Clinical Study of Microbial Markers by Mass Spectrometry Method

G.A. Osipov[1], N.V. Verkhovtseva[2], A.N. Vedenin[3], D.A. Koluntaev[3], A.N. Verenchikov[3]
[1]Academician Yu. Isakov Research Group, Bakulev Scientific Center for Cardiovascular Surgery,
[2]Lomonosov Moscow State University;
[3]Q-Technologies RD Center, Bar, Montenegro

Keywords: microbiological analysis, express test, fatty acids, mass spectrometry, clinical applications.

ABSTRACT: Determination of semi-pathological bacteria is considered an important tool at disease diagnostics, selection of appropriate treatment methods and verification of the correct choice. Wide spread methods include biota sampling and bacterial incubation. The method is hardly quantitative and takes several days. No need to say that in some cases it is too slow to treat the disease timely.

This review presents a novel express method of quantitative and reliable measurement of wide variety of semi-pathogenic bacteria - Mass Spectral measurement of Microbial Markers (MSMM). The method allows simultaneous determination of more than one hundred microbial fatty acids *in situ* for clinical, biotechnological or environmental samples. The method is quantitative and express, since the measurements are made without prolonged incubation (precultivation) and without using of biochemical test-materials and primers.

The proposed MSMM method has been applied for clinical studies. Unprecedented information on the quantity of anaerobes and uncultivated aerobes, as well as actinobacteria, yeasts, viruses and microscopic fungi provides full understanding of microbial etiology, and such information may be obtained quick in critical cases in clinical practice. In one particular study of intestine dysbiosis, the tested mass spectral method has confirmed the hypothesis about the nosological specificity of changes in the intestinal microbiota. It has been proven that infectious processes are polymicrobial. Measurements have shown that anaerobes dominate by number and by functional activities at inflammations. The division of microbes into pathogenic and non-pathogenic is artificial – most of microbes in a human body exist in both forms simultaneously. Lactobacilli and bifidobacteria appear as agents of septic conditions and endocarditis. MSMM data confirm that anaerobes of *Clostridium*, *Eubacterium*, *Propionibacterium*, as well as actinobacteria of *Streptomyces*, *Nocardia*, *Rhodococcus* are infection dominants and usually act in groups. The data testify translocation of

these microbes in inflammation loci from the intestine. Relative markers concentration stay consistent between different biota samples, thus, any clinically convenient sample may be used for determining microbes within the inflamed organ or at any specific locus. Quantification using GC-MS reveals that the influence of antibiotics on the normal intestine's microbiota are not as dramatic as believed. Growth-promoting effects are the most important benefits of probiotic applications. The probiotic essence is not the microbial biomass itself, but growth factors, alarm molecules, and other factors of intestinal microbes. There are new possibilities in improving probiotics by using microbial "consortia"

Using Architectural Abstractions in Embedded System Design

Vasiliy Pinkevich
Computer Science Department
ITMO University
Saint-Petersburg, Russia
vpinkevich@niuitmo.ru

Alexey Platunov
Computer Science Department
ITMO University
Saint-Petersburg, Russia
platunov@lmt.ifmo.ru

Abstract— **The conceptual part of complex embedded systems design includes the following key stages: system analysis of initial requirements, architectural and micro-architectural decisions generation, evaluation of decisions. During these stages, many important mechanisms of subsequent implementation are defined. These are the stages that are the least formalized and automated. The proposed method allows the design process to be partially formalized by the usage of computational mechanism concept as the central abstraction. The considered example regards to analysis of languages used together in complex embedded systems design with "immersion" to the level of custom system on a chip design. The comparison of design languages, carried out on the basis on the proposed approach, allows the design means for subtasks and subsystems to be chosen more effectively. The source code markup method is proposed as a tool for automated processing of multi-language projects targeted to work with design entities, which cannot be adequately and directly expressed by the standard languages means. In general, the demonstrated approach stimulates the designers to concentrate on "cross-cutting" conceptual mechanisms of a project and provides a way to monitor the adequacy of their multi-stage implementation.**

Keywords – embedded system; system level design; architectural abstraction; design space exploration; multi-language design.

I. INTRODUCTION

Embedded systems (ES) design process in its conceptual phase has to be based on methodologies of their complex representation [1]. In the literature, this level of consideration is typically attributed to electronic system-level (ESL) design [2, 3] or system-level design (SLD) fields, however, already in [4] it is noted that activities in this design stage are wider. We call these activities HLD – High Level Design. Abstract system concepts are used at the stage of ES high-level design. They largely form the ES project, but are not fixed in the ES implementation. This greatly complicates the control over the adequacy of system implementation in its "top-down" transition from one level to another within design process. Therefore, "cross-cutting" methods of working with conceptual information, that cover all ES design phases, are needed.

II. DISCUSSED PROBLEM

Languages for design, programming, modeling and other problems are critically important ES implementation instruments. They are actually platforms containing abstraction means to allow explicit allocation of conceptually important design units – classes, functions, macros, modules and other units. However, the serious problem with standard programming languages is that they do not allow system specification to be composed exactly in the same terms as designer thinks about it. The examples of entities, which are difficult to be expressed, include: cross-cutting mechanisms, with support scattered over the entire specification code (means of ensuring reliability, lower power consumption, etc.); mechanisms that affect multiple levels of a system, described in several languages (e.g. hardware description language and software programming language); any significant logical structures, unsupported by the language means. If developer has used abstractions of higher level than the language and standard library, these abstractions are typically left in his mind. Thus, it is necessary to have an opportunity to establish consistency between design abstractions, which are generated by the developer "in free mode", and constructions that are directly provided by the languages used in the design. For this purpose, both methodological framework and automation toolset are needed to allow the developer to use this approach in practice.

III. RELATED WORK

One of the known methods to solve the stated problem is the usage of domain-specific languages (DSL) and related tools [5]. However their usage may be limited due to the following reasons:

- initial project of the system is not formalized enough to be unambiguously realizable (synthesizable) from the specification;
- architectural information is unavailable so the system is considered only via its implementation;
- too much overhead for the creation or implementation of the language and tools that provide the required conceptual entities;
- legacy system support is required;
- manual optimization with the usage of low-level language is required.

This work was partially financially supported by Government of Russian Federation, Grant 074-U01.

There are examples of allocation and classification of conceptual elements for ES design and analysis in the literature. Typically, several maximally independent axes in the possible solution space is provided. Axes contain marks, which designate possible problem solutions. The marks may represent either various abstraction levels (see Fig. 1a, "design cube" – model for VHDL language [6]) or technical decisions (see Fig. 1b, "design space evolution" [7]). Furthermore, axis may represent the process that evolves in time within computer system life cycle (see Fig. 1c, "rugby model" [8]). Also, the known examples of this approach usage are VSIA taxonomy and ESL taxonomy [9], which consider the properties of ES structural units models. They can be applied for analysis of programming and design languages properties [3, 10]. However, it is very important, that the presented models do not consider the cross-cutting mechanisms problem.

IV. PROPOSED METHOD

Based on the concepts, models and principles of ES HLD-methodology [11–14], the following method of analysis of design entities and implementation languages is proposed.

ES design process should be carried out within aspect approach [15–17]. In the initial step, system architect allocates important design space segments (aspects). Each aspect reflects a particular problem space in the project execution. Within a single aspect, the sets of design space axes (subspaces) are allocated. An axis is the certain problem within the project. On each axis, the set of computational (and other) mechanisms, ranged by the certain criteria, are located. The mechanisms provide the means to solve design problems.

Computational mechanism (CM) is the central concept of the proposed method. It is an architectural pattern that demonstrates the principles of computational process organization. In contrast with the popular concept of "design pattern" that does not have fixed requirements for abstractness of description and internals demonstration, CM has to transparently provide with useful "computational" technical principles without fixating of their implementation. Thus, CM should be considered as a specific category of patterns of computer systems design. Along with computational mechanisms, other categories of mechanisms are used, e.g. mechanisms of interaction, verification, debug. Thus, the mechanism is the universal element that can be allocated both within a single design language and across several layers, which involve several languages to work with. The marks on the axes that are proposed within the certain methodologies [6–9], can be treated as the variants of the mechanisms, while the proposed axes can be used as design space axes.

During ES implementation, the set of aspects, design space axes and mechanisms within each axis is used by developers as a library of design decisions, primarily, at the conceptual level [18]. Also, the models that are constructed in these terms can be used at the verification step [19].

Annotation of the source text of the project (primarily multi-language) is proposed to enable the automated support for this approach. The tag language, based on comments of the special format, has been developed. Annotated code allows the fast navigation through mechanisms implementation fragments to be carried out that simplifies manual control over their implementation correctness.

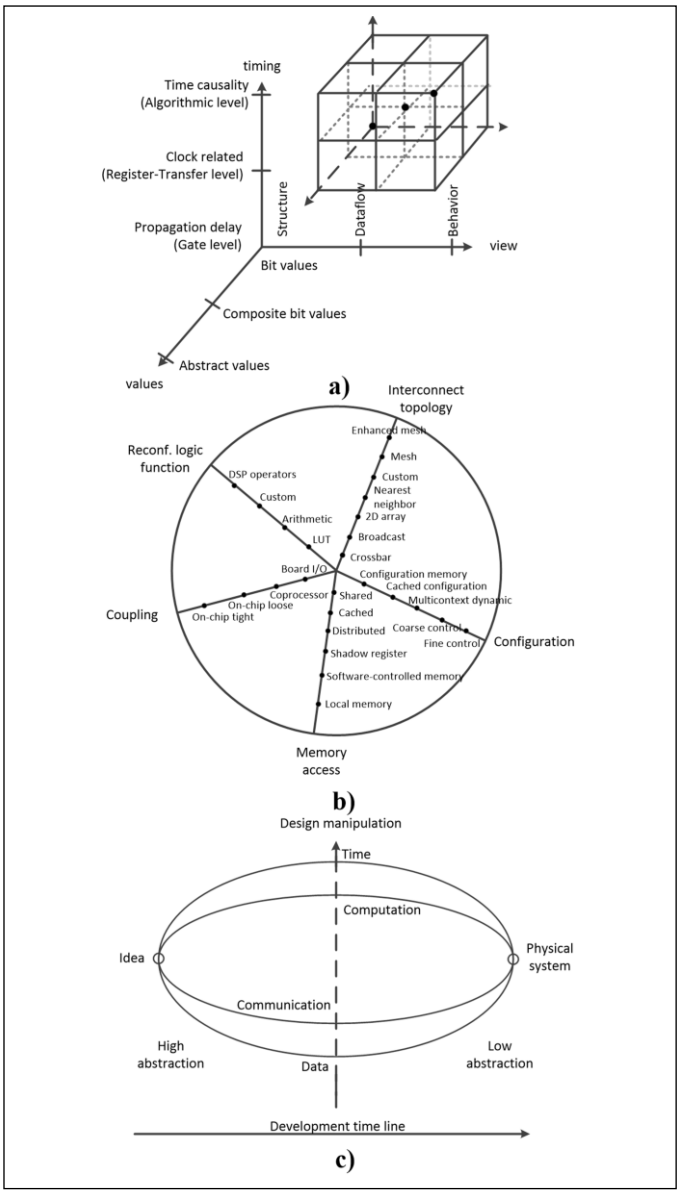

Figure 1. Variants of aspect and design spaces representation in embedded systems design and analysis methodologies.

V. THE USE CASE OF THE METHOD

The proposed method has been applied to the typical set of languages that are used for ES design and development with utilization of programmable processors and dedicated hardware units implemented in FPGA or ASIC (see Fig. 2 and Table 1).

The set of design space axes has been allocated for analysis. This set includes four axes from ESL taxonomy [3, 9] and two extra axes that have been added: data flow / control flow ratio of structural unit functional implementation and axis

of functional verification mechanisms. Mechanisms of first extra axis variously combine involvement of instruction and data streams in computational process control. The second axis contains the mechanisms that can be used for computer system testing and verification during its design, depending on the scale of the element under verification. The number of marks on the concurrency and communication axes has been reduced and only important mechanisms, which are not tied to specific implementation, have been left.

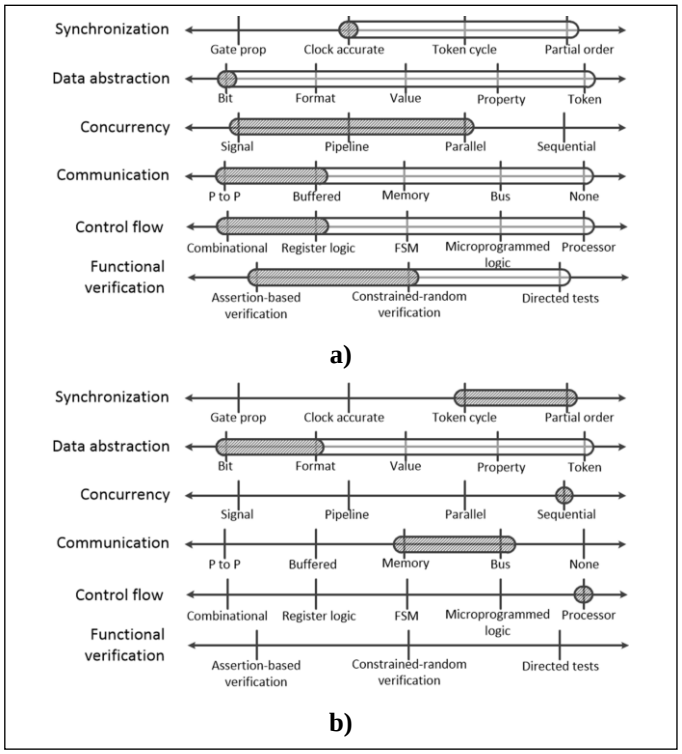

Figure 2. Mechanisms of languages: a) SystemVerilog (synthesizable subset + verification) and b) generic assembler. Shaded outlined areas – built-in mechanisms, outlined areas – mechanisms which can be implemented on the basis of built-in mechanisms.

Generic assembler (assembler of abstract programmable sequential processor core) reflects the capabilities of software implementation and is not tied to specific architectures or processor configurations. If needed, the set of language mechanisms can be extended through processor specialization (by redesign, IP-core configuration or custom extensions). The examples are Microblaze (Xilinx), NIOS (Altera) and other processor cores.

Languages have built-in support of the mechanisms of a certain complexity (or abstractness) and, in most cases, means for combining simple mechanisms to form complex ones. It is assumed that the level of mechanisms' complexity within design space axis can be increased in case mechanism implementation fits into language capabilities. Within this approach, the languages have been analyzed from two perspectives – from the viewpoint of the mechanisms that have built-in support in the language and from the viewpoint of the mechanisms that can be effectively realized by the language means based on built-in mechanisms.

TABLE I. BUILT-IN MECHANISMS OF LANGUAGES.

Design space axes	**Languages**				
	System-Verilog (synthesizable)	***System-Verilog (for simulation)***	***SystemC (for simulation***	***Generic assembler***	***SysML (modeling only)***
Synchronization	Cycle-accurate	Cycle-accurate, system events, partially ordered	Cycle-accurate, system events, partially ordered	Instr. cycle	Partially ordered (sequence, activity diagrams)
Data abstraction	Bit and format (with wire and reg vectors)	All (with enums and structures)	All (with structures and classes)	Bit	From format to token (package diagrams)
Concurrency	Signal and block parallelism	Signal and block parallelism	Signal, block, software processes (with sc_process_handle)	Sequential	Signal, block, multi-application (with internal block diagram)
Communication	P to P and buffered (with wires and regs)	Same as synthesizable	P to P, buffered, memory (with pointers)	No	No specific mechanism
Control flow	Comb. and register logic	Same as synthesizable	Same as System-Verilog	Pro-process or as a platform	FSM (state-charts), sequential (sequence diagrams)
Func. verification	No	Assertion-based, constrained-random	No	No	No

The example of using multi-language source code annotation is demonstrated in Fig. 3.

```
File: .\asm\boot.asm, line: 176
CALL P_LOAD_CRC # load from coprocessor to ACC
SUB  CRC_REG    # checking CRC
JEQ  LABEL_BLOCK_CRC_OK
...
File: .\asm\asm.py, line: 198
"RWRK": [11, "R"], # read coprocessor cmd
"WWRK": [12, "R"], # write coprocessor cmd
...
File: .\hdl\wrk.sv, line: 236
if (ctrl_a == ADDR_CRC) begin
    ctrl_do <= crc;
end
...
File: .\hdl\crc32.sv, line: 35
always @* begin
    crc_new[0] = crc_old[0] ^ crc_old[6] ^ ...
```

Figure 3. Fragment of CRC implementation mechanism.

Here, the fragment of cross-cutting implementation of cyclic redundancy checksum (CRC) mechanism is demonstrated. CRC is used for integrity control of bootable software images for heterogeneous multiprocessor system. The system has been implemented as system on chip (SoC). Embedded boot manager is a specialized processor being programmed in assembly language. The cut has been acquired from the annotated code automatically. The source code has been realized in assembly (program for the processor, boot.asm), Python (compiler, asm.py), SystemVerilog (CRC coprocessor, wrk.sv and crc32.sv).

VI. FUTURE WORK

The proposed approach is not formal, thus, quality of its application depends greatly on expert's qualification. Thus, further refinement of the used concepts has to be carried out. Also, extraction of individual subspaces with clearly defined axes and mechanisms sets has to be done. Such axes could be recommended as typical for certain class of projects and problems. The mechanisms that require cross-level implementation are of special interest.

CONCLUSION

The proposed method allows design process to be partially formalized using HLD-methodology system of concepts. Computational mechanism is the central abstraction. Design languages comparison, carried out on the basis of the proposed method and applied to the languages being used in complex hardware-software projects, allows design tools to be chosen more effectively for subtasks and subsystems. The demonstrated approach makes the developers to concentrate on "cross-cutting" conceptual mechanisms and enables control over the adequacy of their multi-stage implementation.

REFERENCES

[1] J. Teich, "Hardware/software codesign: the past, the present, and predicting the future", Proceedings of the IEEE, 2012, vol. 100, pp.1411 – 1430.

[2] D. Densmore, R. Passerone, A. Sangiovanni-Vincentelli, "A Platform-Based Taxonomy for ESL Design", IEEE Design and Test of Computers, September 2006.

[3] B. Bailey, G. Martin, "ESL models and their application", New York: Springer Publication, 2010.

[4] A. Sangiovanni-Vincentelli, "Quo vadis SLD: reasoning about trends and challenges of system-level design", Proceedings of the IEEE, 95(3), 2007, pp.467-506.

[5] M. P. Ward, "Language-Oriented Programming", Software - Concepts and Tools 15(4): 147-161 (1994)

[6] W. Ecker, M. Hofmeister, "The design cube - a model for VHDL designflow representation", Proceedings of the European Design Automation Conference (EuroDAC), Hamburg 1992, 752-757.

[7] A. Chattopadhyay, "Ingredients of adaptability: a survey of reconfigurable processors", VLSI Design, 2013, vol. 2013, p.18.

[8] A. Jantsch, S. Kumar, A. Hemani, "A Metamodel for Studying Concepts in Electronic System Design", IEEE Design & Test of Computers, vol. 17, no. 3, pp. 78-85, Jul. 2000.

[9] B. Bailey, G. Martin, A. Piziali, "ESL Design and Verification: A Prescription for Electronic System Level Methodology", Elsevier Morgan Kaufmann, 2007.

[10] Panagopoulos, G. Papakonstantinou, N. Alexandridis, and T. El-Ghazawi, "A comparative evaluation of models and specification languages for Embedded System design", Languages, Compilers, and Tools for Embedded Systems (LCTES-03), San Diego, Ca., June 11-13, 2003.

[11] A. Platunov, A. Nickolaenkov, and A. Penskoy, "Architectural representation of embedded systems", 2012 Mediterranean Conference on Embedded Computing (MECO), June 2012, pp.80-83.

[12] A. Platunov, A. Kluchev, and A. Penskoi, "HLD Methodology: The Role of Architectural Abstractions in Embedded Systems Design", 14th GeoConference on Informatics, Geoinformatics and Remote Sensing, 2014, pp. 209–218.

[13] Platunov A., Kluchev A., Penskoi A., "Expanding Design Space for Complex Embedded Systems with HLD-methodology", Proc. of the 6th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT) - 2014, pp. 253-260.

[14] A. Platunov, A. Penskoi, and A. Kluchev, "The Architectural Specification of Embedded Systems", 2014 3rd Mediterranean Conference on Embedded Computing (MECO), June 2014, pp. 48-51.

[15] D. Broman, Ed. A. Lee, S. Tripakis, and M. Toerngren, "Viewpoints, formalisms, languages, and tools for cyber-physical systems", 6th International Workshop on Multi-Paradigm Modeling - MPM'12, October 2012, pp.49–54.

[16] A. Platunov, and A. Nickolaenkov, "Aspects in the design of software-intensive systems", 2012 Mediterranean Conference on Embedded Computing (MECO), June 2012, pp.84-87.

[17] J. M. P. Cardoso, P. C. Diniz, J. G. de F. Coutinho, and Z. M. Petrov, "Compilation and Synthesis for Embedded Reconfigurable Systems: An Aspect-Oriented Approach (Google eBook)", Springer, 2013, p. 215.

[18] Kustarev P., Bikovsky S., Antonov A., Yanalov R., "Process control and synchronization patterns for SOC", 14th GeoConference on Informatics, Geoinformatics and Remote Sensing, 2014, Vol. 1, No. 2, pp. 287-294.

[19] Kustarev P., Bikovsky S., Pinkevich V., "Functional monitoring of SoC with dynamic actualization of behavioral model", unpublished.

Hardware Approach of Text-to-Speech in Embedded Applications: Work in Progress

Gordana Laštovička-Medin, Itana Bubanja
Faculty of Science and Mathematics
University of Montenegro
Podgorica, Montenegro

Abstract—**This paper presents work in progress. Here, we describe our ongoing experience teaching embedded systems to physics students who've been given access to the Arduino platform and its open source community. Our chosen topic, Design and Applications of Embedded Systems for Speech Processing, was researched as part of the Basic Measurement in Physics course of the Faculty of Science and Mathematics at the University of Montenegro. During the student research the SpeakJet sound synthesizer was explored. Building embedded systems has enormous potential for developing students' skills and cultivating a culture of thinking and participating**

Keywords-speech processing, speech synthesis, SpeakJet, processor TTS, Arduino

I. INTRODUCTION

Speech processing, especially audio manipulation and sound processing is commonly performed by many everyday electronic devices. Examples of such devices are are digital voice recorders, speaking GPS receivers, and many others. In general, the speech processing capabilities that can be added to an electronic device are voice recording, voice playback, text-to-speech (TTS) synthesis and speech recognition (SR). Voice recording and voice playback are used in digital voice recorders to store speech in non-volatile memory and then replay it at a later time. TTS involves reading a written text and converting it into spoken words that can be played through speakers. People with reading or visual difficulties may find such systems extremely useful.

TTS synthesis transforms any linguistic information stored as data or text into speech. We can make robots speak by simply recording human speech, and playing it back when needed. True speech synthesis, however, is to allow robots to generate boundless speech output- in other words to let them speak their mind. TTS is true speech synthesis. It is the synthesis of speech based on unrestricted text input. There are three main classes of TTS synthesis: articulatory, formant, and concatenative [1]. Articulatory synthesis is based on a complete 3D model of the human speech apparatus. It uses acoustic parameters extracted from the model to synthesize speech. Articulatory synthesis is the most powerful process, but also the most complex. It requires analyzing Magnetic Resonance Imaging (MRI) scans of speech production. It scores high in intelligibility, but low in naturalness. Formant synthesis uses a black-box modeling approach to speech production It analyzes the end transfer function of the vocal tract, rather than the way it is made. Formants are the vocal tract's resonant frequencies. They give the phonetic character to speech sounds. The voice of Stephen Hawking is an example of formant synthesis. It shows high intelligibility but low naturalness. Concatenative synthesis does not seek to model speech production. It uses a database of prerecorded segments of natural speech that it concatenates one after the other. This approach gives the synthetic speech a very natural sound.

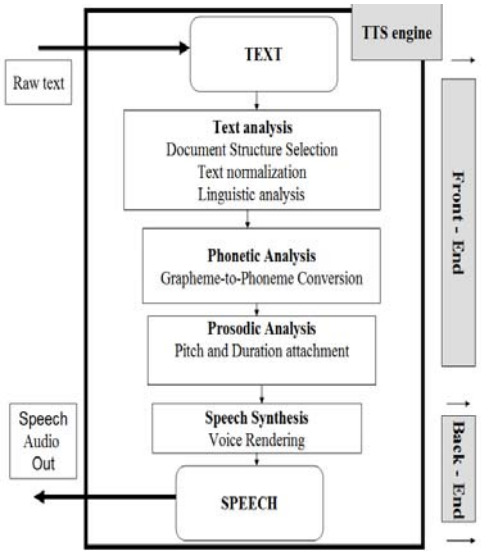

Figure 1. Block diagram of a general text-to-speech system. The figure has been adopted from [1].

A text-to-speech system (or "engine") is composed of two parts: front end and back end. The front end has two major tasks, as illustrated in Fig. 1 [1]. Firstly, it converts raw text containing symbols, such as numbers and mabbreviations, into the equivalent of written words. This process is often called text normalization, pre-processing, or tokenization. The front end then assigns phonetic transcriptions to each word, and divides and marks the text into prosodic units, like phrases, clauses, and sentences. The process of assigning phonetic

transcriptions to words is called text-to-phoneme or grapheme-to-phoneme conversion. Phonetic transcriptions and prosodic information together make up the symbolic linguistic representation that is output by the front end. The back end, often referred to as the synthesizer, then converts the symbolic linguistic representation into sound. For more details we refer to [2,3,4].

TTS in embedded systems can be analyzed from several points of view: as TTS integrated circuits, TTS modules, TTS across embedded operating systems and TTS software applications for embedded devices. Details can be found in [5]. In this article we are interested in TTS integrated circuits (ICs). We use a SpeakJet sound synthesizer chip and the TTS256 Text-to-Speech chip for SpeakJet, in order to create a text-to-speech solution. The work is under progress.

II. SPEKAJET: ALLOPHONE BASED SPEECH SYNTHESIS

The central hardware component of this project is the Magnevation's Speakjet which is a 20-pin IC [6]. It is designed to add speech and audio to embedded microcontroller applications. The chip is self-contained and requires just an external +5V supply and a speaker for its operation. A mathematical sound algorithm is used to control its five channel internal sound synthesizer to generate vocabulary speech synthesis and complex sound generation. The chip is low cost and is aimed primarily at the hobby market. The Speakjet is programmed with 72 speech elements, 43 sound effects and 12 DTMF touch tones. In addition, sound effects such as the pitch, rate, bend and volume can be controlled. The chip can easily be controlled from a microcontroller.

The SpeakJet uses a technique called allophone based speech synthesis in order to create the sound that we interpret as intelligible speech. A string of letters spelled out as "hello world" can not be sent to the SpeakJet because the way the words are written down in English and the way they are spoken is very different. Written English text consists of a series of letters but spoken text consists of a series of phonemes The smallest meaningful unit of sound in human speech is called a "phoneme". Phonemes, in turn are represented by allophones which are sets of multiple possible spoken sounds used to pronounce a single phoneme. To be able to generate intelligible speech from an allophone based synthesizer it is important to understand the difference between letters and allophones. There are 26 letters in the English alphabet but hundreds of allophones. English language is not spoken phonetically since subconsciously speakers apply various conventions that change the sound represented by particular letters based the context surrounding the word or sentence. For example, the letter "e" may be short as in "set" or it can be long, as in the first e in "concrete". Contrary, the most southern Slavic languages (Montenegrin, Bosnian, Serbian) are spelt phonetically. The writing Serbian system does not take into account allophones while as we saw before the allophones play a critical key in naturalness of English synthesized speech.

Figure 2 shows block diagrams of the system designed to control, manipulate and program the SpeakJet. The circuit diagram is displayed in Fig. 3. The aim was to program chip to be used in talking puppet, thus there are 3 tackle switches (S1,S2,S3) in order to provide chip activation by pressing toy\s hands on three different places. Besides the SpeakJet, two additional integrated circuits were used to build the complete electronic circuit: an LM386 low-voltage audio power amplifier and a MAX232 driver/receiver.

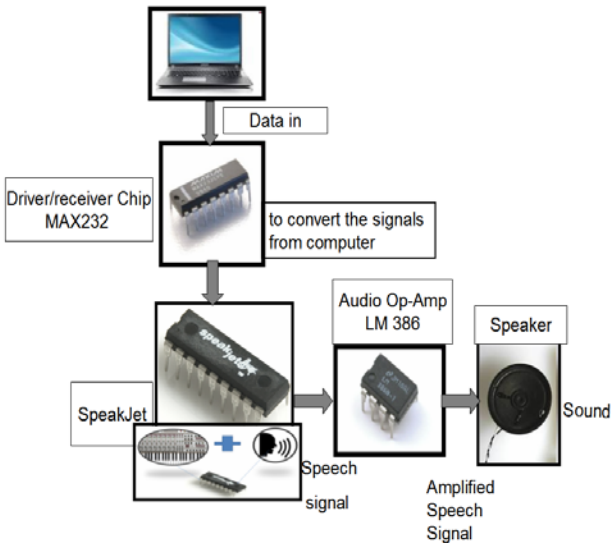

Figure 2. Block diagram designed used to program SpeakJet.

The MAX232N-1 integrated circuit was used to convert the signal from a serial port to signals suitable for use in digital logic circuits. The created signal was transferred using 6 phono jacks and phono plugs. By pressing one of three tactile switches connected to the phono plugs, an electric circuit was created and the SpeakJet was "ready" for uploading. A tactile switch also known as a momentary button or push-to-make switch, is commonly used for inputs and controller resets. These types of switches create a temporary electrical connection when pressed. One pin is supplied with +5 volts and the other pin is grounded. The LM386N-1 audio amplifier takes the electrical signal generated by the SpeakJet when we push and release one of the tactile switches (S1 S2, S3) and then amplifies the electrical signal to create enough power to drive the speaker. Switches S1, S2, and S3 were used to control the voltage on SpeakJet's pins 2, 4, and 7. The switches are normally open, which means each pin is connected to ground. When we push one of the switches, the voltage on the corresponding pin raises to +4.5 volts. This means that electric circuit is created and SpeakJet is "ready" for software uploading. for use in digital logic circuits.

The hardware system accompanying the circuit diagram is displayed in Fig. 4. A breadboard was used for circuit prototyping. Fig. 9 shows the same design as presented in Fig. 4 but with added phono plugs and phono jacks. A useful tool from Magnevatron is a Windows program called Phrase-A-Lator which has a great dictionary of word-to-allophone translations, and also has the ability to pump information directly at a Speakjet connected to a PC via a serial port. This

is very useful because in this way phrases are directly programmed into the EEPROM, and can be tested for sound. Using the Phrase-A-Lator, we can convert the text/allophone string into the proper codes to be sent to the Speakjet chip, and apply those to our Arduino code (see next chapter).

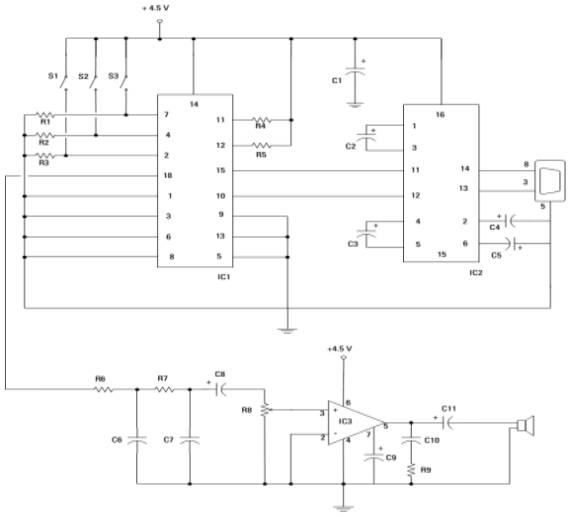

Figure 3. A circuit diagram with SpeakJet, Max232 and audio op-amplifier LM386 to program SpeakJet.

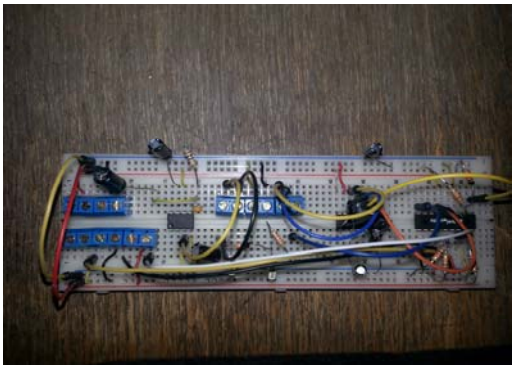

Figure 4. Breadboard SpeakJet prototyping

Figure 5. Breadboard as shown in Figure 4 but with added phono plugs and phono jacks.

III. TEXT-SPEECH SOLUTION WITH SPEAKJET AND TTS256

Speech processing with the IC SpeakJet works well for a limited vocabulary, but if an application requires unlimited or arbitrary speech output, the TTS chip does the work of translating any English text into the allophenes that the SpeakJet understands. The TTS256 chip [6,7,8], which contains a dictionary of words-to-allophones converts English text into a sequence of phonemes. This chip is a companion to the SpeakJet and comes with a built-in 600-rule database to convert English text to phoneme codes. Speech can easily be generated from ASCII text in microcontroller-based embedded applications, making the chip extremely easy to use in applications where speech generation is required. The TTS256 is controlled from its serial port and, thus, it is compatible with any microcontroller with such a port. Then a SpeakJet chip converts the phonemes into sound. The problem is that the Magnevation software can not communicate directly with this chip, since the communication is with Arduino itself. All that is needed is a simple little host program on the Arduino to redirect information to the Speakjet.

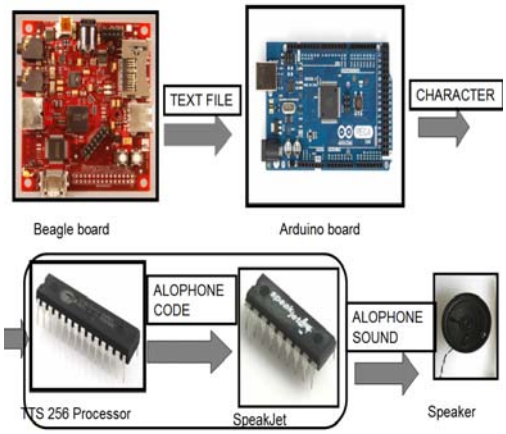

Figure 6. Block diagram of speech synthesizer. It consists of BeagleBoard, Arduino, TTS256, SpeakJet and Speaker

Figure 6 shows a block diagram of a speech synthesizer. It consists of the BeagleBoard, Arduino, the TTS256, the SpeakJet and a Speaker. The Arduino was used to control TTS256 while the Beagle Board [9] was used as a single-board computer which is is low-power, open-source hardware [9]. The TTS-Speakjet breadboard prototyping is displayed in Figure 7.

In what follows we will show how the Magnevation software works in order to program chip. The Magnevation software opens and closes the serial port every time we send information down. Arduino's default setup will run the following process: when the USB-Serial connection is opened up by the host, it institutes a chip reset, and the program restarts. Figure 8 shows a couple of Phrase-A-Lator screenshots. When the phrase is completed (with the help of "Say it"), we have to select "View Codes" to get the numerical allophone sequence and then to paste it into the Arduino code. Alternatively to the setup shown in Fig. 9 the VoiceBox shield

can be used. It contains a SpeakJet chip while the TTS chip can be soldered directly onto the VoiceBox shield, as shown in Fig. 9.

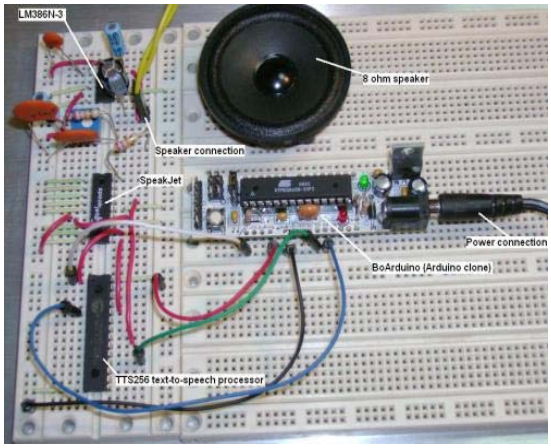

Figure 7. Breadboarding design of electronic circuit containing of TTS256, Speakjet and Speaker

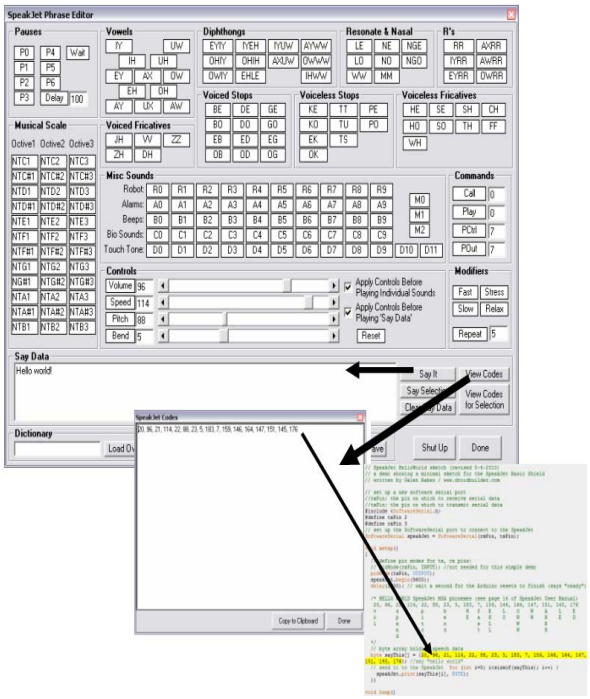

Figure 8. PhraseALator's control panels.

IV. FUTURE WORK AND DISCUSSIONS

Our research towards speech processing is at its early stage. An attempt had been made in order to implement and test the text-to-speech solution. The hardware design is based on a TTS256 chip, a sound synthesizer SpeakJet and an open source platform. Paper also presents an pedagogical aspect of TTS engine where students explored the hardware approach to TTS and how to program SpeakJet. Currently we extended our research towards the design of the audio hardware interfaces in order to visualize and manipulate the audio signal. The issues we are faced are the choice of the signal encoding schemes, methods for signal visualization and the selection of deployment platform. Additionally, the TTS-interface can also be used creatively to create dynamic facial animation. The project has potential to be additionally extended towards designing the multimodal platforms in order to study the various combinations of audio-visual speech processing, including real-time lip motion analysis, real-time synthesis of models of the lips and of the face, audiovisual speech recognition of isolated words, and text-to-audio-visual speech synthesis in Montenegrin

.

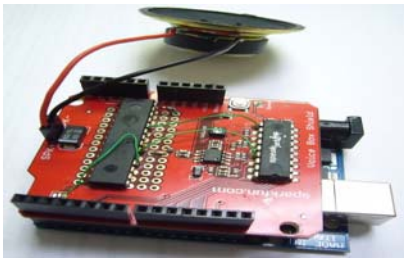

Figure 9. VoiceBox with SpeakJet and TTS256

REFERENCES

[1] X. Huang, A. Acero, H.-W. Hon, Spoken Language Processing, Prentice Hall PTR, 2001

[2] Tanja Schultz (Edit.), Katrin Kirchhoff (Edit.), Multilingual Speech Processing, Elsevier, 2006

[3] Thierry Dutoit , An Introduction to Text-to-Speech Synthesis, Published by Kluwer Academic Publishers, ISBN 0-7923-4498-7, The Netherlands, 1997

[4] Vincent J. van Heuven, Louis C. W. Pols, Analysis and Synhesis of speech: Strategic Research towards High-quality text-speech-generation, Published by Mounton de Grayter, 1993

[5] Shrikanth Narayanan, Abeer Alwan, Text to Speech Synthesis: New Paradigms and Advances, Published by Prentice Hall. Part of the IMSC Press Multimedia Series, 2004-ISBN-13: 978-0-13-145661-7

[6] http://magnevation.com/software.htm

[7] htto://www.magnevation.com/pdfs/speakjetusermanual.pdf

[8] www.speechchips.com

[9] http://beagleboard.org/

New Study Program in Bioengineering and Medical Informatics at University of Defense from Belgrade

Spasic-Jokic V. M.[1,2], Vasilijic S.R..[1], Ninkovic M.B. [1], Vucevic D.B. [1] and Ilic T.V.[1]

[1]Medical Faculty Military Medical Academy,
University of Defence,
Belgrade, Serbia
[2]Faculty of Technical Sciences,
University of Novi Sad,
Novi Sad, Serbia

Abstract— **The paper presents new study program in bioengineering and medical informatics developed at Medical Faculty Military Medical Academy, University of Defense in Belgrade, Serbia, in the frame of BioEMIS TEMPUS activities. The courses were aimed to educate well trained specialists in medical physics, medical engineering and medical informatics and to establish new programs in specialist studies incorporated in continuing professional development programs for professional licensing. This electronic document is a "live" template. The various components of your paper [title, text, heads, etc.] are already defined on the style sheet, as illustrated by the portions given in this document.**

Keywords- bioengineering, medical informatics, modular approach, specialist studies, BioEMIS.

I. INTRODUCTION

Biomedical engineering uses engineering, mathematics and computational tools to simulate and understand real-world medical engineering problems. Bioengineering is an evolving discipline in engineering that involves collaboration among engineers, physicians, and scientists to provide interdisciplinary insight into medical and biological problems. Modern Health Care Services are provided with ever-increasing demands for competence, specialization and cost effectiveness. [1-3]

EEC Directive 97/43/Euratom (Official Journal of the European Communities No L180, 9.7.1997) recognized special groups of, generally, technical professionals whose training and competence enable the development and use of complex techniques and equipment, optimization, quality assurance, including quality control, and other matters relating to diagnostic and therapy techniques including ionizing and non-ionizing radiation protection of patients, staff and general public. Biomedical engineering Departments generally serve a variety of medical specialities as are radiological field (radiotherapy, nuclear medicine, X-ray diagnostics and radiation protection), magnetic resonance and ultrasound imaging, physiological measurements, clinical applications of non-ionising radiations (lasers, ultraviolet light and microwaves), bioengineering, electronics, information technology, general data processing and computer technology [1-3]. The role of biomedical engineers in these areas is expected to increase in the future.

Generally the total number of staff required in hospital depends upon:(i) the range of applications of technical service to medicine; (ii) the scale of organizational and management responsibilities (number of clinics, population served); (iii) the amount and complexity of equipment and procedures used in related clinical specialities; (iv) the number of patients examined and treated in the relevant modalities and the complexities of these examinations or treatments; (v) the load for formal teaching and training and (vi) the level of participation in maintenance, development, research and clinical trials [4]. Minimum staffing levels should be calculated from factors depending both on equipment load, number of patients treated and sophistication of treatments. General guidelines are based upon WTE (whole time equivalent) for assessment of minimum staffing levels for routine clinical work in various medical disciplines [4].

There is no doubt that a biomedical engineer must be an engineer that possess broad knowledge of fundamental engineering and physical sciences' principles, and must be able to apply a multidisciplinary approach to solve problems dealing with diagnostics, treatment, and prophylactics of the patients and population. To arrive at this capability, a student interested in Biomedical Engineering (BME) must be offered a study program that provides specific education and training in biomedical engineering. Problems that biomedical engineers are expected to solve today vary tremendously and this diversification can only be expected to increase further with new and rapidly emerging technologies and demands of the health sector. For this reason any BME study program must provide a sound BME foundation together with specialisation elements within a narrow field of BME, which address the current and future needs of the Society [5-6].

The main objective of our work in the frame of TEMPUS project titled Studies in Bioengineering and Medical Informatics (BioEMIS), was to propose an updated specialization study curriculum in the field of biomedical engineering, in order to meet recent and future developments in the area and address new and emerging interdisciplinary

domains that appear as a result of the R&D progress and respond to the demands of the BME job market. Adoption of the core program structure will facilitate harmonization of studies as well as student and staff exchange in Europe.

II. SPECIALISTIC PROGRAM AT UNIVERSITY OF DEFENSE

A. General

The programs offered at Military Medical Academy (MMA) will emphasize the confluence of basic engineering science and applied engineering with the physical and biological sciences, with emphasis in the areas of biomechanics, cell and tissue engineering, therapy and biomedical imaging. This joining of the diverse scientific fields is complemented by strong academic and research collaboration with various MMA departments.

The specialization programs in biomedical engineering prepare students to apply the principles of engineering and applied science to problems in biology and medicine, to understand the dynamics of living systems, and to develop biomedical systems and devices. Modern engineering encompasses sophisticated approaches to measurement, acquisition, storage and analysis of data, model simulations, and materials and systems identification. These techniques are used in the study of individual cells, tissues, organs, and entire organisms. The increasing value of mathematical models in the analysis of living systems is an important sign of the success of contemporary biomedical engineering activity [3,7].

After achieving the learning outcomes the students should use the acquired knowledge in further education including lifelong learning. Students should be able to communicate effectively in both oral and written form, to participate in social debates pertaining to technology, to carry out independent and multidisciplinary team work, to perform project management duties, and to understand the societal impact of engineering solutions. Students will be able to identify and formulate challenges in the biomedical and health science domains which belong to the biomedical engineers. Students should learn to work as members of multidisciplinary teams and to apply advanced methodologies at the interface between engineering and medical sciences [3, 7-9].

Students will be able to apply engineering expertise to the design of devices, systems, algorithms, software, models, materials, methods or processes in order to meet the desired functional and regulatory requirements for the commercialization of medical devices. They also should be able to design and carry out a research plan to test hypotheses, to analyze and interpret the results in the context of the research, and to report the results according to scientific principles [3,7-9].

B. The Structure of the Specialization Program

The main structure of the study program consists of 5 mandatory courses (core courses) which are evaluated with total 30 ECTS and certain number of elective courses (30 ECTS in total). Specialization work brings 10 ECTS. Core subjects consist of four courses and one research project related to the selected topics. Students are required to choose at least 10 ECTS electives from the selected specialization. Study program and ECTS distribution are given in Table 1 [7,10-11].

TABLE I. DISTRIBUTION OF ECTS

No.	***Course***	***ECTS***	***Structure***	***Note***
1.	Physics and regulatory mechanisms of the human body	5	3+1+1	Mandatory
2.	Introduction to telemedicine	5	2+3	Mandatory
3.	Ionizing and non-ionizing radiation and protection	5	3+1+1	Mandatory
4.	Ethics in biomedical engineering	5	2+2	Mandatory
5.	Research Project related to the selected module	10	8	Mandatory
6.	Processing of Physiological Signals	6	2+3	Elective
7.	Computer Networking	5	2+2	Elective
8.	Medical Imaging Methods in Radiology	5	1+2	Elective
9.	Visualization techniques in nuclear medicine	5	1+2	Elective
10.	Laser application in therapy	6	3+3	Elective
11.	Methods of Radiotherapy	6	2+2	Elective
12.	Non-invasive brain stimulation	4	1+1	Elective
13.	Biomaterials and biocompatibility	5	2+2	Elective
14.	Stem cells in therapy	5	1+1	Elective
15.	Cell biology and immunology for engineers	5	2+2	Elective
16.	Techniques in Molecular Biology and Applications to Gene Expression	6	2+2	Elective

C. Modular Approach

We recognized four study specialization directions as the defining components for the BME specialization program. We have not strictly and officially divided them. In any case, it is possible to identify the four directions: Medical informatics; Bioimaging; Biomedical Engineering in Therapy and Cell and Tissue Bioengineering [11-14].

III. DISCUSION

The development of a biomedical engineering practioner working in health care or in industry, and the maintenance of effective performance depends on three components- Education, Training and Continuing Professional Development.

A. Education

The results of the review of the existing Biomedical Engineering educational programs in Europe have shown that the number and proportion of undergraduate and postgraduate BME programs is increasing. Biomedical Engineering programs experienced a rapid growth after the year 2005 and especially during the last five years. The study identified that Biomedical Engineering programs are available in almost all European countries. Approximately 200 Universities across Europe offer in total more than 300 BME programs, even around 50 % at the level of MSc and specialization study. This results in an increased number of Biomedical Engineers available on the market today. It can be expected that this trend will continue as a response to an increasing demand of health sector and relevant industry demand for BME specialists. This increased demand and rapid emergence of new technologies in biomedical instrumentation both require an interdisciplinary approach to problem solving [2-3,5-6,8].

The results of the survey were then used as the basis for discussions and to facilitate the definition of the core curriculum for BME programs. The contents of a curriculum are usually described by the titles of the courses included in the curriculum. However, the actual contents of courses with the same title can be quite different among the programs. Moreover, the same contents may be covered by different courses in different programs [2-3,5-6,8].

To avoid such ambiguity, it was decided that a more appropriate way to define the curriculum is in term of modules with well defined contents and with no overlap in contents between the modules. For example, the contents of one model can be covered by more than two courses. On the other hand, the contents from different modules can be combined within the same selected topic. The use of modules instead of fix program provides for a higher degree of flexibility in designing new BME specialization study programs.

B. Training

Education is primarily related to the acquisition and integration of knowledge leading to understanding. The outcome is generally evaluated by examination and thesis. Training on the other hand is related to the utilization of understanding leading to competencies. There is strikingly little information on training programmes and how they are delivered and evaluated. It appears, however, that like Biomedical Engineering Education, there is considerable national variation. Engineering scheme is of long standing, organized by IPEM and EFOMP and clearly documented [2-3,5-6,8].

The training should be undertaken in hospital based training centre accredited by country authorities. Training should be divided into Part I, which is basic training and Part II, which shows increasing professional responsibility. Part I training often include the acquisition of a specialization degree and lasts for 1 year. Assessment of the non-academic component of the training is carried out in three ways; by continuous assessment during training, by examination of a portfolio of evidence of training, and by viva voce examination. Part II training normally should lasts for a minimum of 1 year and its competence implies the ability, in most instances, to perform without supervision, to make independent professional calculations and judgments, to supervise junior staff and to provide a service in a specified area of work [2-3,5-6,8].

C. Continuing Professional Development (CPD)

It is the responsibility of professionals to maintain and enhance their levels of knowledge, skills and professional competence throughout their working life. CPD is essentially structured and planned method of doing so. Many professional organizations require those registered with, or accredited by, the organization to show evidence of CPD. This also applies to Biomedical Engineering. The essential features of a CPD scheme should cover: definition of what is considered acceptable CPD activity; CPD registration; CPD record and CPD outcomes of individual activities [1,5,8,15].

IV. Conclusion

Within Europe there is considerable variation in the education and training of Biomedical Engineers at the specialization level. This Report advances a harmonized syllabus and structure for education, supplemented by a notional time-table, showing how the syllabus could be implemented.

Training has not been considered within the framework of the project, but there is even more variability than in education. There is a need to evaluate the national practices to identify best practice and develop a harmonized structure for the training of Biomedical Engineers

Acknowledgment

This study is supported by the grant (BioEMIS, 530423-TEMPUS-1-2012-1-UK-TEMPUS-JPCR) of the European Commission Education, Audiovisual and Culture Executive Agency.

References

[1] Grimes, S.L., The future of clinical engineering: the challenge of change. Engineering in Medicine and Biology Magazine, IEEE, vol. 22(2): p. 91-99, 2003.

[2] S.Grimes (2006): Mission, Function & Organizational Structure of Clinical Engineering Services, Strategic Health Care Technology Associates, www.SHCTA.com

[3] Kolitsi, Z., ed. "Towards a European framework for education and training in medical physics and biomedical engineering", IOS Press: Amsterdam, 2001.

[4] The European Federation of Organisations for Medical Physics (1998): Criteria for the staffing levels in a Medical Physics Department. http://www.efomp.org/ No. 7: Criteria for the Staffing Levels in a Medical Physics Department *(pdf file-40 kB)*, Sept. 1997 [*Physica Medica* **XIII** (1997) 187-194]

[5] Eudaldo, T. and K. Olsen, The European Federation of Organisations for Medical Physics. Policy Statement No. 12: The present status of Medical Physics Education and Training in Europe. New perspectives and

EFOMP recommendations. Physica Medica: European Journal of Medical Physics. 26(1): p. 1-5, 2010.

[6] Christofides, S., et al., "Education and Training of the Medical Physicist in Europe", in World Congress on Medical Physics and Biomedical Engineering, September 7 - 12, 2009, Munich, Germany, O. Dössel and W. Schlegel, Eds. Springer Berlin Heidelberg. p. 1-4., 2010.

[7] Institute of Electrical and Electronics Engineers (2003): Designing a Career in Biomedical Engineering Healthcare Technology Certification Commission (2005): Certification in Clinical Engineering.

[8] European Alliance for Medical and Biology Engineering and Science (2005): Protocol for the training of clinical engineers in Europe.

[9] S. Stankovic, V. Spasic Jokic and M. Veskovic, "Medical Physics Education in Serbia: Current State and Perspectives," Biomedizinishe Technik, Berlin, vol.50, Supl.1/2, pp. 1376-1377, 2005.

[10] L.Christensen: WHO/WFME practical guidelines for allocation and use of ects credits in medical education, WFME, University of Copenhagen http://wfme.org/projects/wfme-publications/76-use-of-ects-credits-in-medical-education/file

[11] ECTS: European Credit Transfer and Accumulation System. http://ec.europa.eu/education/lifelong-learning-policy/doc48_en.htm

[12] Bronzino JD, "The Biomedical Engineering Handbook, Ed 3 Section XX Ethical Issues Associated with the use of Medical Technology", Sections 189-192. CRC Press/Taylor and Francis, Boca Raton FL, 2006.

[13] Monzon JE and Monzon-Wyngaard A, "Ethics and biomedical Engineering education: the continual defiance" in Proc of 31st Annual International Conference of the IEE EMBS, 2009

[14] Syllabus for postgraduate specialisation in nuclear medicine. 2002 Update. European Journal of Nuclear Medicine and Molecular Imaging, vol. 30 (3): p. B1-B2, 2003.

[15] European Alliance for Medical and Biology Engineering and Science (2005): Protocol for continuing education of clinical engineers in Europe.

Wireless Communication Architecture of a Robotic Services System in an Unknown Environment

Eva Cipi,
Department of computer sciences
University of Vlora "Imail Qemali"
Vlore, Albania
eva.cipi@univlora.edu.al

Betim Cico
Department of informatics engineering
Polytechnic University of Tirana
Tirana, Albania
betim.cico@gmail.com

***Abstract*—This paper is focused on presenting the architecture and the implementation of a semi autonomous simulation based system which is able to navigate into a partially known environment. Most of robotic agent does not support the mobility according the requirements of application. Our work of implementation provides a semi autonomous system which is guided by an operator performing several services with better quality and low costs. The design is a module based architecture which supports the robot navigation using simulation offline software and on line at the moment when the agency percepts obstacles. The robot changes states of its mobility in real time building a new strategy to achieve the normal path which is received from a simulator that executes and communicates the path calculated by a simple navigational algorithm in the virtual static environment.**

Using a communication protocol between robotic unit and simulator software, it is possible to correct data and to improve the system behavior using a wireless communication. The physical system is tested in a laboratory environment. It is situated in a environment which is the same designed in the simulator interface. The robot updates its coordinates in the virtual environment and the simulation runs exactly according the navigation algorithms until the sensor module transmits to simulation software new data of obstacle presences. We evaluate the performance of the system and the results confirm an improved behavior of robotic agent in extreme situations of dynamic environment.

Keywords—semi-autonomous system, simulation offline, simulation on line, robotic agent, module based architecture, communication protocol

I. INTRODUCTION

Autonomous applications are those in which the user is interested in the results of self processing large amounts of data in several distributed locations in order to achieve some goals. These applications are extremely useful in areas such as intrusion detection systems, self service system or unknown environment map analysis.

Robotic agent systems appear to be the most feasible solution for their implementation. In these systems, autonomous software entities (robots) may move across an environment of execution platforms. The application is supported by an architecture which seems to be based on two modalities of agency work.

In this paper aims to study a semi-autonomous system that operates into a partially known environment which is able to bring services of transportation using robotic agents to every point addresses generated from a central management system. The robots move autonomously into a physical environment area controlled by an operator but the robot path has been fixed over a virtual partially known environment. A wireless communication system supports the connection between the central system and robotic entities. There are two channel transmissions in separated frequencies. The system passes from one modality (simulation off-line) in another one (simulation online) when the robots percept the presence of unknown obstacles.

The work aims to contribute on bringing new solutions on solving the problem of the navigation in partially unknown environments in order to avoid collisions with obstacles. Another element is the designing process tents to be easier, programming intelligent behaviors in virtual environments. The robotic entity is equipped by a set of sensors to sense obstacles. We have tested several behaviors of the system in different environments and the performance of the system was very good. In the Fig. 1 you can see a robotic prototype used to be programmed as one of subsystems.

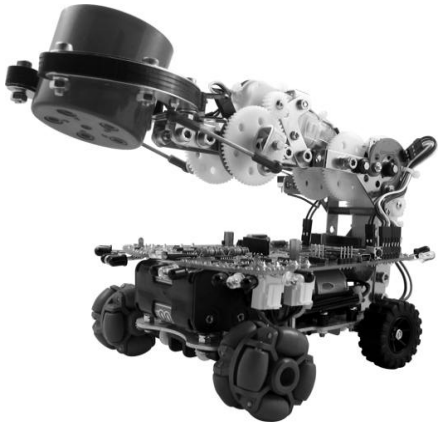

Fig. 1. The prototype used in this work

II. RESEARCH OBJECTIVES

In this research we propose a modulated architecture agent based approach to develop complex applications. The approach aims to increase the performance of systems on managing autonomously the dynamism and changes of the application environment. The general idea of self-management is to endow computing systems with the ability to manage themselves according to high-level objectives specified by humans. Researchers divide self-management into four functional areas [3]:

- Self-configuration: systems automatically configure components to adapt themselves to different environments;
- Self-healing: systems automatically discover, diagnose, and correct faults;
- Self-optimization: system automatically monitor and adapt resources to ensure optimal functioning regarding the defined requirements; and
- Self-protection: they identify and protect against attacks.

The environment will occupy an important role in this project. An agent based system which pretends to be self managed must take the appropriate actions based on a sensed situation of the environment. This requires some functionalities of system such as environment monitoring, decision making, and action execution. This requires the coordination of behavior of agents inside of the same environment. Another positive side is that same approach can be used in different applications. Here we show two modalities of the same system. This helps designers and developers to resolve effectively problems of software engineering processes.

III. MULTIAGENT SYSTEMS AND THE SOFTWARE ARCHITECTURE

A multi agent system provides the software to solve a problem by structuring the system into a number of interacting autonomous entities embedded in an environment in order to achieve the functional and quality requirements of the system. In particular, a multi agent system structures the system as a number of interacting elements in order to achieve the requirements of the system. This is exactly what software architecture is about. [4] defines software architecture as:

Software elements (or in general architectural elements) that provide the functionality of the system, while the required quality attributes (performance, usability, modifiability, etc.) are primarily achieved through the structures of the software architecture.

Typical architectural elements of multi agent system software architecture are agents, environment, resources, services, etc. The relationships between the elements are very diverse. In short, multi agent systems are a rich family of architectural approaches with specific characteristics, useful for a diversity of challenging application domains. [5] There are applications that provide different levels of complexity and various forms of dynamism and change. The architecture for many of them is a set of agents, for reuse, they can serve to develop software architectures transporting them as ready entities with specific attributes such as: each agent has incomplete information or capabilities to solve the problem and, thus, has a limited viewpoint and the computation is an asynchronous process.

A multi agent system consists of a (distributed) environment populated with a set of agents that cooperate to solve a complex problem in a decentralized way. [6] Behavior-based action selection is driven by stimuli perceived in the environment as well as internal stimuli. Situated agents employ internal state for decision making relates to:

- Planning off line (static information of the system).
- Planning on line (dynamic information related to the changes of the environment); or issues internal to the agent.
- Environment encapsulates resources and enables agents to access the resources. [7]

IV. A ROBOTIC SYSTEM: CASE STUDY

The system we propose, is a prototype which has to transports loads from one point to another performing intelligent behaviors being oriented themselves of service points in a partially known market place, completing successfully its tasks, avoiding the collision with obstacles displayed dynamically and creating unexpected situations in its path, which is been pre planed by a simulator software. The tasks are generated by the information agent which is part of a central information system, typically for a business management program discussed in our past works. The task is composed out of some processes like receiving order from service agent acquiring the first service point address, moving to first service point, receiving the second service point address, moving to the second service point. In order to execute this task, the system has to perform:

- Route assignment: paths are generated by the information systems and have to be assigned to robot vehicle that can execute them.
- Routing: the robot must route efficiently through the environment layout of the warehouse when executing transports.
- Gathering traffic information: although the layout of the system is static, the best route for the robot in general is dynamic, and depends on the actual traffic condition. Using visual sensors, the robot routes efficiently without collision with other objects. [8]

This model integrates the environment and agent integrating mechanisms of agent adaption. We have divided the model in two parts: environment and situated agent. The environment model consists of a set of modules with flows between the modules. The modules represent the core functionalities of the environment. The model consists of two main modules:

- the deployment context (referred to the given hardware and software and external resources with

which the multi agent system interacts (sensors and actuators, a printer, a network, a database, a web service, etc.).and

- the application environment(refereed to the part of the environment that has to be designed for an application) .

The data flow diagram is showed in the Fig.2.

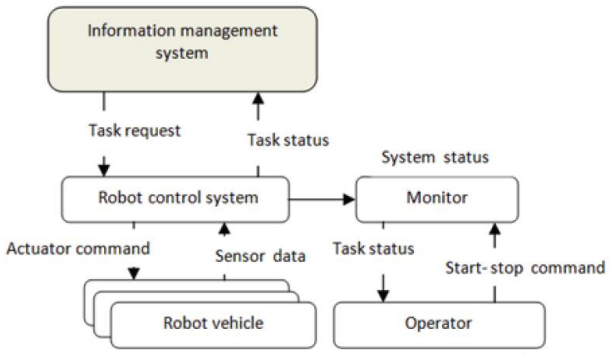

Figure. 2. The diagram of the robotic system for semi-automatic services.

The robotic agent model consists of four modules with flows between the modules. Knowledge module provides the functionality to access and update the agent's current knowledge. Sensing module receives perception requests from the Decision Making and Communication modules to update the agent's knowledge about the environment. Decision making and communication module use the agent's current knowledge to make appropriate decisions. The communication module writes the location in the agent's current knowledge which in turn will be used by the decision making module to move the agent efficiently towards the destination. Decision Making provides the functionality to an agent for selecting and invoking influences in the environment. Decision making consists of two basic functions: Influence selection and actuator execution. To select appropriate influences, an agent uses a behavior based action selection mechanism extended with roles and situated commitments. Execution provides the functionality to invoke selected influences in the environment.

A. The Wireless Communication

The Fig. 3 is a view of environment communication of robotic vehicles with the simulation software. We consider robotic agent as independent entities that communicate with a central management system which generates transportation tasks through a generated path from a simulated known environment but not updated. The control of task distribution is centralized but the vehicle path is not completed autonomous. We consider the presence of collision point but this is provided by robotic agent which changes its status in simulation on line. The central system sends instructions according the simulation software which pretends to know the environment in its first statement. All the exchanged messages are done through a wireless module of communication.

Here is been presented some technical features of the Wireless device which is composed by two modules:

- Receivers RX-B1 (433 MHz and 315 MHz) and
- Transmitters TX-C1 (433 MHz and 315 MHz)

Each message is transmitted as:

- 36 bit training preamble consisting of 0-1 bit pairs
- 12 bit start symbol 0xb38
- 1 byte of message length byte count (4 to 30), count includes byte count and FCS bytes
- 2 bytes FCS, sent low byte-hi byte

Everything after the start symbol is encoded 4 to 6 bits, Therefore a byte in the message is encoded as 2x6 bit symbols, sent hi nibble, low nibble. Each symbol is sent LSB it first. The Arduino clock rate is 16MHz => 62.5ns/cycle. For an RF bit rate of 2000 bps, need 500microsec bit period. The ramp requires 8 samples per bit period, so need 62.5microsec per sample => interrupt tick is 62.5microsec. The maximum message length consists of (6 + 1 + VW_MAX_MESSAGE_LEN) * 6 = 222 bits = 0.11 seconds (at 2000 bps). Throughout the range there are nulls and strong points due to multipath reflection. Similar performance figures were found for DR3100. 9000bps worked. Arduino and TX-C1 transmitter draws 27mA at 9V. Arduino and RX-B1 receiver draws 31mA at 9V.

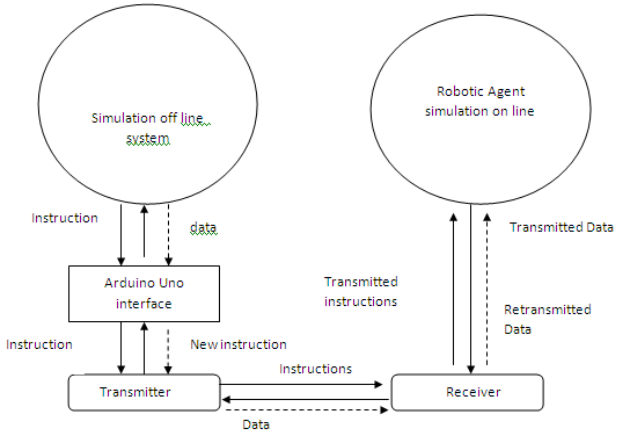

Fig. 3. Receiver and transmitter modules of the wireless communication

Here we present two protocols which allow the system to send commands to robots and to receive data from its. There are two simple protocols which are used to link the simulation software with the physical entity. These messages are transmitted in different frequencies to avoid signal interferences and message errors:

- **Receiving data protocol**

```
int GetMessage()
uint8_t buflen = VW_MAX_MESSAGE_LEN;
   if (vw_get_message(buf, &buflen))
   {
```

```
        int i;
    char inStr[25];
    char inChar=-1;
        for (i = 0; i < buflen; i++)
        {
      inChar = buf[i];
      inStr[i] = inChar;
        }
    inStr[i]='\0';
  }
  return  command;
}
```

- **Sending commands protocol**

```
void SendMessage(String msgToSend)
{
  int command=-1;
  uint8_t buf[VW_MAX_MESSAGE_LEN];
{
 char msgtmp[25];
 msgToSend.toCharArray(msgtmp,25);
  vw_send((uint8_t *)msgtmp, strlen(msgtmp));
  vw_wait_tx();
}
```

The operator can use a robot interface to view the messages which are transmitted in both sides of the communication. The fig. 4 presents the interface which is used also to give basic manual command as *connect* or *stop* to the robotic prototype.

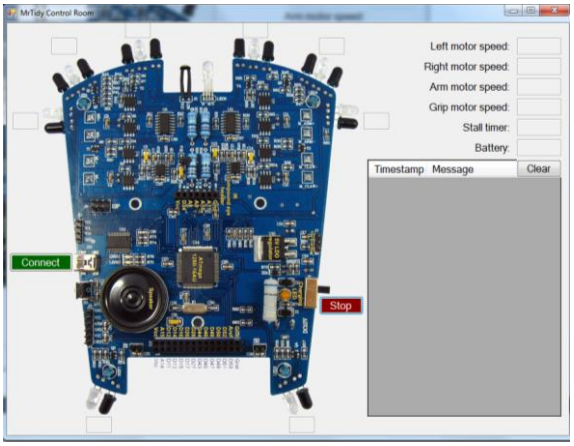

Figure 4. The view of the robot interface

B. *Off line simulation*

Two robotic agents operate in a two dimensional environments, first they act into a space without obstacles; then the obstacles are generated randomly by mouse clicks; i.e. in random sizes and distance between them, and their black shapes are circular. We can add other obstacles during the simulation time. A great number of obstacles increases the complexity of the virtual world but this required more calculation power to manage the dynamic information. At each step, agent should perform two actions:

- interaction with simulation software if agents do not sense an obstacle,
- interaction with environment if agents sense one,
- Orientation to choose the next step.

The final objective is to complete instructions came from the central management system. For this purpose, we combine two types of agent behavior in order to achieve the goal. The system is designed to control constantly the changes of the environment avoiding the failures. Our system is designed to have a set of sensors. The agent can percept obstacles and can overview its distance from them. Hanging around the motion map, the agent can discover invisible areas behind the obstacles to find the target which is not in the visible area. Here he finds the motivation to change position for a new state with purpose to reach his objective. He starts his movement following a predicated plan in base of the visual sensor information and simulation instructions.

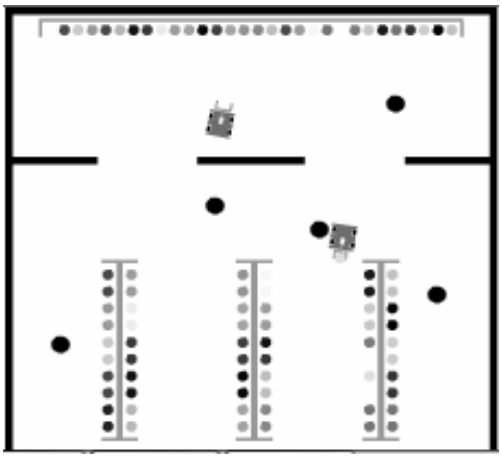

Figure. 5 The virtual environment of the simulation off line with obstacles.

This is an important moment for showing the new state after the action execution. These actions are divided in two categories: normal actions that change the environment sensing actions that accumulate information over the environment to make decisions. In intelligent systems during the execution the agent decides itself about the state of conditions: true or false.

C. *The simulation on line- Explorer modality*

In this modality, the robotic agent does not know the environment. The agent navigates choosing an explorer path until it senses the presence of an obstacle. In this case, the robot transmits data of its location and calculates the distance from the perceived obstacle. Each new state of the robot is calculated and it is showed in the virtual environment reaching the knowledge about it. The robot must perform different ways within the restricted environment by exploring the presence of obstacles and transmitting information about this.

We finally assembled and integrated graphic information of calculated points, showing how the environment becomes more complete and revealing. In the figure 6, we have showed the simulation view. There are two windows, the first is simulation view that would be in a physic environment and the second is the map that the robot makes exploring the world. The yellow obstacles are invisible for a human operator.

After 5 min of simulation, the robotic agents have transmitted data to the environment map window translated in some red points which are points of addresses where the robot senses an obstacle. The third view is the suggested map after discovering process. The same process is for the fourth view after 12 minutes. The simulation integrates more red points considering the distance criteria between them. The green area is considered as obstacle area. The simulation software calculates the rate of error made as the division of green surface with white surface in percentage.

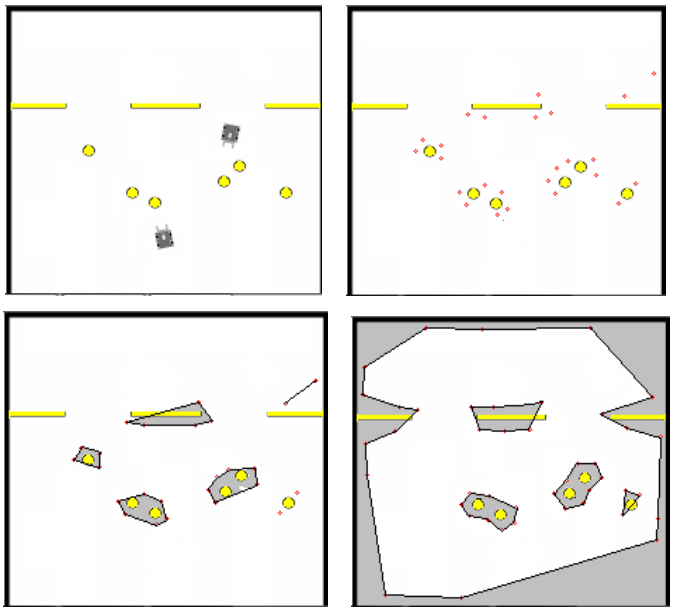

Fig. 6 View of Explorer Simulation modality

$$\text{Error\_rate} = 1 - S_{green} / S_{white} \ \%$$

In the first case, the error rate is 94 % considering the walls of environment while, after 12 minute , this error is 66%. The robotic agent needs much more time to discover exactly the map.

V. CONCLUSIONS

During the simulation, we can observe the agent environment relations and their dependencies:

- We have designed a software system that is capable to integrate successfully the execution ability of complex tasks using reflexive capacities needed to manage uncertain situations in dynamic environment.
- We observe temptations for more reactive behaviors in expected situations.
- The speed of mission is another of agent exigencies. However the environment changes with certain speed and the agent has not time enough to make a perfect decision and to choose accurately the next action.
- Time in disposition do not compromises the agent performance in the dynamic environment.
- The simulation software that we have designed provides a simple way to study the complex interactions between different types of environment and agents.
- We were interested to study the stability on making decisions of the system. Our attention was focused on how much security offers an agent based system in difficult situations.
- Many different types of robotic agent behaviors have been introduced. We could observe the agent efforts to reach the goal and the results were interesting about agent intelligence.
- Combining simulation off line with simulation on line, the agent can perform better behaviors in a partially known environment giving a new solution in software engineering.
- In further research it would be interesting to introduce new types of environment making them more complex. We aim to extend agent based systems on modeling hierarchical structures of control system and other industrial applications

REFERENCES

[1] Cipi, E., Cico, B.: Information Agents as a New Paradigm for Developing Software, Applications in Database Systems, Conference Proceeding DSC 2010, Thessaloniki, Greece , Vol. 1, 514–519, (2010).

[2] Bass, L., Clements, P., Kazman, R.: Software Architecture in Practice., Addison Wesley Publishing Comp,95--97, (2003).

[3] Kpheard, J., Kephart., J., Chess, D.:The Vision of Autonomic Computing. IEEE Computer Magazine, No. 36(1), (2004.

[4] Bandini, S., Manzoni, S..and Simone. C.: Dealing with Space in Multiagent Systems: A Model for Situated Multiagent Systems. In 1st International Joint Conference on Autonomous Agents and Multiagent Systems. ACM Press, (2002).

[5] Buchmann, F., Bass, L.: Introduction to the Attribute Driven Design Method. 23rd International Conference on Software Engineering, IEEE Computer Society.Toronto, Ontario, Canada, (2001) .Intelligent Agents Solutions to Improve Strategic Level of Self-Management... 525

[6] Clements, P., Kazman, R., Klein, M.: Evaluating Software Architectures: Methods and Case Studies. Addison Wesley Publishing Comp. (2002).

[7] Steegmans, E., Weyns, D., Holvoet,T., Berbers, Y.: A Design Process for Adaptive Behavior of Situated Agents. In Agent-Oriented Software Engineering V, 5th International Workshop, AOSE, New York, NY, USA, Lecture Notes in Computer Science, Vol. 3382. Springer, (2004).

[8] Weyns, D., Holvoet, T.: Multiagent systems and Software Architecture. In Special Track on Multiagent Systems and Software Architecture, Net.ObjectDays, Erfurt, Germany, (2006).