

METRICS FOR AGENT-BASED SOFTWARE DEVELOPMENT

Behrouz Homayoun Far and Tom Wanyama
Department of Electrical and Computer Engineering
University of Calgary
2500 University Drive, N.W.
Calgary, Alberta, Canada, T2N 1N4
{far,twanyama}@enel.ucalgary.ca

Abstract

In software engineering community an increasing effort has been put into design and development of multi-agent systems (MAS). However, agent system development is currently dominated by informal guidelines, heuristics and inspirations rather than formal principles and well defined engineering techniques. In this paper we define a set of objective and subjective metrics to measure the complexity of MAS. The subjective metrics is a modified version of function Point (FP) including the algorithmic complexity and knowledge complexity factor. The objective metrics is a measure for nearly-decomposability, measured by the communicative cohesion. Such metrics can be used to select the best architecture for the MAS. A methodology for agent-based software development based on such metrics is proposed.

Keywords: multi-agent system, software metrics, agent system design.

1. INTRODUCTION

Nowadays, an increasing number of software projects are being revised and restructured in terms of multi-agent systems (MAS). Software agents are considered as a new experimental embodiment of computer programs and are being advocated as a next generation model for engineering complex, heterogeneous, scalable, open, distributed and networked systems. However, agent system development is currently dominated by informal guidelines, heuristics and inspirations rather than formal principles and well-defined engineering techniques.

We have already proposed a methodology for multi-agent system design, called Agent-SE [1]. The Agent-SE offers a way of decomposition (partitioning the problem space), synthesis of the MAS and an appropriate way of modeling and viewing organizational relationships by dynamic derivation of organizational properties. In this paper, we extend the Agent-SE methodology to account

for the decomposition problem. Specifically, we propose metrics for measuring complexity of the participant agents to guide selection of the best architecture.

The structure of this paper is as follows: Section 2 gives a brief description of the Agent-SE approach. In Section 3 the MAS complexity metrics are introduced. In Section 4 an intuitive example is presented and finally a conclusion is given in Section 5.

2. AGENT-SE APPROACH

The method for multiagent system design, called Agent-SE, is based on the abstraction and decomposition, ontology and synthesis and organizational properties. The Agent-SE design steps are as follows:

1. Identify roles and decompose the problem based on function/input/output into an organization of agents.
2. Design the task ontology of the problem.
3. Build an abstraction model and add interactions and signal level organizational relationships using the task ontology.
4. Design each agent and its internal intra-actions using conventional SE techniques (preferably, object-oriented design with UML) and a predefined agent model, if necessary.
5. Based on the domain ontology, design each agent's knowledge-base using Symbol Structure.
6. Derive and record symbol level organizational properties based on interactions of pairs of cooperative or coordinative agents (optional).
7. Derive strategic properties based on interaction of competitive agents (optional).

Agent-SE offers:

- An effective way of decomposition (partitioning the problem space) and synthesis.
- A means of introducing abstraction to the model.
- An appropriate way of modeling and viewing organizational relationships of complex systems.
- Dynamic derivation of organizational properties.

3. MAS COMPLEXITY

In conventional software systems complexity is structural in nature. As the system evolves new components or functions may be added to the system. By doing so, the structure of the software may deteriorate to the extent that major effort is needed to maintain its consistency and conformity with the requirements. On the other hand, complexity of MAS is both structural and algorithmic. These both can be defined in either objective or subjective way. We elaborate on this in the following subsections.

3.1 Structural complexity in MAS

A main complexity component in MAS is structural because new agents may be added to the system or new functions, program modules or packages may be added to the existing agents.

The MAS architecture is the primary artifact for conceptualizing, constructing, managing, and evolving the system under development. It is difficult to draw a sharp line between software design and its architecture. Software architecture is a level of design concerned with issues beyond the computation. Architectural issues include strategic decisions upon:

- Structural issues including gross organization and global control structure.
- Selection among design alternatives.
- Assignment of functionality to constituent agents.
- Composition of constituent agents.
- Protocols for communication, synchronization, etc.
- Physical distribution.
- Scaling and performance.

Hierarchical decomposition is a major method for handling complexity in conventional software analysis and design, assuming that the final product shall have the hierarchical architecture. Unfortunately, hierarchical decomposition cannot be used directly in MAS system development due to the facts that the MAS architecture may not necessarily be hierarchical and MAS analysis and design is not top-down or bottom-up. That is, the participating agents of the MAS cannot be defined at the outset in a hierarchical way. The interactions of the MAS system with the outside world, i.e., use case models, usually come first and then the architectural pattern and participating agents may be decided upon. This is equivalent to moving up the hierarchy. Defining detailed design for each agent is equivalent to moving down the hierarchy.

An architectural pattern expresses a fundamental structural organization schema for the MAS systems. It provides a set of predefined agents, specifies their responsibilities, and includes rules and guidelines for organizing the relationships between them. The most popular architectural patterns for MAS are:

- **Layers:** application is decomposed into different levels of abstraction, such as application layer, business specific layer, middleware layer, etc., and each constituent agent may belong to one layer only.
- **Blackboard:** independent specialized agents collaborate to derive a solution, working on a common data structure called blackboard.

The architectural pattern may be extended to devise the internal architecture of each constituent agent. Common internal architectural patterns are:

- **Model-view-controller (M-V-C):** application is divided into three partitions. The model, which is the business rules and underlying data; the view, which is how information is displayed to the user; and the controllers, which process the user input.
- **Reasoning-communication-documentation engine (R-C-D):** the application is composed of three processing engines. The reasoning engine to process the basic business logic; the communication engine to manage messages to and from the outside world; and the documentation engine to manage data internally [2].

In MAS the relationships among the agents are dynamic. If not, the system can be developed in a much easier way of using *pipes and filters* architecture, in which the data is processed in streams that flow through pipes from filter to filter.

Two kinds of dynamic relationships can be devised: interactions among subsystems and intra-actions within subsystems. **Interactions** are between an agent and its outer environment and manifested by the messages sent, received (in case of cooperation and coordination) and perceived (in case of competition). **Intra-actions** are the characteristics of the agent's inner environment.

Contemporary software engineering techniques can manage the intra-actions using decomposition and abstraction techniques and interactions using RPC, RMI, etc.

3.2 Algorithmic complexity in MAS

Algorithmic complexity stands for the mechanisms for knowledge processing and knowledge sharing as well as the ability to engage with the other agents in cooperative, coordinative and competitive tasks [1].

3.3 MAS complexity metrics

Subjective metrics:

Subjective complexity accounts for the way that a human user evaluates the complexity of the agent system. A modified version of *Function Point (FP)* [3] that accounts for algorithmic complexity can be used. For each participant agent, the parameters involved in the model are: External inputs (N_i) and external outputs (N_o),

external inquiries (N_q), external interfaces (N_{ef}), internal data structures (N_{if}), algorithmic complexity (N_m) and knowledge complexity factor (N_k). The algorithmic complexity (N_m) factor is the sum of three Boolean variables stating whether cooperative, coordinative and competitive mechanisms are implemented or not ($0 \leq N_m \leq 3$). The knowledge complexity factor (N_k) has a value between 0 and 5 depending whether the agent has a knowledge-base and whether the knowledge-base is sharable or is based on a shared ontology.

$$UF_eC = 4N_i + 5N_o + 4N_q + 7N_{ef} + 7N_{if} + 10N_m + 6N_k$$

The adjusted MAS function point (MAS-FP) is derived by multiplying UF_eC with the subjective assessment of technical complexity, the TCF factor [3]. The overall complexity of the MAS will be the mean of the adjusted MAS function point of its constituent agents.

Objective metrics:

Objective complexity accounts for complexity as an internal property of the agent system. If the MAS system is nearly-decomposable, the cyclomatic complexity [4] metrics can be used. Complexity of the MAS is the sum of cyclomatic complexity of its constituent agents. As a measure for nearly-decomposability, the communicative cohesion metrics can be examined. The communicative cohesion metrics (CCM) for an agent g_i is defined in terms of the ratio of internal relationships (inter-actions) to the total number of relationships (i.e., sum of inter-actions and intra-actions).

$$CCM(g_i) = \frac{R_{\text{internal}}}{R_{\text{internal}} + R_{\text{external}}}$$

The CCM for the MAS is the statistical mean of CCM of its constituent agents. Systems with $CCM \geq 0.9$ are usually considered to be nearly-decomposable.

3.4 Deployment

The first step in design of a MAS system is to identify roles and decompose the problem based on function/ input/ output into an organization of agents. The problem is how to assign roles to the constituent agents at the first place. The MAS complexity metrics can be used for such purpose. First, the target CCM and UF_eC is set and decomposition is performed to devise a tentative set of agents with CCM greater than the target value. Then the UF_eC is measured for each agent and those with higher UF_eC value will be the target for further decomposition. These steps are repeated until all the agents have satisfactory CCM and UF_eC .

4. EXAMPLE

This example explains an agent-based system called Electronic Realtor (eRealtor) that uses the advanced computer and communication technologies to assist the users in the highly dynamic real estate market.

The real estate market involves buyer and sellers who search for counterparts. It also involves one or more middlepersons who holds information about sellers and buyers. The middleperson usually provides transaction services such as *search*, *coordination* and *settlement*. *Search* reflects effort by a seller or buyer to obtain information on counterparts that best fit his/her transaction conditions. *Coordination* is the next step after deciding upon a few trading candidates. It is the effort of the seller and buyer to increase their profit and value (i.e., utility function). *Settlement* clears the transactions through physical exchange of the home and the accompanying payment. The proposed eRealtor takes on the role of middleperson by providing the real estate market participants with transaction services using the Internet (see Fig. 1). In addition, the eRealtor deals with multiple mortgage companies and electronic newspapers to provide advisory services to its clients. The eRealtor moderates and provides negotiation support during the coordination stage of the transaction on the Internet.

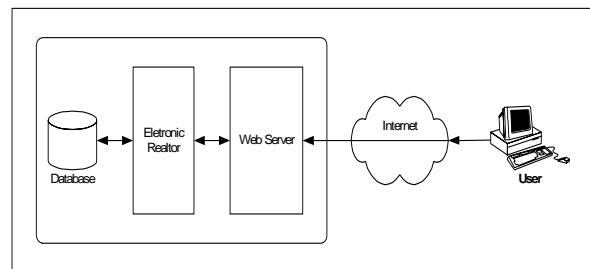


Figure 1. eRealtor system overview

4.1 System Requirements

The minimum set of requirements for such system is:

- The eRealtor provides home listing services.
- The eRealtor provides home searching service.
- The eRealtor introduces buyers (if they so wish) to mortgage companies.
- The eRealtor deals with various mortgage companies and electronic real estate newspapers.
- The eRealtor automatically matches buyers and sellers, and provides an indirect communication link between them to negotiate and schedule meetings.
- The eRealtor acts as the moderator during negotiations.
- The eRealtor helps the buyer to choose a mortgage company depending on his financial status, and

provide a communication link between them to schedule meeting.

- The eRealtor provides advisory services to buyers and sellers. The advice given depends on the role of the participant.
- The eRealtor provides facilities for calculating taxes and advising participants on this issue.
- The eRealtor provides facility for connecting to other electronic home listing databases, so as to provide participants with a big population of prospective candidates.
- The eRealtor provides facility for billing users.

4.2 Role Identification

In the eRealtor, the following roles are required.

1. Role of home buyer's personal assistant
2. Role of home seller's personal assistant
3. Role of banker's personal assistant
4. Role of mortgage and tax assistant
5. Role of handling organizational security and user registrations
6. Role of moderator and coordinator
7. Role of office assistant for information storage and retrieval (filling)
8. Role of real estate economic advisor
9. Role of economic assistant and gathering real estate information from electronic newspapers and journals or publishing real estate information.

4.3 Constituent Agents

From this specification, one may naturally derive that the candidate agents should be buyer-agent (role 1), seller-agent (role 2), banker-agent (roles 3-4) and realtor-agent (roles 5-9). We set the CMM be greater than 0.9 and UF_cC be less than 250. Although all the proposed agents pass the CCM test, the realtor-agent failed the UF_cC test. The UF_cC was calculated as around 450 indicating too much complexity in this agent.

We revised the architecture and decomposed the realtor-agent into 3 agents, namely, the administrator-agent, economic-analysis-agent and market-signal-agent. The administrator-agent is assigned to implement roles 6-7. The market-signal-agent and the economic-analysis-agent implement roles 8 and 9, respectively. Also the login-agent was assigned to implement role 5. The new architecture could pass both CCM and UF_cC tests.

5. CONCLUSIONS

Although there are other methodologies for MAS system analysis and design, such as [5] and [6] none of them address the decomposition problem of how to assign the function or roles to the constituent agents. The

authors' Agent-SE method suggests decomposition based on roles, functions, inputs or outputs [1]. In this paper, we extended the Agent-SE method by addressing the decomposition problem and proposed metrics to measure the complexity of the agents. The metrics were used for devising a candidate set of agents for MAS analysis and design.

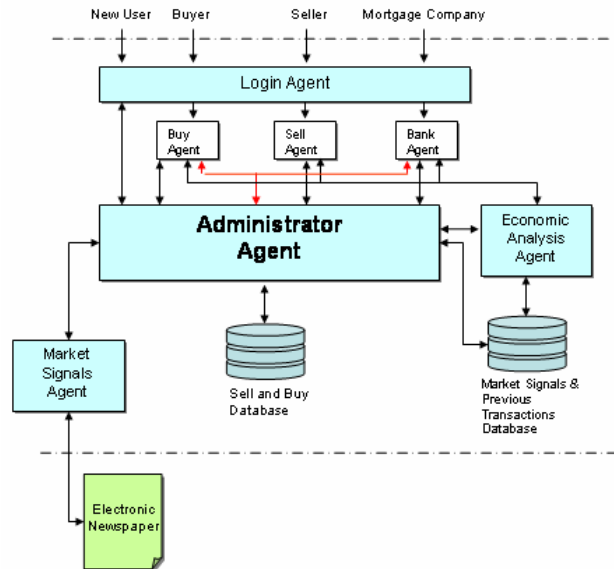


Figure 2. The final eRealtor system architecture

References

- [1] Far, B.H., Agent-SE: A Methodology for Agent Oriented Software Engineering, Enabling Society with Information Technology, Q. Jin et al. eds., pp. 357-366, Springer, 2001.
- [2] Far, B.H., et al., An Integrated Reasoning and Learning Environment for WWW Based Software Agents for Electronic Commerce, IEICE Trans. Inf. and Syst., vol. E81-D, no. 12, pp. 1374-1386, 1998.
- [3] Albrecht, A.J. and Gaffney, J.F., Software Function, Source Lines of Code and Development Effort Prediction: A Software Science Validation, IEEE Trans. Software Engineering, vol. 9, no. 6, pp. 639-648, 1983.
- [4] McCabe, T.J., A Complexity Measure, IEEE Transactions on Software Engineering, vol.2, no.4, pp. 308-320, 1976.
- [5] Lind, J., The MASSIVE Development Method for Multiagent Systems. Proceedings of the 5th International Conference on the Practical Application of Intelligent Agents and Multi-Agents (PAAM2000), 2000.
- [6] Wooldridge, M., Jennings, N.R., and Kinny, D., A Methodology for Agent-Oriented Analysis and Design. In O. Etzioni, J. P. Muller, and J. Bradshaw, editors: Agents'99: Proceedings of the 3rd International Conference on Autonomous Agents Seattle, 1998.