Improving Goal and Role Oriented Analysis for Agent Based Systems

Pei Pei Kuan, Shanika Rarunasekera, Leon Sterling
Department of Computer Science and Software Engineering
The University of Melbourne, Parkville, Australia 3052
ppk@students.cs.mu.oz.au
shanika@cs.mu.oz.au
leon@cs.mu.oz.au

Abstract

The separation between analysis and design phases has long been advocated in software engineering literature. There has been active interest in the the area of agent oriented software engineering but the methodologies developed do not focus on a clear separation between the two phases. Furthermore, existing agent oriented methodologies tend to be tied to a particular design architecture and applicable only for small systems. In this paper, we describe a goal and role based analysis methodology that is both unbiased towards any design architecture and is scalable. The model is derived from improvements to the ROADMAP methodology for agent onented systems developed at the University of Melbourne. We also present ***REBEL*** *- a* ***CASE tool*** *developed to support the methodology. Furthermore, several examples and experiences with the method are discussed We conclude by comparing analysis models of other agent oriented methodologies to ours.*

Keywords: *Agent oriented software engineering, role oriented analysis, goal oriented analysis, methodology*

1. Introduction

Software analysis is a crucial phase in the software development lifecycle. During analysis, the 'what' as oppose to the 'how' of the system is described [10]. It is during this phase that the basis for understanding the problem domain is established. Analysis models then serve as the common communication medium between two often disparate groups - domain experts and software developers. Besides these two groups, clients would also be interested in the analysis models. Following analysis, the 'how' of the system can be described during the design phase.

Agent-oriented software engineering (AOSE) has become a widely researched area in recent years. Many methodologies for supporting the development of multi-agent systems have been proposed in the literature [3], [8], [7], [1], [12]. Many of these methodologies introduce analysis and design models for multi-agent systems. However, they do not draw a clear separation between the detail that should go into analysis and design models. Most of these models have too much design details in the analysis models and thereby, taking away the flexibility of making important design decisions during the design phase. Furthermore, by not providing a clear separation between analysis and design, there methodologies have in effect impaired the communication channel between domain experts and software developers. It also makes the models difficult to understand for the clients.

Another major drawback of the existing AOSE methodologies is that they are tied to a particular agent architecture from the early phases, and therefore are too constrained. Having analysis models which are agent architecture independent will make the models more understandable to domain experts. It will also give the designers the choice of choosing an appropriate architecture once the requirements are properly analysed. Scalability is also an issue that needs to be considered since agent systems are typically large systems.

In our previous work we presented ROADMAP - a methodology for modeling open, adaptive intelligent agent systems. Since then we have applied the methodology to develop many multi-agent systems both in academia and industry. Based on these experiences we have refined our methodology, especially the analysis models, to meet the challenges described above and elaborated on in the next section. In this paper, we describe the analysis models of ROADMAP and demonstrate how they provide a clear separation between analysis and design. We also present the Roadmap Editor Built for Easy deveLopment (REBEL), a CASE tool we developed to aid usage of ROADMAP's analysis models.

The paper is organised into several sections. Section 2

Proceedings of the 2005 Australian Software Engineering Conference (ASWEC'05)
1530-0803/05 $20.00 © 2005 IEEE

describes the criteria that our analysis methodology should meet. Section 3 provides an overview of the ROADMAP methodology. Section 4 explains the Goal Model. Section 5 explains the Role Model. Section 6 explains the Social Model. Section 7 presents REBEL which is a tool developed to support our methodology. Section 8 discusses some examples of how the methodology has been used. Section 9 compares our analysis methodology with other agent oriented methodologies. Section 10 concludes the paper.

2. Criteria for the Analysis Methodology

In this section, we discuss the criteria that motivated us to devise the methodology. The criteria are design independence, scalability and ease of use.

2.1. Design Independence

By design independence, we mean that the methodology should provide a clear separation between analysis and design. In other words, the analysis model should not be tied to any particular design architecture. The design architecture should be decided during the design stage and not at onset of analysis.

Software analysis should be performed at a high level of abstraction. It is important that the analysis model captures the high level requirements and quality constraints of the system. Only during design are the lower level details about architecture and implementation considered. A clear separation is important because the target audience for each phase is different. Domain experts are able to understand an overview of the system during analysis. They are not interested in technical details. On the other hand, technical details are important to software developers at design time. Besides, providing a separation between the two phases facilitates clear traceability of requirements.

2.2. Scalability

Multi-agent systems, especially open systems are inherently large systems managing different organizational aspects. Therefore, an important aspect of a methodology for modeling such systems is scalability. It is very unlikely that a single analysis model will be able to capture the requirements for the complete system. It is necessary to be able to have models at different levels of abstraction so that both domain experts and developers alike can get an idea of the overall system behaviour or focus on a particular part of the system in more detail if required.

23. Ease of Use/Understandability

The methodology should be easy to use and easy to learn. It should not be complicated by complex notations or formal specifications. For example, to use UML even in the analysis stage, one has to learn and understand myriad notational concepa such as triggers, preconditions, inheritance, etc. By meeting this criterion, the methodology makes it easy for domain experts to understand the models produced, thus simplifying the sharing of domain knowledge with developers. It should also be easy for clients to understand the models. It may even provide the additional benefit that the methodology is not restricted to agent oriented systems only.

3. ROADMAP Methodology Overview

The ROADMAP methodology [12] was first presented as an extension to GAIA [3]. However, feedback and experiences with the methodology then uncovered several limitations. Work on the methodology has since progressed through several evolutions ([13], [6], [14]) such that ROADMAP is now disparate from GAIA.

The analysis methodology we are presenting here is a part of the ROADMAP methodology. However, this paper is not meant to describe. all the models in the methodology, Here, we concentrate only on the analysis models. We described the Goal Model, Role Model and Social Model in the next few sections.

Figure 1 shows the models in the current ROADMAP methodology. The dotted horizontal line in the figure shows the separation between models in the analysis and design phase.

In the current ROADMAP methodology, the models are divided vertically into 3 areas: Domain Specific Models. Application Specific Models and Reusable Services Models. The Environment Model and Knowledge Model capture information about a specific domain and do not belong to a particular software development phase. On the other hand, Goal Model, Role Model, Agent Model and Interaction Model are tied to the system being modelled. Generic and reusable components in the system are captured by the Social Model and Service Model.

The models are also split horizontally according to the analysis and design phases. As discussed, it is important to keep implementation details out of the analysis models. The Goal Model, Role Model and Social Model are part of the analysis phase whereas Agent Model, Protocol Model and Service Model are part of the design phase.

All the models do not use formal notation. Feedback from teaching the methodology in the 433-682 Software Agents subject at the University of Melbourne and from

many presentations at workshops shows that the models are easy to understand.

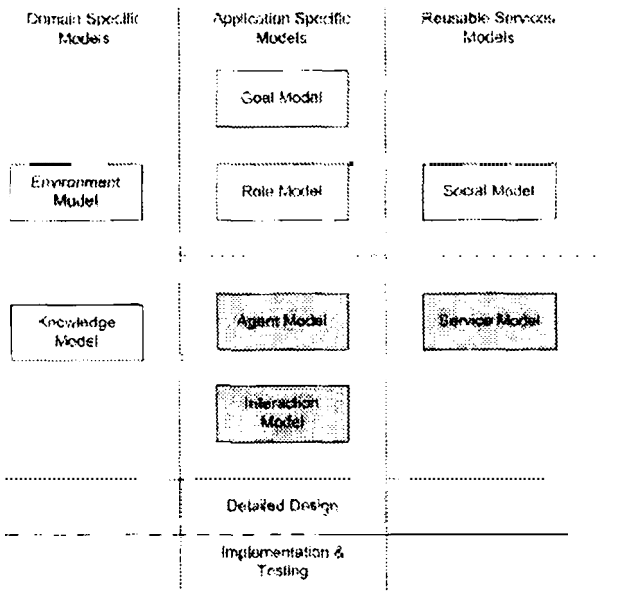

Figure 1. ROADMAP Models

4. The Goal Model

The Goal Model provides a high level overview of the system requirements. Its main objective is to enable both domain experts and developers to pinpoint &he*goals* (and *quality goals*) of the system and thus the separate *roles* the system needs to fulfill in order to meet those goals. Implementation details arc not described at all as they arc of no concern in our analysis stage.

The reader can simply think of the Goal Model as a container of three components: Goals, Quality Goals and Roles. A Goal is a representation of a functional requirement of the system. A Quality Goal, as its name implies, is a nonfunctional or quality requirement of the system. We use the term ***quality*** *goal* as opposed to ***soft goal*** because the former comes from a software engineering mindset whereas the latter is from an artificial intelligence mindset. A Role is some sort of capacity or position that the system requires in order to achieve its Goals. Later on in the design stage, a Role may well be decomposed into one or more agents.

A Goal, Quality Goal or Role may appear in more than one Goal Model. This simplifies the analysis of large system because it can be decomposed into multiple Goal Models. The Goal Model does not require its constituent components *to* be unique across the system.

Goals and Quality Goals can be further decomposed into smaller related sub-Goals and sub-Quality Goals. This seem to imply some hierachical structure between the Goal/Quality Goal and its sub-Goals/sub-Quality Goals. However, this is by no means an "is-a" relationship as is common in OO methodologies. Rather the hierachical structure is just to show that the sub-component is an aspect of the top level component.

On the other hand, Roles do not have such a hierachy. This is because the Roles required in the system are determined from the Goals. Each Goal is analysed to acertain the type of Roles needed to fulfill it. As such, Roles can be considered *to* be *atomic* though they may share common Goals and Quality Goals. Who or rather which human or software agent fulfills a particular Role is not a concern at this stage. An agent may also play the part of more than one Role.

As an example, let us consider modelling a university. For simplicity, let us assume that this university has only one Goal which is to provide tertiary education. This Goal can be decomposed into two sub-Goals - to conduct lectures and to conduct tutorials. To accomplish these, it requires the Lecturer and Tutor positions i.e. these are the separate Roles it requires. The university must also provide quality courses if it wants to be regarded highly by potential students. Factors that affect the quality of a course are for example its practical value, its cost and whether or not it has professional accreditation. As such, the university has a top level Quality Goal which in turn is related to 3 sub-Quality Goals.

The Goal Model is presented in a graphical diagram and no formal notations is used. A summary of the notations used is shown in Figure 2 whereas Figure 3 portrays the university model example diagrammatically.

Notation	Representation/Meaning
▱	Goal
⌓	Quality Goal
웃	Role
←	To show relationship between the Goal/Quality Goal with its related sub-Goals/sub-Quality Goals
—	To show connection between elements

Figure 2. Summary of Notations

We clearly do not utilisc a use case-like approach as do some other methodologies. The inital version of ROADMAP [12] utilised use cases which we later found to be a disadvantage. This is because use cases impose some limitations when the system is a multi-agent

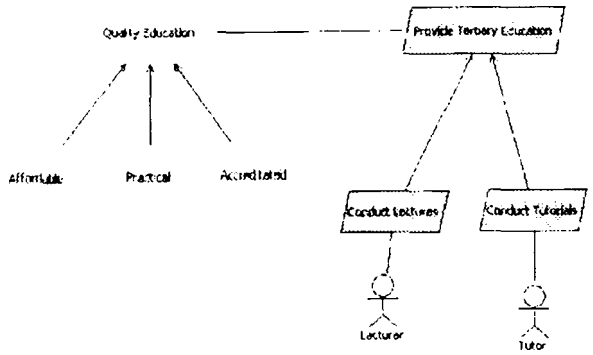

Figure 3. An Example Goal Model for University System

one. Firstly, use cases inherently impose a system boundary. This is unsuitable for an open system where any number of agents may enter or leave. Secondly, the approach places a strong focus on user actions which can only give a restricted view of the system. Agent systems are unlike traditional system. Agents potentially are responsible for every aspect of the system and not just an abstraction of the users' point of view.

5. The Role Model

The Role Model describes the properties of a Role. The term Role Schema is used interchangably with Role Model, The Role Model consists of four elements to describe the Role:

Role Name A name identifying the Role.

Description A textual description of the Role.

Responsibilities A list of duties/tasks that the Role must perform in order for a set of Goals and/or Quality Goals to be achieved.

Constraints A list of conditions that the Role must take into consideration when performing its responsibilities. These include guard conditions, safety conditions and quality conditions.

Clearly, this is analogous to the delegation of work through the creation of positions in a human organisation. Every employee in the organisation holds a particular position in order to realise some business functions. Different positions entails different degree of autonomy, decision making and responsibilities. Taking this analogy, a Role Schema is the "position description" for a particular Role.

Table 1 shows an example Role Model/Role Schema.

Role Name	Lecturer
Description	Teaches one or more courses,
Responsibilities	Make preparations for lectures such as slides, notes. etc.
	Present lectures.
	Provide consultation hours.
	Decide on course content.
Constraints	Lectures must not go on for too long, i.e. it fits in X hour as per the timetable.
	Delivery style should not bore the audience.
	Consultation hours must be suitable and accessible - e.g. not 7am when its too early for most students, it does not clash with other commitments, etc.
	Course content must respect requirements of relevant accreditation bodies.

Table 1. Role schema for Lecturer - an example

Role Models (and hence Roles) can be merged or splitted. This is useful because Roles can be refined as more understanding of the domain space is obtained during analysis. For example, a more complex Lecturer Role than in Table 1 can be divided into the Educator Role and Researcher Role. The Educator Role deals with the teaching responsibilites of the Lecturer whereas the Researcher Role ensapsulates his research activities. Conversely, if the analyst had started with the Educator Role and Researcher Role, he may amalgate them into the Lecturer Role.

6. The Social Model

The Social Model captures the relations between the Roles in the system and policies for interaction. Policies for interaction between the Roles encompasses areas such as security, privacy and communication.

In a multi-agent system, it is common to have team structures. The team interaction then is a type of relation among the Roles participating in the team. Another basic relation is a reporting relation similar to a human organisation where

you have junior employees reporting to senior employees. In our university example, Tutor X may plan and discusss his/her tutorial plan with other Tutors. Every fortnight, Tutor **X** has to meet with Lecturer Y to report on his team's plan or suggestions for the next scheduled tutorial.

The model is open to all possible types of relations depending on the type of system. To determined the possible relations. analysts will need to ask probing questions about how a particular Role will fit in the system and how it will interact with its environment. Research on the Social Model is still evolving. Nonetheless, our aim is to achieve a level whereby common relations that exist in an agent-system can be easily described and yet understandable by domain **experts.**

7. REBEL Tool

The Roadmap Editor Built for Easy deveLopment (REBEL) is a tool **for** building Goal Models and Role Models (or Role Schemas) during the analysis stage. In the future it will be extended to support later stages of the software development life cycle.

REBEL has been developed in Java as an Eclipse plugin. Eclipse was chosen because of the following reasons:

- Its extensible plugin architecture [9] allows easy extension to our plugin in the future. It also allows reuse of functionalities provided by other plugins.
- It enables the developer to concentrate on the plugin without worrying about IDE capabilities such as file management, team sharing, common views, etc.
- Eclipse's Graphical Editing Framework (GEF) and Eclipse Modeling Framework (EMF) technologies can be utilised (as discussed further in Section 7.2).
- It has strong community support plus its API **documentation** is well maintained.

7.1. Features

The Goal Models and Role Schema5 created are part of an application model of the system to be built. This application model persists in a file on the file system. **A** Roadmap Project is a folder for such application model files. In REBEL, the user *can* create Roadmap Projects and then create application model files **for** the relevant Project.

REBEL has **3** main views - Navigator, Model Outline and Graphical views. **The** Navigator view **is** a standard Eclipse view that shows the files in **a** particular project. **The** Model Outline view provides a tree view of the persisted application model. This view allows the user to save the model. add another Goal Model/Role Schema and delete an existing Goal Model/Role Schema. Saving the application model is also done **through** the Model Outline view. The Graphical view provides graphical editing capabilities. It has a palette to drag and drop elements onto the drawing area. It provides functionalities such as move, resize, delete and grouping. Double clicking on an element in the drawing area will enable the user to edit the element's properties, if applicable.

Figure 4 and 5 show some scrcenshots of REBEL. In Figure **4,** the top left panel is the Navigator view, the bottom left panel is the Model Outline view and the main panel is the Graphical view.

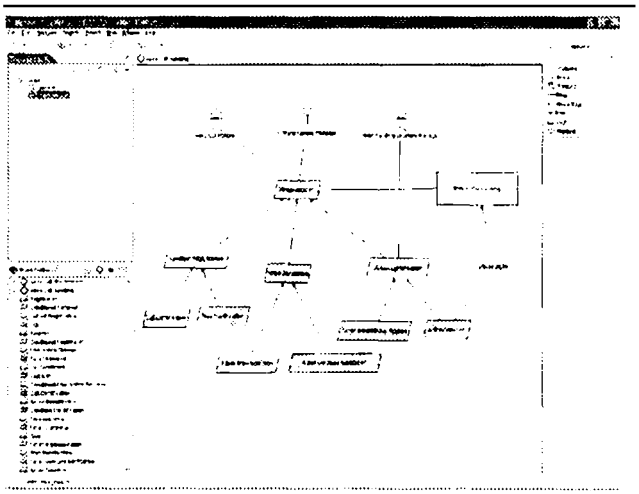

Figure 4. REBEL Navigator, Model Outline and Graphical Views

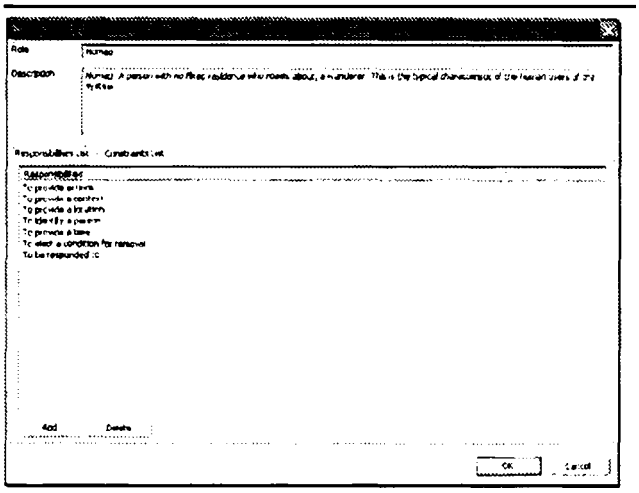

Figure 5. Editing a Role Schema with REBEL

The application model file is actually an XML Meta-

data Interchange (XMI) [18] compliant representation of the model. Saving it thus enables the analysis model to be vendor neutral and simplifies transformation of the model (e.g. for presentation purposes).

7.2. Architecture

REBEL consists of 4 main parts as shown in Figure 6.

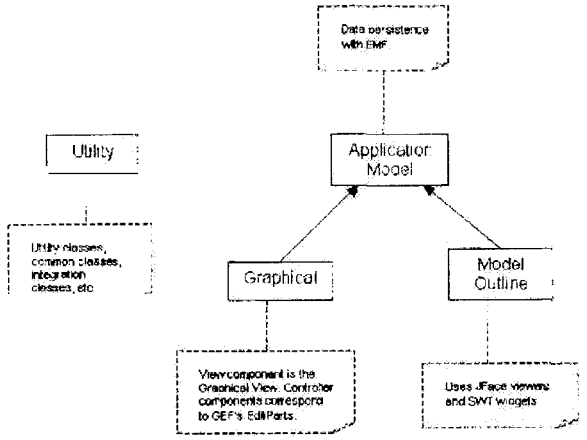

Figure 6. REBEL Design Architecture

The Utility component consists mainly of classes needed to integrate into the Eclipse IDE and utility classes for creating files, project, etc.

The rest of the components - Application Model, Graphical and Model Outline implements the MVC architecture. The Application Model is the model in the architecture which is used by the views in the Graphical and Model Outline components. The Graphical component utilises Eclipse's Graphical Editing Framework (GEF) for its graphical capabilities. Its View and Controller parts corresponds to those required by GEF as described in [4], [17] (Chapters 3 and 4). The Model Outline view uses JFace Viewers and SWT widgets for its implementation. [5], [11] describe the JFace and SWT architecture.

As mentioned before, the Application Model is saved as an XMI model. To be more precise it uses Eclipse's implementation of XMI - the Ecore model [2] (Chapter 5), [16]. The Eclipse Modelling Framework (EMF) [2], [15] was used to auto-generate skeleton classes from a UML class diagram which represented the meta-model for the Goal Model and Role Model. The main benefit of using EMF was that it enabled the developer to model the component at a very high-level. The generated classes were later modified and customised as required.

8. Examples and Experiences

The methodology has been used in several different instances by people with varying degree of exposure and experience with agent-oriented paradigms. Here is an overview of four of the applications:

Secure Identity Management (SIM) Project This was a project in collaboration with our industry partner, Adacel. An employee of Adacel applied the methodology to model a multi-agent system. The employee has a strong industry background in object-oriented concepts but none at all with agent-oriented ones. The employee was easily able to use the methodology and it helped him with understanding agent concepts. Examples of some of the Coal Models developed (reproduced with REBEL) are show in Figures 7 and 8.

440 Project - Intelligent Lifestyle This is a fourth year software engineering project. The team of 14 undergraduate students used the methodology to analyse requirements for building a demonstration of intelligent agents in a smart home environment. All the students had not taken any agent-oriented courses prior to this project.

Teaching Experience The methodology was taught as part of a masters level subject: 433-682 Software Agents at the University of Melbourne. Students were exposed to agent-oriented concepts and methodologies. The students also used the ROADMAP methodology to analyse example systems as part of their assessment.

Real Estate Multi Agent Systems (REMAS) A post-graduate student used the methodology to develop Goal Model diagrams, Role Models and Social Relationship Models to model an intelligent real estate system. This student has knowledge of agent concepts and had also taken the 433-682 subject. This project is now progressing to the design phase.

All the examples mentioned, except for the Intelligent Lifestyle project, were carried out in the first 6-9 months of 2004. The 440 Project will progress till the end of the year. Feedback from the participants in these examples was valuable *to* the continuous improvement of the methodology. Of particular note was the fact that the methodology was used favourably both by people who had prior knowledge about agent-oriented concepts and those who do not. Furthennore. the recency of these examples gives further assurance that it is applicable for developing systems in the present and near future.

9. Comparison to Similar Methodologies

In this section, we compare our analysis methodology with other agent oriented methodologies. We will consider

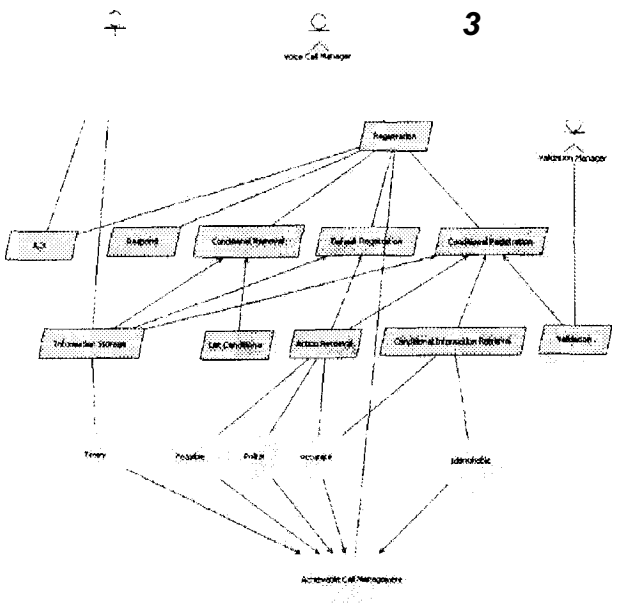

Figure 7. Voice Call Management Goal Model Diagram - from SIM Project (reproduced with REBEL)

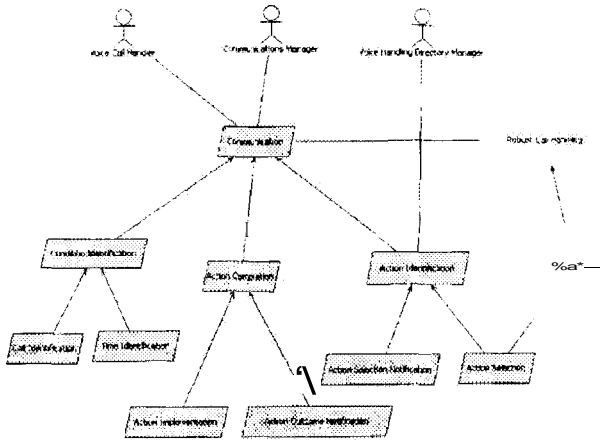

Figure 8. Voice Call Handling Goal Model Diagram - from SIM Project (reproduced with REBEL)

GAIA [3], Tropos [8], Prometheus [7] and Message/UML [1] in our discussion.

The GAIA methodoloy provides a complete analysis and design framework,It is architecturally independent but does not have a clear separation between analysis and design, Its analysis phase has design decisions and computational information. For example, its role model has "Liveness" and "Safety" properties that require low level information that may not be known at analysis phase. The notations used in its analysis models are unfamiliar for developers and hard for domain experts to understand.

Tropos on the other has good support for early requirement analysis. However, it focuses on the BDI architecture and uses BDI notions in its analysis phase. Knowledge level concepts, such as goals, plans, capabilities. beliefs, etc. are used throughout all phases of development. It requires the system under construction to be shown as a actor in an actor diagram in its Late Requirements phase. The goals and plans of the system are subsequently identified. However. *this* is not suitable when there are no identifiable domain stakeholders to serve as the main actor or for the design of reactive agent systems. Furthermore, the actor diagrams used in Tropos may not scale. This is because the actor diagram has to depict the entire network of dependency relationships between the system-to-be and other actors, goals and soft goals.

Prometheus also is not design independent. It specifically supports the development of BDI agents and is strongly tied to the JACK agent language. While the functionalities specified in its analysis phase (termed the system specification phase) are similar to Goals in our methodology, Prometheus does not have any concept similar to Quality Goals. Moreover, Prometheus employs use case scenarios to capture information about the system in operation. These scenarios only capture sequential system operations **and** are unable to provide a high level system overview at analysis phase. An overview of the system is provided by Prometheus but only during design. Even then, their system overview diagram is not scalable as all information has to be created in a single diagram.

The Message/UML methodology is an extension of UML to include agent concepts. It is implementation unbiased. During analysis, different views of the system are decomposed **and** refined. It lets the analyst have the flexibility of choosing appropriate views and refinement strategies. While this is acceptable for experience users, novice users may have difficulty deciding on a view. Certainly, **a** guideline **of** situations where a particular view is more applicable would help ease the learning process. Its heavy use of UML notations also complicates the analysis process because it impedes understanding by domain experts.

Our analysis methodology does not have the drawbacks of the other methodologies. It is clearly not biased toward any specific design or implementation. No assumption has been made about the design architecture nor is one required to be chosen before the methodology can be applied. In the methodology, Goals, Quality Goals and Roles can be re-

peated in different Goal Models. Roles can also be merged or splitted. Furthermore, the Social Model **does** not need to be portrayed in the Goal Model diagram (as Tropos' dependency model does in its actor diagram). These ensures that our diagrams can be used to depict large systems thus meeting the scalability criteria. We have also introduced **a** relatively small set of notations **and** concepts. This make it easy to learn and use both by **domain** experts and software developers.

10. Conclusion

In this paper, we presented the ROADMAP methodology for improving role and goal oriented analysis of multi-agent systems. We discussed how the methodology is able to fulfill the three criteria of design independence, scalability and ease of use. In fact, the methodology can also be used on object-oriented systems. Future **work** will focus on maturing the other models so that ROADMAP can support the entire software lifecycle. We also plan to develop REBEL further as the inethodology matures.

11. Acknowledgement

We would like to thank Andrew Seiberras for his feedback and experiences with using the methodology in the SIM project. Thanks also to Srutilekha Bhaltacahrya for her insights in the REMAS project and Thomas Juan for his work on the initial version of ROADMAP. We also thank the Smart Internet Technology CRC for supporting our work,

References

[1] G. Caire et al. Agent oriented analysis using message/UML. In ***Second International Workshop on Agent-Oriented Software Engineering (AOSE)***, pages 101–108, 2001.

[2] F.Budinsky et al. ***Eclipse Modeling Framework.*** Addison Wesley Professional, 2004.

[3] F.Zambonelli, N Jennings, and M.Wooldridge. Developing Multiagent Systems: The Gaia Methodology. ***ACM Transactions on Software Engineering and Methodology (TOSEM)***, 12(3):317–370, July 2003.

[4] Create an Eclipse-based application using the Graphical Editing Framework. http://www-106.ibm.com/developerworks/opensource/library/os-gef/, 3003.

[5] Eclipse **3.0** Platform Plug-in Development Guide. http://help.eclipse.org/help30/index.jsp, 2004.

[6] K.Chan and L.Sterling. Adapting Roles for Agent Oriented Software Engineering. ICSE, 2004.

[7] L.Padgham and M.Winikoff. Prometheus: **A** Methodology for Developing Intelligent Agents. In ***Proceedings of the 3rd International Workshop on Agent Oriented Software Engineering***, July 2002.

[8] P.Bresciani et al. Tropos: An Agent Oriented Development Methodology. ***Journal of Autonomous Agents and Multi-Agent Systems***, 8(3):203–236, May 2004.

[9] Notes on the Eclipse Plug-in Architecture. http://eclipse.org/articles/Article-Plug-in-architecture/plugin_architecture.html, 2003.

[10] R.Pressman. ***Software Engineering: A Practitioner's Approach***, page 176. McGraw-Hill, 2004.

[11] Getting Started with **Eclipse** and the SWT (Tutorials) . http://www.cs.urnanitoba.ca/ eclipse/, 2003.

[12] T.Juan, A.Pearce, and L.Sterling. ROADMAP:Extending the Gaia Methodology for Complex Open Systems. In ***Proceedings of the First International Joinr Conference on Autonomous Agents und Multi-Agent Systems (AAMAS)***, pages 3–10, July 2002.

[13] T.Juan and L.Sterling. The KOADMAP Meta-model for Intelligent Adaptive Multi-Agent Systems in Open Environment. In ***Proceedings of the 4th International Workshop on Agent Oriented Software Engineering,*** July 2003.

[14] T.Juan and L.Sterling. Achieving Dynamic Interfaces with Agent Concepts. In ***Proceedings of the 3rd International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS).*** July 2004.

[15] The Eclipse Modeling Framework (EMF1 Overview. `http://download.eclipse.org/tools/emf/saipts/dncs.php?doc=` references/overview/EMF.html, 2004.

[16] Using EMF. http://eclipse.org/articles/Article-Using%20EMF/using-emf.html, 2004.

[17] W.Moore et al. ***Eclipse Development using the Graphicul Editing Framework and the Eclipse Modeling Frumework.*** IBM Publications, 2004.

[18] XMI Metadata Interchange(XMI) Specification. http://www.omg.org/cgi-bin/doc?formal/2003-05-02, 2003.