Agent-based Simulation of Processes in Medicine

Post-Graduate Student:
MGR. BRANISLAV BOŠANSKÝ

Department of Medical Informatics
Instutite of Computer Science of the ASCR, v. v. i.
Pod Vodárenskou věží 2

182 07 Prague, Czech Republic

bosansky@euromise.cz

Supervisor:
DOC. ING. LENKA LHOTSKÁ, CSC.

Department of Cybernetics
Faculty of Electrical Engineering
Czech Technical University in Prague
Technická 2

166 27 Prague, Czech Republic

lhotska@labe.felk.cvut.cz

Field of Study:
Biomedical Informatics

This research was partially supported by the project of the Institute of Computer Science of Academy of Sciences AV0Z10300504, the project of the Ministry of Education of the Czech Republic No. 1M06014 and by the research program No. MSM 6840770012 ”Transdisciplinary Research in Biomedical Engineering II” of the CTU in Prague.

Abstract

Process modelling 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 medical care. We analyze the problem of the simulation of processes and present an approach based on agent-based simulations. We formally define an enhanced process 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 processes in some environment. The example of usage is given in the form of a critiquing expert system proposal that uses formalized medical guidelines as the knowledge base.

1. Introduction

Process modelling is a widely used technique offering a simple and understandable view on the work practice within a team or a company, and it is mainly utilized by managers and executives in various fields of industry. The area of medical care also offers an opportunity for process modelling, and its usage in computer systems, such as hospital information system (HIS) or workflow management systems (WfMS). However, there are many problems with applying this proved technique into the medical care [1], hence it is not as spread as it could be. In our work, we focus on the simulation of processes in general, we study the possibility of using agents and multi-agent system for this purpose, but we also want to apply these state-of-the-art methods into a development of an expert system for physicians that would use processes as a knowledge base.

In the area of processes in medical care, most of the authors distinguish two main categories [1]. Firstly, there are processes that directly relate to treatment of a patient (e.g. describing treatment of a patient with a chronic ailment), and secondly there are processes that relate to organizational duties (e.g. the process of a reservation of a clinical bed for a patient within different hospital facilities). In our approach, however, we do not differentiate between these two types of processes and we try to work with them in the same way as a set of process diagrams and use them together. The reason for combining different sources of knowledge is to enable validation of applied procedures, on a general level, as well as with a local practice in a hospital, that can differ in each facility and that is captured as clinical processes.

The main goal of this paper is to summarize all aspects necessary for agent-based process simulation in a medical environment leading to an critiquing expert system. Therefore we firstly discuss more exhaustively processes and process simulation and its specific characteristics in the medical domain in Secion 2, where we also reason about the advantages that utilization of agents can bring into the field of process simulations. In Section 3 we present formal definitions of our enhanced processes. We describe the architecture of a multi-agent system that can simulate these processes in an environment in Section 4. Then in Section 5 we propose the vision of the whole expert critiquing systems, that can use this approach, and conclude in Section 6.

2. Process Modelling in Medicine

The work practice (i.e. duties of employees and organizational procedures – such as a specification of an activities' order, an assignment of employees as well as necessary resources to these activities, etc.) is usually captured using a set of processes describing the functioning of a work team or the whole company. These processes can be stored as a document in a textual form, and often these documents also contains their models, made using some of process modelling languages, as visual diagrams, which improve understanding and lucidity of the information.

There are several studies [1, 2, 3] that analyze the problems of applying process modelling or usage of workflow management systems in medical care. They all agree that the implementation of this approach can improve current problems with organization, reduce the time of hospitalization and finally reduce the costs. However, they also point out, that till now is the usage of processes rather low and insufficient. The main reasons were identified as more complex processes than in other fields of industry, or problems with interoperability resulting from inconsistencies of databases and used ontology or protocols. Finally, real processes in medical care are very variable hence the system that uses them has to be prepared for such a dynamic environment and multiple variations of similar processes. This factor prohibits us from applying standart workflow systems, that can not handle exceptions nor irregular situations.

As we have already stated, processes in medical care can be seen in several levels. Using terminology from [1] we can differentiate the *organizational processes* and the *medical treatment processes*. The latter type can be seen as medical guidelines that represent the recommended diagnosis and treatment procedures for a patient in a specific area of healthcare. They are approved by medical experts in related field based on the newest studies, literature reviews, and expert knowledge. There have been several surveys aimed at the importance of guidelines and generally they are considered to be a useful method for standardizing the medical practice, improving quality of treatment [4], or lowering the patient's medical expenses [5]. Currently, the guidelines are being approved as a document (i.e. in a textual form), which prohibits one from using them directly in a computer-based system – such as hospital information system, or an expert system helping a physician or a patient. Hence there has been a significant focus on the formalization of medical guidelines into a formal language [6]. Many formal languages have been developed, such as ASBRU [7], EON [8], GLIF [9], or PRO*forma* [10]. They are all quite different and based on different foundations, but they are all trying to capture the same thing – the recommended process of treatment of a patient in the specific area of healthcare.

In order to achieve corresponding simulation of processes, we need to simulate both of these types, as they affect each other – an organizational process is strictly limited by patient's health conditions, on the other hand, a physician has to take specific clinical processes and hospital organization into consideration when treating a patient. Furthermote we use both types in the same way – i.e. formalized using the same theoretical foundations, but in possibly different languages. In spite of several other proposed approaches (e.g. like in [11]), we do not try to convert one formalism into other one (e.g. formalize medical guideline using a business process modelling language), but modify existing formal languages in order to capture all necessary information for the agent-based simulation.

2.1. Using Agents in Simulations of Processes

Generally, we are interested in a simulation of processes in a certain environment by means of agents and we want to create a multi-agent system that would be coordinated and organized by a set of processes. Let us therefore discuss the advantages and disadvantages of this approach.

Using agents to simulate processes in companies is a promising alternative to standard process simulation methods based on statistical calculations [12]. There are several studies addressing this issue [13, 14, 15], and they all highlight the advantages, that agents are more accordant with people, they can be autonomous, they can plan their assignments and they can distribute and coordinate their activities. However, in practice, there are not many existing applications that would interpret a process language in a multi-agent system and let agents be guided directly by modeled processes. In some cases, even though the agents are supposed to simulate processes, their behavior is hand-coded depending on processes (e.g. in [14]) using some standard decision mechanism (e.g. rules, FSM, etc.). Several approaches in the area of WfMS were discussed in [16] or even processes modelling in [17], however no existing implementation or transforming algorithm for agents' behavioral definition was presented. In both these cases authors try to cover much wider concepts (e.g. different views on a single process by different agents, concept of trust etc.) which prohibits them from proposing the universal MAS architecture and algorithm interpreting the processes into behavior of agents. Therefore we introduced a new approach to a process simulation in [18] that defines a universal multi-agent system and transforming algorithm that enables process simulation by means of reactive autonomous agents.

When we closely focus on using agents in process simulation in medical care, we can see that several problems, mentioned in previous section, can be overcome. When agents represent the hospital staff or patinets, they do not have only to follow the modeled processes, but also their own pre-defined goals, hence the exceptions or interruptions of process execution can be handled much easier. Furthermore, using enhanced processes described in [18], the variability of processes can be assured.

3. Formal Definition of Agent-based Process Simulation

In order to correctly define multi-agent system that simulates modeled processes we firstly need to properly define processes modeled in process diagrams.

Definition 1: We call a seven-tuple $D = (P, S, E, C, O, A, R)$ a process diagram, when:

- P is a non-empty set of processes (activities).
- S is a set of passive states that describes current state of environment.
- C is a set of connectors that can split or join the control flow.
- $E \subseteq (\{P, S, C\} \times \{P, S, C\})$ is a non-empty set of control edges that connect processes and define a control flow of a diagram.
- O is a set of objects from the environment. Each object has a set of parameters that can be modified by processes.
- A is a non-empty set of roles of agents that participate in activities.
- $R \subseteq (\{P, O, A\} \times \{P, O, A\})$ is a set of auxiliary edges (relations) connecting agents and objects with processes.
- Process diagram is a directed graph $G = (V, X)$, where $V = (P \cup S \cup C \cup O \cup A)$ and $X = (E \cup R)$

Furthermore, when D is a process diagram, and

- p_v is a set of vertexes preceding to the vertex v; $p_v = \{n \in V; (n, v) \in E\}$
- s_v is a set of vertexes succeeding to the vertex v; $s_v = \{n \in V; (v, n) \in E\}$

following conditions have to hold:

- The sets of vertexes P, S, C, O, A are pairwise disjoint. The same condition holds for the sets of edges E and R.
- There is at most one edge outgoing of and incoming to each node except connectors; $\forall v \in \{V \setminus C\} : (|p_v| \leq 1) \wedge (|s_v| \leq 1)$
- Logical connectors have at least one incoming and at least one outgoing edge. We distinguish exactly two types of connectors – splitters and joiners. We thus define two disjoint subsets of the set of connectors as: $C = T \cup J$, where $T \cap J = \emptyset$. Now the following corollaries hold:
  - splitters – connectors that have exactly one incoming edge and at least two outgoing edges; $\forall t \in T : (|p_t| = 1) \wedge (|s_t| > 1)$
  - joiners – connectors that have at least two incoming arc-edges and exactly one outgoing arc-edge; $\forall j \in J : (|p_j| > 1) \wedge (|s_j| = 1)$

This definition of process is quite universal. It is based on EPC language definition [19] that is widely used in business process modelling. However, it is extended in order to cover other specific languages as well, specifically GLIF, that we use to formalize medical guidelines. We use three control entities (processes, states and connectors) that forms the control flow, and two auxiliary entities (agents, objects) that describe processes in more detail. Note, that in definition of relations (the set R), we allow the connections between different roles as well. This corresponds to definition of organizational hierarchy in a team using roles (e.g. Jane is a nurse and she also is a general employee).

Now, let us define the enhancements for a general process language identified in [18], in order to be able to properly simulate them using a multi-agent system.

Definition 2: We say, that $D' = (P, S, C, E, O, A, R)$ is an enhanced process diagram, when for each $p \in P$ hold:

- $O_{p_i} = \{o \in O; (o, p) \in R\}$ is a set of input objects of the process. Following properties of each input object have to be specified:

  optional – relation that represents whether this object is necessary for executing the process or not

  utilization – float number representing the amount of usage of the input object in order to use it in several processes at the same time

- $O_{p_o} = \{o \in O; (p,o) \in R\}$ is a set of output objects of the process. If an output object is not also an input object, the process creates a new object in the environment.
- $A_p = \{a \in A; (a,p) \in R\}$ is a set of roles of executing agents. Following properties have to be specified for each role:

  optional – a relation that represents whether this agent is necessary for executing the process or not

  utilization – a float number representing the amount of agent's utilization in order to enable the possibility of multi-tasking of agents

  replace – a relation that represents whether agent should be replaced by another agent possesing this role when it interrupts the execution of this process, or not
- *location* – an optional characteristic represented by one of the input objects. As we are running the simulation in a certain environment, there can be a need for executing each process at precise location (e.g. an examination should be executed in the appropriate room of hospital that can be modeled as a virtual world for the visualization of the whole simulation).
- *priority* – an integer number representing the priority of the process
- *transition function* – a description of the course of the activity as such. Let X_i be the domains of changing parameters of output objects of the process. Then we say, that

  $$f^p_{O_{p_i}} : \mathbb{N} \longmapsto (X_1, X_2, \ldots, X_m)$$

  is a transition function of the process that for each timestep (a natural number) returns the actual values for each changing parameters of the output objects.

These enhancements are mostly natural and correctly specify input and output objects with their characteristics, or participating agents. Note, that we allow cooperation of several agents on a single process ($| A_p | \geq 1$), and we introduce multi-tasking of agents as well.

We explain the definition of the transition function, that represents the course of the process. We use a concept of mathematical functions 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 parameter (e.g. the state of a patient examination request can change during an execution of a single process from "new" through "verified" to "prepared"). 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 of the process 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 accomplish a tas). Hence the transition function is parametrized by these aspects.

4. Multi-agent System for a Process Simulation

In previous section we defined the enhanced processes nad now let us define a multi-agent system (MAS), that simulates them. Firstly we present an existing MAS architecture, which was proved usefull as prototype implementation in [18], following by several enhancements that we want to implement in our current approach in order to improve the course of the simulation as such.

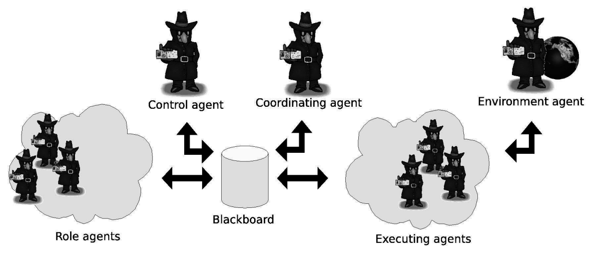

Figure 1: The architecture of the MAS simulating enhanced processes.

The organizational scheme, shown in Figure 1, represent a multi-agent system that can simulate processes captured in a formalism for enhanced processes. We differentiate several types of agents, but there are three main groups. The first one is an agent representing the environment in which the simulation proceeds. Secondly, there are *executing agents* that correspond with the modeled hospital staff (e.g. physicians) that act within the environment. Finally, we identify three types of auxiliary agents (*control*, *coordinating* and *role agents*) which help to organize *executing agents* in case of more complicated scenarios. Communication of agents uses the blackboard architecture, where every agent is able to read and write facts (e.g. activation of specific processes for an execution agent, etc.) at the common blackboard. As the decision mechanism for the execution agents, the hierarchical reactive plans were used, as they are easy to automatically generate from process diagrams and they can define reasonably complex behavior of an agent.

Let us now describe the functioning of the system. For auxiliary agents, we present their behavior in the pseudocode, that for brevity handles with only one instance of process diagrams in the system. In the implementation, however, several instances of the same process diagram can be active. Firstly, we focus on a simple scenario – simulation of a single process. An *executing agent* reads from the blackboard a set of currently allowed actions (they are allowed entirely based on progress in process diagrams), it autonomously chooses one of them on the basis of its internal rules, priority of the processes, 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 parameters), 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.

Algorithm 1 Rules for the *control agent*

```
1: if $\exists p \in P : finished(CoordAgent, p)$ then
2:     choose processes $P' \subseteq P$ subsequent to $p$ according to the process rules
3:     if $P' \neq \emptyset$ then
4:         $remove(finished(CoordAgent, p))$
5:         for all $p' \in P'$ do
6:             $store(active(CoordAgent, p'))$
7:         end for
8:     end if
9: end if
```

Described scenario was the simplest one, however in more advanced cases, the three auxiliary agent types are used. The *control agent* is the one who controls the correct order of the process execution according to the process diagrams and sets the set of currently allowed activities. We can demonstrate its behavior using pseudocode shown in Algorithm 1. Note, that movement in the process chain in line 2 can contain several steps or possibly splitting or joining the flow using a connector.

In the case of cooperation of several agents in a process execution, the *coordinating agent* takes responsibility for notifying the correct subordinate agents (lines 1–4), it selects which agent is so called *master agent* (i.e. the one, that actually modifies objects used in the process; lines 5–6), and monitors the progress of the execution (lines 8–27). Coordination agent is also necessary in the case of an interruption, when it chooses one of the other participating agents to be the master agent (lines 16–25):

Algorithm 2 Rules for the *coordinating agent*

```
1: if $\exists p \in P : (active(CoordAgent, p) \wedge (\neg\exists a \in A_p, \exists p' \in P : \neg optional(a, p) \wedge active(a, p') \wedge (priority(p') > priority(p))))$ then
2:     for all $a \in A_p$ do
3:         $store(active(a, p))$
4:     end for
5:     choose one $a \in A_p$
6:     $store(master(a, p))$
7: end if
8: for all $p \in P$ do
9:     if $(\exists a \in A_p) : (active(a, p) \wedge master(a, p) \wedge \neg working(a, p))$ then
10:        if $finished(a, p)$ then
11:            $remove(finished(a, p))$
12:            for all $a' \in A_p$ do
13:                $remove(active(a', p))$
14:            end for
15:            $store(finished(CoordAgent, p))$
16:        else if $interrupted(a, p)$ then
17:            if $\neg optional(a, p)$ then
18:                for all $a' \in A_p$ do
19:                    $remove(active(a', p))$
20:                end for
21:            else
22:                choose one $a' \in \{e \in \{A_p \setminus a\} : working(e, p)\}$
23:                $store(master(a', p))$
24:            end if
25:        end if
26:    end if
27: end for
```

Finally, we describe the *role agents*. We are using the concept of roles, hence the role agent reads the set of currently active processes for the given role (set by the coordinating agent, line 1) and activates them for selected executing agent (lines 2–4). When an interruption occurs and the suspended agent should be replaced, a role agent is responsible for notifying another executing agent possessing the same role (lines 11–14).

Algorithm 3 Rules for a *role agent*

```
Input: $a$ is this role agent; $Ex_a$ is a set of executing agents that posses this role
 1: if $(\exists p \in P) : active(a,p) \wedge \neg working(a,p)$ then
 2:     choose one $e \in \{c; c \in Ex_a \wedge (\neg\exists p' \in P : working(c,p') \wedge (priority(p') > priority(p)))\}$
 3:     $remove(interrupted(a,p))$
 4:     $store(\{active(e,p)), deleg(a,e,p), working(a,p)\})$
 5: end if
 6: for all $p \in P$ do
 7:     if $\exists e \in Ex_a : deleg(a,e,p) \wedge \neg working(e,p)$ then
 8:         if $finished(e,p)$ then
 9:             $remove(\{active(a,p), deleg(a,e,p)\})$
10:             $store(finished(a,p))$
11:         else if $\neg finished(e,p) \wedge replace(a,p)$ then
12:             $remove(\{active(e,p), deleg(a,e,p)\})$
13:             choose one $e' \in \{c; c \in Ex_a \setminus \{e\} \wedge (\neg\exists p' \in P : working(c,p') \wedge (priority(p') > priority(p)))\}$
14:             $store(\{active(e',p), deleg(a,e',p)\})$
15:         else
16:             $remove(working(a,p))$
17:             $store(interrupted(a,p))$
18:         end if
19:     else if $working(a,p) \wedge \neg active(a,p)$ then
20:         for all $e \in Ex_a : deleg(a,e,p) \wedge active(e,p)$ do
21:             $remove(\{active(e,p), deleg(a,e,p)\})$
22:         end for
23:     end if
24: end for
```

4.1. Transforming Algorithm

Let us now describe how the set of rules for an executing agent is automatically generated and how its action-selection mechanism works.

As we have already stated, we are using the reactive architecture for *execution agents*, hence each goal of the plan is represented by a fuzzy if-then rule. For each process the executing 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 create the first level of hierarchical architecture of the agent. The second layer is created by several sets of rules, where each set is related to one first-level rule. This second-level set of rules represents several partial activities that are necessary to execute according to the conventions in the environment (e.g. transporting movable objects to the location 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.

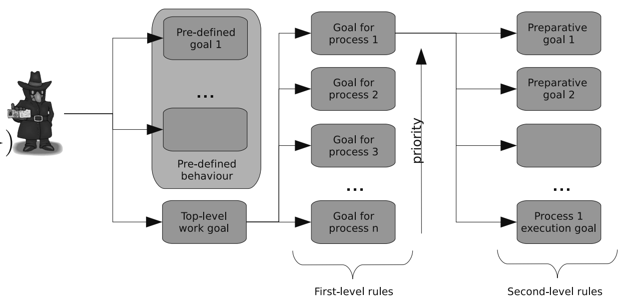

Figure 2: A hierarchy of reactive plans of each *executing agent*

The condition of a first-level rule is created as a conjunction of all constrains related to properties of input objects and agents (i.e. correct values of their utilization (whether they can execute this activity) and possibly other attributes, such as the state of an patient 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.

4.2. Improved Architecture of the MAS

The architecture presented in previous section can successfully simulate modeled processes and as such can suit our intention to create an expert critiquing system based on the simulation of clinical processes and formalized medical guidelines. However, several issues can be improved in previous approach. First of all, the control and coordination of *execution agents* using specialized auxiliary agents together with a blackboard architecture is quite stiff and it partially limits the autonomy of *executing agents*. Moreover, in order to enhance *executing agents* with planning or advanced architectures, much more organization-related communication would be needed.

Therefore we propose a new architecture that, according to our experiences gained during implementation and testing the previous one, should emphasize more the positive concepts of agents paradigm and enable implementation of further improvements and functionality, such as planning and better *executing agents* coordination. Currently, we do not change the reactive architecture of the *executing agents* as there is not known correct interpretation of common knowledge in form of processes for more deliberative agents. We argue that processes have stronger conceptual meaning than a plan library for an agents, as not only an agent knows what actions it needs to execute, but also what actions other agents should execute and how their actions would affect the state of the environment. This remains an open problem which we want to address in further research.

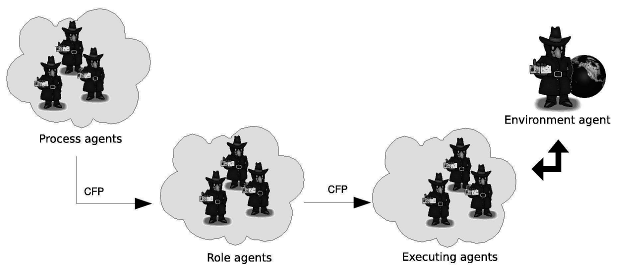

Figure 3: The improved architecture of the MAS simulating enhanced processes.

The schema of the new architecture is shown in Figure 3. Both control and coordinating agent were replaced by a set of agents – for each of modeled process one *process agent* is automatically generated. Each of them is responsible for executing one type of activity (possibly several instances of one process) and the duties of removed auxiliary agents are distributed within this set. Note that in the new architectural schema, the blackboard is no longer used. The simplified organizational concept enables the possibility of usage the direct messaging as well as a standard concept of the Contract-Net Protocol (CNP) [20].

The pseudocode of a *process agent* is shown in Algorithm 4 and it presents how a it acts in the simulation. Note that the pseudocode is reduced (several lines regarding the responses to rejections are omitted). We can see, that agent keeps to the CNP and each *process agent*, when notified (lines 2–6), finds appropriate role agents (lines 25–26 and 11–15), monitors the progress of the master agent in case of cooperation, and passes the information of the success to the next process agent(lines 16–24). Other agents, *role* and *executing*, are acting in the same way, except the changes in the communication.

Let us point out the advantages, that these modifications can bring. The key change is the shift from the blackboard architecture to the direct messaging within agent community together with using standard protocols. At the cost of increasing the overall number of agents we simplify the communication within agents (compare the organizational communication issues in *control* and *coordinating agent* with *process agents*). Moreover, we expect easier integration of planning that can be added as further communication within *process* and *role agents* (e.g. one *process agent* knows, what the subsequent processes are, hence it can notify appropriate agents in advance and negotiate executing some of the auxiliary actions (see the second-level rules in Section 4.1) to save time).

Algorithm 4 Rules for a *process agent*

```
Input: $p$ is a process assigned to this agent; $I_p$ is the set
       of currently active instances of $p$; $m_i$ master agent
       of the process for $i \in I_p$; $X_i$ is a set of returned
       proposals for $i \in I_p$ asdasd
 1: for all $msg \in IncMsgQueue$ do
 2:     if $msg$ is activation of $i$ then
 3:         $I_p = I_p \cup i$
 4:         $m_i = \emptyset$
 5:         $i$ is new
 6:         $X_i = \emptyset$
 7:     else if $msg$ is proposal for $i$ then
 8:         $X_i = X_i \cup msg$
 9:     end if
10: end for
11: if $CFPTimeOut \wedge X_p \neq \emptyset$ then
12:     choose one agent, $m_i$, from $X_i$
13:     $sendAcceptProposal(i, m_i)$
14:     $i$ is started
15: end if
16: for all $i \in I_p$ do
17:     if $i$ is finished then
18:         $I_p = I_p \setminus i$
19:         for all $p \in (s_p \cap P)$ do
20:             $sendActivation(success(i), a)$
21:         end for
22:     else if $i$ is interrupted $\wedge\{A_p \setminus m_i\} \neq \emptyset$ then
23:         $X_i = \emptyset$
24:         $sendProposal(i, A_p)$
25:     else if $i$ is new then
26:         $sendProposal(i, A_p)$
27:     end if
28: end for
```

5. Future Work

So far we discussed processes, problems related to their simulation, and proposed a solution based on a multi-agent system. Now we present the vision of the critiquing expert system which can profit from these methods.

The critiquing system runs in the background of the standard applications of HIS and controls the inserted data about a patient. From these data values it tries to recognize a medical guideline that physician is following and furthermore recognize the state of the patient. After a successfull matching, it further predicts the future progress of possible patient's treatment with respect to the guidelines and database of existing cases in the facility. This prediction follows the next steps in guidelines (note, that patient can have several diseases hence we need to take all of them into consideration) and tries to simulate the future actions of the physician and in case of missing current data value (e.g. a result from an examination that patient have not undergone yet) the approximation using similar patients from the database is made. Also, this prediction would be probabilistic, hence multiple branches of the guidelines would be evaluated. Therefore, in case of for example omitting an optional examination, physician can be alerted by the system that similar patients had results that negatively affect their further progress. Finally, the simultaneous work with several guidelines for different diseases can bring attention of the physician that treatment of a disease she/he is focused on can conflict with another treatment that this patient is going through.

In this system, we want to combine several existing techniques. For a guideline recognition we want to use ideas from existing plan recognition techniques (such as using Bayesian network), and for guideline simulation we want to apply the approach described in this paper. However, the advantage of usage of agents for a guideline simulation purpose (and the whole system as such) is not so evident. We argue that focusing on distributed artificial intelligence can simplify the implementation and also the adaptivity of the system (e.g. learning of the specialized *process agents*). Finally, in the future a system designed on such general principles could also be integrated into more advanced HIS based on processes, which could help to plan and organize work in a hospital facility with a close relation to specific patients' treatment.

6. Conclusions

In this paper we presented an approach to an agent-based simulation of processes in an environment and described its possible utilization in medical care – specifically in the development of an critiquing expert system that would use formalized medical guidelines as a knowledge base. We formally defined processes and their enhancement which helped us to closely describe the functioning of the multi-agent system that simulates the processes and finally, we presented our vision of application of this approach in medicine.

Because such a direct usage of processes to control a multi-agent system has not been till now a not very explored area, there are several open issues: further improvement of the architecture of the MAS, implementation of planning and learning, or using more deliberative decision mechanisms for *executing agents*. In the following work we want to address some of them and prove the usefulness of this method by implementation of the working critiquing system that would help the physician with their work.

References

[1] R. Lenz and M. Reichert, "It support for healthcare processes - premises, challenges, perspectives," *Data Knowl. Eng.*, vol. 61, no. 1, pp. 39–58, 2007.

[2] X. Song, B. Hwong, G. Matos, A. Rudorfer, C. Nelson, M. Han, and A. Girenkov, "Understanding requirements for computer-aided healthcare workflows: experiences and challenges," in *ICSE '06: Proceedings of the 28th international conference on Software engineering*, (New York, NY, USA), pp. 930–934, ACM, 2006.

[3] A. Kumar, B. Smith, M. Pisanelli, A. Gangemi, and M. Stefanelli, "Clinical guidelines as plans: An ontological theory," *Methods of Information in Medicine*, vol. 2, 2006.

[4] A. G. Ellrodt, L. Conner, M. Riedinger, and S. Weingarten, "Measuring and Improving Physician Compliance with Clinical Practice Guidelines: A Controlled Interventional Trial," *Ann Intern Med*, vol. 122, no. 4, pp. 277–282, 1995.

[5] J. Cartwright, S. de Sylva, M. Glasgow, R. Rivard, and J. Whiting, "Inaccessible information is useless information: addressing the knowledge gap," *J Med Pract Management*, vol. 18, pp. 36–41, 2002.

[6] P. A. de Clercq, J. A. Blom, H. H. M. Korsten, and A. Hasman, "Approaches for creating computer-interpretable guidelines that facilitate decision support.," *Artificial Intelligence in Medicine*, vol. 31, pp. 1–27, 2004.

[7] Y. Shahar, S. Miksch, and P. Johnson, "The asgaard project: a task-specific framework for the application and critiquing of time-oriented clinical guidelines.," *Artificial Intelligence in Medicine*, vol. 14, pp. 29–51, 1998.

[8] S. Tu and M. Musen, "A flexible approach to guideline modeling," *Proc AMIA Symp.*, pp. 420–424, 1999.

[9] M. Peleg, A. Boxwala, and O. Ogunyemi, “Glif3: The evolution of a guideline representation format.,” *Proc AMIA Annu Fall Symp.*, pp. 645–649, 2000.

[10] J. Fox, N. Johns, A. Rahmanzadeh, and R. Thomson, “Proforma: A method and language for specifying clinical guidelines and protocols,” in *Amsterdam*, 1996.

[11] L. Dazzi, C. Fassino, R. Saracco, S. Quaglini, and M. Stefanelli, “A patient workflow management system built on guidelines.,” *Proc AMIA Annu Fall Symp*, pp. 146–150, 1997.

[12] A. W. Scheer and M. Nüttgens, “ARIS architecture and reference models for business process management,” in *Bus. Proc. Management, Models, Techniques, and Empirical Studies*, (London, UK), pp. 376–389, Springer-Verlag, 2000.

[13] M. Sierhuis, *Modeling and Simulating Work Practice*. PhD thesis, University of Amsterdam, 2001.

[14] N. R. Jennings, P. Faratin, T. J. Norman, P. O’Brien, and B. Odgers, “Autonomous agents for business process management,” *Int. Journal of Applied Artificial Intelligence*, vol. 14, no. 2, pp. 145–189, 2000.

[15] A. Moreno, A. Valls, and M. Marín, “Multi-agent simulation of work teams,” in *Multi-Agent Systems and Applications III: 3rd Int. CEEMAS*, (Prague, Czech Republic), June 16-18 2003.

[16] M. P. Singh and M. N. Huhns, “Multiagent systems for workflow,” *Int. Journal of Intelligent Syst. in Accounting, Finance and Management*, vol. 8, pp. 105–117, 1999.

[17] C. de Snoo, “Modelling planning processes with talmod,” Master’s thesis, University of Groningen, 2005.

[18] B. Bosansky, “A virtual company simulation by means of autonomous agents,” Master’s thesis, Charles University in Prague, 2007.

[19] A. Finkelstein, J. Kramer, B. Nuseibeh, L. Finkelstein, and M. Goedicke, “Viewpoints: A framework for integrating multiple perspectives in system development,” *Int. Journal of Software Eng. and Knowledge Engineering*, vol. 2, no. 1, pp. 31–57, 1992.

[20] R. G. Smith, “The contract net protocol: high-level communication and control in a distributed problem solver,” pp. 357–366, 1988.