지능 로봇의 에이전트 기반 정보 통합

이 재 호*

◈ 목 차 ◈

1. Introduction

Intelligent service robots consist of variety of complex software components that are usually distributed over several single board computers (SBCs). Those components need to be integrated into a single unit to form an integrated identity of a robot. We have been building an integrated software framework for intelligent service robots for the past three years exploiting the benefits of agent-based approach to both establishing the software engineering process and designing the robot architecture. In this paper, we present an agent-based approach at the deliberative layer of robotarchitecture to develop large complex quality software systems consisting of heterogeneous and distributed software components. We show the level of abstraction provided by the agent-oriented methodology and design steps to build software systems of intelligent service robots.

While Industrial robots provide simple, repetitive, and physical supports, intelligent service robots are characterized by the integratingvarious intelligent functionalities. Designing and building intelligentrobots thus involves integration of various functionalities such as manipulation, navigation, various recognitions, speech understanding and expression, reasoning, planning, and so on. Furthermore, such functional components are often distributed over several processors even in a single robotic system. As the number of intelligent functionalities grows steadily and the demand for more flexible, adaptable, extensible, and robust intelligent components increases, software development complexities are also rapidly growing and new software engineering methodologies and development strategies becomes critical.

2. Agent-Based Approaches

Agent technology is now being recognized as an indispensable new software engineering methodology and development strategy to manage the intertwined complex interactions between near-independent software components in complex software systems. In this section, the characteristics of agent-based software development are presented and then an agent-based software engineering approach is proposed. In general, an agent is a proactive autonomous software unit interacting with its environment and other agents and decides autonomously its behavior to act on behalf of the user based on its goal and belief. As a software component, an agent exhibits some or all of the following additional characteristics.

* 서울시립대학교 조교수, 부교수

Adaptability: The agent can update or augment its knowledge and capabilities by its own learning mechanism or instruction from others so that it can change its behavior appropriate for the new environment.

Autonomy: The agent can act proactively independent of other agents on its own thread of control in accord with its goal.

Consciousness and knowledge ability: Theagent is aware of, and can reason about the goals, capabilities and knowledge of itself and other agents.

Persistency: The agent retains its state including knowledge and goals over extended periods of time even when unexpected failures occur.

Sociability: The agent is conscious of the existence of other agents and is able to communicate and interact with other agents to form an agent community.

Mobility: The agent can move from one execution environment to another and then can continue execution in the new execution environment by moving its code as well as its state and context to continue its execution.

These characteristics combined with the advantages of conventional component-based software engineering offer an effective alternative for building large distributed web-based application systems. The agent-based software system design phase includes new design activities such as identification of agents for the large distributed systems.

Each agent has its own belief, desire, and intention as the basic an autonomous software component of the system. An agent acts as an actor of interactions between these basic components. Agents may be implemented using active distributed objects with scripted behavior specifications or using specialized agent programming language. Agents are located in an agency which provides facilities for locating other agents and communicating messages. Agent communication requires standard patterns of interactions between agents and structure of the communicating messages. An agent communication language (ACL) such as KQML [1] and FIPA ACL (see http://www.fipa.org) defines rules of communications including how communication is to be started and finished. Recently these languages have been converted to a standard XML [2] encodings.

The interacting agents need to understand the vocabulary used to communicate with each other. The set of vocabularies are also called ontology. The ontology describes objects, attributes, operations in the given domain, their relationships and meanings, and rules of interactions with other agents and services.

Multiagent Systems integrate a group of possibly heterogeneous agents and users to coordinate their activities. Agents in a multiagent system may dynamically join or leave the system as in many e-commerce applications. The interaction pattern determines the characteristics of the multiagent system. The interaction patterns may be specified in declarative rules, interaction protocols, or workflow [3].

Agent-oriented programming encompasses all of the above mentioned components, that is, identification of agents, agency, agent communication languages, ontology, and interaction patterns in multiagent systems. Note that the basic components of the agent are conceptually consistent with the components of the knowledge level introduced by Allen Newell [4]. At the knowledge level an agent is described as having a set of goals, a set of possible actions, and a body of knowledge. Agent's behavior is predicted by the principle of rationality.

Currently we are developing agent-based robot architecture to support integration of intelligent robot software components. Our architecture providesa suit of tools to develop individual agents and interactions based on blackboard architecture. Agents are currently implemented as blackboard agents and the architecture works as a constraint or guideline to the development of agent systems. A sketch of the development phase in this guideline is described in the following sections.

2.1 Requirement Analysis and Definition of Belief

As in the conventional software engineering, requirement analysis is performed as the first step of the development. One of the major results of requirement analysis is the definition of agent's belief. Belief is represented in terms of basic objects and predicates defined over the objects. In this process, existing ontologies may be searched, referred, specialized and updated. Belief system may exist in a reusable library. In this case, new belief system can be derived from the existing one through parameter initialization.

2.2 Definition of Agent's Actions

Rigorous formal definition of agent's intention [5] has been applied various agent systems. In this paper, we adopt a simple notion of intentions. Agent's intention simply means a series of actions committed to achieve goals. Thus agent's actions are the means of realizing agent's intention.

The series of actions are represented as the state of progress made toward the agent's goals. Basic actions of an agent thus need to be identified. In this stage, agent's basic actions are identified and represented as primitive actions. Primitive actions are atomic actions which run to completion without being interrupted until either successful or failed termination of the action. Complex behaviors can be expressed as combinations of primitive actions using plans (Section 2.4) or goals (Section 2.3) to be explained in the following sections.

2.3 Definition of Agent's Goals

An agent has goals to achieve or maintain. A goal specifies the state to bring about or keep up. Goals are achieved or maintained by means of agent's plans (see Section 2.4).

An agent's goals are classified into two categories: top-level goals and subgoals. Top-level goals are the highest-level goals that the agent has. Top-level goals are usually given to the agent by the user of the agents as a specification of tasks. Top-level goals are persistent in that they are pursued until either they are either satisfied or removed explicitly. Goals can be satisfied by successful plan execution or opportunistically by some other means such as other agents.

Subgoals are goals that the agent creates as subtasks from within plans during plan execution. Subgoals are not persistent by default. If a plan to achieve a subgoal fails, the subgoaling action is considered to have failed as if it were any other type of valid plan action. Subgoaling can be performed to arbitrary depth levels and can be recursive, where a plan for a particular goal can subgoal to the same goal.

Agent's goals have prioritiesassociated them. The priority is dynamically calculated and thus situation-dependent. On every cycle of execution, the execution thread switches to the plan for the goal with highest priority.

In this stage, top-level goals and subgoals are identified and structured in a goal hierarchy. The dependency between goals can be either AND-dependency or OR-dependency. In order to achieve a task, all the subtasks in AND-dependency and at least one subtask in OR-dependency must be satisfied. Since XML is good for representing hierarchical structure and RDF for representing task dependencies, we adopt both XML and RDF representations for agent's goals.

2.4 Implementation of Agent Plans

Agent's plans are the means to achieve the goals. A plan defines a procedural specification for accomplishing a goal. Applicability of a plan for a goal is limited to a particular goal and constrained to a certain condition or context. The procedure to follow in order to accomplish the goal is given as a combination of primitive actions and subgoals using plan constructors. Plan constructors specify common programming constructs such as sequential execution, conditional execution, and conditional iteration. Each plan needs a unique name that can be used to distinguish between procedures. As in the goals, associated with each plan is an explicitly or implicitly defined utility value or function, which is used to influence selection of certain procedures over others through the default utility-based selection of plans. A plan may either succeed or fail. Additional separate bookkeeping actions for successful termination and for failed termination of the plan. The sufficiencyof plans for the agent's goals needs to be checked against the goal structure (see Section 2.2) through a classification of the goal and condition of the plans.

2.5 Definition of Agent Interactions

Agents interact throughagent communications. Agent communication requires predefined ontology and agent interaction protocols. In our framework, we adopt a state-based protocol and states use the same representation as that of goals because goals are basically specifies the state to reach. The content of agent communication is also dependent on ontology and agent communication supports XML or SOAP (Simple Object Activation Protocol) type of messages. The interaction structure of agents is again represented using RDF (Resource Description Framework).

3. Rationales

The concept of an agent as an autonomous system, capable of interacting with other agents in order to satisfy its design objective has led to the growth of interest within software designers as a new software engineering paradigm [6]. Agent-based systems are increasingly being applied in a wide range of areas including business process modeling, telecommunications, computer games, military simulations, and intelligent robots. The agent paradigm has an enormous influence on model building and information integration because of the natural capacity of multi-agent systems to design distributed complex systems. Multi-agent systems that are composed of collections of autonomous, interacting agents offers great means to model and simulate real worlds with complex, dynamic, and nondeterministic environments where agents need to function with exogenous events and uncertain effects as well as other agents. Our approach borrows the concept and methodology of multi-agent systems since intelligent robots as autonomous systems have the following properties:

Heterogeneous multiple components: Intelligent robots require integration of multiple functional components such as manipulations, navigations, recognitions, speech understanding and expression, reasoning, planning, and so on. Those components are furthermore heterogeneous in terms of implementation languages and platforms.

Distributed components: The platform for an intelligent robot typically has multiple Single Board Computers (SBCs) to provide enough processing power to meet the real-time requirements and to properly handle parallel and layered robot activities.

Intelligent systems: Intelligent robots are inherently intelligent systems. According to Erman [7] and recitation by Booch [8] in the context of software engineering, intelligent systems differ from conventional systems by a number of attributes as follows:

- They pursue goals which vary over time. Goals form a larger context for the operation of the system. The dynamic nature of that context often makes static algorithms insufficient, requiring the system to exhibit more flexible behavior than conventional systems.
- They incorporate, use, and maintain knowledge. Knowledge includes information stored in traditional knowledge engineering forms, such as

production rules, but also data stored in conventional databases.

- They exploit diverse, ad hoc subsystems embodying a variety of selected methods. The subsystems may be "intelligent" or conventional.
- They interact intelligibly with users and other systems. Intelligibility is one of the most striking attributes of knowledge systems.
- They allocate their own resources and attention. Intelligent systems often need to be introspective and aware of their progress in applying their knowledge and subsystems in pursuit of their goals.

Evolving systems: Service robots are one type of intelligent machine that is designed to interact with human users in relatively unstructured environments [9]. A robot is also commonly defined as "reprogrammable, multifunctional manipulator designed to move material, parts, tools or specialized devices through various programmed motions for the performance of a variety of tasks [10]," emphasizing the "programmable" or "reprogrammable" aspects of robots. Especially the performance goals for an intelligent robot system are continually increasing in complexity. This suggests that the robot architecture should be designed to evolve over time and have new capabilities added to it.

Conventional control systems and architectures are no longer adequate to realize the characteristics of the intelligent systems completely, nor are they sufficient to master the complexity of such systems. Our approach is to design the system as a multiagent systems by ascribing to selected components "autonomous systems" which can act mainly on themselves without external control most of the time. We model the autonomous systems as agents since an agent is a self-contained, concurrently executing thread of control that encapsulates some state and communicates with its environment and possibly other agents via some sort of message passing [11]. Higher degree of flexibility can then be achieved based on a control system consisting of such highly independent modules composed to fit for an application. In our agent-based control system, the resources as well as the agents are designed to operate as autonomous subsystems. Resources are mostly passive or simple active units such as manipulable objects or grippers.

Control of a complex system like intelligent robots can efficiently be realized by successively delegating control competence in hierarchically structured, autonomously operative modules, each of which implements a manageable part of the whole system. The top-most controller of the system is especially built using the BDI model. BDI agents [12] are modeled in terms of beliefs, desires and intentions and believed to provide sufficient level of abstraction to overcome the complexity of intelligent robots.

Our objectives in designing and building a control structure for intelligent robots are summarized as follows:

- *Simple and unified control for heterogeneous components*: Integration of multiple functional components that are developed independently requires the control interface between the components and the controller to be simple and unified. The control interface should be also independent of the language that is used to build the components and the platform where the components run. Ontological commands using command messages instead of application program interfaces (APIs) are much desired to achieve these ends because a single simple API can be used for diverse commands that are agreed upon the ontological commitment.
- *Location transparency:* A common issue of distributed software systems is the allocation of the logical software components of a system to physical nodes. In a distributed system, location transparency allows the resources to be accessed by a user anywhere on the network without concerning where the resource is located. Location transparency is the ability to map a

single specification to a number of different physical environments without requiring changes to the code. Likewise, the various components located in different SBCs need to be accessed without prior knowledge of their location and to operate independently of their locations. The actual locations of components, however, need to be specified in explicit configuration of the system to support location transparency. Location transparency eliminates location as a concern for the developer and the need to overlap of development processes among design, development and the deployment phase. This is likely to produce benefit in the development of intelligent robots because various components are independently developed on their own platform to be integrated later in the development phase.

- *High-level control:* As discussed in Section 3, intelligent robots as intelligent systems are complex systems that should be able to use their computational, sensory, and physical resources to achieve goals and respond to changes in the environment. Such complex systems need higher level of control than the level for the conventional systems. The perceive-reason-act cycle of agent-based approaches combined with the agent-based communication using ACL messages seem to provide an appropriate level of abstraction for high-level control of intelligent robots. An agent model provides a conceptual boundary between the interior and the exterior. The interior of the boundary gives freedom of being independent from other agents or being autonomous. The exterior of the boundary provides a sufficient level of abstraction to other agents so that the complexity of interactions between objects can be reduced by means of the agreement or protocol of interaction between agents. This modularity provided by the notion of agent allows us to develop components independently and easily integrate into the intelligent system. The integration does not require any modification of the system since it uses predefined goal-based interaction instead of interdependent function calls or direct method invocation as in the object-oriented methodology. The goal-based interaction also provides additional benefit for the intelligent systems. Function calls or method invocations in the object-oriented programming tend to generate deterministic outcomes. On the other hand, in our agent-based approach, the desired effect is given to each agent in the form of goals and thus agents are free to choose whatever suitable course of actions to achieve the goal adapting to the dynamically changing situation.

4. Conclusions

Intelligent service robots consist of variety of complex distributed components that are to be integrated into a single robot software system. Agent-oriented software development is a new promising software engineering paradigm to promote the development of such complex distributed software systems. An agent model provides a conceptual boundary between the interior and the exterior. The interior of the boundary gives freedom of being independent from other agents or being autonomous.

In this paper, we presented our rationalesof our agent-based integration approach for building intelligent service robots. Our effort to develop an agent-based architecture for intelligent service robots reflects the need for systematic engineering methodologies to utilize the power of agent technology. We believe that rapid changes in computing environment and the growing industry for intelligent robots will accelerate the use of agent-based approaches.

Acknowledgment

This research was supported by the Intelligent Robotics Development Program, one of the 21st Century Frontier R&D Programs funded by the Ministry of Commerce, Industry and Energy of Korea.

References

[1] Finin, T., Labrou, Y., Mayfield, J.: KQML as an agent communication language. In Bradshaw, J.M., ed.: Software Agents. MIT Press, Cambridge, Mass. (1997) 291 - 316

[2] Klein, M.: XML, RDF, and relatives. IEEE Intelligent Systems 16(2) (2001) 26-28

[3] Kaiser, G., Stone, A., Dossick, S.: A mobile agent approach to lightweight process workflow. In: Proceedings of the International Process Technology Workshop (IPTW), Grenoble, France (1999)

[4] Newell, A.: The knowledge level. Artificial Intelligence 18 (1982) 87-127

[5] Cohen, P.R., Levesque, H.J.: Intention is choice with commitment. Artificial Intelligence 42(2-3) (1990) 213-261

[6] Jennings, N.R., Sycara, K., Wooldridge, M.: A roadmap of agent research and development. Autonomous Agents and Multi-Agent Systems 1 (1998) 7-38

[7] Erman, L.D., Lark, J.S., Hayes-Roth, F.: ABE: An environment for engineering intelligent systems. IEEE Transactions on Software Engineering 14(12) (1988) 1758-1770

[8] Booch, G.: Object-Oriented Analysis and Design. Addison-Wesley (1994)

[9] Pack, R.T.: IMA: The Intelligent Machine Architecture. PhD thesis, Vanderbilt University, Nashville, Tennessee (2003)

[10] Dowling, K.: Robotics: comp.robotics frequently asked questions. http://www.faqs.org/faqs/robotics-faq/ (1995)

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

[12] Rao, A.S., Georgeff, M.P.: BDI agents: From theory to practice. In: Proceedings of the First International Conference on Multiagent Systems, San Francisco, California (1995) 312-319

◐ 저 자 소 개 ◑

이 재 호

1985년 서울대학교 계산통계학과 학사
1987년 서울대학교 대학원 계산통계학과 석사
1997년 University of Michigan, EECS, 공학 박사
1998~현재 서울시립대학교 조교수, 부교수