



UNIVERSITY OF
CALGARY

A tutorial report for SENG 609.22

Agent Based Software Engineering

Course Instructor: Dr. Behrouz H. Far

Agent-based System Architecture

Prepared by: Tong Chen

University of Calgary

Department of Electrical and Computer Engineering

2500 University Dr. NW, Calgary, Alberta, Canada, T2N 1N4

E-mail: tche@ucalgary.ca

Abstract

The different system architectures proposed in the literature for agent-based manufacturing systems can be classified into three categories: the Hierarchical approach, the Federation approach and the Autonomous Agent approach. According to our experience, the Autonomous Agent approach is well suited for developing distributed intelligent design systems where existing engineering tools are encapsulated as agents and connected to the system for providing special services, and the system consists of a small number of agents. This type of architecture is also very useful for developing autonomous multiple robotic systems.

1. Introduction

The different system architectures proposed in the literature for agent-based manufacturing systems can be classified into three categories: the Hierarchical approach, the Federation approach and the Autonomous Agent approach. [2]

A typical modern manufacturing enterprise consists of a number of, often physically distributed, semi-autonomous units, each with a degree of control over local resources or with different information requirements. For such a real situation, a number of practical agent-based industrial applications still use the hierarchical architecture, though it may be criticized for its centralized appearance.

Three approaches have been proposed for federation architectures: Facilitators, Brokers and Mediators. In the Facilitator approach, several related agents are combined into a group. Communication between agents takes place always through an interface called a facilitator. Each facilitator is responsible for providing an inter-medium between a local collection of agents and remote agents, usually by providing two main services:

- (1) Routing outgoing messages to the appropriate destinations;
- (2) Translating incoming messages for consumption by its agents.

Brokers (also called broker agents) are similar to facilitators with some additional functions such as monitoring and notification. The functional difference between a facilitator and a broker is that a facilitator is responsible only for a designated group of agents, whereas any agent may contact any broker in the same system for finding service agents to complete a special task.

The Mediator approach is another type of federation architecture. In addition to the functions of a facilitator and a broker, a mediator assumes the role of system coordinator by promoting cooperation among intelligent agents and learning from the agents' behavior. A detailed description of the mediator concept and architecture can be found in (Gaines et al 1995).

Federation multi-agent architectures (the Facilitator approach, the Broker approach and the Mediator approach) are able to coordinate multi-agent activity via facilitation as a means of reducing overheads, ensuring stability, and providing scalability.

The Autonomous Agent approach is different. Although different definitions have been proposed for autonomous agents, we argue that an autonomous agent should have at least the following characteristics:

- (1) It is not controlled or managed by any other software agents or human beings;
- (2) It can communicate/interact directly with any other agents in the system and also with other external systems;
- (3) It has knowledge about other agents and its environment;

(4) It has its own goals and an associated set of motivations.

2. Concepts

Open Agent Architecture: A framework for building distributed software system. [1]

ATAM: Architecture Tradeoff Analysis Method, a method for evaluating software architecture, can be used in agent-based system analysis. [4]

3. Case Study

Designs for real-world agent-based systems must reflect both domain requirements and technical capabilities. We illustrate some of the requirements for agents in manufacturing scheduling and an architecture that addresses them with a case study of AARIA (Autonomous Agents for Rock Island Arsenal), an industrial-strength agent-based shop-floor control and scheduling architecture being developed for an Army manufacturing facility.

3.1 System Requirements

Requirements Definition defines the problems that the project must solve. The focus is on why an effort is needed in the first place, not on what the project will do or how it will do it. This section outlines the problem we are addressing, and then discusses requirements in two categories: Those imposed by the system's interface with its external environment, and those needed in its internal operations to achieve necessary functionality.

3.1.1 Shop-Floor Scheduling and Control

A manufacturing enterprise is measured by the cost of the goods it produces, their quality, and the timing of their availability relative to the customer's need. The task of shop floor scheduling and control is to deploy resources to produce high-quality goods as inexpensively as possible and when the customer wants them. It governs decisions such as when which machines should be used for which products, what order products should be manufactured, when new jobs should be started, what level of inventory should be carried, and when machine maintenance should be performed. The problem is the subject of extensive research in the industrial engineering and operations research communities. Because of its commercial potential, researchers in agent-based systems find it increasingly attractive as an application problem.

These specific requirements were identified on the basis of detailed discussions with manufacturing staff at Rock Island Arsenal and at various commercial manufacturing facilities, taking into account our knowledge of the functionality of commercial systems and systems documented in the literature. We have drawn from our awareness of the functionality that people have demonstrated with previous agent systems. While these requirements are not specific to an agent-based manufacturing system, this

paper demonstrates how agents can support them.

3.1.2 External Interfaces

- ◆ Support of Customer/Supplier Interaction
- ◆ Support of Operational Personnel and Equipment
- ◆ Support of Manufacturing Engineers and Management

3.1.3 Internal Operations

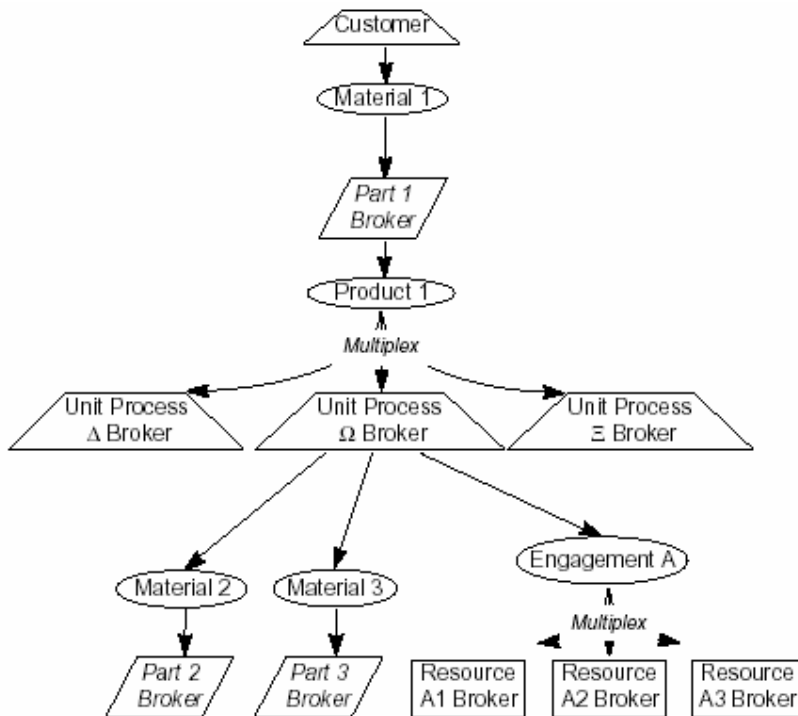
- ◆ Frequent Change
- ◆ ERP Functionality
- ◆ Different Factory Control Modalities
- ◆ Metamorphosis
- ◆ Uniform Internal/External Presentation

3.2 Refinement

The Requirements developed earlier enable us to refine this preliminary set of agents:

- ◆ Implications of Uniformity
- ◆ Implications of Metamorphosis and Least Commitment
- ◆ Implications of Modality Emergence
- ◆ Implications of Frequent Change
- ◆ Implications of Empowerment
- ◆ Implications of ERP Functionality

3.3 An Example of AARIA in Action



Though this example is initiated by a Unit Process Broker (a Customer) by way of a Material agent, the architecture is flexible enough to permit any persistent agent to initiate action, under appropriate circumstances.

- A Unit Process Broker may initiate activity through a Product agent instead of a Material agent, thus signaling the Part Broker for its output that it would like to produce. Such an action might result from an advance schedule imposed by management that calls for the operation to run at a given time, and would be plausible in a make-to-stock operation.
- A Part Broker may spawn a Product agent if its inventory falls below a specified value, thus imposing a pull dynamic on its Suppliers. It should also be able to spawn a Material agent if its inventory rises above a certain level, thus actively seeking out consumers for its stock. If the price it offers for excess inventory is low enough, it may be able to find Unit Process Brokers willing to consume from it.
- A Resource Broker may also initiate processing by spawning an Engagement with a selected operation. This mode of operation may characterize a bottleneck resource, which thus offers services on a “Don't call me; I'll call you” basis. [3]

4. Methodology of Agent-based System Architecture Analysis

4.1 Architecture Tradeoff Analysis Method for Software Systems [4]

In large software systems, software qualities such as performance, security, and modifiability depend not only upon code-level practices (e.g., language choice, detailed design, algorithms), but also upon the overall software architecture. A variety of qualitative and quantitative techniques are used for analyzing specific quality attributes. These techniques have evolved in separate communities, each with its own vernacular and point of view, and typically have been performed in isolation. However, the attribute-specific analyses are interdependent; for example, performance affects modifiability, availability affects safety, security affects performance, and everything affects cost.

The Architecture Tradeoff Analysis Method (ATAM) is an evolving architecture evaluation technique. The input to the ATAM consists of system architecture and the perspectives of stakeholders involved with that system. The ATAM relies on generating scenarios to assess the architecture, where the scenarios are stimuli used to represent stakeholders' interests and to understand quality-attribute requirements. The scenarios give specific instances of anticipated use, performance requirements or growth, various types of failures, various possible threats, various likely modifications, etc.

ATAM, by itself, does not provide an absolute measure of architecture quality; rather it serves to identify scenarios from the point of view of diverse stakeholders (e.g., the architect, developers, and users) and to identify risks (e.g., inadequate performance, successful denial-of-service attack) and possible mitigation strategies. The results of an ATAM evaluation reflect the concerns of stakeholders and the scenarios that they consider important. In section 4, we identify typical quality attributes, scenarios, and

agent-system architectural patterns as a way to expedite evaluations of real agent systems.

4.2 Architecture Tradeoff Analysis Method for Agent Systems

Very broadly, agents may be thought of as software entities with the ability to undertake action autonomously in a specific embedded environment. Typically, the autonomous behavior is performed according to a general set of requests or desired goals. Additionally, agents are considered capable of communicating with other agents as determined by their own initiative. In this section, we illustrate the nature of the architecture decisions and their impact on various qualities attributes of agent systems.

A number of publications on the subject of agent-based systems contain typical design patterns for collaborating agents. However, critical aspects of evaluating agent-based systems in terms of software quality attributes are not presented. We see the following set of attributes as particularly relevant to agent architecture evaluation:

- Performance predictability. Since agents have a high degree of autonomy, it is difficult to predict, from a high level of abstraction, individual agent characteristics (e.g., timeliness and latency) when determining the behavior of a system at large.
- Security against data corruption and spoofing. Agents are free to identify their own data sources (e.g., Web agents). In addition to possibly misleading information acquired by agents, there is the distinct possibility of hostile agents attempting to spoof the system agents. The protocols of verifying authenticity for data sources by agents are a concern in the evaluation of system quality.
- Adaptability to changes in the environment. Agents may be required to adapt to modifications in their environment, such as changes to the agent's communication language.
- Availability and fault tolerance. Agents that offer services to other agents may make promises about the availability of the offered services. A related attribute is the ability of the agent system to detect, contain, and deal appropriately with external agents such that erroneous inputs are not propagated into the agent system itself.

5. Conclusion

According to our experience, the Autonomous Agent approach is well suited for developing distributed intelligent design systems where existing engineering tools are encapsulated as agents and connected to the system for providing special services, and the system consists of a small number of agents. This type of architecture is also very useful for developing autonomous multiple robotic systems.

In my survey, the Autonomous Agent approach is more broadly used in business cases. According the features of it:

- (1) It is not controlled or managed by any other software agents or human beings;
- (2) It can communicate/interact directly with any other agents in the system and also with other external systems;
- (3) It has knowledge about other agents and its environment;
- (4) It has its own goals and an associated set of motivations.

I think in the research area of agent-based system, we should pay more attention to this approach. Its architecture possesses more flexible, intelligent, autonomous, and powerful ability.

Because all these judgments come from simple survey and experience, we should find some reasonable, reliable and systematic methodology to analyze the architecture of agent-based system. ATAM is an appropriate and relatively mature methodology to evaluate which approach of agent-based system is better. As future work, we should do some evaluation and analysis for agent-based system architecture, instead of instinctive, experienced estimation.

It is an assumption that formal method also would validate that Autonomous Agent approach is prior other approaches in most aspects.

6. Acknowledgements

I would like acknowledge Dr. Far for giving me enough help, suggestion, and support when I am in very difficult days. In fact, in these days, Dr. Far gives me help both in actual and mental aspects. Or I might give up something, which I should not give up. I also would like thank Dr. Far provide knowledge and method in agent-based system software engineering.

At same time, I would like acknowledge my project group members: Edidiong Ekaette and Cathy Wei Wu, they gave me lots of help in the course project.

Reference

- [1] Dr. Far, SENG 609.22 Course slides and papers
- [2] Weiming Shen and Douglas H. Norrie, “Agent-Based Systems for Intelligent Manufacturing: A State-of-the-Art Survey”
- [3] Eleni Kaldoudi, Marcos Zikos, E. Leisch, Stelios Orphanoudakis, “PLEGMA: An Agent-Based Architecture for Developing Network-Centric Information Processing Service”
- [4] Nenad Ivezic, Mario Barbacci, Don Libes, Tom Potok, and John Robert, “An Architecture Analysis of Agent-based Systems”
- [5] H. Van Dyke Parunak, Albert D. Baker, Steven J. Clark, “The AARIA Agent Architecture: From Manufacturing: Requirements to Agent-Based System Design”