Agent-Based Simulation of Business Processes in a Virtual World

Branislav Bošanský[1] and Cyril Brom[2]

[1] Center of Biomedical Informatics, Institute of Computer Science
Academy of Sciences of the Czech Republic
bosansky@euromise.cz
[2] Department of Software and Computer Science Education, Faculty of Mathematics and Physics, Charles University in Prague
brom@ksvi.mff.cuni.cz

Abstract. Business process modeling (BPM) has proven itself as a useful technique for capturing the work practice in companies. In this paper, we focus on its usage in the domain of company simulation and we present a novel approach combining the clarity of BPM with the strength of agent-based simulations. We describe the enhancement of a general process modeling language, the algorithm transforming these enhanced processes into the definition of agents' behavior, and the architecture of the target multi-agent system simulating the modeled company. The example is given as the implemented prototype of all proposed methods leading towards the simulation of a virtual company.

1 Introduction

Business process modeling (BPM) is a widely used technique offering a simple and understandable view on the work practice and it is mainly utilized by managers and executives. However, it has limited usage and can be unsuitable in the domain of company simulation, usually based on a statistical calculation, which cannot sufficiently represent human behavior or social factors. On the other hand, as proven by many existing applications, multi-agent systems and agent-based simulations can at the cost of typically more complicated specification provide more realistic model.

In our research we intend to create an agent-based simulation of the work practice in a company that is defined using a BPM technique. Our goal is to reach following characteristics of a company simulation program:

- A user can define a work assignment for agents (e.g. virtual subordinates) through some process modeling visual tool.
- Employees, represented by agents, should be human-like and they should act in a virtual world in order to accomplish given tasks and predefined goals.
- A user can see the virtual world, he/she can see the visual interpretation of agents representing employees, and can interact with this world.

The motivation behind described goals is to allow an easy modification of an agent-based simulation by a non-specialized user. An example can be given as an educational program for students of economy fields, where they can define what employees

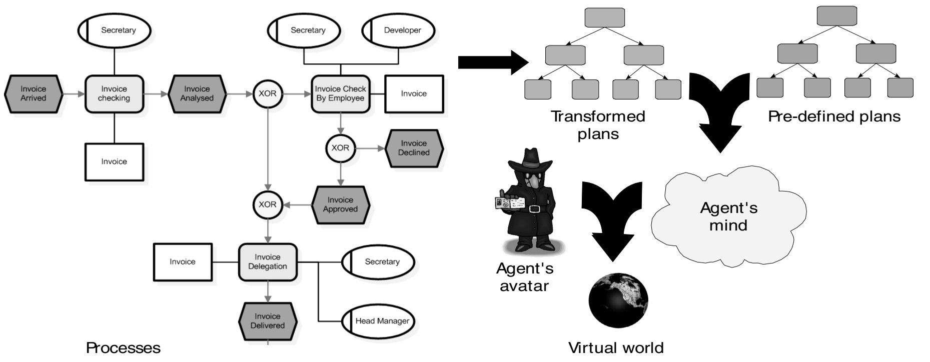

Fig. 1. The ground idea of presented approach

should do and how the employees should cooperate, and the program simulates and shows the progress of the work in the virtual world that represents the company.

In order to accomplish our objectives, we enhance a process language that can be automatically transformed into behavior description of agents that act in a multi-agent system (MAS), which represents the target simulation. The main aim of this paper is to present the algorithm transforming these enhanced processes into the hierarchical reactive plans for agents as well as to describe the architecture of a multi-agent system that can simulate virtual employees in a virtual world. Note that we understand these enhanced processes to be a parameterization for the target MAS and we assume other mandatory aspects for the agent-based simulation, such as the description of a virtual world, *a priori* knowledge of agents and their basic behavior (e.g. picking up an object), to be already captured (see Figure 1).

This paper is organized as follows. In Section 2 we analyze the problem domain and discuss different approaches in the area of company simulation. Section 3 is devoted to the description of a process modeling language enhancement. In Section 4 we describe the functioning of the target multi-agent system together with the algorithm transforming these enhanced processes into hierarchical reactive plans. In Section 5 we briefly present implemented prototype and the case study and we give our conclusions and discuss the possibilities of the future work in Section 6.

2 Business Process Simulation

We understand a business process to be an activity relevant for adding a value to an organization and capturing these activities as the business process modeling (BPM). There are several different languages that can be used for BPM, such as Unified Modeling Language, Business Process Modeling Notation or Event-Driven Process Chains (EPC) [5]. They are all related to some visualization and as they are very similar to each other, we will further talk about a general process language that represents a set of sequences of activities (processes), and that can be split into several flows or joined back together.

Usual process simulation approaches based on BPM methods are based on statistical calculation (e.g. as in [9]). However, several problems can be identified while using this method. As shown in [10], the concept of *work* itself consists of more than a functional transformation of work products by people and there are a lot of influences that cannot be captured using any business process model (e.g. effects of collaboration of employees or their communication). Moreover, employees are considered to be only the resources and it is hard to simulate their specific aspects (e.g. experience level, cultural or social factors) [10]. Finally, this method has only limited capabilities of visual presentation of running simulation and we do not actually see the employees working.

2.1 Agent-Based Company Simulations and Related Work

Agent-based simulations and their usage in a simulation of a company can bring several crucial advantages in the course of the simulation as such [4, 6, 8, 10], and can overcome some of the problems identified in the previous section. Firstly, agents that represent employees are more accordant with people and issues like communication, cooperation or coordination are all the basic characteristics of a multi-agent system. Agents can also be specialized (e.g. in life experience or adaptability in a new environment), agents are able to plan the assigned work or assign work to other agents, and modeling of interruptions or human behavior (e.g. personal characteristics, basic needs) is a lot easier too. Finally, in an agent-based simulation, which is set in a virtual environment, possible non-modeled behaviors can emerge (e.g. an agent carrying paper can be affected by other agents that are blocking the way).

However, the standard specification of an agent-based simulation is usually being done by defining agents' duties separately (e.g. using rules or finite states machines) in a correct way in order to achieve a global goal. Such an approach puts higher demands on a user and it is in contradiction to our requirements.

Hence we need to analyze possibilities of combining the BPM with agents, which is, according to our research, a not well documented approach and it is, to our best knowledge, not covered by any theory that would interpret a relationship between a general process-like language and an agents' architecture from the artificial intelligence point of view. We found several practical solutions in the domain of process modeling [4], in workflow systems [2, 11], or in biomedical informatics [3]. Their common characteristic is the attempt to formalize actions on a very low level (i.e. the atomic actions are rather basic, such as checking specific fields on an invoice) in order to be automatically executed by agents. Also, more advanced issues like the concept of trust, or different perspective of agents on the execution of their joint activity [4, 11] are addressed, which inhibits the development of the general transforming algorithm.

Approach in this paper presents several crucial distinctions. The first evolves in modeling actions on a higher level (i.e. handling an invoice can be the atomic activity). As we intend to apply the system in a simulation, we need to describe the activities and their expected results, but we do not expect the system to actually solve the problems. Moreover, we are integrating the concept of a process modeling technique into an existing agent framework (i.e. we do not model the whole MAS) and we set the simulation into a virtual environment.

3 Agent-Based Simulation Enhancement of a Process Language

In order to successfully simulate processes by means of autonomous agents, we can not use pure BPM language and we need to enhance it with several aspects. As we talk about these enhancements using a general process-like language, we mention the key ideas only and the enhancement of a specific language (EPC) is described in [1] more in detail. Firstly, we discuss inputs, effects, actors, and location of the activity, followed by more advanced concepts.

Inputs are representing the objects that are used during the execution. Moreover, they are also interpreted as a logical input condition that describes what state the attributes of the object should be in, in order to use it in the activity (e.g. we want to handle only new invoices in a process, hence the state of the invoice object should be "new"). *Effects* represent the adding value of a process. Two cases have to be distinguished: when a new object is created, and when an existing object (the one that is already used by the process as an input object) is modified. *Actors* represent a role that agent has to posses in order to execute the activity. We assume that the roles are organized in a tree hierarchy, agents can posses several roles, and a single role can be handled by several agents (e.g. agent Jane is a secretary, but she is also an employee and a human). Finally, *location* points to a place where the activity should be executed in a virtual world.

Beyond these quite straightforward aspects, we identify several more advanced characteristics. Objects and actors can be *optional* in their participation in an action, and they both can be used in more activities *simultaneously*, thus parameters regarding their utilization have to be specified for every activity. Furthermore, we want to utilize the *cooperation* of several agents participating on a single activity, hence we have to allow the assignment of more actors to a process.

The crucial part of the simulation is the course of an activity as such. We describe it using a mathematical function that can be defined for each output effect of a single process separately, meaning we are modeling several courses of changes in time – one for each output effect (e.g. the state of an invoice can during an execution of a single process change from "new" through "verified" to "prepared for payment", but the amount of money at the same invoice can be increasing linearly during the process due to a fine). Thanks to using general mathematical functions we can determine the precise state of all output effects at any time and we are able to apply partial results into a virtual world when an interruption occurs. Because all of the described functions have only one input variable – discrete time – we can transform the set of functions as a single multidimensional *transition function* of the process. Finally, according to a real-life practice, we expect the real course of the function during the simulation to depend on the actual state of environment and input objects (e.g. if we have some of the optional input objects we need less time to finish). Hence the transition function has to be parameterized by these aspects.

In order to successfully simulate the passage through a process sequence that contains splitting points, we also have to decide which of the continuations shall we follow. We prefer using a probabilistic values at the decision point rather than an attempt to formalize the logic of the decision been made. We argue that gaining the probabilistic values can be easier than the latter approach, as the real-life decisions are usually too hard to formalize.

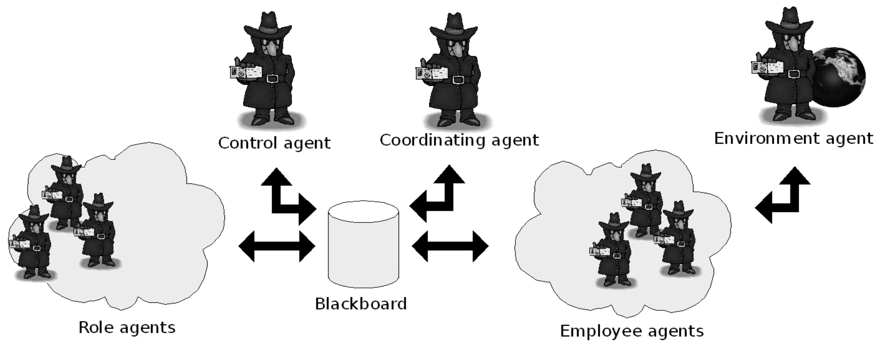

Fig. 2. Architecture of the multi-agent system for a company simulation

4 From Enhanced Processes to an Agent-Based Simulation

In this section we describe the transformation of extended processes into an agent-based simulation together with the architecture of the target MAS that simulates a modeled company. At first we focus on duties of individual agents in MAS followed by the description of the transforming algorithm.

4.1 Architecture of the Multi-Agent System

The architecture of the MAS is shown in Figure 2. We differentiate several types of agents, but there are three main groups. The first is the *environment agent* representing the virtual world. Secondly, there are *employees agents* that correspond with the virtual employees and they have their virtual body in the virtual world. Finally, we identify three types of auxiliary agents (*control*, *coordinating* and *role agents*) which help to organize *employee agents* in case of more complicated scenarios. In our approach we are using the blackboard architecture, where every agent is able to read and write at the common blackboard. The reason for this choice results from an easier realization of coordination of agents as well as an easier implementation. For *employee agents* we use hierarchical reactive plans as the inner control architecture. We based this decision on the fact, that hierarchical rules can already define a complex behavior and they further can be enhanced with planning or learning.

Let us present the simplest scenario of the simulation of a single activity: An *employee agent* reads from the blackboard the set of currently allowed actions (based entirely on process chains, the input conditions of actions are not considered here), it autonomously chooses one of them on the basis of its internal rules and the ability to satisfy the input conditions, and commits itself to execute it. It asks the transition function of the process, what is the expected finish time of this instance of the activity (as it can depend on the actual values of input objects attributes), and after the specified time it applies the target values of the effects of the activity as provided by the transition function and marks the activity as finished at the blackboard. However, during the execution of the activity, the agent can suspend its work (e.g. because it needs to accomplish a task with higher priority). At the time of the occurrence of this suspension, the agent asks the transition function for actual values of all effects and reflects the partial changes in the environment.

Now, we focus on the functioning of the whole system. The *control agen*t is the one who controls the correct order of the process execution according to the process chains and sets the set of currently allowed activities. In the case of cooperation of several agents in a process execution, the *coordinating agent* takes responsibility for notifying the correct subordinate agents and monitors their progress. Note that in this case, only one *employee agent* gets to actually execute the activity as explained above (so called *master agent*) and the other ones are considered to be only passive observers. However, the *coordinating agent* is necessary again in the case of an interruption, where it chooses one of the other participating agents to be the master. Finally, we describe the duties of the *role agents*. As we have stated in Section 3, we are using the concept of roles. Hence the *role agent* reads the set of currently active processes for the given role (set by *control* or *coordinating agent*) and activates them for selected *employee agent* that will execute them. Furthermore, when an interruption occurs and the suspended agent should be replaced by another one, *role agent* is responsible for notifying another *employee agent* possessing the same role.

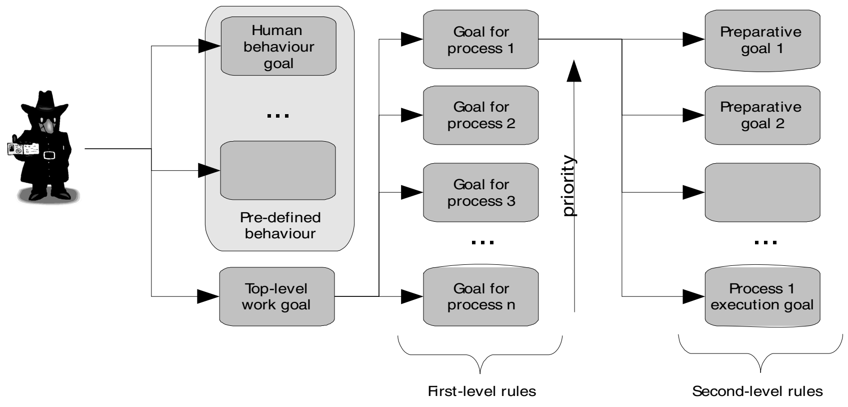

Fig. 3. Plan visualization for an employee agent

4.2 Transforming Algorithm

Let us now describe how the algorithm for automatic hierarchical reactive plans generation works (see Figure 3). As we have already stated, we are using the reactive architecture in our implementation of the algorithm, hence each goal of the plan is represented by a fuzzy if-then rule. For each process the *employee agent* can participate in, one rule is automatically generated. These rules are for each agent ordered by the descending priority of the activities and they form the first level of hierarchical architecture of the agent. The second layer is created by several sets of rules, where each set is collocated with one first-level rule. This second-level set of rules then represent several partial activities that are necessary to accomplish according to the conventions in the virtual world (e.g. transporting movable objects to the place of the execution of the process), and one rule for executing the simulation of the activity as such (modeled by a *transition function* as described in Section 3). Except the last one, the nature of these rules depends on the conventions that hold in the virtual world and therefore cannot be generalized.

The condition of a first-level rule is created as a conjunction of all constrains related to properties of input objects and participants (i.e. correct values of their utilization (whether they can execute this activity) and possibly other attributes, such as the state of an invoice etc.), and activation of an appropriate process. Moreover, if an input object or a participant is not mandatory, related conditions do not need to hold in order to fire the rule.

5 Prototype and a Case Study

In order to verify proposed process language enhancement, transforming algorithm together with the multi-agent system, we implemented a working prototype of a company simulation. For brevity, we highlight only the key parts and the details can be found in [1].

From several possible BPM languages, we chose and enhanced EPC. The observations defined in Section 3 are mostly semantic and we use standard EPC objects and their properties to capture them (e.g. we use orientation of "relation edges" connecting processes with data objects to differentiate input and output objects, we specify properties to capture their utilization in the process, obligation, etc.). Furthermore, we implemented these process language enhancements as an extension of existing EPC formal notation language EPML [7], and we created a new language which we call A-EPML. As a formalization of the transition function we used a Java class that can be highly configured through XML parameters stored in an A-EPML file. Using the new element of the A-EPML language, we can connect the Java class with the activity – e.g. a general linear function for each output effect is implemented in Java, but the parameters (e.g. different slopes of linear functions for separate effects) are passed through the XML configuration. This way we satisfied all demands on the transition function defined in Section 3 (we can easily implement a configurable multidimensional mathematical function using Java) together with preserving the possibility of easy visual modeling (user can set these properties using a visual tool).

As the next step we selected a proper multi-agent simulation framework – Intelligent Virtual Environment[1] – and implemented the transforming algorithm that creates sets of hierarchical rules from A-EPML files for each employee agent as described in Section 4.2. Finally, we created a case-study: a simulation of a small software development team. We designed the virtual world representing several offices with computers, desks and six employees (see Figure 4), and implemented pre-defined behavior of auxiliary agents and employee agents (e.g. walking, transporting objects). We modeled four EPC process chains on the basis of a real company. We selected the ones that, on the one hand, cover some cognate areas, but they as well cover most of the possibilities in enhanced EPC modeling – we chose repairing a broken computer, fixing a bug in software, implementing and testing a new feature in a software product and handling an invoice. We successfully applied our algorithm to transform them into rules for agents in IVE, where they acted according to the process model.

[1] http://urtax.ms.mff.cuni.cz/ive

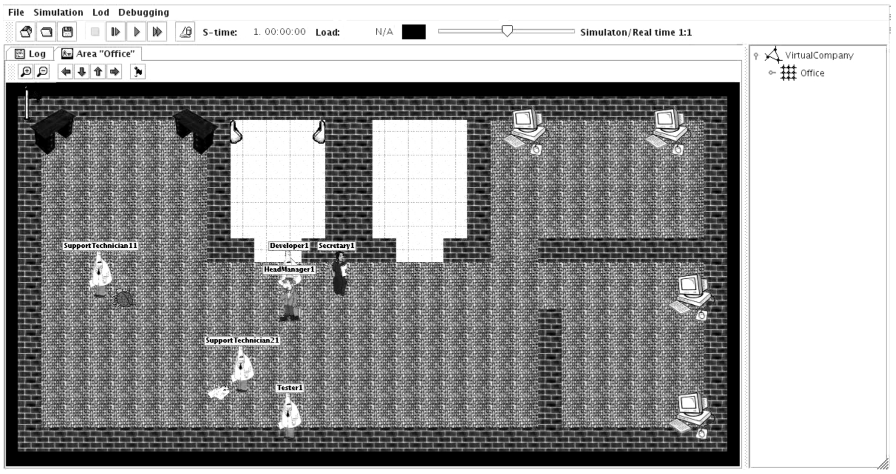

Fig. 4. Screenshot of the virtual world of implemented case study

6 Conclusion and Future Work

In this paper we have presented a novel approach combining the process language with the description of agents' behavior creating that way an easy modifiable agent-based simulation of the work practice in a company. We identified necessary additional characteristics that we need to enhance a process language with, designed the algorithm transforming these processes into reactive hierarchical plans for agents, and defined the functioning of a multi-agent system simulating the modeled company. To prove the functionality of our proposal we implemented described enhancement into the EPC language, created an appropriate A-EPML notation to store these processes, and implemented the algorithm transforming them into a multi-agent framework resulting into an agent-based simulation.

Our approach has several advantages. It opens the possibility of the behavior description of agents in the simulation using a simple graphical tool in the form of processes. Furthermore, these agents can easily be visualized in the virtual world representing a company, giving that way a good overview at the course of the simulation.

We argue that the presented method can also have other applications, than the one shown in this paper. Because the process modeling method is quite natural it can be applied in various areas (e.g. formalized medical guidelines). However, there are some limitations in the architecture of the employee agents and dependence on the rules of the virtual world during the transforming algorithm. Hence the future work will be mainly focused on overcoming these disadvantages and applying proposed approach in other areas as well.

Acknowledgments

This research was partially supported by the project of the Institute of Computer Science of Academy of Sciences AV0Z10300504, the projects of the Ministry of

Education of the Czech Republic No. 1M06014 and MSM0021620838, and the project "Information Society" under project 1ET100300517. We would like to thank Lenka Lhotská and Roman Neruda for valuable advices, and we also thank Petr Kocáb from SoftDec company for lending us the ISO protocols for the prototype implementation.

References

1. Bosansky, B.: A Virtual Company Simulation by Means of Autonomous Agents. Master's thesis, Charles University in Prague (2007)
2. Buhler, P.A., Vidal, J.M.: Towards adaptive workflow enactment using multiagent systems. Inf. Technology and Management 6, 61–87 (2005)
3. Corradini, F., Merelli, E.: Hermes: Agent-Based Middleware for Mobile Computing. In: Bernardo, M., Bogliolo, A. (eds.) SFM-Moby 2005. LNCS, vol. 3465, pp. 234–270. Springer, Heidelberg (2005)
4. De Snoo, C.: Modelling planning processes with TALMOD. Master's thesis, University of Groningen (2005)
5. Finkelstein, A., Kramer, J., Nuseibeh, B., Finkelstein, L., Goedicke, M.: Viewpoints: A framework for integrating multiple perspectives in system development. Int. Journal of Software Eng. and Knowledge Engineering 2, 31–57 (1992)
6. Jennings, N.R., Faratin, P., Norman, T.J., O'Brien, P., Odgers, B.: Autonomous agents for business process management. Int. Journal of Applied Artificial Intelligence 14, 145–189 (2000)
7. Mendling, J., Nuttgens, M.: XML-based reference modelling: Foundations of an EPC markup language. In: Becker, J., Delfmann, P. (eds.) Proc. of the 8th GI Workshop Referenzmodellierung 2004 at MKWI 2004, pp. 51–72 (2004)
8. Moreno, A., Valls, A., Marin, M.: Multi-agent simulation of work teams. In: Mařík, V., Müller, J.P., Pěchouček, M. (eds.) CEEMAS 2003. LNCS (LNAI), vol. 2691, p. 281. Springer, Heidelberg (2003)
9. Scheer, A.W., Nuttgens, M.: ARIS architecture and reference models for business process management, Bus. In: van der Aalst, W.M.P., Desel, J., Oberweis, A. (eds.) Business Process Management. LNCS, vol. 1806, pp. 376–389. Springer, Heidelberg (2000)
10. Sierhuis, M.: Modeling and Simulating Work Practice. PhD thesis, University of Amsterdam (2001)
11. Singh, M.P., Huhns, M.N.: Multiagent systems for workflow. Int. Journal of Intelligent Syst. in Accounting, Finance and Management 8, 105–117 (1999)