Agent Design for LCC Information Gathering

T. I. Zhang
Computer Systems Engineering, Royal Melbourne Institute of Technology, Australia
Em: t.zhang@rmit.edu.au

H. C. Jiang
Object-Oriented PTY. LTD., Australia
Em: harveyj@oopl.com.au

E. A. Kendall
The School of Network Computing, Monash University, Australia
Em: kendall@infotech.monash.edu.au

Keywords Multi-agent system, Information gathering, Life cycle costing (LCC)

Abstract To respond to the challenge of global economic competition, manufacturers are searching for ways to bring high-quality products, system, and structures into being in response to established needs. Simultaneously, as cost is a key factor among many physical and social factors to determine the success of a product, they attempt to reduce costs during every phase of the product's life cycle. The major obstacle for LCC models is data gathering from a highly distributed heterogeneous environment with a huge number of information sources. This paper presents a system analysis approach to design agents for information gathering for CASA model that is one of popular LCC models.

1 INTRODUCTION

Cost is a key factor to determining the success of a product [10]. The life cycle costs of the product can be the total costs at phases of research and development (R&D), production and construction, and operation and support (O&S) [3]. To respond to the challenge of global competition, manufacturers need to reduce costs during every phase of a product's life cycle [18].

However, there are obstacles to use LCC models [2]. One major barrier is data gathering from a highly distributed heterogeneous environment with a huge number of information sources in an organization. When data-processing systems are distributed in various formats, manufacturers have to search them separately and manually integrate information from flat files, relational databases, and remote supplier parts catalogs. CASA model is a typical example [19]. Due to the explosion in the amount of information, it is more useful for collectors to understand customer needs, develop a product to meet these needs, and bring that product to market quickly and at fair value [11]. In recent years, a new wave of changes to the business environment has emerged [13]. The exponential growth of the Internet throughout the last decade has led manufacturing companies to move into a globalized business environment. They can interact with business partners and customers around the world over the Internet. As information from the Internet is diverse, this barrier becomes outstanding.

To overcome this barrier, agent technology is more advanced to deal with information gathering [17, 21]. That is because an agent is able to carry out activities in a flexible and intelligent manner that is responsive to changes in the environment without requiring constant human guidance or intervention [4]. In this paper, we present a system analysis approach for an agent-based system. We then introduce a layered conceptual model for information gathering based on the architecture of InfoSleuth [20].

2 A SYSTEM ANALYSIS APPROACH

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 of OO systems. But as agent technology is getting more popular today, we need an agent-oriented methodology for multi-agent system development. However, as an agent is autonomous, social, reactive and proactive [22], we can not directly apply the OO methodology to agent-oriented software engineering. Current research in role models shows promising results for agent analysis and design [16]. We have combined these two methodologies to specify and develop an agent-based information gathering system for product life cycle cost estimation.

Figure 1 depicts the process of building models for specifying agents. In this figure, each activity represented by a solid box uses several different models, transforming and refining them from activity to activity. The box is given an ICOM (Input, control, output and mechanism) representation adopted from the functional model of IDEF [5,6]. Note that IDEF is a standard modelling tool widely used in manufacturing industry. The thick lines with arrows to connect activities represent interaction collaboration between these two activities.

All activities as shown in Figure 1 can be classified into two categories: agent-oriented analysis (bounded by a dashed polygon) and OO analysis including activities (not bounded by the dashed polygon). OO analysis consists of four activities: "Identify Actors", "Identify Use Cases", "Identify Objects" and "Determine Business Objects". These four activities use the traditional methods developed by Jacobson et al [12].

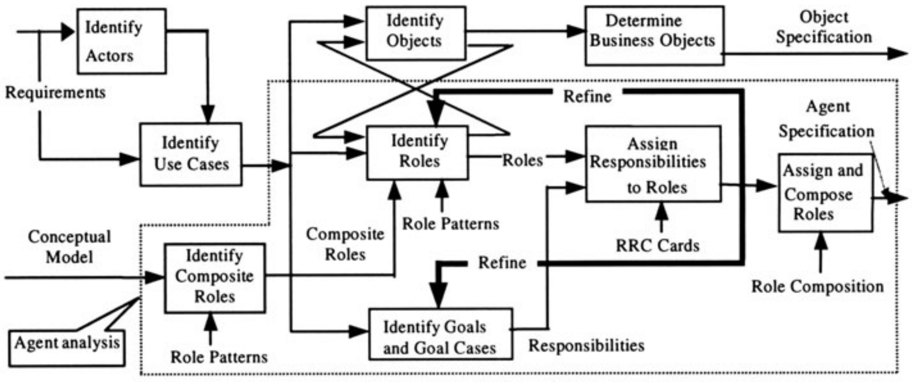

Figure 1 *Agent-oriented analysis process*

The identified use cases are also fed to the activity "Identify Goals and Goal Cases" in agent-oriented analysis processes. A goal is an objective or a desired state that can be achieved or ascertained by an agent. We use $G = \{g_i \mid i \in N, g_i = goal\}$ to define a set of goals. A goal case is a goal-based use case and it belongs to a set of goal cases defined by $U = \{u_i \mid i \in N, u_i = goal\ case\}$. A goal case is a collection of scenarios about the agent's interaction with a system. An agent can start a goal case when the corresponding goal is triggered. We use the existing method [15] to identify goals, goal cases and present them in the hierarchical diagram.

Role models that commonly occur can then be documented as role patterns [14]. One important method is to use role responsibility and collaboration (RRC) cards as shown in Table 1 to specify responsibilities and collaborations of agents, especially in the early phase of agent-oriented software development.

Table 1 Template for a RRC card

Role type		Collaborator
Names of Roles	List all responsibilities	List all collaborators

As a result, all the roles represented by R can be defined as $R = 2^S \times 2^C$ where $S = \{s_i \mid i \in N, s_i = responsibility\}$, 2^S is a power set of S,

$C=\{c_i \mid i \in N, c_i = collaborator\}$ and 2^C is a power set of C. Furthermore, responsibilities can be refined as $S = G' \times U'$ by assigning all potential goals (G') and goal cases (U') for a system.

A conceptual model is used to organize a system in a structure with the functionality to be best supported. We can apply role patterns to this model for identifying composite roles that is a classification of roles for the system. This classification can be instanced in an application. Details of the activity for our application is given in §3. To identify roles from use cases that are the output of the activity "Identify Use Case", the activity "Identify Roles" follows such steps [16]:

- Instance composite roles if they are available;
- Examine role patterns from the existing role patterns. The determined role patterns can specify types of interaction and collaboration of the role with other roles.
- If there are no relevant patterns, partition goals to form roles.
- Determine all roles for the identified interactions and collaborations.

After identifying goals and constructing goal case models, the activity of "Assign Responsibilities to Roles" starts at the bottom of the hierarchy goal diagram, of which the goals are very detailed and would not have any sub goal. This activity determines goal cases that each role can achieve and assigns them as responsibilities of that role.

Once responsibilities are assigned to roles in a multi-agent system analysis, we can partition either identified roles (R) or identified goals (G) and goal cases (U) for agent design. Based on G and U, we can partition $B = 2^{(GxU)} \times 2^C$ for designing a multi-agent system represented by $\bigcup_{\beta} B_\beta = B$. To partition R, we can design another multi-agent system that is expressed by $\bigcup_{\alpha} A_a = R = 2^S \times 2^C = 2^{(G'xU')} \times 2^C$. Note that A_α is a partition in R and stands for an agent. This role partition implies that agent design is to compose elements (roles) that belong to R into different categories (agents) which are subsets of $2^{(GxU)} \times 2^C$. We call this partition procedure as role model composition that is the integration of the relevant role models in an application [1]. The role composition is more than just the sum of the constituent patterns. It captures the synergy arising from the different roles an agent plays in the overall composition structure. $\because R \supseteq B$ $\therefore \bigcup_{\alpha} A_a \supseteq \bigcup_{\beta} B_\beta$. Therefore, role composition method for agents is better and more systematic than directly combination of goals, goal cases and collaborators that are identified from use cases. To assign and compose roles to agents, we have to have the view of the whole agent organization.

- Design agent in appropriated size i.e. to assign appropriated number of roles to each agent.
- Compose the roles for the agents with differentiation emphasizing the specialization that is goal oriented. Split an agent if it has too many responsibilities for the different sub-goals. Merge agents if they got similar responsibilities.
- Compose the roles to agents with good quality of collaboration, for which some aspects are essential such as cohesion, lower coupling, and minimum need for communication.

3. LAYERED CONCEPTUAL MODEL

We develop a conceptual model to organize our system in a structure with the functionality to be best supported. The Layered Architecture pattern [7] has been widely accepted as a standard in network design and software engineering. Here, for our conceptual model, the information-gathering domain is classified into different layers as shown in Figure 2.

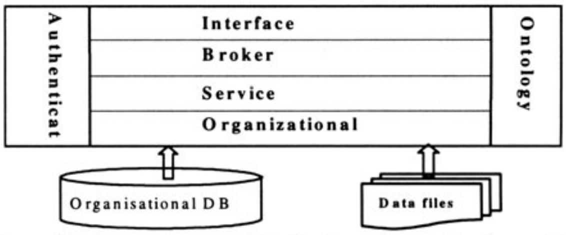

Figure 2. Conceptual model of information gathering system

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 language that would be used for specifying and translating requests for information. The authentication layer performs the task of checking and validating users. The interface layer is used to predict the user's intentions and to request services provided by the remaining modules. This layer acts on behalf of users to relay specifications and obtain results. The broker layer predicts or models the intentions of the overall organization and then provides services 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 have identified composite roles by using role patterns such as Observer, Broker, Master/Slave, Manager, Bodyguard, and Adapter to this model [24]. These composite roles that can instanced in our application are tabulated in Table 2.

Table 2. Summary of the roles for the layered model

Composite Roles	Role Types	Descriptions
Interface Role	• Observer (Observer) • Client-proxy (Broker) • Client (Bodyguard) • Client and target (Adapter)	• Observe user request • Request to provide services. • Obtain result
Broker Role	• Broker (Broker) • Client (Bodyguard) • Subject (Bodyguard) • Client and Target (Adapter)	• Match user's request to service and send the request to service provider • Reply result
Service Role	• Master (Master/Slave) • Server-proxy (Broker) • Client and Subject (Bodyguard) • Client and Target (Adapter)	• Accept the request • Provide service • Distribute work if need • Get final result Send result
Organizational Role	• Manager (Manager) • Slave (Master/Slave) • Client and Subject (Bodyguard) • Client and Target (Adapter)	• Manage information resources • Query information • Reply result
Authentication Role	• Bodyguard (Bodyguard) • Client and Target (Adapter)	• Verify and validate user • Send out permission
Ontology Role	• Adapter (Adapter) • Client (Bodyguard) • Target (Bodyguard)	• Get help request • Find alternatives • Translate Terms

4 AGENT DESIGN

Our application can be modeled by the simplified use case:

(1) The user observes a request about operating and maintaining a product and then waiting for results.

(2) When receiving a request, the maintainer sends requests to a planner for a maintenance plan and to a cost estimator for operating and maintaining (O&S) cost.

(3) The planner distributes work to a relevant information keeper who manages the database for product breakdown structures and requirements.
(4) The cost estimator distributes work to information keepers who manage data for labor, equipment, and material and then calculates the O&S cost. If a material is not available, the estimator distributes work to the purchaser who manages to get it from suppliers.

This use case model describes the system requirement briefly and simply. By using information in Table 2 and interrogating this user case, we have identified roles as shown in Column 2 of Table 3 that are instances of composite roles. By applying rules demonstrated in §2, we partition these instances into different categories separated by dashed lines in Table 3. These categories shown in Column 3 of Table 3 are agents that play the roles that have been instanced.

Table 3 Agent identification

Composite Roles	Instanced Roles	Potential agents used
Interface Role	User/Customer	User agent
Broker Role	Maintainer	Maintenance agent
Service Role	Estimator	Estimation agent
	Planner	Planner agent
Organizational Role	Projects Infokeeper,	Project manger agent
	Labors and Equipment Infokeepers	Resource agent
	Materials and Supplier Infokeepers	Inventory agent
	Support & Management Infokeeper Miscellaneous Infokeeper	Support agent
Authentication Role	Security	Security guard agent
Ontology Role	Term and Location Helper	Information agent

All the agents in our research are implemented by using Jack Intelligent Agents [8, 9]. The organizational databases are accessed using Java Database Connectivity JDBC [23].

5 CONCLUSION AND FUTURE WORK

This paper has proposed a system analysis approach, which provides systematic processes for agent-oriented software development. By applying this approach, we have designed agents used to gathering information for O&S cost by using CASA model. The further research is to develop such a system for the costs of other phases of a product life cycle.

6 ACKNOWLEDGEMENTS

This research is supported by Cooperative Research Centre for Advanced Composite Structures Ltd. (CRC-ACS).

7 REFERENCES

[1] Andersen, E. (1997). Conceptual Modelling of Objects: A Role Modelling Approach, *PhD Thesis, University of Oslo*

[2] Benson, S. (1998). Life Cycle Costs and Dic Pump, preprint, *http://www.discflo.com/lccart.html*

[3] Blanchard, B.S., Fabrycky W.J. (1990). *System Engineering and Analysis*, Prentice-Hall, Inc., New Jersey, USA.

[4] Bradshaw, J M. (1997). *Software Agents*, Menlo Park, Calif.: AAAI/The MIT Press.

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

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

[7] Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P., and Stal, M. (1996) *Pattern-Oriented Software Architecture: A System of Patterns*, Wiley, USA.

[8] Busetta, P., Ronnquist, R., Hodgson, A., and Lucas, A. (1999). JACK Intelligent Agents-Components for Intelligent Agents in Java, *Agent Link News* **2**, January.

[9] Cross, M., Ronnquist, R. (1999). A Java Agent Environment for Simulation and Modelling, *SimTech 99*, Melbourne, Australia.

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

[11] Hennecke, F. (1999). Life Cycle Costs of pumps in chemical industry, *Chemical Engineering and Processing*, **38**, 511-516

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

[13] Jiang, H. C., Mo, J. (1999). Internet Based Design System for Global CE, *Proc. of the 2nd International Conference on Managing Enterprises*, Newcastle, Australia, 150-156.

[14] Kendall, E. A. (1998). Agent Roles and Role Models: New Abstractions for Multiagent System Analysis and Design, *International Workshop on Intelligent Agents in Information and Process Management*, Germany, September.

[15] Kendall, E. A., Palanivelan, U., Kalikivayi, S. (1998). Capturing and Structuring Goals: Analysis Patterns, *European Pattern Languages of Programming*, Germany, July.

[16] Kendall, E. A. (1999). Role Modelling for Agent System Analysis, Design, and Implementation, *First International Symposium on Agent Systems and Applications (ASA'99), Third International Symposium on Mobile Agents (MA'99)*, Palm Springs, Oct.

[17] Knoblock, C. A., Ambite, J. L. (1997). Agents for Information Gathering, *Software Agents*, Edited by Jeffrey M. Bradshaw, AAAI Press/The MIT Press, 347-373.

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

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

[20] Nodine, M., Perry, B., Unruh, A. (1998). Experience with the InfoSleuth Agent Architecture, *Proc. of AAAI-98 Workshop on Software Tools for Developing Agents.*

[21] Sycara, K., Zeng, D. (1996). Multi-Agent Integration of Information Gathering and Decision Support, *ECAI96: 12th European Conference on AI,* Edited by W. Wahlster.

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

[23] Zhang, T., Kendall, E. A. (1999). Agent-based Information Gathering System for LCC, *Proc. of the 1st Asia-Pacific Conference on IAT (Intelligent Agent Technology),* 483-487.

[24] Zhang, T., Kendall, E.A. (2000). System Analysis of Agent-Based LCC Information Gathering, *Proceedings of the 1st Pacific Rim International Workshop on Intelligent Information Agents (PRIIA),* Melbourne, September.