Method for Rapid Prototyping of Societal Information Systems

Kuldar Taveter
Tallinn University of Technology,
Raja 15, 12618, Tallinn
Estonia
kuldar.taveter@ttu.ee

Hongying Du
University of South Carolina, 315
Main St. Columbia, SC 29208
USA
du5@email.sc.edu

Michael N. Huhns
University of South Carolina, 315
Main St. Columbia, SC 29208
USA
huhns@sc.edu

[1]*Abstract*—**We first define and explain the notions of rapid prototyping and societal information systems. Thereafter we introduce a design method appropriate for designing and rapid prototyping of societal information systems – agent-oriented modeling. Following, we describe a "proof-of-concept" case study of applying agent-oriented modeling to rapid prototyping of a societal information system for finding an appropriate physician. In the description, we first present analysis models and then show how they can be mapped to the respective design models. Finally, we explain how the resulting design constructs can be turned into the programming constructs of NetLogo for rapid prototyping. The article finishes by drawing conclusions on designing societal information systems.**

I. INTRODUCTION

THIS article is concerned with rapid prototyping of societal information systems. We first explain rapid prototyping and then discuss societal information systems. Rapid prototyping stands for implementing proof-of-concept prototypes in an agile way by directly mapping the modeling constructs to the constructs of a scripting environment like NetLogo or some agent-oriented environment like JADE. Previously we have investigated this technique in [5]. By societal information systems we mean large-scale information systems that gather information from hundreds, perhaps thousands, of nodes, each associated with a person, and then process and use the information to affect the behaviors of the people at the nodes. In today's world, a person's behavior is affected by means of social networking services, such as Facebook or Twitter. However, the amount of information to be processed can be overwhelming for users of such systems. To further automate the sharing and processing of information within a large social network, we are investigating the use of software agents – distributed reactive and proactive software entities representing and working on behalf of each person in the network. Such agents gather information from humans and other agents at the nodes of the network and aggregate and process it in ways that can augment the capabilities of the humans at the nodes. The resulting system is a kind of multi-agent system (MAS) [1], [2]. The key metaphor for such a MAS is interaction. MASs emphasize the design-time autonomy of the nodes and the importance of the environment in which the nodes interact with each other, which itself must often be designed [3].

The areas where societal information systems can help are regulation (e.g., banking), allocation of scarce resources (e.g., electric power, parking spaces, and emergency care), distributed situation assessment (e.g., traffic jams), system control (e.g., traffic management), and decentralized decision-making (e.g., finding a healthcare provider), which represent five kinds of problems that societies confront.

Engineering societal information systems requires methods different from those meant for engineering conventional information systems. A method of engineering societal information systems should first support the design of distributed systems consisting of autonomous, heterogeneous, and local nodes. Second, such a method should support the engineering of distributed systems that are open, adaptive, and intelligent. Societal information systems are *open* systems because members of the society (e.g., commuters, patients, or shoppers) may join and leave the system at any time. Societal information systems are *adaptive* systems, because they should react to their constantly changing environment, which for example can take the form of changes in traffic infrastructure, health insurance coverage, and product prices. We also term societal information systems as *intelligent* systems, because they reflect the "wisdom of crowds" when, for example, recommending a healthcare provider to a patient. In addition to the requirement of supporting the design of open, adaptive, and intelligent systems, a method for designing societal information systems should support the *purposefulness* and *understandability* of the design.

Considering the requirements outlined above, we have chosen agent-oriented modeling for designing societal information systems. Agent-oriented modeling as described in [4] is a holistic approach for analyzing, designing, and rapid prototyping of socio-technical systems consisting of humans and technical components. Its support for rapid prototyping of information systems [5] conforms well to the agile approach of developing software [e.g., 6].

[1] The first author expresses his gratitude to the Institute of International Education for supporting his work by a Fulbright Scholarship.

Other methodologies and tools have been shown to be useful for modeling and then developing large-scale agent-based systems, such as Gaia [7], Tropos [8], and O-MaSE [9]. In particular, the Organization-based Multiagent System Engineering (O-MaSE) metamodel defines the key concepts needed to design and implement multiagent systems to capture the organizational concepts identified in an organization metamodel. However, the agents in these systems are considered to have individual goals, rather than a combination of both individual and societal goals, on which we focus herein.

Other approaches of developing socio-technical systems for healthcare have emphasized a centralized approach, even when the individuals being assisted are decentralized [10].

This article focuses on designing societal information systems for applications requiring decentralized decision-making. In this article, we present as a "proof-of-concept" case study, the rapid prototyping of a societal information system for deciding on an appropriate physician. It is difficult to experiment with such information systems in a real human society, especially when patient health, privacy, and rights must be safeguarded. We therefore have relied on simulations for evaluating our prototypes.

The rest of this article is structured as follows. We first introduce the method we suggest to use for the rapid prototyping of societal healthcare information systems – agent-oriented modeling. This is followed by an overview of the analysis and design by agent-oriented modeling of our prototypical societal information system of finding a physician. We then describe how the modeling constructs of agent-oriented modeling can be mapped to the programming constructs of the NetLogo simulation platform [11]. The article concludes by drawing conclusions about engineering societal information systems, and particularly those for healthcare.

II. Related Work

Rapid prototyping of multi-agent systems has been addressed in [12, 13]. In [12], a MAS must be described for rapid prototyping by an organizational model which semantics is given in term of a formal framework. Similarly to our approach, [12] relies on simulation for evaluating the prototypes. The difference from our approach is that while our method is a lightweight agile approach, [12] uses formal transformations between models, which require the usage of a specialized software engineering tool. The article [13] provides a survey of approaches and then describes a prototype of a platform-independent metamodel for developing agent applications in a generalized manner. Similarly to agent-oriented modeling, [13] proposes mapping rules to transform platform-independent models into platform-specific models. Differently from us, their problem domain is workflow tasks.

III. Analysis and Design

A. Method

Agent-oriented modeling [4] comprises a set of canonical models, whose types are represented in Table I. In addition to representing each model with an abstraction layer (analysis, design, or prototyping), Table I maps each model to the vertical viewpoint aspect of interaction, information, or behavior. Each cell in the table represents a specific viewpoint. We next give an overview of agent-oriented models proceeding by viewpoints.

From the viewpoint of interaction analysis, the properties of roles are expressed by *role models* and the relationships between the roles – by an *organization model*. From the viewpoint of information analysis, a *domain model* represents the knowledge to be handled by the socio-technical system. From the viewpoint of behavior analysis, a *goal model* can be considered as a container of three components: goals, quality goals, and roles. From the viewpoint of interaction design, *agent models* transform the abstract constructs from the analysis stage, roles, to design constructs, agent types, which will be realized in the implementation process. From the same viewpoint, *interaction models* represent interaction patterns between agents of the given types. From the viewpoint of information design, agents' *knowledge models* are used for representing both private and shared knowledge by agents. From the viewpoint of behavior design, we model by *behavioral scenarios* how agents make decisions and perform activities [4]. A more detailed explanation of the model types can be found in [4].

Table I.
The model types of agent-oriented modeling

	Viewpoint aspect		
Abstraction layer	Interaction	Information	Behavior
Analysis	Role models and organization model	Domain model	Goal models
Design	Agent models and interaction models	Knowledge models	Behavioral scenarios
Prototyping	Interaction prototyping	Information prototyping	Behavior prototyping

B. Analysis

We begin the analysis of the societal healthcare system from the viewpoint of *behavior analysis* by deciding the system's purpose. A societal information system of healthcare can be viewed as a socio-technical system with the overall purpose "Allocate Healthcare Resources" among the members of the society. Our case study is limited to the allocation of healthcare resources of a particular kind – physicians. Achieving the functional goal "Allocate Healthcare Resources" is characterized by the quality goal "Maximal Societal Health", which determines the quality criterion according to which healthcare resources should be allocated in a society. A possible metric for this criterion is an average number of annual sick days per person in a society. Regardless of the global quality goal measurement, we achieve it by the decentralized agent-oriented method as described in Section *C*, rather than by a centralized method. The roles attached to the functional goal models – Patient, Healthcare Provider, and Government – constitute the major stakeholders in a healthcare system. The goal model of a societal information system of healthcare is represented in Fig. 1. In the figure, rectangles stand for functional goals and clouds for quality goals. Roles are denoted by stick figures.

We next elaborate the goal tree as follows. In our limited case study, allocating healthcare resources entails finding a healthcare provider – physician – for each patient, providing care, evaluating care, and recommending healthcare providers to other patients. Each of these sub-goals represents a particular aspect of allocating healthcare resources.

In addition to functional goals, we need a number of quality goals in the goal model. First, we add "Quickly" pertaining to the functional goal "Find Healthcare Provider". The meaning of this quality goal is obvious. Second, we express that a healthcare provider to be found should be appropriate. In the analysis phase, we do not need to specify the precise meaning of the "Appropriate" quality goal, because it is elaborated in the design phase where we decide how exactly appropriateness can be represented and what algorithms and software solutions are available for supporting it. However, it is highly relevant to capture this quality goal in analysis models that are used in round-table discussions between customers and other non-technical stakeholders and information system developers.

As we plan to use social networking for finding a healthcare provider, we elaborate the "Find Healthcare Provider" functional goal into two sub-goals: "Ask Friends" and "Choose". We characterize the second of these functional goals by the "Good Quality Provider" quality goal, meaning that the healthcare provider who offers the best overall quality should be chosen. Again, we do not worry here how to measure the overall quality and postpone this until the design phase, where we decide technical means for supporting quality appraisals and social networking.

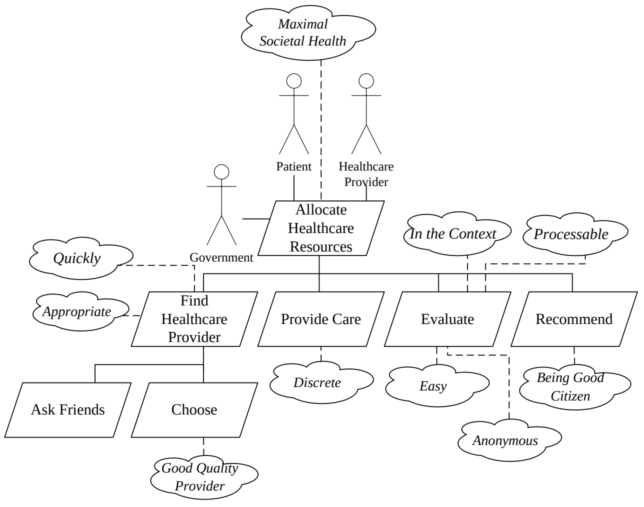

Fig. 1. The goal model

The "Provide Care" functional goal is characterized by the "Discrete" quality goal with an obvious meaning. The "Evaluate" functional goal is modified by four quality goals. The quality goal "In the Context" represents that evaluation has to occur in the context of receiving the service, preferably before leaving the facilities of the healthcare provider or at least on the same day. This quality goal implies the need to introduce some context awareness into the system. The "Easy" quality goal means that evaluating a healthcare provider should be easy for a patient. Potential design decisions for achieving this quality goal involve using a cell phone or a specialized device for evaluation. The "Processable" quality goal means that the evaluation should be presented in a form amenable to computer processing. What exactly it means is again left up to the design. For example, depending on the system design, it could mean that all evaluations should be expressed on a scale from 1 to 5. Or alternatively, if the system includes a data-mining component, it could mean that evaluations can be expressed in a natural language that is controlled or restricted to a smaller or greater extent. Finally, the "Anonymous" quality goal expresses that no evaluation by a patient should identify the patient.

The "Recommend" functional goal is modified by the "Being Good Citizen" quality goal, meaning that recommending healthcare providers to other patients is seen as a voluntary activity benefiting a society as a whole.

Having defined the goals for the system, we now proceed to the viewpoint of *interaction analysis* by deciding the roles that are required for achieving the goals. In the given case study the roles are obvious: Patient and Healthcare Provider. We represent each of these roles in terms of its responsibilities and constraints. The resulting role models are described by Table II and Table III. There is also a third role – Government – but its modeling is not relevant for the system to be designed.

TABLE II.
THE ROLE MODEL FOR PATIENT

Role	**Patient**
Description	The role of a patient in U.S. healthcare
Responsibilities	Ask friends for recommendations
	Choose a healthcare provider
	Receive care
	Evaluate care
	Recommend healthcare providers
Constraints	A patient should choose the best available healthcare provider
	The evaluation by a patient should not reveal the identity of the patient
	The evaluation by a patient should be processable by computers
	The evaluation by a patient should be given in the context of receiving the care
	A patient should be willing to help his/her friends

TABLE III.
THE ROLE MODEL FOR HEALTHCARE PROVIDER

Role	**Healthcare Provider**
Description	The role of a healthcare provider in U.S. healthcare
Responsibilities	Provide medical service
Constraints	Medical service should be provided in a discrete manner

TABLE IV.
THE ROLE MODEL FOR ASSISTANT

Role	**Assistant**
Description	The role of a patient's assistant in healthcare
Responsibilities	Ask friends for recommendations
	Choose a healthcare provider
	Assist in evaluating the care
Constraints	The best possible healthcare provider should be chosen
	Appropriate for the given problem healthcare provider should be found
	Healthcare provider should be found as quickly as possible
	The evaluation by a patient should not reveal the identity of the patient
	The evaluation by a patient should be processa-ble by computers

According to the metaphor of hiring new staff proposed in [14], we next ask what positions would be needed to be filled if one was to hire more staff to handle the problem. The answer is that the Assistant position would need to be filled, because some help would make finding a healthcare provider easier for a patient. To reflect this, we complement the goal model with a new Assistant role. The Assistant role takes up the responsibilities of asking friends for recommendations, choosing a healthcare provider, and assisting in evaluating the care. The Assistant role is modeled in terms of its responsibilities and constraints as described by Table IV.

We proceed by modeling the organizational structure of the societal information system to be developed. The organization model is depicted in Fig. 2. All three major relationship types – peer, benevolence, and control – are represented in the organization model. First, as we are addressing social networks, there is the "IsPeerTo" relationship attached to the Patient role. Second, since healthcare providers provide services to patients, there is the "IsBenevolentTo" relationship between the roles Healthcare Provider and Patient. Third, in finding healthcare providers, a patient needs help that is provided by his/her assistant. This is reflected by the "Controls" relationship between the roles Patient and Assistant.

The organization model also shows that there can be different types of healthcare providers, out of which physicians and hospitals are modeled in the figure. As stated above, the design of the societal information system focuses on patients finding physicians.

Visualizing the organization model assisted us in exploring three kinds of social networks, which vary according to how the "isPeerTo" or "being friend to" relationship of the Patient role is instantiated. The three kinds of relationships are the following ones:

- Random network: the relationships between pairs of patients are created randomly.
- Small-world network: most nodes are not neighbors to one another, but most nodes can be reached from any other node by a small number of hops [15].
- Scale-free network: the shortest paths between nodes flow through hubs, and if a peripheral node is deleted, it is unlikely that this will interfere with passing a message between other peripheral nodes. We follow the Barabási–Albert model [16] to construct a scale-free network for our simulation. Scale-free network is a common model for a collaboration network.

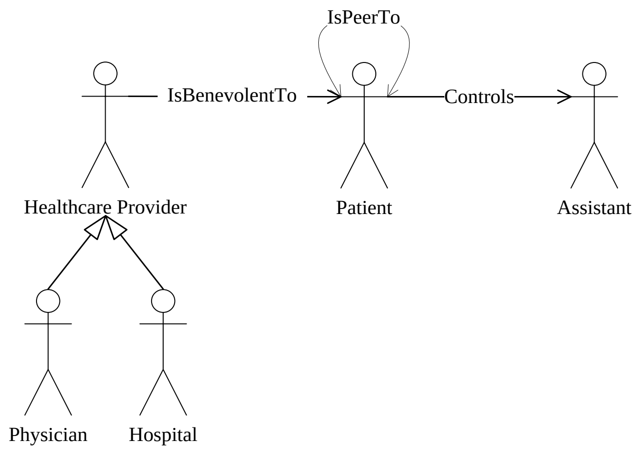

Fig. 2. The organization model

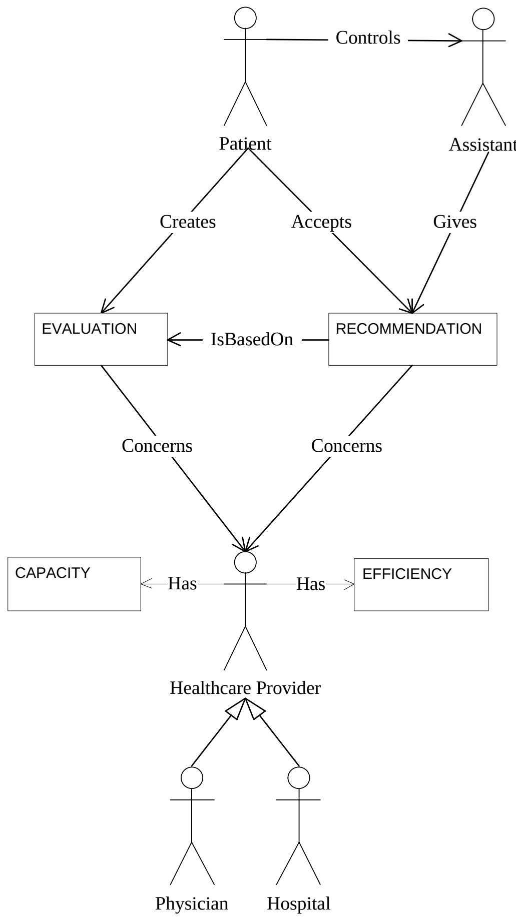

Fig. 3. The domain model

Based on the above behavior and interaction analysis, we proceed to the viewpoint of *information analysis* by addressing the knowledge to be represented within the system. We do this by identifying the types of knowledge entities related to the roles. As each healthcare provider has predefined capacity and efficiency, we attach the Capacity and Efficiency knowledge entity types to the Healthcare Provider role. According to the role models represented in Tables II and IV, patients evaluate and their assistants recommend healthcare providers. We accordingly place the Evaluation and Recommendation knowledge entity types between the roles Patient, Assistant, and Healthcare Provider. In this way we obtain a domain model, represented in Fig. 3, from the organization model.

C. Design

Having created the goal model and the models of relevant roles, as well as the organization model and domain model, we have completed the analysis phase of agent-oriented modeling. We now proceed with design and decide from the viewpoint of *interaction design* the agent types for the prototype. In the prototypical societal information system to be designed, the role Assistant should obviously be mapped to the Assistant Agent software agent type. Since a patient is a real human that is treated by another real human – a physician – we map both the roles Patient and Healthcare Provider to the Human Agent type. The software system boundary of the societal information system is obviously going to be between the roles Patient and Assistant. Regarding the Healthcare Provider role, because of the need to limit the scope of our case study, the societal information system to be designed by us does not include any software agents for healthcare providers.

Finding a physician involves interactions among Assistant Agents representing patients. We represent these interactions as an interaction protocol among agents of the type Assistant Agent. It is appropriate to remind here that the difference between interaction protocol and other kinds of interaction models is that interaction protocol models some aspects of the agent behaviors along with their interactions [4].

Interaction protocol is an important model for the societal healthcare information system, because it describes the patient's strategy of choosing a physician. Visualizing interaction protocols assisted us in exploring the following four possible strategies of choosing a physician:

- Random strategy. The patient's Assistant Agent randomly chooses a physician.
- The "Choose one" strategy. The patient's Assistant Agent chooses the best physician according to the patient's evaluations for physicians. If the patient has no evaluations, the Assistant Agent asks her friends' Assistant Agents for recommendations.

In addition to the random and "Choose one" strategies, the "Borda voting" [Borda voting is a single-winner election method in which voters rank candidates in order of preference, named for the 18th-century French mathematician and political scientist Jean-Charles de Borda, who devised the system in 1770] and "Add and minimize" strategies were explored. In the "Borda voting" strategy, the Assistant Agent of the patient asks for recommendations and the Assistant Agents of the patient's friends give back recommendations just like in the "Choose one" strategy. Physicians then earn points which are equal to the number of physicians whose evaluations are worse than the given physician. For each friend who gives a recommendation, the points are calculated individually for each physician with whom the friend has experience. Then the Assistant Agent of the patient calculates the total points of each recommended physician and chooses the one with the highest points. In the "Add and minimize" strategy, the Assistant Agent of the patient calculates the mean value of all the non-zero evaluations for each recommended physician and chooses the one with the minimal mean evaluation. These strategies are explained in more detail in [17]. The interaction protocol modeled in Fig. 4 describes interactions between patients' Assistant Agents according to the "Choose one" strategy.

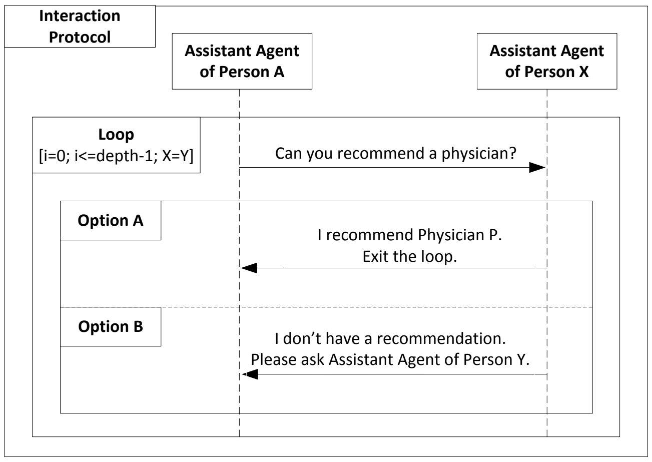
Interaction Protocol
Assistant Agent of Person A
Assistant Agent of Person X
Loop [i=0; i<=depth-1; X=Y]
Can you recommend a physician?
Option A
I recommend Physician P. Exit the loop.
Option B
I don't have a recommendation. Please ask Assistant Agent of Person Y.

Fig. 4. The interaction protocol for "Choose one" strategy

We also created similar interaction protocols for the two other strategies of choosing a physician by the patient. The interaction protocol shown in Fig. 4 models that in case of the "Choose one" strategy, the Assistant Agent acting on behalf of the patient's friend may deal with the request in one of the following ways:

o Reply with a recommendation.

o Provide the requesting agent with the address of the Assistant Agent of one of its principal's friends if there is no recommendation to give. This process continues recursively until the first recommendation is received or until all the friends until the maximum forwarding depth have been asked. The forwarding depth is defined as follows: the originator's friends are at depth 1; the originator's friends' friends at depth 2, and so on. This means that the interaction protocol is recursive, which is represented by the "Loop" behavioral construct, whose repeating condition is presented in the programming style. A friend's Assistant Agent may also ignore a request, in which case neither of the Option boxes shown in Fig. 4 is chosen.

From the viewpoint of *behavior design*, to model the behaviors of agents of the decided types, we transform responsibilities of the roles into activities attached to the agent types. This results in behavioral scenarios for agents playing the roles Patient, Assistant, and Physician.

Table V represents the behavioral scenario for the role Assistant played by a software agent of the type Assistant Agent when finding a physician for its principal. The behavioral scenario represented in Table V models that "Find a physician" and "Evaluate" activities are performed sequentially. In the societal information system for healthcare being designed by us this is always the case because the Assistant Agent does not perform any activities between these activities while a patient is attended by a physician.

Another aspect of the Assistant Agent's behavior in choosing a physician deals with what the agent should do if the physician is not available on the given day. By using agent-oriented modeling for representing agent behaviors, we have decided to consider the following three waiting strategies of a patient:

- Waiting. The patient's Assistant Agent chooses the best physician by adopting one of the physician choosing strategies that were explained above and sticks to its choice. If the physician is busy, the patient will still make an appointment with the physician and will wait until the physician becomes available.

- No waiting. If the physician chosen is busy, the patient's Assistant Agent will choose a physician randomly according to the "Random" strategy or the next best physician according to the other physician choosing strategies until it finds an available physician.

- Waiting with limit. If the physician chosen is not available, the patient's Assistant Agent will check whether the physician could be reached in a certain number of days. If it is possible, the patient will make an appointment and wait. If not, the Assistant Agent will choose another physician according to the rules of the same waiting strategy. If no physician is available in a certain number of days, the Assistant Agent will choose a physician, who requires the minimum number of days to wait.

As is modeled in Table V, the activity "Evaluate" performed by the Assistant Agent is triggered by a patient leaving the physician's office. This reflects the "In the Context" quality goal, which in Fig. 1 is attached to the "Evaluate" functional goal. How the leaving is to be perceived is left to more detailed design, which we do not address here because of the scope of this paper. A possible solution may involve the timeframe of the physician office visit in question and perceiving the geographical coordinates of the patient [18].

TABLE V.
THE BEHAVIORAL SCENARIO FOR AN ASSISTANT AGENT PLAYING THE ROLE OF ASSISTANT

BEHAVIORAL SCENARIO						
Role			Assistant			
Agent type			Assistant Agent			
DESCRIPTION						
Trigger	**Condition**	**Step**	**Activity**	**Other roles/agent types involved**	**Knowledge entities**	**Relevant goals (quality goals)**
Request by the patient	Sequential	1	Find a physician	Patient/Human Agent, Assistant/Assistant Agent	Recom-mendation	Find healthcare provider (Quickly, Appropriate, Good Quality Provider)
Patient leaves the physician's office		2	Evaluate	Patient/Human Agent, Assistant/Assistant Agent	Efficiency, Evaluation	Evaluate (In the Context, Proces-sable, Anonymous, Easy)

In accordance with another quality goal – “Quickly” – which was introduced by the goal model shown in Fig. 1, we assume that a patient is willing to get healthy as soon as possible.

Finally, distinguishing between private and public knowledge entities from the viewpoint of *information design* is straightforward, because the knowledge entity Evaluation is private to the patient and Assistant Agent helping him/her, while the knowledge entity Recommendation is shared between different patients and instances of Assistant Agent. Similarly, the knowledge entity Efficiency is private to each Healthcare Provider, but at the same time naturally forms a basis for how patients evaluate healthcare providers.

IV. MAPPING AGENT-ORIENTED MODELS TO NETLOGO

In this section, we give an overview of some basic programming constructs of NetLogo and show how agent-oriented models described in Section III can be mapped to them.

NetLogo [11] is a programmable modeling environment for simulating natural and social phenomena. System designers using NetLogo can give instructions to hundreds or thousands of agents all operating independently. This makes it possible to explore the connection between the micro-level behavior of individuals and the macro-level patterns that emerge from the interactions between many individuals. Because of this, NetLogo is a suitable environment for rapid prototyping of societal information systems. In this subsection, we give an overview of some basic programming constructs of NetLogo and show how agent-oriented models can be mapped to them.

The NetLogo world is made up of agents that can follow instructions. Each agent can carry out its own activity simultaneously with the activities performed by other agents. Detailed overview of the types of agents in NetLogo can be found from [11].

The programming constructs of NetLogo are seemingly quite different from the modeling concepts of agent-oriented modeling. However, at a closer look, the NetLogo programming constructs can be understood as the ones defining agents and their environments. As has been pointed out in [4], an environment can be either a real physical environment or a virtual environment. An environment simulated by means of NetLogo is an example of a virtual environment.

In NetLogo, knowledge entities of agent-oriented modeling that are private for specific agents can be represented by means of agents’, which in NetLogo are called turtles, local variables and knowledge entities shared by agents – by global variables. The relationships between knowledge entities are represented in NetLogo as calculations or derivations involving the respective NetLogo variables. Acquaintances (communication pathways) between agents can be simulated as links between NetLogo turtles. The environment in which the agents are situated can be simulated as a set of patches. All in all, such a view is consistent with the one treating both agents and their environments as first-class citizens [19].

When we turn from the level of instances to the level of types, we also discover obvious mappings between agent-oriented modeling and NetLogo. For example, roles of agent-oriented modeling are mapped to agent types, which are in turn mapped to breeds of turtles. Similarly, private and shared knowledge entities from the agents’ knowledge models are respectively mapped to turtles’ local variables and global variables of NetLogo. The types of organizational relationships between agents, such as control, benevolence, and peer, correspond to breeds of links between turtles. Behavioral scenarios of agent-oriented modeling correspond to procedures of NetLogo with the difference that the procedures typically define the behavior of a set of turtles rather than just one turtle. The biggest disadvantage of using NetLogo for simulating multi-agent systems is that NetLogo does not directly support interactions between agents, and interactions therefore have to be implemented indirectly through using global variables. The mapping between the concepts of agent-oriented modeling and the programming constructs of NetLogo required for rapid prototyping is presented in Table VI.

TABLE VI. THE MAPPINGS BETWEEN AGENT-ORIENTED MODELING AND NETLOGO

Modeling concept of analysis	*Modeling concept of design*	*Programming construct of NetLogo*
Role (role model)	Agent (agent model)	Turtle
Role (role model)	Agent type (agent model)	Turtle breed
Goal (goal model)	Behavioral scenario	Procedure
Domain entity (domain model)	Private knowledge item (knowledge model)	Local (to turtle) variable
Domain entity (domain model)	Shared knowledge item (knowledge model)	Global variable
Relationship between roles in a domain model (organization model)	Acquaintance (agent acquaintance model)	Link between turtles
Relationship between domain entities (domain model)	Relationship between knowledge items (knowledge model)	Calculation or derivation involving the knowledge items
Relationship type (domain model)	Relationship type (knowledge model)	Link breed

V. Conclusions

We proposed a method for designing and rapid prototyping of societal information systems. For developing societal information systems, both their social and technical aspects should be considered. We have chosen to use agent-oriented modeling for developing societal information systems, because this approach explicitly addresses the design of socio-technical systems where the activities of humans are supported by software agents. What makes agent-oriented modeling particularly appropriate for developing societal information systems is that the design process starts with specifying goals for a socio-technical system as a whole and then with defining roles required for achieving the goals. Technical and social subsystems of the system are identified only later in the design process when roles are mapped to the types of agents enacting them. This is also a stage when the decisions of architectural design can be made by mapping roles to different possible configurations of agents. Alternatively, the system architecture can be designed already when deciding roles. Another advantage of agent-oriented modeling is that it enables a system designer to address the problem domain of an information system from three balanced perspectives – information, interaction, and behavior – and at three abstraction layers.

Designing the prototypical societal information system confirmed that agent-oriented modeling can effectively combine models presenting the overall view of the information system to be designed – goal model, organization model, and domain model, with the models representing the perspectives of individual participants – role and interaction models, behavioral scenarios, and agents' knowledge models. This supports well the requirements for autonomy, heterogeneity, and locality of individual members, while achieving the goals set for the system as a whole. At the same time, the usage of agent-oriented modeling ensures that agents implemented and deployed by all of the participants in the information system are designed in a uniform way to behave benevolently and safely rather than maliciously. As a result, each patient can in a secure and fast manner launch and personalize his/her own healthcare agent and can task it to find a physician.

The results from performing simulations with the "proof-of-concept" societal information system for finding a physician are described in [17]. One main conclusion is that the number of annual sick days per person is decreased by 6.2%-27.1% using some strategies developed in the system compared to using random strategy.

Acknowledgements

We express our gratitude to Professor Leon Sterling from Swinburne University of Technology, Australia for comments and suggestions on the draft of this paper.

References

[1] Huhns, M. N. and Stephens, L. M. (1999). Multiagent systems and societies of agents. In G. Weiss (ed.), *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*. Cambridge, MA, and London, England: MIT Press, Chapter 2.

[2] Wooldridge, M. (2009). *An Introduction to Multiagent Systems*. 2nd Edition. Chichester, UK: John Wiley & Sons.

[3] Huhns, M. N. (2009). From DPS to MAS to ...: continuing the trends. In C. Sierra, C. Castelfranchi, K. S. Decker, and J. S. Sichman (eds.), *8th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2009),* Budapest, Hungary, May 10-15, Volume 1 (pp. 43-48). New York, NY: ACM.

[4] Sterling, L. and Taveter, K. (2009). *The Art of Agent-Oriented Modeling*. Cambridge, MA, and London, England: MIT Press.

[5] Taveter, K. and Sterling, L. (2008). An Expressway from Agent-Oriented Models to Prototype Systems. In M. Luck and L. Padgham (eds.), *Agent Oriented Software Engineering VIII, Proceedings of the 8th International Workshop on Agent-Oriented Software Engineering (AOSE 2007),* LNCS 4951, 149-166. Berlin, Germany: Springer-Verlag.

[6] Ambler, S. W. (2002). *Agile Modeling*. Chichester, UK: John Wiley & Sons.

[7] Wooldridge, M., Jennings, N. R., and Kinny, D. (2000) The Gaia Methodology for Agent-Oriented Analysis and Design. *Journal of Autonomous Agents and Multi-Agent Systems, 3*(3), 285-312.

[8] Bresciani, P., Perini, A., Giorgini, P.,Giunchiglia, F., and Mylopoulos, J. (2004). Tropos: An Agent-Oriented Software Development Methodology. *Autonomous Agents and Multi-Agent Systems, 8,* 203–236.

[9] DeLoach, S. and García-Ojeda, J.C. (2010). O-MaSE: a customisable approach to designing and building complex, adaptive multi-agent systems. *Int. J. Agent-Oriented Software Engineering, 4*(3), 244-280.

[10] Berg, M. (1999). Patient care information systems and health care work: a sociotechnical approach. *Int J Med Inform. 55*(2), 87-101.

[11] Wilensky, U. (1999). *NetLogo*. Last accessed on February 16, 2012, from *http://ccl.northwestern.edu/netlogo/*. Center for Connected Learning and Computer-Based Modeling. Evanston, IL: Northwestern University.

[12] Hilaire, V., Koukam, A., Gruer P., and Müller, J.-P. (2000). Formal Specification and Prototyping of Multi-agent Systems. In A.Omicini, R. Tolksdorf and F. Zambonelli (eds.), *Engineering Societies in the Agent World, First International Workshop, ESAW 2000, Berlin, Germany, August 21, Revised Papers*, LNCS 1972, 114-127. Berlin, Germany: Springer-Verlag.

[13] Hahn, C., Madrigal-Mora, C., and Fischer, K. (2009). A platform-independent metamodel for multiagent systems. *Journal of Autonomous Agents and Multi-Agent Systems, 18,* 239–266.

[14] Wilmann, D and Sterling, L. (2005). Guiding agent-oriented requirements elicitation: HOMER. In *The 2005 NASA / DoD Conference on Evolvable Hardware (EH 2005),* 29 June - 1 July, Washington, DC, USA (pp. 419–424). Washington, DC: IEEE Computer Society.

[15] Watts, D. J. and Strogatz S. H. (1998). Collective dynamics of 'small-world' networks. *Nature, 393,* 440-442.

[16] Barabasi A.-L. and Albert R. (1999). Emergence of Scaling in Random Networks. *Science, 286*(5439), 509–512.

[17] Du, H., Taveter, K. and Huhns, M. N. (2012). Simulating a Societal Information System for Healthcare. In *Proceedings of the 6th International Workshop on Multi-Agent Systems and Simulation (MAS&S), Wrocław, Poland, September 9-12, 2012,* to be published.

[18] Nguyen, T., Loke, S. W., Torabi, T., and Lu, H. (2011). PlaceComm: A framework for context-aware applications in place-based virtual communities, *Journal of Ambient Intelligence and Smart Environments, 1*(3), 51–64.

[19] Weyns, D., Omicini, A., and Odell, J. (2007). Environment as a first class abstraction in multiagent systems. *Autonomous Agents and Multiagent Systems, 14*(1), 5–30.