2014 IEEE 38th Annual International Computers, Software and Applications Conference Workshops

Heuristics for Designing and Evaluating Socio-Technical Agent-Oriented Behaviour Models with Coloured Petri Nets

Msury Mahunnah*, Alex Norta*, Lixin Ma*†, Kuldar Taveter*

*Department of Informatics, Tallinn University of Technology, Tallinn, Estonia,
†Department of Computer Science, University of Shanghai for Science and Technology, Shanghai, China

Email: msurym@gmail.com; alex.norta@gmail.com; malixin.usst@gmail.com; kuldar.taveter@ttu.ee

***Abstract*—Software agents are a means to support socio-technical decentralised systems that increase the complexity of daily life. Designing multi-agent systems involves modelling methods for which it is currently not possible to check for soundness before a technical implementation. To improve the design process, the agent models require a mapping to a formalisation that is sufficiently expressive to represent equivalent model properties. The formalized presentation must cater for evaluating the model soundness, simulation and performance experimentations with different test data. The paper gives a set of mapping heuristics from agent models to a sufficiently expressive formalisation representation that follows a real life running case stemming from the healthcare domain.**

***Keywords*—Socio-Technical, Agents, Coloured Petri Nets, Heuristics, Behaviour, Design, Evaluation.**

I. Introduction

Our society is becoming increasingly dependant on complex information technology (IT) systems for carrying out daily activities. The complexity of IT systems, mainly, stems from the integration and orchestration of independently managed software systems that are distributed in dynamic environments [1], such as healthcare, aviation, air traffic control, telecommunications, and so on. In addition, the behaviour of people who work across organizational, geographical, cultural and temporal boundaries [2] influences the complexity of such socio-technical IT systems [3] and thus, poses a great engineering challenge. We define a socio-technical system as an approach to complex organizational work design that recognizes the interaction between people and technology in workplaces.

In recent years, researchers have undertaken various studies in modelling the behaviour and knowledge sharing, of socio-technical systems, among interacting technical, societal and organisational aspects. These studies have focus on domains such as healthcare [4], military [5] and sociology [6], [7] using an agent-oriented paradigm [8]. The latter is a top-down holistic approach for modelling socio-technical systems by engaging all stakeholders during the analysis and design phases of a system's development life cycle. However, a gap exists in formalising and evaluating agent-oriented behaviour, knowledge and interaction models before the actual implementation of these kinds of systems.

In this paper, we fill the identified gap by answering the research question, how to systematically formalise and evaluate agent-oriented behaviour models for socio-technical systems? To establish complexity-reducing separation of concerns, we deduce the following sub-questions: What is a suitable way for conceptualizing the behaviour of socio-technical agent-oriented systems based on a set of heuristics? What formalization is suitable for the first syntactically correctly designed agent-oriented behaviour models? What means exist to evaluate the soundness of agent-oriented behaviour models? This set of sub-questions assumes that a syntactic designing of agent-oriented behaviour models precedes the mapping to formalizations that carry equivalent model properties.

The paper structure is as follows. Section II describes a running case from the healthcare domain that helps in clarifying what artefacts we consider for this paper. Section III presents an agent-oriented goal model and behaviour-interface model for capturing socio-technical system behaviour. Section IV gives mapping heuristics towards a formalization and evaluation of agent-oriented behaviour models. Section V shows and explains the resulting formalised model of the running case and evaluates simulation results. Section VI presents related work and finally, Section VII gives the conclusion and provides future work.

II. Running Case from the Healthcare Domain

Healthcare organizations aim to provide better quality of care by improving the information logistics among caregivers and patients, who work in a distributed way. In this paper, we consider a case study [9] for reporting Critical Laboratory Results (CLRs) to an appropriate caregiver from the North Estonian Medical Centre (NEMC) laboratory.

The case study identifies two weaknesses in the procedure for reporting CLRs at the NEMC laboratory. First, the procedure for reporting CLRs involves many people. This leads to two major problems: (1) high risk of human errors, (2) delay in reporting CLRs. A second weakness arises from the handling of CLRs similar to Normal Laboratory Results (NRLs). When a doctor who orders the laboratory tests is unreachable by phone, the laboratory guidelines suggest that staff make a phone call instead to the departmental nurse who tries to find another appropriate doctor. If the departmental nurse is also unreachable, the CLRs are sent to the Hospital Information System (HIS) comparable to NRLs. The delay poses the risk that patients do not receive adequate treatment in a state of emergency.

Results of the case study in [9] propose to improve the current system for reporting CLRs as follows. NEMC must

978-1-4799-3578-9/14 $31.00 © 2014 IEEE
DOI 10.1109/COMPSACW.2014.74

consider mobile technologies, to accurately identify the location of caregivers. The new intelligent information system is a socio-technical system, i.e., a software intensive system that has defined operational processes followed by human operators and that operates within an organization [10]. The envisioned socio-technical system specifies roles for human agents in healthcare organizations, such as patients, nurses and doctors. The human agents receive support by software agents that we define [8] as an entity that performs a specific activity in an environment of which it is aware, and that can respond to changes.

The execution of rules determine the behaviour of a software agent [11]. The rules execute upon detection of changes in the environment, such as event occurrences. Events may also stem from other collaborating agents. The dynamic environment influences the possible reachable states of agents after executing activities. For instance, when the doctor for receiving CLRs is unavailable, the socio-technical agent proactively identifies and suggests another appropriate caregiver according to availability, medical knowledge and speciality.

In our previous work [9], we evaluate the analysis- and design models for socio-technical systems by engaging domain experts, i.e., healthcare professionals. In this paper, we suggest heuristics for evaluating syntactical correctness and soundness of socio-technical agent-oriented behaviour models by mapping to a formalization that carries equivalent model properties.

III. Behaviour Modelling

The analysis of socio-technical systems where humans receive support from intelligent software agents, must follow an appropriate methodology. Various Agent Oriented Software Engineering (AOSE) methodologies exist with a technical emphasis on designing systems consisting of software agents, e.g., Tropos [12], MaSE [13], or Prometheus [14]. In [8], the described Agent-Oriented Modelling (AOM) method is a socio-technical approach that includes features similar to those in mentioned AOSE methodologies while taking into account the combination of human- and man-made agents in the system design process. In this section, we present two AOM model types, i.e., the goal model and behaviour interface model, that capture important socio-technical behavioural features from the running case.

The goal model serves as a container for three main components: functional requirements commonly referred to as goals, roles, and non-functional requirements. The latter has two categories, quality goals for non-functions requirements related to software and emotional goals for those related to humans. Parallelograms represent goals, sticky men are roles, clouds are quality goals and hearts are emotional goals as depicted in Figure 1. Goal models serve as communication media between technical and non-technical stakeholders to establish a better understanding of the problem domain. The goal model starts with an overall objective of the socio-technical system that is known and clear to all stakeholders. Goals decompose into sub-goals where each sub-goal represents some aspect for achieving its parent goal [15]. Note that the lowest sub-goal must be atomic.

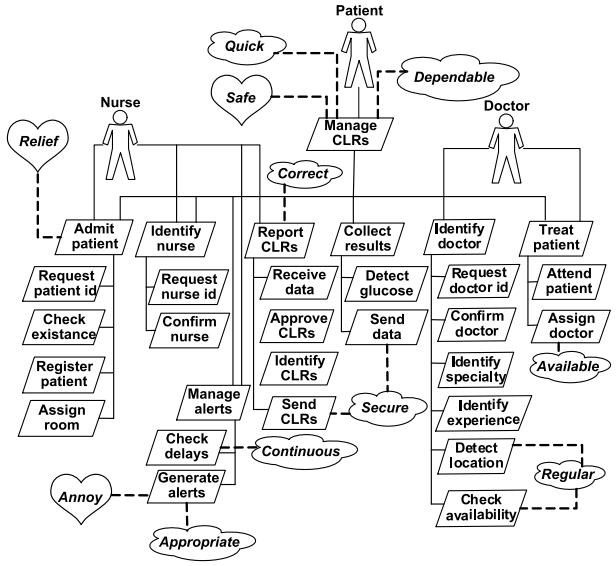


Fig. 1. Goal model of the socio-technical running healthcare case.

In the goal model of Figure 1, we first present the uppermost goal, viz., *Manage CLRs* with the attached role of *Patient* who is the focus of the analysis. The emotional goal *Safe*, and two quality goals *Quick* and *Dependable* are also attached to the main goal. The latter means to avoid service failures that are more frequent and more severe than acceptable [16]. The main goal *Manage CLRs* splits into seven sub-goals: *Admit patient*, *Identify nurse*, *Manage alerts*, *Report CLRs*, *Collect results*, *Identify doctor* and *Treat patient*. In addition to the role *Patient* who is the only role responsible for achieving the sub-goal *Collect results*, the role *Nurse* is responsible for the first four sub-goals and the role *Doctor* is responsible for the last two sub-goals. Each of the identified sub-goals has further refining third level sub-goals that are the lowest-level sub-goals for this running case. These sub-goals represent the activities of the socio-technical system. Parallel to the process of breaking down the sub-goals, we also identify suitable quality goals and emotional goals. For example, the sub-goal *Admit patient* in Figure 1 has the emotional goal *Relief*, meaning execution of the four activities of *Request patient ID*, *Check existence*, *Register patient* and *Assign room* targets at ensuring the patient feels a relief during the admission process.

Goal models focus on identifying functional and non-functional requirements of the whole socio-technical system rather than simple activities conducted by individual agents. During the design phase, behaviour models for individual agents facilitate the refinement of the goal model resulting from the analysis of the running case. A behaviour model in AOM has two parts: an agent behaviour model coupled with a behaviour interface model [8]. The former describes the rule-based behaviour of an agent, while the latter focuses on identifying the activities associated triggers, preconditions, and postconditions.

Table I presents the behavioural interfaces of four important activities in fulfilling the goal *Admit patient*. Each activity must have one trigger and at least one postcondition. Preconditions may either exist or not, depending on the nature of

TABLE I. BEHAVIOURAL INTERFACES OF ACTIVITIES FOR THE SUB-GOAL "ADMIT PATIENT"

Activity	Trigger	Preconditions	Postconditions
Request patient ID	New case detected		Received Patient ID
Check existence	Patient ID received	ID found	Registered patient
Register patient	Patient ID received	ID not found	Updated patient DB Registered Patient
Assign room	Patient registered	Room is available	Assigned room

the corresponding activity, e.g., the activity *Request patient ID* has only one trigger and one postcondition without any precondition. The execution of an activity is either triggered by the occurrence of an event, or by a pre-condition after the occurrence of the event. For example, the activity *Assign room* has three interfaces, *Patient registered* as a trigger, *Room is available* as precondition and *Assigned room* as postcondition. The given interface for the activity *Assign room* assumes room availability before patient registration. If the room is available after the registration of a patient then *Patient registered* becomes the precondition and *Room is available* as the trigger. In other words, the trigger and precondition may exchange their roles at runtime.

The behaviour interface models, designed by the domain experts, require a mapping to a formalization for an evaluation that ensures the models are sound before a technical implementation. The following section provides a step-by-step procedure for formalising agent-oriented behaviour models.

IV. MAPPING AGENT-ORIENTED BEHAVIOUR MODELS TO A FORMALIZATION

In order to formulate sound agent-oriented behaviour models for the socio-technical system, it is important to map AOM models to a formal and deterministic notation that allows for a strong evaluation. Consequently, we consider for Colored Petri Nets [17] (CPN) with mature tool support[1] as a mapping target. CPN is a graphical oriented language for design, specification, simulation and verification of systems. CPN has an intuitive, graphical representation that consists of a set of modules (pages), each containing a network of places, transitions and arcs. The modules interact with each other through a set of well-defined interfaces in a similar way as known from many modern programming languages. Places may hold multiple tokens that carry colour, i.e., attributes with values. Transitions fire when all input places hold the required sets of tokens and produce condition-adhering tokens into output places.

We next explain the mapping between AOM and CPN that Table II summarizes. Places and transitions are connected by directed arcs denoting the flow during the execution of activities and resources in AOM. Rectangles depict transitions that represent simple activities performed by agents. Ovals depict places that may either be attached with an outgoing arc to the transition or incoming arc from the transition. The former represents a trigger or precondition while the latter represents the postcondition of given activity in AOM. Double-boarded rectangles depict modules that represent goals in the socio-technical system that can further be broken-down into simpler sub-goals or activities. During the enactment of a CPN model, flow of control passes to the sub-goals or activities (in the AOM equivalent) associated with a parent goal represented as module. This way, a CPN model represents a hierarchical structure of the goal model in AOM. When mapping agent-oriented behaviour models to CPN, behaviour interface models as shown in Table I represent each identified activity of the socio-technical system found in the goal model of Figure 1, i.e., identifying and deciding about the triggers, preconditions and postconditions of each activity.

[1]http://cpntools.org/

TABLE II. NOTATIONS FOR MAPPING AOM TO CPN

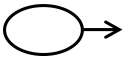
Notation	Name
	Connecting Arc
	Sub-Goal or Activity
	Trigger or Precondition
	Postcondition
	Goal

Table III describes a sample behaviour interface model for two consecutive activities. Activity 1 has Trigger 1, Precondition 1 and 2, and Postcondition 1. When mapping Activity 1 to CPN, it turns into a transition connected by arcs to four different places. Among them, three are incoming arcs from the three places representing Trigger 1, Precondition 1 and Precondition 2. The other connection to Activity 1 is an outgoing arc to a place representing Postcondition 1. Furthermore, Table III shows that Postcondition 1 triggers Activity 2 since its execution follows just after completion of Activity 1. Thus, making Postcondition 1 connected to Activity 2 by an outgoing arc and referred as a trigger named Trigger 2 by Activity 2. Following that, the outgoing arcs from Activity 2 connect to two places,namely, Postcondition 2 and Postcondition 3.

TABLE III. BEHAVIOUR INTERFACES FOR ACTIVITY 1 AND ACTIVITY 2

Activity	Trigger	Preconditions	Postconditions
Activity 1	Trigger 1	Precondition 1 Precondition 2	Postcondition 1
Activity 2	Trigger 2		Postcondition 2 Postcondition 3

Figure 2 presents a CPN model of interconnected nodes representing triggers, preconditions and postconditions of Activity 1 and Activity 2 mapped from the behaviour interface model given in Table III. Data-flows are not captured in the sample CPN model of Figure 2. Heuristics for modelling data-flows are out of scope for this paper and left as future work.

The following section gives a full implementation of the running case's CPN model for studying the behaviour of socio-technical systems and evaluating soundness.

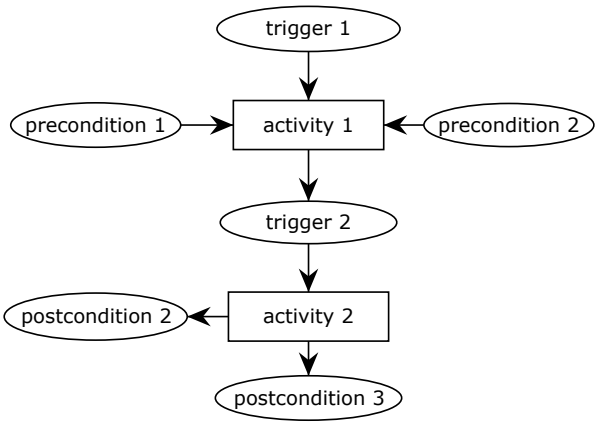


Fig. 2. A CPN model for sample behaviour interface model.

V. Formalized CPN Model and Evaluation

Following the procedure for mapping agent-oriented behaviour models to CPN, Section V-A presents a formalised CPN model of the running case. In Section V-B,we simulate the CPN model for studying socio-technical behaviours by altering some factors such as availability of the doctors and the average time it takes for attending patients with CLRs.

A. CPN Model for the Running Case

A CPN model in Figure 4 is equivalent to the earlier presented agent-oriented goal model of the running case. The atomic activities from the goal model we map to the behaviour interface models. Due to page limitations, it is not possible to show all models in this paper. Instead, we refer the reader to the full version[2] in the footnote for the complete CPN model of the running case. Behaviour interface models consist of triggers, preconditions and postconditions for each activity depicted by the lowest level sub-goals in the goal model.

For representing the goal *Admit patient*, Figure 3 shows the equivalent refinement as a CPN module. The refinement consists of four transitions mapped from the activities in the behaviour interface model given in Table I. Each transition is connected to places by at least one incoming arc and one outgoing arc. The former describes a trigger, or precondition while the latter describes a postcondition identified by the help of AOM models in Section III. The CPN modules comply to the guidelines given in the previous section that each activity must have a trigger and at least one postcondition.

For the running case, the module simulation triggers when a new patient arrives at the hospital. The transition *request patient ID* fires and results in a new place *received ID*. This new place acts as a trigger to two possible transitions, namely, *check existence* and *register patient*. In addition, the execution of the former requires existence of a suitable token in *patient DB* place as a precondition. The place *patient DB* also serves as the postcondition together with the place *registered patient* for the mentioned two transitions.

[2]https://www.dropbox.com/s/9efc9t9zn2oqttn/aom_cpn_model.cpn

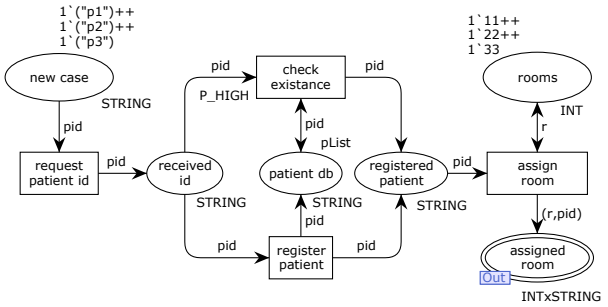


Fig. 3. A formalised CPN model for the "Admit patient" sub-goal.

We introduce the label `P_HIGH` to the transition *check existence* indicating firing priority. If *patient ID* is not found in the *patient DB*, the transition *check existence* never fires. Therefore, the alternative transition *register patient* fires by registering the patient into *patient DB* and having the output place *registered patient*. With the existence of available rooms, the *assign room* transition is triggered by the place *registered patient*. The execution of this module ends with the place *assigned room* that connects to remaining activities of the healthcare system represented in CPN. These activities are refinements of different modules corresponding to their parent goals such as *Collect Results*, *Identify Doctor* and so on. Figure 4 represents a higher-level CPN model of the running case with all the modules and relationships among them.

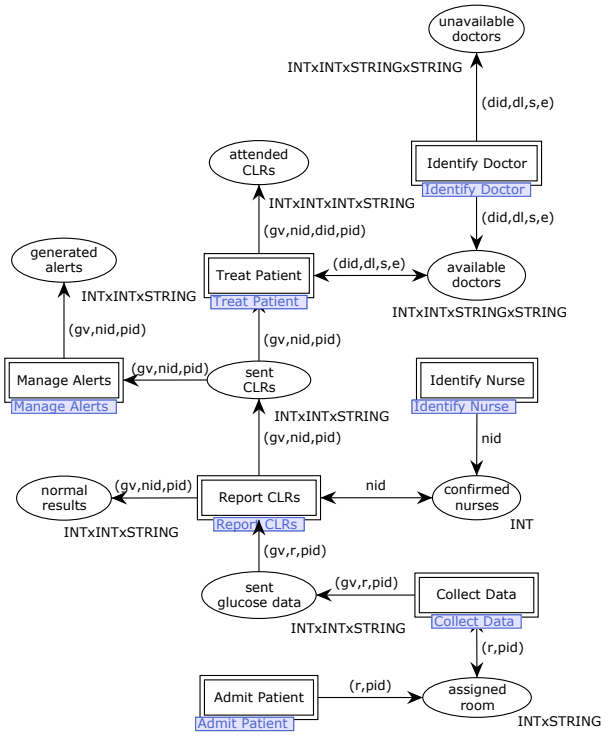


Fig. 4. A formalised CPN Model from AOM.

B. Simulation and Results

The simulation of the complete CPN Model for the running case aims at identifying the optimal number of available doctors that attend CLRs. The simulation also assures a minimal number of generated alerts. According to the depiction of CPN sub-model Figure 5, the system generates an alert when there is no available doctor to attend CLRs. A firing of transition *assign doctor* requires a fulfilled precondition *available doctor*. Otherwise, the transition *check delays* fires, followed by transition *generate alert* that Figure 5 does not capture.

For one CPN simulation, three different patients generate a total of 100 CLRs in an interval of 10 time units. We record the number of available doctors, generated alerts, attended CLRs, and average time taken by doctors when attending CLRs. The amount of activities carried out determines the availability of doctors, e.g., for attending patients. Table IV summarises the results of the CPN simulation and assumes attending CLRs consumes between 0 and 10 time units. The results in Table V assume the time taken for attending CLRs consumes between 10 and 20 time units.

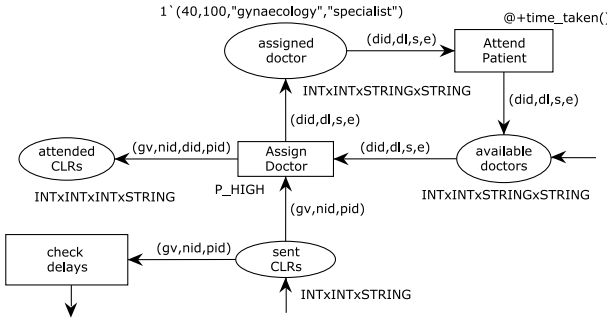


Fig. 5. A formalised CPN sub-model for generating alerts.

TABLE IV. RESULTS TABLE 1

Available Doctors	Attended CLRs	Generated Alerts
1	32	64
2	72	28
3	100	0

Table IV and Table V deduce approximate numbers for respective doctors who attend CLRs and the number of generated alerts. In both tables, the trends of the results comply with real-life observations [9] where the number of generated alerts decreases with an increase in the number available doctors. With a fixed time interval for generating CLRs, the results in Table IV suggest a need for 3 available doctors to minimize the number of generated alerts, while the results in Table V show 6 available doctors to achieve the same results. The doubled number of available doctors in Table V complies with the doubled range in the time taken for attending CLRs. The summarised results in these two tables not only show the real-life coherent behaviour of the designed socio-technical system but also the correctness and soundness of the designed CPN model.

In the next section, we discuss related work for formalising and evaluating socio-technical agent-oriented behaviour models.

TABLE V. RESULTS TABLE 2

Available Doctors	Attended CLRs	Generated Alerts
1	16	84
2	34	66
3	53	47
4	70	30
5	85	15
6	100	0

VI. RELATED WORK

The trend towards using Petri Nets for modelling and analysing is gaining prominence. For example, the conceptual framework AgOS [18] allows for a high level representation of a multi-agent system environment using classical Petri Nets. The disadvantage of using classical Petri nets in AgOS is a decrease of expressiveness for large systems. As CPN allow for modelling hierarchies, the approach in our paper is more scalable. In [19], agents for the management of computing resources in clouds the authors formalise using Petri Nets. These examples have a technical focus for using Petri Nets in the design of multi-agent systems. Instead, the relatively new AOM focus is socio-technical in nature and thus, recognizes the interaction between people and technology in workplaces that other research work does not consider.

Automating a technical realisation of multi-agent system research in [20] presents. The authors show a domain engineering process for developing multi-agent system product lines including supporting agent variability and providing agent feature traceability resulting in reduced time-to-market and lower development costs. CPN Tools also offers an automatic translation to Java code that a programmer can implement to full completion.

For formalising agent models, other options exist too. A tool called Rodin [21] supports system formalization with Event-B that uses set theory and refinement through theorem proving to represent systems at different abstraction levels. The generated mathematical proof verifies the refinements. The Rodin tool integrates system modelling and proving of formalised systems.

The so-called PiVizTool [22] supports system design with π-calculus. The original purpose is to model and analyse Web-service choreographies and it is also a candidate for formalising aspects of AOM. However, as [23] discusses, π-calculus is differently to Petri Nets not a graphical notation that system modelling and analysis more challenging for laymen. Using mature CPN Tools is easier to accomplish and it suffices to understand the use of the integrated tools for simulation, performance testing and verification to generate quickly soundness checks for AOM.

VII. CONCLUSION

In this paper, we define heuristics for formalising and evaluating agent-oriented behaviour models of socio-technical systems. The aim is to ensure before an actual implementation that the models are sound and coherent. A running case from

the healthcare domain demonstrates the mapping from AOM to CPN. The running case focuses on the management of Critical Laboratory Results by utilising a minimum number of resources, including humans such as doctors and nurses.

For capturing socio-technical requirements of the running case, the AOM approach is suitable for engaging technical and non-technical stakeholders from the healthcare domain. A goal model and behaviour interface model summarise the results of the AOM-based design. The former specifies in a hierarchically refining way, the objectives of a socio-technical system while the latter defines the triggers, preconditions and postconditions for each identified activity.

We give a set of mapping heuristics from AOM to CPN with the latter having the advantage of providing visual elements of places, transitions, modules, arcs that may carry condition statements with the required expressiveness to capture the properties of the equivalent AOM model. The advantage of this mapping is that the CPN model allows for tool supported simulation, performance testing and model-checking based verification that is currently not possible in AOM. Consequently, such a CPN-based evaluation gives indications about the soundness of the AOM models and coherent behaviour of the designed socio-technical system. The running case of the paper shows that the results of the CPN-model simulations yields results corresponding to empirical data collected from the healthcare domain.

As future work, we plan to develop tool-support for mapping from AOM to CPN that requires a detailed definition of the mapping rules beyond the heuristics given in this paper. Furthermore, this tool must also comprise mechanisms for a rapid system implementation, for example, by mapping automatically to programming code that reduces full development time to a minimum.

ACKNOWLEDGEMENT

This work is partially supported by the project SF0140013s10 "Model-based Creation and Management of Evolutionary Information Systems" (2010-2014) by the Estonian Ministry of Education and Research; The IT Academy Programme for Information and Communication Technology Research of Tallinn University of Technology (13-09-00-1); The Dawn Program of Shanghai Education Commission (11SG44) and The Research Fund for the Doctoral Program of Higher Education of China 20123120130001).

REFERENCES

[1] I. Sommerville, D. Cliff, R. Calinescu, J. Keen, T. Kelly, M. Kwiatkowska, J. Mcdermid, and R. Paige, "Large-scale complex it systems," *Communications of the ACM*, vol. 55, no. 7, pp. 71–77, 2012.

[2] P. Carayon, "Human factors of complex sociotechnical systems," *Applied ergonomics*, vol. 37, no. 4, pp. 525–535, 2006.

[3] E. Trist, "Some social and psycho," *Human relations*, vol. 4, p. 3, 1951.

[4] M. Mahunnah and K. Taveter, "A scalable multi-agent architecture in environments with limited connectivity: Case study on individualised care for healthy pregnancy," in *7th IEEE International Conference on Digital Ecosystems and Technologies (DEST)*. IEEE, 2013, pp. 84–89.

[5] I. Shvartsman and K. Taveter, "From agent-oriented models to profile driven military training scenarios," in *Intelligent Distributed Computing VII*. Springer, 2014, pp. 317–322.

[6] S. Pedell and L. Sterling, "Agent-based modelling for understanding sustainability," in *Agents in Principle, Agents in Practice*. Springer, 2011, pp. 398–409.

[7] S. Pedell, T. Miller, L. Sterling, F. Vetere, and S. Howard, "Substantiating agent-based quality goals for understanding socio-technical systems," in *Advanced Agent Technology*. Springer, 2012, pp. 80–95.

[8] L. Sterling and K. Taveter, *The art of agent-oriented modeling*. MIT Press, 2009.

[9] M. Mahunnah, A. Koorts, and K. Taveter, "Towards distributed sociotechnical system for reporting critical laboratory results." in *The 6th International Conference on Health Informatics (HEALTHINF), Barcelona, Spain*. SciTePress-Science and Technology Publications, 2013.

[10] E. Trist, "The evolution of socio-technical systems," *Occasional paper*, vol. 2, p. 1981, 1981.

[11] S. A. DeLoach, "Modeling organizational rules in the multi-agent systems engineering methodology," in *Advances in Artificial Intelligence*. Springer, 2002, pp. 1–15.

[12] P. Giorgini, J. Mylopoulos, and R. Sebastiani, "Goal-oriented requirements analysis and reasoning in the tropos methodology," *Engineering Applications of Artificial Intelligence*, vol. 18, no. 2, pp. 159–171, 2005.

[13] S. A. DeLoach, "Analysis and design using mase and agenttool," DTIC Document, Tech. Rep., 2001.

[14] L. Padgham and M. Winikoff, "Prometheus: A methodology for developing intelligent agents," in *Agent-oriented software engineering III*. Springer, 2003, pp. 174–185.

[15] J. Marshall, "Agent-based modelling of emotional goals in digital media design projects," *International Journal of People-Oriented Programming*, 2014.

[16] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr, "Basic concepts and taxonomy of dependable and secure computing," *Dependable and Secure Computing, IEEE Transactions on*, vol. 1, no. 1, pp. 11–33, 2004.

[17] K. Jensen, L. Michael, K. L. Wells, K. Jensen, and L. M. Kristensen, "Coloured petri nets and cpn tools for modelling and validation of concurrent systems," in *International Journal on Software Tools for Technology Transfer*, 2007, p. 2007.

[18] R. K. Chatterjee, A. Sarkar, and S. Bhattacharya, "Modeling and analysis of agent oriented system: Petri net based approach," in *11 th International Conference on Software Engineering Research and Practice (SERP 11, WORLDCOMP 2011)*, vol. 1, 2011, pp. 17–23.

[19] W. Iqbal and S. Yousaf, "Formal modeling of agent based cloud computing services using petri nets," *VFAST Transactions on Software Engineering*, vol. 1, no. 2, pp. 1–6, 2013.

[20] I. Nunes, C. J. D. Lucena, D. Cowan, U. Kulesza, P. Alencar, and C. Nunes, "Developing multi-agent system product lines: from requirements to code," *International Journal of Agent-Oriented Software Engineering*, vol. 4, no. 4, pp. 353–389, 2011.

[21] J.-R. Abrial, M. Butler, S. Hallerstede, T. S. Hoang, F. Mehta, and L. Voisin, "Rodin: an open toolset for modelling and reasoning in event-b," *International journal on software tools for technology transfer*, vol. 12, no. 6, pp. 447–466, 2010.

[22] P. Papapanagiotou and J. D. Fleuriot, "A theorem proving framework for the formal verification of web services composition," *arXiv preprint arXiv:1108.2348*, 2011.

[23] W. van der Aalst, "Pi calculus versus petri nets: Let us eat "humble pie" rather than further inflate the "pi hype"," 2003.