An Agent-Oriented Software Engineering Methodology with Application of Information Gathering Systems for LCC

Tiemei Irene Zhang [1], Elizabeth Kendall [2], and Harvey Jiang [3]

[1] School of Network Computing, Monash University
McMahons Rd., Frankston, VIC. 3199, Australia
Irene.Zhang@infotech.monash.edu.au

[2] Faculty of Information Technology, Monash University
McMahons Rd., Frankston, VIC. 3199, Australia
Kendall@infotech.monash.edu.au

[3] Object Oriented Pty. Ltd., Level 11, 484 St. Kilda Rd.
Melbourne VIC. 3004, Australia
harveyj@oopl.com.au

Abstract. Life Cycle Cost (LCC) is a very important issue for organizations, which determines the success of their businesses. However, information gathering from a highly distributed heterogeneous environment with a huge number of information sources is an obstacle to the use of the existing LCC models. To overcome this obstacle, we took an agent-based information gathering system as a solution. In order to develop an agent-based system in a systematic way, we established a methodology of agent-oriented system engineering. Therefore, the system development follows a step by step process from the stage of system requirement to implementation. This paper presents this methodology and illustrates the processes of system analysis, design, and implementation by applying this methodology to information gathering for the CASA (Cost Analysis Strategy Assessment) model. The experimental results show that the agent technology is useful and beneficial for LCC information gathering.

1 Introduction

As cost is a key factor to determining the success of a product [6], manufacturers attempt to reduce costs during every phase of the product's life cycle [16]. They use life cycle cost (LCC) models [23] to estimate the life cycle costs of their products before making decisions. A typical model is CASA that is used in many organizations,

such as the U.S. Department of Defense. The CASA model covers the entire life of a product and employs some 82 algorithms with 190 variables. Similar to the other LCC models, CASA requires extensive information, manually gathered from different data sources in a highly distributed and heterogeneous environment. however, this approach cannot deal with the need to respond to global economic competition, because this environment involves unpredictable changes and uncertainty. Further, the Internet is changing today's business environment, and this also has a large impact on the problem.

Having reviewed the available literature, we determined that an agent-based information gathering system can potentially solve our problem. This is because an agent is autonomous, social, reactive and proactive [26], and it can also model policies and proactive behaviours [12]. In other words, agents provide high level communication and interaction, which can perceive the situation of the environment and respond appropriately. Agents are different from objects, which are static and cannot change with the environment. For example, to gather information, an agent will first search its own knowledge base. If the information is not available, it will be able to interact with the other agents. An agent stores the information in its knowledge base once it finds it. If there are multiple data sources, an agent will be able to gather information in an efficient manner. In contrast, it involves considerable difficulty and complexity to realize the above scenario with object technology. So agent technology provides perceived advantages over objects for solving our problem.

This paper aims to develop an agent-based system to gather information for the CASA model. It establishes a methodology for analyzing and designing agents, and then applies it to the CASA model [17]. We propose a conceptual model that takes Infosleuth [19] as a reference and also we use the BDI (Belief, Desire, and Intension) [20] agent as the agent architecture in our system.

2 Literature Review

A number of methodologies have been reported to address agent-oriented software engineering [24]. Wooldridge, Jennings and Kinny [27, 28] present the Gaia methodology for agent-oriented analysis and design. Gaia is a general methodology that supports both the micro-level (agent structure) and macro-level (agent society and organization structure) of agent development. It requires that the inter-agent relationships (organization) and agent abilities are known at run-time. Gaia includes analysis and design processes. Gaia's analysis process can find the roles in the system, and then it models interactions between the roles that have been found. The design process maps roles into agent types, and then creates the right number of agent instances of each type. Next, Gaia can be employed to determine the services needed to fulfill a role in one or several agents, and to the final step involves creating acquaintance models for the representation of communication between the agents. The Gaia methodology emphasizes a few models that can be utilised to form the whole system. It describes

what these models are, but the processes used to develop these models are vague. In Gaia a *role* is viewed to be one of the roles in an organization, and role identification itself is ad hoc.

Wood and DeLoach [5, 25] suggest the Multiagent Systems Engineering Methodology (MaSE). Similar to Gaia, MaSE respects to generality and the application domain supported, but in addition, MaSE goes further regarding support for automatic code creation. It includes seven sections of capturing goals, applying use cases, refining roles, creating agent classes, constructing conversations, assembling agent classes, and system design. The identified roles are driven by the capturing goals. The goal of MaSE is to lead the designer from the initial system specification to the implemented agent system. Domain restrictions of MaSE are similar to those of Gaia's, but in addition it requires that agent-interactions are one to one and not multicast

MAS-CommonKADS [9] is extends CommonKADS [21] for multi-agent systems. It starts with a conceptualization phase that is an informal phase for collecting the user requirements. This methodology defines the models for the analysis and design of the system, which includes the following models: agent, task, expertise, coordination, organization, communication, and design. Although MAS-Common KADS employs the notion of an agent's role, it does not formally define what this means. Also, the concepts of role and class are used interchangeably [13] as the roles that used to analyze and design agents actually are the attributes of an agent class. The distinction is important; a class stipulates the capabilities of an individual object, while a role focuses on the position and responsibilities of an entity in an overall structure or system. In particular, MAS-CommonKADS states that a CRC card describes an agent's class.

Although the above methodologies use the concept of a role to design agents, no formal techniques/representations, such as role models (role patterns), are used to identify roles. This may lead to inappropriate behavior to appear in the agents (to which roles are mapped) because roles must be clear and unambiguous to provide detailed descriptions. This paper describes a methodology for designing agents that is based on the use of role models.

3 Methodology

3.1 Overview

Object-oriented (OO) methodology with use cases has been widely used in software development, and use case analysis has proved to be useful and successful for requirement specification and analysis of OO systems. It is useful to investigate the use of OO methodologies in agent-oriented software engineering. However, an agent is autonomous, social, reactive and proactive [26], while an object does not possess these characteristics. Therefore, we cannot directly apply the OO methodology to agent-oriented software engineering. Current research in role models shows promising

results for software agent analysis and design [15]. Therefore, our methodology combines these two approaches to develop an agent-based information gathering system for product life cycle cost estimation.

To describe interactions between activities, an ICOM (input, control, output and mechanism) presentation (as shown in Figure 1) is used to clarify constraints and resources pertaining to an activity. This notation is adopted from the functional model of IDEF [2, 3] that has widely been used for modeling manufacturing process.

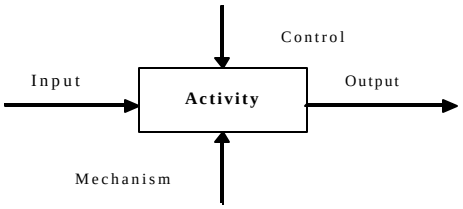

Fig. 1. An activity model with ICOM notation

We apply the ICOM representation to depict the processes involved in our methodology, as shown in Figure 2.

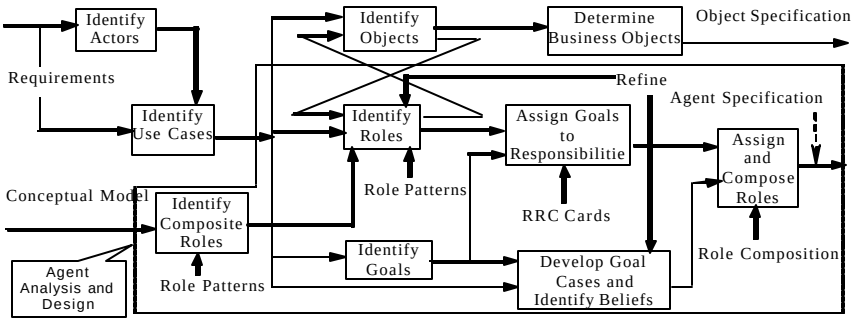

Fig. 2. Agent-oriented software engineering process

3.2 Object-Oriented Analysis

The object-oriented analysis depicted in Figure 2 consists of four activities: "Identify Actors", "Identify Use Cases", "Identify Objects" and "Determine Business Objects". These four activities use the traditional methods developed by [10] to identify actors, use cases, and objects. Note that an actor is a user who interacts with a system or an external system. A use case can be defined as a specific way of using the system by performing some part of the functionality, and it includes preconditions, flow of events, and post conditions. A use case model is closest to the requirements, and

with the least amount of detail [11]. Therefore, most business objects required by the system can be identified from the output of the "Identify Use Case" activity.

3.3 Goal Identification

The identified use cases are fed to the activity "Identify Goals" in Figure 2 for capturing goals. A goal is an objective or a desired state that can be achieved or ascertained by an agent. A goal identifies *what is to be done*, and it should change less often than more detailed processes/activities. This is because a process or activity identifies *how things are to be done*. Goals are important to agent-based systems because agents are autonomous and proactive. Agents achieve goals on the behalf of users through their autonomous and proactive behavior. To identify goals from the use cases, we should [14]:

- Identify the most top goal;
- Decompose it to the sub goals necessary to fulfill the top goal;
- Place the first set of sub goals as the first level goals;
- Identify the next level of goals in a similar mode;
- Place them as the second level goals;
- Derive all the goals in an iterative form;
- Stop when a goal cannot be structurally or temporally decomposed.

The result of goal identification is a goal hierarchy diagram where each level of goals fulfils the goal on the level above. The identification of goals is an iterative process because additional details may be uncovered and duplicated/unnecessary items may be deleted, modified, or combined.

3.4 Goal Case Development and Belief Identification

Having identified goals, we next define a collection of scenarios about the agent's interactions in terms of the corresponding goals. Each scenario is called a goal-based use case (in short, a goal case). This scenario describes a sequence of plans that handle events that the agent initiates. An agent can start a goal case when the corresponding goal is triggered. The use of goal cases also helps with traceability because they are developed according to goals that link to the system requirements. To specify the goal cases, the following steps are taken:

- Determine if the goal case is a reaction to an event or an outcome to be achieved.
- Determine what triggers the goal case.
- Elaborate the context condition of the goal case.
- Determine the activities that have to be carried out for the goal case to be satisfied
- Describe the conditions on the transitions.

- Determine the input data and output results.
- Determine the performance measures that need to be collected.

Goal cases are the core of agent specification. Once goal cases are developed, we can assign them to agents according to the goals that each agent has. In our system, the beliefs are the knowledge that agents have. When an agent wants to achieve a goal and carry out a set of goal cases, the agent should have the knowledge to support its actions or evaluate the results. For example, the costing formulas are the beliefs of the agent who does the cost estimation . The beliefs can be identified and extracted from the goal cases in terms of the knowledge and expertise that are the basis for performing some activities.

3.5 Role Identification

The activity "Identify Roles" in Figure 2 occurs after "Identify Use Cases". Here a role involves a set of activities which, taken together, carry out a particular responsibility or set of responsibilities. A role has the resources that are necessary for it to do its activities. Those resources might reside permanently with the role or be passed to it. Role names should be verbs in the gerund form. Roles can be identified from the relevant role models, where role models are patterns of interaction and collaboration. Many role models may appear in a given agent application. Because they are patterns, they can be used during analysis and design as conceptual and analytical models. The activity of identifying roles from the use cases is to [15]:

- Examine role patterns from the existing role patterns literature and documentation. Relevant role patterns can be used to identify or recognize types of interaction and collaboration.
- Partition goals to form roles if there are no relevant role patterns that the goal can be assigned to. This includes extracting the goals in a generic way and taking these as the responsibilities of the role. Also, the other roles that collaborate with this role can be taken as collaborators. (In the next section, we describe the relationships a role can have, along with responsibilities and collaborators.
- Determine all roles for the identified interactions and collaborations.

To document role models of agent systems, one important method is to use role responsibility and collaborations (RRC) [13] cards (refer to Table 1). These are used to specify responsibilities and collaborations of roles, especially in the early phase of agent-oriented software development.

Table 1. RRC card

Role model name		
Role type	**Responsibility**	**Collaborator**
Names of Roles	List all responsibilities	List all collaborators

3.6 Assigning Goals to Responsibilities

After identifying RRC cards, we should assign goals as responsibilities of roles by carrying out the activity "Assign Goals to Responsibilities". This assignment starts at the bottom of the goal hierarchy diagram. During this activity, a role is assigned goals that are related to its responsibilities. Once the goals are assigned to responsibilities, collaboration between the roles should be indicated. A RGC (Responsibility, Goal, Collaborator) card, as shown in Table 2, is used to document relationships between responsibilities, goals and collaborators for a role.

Table 2. RGC card

Role		
Responsibilities	**Goals**	**Collaborators**
List of responsibilities	List of goals	List of collaborators

3.7 Assigning and Composing Roles

To design agents, we have to assign and compose the roles identified according to the process in §3.5. When the roles are assigned and composed, their goals and collaborators are allocated to the agents. Note that goals and collaborators are obtained from the RGC card shown in Table 2 whilst the goal cases and the beliefs have been developed for each goal. The designated agents will carry out the roles in order to achieve the goals. All of the agents' actions are based on beliefs and are according to the goal cases. To assign and compose roles for an agent, we should:

- Assign and compose roles for agent design;
- Assign roles to agents with design quality in mind, where cohesion, low coupling, and minimum need for communication are essential;
- The goals form the expertise for the agents. Splitting and merging may be required.

The output of this activity is a GCB (Goals, Goal Cases, Collaborators, and Beliefs) card (refer to Table 3) that is used to document agents. This card can directly be taken to the implementation stages.

Table 3. Agent specification template

Agent Name:			
Goal	**Goal Case**	**Collaborator**	**Belief**
List all goals	List all goal cases	List all collaborators	List all beliefs

4 Agent Analysis

Before proceeding, we present a conceptual model for an agent-based system that focuses entirely on business problems [7]. Taking the InfoSleuth architecture [19] as a

reference, our model consists of six different layers that make the system more compartmentalized and modularized. Each layer provides a level of abstraction and certain services to the layer above it, while hiding the implementation of its services from the higher layer. Also, each layer passes both data and control information to its corresponding neighbors. Figure 3 depicts such a model for gathering information for a life cycle model.

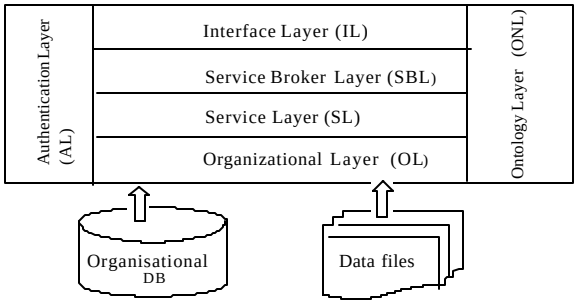

Fig.3. Conceptual model of information gathering system

This conceptual model covers many areas. First, an organization requires the ability to accurately identify a user who is making requests. In our system, the user's identity is verified by checking a password typed in during login. The process that verifies and records the user's identity is called authentication. This process is designed to employ an access-control-list that contains a single entry that is authorized to grant capabilities for other layers. In actuality, agents in every layer in an agent-based information gathering system have to get security clearance from the authentication layer before they can request services.

The ontology layer collectively maintains a knowledge base of the different terminology and concepts that are employed over the whole organization. This layer thus describes the language that will be used for specifying and translating requests for information.

The interface layer is used to predict the user's intentions and to request services that are provided by the remaining modules. This layer acts on behalf of users to relay specifications and obtain results. The broker layer models and delegates the services of the overall organization and then provides them to users via the interface layer. The service layer is used to provide services, which differs from the organizational layer that controls resources. The service layer represents and provides the high level services that can be formed by encoding expertise and by utilizing the organizational layer. The organizational layer can be used to manage the organizational resources. Its main task is to gather data from the various sources.

We then need to examine role models to determine their relevance and applicability to the conceptual model. We concentrate on the organizational layer here, and these roles, which we term organizational roles, are responsible for controlling resources.

We considered the following role models: Master/Slave [1], Manager [22], Bodyguard [18], and Adapter [8]. An organizational role can be another form of an Adapter, as it can reformulate requests to the different databases. However, the organizational role is also responsible for database activation and the supervision of any security restrictions. The activation behavior resembles that of a Manager, while the security supervision facets of this role resemble those found in the Bodyguard role model. When an organizational role acts as a Bodyguard or an Adapter, the database is the Target and the Subject. These roles are summarized in Table 4 [29].

Table 4. Organizational roles

Organization Role		
Role type	**Responsibilities**	**Collaborators**
Manager (Manager)	• To manage the information	resource role
Slave (Master/Slave)	• To receive the request from Master • To perform a task and send the reply to Master	service role
Client (Bodyguard)	• To request the permission of a service	authentication role
Subject (Bodyguard)	• To accept the notification of a service	authentication role
Client (Adapter)	• To send message to Adapter and collaborate with the Target	ontology role
Target (Adapter)	• To receive the message sent by Client • To perform a task and send a reply	ontology role

The roles in the other layers can be identified in a similar manner. However, in the following sections, we will only illustrate the application of our methodology for the organizational layer.

5 Agent Design

5.1 Basic Requirements of the CASA Model

The CASA model estimates costs throughout all life cycle phases, including maintenance. The maintenance of a product involves plans, labor, equipment, material, spare parts, transportation, recurring facilities, recurring item management, software maintenance, contractor services and engineering changes. The relevant information is stored in organization databases. To estimate life cycle costs with the CASA model,

this information must be gathered. We will focus on cost estimation for maintenance and give a brief description for a use case as below:

The user formulates a request for maintaining a product. When receiving the request, the maintainer will ask a planner for a maintenance plan, including schedules and actions. He/she then asks an estimator to estimate costs for all components of the product. To estimate costs, the estimator will gather the information required by the CASA model from organizational databases. This information is related to plans, labor, equipment, material, spare parts, and management, such as transportation, recurring facilities, recurring item management, software maintenance, contractor services and engineering changes. Particularly, if spare part information is not available in the organizational databases, the estimator will ask suppliers to provide it.

5.2 Goals, Goal Cases, and Beliefs.

We can identify the goals for information gathering from section 5.1. Figure 4 shows a goal diagram that represents the goals of the system in hierarchical structure.

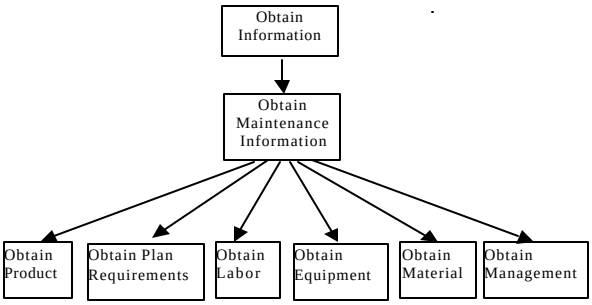

Fig. 4. Goals for searching for maintenance information

There are three levels in this figure. The top level contains "Obtain Information" that is a general goal for obtaining information. The second level is "Obtain Maintenance information", and this is a goal for obtaining information required for maintenance cost estimation. There are six goals in the third level. These goals can be used to search information specific to the CASA model.

To illustrate how to develop a goal case, consider the "Obtain Material" goal as an example. As we know, material information can be obtained from three data sources: organizational databases, suppliers' catalogues and user experiences. Therefore, the "Obtain Material" goal can be achieved by using a goal case described below:

Obtain Material goal case (GC1)
Pre-condition:
The product information is available
Flow of events

Basic paths:

1. The goal case starts when the agent is requested to achieve the "Obtain Material" goal.
2. The agent attempts to achieve the "Manage Material" goal for information from an organizational database.
3. If it cannot find data from organizational database, the agent attempts to achieve the "Manage Supplier" goal.
4. If it cannot find data from suppliers or any error occurs, the agent asks user to enter data by posting the "Manager Manual Entry" goal.
5. The agent replies to the requested agent with the material data.

Post-condition:

The agent stores the material information.

We can identify the belief "Material" from this goal case, which is the knowledge used to describe or present the material information.

5.3 Roles and Goal Assignment

The organizational roles in Table 4 are responsible for gathering information that is needed by algorithms in the CASA model. This information includes the product to be maintained, the plan and requirements to be implemented, labor and equipment to be used, materials to be consumed and spare parts that need to be purchased from suppliers. In addition, CASA also requires management information involving transportation, recurring facilities, recurring item management, software maintenance, contractor services and engineering changes. Therefore, it is practical that each individual agent that is assigned a role is responsible for gathering information in its specialized field. As a result, the organizational roles can be instantiated in our application as shown as in Table 5. After that we assign the goals to roles by using RRC cards. However, this procedure is not discussed in this paper.

Table 5. Instances of the organizational role

Composite Role	Instantiated Roles	Goal
Organizational Role	Labour manager	Obtain Labour
	Equipment manager	Obtain Equipment
	Material manager	Obtain Material Manage Material Manage Supplier Manage Manual Entry
	Management manager	Obtain Management
	Project manager	Obtain Product
	Plan & Requirement manager	Obtain Plan & Requirement

5.4 Agent Specification

According to Table 5 and the goals that we have identified, we assign and compose roles to agents, as shown as in Table 6. It is worth mentioning that this approach allows organizations to create their new assignments when organizational changes occur.

Table 6. Organizational Agent identification

Composite Roles	Instantiated Roles	Potential agents used
Organizational Role	Project manager, Plan & requirement manager	Project agent
	Labor manager Equipment manager	Resource agent
	Material manager Supplier manager. Management manager	Inventory agent Management agent

To document agents, consider the Inventory agent as an example. Table 7 shows a GCB card for specifying this agent. The rest of agents can be documented in a similar manner.

Table 7. Simplified Inventory Agent specification

Inventory Agent			
Goal	**Goal Case**	**Collaborator**	**Belief**
Obtain Material	GC1	Estimator	Material
Manage Material	GC2*	Organization Database	Material
Manage Supplier	GC3*	Supplier Database	Material
Manage Manual Entry	GC4*	User	Material

* Goal cases are not specified in this paper

6 Experimental Evaluation

To realize our system, we used the JACK framework [4] that provides four main class-level constructs: Agent, Database, Event, and Plan for a BDI agent. Note that the Agent construct includes what type of messages and events an agent responds to, and which plan it uses to achieve its goals. It not only has methods and data members just like objects, but it also contains database relations that an agent can use to store beliefs, descriptions of events that the agent can handle, and plans that the agent uses

to handle the events. Table 8 shows rules that can be used to map a GCB card to JACK constructs

Table 8. Mapping from a GCB card to JACK agent

GCB card	Goal	Goal case	Collaborator	Belief
JACK agent constructs	Event	Plan	Agent	Database

By applying the above mapping, we have implemented our agents to gather information for the CASA model. Figure 5 shows a screen shot from our system that is able to estimate a life cycle cost for a computer system.

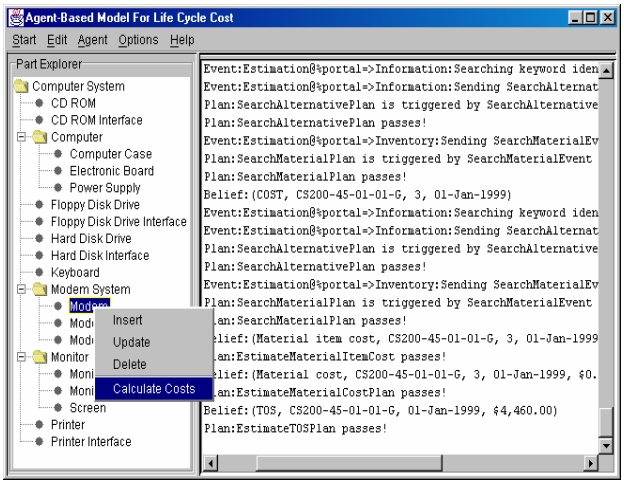

Fig. 5. Results

In this figure, the left side pane shows a tree structure that represents the product of computer system in the form of assembly. For estimating cost for the whole product or a subsystem, just click the manual. The right side pane lists the results, which describes the event, plan and agent tasks in time order.

7 Summary

In this paper, we have presented a methodology of AOSE for identification of goals and goal cases based on an organizational view and roles. We have applied this methodology to an information gathering system for LCC. This methodology is a systematic approach that uses goals and roles. It generates results from the initial system requirement to the implemented agent-based system. Furthermore, this methodology can be applied to other agent-based information systems.

8 References

[1] Aridor, Y., and Lange, D.B.: Agent Design Patters: Elements of Agent Application Design. Autonomous Agents (Agents'98), Minneapolis, (1998), 108-115

[2] Bravoco, R. R., and Yadav, S. B.: Requirements Definition Architecture –An Overview. Computers in Industry, 6, (1985), 237-251.

[3] Bravoco, R. R., and Yadav S. B.: A Methodology to Model the Functional Structure of an Organisation. Computers in Industry, 6, (1985), 345-361.

[4] Busetta, P., Ronnquist, R., Hodgson, A., and Lucas, A.: Light-Weight Intelligent Software Agents in Simulation. SimTech 99, Melbourne, Australia, (1999).

[5] DeLoach S. A.: Multiagent Systems Engineering A Methodology and Language for Designing agent Systems. In Proceedings of Agent Oriented Information Systems, (1999), 45-57.

[6] Fabrycky, W.J., and Blanchard B.: Life-Cycle Cost and Economic Analysis. Prentice-Hall, Inc., New Jersey, USA, (1991).

[7] Flower, M.: Analysis Patterns – Reusable Object Models, Addison Wesley, (1997).

[8] Gamma, E. R., Helm, R., Johnson, R., and Vlissides, J.: Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley, (1994), 139-150.

[9] Iglesias, C. A., Garijo, M., González, J. C., and Velasco, J. R.: Analysis and design of multiagent systems using MAS-CommonKADS. In: Singh, M. P, Rao, A., and Wooldridge, M. J. (eds.): Intelligent Agents IV (LNAI volume 1365). Springer-Verlag: Berlin Germany, (1998), 313-326.

[10] Jacobson, I., Christerson M., and Jonsson P., and Overgaard J.: Object-Oriented Software Engineering – A Use Case Driven Approach, Addison-Wesley, (1992).

[11] Jacobson, I., Griss M., and Jonsson P. Software Reuse - Architecture. Process and Organization for Business Success, ACM Press, (1997).

[12] Kendall, E.A., Malkoun, M. and Jiang C.: A Methodology for Developing Agent Based Systems for Enterprise Integration. EI'95, IFIP TC5 SIG Working Conference on Modeling and Methodologies for Enterprise Integration. Heron Island, Queensland, Australia, (1995).

[13] Kendall, E. A.: Agent Roles and Role Models: New Abstractions for Multi-agent System Analysis and Design. International Workshop on Intelligent Agents in Information and Process Management. German Conference on Artificial Intelligence, Bremen, Germany, September, (1998)..

[14] Kendall, E. A., Krishna M., Pathak C. V., and Suresh C. B.: Patterns of Intelligent and Mobile Agents. Agents '98, May, (1998).

[15] Kendall, E. A.: Role models, aspect oriented programming and agent engineering. Technical report, British Telecom, (1999).

[16] Li, Y., Huang B., and Wu C.: Virtual Enterprise Information System. In Proceedings of the 1st Asia-Pacific Conference on IAT' (Intelligent Agent Technology), (1999), 493-497.

[17] Manary, J.M .: DSMC's CASA Model Still Going Strong. Article in PM: Jan.-Feb., (1996).

[18] Neves, F.D., Garrido, A.: Bodyguard. In Marting, R, Riehle, D, Buschmann, F (eds.): Pattern Languages of Program Design 3. Addison Wesley, (1998), 231-243.

[19] Nodine, M., Perry B., and Unruh A.: Experience with the InfoSleuth Agent Architecture in Proceedings of AAAI-98 Workshop on Software Tools for Developing Agents, (1998).

[20] Rao, A. S., and Georgeff M. P.: BDI Agents: From Theory to Practice. In Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95), San Francisco, USA, June, (1995).

[21] Schreiber, A., Wielinga, B. J., Akkermans, J. M., and Van de Velde W.: CommonKADS: A comprehensive methodology for KBS development. Deliverabel DM1.2a KADS-II/M1RR/UvA/70/1.1, University of Amsterdam, Netherlands Energy Research Foundation ECN and Free University of Brusels, (1994).

[22] Sommerlad, P.: Manager. In Marting, R. Riehle, D., Buschmann, F. (eds.): Pattern Languages of Program Design 3. Addison Wesley, (1998), 19-28

[23] Sterling, J.C., Analysis of Life Cycle Cost Models for DOD & Industry Use in 'Design-to-LCC', (1996), http://nissd.com/sdes/papers/deslcc.htr.

[24] Tveit, A.: A survey of Agent-Oriented Software Engineering. NTNU Computer Science Graduate Student Conference. Norwegian University of Science and Technology, May (2001).

[25] Wood M. F. and DeLoach S. A.: An Overview of the Multiagent Systems Engineering Methodology. The First International Workshop on Agent-Oriented software Engineering (AOSE-2000), (2000).

[26] Wooldridge, M., and Jennings, N. R.: Intelligent agents: theory and practice. The Knowledge Engineering Review, 10(2), (1995), 115-152.

[27] Wooldridge M. J., Jenning N. R. and Kinny D.: A methodology for agent-oriented analysis and design. In Proceedings of the third international conference on Autonomous agents, (1999), 69-76.

[28] Wooldridge M. J., Jennings N. R. and Kinny D.: The Gaia methodology for agent-oriented analysis and design. Autonomous Agents and Multi-Agent Systems, 3(3), September, (2000), 285-312.

[29] Zhang, T. I., Kendall, E. A., and Jiang H. C.: System Analysis of Agent based LCC Information Gathering. In Proceedings of the First Pacific Rim International Workshop on Intelligent Information Agents (PRIIA 2000), Melbourne, Australia, (2000).