Agent-Oriented Software Engineering Models

REDUCING SEMANTIC GAP IN DEVELOPMENT PROCESS OF MANAGEMENT INFORMATION SYSTEMS FOR VIRTUAL ORGANIZATIONS

Jacek Jakieła, Paweł Litwin, Marcin Olech

Abstract: *The paper describes experience gained by developing an agent-oriented methodology suitable for design and implementation process of Management Information Systems for modern business structures called virtual organizations. It starts with description of semantic gap problem, shows main concepts related to agency and virtual organizations and describes similarities between virtual organizations and multi-agent systems. Next we shortly present the state-of-the-art of currently used methodologies aimed at inter-organizational modeling and show motivations which have become the rationale for our approach. What is more we present the main methodology assumptions made, international standards the methodology follows and the framework we have developed so far. The paper ends with conclusions and further research plans.*

Keywords*: business modeling, agent-oriented development methodology, agent-oriented-software engineering, virtual organization, semantic gap, agent-oriented management information systems*

ACM Classification Keywords*: I. Computing Methodologies; I.2 Artificial Intelligence; I.2.11 Distributed Artificial Intelligence; Multi-Agent Systems*

Introduction

Nowadays business organizations are becoming increasingly complex systems. This complexity is well visible during the development process of information system supporting modern organizations' activities. Organizations are no more monolithic structures with clear boundaries and areas of operations, well defined according to functional hierarchies. Modern business structures can take different forms. Enterprise can consist of distributed independent organizations with physical presence, that share resources to achieve common goals [Franke 2002], or can be fully virtual and operate primarily via electronic means e.g. various forms of business webs [Tapscott et al. 2000]. Therefore the structural and behavioral characteristics of business firms have profoundly changed. All these changes have to be taken into consideration during the process of designing the management information systems.

Apart from the complexity of business domain there is also the problem of complexity of software, which is its essential property, not an accidental one. This inherent complexity derives from such elements as: the difficulty of managing the development process, the flexibility possible through software, and the problems of characterizing the behavior of discrete systems [Brooks 1995, Booch et al. 2007].

The third problem is that contemporary methodologies for software development are not equipped with modeling methods for preparing system specification including all new characteristics of business problem domains. As a result semantic gap arises between business models and software models used for implementation of management information system. This gap is the source of many problems during the process of transforming business specification into the software architecture. The main problem is that many aspects of organization operations that should be supported by system under development are not taken into account during development process and therefore software does not support properly business goals of the enterprise. As there will be shown in the paper, the solution of this problem is to use appropriately selected modeling concepts for all of the stages of software development. The concepts defined for every stage have to have high semantic proximity, and thanks to this, the process of transforming business model into software model is an intuitive and unambiguous mapping of specifications' artifacts.

The paper presents the skeleton of management information system development methodology that has two main goals. Firstly it improves the process of business and software complexity management. Secondly it helps to reduce the semantic gap between business and system specifications prepared during software development process. The proposed methodology is based on the concept of software agent and its characteristics which are used as modeling constructs. As will be shown, such approach enables to better manage the complexity of modeling process. What is more the semantic proximity of the agent and modern organizations' characteristics will lead to significant reduction of semantic gap between modeling artifacts used at different stages of development process.

The Semantic Gap Problem

The semantic gap characterizes the difference between two descriptions of an object by different linguistic representations. In computer science, the concept is relevant whenever human activities, observations, and tasks are transferred into a computational representation. More precisely the gap means the difference between ambiguous formulation of knowledge related to the application domain in a business specification and its computational representation in a formal language – at first system specification and then programming language.

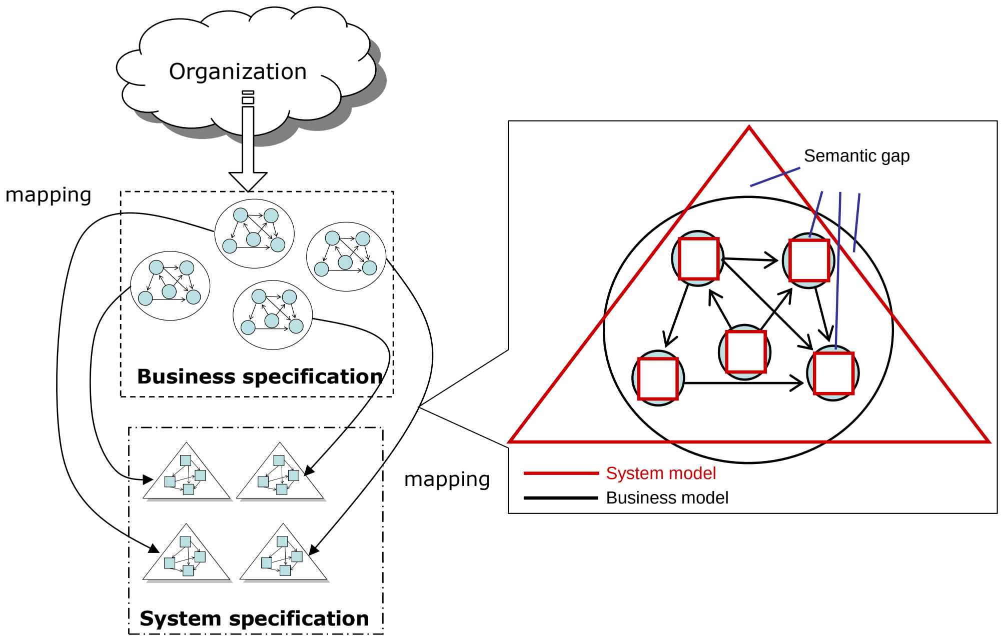

Fig. 1. Semantic Gap Problem

It is a fundamental task of software engineering to close the gap between application specific knowledge and technically doable formalization. For this purpose domain specific (high-level) knowledge must be transferred into algorithms and their parameters (low-level) [Dorai, Venkatesh 2003]. When developing management information system there are two main analysis and design perspectives: business system perspective and software system perspective. In order to describe application domain, analyst with stakeholders prepare business model that formalizes all important aspects of organization's operations and structure. It typically includes organization chart, strategy related documents (vision, mission and goals statements), business processes models and data models.

This constitutes the basis for application domain analysis and requirements elicitation as well as specification. In the next stage the system specification is derived from business specification. It includes the software architecture and functionalities description as well as all the models related to static and dynamic aspects of the system under development. The main question here is *how to express business artifacts in terms of implementation constructs?* At this point the semantic gap problem usually arises because every perspective uses its own set of concepts – different for business and software modeling, and therefore there is no intuitive and unambiguous mapping between business and software mindsets. The problem is visualized on the figure 1. In the following sections there will be shown how to tackle this problem. The cornerstone of proposed solution is to base development methodology on an agent concept.

The Essence of Agency

Before advantages of agent oriented organization modeling and system development will be presented, it seems advisable to explain the essence of agency and define two main concepts our methodology is based on – an *agent* and a *multi-agent system*.

The Concept of Agent

Over the last two decades the concept of an intelligent agent has become really popular. A number of researchers dealing with artificial intelligence domain focused on agency. Consequently numerous definitions of an agent have been coined. Two of them are provided below.

Michael Wooldridge and Nicholas R. Jennings [Wooldridge, Jennings 1995] describe an agent as: *a hardware or (more usually) software-based computer system that enjoys the following properties*:

- *autonomy*: agents operate without the direct intervention of humans or others, and have some kind of control over their actions and internal state;
- *social ability*: agents interact with other agents (and possibly humans) via some kind of agent-communication language, which enables to exchange their knowledge;
- *reactivity*: agents perceive their environment, (which may be the physical world, a user via a graphical user interface, a collection of other agents, the INTERNET, 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.

Another definition has been proposed by S. Franklin and A. Graesser in their paper attempting to distinguish software agents from regular computer programs [Franklin, Graesser 1996]: "*An autonomous agent is a system situated within and a part of an environment that senses that environment and acts on it over time, in pursuit of its own agenda and so as to effect what it senses in the future.*"

Based on definitions presented few vital attributes of an agent can be abstracted:

- an agent exists in a certain *environment* and thus it ceases to be an agent when extracted from such environment,
- an agent *senses its environment*, *acts on this environment* and its *actions can affect what an agent will sense in the future*,

- an agent *operates over time* and acts whenever it “feels” it’s necessary; unlike regular program which terminates once its mission is accomplished,
- an agent *operates autonomously* pursuing its own goals and *is able to undertake pro-active behavior*.

All these basic characteristics constitute conceptual framework that will be used later when trying to show how agent oriented methodology may help to improve complexity management and reduce semantic gap during management information system development process.

The Concept of Multi-Agent System

A Multi-Agent System may be defined as a set (society) of decentralized software components (where every component exhibits the properties of an agent, mentioned in the previous section), that are carrying out tasks collaboratively (often in parallel manner) in order to achieve a goal of the whole society. Later in the paper this definition has been disaggregated and all the properties are used to show why the multi-agent system can be considered as a very intuitive virtual organization modeling metaphor.

As presented definition reveals, software agents have the ability to collaborate with each other what enables the creation of multi-agent systems. Collaboration is defined as a process in which society coordinate its actions in order to achieve common goals. Software agents are able to collaborate with one another as well as human agents.

The corner–stones of inter–agent collaboration are: *communication and knowledge sharing*. *Communication* is basically an exchange of information among agents (agents can send messages to each other, observe each other’s state and behavior, however, communication takes place on the knowledge level). To enable knowledge sharing agents must have common goals and decompose the process of achieving these goals into sequence of actions providing that every agent is capable of performing task assigned to it.

Inter–agent collaboration requires also a *communication language*. Currently the most popular agent communication languages are: Knowledge Query and Manipulation Language (KQML) developed in early 90’s and FIPA-ACL developed by Foundation for Intelligent Physical Agents. Both rely on speech acts theory and define a set of performatives, their meaning and protocol for perfomatives exchange.

Although there are many frameworks and agent architectures developed so far by AI community, we have decided to base our methodology on *Belief-Desire-Intension* approach [Georgeff et al. 1999], which is most widely used framework for multi-agent systems development and offers implementation constructs that are semantically closest to virtual organization characteristics.

The Essence of Virtual Organization

Virtual organizations are (often temporary) value-added partnerships of independent, autonomous actors, such as individuals, companies or research institutes that have established a pre–partnership relationship in order to work together for achieving common goals. As we can see there is very close semantic proximity between basic characteristics of virtual organization and multi-agent system. More detailed insights will be presented in the next section.

Virtual organization can be viewed as a hub of partner firms that are selected according to an actual need in order to carry out a given task on a temporary basis. They can be partnerships of independent firms with physical presence. In such case every partner delegates specific unit of organization which constitutes the part of the virtual organization structure. It can also be the whole organization that takes part in such alliance with all departments and resources it possesses. Possible configuration of virtual organization is presented on figure 2.

Organizations may also take fully virtual forms. With the development of the Internet and modern ICT, new ways of conducting business have evolved. Many firms without physical presence are conducting business on electronic marketplaces. In such case virtual organization is a business web platform which provides the environmental condition, such as trust and coordination mechanisms and tools, necessary for the dynamic configuration of market and customer-driven value chain constellations [Franke 2002]. Therefore the task of designing organization boils down to developing the software (in the form of business web) that will support all operations of e-business partners related to extended supply chain (see figure 3).

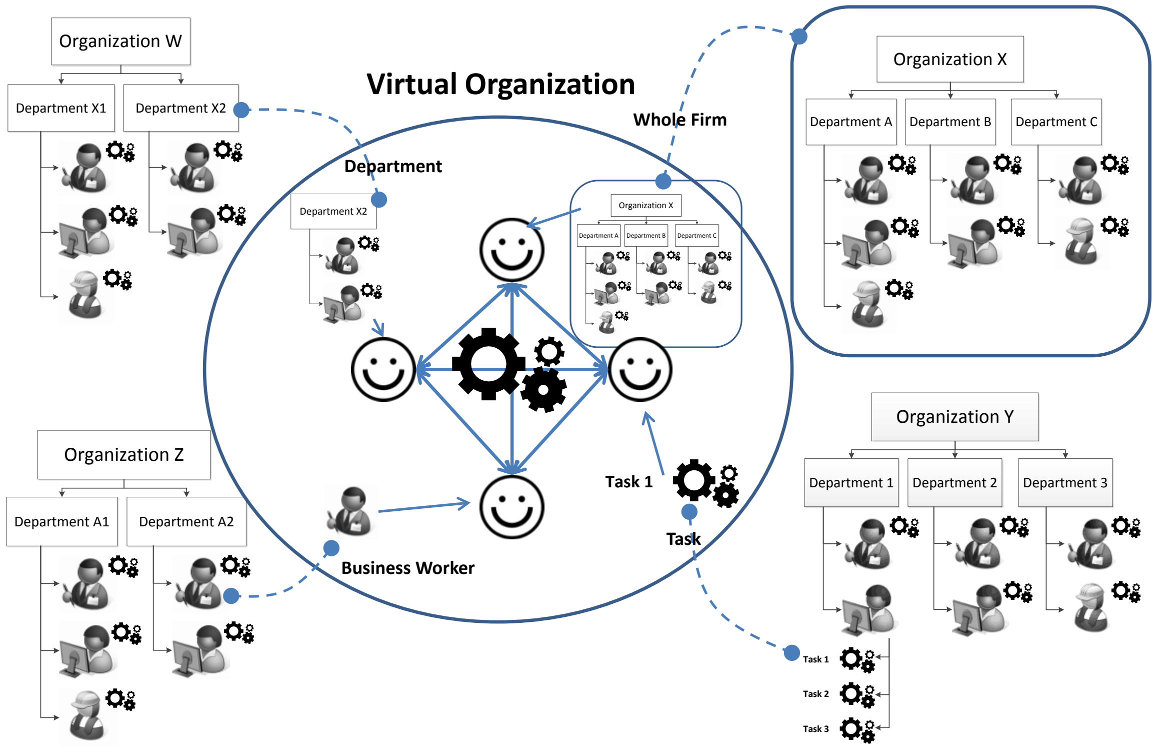

Fig. 2. The Structure of Virtual Organization

In both cases it is possible to abstract the common characteristics of virtual organizations. Most important are the following:

- process oriented organization structure rather than functional hierarchy,
- dynamic nature of organization structure that can change during its lifecycle,
- high autonomy of partners that constitute virtual organization,
- every business partner is self-contained what means that it has all needed competences and resources for conducting specific category of tasks,
- goal orientation, what means that all the tasks are organized toward achievement of common goals,
- physical distribution of partners.

As there will be argued later in the paper, in order to solve the semantic gap problem, all of the presented characteristics have to be taken into account when developing the software supporting operations of virtual organizations.

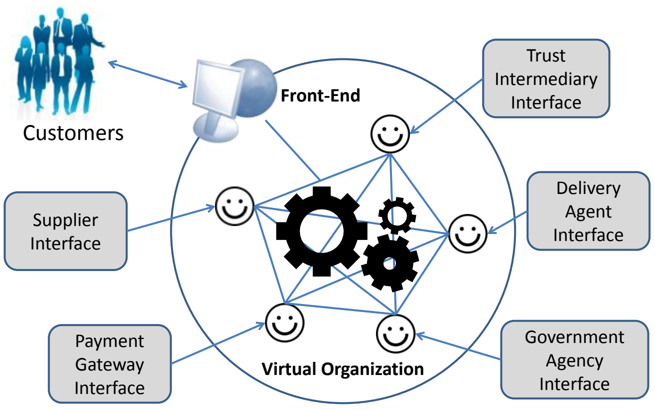
Customers
Front-End
Trust Intermediary Interface
Supplier Interface
Delivery Agent Interface
Payment Gateway Interface
Government Agency Interface
Virtual Organization

Figure 3. The Structure of Business Web Virtual Organization

Agent Paradigm as a Modeling Framework and Complexity Management Tool

When considering agent paradigm as a modeling framework it is essential to answer two fundamental questions. The first one is *why agent and multi-agent system characteristics make agents so natural and intuitive organizational modeling constructs*? The second is *why agent orientation is optimal choice for complexity management*?

One of the fundamental assumptions for our methodology is that business modeling process bases on multi-agent system metaphor. It leads to perceiving and understanding of virtual organization in the way typical for multi-agent system software engineering, but also takes under consideration business aspects including most important virtual organization characteristics.

The similarities between multi-agent systems and business organizations are particularly visible in case of virtual organizations. The table 1 presents observed similarities. In the first column there are basic structural and behavioral characteristics of multi-agent systems, and in the second characteristics of virtual organizations [Jakieła, Pomianek 2009].

Table 1. Resemblances between structural and behavioral characteristics of virtual organizations and multi-agent systems

Multi-agent system	Virtual organization
Multi-agent system is a set of **decentralized** software components.	**Virtual organization** is value-added partnership of **decentralized business actors**, such as individuals, companies or research institutes.
Multi-agent system is a set of **autonomous** software components.	Decentralization of partners that form virtual organization requires **autonomy delegation**. It has drastically changed the role of business actors, because “controlled positions” have been replaced by positions which give full competence. In case of **virtual organization** it is impossible to avoid situation when **partners**, who perform tasks related to common goals, **are fully autonomous entities**.
Multi-agent system is a set of **goal-oriented** software components.	Virtual organization is partnership of independent actors that **work together in order to achieve common goals**.

Multi-agent system is a set of **self-contained** software components that **may carry out tasks in parallel manner**.	Instead of artificial operations order, in virtual organization natural operation order is used. Business processes conducted by virtual organization are de-linearized. It allows for performance acceleration, because **tasks are performed in parallel by self-contained business partners**. Every partner carries out the tasks in the most effective and efficient manner because every virtual organization component possesses competences and resources needed for conducting specific category of activities.
The organization of multi-agent system **may change dynamically** depending on the current goals of society.	The structure of virtual organization **can change during its lifecycle**, depending on the current goals that have to be achieved.

Now let's move to the issue of complexity management. As Grady Booch says "*The task of the software development team is to engineer the illusion of simplicity*" [Booch et al. 2007], however as we have already mentioned, complexity management is a serious problem in case of modern organizations modeling and software development. In order to better explain how an agent paradigm, used as a mindset of our methodology, can improve complexity management it is useful to define the concept of complex system. Booch [Booch et al. 2007] relying on Simon's work [Simon, 1996] has defined the basic characteristics of complex systems:

- complexity frequently takes a form of hierarchy, where the system is composed of sub-systems connected with each other, which have their sub-systems, which in turn have their sub-systems and so on until the elementary level is reached. This hierarchy does not mean the superior-subordinate relation. Thanks to the fact, that complex systems are nearly decomposable we can fully understand them, describe or even perceive. Simon claims that it is highly probable that in reality only the systems that have a hierarchical structure can be understood [Simon 1996]. Looking at virtual organizations from this perspective, it is possible to distinguish such levels of hierarchy as organization actors level, business process level, singular organization level and specific configuration of few firms in a form of virtual organization (see figure 4),

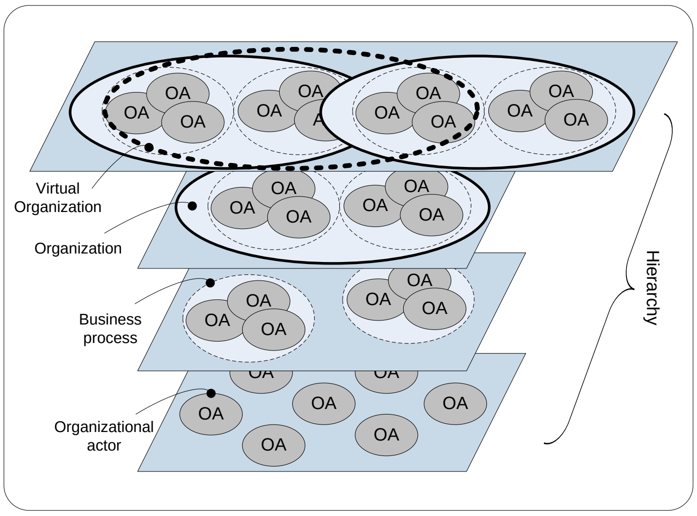

Fig. 4. Virtual Organization as a Complex System

- the choice which components of a system should be treated as elementary is arbitrary and depends on the system observer's decision,
- it is possible to identify interactions taking place between sub-systems as well as inside sub-systems, between their components, however interactions of the second type have one row higher frequency and are more predictable. The interaction frequency will differ depending on the level of hierarchy. For example, within a business organization more interactions will take place between employees working on the same process than between teams of employees working on different processes. The differences in interaction frequency within and between sub-systems allow decomposition and lead to the clear division between domains of analysis. In case of social systems, and undoubtedly every organization can be seen as such system, nearly decomposable character is clearly visible, therefore it is possible to exploit advantages of the decomposition method,
- complex systems are mostly sets of similar elements composed in various combinations. In other words there are certain common templates created on the basis of reuse of similar elementary components or more complex structures in the form of sub-systems,
- systems organized hierarchically tend to evolve over time, and hierarchical systems evolve faster than non-hierarchical ones. Simon claims that complex systems will evolve out of simple systems, if certain intermediary forms exist [Simon 1996].

Taking into consideration characteristics of complex systems as well as agent paradigm it is possible to show advantages of agent approach in the context of complexity management in the organization modeling as well as management information system development process [Jakieła 2006].

As the first argument it can be noticed, that agent oriented decomposition of a problem domain is an effective way to divide the problem space, while modeling organizations and information systems. It can be concluded from a number of factors. Firstly, hierarchical structure of complex systems causes, that modularization of organization components in terms of goals, that are to be achieved is a really intuitive solution. As Jennings and Wooldridge claim hierarchical organization of complex systems causes that at each level of the hierarchy, the purpose of the cooperation between sub-systems is achieving a functionally higher level. Whereas within sub-systems, components which these sub-systems are composed of, cooperate in order to achieve total functionality of a sub-system. As a consequence, decomposition oriented on goals that are to be reached is very natural division [Wooldridge, Jennings 2000]. Applying this schema to an organization the situation emerges where organization actors cooperate in order to achieve goals of the process, in turn processes are realized in order to achieve the goal of the specific firm, and firms combine their inherent competences in order to achieve the goals of virtual organization. It is worth to remember that goal orientation is one of the main characteristics of an agent and thus agent concept can be used without any additional effort.

Another vital issue is presentation of such characteristic of a modern organization as decentralization in the area of information processing and control. In this case agent oriented decomposition seems to be an optimal solution due to such characteristics of an agent as thread of control encapsulation in the form of autonomy property. The distributed organizational components may be thus modeled with autonomous agents as a basic modeling constructs.

Agent oriented approach allows also to solve problems connected with the design of interactions taking place between system components. It is a serious issue due to the dynamics of interactions between organization components. It is really frequent, that organization components enter an interaction in time which is difficult to predict and for unknown at the design stage reasons. As a consequence it's really challenging to predetermine parameters of such interactions. The solution to this problem is existence of system components that can make decisions concerning the type and range of interaction during runtime.

Another argument for an agent oriented approach is that it allows to eliminate semantic gap between agent abstraction used during the information system design phase and structures used during organization modeling. It is directly connected with similarities which appear between structural and behavioral characteristics of a multi-agent system and organization (table 1). Continuing this thread it is advisable to point out the following conveniences:

- Mutual interdependencies among organization actors and organization sub-systems can be naturally mapped into the system architecture in terms of high-level social interactions which take place among agents.
- In virtual organizations dependencies of this type are present in the form of really complex network of dynamically changing relationships. Agent based approach includes mechanisms which allow to describe such relations. For example, interaction protocols such as Contact Net Protocol can be used in order to dynamically create virtual organization structure, which can be, in case of such need, activated and after reaching particular goals deactivated. What is more, there are off-the-shelf structures, which can be used during the community modeling, what is really useful when modeling organization actors and sub-systems [Wooldridge, Jennings 2000].
- The process of organization modeling and system design frequently requires to perceive modeled object from the perspective of various abstraction levels, treating set of elements as atomic modeling structure. The idea of an agent is flexible enough to be used in an elementary level or on any detail level depending on the analyst’s needs. For example, an agent could be organization actor, department or whole organization and components treated as elementary interact only in an integrated form omitting details concerning intra-interactions.
- Organization modeling and system design with agent oriented approach leads to the structure, which has numerous stable intermediary forms, what is really important concerning complexity management. Among others it means that system components in the form of agents can be created rather independently and, in case of such a need, added to the system providing a smooth functionality growth.

After introduction to basic advantages of taken approach, the next section describes the methodology for management information system development, based on all the insights that have been presented so far.

The Skeleton of Development Methodology

Related Works and Methodology Motivations and Assumptions

Motivations for our work have been derived from detailed analysis of research concerning development methodologies for inter-organizational and virtual organizations management information systems [Huemer et al. 2008, Yu 1995, Mylopoulos et al. 2002, Zaborowski 2006, Mili et al. 2010, Telang et al. 2012]. The analysis discovered opportunities for improvements.

The first problem identified is the lack of guidelines for business modeling stage in the system lifecycle and in some methodologies this stage has not been taken into consideration at all. As best practices, prepared by Object Management Group show, business specification including all organization stakeholders’ needs and business goals, is a key determinant of quality of management information system under development.

Next important issue is the ease of methodology adoption to practical applications. We assumed that the key factor of fast methodology adoption is to make use of unified languages that are considered as international standards for business and software systems modeling. Unfortunately part of methodologies in use (e.g. COMMA, TROPOS) is based on non-standard, original notations what considerably decreases the speed of adoption for business and industrial applications and narrows the circle of prospect users.

Analysis of research works enabled to formulate the following motivations for our methodology:

- The designing of information systems for virtual organizations requires the detailed business model that describes structural and behavioral characteristics of application domain for which system is being developed.
- The business model has to be precisely mapped into architecture and functionalities of the software system that will support virtual organization operations.
- The system development should be supported by the process, analysis and design methods for business modeling and system implementation as well as unified modeling language adopted by software industry.

The motivations presented have been used for preparing the following methodology assumptions:

- Development methodology should be equipped with detailed business modeling stage that includes such aspects as business motivation model, business processes model, business rules model and organizational structure model. This enables to include in business specification the most important characteristics of application domain and considerably improves the process of system requirements elicitation and specification.
- All modeling methods developed as a part of methodology should be based on unified languages, what will increase the speed of its adoption to business and industrial applications.
- Methodology should be agent oriented what will enable to reduce the semantic gap between business and system requirements and facilitate the management of business and software modeling complexity. Agent orientation of the methodology means that all modeling concepts used are derived from agent paradigm and related to virtual organizations characteristics. What is more, during implementation stage management information system should be developed as a multi-agent software solution based on *Belief-Desire-Intension* (BDI) *architecture*.

Modeling Standards, Frameworks and Implementation Architectures Used

In order to increase the speed of methodology adoption it has been based on international modeling standards. All the standards were used as the meta-models that have been extended and adjusted for our methodology modeling methods. The following standards have been used:

- *Business Motivation Model* (BMM) – a standard developed by Object Management group which allows a business plan to be developed, communicated and managed in an organized manner. Business strategy is modeled in terms of Vision, Goals, Objectives, Mission, Strategies and Tactics, and internal as well as external Influences. These influences are then assessed to identify the potential impact they may have on the business. What is important all elements of the BMM are developed from a business perspective. The main idea is to create a business model for the elements of the business plan, before system design or technical development is begun. Thanks to this, the business strategy can become the foundation for system requirements specification and connects system solutions to their business intent [OMG 2013a].
- *Business Process Modeling Notation* (BPMN) – it is graphical notation that depicts the steps in business processes. BPMN depicts the end to end flow of a business process. The notation has been specifically designed to coordinate the sequence of processes and the messages that flow between different process participants in a related set of activities. BPMN is targeted at a high level for business users and at a lower level for process implementers. The business users should be able to easily read and understand a BPMN business process diagram. The process implementer should be able to adorn a business process diagram with further detail in order to represent the process in a physical implementation [OMG 2013b].
- *Unified Modeling Language* (UML) – this is an industry standard modeling language with a rich graphical notation, and comprehensive set of diagrams and elements. We have based all modeling

methods on UML notation. As we have already mentioned the rationale for this was to increase the speed of methodology adoption to business and industrial applications [OMG 2011].

- *Eriksson and Penker Business Patterns* – this is a set of UML extensions that enables to conduct business modeling with the use of UML notation. Part of these patterns is used as a foundation for organization layer in business modeling stage of our methodology [Eriksson, Penker 2000].
- *Belief-Desire-Intension Architecture* (BDI) – is one of the major approaches to building agents and multi-agent systems, including commercial agent software. It is inspired by logics and psychology. The main idea is to build agents using symbolic representations of agents' beliefs, desires, and intentions. It provides a mechanism for separating the activity of selecting a plan (from a plan library) from the execution of currently active plans. Consequently, BDI agents are able to balance the time spent on deliberating about plans (choosing what to do) and executing those plans (doing it) [Borodini et al. 2007, Georgeff et al. 1999]. We used this architecture as a foundation for implementation discipline. According to methodology assumptions presented before, the management information system is developed as a multi-agent software solution which conforms to BDI architecture.

Structure of Methodology

Our methodology has multi-layer organization. Its structure may be presented in two dimensions – *static* and *dynamic*. Static structure describes how process elements are logically grouped into core process disciplines. Basic process elements are: modeling methods, disciplines, artifacts, and roles. Dynamic structure shows how the process, expressed in terms of cycles, phases, iterations, and milestones, unfolds over the lifecycle of a project (figure 5). We borrowed dynamic structure from *Rational Unified Process* (RUP), which defines four main phases: inception, elaboration, construction and transition. In *inception phase* a good understanding of what system to build is gotten. It is done by getting a high-level understanding of all the requirements and defining the system's scope. In this stage the focus is also on mitigating business risks, and producing the business case for building the system. Finally it is important to get acceptance of all stakeholders and decide whether to proceed with the project. During *elaboration phase* most technically difficult tasks such as: design, implementation, testing, and baselining an executable architecture (including subsystems, their interfaces, key components, and architectural mechanisms) are undertaken. What is more, major technical risks are addressed by code implementation and validation [Barnes 2007].

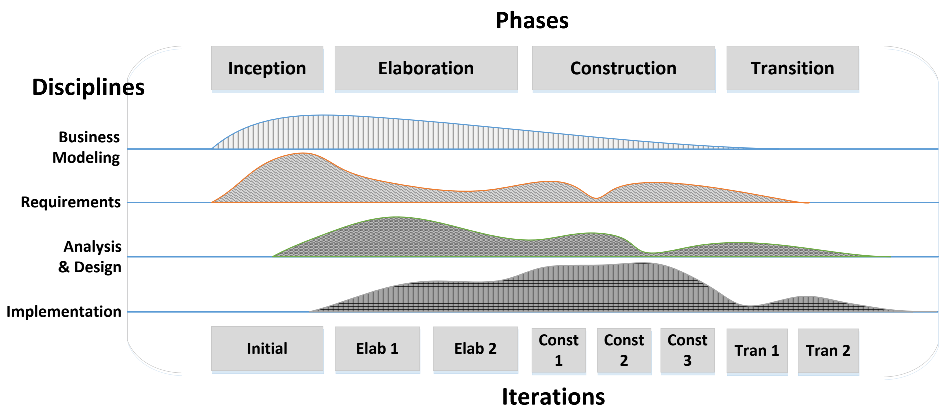

Fig. 5. The Agent Oriented Development Methodology Dynamic Structure

Most of the implementation is done during *construction phase*. Programmers are developing first operational version of the system on the basis of executable architecture. Then they deploy alpha releases to verify if system under development meets stakeholders' needs. At the end of this stage fully functional beta version is deployed, however system still requires improvements and tuning related to overall functional and non-functional requirements as well as quality.

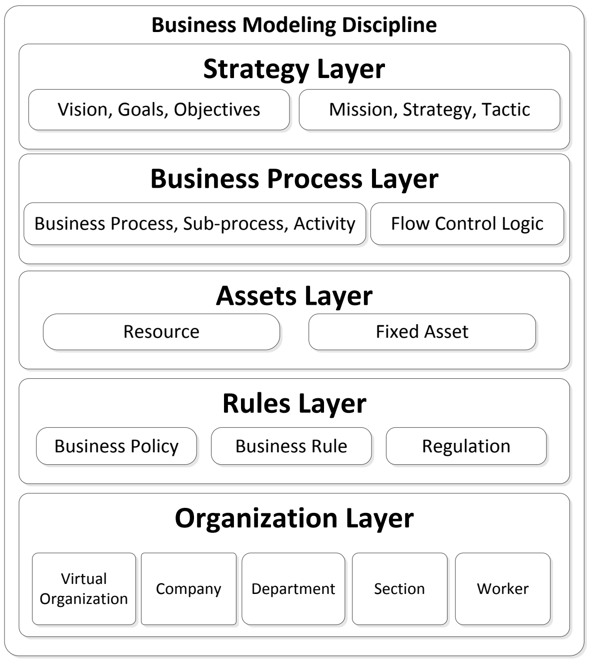

Fig. 6. Structure of Business Modeling Discipline

Main aim of the *transition stage* is to collect final feedback and ensure the release of the system under development addresses needs of all stakeholders. During this stage testing and minor adjustments are made. Basic activities include fine-tuning of the product, configuration and usability analysis. Focus is also on users training and integration issues.

Our original contribution is related to static structure of the methodology. We have extended business modeling, requirements as well as analysis and design disciplines. What is more, according to methodology assumptions, we have made implementation discipline agent oriented and prepare the transformations of design artifacts to implementation constructs. The figure 6 presents our approach in the area of the business modeling discipline.

Every layer is responsible for modeling specific aspect of the virtual organization. They are as follows:

- *Strategy layer* is split into two sections: Ends and Means. Ends section is used for describing state the virtual organization wants to achieve. This section includes such elements as: Visions, Goals and Objectives. Means section describes what courses of action need to be taken and how should be executed. In this section there are store elements like Missions and Courses of Action expressed in terms of Strategies and Tactics. Modeling concepts in the strategy layer have been adapted from Business Motivation Model. Elements of the strategy layer are visually modeled with the use of UML extensions developed for every modeling concept.
- *Business Process layer* is meant to model business processes, sub-processes, activities and finally the logic of control flows. Elements of the business process layer are visually modeled with Business Process Modeling Notation (BPMN) constructs.
- *Assets layer* is responsible for modeling all the resources and fixed assets which are used by virtual organization business processes. Concepts in the assets layer are based on BMM and are visually modeled with UML extensions.

- *Business Rules layer* is used for modeling directives which govern or guide business processes. There are three categories of directives: *Business Policy, Business Rules and Regulations*. Concepts in the business rules layer are based on BMM and its visual notation is developed in the form of UML extensions.
- *Organization layer* is meant to model the structure of the virtual organization. It includes such elements as: workers, sections, departments and companies that form virtual organization. Concepts in this layer are based on Eriksson & Penker patterns visually modeled with extended UML language.

All layers are interconnected and therefore it is possible to trace any artefact between layers. For every layer modeling methods have been developed according to framework published in [Mayer et al. 1995]. Every modeling method is described in the structure consisted of name, method definition (concepts and motivation), discipline (dictionary, grammar, detailed procedure) and use (how the method is used in the system development process).

Integral part of every layer is a visual modeling language. The notation of the language has been based on UML and concepts derived from standards presented before. The visual language was developed for every layer for both business modeling and analysis and design disciplines with the use of meta-modeling approach provided by OMG [OMG 2013c]. Sample meta-model of modeling language created for *strategy layer* and *means section* is presented on figure 7.

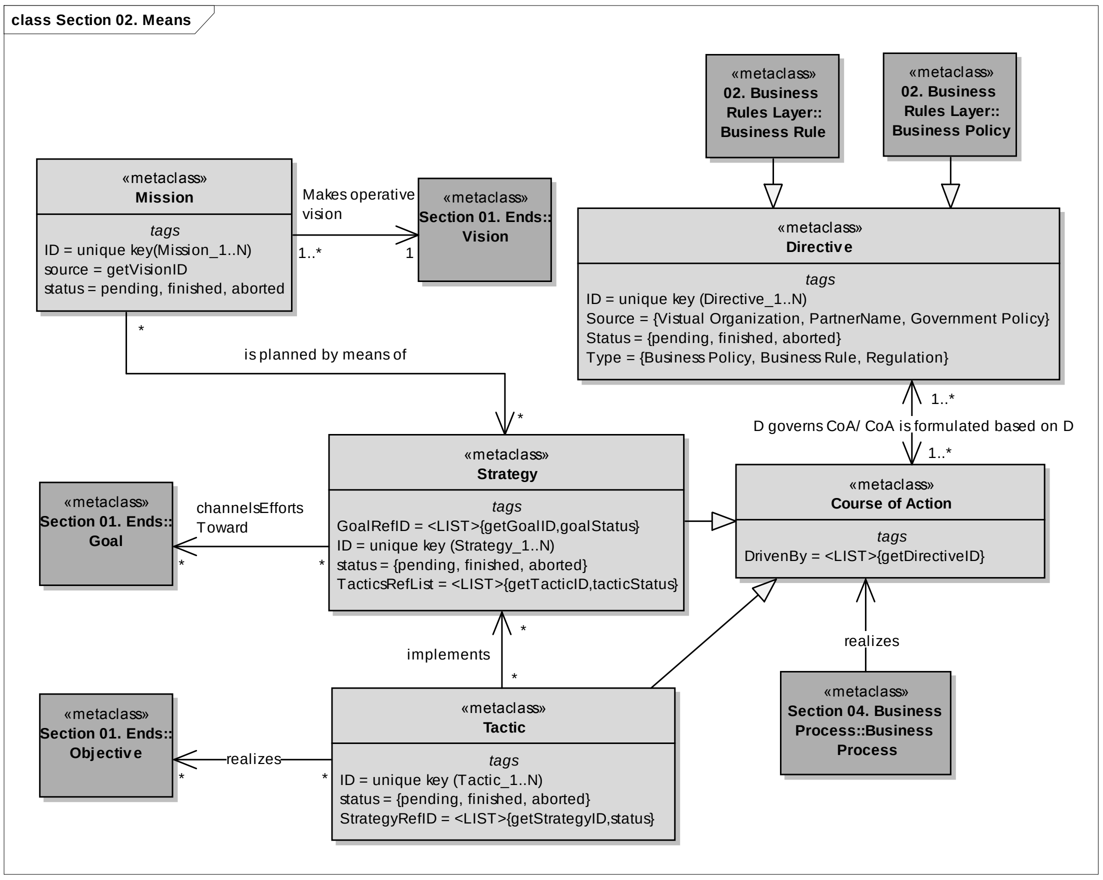
class Section 02. Means
«metaclass» 02. Business Rules Layer:: Business Rule
«metaclass» 02. Business Rules Layer:: Business Policy
«metaclass» Mission
tags
ID = unique key(Mission_1..N)
source = getVisionID
status = pending, finished, aborted
Makes operative vision
1..*
1
«metaclass» Section 01. Ends:: Vision
«metaclass» Directive
tags
ID = unique key (Directive_1..N)
Source = {Vistual Organization, PartnerName, Government Policy}
Status = {pending, finished, aborted}
Type = {Business Policy, Business Rule, Regulation}
*
is planned by means of
*
1..*
D governs CoA/ CoA is formulated based on D
1..*
«metaclass» Strategy
tags
GoalRefID = <LIST>{getGoalID,goalStatus}
ID = unique key (Strategy_1..N)
status = {pending, finished, aborted}
TacticsRefList = <LIST>{getTacticID,tacticStatus}
«metaclass» Section 01. Ends:: Goal
channelsEfforts Toward
*
*
«metaclass» Course of Action
tags
DrivenBy = <LIST>{getDirectiveID}
implements
*
*
realizes
«metaclass» Section 04. Business Process::Business Process
«metaclass» Section 01. Ends:: Objective
realizes
*
*
«metaclass» Tactic
tags
ID = unique key (Tactic_1..N)
status = {pending, finished, aborted}
StrategyRefID = <LIST>{getStrategyID,status}

Fig. 7. Part of Visual Language Meta-model for Strategic Layer of Business Modeling Discipline

Elements filled with darker color comes from different sections/layers. Elements with brighter background are native to the current layer and section. As can be seen, basic element in the Means section is a *Mission*, which indicates the ongoing operational activity of the enterprise. Mission makes *Vision* operative. Every

Mission is planned by means of *Strategy*, which represent the essential *Course of Action* to achieve, *Ends — Goals* in particular. Every Strategy is implemented by *Tactic*, which is a Course of Action that represents part of the Strategy details. Tactics generally channel efforts towards *Objectives*. Every Strategy and Tactic is Course of Action which is formulated based on *Directive*. Directives indicate how the Courses of Action should, or should not be carried out — in other words, they govern Courses of Action. Specifically, a Directive defines constraints or liberates some aspect of an enterprise.

After all activities related to business modeling discipline planned for specific iteration are done, Analysis and Design discipline activities are carried out. The structure of Analysis and Design discipline is presented on figure 8.

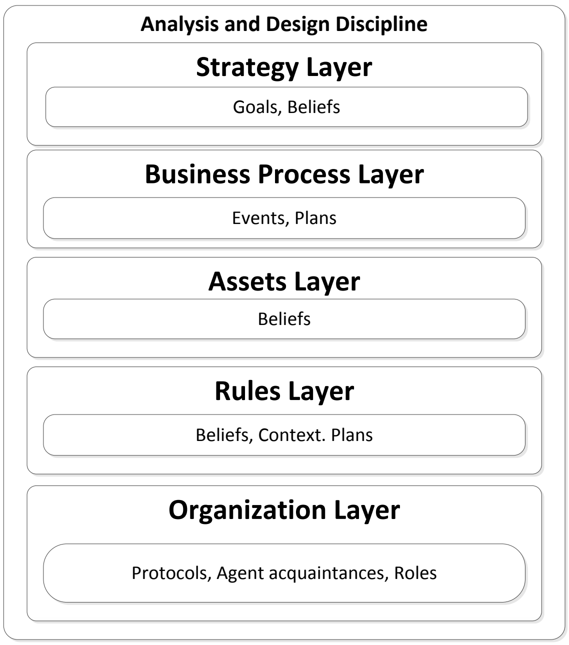

Fig. 8. The Structure of Analysis and Design Discipline

As has already been mentioned, analysis and design as well as implementation disciplines have been developed according to an agent paradigm. Therefore all the modeling concepts that are used for preparing agent oriented system specification conform to BDI architecture. What is more all the concepts are related to business modeling discipline concepts in such a way that there is a direct and unambiguous mapping between business and systems specifications. Finally implementation discipline uses the programming constructs from AgentSpeak language interpreted by Jason [Borodini et al. 2007].

Agent oriented modeling concepts are the following:

- *Beliefs*, which are used to represent information agent stores about environment, other agents and itself. Interesting fact about beliefs in Jason is that they are annotated and therefore may be maintained on the meta-level. There are three main annotations such as: percept, self and agent name. Percept is used to denote information from the agent sensors (received from environment). Self means that the belief is created by agent as mental note. Agent name suggests that source of the belief is other agent.
- *Goals* represent the state of affairs the agent strives for. The representation of goal is the same of a belief except that it is prefixed by exclamation mark.

- *Plans* constitute courses of action an agent will execute in order to achieve its goals or to react to changes in its environment. Every agent has the library of plans that determine its behavior. The plan is structured as presented below.

  ```
  triggering_event : context <- body.
  ```

  The *triggering event* represents event that will be handled by plan. *Context* describes circumstances under which the plan is suitable to handle the event. The *body* is a sequence of actions that will be executed or new goals for the agent to achieve. The agent behavior may change over time if new plans are acquired during the communication with other agents. Events result from changes in beliefs and goals. Beliefs may be updated and new goals set or received from other agents as a part of the delegation process. Events trigger execution of plans, provided that event matches the triggering event and is applicable in the time it is selected.
- *Protocols* specify rules of how all messages will be exchanged between agents. Agent acquaintances describe how agents are related to one another.
- *Roles* describe what agent is responsible for.

According to methodology assumptions all elements modeled in business modeling discipline have to be mapped into agent paradigm concepts in analysis and design discipline and finally implemented as management information system solution. What is also very important this conversion should be done in the way that reduces semantic gap between business and system aspects.

Figure 9 presents the mapping between modeling concepts. Elements from Organization Layer are represented using agents' roles, agents' acquaintances and composed protocols. Different resources modeled in Assets Layer are mainly described using agents' belief base. Concepts from Strategy Layer and Ends section are mapped into the agents' goals and beliefs. Concepts from Means section are mapped into agents' goals and plans. Business Policies, Business Rules and Regulations are used in plans mainly to check their context. Business processes are converted to events or messages and plans.

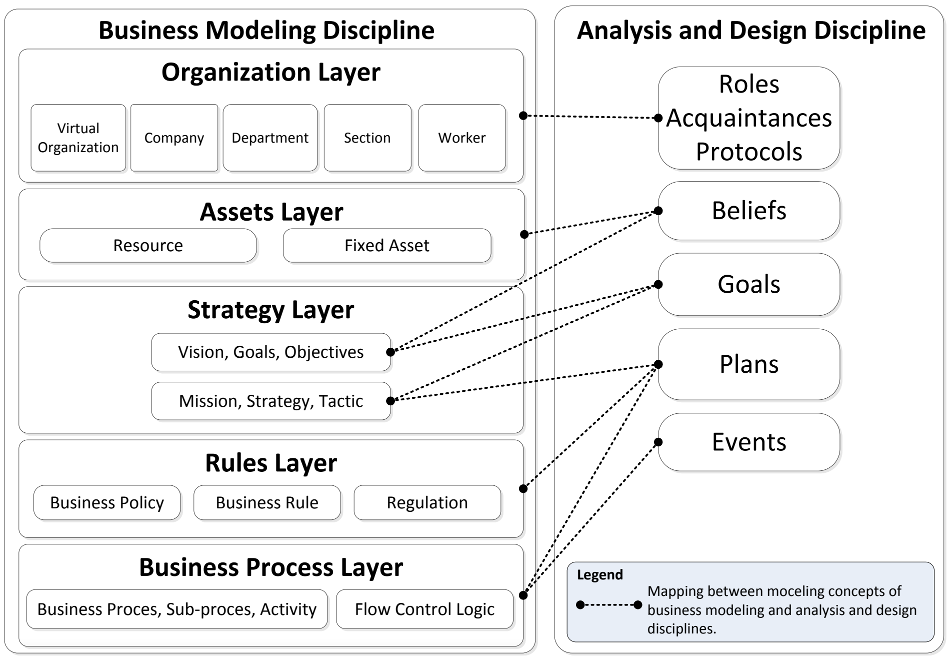

Fig. 9. Mappings between Modeling Concepts

Final development activities are related to implementation of the design model. This is done with *AgentSpeak* language and Jason interpreter. Because in Analysis and Design discipline all of the constructs come from BDI approach, the implementation boils down to the conversion of design model to agent oriented implementation constructs. However this is behind the scope of the paper as its aim was to present of how an agent paradigm may be used as tool for reducing semantic gap and improving development complexity management.

Conclusions and Further Research

High complexity and newly emerged characteristics of contemporary business structures require new approach to management information systems development. This approach should address to main problems: enable to reduce sematic gap between business model of application domain and design model of the system under development as well as improve the process of complexity management. Paper presents the solution in the form of agent oriented development methodology for systems supporting virtual organizations' operations. As was shown in the paper, agent orientation is an effective way to capture and include in business model the structure and behavior of modern enterprises. It is possible because of very high semantic proximity of agent paradigm modeling constructs and contemporary organizations characteristics. Our methodology is generic what means that it can be used in system development for organizations of any specificity and industry sector. Thanks to unified modeling language and process used in our methodology, it can be fast adopted to business and engineering applications. What is more, because of very detailed business modeling discipline it is possible to model all the important aspects of every organization and finally elicit the system features that will fully support virtual enterprise's business goals and objectives.

The methodology is still under development. The future research will concern the detailed meta-model for implementation discipline as well as automatic Model-Driven-Architecture transformations between specifications artifacts. However the methodology in its current state of development may be used in management information system prototype projects. This should be done in order to verify our approach and improve its meta-models of visual language and the process (disciplines).

Bibliography

[Barnes 2007] Barnes J.: Implementing the IBM Rational Unified Process and Solutions: A Guide to Improving Your Software Development Capability and Maturity. IBM Press, 2007.

[Booch et al. 2007] Booch G., Maksimchuk R., A., Engel M., W., Young B., J.: Object-Oriented Analysis and Design with Applications. Addison-Wesley, 2007.

[Borodini et al. 2007] Borodini R. H., Hubner J. F., Wooldridge M.: Programming Multi-Agent Systems in AgentSpeak Using Jason. Wiley, Chichester, 2007.

[Brooks 1995] Brooks F., P.: The Mythical Man-Month: Essays on Software Engineering. Addison-Wesley, 1995.

[Dorai, Venkatesh 2003] Dorai, C., Venkatesh, S.: Bridging the Semantic Gap with Computational Media Aesthetics. IEEE MultiMedia, 2003.

[Eriksson, Penker 2000] Eriksson H. E., Penker M.: Business modeling with UML: Business patterns at work. John Wiley & Sons, 2000.

[Franke 2002] Franke U., J.: Managing Virtual Web Organizations in the 21st Century: Issues and Challenges. Idea Group Publishing, 2002.

[Franklin, Greasser 1996] Franklin, S., Greasser, A.: Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents. University of Memphis, 1996.

[Georgeff et al. 1999] Georgeff, M., Pell B., Pollack M., Tambe M., Wooldridge M.: The Belief-Desire-Intention Model of Agency. Intelligent Agents V: Agents Theories, Architectures, and Languages. Spronger-Verlag, 1999.

[Huemer et al. 2008] Huemer Ch. et al.: The Development Process of the UN/CEFACT Modeling Methodology. 10th International Conference on Electronic Commerce (ICEC) '08 Innsbruck, Austria.

[Jakieła 2006] Jakieła, J.: AROMA – Agentowo zoRientowana metOdologia Modelowania orgAnizacji. WAEiI, Politechnika Śląska, Gliwice, 2006

[Jakieła, Pomianek 2009] Jakieła J., Pomianek B.: Agent Orientation as a Toolbox for Organizational Modeling and Performance Improvement. International Book Series "Information Science and Computing", Book 13, Intelligent Information and Engineering Systems, INFOS 2009, pp. 113-124, 2009.

[Jennings, Wooldridge 2000] Jennings N., R., Wooldridge M.: Agent-oriented software engineering. Proceedings of the 9th European Workshop on Modeling Autonomous Agents in a Multi-Agent World : Multi-Agent System Engineering, 2000.

[Mayer et al. 1995] Mayer, R., J., Crump, J., W., Fernandes, R., Painter, M., K., Keen A.: Information Integration for Concurrent Engineering. Compendium of Methods Report. Interim Technical Paper. Wright-Patterson Air Force Base. Ohio, 1995.

[Mili et al. 2010] Mili H. et al.: Business Process Modeling Languages: Sorting Through the Alphabet Soup, ACM Computing Surveys, Vol. 43, No. 1, Article 4, 2010.

[Mylopoulos et al. 2002] Mylopoulos J., Castro J., Kolp M.: Towards requirements-driven information systems engineering: the Tropos project, Information Systems 27 (2002) 365–389. Elsevier, 2002.

[OMG 2011] Object Management Group: Unified Modeling Language, ver. 2.4.1, August 2011.

[OMG 2013a] Object Management Group: Business Motivation Model, ver. 1.2b2, August 2013.

[OMG 2013b] Object Management Group: Business Process Model and Notation, ver. 2.0.2, December 2013.

[OMG 2013c] Object Management Group: Meta-Object Facility, ver. 2.4.1, June 2013.

[Simon 1996] Simon, H.: The Sciences of Artificial. MIT Press, 1996.

[Tapscott et al. 2000] Tapscott, D., Lowy, A., Ticoll, D.: Digital Capital: Harnessing the Power of Business Webs. Harvard Business Review Press, 2000.

[Telang et al. 2012] Telang R. P., Singh P. M.: Comma: A Commitment-Based Business Modeling Methodology and its Empirical Evaluation, Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012), Conitzer, Winikoff, Padgham, and van der Hoek (eds.), 4-8 June 2012, Valencia, Spain.

[Wooldridge, Jennings 1995] Wooldridge, M., Jennings, N., R.: Agent Theories, Architectures, and Languages: A Survey. Springer-Verlag, 1995.

[Yu 1995] Yu E.: Modelling Strategic Relationships for Process Reengineering, Ph.D. Thesis, University of Toronto, Department of Computer Science, Toronto, 1995

[Zaborowski 2006] Zaborowski M. „Model informacyjno-decyzyjny struktury danych o obiektach zarządzania", W: Kozielski St. i inni (red.) „Bazy danych. Modele, technologie, narzędzia. Analiza danych i wybrane zastosowania", WKŁ, 2006.

Authors' Information

Jacek Jakieła, Ph.D., Eng. – Department of Computer Science FMEA RUT; W. Pola 2, 35-959 Rzeszow, Poland; e-mail: jjakiela@prz.edu.pl

Major Fields of Scientific Research: Software Development Methodologies, Agent and Object-Oriented Business Modeling, Internet Enterprises Models, Computational Organization Theory and Multi-Agent Simulation of Business Architectures.

Paweł Litwin, Ph.D., Eng. – Department of Computer Science FMEA RUT; W. Pola 2, 35-959 Rzeszow, Poland; e-mail: plitwin@prz.edu.pl

Major Fields of Scientific Research: Applications of Neural Networks in Mechanics, Computer Simulations, Finite Element Method.

Marcin Olech, M.Phil., Eng. – Department of Computer Science FMEA RUT; W. Pola 2, 35-959 Rzeszow, Poland; e-mail: molech@prz.edu.pl

Major Fields of Scientific Research: Multiagent Based Simulations, Application of Artificial Intelligence in Industry.