A Process for BDI Agent-based Software Construction

Chang-Hyun Jo
Department of Computer Science
California State University Fullerton
Fullerton, CA 92834-6870
714-278-7255
jo@ecs.fullerton.edu

Jeffery M. Einhorn
Department of Computer Science
University of North Dakota
Grand Forks, ND 58202-9015
701-777-4105
jeinhorn@cs.und.edu

Abstract

Agent-based programming comes us as a next generation programming paradigm. However, we have not been ready yet to fully use it without having sound and concrete software engineering methods and tools to facilitate agent-based software development.

In this paper we propose a new software engineering process based on the BDI agent concept. We have refined and extend substantially our previous work, Agent-based Modeling Technique (AMT) and Agent-based Software Development Process (ASP), so that a systematic and realistic process has been born to construct BDI agent-based software. This paper introduces our new approach to the BDI agent-based software development process.

The Belief-Desire-Intention (BDI) model has been proved as a dominant view in contemporary philosophy of human mind and action. We utilize BDI as a tool to analyze agents' environments, goals, and behaviors. Use cases have been proved as a useful tool for requirement analysis. However, use cases cannot be neither agent-oriented nor object-oriented even though it has been used as a tool for analysis for a while. We have extended the existing use cases, and use a new sort of use cases to identify BDIs of agents in the real-world problem.

We use external use cases to get the basic behaviors (intentions) needed to provide the services in the system. We use then internal use cases to define goals (desires) of the system and to discover more specified behaviors (intentions) to achieve the goals. By analyzing the behaviors (intentions) for each goal (desire), we can obtain environments (beliefs) on which the system behaves to perform the goal.

The goal of this paper is to provide a very practical and systematic way to analyze and design the agent software based on the BDI concepts. We have started by using the existing proven tools and methods such as the use case approach, however, we have made a substantial modifications and improvements to these existing techniques so that we can analyze and design the system very realistically based on the BDI agent concept.

This paper provides a systematic and seamless approach to the BDI agent-based software development. The way we suggest here to find BDI agents through requirements analysis is a unique and novel approach. This technique suggests a new way of thinking for BDI agent-based modeling.

Keywords

Agents, BDI agent-based software development process

1. Introduction

Agent-based software development provides a next generation of software construction. Agent-based software consists of agents cooperating to achieve a common goal. To succeed the common goals, agents can be working in the form of highly distributed, mobile, autonomous, intelligent and cooperative entities. Building systems based on agents gives a more natural way to simulate complex real-world systems [12]. Therefore, agents have been next generation entities to model the complex systems in many research works [13, 24, 21, 22, 23, 6, 11, 14, 18, 20].

The rationality of intentional action is viewed as a primary function of the agent's desire-belief reasons for action [4]. An agent's desires and beliefs at a certain time provide her with reasons for acting in various ways. This Belief-Desire-Intention (BDI) model has been used to

--
IMCSE 2003 – SERP'03
Las Vegas, Nevada, USA
June 23-26, 2003

describe the behavior of agents with certain goals on a certain environment [19, 24].

The goal of this paper is to provide a very practical and systematic way to analyze and design the agent-based system based on the BDI agent model. In this paper we present a new way of development process by using various kind of artifacts to model agents of the system based on their belief, desire, and intentions. In our modeling, real-world entities are described as agents when we assign their beliefs, desires, and intentions.

Previously we have once introduced the Agent-based Modeling Technique, AMT [14]. We have also briefly described a draft definition of our Agent-based Software Development Process (ASP) in the previous work. This paper is to provide more mature and concrete steps and artifacts in each step. We have also adopted other techniques such as different kinds of use cases to analyze the requirements and to aid in finding beliefs, desires and intentions of agents in the system. Our method presented here is a very unique and systematic approach to find BDIs and to assign them to appropriate agents to perform the system services. The new technique we presented here is realistically useful. It provides a seamless and systematic approach from the analysis to the implementation.

Adoption of use case approach has made our process more realistic and systematic. Use cases have been a proven tool in analyzing systems [5]. Use cases are neither object-oriented nor agent-oriented approach. Using use cases is rather functional approach. However, use cases have been well used to gather requirements in the analysis phase. Functional approach has also been well used in object-oriented analysis [17]. We have adopted this approach first and then extended substantially to find beliefs, desires, and intentions of the system.

To properly model both BDIs and agents in the system, we have developed a unique technique for requirements analysis to use various kinds of use cases. We have extended the existing use cases technique by categorizing external use cases and internal use cases. We have developed a new way to describe a system by using both external use cases and internal use cases. We use external use cases to get the basic behaviors (intentions) needed to provide the services in the system. We use then internal use cases to define goals (desires) of the system and to discover more specified behaviors (intentions) to achieve the goals. By analyzing the behaviors (intentions) for each goal (desire), we can obtain environments (beliefs) on which the system behaves to perform the goal.

In our model, a system consists of cooperative and communicating agents. An agent is defined by a set of BDIs [14]. To model the system by using the BDI agents, we go through two steps: (1) In analysis, we use various artifacts including use cases to find BDIs and agents in the system; (2) In design, we assign BDIs to appropriate agents in the system.

We propose a new BDI agent-based software development process in the next sections. We show this step by step, and we will show how the information previously we found in the analysis is correlated and transferred into the later artifacts. The artifacts we generate in the requirements analysis will be naturally used in the artifacts in other phases of both analysis and design. We explain here briefly how the rest of phases will go through, but the more detailed version for other phases will be introduced in separate papers.

In the next section we present our modeling process with corresponding steps suggested in AMT. Section 3 describes each activities involved in the analysis step of the AMT. Section 4 explains the activities and artifacts needed in the design step of AMT. Section 5 concludes our presentation and lists references used.

2. A process for BDI-agent software development

In our previous research, Agent-based Software Development Process (ASP) and Agent-based Modeling Technique (AMT) have defined four development phases such as requirements analysis, modeling, construction, and deployment. Each phase suggests four steps analysis, design, build and test iteratively and evolutionally [14].

AMT suggests the analysis step include the following sub-steps:

- Analyze the system requirements
- Construct BDI agent cards
- Identify agents and concepts
- Identify relationships among agents
- Build agent scenarios
- Identify agent boundary

In the design step, AMT suggests the following sub-steps:

- Build agent relationship diagrams
- Build agent interaction diagrams
- Use agent patterns
- Build agent component diagrams

Figure 1 shows dependencies among deliverables generated in the sequent steps in AMT.

In this research we refine the ASP by defining more precise activities and concrete artifacts. In the next section we will introduce our new version of software development process named the BDI Agent-based Software Process (BDI ASP). We should present both a brief discussion of each artifact created in each phase as principles and an example of the artifact as practices. Because of the limitation of the paper length, however, we cannot provide all examples of the artifacts. To see a more detailed example and a more realistic case study, you may refer to our web page [16].

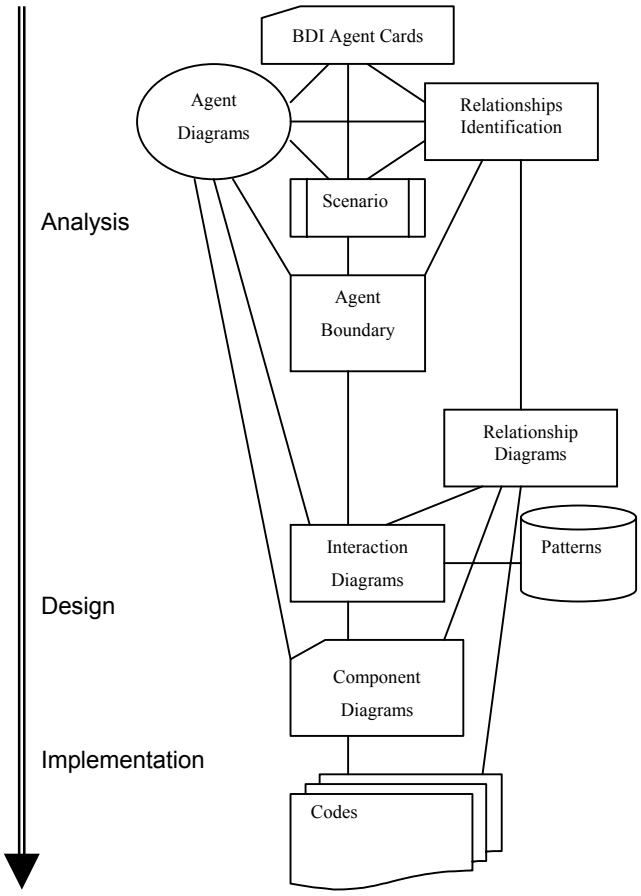

Figure 1. Dependencies among deliverables

3. BDI agent software analysis

Here we describe the steps and artifacts for the analysis step in our BDI Agent-based Software Development Process (BDI ASP). The steps described here are rather concurrent than strictly sequential steps. Many artifacts here are correlated one another, therefore they can be concurrently constructed and use information to construct one part of artifacts from others.

3.1 System requirements analysis

3.1.1 **Problem Statements.** Our requirements analysis phase starts by building initial problem statements. After an iteration of full phases in the process, the initial problem statements are revisited and refined.

3.1.2 **Enterprise assessment.** We then assess the enterprise software to find how new requirements fit the existing enterprise. In each iteration, we redo the enterprise assessment.

3.1.3 **External system view.** The next step is to discover all the functions that system should provide for the new requirements. This is gone through by writing external use cases. With external use cases, we can describe the system functions from the external point of view. This system functions will be foundations of behavioral plans (intentions). External use cases become to the detailed external use cases during the several iterative steps.

3.2 Conceptual agent identification

Once we create external system view, we try to create a conceptual agent list. Conceptual agents are candidate agents, which might or might not be adopted as software agents in the design phase. Conceptual agents provide us with information for possible candidates so that we can focus on those agents to assign proper BDIs later. Conceptual agents can be found by using linguistic analysis as widely used in the object-oriented analysis. We use nouns or noun phrases to find candidate agents from the previous artifacts such as problem statements and external use cases.

3.3 Scenario construction

An initial scenario describes the system process in plain sentences. The scenarios will become more formalized by using a form to describe a scenario name, a set of clients, a set of servers, goals and plans. Scenario helps us to find out the system process, related client agents and server agents, their goals, and plans.

3.4 Agent relationship identification

After we are familiar with conceptual agents in the system and the system process through the scenarios, we build the conceptual agent relationship diagram (ARD). A conceptual ARD provides conceptual relationships to show how the system functions are achieved among the conceptual agents. Conceptual ARDs will be used to find out related agents in the design step, and conceptual ARDs do not guarantee any software implementation yet. During the iteration, conceptual ARDs will become detailed ARDs. ARDs use oval representation for external agents and rectangular representation for internal agents.

3.5 Internal system view

We decompose the system functions investigated from the external use cases by building internal use cases into the goals (desires) that are supposed to be assigned to appropriate agents. Agent decomposition is done in the reverse order of belief-desire-intention lists suggested in Jo's work [14]. From the external system view through the external use cases construction, we list the goals. A system service described in an external use case is decomposed into one or more goals. Each decomposition step creates an internal use case. An internal use case describes the plans for each goal that is identified to provide a system service described in an external use case. Each goal has a plan that may include other goals, which in turn include their own plans.

3.6 Agent belief lists

We have found so far plans (intentions) and goals (desires) through the previous steps. Now it is time for us to identify the environments (beliefs) required to achieve goals for agents. Goals and plans certainly require information to access and update. Such information forms environment on which agents are working to achieve goals by using plans. This environment can be represented as states in a knowledge base, and it is called as belief in our model. The agent belief lists (ABLs) show what kinds of beliefs are needed to achieve goals. An agent belief list consists of a system service name, a set of triples (goal, belief, reason) to perform this service.

3.7 Brief BDI agent cards construction

A BDI agent card summarizes the system requirements, agents, and their BDIs. A BDI agent card lists agents' names with a set of their BDIs, collaborators, pre-conditions, and post-conditions respectively. BDI agent cards explain the static property of the agent system. The goal and belief identified from agent belief lists (ABLs) become concrete desire and belief. The plans we identified by external system view and internal system view are used to define intentions. In the first iteration, a BDI agent card might be brief and abstract. It may not include any detailed idea, but it gives a guide to identify proper agents and their BDIs. In the consecutive iterations, brief BDI agent cards become more detailed and concrete.

3.8 Agent boundary identification

An agent boundary diagram (ABD) provides conceptual view of the system with the external services. All conceptual agents will be categorized into their agent boundaries. ABDs will be eventually foundations of components diagrams in the design step that will be useful in the construction and deployment phases.

4. BDI agent software design

In the previous sections we have shown the steps and artifacts for in the analysis phase of our BDI agent-based software development process. The next consequence is the design phase of our BDI agent-based software development process.

4.1 Detailed agent relationship diagrams

In the design step, we suggest to refine the conceptual agent relationship diagrams (ARDs). Conceptual ARDs will become more detailed and concrete through the several iterative and evolutionally steps in different phases. With the concrete agent relationship diagrams we can identify collaborators to achieve common goals among cooperative agents in the system.

4.2 Agent interaction diagrams

Here we assign goals (desires) to appropriate agents, plans (intentions) by which agents achieve the goals, and environments (beliefs) on which agents work to achieve the goals by plans. Not only an agent interaction diagram shows goals and participating agents, but also it describes the dynamic characteristic of the agent system.

To achieve the system service we have to find out which intentions are proper to fulfill the system requirements gathered through the external view of the system. Then we find what are the goals to perform these

intentions, and which agents are responsible for it. We assign goals to appropriate agents at this time. Certain patterns such as the agent goal assignment pattern [7] are useful to find the most appropriate agents to fulfill the goals. An agent interaction diagrams shows a set of agents and their communications via invoking their goals.

4.3 Agent patterns

During the software design through the several steps and artifacts listed above such as agent interaction diagrams, some patterns for BDI agents can be very useful. There are not many patterns for agent development [10], but we may use patterns for object-oriented programming to some extent [9]. However, they do not provide enough information to find BDIs and appropriate agents, we have to develop own patterns to be used for BDI agents construction. Einhorn [7] suggests some patterns useful for the BDI agent software design.

4.4 Detailed BDI agent cards

The brief BDI agent cards in the analysis step are used for capturing system requirements and finding corresponding responsibilities. The detailed BDI agent cards we build in the design step are used to assign the responsibilities to the appropriate agents by defining their BDIs and collaborators.

Using patterns, we can assign BDIs to appropriate agents more efficiently. The brief BDI agent cards constructed in the previous phase are refining more precisely at this time. Within a few iterations, the detailed BDI agent cards show enough information ready to implement from which programmers can construct software agents and testers can construct test cases at least for prototyping. BDI agent cards with other artifacts will be refined and evolved during continuous iterations.

Mostly detailed BDI agent cards are constructed in parallel with agent interaction diagrams.

4.5 Agent component diagrams

We also build agent component diagrams (ACDs) based on artifacts, we constructed during the analysis step and design step, such as agent boundary diagrams (ABDs), agent relationship diagrams (ARDs), agent interaction diagrams (AIDs), and BDI agent cards. The agent component diagrams will be eventually used as guidance for packaging of agents and their BDI packages in the construction and deployment phases.

4.6 Guidelines

A BDI agent software development guideline suggests how we can develop software agents and how we can implement our models exactly by suggestive mapping models into agent-based programming languages. This is also a very important guideline because it gives both precise and exact steps we have to follow to develop the BDI agent software applications from our models.

4.7 Mapping to codes

The next step is the implementation of the BDI agents figured out from the requirements analysis and the agent design models. The artifacts in both analysis and design and the guidelines can show us how to implement BDI agent-based software in the real programming languages.

There are many ways to implement our BDI agent models in different languages. A good way might be choosing a proper agent-based programming language in which BDI models can be implemented properly and naturally. One of our research works in agent computing includes a design and implementation of a new agent-based programming language, APL [15]. APL provides a natural syntax and semantics to implement the BDI agent models suggested by the Agent-Modeling Technique (AMT) [14].

We will show this mapping in a separate paper because of the limitation of the paper length here. In this paper, we emphasize the agent software analysis and design only based on the BDI agent concept.

4.8 Concurrent artifacts construction

Some artifacts can be (or must be) constructed in parallel. For example, the BDI agent cards can be concurrently constructed with agent interaction diagrams. While BDI agent cards describe the agent system architecture globally and statically, each agent interaction diagram describes the system service locally but dynamically.

4.9 Evolutionary and iterative approach

In the rest of phases, construction and deployment, while we evolutionally and iteratively refine BDIs and assign BDIs to proper agents, we implement software agents in agent-based programming languages. The artifacts that are modeled and their corresponding software that is implemented are evolving more and more, not only

through the iterative software construction but also through the whole agent software life cycle.

5. Conclusions

Systems go more complex and embedded so that software engineers are hard to analyze and design the system with a simple model. The BDI model teaches us how agents plan their intentions with reasoning desires on their belief naturally.

In this paper we use the BDI concept to model agent-based system. We have shown our development process for the agent-based software construction based on the BDI agent model.

Even though at first we have started to use the existing proven methods and tools used in the object-oriented modeling techniques [1, 2, 3, 8], we have refined and extended substantially the existing methods to adapt into the BDI agent-based software construction. As a result, we have concluded with our own BDI agent-based software development process. We have briefly but precisely defined the phases and their steps in the BDI agent-based software development process, and described them reasonably.

This work is a very unique and new approach in the agent-oriented software engineering. Our method will provide a sound, realistic and practical modeling technique for agent-oriented software development.

With the limitation of the paper length we could not list all steps and examples of the whole process. To see a more detailed example and a more realistic case study, you may refer to our web page [16].

6. References

[1] Bellin, David and Simone, Susan, The CRC Card Book, *Addison-Wesley*, 1997.

[2] Booch, G., Object-Oriented Analysis and Design with Applications, *Addison Wesley*, 1994.

[3] Booch, G., Rumbaugh, J., and Jacobson, I., The Unified Software Development Process, *Addison-Wesley*, 1999.

[4] Bratman, M. E., Intention, Plans, and Practical Reason, *Harvard University Press*, 1987. (also available from CSLI Publications, 1999)

[5] Cockburn, Alistair, Writing Effective Use Cases, *Addison-Wesley,* 2001.

[6] Depke, Ralph, Heckel, Reiko, and Kuster, Jochen, Improving the Agent-Oriented Modeling Process by Roles, *AGENTS'01*, June 2001, 640-647.

[7] Einhorn, M. Jeffery, A BDI Agent Software Development Process, MS Thesis, (Advisor: Chang-Hyun Jo), Department of Computer Science, University of North Dakota, May 2002.

[8] Fowler, Martin and Scott, Kendall, UML Distilled Second Edition: A Brief Guide to the to the Standard Object Modeling Language, *Addison-Wesley*, 2000.

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

[10] Hayden, Sandra, Carrick, Christina, and Yang, Qiang, A Catalog of Agent Coordination Patterns, ACM Press, 1999, 412-413.

[11] Iglesias C. A., Garijo M, Gonzalez J. C., and Juan R. Velasco, Analysis and Design of Multiagent Systems using MAS-CommonKADS, In M.P. Singh, A. Rao, and M.J. Wooldridge, editors, *Proc. 4th Int. Workshop on Agent Theories, Architectures, and Languages (ATAL-97)*, volume 1365 of LNAI, Springer-Verlag, July 24-26, 1998, 313-328.

[12] Jennings, Nicholas R., On agent-based software engineering, *Artificial Intelligence,* volume 117, February 2000, 277-296.

[13] Jennings, Nicholas R., An Agent-Based Approach for Building Complex Software Systems, *Communications of the ACM*, 44(4), April 2001, 35-41.

[14] Jo, Chang-Hyun, A Seamless Approach to the Agent Development, *ACM SAC 2001*, Las Vegas, March, 2001, 641-647.

[15] Jo, Chang-Hyun and Allen J. Arnold, Agent-Based Programming Language (APL), *ACM SAC 2002*, Madrid, Spain, March 2002, 27-31.

[16] Jo, Chang-Hyun, Case Studies of the Agent-based Modeling Technique (AMT), http://www.ecs.fullerton.edu/~jo/research, 2003.

[17] Larman, Craig, Applying UML and Patterns: Second Edition, *Prentice-Hall*, 2002.

[18] Petrie, Charles, Agent-Based Software Engineering, *Agent-Oriented Software Engineering, Lecture Notes in AI*, *Springer-Verlag,* 2001, 58-76.

[19] Rao, Anand S. and Georgeff, Michael P., BDI Agents: From Theory to Practice, *Australian Artificial Intelligence Institute*, April, 1995.

[20] Weiss G., editor, Multi-Agent Systems, *The MIT Press: Cambridge, MA*, 1999.

[21] Wooldridge, M. and Jennings, N. R., Intelligent Agents: Theory and Practice, *Knowledge Engineering Review, Cambridge Univ. Press,* 10(2), June 1995, 115-152.

[22] Wooldridge, M. and Jennings, N. R., Software Engineering With Agents: Pitfalls and Pratfalls, *IEEE Internet Computing*, May-June 1999, 20-27.

[23] Wooldridge, M., Jennings, N. R., and Kinny, D., A Methodology for Agent-Oriented Analysis and Design, *Autonomous Agents 1999, Seattle, WA,* 1999. 69-76.

[24] Wooldridge, M., Reasoning about Rational Agents, *The MIT Press: Cambridge, MA*, 2000.