Association for Information Systems
AIS Electronic Library (AISeL)

AMCIS 2005 Proceedings

Americas Conference on Information Systems (AMCIS)

2005

Designing Multi-Agent Systems - The NDA Approach Applied in Health Care

Andreas Schweiger
Technische Universitat Munchen, schweiga@in.tum.de

Helmut Krcmar
Technische Universitat Munchen, krcmar@in.tum.de

Designing Multi-Agent Systems – The NDA-Approach Applied in Health Care

Andreas Schweiger
Technische Universitaet Muenchen
schweiga@in.tum.de

Helmut Krcmar
Technische Universitaet Muenchen
krcmar@in.tum.de

ABSTRACT

In this paper we introduce inherent problems of information logistics in health care. Promising research results on agent-based systems have allowed us to conclude that this approach is especially suitable to coping with these problems. In order to adequately capture the requirements of a complex setting, we present an approach for the design of agent-based systems. The basis forms the ethnography-based requirements analysis approach Needs Driven Approach (NDA). The NDA supports the participating observation of work processes and guides the construction of domain models. As a result of a field study, a meta-model is constructed which reflects the interrelationships of its elements. According to basic ideas of the Model Driven Architecture (MDA), the elements of the meta-model are mapped to constructs of software engineering.

Keywords

Requirements elicitation, domain analysis, Needs Driven Approach, transformation rules.

INTRODUCTION

Although measures to reduce costs have driven developments in health care IT (Kuhn and Giuse, 2001), information logistics in hospitals is still not adequately supported by information systems (Paulussen, Herrler, Hoffmann, Heine, Becker, Franck, Reinke and Strasser, 2003). Complex treatment processes, distribution of medical data, interdependencies between the actors of the health care team and unpredictable emergency situations still need to be tackled. The deployment of agent-based information systems seems to be an adequate solution for addressing these problems.

This paper is structured as follows: In section two we discuss specifics of the health care domain and describe in section three why these challenges can be tackled adequately by agent-based systems. Section three deals with requirements analysis. The introduction to requirements and domain analysis is followed by the presentation of the NDA. We describe the experiences of and refinements for the NDA as identified by the results of a field study. Furthermore, we present the integration of the elaborated domain models into active, dynamic activity areas as a means for integrating cross-institutional processes. The section is concluded by the comparison of the NDA to other approaches. Section four on system design begins by outlining the concepts of the MDA and shows how the domain models of the requirements phase can be mapped to constructs of software engineering. We conclude the paper by summarizing the research results and giving directions on further research.

SPECIFICS OF THE HEALTH CARE DOMAIN

Health care is characterized by special challenges regarding information logistics. For the further discussion, we concentrate on hospital information systems. Hospitals are usually divided into several functional areas. These are either autonomous or are allocated to a certain department. In the former case, hospitals make use of the centralization of services which are available to other departments. There are clear advantages of this organization as facilities can be shared among several entities and used efficiently. The central provision of services demands an all-embracing planning mechanism which takes into account competing requests from different departments. Up to now, these planning processes were usually performed manually, i.e. without IT-support. This leads to considerable organizational overhead.

Usually the treatment process is initially not determined in terms of treatment steps. This is due to the fact, that the diagnosis is originally unknown, but is elaborated during the hospital stay. Furthermore, undeterminable and emergency situations occur and disturbe the regular treatment processes.

The cooperation, communication and coordination of medical personnel are not generally supported through information systems. The interconnectedness of treatment processes and their necessary logistics cycles cause interdependencies between these processes. The result is the need to resolve conflicts that are due to tight resources or contradictory goals (Kirn, Heine, Petsch, Puppe, Kluegl and Herrler, 2000).

SUITABILITY OF MULTI-AGENT SYSTEMS

Currently available hospital information systems are not capable of providing adequate solutions to the described problems (Paulussen et al., 2003). Agent-based software systems, however, seem to provide valuable solutions.

Software agents are capable of handling automatic negotiations. Using this mechanism, the planning of resource assignments can be supported by mapping principles of market economy to requirements in health care. Agent-based systems are an adequate means for implementing complex, distributed systems (Jennings, 2001) and have the ability to react to changes in their environment due to their reactive behaviour (Wooldridge, 1997). Emergency situations can therefore be treated adequately and both resources and personnel rescheduled. Ensuing resource conflicts can be managed by taking into account different priorities for personnel and resources.

Software agents exhibit proactive behaviour and are therefore capable of properly representing real-world actors. The paradigm of agent systems allows for a natural and straightforward mapping of real-world entities and their behaviour to elements of software engineering. Furthermore, the semantically rich interaction relationships are also appropriately mapped to the agent paradigm since the interactions between real-world actors are modelled by complex interaction protocols. These protocols are not inherent to the agent object itself, but are rather encapsulated into the behaviour of the agent. Therefore, the agent paradigm allows for coping with complexity by separating interactions from the remaining implementation.

The diverse actors in the health care domain often take on several different roles simultaneously. This situation is also adequately modelled through software agents as these are capable of adopting several roles at the same time and of changing their current roles at run-time. Taking different roles into account provides a means for further extensions: agents representing actors of different roles can be deployed to provide the appropriate set of medical data at the appropriate time and place.

First research results that demonstrate the suitability of the agent-based approach for dealing with challenges related to information logistics in health care (Heine, Herrler and Kirn, 2005; Heine and Kirn, 2004; Paulussen et al., 2003) are now becoming available.

REQUIREMENTS ANALYSIS

One of the challenges in developing software systems is building the right system (Fowler and Scott, 2002). To accomplish this, a lot of effort needs to be put into domain analysis. Developing software systems using object-oriented approaches is supported by established methodologies such as “The Unified Software Development Process” (Jacobson, Booch and Rumbaugh, 1999). These methodologies guide the development of software systems by decomposing the process into several activities that range from requirements elicitation, analysis, system design, object design to implementation (Bruegge and Dutoit, 2000). Although there are established methodologies for the object-oriented development of software systems, advances are still necessary as the development faces both demanding and increasingly more complex systems. This is especially true when developing multi-agent systems. These present a need for adequate methodologies that support the development and deployment of multi-agent systems (Luck, McBurney, Shehory and Willmott, 2004).

Domain Analysis

Domain analysis is defined as the process of identifying, capturing and organizing information used in developing software systems (Prieto-Díaz, 1990). Object-oriented modeling methodologies suggest modeling languages such as UML (Object Management Group, 2003; Rumbaugh, Jacobson and Booch, 1999) for the notation of use cases, activity diagrams, class diagrams etc. as a starting point for domain modeling. Use case diagrams describe the system from the point of view of the user. Once these diagrams are elaborated on, the first step for requirements elicitation is performed. But before the software engineer is capable of covering the communication between the user and the system in a use case diagram, an appropriate analysis approach must precede the phase of defining such diagrams. For this purpose, the ethnography-based NDA seems to be suitable.

Needs Driven Approach

Since users are accustomed to performing their work tasks, they often do not think explicitly about the sequence of their tasks (Bruegge and Dutoit, 2000). When asked for the specification of and the requirements for information systems, users often fail to describe their needs appropriately (Bruegge and Dutoit, 2000). Therefore, it is assumed that interviewing users about their work processes is inefficient (Bruegge and Dutoit, 2000) since not all relevant information for building the desired system is captured. As a result, we considered requirements engineering based on participating observation of work processes

to be appropriate. We introduced the ethnography-based NDA (Schwabe and Krcmar, 1996) to requirements analysis and extended it in order to provide a seamless integration into the early phases of the development process.

The NDA (Schwabe and Krcmar, 1996; Schwabe, Streitz and Unland, 2001) as an approach based on ethnography exhibits the prerequisite as an observation-based mechanism for capturing work processes and their related elements. The observation is guided by a set of focus areas which are derived as follows (Schwabe and Krcmar, 1996): Tasks, processes, structure of interaction, means of work, locations the work is performed in, adoption of technology and structure of informational memory.

Experiences of and Refinements for Deploying the NDA in the Health Care Domain

We deployed the NDA in a field study carried out in a hospital. During the application, a software engineer observed the everyday work-environment of potential information system users, i.e. hospital employees such as physicians and nurses. The participating observation took place in a ward in the department of internal medicine in a general hospital over a period of several weeks. The field study revealed that it is difficult to orientate oneself and to focus relevant aspects in an unfamiliar domain at the beginning of domain analysis. Therefore, the original NDA was refined by detailing guidelines for the observation of the domain to be analyzed. The guidelines support the analysis engineer in capturing the requirements step by step. As a result, the guidelines (Figure 1) were elaborated on.

The guidelines are separated into several parts and comprise the following five sections. Firstly, the work setting, rooms, organizational structure and involved actors, is described. This allows the observer to acquire an orientation within the domain which is needed for gathering further information. The results of these observations are captured in floor plans, organizational charts and a list of actors. In section two, the description of tasks and locations where actors accomplish those tasks follows. Users and their tasks are depicted in use case diagrams. By creating these diagrams for different locations, users are mapped to rooms and tasks performed in those rooms are identified. An observation of partial work flows and their integration are elaborated on in section three. The treatment process in health care is taken into account as it represents the central building block of this domain. Process models describe actions and their dependencies. Actors, necessary material and tools are associated with these actions. Parts of the interaction between actors are often already known at this point as they are implicitly provided by the description of the observed work processes. In section four, these interaction parts are completed, put into a new perspective by detaching them from the work processes and described in interaction diagrams. These diagrams constitute an integral part of the Needs Driven Analysis since interactions play an important role in health care: Many different health care professionals are involved in a treatment process and interactions and processes are closely linked. The association between interactions and processes is described in detail in section five. By separating the aspects processes and interactions, the complexity of diagrams is reduced.

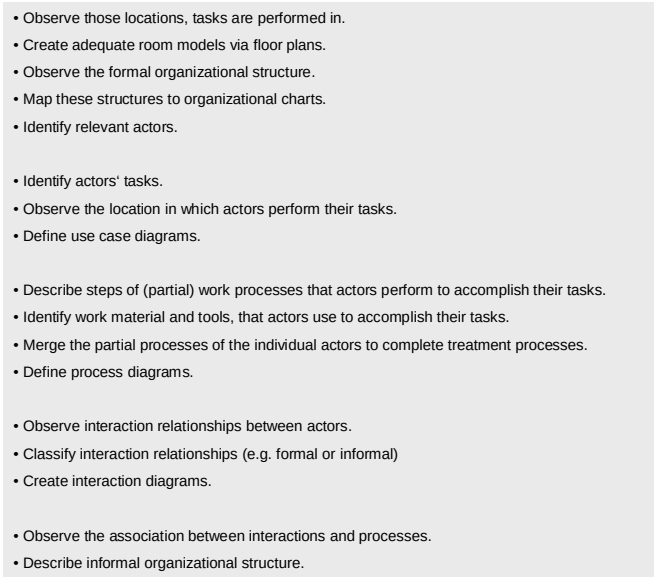

Figure 1. NDA-Guidelines for the Requirements Engineer

The described guidelines provide a general starting point of the requirements analysis and they allow the analyst to get quickly into the field of domain analysis. The guidelines are generally verbalized and are therefore capable of being applied to other domains. Thus, they provide an open framework for requirements analysis.

See Figure 2 and Figure 3 for sample diagrams, which represent exemplary results of sections three and four of the NDA-guidelines. Figure 2 shows a work process using an extended EPC notation. The notation is derived from the well-known Event Driven Process Chains (EPC; Keller, Nuettgens and Scheer, 1992). For an overview of references on the EPC notation see the publication list of the Special Interest Group on Process Modelling with EPCs at the German Informatics Society (http://www.epk-community.de/). To improve readability, our modified EPC notation has no tight bipartite alternation of functions and events. Functions are represented by rectangles with rounded corners and denote tasks. Events (left out in our modified notation) trigger functions and show their completion. Organisational units and actors are associated with functions and are represented by ovals. Although EPCs have their own drawback of not having well defined syntax nor semantics, this notation is considered to be useful in our approach. In order to strengthen the use of EPCs, the work on defining formal syntax and semantics (Kindler, 2004; Nuettgens and Rump, 2002) can be integrated into the further design process. This allows the automated transformation between different models as described in "Mapping of Domain Models to Software Engineering Constructs".

Figure 3 illustrates the notation of an NDA Interaction Diagram which depicts associated actors and actions within an interaction. Actions are represented by rectangles. The drawn through arrows explicate the direction of communication between participating parties, whereas dashed arrows denote the control flow. Organizational units and their association with functions as depicted in an EPC lack the provision of communication directions. Furthermore, EPCs focus the modelling of work processes, whereas interaction diagrams denote only interaction relationships. Therefore, we provided different views in order to reduce the complexity of more extensive models and introduced NDA Interaction Diagrams.

As a result of the application of the NDA, we developed an object-oriented meta-model denoted in UML (Figure 4), which defines the interrelations between the elements of the obtained domain models. Possible omissions in domain analysis may be detected and further observations are triggered by looking at this model. The meta-model identifies a process step as a collection of associated actors, location, material and tools. This meta-model also forms the starting point for defining transformation rules for the mapping between domain models and software engineering models (Reinke, 2003). Thus, the meta-model facilitates bridging semantic and syntactic gaps between the domain and the software engineering area.

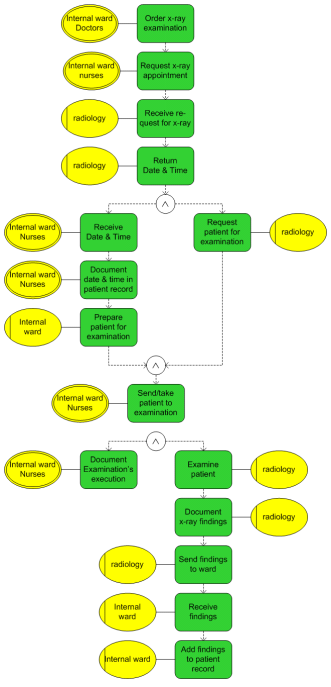

Figure 2. Sample EPC Process Diagram Describing the Work Flow for a Diagnostic Examination

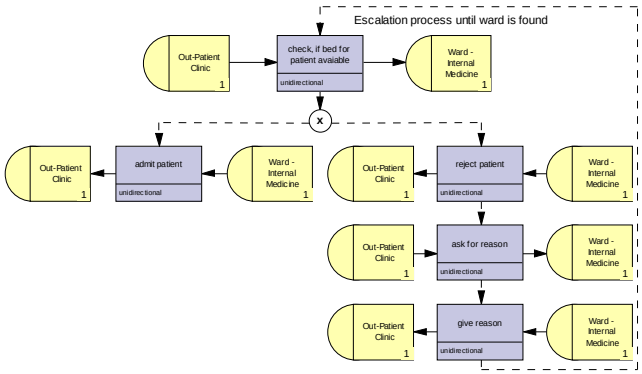

Figure 3. Sample NDA Interaction Diagram

Integration of Domain Models into Active, Dynamic Activity Areas

Active, dynamic activity areas (Schweiger and Krcmar, 2004) are an adequate means for interrelating parts of the process models and describing flexible work processes. Thereby, the integration of cross-institutional work flows is enabled. An activity area is comprised of actors, their tasks, and the necessary data and tools, e.g. medical devices. It contains explicit or implicit knowledge about the processing of tasks, their decomposition, their forwarding, and delegation. Activity areas are dynamic in terms of adding, removing, or changing the actors, i.e. their migration. Actions such as making-up and finishing interactions, changing the process knowledge, and the integration of new data sources or devices add further dynamic aspects. Additionally, activity areas are heterogeneous with respect to their properties, e.g. the used data and their storage.

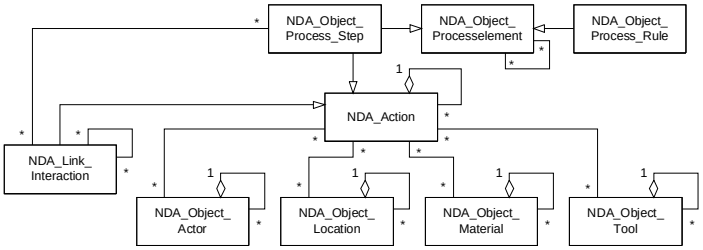

Figure 4. Object-oriented Domain Meta-model

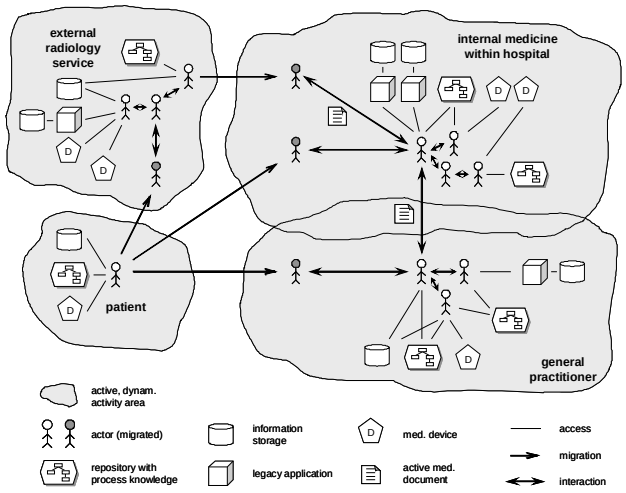

Figure 5. Active, Dynamic Activity Areas

Figure 5 exemplifies activity areas for a section of the health care domain. Consider a patient visiting an internal medicine ward within a hospital. The patient's and the ward's activity areas including their physicians, nurses, information systems and medical devices are merged. The information exchange between physicians regarding the medical information about the patient is facilitated by medical documents. Further information about the patient's health status is extracted from clinical information systems. Results of the physical examination, which are relevant for follow-up care, are shared with a general practitioner via medical documents.

The concept of activity areas allows for the explication of integration and cooperation between different institutions and thus adds further expressive power to the domain analysis. By means of this concept, separated work processes as identified up to now in the analysis phase are connected by merging activity areas of different institutions and therefore bridging the gap between the boundaries of organizations. This modelling technique reflects the observable trend from separated information systems to integrated health information systems (Kuhn and Guise, 2001; Waegemann, 1999).

Comparison to other Approaches

In comparison to the well established methodology Gaia (Wooldridge, Jennings and Kinny, 1999; 2000) our approach differs in the following ways. Gaia focuses on the construction of agent-based software systems that allow agents to collaborate in order to solve a common problem. Our approach also allows for modelling competing goals of work processes, as they appear e.g. when taking into account emergency scenarios in a hospital. A usual work process may be interrupted due to an emergency situation, which requires actors to change their tasks and roles due to changing priorities.

Gaia takes into account only roles and disregards modelling the overall structure of the organization of the system. In comparison, our approach also models the organizational structure by floor plans and organizational charts. Locations are also taken into account in the meta-model thus presenting the locations in which actors perform their tasks. This concept is especially useful for the exemplified health care domain, since the need for information varies depending on the room in which the actors currently find themselves.

Gaia makes no use of integrating different processes. Our approach integrates processes across organizational borders into common activity areas as explained in the section "Integration of Domain Models into Active, Dynamic Activity Areas" above.

As described in the section "System Design" below, the domain models can be mapped to directly codable software entities, whereas the design phase of Gaia has to be followed by usual object-oriented methodologies before implementation.

Another observation-based task analysis method is the Knowledge Analysis of Tasks (KAT; Johnson and Johnson, 1989, 1991; Johnson, 1992), which comprises the following steps:

- Identify objects and actions.
- Identify procedures.
- Identify goals and subgoals.
- Identify typicality and importance.
- Construct a model.

Table 1 summarizes criteria necessary for the adequate support of analysis and design of multi-agent systems in health care. The crucial differences between the compared methods are the goals of agents, models for the organization's structure, the integration of processes and implementation steps.

	NDA	**Gaia**	**KAT**
Modeled elements	Locations, actors, tasks, partial work processes, work material, tools, integrated processes, interactions	Roles (permissions, responsibilities), interactions, agents, services, acquaintance	Objects, actions, procedures, goals, typicality, importance
Goals of agent interaction	Competing	Cooperating	No explicit distinction between competing and cooperating
Structure of organization	Floor plans, organizational charts	Not supported	Not supported
Integration of processes	Active, dynamic activity areas	Not supported	Not supported
Implementation	Transformed elements can be coded directly	Design phase has to be followed by further methodologies	No focus on agents, models cannot be coded directly

Table 1. Criteria for Comparing Different Approaches

SYSTEM DESIGN

Model-Driven Architecture

The Object Management Group (OMG) proposed the concept of model driven-software development. Its goal is "to support interoperability with specifications that address integration through the entire systems life cycle" (Object Management Group, 2001). The term Model Driven Architecture (MDA; Object Management Group, 2000) defines several abstraction layers. The goal of this approach is to allow the software developer to focus on modeling business concepts. Through defining transformation rules, the mapping from business models to programming code can be automated. The MDA distinguishes between the terms PIM (Platform Independent Model), PSM (Platform Specific Model) and code. There need not necessarily be only these three models. There is, for example, an unbounded amount of different PSM, which refine models step by step. Business models are depicted in PIM. These models are mapped to one or several PSM by user-defined transformation rules. Depending on the point of view, a PSM can also be identified as a PIM. The mappings between models are refined until the actual implementation is achieved. The benefit of this approach is the introduction of an additional abstraction layer. Once the transformation rules are provided, changing business requirements can be automatically mapped to executable code.

Mapping of Domain Models to Software Engineering Constructs

As a result of the application of the NDA, the central meta-model (Figure 4) was created. This meta-model defines the interrelationships between its elements. During the analysis phase, we identified locations, actors, tasks, partial work

processes, work materials and tools, integrated processes and interactions. As depicted in Figure 6, these elements are mapped to constructs of software engineering. According to this mapping, an actor of the domain is represented by the authority on behalf of which an agent acts. Locations are mapped to agent platforms, which can reside on locally distributed nodes. Identified process steps are carried out through the agents' tasks. Documents are represented by resources utilized by the agents when performing tasks.

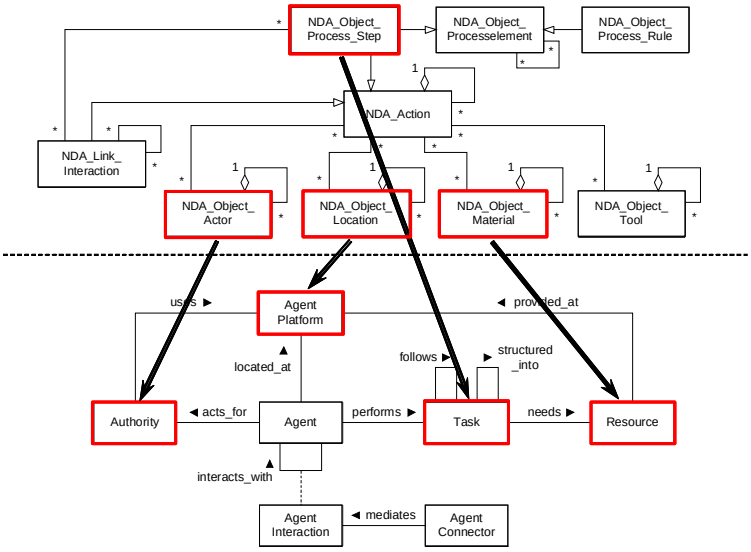

Figure 6. Transformation Rules between Domain Models and Elements of Agent-based Systems

Major parts of the domain models are mapped directly to elements of agent-based systems. Therefore, the applied analysis approach NDA is especially useful when deployed in the context of agent-based software construction. Once the design phase, i.e. the construction of agent-based system models, is completed, refinement and implementation can be addressed. These phases can be supported through an architecture-based methodology (Reinke, 2003) which provides an approach for the stepwise and guided construction of an agent-based system covering domain modeling, specializing of a general architecture, construction of subsystems and integration and allocation phases.

CONCLUSION

In this paper we introduced problems associated with information logistics in health care and showed that agent-based information systems are capable of solving these problems. To better represent the requirements for such systems, we presented an approach for the support of early phases of a software development process for agent-based systems. The proposed NDA is based on ethnography, focuses on the participative observation of work settings and is designed for requirements analysis. For the purpose of collecting information about the system to be implemented, typical work flows and processes were observed in a field study. Detailed and enhanced NDA guidelines allow the analysis engineer to focus on relevant aspects. In a step by step fashion, element locations, actors, tasks, partial processes, tools, materials, integrated processes and interactions are identified. Several domain models were developed which reflect the study findings. The result of the applied NDA in the health care domain was the construction of a meta-model describing the interrelationships between the mentioned domain model elements. Finally, the newly developed models were integrated using the concept of active, dynamic activity areas which are especially suited to map cross-organizational cooperation into an appropriate model. We presented basic transformation rules for the mapping of the meta-model's elements to agent-based concepts. This approach makes use of MDA which assumes that the software developer focuses on modeling business models.

The presented approach seems to be suitable for building agent-based systems as identified actors and their performed process steps can be directly mapped to elements of agent-based systems. Existing semantically rich interaction relationships, captured in NDA interaction diagrams, can be addressed in a straightforward manner by deploying the communication mechanisms of agent-based systems.

The NDA and its extended guidelines should be further evaluated to demonstrate their applicability and usefulness. An agent-based prototype is currently under development. Its purpose will be to prove the concept of the outlined approach and provide an explication of a solution for previously identified problems in information logistics in health care. Further research is to be conducted for mapping active, dynamic activity areas to constructs of software engineering. A means for expressing dynamically growing and resolving activity areas needs to be developed. Up to now, only a partial mapping from the domain meta-model to the agent-based approach has been elaborated on. Therefore, the transformation of missing parts of the meta-model to models of software engineering is subject to ongoing research. As a result, we aim for the seamless mapping of domain models to constructs of agent-based software engineering. The integration of formal descriptions of EPCs facilitates automatic mappings between models.

ACKNOWLEDGMENTS

Thomas Reinke and Astrid Hoffmann collaborated on parts of the research. The authors thank them for their ideas and insightful discussions.

REFERENCES

1. Bruegge, B. and Dutoit, A. H. (2000) Object-Oriented Software-Engineering: Conquering Complex and Changing Systems, Prentice Hall, Upper Saddle River.
2. Fowler, M. and Scott, K. (2000) UML Distilled: A Brief Guide to the Standard Object Modeling Language. 2nd Edition. ADDISON-WESLEY, Boston.
3. Heine, C., Herrler, R. and Kirn, S. (2005) ADAPT@AGENT.HOSPITAL: Agent-based Optimization and Management of Clinical Processes, *International Journal of Intelligent Information Technologies (IJIIT)*, 1, 1, 30-48.
4. Heine, C. and Kirn, S. (2004) ADAPT@Agent.Hospital – Agent Based Support of Clinical Processes. *Proceedings of the European Conference on Information Systems (ECIS 2004)*, Turku, Finland.
5. Jacobson, I., Booch, G. and Rumbaugh, J. (1999) The Unified Software Development Process, Addison-Wesley, Reading, MA et al.
6. Jennings, N.R. (2001) An Agent-Based Approach for Building Complex Software Systems. *Communications of the ACM*, 44, 4, 35-41.
7. Johnson, H. and Johnson, P. (1989) Integrating Task Analysis into System Design: Surveying Designers' Needs. *Ergonomics*, 32, 11, 1451-1467.
8. Johnson, H. and Johnson, P. (1991) Task knowledge structures: Psychological basis and integration into system design. *Acta Psychologica*, 78, 1-3, 3-26.
9. Johnson, P. (1992) Human Computer Interaction: Psychology, Task Analysis and Software Engineering, McGraw-Hill, London et al.
10. Keller, G. Nuettgens, M. and Scheer, A.-W. (1992) Semantische Prozessmodellierung auf der Grundlage "Ereignisgesteuerter Prozessketten (EPK)". Saarbruecken: Universitaet des Saarlandes.
11. Kindler, E. (2004) On the Semantics of EPCs: A Framework for Resolving the Vicious Circle. *Proceedings of the Business Process Management: Second International Conference, BPM 2004*, Potsdam, Germany, 82-97.
12. Kirn, S., Heine, C., Petsch, M., Puppe, F., Kluegl, F. and Herrler, R. (2000) Agentenorientierte Modellierung vernetzter Logistikkreislaeufe als Ausgangspunkt agentenbasierter Simulation – Beispiel Woechnerinnenstation. *Proceedings of the 2nd Kolloquium des DFG-Schwerpunktprogramms "Intelligente Softwareagenten und betriebswirtschaftliche Anwendungsszenarien"*, Ilmenau, Germany.
13. Kuhn, K. A., Guise, D. A. (2001): From Hospital Information Systems to Health Information Systems: Problems, Challenges, Perspectives. *Methods of Information in Medicine*, 40, 4, 275-287.
14. Luck, M., McBurney, P., Shehory, O. and Willmott, S. (2004) Agent Technology Roadmap: Overview and Consultation Report. http://www.agentlink.org/roadmap/, Last Access 04/07/2005.
15. Nuettgens, M., Rump, F. J. (2002) Syntax und Semantik Ereignisgesteuerter Prozessketten (EPK). *Proceedings of the Promise 2002 – Prozessorientierte Methoden und Werkzeuge fuer die Entwicklung von Informationssystemen*, Potsdam, Germany, 64-77.
16. Object Management Group (2000) Model Driven Architecture, White Paper, Draft 3.2, November 27, 2000. In: ftp://ftp.omg.org/pub/docs/omg/00-11-05.pdf, Last Access 04/02/2005.

17. Object Management Group (2001) Model Driven Architecture (MDA), Document number ormsc/2001-07-01. In: http://www.omg.org/cgi-bin/doc?ormsc/2001-07-01, Last Access 02/07/2005.
18. Object Management Group (2003) Unified Modeling Language, Version 1.5. OMG Document formal/2003-03-01.
19. Paulussen, T. O., Herrler, R., Hoffmann, A., Heine, C., Becker, M., Franck, M., Reinke, T. and Strasser, M. (2003) Intelligente Softwareagenten und betriebswirtschaftliche Anwendungsszenarien im Gesundheitswesen. *Proceedings of the Informatik 2003: Innovative Informatikanwendungen*, Frankfurt, Germany, 64-82.
20. Prieto-Díaz, R. (1990) Domain Analysis: An Introduction. *Software Engineering Notes*, 15, 2, 47-54.
21. Reinke, T. (2003) Architekturbasierte Konstruktion von Multiagentensystemen. Ph.D. Thesis, Universitaet Potsdam, Germany.
22. Rumbaugh, J., Jacobson, I. and Booch, G. (1999) The Unified Modeling Language Reference Manual. 1st Edition. ADDISON-WESLEY, Boston.
23. Schwabe, G. and Krcmar, H. (1996) Der Needs Driven Approach. In Krcmar, H., Lewe, H. and Schwabe, G. (Eds.) Herausforderung Telekooperation. *Proceedings of the DCSCW 1996*. Springer, Heidelberg et al., 69-88.
24. Schwabe, G., Streitz, N. and Unland, R. (Eds.) (2001) CSCW-Kompendium. 1st Edition, Springer, Heidelberg.
25. Schweiger, A. and Krcmar, H. (2004) Multi-Agent Systems for Active, Dynamic Activity Areas. *Proceedings of the Americas Conference on Information Systems (AMCIS)*, New York City, USA.
26. Waegemann, C. P. (1999) Current Status of EPR Development in the US, *Proceedings of the Toward An Electronic Health Record Europe*, London, 116-118.
27. Wooldridge, M., Jennings, N. R. and Kinny, D. (1999) A Methodology for Agent-Oriented Analysis and Design. *Proceedings of the Third International Conference on Autonomous Agents (Agents 99)*, Seattle, WA, 69-75.
28. 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.
29. Wooldridge, M. (1997) Agent-Based Software Engineering. *Software Engineering, IEE Proceedings*, 144, 1, 26-37.