VILNIUS UNIVERSITY

GEDIMINAS GRICIUS

DEVELOPMENT OF MULTI-AGENT BASED METHODS FOR INTEGRATION OF SMALL-SCALE EMBEDDED SYSTEMS

Summary of Doctoral Dissertation

Physical Sciences, Informatics (09 P)

Vilnius, 2015

The doctoral dissertation was prepared at the Institute of Mathematics and Informatics of Vilnius University in 2010 to 2014.

Scientific Supervisor

prof. dr. Dalė Dzemydienė (Vilnius University, Physical Sciences, Informatics – 09 P).

The dissertation will be defended at the Council of the Scientific Field of Informatics of Vilnius University:

Chairman

prof. dr. Romas Baronas (Vilnius University, Physical Sciences, Informatics – 09 P).

Members:

prof. dr. Albertas Čaplinskas (Vilnius University, Physical Sciences, Informatics – 09 P),

prof. dr. Jānis Grabis (Riga Technical University, Physical Sciences, Informatics – 09 P),

prof. dr. Saulius Gudas (Vilnius University, Physical Sciences, Informatics – 09 P) ,

prof. dr. habil. Artūras Kaklauskas (Vilnius Gediminas Technical University, Technological Sciences, Informatics Engineering – 07 T).

The dissertation will be defended at the public session of the Scientific Council of the Scientific Field of Informatics in the auditorium number 203 at Vilnius University Institute of Mathematics and Informatics, at 1 p.m. on the 25 of September, 2015.

Address: Akademijos st. 4, LT-08663 Vilnius, Lithuania.

The summary of the doctoral dissertation was distributed on the 25th of August 2014.

A copy of the doctoral dissertation is available for review at the Library of Vilnius University or on this website: www.vu.lt/lt/naujienos/ivykiu-kalendorius.

VILNIAUS UNIVERSITETAS

GEDIMINAS GRICIUS

DAUGIAAGENTINIŲ SISTEMŲ KŪRIMO METODŲ IŠVYSTYMAS NEDIDELIO NAŠUMO ĮTERPTINIŲ SISTEMŲ INTEGRAVIMUI

Daktaro disertacijos santrauka

Fiziniai mokslai, informatika (09 P)

Vilnius, 2015

Disertacija rengta 2010–2014 metais Vilniaus universiteto Matematikos ir informatikos institute.

Mokslinė vadovė

prof. dr. Dalė Dzemydienė (Vilniaus universitetas, fiziniai mokslai, informatika – 09 P).

Disertacija ginama Vilniaus universiteto Informatikos mokslo krypties taryboje:

Pirmininkas

prof. dr. Romas Baronas (Vilniaus universitetas, fiziniai mokslai, informatika – 09 P).

Nariai:

prof. dr. Albertas Čaplinskas (Vilniaus universitetas, fiziniai mokslai, informatika – 09 P),

prof. dr. Jānis Grabis (Rygos technikos universitetas, fiziniai mokslai, informatika – 09 P),

prof. dr. Saulius Gudas (Vilniaus universitetas, fiziniai mokslai, informatika – 09 P),

prof. habil dr. Artūras Kaklauskas (Vilniaus Gedimino technikos universitetas, technologijos mokslai, informatikos inžinerija – 07 T).

Disertacija bus ginama viešame Vilniaus universiteto Informatikos mokslo krypties tarybos posėdyje 2015 m. rugsėjo 25 d. 13 val. Vilniaus universiteto Matematikos ir informatikos instituto 203 auditorijoje

Adresas: Akademijos g. 4, Vilnius, Lietuva.

Disertacijos santrauka išsiuntinėta 2015 m. rugpjūčio mėn. 25 d.

Disertaciją galima peržiūrėti Vilniaus universiteto bibliotekoje ir VU interneto svetainėje adresu: www.vu.lt/lt/naujienos/ivykiu-kalendorius.

.

INTRODUCTION

Scope and Topicality of the Problem

Modern artificial intelligence systems (AIS) are complex and integrate different components of hardware and software that are supposed to communicate with each other and to achieve a common objective. In order to ensure operationalization of intelligent systems that are operating in a real-time and at the complex external conditions, it is necessary to deal with the architecture of heterogeneous components and synergy issues, to offer methodology for architectural design and system implementation. This doctoral thesis is devoted to integration of heterogeneous small-scale embedded systems (with limited computing power, memory capacity, etc.) into joint integrated system architecture using the paradigm of multi-agent systems' development.

Embedded systems class include multi-purpose sensors and actuators that are based on special microcontrollers and are designed for specialized functions. They can be integrated into a variety of computerized products: appliances, devices. There is increasing number of successful examples and fields of application of such systems such as control of mode of the car, remote monitoring of a road or building condition, patient's condition monitoring by means of special medical equipment, mechatronic gearing and robotic control.

In most cases, operation of embedded systems is limited by computer performance capabilities. Requirements for these systems are to be as small size as possible, not to use a lot of energy, to use minimum memory resources of the computer, not to use large amounts of RAM, etc. Small-scale embedded systems are characterised with limited CPU power (1-100 Mips), limited program memory (1-256 KB), limited operational memory (64-16384 bytes). These systems are usually composed of a microcontroller (-s), sensors or other input devices, communication interface and indication or output device, but they still must operate as long as possible, using as low-power and yet, at times, to carry out their functions under unfavourable environmental conditions. It is also preferred that such systems would work in wireless network conditions.

Analyzing technical components of embedded systems for real-time process monitoring, a number of issues arise: the integration of these components, assurance of their interaction, selection of wireless network protocols and assurance of their work during interaction of separate wireless network components. It is important to identify the best ways and methods so distributed heterogeneous systems could be merged into a single,

comprehensive, intellectual approach based system, also, to ensure the functioning of the system under specific conditions when certain circumstances do not allow using high power systems. Developing such systems, behavioral aspects that have to be ensured become very important, i.e. their autonomy, responsiveness to the environment and decision-making. Integrating individual characteristics required for embedded systems into a single system, it is necessary to deal with the issue - how they will be able to communicate and interact. Multi-agent systems' development methods allow designing systems, that are characterized by dynamism, easy scalable distribution and the ability to connect heterogeneous elements into one entirety, therefore, the multi-agent systems paradigm that was chosen for this work appeared to be appropriate for integration of our aperture and software systems. These systems are becoming more distributed and decentralized and combining components into a single system that share common objectives and can work in a wireless network, the following problems are encountered: how to ensure the functioning of different interaction protocols, how to integrate the individual components that were implemented while using different physical platforms and different software tools.

Communication and cooperation of separate embedded systems become a more relevant and widely discussed topic. In this paper, the methodology for the construction of such systems using agent system development paradigm is suggested. Basically, the main problem to be solved is how to represent real embedded systems into existing models of multi-agent systems, i.e. how to implement the desired properties of embedded systems applying methodologies of multi-agent systems' development.

In addition, the requirements for these systems are formulated: they have to be independent, active and communicative, but at the same time not receptive to the calculation and energy resources. Therefore, the question is how to efficiently apply widely used methodology of development of multi-agent systems for creation of small-scale embedded systems in networked communication and cooperation.

The system architecture proposed in this work would be characterized with the proactiveness and capacity to adapt, thus increasing the resilience of the system while working in complex conditions. The field of application of such systems could include marine research and environmental monitoring; in this case systems should be working in saline water, at high waviness, wind changes and water freezing. Moreover, application of multi-agent technologies would give opportunities for dynamic expansion of these systems according to the needs.

Relevance of the problem

Software agents (as autonomously operating computer programs) and multi-agent systems composed from them are a rapidly developing software engineering and artificial intelligence branch. The possibilities of their application are well known from such systems as e-advisors, e-mail filters up to sophisticated systems that are able to manage complex technical systems and even space ships. Multi-agent systems' development methods allow designing systems that are characterized by dynamism, easy scalable distribution and the ability to connect heterogeneous elements into one entirety, therefore, application of multi-agent systems paradigm in machinery and software systems integration are analyzed in many scientific publications e.g. Carrasco, 2010; Ostaševičiūtė 2007. When analyzing and solving complex problems, a special role is given to the agent and multi-agent system development phases (Borodini, 2007). Agent means an abstract entity that can solve a particular or a partial problem while several agents can be integrated into a single system and while operating together they can deal with complex problems. Hereinafter the main provision that one embedded system will correspond one agent will be followed. For the realization of traditional multi-agent systems high-level interpreted programming languages (usually JAVA) with substantial computing capacity are applied, but these languages, due to the relatively large requirements for computer resources, are not suitable for the realization of multi-agent systems in groups of small-scale embedded systems.

Combining several embedded systems into one, so that they would seek for a common goal, we are facing such substantial problems: different interaction protocols, subsystems implemented in different physical (computer) platforms with different logic (programmable) measures. For example, sensors, performing environmental monitoring role, are very different due to different physical nature of observed state variables and parameters (from the temperature measurements till measuring the concentrations of chemicals), moreover, their emitted signals are also different, that is why different hardware and program acquisition, analysis and transmission methods are required for their processing and transmission. Each component of such heterogeneous embedded system can have its own targets, but at the same time they also take part in the system and contribute to the overall objective of the system. To ensure the work of heterogeneous systems, it is necessary to solve resource allocation objectives. They are becoming more distributed and decentralized, therefore, combining components into a single system that share common objectives and can work in a wireless network, the following problems are encountered: how to ensure the functioning of the different interaction protocols, how to integrate the individual components that were

implemented while using different physical platforms and different software tools (Jamont et al., 2013). These systems have to meet these generalized requirements: the system should be able to collect information from the surrounding environment, to operate autonomously, to analyze the situation and changes in the environment, to take independent decisions and to implement them. Therefore, their communication and cooperation require efficient operation protocols of wireless networks that enable reliable interaction between small-scale embedded systems. It is important to develop identification methods for various situations in these systems, but due to a high volume of such work, it will be limited to identification of monitoring problems only for some parameters.

Heterogeneous embedded systems are characterized by higher hardware and software complexity. They require common platform for hardware and software supply, also, separate components that could be combined only at the final step of the development process. These systems often require to create encoding-decoding algorithms integrated in a technical level, transformation algorithms, data transfer protocols (such as TCP / IP) and network drivers, that would reduce the load on the CPU for additional calculations. Real-time operating systems usually ensure the fulfilment of operation of all application software.

Object of Research

Methods and software tools that allow developing an interaction between software agents in small-scale embedded systems networking.

Aim and Objectives of Research

To design and implement small-scale embedded systems communication and collaboration platform based on developed software agents interaction methods and software, also to suggest and experimentally test its prototype, applying it to the Baltic Sea hydrometeorological data monitoring. To achieve the aim, the following objectives are considered:

1. To investigate the structure, functions and properties of multi-agent systems (MAS), to analyze the existing methodologies of their development, and to suggest the method suitable to integrate small-scale heterogeneous embedded systems, taking into account the restrictions and constrain requirements for their application.
2. To perform a comparative analysis of frameworks and implementation platforms for multi-agent systems development and, based on that, to indicate

and select the functional properties and implementation features most suitable for small-scale heterogeneous embedded systems.

3. To perform an analysis and a computer experiment aimed at proper selection of a real-time operating system architecture that would fit best for the implementation of MAS based small-scale embedded systems.
4. To suggest a methodology and provide a platform for implementation of heterogeneous MAS in small-scale embedded systems.
5. To verify the proposed methodology and solutions by implementing the sea hydrometeorological data monitoring system capable to functioning under changing climate conditions; to perform experimental assessment of the efficiency of the created system.

The research methods

In the analytical part of the dissertation the scientific literature analysis, MAS design methods and software analysis are presented, it was carried out using qualitative research methods: search for information, systematization, comparative analysis and generalization of collected material. Based on the results of these studies multi-agent systems development methodology, platform for MAS development and other software tools and techniques for the implementation of the stated goal of this work have been chosen.

When designing architecture of the real-time multi-agent platform for the small-scale embedded systems, constructive research method was applied. This method is the investigation procedure, generating innovative constructions (models, charts, methods, algorithms, etc.) to solve real-world problems and using them to make a contribution to the theory of the examined discipline (Kasanen 1993; Vaira 2012). This study covers the analysis of real-time OS architectures for small-scale embedded systems, the design and development of multi-agent platforms, furthermore, new methods and models, enabling the application of MAS in this type of systems were suggested.

In order to conduct validation for the proposed architecture of software agents for small-scale embedded systems, the experimental research was carried out. During the study based on the procedure described, the integrated multi-agent buoys system, performing hydrometeorological observations, was created. Experimental study analysed the characteristics of such a system as data transmission amount and system adaptability to rapidly changing climatic conditions. The results were compared with the characteristics of currently used buoys systems.

Results

Investigation of interaction methods of many agents and analysis of application of real-time small-scale operating systems enabled to assess that the real-time operating system FreeRTOS is best suited for the management of such systems with the needs of low memory resources (e.g. 16-20 MIPS computing speed, 2-8 MB SDRAM memory type, 32-256 KB of flash memory).

For communication of small-scale embedded controllers that function as separate agents of multi-agent system, the most suitable appeared the ZigBee Mesh wireless network protocols. They have been assessed and their suitability for small-scale embedded systems integration into a single wireless network system was determined.

The suggested sophisticated infrastructure and its design methodology for many agents' interaction systems were based on PROMETHEUS and DIAMOND multi-agent system development principles, that seemed to be the best for integration of small-scale embedded systems into common wireless network architecture.

Integrated functioning of the system has been experimentally tested, implementing in it many features of communication and cooperation in agents' system. It was shown that the selection of proposed method is more suitable for integration of small-scale embedded systems than other existing methods. During the experiment, practical significance of proposed and implemented solutions was demonstrated. It was shown that this type of system can be applied for solving ecological monitoring tasks because they can successfully work in a realistic environment and climatic conditions of the Baltic Sea.

Scientific Novelty

The methods and their application methodology suggested in this work extend development capabilities of multi-agent intelligent embedded systems for integration of limited potential embedded systems. Most of the proposed agent system development platforms are designed for development of a high capacity embedded systems that are operating under stationary conditions and with high data storage (computer memory). During their realization, a high-level programming languages (e.g. JAVA) are mainly applied, but they are not suitable to ensure the operation of small-scale embedded systems. Methodology proposed in this work is based on the principles of many agents' interaction and is designed for integration of distributed and heterogeneous embedded systems. Each embedded system, as individual component for certain processes monitoring, may have its own specific sensor, microcontroller

and other elements. Integration of embedded systems into a common system based on artificial intelligence techniques have helped to implement a targeting and interaction development techniques, also, use and apply capabilities of the interactive and functional agents. This gives an assumption that scientific results obtained by the author will benefit in the further development of artificial intelligence systems, in particular with the development of management systems for smart facilities.

Designed agent platform provides opportunities not only to facilitate and improve the physical realization of multi-agent systems, but also to use agent technologies in heterogeneous embedded systems. Furthermore, at the same time it ensures the compatibility between agents-components and expansion possibilities for such systems.

Defended Propositions

1. For communication and cooperation of small-scale embedded systems in heterogeneous environments, it is appropriate to apply a modified software agent systems development methodology DIAMOND.
2. In a complex integrated embedded systems, which consist of many different heterogeneous embedded systems (microcontrollers), it is more appropriate to use distributed multi-agent systems than a centralized agent system, located in a single computer; while for individual components (systems) use a wireless communication protocol DigiMesh.
3. The suggested methodology and the development platform are adapted for development of small-scale embedded systems. Also experimentally tested for development of meteorological phenomena monitoring system, which reduces the quantity of data transfer and makes the system adaptive - provides the system the adaptivity properties.

Practical Importance

Microcontroller capabilities are growing much faster than their price, so the embedded system are integrated into devices, machinery, technological equipment and in this way determines their unique and competitive characteristics. The results of this work can be successfully used in practice for development of intelligent systems that can operate in extreme conditions.

The methodology for multi-agent small-scale embedded systems development was practically implemented and experimentally tested; the technological platform was designed, experimentally tested and evaluated proving the validity of this methodology.

Suggested multi-agent collaboration system and its development were based on PROMETHEUS multi-agent systems platform development principles that seemed the best for small-scale embedded system integration into a joint wireless network architecture.

The functioning of the integrated system while implementing in it many features of communication and cooperation of the agents system was experimentally tested. It was shown that such methodology was the best for integration of small-scale embedded systems. Also, it was demonstrated that these systems can work in the real environmental and climatic conditions of the Baltic Sea. During the research, a real testing platform in Klaipeda University scientific laboratory was created. The results presented in this paper were used in the following MTEP projects:

- "Development of embedded systems' laboratory equipment" (No.: VP2-1.3-ŪM-05-K-02-023) project partly funded by European Regional Development Fund (ERDF)
- "Customer information techniques analysis and development of adaptive information system" (No.: VP2-1.3-ŪM-05-K-03-475), project partly funded by European Regional Development Fund (ERDF)
- "Baltic Sea hydrometeorological monitoring buoys data transfer systems modernization" project funded by Research Council of Lithuania (2014).

Approbation and publication of the research

The main results of this dissertation were published in 7 scientific papers: 4 articles in a journal abstracted in Thomson ISI Web of Science database; 3 articles in the proceedings of scientific conferences. The main results of the work have been presented and discussed in 6 international and 2 national conferences..

The scope of the scientific work

The work is written in Lithuanian. It consists of 5 chapters, and the list of references. There are 119 pages of the text, 35 figures, 10 tables.

1. MULTI-AGENT SYSTEMS AND DEVELOPMENT METHODOLOGIES

Software agents and multi-agent systems formed from them are a rapidly evolving software engineering and artificial intelligence branch. **The software agent** is defined in the literature as an autonomous part of the software that performs user instructions or identification of programmed situations and its management solutions (O'Brian, 1998; Nwana, 1996; Wooldridge, 2006). Software agents are relatively new artificial intelligence tools for creation and analysis of sophisticated computer programs. Depending on the types of software agents they can learn, draw conclusions, communicate with each other in order to achieve common goals, to change the location.

Multi-agent system is an expansion of component programming paradigm, which consists of interacting agents. Multi-agent systems (MAS) can be described as a connected network of problem solvers (agents) who work together while solving the problem, which cannot be solved individually (Wooldridge, 1995). These systems are more efficient and have more advantages compared to a single agent (monolithic) systems: problems are solved faster by exploiting parallelism, transferring of high-level partial solutions (rather than raw data), the amount of data transferred is reduced and the higher credibility of the system is achieved, allowing to take over the positions of non-performing agents (Moulin, 1996). MAS is widely used for operative real time process management and smart electronic services infrastructure development.

After deciding to use the agent technology for system development, it is important to choose the appropriate methods and techniques to achieve this objective. We will review some of the methodologies for agent systems development that are offering agent system development and design techniques. When choosing a method, it is important to assess both: the essential characteristics of object created and the key features of the method. As each system has its specific requirements set, they indicate which evaluation criteria are the most important and must be ensured during the development process.

Embedded systems with hardware that do not only concern software components include the environment in which they operate. Due to the fact, these systems require integration of software elements and the hardware (e-cards, sensors, actuators).

Methodology for creation of the agent systems should consist of the following stages: analysis, architecture design, detailed planning, realization as well as the operation and maintenance that include possibility to redesign the system during the operation if necessary. Together, such methodology should maintain specifications of real-time and reliability requirements.

Six methodologies of agent system development were analysed: GAIA (Zambonelli, F. 2003), PASSI (Cossentino M., 2005), MaSE (DeLoach, S. A., 2004), Prometheus (Padgham, L.2006), IĮS and DIAMOND. Generally, in multi-agent methodology concepts and models of only one origin, such as UML ("Mase", AAII, MESSAGE, PASSI), are used. In other methodologies many concepts are used, e.g. TROPOS (*"i" concept is obtained on the basis of engineering knowledge, A-UML concept is for interoperability protocols and plans) or DESIRE (graphic concept based on the modelling expertise, and specific hierarchical concept for description of the tasks). Therefore, in order to cover all stages of the life cycle, for different levels of abstraction several formalized languages are required.

The analysis of multi-agent systems development methodologies was carried out. The weakness of existing agent methodologies were identified: procedures are too abstract, there is no such agent methodology that would take into account the important features of the systems, such as the interface with the apparatus, its restrictions and time requirement specification. It was decided to choose one most appropriate methodology and to create specialized multi-agent system development methodology on its basis.

Many existing multi-agent techniques generally distinguish only the stages of analysis and development. Only few methodologies examine other steps. For example, in MASSIVE or "Vowels" we can find the arrangement stage. This arrangement stage is extremely important for embedded systems as it includes hardware/software distribution. The last and the most significant difference between DIAMOND and other multi-agent approaches is that DIAMOND, unlike other methods, combines the development of hardware and software parts. Developing the traditional system, the division is carried out at the beginning of the cycle.

In order to cover the entire life cycle, expressing the different things at various levels, the different formalities are required. For this reason, the life cycle is applied using four steps, where using more or less formal paradigms and languages, different expressions are mixed (agents, components, limited state machinery, hardware definition languages). The latest existing life cycle used in multi-agent methodologies is a classic cascade life cycle. Even if some works are trying to present such repetitive cycles as Kasiopeja (W) or GAJA, the suggestion of DIAMOND spiral lifecycle is very original.

While defining the needs of the stage, the study of functioning and suspension modes is included into general operation system. In the design phase, it is permitted to abstractly deal with software and hardware. DIAMOND methodology uses the components to create agents, because only few multi-agent methodologies provide the actual componential dimension. Because the analogy of flexible components and chips allow hardware tools and software tools

to be guided by a common vision, these components are used for facilitating the work of developers with a visual programming, for managing the complexity while making functional separations, increasing multiple with reusability, also, to facilitate the division.

DIAMOND allows analysing the areas in which co-operating subjects are combined with the physical devices that must control or supervise. DIAMOND uses the cases of UML use, because they contain reliable definition of requirements. The interpretations of the cases of use are slightly different from their common use, because the actors are not included in the system or its components. In addition, in the cases of physical interactions, there cannot be actors in the interaction schemes.

In the design stage, DIAMOND uses components as operational units. In these components, the limited state machinery or component set out to describe the internal operations are used. These formalities allow generating software code or hardware specifications.

2. THE CONCEPT OF MULTIAGENT SYSTEMS' DEVELOPMENT PLATFORMS

Worldwide there are many developed agent systems, but most of them cannot or only partially can be adapted to the development of small-scale embedded systems. It is, therefore, necessary to review them. The first overviewed agent system which was developed realizing FIPA (O'Brien, P. D., 1998) standards is FIPA-OS. Another agent platform is "APRIL Agent Platform (AAP)" which, unlike most of agent platforms, is realized in APRIL programming language (McCabe, 1995). AAP creation and development has been terminated, but the platform stands out with another meaningful difference from other systems: it makes it possible to easily create and deploy agents in the internet space and supports web services and semantic web standards. Another multi-agent platform is JASON (Bordini, 2005; Bordini, 2007) which is still under development. Its main advantage is easy development of BDI (Belief-Desire-Intention) type agents. The platform was developed in JAVA programming language and the behaviour of its agents is described in AgentSpeak programming language (Rao, A.S., 1996). JASON can work in two ways: the first, when all agents are running on a single computer, the second method allows agents to spread to different machines using SACI (Simple Agent Communication Infrastructure) that uses the KQML language (Finin T., 1997). In practice, the most commonly used agent platform for development of many agents systems is JADF (Java Agent Development Framework). JADF platform focuses on implementation of the FIPA

model, as well as providing communication infrastructure and additional tools such as management agents, help for their development and debug.

Comparative analysis of agent system modelling languages

Formal specification languages usually used in agent systems analysis and design are such as ConGolog and CASL, Z languages. Informal development methods of agent systems are generally based on a structured language and graphic notations, such as UML diagrams. There are a few graphical tools, specialized for modelling of agent systems, such as PROMETHEUS, JAL or AUML. AUML is UML extended with necessary notations for agent simulation and other specialized languages for modelling of agents.

Agent systems can be realized and implemented on multiple platforms both specialized standards-based and traditional platforms such as JAVA objects or components. Various programming languages can be used for realisation. Traditional object-oriented languages are considered suitable for multi-agent system development as the concept of agent is similar to the concept of the object, while the special agent programming languages such as FLUX, JACK or 3 APL are considered as new language class. For realization of the agent system, it is the most affective to use agent platforms that provide a set of reusable components and services necessary for realization and implementation of agents.

The main difference between the agents and components is the communication mechanism. Agents use the communication languages, and component which is the interaction protocols. In an agent paradigm, a message is sent to transferee part of the sender's mental state to the recipient. When another agent receives a message, it can assert that the eligibility conditions are suitable for the sender and the sender tries to achieve an appropriate rational effect. Using structured ACL, such as in JADE platform, the development of reactive agents that could be involved in a complex interactions is facilitated.

Both for agents as well as for components the delegation of responsibility depends on their communication; and different ways of delegation justify the differences of their communication models. In the component model, the sender himself is responsible for the effects of the notification. In other words, components do not delegate responsibility to other components. In an agent model, the recipient itself is responsible for the consequences of its actions and the sender has to identify why he requested the service. Very important communication act, which the agent can perform, is delegation of one of its goals to another agent. This is the main mechanism by which agents delegate responsibilities.

3. COMPARISION OF RTOS FOR SMALL-SCALE EMBEDDED SYSTEMS

Embedded systems generally are classified into three groups: small-scale, medium scale and sophisticated embedded systems (Kamal R., 2009).

Small-scale embedded systems are developed using 8 or 16-bit microcontroller, they usually have little additional hardware, the software is simple. When programming these types of systems, an assembly or “C” programming language is often used; the source code, written in this language, is compiled into executable code and is placed in the microcontroller's memory. The software must fit into the limited memory of the microcontroller, and to ensure the smooth execution of the entire system.

Medium-scale embedded systems are usually created using one or more 16 or 32-bit microcontrollers, and other additional equipment DSP, RISC. These systems may also include additional purpose single processors for the operation of various system functions (such as data transmission protocols, scanning and digitizing of various sensors). This type of systems have much more sophisticated hardware; therefore, C/C++/Java programming languages and real-time operating system (RTOS) are often used for software development.

Real-time operating system (RTOS) works as any other operating system (hereinafter OS). This software has an application programming interface (API) set, which can be used to create systems and solutions. But these operating systems are special because they help embedded systems to comply with the deadlines. The selection of RTOS for a particular case is performed during analysis of certain parameters that determine its operation. Certain systems can require the least interruption delay time, but it gives RTOS a higher probability of priority inversion.

Each RTOS has a kernel which acts like a core. The core is surrounded by cladding layers that provides protection and access permissions. Manufacturers of RTOS provide a set of Application Programming Interface (API), which is used to reach the nucleus to accomplish the task. RTOS memory is divided into kernel space that consists of kernel code, and a user space consisting of user code. Threads of real-time operating system are as functions having their temporary information storage and thread control block. The core consists of a planner which fulfils these threads in a consistent manner. There are three types of RTOS: hard real-time, firm real-time and soft real-time. In the hard real-time system, tasks and interruption inquiries must be carried out within the given deadlines. The failure to comply one term may result in failure of the whole system. Firm real-time system allows a number of exemptions from the deadlines, but missing more terms can cause a system break. Soft real-time system is

one in which the terms are most frequently met, but this limitation is not very rigorous. Each RTOS has auxiliary tools for multiprogramming mode (threads), they usually support synchronization of thread using semaphores or mutual exclusion locks. Each RTOS must have a sufficient number of priority levels and must avoid priority inversion.

According to the requirements for developing a small-scale embedded systems (processing speed up to 16 MIPS, flash memory size up to 2560 Kb, data memory size up to 8KB). Atmel microcontroller ATmega2560 (Fig. 8 a) was chosen for comparison with RTOS.

RISC controller architecture was used for the study; the controller computing speed of 16 MIPS at 16MHz clock speed, program memory type: flash, program memory 2560 Kb; data memory type: EEPROM, data memory 8 Kb, ROM type EEPROM, memory 4096 b. The controller has 6 timers, RTC, 32 external interruptions 86 input/output ports (Atmel, 2014). To connect the controller the Arduino MEGA controller board (Fig. 8 b) with a 16 MHz quartz resonator that provides a maximum working speed of the processor was used. The controller board is powered by a 5V voltage through the USB interface.

During the investigation, the Arduino software has been deleted in the controller and with the help of ISP connector each studied RTOS was uploaded into microcontroller. Before uploading, each RTOS was rebuilt specific for this type of 8-bit microcontroller and additional functions required to measure the parameters of RTOS were added.

Firstly, the operating system program memory space (flash), which consists of the operating system kernel, program loader, libraries, and drivers, were measured. The measurements were carried out compiling operating systems; also, from their loading file while separating the memory, the occupied space was measured. The study results are presented in chart 1.

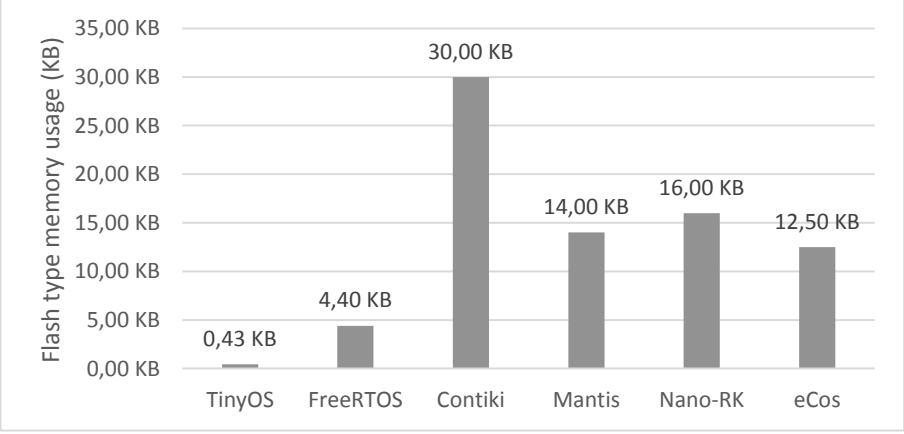


Fig 1. *RTOS flash type memory usage*

For its simple architecture the real-time operating system TinyOS used the least of programs memory - merely just 432 b, a little more space was used by FreeRTOS operating system - 4.4 Kb. Mainly due to additional libraries, Contaki operating system used 30 Kb, but none of the tested RTOS exceeded the storage space of microcontroller. During the examination of memory usage by RTOS system (SRAM memory type was used in microcontroller), RTOS was released and the amount of memory reserved was measured. The results are presented in chart 2.

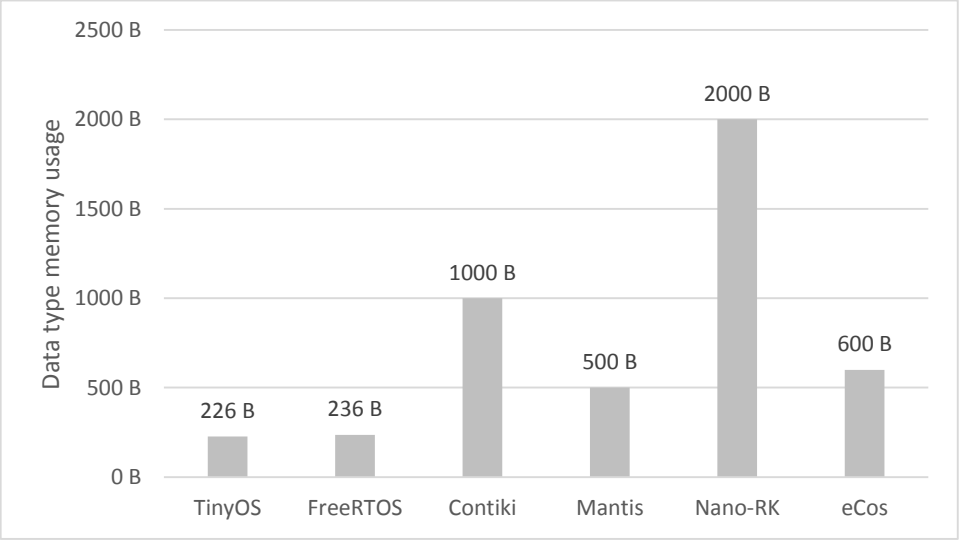


Fig 2. *RTOS data type memory usage*

The least memory usage had TinyOS and FreeRTOS operating system 236b and 226b, respectively, while the most space was used by Nano-RK operating system - 2Kb, and this is one quarter of the microcontroller data memory.

Processor's cycles number required for switching threads was measured after carrying out two processes in each of operating systems and the hardware timer Amtega2580 microcontroller with a processor's frequency multiplier equal to one. The first process starts the timer when finishing the work, and the second process stops the timer when it starts to work. The timer's counter value is equal to the number of CPU cycles when switching execution threads. The results are presented in chart 3.

The minimum number of CPU cycles to switch the threads was required by TinyOS and FreeRTOS operating systems, respectively 80 and 84 CPU cycles, significantly worse results were showed by Contiki, Mantis and Nano-RK operating systems - 320 to 400 processor beats. Since the CPU frequency was 16MHz, TinyOS and FreeRTOS took only about 5μs for switching the thread.

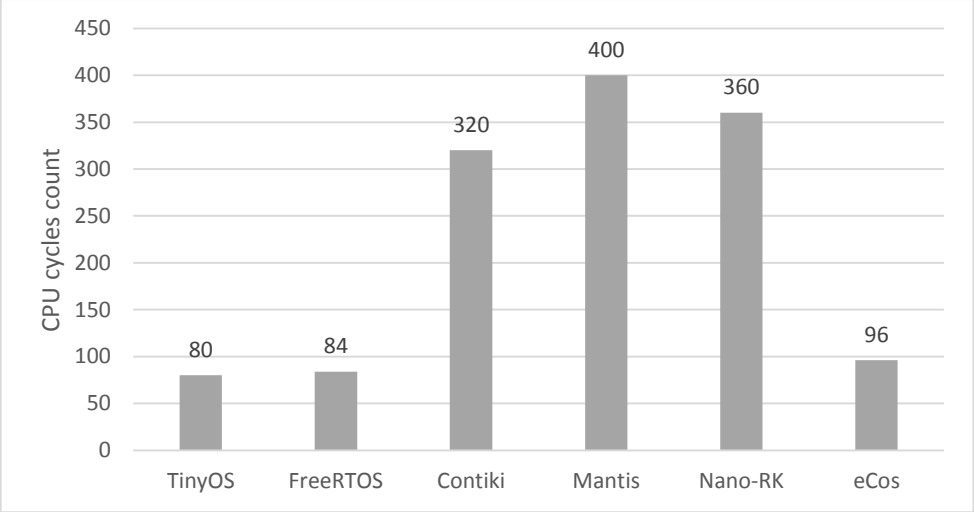


Fig 3. *Processor cycles count required for switching between threads*

For further comparative analysis of RTOS systems the following criteria were analysed: multitasking, supported network protocols, programming language support, distribution license, documentation, the latest version and release date.

The results presented in Table 1 show that all of the analysed real-time operating systems, except for TinyOS, are using C programming language for programming. For programming TinyOS uses the C language extension NesC *(network embedded systems C)* specifically created for it. Almost all examined RTOS (except for TinyOS and Mantis) had TCP/IP protocol support implemented in a real-time operating system. This support is necessary for the created agent systems so the constituent embedded systems could communicate with each other and with the external systems. TCP/IP support can be implemented in the user field as well, but this would require additional programs and data space. The most comprehensive documentation was prepared for FreeRTOS, Contiki and eCos, the rest of the real-time operating system had only provided the description of their functions, structures and nuclear. The most often updated (the official versions released) was FreeRTOS operating system. Although eCos and Nano-RK official versions were released long ago, but the source code is available online and can be adjusted for each registered user so their informal versions are given often enough.

Table 1. RTOS comparison

RTOS	Multitasking	Supported network protocols	Programming language	License	Documentation	Latest version
TinyOS	Based on events	-	NesC	Free, Opensource	Functions and API description	2012.08.08 2.1.2.
FreeRTOS	Round Robin, preemp-tive	TCP/IP, Ipv6	C	Free, Opensource	Detailed documentation	2015.03.24 8.2.1
Contiki	Based on events	TCP/IP, Ipv6	C	Free, Opensource	Detailed documentation	2013.11.15 2.7
Mantis	Priority based	-	C	Free, Opensource	Functions and API description	2012.07.03 1.0b
Nano-RK	Priority based	TCP/IP, Ipv6	C	Pay-per-use for commercial purposes, Opensource	Functions and API description	2009.07.15 Rev 877
eCos	FIFO, Round Robin, BitMAp	TCP/IP, IPV6, FTP, SNMP, DNS, DHCP, HTTP, SNTP	C	Free, Opensource	Detailed documentation	2009.03.30 3.0

After summarising the results of the comparative analysis, we can state that the most appropriate real-time operating system to create MAS for small-scale embedded systems is FreeRTOS. FreeRTOS, like the TinyOS, needs low software and data storage space, but has implemented TCP/IP protocols; their implementation in the field of application software would require additional resources in TinyOS system; furthermore, FreeRTOS have a much more detailed documentation and is more frequently updated.

4. METHODOLOGIE FOR MAS DEVELOPMENT IN SMALL-SCALE EMBEDDED SYSTEM

MAS can be divided into several sub-systems according to management devices and functions. There are three possible separation methods of a developed system:

- Each embedded system works as a single agent and each agent is responsible for management and control of one embedded system, all incoming and outgoing data is merged into one set of rules. This can create a large database of rules, but it is clear that some of the control data may be irrelevant. This method is a decentralized in respect of control.
- The entire system is classified in several functional modules. For example, such as temperature control, lighting, etc. Each function of the system is controlled by the individual uniform level agent.
- The entire system can be divided into certain hierarchical levels. Each embedded system is controlled by individual agents and they, according to functional modules, are managed by the agents responsible for certain function. The entire system is controlled by the management agent of the highest hierarchical level.

Agents' organization modelled in hierarchical levels is presented in Figure 4. In interoperability level agents interact with physical sensors and controllers as the final ZigBee network devices.

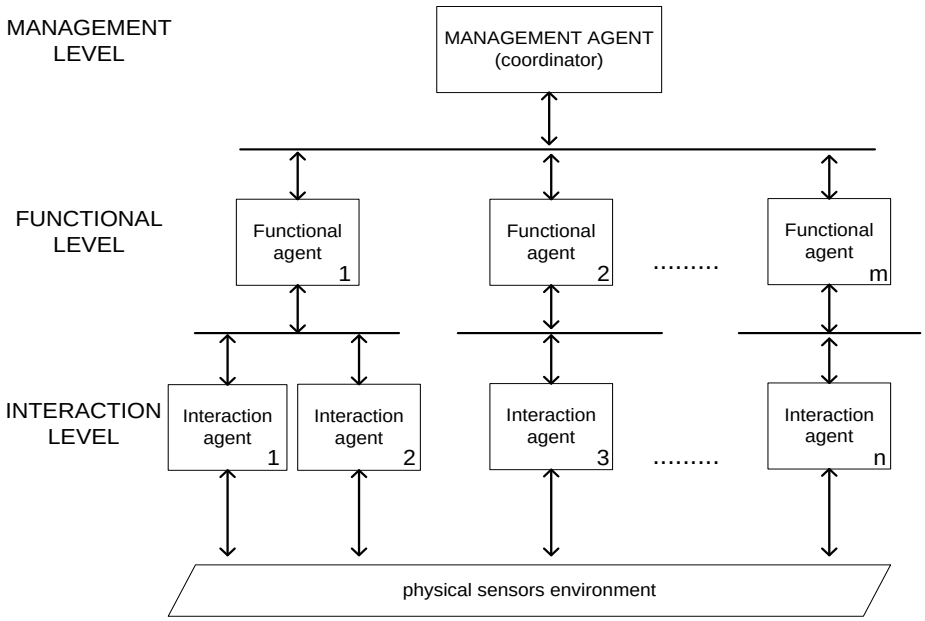


Fig. 4. *MAS structural levels*

Within the functional level, each agent performs a specific function. Each functional module is an agent that manages the corresponding lowest level agents. In addition, the agent can determine the behaviour and priorities, to improve the system's intelligence and to understand the environmental changes or concept of user control. Physically, in the technical system, functional agent interacts with other agents as ZigBee network router.

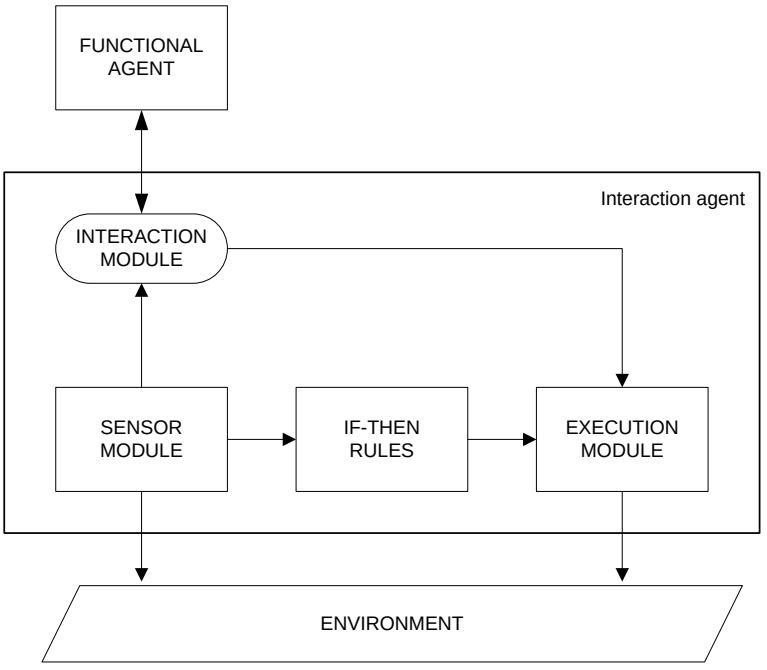
The highest is management level. Management agent manages the entire system. Management agent corresponds the ZigBee network coordinator.

Interaction agents, including internal environmental parameters agents and external environmental parameters agents, transmit the collected information to the functional agents via wireless sensor network and receive control signals from functional agents for the environmental control. Its internal model consists of the sensor module, execution module, communication module and If-Then pattern. Terms-action rules associate the sensor with operation model.

Not all of the decisions necessary for operation of the system are accepted by the functional agents, some of them can be directly accepted by the interaction agents at a lower level in order to reduce the complexity and increase system stability. Purely reactive decisions are carried out by the interaction agents directly through If-Then rules. If-then rules are between switches and the respective connections. This method allows the system to operate only in the main level, even if the functional agent does not work. The graph indicates that the control module entrance consists of interaction agent solution and control information from a

functional agent. Interaction agent solution is superior to functional agents, so the execution agent chooses interaction agents' decision if they clash. As shown in the interaction agent operation scheme (Fig. 5), the corresponding act of the rule is carried out, if the rule is applied successfully. This means that the interaction agent has tasks of actions. Otherwise, if there is a communication event from the functional agent, then it is executed.

Functional agents are responsible for a specific function that system operates. Usually internal structure of functional agents is the same but their functions are different. Registers server module is responsible for network communication and records it in the network connection register. When it is created, it determines the network routing lists in wireless network. Agents, associated with the function managed by functional agents, such as interface or control agent, must plan routes together with it. To increase the flexibility of the system, some agents can be added and others eliminated during the network development process. Information from the interaction of agents usually consists of environmental parameters, the external environmental parameters, device status and so on.



Source: Hagras H. 2004, Ostaševičiūtė L. 2009

Fig. 5. *Structural components of interaction agent*

Due to continuous changes in the environment and user priorities, a functional agent is an evolutionary function and often uses certain artificial intelligence techniques (such as fuzzy logic, etc.). When there are changes in the environment in order to improve the decision-making function, both the base of rules and the base knowledge are changed.

Applying the base of knowledge and skills, control agent manages the entire system at the highest level. Agent confirms the common tasks and distributes tasks to different functional agents. He is responsible for the cooperation of the tasks and conflict elimination. The agent has an automatic control function, which keeps systems functioning and examines system's events.

In addition, it has an independent planning possibility. The internal structure can be divided into three layers. The highest layer is designed to hold all system's information and based on ontology base, knowledge base and database, to transfer the corresponding knowledge to other agents. The database contains the environmental information data, device data and user information data. The knowledge base consists of global knowledge, task knowledge, cooperation knowledge, conflict elimination knowledge, etc.

After analysing application field of agent technologies and their characteristics, it was found that the agent paradigm is suitable for designed system development, because it ensures a high degree of abstraction, simple scalable distribution of the system, integrity, efficient communication mechanism and other requirements for this class of systems and their development process.

5. EXPERIMENTAL RESULTS OF MAS APPLICATION FOR SMALL-SCALE EMBEDDED SYSTEMS

Usually, a hydrometeorological information system is faced with great data flows, but the data levels are often excessive, depending on the observed region of the water. The paper presents advanced buoy communication technologies based on multi-agent interaction and data exchange between several monitoring system nodes. The proposed management of buoy communication is based on a clustering algorithm, which enables the performance of the hydrometeorological information system to be enhanced. The experiment is based on the design and analysis of the inexpensive but reliable the Baltic Sea autonomous monitoring network (buoys), which would be able to continuously monitor and collect temperature, waviness, and other required data. The proposed approach of multi-agent based buoy communication enables all the data from the costal-based station to be monitored with limited

transition speed by setting different tasks for the agent-based buoy system according to the clustering information.

There is a variety of tools to monitor and evaluate Baltic Sea hydrometeorological data, but most received information has low spatial coverage and low level of detail in time (Barrera 2006). Sea wave height, water temperature, and underwater noise data, used for many practical applications, are usually obtained from three sources: buoy measurements, model calculations, and ship observations. Compared to other data acquisition methods, buoy measurements are the most reliable and readily data source available continuously for years (Bender 2011). Basically, the network of buoys is involved in mapping the temperature, wave height, and underwater noise at a buoy location using the data retrieved from other buoy locations. However, many hydrometeorological data measurements using sea buoys can be lost due to malfunctions, maintenance, connection problems, or dubious data recorded by the buoy. In order to ensure greater reliability of data collection, it is necessary to develop a distributed information system, predicting complex situations and supporting decision-making processes. Information provided from such system is important for decision-makers and is needed to ensure the provision of information for decision-making institutions (Collins 2012; Dzemydienė 2013). An important feature of the buoy network is the ability to monitor, collect, and evaluate wide spatial coverage and real-time hydrometeorological data of the Baltic Sea (Bykov 2013). A hydrometeorological information system is faced with great data flows, but the data levels are often excessive, depending on the observed region of the water. Therefore, current traditional methods are no longer sufficient to ensure the rapid collection of data and valuable information extraction.

The purpose of this study is to show the possibilities of developing a hydrometeorological data collection system (HMDCS) involving advanced technologies such as a multi-agent based interaction and data collection between several monitoring system nodes (i.e., buoys). The experiment is based on the design of the inexpensive but reliable Baltic Sea autonomous monitoring network (buoys), which would be able to continuously monitor and collect temperature, waviness, and other required data. Moreover, it has the ability to monitor all the data from the coast-based station with limited transition speed by setting different tasks for the agent-based buoy system.

Sea hydrometeorological data monitoring

Nowadays, there are numerous and varied designs for autonomous systems used for meteorological and oceanographic monitoring with different integration degrees. The buoy network system used in the Canary Islands is one of them (Barrera 2006). It has a control center that manages the transmission communications and provides data in a useful form to diverse socioeconomically important sectors which make exhaustive use of the littoral, and data from the buoys are used to manage the coastal environment. The buoys monitor water temperature, salinity, dissolved oxygen, hydrocarbons, and other characteristics, which they can measure when equipped with other sensors such as a fluorometer and a turbidimeter, and each buoy is also able to communicate via GSM modem. Following a programmed sampling rate (every hour), the ECU sends to the central receiver unit an SMS message, which includes a sensor data set, GPS position, and battery level. However, deeper analysis of the data has shown that such a sampling rate is not sufficient, which means that the data transmit protocol must be reevaluated.

In order to provide greater hydrometeorological data monitoring reliability and faster data retrieval, a variety of sensory system networks (Larhuis 2010; Li 2008) have been proposed. Such communication technologies enable communication between sensor nodes (Mirza 2011), systems for communication between maritime platforms like vessels, commercial ships, or buoys, and real-time monitoring of the underwater environment where an acoustic mesh network is located between the underwater sensor networks and the central monitoring system. The proposed models can solve various problems but require more flexible solutions for complex data transfer problems. This problem can be solved by developing an active autonomous sensor multi-agent based system, which is able to combine data processing methods according to the situation.

Hydrometeorological data sensory system

Temperature Data Collection. During the investigation stage of the HMDCS development, several types of temperature sensors were compared. The comparison possibilities are made by analyzing their parameters according to the technical specifications presented in datasheets.

After a comparative analysis of the temperature sensors, we selected the DS18B20 digital sensor. This digital temperature sensor can measure temperatures within the range from −55°C to +125°C at 12-bit precision, with accuracy ±0.50°C (Maxim Integrated, 2008). However, after additional calculations, it is possible to reduce the temperature measurement error down to 0.10° C. The most attractive feature is the fact that these sensors have already

been calibrated at the factory and their accuracy error is ±0.5° C in the range from−10°C to +85°C and ±2°C error over the operating range (55° C to +125°C). Sensor supply voltage is in the range of +3 to +5.5 V. In standby mode, current consumption is close to zero (less than 1μA), while temperature conversion power use is about 1 mA. The measurement process lasts no more than 0.75 sec. The DS18B20 communicates over a 1-Wire bus that by definition requires only one data line (and ground) for communication with a central microprocessor. In addition, the DS18B20 can derive power directly from the data line ("parasite power"), eliminating the need for an external power supply. Each DS18B20 has a unique 64-bit serial code, which allows multiple DS18B20 to function on the same 1-Wire bus. Thus, it is simple to use one microprocessor to control many DS18B20 distributed over an area of few square meters (in our case they are used to measure temperature in different depth of the sea).

At present, sea and ocean waviness measurements use a variety of methods, depending on the geographic region, measuring accuracy, and general tasks (Collins 2012). The main and most commonly used are as follows:

- ultrasound-based sensors:
  - pros: suitable for measuring waves with a height of over 5 meters,
  - cons: significant measurement errors,
- rheostat-type structures:
  - pros: allow you to get fairly accurate data,
  - cons: because of their design features they have a short lifetime,
- satellite image analysis:
  - cons: due to the inherent large errors, this can be used only for ocean waviness measurements,
- GPS system:
  - cons: not suitable for measuring waves with a height of 0.5–2.0 meters,
- accelerometer and gyroscope design:
  - pros: small measurement errors, easy implementation.

For our experiment a couple of accelerometer and gyroscope was used. Based on the experience of other scientists (Bender et al. 2010), accelerometer data were processed by removing the component of gravity, according to the following:

$$\begin{bmatrix} X_E \\ Y_E \\ Z_E \end{bmatrix} = \begin{bmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ a_3 & b_3 & c_3 \end{bmatrix} \begin{bmatrix} X_s \\ Y_s \\ Z_s \end{bmatrix} \quad (1)$$

Here, X_S, Y_S, Z_S represent the accelerations measured in the sensor frame, X_E, Y_E, Z_E are the accelerations rotated into the earth coordinate frame, and the direction cosines for the above transformation are in terms of the Euler attitude angles.

The coefficients a, b, and c are calculated using the following formulas:

$$a_1 = cos\theta cos\psi \tag{2}$$

$$b_1 = sin\varphi sin\theta cos\psi - cos\varphi sin\psi \tag{3}$$

$$c_1 = sin\varphi sin\theta cos\psi + cos\varphi sin\psi \tag{4}$$

$$a_2 = cos\theta sin\psi \tag{5}$$

$$b_2 = sin\varphi sin\theta cos\psi + cos\varphi sin\psi \tag{6}$$

$$c_2 = sin\varphi sin\theta cos\psi - cos\varphi sin\psi \tag{7}$$

$$a_3 = -sin\theta \tag{8}$$

$$b_3 = sin\varphi cos\theta \tag{9}$$

$$c_3 = cos\varphi cos\theta \tag{10}$$

Here, θ, ψ and φ are data from the gyroscope. After the accelerations have been rotated into the earth frame, the earth-referenced accelerations of the buoy are given by

$$A_x = -gX_E \tag{11}$$

$$\mathrm{A_y} = -gY_E \tag{12}$$

$$A_Z = g(1 - Z_E) \tag{13}$$

WhereA_X, A_Y, andA_Z are accelerations with eliminated gravity force along the earth-oriented x, y, and z axes.

After comparing the most popular data transmission protocols such as Bluetooth, UWB, ZigBee, Wi-Fi, and others, it was decided that ZigBee is the most suitable transmission protocol for such a task (lowcost, low power, mesh network support). Thus, this mesh-type network protocol was used for developing the HMDCS buoy network. ZigBee is an open standard for short-range wireless networks based on the Physical Layer and the Media Access Control from IEEE 802.15.4, focusing on minimizing the overall power consumption and at the same time maximizing network reliability (Sieber 2008).

The ZigBee protocol offers three kinds of devices to form a PAN (personal area network):

- end-devices, which periodically collect data and transmit it,
- routers: they collect data from end-devices and forward it to the destination (like another router or to the final coordinator),

– coordinator: one of the routers in a PAN is usually configured as a coordinator. The main function of the coordinator is the parameterization and management of the PAN, and the collection of network data.

In our case, we used so-called “full function devices” which collect data and work as a router and coordinator, which manages the PAN network and sends collected data via GSM to a coastal station (Figure 6). The following ZigBee network configuration was used for transmitting data to the coastal station.

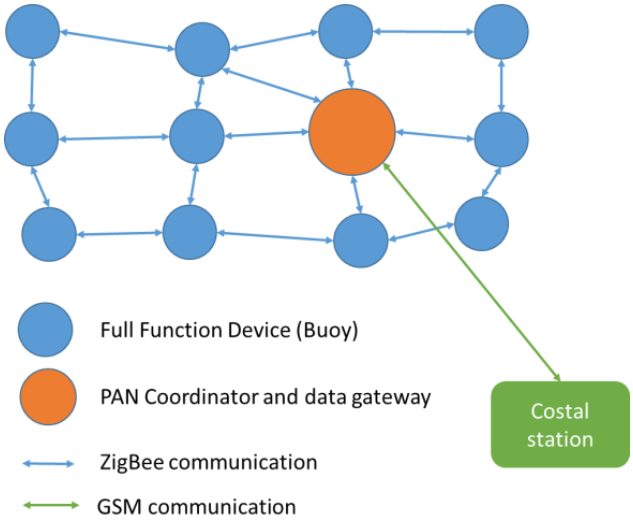


Fig 6. *HDRS Mesh network*

Power requirements for the electronic buoy system are 5VDC at 76mA in active cycle (active sensors and microcontroller are calculating data; transmitter is sending information) and about 27 mA at passive cycle (microcontroller is in sleep mode, and only the receiver is powered up for wake up using external interruption).

Active cycle lengths is about 5 seconds in every 10 minutes, so the duty cycle of the buoy is 0,0083. The buoy power supply consists of a battery bank of 18 AA type Ni-MH battery cells, arranged in 3 parallel groups of 6 cells connected in series. The capacity of each battery used in our buoy system is 2200 mAh, therefore, the capacity of each group of 3 batteries combined in parallel groups is 6600 mAh at 7.2 V. Expected lifetime of such configuration system at 25°C temperature would be about 10 days. Batteries placed in the bottom of the buoy also serve as ballast.

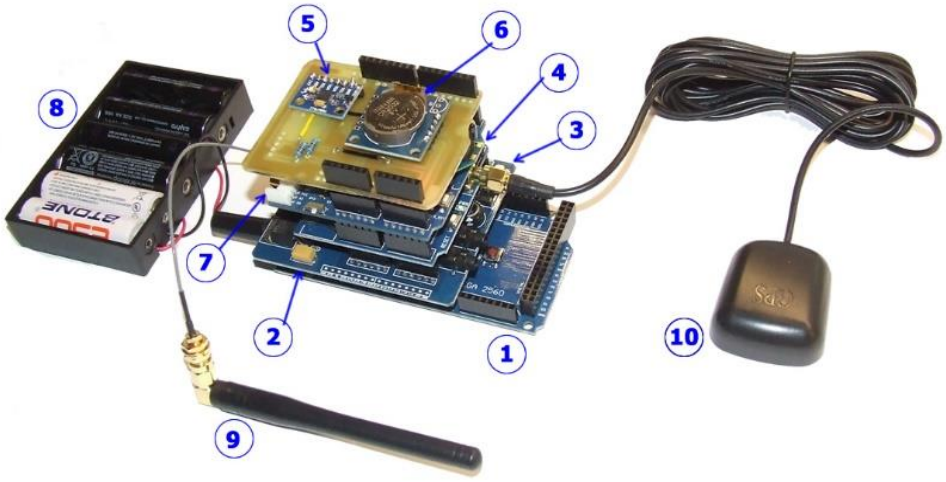


Fig. 7. *Buoy electronic system prototype*

For testing purposes the experimental buoy sensory system was developed. The core component of the prototype is Arduino Mega platform with ATmega2560 microcontroller which operates at 16 MHz clock frequency, (Fig. 7 - 1). The experimental buoy system is powered by solar power supply, which also recharges Ni-Mh batteries, which allow buoy sensory system to operate at night (Fig. 7 - 8). XBee Pro modules (Fig. 7- 4) implement communication via ZigBee protocol, which have 10mW transmission power and, according to the specifications, expected distance is about 1-1.5 km outdoors. Temperature measurements (underwater and weather) are implemented using DS18B20 sensors array connected in to the 1-wire network (Fig. 7 - 7). Data logging to MMC (Fig. 7 – 3). The wave height is measured using MPU6050 (Fig. 7 – 5) accelerometer/gyroscope and calculated by a provided method.

Multi-agent system model for hydrometeorological sensory system

The analysis overview diagram is designed to show the interactions between the system and the environment. At this abstract level, it is necessary to identify the actors, scenarios, percepts and actions involved in the system. This consists of two steps: identifying the actors and the scenarios they participate in with the system and identifying and defining the actions and percepts between the actors and the system. HDRS analysis overview diagram presented in figure 8.

The actors are all the people or external systems associated with the system. In our case, there will be two actors:

"Buoy" – defines buoy hardware (sensors), which monitors data and transmits them to HDRS multi-agent system;

"Station" – monitoring station, which receives data and if necessary, remotely controls buoys.

In the next step, we describe scenarios. The scenarios are the processes which the system uses to handle the percepts and produce the actions. In our system (HDRS), we can identify three high level scenarios:

"Obtain data scenario" – scenario during which multi-agent system receives the measured parameters from buoy hardware;

"Subscribe user scenario" – scenario during which a connection between hydrometeorological monitoring station and the user (or other system) to obtain monitoring data is created;

"Receive buoy data scenario" – scenario during which a hydrometeorological monitoring buoy is measured and processed data are transmitted to monitoring stations.

The percepts are all types of information which come into the system from the environment. In HDRS, we can identify three percepts:

"RS dataE" – data obtained from sensors;

"RFrequency dataE" – new sample rate information;

"Subscription from user" – user (or system) information.

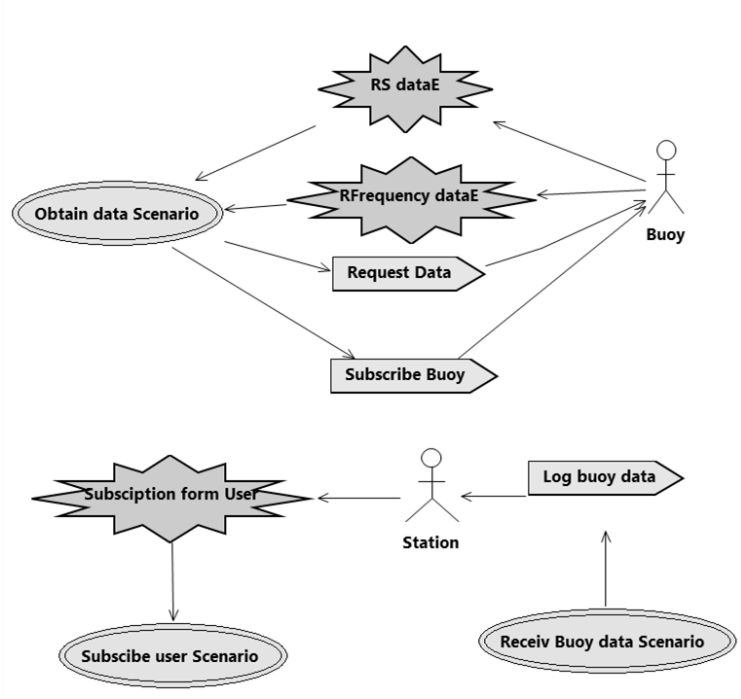


Fig. 8 *HDRS analysis overview diagram*

The actions describe everything that is sent from the system to the environment. In our system, we can identify three actions:

"Request Data" – MAS, which manages the buoy, may require hardware to measure environment parameters;

"Subscribe Buoy" – during this action MAS, which manages the buoy, subscribes to the monitoring station;

"Log buoy data" – measured and processed environmental parameters transmission to the monitoring stations.

Goals diagram shows the highest levels goals of the whole system and the individual components. From these high-level goals, several sub goals can be defined. For each set of sub-goals either an AND or OR constraint can be attached. AND constraints indicate that all of the sub-goals must be achieved in order to achieve the parent goal. An OR constraint indicates that only one of the sub-goals needs to be achieved in order for the parent goal to be achieved.

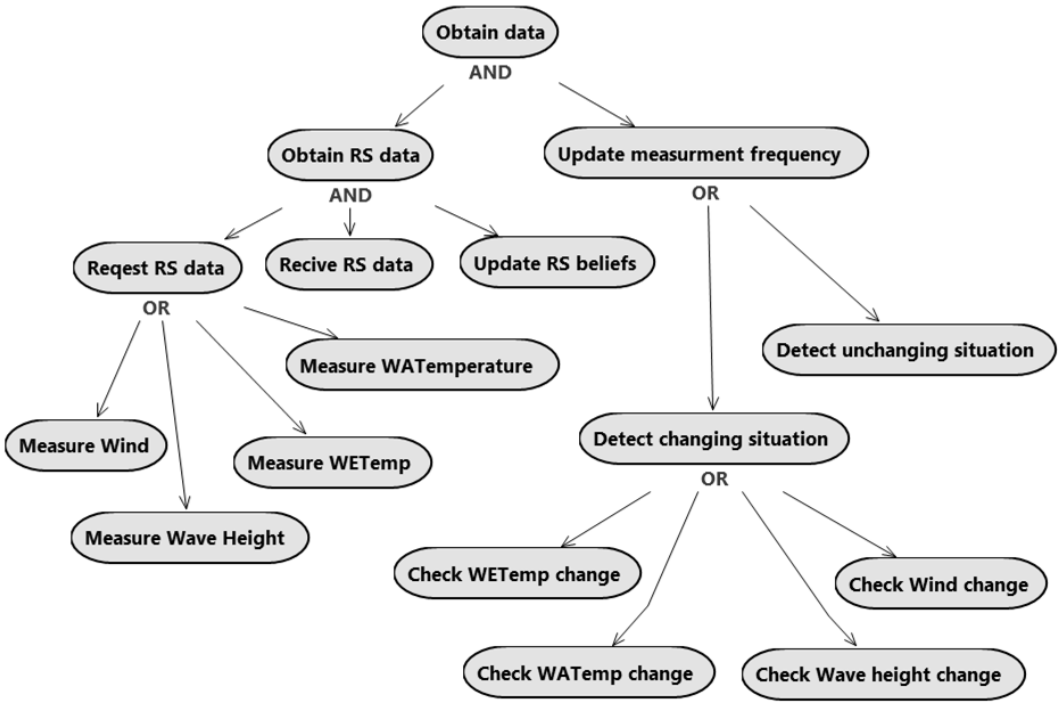


Fig. 9 *HDRS buoy goals diagram*

"Obtain data" goal is the highest level which is divided into two sub-goals:

- "Obtain RS data" – measure actual data from buoys sensors;
- "Update measurement frequency" – by environmental parameters (wave height, temperature, wind direction and speed) changing speed of the

environment are calculated, sample rate and data reading and frequency of transfers to the station are updated.

Goal “Obtain RS data” is divided into three sub-goals:

- “Request RS data” – the requirement to read data from the buoy sensors. This goal has four sub-goals: Wind, Wave height, Water temperature, Weather temperature. As it is only necessary for one of the sub-goals to succeed in order for the parent goal to succeed, an OR constraint can be used.
- “Receive RS data” – processing of received data and conversion to the appropriate format.

“Update RS beliefs” – place processed data into the data warehouse.

Goal “Update measurement frequency” is divided into two sub-goals: “Detect Unchanging situation”, “Detect changing situation”

The next phase of system design is to group similar goals to separate the roles. HDRS grouping is done using the System Roles diagram. Each percept and action should be part of some role. They can also be part of multiple roles.

In the HDRS, roles are distributed in two parts: the roles associated with the management of a buoy (fig. 10) and roles associated with meteorological station (fig. 11).

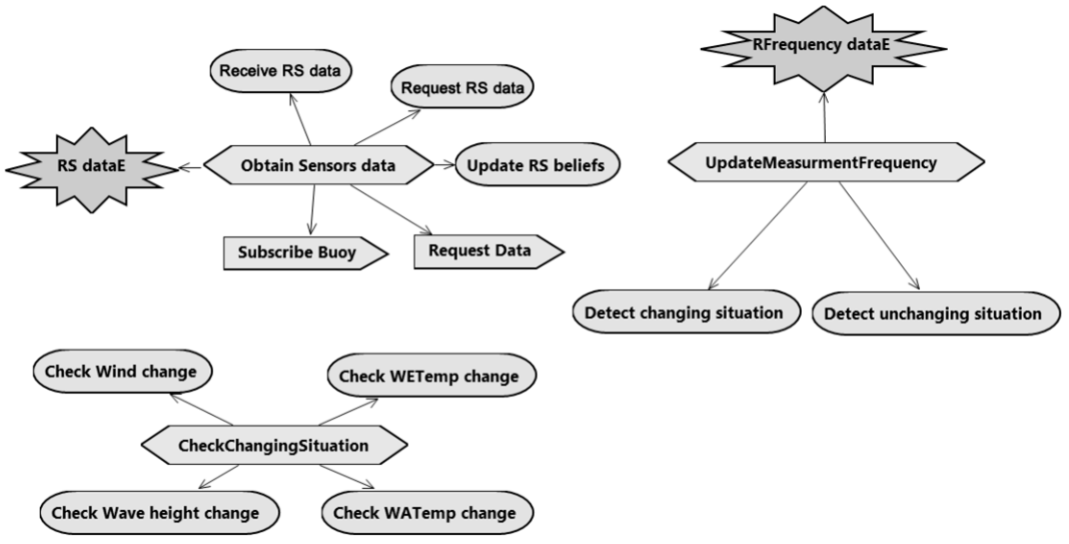


Fig. 10 *HDRS buoy roles diagram*

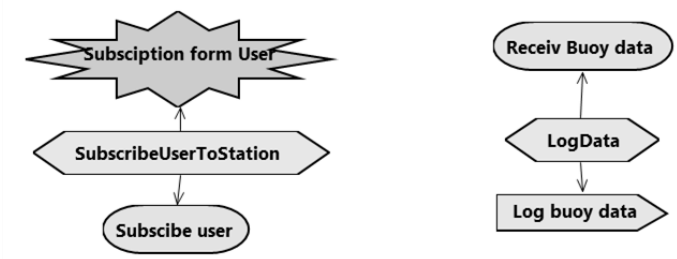


Fig. 11 *HDRS station roles diagram*

At this point in the design process, it is necessary to return to the key scenarios we identified earlier and add in the information we have on roles. This will also help us to determine the data stores that will be required by the system.

HDRS architectural design

There are three key actions that must be performed in a system specification phase: roles are grouped into certain groups, which are assigned to the agents, protocols for agent interactions (using AUML notation) are described, and data stores in data description are specified.

At agents' roles assignment stage, it is necessary to identify the different types of agents and assign them pre-specified roles. Each described agent type may have one or several agent realizations in the system. In the agent role diagram, there are descriptions of each agent type, but not a specific agent. Describing agent types, we must take into account these main principles: if possible, roles using the same data storage should be assigned to the same type of agents; it is better to create agents that combine several roles than for each role create a different type agent.

With these considerations in mind, we can decided upon the following agent types in HDRS (fig. 12): "BuoyAgent", "StationAgent."

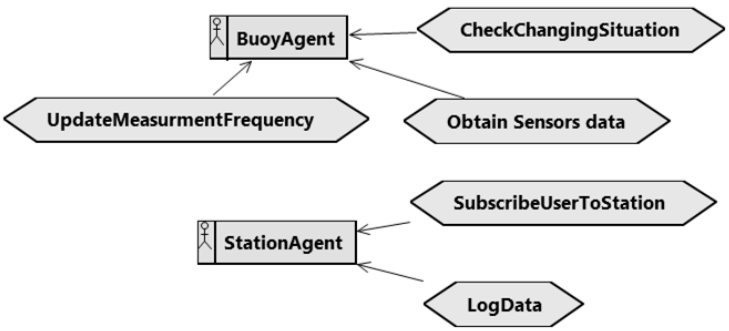


Fig. 12 *HDRS Agent-role grouping diagram*

Communication between agents and the environment (other systems or hardware) are described in the protocols and are specified using AUML diagrams. Data can be transferred in separate messages or described protocols (protocols group individual messages into certain sequences of communication). In the HDRS, there will be used two data transfer protocols: the measured data to and from the buoy hardware and the transfer of data to closely located other agents; transfer of measured data from buoys to monitoring station.

The first protocol includes two system entities: the actor “Buoy” (Buoy hardware) and agent “BuoyAgent” (agent controlling buoy) (fig. 13). Communication protocol is split into several parts, firstly, agent “BuoyAgent” completes an action "Subscribe buoy“, which attaches agent’s software to the buoy hardware (at this step, data transmission protocols such as I2C, SPI, USART are initialized which are used for microcontroller and sensors communication). Next part of the protocol is implemented in cycles, firstly, Agent “BuoyAgent” can require hardware to measure new environment data (this step is contained in a separate container “opt” and may optionally be called for each iteration through the loop, agent itself decides how often environment parameters are measured). In the next step, the agent receives the data from the hardware (if they are sent). In the last step of the cycle, the agent can send the measured environment data to other nearby buoys if environment parameters are changing a lot.

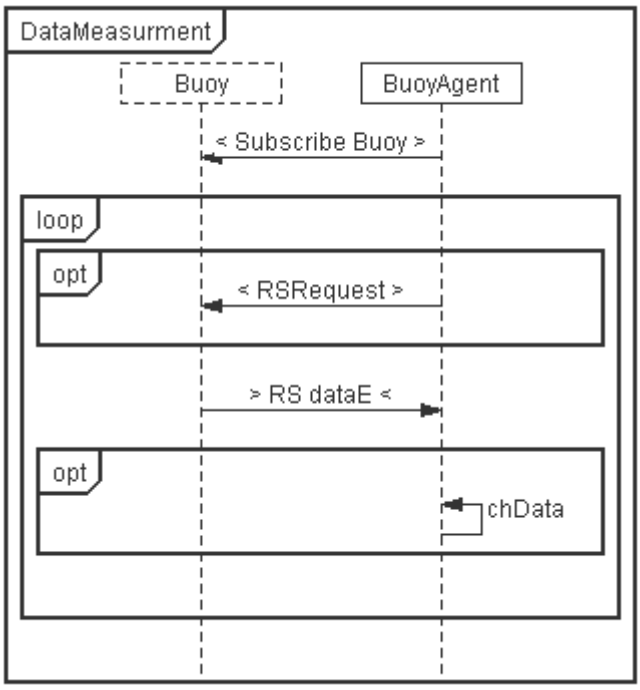


Fig. 13 *Measured data AUML Chart*

Another data transmission protocol is designed for measured data transmission to the monitoring station agent (fig. 14).

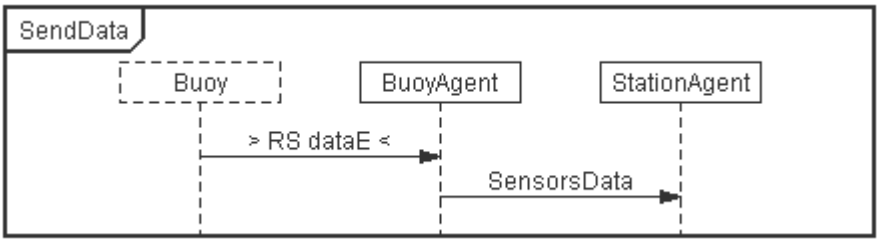


Fig. 14 *Measured data transmission from buoy to the station AUML diagram*

Experimental results

During experimental study, two types of buoys were compared: regular buoy which reads and transfers data to the central station in fixed time intervals and multi-agent systems using buoy, which reads and transmits data at different frequencies (depending on how fast the environment is changing).

Both types of buoys were set to collect the same type of measurements: temperature (water and air), barometric air pressure, wave height, wind speed and direction. Information was transmitted by data packet described in table 2.

Table 2. Collected data packet structure

Transmitted information	**Size**
Buoy identification code	32 bits
Measured parameter identification code	8 bits
GPS longitude	32 bits
GPS latitude	32 bits
Water temperature	16 bits
Air temperature	16 bits
Barometric air pressure	16 bits
Wave height	16 bits
Wind speed	8 bits
Wind direction	8 bits

Each data packet consists of buoy identification code, geographical location, measured parameter identification code and value of measured parameter. It's about 104 bits of information (without DigiMesh headers) in average. Regular buoys read information and transmit all measured data values to the station in average every two-three minutes. Experimental investigations were carried out under laboratory conditions (to ensure results accuracy all sensors' measurements were identical for both buoys).

During the first experiment, 24 hours, all environment parameters were constant. A number of transmitted packets from each buoy was measured. Figure 15 presents the results of the first experiment: on the left axis of the graph are indicated both buoys' transmitted data packets per hour, on the right axis, temperature from temperature sensor (constant 8°C temperature). All other buoys sensors were also fed by constant values. The regular buoy in one hour transmitted average of 119 data packet while the buoy with MAS within the first hour sent 102 data packets, and subsequent of only 12 packets per hour. During the first hour HDRS multi-agent system detected that environment parameters remain unchanged so the system automatically began to lengthen intervals between readings. When the measured values were constant, HDRS transmitted all data set (5 measured parameters) to the central station on average every 25 minutes.

This experimental study shows that the using of proposed MAS for buoys, at unchanging environmental conditions, significantly reduces the amount of transmitted duplicate data. HDRS buoy, during 24-hour interval, amount of transmitted data was 87% lower than regular buoys using fixed time for readings.

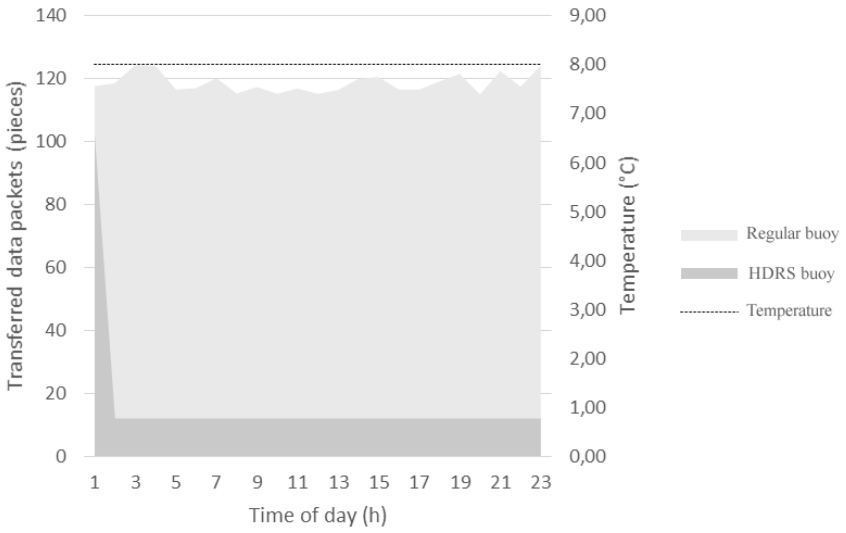


Fig. 15 *Transmissions of data packets under sustainable climatic conditions*

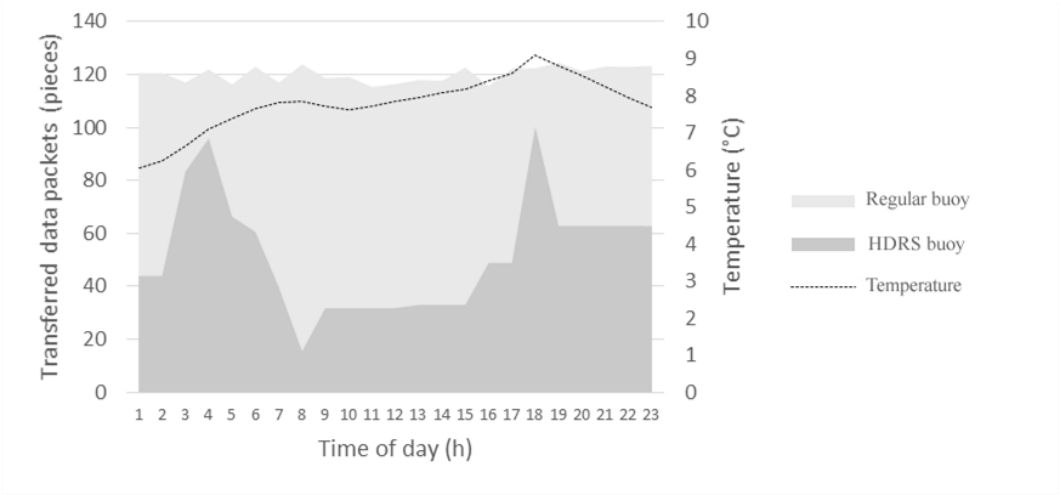


Fig. 16 *Transmissions of data packets of changing climatic conditions*

During the second experiment, real Baltic Sea hydrometeorological measurements were sent to the both buoys (fig 16). The results show that HDRS frequency of measurements and data transmission depends on how fast environment parameters are changing. During 24-hour interval, HDRS amount of transmitted data was 63% lower than regular buoys using fixed time intervals for readings.

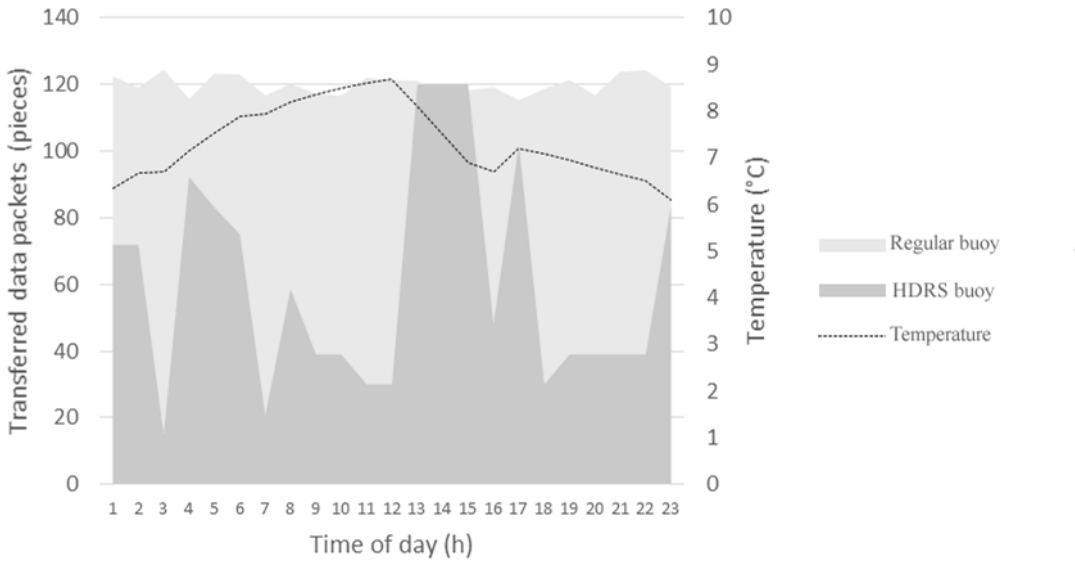


Fig. 17 *Transmissions of data packets of rapidly changing climatic conditions*

During the third experiment, very fast changing hydrometeorological measurements were sent to the both buoys. The obtained results showed that the HDRS quickly adapts to changing hydrometeorological parameters and increases the frequency of measurements (fig. 17). At 13-16 hours, sudden change of weather was recorded, so in this period HDRS started

to measure parameters very often (121-122 per hour). This allows to record detailed environment parameters during rapid marine processes. However, during the whole 24-hour interval HDRS transmitted 46% less data packages than regular fixed time buoy.

GENERAL CONCLUSIONS

1. After examining the methods for development of general-purpose multi-agent systems, it was established that they are too abstract and fail to address the characteristics of small-scale embedded systems and their limitations. Also, analysis of the specialized versions of these techniques for development of embedded multi-agent systems showed that they cannot be directly applied for the following reasons:
   - IIS methodology is based on JADE framework, which requires additional resources consuming JVM;
   - Neither IIS, nor DIAMOND methodology offers the platform and code generation tools for small-scale embedded systems;
   - Distribution stage (software/hardware separation) is performed at the beginning of the development cycle, so there is no possibility to integrate hardware and software during the whole multi-agent system development process.
2. During the examination of MAS realization platforms, it was found that most of them are designed for computer systems with large capacity and data storage and are operating under stationary conditions. It was found that the requirements raised by the author are partly satisfied by PDT "PROMETHEUS Design tool" platform, which allows to generate a primary software code for embedded systems (up to class diagrams), and can be integrated with DIAMOND development methodology.
3. After the experimental studies of real-time operating system capabilities and suitability for small-scale embedded systems it can be stated that the most appropriate is FreeRTOS operating system for development of MAS.
4. After modifications of the selected development methodology (DIAMOND) the infrastructure and its development methodology based on multi agents collaboration for development of small-scale embedded systems was proposed. Collocation stage, covering hardware and software partition was moved to the

end of the development cycle, thereby enabling the connection of hardware and software development during the whole multi-agent system development cycle. The transformation model for mapping the projected MAS agents into the RTOS controlled system was created. Also, the methods for realization of different types of agents in small-scale embedded systems are presented.

5. The proposed methodology for designing MAS has been experimentally tested through the development and testing of hydrometeorological data collection and monitoring system. The system of this type is applied in solving ecological monitoring tasks in the real environmental and climatic conditions of the Baltic Sea. Obtained experimental results demonstrate that when observed climate and environmental parameters are constant (or slowly varying), the number of transmitted samples is reduced by 87%. The system displays the appropriate adaptability to rapidly changing environmental conditions and allows collecting more detailed measurements than conventional fixed-time systems.
6. During practical realization the data transfer protocols were evaluated and it was found that DigiMesh wireless networking protocol is the best for integration of small-scale embedded systems into joint wireless network system.

LIST OF PUBLICATIONS BY THE AUTHOR ON THE SUBJECT OF DISSERTATION

The articles published in the peer-reviewed journals

1. Gricius G., Drungilas D., Andziulis A., Dzemydiene D., Voznak M., Kurmis M., Jakovlev S. 2014, Advanced Approach of Multi-Agent Based Buoy Communication, Computer Intelligence in Modeling, Prediction, and Analysis of Complex Dynamical Systems (COMD), 2014
2. Bielskis A.A., Gricius G., Drungilas D., Dzemydienė D., Guseinovienė E., Ambient Lighting Controller Based on Reinforcement Learning Components of Multi-Agents, Electronics and Electrical Engineering, ISSN 1392 – 1215, 5 (121), p. 79 – 84, Kaunas, Lithuania, 2012
3. Gricius, G., Bielskis, A.A., Denisovas, V., Drungilas, D., Dzemydienė, D. Multi-Agent-Based human Computer Interaction of E-Health Care System for People with Movement Disabilities // Electronics and Electrical Engineering. ISSN 1392-1215, 2010, 7(103):77-82
4. Dzemydiene, D., Bielskis, A.A., Andziulis A., Drungilas, D., Gricius, G. Recognition of Human Emotions in Reasoning Algorithms of Wheelchair Type Robots // Informatica, 2010, Vol.21 Issue 4, p. 521-532.

The articles in other scientific journals

1. Gricius G., Drungilas D., Guseinovaitė J., Grigaitis K., Bielskis A., Modeling of Human Friendly Multi-Agent Based Sustainable Power Controller, Balkan journal of electrical & computer engineering vol.1, no.2 (ISSN 2147-284X), 2013, Kirklareli / Turkey
2. Bielskis A., Drungilas D., Gricius G., Guseinoviene E., Dzemydiene D., Žutauta L., Modeling of Ambient Comfort Affect Reward Based on Multi-Agents in Cloud Interconnection Environment for Developing the Sustainable Home Controlle, Proceedings of 2013 Eighth International Conference and Exhibition on Ecological Vehicles and Renewable Energies (EVER), ISBN: 978-1-4673-5270-3, EEE Catalog Number: CFP1328U-CDR, Monaco, 2013
3. Gricius G. , Ramašauskas O., Non-rigid image recognition algorithms in applied robotics, Proceedings of International Workshop STOPROG-2012, Stochastic Programming for Implementation and Advanced Applications, ISBN 978-609-9524 1-4-6, Vilnius, Lithuania, 2012

SHORT DESCRIPTION ABOUT THE AUTHOR

Gediminas Gricius was born on 18 November, 1984. He graduated from Klaipėda University Faculty of Nature and Mathematics in 2007 acquiring Bachelor's Degree in Informatics. He gained his Master's Degree in Informatics at Klaipėda University Faculty of Nature and Mathematics in 2010. Since 2010 to 2014 he has been a PhD student at Vilnius University Institute of Mathematics and Informatics. From 2010 he works as a lecturer at Klaipėda University, Department of Informatics.

DAUGIAAGENTINIŲ SISTEMŲ KŪRIMO METODŲ IŠVYSTYMAS NEDIDELIO NAŠUMO ĮTERPTINIŲ SISTEMŲ INTEGRAVIMUI

Temos aktualumas

Šiuolaikinės dirbtinio intelekto sistemos (DIS) yra sudėtingos, integruojančios skirtingas, tarpusavyje turinčias komunikuoti programinės bei techninės įrangos komponentes. Norint užtikrinti realiame laike sudėtingomis išorinės sąlygomis dirbančių sistemų veikimą, tenka spręsti heterogeninių komponentų architektūros ir sąveikos klausimus, siūlyti architektūros projektavimo bei sistemos įgyvendinimo metodiką. Šis disertacinis darbas skirtas nedidelio našumo (turinčias ribotą skaičiavimo pajėgumą, atminties talpą ir kt.) heterogeninių įterptinių sistemų integravimui į bendrą integruotos sistemos architektūrą, taikant daugiaagentinių sistemų kūrimo paradigmą.

Įterptinių sistemų klasei priskiriami įvairios paskirties jutikliai ir vykdikliai, sukurti specialių mikrovaldiklių pagrindu ir skirti specializuotų funkcijų vykdymui. Jos gali būti integruojamos į įvairius kompiuterizuotus produktus: prietaisus, įrenginius. Atsiranda vis daugiau šių sistemų taikymo sričių ir sėkmės pavyzdžių: automobilio darbo režimo valdymas, nuotolinis kelių eismo ar statinių būklės stebėjimas, ligonių būklės stebėjimas, pasitelkiant specialiąją medicininę aparatūrą, mechatroninių pavarų ir robotų valdymas.

Dažniausiai įterptinių sistemų veikimą apriboja jų kompiuterinio našumo pajėgumai. Šioms sistemoms keliami reikalavimai: kuo mažesni gabaritai, nedideli energijos ištekliai, kuo mažesni kompiuterinės atminties resursai, nedidelis operatyviosios atminties kiekio naudojimas ir pan. Nedidelio našumo įterptinėmis sistemomis (angl. *small-scale embeded systems*) laikomos tokios sistemos, kurios turi ribotą procesoriaus galingumą (1-100 Mips), ribotą programinę (1-256 KB) ir operatyvinę (64-16384 baitų) atmintį. Šias sistemas dažniausiai sudaro mikrovaldiklis (-iai), jutikliai (sensoriai) ar kiti įvedimo įrenginiai, komunikacijos sąsajos bei indikacijos ar išvedimo įrenginys. Jos privalo veikti kuo ilgiau, sunaudodamos minimalų elektros energijos kiekį, ir kartais vykdyti savo funkcijas esant nepalankioms aplinkos sąlygoms. Taip pat pageidautina, kad tokios sistemos dirbtų belaidžio tinklo sąlygomis.

Nagrinėjant įterptinių sistemų, realaus laiko procesų stebėjimui (monitoringui) skirtas technines komponentes, susiduriama su problemomis: šių komponenčių integravimo, jų sąveikavimo užtikrinimas, belaidžių tinklų protokolų parinkimas bei jų darbo užtikrinimas atskirų belaidžio tinklo komponenčių sąveikoje. Svarbu nustatyti, kokie yra tinkami būdai ir metodai, kad būtų galima sujungti išskirstytas heterogenines sistemas į bendrą, kompleksinę, intelektualiais metodais paremtą sistemą, užtikrinti sistemos veikimą esant savitoms sąlygoms,

kai objektyvios aplinkybės neleidžia pasitelkti didelį galingumą turinčias sistemas. Kuriant tokias sistemas tampa labai svarbūs elgsenos aspektai, kuriuos reikia užtikrinti, t. y. jų savarankiškumas, reagavimas į aplinką bei sprendimų priėmimas. Atskiroms įterptinėms sistemoms reikalingas savybes integruojant į bendrą sistemą svarstoma, kaip jos galės bendrauti ir sąveikauti. Daugiaagentinių sistemų kūrimo metodai leidžia projektuoti sistemas, kurios yra dinamiškos, lengvai plečiamos, geba sujungti heterogeninius elementus į visumą, todėl darbe pasirinkta daugiaagentinių sistemų paradigma, kuri pasirodė tinkama kuriamų aparatūrinių ir programinių sistemų integravimui. Aukščiau minėtos sistemos tampa vis labiau išskirstytomis, decentralizuotomis, o komponentus jungiant į bendrą sistemą, kuri siekia bendro tikslo ir veikia belaidžiame tinkle, susiduriama su šiomis problemomis: skirtingų bendravimo protokolų veikimo užtikrinimas, atskirų komponentų, realizuotų skirtingose fizinėse platformose ir skirtingomis programinėmis priemonėmis, integracija.

Atskirų įterptinių sistemų tarpusavio bendravimas ir bendradarbiavimas yra svarbi iki šiol pilnai neišspręsta problema. Šiame darbe siūloma sukurti nedidelio našumo įterptinių sistemų konstravimo metodiką, pasitelkiant agentinių sistemų kūrimo paradigmą. Keliami reikalavimai, jog tokios sistemos būtų savarankiškos, aktyvios ir komunikuojančios, bet tuo pačiu metu nesuvartojančios didelių skaičiavimo ir energijos kiekių. Todėl būtina tikslingai pritaikyti dažnai naudojamą daugiaagentinių sistemų kūrimo metodiką nedidelio našumo įterptinių sistemų įtinklinto bendravimo ir bendradarbiavimo kūrimui.

Siūloma sistemos architektūra, kuri pasižymėtų proaktyvumu, galimybėmis prisitaikyti ir taip padidinti sistemos atsparumą darbui, esant sudėtingoms sąlygoms. Tokių sistemų taikymo sritis galėtų apimti jūros tyrimus ir aplinkos monitoringą, šiuo atveju, sistemos turėtų dirbti sūriame jūriniame vandenyje, esant dideliam banguotumui, vėjo pokyčiams, užšalimui. Be to, daugiaagentinių technologijų taikymas suteiktų galimybes šių sistemų dinaminiam išplėtimui pagal poreikius.

Problemos formulavimas

Programiniai agentai (kaip autonomiškai dirbančios kompiuterinės programos) ir iš jų sudarytos daugiaagentinės sistemos yra sparčiai besivystanti programų inžinerijos ir dirbtinio intelekto šaka. Jų taikymo galimybės gerai žinomos tokiose sistemose kaip e. patarėjai, e. pašto filtrai ir sudėtingesnėse sistemose, sugebančiose valdyti komplikuotas technines sistemas ir net kosminius laivus. Daugiaagentinių sistemų kūrimo metodai leidžia projektuoti sistemas, kurios pasižymi dinamiškumu, lengvu plečiamumu, gebėjimu jungti heterogeninius elementus į visumą, todėl daugiaagentinų sistemų paradigmos taikymas aparatūrinių ir programinių sistemų integravimui nagrinėjamas daugelyje mokslinių darbų: Carrasco, 2010; Ostaševičiūtė,

2007. Analizuojant ir sprendžiant sudėtingas problemas specialus vaidmuo yra skiriamas agento ir daugiaagentinės sistemos kūrimo etapams (Borodini, 2007). Agentas reiškia abstraktų subjektą, kuris gali išspręsti tam tikrą arba dalinę problemą, keli agentai gali būti integruoti į bendrą sistemą, kurioje veikdami kartu gali susidoroti su sudėtingesnėmis problemomis. Toliau bus laikomasi pagrindinės nuostatos, jog viena įterptinė sistema atitiks vieną agentą. Tradicinių daugiaagentinių sistemų realizacijai taikomos aukšto lygio, interpretuojamos programavimo kalbos (dažniausiai JAVA), reikalaujančios didelių kompiuterinių pajėgumų, tačiau tokios kalbos nėra tinkamos daugiaagentinių sistemų realizavimui nedidelio našumo įterptinių sistemų grupėse dėl keliamų didelių reikalavimų kompiuteriniams resursams.

Integruojant heterogenines nedidelio našumo įterptines sistemas į bendrą sistemą kuri siektų bendro tikslo, susiduriama su šiomis esminėmis problemomis:

- naudojami skirtingi bendravimo ir belaidžio tinklo protokolai,
- posistemės dažniausiai realizuotos skirtingose fizinėse (kompiuterinėse) platformose
- realizuotos skirtingomis loginėmis (programinėmis) priemonėmis.

Pavyzdžiui, jutikliai, atliekantys aplinkos stebėjimo vaidmenį, yra labai diferenciniai dėl vis kitokios stebimų būsenos kintamųjų ir parametrų fizinės prigimties (nuo temperatūros matavimo iki cheminių medžiagų koncentracijų matavimo). Jų duodami signalai yra taip pat skirtingi, todėl jų apdorojimui ir perdavimui reikalinga skirtinga techninė įranga bei programiniai gavimo, analizavimo ir perdavimo metodai. Kiekvienas tokios heterogeninės sistemos įterptinis komponentas gali turėti savo tikslus, tačiau kiekvienas jų taip pat dalyvauja bendroje sistemoje ir prisideda prie bendro sistemos tikslo siekimo. Siekiant užtikrinti heterogeninių sistemų darbą, tenka spręsti resursų išskirstymo uždavinius.

Išskirstytų decentralizuotų sistemų komponentus jungiant į bendrą sistemą, kuri siekia bendro tikslo ir gali dirbti belaidžiame tinkle, susiduriama su šiomis problemomis: kaip užtikrinti skirtingų bendravimo protokolų veikimą, kaip integruoti atskirus komponentus, realizuotus skirtingose fizinėse platformose bei skirtingomis programinėmis priemonėmis (Jamont ir kt., 2013). Šioms sistemoms keliami reikalavimai: sistemos turėtų ir galėtų rinkti informaciją iš supančios aplinkos, veikti autonomiškai, analizuoti situacijos ir aplinkos pasikeitimus, savarankiškai priimti sprendimus ir juos įvykdyti. Todėl jų bendravimui ir bendradarbiavimui reikalingi efektyvūs belaidžių tinklų protokolai, kurie įgalintų patikimą nedidelio našumo įterptinių sistemų sąveiką. Svarbu šiose sistemose išvystyti ir įvairių situacijų

atpažinimo metodus, tačiau dėl per didelės tokių darbų apimties bus apsiribojama tik kai kurių parametrų stebėjimo atpažinimo problemomis.

Heterogeninės įterptinės sistemos pasižymi didesniu techniniu bei programiniu sudėtingumu. Joms reikalinga bendra techninio bei programinio aprūpinimo platforma ir atskiri komponentai, kurie galėtų būti apjungiami tik paskutiniame kūrimo proceso žingsnyje. Šioms sistemoms neretai tenka kurti techniniame lygyje integruotus kodavimo – dekodavimo algoritmus, transformacijų algoritmus, tinklo tvarkykles, duomenų perdavimo protokolus (TCP/IP, ZigBee), kurie padėtų sumažinti procesoriaus apkrovą dėl papildomų skaičiavimų. Visų šių taikomųjų programų veikimą dažniausiai užtikrina realaus laiko operacinės sistemos.

Tyrimo objektas

Metodai ir programinės priemonės, kurios leistų išvystyti daugelio programinių agentų sąveiką ir užtikrinti jų bendravimą taikomą nedidelio našumo įterptinų sistemų tinklaveikoje.

Disertacinio darbo tikslas

Sukurti nedidelio našumo įterptini sistemų bendravimo ir bendradarbiavimo platformą, grindžiamą daugelio programinių agentų sąveikavimo metodais ir priemonėmis, realizuoti ir eksperimentiškai išbandyti jos prototipą, taikant jį Baltijos jūros hidrometeorologinių duomenų stebėsenai.

Tikslui pasiekti keliami šie disertacinio **darbo uždaviniai:**

1. Išnagrinėti daugiaagentinių sistemų veikimo principus, modelius, jų kūrimui skirtas metodikas ir pasiūlyti metodiką, kuri būtų tinkamą nedidelio našumo heterogeninių įterptinių sistemų sąveikai ir integracijai.
2. Ištirti daugiaagentinių sistemų kūrimo platformų funkcines savybes, atlikti jų lyginamąją analizę ir parinkti tas, kurios būtų tinkamos heterogeninių sistemų sąveikai užtikrinti.
3. Išanalizuoti realaus laiko operacines sistemas (RTOS) ir atlikti (greitaveikos) eksperimentinį tyrimą, nustatyti jų tinkamumą daugiaagentinės sistemos realizavimui nedidelio našumo įterptinių sistemų lygmenyje.
4. Sukurti daugiaagentinių sistemų kūrimo metodiką ir realizavimo platformą, skirtą integruoti nedidelio našumo heterogenines įterptines sistemas.

5. Eksperimentiškai verifikuoti pasiūlytą metodiką, realizuojant integruotą sistemos prototipą skirtą jūros hidrometeorologinių duomenų stebėjimui esant skirtingoms klimato kaitos sąlygoms.

Tyrimo metodai

Analitinėje disertacijos dalyje pateikiama mokslinės literatūros, egzistuojančių daugiaagentinių sistemų kūrimo metodikų ir programinės įrangos analizė. Šioje dalyje taikomi kokybinio tyrimo metodai: informacijos paieška, sisteminimas, lyginamoji analizė bei sukauptos informacijos apibendrinimas.

Projektuojant realaus laiko daugiaagentinės platformos architektūrą, skirtą nedidelio našumo įterptinėms sistemoms, taikytas tyrimo konstravimu metodas (angl. *constructive research*). Šis metodas yra tyrimo procedūra, generuojanti naujoviškas konstrukcijas (modelius, diagramas, metodus, algoritmus ir kt.), skirtas spręsti realaus pasaulio problemoms ir, taikant jas, padaryti tam tikrą įnašą, nagrinėjamos disciplinos teorijoje (Kasanen 1993). Šis tyrimas apėmė realaus laiko OS architektūrų, skirtų nedidelio našumo įterptinėms sistemoms, nagrinėjimą, daugiaagentinės platformos, skirtos NNĮS, projektavimą ir kūrimą, kurio metu pasiūlyti nauji metodai bei modeliai, įgalinantys DAS taikymą tokio tipo sistemose.

Siekiant atlikti pasiūlytos programinių agentų architektūros, skirtos nedidelio našumo įterptinėms sistemoms, validaciją buvo atliktas eksperimentinis tyrimas. Tyrimo metu taikant aprašytąją metodiką sukurta integruota daugiaagentinė hidrometeorologinių stebėjimus atliekančių plūdurų sistema. Eksperimentinio tyrimo metu buvo analizuojamos šios sistemos charakteristikos: duomenų perdavimo kiekis ir sistemos adaptacija greitai besikeičiančiose klimato sąlygose. Gauti rezultatai palyginti su šiuo metu naudojamų plūdurų sistemų charakteristikomis.

Gauti darbo rezultatai

Ištirti daugelio agentų sąveikos metodai ir išanalizuotos realaus laiko nedidelio našumo operacinių sistemų taikymo galimybės leido įvertinti, kad realaus laiko operacinė sistema FreeRTOS geriausiai tinka tokių sistemų valdymui, kai yra nedidelis resursų atminties poreikis (kaip pvz., 16-20 MIPS skaičiavimo greičio, 2-8 KB SDRAM tipo atminties, 32-256 KB flash tipo atminties).

Mažo našumo įterptinių valdiklių, kurie veikia kaip atskiri daugiaagentinės sistemos agentai, bendravimui labiausiai tinkami ZigBee Mesh belaidžių tinklų protokolai, kurie buvo

įvertinti ir nustatytas jų tinkamumas nedidelio našumo įterptinių sistemų integracijai į bendrą belaidžio tinklo sistemą.

Pasiūlyta daugelio agentų sąveikavimu grindžiamos sistemos išplėtota infrastruktūra ir jos kūrimo metodika, grindžiama DIAMOND metodika ir PROMETHEUS daugiaagentinių sistemų kūrimo platformos principais, kurie yra labiausiai tinkami nedidelio našumo įterptinių sistemų integracijai į bendrą belaidžio tinklo architektūrą.

Eksperimentiškai išbandytas integruotos sistemos veikimas, realizuojant joje daugelio agentų sistemos bendravimo ir bendradarbiavimo savybes. Parodyta, jog pasiūlytos metodikos parinkimas nedidelio našumo įterptinių sistemų integracijai tinka labiau nei kitos parengtos metodikos. Vykdant eksperimentą pademonstruota pasiūlytų ir įgyvendintų sprendimų praktinė reikšmė: parodyta, kad tokio tipo sistema gali būti taikoma sprendžiant ekologinio monitoringo uždavinius, nes veikia esant realioms Baltijos jūros aplinkos ir klimato sąlygoms.

Mokslinis naujumas ir rezultatai

Darbe pasiūlyti metodai ir jų taikymo metodika praplečia daugiaagentinių intelektinių įterptinių sistemų kūrimo galimybes nedidelio našumo įterptinių sistemų integravime. Dauguma siūlomų agentinių sistemų kūrimo platformų yra skirtos didelį pajėgumą turinčių įterptinių sistemų kūrimui, veikiančioms esant stacionarioms sąlygoms ir turinčioms dideles duomenų saugyklas (kompiuterinės atminties). Jų realizacijai dažniausiai taikomos aukšto lygio programavimo kalbos (pvz. JAVA), tačiau jos nėra tinkamos nedidelio našumo įterptinių sistemų darbo užtikrinimui.

Šiame darbe pasiūlyta metodika grindžiama daugelio agentų sąveikos principais ir yra skirta išskirstytų, heterogeninių įterptinių sistemų integravimui. Kiekviena įterptinė sistema kaip atskiras tam tikrų procesų stebėjimui skirtas komponentas, gali turėti savo mikrovaldiklį, specifinių sensorių ir kitų elementų. Įterptinių sistemų integravimas bendroje daugiaagentinių sistemų metodais grindžiamoje sistemoje leido įgyvendinti tikslo siekimo ir sąveikos metodus, pasitelkti ir taikyti sąveikos bei funkcinių agentų galimybes. Manoma, jog gauti moksliniai rezultatai turės įtakos tolimesnėje dirbtinio intelekto sistemų plėtroje, ypač vystant išmaniųjų įrenginių valdymo sistemas.

Sukurta integruota platforma suteikianti galimybes ne tik palengvinti ir pagreitinti daugiaagentinių sistemų fizinę realizaciją, bet ir efektyviai naudoti daugiaagentines technologijas heterogeninėse įterptinėse sistemose, kas leidžia užtikrinti visų posistemių (agentų-komponentų) tarpusavio suderinamumą ir tokių sistemų praplečiamumą.

Praktinė rezultatų svarba

Mikrovaldiklių taikymo galimybių daugėja daug sparčiau negu jų kaina, todėl įterptinės sistemos integruojamos į prietaisus, mašinas, technologinę įrangą ir vis daugiau lemia jų išmanumą bei išskirtines konkurencines savybes. Darbo rezultatai gali būti sėkmingai panaudoti praktikoje, kuriant ekstremaliomis sąlygomis dirbančias išmaniąsias sistemas.

Praktiškai realizuota ir eksperimentiškai išbandyta daugiaagentinių mažo našumo įterptinių sistemų kūrimo metodika, sukurta technologinė platforma ir atliktas jos eksperimentinis tyrimas bei vertinimas, įrodantis metodikos pagrįstumą.

Pasiūlyta daugelio agentų sąveikavimu grindžiamos sistemos išplėtota infrastruktūra ir jos kūrimo metodika, grindžiama PROMETHEUS daugiaagentinių sistemų kūrimo platformos principais, kurie pasirodė labiausiai tinkami nedidelio našumo įterptinių sistemų integravimui į bendrą belaidžio tinklo architektūrą.

Eksperimentiškai išbandytas integruotos sistemos veikimas, realizuojant joje daugelio agentų sistemos bendravimo ir bendradarbiavimo savybes. Parodyta, kad tokios metodikos parinkimas geriausiai tiko nedidelio našumo įterptinių sistemų integravimui. Taip pat pademonstruota, kad tokio tipo sistema veikia esant realioms Baltijos jūros aplinkos ir klimato sąlygoms. Darbo metu buvo sukurta reali bandymų platforma Klaipėdos universiteto mokslinių bandomųjų tyrimų laboratorijoje. Šiame darbe pateikti rezultatai buvo panaudoti vykdant šiuos MTEP projektus:

- „Įterptinių sistemų mokomųjų laboratorinių stendų kūrimas“ (projekto Nr.: VP2-1.3-ŪM-05-K-02-023), projektas finansuotas pagal Ekonomikos augimo veiksmų programos 1 prioriteto „Ūkio konkurencingumui ir ekonomikos augimui skirti moksliniai tyrimai ir technologinė plėtra“ įgyvendinimo priemonės VP2-1.3-ŪM-05-K „Inočekiai LT“ įgyvendinimą visuotinės dotacijos būdu (2013 -2014 m.).
- „Įmonės klientų informavimo būdų analizė bei šiuolaikiškos ir ekonomiškai adaptyvios informavimo sistemos sukūrimas“ (projekto Nr.: VP2-1.3-ŪM-05-K-03-475), projektas finansuotas pagal Ekonomikos augimo veiksmų programos 1 prioriteto „Ūkio konkurencingumui ir ekonomikos augimui skirti moksliniai tyrimai ir technologinė plėtra“ įgyvendinimo priemonės VP2-1.3-ŪM-05-K „Inočekiai LT“ įgyvendinimą visuotinės dotacijos būdu (2013 -2014 m.).
- „Baltijos Jūros hidrometeorologinių stebėjimo plūdurų duomenų perdavimo sistemų modernizavimas“ projektas, finansuotas pagal Žmogiškųjų išteklių plėtros veiksmų programos 3 prioritetą „Tyrėjų gebėjimų stiprinimas“

įgyvendinimo VP1-3.1-ŠMM-01-V priemonę „Mokslininkų ir kitų tyrėjų mobilumo ir studentų mokslinių darbų skatinimas“ (2014 m.).

Ginamieji teiginiai

1. Mažo našumo įterptinių sistemų bendravimui ir bendradarbiavimui heterogeninėje aplinkoje tikslinga taikyti modifikuotą daugelio programinių agentinių sistemų kūrimo DIAMOND metodiką.
2. Sudėtingose integruotose įterptinėse sistemose, kurias sudaro daug atskirų heterogeninių įterptinių sistemų (mikrovaldiklių), tikslingiau naudoti išskirstytas daugiaagentines sistemas nei vieną centralizuotą agentinę sistemą, esančią viename kompiuteryje, o atskirų komponentų (sistemų) bendravimui naudoti belaidį DigiMesh protokolą.
3. Pasiūlyta metodika ir kūrimo platforma kompleksiškai pritaikyta mažo našumo įterptinių sistemų kūrimui. Eksperimentiškai išbandyta meteorologinių reiškinių stebėjimo sistemai kurti, tai sumažina duomenų perdavimo kiekius, padaro sistemą adaptyvesnę.

Darbo rezultatų aprobavimas

Disertacijos tema yra paskelbti 7 moksliniai straipsniai: 4 recenzuojamuose mokslo žurnaluose, vienas – leidinyje, įtrauktame į Mokslinės informacijos instituto konferencijų darbų sąrašą (ISI Proceedings), 3 kituose respublikinių konferencijų medžiagose (publikacijų sąrašas pateiktas darbo gale).

Disertacinio darbo rezultatai pristatyti 6 tarptautinėse konferencijose ir 2 nacionalinėse konferencijose:

1. International Academic Conference in Social Technologies '13: Social Innovations: Theoretical and Practical Insights, November 10 – 11, 2013, Vilnius.
2. Practical Conference Virtual Instruments in Biomedicine, 2013, Klaipėda, Lithuania.
3. International Workshop, Stochastic Programming for Implementation and Advanced Applications, July 3-6, 2012, Neringa, Lithuania.
4. International Academic Conference Social Technologies '11: ICT for social transformations, November 17 – 18, 2011, Vilnius.
5. 15–oji Respublikinė konferencija Kompiuterininkų dienos (Klaipėda, 2011); rugsėjo 22-24, 2011.
6. The 14th International Conference Electronics‘2010, May 18-20, 2010, Kaunas.

7. The 10th International Conference Reliability and Statistics in Transportation and Communication (RelStat'10), October 20-23, 2010, Riga, Latvia.
8. 7-oji mokslinė konferencija Technologijos mokslo darbai Vakarų Lietuvoje, 2010 m. gegužės 14 d., Klaipėda.

Darbo apimtis ir struktūra

Disertaciją sudaro įvadas, 5 skyriai, išvados, naudotos literatūros sąrašas, autoriaus publikacijų sąrašas.

Disertacijos apimtis – 120 puslapių, iš jų - 10 lentelių ir 35 paveikslėliai.

Įvade aprašomas mokslinio tyrimo aktualumas, analizuojama jo reikšmė, pateikiamas šiuolaikinių mokslinių tyrimų kontekstas ir problemos formuluotė, pagrindžiamas problemos aktualumas, formuluojamas tikslas bei uždaviniai, apibrėžiamos tyrimo hipotezės ir pristatoma tyrimo metodika. Aptariami pagrindiniai disertacinio darbo rezultatai, jų praktinė vertė ir naujumas, disertacijos rezultatų publikavimo rodikliai bei aprobavimas.

Pirmajame skyriuje nagrinėjami daugiaagentinių sistemų veikimo principai, funkcinės galimybės ir keliami reikalavimai intelektualizuotų kompiuterinių sistemų kūrimui. Analizuojamos daugiaagentinių sistemų kūrimo metodikos, ieškoma būdų, kaip pritaikyti jų veikimo principus ir metodus nedidelio našumo įterptinių sistemų veikimo programavimui. Pateikiama dažniausiai naudojamų agentinių sistemų kūrimo metodikų apžvalga. Įvertinus populiarių agentinių sistemų kūrimo metodikų GAIA ir DIAMOND savybes, parodyta, jog tolimesniam darbui labiau tinkama DIAMOND metodika, kuri gali būti taikoma ir nedidelio našumo išskirstytų įterptinių sistemų kūrimui.

Antrajame skyriuje nagrinėjamos esamos daugiaagentinių sistemų platformos, jų darbinės aplinkos ir teikiamos funkcinės struktūros, aprašomi svarbiausi šių sistemų komponentai. Nurodomi naudojamų daugiaagentinių sistemų kūrimo platformų trūkumai, kuriuos reikia įvertinti ir pašalinti, norint jas taikyti nedidelio našumo įterptinėse sistemose.

Trečiajame skyriuje nagrinėjamos skirtingos realaus laiko operacinės sistemos ir jų tinkamumas, siekiant užtikrinti nedidelio našumo įterptinių sistemų darbą apribotomis sąlygomis. Remiantis atlikta OS savybių analize ir jų našumo eksperimentiniais tyrimais nustatyta, jog tinkamiausia DAS kūrimui yra operacinė sistema FreeRTOS.

Ketvirtajame skyriuje projektuojama DAS platforma ir apibrėžiama jos architektūra, skirta nedidelio našumo įterptinių sistemų kūrimui. Nagrinėjant atskiras įterptines sistemas kaip DAS agentus, pateikiami jų bendravimo ir sąveikos metodai, tinkami realizuoti belaidžių kompiuterinių tinklų priemonėmis.

Penktajame skyriuje pateikiami eksperimentinių tyrimų rezultatai įvertinant sukurtą hidrometeorologinių stebėjimų Baltijos jūros akvatorijoje sistemą. Pateikiami realizuotos sistemos techninės ir programinės įrangos vertinimo rezultatai, kurie parodo, jog pasiūlyta daugiaagentinės sistemos taikymas palengvina funkcinį plečiamumą ir nedidelio našumo heterogeninių įterptinių sistemų integravimą. Eksperimentiniu būdu pademonstruota kad tokios integruotos įterptinės sistemos yra pajėgios efektyviai dirbti esant ribotam atskirų posistemių darbo našumui.

Galutiniame skyriuje pateikiami darbo rezultatai ir išvados, kurios pagrindžia ginamus teiginius.

Bendrosios išvados

1. Išanalizavus metodikas, skirtas bendrosios paskirties daugiaagentinėms sistemoms kurti, nustatyta, kad jos yra per daug abstrakčios, jose neatsižvelgiama į nedidelio našumo įterptinėms sistemoms būdingas savybes ir apribojimus. Išnagrinėjus šių metodikų specializuotus variantus, skirtus įterptinėms daugiaagentinėms sistemoms kurti, nustatyta, kad jos taip pat negali būti tiesiogiai taikomos dėl šių priežasčių:
   - IĮS metodika grindžiama JADE karkasu, kurio vykdymui reikalinga papildomų resursų reikalaujanti JVM;
   - nei IĮS, nei DIAMOND metodika nesiūlo jas realizuojančios platformos ir kodo generavimo įrankių, skirtų nedidelio našumo įterptinėms sistemoms;
   - išdėstymo stadija (apimanti programinės/aparatinės įrangos atskyrimas) atliekama kūrimo ciklo pradžioje, todėl nėra galimybės integruoti aparatinę ir programinę įrangą viso daugiaagentinės sistemos kūrimo metu.
2. Tiriant DAS realizavimo platformas, nustatyta, kad dauguma jų yra skirtos didelį pajėgumą ir duomenų saugyklas turinčioms kompiuterinėms sistemoms veikiančioms prie stacionarių sąlygų. Nustatyta, kad autoriaus iškeltus reikalavimus iš dalies tenkina PDT „PROMETHEUS Design tool“ platforma, kuri leidžia generuoti pirminį programinį kodą įterptinėms sistemoms (iki klasių diagramų), bei gali būti integruojama su DIAMOND kūrimo metodika.
3. Atlikus eksperimentinius realaus laiko operacinių sistemų galimybių ir tinkamumo nedidelio našumo įterptinėms sistemoms tyrimą galima teigti jog tinkamiausia DAS kūrimui yra FreeRTOS operacinė sistema.

4. Modifikavus atrinktą kūrimo metodiką (DIAMOND) buvo pasiūlyta daugelio agentų sąveikavimu grindžiamos sistemos išplėtota infrastruktūra ir jos kūrimo metodika nedidelio našumo įterptinėms sistemoms kurti. Išdėstymo stadija, apimanti aparatinės ir programinės įrangos padalijimą, perkelta į kūrimo ciklo pabaigą, taip sudaro galimybę jungti aparatinės įrangos dalies ir programinės įrangos dalies vystymą viso daugiaagentinės sistemos kūrimo ciklo metu. Sudarytas transformavimo modelis skirtas, suprojektuotų DAS agentų atvaizdavimui į RTOS valdomą sistemą. Taip pat pateikiamas skirtingų tipų agentų realizavimo metodai nedidelio našumo įterptinėse sistemose.
5. Pasiūlyta DAS kūrimo metodika eksperimentiškai išbandyta sukuriant ir išbandant hidrometeorologinių duomenų rinkimo ir stebėjimo sistemą. Tokio tipo sistema taikoma sprendžiant ekologinio monitoringo uždavinius, esant realioms Baltijos jūros aplinkos ir klimatinėms sąlygoms. Gauti eksperimentiniai rezultatai demonstruoja, kai stebimi klimato ir aplinkos parametrai nekinta (arba kinta lėtai), sumažėja perduodamų nuoskaitų skaičius sumažėja 87%. Sistema demonstruoja tinkamą adaptyvumo prie greitai besikeičiančių aplinkos sąlygų savybę ir leidžia surinkti išsamesnius matavimų duomenis, nei įprastinės fiksuoto laiko sistemos.
6. Praktinio realizavimo metu buvo įvertinti duomenų perdavimo protokolai ir buvo nustatyta kad nedidelio našumo įterptinių sistemų integravimui į bendrą belaidžio tinklo sistemą – tinkamiausias DigiMesh belaidžių tinklų protokolas.

Trumpos žinios apie autorių

Gediminas Gricius gimė 1984 m. lapkričio 29d. 2007 m. Klaipėdos universiteto gamtos ir matematikos mokslų fakultete įgijo informatikos bakalauro laipsnį. 2010 m. Klaipėdos universiteto gamtos ir matematikos mokslų fakultete įgijo informatikos magistro laipsnį. 2010–2014 m. studijavo doktorantūroje Vilniaus universitete, Matematikos ir informatikos institute (fiziniai mokslai, informatika). Nuo 2010 metų dirba lektoriumi Klaipėdos universitete Informatikos katedroje.

Gediminas Gricius

DEVELOPMENT OF A MULTI-AGENT BASED METHODS FOR INTEGRATION OF SMALL-SCALE EMBEDDED SYSTEMS

Summary of Doctoral Dissertation
Physical Sciences,
Informatics (09 P)

Editor Šarūnė Tilvikaitė

Gediminas Gricius

DAUGIAAGENTINIŲ SISTEMŲ KŪRIMO METODŲ IŠVYSTYMAS NEDIDELIO NAŠUMO ĮTERPTINIŲ SISTEMŲ INTEGRAVIMUI

Daktaro disertacijos santrauka
Fiziniai mokslai,
Informatika (09 P)

Redaktorė Toma Arcišauskaitė-Lukošienė