Design and Implementation of a Robot System Architecture Based on Agent Technology and Delegate Mechanism

Jialin Yu, Yonghua Yan(✉), and Jianrong Zhang

State Key Laboratory of Mechanical System and Vibration,
School of Mechanical Engineering,
Shanghai Jiao Tong University, Shanghai 200240, China
{yjl_sjtu,yhyan,jrzhang}@sjtu.edu.cn

Abstract. The purpose of this paper is to build a robot system architecture based on agent technology and delegate mechanism. Agent technology has penetrated in various fields and delegate mechanism has been widely used to make software architectures scalable. This paper focuses on three aspects: (i) introduction and principle of agent and delegate; (ii) design of a robot system architecture based on agent technology and delegate mechanism; (iii) implementation and verification of the robot system architecture by corresponding experiments. Based on agent technology and delegate mechanism, the robot system architecture can be developed in different developers and period.

Keywords: Agent technology · Delegate mechanism · Robot system architecture

1 Introduction

Etzioni proposed that agent technology was ninety-nine percent computer science and one percent artificial intelligence (AI), so agent technology needs to rely on a lot of basic computer science, such as communication technology and programming language [1]. Over the past few decades, robot systems have been widely used in various fields, while object oriented programming(OOP) has becoming the mainstream programming language. However, the robot systems based on OOP are complicated because of invoking functions from different classes frequently, and the whole architectures is unclear because classes of OOP cannot match the independent robot modules completely. In addition, the most criticized issue is that the objects of OOP are not appropriate to human thinking and unable to keep up with the pace of artificial intelligence because they are static and have no capable of reactivity and pro-activeness. Considering practical applications, such as inserting a simple robot module, the whole program must be analyzed and handled, it costs a lot of time and effort, affects the efficiency of development greatly and increases difficulty of debugging. Therefore, agent technology is used in the design of a robot system architecture in this paper.

H. Liu et al. (Eds.): ICIRA 2015, Part III, LNAI 9246, pp. 116–125, 2015.
DOI: 10.1007/978-3-319-22873-0_11

Agent technology originated from artificial intelligence, the concept of agent had been put out in 1960s and developed in 1990s. Nowadays, agent oriented programming (AOP) has been known and recognized by the majority of programming enthusiasts, it is inheritance and development of object oriented programming. While agent technology is penetrating in various fields and becoming a promising way to develop many complex applications, ranging from electronic commerce to industrial process control[2]. However, there is not a uniform definition about agent[3], Wooldridge defined it as "an autonomous decision making system, which senses and acts in some environment"[4]. In this paper, agent technology is more inclined to be defined as "an autonomous body which runs on the dynamic environment is relatively independent of other bodies and responses by perception of dynamic environment." Although the theoretical study of agent has made great progress, practical applications still exist considerable hysteresis[5]. At present, the agent language is still developing, most of the agent applications are still using Java, C++ and C#, while C++ and platform of Visual Studio 2008 is used to build the robot system architecture in this paper.

With agent technology, there are still some problems. For one thing, the function and size of traditional architectures are set in stone and there is less flexibility in the architectures because they are cannot be trimmed. For another, the traditional architectures are not scalable and are difficult to be configured because the program is protected and encapsulated. If other developers want to develop the architecture, it will cost too much, this is "pull one hair and the whole body is affected". Therefore, delegate mechanism is used here in the design of the robot system architecture. With delegate mechanism, other developers only need to choose configuration or provide new functions to delegate mechanism, the architecture can be personalized and the new functions can be feasible. Delegate mechanism which focuses abstract structure type has been used widely by C# or JAVA language, while delegate mechanism which focuses specific applications is built by C++ language here.

In this paper, the concept of agent technology and delegate mechanism is firstly introduced. And then the robot system architecture based on agent technology and delegate mechanism is designed. Finally, the feasibility of robot system architecture is verified by corresponding experiments.

2 Principle of Agent Technology and Delegate Mechanism

2.1 Principle of Agent Technology

Principle of agent technology is mainly reflected in agent characteristics, agent structure and agent communication modes. There are a lot of agent characteristics, which associated with robot system are autonomy, social ability, reactivity and pro-activeness[6].

1) Autonomy

Agents can start spontaneously and control their own behavior and internal state without the direct intervention of humans or others;

2) Reactivity

Agents can be aware of their environment and respond with changes of environment;

3) Pro-activeness

Agents can actively meet the target behavior.

4) Social ability

Agents can communicate with other agents by certain language and cooperation with other agents to finish the task;

Agent structure shows the operation way inside agent. There are a variety of forms, which is used in the robot system should contain a message queue, a message processor and a lot of agent methods at least[7]. The message queue is used for storing message, which is usually programmed by an array or a list. The essence of the message processor is a thread, which is used to deal with the messages from message queues and invoke agent methods. After agents start spontaneously, the agent processor is detecting messages and environment circularly and constantly. The essence of agent methods are the member function of OOP classes, which are important to finish specific tasks. When the agent processor receives the messages or is aware of changes of environment, it responds and invokes certain agent methods to finish movement[8]. Agent structure diagram is shown in Fig. 1.

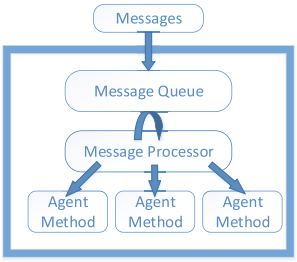

Fig. 1. Agent structure diagram

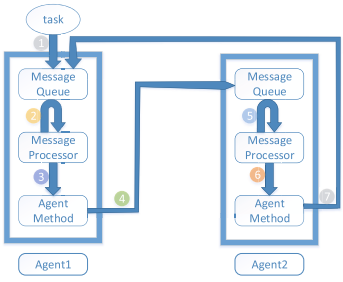

Fig. 2. Diagram of agent communication by message passing model

Agent communication mode decides the way of messages flow between different agents. There are five basic communication modes between agents: no communication model, message passing model, scheme passing mode, the blackboard mode and the agent language model[9]. Message passing model is used in this paper because it is easy and effective to satisfy the needs of robot system architecture design.

Message passing model means that messages are the media between agents, and messages should be formulated and ruled in advance. Only the messages that conform to these rules and formats are useful. When a particular state (for example, the agent get messages from other agents, or the agent is aware of some changes in the environment) happens, agent can make corresponding actions.

The process of agent communication is shown in Fig. 2. Firstly, Agent1 receives the messages from the task, then the messages are stored in message queue, message processor is detecting the messages in message queue cyclically and constantly, when detecting the useful message, message processor invokes corresponding agent method, then the agent method makes operation and sends messages to Agent2, while Agent2's message processor is detecting the messages of message queue, after Agent'2 detects the messages, it invokes corresponding agent method, then the task is finished. After the task has been finished, the agent method of Agent2 sends messages to Agent1, and then Agent1 cannot detect the useful messages any more. Agent1 and Agent2 work together to finish a task, this is the communication process between different agents by message passing model[10].

2.2 Principle of Delegate Mechanism

There is less flexibility in traditional architectures and they are difficult to be developed as they cannot be trimmed, while delegate mechanism in java or C# language focuses on abstract data type. Therefore, delegate mechanism which focuses on specific application about robot system is built in C++ language.

There are a data queue, a serial of registration and initialization functions and a serial of library files in delegate mechanism of this paper. The data queue is in charge of storing the first address of registered functions. Registration functions are charge of providing a way to configure functions to other developers, while initialization functions are in charge of invoking registered functions by invoking the pointers in date queue. The C++ function points are used to finish the process of registration and the process of initialization which are known collectively as process of configuration. The library files are in charge of storing functions that could be registered. In the process of configuration, developers only register the functions needed, and they can register the new functions that are not in library. With delegate mechanism, the architecture is scalable and can be developed.

3 Design of System Architecture

3.1 Design of Agent-Based System

As the agent concept cannot appear in a vacuum, a robot system architecture based on agent technology is designed[11]. There are several functions in traditional robot system architecture, such as configuring parameters, running in different coordinates and running in different model. With the needs of traditional robot system, five agents are designed: Agent Interface, Agent Management, Agent Motion-mode, Agent Interpolation and Agent Controller. In this design, every agent module of the new system has the ability of finishing a simple task[12], while a serial of agent modules can finish a complex task, introduction of every agent function is as follows:

1) *Agent Interface* is the media between the user and machine, which is in charge of providing display and operation for the user.

2) *Agent Management* is in charge of internal data management and external serial management. Internal data consist of reduction ratio, encoder resolution and joint size. External serial management is in charge of reading the value of absolute encoder.

3) *Agent Motion-mode* is in charge of provide several different mode of motion for the user, which includes jog mode, job mode etc. Job motion mode means moving according to job instructions that written in advance. After analysing these instruction, Agent Motion-mode decomposes them into a serial of object points.

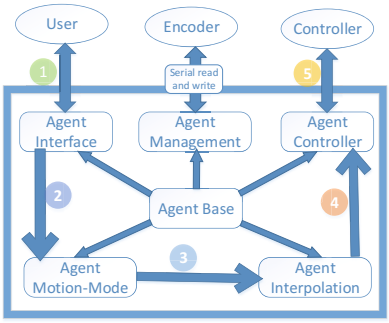

Fig. 3. Structure and information flow of multi-agent

4) *Agent Interpolation* is in charge of receiving the object points from Agent Motion-mode and interpolating into a lot of refined points with professional interpolation method.

5) *Agent Controller* is in charge of managing functions from the controller providers (Controller from GOOGOLTECH company in this paper is used), such as "GT_Open()" and "GT_SetEncPos()".

In this paper, a basic agent module named BaseAgent is designed in advance. BaseAgent has the simplest agent structure: a simple message queue, a message processor that starts spontaneously and cycles continuously and many agent methods that are virtual functions in nature and have nothing meaning. The five agents are derived by the basic module. Fig. 3 is the structure and information flow of multi-agent.

3.2 Improvement with Delegate Mechanism

There are many advantages in delegate mechanism of this paper, the most representative one is that it is feasible to support open and scalable architecture. With the delegate mechanism, both a simplified architecture and a powerful architecture can be built, in other words, an open architecture can be built, which is

a bit like the embedded system. In addition, the robot system architecture is scalable, other developers can replace or add functions according to their needs.

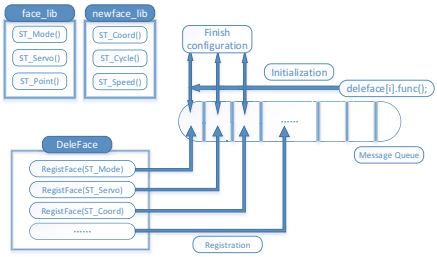

Fig. 4. Process of configuration

In the design, a serial of library files about robot system is built, such as "face_lib", "key_lib" and "Ipolation_lib". There are a lot of functions in every library file, other developers can configure them and get their own architecture. While the developers can write new functions in new library files, such as "newface_lib" in Fig. 4.

A data queue which is programmed by arrays and structures is designed to store the first address of functions. The developers can register functions in library files with the registration functions of the delegate mechanism, and then the first address of functions is written in data queue. There are many registration functions, such as "RegistFace()", "RegistKey()" and "RegistPara()".

After process of registration, these registration functions is configured, which is called process of initialization. There are many initialization functions like "InitFace()", "InitKey()" and "InitPara()".

With process of registration and initialization, the functions registered in data queue are configured one by one. Fig. 4 is the process of interface registration and initialization with the delegate mechanism.

4 Implementation of System Architecture

The process of system program running is shown in Fig. 5. The program begins from the entrance function "OnInitDialog()", and then is the process of registration and initialization, after configuration finished, the five agents start to run, and the message processors are the state of detection, at that time, the user can control the machine to finish movement.

Fig. 6 reflects the information flow of Jog-mode movement. Firstly, the user choose jog mode from the interface and the message "P_Motionmode" is turned to 1(1 means the message is valid), when message processor of Agent Interface detects the changes, it invokes "Change_Mode()" which changes the state to Jog-mode.

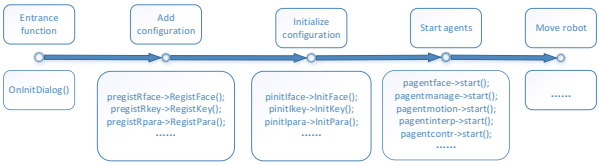

Fig. 5. The process of system program running

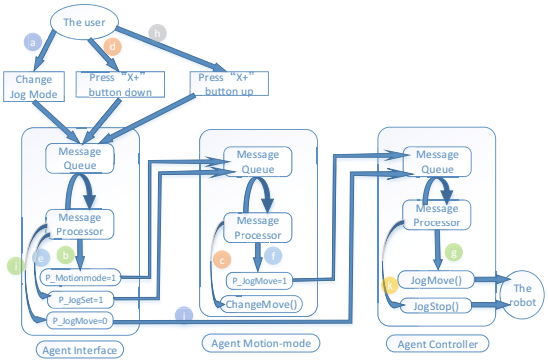

Fig. 6. The information flow diagram of jog movement

When the user pressed the "Y+" button, the message "P_JogSet" is turned to 1, in the same way, the message processor detects the changes and invokes corresponding function to set jog parameter and turns message "P_JogMove" to 1, at last, "JogMove()" is invoked and the robot starts to run in jog mode. When the user press the button up, the message "P_JogMove" is turned to 0, subsequently, "JogStop()" is invoked and the robot is stop.

5 Experimental Studies

5.1 Experimental Setup

The system architecture is used in a six-axis industrial robot based on Windows CE 5.0 , which is shown in Fig. 7. The control cabinet of the robot is shown in in Fig. 8. In order to verify the feasibility of the system architecture, corresponding experiments are carried out.

The experiments collect the working time of every agent when Y-axis is turn from the position in Fig. 9 to the position in Fig. 10 in jog mode. Working time is the time of detecting messages and invoking corresponding functions, which is different from running time. In other words, agents are always in running time after activated, only the agents detect effective messages are in working time.

Fig. 7. Programs of registration

Fig. 8. Programs of initialization

Fig. 9. Programs of registration

Fig. 10. Programs of initialization

5.2 Feasibility Analysis of System Architecture

As shown in Fig. 11, only three agents (Agent Interface, Agent Motion-mode and Agent Controller) receive valid messages and are in working time. The "Power On" button is put down at point a, the span "ab" is the time that Agent Interface detects the message "P_Power" and makes it changed. Subsequently, Agent Controller detects the changed message "P_Power" and makes the power open in span "cd". The time between b and c is 15 milliseconds, which is the time Agent Controller detects the message "P_Power". The jog mode is chose at point e, and Agent Motion-mode detects the messages at point f, and the point g and h is similar in the operation of putting "Y+"button down. These spans reflect the time of detecting corresponding messages, they are usually 5 to 16 milliseconds. When "Y+" button is put up at point m, the movement is stop slowly in span "np". As shown in Fig. 11, the working time of Agent Controller is far more than the others because the most time-consuming operation is finished in Agent Controller.

Fig. 12 is made by collecting the message state of agents and it reflects message state diagram of agents in jog mode. After Agent Controller detects "P_Power" turn to 1(1 means power on message is valid) at point b, the corresponding function makes the power open and "P_Power" is turn to 0 quickly

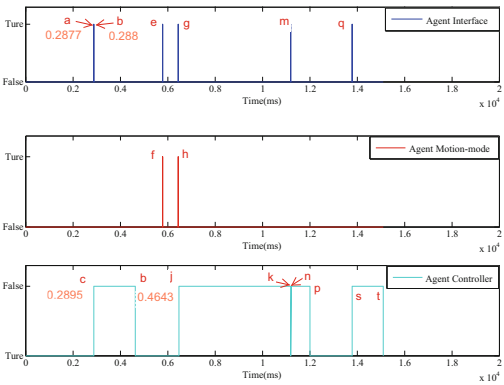

Fig. 11. Sequence diagrams of agents working time in jog mode

in order the operation is done once, similarly, when Agent Controller detects "P_Power" turn to 2(2 means power off message is valid) at point d, the corresponding function makes the power close and "P_Power" is turn to 0.

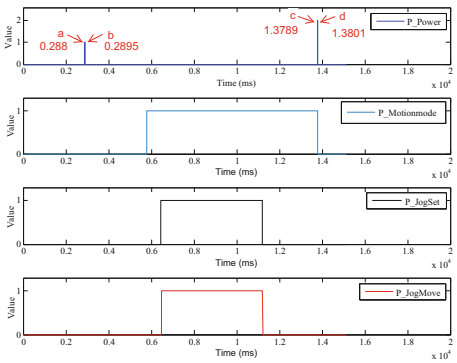

Fig. 12. Message state diagram of agents in jog mode

Although the experiments are simple, they can reflect working time and state of every agent in the process of the system architecture running.

6 Conclusions

Agent technology is a promising development which can provide a solution to many important software problems. In this paper, the robot system architecture combines agent technology with the robot systems creatively, while it is a simplified version of agent technology and cannot finish many things that theoretical

agent technology promises to do. Although the robot system architecture is a small step towards intelligent agent technology, the big step is not far with efforts of other developers. With delegate mechanism, more personalized configuration can be easily completed because the robot system architecture is scalable and can be developed. In the future, the robot system architecture can be widely used in the field of robot systems.

Acknowledgments. This research was supported in part by National Key Basic Research Program of China under Grant 2013CB035804, National Natural Science Foundation of China under Grant 51120155001 and U1201244 and was sponsored by Shanghai Economic and Information Technology Commission (No.CXY-2013-21).

References

1. Etzioni, O.: Moving up the information food chain: deploying softbots on the world wide web. In: Proceedings of the National Conference on Artificial Intelligence, pp. 1322–1326 (1996)
2. Jennings, N.R., Wooldridge, M.: Applying agent technology. Applied Artificial Intelligence an International Journal **9**(4), 357–369 (1995)
3. Bellifemine, F.L., Poggi, A., Rimassa, G.: Developing multi-agent systems with jade. In: Castelfranchi, C., Lespérance, Y. (eds.) ATAL 2000. LNCS (LNAI), vol. 1986, pp. 89–103. Springer, Heidelberg (2001)
4. Wooldridge, M., Jennings, N.R.: Intelligent agents: Theory and practice. The Knowledge Engineering Review **10**(02), 115–152 (1995)
5. Wooldridge, M.: An introduction to multiagent systems. Wiley & Sons **4**(2), 125–128 (2011)
6. Wooldridge, M.: Agent-based software engineering. IEE Proceedings-Software **144**(1), 26–37 (1997)
7. Lee, J., Barley, M., Systems, M.A., Web, S.: Intelligent agents and multi-agent systems. Studies in Fuzziness & Soft Computing **2**(12), 67–96 (2009)
8. Rao, A.S., Georgeff, M.P., et al.: BDI agents: From theory to practice. ICMAS **95**, 312–319 (1995)
9. Luck, M., d'Inverno, M., et al.: A formal framework for agency and autonomy. ICMAS **95**, 254–260 (1995)
10. Olfati-Saber, R., Fax, J.A., Murray, R.M.: Consensus and cooperation in networked multi-agent systems. Proceedings of IEEE **95**(1), 215–233 (2007)
11. Corchado, J.M., Bajo, J., Paz, Y.D., Tapia, D.I.: Intelligent environment for monitoring alzheimer patients, agent technology for health care. Decision Support Systems **44**(2), 382–396 (2008)
12. Maes, P., et al.: Agents that reduce work and information overload. Communications of the ACM **37**(7), 30–40 (1994)