

THESIS ON INFORMATICS AND SYSTEM ENGINEERING C77

Simulations in Multi-Agent Communication System

VADIM KIMLAYCHUK

TUT
PRESS

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER CONTROL
TALLINN UNIVERSITY OF TECHNOLOGY

**Dissertation was accepted for the defense of the degree of Doctor of Philosophy
in computer and system engineering on 30th of June 2012.**

Supervisor:

Professor, DSc Leo Mõtus, Department of Computer Control,
Tallinn University of Technology

Opponents:

Professor Janis Grundspenkis, Head of the Department of Systems Theory
and Design, Director of the Institute of Applied Computer Systems,
Riga Technical University, Latvia

Professor Kuldar Taveter, Department of Informatics, Head of the Chair of
Software Engineering, Tallinn University of Technology, Estonia

Defence of the thesis: 15 august 2012 at 14-00, II-309

Declaration: Hereby I declare that this doctoral thesis, my original investigation and
achievement, submitted for the doctoral degree at Tallinn University of Technology has
not been submitted for any degree or examination.

/Vadim Kimlaychuk/



Copyright: Vadim Kimlaychuk 2012
ISSN 1406-4731
ISBN 978-9949-23-335-9 (publication)
ISBN 978-9949-23-336-6 (PDF)

INFORMAATIKA JA SÜSTEEMITEHNIKA C77

Simulatsioonid multiagentsüsteemis

VADIM KIMLAYCHUK

Contents

ABSTRACT	9
KOKKUVÕTE	11
LIST OF ABBREVIATIONS	12
1 INTRODUCTION	13
1.1 Novelty and original contribution of the thesis	17
2 APPLICATIONS FOR INTELLIGENT AGENTS	20
2.1 Agent definition and agent application area	20
2.2 Agent structure and agent types	22
2.3 Communication between agents	25
2.4 Information security as a part of social life	26
2.5 JADE simulation platform	27
3 ANT COLONY SIMULATION	30
3.1 Model structure	30
3.1.1 Internal goals	31
3.1.2 Foreign affairs	32
3.2 Agent approach	32
3.3 Distributed model of mobile agents	33
3.4 Ontology	34
3.5 Implementation in JADE agent development environment	36
3.5.1 Running system	36
3.5.2 Examples of GUI	37
3.6 Conclusion	39

4	OPTIMIZE SHARED RESOURCES. 5 HUNGRY PHILOSOPHERS PROBLEM	40
4.1	Process modeling	40
4.2	Agent approach and ontology	41
4.3	Implementation in JADE	42
4.3.1	System	42
4.3.2	Agents	43
4.3.3	Ontology	45
4.3.4	Executive Environment	46
4.4	Results obtained from simulation in JADE	46
4.4.1	Time constraints	46
4.4.2	Homogeneous systems	47
4.4.3	Heterogeneous systems	48
5	COMMERCIAL OFF-THE-SHELF PRODUCT IMPROVEMENTS	49
5.1	SOA structure and main principles	49
5.2	Major integration problems	50
5.2.1	Addresses and names	51
5.2.2	Services are not intelligent	52
5.3	Intelligent agents as an improvement for services	52
5.3.1	Addresses and names	52
5.3.2	Intelligent services	53
5.4	JADE implementation	55
5.4.1	Web services and software agents	55
5.4.2	Making agents work	56
5.4.3	Conclusion and further work	59
6	DISTRIBUTED SENSOR AGENT NETWORKS	60
6.1	Overview of sensor networks	60
6.2	Security problems in WSN-s	61
6.3	Target implementation domain	63

6.3.1	Structure of MICA2/MICA2DOT motes	63
6.3.2	Conception of operation for DSN	63
6.3.3	General network structure	64
6.3.4	Non-symmetric cryptography	65
6.4	TLMK protocol	65
6.4.1	Key pre-deployment	66
6.4.2	Key update/revoke	67
6.4.3	Message exchange	68
6.5	Complex scenarios	69
6.5.1	Multi-hop network	69
6.5.2	Coalitions and group-level security	71
6.6	Simulation of TLMK protocol in MASE	71
6.6.1	Secure mote architecture for MASE	72
6.6.2	Special assumptions made for simulation and the result	73
6.7	TLMK implementation summary	74
7	ACCESS CONTROL BASED ON SHARED KNOWLEDGE	76
7.1	Problem background	76
7.2	Recent situation	77
7.3	Trust function	78
7.4	Requirements for authentication system	79
7.5	Difficulties to design such a system	79
7.6	Teaching agent structure and basic notions	81
7.6.1	Agents roles and properties	81
7.6.2	Agents ontology	82
7.6.3	Results in math study for student/teacher simulation in JADE	82
7.7	Conclusion and further work	84
	CONCLUSION	85
	BIBLIOGRAPHY	86

LIST OF PUBLICATIONS	93
ANNEX A.	95
ANNEX B	97
ANNEX C	99

Abstract

We observe crisis of computational systems that can calculate at very high speed over large data sets, but cannot manage emergent situations that always happen in real world. Today there are many attempts that try to overcome the restrictive boundaries of Turing computational model and create basis for non-classical computations. One of them is self-organized computational systems composed of agents that act in some environment. Individual agents in such a multi-agent systems (MAS) are usually simple entities, but system itself may demonstrate complex behaviors. Interactions in such systems are established by communication acts between agents and the way it happens is important for the sake of functionality. That is why ontologies of agent languages are a large part of this research.

This thesis is devoted to simulation of agent-based systems. We do simulation on our models because we are not able to model emergent behavior (i.e. to describe the emergency in terms of Turing computable functions) in order to build complete system with emergent behavior.

We are studying the properties of multi-agent systems by simulation according to the rules permitted by non-classical models of computation.

Five practical cases have been studied in this thesis. All of them apply multi-agent paradigm to model different real world problems. Systems that comprise entities by their nature were studied first. Simulation of ant colony is the first experiment in applying MAS development principles. The agent-based model of ant colony can further be elaborated, if necessary, to capture features that are essential to model the operation of real-world ant colony.

The other cases have conventionally been resolved without applying agent-based paradigm. We formulated a modified “dining philosophers” problem and web-services management problem in an enterprise as multi-agent systems and demonstrated by simulation the feasibility of this approach. Last two experiments were dedicated to study of features present in any multi-agent system – those related to security of communication. We have suggested and studied by simulation a new security protocol for WSN networks. We have also introduced a new, shared knowledge authentication based process, and have simulated knowledge transfer between software agents. The design of an intelligent agent that can adopt new ontology and perform new actions was deduced from this experiment.

Most of the simulations have been implemented within JADE MAS framework. The results of the simulations demonstrate that agent-oriented

approach gives better results than conventional software development paradigms especially in the domain of the problems that natively can be resolved by using communications between entities.

Kokkuvõte

Viimaste aastakümnete jooksul on püsinud kustumatu huvi multiagentsüsteemide ja agent-orienteeritud süsteemide arenduse vastu. Selle peamiseks põhjuseks, minu arvates, on selle oma nišš rakenduste domeenis. Näiteks, tarkvara arenduses enamik paradigme, nagu funktsionaalne programmeerimine või objekt-orienteeritud programmeerimine katavad ainult täieliku Turingi algoritmi. Isegi paralleelse arvutamise, mis on teostamise mustri järgi väga lähedane MAS-ile (kus iga agent töötab paralleelselt teiste agentidega), eesmärgiks on lahendada samu täielikke Turingi algoritme.

MAS ulatub üle selle printsiibi ja selle ajal kui iga üksik agent võib kasutada üht stukuurse programmeerimise paradigmat (tavaliselt sündmuse-põhist metodoloogiat), terve süsteem võib käituda kui super-Turingi masin. See MAS-i omadus ongi minu jaoks huvitav.

Selles väitekirjas tehakse rida katseid ehitada erinevaid MAS süsteeme ja uurida nende käitumist tuginedes ilmneva printsiibile. Vaatamata sellele, et kõik süsteemi sisendid olid simuleeritud ja ma ei saa väita, et arendatud süsteem demonstreerib ilmneva käitumist, saab hinnata iga süsteemi potentsiaali demonstreerida sellist käitumist reaalses keskkonnas. Iga simuleeritud süsteemi võime demonstreerida ilmneva käitumist rangelt sõltub sisenditest. Simulatsiooni on kasutatud selleks, et lihtsustada MAS-i arendusprotsessi ja ennustada tulemusi eksperimendi varajases staadiumis. Veel enam igal katsel on olnud kasulikud praktilised tulemused, mis on kasutatavad olemasolevate probleemide lahendamisel erinevates rakenduste domeenides. Näiteks, TLMK protokoll sensor sõlmede võrgu turvalisemaks tegemiseks ja SOA agendid web-teenuste funktsionaalsuse laiendamiseks.

MAS süsteemide programmeerimisel katsete jaoks kasutades OO keeli nagu JAVA ja C++ on täheldatud OOP paradigmi suundumus läheneda AOP-le.

Üks kõige huvitavamatest teemadest järgmiseks teadustööks on agentide võime õppida uusi ontoloogiaid ja jagada neid teadmisi teistega. Väitekirji näitab kuidas see teostatakse viimases peatükis. Kuna õpetamise eesmärk on uued käitumised aga mitte lihtsalt olemused ja nende tähendused, on suur täenäosus ehitada tõeline ilmnev MAS, kui selle sisendid ei ole simuleeritud, vaid on loetud sensoritest.

List of abbreviations

AI	Artificial Intelligence
ACL	Agent Communication Language
ACO	Ant Colony Optimization
AMS	Agent Management System
AOP	Agent-Oriented Programming
API	Application Programming Interface
CDC	Connected Device Configuration
CLDC	Connected Limited Device Configuration
COTS	Commercially available Off-The-Shelf
FIPA	Foundation for Intelligent Physical Agents
FSM	Finite State Machine
GUI	Graphical User Interface
JVM	Java Virtual Machine
LAN	Local Area Network
MAS	Multi-agent System
OOP	Object-Oriented Programming
RTJ	Real Time for JAVA
SOA	Service-Oriented Architecture
TLMK	Time Limited Memory Keys (protocol)
TSP	Travelling Salesman Problem
WSIG	Web Services Integration Gateway
WSN	Wireless Sensor Networks

Unless real AI will appear all the attempts to create it are simulations¹.

1 Introduction

Humans have always tried to find someone who is smart enough to help them with every day work. They started from training animals that helped them to hunt, pasture and farm, but the abilities of trained animals are very limited. Only scientific progress gave humans a real hope to get something as intelligent as they or even more intelligent.

Real explosion of interest to intelligent artificial creatures took place half a century ago when computational theory and computers appeared. Together with automation they gave us what we have now – thousands of electro-mechanical devices that help humans everywhere. Nowadays a car (or an aircraft) has so many sensory and processing systems that it can drive without a human driver. Still their intelligence is not high enough to allow them act totally by themselves, in spite of their computational power.

Simulation is very popular approach to develop, test, predict the behavior and study any kind of physical or virtual reality, including AI. Simulation is used in very many contexts and can be executed in different ways. In this work we will focus on computer simulations applying the paradigm of intelligent software agents. Agent-oriented techniques represent one of the new approaches to analyzing, designing and building complex systems. This technology has strong impact on software as the most flexible and versatile area for developers. In this thesis I only consider software agents that operate in cyber space. Those agents that act outside of the cyber space are beyond the scope of this thesis, but it does not mean that principles described here cannot be applied to design other types of agents.

Nowadays agents provide information services to their owners acting successfully in complex heterogeneous networked environment with unstable and unpredictable network configuration. They have potential to improve significantly current practice in software engineering and can easily tackle the

¹ Not taking into account AI as a set of scientific disciplines that study reasoning, knowledge representation, etc., but rather focusing on the entity that possibly can be considered as intelligent being. Considering different philosophical approaches of what AI is (or can be) Turing's behavioral judgment of the machine intelligence is what we are taking as a basement for AI identification. Ultimately – real AI is a machine that acts as intelligently as human being. Dartmouth [McCarthy, et al., 1955] participants and most AI researchers believe that such machine is possible.

extended range of applications. This potential is far from being fully realized and the main reason is that agents are typically introduced too late in development cycle of system of systems.

This thesis projects some real world problems onto the cyber space and applies simulation to resolve some of those problems. I will show that agents can help to solve not only mathematical (e.g. monitoring or decision-making) problems, but also help to improve the existing COTS software products.

The main objective pursued in this thesis is to develop experience and new methods, of how the multi-agent systems can be built and used to simulate the properties and behavior of different real world processes and tasks.

Problems that are studied in this work have been taken from different domains and are related to each other at least by ability to generate complex (emergent) behaviors. These problems are simulated using agent-based techniques.

This thesis focuses on 5 disparate cases listed in the order of their presentation: “Ant colony simulation” (chapter 3, page 30), “five hungry philosophers” (chapter 4, page 40), “Agents for COTS²” (chapter 5, page 49), “Time limited memory keys (TLMK) protocol” (chapter 6, page 60) and “Access control based on shared knowledge” (chapter 7, page 76).

We start from behavior simulation of an ant colony as asset of interacting software agents (ants, map of the environment, resources, and nests). The ant colony is described as multi-agent system (MAS). The importance of ontology harmonization and agent communication language (ACL) is discussed. The system is implemented in JADE platform [Bellifemine, et al., 2003] [Bellifemine, et al., 2007]. The agents here are software programs written in JAVA and running on a variety of hosts. Graphical user interface shows simulation results: separately for each agent or for groups of agents. Agent conversations can be controlled visually by Sniffer agent (a part of JADE platform), or using plain text files generated by each agent. The system can be used to solve such practical problems as routing, scheduling, capacity planning, travel salesman problem (TSP), etc.

Next part of the thesis has less social orientation and shows how agent-based approach helps to solve tasks of using resources and improving working capacity of individual agents, by modifying a well-known setting of “five hungry philosopher” originally formulated in 1965 by Edsger Dijkstra as a student exam exercise. JADE agent platform and JAVA is again used for this simulation

² This is an acronym for Commercial Of-The-Shelf products. Particularly here we mean the software that can be purchased on the market.

again. Main objective of this experiment is to apply methods of extreme programming for building MAS and introduce time constraints for agents. The system also has GUI to control “philosophers” and for visualization the inter-agent conversations.

Another experiment is related to one of the complex areas of Telecom industry – information systems integration. Many companies use web services within SOA (service oriented architecture) architecture to support their business [Bieberstein et al., 2005] [Marks et al., 2006]. In any enterprise business, production, and logistics related information flows use many different data stores that are distributed in space and are based on heterogeneous technologies. I used JADE agents to improve the performance of SOA by introducing agents to control services in the web, and into business processes that comprise commercial product within a company. Agent’s proactive behavior supplies web-services with service/client discovery, usage statistics, service version control and connectivity capabilities that are currently missing. The main objective of this experiment is to show that software agents, particularly JAVA-based, can be integrated with commercial software. MAS is not a separately standing technology, but can be integrated into (or cooperate with) any existing application.

Finally an attempt to cross boundary between cyber space (of software agents) and physical space (of the real world environment) has been made. Here I took a quick look on ad-hoc wireless sensor networks (WSN) [Akyildiz et al., 2002] and communication between its nodes. I started with wireless distributed sensor network built on MICA2 motes [Bramwell, 2006]. A mote comprises several small independent, but interacting devices, e.g. a set of sensors (for instance, light, temperature, and sound- sensors), CPU, small flash memory, battery, and radio to transmit and receive data. It works under TinyOS operating system that allows user to write software. These restrictions together with multi-hop message passing over open wireless communication channels make the task of developing simple and reliable security protocol very complex. I underline that authentication in such a system (without direct access to central node, or in the absence of a central node) is not trivial thing as well. It requires secure routing protocol that can be managed by nodes themselves. I have suggested Time Limited Memory Keys (TLMK) protocol that solves some critical security problems in distributed sensor networks at low cost in terms of resources. Together with appropriate routing protocol the level of security in wireless ad-hoc networks can be enhanced. The objective of this case was to find a security protocol that enables secure communication between agents, in this case

between MICA2 motes sensors. TLMK protocol has low demands to hardware resources in network nodes and provides appropriate³ level of security for WSN.

It is not possible, at the moment, to run JADE agents on MICA motes although JADE exists for mobile devices built on Android, CDC and CLDC configurations. The reason for that are insufficient HW resources to run JVM. . Since almost every single CPU tact is valuable for these devices there have been no serious efforts towards implementing any virtual execution environment on MICA motes. That is why our simulations this time were made in special simulation environment developed at Research Laboratory for Proactive Technologies called “Multi-agent environment Simulator” [Tomson, 2009]. This provides a Tiny OS simulated environment that allows the developer to run the studied source code on real MICA2 sensor nodes without modification. This gives us confidence that simulated agent is very close by all features to the real one especially with respect to behavior and resource consumption. The code of TLMK protocol has been tested in simulator to assess its performance and resource consumption.

There is a drawback in all identity mechanisms [Magno, 1996] of modern access control – they reveal no other properties of the identified object but whether its identity token is recognized or not [Sasse, 2005]. This thesis tries to overcome the problem by extending authentication process with history awareness – e.g. by considering the use of shared knowledge. This assumes that the communicating agents are intelligent enough to determine the level of trust they can have to their partner by analyzing the knowledge shared between partners in the previous communication acts (i.e. history of interactions).

Shared knowledge-based authentication assumes the ability to automatically harmonize, or transform, the ontologies used by the group that wants to apply shared knowledge-based authentication. The idea is illustrated by observing the communication of two agents: “teacher” and “student” who exchange the knowledge about a new concept. As a result of the communication act “student” obtains new knowledge and becomes able to extend its ontology, simulating thus the process of real learning where software does not need to be reloaded or recompiled in order to capture new functionality.

We also define probabilistic trust-function that measures level of shared knowledge and gives us level of trust towards the opponent as a rational number between 0 and 1. To evaluate this function in simulation environment agent must be able to work with ontology and study new concepts.

³ Appropriate level of security is the level when the cost of network capture by intruder is higher than the cost of the network

1.1 Novelty and original contribution of the thesis

Agent-oriented software development is not a new approach [Wooldridge, 1997] [Wooldridge et al., 1995] but despite advantages over object-oriented programming (OOP) it can't get yet leading position on the market and support from major compiler vendors.

AOP also lacks standards and every framework developer defines their own. There is an attempt to unify the way agents communicate to each other and propose standards for message exchange done by FIPA [FIPA, homepage]. However not all agent framework suppliers follow these standards. We believe AOP engineering will stand for a long time as a separate framework for major compilers and will not produce a new programming language in the nearest future. For the agent developers it will matter which framework to choose rather than programming language choice.

We used JADE to simulate most of our projects. When we started in 2003 JADE platform was very young but promising framework for agent development and test. By now this framework is almost 10 years under continuous development and new releases of the software are done on a regular basis. We contributed to the JADE source code development [Kimlaychuk, 2010] and now it is possible to serialize concepts, predicates and actions for bean ontology (and not only for that type of ontology).

The other contributions are summarized below:

In “Ant colony simulation” project we simulated ants, nests and 2-D map with a food sources. Real ants use special substance called *pheromone*. Ants moving towards the food source and back deposit this substance on the ground making *pheromone trails* that other ants can smell. The decision which way to move every ant makes depending on trails density. The mathematical model based on real ant behavior and researched in project “Ant colony optimization” [Dorigo et al., 2007] (ACO) helps to find solutions for some practical tasks such as routing, scheduling, travel salesman problem (TSP), etc. In our project we map this mathematical model to software agents and run different simulation tasks in JADE. Particularly we combine the power of distributed agent platform with central web-server to control using GUI the TSP problem solving in a real time. Software agents represent here ants, ant nests and map where ant movements take place. Special ontology is developed for agent communication using Protégé tool [Mussen, 2011]. Ontology is used for generating JAVA classes agents can communicate with using FIPA language. There is an article published in proceedings at EISTA2004 conference [Kimlaychuk, 2004]. The novelty of this work is in making agent oriented design principles work for simulation of real insects.

In “five hungry philosophers” project we simulated virtual entities – philosophers, hospital, resources and resource managers. This works proves the concept of AOP design for simulation once again. We used more sophisticated ontology and the behavior of agents was less predictable (more emergent). This happens because of real-time constraints we tried to implement in this project. The novelty of this project is implementation of real-time for Java principles (RTJ) [Bollella, et al., 2000] in AOP and simulation. The results were introduced in electronic journal EXP [Kimlaychuk, 2003].

In “Agents for COTS” project we extended the functionality of industrial product. Usually commercial products are supported by the manufacturer. There are public APIs that allow some customization of the product. In many cases academic world and the manufacturer of the commercial software tools live separate lives. There is a long process of standardization and approval for agent-oriented technology to be widely accepted as software production technology. This project is a good example that shows how easily agent-oriented software engineering methods can be fitted into existing programming paradigms. We use JADE WSIG add-on as a basement for creating agent-based web-service search engine and directory tree for Oracle SOA Suite. Project shows how web-services (passive by their nature) can be organically combined with pro-active java agents. There are 2 published articles for this theme [Kimlaychuk, 2008] [Kimlaychuk SOA, 2008]. Major achievement in this project is integration between two different software groups based on AOP principles.

In “TLMK protocol” project we developed new protocol for wireless sensor networks that has very low demands for hardware and at the same time gave us proper level of security. There are many methods, models, routing protocols and standards that try to reduce the overhead in the WSN with still maintained some level of security. Adding security to any application increases the overhead, but for sensor networks that operate with limited memory and power resources this overhead is especially important. TLMK is new protocol to secure agent communications in WSN. It is based on Ottway-Rees protocol [Ottway et al., 1987] where in addition to key and “salt” there is a key lifetime transferred to the target node. There are also some specific implementation patterns that make this protocol very attractive to be used as authentication and encryption protocol for wireless sensor networks. Simulation of the protocol is done in MAS simulator and agent's source code can be distributed to the target platform without modification. Protocol is introduced at BMEI'2011 conference [Kimlaychuk V., 2011]. TLMK protocol can be more valuable (secure) in combination with secure routing protocol.

In “Access control based on shared knowledge” project we investigated problem of user authentication described in [Toomim et al., 2008]. Authors define the knowledge to share and build the database, users authenticate through, but this practical work lacks mathematical model definition for doing appropriate conclusions about quality of access control. Recently only statistical methods are used. We have introduced trust function and probabilistic authentication function and made assumptions on how to evaluate them. Statistical methods are used to make trust function more accurate. Practical part accomplished within this project includes ontology development for simulated agent’s learning process. There is a concept that one agent “knows” but the other does not. During communication act the other agent gets this new concept and can start to use it in its actions. We underline here that knowledge is not a data in information technology but rather an execution block of code that can be integrated into agent’s existing code. This execution block usually contains some static data as well (like constants, parameter values, references). Agents are programmed in JAVA using JADE platform. This project is still on-going. However, we already developed a new method on how to expand ontology dynamically. The results are published in paper [Kimlaychuk, V. 2012]

2 Applications for intelligent agents

Software engineering has already passed some milestones on the way of technological improvements that come with every new CPU. We will not be very scrupulous to argue about what was the first programming device, rather count the time from the invention of the CPU and first programming paradigm – imperative programming. Since then many different paradigms were used and the most known at the moment is object-oriented programming. It became the dominant programming methodology during the mid-1990s, largely due to the influence of C++. Nowadays OOP is a major programming paradigm, but it does not mean, that it can cover all engineering demands even in the field of classical computations [Cardelli, 1996]. For the non-classical computations (or super-Turing machines) this paradigm does not work at all [Stepney, et al., 2005].

If we take the original definition for agent oriented programming (AOP) proposed by paradigm inventor [Shoham, 1993] and the modern state of being in OOP we can see quite few differences. They all concern the notion of base entity in both paradigms: class for OOP and agent for AOP. During the last decade requirements for AOP has also being changed, but the gap between these two paradigms constantly decreases. That is why recent agent programming is done in powerful OOP languages – JAVA or C++/C# nevertheless some of the key notions of AOP are not explicitly described in OOP.

For systems, that can be naturally modeled as societies of interacting autonomous entities (interaction computations) [Motus, et al., 2005] the agent-oriented paradigm suits better. Nevertheless objects, as they are defined in OOP, have become a foundation for more complex entities – agents.

2.1 Agent definition and agent application area

There are many definitions of what is agent [Russel et al., 2003] [Wooldridge et al., 1995]. I take one that describes the way I used them better: “an agent is an encapsulated computer system that is situated in some environment, and that is capable of flexible, autonomous action in that environment in order to meet its design objectives” [Wooldridge, 1997]. This definition underlines the most important properties of the software agent – it is a computer program and it works autonomously (like daemon [Burk et al., 1998]) without interaction with human. The full power of agent-oriented software engineering, however, appears only in multi-agent systems [Weiss, 1999]. One agent (either software or hardware) has not enough power to compete with large multipurpose systems, but a set of different agents united together can give us missing functionality. Under the meaning of “has not enough power” can be defined wide range of artificial devices with different CPU/power outputs from “smart-dust” to

personal computers and servers standing along and, of course, pure software agents running in real or virtual environment. Under virtual environment we mean one created by human, i.e. artificial. Internet, for instance, is a good example of virtual environment. There is a specific type of virtual environment created for testing agents – simulation environment.

Very often the autonomous computer program (and hardware it runs) is embedded into a natural system or into some autonomously functioning artifact to enhance their behavior. The main advantage of an agent is ability to communicate with the other agents in order to achieve its goals.

From this general description could be specified the primary properties of agents:

- **Autonomy** – agents can operate without direct intervention of humans or others, and have some control over their actions and internal states;
- **Social ability** – agents interact with other agents (and possibly humans) via some kind of agent-communication language or set of actions;
- **Reactivity** – agents perceive their environment, (which may be the physical world, a human (e.g. via a graphical user interface), a collection of other agents, the cyberspace, or perhaps all of these combined), and respond in a timely fashion to changes that occur in it;
- **Pro-activeness** – agents do not simply act in response to their environment, they are able to exhibit goal-directed behavior by taking the initiative
- **Veracity** – the ability of an agent to choose will it voluntarily communicate false information or not;
- **Mobility** – the ability of an agent to move, or that an agent is residing on a moving platform (that it may control or may not). For cyberspace this means changing the host.
- **Learning/adaptation** – agents can improve their performance and modify their functionality over time.

These are desired properties of an intelligent agent. Not all the agents possess all the properties, but the more of those properties they possess – the better they perform. All depends on the goals and application area the agents are applied to. Some typical application areas of agents are:

- **Military:** monitoring friendly forces and adversary, battlefield surveillance, biological attack detection, targeting, modeling and analyzing systems of systems, etc.
- **Ecological:** fire detection, flood detection, pollution monitoring, etc.

- **Health related:** monitoring physiological data, medical data processing, controlling medical equipment, etc.
- **IT/WWW:** customer tracking, customer profile generation, security, information search, statistics collection, viruses, cyber defense
- **Social applications:** cooperative information exchange, individualized traffic control, microclimate monitoring using personal mobile devices, grocery shopping for consumers, etc.
- **Miscellaneous:** car theft detection, automatic car parking device, inventory control, habitat monitoring, home applications, etc.

2.2 Agent structure and agent types

There are different approaches to build intelligent agents. Similar to every living creature agent lives in some environment, senses it, acts upon the information that it gets from the sensors and possibly has enough intelligence to remember what it has done and what were the consequences so as to correct it's behavior.

As shown in Fig 1, agent may have several inputs and outputs. They can be divided into 2 big groups: inputs/outputs to be used by the agent in real word (e.g. sensors, actuators, etc.) and inputs/outputs to communicate with other agents and entities that understand agent communication language.

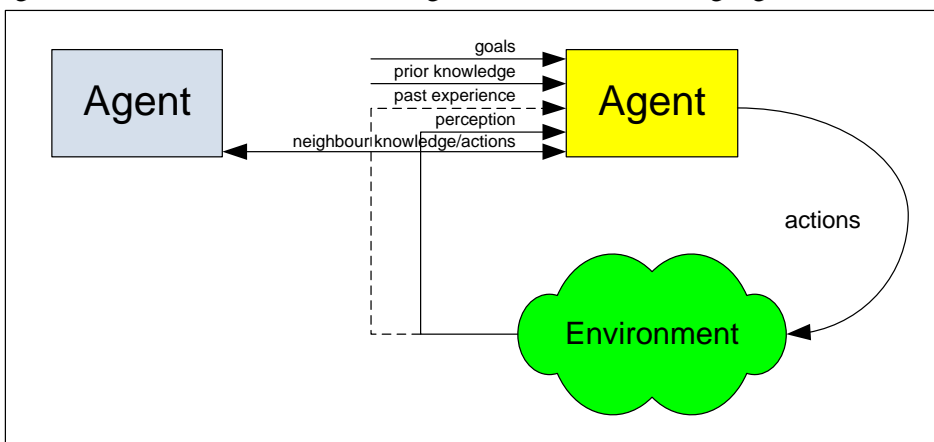


Figure 1. Agent life cycle

Some of the properties mentioned above may be absent. For instance, sensor has no ability to change the environment (pure observer). Usually it is not be able to move on its own, and correspondingly never plans its movements.

The generic agent's life cycle contains six important domains to operate with (Fig 2) [Murray, 2002].

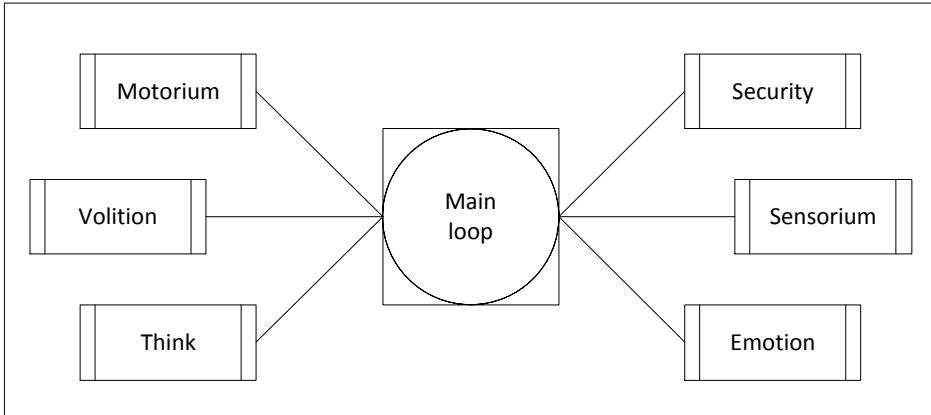


Figure 2. Structure of the agent

The functionality of those domains is:

- **Security** – Ensures that agent by its actions does not harm itself or other parties and does not corrupt mission objectives.
- **Sensorium** – Sensors and other means of collecting input information.
- **Emotion** – Quasi-physiological influence upon reasoning.
- **Think** – Syntax and vocabulary of natural/artificial/agent languages, formulating messages, goals, memory, learning.
- **Volition** – Contemplative or rational selection of motoric options.
- **Motorium** – Robotic activation and implementation of motoric initiatives.

A generic agent has to spend a large share of its resources to motion planning and executing. For our work this is really not important and lay behind our goals. That is why such a big parts of the real agent body as sensors, motors and their calibration are not considered in the simulation. Instead of the physical sensors simulation uses input signals from a computer. The same is done with movements. Agent may travel between different computers instead of moving in a real area. We do not focus on the mechanics here. Instead we focus on agent's communication and behavior. Thus our target agent is simulated as pure software agent.

Other agent architecture is proposed by M. d'Inverno, and M. Luck in [d'Inverno et al., 2001]. This type of agent consists of four main parts: optional sensor(s), information storage(s), controller(s) and actuator(s) (Fig 3). Information flows from the sensor towards memory of the agent or directly to controller which affects actuator. Actions of the agents can be proactive as well when it affects actuator by the intention based on prior knowledge.

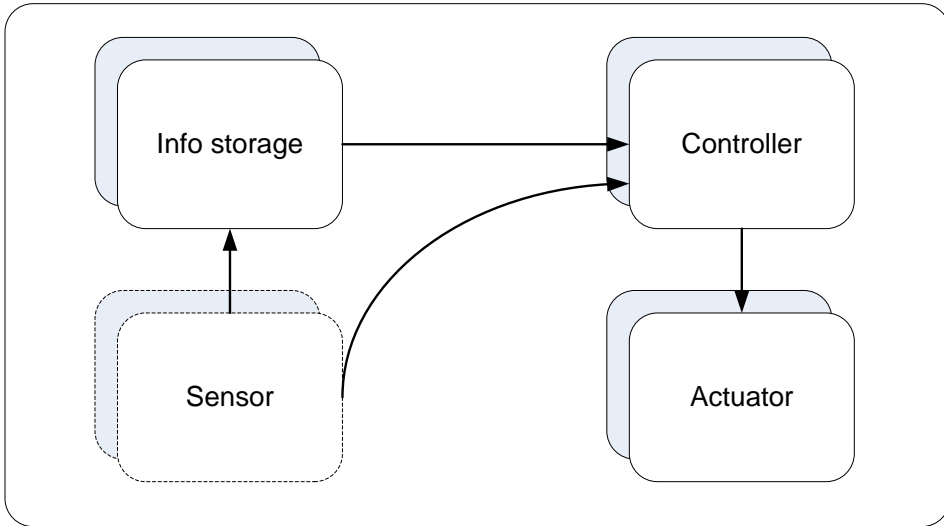


Figure 3. Structure of the SMART agent

By the type of agent's actions they can be divided into three types: reactive (reflex-based), rational and goal-oriented [Russel et al., 2003].

Agents can be very simple as well as very complex entities. Even reflex-based agents can be designed with adaptation mechanisms. For software agents it is very hard to evaluate their behavioral complexity. Assessing agent behaviors one must consider the number of implemented behavior patterns and complexity of each pattern. Correlating the complexity with the size of the agent (program) seems to be inappropriate. Agent with simple behavior may consume large data array whereas agent with complex behavior might use small data sets. They both may occupy the same number of memory cells during operation.

According to FIPA specification agent life-cycle may have several states (Fig 4). The state before agent is created and after agent is destroyed is *Unknown*.

The rest of the states when implemented in JADE are:

- *Initiated*: the Agent is built, but hasn't registered itself yet with the AMS, therefore it has neither name nor address and is not able to communicate with other agents.
- *Active*: the Agent is registered with the AMS, has received legal name and address, and it can access all the various JADE services.
- *Suspended*: the Agent is currently blocked from operating. Its internal thread is suspended and its behavior is not being executed.
- *Waiting*: the Agent is conditionally blocked, waiting for some event to occur. Its internal thread is sleeping on a Java monitor and it will wake up when the condition is met (typically when a message arrives).

- *Transit*: a mobile agent enters this state while it is migrating to a new location. The system continues to buffer messages that will be sent to its new location after it resumes “active” state.

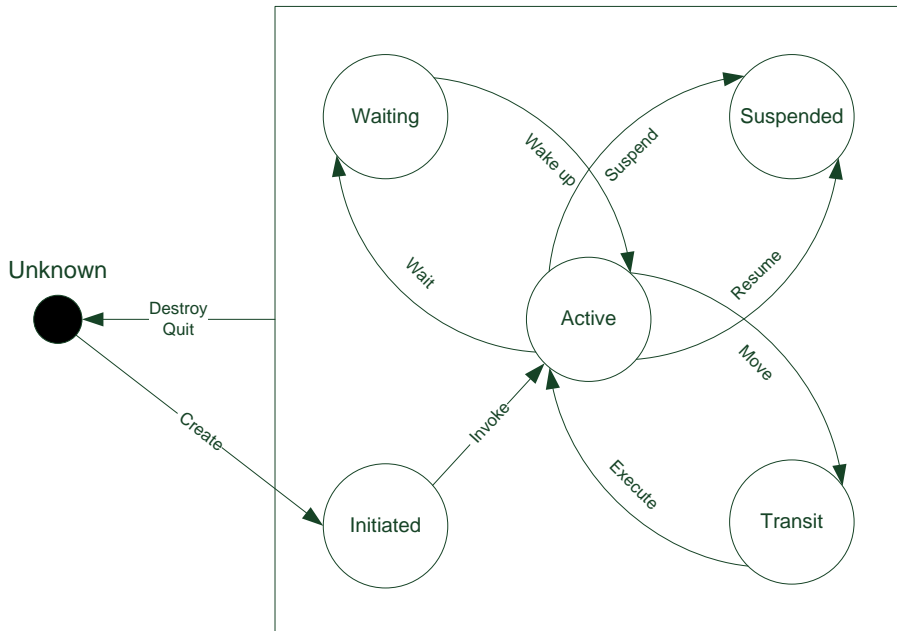


Figure 4. Agent life-cycle as defined by FIPA

2.3 Communication between agents

Communication between software agents is usually done in a specially designed communication language. Most popular languages are based on XML. One of the standard languages is FIPA ACL proposed by Foundation of Intelligent Physical Agents [Odell, 2011]. This language describes the rules of how agent can send queries to another agent and how to construe the answer. The definitions of objects in the universe are given by ontology that the agent uses.

Ontology is a formal representation of a set of concepts and actions within a specific domain and the relationships between those concepts. It is used to reason about the properties of that domain, and may be used to define the domain. Typical mathematical domain will include definitions of mathematical operations, numbers, sets, equation, etc. To make agents understand each other they have to speak the same language and use the same ontology. Ontology is part of agent's knowledge base that describes what kind of things the agent can deal with and how those things are related to each other. Typically the ontology is predefined for a specific domain. It is easier to generate a specific domain with appropriate level of abstraction than to make agents smart enough to learn

new concepts. This approach is promising for designers who construct agents that always execute the same computation but with different data sets. In section 7.6 we will demonstrate that such kind of agent can be created even at current technological level.

Agent that achieves its goals through communication acts with other agents is by all means – a social agent. Socialization is a powerful mechanism that solves tasks by delegating specific tasks to specific types of agents instead of building a universal agent. Dedicating agents to specific tasks makes agent development less demanding.

Social approach, applicable to agents, is more reliable, versatile, agile and distensible than OOP or component-based approach. Validity of this statement is demonstrated by success of service-oriented programming. Services are not yet intelligent but still very powerful entities that deliver simple functionality for wide range of applications over the network. During the last 5 years the number of services based on HTTP protocol (web services) has increased dramatically. This is also stipulated by widely adopted SOA principles [Bell, 2008].

2.4 Information security as a part of social life

The information security cannot be considered separately of social aspects – human being who often is end-user, or information holder, is always vulnerable to different kind of manipulations [Mitnick et al., 2002]. Technically, the existing mathematical algorithms and random number generators allow us to encrypt data at a level of security that forceful decryption has NP-complexity [Schneier, 1996]. Potential attacker circumvents encryption by exploiting either software weaknesses or human's nature [CVE, homepage]. This trend shifts efforts in building information security systems to the weakest link – human-being. Comparatively new problem in computer systems is to determine with whom it interacts – with a human being or with another computer.

CAPTCHA⁴ is one of good examples how to determine human being. It exploits human's ability to work with images and recognize damaged text exposed as picture. CAPTCHA's wide usage that has spread over internet demonstrates that attention is moving from the reliability of software-hardware systems to the most unreliable element – human being. To work efficiently with human computer system must recognize human being and help him to do his tasks efficiently. Such tools as fingerprint identification and voice assistance are already part of our life.

⁴ Completely Automated Public Turing test to tell Computers and Humans Apart

Computer tries to create a personal profile for the human user – determine the way human types in words, analysis pattern of mistakes, preferred time to work, favorite web-sites he visits, etc. This information can be used in security protocols to authenticate the user. Combination of user name and password is not sufficient to identify personality of the user. The system can identify the user by his behavior that it previously has learned from the user. This knowledge can be shared then between the computer systems or agents like is demonstrated in this thesis.

2.5 JADE simulation platform

Most of the simulations in this thesis were done on JADE platform. Platform is written in JAVA and distributed as a set of libraries to be used in agent development, test and execution. General architecture of JADE platform refers to FIPA standard and includes special agents that constitute the core of the system: Agent Management System (AMS) and Directory Facilitator (DF) (Fig 5) [Bellifemine et al., 2010].

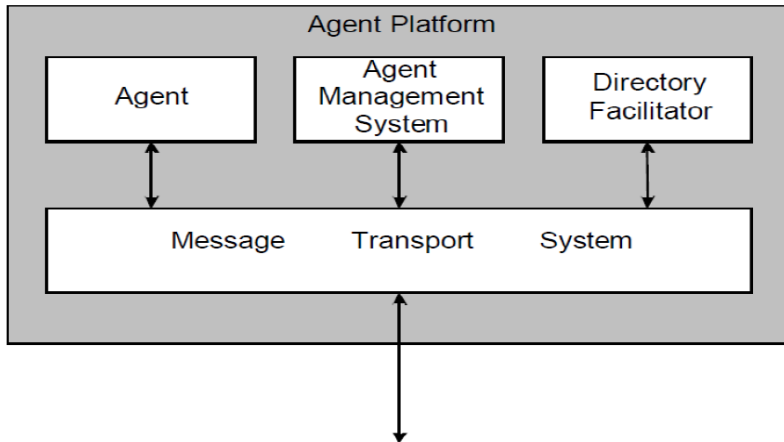


Figure 5. Reference architecture of a FIPA Agent Platform

Platform may be distributed across multiple hosts and JVM-s. There should be one and only one Main container where AMS and DF agents live and the rest of agents could spread over different containers that are connected to the main container (Fig 6).

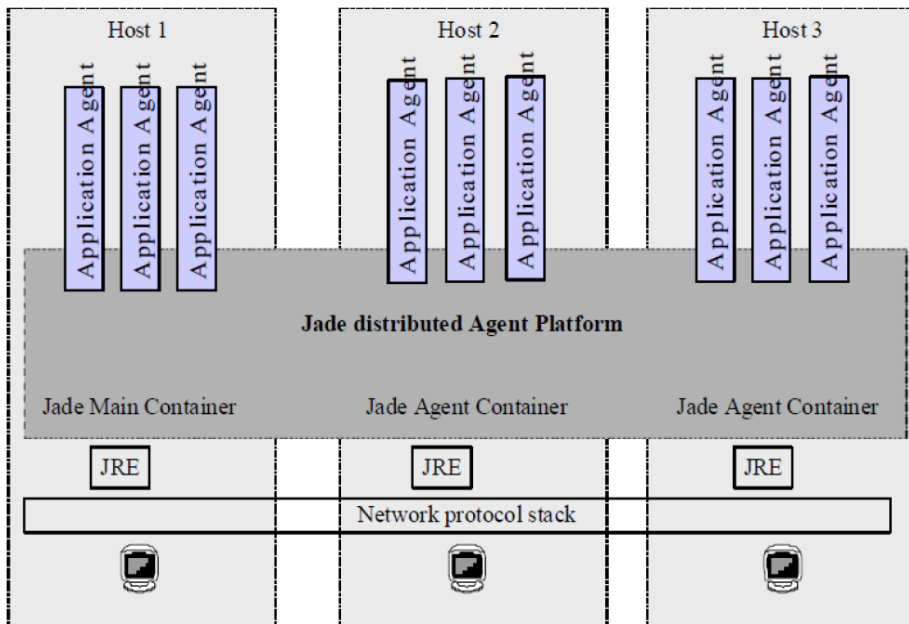


Figure 6. JADE Agent Platform distributed over several containers

There are also special-purpose JADE agents that help to develop and test agents. They are:

- Remote Monitoring Agent (RMA) – allows controlling the life cycle of the agent platform and of all the registered agents. The distributed architecture of JADE allows also remote controlling, where the GUI is used to control the execution of agents and their life cycle from a remote host.
- DummyAgent – allows users to interact with JADE agents in a customary way. The GUI allows composing and sending ACL messages and maintains a list of all ACL messages sent and received.
- Sniffer Agent – visualize messages sent between agents. When the user decides to sniff an agent or a group of agents, every message directed to/from that agent/agent group is tracked and displayed in the Sniffer Agent's GUI.
- Introspector Agent – monitors and controls the life-cycle of a running agent and its exchanged messages, both the queue of sent and received messages. It allows also monitoring the queue of behaviors, including executing them step-by-step

User agents are obliged to interact with AMS and DF in order to be tracked by special agents and in order to utilize all platform features. Agents must implement one or more behaviors from the patters below:

- SimpleBehaviour – models generic atomic behavior.

- OneShotBehaviour – models atomic behaviors that must be executed only once and cannot be blocked.
- CyclicBehaviour – models atomic behaviors that must be executed forever.
 - CompositeBehaviour – this abstract behavior that are made up by composing a number of other behaviors (children). This behavior must be explicitly defined by sequential, parallel of FSM behavior.
 - SequentialBehaviour – this is a CompositeBehaviour that executes its sub-behaviors sequentially and terminates when all sub-behaviors are done.
 - ParallelBehaviour – this is a CompositeBehaviour that executes its sub-behaviors concurrently and terminates when a particular condition on its sub-behaviors is met.
 - FSMBehaviour – this is a CompositeBehaviour that executes its children according to a Finite State Machine defined by the user. Each child represents the activity to be performed within a state of the FSM and the user can define the transitions between the states of the FSM.
 - WakerBehaviour – one-shot task that must be executed once after given timeout is elapsed.
 - TickerBehaviour – cyclic task that is executed periodically.

3 Ant colony simulation

This project has started when I accidentally got the book [Bonabeau et al., 1999] that describes the behavior of real ants and shows some practical applications based on the research of their life. This conception was to build up the system of intelligent agents (IA). From one side it supposes to simulate the behavior of real ants and from the other it describes the process of building software-based agents. Ant colony simulation suits for such a task almost perfectly.

Project implements the model based on ant colony structure. Originally the idea of ant colony optimization project is to use “pheromone's trails” representation to find the solutions for some practical tasks such as routing, scheduling, travel salesman problem (TSP), etc. In this project “trails” as defined in the book are not used at all. Instead of the “trails” agent ability to exchange messages directly is used. I have developed special “ant” language that is represented by ontology.

My main area of interest is IA structure, different types of behaviors and collaborations between agents with an ability to create social structures (coalitions) according to the agent’s goals and beliefs. System uses the notion of “time” [Mõtus, 2003] to be able to carry out all its functions. Thus more complex model of inter-agent communication is used instead of “pheromone trails” and also message security and real time issues have being taking into account.

The system implements some mathematical algorithms for the particular problem solving (like finding the shortest path between two points) or even several at the same time for different agents/coalitions, but we will not be focused on describing each of them in details, because they are well-known. It is more interesting for me to obtain results using different types of interactions between agents, implementing different types of agents, use coalitions and time constraints.

3.1 Model structure

Here I define the internal structure of the whole system that implements the desired functionality. UML approach has being used to build the system from different top-level views. This diagram is analogue to Use Case Diagram in UML notation [Booch et al., 1998] [Soley, 2003], where the users of the system are agents themselves. There is a better approach now to represent the goal hierarchy developed recently [Sterling et al., 2009], but at the time system was designed this approach did not exist. The goal hierarchy looks like on the Fig 7.

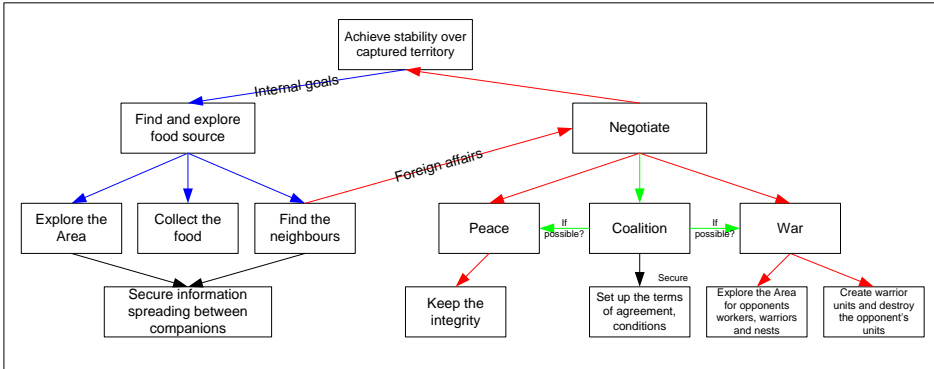


Figure 7. Goal hierarchy

The goal hierarchy tree has two main branches – “internal goals” (blue) and “foreign affairs” (red). Agent’s behavior entirely depends on internal B-D-I (beliefs, desires, and intentions) logic [Haddadi et al., 1996]. In complex situations, besides agent’s BDI logic, agent may use more sophisticated structures like different RRS (representation and reasoning systems) [Poole et al., 1998]. The behavior of the agent also depends on which class agent belongs to. We will consider several of them later on.

Top-level goal is to find the balance in the system with different (sometimes opposite) agent’s desires or within population (growth, death, food income, etc.) over certain period of time.

In a case of one population there is no need of warriors and system goal is to find the food source and bring food to the nest. The tasks degenerate into finding the shortest path between two points and avoid collisions. Generally this is a goal of ACO project.

In a case of several populations much more problems arise. It depends on the social life of such a structure. System top-level goal comes from this kind of structure, but in the beginning I am going to implement the simplest one – without social impact.

3.1.1 Internal goals

These goals determine what an agent should do for himself, then for his relatives and social groups he belongs to. In our particular case the Ant belongs to some Nest and all the ants that belong to the same nest organize the “family” (or social group with non-contradictory goals). Within this group it may be the other groups (subgroups) that are created by dividing the functions between agents, i.e. workers, warriors, etc. They work together to achieve common goal.

Regarding Ant colony simulation the main tasks for the units are to explore the area, collect the food and find the neighbors (if any). All information about

the food sources and foreign affairs should be secured inside trusted groups within the colony or coalition.

3.1.2 Foreign affairs

All relations to ants that do not belong to the same nest are considered as foreign inter communications. The most important part of it is Negotiation. The result of the negotiation process is the decision about war, peace or coalition in one of the area. Here I define peace as co-existence of two or more parties whose interests are not in conflict. The war in this case is the conflict between two or more parties that cannot be solved by coalition. Good example of such situation is struggle for the shared resource. Coalition has a priority over other two choices, because it brings mutual benefits for the participants either in war or peace.

There are, certainly, more different types of possible interactions like master/slave situation or long term cooperation that can bring no benefit (even losses) in the beginning, but may have good output in the future. I do not consider them here for a while in order not to make the system too complex to build and analyze. This social part of the system remains open for future investigation.

3.2 Agent approach

Basically the model consists of four types of agents: environment agents, nests, workers and warrior agents and may be easily extended in the future by adding other types of the agents (Food Agents, Obstacle Agents, etc.).

Here I define what abilities and actions each agent does:

Area Agent – represents the area where each other agent acts. This is a matrix of cells. Each agent occupies one cell with its own coordinates on this area and some of them can change their position by moving to another empty neighbor cell. Cell could contain food source, obstacle, another agent or empty. This information keeps and manages the Area Agent. It belongs to environment type of the agents.

Nest Agent – represent the Nest for one ant colony. It is placed randomly on the map and then produces Worker agents and Warrior agents to explore the Area and find the food sources and neighbors. It has unchangeable coordinates in the Area. Nest is characterized by number of issued workers and warriors and by amount of collected food.

Worker Agent – searches through the Area for the food and then transfers the food from the food-source to its Nest agent. It modifies the Area and changes the

information in Nest agent. The algorithm of food search and orientation is not strictly defined and may be specified by external module.

Warrior Agent – searches through the Area for the other agents. Identifies the opponent whether it belongs to the same Nest or not. For the friendly agents warrior can be the messenger. For the unknown agents it determines the politics of foreign affairs. It executes War and Peace strategies. Simple behavior is to capture (kill) all unfriendly agents without analyzing consequences and conditions.

3.3 Distributed model of mobile agents

From the beginning there was a demand to the system to be distributed over the network. The system uses client-server model. Server runs the main-container (JADE term) and *Area Agent* that generates and maintains the area map. It also has HTTP-server to be able to serve requests from the clients. It contains client applet. The structure is shown on Fig 8.

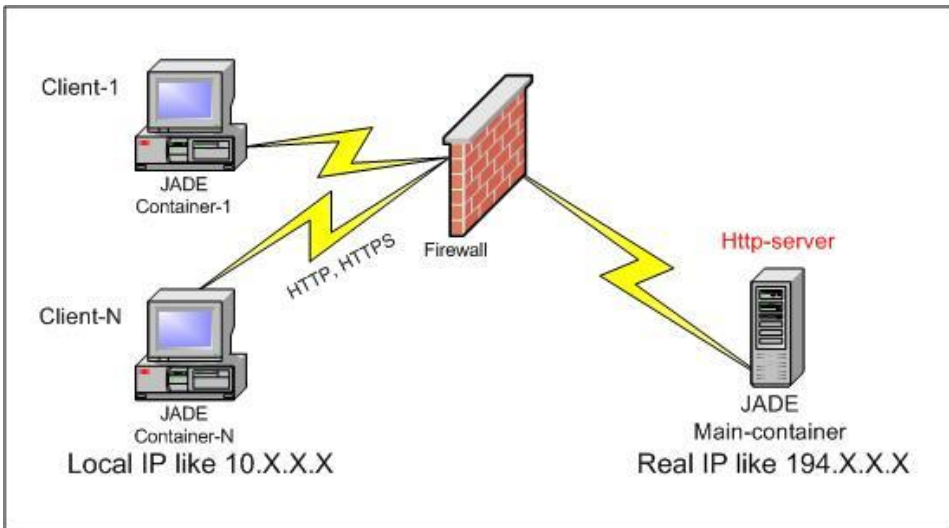


Figure 8. Distributed JADE platform

Any agent (i.e. nest, worker of warrior) can physically move from one container to another and live there without being connected to the origin. This resembles virus behavior, but this is not. The function of moving between platforms is supported by JADE.

One of the main demands is that client doesn't have to have JADE installation on the local PC and also may not have real IP address. The only required software for the client is JAVA virtual machine and web-browser.

Client initiates the connection to the server through HTTP (HTTPS) request, downloads the applet with JADE libraries and creates JADE container on the

local PC. All the agents run within a container and communicate with the other agents using JADE main-container. Agents have to implement special type of map-queries to be able to explore the map more efficiently. The diagram is shown on Fig 9.

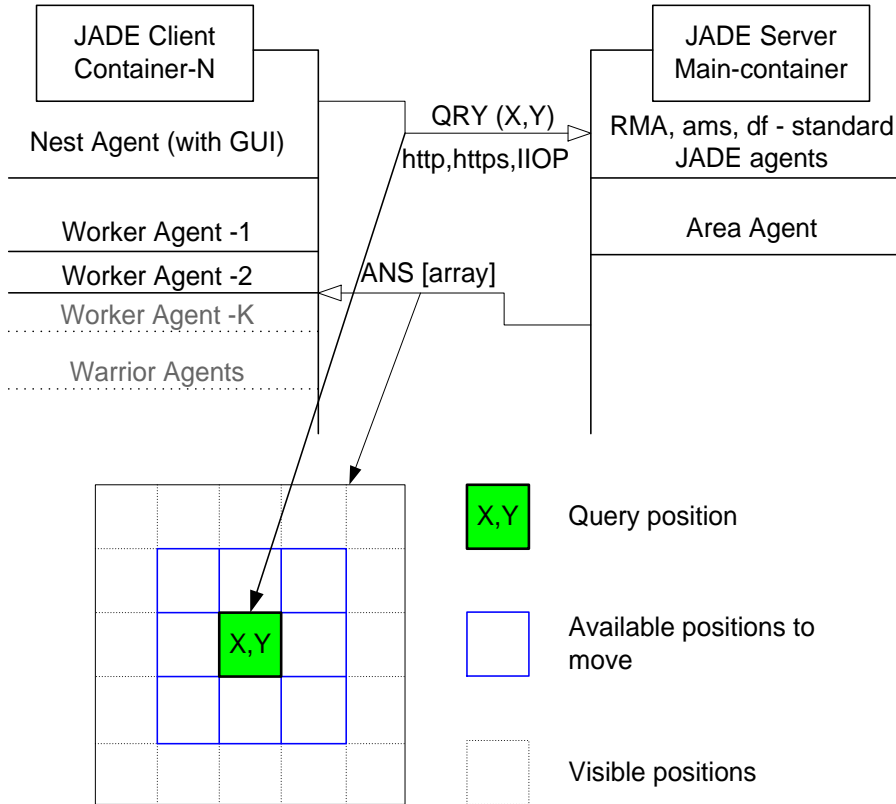


Figure 9. Map navigation messages

Every agent queries the *Area Agent* with (X, Y)-coordinates and has an answer in reply as an array of Cells in a visible range. This range is different for different type of agents and may change during agent's life (experience). Agents that have ability to move between cells can move only in a neighbor cell, despite his visible range may be bigger. The *Area Agent* manages all these movements as well.

3.4 Ontology

Each agent, involved in the project, communicates with the other agents using FIPA ACL. Rather FIPA-SL [FIPA, SL] content language has being used because it is supported by Protégé and JADE platforms. To be able to communicate agents should define/use the ontology [Cranefield et al., 2001], i.e. domain of terms and meanings agents operate with.

The ontology for “Ants” project was generated using Protégé-2000 development tool. It contains 7 concepts and 5 predicates. We use special add-on, called “bean generator”⁵, to be able to generate the ontology for JADE platform [Noy et al., 2001] automatically. Short description of used notions is given in (Table 1).

Table 1. Ontology components

Type	Subtype	Slots	Data type	Description
Concept	AID	Name Addresses Resolvers	String String, ∞ Class, ∞	Agent name Address AID
Concept	AreaAgent	maxX maxY	Integer Integer	Size (X coord.) Size (Y coord.)
Concept	NestAgent	Vrange NestCapacity Neighbors NestWorkers NestWarriors Cell	Integer Integer String, ∞ Integer Integer Class	visible range food amount neighbour agents number of workers number of warriors occupied cell
Concept	WorkerAgent	Vrange Neighbours Prev_cell Cell	Integer String, ∞ Class Class	Visible range Neighbour agents Previous location Current location
Concept	WarriorAgent	N/A	N/A	N/A
Concept	FoodAgent	N/A	N/A	N/A
Concept	ObstacleAgent	N/A	N/A	N/A
Concept	Cell	coordX coordY cellOWNER cellSTATUS	Integer Integer Class Integer	X coordinate Y coordinate Cell content Status (0 - empty)
Predicate	Consists	area cell	Class Class	Area Agent Cell
Predicate	AreaStatus	area	Class	Area Agent
Predicate	WorkerStatus	worker	Class	Worker Agent
Predicate	NestStatus	nest	Class	Nest Agent
Predicate	isSituaded	location worker	Instance Class	Cell Worker Agent

The ontology is not universal enough to be able to communicate with the agents of other type. Theoretically it is possible to teach agents other ontology like Finnish SUO [Henriksson et al., 2008]. Practically within this project there is no need of external ontologies. For this project I have developed stand-alone

⁵ In the latest releases on JADE (since version 3.6.1) there is no need to use external tools to generate ontologies. They can be developed using BeanOntology class.

ontology that, from one side, separates our agents from the other world, but from the other side it is easier to start the project with custom ontology and not spend much time adopting the external one.

3.5 Implementation in JADE agent development environment

Ant project is a template to investigate interactions between agents and potential of JADE platform. Currently project doesn't include Warrior Agents and simulates the behavior of one isolated ant colony. It can be easily modified to more complex research.

Main features of the Ant project are:

- The system has been entirely written in JAVA so it is platform independent.
- JADE platform support distributed architecture and software implements conceptions of parallel programming and (possibly) real time.
- JADE libraries are included in the client software.
- Configuration of the system is concentrated on the server. Clients are configured automatically.

3.5.1 Running system

Area agent – generates the map as matrix $X*Y$ where X and Y parameters may be passed to him during start-up. The default values – $100*100$. The area map is two-dimensional. Each cell is an object that contains 4 fields (see Table 1). Each cell has a status field - an integer number that shows cell's status (empty, occupied or food).

The map is stored in a memory and text file. Text file contains only the initial map (for a debugging) and map changes during system run can be seen on the Area Agent Graphical User Interface (GUI).

Agent has 1 behavior: it listens for a message which in FIPA specification called “query-ref” with coordinates “(X, Y)” in its body. Agent replies with “inform-ref” message that contains an array of neighbor cells. Thus agent who queries the cell will know whether the cell is empty, obstacle or contains the food and what is situated in neighbor cells within his visible range.

Nest agent – searches for the *Area Agent* and gets its location over the territory. It happens during the conversations between the *Nest Agent* and the *Area Agent*. Coordinates of the nest location are assigned randomly. Typically Nest is placed in the initialization phase and its coordinates does not change over time. So there is no possibility for the ant colony to change the place of the nest or to set up new Nest in addition to the initial one.

After Nest is placed on the map it generates workers. Workers begin to live independently from the Nest, trying to achieve their goals.

Nest Agent software is downloaded from the server as JAVA applet and runs on the client computer if server is accessed over HTTP. Nest has its own GUI – copy of the area map with cells, visible by the agent, and all generated workers.

Worker agent – is placed on the Area by the Nest and starts searching for the food source. When the food source is located Worker gets one piece of food and returns to the Nest. The Agent can be in one of the 3 states: searching for the food, getting food to the nest or returning back to the food source for the next portion.


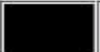





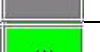


The movements are displayed on the Nest GUI.

3.5.2 Examples of GUI

Area map graphical representation is common for both client and server. The only difference is that server has this map completely open, but client can see only the part within certain range. See the examples below.

Each cell on a map may show different pictures. They are shown in Table 2.

Table 2. GUI cell icons description

Picture in applet	Picture at server	Description
	the same	Food source (with value)
	the same	Obstacle
	the same	Unexplored territory
	the same	Empty cell
	the same	Nest
		Worker searches the food
		Worker transfers the food to the NEST
N/A		Worker returns to the food source for the next “portion”

Interactions between agents (i.e. messages and their context) user can trace using JADE build-in agents: Sniffer and Dummy Agent (only for main-container). Additionally they can be traced on the console (java console for the applet). Example of server GUI (Area agent) with the area 20x15 cells is shown on (Fig 10).

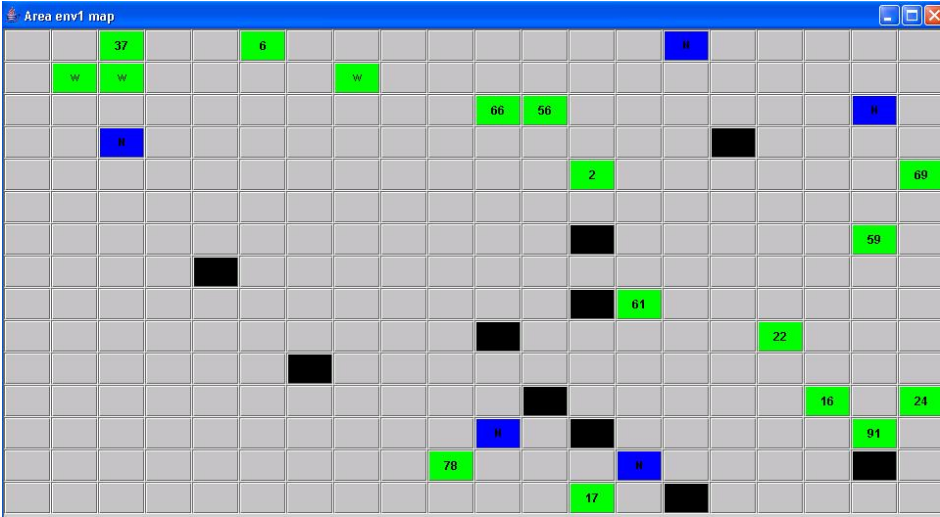


Figure 10. Area agent GUI example1

Example of client's applet GUI for the same area (visible range = 1 cell) is shown on (Fig 11).

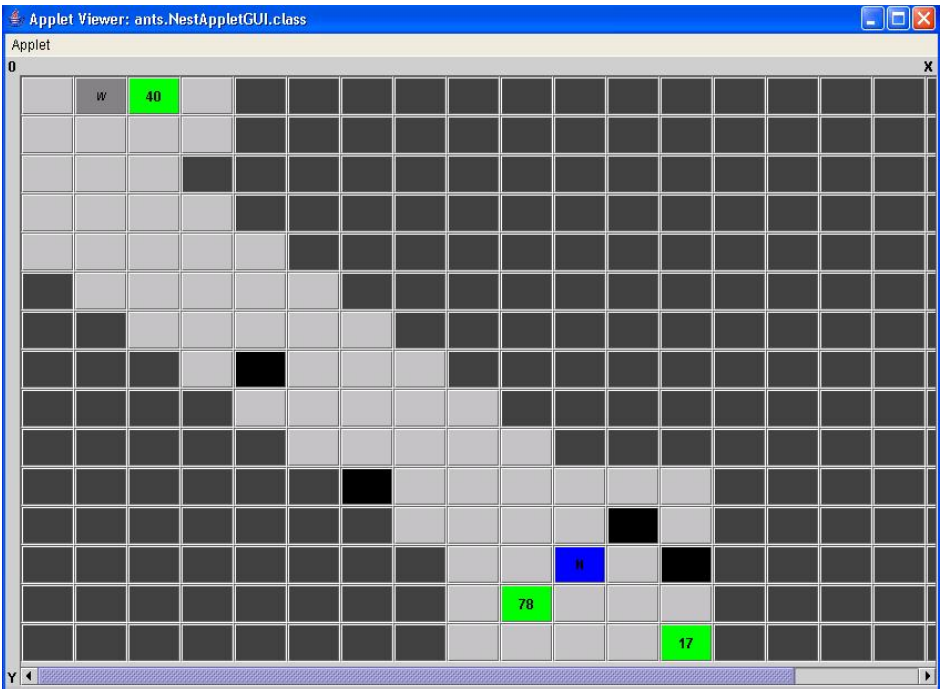


Figure 11. Nest GUI example

3.6 Conclusion

I have studied basic principles of building multi agent systems and tried JADE framework to program software agents. Distributed MAS with more than 50 agents has being created as a result. Each agent has being started either by local JVM thread or by remote container in JSP. Main JADE container was launched from JSP within Apache Tomcat server [Apache, 2012] and locally.

JADE platform may be used by universities to study MAS development and train students on agent programming. Nevertheless JADE has potential to be used as production system for many application areas where real time constraint is not strict.

Strong side of the platform is ability to use ontologies in messages interactions. Despite the ontology for this particular project is not optimal I get the way to make it better. In the latest versions of the JADE ontology support has been drastically improved.

4 Optimize shared resources. 5 hungry philosophers problem

The management of shared resources in the systems sensitive to time is tried to be analyzed in multi-agent environment – JADE. As a model for that “dining philosophers” problem (firstly formulated by Dijkstra) [Hoare, 1985] [Mötus, 1990] was taken to be investigated. This project shows how this model can be represented using agent conception and methods of Extreme Programming [Knublauch, 2002]. The latter I used to simulate modeling in rapidly changed environment while designing the system from “classical” definition to extended one. Intermediate steps are not presented here for the sake of conciseness.

The main goal is to build up the system to be capable to investigate the behavior of the agents under different conditions. Since the environment is represented also by the agent it is possible to model not only static but also highly dynamic environment. Basically agents are not capable to learn and don’t have the memory of past events, but the system can be extended to do that in the future.

Classical description of the “dining philosophers” problem assumes that there are five philosophers who work and feel hungry from time to time. When a philosopher feels hungry he goes to the table where there are 5 plates with rice and only one stick between neighbor plates. One of the possible (worst) states of such system is achieved when all 5 philosophers go to the table simultaneously, take one stick each to eat rise and no one can eat (we also assume that to be able to eat rise person needs 2 sticks).

There are many goals which could be achieved by solving this problem. How to increase time the philosophers are working? How to avoid the worst situation (when all philosophers die)? How to keep satisfaction of the philosopher at the maximum level? And so on.

More parameters can be added to the system in order to simulate some real object. For instance, the level of unconsciousness can be added to the philosopher to represent the state when he is unable to go for the lunch by himself and external help (like hospital) is needed. Some examples of more sophisticated formulations of the problem can be found in [Chandy et al., 1988], and [Mötus et al., 1994].

4.1 Process modeling

As there is no need to be strictly followed by classical description, I have slightly modified the task and refused from “sticks”. Instead there is a table with

maximum 3 simultaneously available free places for 5 philosophers. The model of system with additional parameters is shown on Fig 12.

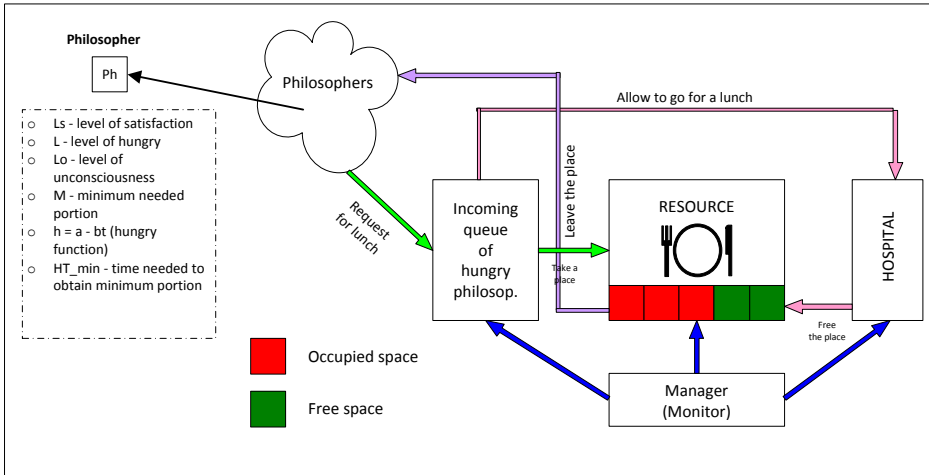


Figure 12. Functional diagram of the system

4.2 Agent approach and ontology

All the actions could be divided between four types of agents.

Philosopher:

- thinks and works
- keeps account of the consumed food (in a fixed period)
- becomes hungry (according to individual schedule) and applies for the food
- falls ill and applies for hospital, unless he regularly receives sufficient quantity of food in due time

Manager:

- monitors the status of philosophers (and other involved agents)
- attempts to influence the decisions of the table
- defines the functioning goal of the system – e.g. maximizes philosophers working time, or minimizes hospital expenses, or minimizes the consumed food, etc.

Table:

- controls the use of resources (simultaneously 3 of 5 is maximum usage)
- assigns the quantity of food to a customer
- selects customers from the queue according to its own rules

Hospital:

- cures philosophers suffering from being hungry too long

- takes care of the philosophers, gets extra food in a priority queue for them
- if and when a philosopher's health has improved, sends them back to work

Different types of agents carry the role for the different activities in the system, however basically the functions of the Hospital and Table were implemented in the Manager agent.

This is done because all the interactions between the Philosopher and other agents are done through the Manager. And for the Philosopher there is no difference who issues the command because all of them are going through manager.

The decision to refuse from 2 types of agents simplifies the process of design and allows obtaining the results about system behavior more quickly.

4.3 Implementation in JADE

4.3.1 System

The source code was written in JAVA using custom developed ontology which is also a part of the project. JADE environment was used to execute agents and monitor their states. GUI interface of the Managers shows the state of a particular agent with the precision depending from Agent local time period (see section „Time constraints Time constraints“).

The example of manager agent GUI is shown on Fig 13.

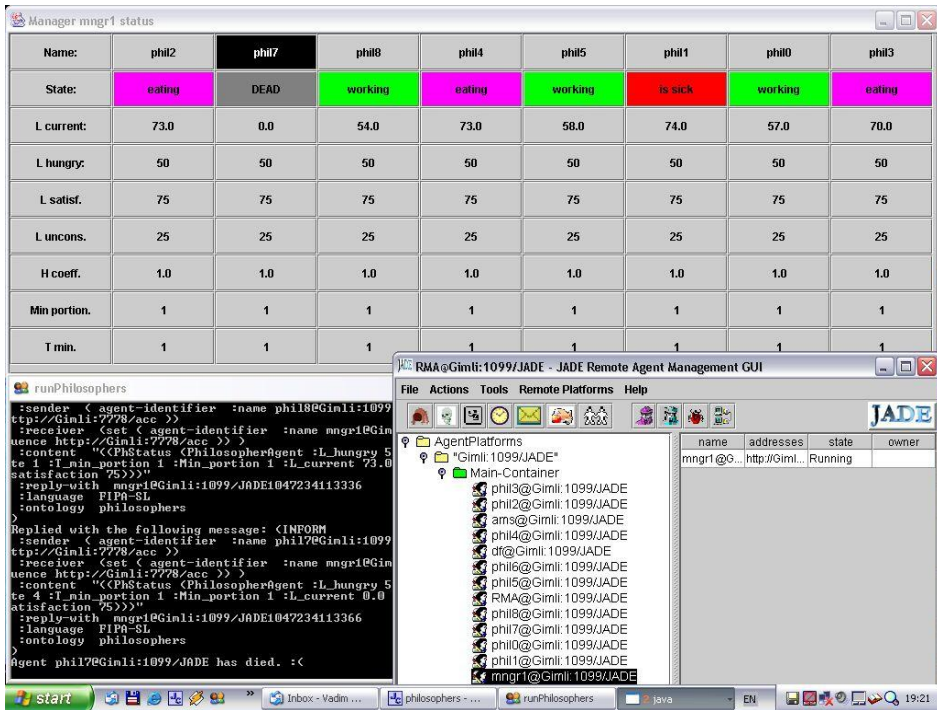


Figure 13. Screenshot of a working system

There are 3 windows on the picture:

- Manager status GUI
- JADE console
- JADE Remote Agent Management GUI

While “Manager status GUI” is only informational from JADE console output information can be saved to the file. JADE RMA GUI is used to control agents, i.e. add/delete, pause/continue, migrate, sniff, and send messages. Parameters to the philosophers are passed through command prompt. If they are not mentioned or their order or number is incorrect agent uses default settings (see Table 4 for details).

4.3.2 Agents

All the agent’s functions in this project depend from time linearly though there is no constraints do define the behavior as a non-linear time function.

4.3.2.1 Manager

Manager agent is really the core of the system. It receives the agent’s requests and sends additional messages in order to “learn” the status of each agent in the system. Its main characteristics are $t_{manager}$ (see Time constraints

for how to calculate the maximum value of this parameter) and strategy to implement. Strategy represents the goal for the system. It is the function:

$$S(a_{i=1..k}^n, n, t),$$

which should be maximized (minimized) according to agent parameter(s) a_i (the number of which can be up to k – maximum number of parameters), during time t over n agents. In our case the function:

$$S(a_{state=eating}^5, 5, t) \rightarrow max,$$

In other words all our 5 philosophers should be able to work (without dying case) and eat up to their maximum needs.

4.3.2.2 Philosopher

Philosopher is an active element of the system. It “lives” according to its internal time, works and feels sometimes hungry (self-control). There is no measure on the quality of his work since the productivity of philosophers is of no interest in our current project. It may be done in the future.

Main parameters are represented in Table 3.

Table 3. Default parameters

Parameter	Description	Default value
t_{phil}	Minimum time interval which philosopher can sense	1
hunger function	$L_{current} = L_{current} - H_{coefficient} * t_{phil}$	
$L_{current}$	Current level of hungriness (amount of food, energy philosopher has at the time being).	100
$H_{coefficient}$	Hungriness coefficient. Determines how fast the philosopher becomes hungry. $H_{coefficient} > 0$	1
L_{hungry}	Limit of hunger. When this level is reached philosopher feels hungry and sends request to eat.	50
$L_{satisfaction}$	Satisfaction level. The minimum level that enables working of a philosopher after eating.	75
$L_{unconsciousness}$	Level after which a philosopher can't take meal by himself. Cure is needed. He submits request to be taken to the hospital.	25
$Min_{portion}$	Amount of food which philosopher consumes during $T_{min.portion}$	1
$T_{min.portion}$	Minimum amount of time to get $Min_{portion}$. Measures in philosopher's time units t_{phil} .	1
State	Can have one from 4 values: 1-“working”, 2-“eating”, 3-“sick”, 4-“dead”	1

4.3.2.3 Hospital

Not yet implemented as an agent. His function was divided between Manager and Philosopher by representing “healing action” as a linear function of time:

$$L_{current} = L_{current} + h * t_{phil}, h > 0 \text{ where } h - \text{healing coefficient.}$$

4.3.2.4 Table

Not yet implemented as an agent. His function was divided between Manager and Philosopher by representing “eating action” as a linear function of time:

$$L_{current} = L_{current} + e * t_{phil}, e > h > 0 \text{ where } e - \text{eating coefficient}$$

4.3.3 Ontology

The ontology for this project was created and generated using Protégé-2000 development tool with special bean generator plug-in which creates the structure compatible to JADE.

The ontology consists of 7 predicates and 4 agent AIDs. They are specially designed for this project only thus can hardly be reused (see Table 4).

Table 4. Philosophers ontology

Subtype	Slots	Data type	Description
AID	resolvers	Class, ∞	AID in JADE
PhilosopherAgent	resolvers hungry_coefficient L_hungry L_current Min_portion State T_min_portion L_unconsciousness L_satisfaction	Class Float Integer Float Integer Integer Integer Integer Integer	Subclass of AID Hungry coefficient Hungry level Level of satisfaction Minimum portion Agent current state Time to get portion Unconsciousness level Level of satisfaction
TableAgent	resolvers capacity free_places table_queue	Class Integer Integer Class, ∞	Subclass of AID Qty. of available places Qty. of free places Queue of philosophers
ManagerAgent	resolvers N_philosophers strategy current_agent	Class Integer String Class	Subclass of AID Number of philosophers Goal of the system Subclass of AID
HospitalAgent	resolvers disease_level hospital_queue	Class Integer Class, ∞	Subclass of AID Priority of agent Queue of agents

	sick_num	Integer	Sick philosophers qty.
PhStatus	philosopher	Class	Philosopher agent
TableStatus	table	Class	Table agent
HospitalStatus	hospital	Class	Hospital agent
Go4Lunch	philosopher	Class	Task for philosopher
Go2Work	philosopher	Class	Task for philosopher
SetState	state	Integer	Set state task
Go2Hospital	philosopher	Class	Task for philosopher

4.3.4 Executive Environment

- JADE 2.61 (for the 3.0b small changes and recompilation is needed)
- JRE 1.3.1
- For the processor < PII-300 and RAM<64Mb recompilation with appropriate timing parameters may be needed (see section 5.1)
 - OS with GUI (tested on Windows 2000/XP and Linux Mandrake 8.2+KDE)

Since each agent synchronizes its actions with RTC (real-time clock) the whole system is very sensitive to the working environment. Thus the performance on the slowest computer can even be better than on the fastest one depending from which services and programs are running simultaneously. This can be stabilized by using real-time OS (like QNX) together with conceptions of real-time programming in JAVA [Bollella et al., 2000].

4.4 Results obtained from simulation in JADE

4.4.1 Time constraints

Before assigning any specific value to the characteristics of a philosopher (like hungriness level, level of satisfaction and so on) the time notion for each agent should be defined because the system is time-sensitive see, for instance Mōtus in [Selic et al., 2003]. It should be done in conjunction with the other agents, especially Manager. Manager's internal time interval value is calculated by formula:

$$t_{max.manager} < \left(\frac{N_{phil}}{t_{phil}} + \frac{N_{hosp}}{t_{hosp}} * \frac{N_{tbl}}{t_{tbl}} + C \right),$$

where t_{phil} – time interval for the philosopher, N_{phil} – number of philosophers, t_{hosp} – time interval for the hospital, N_{hosp} – number of hospitals, t_{tbl} – time interval for the table, N_{tbl} – number of tables, C – constant which depends on the executive environment⁶ (the speed of the computer, operating system,

⁶ This constant helps Manager to calculate maximum time interval for every philosopher easily, but it is certainly not – the execution environment which consists of

version of JRE). For fast computers and agents number <10 it can be assigned to “0”.

The above formula is valid in a case there is only 1 manager to control the other agents. If there are at least 2, the task becomes more complicated. System is facing with problems of parallel execution, which demands from the software to focus on synchronization between managers.

4.4.2 Homogeneous systems

Homogeneous linear system in this case is the system where all agent instances (all philosophers) have the same linear parameters and start their execution at the same time (from the batch during the system start up)⁷.

Theoretical results:

In this type of system when agents start working with the same initial parameters they feel hungry also simultaneously. So, the peak of requests starts at the time $t_{peak} = \frac{L_{initial} - L_{hungry}}{t_{phil}}$ and if during $t_{critical} = \frac{L_{hungry} - L_{unconsciousness}}{t_{phil}}$ agent do not receive invitation to the table he becomes “sick”. And Hospital agent can restore his ability to work. However there is also a situation when hospital has no free spaces (if we limit the capacity of hospital) to serve the sick philosopher and after $t_{lifecritical} = \frac{L_{unconsciousness}}{t_{phil}}$ it will die.

Practical results:

Practical results with default initial values are shown on Fig 14.

SW and HW components cannot be constant. Environment parameters (like amount of free RAM, swap size, frequency, etc.) can change eventually.

⁷ There is no possibility to launch the agents “at the same time”, because JADE launches agents sequentially one after the other. When there is $t_{phil} > t_{batch}$ we can say that agents are executed at the same time. For our task this level of approximation is acceptable.

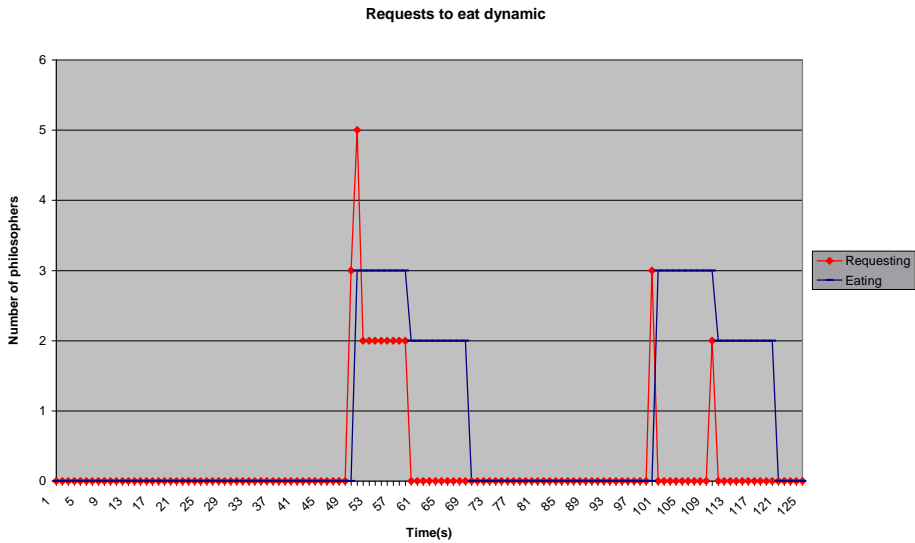


Figure 14. Requests for eating over time

Homogeneous non-linear system in this case is the system where all agent instances (all philosophers) have the same parameters and start their execution at the same time (from the batch during the system start up). At least one parameter of the agent is non-linear.

4.4.3 Heterogeneous systems

Heterogeneous linear system in this case is the system where at least 2 agents are different from each other by parameters. All their parameters are linear.

Heterogeneous non-linear system in this case is the system where at least 2 agents are different from each other by parameters. At least one parameter of the agent is non-linear.

Those two types of systems I did not simulate, because they just have another mathematical function behind agent parameters and all the agents, their interactions and environment remains the same. The goal of the project did not have the requirement to get output for all types of the systems, but to build the MAS that anyone can adjust to their particular needs.

5 Commercial off-the-shelf product improvements

Large companies use a number of different software systems to support and/or automate their work. Usually these software systems include customer/partner management software surrounded by various service applications and databases that help to keep finances, personnel, resources, etc. These applications are often implemented on different platforms, created at different time by using a wide range of technologies. In this situation a very important task is to create homogeneous IS structure for the entire company – so as to use all those systems at their maximum performance, and to have clear control rules within the company. For many companies this problem is weighed down by rapid market changes requiring to integrate new technologies fast, and tendency to keep their own system open for integration of new (and changing) partners/clients.

5.1 SOA structure and main principles

SOA (Service Oriented Architecture) as a method of integrating different applications is not new. Many books and articles are written about SOA [Erl, 2005]. The related theory promises IT-professionals many benefits stemming from SOA. The paper will not discuss the details of different implementation patterns, nor will give definitions to SOA terms, but rather focus on the implementation pitfalls and general weaknesses that we have found after some practical experiments within our telecommunication company. These drawbacks forced us to look for a way to improve SOA. The core components of SOA are Enterprise Service Bus (ESB) and BPEL (Business Process Execution Language). ESB is a pattern of middle ware that unifies and connects services, applications and resources within a business [Chappell, 2004]. BPEL is a language for the formal specification of business processes and business interaction protocols. It is platform-independent and based on XML [Khodabakchian et al., 2011]. We use BPEL to orchestrate services within our company [Schittko, 2003]. BPEL is connected to ESB and is a part of entire system. While ESB performs synchronously, BPEL by its logic allows both synchronous and asynchronous operations. This creates a situation when not all service calls produce response message for the customer. Process may wait until certain criteria is met and then call back customer. It is important for our intelligent agents that will check service availability.

I have created a model similar to classical ESB structure by slightly modifying it by adding one extra layer – “Network names management” (see Fig 15).

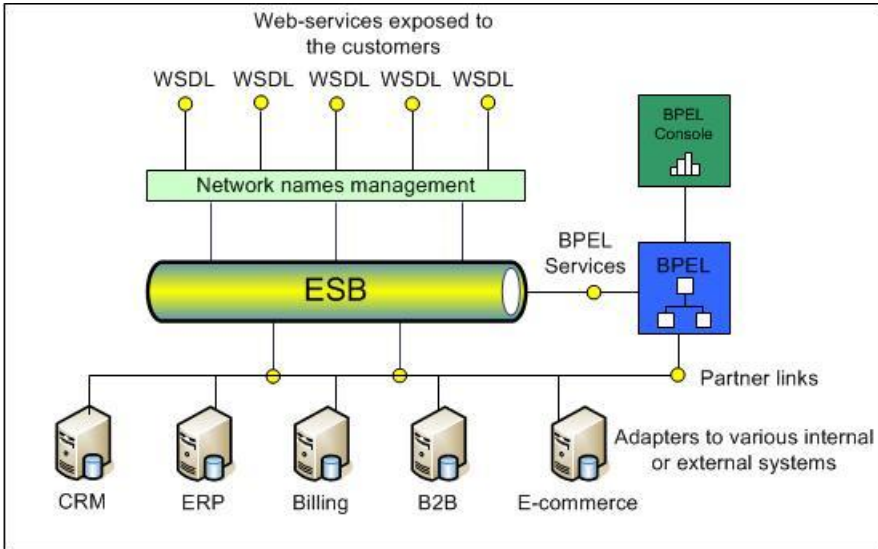


Figure 15. ESB structure

Network names management layer has been introduced in order to manage web-service external addresses in the unified way and separate end-point location from its realization. Both external and internal customers call the same WS address endpoint though internally address physically can be mapped to different systems (production or development) and different versions of the service. Security is also brought to a higher level by having external customers that are working over HTTPS, while internal are working over HTTP.

5.2 Major integration problems

SOA integration requires tight cooperation between different departments and partners/customers. Everybody must understand SOA architecture and principles to be able to work as a team. Usually SOA team consists of architects, IT and business analysts, designers, testers, network administrators, development partners and customers. Analysts analyze business and system requirements and propose service candidates. Architects control overall design process and define company standards for WS development like namespaces, BPEL domains, ESB structure and services registry. Designers create services, test and publish them. Testers check that applications that use services are working correctly and after that they are available for the customers/partners to use. The complexity of information flow that must be followed each time new service is created or old service is updated is shown at Fig 16.

As you can see there are still a lot of human-tasks around SOA integration process. The process is hard to synchronize and plan, because all participants belong to different departments and even companies.

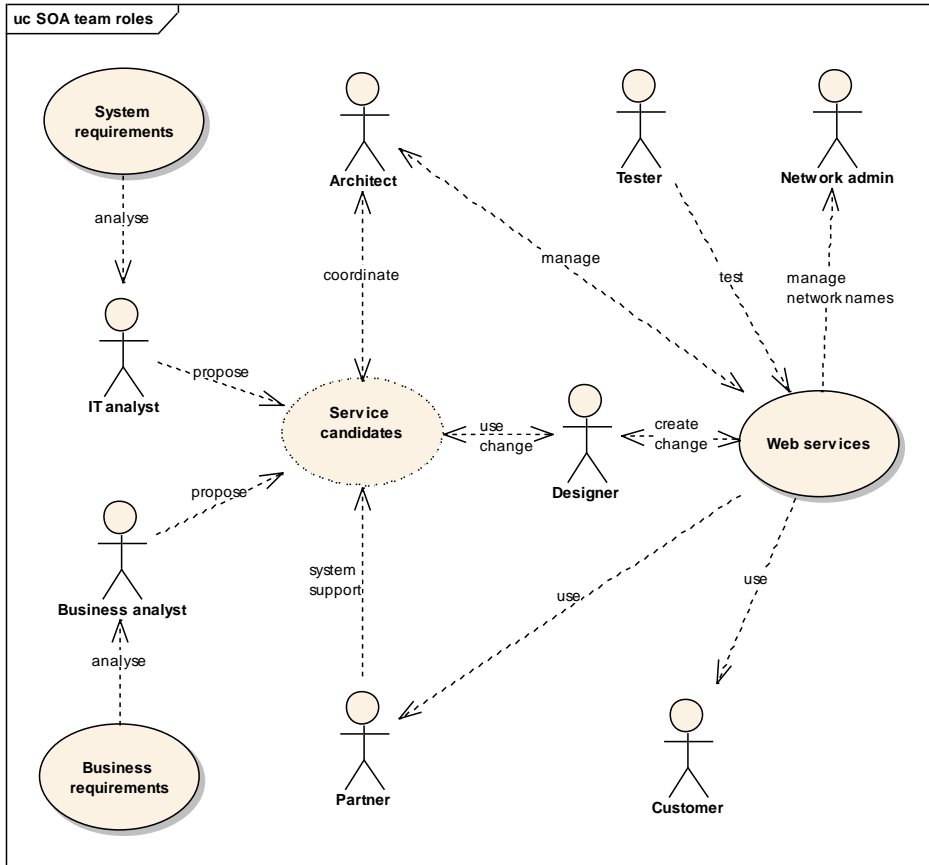


Figure 16. Roles and activities of a SOA team

5.2.1 Addresses and names

One of the major problems that a customer/client faces is service discovery. If a customer does not know the service address in advance he cannot use the service. UDDI (Universal Description Discovery and Integration) is a platform-independent, XML-based registry for businesses worldwide to list themselves on the Internet. UDDI is an open industry initiative, enabling businesses to publish service listings and discover each other and define how the services or software applications interact over the Internet. Even UDDI is not sufficiently good solution to manage the list of existing services, because this list must be created manually. Service deployment already has all information about the service, but it is hidden from the user and all the notifications about service list updates are done manually by sending e-mails or calling ESB console and manually checking service directory. From my experience company waste up to 20% of

project time synchronizing addresses and names with all the involved parties (same Fig 15).

Service discovery must be essential part of the SOA platform. Client looking for a service should have a chance to specify and get extra information about service version, uptime and functions that service offer.

5.2.2 Services are not intelligent

Services are entities that are independently deployed, assigned a version and managed. Service management can be done by various monitors that are available from the vendor, but they can only control the execution process of the service (whether it passed or failed). This happens because a service is not able to control itself – it does not have the minimal required autonomy. A service instance exists only during service invocation. And during the rest of the time the environment can be dramatically changed. Services cannot find other services and connect to them depending on different conditions like version, service type, response time, etc. This leads to enormous amount of time wasting by designers/support to organize and keep track on existing services. Changing a service in these conditions is even more difficult - all partners/customers must be warned to update and re-deploy the service, because changes in WSDL file (Web Services Description Language) cannot be automatically implemented. This happens because service connection point is created during design and has no “re-load” methods that can modify endpoint parameters during runtime. From our experience this overhead can create a persistent flow of corrections that “eats” service uptime. Overall service malfunction correction can take from 1 day to several weeks depending on how early we find the problem root cause and how fast the changes will be implemented by our partners. During all this time service is unavailable.

5.3 Intelligent agents as an improvement for services

Agent cannot replace human in the complex SOA implementation process. They can help to reduce the amount of time SOA team spends to manage the existing services.

5.3.1 Addresses and names

This problem can partly be solved by our Network names management layer that we described before and showed on Fig 14. Services always have the same naming rules for the partner/client. And their end point can always be connected to the newest version of the service (unless WSDL is unchanged) or to the one that is currently working (in case the main service fails). Nevertheless it is done

manually and requires extra work (and especially time) from our network administrators to configure routing. With a number of services to grow this solution becomes more and more complex to maintain. This must be automated and one of the possible solutions is agent approach that solves both problems mentioned above.

5.3.2 Intelligent services

Web services are usually intelligent, but not proactive [Huhns, 2002]. Agents, by definition are proactive and intelligent. Some of them to a greater extent than others, some of them are with very little intelligence, but the main point is - they act. If every service has at least one intelligent function – self-control we will identify system failure long before the customer rise trouble ticket. Unfortunately SOA does not imply that its components are actively collaborating with each other. All actions supposed to be initiated by external systems (or humans) and SOA place is to route (manage) system interconnections. In our experiments we took Oracle ESB that is a part of Oracle SOA Suite 10g middleware. ESB is instance-based and only functionality that API gives is instance control. It means that agent platform must be deployed separately.

Related work has being done by number of persons [Cooney et al., 2003] [Peters, 2005], but new to this paper is integration with industry accepted technologies like ESB or BPEL and less strict limitation for agent’s mobility.

General idea of agent integration into SOA is represented on Figure 17.

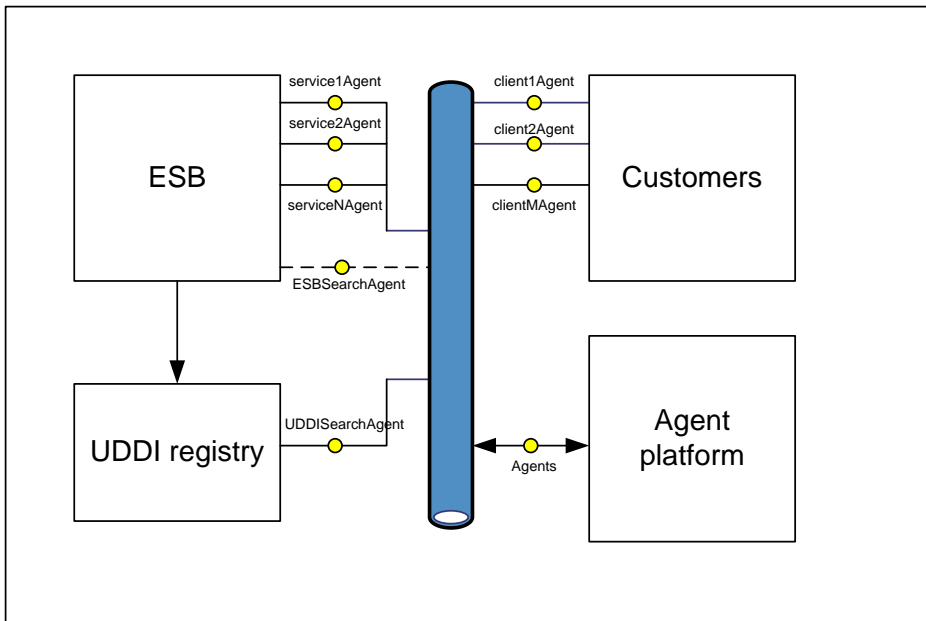


Figure 17. Agent platform integration with SOA

For the project structure it is not important whether agent will physically move to customer platform or it will stay within home platform. Agent platform itself is also not very important. It must support agents that know SOAP, XML, HTTP/HTTS and WSDL. It seems easier to choose one that is based on JAVA language, for instance – JADE.

SOA platform and UDDI part (if exists) remains unchanged. Agent platform is set up between service provider and customer. Customer will get URL to agent platform instead of direct link to the service.

There are 3 possible configurations:

1. With UDDI and without ESBSearch agent
2. Without UDDI and with ESBSearch agent
3. With UDDI and with ESBSearch agent

There are three major types of the agents – search agent, client agent and service agent. If there is UDDI registry defined one more type is to appear – UDDI search agent. As it is defined in specification for UDDIv3 – client can subscribe to specific service. Subscription provides clients, known as subscribers, with the ability to register their interest in receiving information concerning changes made in a UDDI registry. The main problem is that information to the registry is entered by human. That means service registry will never be 100% up to date. Search agent type that is represented by ESBSearch agent is created to fix this problem. It scans ESB for new services to appear and communicates with existing service agents regarding services changes. If search agent finds endpoint without service type agent (new service) assigned to it, it initiates agent platform to create a new service type agent for this endpoint. ServiceN agents are simple agents that keep track on the specific service (1 agent per service). They control service functionality and changes. If new service is created Agent platform gets information from ESBSearch or UDDISearch agents and creates new ServiceN+1 agent to control this end point.

ClientM agents are the ones to help customer to find and connect client to the service he wants. This type of agent is created for every customer connected to Agent platform. ClientM agent supplies the client with all information gathered by all the agents. Such a structure gives us number of benefits and fixes the major problems:

- Human should not maintain service list directory and track addresses
- Clients should not maintain their endpoint lists and request for service updates
- Service list updates are not a manual work any more

The problem that cannot be solved by cooperation with agent platform is service endpoint re-deployment. If connected service is changed customer must update their connected adapter.

Practical implementation of this model is not trivial. If I take, for example, Oracle ESB within Oracle SOA suite there is no API that supports creating such an agent as ESBSearch agent. Customer can wait until Oracle publishes platform API or do reverse engineering for oraesb.jar library that is responsible for service registration within ESB.

5.4 JADE implementation

Implementing SOA is not easy. In practice, new architecture brings new problems instead of old. And this is not because of bad implementation. Principles that SOA is built on are rather old. ESB in general is big routing hub where SOAP application requests are routed. Routing protocols are well-known objects and principles they are built on could be used here as well. The result will be much better than updating routing rules manually. Still this is not a good solution. Using active components (like agents) that act autonomously and do major low-level work will create much better results. With existing SOA platform this is not possible. Either vendor will open API that enables platform extensions or will build new system based on new approach. For large companies it is suggested to consider agent-oriented service design as a next step for IS integration.

5.4.1 Web services and software agents

The first bridge between web-services and JAVA agents was made by Whitestein Technologies and their product WSIG [Greenwood, 2005] (Web-Service Integration Gateway). They found many points of contact between these two technologies and propose method how to integrate two worlds. Their model is shown on the Fig. 18.

As a result – agents can exchange messages with web-services⁸. We use this property to integrate agent platform with ESB and delegate some functions that are performed by human to intelligent agents.

Joining agents and services gives intelligence for web services that is missing for now.

⁸ Web-services, due to their passive nature, can't discover services published by agents and dynamically use their functions.

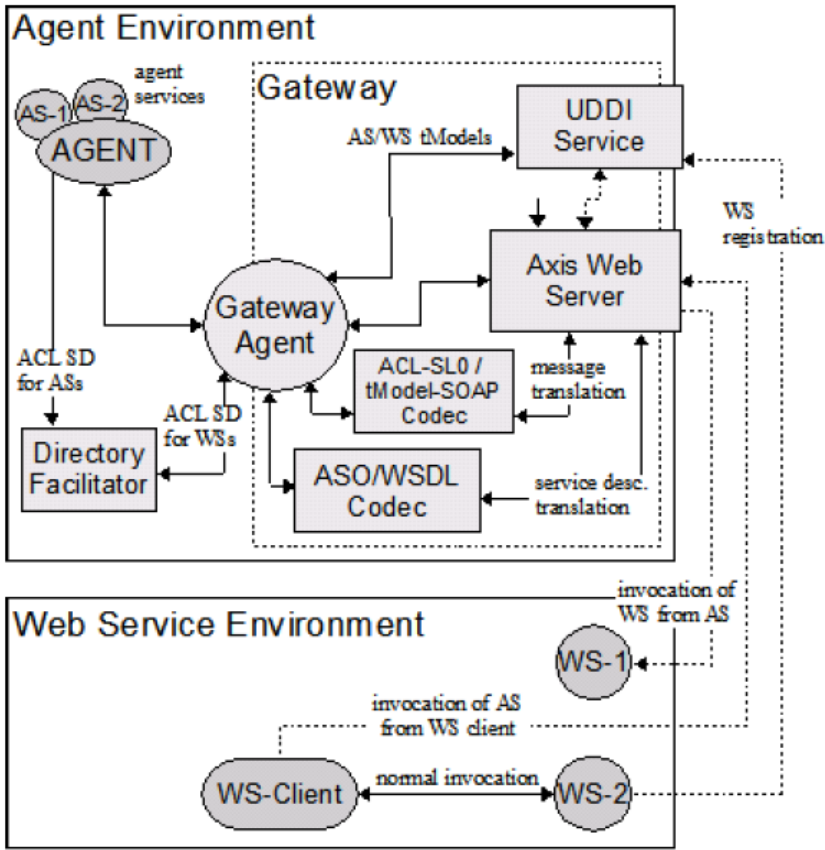


Figure 18. Standard WSIG architectural model

5.4.2 Making agents work

In the next Fig. 19 I introduce the vision of how agent's platform can help to automate routine work. Many of the system parts are already up and running while the others are our potential to grow.

5.4.2.1 Transform ESB metadata into agent ontology

Oracle ESB agent performs translation function between server metadata and agent's internal structure. It gets XML data from the server API, extracts and maps oracle objects to agent's ontology concepts and caches this information for 15 min after which an update has to come. Agent provides this information to other agents upon the query. ESB agent implements 2 behaviors – one to update its knowledge base and other – to reply for services list request. ESB agent is core part of the system and its behaviors are cyclic opposite to applet agents that work only in time period from client page open to browser close.

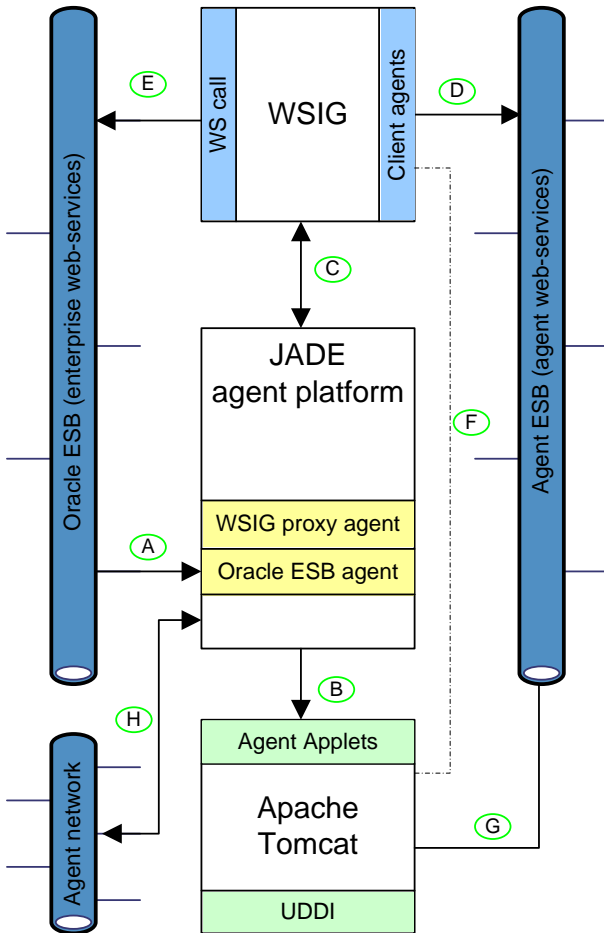


Figure 19. New enterprise service bus model

5.4.2.2 *ESB service lists for development team*

Data extracted from Oracle ESB server is already interesting for the developers and customers. We created applets to display information in a convenient way. User can search, sort and filter output using different parameters and get extra information from WSDL <documentation> tags.

5.4.2.3 *WSIG proxy agent*

WSIG proxy agent translates SOAP message requests/responses into agent ACL. It also maps WS schema with agent service description. Agent allows calling agent functions as web-services and vice versa. If UDDI option is activated agent registers services in UDDI register.

5.4.2.4 Agents as intelligent web services

Every Oracle ESB service can be represented by agent that publishes service operations using WSIG proxy agent in WSDL format. This conversion from service-to-service is needed to make service proactive and have more control over client's connections. Oracle ESB service control is far from being easy and informative enough. Client's statistics suffer from address and process data absence.

5.4.2.5 Oracle WS call

Agent calls web service function after client calls service representation at Agent ESB. For the customer there is no difference between calling Oracle ESB directly and calling agent representation of the service at Agent ESB.

5.4.2.6 Agent's WS management

Since agent is situated in the middle between customer and web service it can carry many useful functions like gathering client's statistics, notifying client about service updates or managing subscriptions. These functions can be exposed as applets on application server. WSIG supports automatic SOAP envelope request/response generation and WS invocation through server applet. This is very useful for testing purposes when client needs to know if service is alive without running special software like SoapUI. In fact this can also be automated by agents. They can call web service and check if the reply is reasonable (i.e. has no error in response) and set appropriate status for the service. The problem here is services that perform insert/update operations in databases.

Furthermore agents can store statistics into the database to be less dependent on platform failures and amount of system memory (not shown on the picture).

5.4.2.7 UDDI service

UDDI is widely used standard. WSIG has ability to publish service in UDDI registry. Here, again, agents can automate this manual work⁹.

For the described configuration of the system we recommend using either jUDDI or UDDI4j (default for WSIG) software.

5.4.2.8 Open agent platform

JADE is an open agent platform that follows FIPA standards and can adopt many different types of intelligent agents. JADE can also work in a cluster with other JADE platforms. Company doesn't use this ability like it doesn't use

⁹ Publishing operations are manual

UDDI at everyday work, but it can use this ability in the future to load balance between client and server.

5.4.3 Conclusion and further work

I used agents to improve company's everyday work with commercial off-the-shelf product - Oracle SOA suite and particularly in its weakest part - Oracle ESB.

Implementation of the system gave number of benefits compare to the original system:

1. Lower down response time to the client
2. Ability to search services by name, domain, deployment time, etc.
3. Ability to read service and operation description (documentation) directly from WSDL
4. Ability to share information about services availability between different user groups (developers, partners, customers, analysts, etc.) without getting access to the system
5. Ability to connect to the RPC-style services
6. Possibility to create "active" services like agents

The next step is to improve single services and bind them with WSIG. This step requires database for statistics and applets for management. As the primary goal was to prove concept of successful integration of web services with agents, future work will either be outsourced to professional programmers or be marked as internal company standard and developed inside.

6 Distributed sensor agent networks

Sensor networks are being used and developed for more than half a century. High impact to their growth was initially stimulated by aerospace industry that needs a lot of data about environment where aircraft is on. Automotive industry is the next big area that uses a lot of sensors that communicate with CAN protocol. All they are usually wired networks that are expensive and static. In opposite to them wireless sensor networks (WSN) become more and more attractive alternative thanks to technological progress. It makes the price, size and computational power of embedded devices reasonable enough to implement huge number of small independent devices. Here come the next problem – how to organize them?

6.1 Overview of sensor networks

There is no strict classification of the sensor networks. They can be divided in two big parts - wired and wireless. If it is not explicitly said in the text the further concepts will cover wireless networks, as for wired the majority of the problems do not exist. Notion "distributed" is hardly applicable for wired networks as well, because usually wired sensor network is physically constrained by the size of aircraft, car or building. Buildings are probably the biggest wired sensor networks holders, though nowadays they start to use more and more wireless devices. Wired networks also tend not to suffer from ad-hoc network structure because all the nodes usually have static position (address). In general wired network is special case of wireless network where connections between elements are fixed. Wireless networks allow designers to develop new functions of the elements. They are:

1. sensors can change position (i.e. move)
2. sensors can be massively deployed
3. maintenance free sensors

Typically the node in WSN is sensor with CPU, battery, wireless communication module and some flash memory on board. It is self-sufficient element that can operate independently from other nodes or in cooperation with them. Here I want to underline basic similarity between sensor nodes and agents. Nodes have all possibilities to play the role of intelligent agent.

Securing information in wireless ad-hoc networks [Carman et al., 2000] is a non-trivial task due to the nature of their structure. Radio communication between sensors requires encryption of the messages and physical implementation of the sensor nodes in real environment requires from the node to be tamper-protected. It is very hard to comply with these two demands, because physical tamper-protection, in most cases, is unachievable. Encryption

is usually either non-symmetric or symmetric. The latter is very sensitive to key revival. It is hard to distribute key securely and more - to keep it secret during operating time. The further chapters describe Time Limited Memory Keys (TLMK) protocol of key distribution/storage for the ad hoc sensor network based on smart sensors with limited CPU power resources. Originally protocol is designed to fulfill MICA2/MICA2DOT platform specification.

6.2 Security problems in WSN-s

There are much more security problems in ad-hoc sensor networks (see Table 5) [Bhargava et al., 2002] than in the others and the main reason for this is power limitation. Each sensor can rely only on the battery it has being deployed with and thus the operating time of the sensor network directly depends on the node's ability to retrench battery power. There are sensor nodes that are able to recharge accumulator from solar battery or other sources, but we consider them as very rare type and still not having unlimited power source.

Distributed sensor networks (DSN) whose mission is very short in time or information security is not the part of requirements specification may have benefit from not using any protection at all. Some hardware allows IEEE 802.11i-2004 (WPA2) standardized protection. In other cases sensor network security will reach higher level implementing TLMK protocol.

For the sensor networks that operate on batteries the importance to have simple, reliable and secure protocol is very high. It is also important to set up implementation domain, because there are different approaches to build networks depending on the network size, topology, and sensor operation modes. It is not possible to fit all various networks in one method.

Table 5. Security problems for different type of networks

Security problem	Wired	Wireless	Ad-hoc
Accidental Attack	yes	yes	yes
Passive Attack	possible	yes	yes
Active Attack	yes	yes	yes
Broadcast based communications	possible	yes	yes
Highly distributed	yes	yes	yes
Heterogeneity	yes	yes	yes
Limited computational ability			yes
Easy theft of nodes			yes
Vulnerability to tempering			yes
Battery power operation			yes
Transient nature of service and devices			yes
Physical Protection is not possible			yes
Cooperative nature of nodes			yes

Major active attacks on wireless sensor networks (WSN) are presented in Table 6 [Lupu 2009] [Bojkovic et al., 2008]. We do not consider, of course, the situation when strong electromagnetic field totally disallows communication making entire network unavailable.

Table 6. Possible attacks in multi-hop WSN

Layer	Name	Type	Description
Network	Wormhole	active	Tunnel packets received on one part of the network to another
Network	Black hole	passive	Does not forward or replay messages
Network	Byzantine	active	Attacker gains access to the node keys and compromise the network from inside. This type of attack includes such a malicious actions as selective forwarding or false route injection
Network	Flooding	active	Generates messages to flood network and cause DoS attack
Network	Resource consumption (sleep deprivation)	active	An attacker or a compromised node can attempt to consume battery life by requesting excessive route discovery, or by forwarding unnecessary packets to the victim node
Network	ACK spoof	active	Convince that weak link is strong or that dead node is alive
Network	HELLO flood	active	Attacker can mess neighbor discovery phase
Network	Sinkhole	active	Attacker creates metaphorical sinkhole by advertising for example high quality route to a base station (KDC)
Network	Location disclosure	active	An attacker reveals information regarding the location of nodes or the structure of the network
Physical	Jamming	active	Radio signals can be jammed or interfered with, which causes the message to be corrupted or lost
Physical	Eavesdropping	passive	Eavesdropping is the intercepting and reading of messages and conversations by unintended receivers
Multi-	Replay	active	Replays message to simulate the

layer			original sender
Multi-layer	Man-in-the-middle	passive	An attacker sits between the sender and the receiver and sniffs any information being sent between two ends
Multi-layer	Impersonation	active	Use other node's identity

Active attacks lead to routing loops, increased latency, decreased lifetime of the network, low reliability, information capture, etc.

6.3 Target implementation domain

Here I will consider sensor network build on MICA2/ MICA2DOT nodes and having single central node -‘Sink node’ with no power or processor (CPU) limitations (uses fixed electrical network and fixed position). Typical implementation area for this can be field/building monitoring. For smaller types of sensor nodes – human body area network [Chen et al., 2010]. In this case there may be multiple sink nodes that are synchronized with each other.

6.3.1 Structure of MICA2/MICA2DOT notes

The physical structure of MICA2/MICA2DOT mote includes CPU (ATmega128L), external flash memory and radio see Fig 20. Usually developer stores the program/data in internal (or external) Flash memory. This is not secure because attacker can capture the node and read the contents of the memory (tamper the node) and reveal the key material. Many of the researchers ignore this issue or consider tamper-protection rather expensive and thus – unacceptable [Karlof et al., 2003]. But I will show here that node's high tamper protection is not something very expensive, but just a matter of organizing key-material storage.

Here I propose to save the keys in random-access memory (RAM) of the device, i.e. keys will only be accessed during node operation mode. It will be tremendously hard to read RAM of the running program from the outside. Program reset will automatically erase contents of the memory and all sensitive information. Functioning longer than lifetime of the key will also erase critical key information.

6.3.2 Conception of operation for DSN

General life-cycle for the nodes in distributed sensor networks starts after Manufacturing and Storage phase with Pre-deployment at Fig 21. During this stage each node gets some specific data about the mission. Nodes may get software, unique IDs and secret keys for exchange. Then it goes Deployment

phase when nodes are physically implemented into environment and Mission itself.

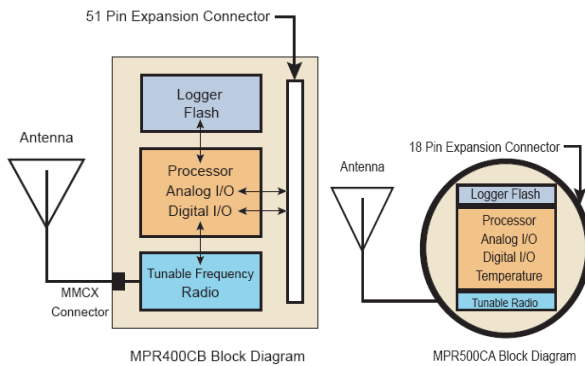


Figure 20. Structure of MICA2/MICA2DOT motes

After mission is complete nodes may be taken out from the field and re-initialized for the next mission. Notice, that in all these states, except pre-deployment, we cannot guarantee nodes security because of the lack of control. Deployment and Mission take place in hostile environment. Manufacturing and storage can also be considered as hostile unless steps 1-3 are done by one organization.

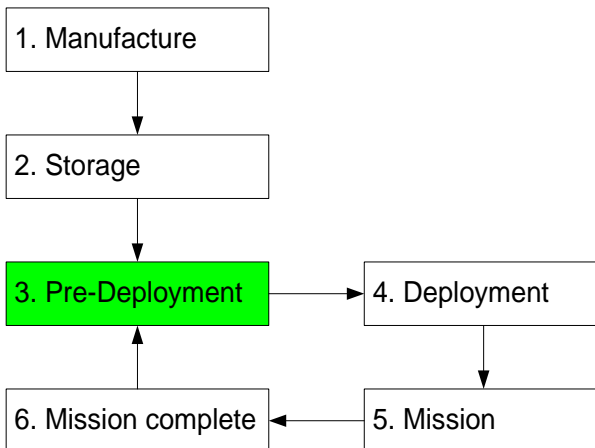
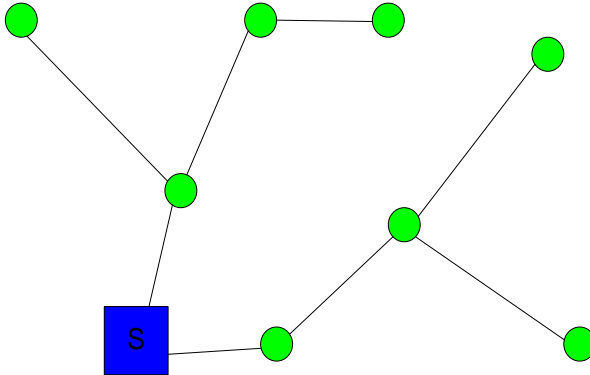


Figure 21. Concept of operation for DSN

6.3.3 General network structure

In sensor networks there are no reasons in general to keep information inside the network. User collects information using nodes and transfers it to some fixed location (central hub). This location is called ‘Sink node’ that may be represented by PC with wireless interface to DSN. Such structure is shown on Fig 22 [Radzevych et al., 2004].

Sink node is the target for any node to transfer information. It has unlimited battery (power network connection) and high CPU speed in comparison with sensor nodes. Physically there can be more than one sink node in the network, but in general information should hit one place that plays role of central hub that manages the network.



Sink node

Figure 22. Typical network structure

6.3.4 Non-symmetric cryptography

To provide the same secrecy with non-symmetric key its length must be at least 10 times longer than symmetric one [Schneier, 1996]. Computational energy behind it is also several times bigger because of the complexity [Fokine 2002] [Čapkun et al., 2003] and transmission energy spent exchanging keys linearly depends on key size. Though network structure is suitable to implement public key infrastructure (PKI) with sink node as key distribution center (KDC), we will try to avoid using non-symmetric key cryptography because of high energy and memory consumption.

6.4 TLMK protocol

TLMK is based on Otway-Rees protocol where in addition to key and "salt" there is a key lifetime (L) transferred to the target node. Central node (sink node) will provide key generation and management (like key updates, revocations). Since the key will be stored in the RAM key pre-deployment should be implemented just before node implementation. Mission program cycle of the node will run only after it obtains the initial key for communication. To build the

network in specified domain with appropriate level of security¹⁰ we need to define the protocol.

Key management goals:

- The protocol must establish a key between all sensor nodes that must exchange data securely.
- Node addition / deletion should be supported
- Unauthorized nodes should not be allowed to establish communication with network nodes

Key management constraints:

- battery power
- memory constraints
- transmission range
- tamper protection
- sleep pattern
- ad-hoc nature
- packet size

It also would be good to have reliable routing protocol [Hu et al., 2003] [Zapata et al., 2002], but this is beyond the scope of this research. All assumptions are made from the point of fixed routes in the table, like in DSDV protocol [Tripathi et al., 2010].

6.4.1 Key pre-deployment

No one can trust the node's hardware and software it's running before and after pre-deployment phase. That is why key pre-deployment must be taken inside secure environment, i.e. shielded chamber that no one can listen to the communication. Also we possibly re-flash the node's software on this stage to be sure that no nodes are operating with other algorithms. The sequence of key transport in this case is shown on Fig 23.

¹⁰ Appropriate level of security is the level when the cost of network capture by intruder is more than the cost of the network

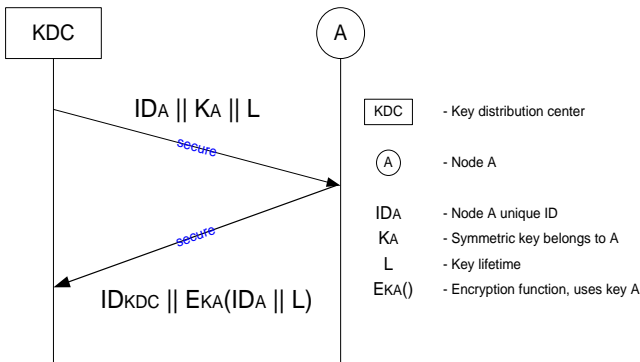


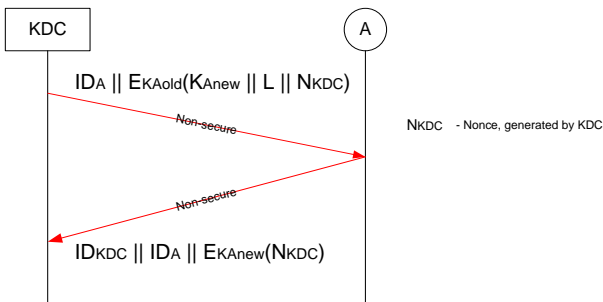
Figure 23. Key pre-deployment

Each node has unique ID to get initial key. The assumption to have secured pre-deployment phase is not strict. Originally mutual authentication is not necessary because we have secured pre-deploy phase where each node gets its unique id and key. But if such a condition is not achieved we use original Otway-Rees protocol authentication.

During pre-deployment communication act each node is initialized with the first key. Optionally message may contain initial salt (furthermore nonce) if KDC has good pseudo-random generator¹¹.

6.4.2 Key update/revoke

This operation takes place in hostile environment so according to our base rules any component of the message can be compromised. It is not necessary to specify source address of the sender because all update/revoke messages will always come from KDC and reply can always be verified by KDC trying all known keys to search for valid nonce. To reduce unnecessary operations node “A” sends its ID as unencrypted value. KDC, if verification succeeded used only 1 “comparison” operation and if not may search for valid sender and report malicious action.



¹¹ Usually KDC has better hardware hence – better PRG. It is recommended to initialize each node with salt. Figure 22 shows simplest communication act.

Figure 24. Key update/revoke

In the terms of time sensitive keys the revocation of the node is fairly simple – just do not update the nodes key after time L. Node will be automatically removed. To keep it inside the network node’s key must be regularly updated within interval smaller that L at Fig 24.

6.4.3 Message exchange

Nodes in the network have to exchange the information between each other, though most of the messages will be between node and KDC. At the moment node secures the message content with symmetric key cryptography and the routing of the message remains unsecured (i.e. open). This assumption came from the fact that the simplest scenario is KDC with all the sensor nodes within visible range (i.e. no routing is required). These interactions are shown on Figure 21. Here we describe native TLMK exchanging method and Kerberos V modified for DSN for comparison. Both protocols use KDC as a third trusted party. Simple scenario does not also include node partitioning/grouping that requires additional group key management protocol to use.

6.4.3.1 Transparent message encryption

To exchange data securely between nodes we do transparent symmetric message encryption through KDC at Fig 25.

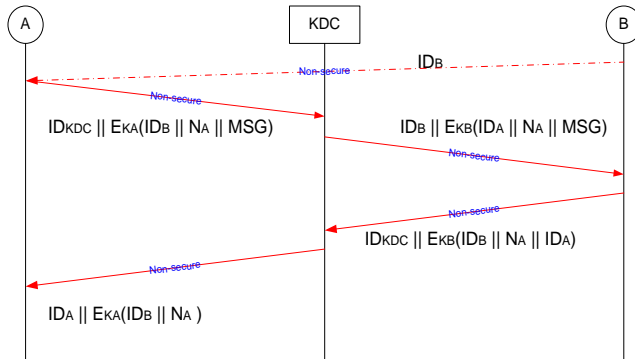


Figure 25. Transparent message encryption

This protocol provides mutual authentication and message delivery acknowledgment. Due to its symmetrical nature both nodes have to accomplish almost the same amount of computation/ communication tasks thus having the same power consumption.

It is also easy to modify it for communication between node and KDC. Just remove right hand communications from the diagram and ID_B from the message.

6.4.3.2 Kerberos V modified for DSN

This protocol [Menezes et al., 1996], developed by NAILabs has similar structure. It also provides authentication for both parties and message acknowledgment. Its structure is more robust due to the usage of extra session key at Fig 26.

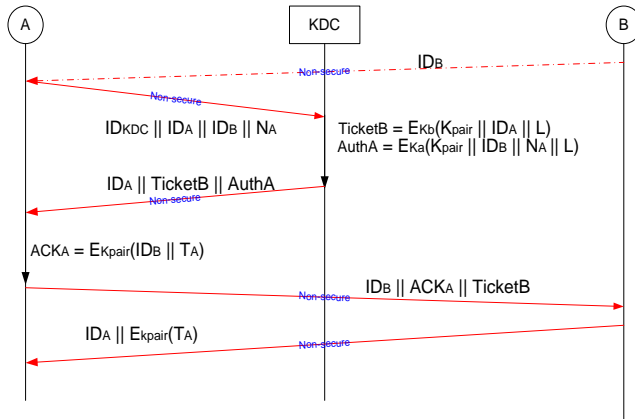


Figure 26. Kerberos V for DSN

Session key has also time limit that can be different from the node's secret keys lifetime.

6.5 Complex scenarios

Next step for the protocol to be applied to is multi-hop network. Message passing the network must be either routed or replayed. Since it is not possible to explicitly transmit the message to specified destination like in fixed networks (we don't use directional antennas and signal spreads in all directions with the same power) we may use only Dynamic Source Routing [Johnson, 1994] [Johnson et al., 2007]. Thus the only critical parameter for routed WSN-s to protect is source address. Rule for destination is computed by local node that constructs local routing table. For the message replay strategy (no routing in fact at all) developer only need to avoid situation when message can be re-send infinitely (i.e. message loops).

6.5.1 Multi-hop network

The goal is still to have single KDC, but it is not visible for all the nodes directly. Typically the part of multi-hop network can be described as on the Figure 27.

Such network topology has all basic element connections: sequential (each node has at least 1 and most 2 connections like, for example, nodes C, D, G or H

on the graph) and mixed connections (each node has more than 2 connections, like nodes A, B, E, F).

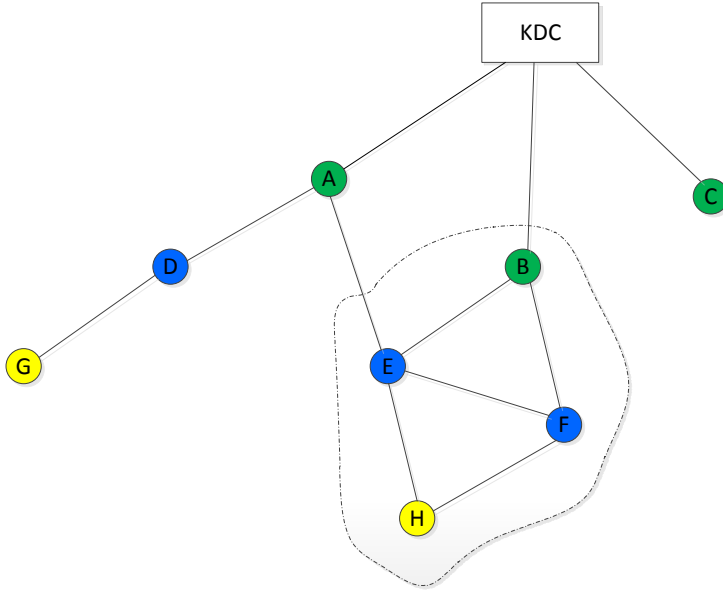


Figure 27. Partitioning the network

1. Unsecured routing

TSMK protocol works the same way in networks with message replay strategy like in peer to peer communication with KDC (Fig. 28). Having node(s) in between the source node and KDC opens possibility for black hole attack but this type of offense can't be defended by encryption. Wormhole or sinkhole attacks are not possible due to absence of routing table. Each node either replays message or does not replay (like on the picture below). It may seem there is no need for routing in WSN, but for large number of nodes and frequent message exchange routing will prevent network from collapse. I estimate TSMK in multi-hop network will be efficient for maximum at 100 nodes and 1 message/minute.

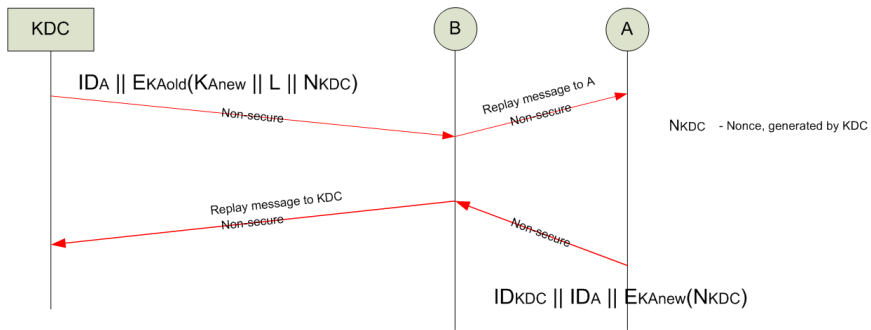


Figure 28. Unsecured message replay. No routing.

2. Secured routing

In general to be success at secure routing we need to protect source address from being changed by anyone except the sender of the message. At the same time we must assure that every node that forwards the message is able to read source address and authenticate it. Here can be used Kerberos V modified for DSNs, but that increases traffic in network and lowers down efficiency of symmetric cryptography. The cost of extra message exchange may be too high from the point of energy consumption and may overlap the cost of CPU time used in non-symmetric key encryption. I estimate that TSMK may be efficient for devices with limited CPU/RAM and the network size up to 50 nodes and frequency of 1 message per minute. This is also subject of simulation.

6.5.2 Coalitions and group-level security

From the beginning the TSMK protocol does not support group management and multiple KDCs. If typical tasks that must be accomplished within groups are taken [Aurisch et al., 2008] it becomes clear that there is no exclusive central point. Nodes within groups should have local autonomy and ability to choose privileged nodes by themselves. They should also update local group membership without KDC. I think that such a task cannot be efficiently achieved without public-key cryptography implementation. And this is another protocol.

6.6 Simulation of TLMK protocol in MASE

Our research laboratory has developed a simulator [Tomson, 2009] for TinyOS to make software agent development more productive. The primary goal of simulation here was to check the size of the code deployed to each agent and visualize sensor nodes communications. It is still hard to estimate energy consumption especially in comparison to other media access control (MAC) protocols [Gopalan et al., 2010]. We estimate it to be proportional to the key size and cipher algorithm used in message encryption/decryption. In simulation I used key size of 16 bytes and Rijndael [Daemen et al., 1998] encryption algorithm. Altogether these are basic rules we follow in our simulation:

1. We trust any communication in open form only during pre-deployment phase. All the nodes that participate in pre-deployment key distribution trusted to have no malicious software and they strictly follow the protocol.
2. All the key material must be stored in RAM of the sensor node. For KDC this rule does not apply assuming that it is protected from being physically compromised.

3. We trust no communication unless it is verified by the receiver. Verification is done using message content decryption with a shared key and payload validation against the protocol.

4. There must be 1 and only 1 KDC for protocol operations.

5. KDC controls the expiration time of each key and updates it when necessary.

For multi-hop network structure each node must have a message replay protection procedure. Since every message in the network is unique there is sufficient in the simplest case to have a cache of last replayed/sent messages and check the new incoming messages against cached one. The size of the cache depends on the number of member nodes and density of the messages coming through the node. It has to be at least one to prevent echo-effect (when single member node replays message back to the sender in order to transmit it further on the network).

6.6.1 Secure mote architecture for MASE

SecureMote follows the typical MACE architecture, i.e. uses SecureMote.cpp for agent definition and LUA script for environment configuration and execution. As a template for SecureMote we used PhoneMote that was developed among other applications in MACE simulator. We need to define the special mote that will play role of the sink node and more specifically – key distribution centre. Since every other node inherits the same code base – every other node can play role of the KDC or sink-node. This is not a problem for simulator that runs on PC and have “unlimited” resources from the embedded devices point of view. In the real environment this code might be split into two parts due to physical memory limitation.

SecureMote consists of:

- TSMK key structure (consists of key itself 128 bits, L – key lifetime measured in seconds and salt or nonce for message uniqueness). Key lifetime is relative because we can't guarantee clock synchronization for the sensor nodes¹².
 - Function to generate new key (operates only in KDC mode)
 - Function to pre-deploy keys (operates only in KDC mode)
 - Function to update/revoke key (operates only in KDC mode)
 - Functions to exchange messages (uses standard DataIn, DataOut simulator structures)

¹² There is no need to control the key validity at the sensor node because we use PUSH-like messages from the KDC to update the key. These update messages are send long before the key becomes invalid.

6.6.2 Special assumptions made for simulation and the result

In the real world scenario some functions will be executed in the different way from the one that was programmed in simulator. More specific they are:

- Key pre-deployment. In the real world it supposes to be mass-broadcast from the KDC to all the motes within the range inside secured chamber. Each mote is passed the clearance test not to have “evil” software on board to log the keys that are distributed to neighbors and accepts only key that is assigned to its ID. In simulator we use virtual “broadcast” function that in fact peer-to-peer communication between environment and the node.
- There is no definition for symmetric encryption algorithm selected to encrypt/decrypt messages. It is beyond the scope of this simulation which one to use. Obviously cipher complexity, strength and speed should be considered before real-world scenario implementation. At the moment simulation uses Rijndael encryption¹³.

The example of simulation one can see on Fig 29. There is a "star" network layout with KDC to execute key pre-deployment phase. The result of simulation is very small agent code size, around 90kbytes in total.

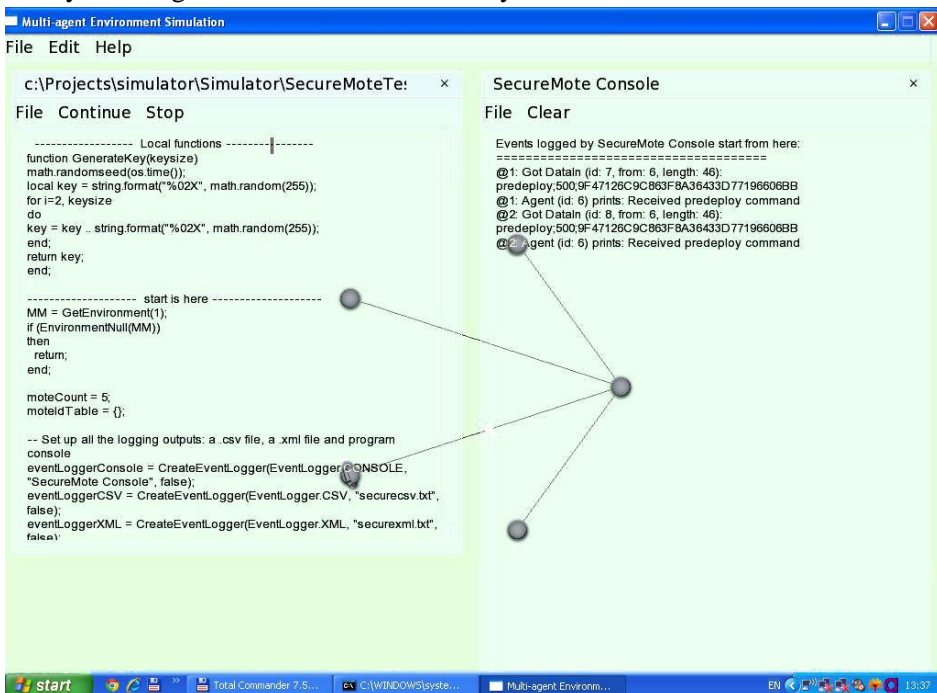


Figure 29. TLMK protocol simulation in MASE Simulator

¹³ This AES candidate is optimal for the trade-off between resistance, efficiency, hardware demands (CPU cycles) and flexibility [Зенин et al., 2002]

6.7 TLMK implementation summary

TLMK protocol inherits Otway-Rees protocol properties and in addition puts more constraints to key handling. Security strength totally depends from encryption protocol being used (in our case it is AES). TLMK has many positive sides to be implemented in sensor networks.

For the comparison 3 types of network configuration patters has being considered: single-hop, multi-hop and routed multi-hop WSN. All of them have sink node that plays role of KDC. In each column I put estimation how high is the probability of certain type of attack for configuration without any protection and with TSMK protection (column name "TSMK"). There are 3 major levels:

"N/A" – this type of attack is not applicable for the configuration

"-" – this type of attack is possible with a complexity equal to complexity of breaking encryption algorithm used in TSMK. In our case it is AES. That is why simple padding algorithms are strongly not recommended.

"+" – this type of attack is possible and have very low complexity. If "+" is in both cells for selected type of the attack that means it cannot be defended with encryption or by this protocol.

The summary is shown in Table 7

Table 7. TSMK protection level

Type of attack	single hop	TSMK	multi-hop	TSMK	routed multi-hop	TSMK
Wormhole	N/A	N/A	N/A	N/A	+	+
Black hole	N/A	N/A	+	-	+	-
Byzantine	+	-	+	-	+	-
Flooding	N/A	N/A	+	+	+	+
Resource consumption	+	-	+	-	+	-
ACK spoof	N/A	N/A	N/A	N/A	+	-
HELLO flood	N/A	N/A	N/A	N/A	+	-
Sinkhole	N/A	N/A	+	-	+	-
Location disclosure	+	-	+	-	+	-
Jamming	+	+	+	+	+	+
Eavesdropping	+	-	+	-	+	-
Replay	+	-	+	-	+	-
Man-in-the-middle	N/A	N/A	N/A	N/A	+	-
Impersonation	+	-	+	-	+	-

Strong sides of the protocol are:

- simple, hardware demands are low
- keys are secure
- provides mutual authentication
- possible to use one time pads
- energy consumption is low and balanced
- easy to identify intruders (wrong ID, key)

Weak sides or constraints of the protocol:

- need to have secured KDC. Compromising KDC will compromise all the network
- not suitable for large networks. Groups, coalitions can only be made by dedicated KDCs that must have trusted relations between each other through master KDC
- if KDC is blocked (separated) from all the nodes for time $> L$ entire network will be destroyed

Protocol is not vulnerable for Sinkhole/Sybil attacks. Others are still possible due to unencrypted link layer. Developer must specify routing protocol to eliminate the rest of the attacks. Passive attack may only reveal the number of nodes in the network and their activity level.

Future work will focus on finding the best suitable routing protocol for TLMK. Link layer in routing messages is not encrypted, thus such attacks as wormholes are still possible.

It is also good to move from simulations to practical implementation for some real project. Right now protocol lacks field tests. There we can investigate how protocol can be used within groups, though major implementation domain is still small to middle size networks with one sink node (KDC) at the beginning.

In general it would be good to decrease the responsibility of the central node to make network more independent from the sink node [Buchegger et al., 2001]. In this case emergent sensor node behavior is also possible. The ultimate goal is self-organized sensor networks with the same (or higher) security level.

7 Access control based on shared knowledge

The way to identify user in existing computer systems has for a long time been very simple. It is enough to provide correct password and the majority of the systems will “recognize” you as legal user of given user name. There is no difference for such a system if there is one person behind this user name or many. Authentication of the user for many years has been reduced to the problem of the user name validation and is not answering question “Is the user really who he/she represents himself to be?” This problem was identified from the very beginning and there were many attempts to solve it, but without success. General idea to overcome this mismatch is to know more about personality of the user and ask more questions from the user about him. The key concept here is shared information knowledge that must be unique and recognized by both sides. In this article we show how intelligent agents can help to improve user’s authentication in computer systems without user being asked the questions.

7.1 Problem background

It is natural for the human to recognize other human by voice, appearance, gait, gestures, etc. You probably never ask your mother to say password in order to let her in. There can be mistakes to identify who is your opponent for instance if we talk about twins who want to cheat teacher and appearance, voice and gesture is not enough to be sure who is in front of you. Even though for the mother and other close relatives this is usually not the case. The main reason for this is the unique information that is shared between the parties during their life. The more people communicate with each other, the more they share experience from the common events the better and more precise they can identify each other. We claim to say that for the human there is no problem to identify other human he knows if it is possible to talk to the opponent personally.

Unfortunately in our modern life personal contacts become less frequent because of wide range of communication services that are offered. People tend to communicate through e-mails, messenger services, phones, etc. more often than personal meetings. This contributes to the success of social engineering [Long et al., 2008] techniques where intruder uses open information in order to pretend to be some key person to get sensitive information.

For the computer systems situation is even worse. Computers cannot recognize human face, voice and gestures as efficient as people do. There are number of reliable systems with very high probability of identifying person by fingerprint, pupil of the eye, voice and, of course password, but all these properties nowadays can be easily copied and re-produced. That is why more

sophisticated systems check not only physical parameters of the user but also try to be more intelligent to ask some specific questions. As I will show later – asking the question from the user is only half a problem. The more difficult is to get and construe answer from the user especially when we expect it in a form of clear text input¹⁴.

In general all the systems that try to get more information about personality of the user are built on 3 elephants: creating the question, asking the question, parsing the answer. Each of these steps can be done in a numerous number of ways and can be automated with different level of the success.

Here there is a major difference between systems – the success rate and its measurement. The prevalent approach is deterministic. Either user knows password (has key) or not and thus – authenticated or not. Overall result is binary AND of authentication functions if they are many. The other approach is probabilistic. System assumes that the user is who he pretends to be if he knows shared secret with some probability. Overall result is the sum of series of authentication functions with a limit = 1. Here system also assumes that some number of iterations should be made in order to reach appropriate level of trust towards the user, because single authentication function cannot give us appropriate level of authentication.

The first type of the system is in fact special case of the second one. If system rises up trust level for authentication function till 100% then it has binary answer to the question if user is authenticated or not.

7.2 Recent situation

There are still only 3 ways to authenticate the user in modern computer systems [Magno, 1996]. They are:

- knowledge-based authentication (passwords and pass phrases, PINs, graphical passwords)
- token-based authentication (physical tokens such as smartcards or badges)
- biometric-based authentication (using users' physical characteristics such as fingerprint, hand geometry, iris pattern or face)

Ultimate systems use all 3 at the same time to diminish risk of wrong person to access valuable resource. Unfortunately, as we mentioned above, all 3 types can be copied. That is why people continue to evaluate computer systems that will be smart enough to identify human by something very personal that cannot

¹⁴ One of the successful implementation of the human identification is CAPTCHA [Yan et al., 2008]. It works quite well in their domain where there is a need to separate human answer from the machine generated one. Unfortunately it is not a way to separate one human from another.

be reproduced or guessed by someone else. There are number of papers published in this area [Nosseir et al., 2005] [Nosseir et al., 2006] [Zviran et al., 1990]. Probably the closest one to this article that can be found in the public resources is "Access control by testing for shared knowledge" [Toomim et al., 2008]. It underlines the problem of user's identification in social networks and shows how private information can be used to increase the level of trust from the system towards the user. Major problems for implementing shared knowledge authentication are:

1. static set of pre-defined questions that needs to be maintained by human
2. problems identifying the correct answer (in case of direct user input)
3. intra-word deviations and spelling errors (like: behaviour vs. behavior)
4. alternative words (abbreviations, acronyms, and synonyms)
5. perception/feeling (violet/dark blue, dark/light, far/close)
6. extra or missing words (like "and", "or", "the", "a", etc.)
7. problems identifying user if the input is organized as a set of pre-defined answers. High probability of "guesses"
8. extra-time for the user to answer and type-in text and as a result - low satisfaction of such a system that asks questions not related to its main tasks
9. probabilistic user access control instead of deterministic

7.3 Trust function

Let me define the "trust function" – y . Codomain of this function is between 0 and 1, where 1 means the user is authenticated and 0 – is not. For the deterministic authentication we have very simple function as superposition of all the single results for the authentication function (f_n). If at least one of them failed the result of the authentication is 0. It can be described with the formula:

$$y = \bigcap_1^{\infty} f_n, y = (1, 0], f = \{1|0\}.$$

For probabilistic approach the result of summary for all authentication functions lies between 0 and 1. Probability of that user is authenticated cannot be negative and at the same time depending on the results of the authentication functions it can be increased or decreased. It can be described with formula:

$$\begin{cases} y_0 = p_0 \\ y_n = y_{n-1} + p_n * y_{n-1}, y = (0, 1), p = (-1, 1). \end{cases}$$

Initial probability of the user to be authenticated (p_0) should also be defined. Initial probability is a probability of the user to be authenticated without applying any authentication function. In the simplest case it can be equal to $\frac{1}{u}$ where u is total number of users in the system. Probabilistic authentication function (p_n) depends on the result of the previous step. Thus one single

authentication step may have different impact on the result depending on current value of probability.

7.4 Requirements for authentication system

From all the points mentioned above there are certain demands for the system that will allow identifying user in addition to username and password combination.

First of all trust function should give us high probability of correct user authentication. How high depends on the task carried by authentication system. For example: if we are going to determine what group the user belongs to and the follow up activity is connected to commercial proposal that is displayed to the user then it will probably be enough to have trust function equal or greater than 0.6. For access to the private information or bank account it will not be enough to have trust function less than 0.99. All depends on system.

System should not ask questions. Thus problems number 2 to 6 and also 8 are being eliminated.

System should observe user's behavior and transform it into knowledge that can be shared with other parts of the system and can be analyzed for the purpose of getting user's identity. This behavior can include and is not bounded to the speed the user types in words, speed user clicks on the mouse button, URLs he is visiting in Internet, items he is searching in search-engines, favorite software to run, working time, etc. Everything depends on the sensors we have to observe the user.

This knowledge we get cannot be reduced to the number of static parameters – otherwise we cannot evolve our system and need to stick to specific data structure. We see that efficient way of holding information is ontology. Ontology can always be extended with new meanings, actions and behaviors.

7.5 Difficulties to design such a system

Possibility to study new ontological concepts allows us to teach agents that operate in terms of this ontology like humans. That is why agents and multi-agent systems (MAS) are recently the most suitable approach to fulfill the task. Focus must be shift towards social-oriented MAS with certain rules similar to human society. We cannot operate with different releases of the software because it is hard from any point to measure version of the knowledge that may grow on all the agents in parallel.

Observation of the user and its behavior may include measurements of the physical parameters. From one point of view the more information we get from the sensors – the better, but the problem here is how to correlate this information

with user identity? It is very easy and fairly cheap nowadays to deploy a sensor network that measures vibration, temperature, light, etc. but how to use this information in a smart way [Estrin et al., 2001]? Every type of measurement must have an impact to the trust function. Very good example of how PIR sensors can give us an assumption about who the user is was discussed in paper. Having smart correlations between the set of physical parameters and users identity may vastly increase the reliability of the identification.

Some of the user identity algorithms are realized in the software that tracks behavior of the customers in web-shops, search engines and media-players. The main goal for this is to help user find information he searches and promote new goods that may be interesting for potential purchaser. Without knowing user's profile it is almost impossible to guess what he/she prefers. Unfortunately recent methods are quite straight-forward and based on very simple tasks like filling in the questionnaire and passing some tests. Information update is done in most cases annually using the same methods. This kind of information is not enough to make assumptions about real identity of the user. On the other hand this type of information is provided by user. He knows what kind of information being asked for and answers on his free will.

Observation of user's behavior initially is not bounded. But information we get as a result may be considered as private. This is true especially if we observe user's communications. Gathering any kind of information in observation mode is always question of privacy and hence – needs to be legally accepted.

Legacy is not probably the main problem in designing such a system. There is still no good theoretical and practical background of how intelligent systems should be designed in order to be human-like. Multi-agent system theory is probably the closest one because it encompasses sensors, behaviors, agents, social activities – everything we need to fulfill the task.

Implementation and usage is another weak point. It can take some time for the system to make assumption about user identity if we want agents to act insensibly. By all means this kind of system will be more complex and slower than system that is operating with traditional authentication methods. The main difficulty to build shared knowledge authentication system is knowledge itself. What to represent, how to represent, how to share this knowledge and how to evolve it.

Taking all above in consideration we don't see shared knowledge authentication as primary way for authenticating users. The main reason for that is speed. User cannot spend much time to communicate with the system just to be able to log-in. Nevertheless after log-in procedure we have enough time to judge who is working behind this account and may restrict access to critical

resources if there is doubt regarding user identity. Strong side of such a system is that it knows the user and more he works with the system the better system recognizes the user. Authentication is not bounded to initial log-in procedure but lasts during the whole period of work.

Another function that shared knowledge can carry on is system usability. If system knows user's preferences it can serve his needs faster. Good example is user profile in operating system. It holds all customizations that user has made to desktop, applications, system components, etc. manually. Instead system will adapt to user's needs automatically. Traditional application areas like online shopping, information search and gaming can also get benefit from knowing real user's profile.

7.6 Teaching agent structure and basic notions

The main activity of the agent is observation. Another important task is to transform the results of observation into knowledge. For such a specific area there is a better approach to build an agent than BDI model [Kinny et al., 1996]. There is an opinion that artificial entities build on the principles of the neural systems fits better to carry out such task [Жданов, 2009].

The result of observation is pure knowledge in terms of agent's ontology. This ontology, when defined, remains unchanged during agent life cycle. It is hard to describe how invention process should look like in terms of basic ontology. But it is possible to describe how this new notions (ontological elements) can be spread between agents and can be used later on.

The main goal of this simulation is to get software agent that can study new knowledge by means of communication act(s) and then use this knowledge to perform some useful action.

7.6.1 Agents roles and properties

For this task it was defined three types of the agents. First type is agent under test called "student". Initially it knows nothing, but can accept messages from other agents and either adopt new knowledge or demonstrate the result of known actions.

Second type of the agent is "teacher". It knows initially something. In our case it knows mathematical operation *add* on two elements – either integer or fractional numbers. Teacher can also send this knowledge during communication act. Teacher is also able to find any students before communication act happens because only students are able to learn.

There is also third type of the agent. It is called "examiner". Examiner knows the same terms teacher does, but it searches for the student agents in order to

inspect their knowledge by sending them special type of requests. In our case examiner sends to student two integers expecting student to answer with the result of summary.

7.6.2 Agents ontology

Initially “student” agent knows only *studentOntology* that is empty and both “teacher” and “examiner” share the same *teacherOntology*. “Examiner” must operate with terms that teacher trains and thus have a feedback from other agents. It is important that “teacher” agent does not perform actions it trains other agents, but “examiner” does and can control that result of the action is correct from the reply of examinee.

For the “student” should be defined general *response* action in order to create any kind of responses for the request from other agents.

For the “teacher” and “examiner” ontology is static (does not change over the time), but for the “student” ontology is dynamically extendable – agent adds new notions and actions by the set of communication acts. All the initial elements are described in Table 8.

Table 8. *Otologies in knowledge sharing experiment at the beginning*

Ontology name	Action	Arguments	Agent
teacherOntology	add	double, double	teacher, examiner
	add	int, int	teacher, examiner
	add	long, long	teacher, examiner
studentOntology	response	string[]	student

In this experiment I have extended *BeanOntology* class that uses “convention over configuration” principle to create new ontological elements. As a result there is no need in external tools to generate schemas – they are generated automatically from the corresponding JAVA beans.

7.6.3 Results in math study for student/teacher simulation in JADE

There are certain ACL messages that agent will react to. They are described in Table 9.

Table 9. *Message types in agent communications*

Message	Source agent	Target agent	Description
PROPAGATE	teacher	student	teachers new concept
REQUEST	examiner	student	check if concept is studied
INFORM	student	examiner	make use of studied concept

There is also automatic reply from the “student” agent to the “teacher” after obtaining the new concept (in our case addition operation), but this reply is ignored by the teacher.

To make knowledge transfer possible ontological class must be serializable (using JAVA serialization) and the recipient must have opportunity after de-serialization make “union” operation with existing ontology. This is now possible after author’s patch to JADE ontology class.

Thus after first PROPAGATE message “student” gets, in fact, serialized *teacherOntology* and makes after de-serialization the following operation:

$$studentOntology = \bigcup (studentOntology, teacherOntology)$$

The result of simulation is shown on Fig 30.

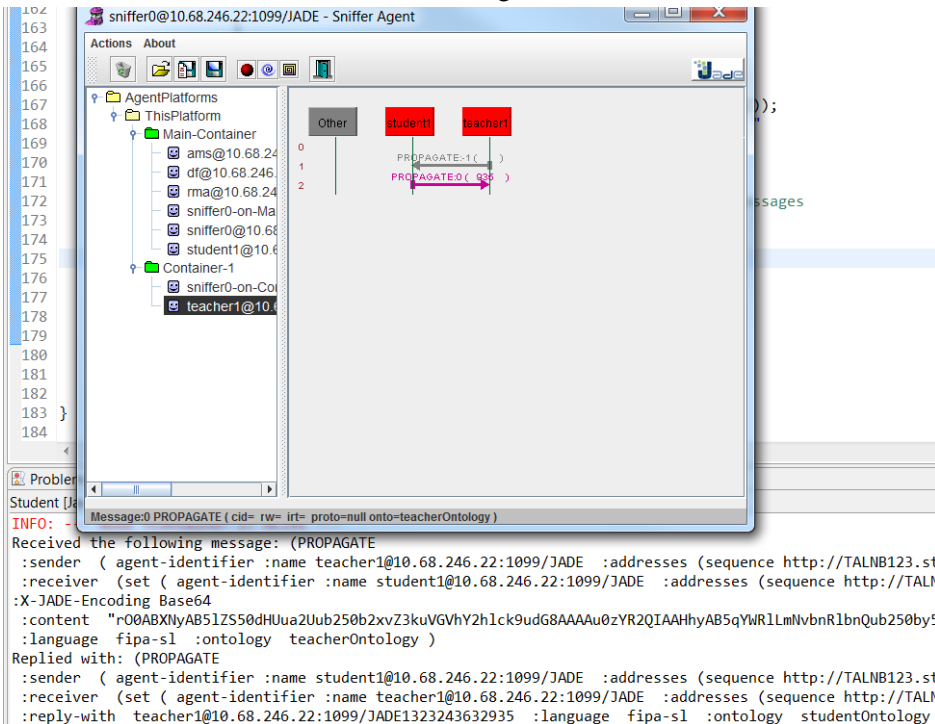


Figure 30. Student obtains new knowledge from the teacher

The next step is exam. To be able to use methods of the class dynamically we used JAVA reflection [Forman et al., 2004]. Since “student” understands *teacherOntology* it can extract arguments of the request automatically. In this particular example we executed all the methods defined in ontology sequentially, but using reflection we can guess arguments types or using more complicated message protocol – guess the required method to execute. This also can be done using “convention over configuration” principles in message exchange.

The result of experiment is shown on Figure 31. There were 2 numbers to summarize and “student” agent executes summary operation correctly giving back 3 answers (by type): integer, float and long. We can update or extend the

knowledge of the student by propagating new ontologies and this is done at runtime without agent being reloaded or restarted.

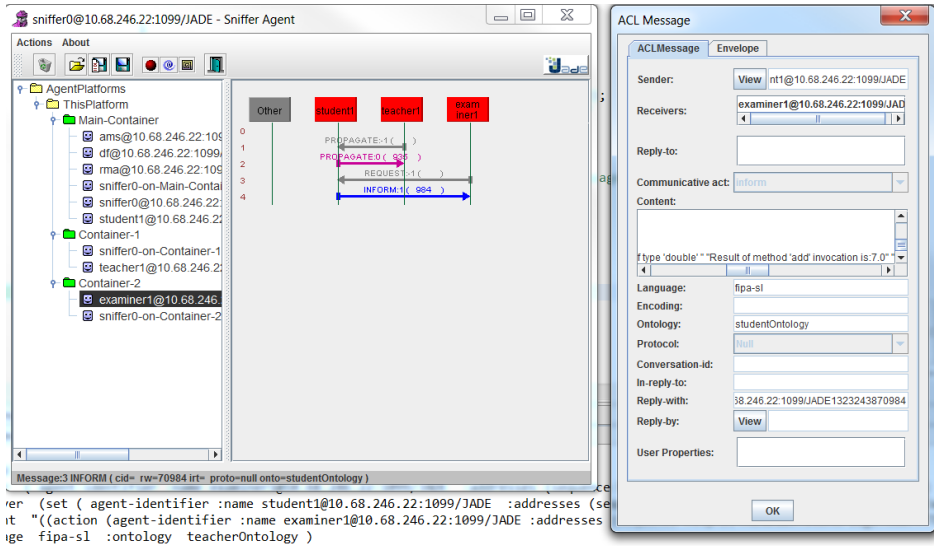


Figure 31. Students answer to examiner with a result of addition operation

7.7 Conclusion and further work

The experiment with knowledge sharing between teacher and student is the first step for creating complex systems that can study new concepts and actions dynamically without being at one of the definite state, hence cannot be described as FSM. Interaction computing model should be used here to describe the behavior of the system. At the simulation stage we work with pre-defined information thus results in most of the cases are clearly predictable. There should be sensors that supply agent with incoming information and events. The role of the developer in such a case is to wrap this information into ontology.

In our further work we will continue to develop the other parts of the system because knowledge representation and sharing is only the first step. To make first working prototype of the entire system some critical parts need to be investigated, particularly – probabilistic function (p_n). For the working prototype application domain must be defined as well, notwithstanding that designing principles are general; ontology elements are domain-dependent.

It will be also very interesting to go further for real world implementation and combine pure software agents with sensor networks. Thus problem of having reliable (not simulated) input can be solved.

Conclusion

There is an inextinguishable interest to multi-agent systems and agent oriented system development over decades. The main reason for that from my point of view is its own niche of the application domain. Particularly in software development most of the paradigms like functional programming or object-oriented programming cover only Turing-complete algorithms. Even parallel computing, that is very close by execution pattern to MAS (where each agent executes in parallel with other agents), has its goal to solve the same Turing-complete algorithms extending single “tape reader” to multiple readers (executors).

MAS goes beyond that principle and while every single agent may use one of the structured programming paradigms (usually event-driven methodology) the entire system itself may operate as super-Turing machine. Particularly this feature of MAS is interesting for me.

This thesis makes a set of attempts to construct different MAS systems and investigate their behavior based on emergency principle. In spite of the fact that all the inputs for the systems were simulated and I can’t say that the resulting system demonstrates emergent behavior there is an opportunity to judge the potential of each system to demonstrate the emergency in real world environment based on experiments. The ability of each simulated system to demonstrate emergent behavior strictly depends on inputs. Simulation is used to simplify the process of MAS development and to predict the results on experiment’s early steps. Furthermore each experiment has had useful practical results that may be used to solve the existing problems in different application domains. For example – TLMK protocol to secure sensor node networks and SOA agents to extend functionality of web-services.

While programming MAS systems for experiments using OO languages like JAVA and C++ it was noticed that there is a trend for OOP paradigm to become closer to AOP. For example “convention over configuration” principle and MEF framework are good practices that can be treated as AOP though are designed and used as a part of OOP.

One of the most interesting topics to investigate further on is agent’s ability to study new ontologies and share them between others. Thesis demonstrates how this is achieved at the last chapter. Nevertheless potential of the self-studying agents is much higher and may be used in any MAS where agent needs to be trained during its lifetime. Since subject of training is new behaviors and not just entities and their meanings there is a possibility to construct true emergent MAS if its inputs are not simulated, but read from sensors.

Bibliography

1. **Akyildiz, I.; Su, W.; Sankarasubramaniam, Y.; Cayirci, E.** 2002. Wireless sensor networks: a survey – [s.l.]: Computer Networks, 38
2. **Apache Software Foundation.** Apache Tomcat. <http://tomcat.apache.org/> – [s.l.] June 2012
3. **Aurisch, T.; Ginzler, T.** 2008. Practical efficiency analysis of a dual mode group key management – [San Diego, USA]:IEEE MILCOM 2008
4. **Bell, Michael.** 2008. Service-Oriented Modeling (SOA): Service Analysis, Design, and Architecture – [s.l.]: John Willey & Sons
5. **Bellifemine, F.; Caire, G.; Poggi, A.; Rimassa, G.** 2003. JADE: A White Paper – Parma: EXP Volume 3 - n. 3 p.6-19
6. **Bellifemine, F.; Caire, G.; Poggi, A.; Rimassa, G.** 2010. JADE programmer's guide – [Parma]: Telecom Italia S.p.A.
7. **Bellifemine, Fabio; Caire, Giovanni; Greenwood, Dominic.** 2007. Developing Multi-Agent Systems with JADE – [s.l.]: John Wiley
8. **Bhargava, S.; Agrawal, D. P.** 2002. Scalable Security Schemes for Ad Hoc Networks – [Anaheim, USA]: IEEE Milcom 2002
9. **Bieberstein, Norbert; Bose, Sanjay; Fiammante, Marc; Jones, Keith; Shah, Rawn.** 2005. Service-Oriented architecture (SOA) Compass: Business Value, Planning and Enterprise Roadmap – [s.l.]: IBM Press.
10. **Bojkovic, Z.; Bakmaz, B.; Bakmaz, M.** 2008. Security Issues in Wireless Sensor Networks – [s.l.]: International Journal of Communications Issue 1, Volume 2
11. **Bollella, Greg; Brogsol, Ben; Dibble, Peter; Furr, Steve; Gosling, James; Hardim, David; Turnbull, Mark.** 2000. The Real-Time Specification for Java – [s.l.]: Addison Wesley Longman
12. **Bonabeau, Eric; Dorigo, Marco; Theraulaz, Guy.** 1999. Swarm Intelligence: From Natural to Artificial Systems – [Santa Fe]: Oxford University Press
13. **Booch, Grady; Jacobson, Ivar; Rumbaugh James.** 1998. The Unified Software Development Process – [s.l.]: Addison-Wesley
14. **Bramwell, M.** 2006. Implementing a MICA2 MOTE sensor network. – [Waterloo, Ontario, Canada]: The University of Waterloo
15. **Buchegger, S; Boudec, J.** 2001. The Selfish Node: Increasing Routing Security in Mobile Ad Hoc Networks – [Lausanne, Switzerland]: LCA-REPORT-2001-008
16. **Burk, Robin; Horvath, David.** 1998. UNIX Unleashed: System Administrator's Edition – [s.l.]: SAMS Publishing

17. **Čapkun, S.; Buttyan, L.; Hubaux, J.** 2003. Self-Organized Public-Key Management for Mobile Ad Hoc Networks – [s.l.]: IEEE Transactions on Mobile Computing, Volume 2 Issue 1
18. **Cardelli, L.** 1996. Bad Engineering Properties of Object-Oriented Languages – [s.l.]: ACM Comput. Surv. (ACM) 17: 471–523
19. **Carman, D.; Kruus, P.; Matt, B.** 2000. Constraints and Approaches for Distributed Sensor Network Security – [s.l.]: NAI Labs Technical Report #00-010
20. **Chandy, Many; Misra, Jayadev.** 1988. Parallel Program Design: A Foundation – [s.l.]: Addison-Wesley
21. **Chappell, David.** 2004. Enterprise Service Bus: Theory in Practice – [s.l.]: O'Reilly Media
22. **Chen, M.; Gonzalez, S.; Vasilakos, A.; Cao, H.; Leung, V.** 2010. Body Area Networks: A Survey – [s.l.]: Springer Mobile Netw Appl (2010) 16
23. **Cooney, D.; Roe, P.** 2003. Mobile Agents Make for Flexible Web Services – [Queensland, Australia]: The 9th Australian World Wide Web Conference
24. **Cranfield, S; Purvis, M.** 2001. Generating ontology-specific content languages – [Dunedin, New Zeland]: 5th International Conference on Autonomous Agents
25. **CVE.** 2011. Computer vulnerabilities and exposures database – [s.l.]: <http://cve.mitre.org/cve/index.html> , December 2011
26. **D'Inverno, M; Luck M.** 2001. Understanding Agent Systems – [Berlin]: Springer-Verlag
27. **Daemen, J; Rijmen, V.** 1998. AES Proposal: Rijndael – [s.l.]: Federal Information Processing Standards Publication 197, available at <http://csrc.nist.gov/> , December 2011
28. **Dorigo, M.; Socha, K.** 2007. An Introduction to Ant Colony Optimization – [s.l.]: Published as a chapter in Approximation Algorithms and Metaheuristics
29. **Erl, Thomas.** 2005. Service-Oriented Architecture: Concepts, Technology, Design – [s.l.]: Prentice Hall
30. **Estrin, D.; Girod, L.; Pottie, G.; Srivastava, M.** 2001. Instrumenting the World with Wireless Sensor Networks – [Salt Lake City]:International Conference on Acoustics, Speech, and Signal Processing (ICASSP)
31. **FIPA homepage** – [s.l.]: <http://www.fipa.org/>, 12 december 2011

32. **FIPA SL.** 2004. FIPA SL Content Language Specification, XC00008G – [s.l.]: <http://www.fipa.org/>, December 2011
33. **Fokine, K.** 2002. Key Management in Ad Hoc Networks – [Linköping, Sweden]: Master. Thesis, LiTH-ISY-EX-3322-2002
34. **Forman, Ira; Forman, Nate.** 2004. Java reflection in action – [s.l.]: Manning Publications
35. **Gopalan, S., Park, J.** 2010. Energy-Efficient MAC Protocols for Wireless Body Area – [Moscow]: Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT)
36. **Greenwood, D.** 2005. JADE Web Service Integration Gateway(WSIG) – [Utrecht, Netherlands]: Autonomous Agents and Multi agent Systems (AAMAS 2005)
37. **Haddadi, Afsaneh; Sundermeyer, Kurt.** 1996. Foundations of distributed artificial intelligence – [New York]: John Wiley & Sons
38. **Henriksson, R.; Kauppinen, T.; Hyvönen, E.** 2008. Core Geographical Concepts: Case Finnish Geo-Ontology – [Ohio, USA]: Location and the Web (LocWeb) 2008 workshop, 17th International World Wide Web Conference WWW 2008
39. **Hoare, Charles Antony Richard.** 1985. Communicating Sequential Processes – [s.l.]: Prentice Hall International
40. **Hu Yih-Chun; Perrig, A.; Johnson D.** 2003. Rushing Attacks and Defense in Wireless Ad Hoc Network Routing Protocols – [San Diego, USA]: Proceedings of the ACM Workshop on Wireless Security (WiSe 2003)
41. **Huhns M.** 2002. Agents as Web Services – [South Carolina]: IEEE Internet computing, Volume 6, Issue 4, pages 93-95
42. **Johnson, D.** 1994. Routing in Ad Hoc Networks of Mobile Hosts – [Santa Cruz, USA]: Proceedings of the Workshop on Mobile Computing Systems and Applications
43. **Johnson, D.; Maltz, D.; Hu, Y.** 2007. The Dynamic Source Routing Protocol (DSR) for Mobile Ad Hoc Networks for IPv4 – [s.l.]: RFC4728
44. **Karlof, C; Wagner, D.** 2003. Secure Routing in Wireless Sensor Networks: Attacks and Countermeasures – [s.l.]: IEEE International Workshop on Sensor Network Protocols and Applications
45. **Khodabakchian E., Shaffer D., GaurH., Zirn M.** 2011. SOA Best Practices: The BPEL Cookbook – [s.l.]: <http://www.oracle.com/technetwork/articles/soa/index-095969.html>, December 2011

46. **Kimlaychuk, V.** 2003. 5 Hungry philosophers' problem. Modelling with JADE – Italy: TILAB "EXP in search of innovation"
47. **Kimlaychuk, V.** 2004. Creating intelligent agents in JADE (an example of ant colony simulation) – [Orlando, FL]: EISTA2004
48. **Kimlaychuk, V.** 2008. SOA Integration Aspects for Large Companies – [Algarve, Portugal]: IADIS International Conference Information Systems, April 9-13
49. **Kimlaychuk, V.** 2008. Integrating Oracle enterprise service bus with JADE agents – [Stara Lesna, Slovakia]: 6th International Conference on Computational Cybernetics, November 27-29.
50. **Kimlaychuk, V.** 2010. Jade contribution – Tallinn: <http://jade.tilab.com/credits.htm> , 12 december 2011
51. **Kimlaychuk, V.** 2011. Security in ad-hoc sensor networks with pre-loaded time limited memory keys – [Shanghai, China]: 4th International Conference on BioMedical Engineering and Informatics, October 15-17
52. **Kimlaychuk, V.** 2012. Authentication using shared knowledge. Learning agents. – [Jeju Island, Korea]: The 12th International Conference on Intelligent Autonomous Systems, 26-29 June 2012
53. **Kinny D.; Georgeff M.; Rao A.** 1996. A Methodology and Modelling Technique for Systems of BDI Agents – [Secaucus, NJ, USA]: Proceedings of the 7th European workshop on Modelling autonomous agents in a multi-agent world: agents breaking away: agents breaking away, pages 56 - 71. Springer-Verlag New York, Inc.
54. **Knublauch, H.** 2002. Extreme Programming of Multi-Agent Systems – [Bologna, Italy]: First International Joint Conference on Autonomous Agents & Multi Agent Systems (AAMAS 2002)
55. **Long, Johnny; Wiles, Jack; Pinzon, Scott; Mitnick, Kevin.** 2008. No Tech Hacking: A Guide to Social Engineering, Dumpster Diving, and Shoulder Surfing – [s.l.]: Syngress
56. **Lupu, T.** 2009. Main Types of Attacks in Wireless Sensor Networks – [Budapest]: WSEAS 2009
57. **Magno, B.** 1996. Survey of user authentication mechanisms – Monterey: Storming Media
58. **Marks, Eric; Bell, Michael.** 2006. Service-Oriented Architecture (SOA): A Planning and Implementation Guide for Business and Technology – [s.l.]: Willey
59. **McCarthy, J.; Minsky, M.; Rochester, N.; Shannon, C.** 1955. A Proposal for the Dartmouth Summer Research Project on Artificial

- Intelligence. Hanover: Dartmouth Summer Research Conference on Artificial Intelligence
60. **Menezes Alfred, Oorschot Paul, Vanstone Scott.** 1996. Handbook of Applied Cryptography – [s.l.]: CRC Press
 61. **Mitnick, Kevin; Simon, William; Wozniak, Steve.** 2002. The art of Deception: Controlling the Human Element of Security – [s.l.]: John Wiley & Sons
 62. **Murray, Arthur.** 2002. Ai4U: Mind-1.1 Programmer's Manual – [s.l.]: iUniverse
 63. **Musen, M.; Noy N.** Protégé homepage – [s.l.]: <http://protege.stanford.edu/> , 12 december 2011
 64. **Mõtus, L.** 1990. Динамика программного обеспечения встроенных систем – [Tallinn]: Valgus
 65. **Mõtus, L; Rodd, M.** 1994. Timing analysis of real-time software – [s.l.]: Elsevier/Pergamon
 66. **Mõtus, L.** 2003. Modeling metric time – [Norwell]: Kluwer Academic Publ.
 67. **Mõtus, L.; Meriste, M. ; Dosch, W.** 2005. Time-awareness and Proactivity in Models of Ineractive Computations – [s.l.]: Elsevier. Theoretical Computer Science 141, p 69-95
 68. **Nosseir, A; Connor, R.; Dunlop, M.** 2005. Internet Authentication Based on Personal History – A Feasibility Test – [Chiba, Japan]:Proceedings of Customer Focused Mobile Services Workshop at WWW2005
 69. **Nosseir, A.; Connor, R.; Revie, C.; Terzis, S.** 2006. Question-based authentication using context data – [Oslo, Norway]: Nordic Conference on Human-Computer Interaction; Vol. 189
 70. **Noy, N.; McGuinness, D.** 2001. Ontology Development 101: A Guide to Creating Your First Ontology – [s.l.]: Stanford University (KSL-01-05)
 71. **Odell, J.** 2011. Foundation for Intelligent Physical Agents – [s.l.]: <http://www.fipa.org/specifications/index.html>, December, 2011
 72. **Otway, D., Rees, O.** 1987. Efficient and Timely Mutual Authentication, Operating Systems Review – [New York, USA]: ACM SIGOPS Operating Systems Review, Volume 21 Issue 1
 73. **Peters, J.** 2005. Integration of Mobile Agents and Web Services – [Leicester , U.K.]: The First European Young Researchers Workshop on Service Oriented Computing

74. **Poole, David; Mackworth, Alan; Goebel, Randy.** 1998. Computational Intelligence. A logical approach – [New York]: Oxford University Press
75. **Radzevych, V.; Mathew, S.** 2004. Security in Sensor Networks: Key Management Approaches – [s.l.]: Unpublished
76. **Russel, Stuart; Norvig Peter.** 2003, Artificial Intelligence a modern approach – [s.l.]: Prentice Hall
77. **Sasse, A.** 2005. Usability and trust in information systems – [s.l.]: Edward Elgar, p. 319-348
78. **Schittko, C.** 2003. WebService Orchestration with BPEL – [Philadelphia, USA]: XML Conference & Exposition 2003
79. **Schneier, Bruce.** 1996. Applied Cryptography – [s.l.]:John Willey & Sons
80. **Selic, Bran; Lavagno, Luciano; Martin Grant.** 2003. "Modeling Metric Time" in UML for Real: Design of Embedded Real-time Systems – [s.l.]: Springer
81. **Shoham Y.** 1993. Agent-oriented programming – [s.l.]: Elsevier Science Publishers
82. **Soley, R.** 2003. Unified modeling language specification – [s.l.]: <http://www.uml.org/> , December 2011
83. **Stepney, S.; Clark, J.; Tyrrell, A.; Johnson, C.; Timmis, J.; Partridge, D.; Adamatzky, A.; Smith, R.** 2005. Journeys in Non-Classical Computation. A Grand Challenge for Computing Research – [s.l.]: International Journal of Parallel, Emergent and Distributed Systems, p. 5-19
84. **Sterling L.; Taviter K.** 2009. The Art of Agent-Oriented Modeling – [London, England]: MIT Press
85. **Tomson, T.** 2009. Research Laboratory for Proactive Technologies annual report – Tallinn: http://www.proactivity-lab.ee/images/proact%20rep_%202009.pdf, p. 48-54, 12 december 2011
86. **Toomim, M.; Zhang, X.; Forgaty, J.; Landay, J.** 2008. Access control by testing for shared knowledge – [Florence, Italy]: 26th Annual CHI Conference on Human Factors in Computing Systems
87. **Tripathi, K.; Agarwal, T.; Dixit, S. D.** 2010. Performance of DSDV Protocol over Sensor Networks – [s.l.]: International Journal of Next-Generation Networks (IJNGN) Vol.2, No.2
88. **Weiss, Gerhard.** 1999. Multi-agent systems: A Modern Approach to Distributed Artificial Intelligence – [s.l.]: MIT Press
89. **Wooldridge, M.** 1997. Agent-based software engineering – [s.l.]: IEEE Proc. on Software Engineering

90. **Wooldridge, M; Jennings, N.** 1995. Intelligent agents: theory and practice – [s.l.]: The Knowledge Engineering Review 10(2)
91. **Yan, J.; Ahmad, A.S.** 2008. A Low-cost Attack on a Microsoft CAPTCHA – [Alexandria, USA]:Proceedings of the 15th ACM conference on Computer and communications security
92. **Zapata, M.G.; Asokan, N.** 2002. Securing Ad-Hoc Routing Protocols – [Singapore]: Proceedings of the ACM Workshop on Wireless Security (WiSe 2002)
93. **Zviran, M; Haga, W.** 1990. User authentication by cognitive passwords: an empirical assessment – [Jerusalem]: Proceedings of the fifth Jerusalem conference on Information technology
94. **Жданов Александр.** 2009. Автономный искусственный интеллект – [Москва]:БИНОМ. Лаборатория знаний
95. **Зенин О.; Иванов М.** 2002. Стандарт криптографической защиты AES. Конечные поля – [Москва]: КУДИЦ-Образ

List of publications

- Kimlaychuk, V. (2012). Authentication using shared knowledge. Learning agents. The 12th International Conference on Intelligent Autonomous Systems, 26-29 June 2012, Jeju Island, Korea
- Kimlaychuk, V. (2011). Security in ad-hoc sensor networks with pre-loaded time limited memory keys. The 4th International Conference on BioMedical Engineering and Informatics, 15 Oct - 17 Oct 2011. IEEE, 2011.
- Kimlaychuk, V. (2008). Integrating Oracle enterprise service bus with JADE agents. In: IEEE 6th International Conference on Computational Cybernetics, November 27-29, 2008: IEEE, 2008, 59 - 61.
- Kimlaychuk, V. (2008). SOA Integration Aspects for Large Companies. In: Proceedings of the IADIS International Conference Information Systems 2008: IADIS International Conference, Algarve, Portugal, APRIL 9-11, 2008. (Toim.) Nunes, M.B.; Isaias, P.; Powell, P. Algarve, Portugal: IADIS Press, 2008.
- Kimlaychuk, V. (2004). Creating intelligent agents in JADE (an example of ant colony simulation). EISTA 2004 (Orlando, FL, July 2004). , 2004, 55 - 60.
- Motus, L.; Meriste, M.; Helekivi, J.; Kelder, T.; Kimlaychuk, V. (2003). A Testbed for Time-sensitive Agents – Some Involved Problems. EFTA 2003: 9th IEEE International Conference on Emerging Technologies and Factory Automation ETFA2003; Lisabon, Portugal; 16-19 September, 2003. IEEE Computer Society Press, 2003, 645 - 651.
- Kimlaychuk, Vadim (2003). 5 Hungry philosophers' problem. Modelling with JADE. Italy, electronic journal TILAB "EXP in search of innovation"

Annex A.

ELULOOKIRJELDUS

1. Isikuandmed

Ees- ja perekonnanimi: Vadim Kimlaychuk
Sünniaeg ja koht: 12 november 1977, Ukraina, Kovel
Kodakondsus: Venemaa

2. Kontaktandmed

Aadress: Paekaare 2-8, Tallinn 13621, Estonia
Telefon: 5233474
E-posti aadress: vadim@dcc.ttu.ee

3. Hariduskäik

Õppeasutus (nimetus lõpetamise ajal)	Lõpetamise aeg	Haridus (eriala/kraad)
Moskva Riiklik Elektrotehnoloogia Instituut	2001	magister
Moskva Riiklik Elektrotehnoloogia Instituut	2001	magister
12 keskkool, Tallinn	1995	keskharidus

4. Keelteoskus (alg-, kesk- või kõrgtase)

Keel	Tase
Vene, Ukraina	emakeel
Inglise	kõrgtase
Eesti	kesktase
Rootsi	algtase

5. Teenistuskäik

Töötamise aeg	Tööandja nimetus	Ametikoht
2012 –	Koolitööde AS	tarkvara arendaja
2011 – 2012	Stoneridge Electronics AS	tarkvara arenduse juht
2006 – 2011	Elion AS	integratsiooni arendaja
2004 – 2006	Stoneridge Electronics AS	tarkvara arendaja
2002 – 2004	Stoneridge Electronics AB	industrialiseerimine, insener

6. Teadustegevus
 - BF38 Gene-Auto, Automaatne koodigeneraator sardsüsteemidele (02.01.2006-31.12.2008), täitja
 - F7007 Gene-Auto, Automaatne koodigeneraator sardsüsteemidele (30.01.2007-01.09.2008), täitja
 - ETF6182 Multiagentsüsteemide uurimine heterogeenses, dünaamiliselt muutuva struktuuriga keskkonnas (Hopad hoc projekt) (01.01.2005-31.12.2008), täitja
 - SF0140113As08 Proaktiivsus ja situatsiooniteadlikkus (01.01.2008-31.12.2013), täitja
7. Kaitstud lõputööd
 - Õppeklassi võrk ehitatud UNIX ja Windows NT integratsiooni alusel – (magister, 2001)
 - Sisevõrgu kaitse Internetis – (magister, 2001)
8. Teadustöö põhisuunad
 - a. Multiagentsüsteemid
 - b. Andurivõrgu simuleerimine
 - c. Tekkiva käitumise süsteemid
9. Teised uurimisprojektid
 - Multiagentsüsteemide tarkvara arendus

Kuupäev: 28/12/2011

Annex B

CURRICULUM VITAE

1. Personal data

Name: Vadim Kimlaychuk

Date and place of birth: 12 November 1977, Ukraine, Kovel

2. Contact information

Address: Paekaare 2-8, Tallinn 13621, Estonia

Phone: 5233474

E-mail: vadim@dcc.ttu.ee

3. Education

Educational institution	Graduation year	Education (field of study/degree)
Moscow Institute of Electronic Engineering	2001	master
Moscow Institute of Electronic Engineering	2001	master
Secondary school nr. 12, Tallinn	1995	secondary

4. Language competence/skills (fluent; average, basic skills)

Language	Level
Russian, Ukrainianian	mothertongue
English	fluent
Estonian	average
Swedish	basic

5. Professional Employment

Period	Organisation	Position
2012 – now	Koolitööde AS	software developer
2011 – 2012	Stoneridge Electronics AS	software development manager
2006 – 2011	Elion AS	integrations developer
2004 – 2006	Stoneridge Electronics AS	software developer
2002 – 2004	Stoneridge Electronics AB	industrialization engineer

6. Scientific work

BF38 Gene-Auto, Automatic Software Code Generation for Real-time Embedded Systems (02.01.2006-31.12.2008), performer

F7007 Gene-Auto, Automatic Software Code Generation for Real-time Embedded Systems (30.01.2007-01.09.2008), performer

ETF6182 Research of multi-agent systems in heterogeneous environment with dynamic structure (Hopad hoc project) (01.01.2005-31.12.2008), performer

SF0140113As08 Proactivity and situation-awareness (01.01.2008-31.12.2013), performer

7. Defended theses

Student's training LAN based on UNIX and Windows NT integration – (M.Sc., 2001)

LAN protection in Internet, based on student's training LAN - (M.Sc., 2001)

8. Main areas of scientific work/current research topics

- a. Multi agent systems
- b. Simulation of the sensor network
- c. Emergent behavior

9. Other research projects

Agent-oriented software development

Date: 28/12/2011

Annex C

DISSERTATIONS DEFENDED AT TALLINN UNIVERSITY OF TECHNOLOGY ON INFORMATICS AND SYSTEM ENGINEERING

1. **Lea Elmik**. Informational Modelling of a Communication Office. 1992.
2. **Kalle Tammemäe**. Control Intensive Digital System Synthesis. 1997.
3. **Eerik Lossmann**. Complex Signal Classification Algorithms, Based on the Third-Order Statistical Models. 1999.
4. **Kaido Kikkas**. Using the Internet in Rehabilitation of People with Mobility Impairments – Case Studies and Views from Estonia. 1999.
5. **Nazmun Nahar**. Global Electronic Commerce Process: Business-to-Business. 1999.
6. **Jevgeni Riipulk**. Microwave Radiometry for Medical Applications. 2000.
7. **Alar Kuusik**. Compact Smart Home Systems: Design and Verification of Cost Effective Hardware Solutions. 2001.
8. **Jaan Raik**. Hierarchical Test Generation for Digital Circuits Represented by Decision Diagrams. 2001.
9. **Andri Riid**. Transparent Fuzzy Systems: Model and Control. 2002.
10. **Marina Brik**. Investigation and Development of Test Generation Methods for Control Part of Digital Systems. 2002.
11. **Raul Land**. Synchronous Approximation and Processing of Sampled Data Signals. 2002.
12. **Ants Ronk**. An Extended Block-Adaptive Fourier Analyser for Analysis and Reproduction of Periodic Components of Band-Limited Discrete-Time Signals. 2002.
13. **Toivo Paavle**. System Level Modeling of the Phase Locked Loops: Behavioral Analysis and Parameterization. 2003.
14. **Irina Astrova**. On Integration of Object-Oriented Applications with Relational Databases. 2003.
15. **Kuldar Taveter**. A Multi-Perspective Methodology for Agent-Oriented Business Modelling and Simulation. 2004.
16. **Taivo Kangilaski**. Eesti Energia käiduhaldussüsteem. 2004.
17. **Artur Jutman**. Selected Issues of Modeling, Verification and Testing of Digital Systems. 2004.

18. **Ander Tenno.** Simulation and Estimation of Electro-Chemical Processes in Maintenance-Free Batteries with Fixed Electrolyte. 2004.
19. **Oleg Korolkov.** Formation of Diffusion Welded Al Contacts to Semiconductor Silicon. 2004.
20. **Risto Vaarandi.** Tools and Techniques for Event Log Analysis. 2005.
21. **Marko Koort.** Transmitter Power Control in Wireless Communication Systems. 2005.
22. **Raul Savimaa.** Modelling Emergent Behaviour of Organizations. Time-Aware, UML and Agent Based Approach. 2005.
23. **Raido Kurel.** Investigation of Electrical Characteristics of SiC Based Complementary JBS Structures. 2005.
24. **Rainer Taniloo.** Ökonoomsete negatiivse diferentsiaaltakistusega astmete ja elementide disainimine ja optimeerimine. 2005.
25. **Pauli Lallo.** Adaptive Secure Data Transmission Method for OSI Level I. 2005.
26. **Deniss Kumlander.** Some Practical Algorithms to Solve the Maximum Clique Problem. 2005.
27. **Tarmo Veskiõja.** Stable Marriage Problem and College Admission. 2005.
28. **Elena Fomina.** Low Power Finite State Machine Synthesis. 2005.
29. **Eero Ivask.** Digital Test in WEB-Based Environment 2006.
30. **Виктор Войтович.** Разработка технологий выращивания из жидкой фазы эпитаксиальных структур арсенида галлия с высоковольтным р-п переходом и изготовления диодов на их основе. 2006.
31. **Tanel Alumäe.** Methods for Estonian Large Vocabulary Speech Recognition. 2006.
32. **Erki Eessaar.** Relational and Object-Relational Database Management Systems as Platforms for Managing Softwareengineering Artefacts. 2006.
33. **Rauno Gordon.** Modelling of Cardiac Dynamics and Intracardiac Bio-impedance. 2007.
34. **Madis Listak.** A Task-Oriented Design of a Biologically Inspired Underwater Robot. 2007.
35. **Elmet Orasson.** Hybrid Built-in Self-Test. Methods and Tools for Analysis and Optimization of BIST. 2007.
36. **Eduard Petlenkov.** Neural Networks Based Identification and Control of Nonlinear Systems: ANARX Model Based Approach. 2007.

37. **Toomas Kirt**. Concept Formation in Exploratory Data Analysis: Case Studies of Linguistic and Banking Data. 2007.
38. **Juhan-Peep Ernits**. Two State Space Reduction Techniques for Explicit State Model Checking. 2007.
39. **Innar Liiv**. Pattern Discovery Using Seriation and Matrix Reordering: A Unified View, Extensions and an Application to Inventory Management. 2008.
40. **Andrei Pokatilov**. Development of National Standard for Voltage Unit Based on Solid-State References. 2008.
41. **Karin Lindroos**. Mapping Social Structures by Formal Non-Linear Information Processing Methods: Case Studies of Estonian Islands Environments. 2008.
42. **Maksim Jenihhin**. Simulation-Based Hardware Verification with High-Level Decision Diagrams. 2008.
43. **Ando Saabas**. Logics for Low-Level Code and Proof-Preserving Program Transformations. 2008.
44. **Ilja Tšahhiov**. Security Protocols Analysis in the Computational Model – Dependency Flow Graphs-Based Approach. 2008.
45. **Toomas Ruuben**. Wideband Digital Beamforming in Sonar Systems. 2009.
46. **Sergei Devadze**. Fault Simulation of Digital Systems. 2009.
47. **Andrei Krivošei**. Model Based Method for Adaptive Decomposition of the Thoracic Bio-Impedance Variations into Cardiac and Respiratory Components. 2009.
48. **Vineeth Govind**. DFT-Based External Test and Diagnosis of Mesh-like Networks on Chips. 2009.
49. **Andres Kull**. Model-Based Testing of Reactive Systems. 2009.
50. **Ants Torim**. Formal Concepts in the Theory of Monotone Systems. 2009.
51. **Erika Matsak**. Discovering Logical Constructs from Estonian Children Language. 2009.
52. **Paul Annus**. Multichannel Bioimpedance Spectroscopy: Instrumentation Methods and Design Principles. 2009.
53. **Maris Tõnso**. Computer Algebra Tools for Modelling, Analysis and Synthesis for Nonlinear Control Systems. 2010.
54. **Aivo Jürgenson**. Efficient Semantics of Parallel and Serial Models of Attack Trees. 2010.

55. **Erkki Joason.** The Tactile Feedback Device for Multi-Touch User Interfaces. 2010.
56. **Jürjo-Sören Preden.** Enhancing Situation – Awareness Cognition and Reasoning of Ad-Hoc Network Agents. 2010.
57. **Pavel Grigorenko.** Higher-Order Attribute Semantics of Flat Languages. 2010.
58. **Anna Rannaste.** Hierarcical Test Pattern Generation and Untestability Identification Techniques for Synchronous Sequential Circuits. 2010.
59. **Sergei Strik.** Battery Charging and Full-Featured Battery Charger Integrated Circuit for Portable Applications. 2011.
60. **Rain Ottis.** A Systematic Approach to Offensive Volunteer Cyber Militia. 2011.
61. **Natalja Sleptšuk.** Investigation of the Intermediate Layer in the Metal-Silicon Carbide Contact Obtained by Diffusion Welding. 2011.
62. **Martin Jaanus.** The Interactive Learning Environment for Mobile Laboratories. 2011.
63. **Argo Kasemaa.** Analog Front End Components for Bio-Impedance Measurement: Current Source Design and Implementation. 2011.
64. **Kenneth Geers.** Strategic Cyber Security: Evaluating Nation-State Cyber Attack Mitigation Strategies. 2011.
65. **Riina Maigre.** Composition of Web Services on Large Service Models. 2011.
66. **Helena Kruus.** Optimization of Built-in Self-Test in Digital Systems. 2011.
67. **Gunnar Pihõ.** Archetypes Based Techniques for Development of Domains, Requirements and Software. 2011.
68. **Juri Gavšin.** Intrinsic Robot Safety Through Reversibility of Actions. 2011.
69. **Dmitri Mihhailov.** Hardware Implementation of Recursive Sorting Algorithms Using Tree-like Structures and HFSM Models. 2012.
70. **Anton Tšertov.** System Modeling for Processor-Centric Test Automation. 2012.
71. **Sergei Kostin.** Self-Diagnosis in Digital Systems. 2012.
72. **Mihkel Tagel.** System-Level Design of Timing-Sensitive Network-on-Chip Based Dependable Systems. 2012.
73. **Juri Belikov.** Polynomial Methods for Nonlinear Control Systems. 2012.

74. **Kristina Vassiljeva.** Restricted Connectivity Neural Networks based Identification for Control. 2012.

75. **Tarmo Robal.** Towards Adaptive Web – Analysing and Recommending Web Users` Behaviour. 2012.

76. **Anton Karputkin.** Formal Verification and Error Correction on High-Level Decision Diagrams. 2012.