FRAMEWORK FOR HIERARCHICAL MOBILE AGENTS: TOWARD SERVICE-ORIENTED AGENT COMPOUND

Fuyuki Ishikawa
Graduate School of Information Science and Technology, The University of Tokyo
7-3-1 Hongo, Bunkyo-ku, Tokyo 113-8654, Japan

Nobukazu Yoshioka, Yasuyuki Tahara, and Shinichi Honiden
National Institute of Informatics
2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo 101-8430, Japan

Keywords: Hierarchical mobile agent, Agent system, Agreement, Service oriented architecture, Multimedia application.

Abstract: Hierarchical mobile agent model is an emerging extension to traditional agent models, where an agent can migrate into another agent and establish a strong partnership. The entering agent and accepting agent are called as a *child* and its *parent*, respectively. This model enables agents to establish the strong (parent-child) partnership and cooperate with each other while maintaining the partnership stable for a long term. This work discusses requirements for frameworks that support development of HMA applications, and describes the basic direction of our proposed HMA framework named MAFEH. MAFEH includes two features: (1) Parent-Child Agreement that specifies an agreement regarding behaviors of a parent and its child, and (2) Interaction Behaviors Description that is used to specify typical synthesis actions separated from the main application logic. Supporting these features, MAFEH aims to facilitate development of an agent system where each agent can obtain required functions on demand by means of synthesis as well as remote interaction. This work considers multimedia application as a motivating scenario, where an agent encapsulates a multimedia content and establishes synthesis with various agents encapsulating other contents or additional services.

1 INTRODUCTION

Agent technologies have attracted widespread attention, orienting towards adding more autonomy and flexibility to software components, especially distributed conpoments interacting and cooperating with each other (Bradshaw, 1997). Mobility of agents has also been investigated, i.e. the ability to migrate from one host to another on its own decision. Mobility has been considered as a powerful way to facilitate flexible, on demand deployment of agents adapting to the surrounding environments e.g. heterogeneity of computing/network resources and movement of users in pervasive computing.

Hierarchical Mobile Agent (HMA) model is an emerging extension to traditional agent models, where an agent can migrate into another agent (Suna and Fallah-Seghrouchni, 2004; Satoh, 2000; Tahara et al., 2003)[1] An agent can enter into another agent or to take one in that provides additional services, and form an agent compound. In this paper, we call the entering agent and accepting agent as a *child* and its *parent*, respectively, and use such kinship terms. As the most notable point in the HMA model, when an agent migrates into an agent or a host, it takes all its descendants along. The HMA model thus enables agents to establish a strong partnership as an agent compound.

The HMA model enables function-migration as well as data-migration as the mobile agent model does. It thus can be used for adaptation to limitation in computing/network resources in mobile computing or in multimedia processing. The HMA model extends the mobile agent model in a sense agents can maintain their partnership stable for a long term, even if the agents migrate around.

In addition, the HMA model can be considered as one way to implement Service-Oriented Architecture (SOA) (Haas, 2004). In SOA, software systems provide services to other applications through published and discoverable interfaces, and the services can be invoked over a network. The HMA model also considers dynamic discovery and coodination of services provided by agents. Moreover, it additonally enables agents to establish stronger partnership by keeping being together with service instances as an agent, if

[1]Currently, there is no widely-accepted name for the model. We follow (Satoh, 2000) for the name HMA. We follow (Satoh, 2000) for the use of kinship terms, too.

Ishikawa F., Yoshioka N., Tahara Y. and Honiden S. (2005).
FRAMEWORK FOR HIERARCHICAL MOBILE AGENTS: TOWARD SERVICE-ORIENTED AGENT COMPOUND.
In *Proceedings of the Seventh International Conference on Enterprise Information Systems*, pages 275-282
DOI: 10.5220/0002529102750282

preferred, not for one-shot invocation but for long-lasting interaction.

Encapsulation of multimedia contents by agents has been considered as an attractive application of the HMA model. Agents encapsulate multimedia contents, form a hierarchy corresponding to the hierarchy of the contents, and achieve presentation, delivery, and composition suitable for the user's situation while obeying policies given by their creators and distributors (Satoh, 2001; Tahara et al., 2003).

Although there seem to be considerable advantages of the HMA model, little effort have been conducted on it. Our objective is to provide a framework that supports development of attractive HMA applications by addressing its unique issues.

The existing HMA frameworks provide only primitive functions for agents to enter into/exit out of another agent and to exchange messages (Suna and Fallah-Seghrouchni, 2004; Satoh, 2000). These functions are derived by directly adopting the mobile agent model and extending typical mobile agent frameworks. Although this enables developers to achieve various behaviors by combining the functions, it leads to complicated codes of low-level functions, where various kinds of behaviors, e.g. discovery and migration, are weaved into the application logic, resulting in difficulty in development and maintenance. In additon, it has not been discussed what behaviors are typical and should be easy to achieve in the HMA model, not in the mobile agent model. It has not been discussed, too, how to support developers to achieve consistent and cooperative behaviors of agents to form an agent compound and work together.

This paper discusses two additional requirements for HMA frameworks: (1) easy control of typical flow upon interaction with a partner agent, i.e. discovery and migration (if preferred), and seperation of it from the main application logic, and (2) support for controls/restrictions over a child's behaviors by its parent. This paper also describes our approach to these requirements and introduces two concepts: Interaction Behavior Description (IBD) and Parent-Child Agreement (PCA) in response to (1) and (2), respectively.

IBD is used by developers to control typical flow mentioned above by setting parameters. IBD is added to the application logic, which specifies how to interact with partners dynamically bound in runtime but does not specify concrete information of them.

PCA denotes an agreement between a child and its parent for strong controls/restrictions over a child by its parent, such as encapsulation of the child or confinement of the child. Before synthesis, the agents establish a PCA and then behave according to it until dissolution of the synthesis. In this work as the first step, we focus on an easily-built agreement concerning on controls/restrictions of domain-independent primitive actions supported by frameworks, such as communication and migration.

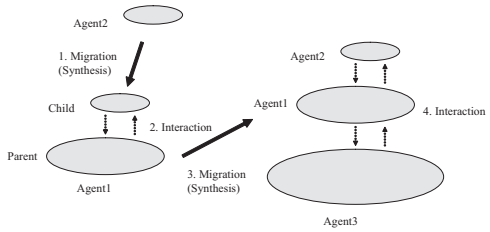

Figure 1: Hierarchical Mobile Agent Model

This paper is a progress report of our challenge and describes the basic direction of our ongoing work on the HMA framework named MAFEH [2], which includes IBD and PCA as its main features. Our work aims to facilitate development of an agent system where agents can request for additional functions on demand and work cooperatively, by synthesis as well as remote interaction. As a motivating scenario, our work considers multimedia application where an agent encapsulates a multimedia content and forms synthesis with various agents encapsulating other contents or services.

In the remainder of this paper, we first introduce the HMA model and discuss requirements for HMA frameworks with the motivating example scenario in Section 2 and 3. We then describe the basic direction of the proposed framework in Section 4. Finally we give some discussion in Section 5 before concluding remarks in Section 6.

2 HIERARCHICAL MOBILE AGENTS

This section introduces the Hierarchical Mobile Agent (HMA) model.

2.1 Basic Concepts

Figure 1 illustrates the HMA model (Suna and Fallah-Seghrouchni, 2004; Satoh, 2000; Tahara et al., 2003), where an agent can migrate into another. The migrating agent and the accepting agent are called a *child* and its *parent*, respectively. An agent can have only one parent and any number of children, forming a tree structure. When an agent migrates into an agent or a host, it takes all its children along (migration of agent1 in Fig. 1). The HMA model enables agents to establilsh a strong partnership and work cooperatively for a long term forming an agent compound.

We claim inter-agent migration should be divided conceptually into two phases: inter-host migration

[2]Mobile Agent Framework with Enhanced Hierarchy.

where one agent migrates to the same host of the other, and the action where they establish a parent-child relationship. This work refers to the latter as *synthesis* and focuses on it.

2.2 Existing Works

SyMPA (Suna and Fallah-Seghrouchni, 2004) is an HMA platform inspired by Ambient Calculus (Cardelli and Gordon, 1998), a calculus for concurrent process forming hierarchy. SyMPA supports basic functions from the calculus for inter-agent migration. However, SyMPA has not focused on how to make use of the HMA model.

MobileSpaces (Satoh, 2000) is a framework that provides basic functions for the HMA model. Multimedia applications on MobileSpaces have been proposed (Satoh, 2001). Agents encapsuate multimedia objects and form a hierarchy, which corresponds to the hierarchy of the multimedia objects, e.g. images included and managed by a word processor document. The agents provide to users, customized functions to operate multimedia objects (play, edit, etc.). In addition, required services can be added at runtime as child agents, e.g. streaming functions.

Active Contents project (Tahara et al., 2003) also proposes encapsulation of multimedia contents in the HMA model. It additionally considers agents that contain multiple content agents and (1) integrate them to provide a united content or (2) provide flexible delivery functionality by means of migration. This model has been formalized using Ambient Calculus, however, implementation issues have not been addressed.

Note that for synthesis control, the existing frameworks (SyMPA, MobileSpaces) provide functions only to enter into/exit out of another agent (either on the same host or on a different host), as an extension to inter-host migration.

3 MOTIVATION

In this section, we first describe the motivating example application. We then discuss additional requirements for HMA frameworks.

3.1 Example Application

In Section 2.2, we have mentioned some multimedia applications of the HMA model. Agents encapsulating multimedia contents and related services in the HMA model are attractive for the following reasons.

Multimedia data cannot be organized like text data. There exist many formats and functions for each format such as playing, editing, and translating formats. There also exist specialized functions to extract useful metadata or provide users customized view. The HMA model enables adoption of functions on demand, without preparing all in advance. Since multimedia data is typically large in size, it is inadequate to send data to remote servers with rich functions. The HMA model enables function migration as well as data migration. Moreover, multimedia data have a hierarchy, e.g. an image in a word-processor document. Presentation of children is controled by their parent and whole contents are delivered as a unit. This is equivalent to the HMA model and the model adds the abilities to move or establish partnerships autonomously. We believe agent technology will help address issues in content synthesis, such as confliction of licenses or presentation policies, as well as systematic issues tackled in this work.

As the motivating example scenario, this work also considers an agent encapsulating a video content (VCA: Video Content Agent). The VCA is given presentation, composition, and delivery policies by creators, editors, and distributors of the contents, repectively. It provides the contents in a way compliant with the policies. For example, it can control how many times the user can move the contents as well as until when the user can play. It can also increase such licenses after the user answers a questionnaire. Such control can be achieved without connecting a central management server, without the need for such functions to be installed in advance. In addition, it can provide value-added services such as customization of contents presentation, merge of related information collected from the Web, and creation / edit management by migrating among hosts of software services or human professionals.

Our concern here is not issues in handling multimedia contents but control of agent synthesis. Figure 2 shows an overview of the VCA lifecycle where it interacts with four types of agents.

Advertiement Agent Upon instantiation of the VCA, it takes in an agent as a child that encapsulates advertisements. The VCA then presents the advertisements integrated to its own video content.

User Agent The VCA migrates to the user host and starts interaction with an agent that represents the user.

Mobile Streaming Agent The VCA provides mobile streaming, i.e. tracking the user and providing streaming at a host nearby the mobile user. To achieve this, the VCA enters into an agent that provides migration and streaming ability.

Privacy Agent To provide presentation customized for the user, the VCA enters into an agent that imposes strong restriction on its behaviors, e.g. outbound communication, in return for the user's information.

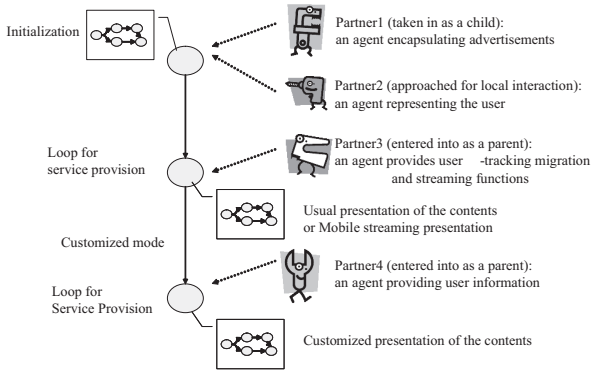

Figure 2: Agent Encapsulating Video Content

3.2 Requirements for HMA frameworks

For the unique issue in the HMA model, i.e. synthesis, the existing frameworks (SyMPA, MobileSpaces) provide functions to enter into/exit out of another agent (either on the same host or on a different host), as an extension to inter-host migration. Synthesis control functions can be classified into two types: (1) building or dissolving a parent-child relationship and (2) cooperation of between a child and its parent while the relationship is being maintained.

3.2.1 Synthesis Control

For (1), the existing frameworks (Suna and Fallah-Seghrouchni, 2004; Satoh, 2000) only provide primitive functions to enter into/exit out of another agent. This fact causes the following two issues. First, it means that the frameworks would require a heavy task of coding, for each partner agent, regarding discovery of a partner instance that can play the required role, migration and synthesis if required, and dissolution of the synthesis. However, this flow is considered rather typical since the HMA model essentially considers on-demand partnerships. Second, in the HMA model, a parent can not only be a host but can also be an agent. It is quite natural that a synthesis is activated by a parent to obtain additional functions as a child (e.g. the advertisement agent in Section 3.1). It is also possible that a parent would migrate to the host of a child, forms a synthesis and provides services (e.g. the mobile streaming agent in Section 3.1). These behaviors have not been facilitated by the frameworks since developers would have to combine the entering action and communication to request the action.

3.2.2 Cooperation of Child and Parent

For (2), consistent behaviors of a child and its parent have not been considered in the existing frameworks. In the frameworks, a child can exit out of its parent on its own discretion. This may cause termination of interaction and then unexpected errors to occur in the parent's behaviors. One idea is to introduce controls and restrictions over children's actions by the parent. It seems like a natural metaphor that a parent would own and control its children (components). Actually, in the MobileSpaces framework, a parent can kill its child or remove it out (Satoh, 2000). However, such control by a parent may in turn cause error in its child's behaviors. Some mechanism is thus required to ensure duration of a partnership. In addition, it has not been discussed whether other types of controls or restrictions over a child by its parent are useful or not.

Below we list types of controls/restrictions considered useful with respect to each purpose. In this work we limit our discussion to primitive functions typically managed by frameworks, namely, lifecycle management, communication, and migration/synthesis.

Duration of Partnership. In the existing work, a child can migrate out of its parent on its own discretion. On the other hand, in MobileSpaces a parent can kill its child or move it away. These actions may cause interruption of interaction that is unexpected by the partner. It should be assured the partnership is maintained for a certain duration.

Service Provision Parents may want their children to interact with the outside in special manners, namely, (1) a black box model and/or (2) encapsulation (the advertisement agent in Section 3.1). (1) A parent makes its child invisible from outside to hide its internal business logic. In this case, all required communication of the child would be conducted in the parent's name, and forwarded by the parent as appropriate. (2) A parent encapsulates its child so that the child's services are not available from the outside, directly or via the parent, and restricted usages are achieved by the parent.

Confinement In MobileSpaces, a parent can kill its child. One usage of this function is to "confine" a child so that it cannot leak some information such as the internal business logic or user privacy information (the privacy agent in Section 3.1). This also requires restriction on outbound communication of the child.

The extent of these controls/restrictions should be decided for each agent pair. In Section 3.1, the mobile streaming agent merely provides carrier functions and imposes no restriction on the video agent while the video agent wants to encapsulate the advertisement agent and control its functions. The above patterns require a mechanism to make agents agreed before synthesis and then work cooperatively according to the

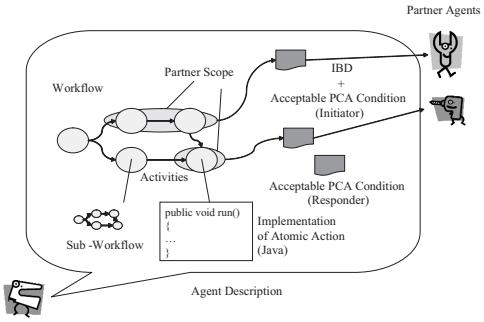

Figure 3: MAFEH Agent Description Model

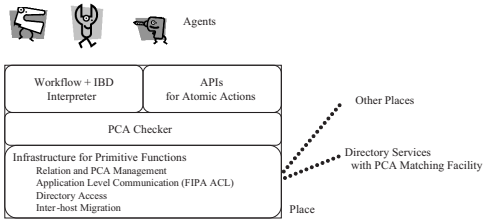

Figure 4: MAFEH Architecture

agreement, since they may have conflicting desires on the extent of the controls/restrictions.

4 MAFEH FRAMEWORK

This section describes the basic direction of our ongoing work on the MAFEH framework.

4.1 Overview

Considering the requirements discussed in Section 3.2, we are developing an HMA framework named MAFEH including two features: *Interaction Behavior Description (IBD)* and *Parent-Child Agreement (PCA)*. Figure 3 and 4 illustrate the agent description model and the architecture of MAFEH, respectively. Although the concepts of PCA and IPA do not essentially depend on enviroments, our current work assumes all agents are registered and can be discovered in global directory services.

The main application logic is described as a *workflow*, which connects sub-workflows and *atomic actions*. An atomic action is a unit of execution (computation or interaction with other agents) and implemented in Java. This logic specifies how to interact with abstract partners without concrete information of them. We assume developers know the type of each partner and interaction protocols for communicating with them.

IBD is then used by agent developers to describe concrete information and detailed action parameters related to each abstract partner declared in the workflow description, for discovery and migration/synthesis.

PCA denotes an agreement on the extent of controls/restrictions over a child by its parent. Developers give each agent *acceptable PCA conditions*. They denote the conditions about the extent of controls/restrictions under which the agent can achieve its expected behavior. An agent has accetable PCA conditions for each partner that it asks to provide services (in case the agent is an initiator of interaction), and for a partner that asks it to provide services (in case the agent is a responder). When an agent activates synthesis, it discovers a partner with which it can establish a PCA using simple matching rules. Management of parent-child relationships are conducted by MAFEH. The relationships are managed as logical links between agents. Calls for primitive functions are hooked and changed or denied according to formed PCAs. The framework is responsible to realize and assure compliance with formed PCAs. Infraction by not using IBD or provided APIs is prevented by using SecurityManager in Java.

4.2 Main Application Logic

The main application logic is described as a combination of (1) implementation of each atomic action in Java and (2) a global workflow that connects *activities* (atomic actions and sub-workflows). A *workflow* consists of *activities* and the control/data flow among them. An activity is either a sub-workflow or an *atomic action*. MAFEH provides APIs for implementation of atomic actions, such as APIs for getting workflow-level inputs and setting outputs, and sending/receiving messages of FIPA ACL (AGENTS, 1997). For description of control/data flow, we followed Web Services Flow Language (Leymann, 2001) and extended its notation and semantics with the concept of exception handling. The control/data flow is expressed by connecting activities with *control links* and *data links*. Loop execution is expressed with an *exit condition* of an activity (until loop).

This two-layer descriptions of the main logic is intended to keep control of migration and synthesis as abstract as possible by handling it at the workflow level. Inter-host migration is not activated during execution of atomic actions but in the interval, according to IBD (described later in Section 4.4). Upon migration all activity states and variable values at the workflow level are preserved, and then rebuilt after migration completion. When a parent initiates migration, the migration is delayed until all the children finish their current atomic actions and preserve their states.

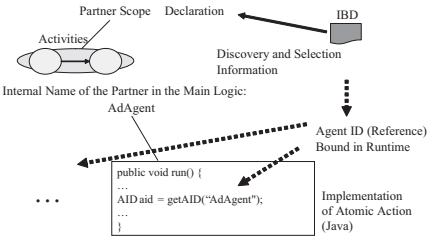

Figure 5: Partner Scope

The main application logic can be specified without concrete information of interaction partners. Figure 5 illustrates how each partner is bound. Names of the involved partners are declared and then can be used in the implementation of atomic actions. At the global workflow level, the *partner scope* of each partner is defined and then used to decide when to initiate synthesis or dissolve it. Concrete information of the declared partners and detailed parameters for migration/synthesis (if preferred) are given in Interaction Behavior Descriptions (IBD) described later in Secton 4.4.

4.3 Parent-Child Agreement

PCA specifies an agreement on the extent of controls/restrictions over the child by the parent. As mentioned in Section 3.2.2, this work handles controls/restrictions related to term of the partnership, black box/encapsulation, and confinement.

The extent of each control or restriction is specified as a value of a predefined parameter in PCA. Each agent is given, by its developers, acceptable PCA conditions for each partner that it requests (for the case where the agent is the initiator of interaction) and for the partner that requests it to provide services (for the case where the agent is the responder). When an agent requests for synthesis, the MAFEH framework matches conditions of two agents (the initiator and responder) and forms a PCA. This work adopts simple matching rules that derive the *least* controls/estrictions compliant with the acceptable conditions. After forming synthesis, the framework achieves the controls and denies actions against the restrictions in the PCA, causing an exception to be thrown (both at the workflow and atomic action levels).

Figure 6 shows an example of acceptable PCA conditons of two agents and the PCA established by mathching them. The parameters are defined to achieve controls/restrictions described in Section 3.2.2. Suppose the parent is the video agent and the child is the advertisement agent in Section 3.1. In the parent conditions, the parent wants black model (*requireBlackBox*) and encapsulation of the advertisement provision functionality (*deniedInboundMessages* and *deniedOutboundMessages*). It does not want to kill the child after the partnership expires (for confinement) (*requireKillResolve*). On the other hand, in the child conditions, the child wants to interact with its master servers for update (*subscribedInboundMessages* and *allowedOutboundMessages*). It also wants a guarantee for term of the partnership (*expireDateLaterThan*). As requirements of the two agents are not inconsistent, matching succeeds, leading to the PCA with the least controls/restrictions compliant with both of the requirements.

```
partnerName:  careerAgent{
role:  MobileStreaming/career
onDemand:  yes
interactionType:  parent
approach:  come
(Other detailed parameters
such as for exception handling)
}
```

Figure 7: IBD

4.4 Interaction Behavior Description

The main application logic specifies how to interact with abstract partners as mentioned in Section 4.2. IBD gives concrete information of the partners and detailed parameters for discovery, migration, and synthesis. Figure 7 shows IBD of the video agent for the mobile streaming agent in Section 3.1. Below, we describe how migration and synthesis actions are conducted upon interaction according to IBD.

When to initiate interaction with the partner is specified in *onDemand* in IBD. If the item is set to *yes*, the partner is discovered on demand, that is, when the interpreter of the workflow first encounters an activity that belongs to the partner's scope. If the item is set to *no*, the partner is discovered when the agent is instantiated, before execution of the main logic.

Agents are discovered from directory services that can play the role in the protocol specified in *role*. When a PCA can be established with an agent in the query result, the agent becomes the partner.

After the partner is decided, migration and/or synthesis is activated if necessary. The interaction type is specified in *interactionType*; (1) *parent*: interaction after entering into the partner, (2) *child*: interaction after taking the partner in, and (3) *parallel*: interaction without synthesis. In addition, the method of approaching the partner is specified in *approach*: its value can be (a) *go*: migrating to the partner's host, and (b) *come*: asking the partner to send a new instance of the partner to the agent's host, and (c) *remote*: interaction by remote messaging without migration only in the case of *parallel* interaction.

In the case of synthesis, dissolution of the synthe-

Parent Conditions

```
expiredDateLaterThan:

requireBlackBox:  yes
deniedInboundMessages:
  (protocol=="playAd")
deniedOutboundMessages:
  (protocol=="playAd")
requireKillDissolve:  no
```

Child Conditions

```
expiredDateLaterThan:
  2004-11-12 00:00:00 JST
acceptBlackBox:  yes
subscribedInboundMessages:
  (protocol=="adUpdate")
allowedOutboundMessages:
  (protocol=="ad")
acceptKillDissolve:  yes
```

Formed Agreement

```
expiredDate:
  2004-11-12 00:00:00 JST
blackBox:  yes
subscribedInboundMessages:
  (protocol=="adUpdate")
deniedOutboundMessages:
  (protocol=="playAd")
killDissolve:  no
```

Figure 6: Acceptable PCA Conditions and Formed PCA

sis is initiated according to the partner scope. Basically, when all the activities in the partner scope have been evaluated (including "skipped" in conditional branch), the synthesis is dissolved. In the case where a partner scope is involved in a loop, it is not clear whether it will be a loop targeting the same partner instance or a loop switches partner instances. It seems typical that an interaction with a partner is described as a loop and then that loop is involved in another loop in which partner instances are switched. In declaration of partner scopes, developers can specify the activity where partner instances are switched.

All the above actions are executed just before or after execution of an activity, according to partner scope declarations and IBDs. Exceptions in these actions are thrown and can be caught at the workflow level.

4.5 Adoption to Example

Below, we describe how IBD and PCA can be used for the example in Section 3.1.

Advertisement Agent The VCA takes in the advertisement agent (*interactionType:child*, *approach:come* in IBD). PCA for this agent pair was described in the example in Section 4.3.

User Agent VCA interacts with an agent that representing the user. When initialized, it migrates to interact with it locally without synthesis (*interactionType:parallel*, *approach:go* in IBD).

Mobile Streaming Agent VCA calls and enters a mobile streaming agent according to the user's request (*interactionType:parent,approach:come* in IBD). This time in PCA, the black box model is not adopted and the function for regular presentation can also be called from the outside. As the agent provides migration strategy based on knowledge of the current environment and supports a streaming protocol supported by the user's device, this agent is switched every time requested (*onDemand: yes* in IBD).

Privacy Agent The VCA enters into the privacy agent that imposes strong restrictions in return for the user's information (*interactionType:parent,approach:go* in IBD). According to the user's policy, the agent inhibits communication with the outside and migration out of it (*deniedOutboundMessages: (true)*, *isEscapable:no* in PCA).

This way, various patterns for agent pairs can be achieved on the framework, MAFEH, especially the interaction patterns in IBD and the control/restriction patterns in PCA.

5 DISCUSSION

We have briefly described the basic concepts of the MAFEH framework, especially IBD and PCA. In this section, we discuss advantages of our approach to IBD and PCA, though MAFEH is currently under development and requires evaluation by constructing real applications. We also discuss further requirements on IBD and PCA.

5.1 IBD

We mentioned needs for facilitation of the typical flow: discovery, synthesis, and dissolution, in Section 3.2.1. In MAFEH, developers explicitly declare partner scopes so that the framework can manage the flow eliminating the need for developers to insert each action appropriately. IBD also facilitates the patterns mentioned in Section 3.2.1, synthesis activation and inter-agent migration (*interactionType* and *approach*). In the existing frameworks, these patterns have not been facilitated as they require combining communication for requests and the primitive function to enter into an agent.

In addition, as details of interaction partners are separated from application logic, it is easier for developers to change them, e.g. to change from synthesis to parallel interaction or to change parameters in the query for discovery. If such details are not separated from application logic, it is difficult to find and modify actions scattered and enweaved within the application logic. However, as this approach of IBD limits description capability, further evaluation is re-

quired by constructing various agents such as those mentioned in 3.1.

IBD takes a similar approach to the one in coordination languages in the Web Services activity for on-demand discovery and coodination of distributed services, e.g. WSFL (Leymann, 2001) and BPEL4WS ((editor) et al., 2003). However, they have not considered strong partnerships as in the HMA model for long-lasting interaction. On the other hand, there have been many languages for controling mobile agents by associating migration with actions (ita, 2001; Ishikawa et al., pear). However, no language has appeared for the HMA model.

It is necessary to extend the current work to facilitate decision of acceptance of agent synthesis based on nonfunctional parameters such as trust and security policies. We are thinking of introducing some policy descriptions similar to acceptable PCA conditions, which allow the framework to support an agent's decision to accept an offer of synthesis.

5.2 PCA

In 3.2.2, we described basic types of agent cooperation in the HMA model: assurance of partnership durations, the black box model and encapsulation, and confinement for information secrecy. MAFEH allows an agent to change the extent of such controls/restrictions for each of its partners, obeying the minimum requirements imposed by its developer.

The framework supports realization of controls and compliance with restrictions in PCA. If they are delegated to each agent, each developer is obligated to implement cooperative behaviors correctly. For example, the black box model requires developers to code appropriately so that a child sends all messages to its parent and the parent forwards inbound and outbound messages correctly. However, it is a heavy burden to code such detailed actions obeying each PCA decided at runtime. It is also very difficult to assure that each agent behaves correctly according to the PCA.

Acceptable PCA conditions are given by developers. Tool support is necessary such as verification of the conditions or (semi-)automatic extraction of them by analysing the main logic. In addition, it should be possible to develop an agent that handles its conditions by itself. For example, the video agent should decide by itself to accept strict conditions of the privacy agent only in return for privacy information, and change its behaviors adapting to the established PCA.

6 CONCLUSION

This paper has described the basic direction of our work on the MAFEH framework for hierarchical mobile agents. MAFEH includes two features: PCA for an agreement regarding behaviors of a parent and its child, and IBD for description of synthesis actions separated from the main application logic. Although our work is still at an early stage, we believe we can mature MAFEH for development of agents that autonomously establish synthesis at runtime, by incrementally focusing on and refining each part of MAFEH.

REFERENCES

(2001). Mobile agents itinerary language project. `http://nemesis.csse.monash.edu.au/~azaslavs/itag/`. (Access: 10 June 2004).

AGENTS, F. F. I. P. (1997). Agent communication language. `http://www.fipa.org/specs/fipa00003/OC00003A.html`.

Bradshaw, J. M. (1997). *An Introduction to Software Agents*. MIT Press.

Cardelli, L. and Gordon, A. D. (1998). Mobile ambients. In *Foundations of Software Science and Computation Structures: First International Conference*.

(editor), S. T. et al. (2003). Business process execution language for web services, version 1.1. `http://www-106.ibm.com/developerworks/webservices/library/ws-bpel/`. (Access: 10 June 2004).

Haas, H. (2004). Web services. `http://www.w3.org/2002/ws/`. (Access: 10 June 2004).

Ishikawa, F., Yoshioka, N., Tahara, Y., and Honiden, S. ((to appear)). Behavior descriptions of mobile agents for web services integration. In *2004 IEEE International Conference on Web Services (ICWS 2004)*.

Leymann, F. (2001). Web services flow language (wsfl1.0). `http://www-3.ibm.com/software/solutions/webservices/pdf/WSFL.pdf`.

Satoh, I. (2000). Mobilespaces: A framework for building adaptive distributed applications using a hierarchical mobile agent system. In *The 20th International Conference on Distributed Computing Systems (ICDCS 2000)*, pages 161–168.

Satoh, I. (2001). Mobile agent-based compound documents. In *ACM Symposium on Document Engineering 2001*, pages 76–84.

Suna, A. and Fallah-Seghrouchni, A. E. (2004). A mobile agents platform; architecture, mobility and security elements. In *The 2nd International Workshop on Programming Multiagent Systems Languages and tools (PROMAS 2004)*.

Tahara, Y., Yoshioka, N., and Honiden, S. (2003). A formal model of active contents based on the ambient calculus. In *The 5th International Workshop on Mobile Agents for Telecommunication Applications*, pages 132–141.