly both kinds of user's requests i.e. commands and queries.

Using the commands, it was possible to control effectively the operation of all the most important systems and devices that an intelligent house is equipped with, including:

– lighting, e.g.: *Turn off the light in the living room.*
– air conditioning, e.g. *Set the temperature in the bedroom on 25° C*.
– alarm and access control system, e.g.: *Block the garden door*.
– home appliances, e.g.: *Start the coffee machine*.
– shutter and blinds, e.g.: *Turn up the blinds in the living room*.
– lawn and garden sprinkling system, e.g.: *Water the lawn in front of the house.*

The commands may refer to the previously defined profiles with settings for groups of connected devices. A good example might be *Turn on 'the film show' profile*, which results in an adjustment of lights and blinders, turning on the TV and DVD player.

Each command can have a defined execution time. This definition can indicate the point in time, e.g.: *Turn the heating in the bedroom off at 2:30*, or a time interval, e.g.: *Keep the light in the living room on between 8 pm 10 pm.* The analyzer correctly recognizes also queries containing the information about repetition of a given operation, e.g. *everyday, every Tuesday, form Monday till Thursday*, etc.

The PAI/IB agent also makes it possible for the user to submit queries regarding the situation at home, or conditions of some systems, such as:

Which devices are on?
Is the light in the living room on?
What is the air temperature in the bathroom?

4. Further research

Several projects have been planned to be performed by students, leading to achieving the full functionality of the prototype, as described in the model. These projects form separate groups regarding the first and the third layer of the model. The projects from the first group are focused on developing various CCH modules. First, internet chat communication module based on Jabber network will be created, with the KVMJab library (http://sourceforge.net/projects/kvmjab/) as its base. The GG network support based on the JGGApi library (http://sourceforge.net/projects/jggapi/) is also planned.

The CCH modules for communication with the PAI/IB agent by e-mail and SMS are also planned. In the first, standard Java classes form JavaMail API library (http://java.sun.com/products/javamail/) will be used. The CCH module handling SMS communication is divided into two parts. The first one, operating in the J2ME environment on a standard mobile phone, will read and send SMS messages. The second will be included in the main PAI/IB agent application operating on a PC computer. Both parts will communicate exclusively through a Bluetooth connection. When building an CCH module, WMA library (http://java.sun.com/products/wma/) will be used for reception and sending SMS, and Marge (https://marge.dev.java.net/) and BlueCove library (http://code.google.com/p/bluecove/), for realizing a Bluetooth connectivity.

The team working on the projects related to PAI/IB system is eager to create an CCH module facilitating voice communication. Unfortunately, the lack of libraries realizing Polish speech recognition, which could become a base for such an CCH module is currently an insurmountable obstacle.

The second group of projects aim at creating an interface modules for building management systems. Two such projects are on the way. The first one includes an interface compatible with KNX/EIB standard (http://www.knx.org/), with Calimero library as its base (http://sourceforge.net/projects/calimero/). In the second one, an interface to BMS implemented in Modbus standard (http://www.modbus.org/) will be developed with JaMod library as its base (http://sourceforge.net/projects/jamod/).

4. References

1. Berglund A., Johansson, P., *Using speech and dialogue for interactive TV navigation*, [in] Universal Access in the Information Society, vol. 3, nr 3, pp. 224-238, 2004
2. Hampicke M., *Smart Home: Speech Based User Interfaces for Smart Home Applications*, [in] Proceedings of the COST219bis Seminar "Speech and Hearing Technology", Cottbus, Germany, 2000
3. Lieberman H., Espinosa J., *A goal-oriented interface to consumer electronics using planning and commonsense reasoning*, [in] Knowledge-Based Systems, vol. 20, nr 6, pp. 592-606, 2007
4. Nichols J., Myers B., Higgins M., Hughes J., Harris T., Rosenfeld R., Pignol M., *Generating remote control interfaces for complex appliances*, [in] Proceedings of the 15th annual ACM Symposium on User Interface Software and Technology, pp. 161–170, ACM Press, 2002
5. Pechmann P., *An object-oriented model of the semantic analysis process of Polish natural language queries to medical databases*, Doctoral dissertation at the Faculty of Computer Science and Information Systems, Szczecin University of Technology, 2006, (in Polish)
6. Pechmann P., *The Model of Object-oriented Representation of Semantics*, [in:] Proceedings of the Conference “SMI 2007”, Polish Journal of Environmental Studies, Vol. 16, No. 4A, 2007
7. Ponnekanti S., Lee B., Fox A., Hanrahan P., Winograd T., *ICrafter: A service framework for ubiquitous computing environments*, Ubicomp 2001, Lecture Notes in Computer Science, vol. 2201, pp. 56-75, Springer, 2001
8. Speicher S., Preuß S., Sedov I., Cap C., *Voice-Controlled Ubiquitous Computing*, [in] Proceedings of the 2004 International Conference on Wireless Networks, pp. 116-122, Las Vegas, USA, 2004
9. Yates A., Etzioni O., Weld D., *A Reliable Natural Language Interface to Household Appliances*, [in] Proceedings of the 8th International Conference on Intelligent User Interfaces, pp. 189-196, Miami, USA, 2003