29

Working Paper

Silesian University in Opava

School of Business Administration in Karvina

Institute of Interdisciplinary Research

Working Papers in Interdisciplinary Economics and Business Research

The Formalization of a Generic Trading Company Model Using Software Agents as Active Elements

Dominik Vymětal, Sohei Ito

April 2016

Working Papers in Interdisciplinary Economics and Business Research

Silesian University in Opava
School of Business Administration in Karviná
Institute of Interdisciplinary Research
Univerzitní nám. 1934/3
733 40 Karviná
Czech Republic
http://www.iivopf.cz/
email: iiv@opf.slu.cz
+420 596 398 237

Citation

VYMĚTAL, D. and ITO, S., 2016. The formalization of a Generic Trading Company Model Using Software Agents as Active elements. *Working Paper in Interdisciplinary Economics and Business Research no 29.* Silesian University in Opava, School of Business Administration in Karviná.

Abstract

Dominik Vymětal, Sohei Ito: **The Formalization of a Generic Trading Company Model Using Software Agents as Active Elements**

Business environment simulation often requires unique knowledge based on the modeler's experience. However, even experience based simulation models need some extent of abstraction and formalization in order to achieve results that are expected. Business process simulation models usually incorporate several essential components such as the model of trading functions that reflect customer behavior, procurement functions for modeling company inputs and the optimization of production & logistics functions. When modeling management decisions, a management function model with a loopback to company economic outputs is also needed. As a solid foundation of such complex business simulations, an abstract multi-agent architecture of a trading company model is proposed. The abstract model is inhabited with active entities – software agents and a method of registering their actions in a simulation run log is proposed. This approach combines the basic notions of abstract agent architectures with process mining methodology. Finally, the correctness of our software-agent system is verified and a validation is provided showing that the proposed system fits the real data company outputs.

Key words:

formal models, business process simulation, software agents, process mining, behavioral patterns

JEL: *C63, C88, D29, M15*

Contacts

Dominik Vymětal, Institute of interdisciplinary research, School of Business Administration in Karviná, Silesian University in Opava, Univerzitní náměstí 1934/3, Karviná, Czech Republic, e-mail: vymetal@opf.slu.cz

Sohei Ito, Department of Fisheries Distribution and Management, National Fisheries University, 2-7-1 Nagata-Honmachi Shimonoseki, Yamaguchi, Japan, e-mail: ito@fish-u.ac.jp

Introduction

Open economies, globalization, price minimization with product quality maximization and other trends result in keen competition due to significant increases in domestic and international business. It is obvious that if consumer demand abates, it may instantly cause serious troubles for any company, which enforces rapid management decisions. However, situations in which management has no relevant data to support their decisions are not rare. Various computer simulation models exist in the domain of artificial intelligence, which facilitate or even model management decision making. Simulations supporting decision support systems are typically based on business process modeling, a topic that has been treated by many researchers (Barnet, 2003; Bucki and Suchanek, 2012; Ericsson and Penker, 2000; Sperka et al., 2013; Suchanek and Vymetal, 2011; Van der Aalst, 2004). While analytical modeling approaches are based mostly on mathematical theories (Baden-Fuller and Haefliger, 2013; Liu and Triverdi, 2011), many research perspectives are based on experimental simulations using various generic business process models. Most business process modeling methods concentrate on the process and control flow (ARIS, 2000; BPMN, 2001). An alternative approach – value chain modeling based on e3-value method (Gordijn and Akkermans, 2002) and the REA (Resources, Events, Agents) ontology (Dunn, Cherrington and Hollander, 2004; Gailly and Poels, 2007; Hruby, 2006; Ito and Vymetal, 2013; McCarthy, 1982; Weigand and Elsas, 2012) can be seen as a complement to process control flow modeling.

However, typical business and value chain modeling methods tend to omit the variability of internal relations in business companies and within the markets. Here we have in mind, the forces of social interaction. The role of social forces within a company organization has already been described in many publications; here we mention a comprehensive one presented in McFarland and Gomez (2014). Social interaction elements in business modeling emerge in various business negotiations, on both the sales and the procurement side, specific management methods, disturbances from the company and in the market environment affecting the company results. This challenge in modeling can be met by incorporating local intelligence elements in the simulation model, leading to a new modeling paradigm – the multi-agent modeling approach (Macal and North, 2006; Odell, 2010; Wooldridge, 2009). Software agents are here defined as modeling entities that can act to some extent independently using their own targets, communicating with each other and coordinating their actions in order to achieve their aims and targets.

When comparing system modeling at the ERP design stage to simulation modeling, we can see differences. First, ERP architects will always use some design compromises in order to achieve speed and to use system resources in an economical way. Such compromises do not hinder the implementation of software but cause semantic differences among various software packages. Second, modeling at design stage is usually finished at a concept level, leaving the concrete steps of ERP functional design and parameterization to further project steps. Third, the ERP system model at the design stage depicts static properties of the system to be implemented.

Differently from system design modeling, business environment simulation requests unique knowledge of the target business processes rather than modeling purely based on end user needs and designers' experience with similar projects. While analytical modeling focuses on precise mathematical tools that often require specific treatment of process input data, the experimental simulations are often based on the experience of modelers. However, even experience based simulation models need some extent of abstraction and formalization in order to achieve expected results. We can observe that simulation models generally incorporate several essential components such as a model of trading function reflecting the customer behavior, procurement function that models company inputs and production and logistics optimization function. When modeling management decisions, a management function model with a loopback to company economic outputs is also needed. After the simulation model is designed, the verification of the model structure and possibly the validation of model outputs using real input data are necessary before it can be deployed in a company.

However, the unique knowledge mentioned above often brings some implicit assumptions in the simulation model that could affect the rigorous descriptions of the components modeled. Distinctly from ERP programming, the verification of agent-based simulation models should not use design compromises or semantic differences or tradeoffs between the concepts and performance. This is why we believe that some formal description of common simulation model features can help to improve the model structure and validation results.

For the formal simulation framework of business company models, we first construct an abstract architecture of a business company modeling also the general management decision functions. The proposed abstract architecture is then inhabited with active entities – software agents – implemented as our simulator MAREA. The abstract agent architecture proposed by FIPA describes the structure and communication rules of multi-agent systems, but in a rather static way. The multi-agent system proposed by Wooldridge (2009) which brings more dynamics into the matter still does not provide tools for the analysis of processes and outputs produced by agent interactions. Thus, in our abstract architecture, we exploit van der Aalst's process mining approach (Van der Aalst, 2004; Van der Aalst, Reijers and Song, 2005; Van der Aalst et al., 2009; Van der Aalst, 2011) through mapping our multi-agent model onto the process mining framework. That is to say, we combine classic abstract agent architecture with a process mining approach to make verification and validation feasible.

The last task is to verify that our multi-agent architecture is correct. Since this is in principle software verification, the rigorous verification is very difficult and is one of the ultimate goals in computer science. Thus we take another approach – software testing. Here the “correctness” of our architecture consists of the process structure, the agent interaction and fitness to real company data. Since we map the multi-agent architecture onto process mining approach, the verification of the first two can be easily done by analyzing the result of simulation (i.e. log) by process mining techniques. This “software testing via process mining techniques” is the advantage of our formalization of multi-agent architecture in the form of process mining framework. The fitness to real company data was validated by comparing the simulation results to the real company data.

In the next sections we first briefly present some related works, their relevance to our topic and the basic notions of a generalized company model. Section 2 introduces a formalized simulation model of a trading company using REA-based ERP and software agents as dynamic entities of the model. Next, a formalized approach to registration of processes in the simulation run resulting in simulation run log is presented. Section 3 briefly describes the method of model verification using functionalities of well-known ProM software as the verification tool and some results of model validation using real data in a case study. Concluding remarks present some discussion on the results and further research steps.

1. Basic Description of the Fundamental Notions

1.1. Software agents and process mining - short state of the art review

In order to enable the modeling of social behavior elements, we will use software agents. Software agents can be defined as software modules that are based on the agents' paradigm. However, no unique definition of software agents could be found in the research papers. Macal and North (2006) briefly defined the following basic features of agents:

- An agent is an identifiable, discrete individual governed by its behavior and decision-making capability;
- An agent is situated in an environment with which it interacts with other agents;
- An agent is goal-directed, autonomous and self-directed;
- An agent is flexible, and has the ability to learn and adapt her behaviors over time based on her experiments.

These features of agents can be seen as justification of their use in modeling complex economic environments like business.

Another definition of agents by Bellifemine et al. (2003) states that:

- Agents are autonomous - they can control their own actions and under circumstances can take decisions;
- Agents are proactive - they do not react in response only, but they can have own goal-oriented behavior and/or take initiative;
- Agents are social - they are able to interact with other agents in order to accomplish their task and achieve the overall goal of the system.

A comprehensive approach to multi-agent models was presented by Wooldridge (2009). His findings concern the single- and multi-agent definitions including formal specifications of the agent behavior, communication and co-operation and other agent properties. An important study on agents and their properties was presented by Odell (2010). He defines the software agent as an autonomous entity that can adapt to and interact with its environment. The notion of adaptation in this definition is important as it opens the way to formal agents 'definition using formal notion of possible worlds as an environment description.

Each modeling and simulation task includes model verification and validation which can be a very challenging issue. At least two main problem areas can be found here:

- The correspondence between the real data acquired from the real company processes and the simulation outputs, (model validation)
- The model structure verification that is the test, if the realized model and simulation corresponds with the abstract specification.

The validation task mainly concerns various real input and output data pre-processing methods, the outputs of which are needed for model parameterization and control. The aim here is to achieve sufficient quality of the model outputs allowing for eventual predictions. The verification task can be seen as the core of the model test itself. Without a correct simulation model structure consisting not only of "static" structures (the modeled structure items) but also of correct process structures, their interrelations and roles played by the process performers the correspondence between the simulation model output and real company output data cannot be achieved. To ensure this verification, the third paradigm of business process modeling can be advantageous – namely the principle of process mining most comprehensively described by van der Aalst (2011). Process mining is a discipline focusing on the extraction of information about processes from event (or transaction) logs. Event logs contain events, each event refers to an activity (activity is a step in the process), each event also refers to a case (i.e., a process instance), each event is instigated by an originator, and events are totally ordered. An event log is then a collection of events. The techniques, generally known as organizational mining, focus on the organizational perspective of process mining. Alongside typical information such as event, case, activity or the time when an event was performed, we can also find information about the originator (device, resource, or person) who initiated the event (activity). The events from the log file can be projected onto their resource and activity attributes. By using this approach, we can learn about people, machines, organizational structures (roles, departments), work distribution and work patterns.

In the simulation domain, the registered records of the simulation run give the modeler the possibility to study the resulting simulation log. Such a log with the proper structure presents the simulated process instances and their parameters such as process start and end timestamps, the participants taking part in the processes, the interrelationships of the processes in the model outputs and so on. Hence, it can be used for model structure evaluation, and, moreover for evaluation of participant relations within the simulated processes. Such participant relations are called "social networks". Recently, a new approach of social network analysis was presented by Slaninova, Martinovic, Sperka and Drazdilova (2013) and Slaninova (2014). By extracting sequences from simulation logs, similar behavior in process participants can be identified and latent ties between participant groups can be visualized. Hence, this approach can be seen as an alternative to the social network analysis performed by ProM (herein below).

The most prominent process mining tool is the open source software ProM, developed by Eindhoven University of Technology[1]. ProM is a comprehensive bundle of process mining software components using special formatted process log files as an input. It delivers various types of control flow diagrams such as Petri nets, social network diagrams like Working together, Subcontracting and others as outputs. Hence, the ProM outputs can be looked upon as ideal tools for model structure verification. The Petri nets, generated from the simulation log files, can be used for general simulation model structure verification, while the social network diagrams obtained from the analysis can help to verify and correct the actual process structure and process – participant relations.

However, using the software agents in the proposed model, the process logging ideas are to be complemented by considerations on architecture of agents taking part in the processes. A generalized approach using agents' formal description helps to avoid model discrepancies. Thus, our motivation was also to present a formalized approach to agent communication which is one of the basic agent properties.

The presented formalization combines the basic approach to process mining by Van der Aalst (2011) pp. 100 – 107 and the findings of Wooldridge, ibid, pp. 34 – 45) as far as agent

[1] http://www.processmining.org/

abstract architecture is concerned. In our solution it is proposed to represent the cases in the process mining sense by sequences of states of the environment caused by messages among agents. Then the messages of the agents are abstracted from in order to obtain suitable representations of the cases (traces) in the process mining sense.

1.2. Basic company model

In our exploration we will use a generic company design model and concentrate on the procurement and sales processes taking also some production and management questions into consideration.

The company model based on the generic business company presented herein below was a topic of other papers such as e.g. Vymetal and Schoeller (2012), Vymetal and Sperka (2013), Vymetal and Sperka (2014) and its more detailed description is beyond the scope of this paper.

The generic company model is presented in Figure. 1.

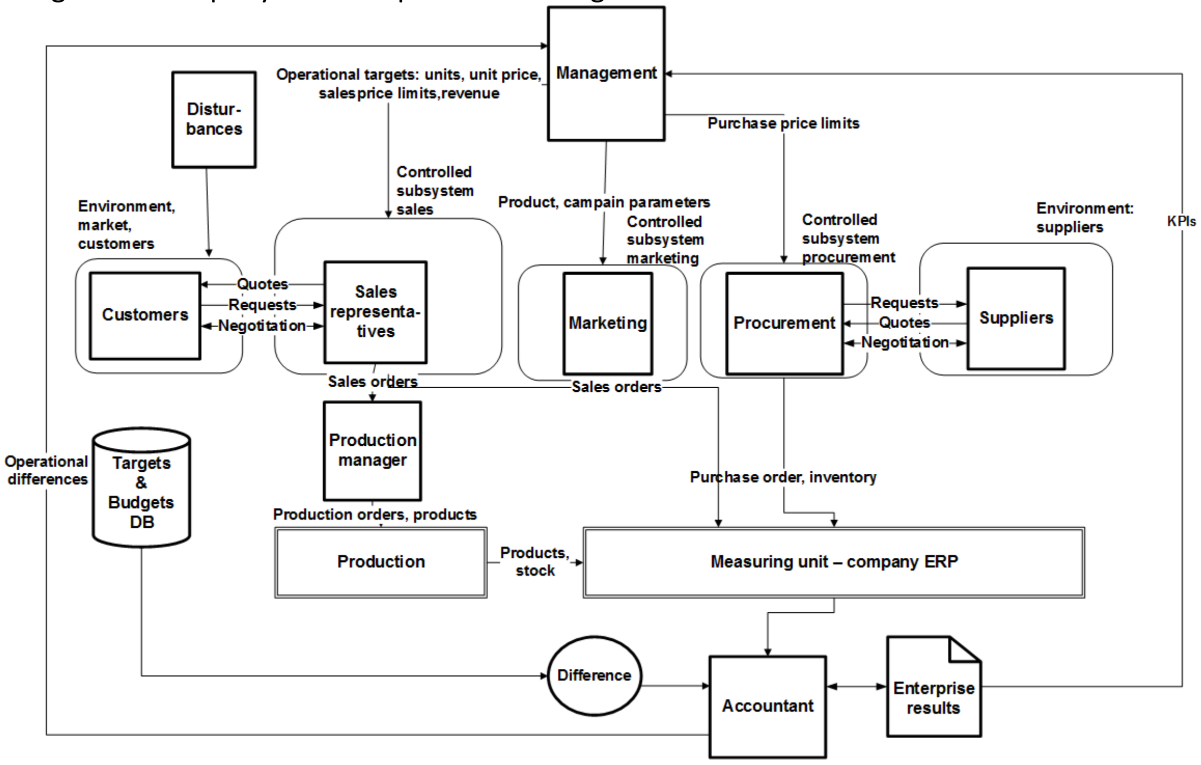


Fig. 1: Generic company model
(Source: own)

The company consists of controlling, measuring and controlled subsystems. The controlled subsystem sales realize the sales of products to customers. The customers communicate with the sales representatives. At the request of the customer, the sales representative sends a sales quotation which starts price negotiations. In case the negotiations end with an agreement a stock or production order and sales order are generated. After the product is shipped, the invoicing starts. The next controlled subsystem is the procurement subsystem. Procurement can be looked upon as the mirror image of sales. The purchase representatives place purchase requests to suppliers and after (possible) negotiation the purchase order is realized and the product or its components are put into stock. Another controlled subsystem is realizing marketing actions. If there is a production part in the company, it also belongs to a controlled subsystem. The value movements within sales, procurement, inventory, production and marketing are registered by the company ERP system. Using the ERP system, the accountants regularly evaluate the company results characterized by Key Performance Indicators (KPIs) and report the differences to management – the controlling subsystem. Based on KPI developments, the managers take decisions in order to keep the company system at the proximity of budgeted targets that is, in this sense the management realizes the back loop function of the management loop.

This simple model can be used for modeling at a higher level of differentiation. So, e.g. the modeling concentrates on sales process and negotiation modeling, customer decision modeling, procurement modeling etc. In Figure 2, an excerpt of the generic model depicting the sales process is presented.

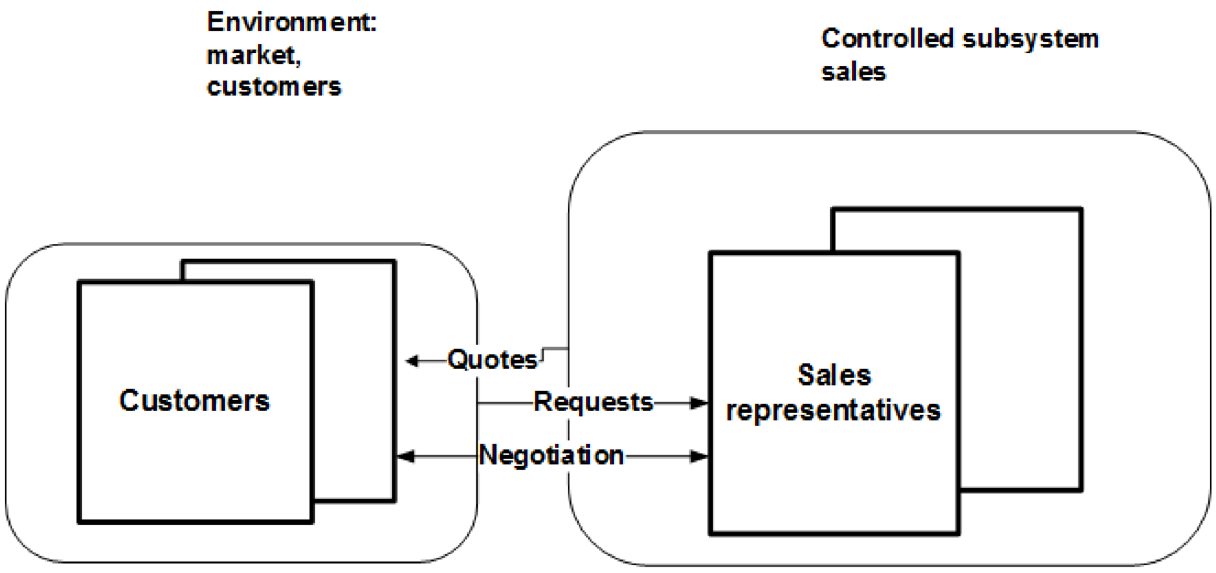


Fig. 2: Partial model – the sales process

(Source: own

The customer sends a sales request to the company represented by a sales representative asking for a product and a quantity to be offered. The sales representative then sends a sales quotation with some price. The customer decides whether to accept or reject the quotation. The decision is based on a decision function. There are several possibilities how to define a decision function. Basically, the decision function can be formally described as a function mapping various quotation prices and other parameters such as customer budget, customer product preferences, market conditions, sales representative ability to sell, quoted quantity, etc. to decision from $\{accept, reject, revoke\}$, where revoke is generated in case the negotiation time limit is exceeded.

In order to get a clear simulation model semantics, we define the negotiation formally:

Let decision domain DD be a set of tuples

$$\langle It, Pr, t, Pf \rangle \qquad (1)$$

$where$ It – quoted product, Pr – $quoted\ price, t - timestamp\ of\ the\ quotation\ and$ $Pf = \{pf_1, pf_2 \dots pf_n\}\ a\ set\ of\ parameters\ used\ for\ the\ decision, n \in \mathbb{Z},$ $n \neq 0\ is\ a\ finite\ number.$

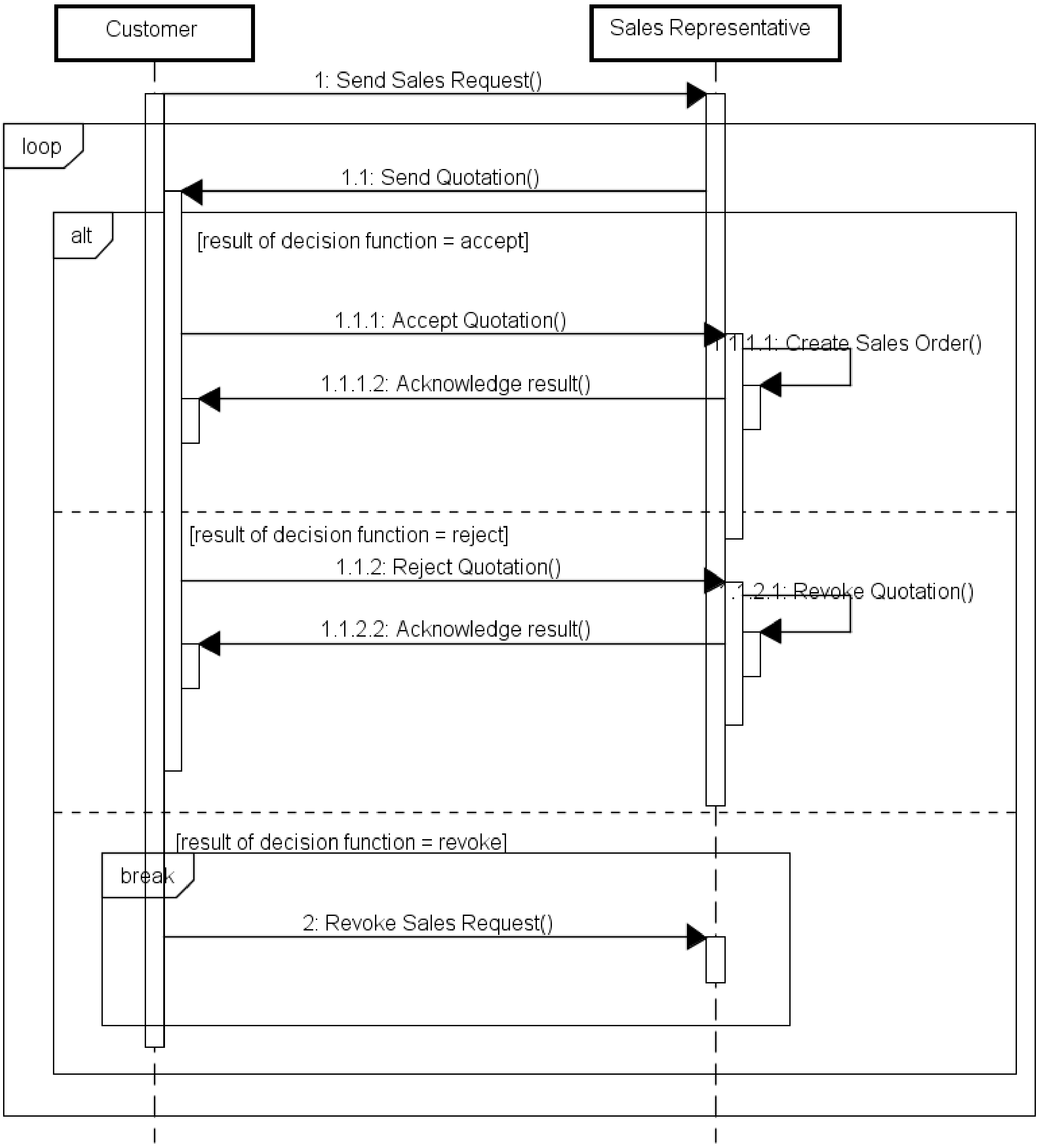
As examples of parameters used for decision we can mention e.g. sales representative ability to sell or customer preferences. Then the customers' decision function can be defined as

$Accept: DD \rightarrow \{accept, reject, revoke\}$

$$Accept(\langle It, Pr, t, Pf\rangle) = \begin{cases} F(\langle It, Pr, Pf\rangle), & if\ t < t_{max} \\ revoke & otherwise \end{cases}$$
$$F(\langle It, Pr, Pf\rangle) \rightarrow \{accept, reject\} \qquad (2)$$

where t_{max} is the set limit of the customer to wait a quotation. If the sales quotation is rejected (Accept = reject), the sales representative sends a new quotation with a new price. The price reduction rule is also a function, in this time based on the company general sales rules.

In case the sales quotation is accepted, the sales order is realized. The formalized price negotiation model defined in equations 1 and 2 is presented in Figure 3.



Customer
Sales Representative
1: Send Sales Request()
loop
1.1: Send Quotation()
alt
[result of decision function = accept]
1.1.1: Accept Quotation()
1.1.1.1: Create Sales Order()
1.1.1.2: Acknowledge result()
[result of decision function = reject]
1.1.2: Reject Quotation()
1.1.2.1: Revoke Quotation()
1.1.2.2: Acknowledge result()
[result of decision function = revoke]
break
2: Revoke Sales Request()

Fig. 3: Price negotiation

(Source: own

Generally, a similar negotiation takes place in the procurement subsystem. The role of the customer is played by the purchase representative, while the role of the sales representative is played by the suppliers. The purchase representative decision function is similar to the customer decision function. The instigation for the purchase request is a message from the ERP system. (Product reorder level and quantity)

The management takes its decisions based on the KPIs. KPIs represent the current state of the company. They show the "score" of the company at a certain point in time. Additionally, companies have their own *target* KPIs. Managers determine actions in order to achieve the target KPI at the end of every period. The target KPI at a company is fixed (at least for a certain duration, e.g. one year). So the manager compares the KPI at every period to the target KPI and decides the management action. Examples of KPIs are e.g. turnover, gross profit, profit. By management decisions, the differences between the KPIs and target KPIs are then mapped into company operational targets such as sales purchase limit price, marketing actions parameters decision on sales representatives' education and others. These decisions have a direct impact on sales and purchase in case of limit prices, while in other cases such as education or marketing they influence the process indirectly. So e.g. education can improve the sales or purchase representative ability to sell or to buy which causes changes in the parameters of the decision functions of customers and purchase representatives leading to the negotiation of other prices. The marketing action can improve the preferences of the customers – the next parameter of the customer decision function.

To define this part of the simulation model semantics following formal statements are presented:

Company target KPIs are represented as a tuple of reals $\langle KPI_1, KPI_2, \dots KPI_j \rangle$ (e.g. profit, revenue, gross profit$\langle 30000, 800000, 60000 \rangle$). The fulfillment of KPIs is evaluated at certain time-points. This is represented by a function from time-point to a tuple of reals.

Let $KPI\colon timepoint \to \mathbb{R}^j$

be a function assigning a tuple of real numbers whose coordinates represent each KPI member at a given time-point. For example, assume we have in April profit = -100000, revenue = 1000000, gross profit = 50000. This is formally represented as$KPI(\text{April}) = \langle -100000, 1000000, 50000 \rangle$, where the first coordinate corresponds to profit, the second to revenue and the third to gross profit. We denote the domain of KPIs as $\mathcal{K}$.

Let AC_{mng} be a set of management actions (e.g. decrease sales limit price, educate sales representatives, start advertising)

By comparing KPI and targetKPI values, the manager makes decisions such as

$Mng(KPI, April, targetKPI) = \{start\ advertising, decrease\ sales\ limit\ price\}$

So the signature of the management decision function is

$$Mng\colon \mathcal{K} \times timepoint \times \mathbb{R}^j \to 2^{AC_{mng}} \tag{3}$$

The rule to choose management actions can be formalized by logical formulas. For example, let us consider the action $decrease\ sales\ limit\ price$. Let us use following acronym for the action: p. Then the formula φ_{dslp} for the action would be:

$$\varphi_{dslp} = KPI(t)_{turnover} > 500 \land KPI(t)_{turnover} < 5000 \tag{4}$$

Likewise, we have formulas φ_a for each management action a. Then the management function Mng can be defined as:

$$Mng(K, t, G) = \{a \mid \langle K, t, G \rangle \vDash \varphi_a \ and\ a \in AC_{mng}\}, \tag{5}$$

where G means goal – target KPI). The definition of the expression $\langle K, t, G \rangle \vDash \varphi_a$ is straightforward.

It is obvious that such model uses some elements of social behavior described herein above. Due to modularity of the model, other modeled items with various behaviors can be included. So e.g., it could be possible to model overhead costs depending on revenue volumes. Last but not least, it could be possible to include some disturbances from the company environment affecting the decisions of the customers and suppliers.

2. Simulation model of the generic business company.

2.1. The multi agent simulation model

The formulation of the generic model presented in the section 1.2. is only the first step to company simulation. For the simulation we divide the generic model into three parts:

The ERP part. This part of the model serves for the economic events registration. From the registered values, the company KPIs can be calculated. The ERP part can also serve for the registration of other activities performed by the active participants of the business processes.

- The dynamic part. This part of the model is realized by software agents, which simulate the activities that would be done by human beings in the real process. This part also includes the social part of the model as the software agent parameters include their preferences and decisions as well as the communication among the agents generally.
- The databases. The files containing the local and global properties of agents, the company budgets, targets, and KPIs and the agent script codes themselves.

Now we inhabit the generic company model presented in Figure 1 with software agents. For this, each flat square in Figure 1 with process description is substituted by a software agent type. The simulation model is presented in Figure 4.

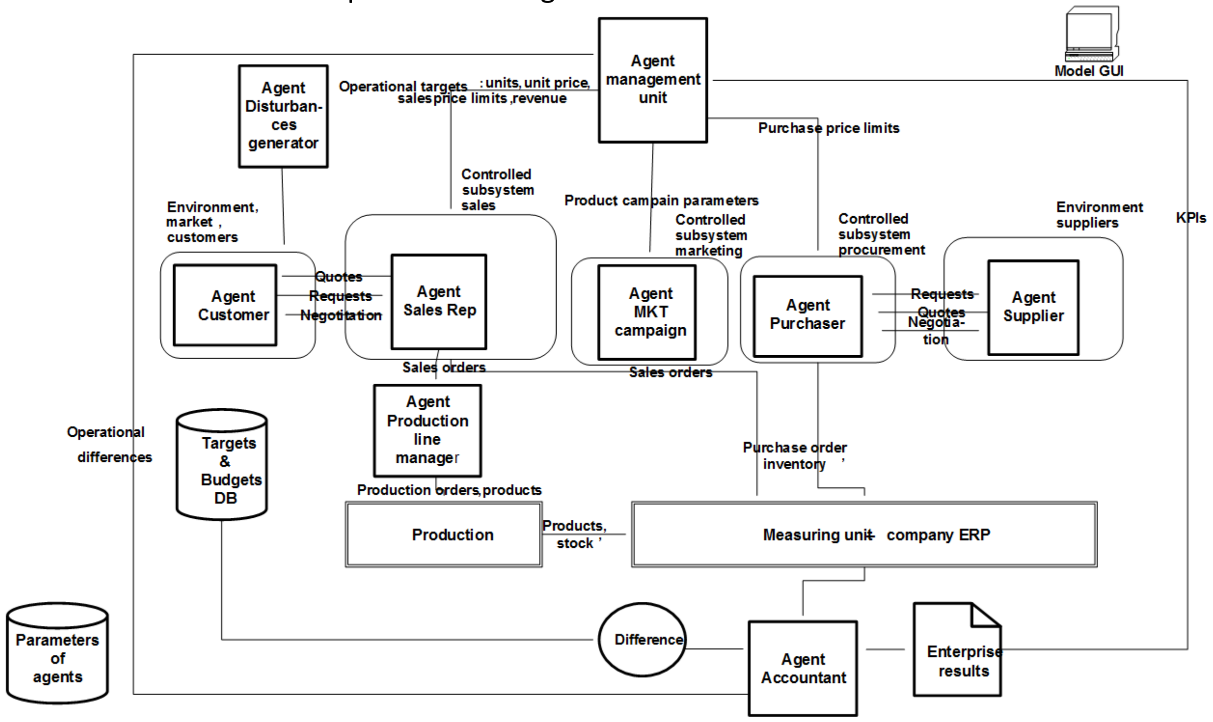


Fig. 4: Simulation model of a business company
(Source: own)

Note, that the general participant types such as Customers and Sales representatives in Figure 1 were substituted by agent types and their instances. The simulation model was also complemented by the runtime parameters of the agents. The REA type ERP implementation makes it possible to run simulations in very short runs due to the REA properties which enable to avoid double entry bookkeeping, accelerating thus the whole simulation run.

2.2. Software agents and process registration

We will start with an intuitive description of notions that will be used further on. We will examine the processes of a trading company. The procedure is explained in the **sales process,** the main part of which is negotiation on the price as presented in Figures 2 and 3 and defined in Section 1.2. A simple intuitive negotiation log example is presented in Table 1 resembling the schemes presented in Figures 2 and 3.

However, it actually shows a small part of concrete activities generated in a model run. What can be deduced from the Table 1 content? We can assume that the log consists of events data which are related to a single process (instance of price negotiation). Let us call the process instance a ***case.*** In Table 1 we see the numbered case (Case ID =10002) with time stamps and other attributes. Case 10002 consists of ***events*** related to corresponding activities (presented here as messages) between customer 45 and Sales representative, Peter Hanson.

Tab. 1: Simple negotiation log

Case ID	Mes-sage ID	Respo nse to	Message type	Message properties					Message content			
				Sender	Reci-pient	Send time	Read time	Qty of lines	Li-ne Nr	Product	Qty	Price
10002	10002	n/a	Request for quota-tion	Cust 45	Peter Hanson	020614 9:15	020614 18:06	1	1	UTP cable	76 m	n/a
10002	10006	10002	Quotation	Peter Hanson	Cust 45	020614 18:08	020614 18:09	1	1	UTP cable	76 m	11,25
10002	10043	10006	Quote rejection	Cust 45	Peter Hanson	050614 19:09	050614 21:03	1	1	n/a	n/a	n/a
10002	10071	10043	Quota-tion	Peter Hanson	Custr 45	070614 10:31	070614 13:41	1	1	UTP cable	76 m	8,38
10002	10075	10071	Quote accepta-tion	Cust 45	Peter Hanson	070614 13:50	070614 14:56	1	1	UTP cable	76 m	8,38
10002	10078	10075	Sales order	Peter Hanson	Accoun tant	070614 14:56	070614 23:47	1	1	UTP cable	76 m	8,38

Source: own

The activities referring to the case in Table 1 are represented by instances of message types *Request quotation*, *Quotation, Quote rejection, Quote acceptation* and *Sales order.* Table 1 also shows that the messages and hence the corresponding activities are ordered. The **ordering** information delivered by time stamps is necessary in order to discover control and message flow in the model. The event information presented in Table 1 includes some other information items. The time stamp information was already mentioned here. Other pieces of information such as message contents, the sender and recipient of the message are also presented here. Generally, each event can be complemented by related ***properties*** (also called ***attributes).***

The events presented in Table 1 also refer to ***participants*** – the agents taking part in the activities. Such participants are shown there as Customer and Sales representative. We assume that the same types of events have the same sets of attributes. It is obvious that the attributes of the events like e.g. the Customer and Sales representative id, the time stamp, the message id and the message types (understood here as activities) need to be brought into a more readable structure, but the figure give a good idea of the notions needed in the process log.

Based on Figures 2 and 3 and also on Table 1, we can make similar assumptions as researchers in process mining (van der Aalst, 2011, p.99 and further).

- A ***process*** consists of ***cases*** that can be logged in an event log.
- A ***case*** consists of ***events.*** Each event relates to exactly one case.
- Events within a case are ordered.
- Events can have properties called ***attributes*** such as time stamp, participant, activity and others

All events should be related to a single process, like e.g. an instance of price negotiation resulting in a sales order. Each event in price negotiation is related to a unique process instance (case) only in order to keep the log unambiguous. Each event is related to an activity pertaining to the activity set of a participating agent type. A finite sequence of events pertaining to one case such that each event occurs only once is called a **trace** in process mining research**.** Thus**,** our assumptions are typical for process mining procedures. Note that each event has one unique label – the activity ID to which the event is actually referring. The ordered set of traces containing the records of all activities pertaining to all cases in the simulation can be intuitively called an event log. Thus, each ***event log*** registers a finite number *n>0* of *cases*. Each case *i* consists of a finite number *k>0* of *events*. The events are sequentially ordered. The formalized "footprint" of each case forms a *trace.*

The formalization of various process mining notions was sufficiently done by Van der Aalst (ibid), and we will use it in our further reasoning.

The structure of the agent activities, (related to events), and the overview of our assumptions regarding their logging is presented in Figure 5.

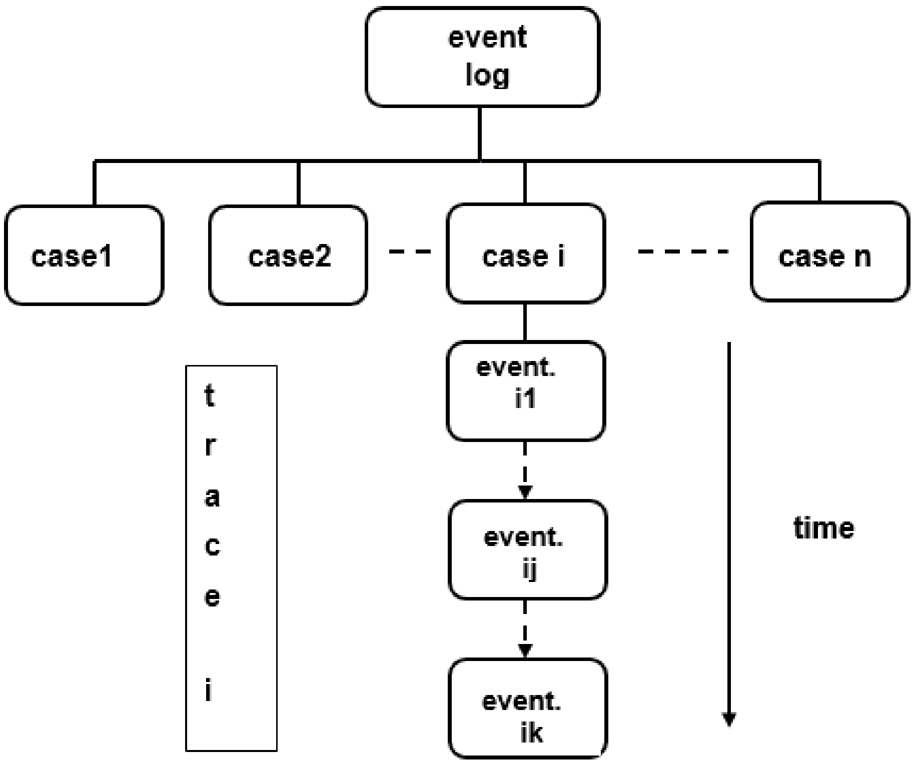


Fig. 5: Event log, cases, events, trace
(Source: own

However, unlike process mining, our intention is to log the activities of the agents in order to verify the simulation model structure later. While the typical process mining approach takes various log files from the existing processes or produces such files from information system database tables as inputs, our approach concentrates not on existing process analyses, but on agent communication within one simulation run in order to verify the simulation model proposed presented in Fig. 4. For this we need another type of inputs than a real process log. The basic idea is that the activities performed by the agents in the model can be complemented by messages representing the outputs of agents' actions – mostly the requirements for the activities to be performed by other agents. The communication and co-operation of agents is the basic idea of the agent paradigm. So, e.g. the agent Customer 45 decides to ask for a sales quotation from the agent Peter Hanson. The result of the customer's decision to buy something is a message of the *Request for quotation* type (message instance ID 10002). Peter Hanson prepares the quotation and sends the message of *Quotation* type as a result of his activity, (message instance ID 10006). The approach using messages as activities (events) representation within cases has an important impact on the log structure. First, any message has a sender and a receiver. So the activity 10002 Request for quotation has a sender, Customer 45 and a receiver, Peter Hanson. Second, any message is characterized at least by a sending time (time stamp) and read time (time stamp). Of course, there could be more time stamps such as receive time, processed time etc. Third, any message must have an attribute "Response to message" showing the message that originated the actual message, or showing an empty set $\varnothing$(in Table 1 marked as n/a). Each message must have an attribute Case ID connecting all messages (events) to one case. The usage of messages as task representation is a major distinction to the process mining concepts defined by other process mining researchers, who use the transactional life cycle model. The main advantage of the presented approach is that we can directly show communications among process participants – the agents.

Now, we present a basic formalization of the event log principles.

Each agent instance lives in her environment and reacts to the changes of state happening there (instigation events). No parallel activities are running inside the agent once a response-to-instigation event was started. Each response process has one start point and one end point ("sink"), thus, the set of agents' interactions can be looked upon as a workflow net. In such a case, the environment of an agent can be presented as a finite set of discrete, instantaneous states w[2].

$W = \{w_0, w_1, \ldots . w_n\}$, (6)

where *n* is the last state index of the set or, in other words, $|W| = n + 1$. The agent instances are grouped in agent types A_i (cf. herein below).

Each agent instance of an agent type communicates with at least one instance of a different agent type. As we already mentioned, we are using the notion of message/action instead of event or task. From now on, we will only use the notion of message.

Each agent type A_i has a finite set $Me_{A_i}$of possible message types representing actions available to her. As the actions of the agents transform the state of the environment and as they are represented by messages, the changes of the environment are also connected with the messages. The message types of the agent types A_i are considered to be disjointed.

$Me_{A_i} = \{\alpha_1, \alpha_2, \ldots \alpha_p\}$, (7)

where p means maximal index of message types available to an agent type A_i.

The agent instances a of agent type A respond to the impulses from their environment by means of a series of message instances from their sets $Me_a \subseteq Me_A$, (the set of agents' a message types $\{\beta_1, \beta_2, \ldots \beta_n\}$ $, n \leq p$ does not necessarily include all message types of its agent type) depending on the state of the environment.

Such a message type can be e.g. "request for quotation", "quotation acceptation", "quotation revoked" etc. for the customer agent type and "quotation", "sales order" etc. for the sales representative agent type.

Formally, for one agent type instance, the series of environment state changes c_a connected with message instances $b_0, b_1, \ldots b_n$ is represented as:$c_a = w_0 \xrightarrow[b_0]{} w_1 \xrightarrow[b_1]{} w_2 \xrightarrow[b_2]{} w_3 \cdots . \xrightarrow[b_{l-1}]{} w_l$, (8)

where $b_i$is an instance of some $\beta_j \in Me_a$

and w are here possible states of environment. (cf. run in Wooldridge, ibid).

However, setting the scene in which several agents take part, we have to take the messages of more agent types into consideration.

Let $A = \{A_1, A_2, \ldots \ldots A_k\}$ be a set of all agent types in the model.

Let $\mathcal{M}e = Me_{A1} \oplus Me_{A2} \oplus \ldots \oplus Me_{Ak}$ be a set of possible agent type interactions realized by messages where $\oplus$ is the direct sum of sets.

Let $Me_c \subseteq \mathcal{M}e$ be a subset of possible messages among agent types (e.g. within a sales negotiation) appearing in case c.

$Me_c = \{d_0, d_1, \ldots \ldots \ldots . . d_{m-1}\}$

[2] We use the letter w in order to keep a symbolic connection to modal logic formalisms, which use the notion of possible worlds w.

Then the case c can be seen as a series of state changes caused by messages of agents taking part in the case

$$c = w_0 \xrightarrow[d_0]{} w_1 \xrightarrow[d_1]{} w_2 \xrightarrow[d_2]{} w_3 \cdots \xrightarrow[d_{m-1}]{} w_m, \qquad (9)$$

where ws are now possible worlds of all agents.

Each business negotiation can have a different outcome depending on message sets used by agent instances involved and message contents. Please note, that the messages exchanged among the agents and the values of message attributes depend on the system history and specific agents' decision functions such as customer decision function defined by equation 2.

Intuitively, it is obvious that the set of actions executed by agent type instances and represented by corresponding messages adequately defines the case for process mining by van der Aalst and such messages in one case can be looked upon as labeled events within one case.

Thus, in principle, the basics of agent abstract architecture (by Wooldridge) perspective can be combined with the basics of process mining (by Van der Aalst). This makes it possible to formalize the logging of the agent activities in the log files.

Definition 1 (case, event, attribute).

Let μ_a be the set of all possible messages, corresponding to possible participants (agent types) actions. Message instances $m\epsilon\ \mu_a$are identified by unique message identifiers (message ID) and referred to by various attributes such as message name, agent sender and receiver instances, time stamps etc. A series of actions sequentially ordered by time stamps, represented by agents' messages and performed by various agent type instances constitute **cases.** Each case is started by one or more messages from the environment by the agent – instigator (e.g. customers' request for quotation). On instigation, one or several agents – performers can react. A typical reaction of more than one agent can happen in e.g. purchase negotiation. Here, the purchase representative sends a purchase quote request (instigation) and more suppliers can react with their quotations. Similarly, if the management agent decides on sales representative education, she sends a message to all sales representatives, which is then reacted to by the agents – sales representatives.

Let $\wp$ be a set of all possible cases (represented by case identifiers). Each case $c\epsilon\wp$ corresponds to a specific process instance and can have its own attributes. At least two attributes are needed for a case – the case ID and the trace. Let all events in cases be represented by messages identified by message identifiers. All messages have their own attributes.

The following standard message attributes will be used further:[3]

- $\#_{agent}(m)$ – the tuple of the agent instances associated to the message m (e.g. Sales representative Peter Hanson, Customer45),
- $\#_{message}(m)$ – the message type of *m* associated(referring) to event (e.g. Sales Quotation)
- $\#_{time}(m)$ – the tuple of the time stamps associated to *m* (e.g. message sent, message received…).

[3] We are using standard notation of Van der Aalst

- $\#_{const}(m)$ – the tuple of message constant fields associated to message m (e.g. product, price etc.),
- $\#_{resp}(m)$ - the message ID originating the message m (e.g. $\#_{resp}(m)$ =10043 – response to Quote rejection with ID=10043)

We could use several other attributes, but for our assumptions, we consider the presented ones sufficient.

Definition 2 (trace, event log). (Van der Aalst revisited)

For any case $c\epsilon\wp$ there exists a special mandatory attribute - *trace,* $\#_{trace}(c) \in {\mu^*}_a$, where ${\mu^*}_a$ is a set of all finite message sequences over μ_a. Thus, a ***trace*** is a finite sequence of messages (representing events) $\sigma \in \mu_a^*$ (with corresponding attributes) such that each message ID appears only once, that is, $for\ 1 \leq j < k < |\sigma\ |: \sigma(j) \neq\ \sigma(k)$.

According to this definition, taking (6) into consideration, we note that a trace of one case c is

- $\#_{trace}(c)\quad = \langle d_0, d_1, \ldots., d_{m-1}\rangle$ and e.g.

(see equation 9),

- $\#_{ID}(c) = 10002$ see Table 1.

The activities/messages of agents can be recorded in an ***event log.***

An event log is a set of cases $L \subseteq\ \wp$ such that each message (representing event and identified by its message ID) appears at most once in the log. As messages and cases are represented by unique identifiers, we can point to a specific case or specific message within the case as seen in Figure 2.

Using this formalism, adapted after van der Aalst, we can describe the event logs produced by agents without using a specific syntax.

In Table 1 we presented one case as a part of an event log.

Here $\#_{trace}(10002)\quad = \langle 10002{,}10006{,}10043{,}10071{,}10073{,}10078\rangle$

(Precisely, this should be $\#_{trace}(c)\quad = \langle d_0, d_1, d_2, d_3, d_4, d_5\rangle$ where $\#_{ID}(c) = 10002, \#_{ID}(d_0) = 10002, \#_{ID}(d_1) = 10006, \ldots.$)

Using various attributes in the log, we can obtain various results. Using message ID attributes, we can obtain Petri nets, or workflow nets, using agent attribute we can discover some type of social net etc. Both mentioned results, namely the Petri net and the social net can then be used for model verification.

3. Simulation model verification and validation: Case study

3.1. General description of verification and validation.

The verification and validation of the proposed model was performed in three steps:

a) Verification of the model structure and main functions – the test whether the modeled entities correspond with the abstract architecture presented in Section 2. Here, the verification of the customer and purchase representative decision functions (equation 2) and negotiation procedure was the main task. For the verification we used the ProM6 software alpha algorithm in order to receive the corresponding Petri net showing the negotiation procedure. As the input for this procedure the message log from the model was used.

b) Verification of the model dynamics – namely the behavior of the agents representing the real entities of the modeled reality. The simulation of the social networks of ProM6 Software was used as a verification tool. Here again, the message log was used.

c) Model validation – the test whether the model output data correspond with the real data delivered by the modeled company.

Only after the structural verification and the simulation outputs – real data comparison has been done, the verification of the management decisions based on the management back loop principle mentioned in section 1.2 would be possible. However, this verification is beyond the scope of the presented results, as this part needs extensive analysis of the management procedures in the modeled company and in reality, it is already the part of the prediction and management support task which could differ substantially in real companies.

The formalism presented in Section 2 was implemented in the MAREA simulation framework presented in several papers before (e.g. Vymetal and Schoeller, 2012; Vymetal and Sperka, 2014). The customer and purchase representative decision function was based on Marshallian demand function and customer utility function using Cobb-Douglas preferences backed up theoretically by Vymetal and Jezek, 2014. The decision function based on global simulation parameters used by i-th customer was defined as follows:

$$x_i = \alpha_i^* * \frac{m_i}{Px}, \tag{10}$$

where

x_i is the requested quantity (units),

α_i^* - i-th customer preference for the product expressed as a randomized global preference which was defined as product market share,

m_i – i-th customer budget expressed as a randomized global budget parameter,

Px – quoted price per unit.

The decision function in simulation script code was expressed as:

```
…
If !Sales quote line.exists(Quantity>Item.Market share*Budget constraints*Preferences/Price)
        Accept quotation
        Set budget constraints = Budgets constraints – Quantity*Price
…
```

The real data used for the model validation represents one year of sales statistics of a medium sized company selling UTP cable (sales unit in meters). The main characteristics of the real data compared to simulation outputs are presented later in Table 3.

The MAREA simulation global parameters are presented in the following Table 2. Please note, that the management function of the model was switched off in order to get “raw” output data.

Tab. 2: Global parameters of the simulations

Simu-lations teps	Custo-mer count	Sales repre-senta-tives	Purchase repre-senta-tives	Global budget (randomiza-tion base) of Customer, CZK	Max quantity sold, m	Starting sales price, CZK/m	Product market share
365	100	1	1	5500	305	7,90	9%

Source: own

Several simulation series were produced during verification and validation. Each simulation series consisted of ten simulation runs. For each simulation series, the average values of simulation outputs presented in Table 3 were calculated. The MAREA framework was changed to produce standard XES file registering messages among agents as process outputs in simulation logs as derived in section 2.1. The simulation logs were used as inputs for ProM6. In ProM6, the Mining for Petri net using alpha algorithm was calculated in order to present the model general workflow structure and to confirm the correspondence of the simulated model structure with the abstract concept of the model. The agents' behavior was analyzed by means of ProM6 social network mining programs, namely the Handover of work, Subcontracting and Working together social networks.

3.2. Process verification

For process verification we will assume that the company model presented in Fig. 4 can be transferred into a Workflow net. The reason for this assumption is that our model describes the life-cycle of the cases simulated in the simulation run. Workflow net is a subclass of a Petri net with a dedicated source place (start) and a dedicated sink place (end) while all nodes are on a path from source to sink places. We considered the process verified if:

a) The structure of the messages representing the processes is adequate to the generic company model (Figure 4). In that sense we e.g. expect that the sequence of the messages in the sales order case without price negotiation should be: Sales request – Sales quote – Sales quote acceptation – Sales order.
b) For process verification we used the standard ProM6 alpha algorithm process mining for Petri net. As shown in Figure 6 which presents an excerpt from the whole Petri net diagram as far as price negotiation is concerned, the simulation respects the expected workflow as defined in equations 1 and 2 and Figure 3. Thus, in our opinion, the process structure corresponds with the abstract company model. In our case, we made a check manually. However, using more sophisticated techniques, such as producing a reachability graph, finding deadlocks or livelocks, automated check can be done. Several analytical tools such as Woflan[4], or some analytical tools of ProM6 software are available. However, in our test the soundness of the resulting Petri net was not tested, as there were at least two domains modeled in one simulation test what would lead to unsoundness of the resulting workflow net. In further research, this type of test would be replaced by

[4] Related papers at: http://www.win.tue.nl/woflan/doku.php?id=publications

Woflan test and separate modeling of the sales and procurement domain will be done in order to produce a process log in the sense of workflow net soundness. Please note, that in the whole model, other activities are started such as bonus payment, production request is sent to the production line manager after successful negotiation. Production can result in other purchase requests in case of low stock. Also, in order not to complicate the Figures 6 - 8, the management back loop function was switched off in the simulation.

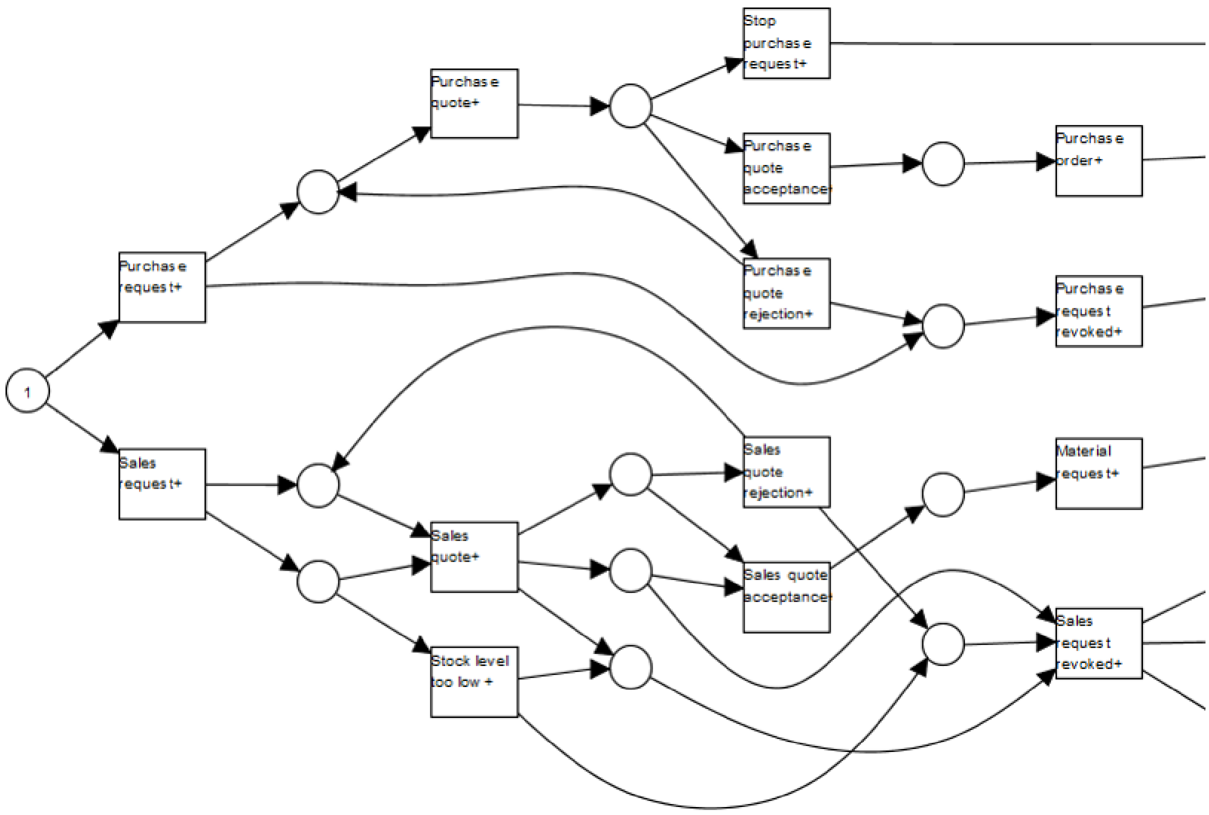


Fig. 6: Model verification - The price negotiation structure as a part of resulting Petri net (Source: own)

3.3. Agents' behavior verification

The aim of the agents´ behavior verification was to find out, whether the agent instances behave correspondingly to their parent agent types behavior defined in the model. Thus the task of the verification consisted of finding the agent clusters in which the agent entities with the expected behavior were grouped. For this, the social network analysis of the simulation log was performed. The social network analysis performed from transaction logs is a part of ProM6 software package. A twofold approach was applied. First, in order to find whether the model follows the causality between the process steps, we applied the Handover of work and Working together metrics for finding the social network. Both types of social networks are described in detail in Van der Aalst, Reijers and Song (2005). The main idea of Handover of work metrics is to find out, whether there is a handover of work between two entities (agents in our case) that is, if there are two subsequent activities where the first activity is performed by agent a_i and the second is completed by agent a_j. In this case we say there is a handover of work from a_i to a_j. The Handover of work metrics allows the checking of the model logic at the agent instance level. Please note, that in our model we do not consider the connections among the customers (such as e.g. information about the prices offered running among the customers with the aim to further improve the price negotiation results). In this case the Handover of work social network should not show any links among customers. Figure 7 shows the result. There are two separate clusters in the social network - the sales cluster and the procurement cluster.

The sales representative (in the center of the customers) hands the work over to the production line manager in case of the quotation acceptance. In the case of low stock, the purchase representative asks for product delivery from vendors. But which quotation was successful is not shown in this network. For this, the working together social network is to be used. However, the Handover of work as presented in this figure corresponds with the logic of the simulation model.

Second, in order to check, whether the agents follow the abstract model which stipulates for the agents to take part in the case as long as their contribution is needed, we used the Working together metrics. Such metrics does not take causal dependencies in consideration; it simply counts how frequently the agents work together on cases. As the model expects that all relevant agents take part in one case if necessary (e.g. the purchase representative has to provide for goods to be sold and the supplier has to deliver) the resulting social network should not contain singleton agent entities within the network. The result is presented in Figure 8.

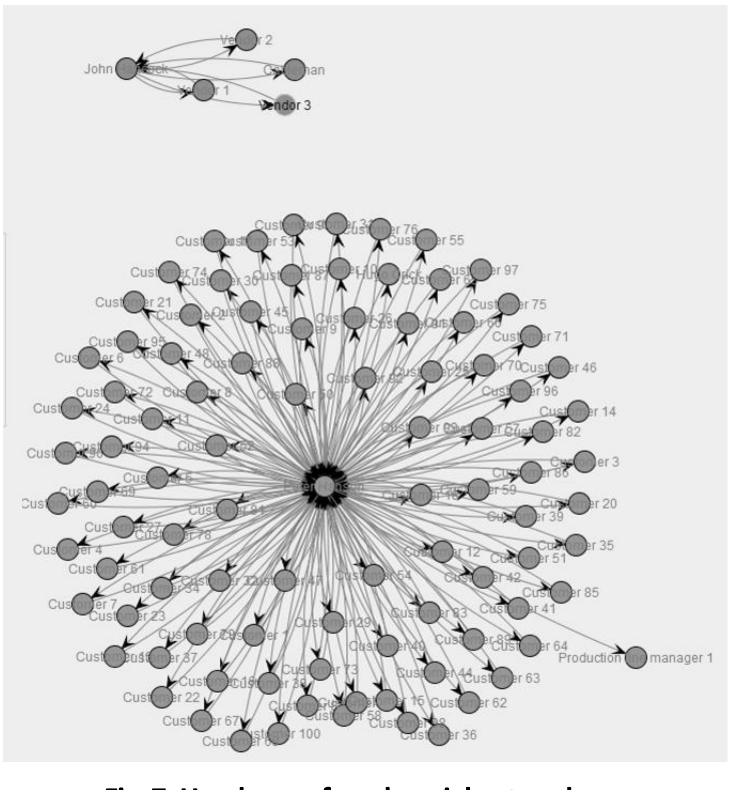


Fig. 7: Handover of work social network
(Source: own)

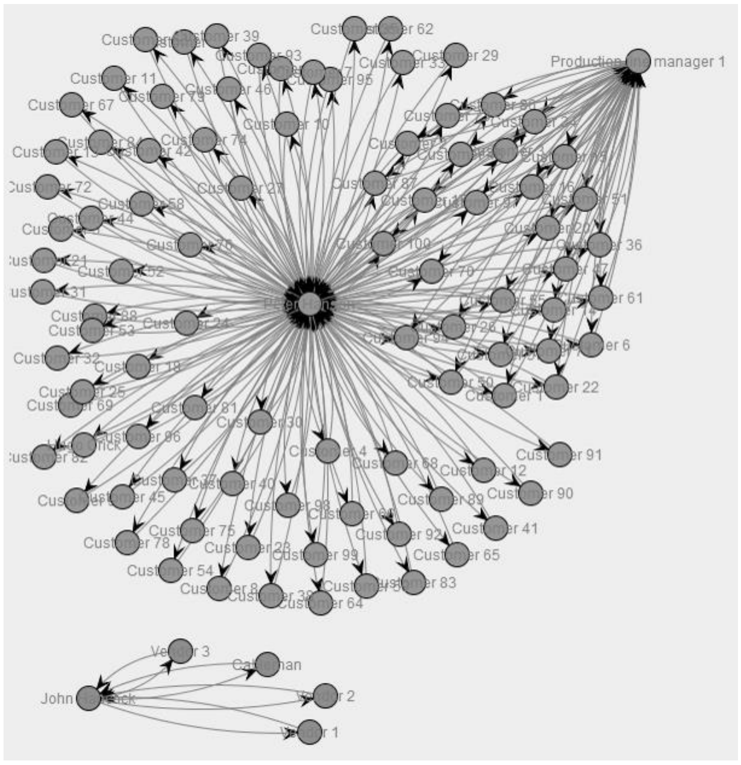


Fig. 8: Working together social network
(Source: own)

In this case we can see two groups of customers – those who accepted the quotation and thus triggered the production line and the second group, those who were in negotiation with the sales representative but did not accept the quotation in time or at all.

Hence, it can also be concluded that in this case that the model functioned according to expectations.

3.4. Comparison of real and simulated output data (model validation)

For model validation, the average simulation outputs were compared with real company data. In individual series of simulation runs the global simulation parameters were slightly changed in order to compare the average simulation outputs with the real company results. Altogether 160 simulation runs were performed during the validation. The best matching results showing average values for one year of simulation are presented in Table 3. The following company KPIs and simulation outputs were compared:

- Company revenue resulting from UTP cable sales, CZK/year. We can see, that the real and calculated values are comparable. (a difference of 3%)
- The gross profit (margin) on sales in %. Also here the calculated values are very near the real ones.
- Average real selling price and the price resulting from the sales negotiation in the model. The difference in this case is 9%. As the accepted price in the negotiation depends on the parameters of the customer decision function (members of the decision domains in equation 1 and 2), which were only partly estimated by the modeled company, here we can see some potential for further research.

- The real number of orders per modeled year was also well approximated by the simulation.

Tab. 3: Comparison of real and simulated output data

	Revenue/year CZK	Margin %	Average price/m, CZK	Number of orders/year
Simulation	51388	52.91%	8.64	171
Real data	49972	53.66%	7.92	167

(Source: own)

Based on the comparison of real and simulated data, following conclusions can be drawn:

- The model correctly simulates the margin and average number of orders per year.
- The higher simulated revenue is probably caused by higher price per meter
- The higher price is caused by the preference used in the customer decision function which most probably indicates that the market share of the company was under-estimated in the real company data.

However, the simulated data outputs are close to the real data and thus, we consider the model validation successful.

Conclusion

In this paper, an abstract multi-agent architecture of a trading company model using formalized approach is proposed. The derived formalism extends the formal description of the REA operational level proposed earlier. The abstract model is inhabited with active entities – software agents and a method of registering their actions in a simulation run log is proposed. This method combines basic notions of the abstract agent architectures with the methodology of process mining. In the presented solution the process mining case notions are represented by sequences of states of the environment caused by messages running among the active elements of the model - the agents. The usage of messages as task representation is a major distinction to the process mining concepts defined by other process mining researchers, who use the transactional life cycle model. The main advantage of the presented approach is that we can directly show communications among process participants – the agents. The derived formalization can be seen as a general approach to multi-agent simulation analysis and verification. The proposed solution is documented by a case study using real data of an anonymous high technology company. Using the proposed formalization, the former MAREA simulation environment was extended in order to reflect the behavior of the agents by means of simulation log and to enable the verification of the simulation model. For the customer decision function in the price negotiation, a method based on Marshallian demand function and customer utility function using Cobb-Douglas preferences previously backed up theoretically, was used. In model verification part, the simulation log was used as an input to ProM6 process mining tool. The structure of the processes is checked by means of Petri net created by the ProM6 alpha algorithm. The behavior of the agents was checked by analysis of Handover of Work and Working together social networks calculated again by corresponding ProM6 modules. This verification became feasible thanks to proposed formalization of multi-agent architecture in terms of process mining framework. Finally, the model validation with real data showed that the company model and the agent structure is viable and corresponds to the real data company outputs.

There are several research topics still open. First, an automated model verification is to be defined in using some more sophisticated Petri net analysis methods and corresponding software. Second, data from more company types has to be collected in order to get more generalized validations. These steps are planned for the future research.

References

[1] Aris, 2000. ARIS - Architecture of Integrated Systems. Workflow Management within the ARIS Framework. http://www.pera.net/Methodologies/ARIS/ARIS.html. Accessed 22 July 2010.

[2] BPMN, 2011. Business Process Model and Notation, version 2.0. http://www.omg.org/spec/BPMN/2.0/pdf. Accessed 3 December 2012.

[3] Barnett, M., 2003. Modeling & simulation in business process management. http://news.bptrends.com/publicationfiles/11-03%20WP%20Mod%20Simulation%20of%20BPM%20-%20Barnett-1.pdf. Accessed 21 December 2013.

[4] Baden-Fuller, & C., Haefliger, S., 2013. Business models and technological innovation. *Long Range Planning,* 46(6), 419 – 426. ISSN 0024-6301.

[5] Bellifemine, F., Caire, G., Poggi, & A., Rimassa, G., 2003. JADE a White Paper, TILAB exp „in search of innovation“, Vol.3, No.3, http://jade.tilab.com/papers/2003/WhitePaperJADEEXP.pdf. Accessed 2 July 2009.

[6] Bucki, R & Suchanek, P., 2012. The method of logistic optimization in e-commerce. *Journal of Universal Computer Science*, 18(10), 1238–1258. DOI: 10.3217/jucs-018-10-1238.

[7] Dunn, C., L., Cherrington, O., J. & Hollander, A., S., 2004. *Enterprise Information Systems: A Pattern Based Approach.* New York: McGraw-Hill/Irwin. ISBN 0-07-2404-29-9

[8] Ericsson, H., E. & Penker, M., 2000). *Business modeling with UML: Business Patterns at Work.* John Wiley &Sons, Inc. 166 p. ISBN: 0-471-29551-5.

[9] Gailly, F., & Poels, G., 2007. Towards Ontology-Driven Information Systems: Redesign and Formalization of the REA Ontology. Witold Abramowicz (Ed.): *Business Information Systems, 10th International Conference, BIS 2007*, Poznan, Poland, April 25-27, 2007, Proceedings. *Lecture Notes in Computer Science* 4439 Springer 2007, pp. 245-259.

[10] Gordijn, J., & Akkermans, J., 2002. Value-based requirements engineering exploring innovative e-commerce ideas. *Requirements Engineering,* 8(2), 114 – 134. ISSN: 0947-3602.

[11] Hruby, P., 2006. *Model/Driven Design Using Business Patterns.* Berlin Heidelberg: Springer Verlag. ISBN 3-540-30154-2

[12] Ito, S., & Vymetal, D., 2013).The Formal REA Model at the Operational Level. Applied Ontology, Vol. 8(4/2013), 275-300. DOI: 10.3233/AO-140129

[13] Liu, Y., & Triverdi, S., 2011. „Survivability Quantification: The Analytical Modeling Approach“. Department of Electrical and Computer Engineering, Duke University,

Durham, NC, USA, http://people.ee.duke.edu/~kst/surv/IoJP.pdf. Accessed 16 January 2012.

[14] Macal, C., & North, M., J., 2006. Tutorial on agent-based modeling and simulation, Part 2: How to model with agents, In: *Proceedings of the 2006 Winter simulation conference.* http://www.informs-sim.org/wsc06papers/008.pdf. Accessed 20 December 2010.

[15] McCarthy, W., E., 1982. The REA Accounting Model: A Generalized Framework for Accounting Systems in a Shared Data Environment. *The Accounting Review* (July 1982), pp. 554-578.

[16] McFarland, D.A. & Gomez, C. J., 2014. *Organizational Analysis.* Pages: 1- 205. http://service.sipx.com/service/php/inspect_document.php?id=x-06fd656e-b146-11e3-b4ce-22000a90058c. Accessed 2 September 2014.

[17] Odell, J., 2010. Agent Technology: An Overview. October, Ann Arbor, MI, USA. http://www.jamesodell.com/Agent_Technology-An_Overview.pdf. Accessed 23 May 2011.

[18] Slaninova, K., Martinovic, J., Sperka, R., & Drazdilova, P., 2013. Extraction of Agent Groups with Similar Behavior Based on Agent Profiles. *12th IFIP TC8 International Conference on Computer Information Systems and Industrial Management Applications (CISIM)*. Lecture Notes in Computer Science pp. 348 – 357. ISBN 978-3-642-40925-7.

[19] Slaninova, K., 2014. User behavioral patterns and reduced user profiles extracted from log files. *13th International Conference on Intelligent Systems Design and Applications, ISDA*, Article number 6920751, pp 289-294. DOI: 10.1109/ISDA.2013.6920751.

[20] Sperka, R., Spišak, M., Slaninová, K., Martinovič, J., & Drážďilová, P., 2013. Control Loop Model of Virtual Company in BPM Simulation. *Advances in Intelligent Systems and Computing,* 188, 515 – 524. ISSN: 2194-5357.

[21] Suchanek, P. & Vymetal, D., 2011. Security and Disturbances in e-Commerce Systems. In: *Proceedings of 10th International Conference Liberec Economic Forum.* pp. 580 – 589. ISBN: 978-80-7372-755-0.

[22] Vandenbosche, P.E.A. & Wortmann, J. C., 2006. Why accounting data models from research are not incorporated in DERP systems. Paper presented at the *2nd International REA Workshop*, June 25, Fira Santorini Island, Greece.

[23] Van der Aalst, W., M., P. 2004. Business process management: a personal view. *Business Process Management Journal,* 10(2), 4, ISSN: 1463-7154.

[24] Van der Aalst, W., M., P., Reijers, H., A. & Song, M., 2005. Discovering Social Networks from Event Logs. *Computer Supported Cooperative Work*, Vol. 14, Springer-Verlag. DOI 10.1007/s10606-005-9005-9.

[25] Van der Aalst, W., M., P. et al. Process Mining Manifesto. IEEE Task Force for Process Mining, 2009. http://www.win.tue.nl/ieeetfpm/lib/exe/fetch.php?media=shared:process_mining_manifesto-small.pdf. Accessed 15 January 2014.

[26] Van der Aalst, W., M., P. (2011). *Process Mining. Discovery, Conformance and Enhancement of Business Processes*. Berlin Heidelberg: Springer-Verlag, Germany, XVI, 352 p. DOI 10.1007/978-3-642-19345-3.

[27] Vymětal, D. & Schoeller, C., 2012. MAREA: Multi-Agent REA-Based Business Process Simulation Framework. In *Conference proceedings, International Scientific Conference ICT for Competitiveness.* OPF SU, 2012. pp. 301 - 310. ISBN 978-80-7248-731-8.

[28] Vymetal, D. & Sperka, R., 2013. Virtual company simulation for distance learning. In *Distance Learning Simulation and Communication Proceedings*. Brno: Univerzita obrany Brno, pp.189-197. ISBN 978-80-7231-919-0.

[29] Vymetal, D. & Jezek, F., 2014. Demand function and its role in a business simulator. *Journal of Advanced Research in Management,* Vol. V, Issue 1(9), Winter 2014, pp. 41 – 47. Doi:10.14505/jarm.v5.1(9).04

[30] Vymetal, D. & Sperka, R., 2014. MAREA – from an agent simulation application to the social network analysis. Proceedings *18th International Conference on Knowledge-Based and Intelligent Information &Engineering Systems – KES2014.* Procedia Computer Science 35, pp.1416 – 1425. DOI: 10.1016/j.procs.2014.08.198

[31] Weigand, H. & Elsas, P., 2012. Model-based auditing using REA. *International Journal of Accounting Information Systems*, Vol. 13, issue 3, pp. 287-310. doi:10.1016/j.accinf.2012.06.013

[32] Wooldridge, M., 2009. *MultiAgent Systems: An Introduction,* 2nd edition. Chichester: John Wiley & Sons Ltd. ISBN 978-0-470-51946-2.