Formal Framework for Adaptive Multi-Agent Systems

Wenpin Jiao  Minghui Zhou  Qianxiang Wang
Institute of Software, School of Electronics Engineering and Computer Science
Peking University, Beijing 100871, China
{jwp, zhmh, wqx}@cs.pku.edu.cn

Abstract

Evolution of software challenges multi-agent systems to be adaptive and the adaptation capabilities should be provided at both the agent level and the system level. In this paper, we study the adaptation of multi-agent systems from the system level and describe a formal framework for multi-agent systems with adaptation capabilities. In the framework, agents are indirectly inter-connected via roles and interaction protocols are defined as entities independent of agents and roles. Thus agents, roles, and interaction protocols involved in multi-agent systems need not always adjust their behaviors or specifications to adapt to the changes of system structures and interaction protocols.

1. Introduction

With the widespread of the Internet, software systems become more and more open and dynamic. Not only the structures but also the behaviors of software systems are evolving all the time. For instance, in multi-agent systems, agents may join in (or withdraw from) the systems timely; which may result in the evolution of structures; and meanwhile interactions among agents may change dynamically, which will be reflected in the evolution of the behaviors of agents. These dynamic characteristics challenge multi-agent systems to be adaptive and flexible.

For a multi-agent system, adaptation represents the ability of the multi-agent system to recognize and response to unanticipated internal and external changes, which is a key capability for the system to improve and adjust its behavior to the evolution.

In [9], evolutions are classified into two categories, knowledge and structure evolution. From the point of view of agents involved in a dynamic evolving multi-agent system, evolutions are actually about changing of agents themselves and their environment.

To adapt to the evolutions, multi-agent systems should make adjustments correspondingly, such as reorganizing, adding/removing agents, redefining interaction protocols, and so on.

From our perspective, a multi-agent system with good adaptability should at least satisfy the following criteria while adjusting its structure or behavior.

1) Transparency. Some adaptations of agents, for instance, the changes of an agent’s internal state, should be transparent to other agents.

2) Impact localization. If the adaptation does not result in reorganizing the whole system’s structure, the behaviors of agents who are not related with the adaptation should be unaffected.

Currently, there has been much research on agent adaptability [4][5][6]. To embody adaptation in a multi-agent system, the techniques they used are usually based on machine learning or decision-making [2], but many of them are focused on centralized processes to formulate models or strategies.

We believe that adaptation should be provided at both the agent level and the system level and further that adaptation at the system level will minimize the impact of system level adjustments on individual agents.

In this paper, we study the adaptation of multi-agent systems at the system level.

- We employ a type of dynamic role-filling mechanism to model the adaptation, in which agents interact via playing roles. Thus if roles involved in the system are not altered, the join, withdrawal, or replacement of agents will not affect the behaviors of other agents.

- Meanwhile, we define interaction protocols as entities independent of roles and agents. Then agents need not to adjust their behavior specifications even if interaction protocols have changed.

To describe the adaptation framework for multi-agent systems, we use a variant of the standard pi-calculus, the polyadic pi-calculus [9], which is very suitable for describing software with dynamic configurations.

In the pi-calculus, processes are active components of a system and they communicate with each other through ports (or names) connected via channels. The processes in the pi-calculus have the following forms.
The main features of the polyadic pi-calculus different from the (monadic) pi-calculus are that prefixes $x(y)$ and $\bar{x}y$ are defined on vectors. We use $x$, $y$, ... to stand for vectors of names, with length $|x|$, $|y|$, ... . The computation between processes is defined by the following communication reduction rule.

$$x(y) \cdot P, \bar{x}z \cdot Q \rightarrow^{n(\bar{z}/\bar{y})} P(x' \bar{y}) \cdot Q',\text{ where } |y|=|z|.$$

In the following sections, Section 2 describes the core conceptual model of the framework for multi-agent systems and some concepts related to the model. And then Section 3 formally defines the framework and its components. In the last section, we conclude our work.

2. Concepts

2.1. Agent

By now, there is no common accepted agent definition. However, an agent is often considered as an active communicating entity, which has many features, such as autonomy, pro-activity, and social ability, and can provide services via interactions with other agents or the environment [7].

2.2. Role

In multi-agent systems, the role focuses on the position and responsibilities of an agent within an overall structure or system. A role can be characterized as a collection of responsibilities, permissions, and interaction protocols. Every role involved in a multi-agent system must be handled by at least one agent. And each agent can play one or more roles [7].

2.3. Interaction Protocol

An interaction protocol indicates what roles are involved in the interaction, how they interact with each other step by step, and what content they transmit while interacting. While implementing their responsibilities via cooperation, roles must adapt at least one interaction protocol.

2.4. Core Conceptual Model

The core conceptual model for our adaptive multi-agent systems is shown in figure 1.
• A role must be played by some agent. Thus, if there exist a role and an agent playing the role, when the agent withdraws from the system, there must be another agent taking the place of the withdrawn agent to handle the role.
• One role has only one responsibility and uses only one interaction protocol.
• If an interaction protocol is used by two different groups of roles, there should be two instances of the protocol. For simplicity, we would rather consider the two instances as two different interaction protocols. Thus, every use of an interaction protocol has a corresponding entity and a unique identification.

3. Formal Framework of Adaptive Multi-Agent Systems

In this section, we will formalize those components occurring in the conceptual model in the polyadic pi-calculus and then use them to form a formal definition for multi-agent systems with adaptation. While defining the pi-calculus processes, we let $a_1, a_2, \ldots$ range over identifications of agents, $r_1, r_2, \ldots$ represent roles, and $p_1, p_2, \ldots$ denote interaction protocols. Then prefixes such as $a(x), r(x), p$ and their co-names have the following denotations.

- $\overline{a(r)}$ denotes that agent $a$ plays role $r$ and $\overline{a(x)}$ is to query which role agent $a$ plays.
- $\overline{r(a)}$ means that role $r$ is played by agent $a$ and $\overline{r(x)}$ is to query by which agent role $r$ is played.
- $\overline{p}$ is used in the initiator role to activate protocol $p$ whilst $\overline{p}$ is used in other roles to respond the run of the protocol.
- Conventionally, $s$ is like a pointer to a Service provided by an agent and $\overline{s}$ is the reference to the pointer. Similarly, $z$ is like a pointer to an Action defined by an interaction protocol and $\overline{z}$ is the reference to the pointer.

3.1. The Role Assignment Manager (RAM)

The responsibilities of $\text{RAM}$ include:
- To assign roles to agents.
- To answer queries about what role an agent is handling.
Formally, suppose that $\text{RAM}$ assigns $r_1$ to $a_1$, $r_2$ to $a_2$, and so on, $\text{RAM}$ can be defined as the following pi-calculus process.

$$\text{RAM} = \overline{a_1(r_1)} | \overline{a_2(r_2)} | \cdots$$

3.2. The Interaction Protocol Manager (IPM)

$\text{IPM}$ is to gather all participants to execute an interaction protocol when the interaction protocol is requested to run by the initiator role.

Suppose the interaction protocol $p_1$ is one of the protocol managed by $\text{IPM}$ and it involves $N$ participants, $r_1, r_2, \ldots, r_N$. Then $\text{IPM}$ can be formally defined as a polyadic pi-calculus process, in which a concretion of the interaction protocol process is included, as follows.

$$\text{IPM} = \overline{(p_1 \cdot p_2 \cdot \ldots \cdot p_N \cdot \{r_1, r_2, \ldots, r_N\})}$$

Intuitively, $\text{IPM}$ always waits for a request from the initiator role (via $p_1$) to activate the protocol. When $\text{IPM}$ is sure that all participants are present and ready, it will activate the protocol through the concretion $[r_1, r_2, \ldots, r_N]$.

3.3. Agents

In our formal model, agents are to play roles and provide services to implement the responsibilities of roles, and agents need not care about how to interact with other agents, which is decided by roles they play.

Suppose there is an agent $a$ to provide a series of services such as $S_1, S_2, \ldots, S_n$ and each service will be used to implement a responsibility of some role. Then to provide services via roles, the agent should know which roles first what roles it plays and then how the services are linked to responsibilities. Agent $a$ can be formally defined as the following pi-calculus process.

$$A = (x, s)a(x) \cdot x(s_1) \cdot s_1 \cdot S_1 \cdots$$

where the agent first obtains the role assigned by $\text{RAM}$ and then outputs a pointer to a service. The pointer to the service will be connected to the responsibility of the role.

3.4. Roles

In our framework, a role is composed of three components.

- A responsibility which will be implemented as a service provided by an agent.
- An interaction protocol used while fulfilling its responsibility.
- Actions related to the interaction protocol.

Suppose that role $r$ will be involved in an interaction protocol, $p$, and acts as the initiator of the protocol (or the first party involved in the protocol). Then role $r$ can be formally defined as the following pi-calculus process.

$$R = ((x, r)X \cdot x \cdot \overline{p}) \cdot \text{Step} \cdot \overline{r}$$

where the role first obtains the agent appointed to play it by $\text{RAM}$ and then gets the pointer to a service to
implement its responsibility. Meanwhile, the role notifies, via $p$, $IPM$ to activate the interaction protocol $p$ and then waits for instructions of interactions step by step from the interaction protocol. $z_r$ is used to get a pointer to the interaction instruction of next step from the interaction protocol.

If role $r$ acts as a normal participant instead of the initiator of the protocol, role $r$ can be formally defined as follows.

$$R = ((\forall x)p(r(x) \cdot z_r | p \cdot !Step) \cdot R$$

$$Step = z_j$$

where the subscript $j$ occurring in $z_j$ indicates that role $r$ is the $j^{th}$ ($j > 1$) party involved in the protocol.

### 3.5. Interaction Protocols

An interaction protocol must be related to and run by roles, and similarly, the interaction protocol can be instantiated many times sequentially. Suppose the interaction protocol $p$ involves $N$ participants, $r_1, r_2, \ldots, r_N$, in which $r_1$ acts as the initiator, and $M$ steps. Then the protocol can be formally defined as an abstraction of a polyadic pi-calculus process.

$$P = (\forall x)(z_1 | Step_1 | z_2 | Step_2 | \cdots | z_m | Step_m)$$

$$Step_j = z_j | Act_{z_j} | z_{j+1} | Act_{z_{j+1}} | \cdots | z_m | Act_{z_m}$$

where each step of the protocol defines a collection of actions and by referring to the pointers (i.e., $z_1, z_2, \ldots$), the protocol tells each participant how to act respectively.

### 3.6. Multi-Agent Systems

By combining all components described in the conceptual model, we can formally define a multi-agent system as a composition process of those pi-calculus processes defined previously.

Suppose there are $N$ roles involved in the multi-agent system, $M$ ($M \leq N$) agents to play these roles, and $K$ ($K \leq N/2$) interaction protocols related with the roles. Then the system can be defined as follows.

$$MAS = A_1 | \cdots | A_m | RAM | R_1 | \cdots | R_n | IPM | P_1 | \cdots | P_k$$

### 4. Conclusions

To develop flexible complex systems, we must cope with rapid and unforeseeable changes of software systems themselves and their environment. Nowadays, many researches have focused on the adaptations of software systems and many methods and frameworks have been proposed from different perspectives to model and design systems.

However, most of research work is processing in the field of component-based software development [1] [3]. In this paper, we studied the adaptation of multi-agent systems starting from the system level. Once a multi-agent system is with the capability of adaptation, entities (including agents, roles, and interaction protocols) involved in the system need not always adjust their behaviors and specifications to adapt to the changes of the system.

Nevertheless, in our current adaptation framework for multi-agent systems, we do not take the adaptation and internal computation capabilities of individual agents into consideration. At next stage, we will study to blend the two level adaptations into a uniform framework and further explore the properties of the framework.

### Acknowledgement

This work is partially sponsored by National “973” Program (granted no. 2002CB312000), National Natural Science Fund, and State 863 High-Tech Program (granted No. 2001AA110236).

### References


