Agent oriented specification for patient-scheduling systems in hospitals

A. Bartelt, W. Lamersdorf
Department of Distributed Systems
University of Hamburg
Vogt-Kölln-Straße 30
Hamburg, Germany
bartelt@informatik.uni-hamburg.de

T. O. Paulussen, A. Heinzl
Department of Information Systems
University of Bayreuth
Universitaetsstrasse 30
Bayreuth, Germany
paulussen@uni-bayreuth.de

1 Introduction

Patient-scheduling in hospitals is a complex task which requires new computational methods, e.g. market mechanisms and enhanced support by software agents. These demands are addressed by the MedPAge-Project (Medical Path Agents) which covers the development of a multi-agent-system for which an agent oriented specification will be presented.

Firstly, based on field studies in five German hospitals, the hospital domain is analysed (c.f. [9]). In this domain analysis, a generic hospital structure is derived and the relevant co-ordination objects for patient-scheduling are identified. Secondly, hospital specific scheduling problems are discussed.

On the foundation of this domain analysis, the architecture of the MedPAge multi-agent-system is developed, taking actual agent-oriented methodologies into account. The agents, consisting of an individual schedule and utility function, are modeled and the co-ordination mechanism, determining the agent interactions, is described. Finally several implementation issues are discussed.

2 Domain analysis

Hospitals are service providers. Their primary aim is to improve the health state of their patients. Therefore, the treatment process is the central value adding process [5].

According to their illness, patients have to undergo several examinations and treatments during their stay in hospital. Because it is in the nature of diagnostics to gain additional information about the patients' diseases, the necessary medical treatments are often not completely determined at the beginning of the treatment process.

Further, the sequence of execution of those actions does partially not follow a given order. But due to medical reasons, there can be minimum and/or maximum delay requirements between two (or more) tasks. Because these constraints can depend on the sequence of execution, those relations are modelled as bothway time-windows instead of strict, binary order constraints. A task *i* might has to start

- *immediately* after,
- *sometime* after,
- *minimum time* after,
- *maximum time* after,
- *between a minimum and maximum time* after, or
- *never* after

a task *i*.

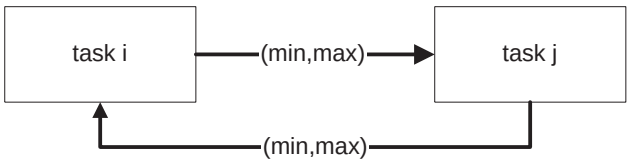

Figure 1. Bothway time-window between task i and task j

For the majority of the tasks the patients have to attend physically. As the patient is an exclusive resource, those tasks can only be performed sequentially. Examples for those patient-requiring tasks are taking x-rays and drawing blood. However, some tasks do not require the patient to be present, like the evaluation of x-rays or laboratory blood-tests. These tasks can be performed simultaneously (c.f. [4][11]).

Due to the individuality of the patients, the duration of the medical actions are stochastic.

The treatments and examinations assigned to the patients and the relations between them, build the pathway for these patients through the hospital.

Additional problems for the patient-scheduling in hospitals arise from complications and emergencies. The immediate need of treatment for emergency patients causes disturbances in the schedule. Complications, which may occur during a treatment, result in waiting times and changed pathways for other patients.

To perform those examinations and treatments, hospital resources, i.e. personnel, rooms and machines, are needed. As hospitals are highly decentralised organisations, those resources are combined in separate functional service units [4]. However, it is not necessary that all resources belong to the same hospital.

The units of a hospital can be aggregated, where the highest aggregated unit represents the hospital itself, being one part of the healthcare chain next to general practitioners, outdoor specialists, pharmacies and rehabilitation facilities.

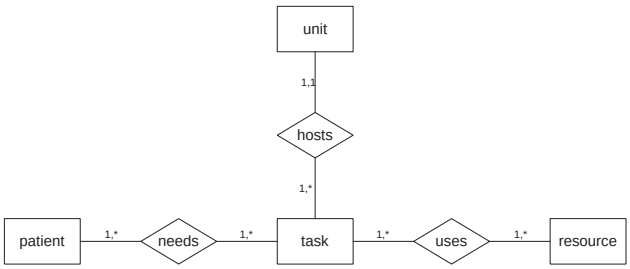

Figure 2. Generic hospital model

Figure 2 shows a generic hospital model, in which the patients need one or more treatments/examinations. Each of those medical actions is assigned to a specific unit and requires at least one resource for execution.

3 Architecture

The architecture of the MedPAge multiagent system is based on the identified agent roles and the dynamic interactions. Therefore, an adequate agent oriented software engineering methodology will be identified and described, which will be used to model the concrete MAS later in this section.

3.1. Agent oriented software engineering methodologies

The research on complex agent oriented systems spans a wide range of application domains (cf. [2]). Design of these systems requires adequate methodologies to cope with agent specific characteristics, which result from agent oriented domain analysis.

In the past there have been various approaches to modeling multiagent systems and since the field of agent oriented software engineering is currently under permanent development. The suggested methodologies can be divided into high level approaches, which usually focus on system architectures, and more detailed design level approaches, e.g. for interactions.

A specific agent oriented software engineering methodology will be identified and used to model high level abstractions which represent the internal conceptual architecture of MedPAge. As interactions are an integral part of the MedPAge project there adequate modelling methods are also required.

3.1.1 Overview

A few major high level methodologies include *Gaia* [16] by Wooldridge et al., *MaSE* [15] by Wood et al., and *MESSAGE/UML* [3] by Caire et al. *Gaia* is a high level approach where models are mostly textually based. One key point is detecting all roles of the system and defining the responsibilities and abilities of each role. Then, for each role the system designer should create a textual scheme - saying what a role can or should do. This leads to the analysis of interactions that are necessary to perform tasks. *MaSE*, also a high level approach, suggests a rather strict phase model for agent oriented modeling. Starting from system requirements (via use cases) and system goals, roles are defined. This leads to concrete agents and to communication between those agents. *MESSAGE/UML* is a graphical modeling approach based on UML and incorporates some concepts from other methodologies. It defines extensions to the UML language and suggests different views and diagrams that can be used for modeling the multiagent system. Further, an analysis process proposed to subsequently refine the models.

Each of the proposed methodologies has its advantages and emphases as well as drawbacks. As currently none of the methodologies is widely adapted or especially focussed on the healthcare domain, MESSAGE/UML appears to be a well suited candidate for the MedPAge project as it provides a competitive set of graphical modeling capabilities and is integrated with the UML metamodel which is broadly known and accepted.

3.1.2 MESSAGE/UML

This section gives a brief overview of the key approaches and benefits of MESSAGE/UML. As the name suggests MESSAGE/UML is an extension of the Unified Modeling Language (UML). MESSAGE/UML uses standard UML diagrams such as class diagrams or activity diagrams and extends them in an agent specific way. There are several reasons which make UML a good base for an agent oriented modeling language (cf. [3]): (a) Being the de facto standard for object oriented modeling UML gains a wide acceptance and a high level of usage. (b) The object- and

agent-oriented paradigms are highly compatible, so agent-oriented concepts can readily be defined in terms of object-oriented ones. (c) UML has a meta model that makes it extensible.

MESSAGE defines agent oriented entities as well as different views and corresponding diagrams on these entities for agent oriented system analysis. There are three groups of new concepts in MESSAGE: *Concrete Entities, Activities* and *Mental State Entities*. Concrete entities are subjects of the agent world such as agents, roles or organizations. Activities are tasks or interactions performed by one agent or a group of agents. Mental state entities are elements of knowledge that one agent has, or that is transported from one agent to another.

Concrete Entities are *Agents, Roles, Organizations* and *Resources*. The graphical representations are shown in figure 3.

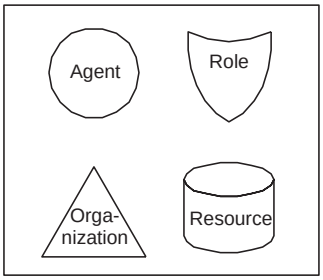

Figure 3. Concrete Entities

An *Agent* is an autonomic unit performing some action. A *Role* stands to an agent, like an interface to a class in the standard objectoriented meaning. A Role defines the outer behavior of an agent in an specific context. An agent can play several roles, and one role can be played by several agents. An *Organization* groups some agents and resources to one unit performing the same means. Note, that an organization is only an entity of analysis, and (in general) will have no direct corresponding program element. *Resources* are databases or external programs used by an agent.

Activities are *Tasks* and *Interactions* (see figure 4). A Task is a kind of elementary operation an agent performs. An interaction describes the exchange of messages (having some information) between two or more agents.

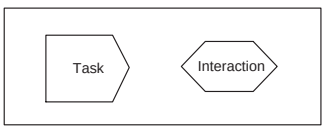

Figure 4. Activities

Mental State Entities are objects that an agent holds in ”mind”. The graphical representation of the two types *Goal* and *Information Entity* are shown in figure 5.

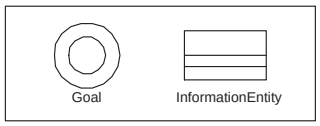

Figure 5. Mental State Entities

A *Goal* describes the wish of an agent to change to another state. Goals can be elementary or composed. *Information entities* are pieces of information that can be held by an agent or be transposed from one agent to another.

MESSAGE suggests five *views* with its diagram types, all taking another sub view of the whole system, so that the views could be overlapping. The views are

- Organization View
- Goal/Task View
- Agent/Role View
- Interaction View
- Domain View

The views will subsequently be introduced at the time of usage.

3.2 The MedPAge architecture

A basic task in designing an agent-based system is to decide which entities should be modelled as agents. A criteria for the identification of agents can be *individuality*. Agent instances are typically unique in their state and lifetime and individual pro-activeness and freedom in the form of autonomy are common agent characteristics.

Therefore, the identified co-ordination objects (patients and resources) are modelled as autonomous agents, i.e. patient-agents (P-Agents) and resource-agents (R-Agents). This allows the representation of every single co-ordination object with its own goals as a single agent, reflecting the existing, decentralized structures in hospitals (see also [4][14]). Each agent only knows its own schedule and constraints, i.e. for the patient-agents the necessary treatments, and for the resource-agents the patients to be treated. Through this encapsulation of information and restrictions to the object, respectively agent, which they belong to, multi-agent-systems are capable to react dynamically to changes, e.g. changed pathways or health state.

Additionally, knowledge agents (K-Agents) were modelled. The roles of K-Agents are implemented by the P- and R-Agents to make their states persistent when needed and to be able to access external legacy systems like healthcare

information systems in hospitals or embedded information systems in hardware resources.

The *organization view* shows concrete entities such as agents, roles and organizations. There are two different types of diagrams: *Structural Relationships* and *Acquaintance Relationships*. Both diagrams are based on UML class diagrams, that are extended by new entities.

The *structural relationship* diagram shows the structure of the whole system being decomposed to sub organizations. In MedPAge (see figure 6) there are two roles: P-Agent (representing a single patient) and R-Agent (managing a hospital resource). A hospital (as parent organization) consists of patient agents (as representatives for patients) and resources. Resources have an R-Agent, that manages the resource, and the resources schedule. A patient agent knows his medical path and has his personal schedule.

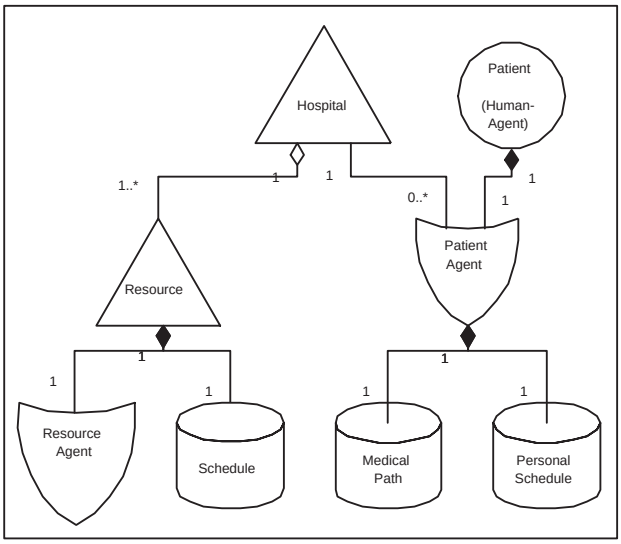

Figure 6. Structural relationships

The *acquaintance relationship* diagram focuses on the data flow between the entities that enable an agent to do his work. The MedPAge patient agent has to communicate with his personal schedule, the medical path and of course the resource agents to book the medical treatments he needs. On the other hand, the MedPAge resource agent has to communicate with the patient agent and with the resources schedule. This leads to the acquaintance relationship diagram shown in figure 7.

4 Co-ordination mechanism

Patient schedules are developed during negotiations between patient agents who want to get time slots for specific treatments, and resource agents who offer treatments at certain points in time. Firstly, the utility-functions for the

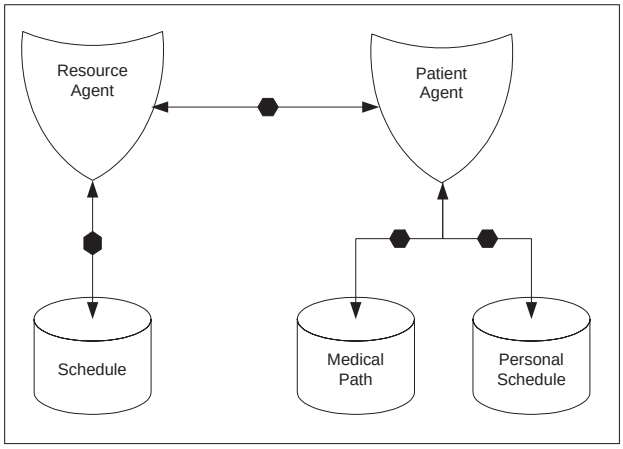

Figure 7. Acquaintance relationships

agents will be derived, and then the interaction protocols will be described.

4.1 Utility functions

4.1.1 Health state dependent utility functions

Obviously, in a hospital the health of a patient is the major determinant of the priority for this patient. The health of a patient can be somewhere between total health and the worst imaginable health state (as death is normally not the worst health state). Furthermore, illness can be interpreted as a loss of utility for a patient. If the disease is treatable, the opportunity cost C^{opp} for not curing the patient right away equals the difference between the achievable health state (through treatment) *z* and the patient's health state over the time $H(t)$. Formally, this can be expressed by

$$C^{opp}(t) = \int_0^t z - H(t)dt.$$

For the necessary cardinal measurement of health, we rely on the concept of "years of well being", because it already includes the time dimension. Therefore the question is, what time period *xT* of total health (1) equals one specific time period $1T$ of a certain health state *H*, i.e.

$$1T * H = xT * 1 \Leftrightarrow H = x.$$

For example, Torrance [12] states that one year with middle angina pectoris equals 0.7 years of well being.

In addition, it has to be considered, that the health state and the achievable health state can change over time. Therefore, the patient's opportunity costs are influenced by his current health state a, the development of his health state over time $H(t)$, and the maximal reachable health state through treatment z. If the health state does not change over time, i.e. $H(t) = a$, the opportunity costs are

$$C^{opp}(t) = \mu t; \mu = z - a.$$

If the health state of a patient worsens over time, assumptions about the course of the health state has to be made by a physician. If we assume - for clarity - a linear reduction by *b* of the health state, i.e. $H(t) = a - bt$, we get

$$C^{opp}(t) = zt - at + \frac{b}{2}t^2 = \mu t + \frac{b}{2}t^2.$$

Figure 8 shows an exemplary course of an illness with linear reduction of the health state, resulting in a quadratic, respectively convex opportunity cost curve.

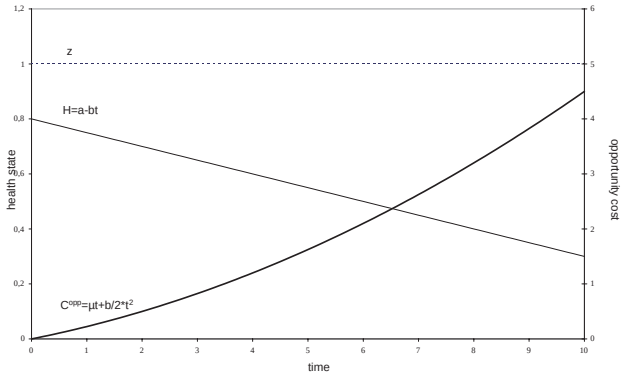

Figure 8. Linear reduction of health state

Finally, the maximal reachable health state can decrease (from z^{old} to z^{new}). In this case a utility loss $\bar{C}^{opp}$ for the rest of his life Δ^{rol} would result for the patient, i.e.

$$\bar{C}^{opp} = (z^{old} - z^{new}) \times \Delta^{rol}.$$

As "rest of life" (Δ^{rol}) is a parameter of this calculation, we face the ethical problem that younger patients get higher priority than older patients due to a longer assumed lifetime from now on. This problem can be avoided by handling those patients as emergency patients, i.e. the opportunity costs for a treatment after a reduction of the achievable health state are infinite.

If the health state of a patient determines his P-Agent's utility respectively cost function, the resulting curve is at least linear (in the case of a constant health state), but in most cases convex (if health decreases over time). If we have a convex curve, a task which could - ceteris paribus - start earlier than another task, has a higher priority (compare [4]).

4.1.2 Multi-attributive utility functions

Single agents can have multiple, even contradictory goals. Examples for possible contradictory goals of a single agent are a short stay in hospital versus blocked examinations/treatments, i.e. gap-minimisation or physician preferences. In figure 9 the case of a decreasing health state and the additional goal of gap minimisation is illustrated.

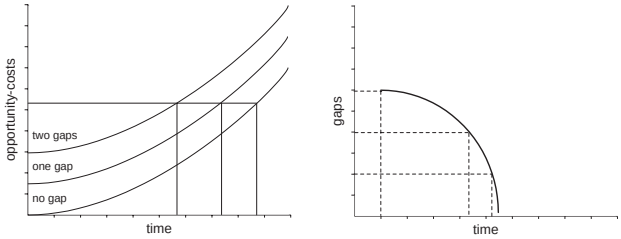

Figure 9. Multi-attributive opportunity-cost curves

To calculate the *value* of a gap, the time-trade off method could be used. In this case it has to be determined what amount of additional waiting time would be equal to one less gap, i.e.

$$C^{opp}(t_1) - (C^{opp}(t_2) + 1 * gap) = 0.$$

If the single subgoals are contradictive, a trade off between those goals is mandatory. However, there are indifferent combinations of these goals. These combinations can be drawn on an indifferent-curve (fig. 8 right). Due to the usage of cost- instead of utility-functions, the area above this (concave) curve is dominated by the area below this curve, i.e. all combinations below this curve indicate a gain of utility for this agent. In this figure it is also visible that the importance/weight of amount of gaps decreases with the time. This results from the decreasing health state.

4.2 Goal Achievement of Agents

Based on the described utility functions each agent tries to achieve its goals. In MESSAGE/UML the *goal/task* view shows goals, their decomposition and their relationships to the accomplishment of a task. The goals decomposition uses UML class diagrams to show the aggregation relationships between goals and their sub goals, while activity diagrams are used to show relationships between the accomplishments of tasks and the achieved goals.

In MedPAge a goal decomposition diagram is used to show how the goal of the whole system is decomposed into sub goals an agent aims to achieve. One decisive goal of the system is: "Let all treatments take place, and let them be optimized". This can be separated into "Plan all treatments" and "Optimize the treatments". Planning all treatments can be decomposed into "get knowledge of all treatments" and "book all treatments" as shown in figure 10.

The *agent/role* view would straight forward point out goals and tasks in a diagrams for each agent/role.

4.3 Agent interaction protocols

To improve their current scheduling situation, the agents need to interact, i.e. to co-ordinate their plans with the

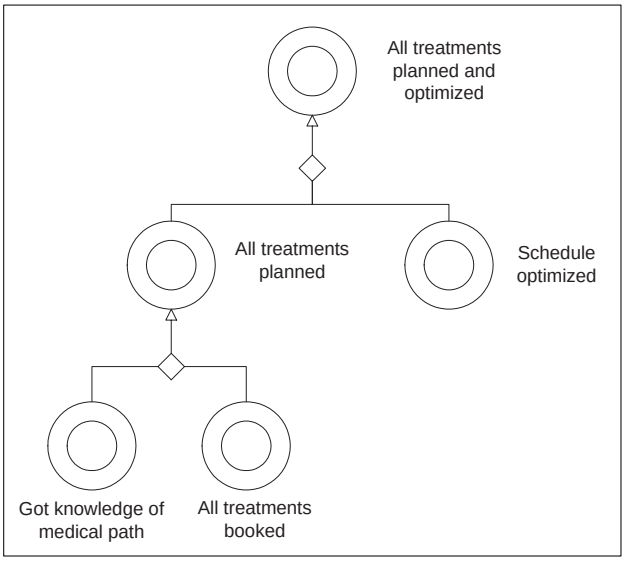

Figure 10. Goal decomposition

other agents (c.f. [10]). To this end a market mechanism for patient-scheduling was developed, in which the agents act as egoistic market participants, trying to maximise their own utility. Through the usage of utility functions, the agents act in a worth oriented domain, enabling them to compromise [10]. Based on economic theory, the total scheduling situation, i.e. the welfare of the multiagent system improves, as long as one agent can improve its situation without harming another agent, which can not happen in a system of non-altruistic agents (c.f. [9][13]).

For selling or buying timeslots, agents must be able to compute the price which they are willing to pay for a specific timeslot respectively the price they have to ask for giving up a timeslot. Based on the above derived utility functions, the price an agent is willing to pay for a timeslot is less or equal to his cost reduction (utility gain). On the other side, the price an agent asks for a timeslot is greater or equal to his additional costs caused by re-scheduling. From this we get

$$p = C^{opp}(t^{old}) - C^{opp}(t^{new}).$$

In our model, only the patient agents are pro-active. If they want to improve the position of a certain task, they contact the responsible unit and ask for a specific timeslot (demanders). In reaction to that query, the resource agent reserves this timeslot and contacts all affected patient-agents, i.e. the agents who own this timeslot, and informs them about the query of the initiator. The affected agents (sellers) now try to re-schedule and notify the initiator about their additional costs. If the alternative timeslots for the sellers are already occupied, they again become users for this timeslot and accumulate the invoked costs. This recursive process ends either when all agents have bought into free slots, or the cost reduction of the initiator is used up (ref. [9]).

For initial timeslots there it is necessary to calculate prices upon improvements, the first-come first-served priority rule is used. To this end, the agents do not need initial assets.

The *interaction view* of MESSAGE/UML describes an interaction between two or more agents. In general there is one interaction diagram showing the participants and the exchanged data for every relevant interaction. Note that here there is no concrete modeling of time aspects of the interaction. It is suggested to use AUML interaction diagrams to model an interaction in detail.

The interaction in MedPAge to book a medical treatment is shown in figure 11. The patient agent and the resource agent participate on this interaction. The patient agent sends an information entity called 'booking wish' and receives an information entity called 'confirmation sent' by the resource agent.

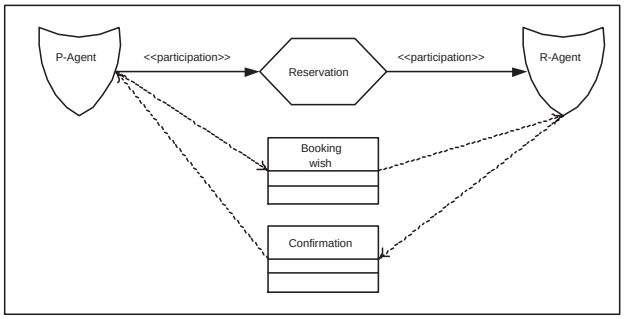

Figure 11. Interaction view

4.3.1 Detailed Interaction Modelling

The different modeling methodologies for agent oriented systems that have been described above also follow different approaches of modeling on more detailed levels such as interactions. The high level approaches MESSAGE and Gaia do not supported detailed interaction modeling. In the case of MESSAGE and Gaia an interaction is modeled by pointing out the participants of the interaction and the submitted information. There is no modeling of timing aspects and such high level approaches for interactions are only of a limited value for the MedPAge project. MaSE uses final state machines to model interactions. One edge of the machine defines what the machine will do when receiving a message. Since every participant of the interaction needs his own interaction diagram, this method is rather hard to read.

There are several specialized methods for modeling interactions in an agent-oriented context like Message Sequence Charts (MSC) [7], UAML [6], UAMLe [6], AUML [8] and EAUML [6].

UAML was the former FIPA standard for protocol diagrams (PD), that was replaced by AUML. AUML basically enhances UML sequence diagrams, which specify which and how interactions can be performed between participating entities. Compared to the final state machine diagrams in MaSE, this is the more intuitive and readable way of modeling interactions. There is no large difference between the two approaches of UAML and AUML in semantics - but UAML is not UML based like AUML. EAUML is a suggested extension of AUML. EAUML defines new connectors for the sequence diagrams that point out dependencies such as direct causality of two messages or message synchronization.

Finally (E)AUML was chosen to specify interactions in MedPAge (see [1] for details) since it provides rich modeling capabilities and has a wide adoption because of the underlying modeling methodology UML.

5 FIPA based implementation

The analysis resulted in the MedPAge architecture shown in figure 12 as the combination of the above described views.

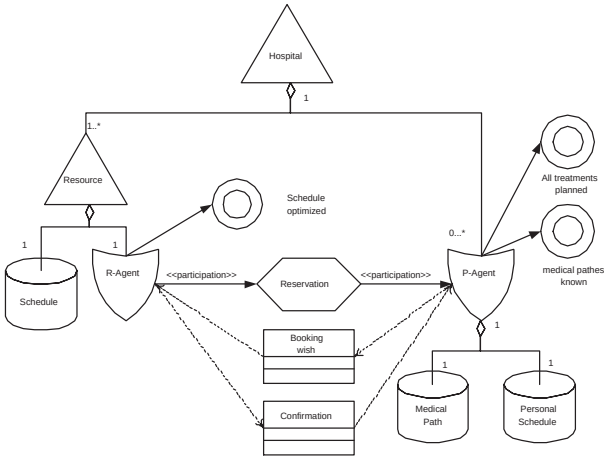

Figure 12. Architectural main entities of MedPAge (adapted MESSAGE/UML diagram)

The MedPAge system itself is implemented using Java and the FIPA-OS agent platform which conforms to the FIPA standard specifications for the interoperation of heterogeneous software agents. Figure 13 shows the details of the highly relevant classes of the implemented MedPAge application logic in an UML diagram.

Patient and resource agents extend the *FIPAOSAgent* and register with the FIPA-OS platform when they are started. Additionally, each resource agent registers with the Directory Facilitator of the platform to advertise the reservation

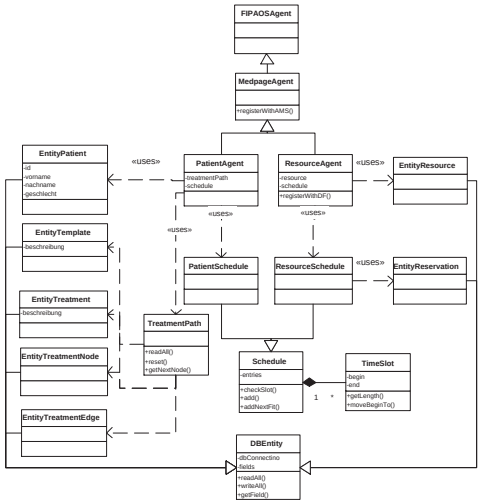

Figure 13. MedPAge UML class diagram (condensed)

service it provides. When a P-Agent needs to make a reservation for a certain resource it is able to locate the appropriate R-Agent by querying the Directory Facilitator Agent.

Communication between agents occurs via the Agent Communication Channel provided by FIPA-OS, allowing a distributed configuration of agents across interconnected platforms. The interactions modelled in (E)AUML are based on standard FIPA speech acts and wherever possible standardised interaction protocols were used, e.g. the FIPA Request Interaction Protocol.

Application data is stored in a relational database (Oracle 9i), accessed via JDBC by a set of classes which provide an object oriented interface to the other components. Thus, the entire system may be easily adapted to changes and extensions in the database model that might be required in subsequent steps of development.

An API to several elements of the domain is provided by another package to be used by the agents to accomplish their goals. TreatmentPath objects are generated from templates stored in the database and allow P-Agents to plan the reservations they need to make. Schedule objects are used by both P- and R-Agents to keep track of scheduled treatments and TimeSlot objects represent time intervals occurring in negotiations between agents.

A GUI is part of the MedPAge system as well. Each R- and P-Agents currently features it's own GUI allowing the agent's activity to be monitored and influenced. Additionally, a control center provides a means to start and stop

the agents and to add patients, resources and templates. A JSP- and applet-based web front-end for remote access is currently under development.

6 Conclusions

For patient scheduling, a generic hospital model was derived in which the patients and resources are identified as the relevant co-ordination objects. Based on their individualism, those co-ordination objects are modelled as autonomous agents, where each of those agents is in possession of its own schedule and goals. The goals of the agents are represented through (multi-attributive) cost functions, which they try to minimize. For agent co-ordination, a market mechanism (MedPaCo) was developed and the agent interaction protocols are modelled. For agent oriented software engineering the MESSAGE/UML methodology and the (E)AUML interaction modelling were adapted to support the FIPA-OS based implementation.

Further work has to address security aspects, as the data and information dealt with in hospitals is very sensitive (patient-records must be strictly protected against third parties) and mission critical (the information about a patient must be correct and always immediately accessible for the persons involved in treatment).

Acknowledgements

We would like to thank *Deutsche Forschungsgemeinschaft (DFG)* for funding this work under the project title *Agentenbasierte Planung und Koordination funktionsübergreifender Aktivitäten in medizinischen Behandlungspfaden (MedPAge)* as part of SPP 1083.

Furthermore we acknowledge the work of Christian Gräfe, Wilfried Röper, Henry Becker, Jürgen Gerstacker and Franz Rothlauf in helping us to develop and implement these concepts.

References

[1] M. Awizen and T. O. Paulussen. Modellierung von Kommunikationsprotokollen für die dezentrale, agentenunterstützte Koordination von Krankenhausprozessen. In W. M. T. Bauknecht, K.; Brauer, editor, *Informatik 2001*, pages 883–888. Österr. Computer-Ges., 9 2001.

[2] A. Bartelt and W. Lamersdorf. Agent-oriented concepts to foster the automation of e-business. In A. M. Tjoa, R. R. Wagner, and A. Al-Zobaidi, editors, *11th International Workshop on Database and Expert Systems (DEXA 2000)*, pages 775–779, London, 2000. IEEE, Los Alamitos, California; Washington; Brussels; Tokyo.

[3] G. Caire, F. Leal, P. Chainho, R. Evans, F. Garijo, J. Gomez, J. Pavon, P. Kearney, J. Stark, and P. Massonet. Agent oriented analysis using message/uml. In *Agent-Oriented Software Engineering (AOSE)*, pages 101–108, Montreal, 2001.

[4] K. Decker and J. Li. Coordinating mutually exclusive resources using GPGP. *Autonomous Agents and Multi-Agent Systems*, 3(2):133–157, 2000.

[5] R. Feinen. Patientenbezogene Organisation von Behandlungsprozessen. *Profitcenter und Prozessorientierung: Optimierung von Budget, Arbeitsprozessen und QualitätProfitcenter und Prozessorientierung: Optimierung von Budget, Arbeitsprozessen und Qualität*, 1999.

[6] J.-L. Koning, M.-P. Huget, J. Wei, and X. Wang. Extended modeling languages for interaction protocol design. In *Agent-Oriented Software Engineering (AOSE)*, pages 93–100, Montreal, 2001.

[7] Mauw, Reniers, and Willemse. Message sequence charts in the software engineering process. 2001.

[8] J. Odell, H. V. D. Parunak, and B. Bauer. Extending uml for agents. In G. Wagner, Y. Lesperance, and E. Yu, editors, *Agent-Oriented Information Systems Workshop at the 17th National conference on Artificial Intelligence; AOIS Worshop at AAAI 2000*, pages 3–17, Austin, TX, 2000.

[9] T. O. Paulussen, F. Rothlauf, and A. Heinzl. Konzeption eines Koordinationsmechanismus zur dezentralen Ablaufplanung in medizinischen Behandlungspfaden (MedPaCo). 2001.

[10] J. S. Rosenschein and G. Zlotkin. *Rules of Encounter*. MIT Press, Cambridge, MA, 1994.

[11] J. Schlüchtermann. *Patientensteuerung*. Verlag Josef Eul, Bergisch Gladbach, Germany, 1990.

[12] G. W. Torrance. Utility approach to measuring health-related quality of life. *Journal of Chronic Diseases*, 1987.

[13] M. Weigelt. *Dezentrale Produktionssteuerung mit Agenten-Systemen: Entwicklung neuer Verfahren und Vergleich mit zentraler Lenkung*. Dt. Univ.-Verl., Wiesbaden, 1994.

[14] C. Weinhardt and P. Gomber. Domänenunabhängige Koordinationsmechanismen für die dezentrale betriebliche Planung. *Information Management*, pages 6–16, 1996.

[15] M. F. Wood and S. A. DeLoach. An overview of the multiagent systems engineering methodology. In *The First International Workshop on Agent-Oriented Software Engineering (AOSE-2000)*, 2000.

[16] M. Wooldrigde, N. R. Jennings, and D. Kinny. The gaia methodology for agent-oriented analysis and design. *Autonomous Agents and Multiagent Systems*, 3(3):285–312, 2000.