Werkzeugunterstützte Prozessanalyse zur Identifikation von Anwendungsszenarien für Agenten

Dipl.-Inf. Holger Knublauch
Holger.Knublauch@faw.uni-ulm.de

Dr. Thomas Rose
Thomas.Rose@faw.uni-ulm.de

Forschungsinstitut für anwendungsorientierte Wissensverarbeitung (FAW)
Helmholtzstraße 16
89081 Ulm

Zusammenfassung

Derzeitige Entwicklungsmethoden für Multi-Agenten-Systeme bieten wenig Unterstützung bei der Beantwortung der Frage, wo und wann welche Arten von Agenten sinnvoll eingesetzt werden können. Dieser Artikel schlägt hierzu einen werkzeugunterstützten Ansatz vor, mit dem die bestehenden Arbeitsprozesse erfasst und dann auf zwei Arten auf Schwachstellen und Optimierungsmöglichkeiten analysiert werden können. Erstens stellt das Werkzeug verschiedene Sichten auf die am Prozess beteiligten Rollen, Aktivitäten und Informationsflüsse bereit. Zweitens umfasst es eine erweiterbare Bibliothek von Mustern, aus der halbautomatisch typische Anwendungsfälle für Agenten in den neuen Prozess eingesetzt werden können. Der Artikel schlägt hierzu einen Ansatz zur Beschreibung wiederverwertbarer, agentenbasierter Prozessbausteine vor.

1 Einleitung

Der diesem Artikel zugrundeliegende Agentenbegriff basiert auf der Definition von Jennings et al. [12], in der ein Agent ein in eine Umgebung eingebettetes, autonomes, und flexibles Software-System ist. Insbesondere verteilte, informationsintensive Arbeitsprozesse, wie die klinische OP-Planung mit Notfallpatienten, können vom Einsatz von Agenten profitieren [15, 17]. Mögliche Einsatzgebiete sind (vgl. [23]) die Akquisition und Sammlung von Patientendaten (*Informations-Agenten*), die Informationsfilterung und -aufbereitung (*Filter-Agenten*) [9]), die proaktive Weiterleitung von Informationen an potenzielle Interessenten (*Notifikations-Agenten* [10]), die Bearbeitung wiederkehrender Standard-Aufgaben (*Nachfrage- und Reminder-Agenten*), die semi-automatische Planung des Einsatzes von Ressourcen anhand der klinischen Situation (*Planungs-Agenten*), und die kontext-sensitive Visualisierung zur Bewältigung der Datenflut (*Interface-Agenten*).

Der Einsatz dieser und ähnlicher Agenten-Typen erfordert eine tiefgreifende Analyse der bestehenden Arbeitsprozesse und einen systematischen Entwurf der neuen, um Agenten erweiterten Anwendungsszenarien. In den vergangenen Jahren wurden einige Entwicklungsmethoden für die agenten-orientierten Software-Entwicklung publiziert (z.B. Gaia [26]). Diese sind jedoch in ihrem Reifegrad noch nicht annähernd mit industriellen, objektorientierten Methoden vergleichbar. Insbesondere fehlen Methoden und Werkzeuge, die den Entwicklern bei der Entscheidung helfen, wo und wann welche Agenten-Typen für die bestehenden Arbeitsprozesse sinnvoll einsetzbar sind.

In diesem Artikel wollen wir die Diskussion zu dieser Fragestellung anregen, und mögliche Lösungsansätze explorieren. Wir stellen hierzu ein Werkzeug vor, das den gesamten Entwicklungsprozess von Multi-Agenten-Systemen begleitet. Ausgehend von der Erfassung der bestehenden Szenarien als Prozessmodelle unterstützt das Werkzeug die Analyse der Modelle auf Schwachstellen und Optimierungsmöglichkeiten, und den Entwurf der (um Agenten erweiterten) neuen Prozessmodelle. Hierzu stellt das Werkzeug verschiedene Sichten auf die in den Prozessen beteiligten Rollen, Aktivitäten und Informationsflüsse bereit, und schlägt mögliche Anwendungsgebiete für Agenten vor. Diese können im nächsten Schritt auf eine ausführbare Agenten-Plattform (wie `FIPA-OS` [6]) abgebildet werden.

2 Bestehende Entwicklungsmethoden

Ebenso wie das Forschungsfeld der Multi-Agenten-Systeme selbst steckt die Arbeit an systematischen Entwicklungsmethoden für Agenten noch in den Kinderschuhen [11, 3]. In einer früheren Arbeit [14] haben die Autoren bei der Entwicklungsmethode *Gaia* [26] beispielsweise auf drei wesentliche Schwachpunkte hingewiesen. Erstens fehlt in Gaia eine "globale" Übersicht über das modellierte Szenario. Dies hat zur Folge, dass die Identifikation von Schwachstellen im Prozess erschwert wird. Zweitens bietet Gaia keinerlei Unterstützung (z.B. in Form von Heuristiken) zur Identifikation möglicher Anwendungsgebiete von Agenten-Typen. Die Agenten werden sozusagen "aus dem Hut gezaubert" und erst nachdem sie identifiziert sind systematisch entworfen. Drittens fehlen Werkzeuge, die für Methoden wie Gaia optimiert sind, und die die verschiedenen im Entwicklungsprozess zu erzeugenden Modelle miteinander verkoppeln und analysieren können.

Ähnliches kann über andere Entwicklungsmethoden, wie z.B. Agent-UML [1], gesagt werden. Hier werden zwar Formalismen zur Beschreibung von Agenten und deren Verhalten bereitgestellt, der Benutzer wird jedoch kaum bei der Identifikation der Agenten unterstützt. In seiner Übersicht über den Stand der Forschung schreibt Lind [19] zu einer ähnlichen Methode "[die Entwicklungsmethode von Kinny und Georgeff] provides only a very sketchy process model that leaves a lot of questions unanswered, eg. how to identify the roles, etc. This is the crucial factor in the entire design process as all of the models that are developed in subsequent steps depend on these decisions".

In diesem Artikel schlagen wir eine werkzeugunterstützte Methode vor, die Ansätze von Methoden wie Gaia aufgreift, jedoch die drei oben genannten Probleme abschwächt.

3 Übersicht des Entwicklungsprozesses

Abbildung 1 gibt eine Übersicht über einen möglichen Entwicklungsprozess für Multi-Agenten-Systeme. Der Prozess gliedert sich in die Phasen Erfassung, Analyse, Design (Entwurf), und Implementation.

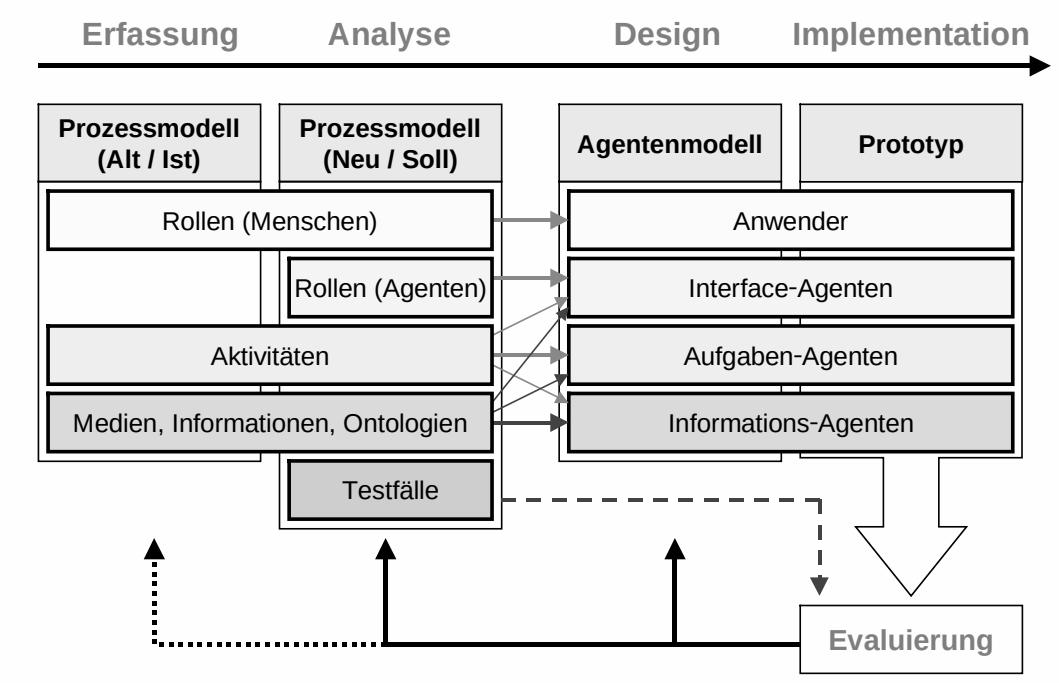


Abbildung 1: Übersicht über den Entwicklungsprocess.

In der Erfassung werden die bestehenden Arbeitsprozesse durch Domänen-Experten modelliert. Im Falle unseres Projektes AGIL [15] wird dies vom medizinischen Projektpartner unter Berücksichtigung von Anwenderfeedback durchgeführt. Die Erfassung liefert ein Ist-Modell, das die Rollen der menschlichen Akteure, deren Aktivitäten, und die ausgetauschten Medien und Informationen enthält. Aufbauend auf diesem Modell wird in der Analyse-Phase in einer Zusammenarbeit von Domänen-Experten und System-Entwicklern ein neues Prozessmodell entworfen, in dem das Ist-Modell durch Agenten erweitert wird. Hier tauchen neben menschlichen Rollen auch von Agenten ausgefüllte Rollen auf, die selbständig Aktivitäten ausführen können. Das entstehende Soll-Modell liefert die Grundlage für den Agenten- und System-Entwurf, der schliesslich zur Implementierung führt.

Die von uns für Design und Implementation gewählte Agenten-Architektur basiert auf der dreischichtigen Architektur von Sycara et al. [25], in der zwischen Interface-, Aufgaben- und Informations-Agenten unterschieden wird. Diese Dreiteilung ermöglicht eine recht intuitive und nahtlose Abbildung der Analyse-Modelle auf die Design-Modelle. Andere Arten von Agenten könnten jedoch ebenso verwendet werden. Im Design werden die in der

Analyse ermittelten menschlichen Rollen auf die Gruppe der Anwender abgebildet. Jede dieser Anwendergruppen interagiert mit dem System über auf sie zugeschnittene Interface-Agenten. Diese steuern die nicht-visuellen Aufgaben- und Informations-Agenten. Während sich Interface-Agenten auf die kontext-sensitive Informations-Präesentation und die Entgegennahme von Anweisungen beschränken, führen Aufgaben-Agenten weitesgehend eigenständige Aktionen durch. Aufgaben-Agenten werden also zumeist nur temporär erzeugt, bis sie eine Aufgabe erfüllt haben. Informations-Agenten stellen eine Schnittstelle zu dauerhaft verfügbaren Datenbeständen bereit, und dienen somit z.B. als Übersetzer zwischen den Aufgaben-Agenten und den zugrundeliegenden (Patienten-)Datenbanken. Alle Agenten verwenden die in der Analyse ermittelten Medien (Agenten-Nachrichten) und die Ontologien der darauf transportierten Informationen.

Schließlich kann der aus diesem Agentenmodell entstehende Prototyp evaluiert werden, wobei Testfälle, die in der Prozess-Analyse gewonnen wurden, herangezogen werden können. Die Schritte nach dem Design werden in diesem Artikel nicht weiter betrachtet, dessen Hauptanliegen die Fragestellung ist, wie aus einem bestehenden (Ist-) Prozessmodell ein um Agenten erweitertes (Soll-) Analyse-Modell gewonnen werden kann. Dies wird im folgenden behandelt.

4 Erfassung der bestehenden Prozesse

Der Prozess- und Agenten-Modellierung liegt das in Abbildung 2 illustrierte Meta-Modell zugrunde. Dieses Meta-Modell wurde entworfen um einen werkzeugunterstützten, nahtlosen Übergang von einem intuitiven Prozessmodell in ein lauffähiges Multi-Agenten-System zu ermöglichen.

Grundlegende Bausteine der Prozessmodelle sind *Prozesse*, die aus Unter-Prozessen und *Aktivitäten* bestehen können. Prozesse und Aktivitäten sind über Nachfolger-/Vorgänger-Beziehungen zu Prozessketten verbunden. Aktivitäten (z.B. "Empfang durch die Pflege") werden von bestimmten *Rollen* (z.B. Rettungsstellen-Arzt, Pflege) an einem gegebenen *Ort* (*Site*, z.B. Behandlungsraum) durchgeführt. Rollen entsprechen dem aus der objektorientierten Analyse bekannten Rollenbegriff, d.h. einzelne Akteure können auch mehrere Rollen annehmen. Rollen werden entweder von Menschen oder von Agenten ausgeführt, wobei verschiedene Typen von Agenten (wie z.B. Aufgaben- oder Interface-Agenten) unterschieden werden. Aktivitäten können Eingabe-Medien erhalten (= Leserecht), und Ausgabe-Medien erzeugen (= Schreibrecht). Medien tragen eine oder mehrere Informationseinheiten und sind entweder analog (z.B. Telefonat, Fax, Patientenakte) oder Nachrichten in einer Agenten-Kommunikationssprache wie FIPA ACL [7]. Die Definition der Medien kann zur Bereitstellung von Kommunikationspfaden und Nachrichten zwischen Agenten genutzt werden. Insbesondere kann das Datenformat (die Ontologie) der Agenten-Nachricht spezifiziert werden, die für die spätere Implementierung bedeutsam ist. Unser Ansatz unterstützt hier vor allem die Spezifikation von JavaBeans/KBeans-Klassen [16], sodass der Code von kommunizierbaren Objekten (den Inhalten der Nachricht) automatisch generiert werden kann. Weiterhin können Agenten-Nachrichten mit Attributen versehen werden, die Sprechakte

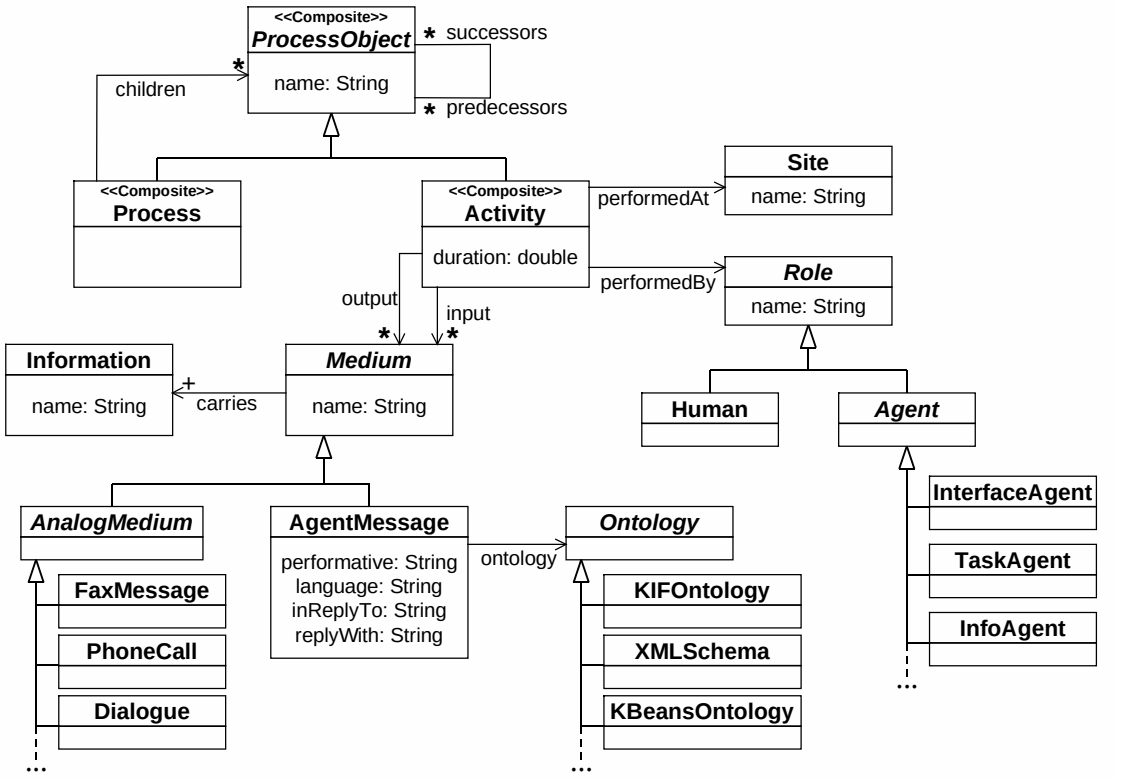


Abbildung 2: Ein Ausschnitt unseres Meta-Modells zur Prozessmodellierung.

gemäß der FIPA-Spezifikation [7] beschreiben. Das Meta-Modell ist bewusst einfach gehalten, sodass es zunächst keine weiterführenden Agenten-Konzepte (etwa den internen Agenten-Zustand gemäß einer BDI-Architektur) unterstützt. Die zugrundeliegende Entwicklungsplattform unter Java ist jedoch leicht erweiterbar, z.B. durch das Hinzufügen neuer Attribute.

Die Instanzen dieses Meta-Modells werden sowohl in der Prozess-Erfassung (ohne Agenten), als auch in der Analyse (mit Agenten und Ontologien) mit einem von uns entwickelten Werkzeug (der *AGILShell*, Abbildung 3) grafisch modelliert. Dieses Werkzeug stellt Prüfmechanismen bereit, die die syntaktische Korrektheit und Vollständigkeit der Modelle (gemäß struktureller Bedingungen) verifizieren können, und somit den (auch unerfahrenen) Modellierern helfen. Desweiteren kann aus den Prozessmodellen automatisch Java-Code generiert werden, der als Schablone für eine Implementierung auf Basis der Agenten-Plattform `FIPA-OS` [6] genutzt werden kann. Details zur Code-Generierung können hier aus Platzgründen nicht ausgeführt werden. Die AGILShell ist (basierend auf Introspektion) sehr generisch, sodass neu hinzugefügte Attribute ohne Anpassung des Werkzeuges mitediert werden können.

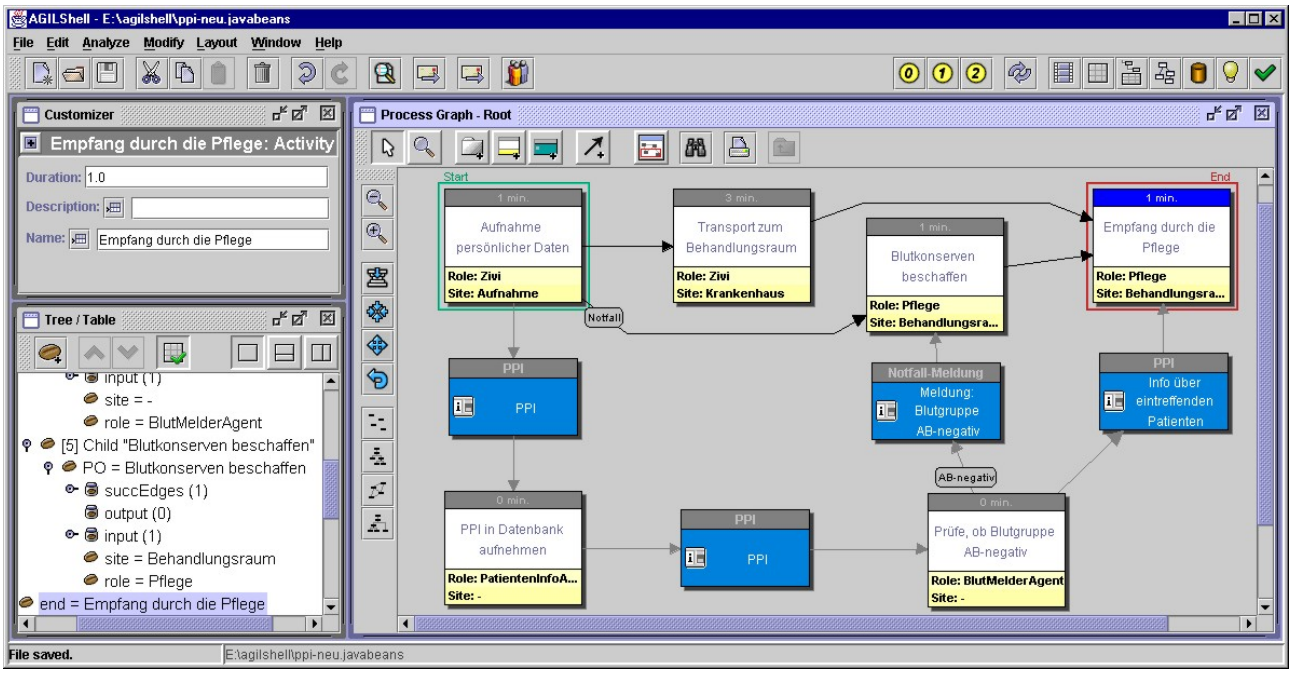


Abbildung 3: Das Prozessmodellierungswerkzeug AGILShell.

5 Verschiedene Sichten auf die Prozessmodelle

Die wie oben beschrieben erzeugten Modelle können mit verschiedenen Sichten/Views visualisiert und analysiert werden. Die Sichten sind dadurch motiviert, dass die visuelle Perzeption ein wichtiges Arbeitsinstrument bei der Modellierung ist [20, 22].

- Die workflow-artige *Prozess-Sicht* (wie in Abbildung 3 gezeigt) ist der wichtigste Bearbeitungsmodus.
- Die lokale *Rollen-Sicht* (siehe Abbildung 4, links) stellt die Lebenszyklen (vgl. die Gaia-Entwicklungsmethode [26]) einer Rolle dar, also die Abfolge der Aktivitäten, in die eine Rolle üblicherweise involviert ist, sowie deren Schreib- und Lese-Rechte auf Medien.
- Die *Kommunikations-Sicht* (siehe Abbildung 4, rechts) stellt Rollen als Knoten, und den Medienaustausch als Kanten dar. Alternativ haben wir eine tabellarische Darstellung implementiert, in der die Rollen als Spalten, und die Medien als Zeilen angezeigt werden. Eine Darstellung als UML-Message-Sequence-Diagramm ist ebenfalls möglich.
- Die *Raum-Sicht* stellt die Räume (des Krankenhauses), an denen die Aktivitäten durchgeführt werden, als Knoten und den Medienfluss als Kanten dar.

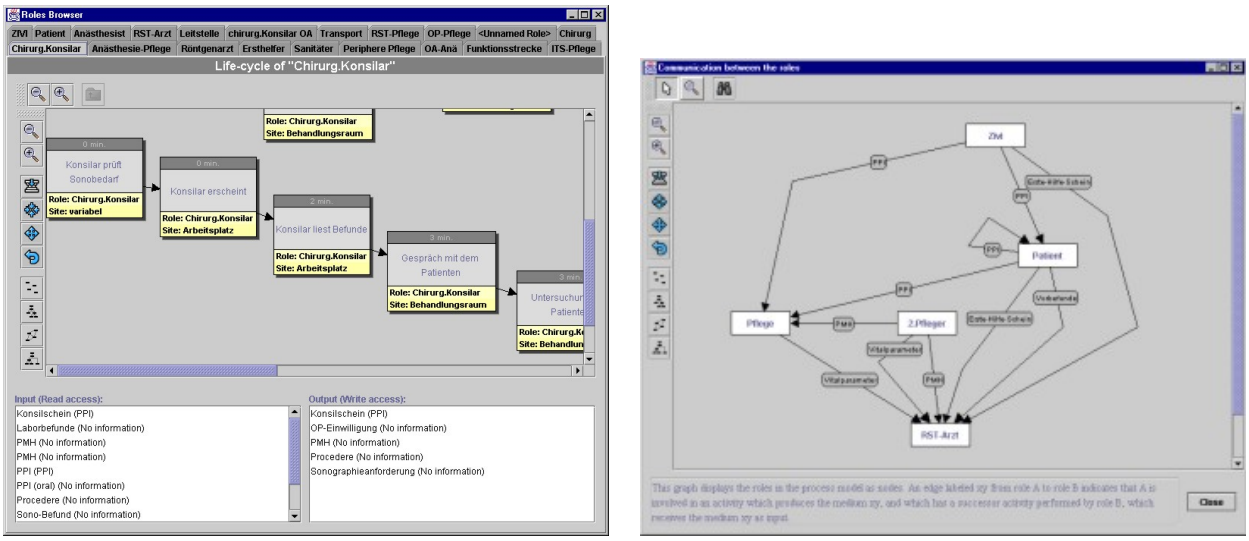


Abbildung 4: Unterschiedliche computergenerierte Sichten auf den modellierten Prozess: Die Rollen-Sicht (links) stellt Lebenszyklen, sowie Schreib- und Leserechte von Rollen dar. Die Kommunikations-Sicht visualisiert den Informationsfluss zwischen den Rollen.

Diese verschiedenen Sichten lassen sich automatisch aus dem Meta-Modell ableiten und erlauben die Verifizierung und Validierung der Modelle, sowie (im Sinne einer Methode wie Gaia) die semi-automatische Ableitung von Agenten-Eigenschaften. Beispielsweise ermöglicht die Kommunikations-Sicht die Aufdeckung von Engpässen und Informationssenken, und gibt Hinweise, welchen Umfang der Interface-Agent einer Rolle haben muss, sowie welche Ontologien die beteiligten Informations-Agenten "verstehen" müssen.

6 Anwendungsmuster von Agenten

Im vorherigen Abschnitt wurden verschiedenen Arten der Visualisierung vorgestellt, die den Entwicklern indirekt bei der (manuellen) Analyse der Ist-Prozesse helfen. Der bestehende Prozess wird hierbei nach Anwendungsgebieten von Agenten durchsucht, und entsprechend zu einem Soll-Prozess umgebaut. Der nächste Schritt besteht nun darin, die in diesen Modellierungsprozess eingeflossenen Entscheidungen und Vorgehensweisen zu verallgemeinern, damit sie in späteren Szenarien wiederverwendet werden können.

Wir schlagen hierzu einen Ansatz vor, der stark an den sog. Entwurfsmustern (*Design Patterns*) [8] aus der objektorientierten Softwareentwicklung angelehnt ist. Entwurfsmuster beschreiben wiederkehrende Lösungen und "Best Practices" zu häufigen Problemstellungen. Entwurfsmuster stellen so Modellierungswissen bereit, von dem andere lernen können, und definieren außerdem ein Vokabular (eine sog. *Pattern Language*), mit dem Entwurfsentscheidungen und Strukturen auf hohen Abstraktionsniveau kommuniziert werden können. Der Gedanke der Entwurfsmuster wurde von Ferstl, Sinz et al. [5] auf den Bereich der

Prozessmodellierung übertragen, und wird von uns nunmehr zur Prozessanalyse für Multi-Agenten-Systeme genutzt.

Die von uns vorgeschlagenen *Agenten-Anwendungsmuster* bestehen im Wesentlichen aus einer konfigurierbaren Schablone von Prozessmodellen, in denen typische Anwendungsszenarien von Agenten beschrieben werden, und aus Hinweisen zur Anwendbarkeit der Agenten und den Konsequenzen der Anwendung. Diese Muster können in einer modularen Bibliothek organisiert, und von einem intelligenten "Browser" angezeigt werden. Wenn die Bedingungen für die Anwendbarkeit eines Musters formal fassbar sind, kann ein Modellierungswerkzeug sogar semi-automatisch Vorschläge generieren, an welchen Stellen sich der Einsatz von Agenten anbietet. Dies kann durch den Vergleich der syntaktischen Strukturen für den modellierten Ist-Prozess implementiert werden. Die semantische Struktur der Prozesse, also die Bedeutung und Aufgabe der einzelnen Aktivitäten im Gesamtprozess, kann jedoch bisher nur manuell verglichen werden. Hier könnten die modellierten Prozesse um weitere semantische Meta-Informationen angereichert werden, sodass wiederum automatisierbare Vergleichsverfahren genutzt werden können.

Bevor wir diesen Ansatz anhand zweier Beispiele verdeutlichen, wird im folgenden ein einheitlicher Beschreibungsrahmen für Anwendungsmuster definiert.

6.1 Beschreibungsschema für Agenten-Anwendungsmuster

Ein Agenten-Anwendungsmuster wird durch folgende Aspekte definiert (vgl. [8, 5]):

1. **Name und Synonyme.**

2. **Übersicht.** Eine kurze Zusammenfassung, die etwa die folgenden Fragen beantwortet: Was tut dieses Muster? Was ist seine Vorgehensweise? Welches spezifische Problem adressiert es?

3. **Anwendbarkeit.**

   Eine Beschreibung der (Ist-) Prozesse, auf die das Muster angewendet werden kann.

   (a) **Szenario (Syntax).**
   Beschreibung der syntaktischen Struktur (Prozessmodelle) der Anwendungsszenarien (z.B. im Meta-Modell aus Abbildung 2, oder ähnlich der Context Check Language aus [5]). Die beschriebenen Teilprozesse können "Ankerpunkte" (z.B. bestimmte Aktitivitäten) haben, an denen die spätere Lösung ansetzen kann.

   (b) **Rahmenbedingungen (Semantik).**
   Zusätzliche (semantische) Rahmenbedingungen, die typische Anwendungsgebiete des Musters einschränken.

4. **Problem.**

   Das Problem (bzw. der Schwachpunkt) der in dem obigen Szenario auftritt und durch den Einsatz von Agenten behoben werden kann (z.B. "Redundanz").

5. **Lösung.**

   Eine Beschreibung der (Soll-) Prozesse, die nach Einbau der Agenten entstehen.

   (a) **Strukturänderungen (Syntax).**
   Beschreibung der Schritte, die zum Umbau des Prozessmodells erforderlich sind. Dies sind (vgl. [18]) Schritte zum *Hinzufügen* neuer Prozessbausteine, zum *Ändern* von Attributen (z.B. Ersetzen einer menschlichen Rolle durch einen Agenten), sowie zum *Löschen* von Teilprozessen. Bei diesen Umbauaktionen sind Aktivitäten und Rollen parametrisiert, d.h. sie müssen bei der Instanziierung durch konkrete Elemente aus dem Ist-Modell ersetzt werden. Hierzu kann auf die Ankerpunkte des Szenarios verwiesen werden.

   (b) **Aktivitäten, Rollen, Beziehungen (Semantik).**
   Details zu den in den Strukturänderungen genannten Modell-Elementen. Hier sollte z.B. eine Beschreibung des Kommunikations- und Verhandlungsverhaltens der neuen Agenten erfolgen.

6. **Beziehungen** (zu anderen Anwendungsmustern der Bibliothek).

   (a) **Komposition.** Liste von Teil-Mustern (falls vorhanden).

   (b) **Varianten.** Liste ähnlicher Muster (falls vorhanden).

7. **Konsequenzen.**

   Textuelle Beschreibung der Auswirkungen des Musters auf den Gesamtprozess.

   (a) **Vorteile.** Erwarteter Nutzen durch den Einsatz der Agenten.

   (b) **Risiken.** Mögliche Probleme und Gefahren durch den Umbau des Prozesses.

8. **Beispiele und Referenzen.**

   Verweise auf Anwendungsbeispiele zur Illustration.

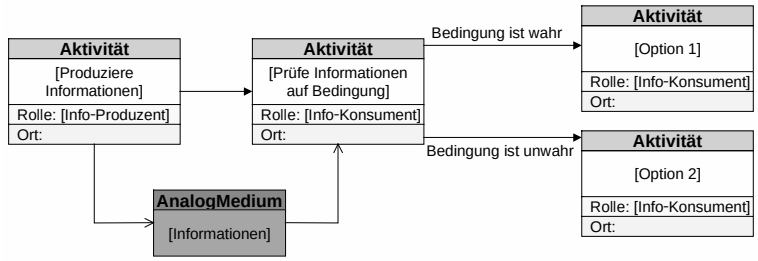
6.2 Beispiel "Melde-Agent"

1. **Name und Synonyme.**

   Notifikation über optionale Informationen, pro-aktive Informationsfilterung.

2. **Übersicht.**

   Melde-Agenten können eingesetzt werden, wenn die zukünftigen Aktivitäten einer Rolle vom Inhalt einer bestimmten Information abhängen. Agenten überwachen das Eintreffen neuer Informationen und informieren den Empfänger, falls diese Informationen eine bestimmte Bedingung erfüllen.



3. **Anwendbarkeit.**

   (a) **Szenario (Syntax).**

   (b) **Rahmenbedingungen (Semantik).**

   Ein Informationsproduzent erzeugt ein (nicht-digitales) Informationsmedium (z.B. eine Patientenakte), das für einen Informationskonsumenten später von Bedeutung ist. Der Konsument prüft das Medium und führt je nach Inhalt der Informationen eine andere Folgeaktivität aus. Diese können zeitkritisch sein, z.B. könnte je nach Wichtigkeit der in einer Patientenakte enthaltenen Informationen ein dringender Zwischenschritt ausgeführt werden.

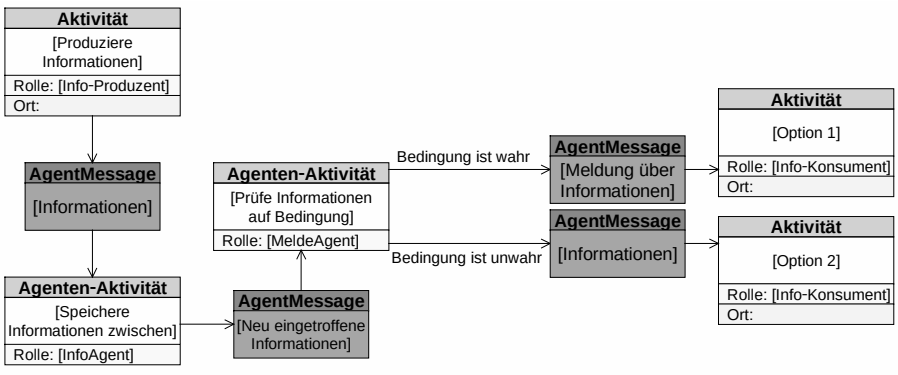
4. **Problem.**

   Ein möglicher Schwachpunkt in diesem Szenario ist, dass der Konsument die für die weitere Planung der Folgeaktivitäten notwendigen Informationen erst zu spät erhält.

5. **Lösung.**

   (a) **Strukturänderungen (Syntax).**

   Der Prozess wird an den oben gezeigten Ankerpunkten um folgende Struktur erweitert. Das analoge Medium wird gelöscht.



(b) **Aktivitäten, Rollen, Beziehungen (Semantik).**
Der Informationsgehalt wird digitalisiert (z.B. digitale Patientenakte) und ein Informationsagent übernimmt die Verwaltung der Informationen. Der Agent speichert das Informationsmedium (in einer Datenbank) zwischen, und leitet neu eintreffende Informationen an die Melde-Agenten der potenziellen Interessenten weiter. Die Melde-Agenten kennen die Preferenzen der Informationskonsumenten, prüfen die Informationen und lösen je nach Ergebnis der Prüfung die entsprechenden Folgeaktivitäten aus.

6. **Beziehungen**.

   (a) **Komposition.**

   (b) **Varianten.**

7. **Konsequenzen.**

   (a) **Vorteile.** Die Effizienz des Informationsaustausches wird gesteigert, weil der Agent dem Konsumenten Routine-Aufgaben (das Prüfen von Informationen) abnimmt. Die Prüfung erfolgt sofort und kann ursprünglich sequenzielle Prozesse parallelisieren.

   (b) **Risiken.** Es besteht die Gefahr, dass wichtige Informationen zum falschen Zeitpunkt weitergeleitet werden, weil die vom Konsumenten vorgegebene Prüfbedingung zu strikt ist.

8. **Beispiele und Referenzen.**

   Ein Anwendungsszenario des Melde-Agenten im Bereich der Aufnahme von Notfallpatienten ist in Abbildung 3 gezeigt. Im ursprünglichen Prozess nimmt ein Zivildienstleistender die persönlichen Daten eines neu eingetroffenen Patienten auf, und transportiert ihn in den Behandlungsraum, wo er von der Pflege empfangen wird. Hat der Patient eine seltene Blutgruppe (hier: AB-negativ), so müssen spezielle Blutkonserven beschafft werden, während dies für häufige Blutgruppen nicht erforderlich ist. Agenten können die Prüfung der Blutgruppe automatisieren, und die Pflegekraft im Behandlungsraum schon frühzeitig informieren. Hierdurch kann die Transportzeit des Patienten zur Beschaffung der Blutkonserven besser genutzt werden.

6.3 Beispiel "Nachfrage-Agent"

1. **Name und Synonyme.**

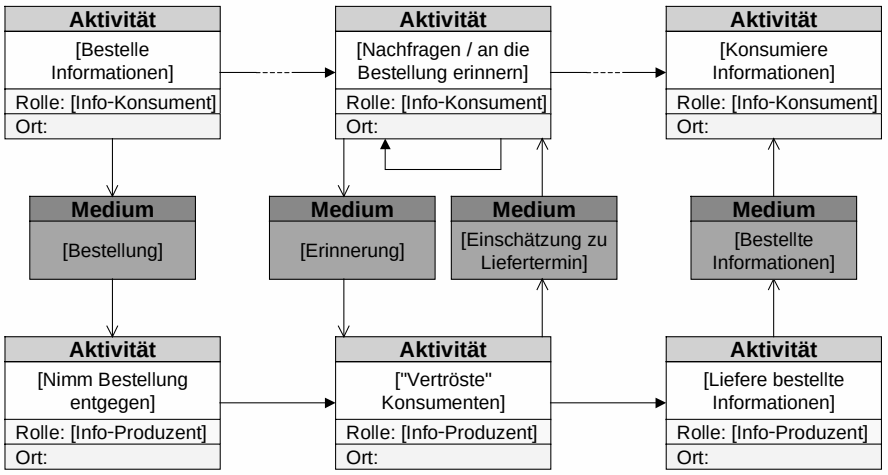
   Nachfrage-Agent, Erinnerer, Reminder.

2. **Übersicht.**

Wenn eine zeitkritische Aktivität auf eine termingerechte Informationslieferung angewiesen ist, kann ein Agent eingesetzt werden, um in regelmäßigen Abständen beim Informationslieferanten nachzufragen.

3. **Anwendbarkeit.**

   (a) **Szenario (Syntax).**



   (b) **Rahmenbedingungen (Semantik).**
   Ein menschlicher Akteur (z.B. ein Chirurg) bestellt Informationen bei einer anderen Abteilung (z.B. Labor). Weil er auf die Lieferung der Informationen zu einem bestimmten Zeitpunkt angewiesen ist, fragt er hin- und wieder beim Informations-Produzenten nach. Dies ist zwar ineffizient, aber eine in der realen Welt übliche Praxis.

4. **Problem.**

   Die wiederholte Nachfragerei kostet Zeit und es fehlt an Planungssicherheit.

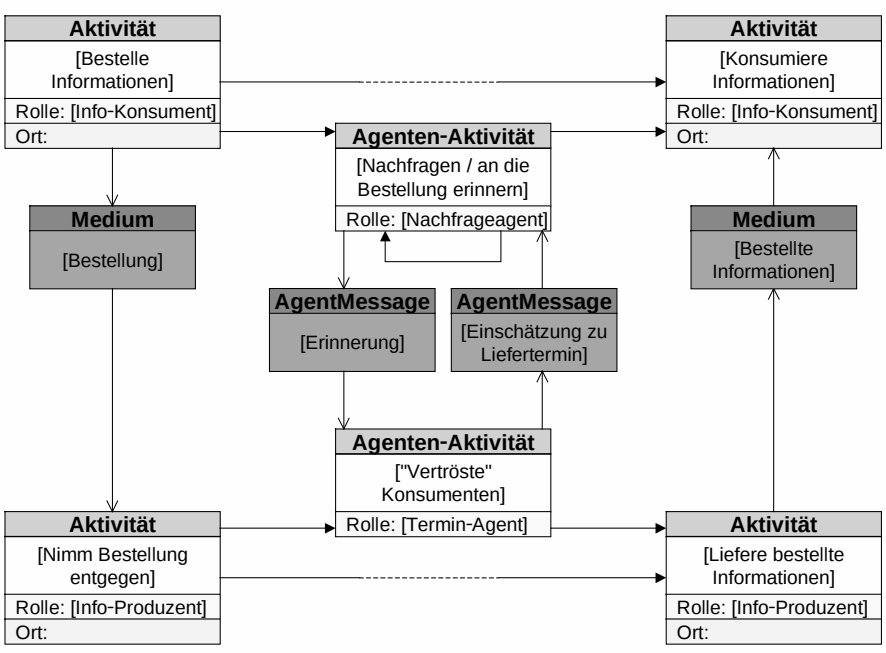
5. **Lösung.**

   (a) **Strukturänderungen (Syntax).**

   (b) **Aktivitäten, Rollen, Beziehungen (Semantik).**
   Das Aushandeln eines Liefertermins für die bestellten Informationen wird von zwei Agenten übernommen, die die jeweiligen Interessen ihrer Auftraggeber kennen und vertreten.

6. **Beziehungen**.

   (a) **Komposition.**



(b) **Varianten.**

7. **Konsequenzen.**

   (a) **Vorteile.** Die Effizienz der "normalen" Arbeit der Beteiligten kann gesteigert werden, weil diese keine Zeit mehr mit organisatorischen Standardaufgaben verbringen müssen.

   (b) **Risiken.** Die menschlichen Akteure werden aus dem Prozess ausgesperrt. Hierbei kann es zu unerwarteten Automatismen kommen, und die sozialen Kontakte können (zu) stark verringert werden.

8. **Beispiele und Referenzen.**

   Ein typisches Anwendungsszenario für dieses Muster findet sich in der abteilungsübergreifenden Kommunikation in Kliniken. Hier ist das Labor ein Dienstleister für andere Abteilungen, wie z.B. der Chirurgie. Wird beispielsweise während einer Operation dringend ein Befund über eine Blutuntersuchung benötigt, so muß das Labor versuchen, seinen Arbeitsplan entsprechend anzupassen. Da in modernen Laboren die Analyse von Blutproben semi-automatisch und mit Hilfe von Rechnern erfolgt, kann ein Agent direkt auf den aktuellen Arbeitsplan zugreifen und autonom auf Anfragen bezüglich der Zeitplanung reagieren. Hierdurch werden weder Labormitarbeiter noch Chirurgen unnötig von ihrer eigentlichen Arbeit abgehalten.

7 Zusammenfassung und Diskussion

In diesem Artikel haben wir einen Ansatz zur werkzeugunterstützten Identifikation von Anwendungsgebieten für Agenten vorgestellt. Der Ansatz basiert auf der Erfassung und Analyse von bestehenden Prozessen, für die der Einsatz von Agententechnologie vorgesehen ist. Aus dem erfassten Ist-Prozess wird schrittweise ein agentenbasierter Soll-Prozess abgeleitet, aus dem wiederum die Topologie eines Multi-Agenten-Systems gewonnen werden kann. Wir haben gezeigt, wie aus einem erfassten Ist-Prozess verschiedene Sichten generiert werden können, die bei der Schwachstellenanalyse helfen. Außerdem haben wir ein Verfahren zur Beschreibung und Wiederverwendung von Modellierungswissen mit Hilfe von Agenten-Anwendungsmustern vorgestellt. Die Unterstützung bei der Identifikation von Agenten-Szenarien ist nur semi-automatisch, weil die reine Struktur von bestehenden Prozessen unzureichend für den Vergleich mit den Mustern ist, und die "Semantik" der Teilprozesse schwierig formal fassbar ist. Wir arbeiten daher mit visuellen Hilfsmitteln, die die manuelle Erkennung von Szenarien erleichtern. Wir sehen ausserdem bereits die Bereitstellung eines durchsuchbaren Agenten-Katalogs als eine große Hilfe an, weil dadurch die Modellierer beim gezielten Durchsuchen der Prozesse unterstützt werden.

Die Methode ist sehr "leichtgewichtig", d.h. sie basiert auf wenigen Modellierungs-Artefakten. Im Gegensatz zu komplexeren Methoden wie CommonKADS oder Gaia (siehe [3] für eine Übersicht), ist der von uns verfolgte Ansatz insbesondere für Projekte mit zunächst unklaren Anforderungen geeignet. Die Leichtgewichtigkeit der Methode bedeutet zudem, dass sie auf spezielle Agenten-Paradigmen (z.B. BDI) angepasst werden kann.

Während die Umsetzung der Sichten im Modellierungswerkzeug AGILShell (siehe Abbildung 3) schon weit fortgeschritten ist und lediglich ein technisches Problem darstellt, befindet sich unsere Arbeit an den Anwendungsmustern noch in einem frühen Stadium. Wir haben mit der Umsetzung einer Bibliothek von Anwendungsmustern begonnen, die mit Hilfe der KBeans-Technologie [4] in die AGILShell eingebaut ist. Wir nutzen dieses Werkzeug im Rahmen des Projekts AGIL zur Analyse und Optimierung der Informationslogistik in klinischen Arbeitsprozessen. Das Werkzeug und die verwendete Datenstruktur zur Prozessmodellierung können relativ einfach um zusätzliche, anwendungsspezifische Klassen und Attribute erweitert werden und so zur Modellierung spezieller Agenteneigenschaften genutzt werden. Wir arbeiten außerdem an einer (mit FIPA-OS implementierten) komponenten-basierten Bibliothek von generischen Agenten-Typen (z.B. Melde-Agenten), die über Parameter (z.B. Art der zu überwachenden Informationen) auf ein gegebenes Szenario angepasst werden können.

Die Agenten-Anwendungsmuster werden in einer Java-basierten Datenstruktur (Ontologie) formalisiert, die von einem Wissens-Ingenieur bearbeitet wird. Es könnte hierbei allerdings noch von Nutzen sein, unseren Ansatz mit verwandten Ansätzen (z.B. [2, 21], Use Cases [11], oder den Role Models von Kendall [13]) zu harmonisieren, um so zu einem generischen Rahmenwerk zur Beschreibung von Agenten zu kommen. Weiterhin sind Ansäetze zur Wiederverwendung von Erfahrung bei der Prozessmodellierung [24] zum Vergleich von Ist- und Soll-Modellen anwendbar.

Danksagung

Die Autoren danken Holger Köth und Prof. Dr. Wolfgang Friesdorf für die Zusammenarbeit im Projekt AGIL. AGIL wird unterstützt von der Deutschen Forschungsgemeinschaft (DFG) im Rahmen des Schwerpunktprogramms 1083 "Intelligente Softwareagenten und realitätsnahe Anwendungsszenarien".

Literatur

[1] B. Bauer, J.P. Mueller, and J. Odell. Agent UML: A Formalism for Specifying Multiagent Software Systems. In P. Ciancarini and M. Wooldridge, editor, *First Int. Workshop on Agent-Oriented Software Engineering (AOSE-2000)*, Limerick, Ireland, 2000.

[2] S. Bussmann, N. Jennings, and M. Wooldridge. On the Identification of Agents in the Design of Production Control Systems. In P. Ciancarini and M. Wooldridge, editor, *Agent-Oriented Software Engineering*, pages 141–162. Springer-Verlag, Berlin, 2001.

[3] P. Ciancarini and M. Wooldridge. First Int. Workshop on Agent-Oriented Software Engineering (AOSE-2000). In 22. Int. Conf. on Software Engineering, 2000.

[4] FAW Ulm. KBeans Homepage. `http://www.faw.uni-ulm.de/kbeans`, 2000.

[5] O.K. Ferstl, E.J. Sinz, C. Hammel, M. Schlitt, S. Wolf, K. Popp, R. Kehlenbeck, A. Pfister, H. Kniep, N. Nielsen, and A. Seitz. WEGA: Wiederverwendbare und erweiterbare Geschäftsprozeß– und Anwendungssystemarchitekturen. Forschungsbericht, `http://www.seda.sowi.uni-bamberg.de/wega`, 1998.

[6] FIPA-OS. FIPA-OS Homepage. `http://fipa-os.sourceforge.net/`, 2001.

[7] Foundation for Intelligent Physical Agents. FIPA Specification. `http://www.fipa.org`, 2000.

[8] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addision-Wesley, 1995.

[9] B. Grote, T. Rose, and G. Peter. Filter & Broker – An Integrated Architecture for Information Mediation of Dynamic Source. *Journal of Organizational Computing and Electronic Commerce*, (Accepted for publication), 2001.

[10] I. Hawryszkiewycz and T. Rose. Notification Agents for Maintaining Awareness. In *Proc. of the Second Int. Conf. on Concurrent Engineering: Research and Applications*, pages 305–314, McLean, Va., 1995.

[11] C.A. Iglesias, M. Garijo, and J. Gonzales. A Survey of Agent-Oriented Methodologies. In *Proc. of the 5th Workshop on Intelligent Agents (ATAL-98)*, Paris, France, 1998.

[12] N. Jennings, K. Sycara, and M. Wooldridge. A Roadmap of Agent Research and Development. *Int. Journal of Autonomous Agents and Multi-Agent Systems*, 1(1):7–38, 1998.

[13] E. Kendall. Role Modeling for Agent System Analysis, Design, and Implementation. In *Proc. of the First Int. Symposium on Agent Systems and Applications (ASA'99), Third International Symposium on Mobile Agents (MA'99)*, Palm Springs, 1999.

[14] H. Knublauch, T. Rose, H. Köth, and W. Friesdorf. Modellierung des Wöchnerinnenszenarios mit der Gaia-Methode nach Wooldridge et al. In *Zweites Kolloquium des DFG-SPP "Intelligente Softwareagenten und betriebswirtschaftliche Anwendungsszenarien"*, Ilmenau, Germany, 2000.

[15] H. Knublauch, T. Rose, and M. Sedlmayr. Towards a Multi-Agent System for Proactive Information Management in Anesthesia. In *Proc. of the Agents-2000 Workshop on Autonomous Agents in Health Care*, Barcelona, Spain, 2000.

[16] H. Knublauch, M. Sedlmayr, and T. Rose. Design Patterns for the Implementation of Constraints on JavaBeans. In *Proc. of the NetObjectDays2000*, Erfurt, Germany, 2000.

[17] H. Köth, W. Friesdorf, H. Knublauch, and T. Rose. Informationslogistik in der OP-Planung. In *Drittes Kolloquium des DFG-SPP "Intelligente Softwareagenten und betriebswirtschaftliche Anwendungsszenarien"*, Hameln, Germany, 2001.

[18] K. Lang. *Gestaltung von Geschäftsprozessen mit Referenzprozeßbausteinen*. Deutscher Universitätsverlag, Gabler Edition Wissenschaft, Wiesbaden, 1997.

[19] J. Lind. *Massive: Software Engineering for Multiagent Systems*. PhD thesis, Universität des Saarlandes, Saarbrücken, 2000.

[20] G. Peter, C. Rupprecht, and T. Rose. Towards Reducing the Complexity of Process Modeling by Advisors, Explicit Context Modeling and Visualization Techniques. In Philippe Lenca, editor, *Proc. of the 10th Mini EURO Conference on Human Centered Processes (HCP '99)*, pages 315–320, Brest, France, 1999.

[21] M. Raphael and S.A. DeLoach. A Knowledge Base for Knowledge-Based Multiagent System Construction. In *National Aerospace and Electronics Conference (NAECON)*, Dayton, OH, 2000.

[22] T. Rose. Visual Assessment of Engineering Processes in Virtual Enterprises. *Communications of the ACM*, 41(12):45–52, 1998.

[23] T. Rose, H. Knublauch, and G. Peinel. Agenten in der pro-aktiven Prozessunterstützung. *Industrie Management*, 4:50–53, 2000.

[24] C. Rupprecht, M. Fünffinger, H. Knublauch, and T. Rose. Capture and Dissemination of Experience about the Construction of Engineering Processes. In *Proc. of the 12th Conference on Advanced Information Systems Engineering (CAISE)*, Stockholm, Sweden, 2000.

[25] K. Sycara, K. Decker, A. Pannu, M. Williamson, and D. Zeng. Distributed Intelligent Agents. *IEEE Expert*, 11(6), 1996.

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