

Service-Oriented and Autonomous Distribution and Provision of Multimedia Contents

Fuyuki Ishikawa
The University of Tokyo
7-3-1 Hongo, Bunkyo-ku,
Tokyo, Japan
f-ishikawa@nii.ac.jp

Nobukazu Yoshioka
National Institute of
Informatics
2-1-2 Hitotsubashi,
Chiyoda-ku, Tokyo, Japan
nobukazu@nii.ac.jp

Shinichi Honiden
National Institute of
Informatics
and The University of Tokyo
honiden@nii.ac.jp

ABSTRACT

This work considers encapsulation of multimedia contents as services, and their autonomous provision and distribution by agents. This approach enables content providers to have their contents provided and distributed together with their own business logics regarding interaction with other services and contents, control logic for various options upon provision, and value-added services. This paper proposes a framework for development of agents that provide and distribute such content services. In the framework, the interaction logic of agents is first developed in the same way as in development of ordinary Web services. Control descriptions are then given for each aspect, such as selection of partner agents, negotiation on service options, and migration. The separated descriptions for each aspect facilitate incremental development and test, as well as later modification, of agents with complex behaviors.

Categories and Subject Descriptors

D.2.11 [Software Engineering]: Software Architectures—*Languages*

General Terms

Design

Keywords

Agent Framework, Service-Oriented Computing, Multimedia, Mobile Agents, Agreement

1. INTRODUCTION

Rapid progress in computing and network technologies has enabled people to use various kinds of multimedia contents, e.g., movies, business documents, and electronic medical charts, with their computers or mobile devices. Operations on multimedia contents include not only to play and edit but also to translate between different formats, extract properties to annotate the contents, and generate

customized view. Currently, multimedia contents are often treated as data and operated using applications, e.g., players, installed on users' computers, and content providers can only set control parameters for them in advance. Consequently, flexible policies regarding how their contents are manipulated after delivered cannot be incorporated, for example, flexible license control such as license division for multiple contents or multiple devices, design and license restriction upon content composition, and content-specific value-added functions such as generation of customized view.

In response to this problem, this work considers encapsulation of multimedia contents by autonomous agents. Multimedia contents are encapsulated together with business logic or control policies given by content providers, provided and distributed autonomously by agents in complying with the policies. This model enables for content providers to keep control on how their contents are handled and to provide value-added services such as integration of related contents or content-specific customization.

Agents encapsulating multimedia contents need to discover and cooperate with other agents providing other contents or functions to form composite services or contents. Issues in implementation of such behavior are not specific to this domain and have been investigated especially in the Service-Oriented Computing (SOC) activity. In SOC, software systems provide services to other applications through published and discoverable interfaces, and the services can be invoked over a network [2]. It thus leads to autonomous composition of a new service by integrating existing ones at runtime. In addition to the above general issues, agent mobility is necessary to implement (possibly autonomous) contents distribution as well as to allow for function migration. In addition, agents should be able to be part of another agent, to represent atomic contents embedded in composite contents or functions embedded in contents. They especially should work cooperatively, for example, migrate between hosts together as a unit and adjust their interfaces and behaviors so that they are seen as one consistent composite agent [4]. Besides, services that provide multimedia contents have various kinds of application-specific service options as mentioned above. It is thus necessary to introduce a mechanism to handle agreements on application-specific service options [3], though the effort on SLAs (Service Level Agreements) for SOC has considered only application-independent service options, especially QoS parameters.

The control flow of agents including these features can be very complex. For example, when an agent wants to initiate interaction with another agent providing required services or contents, it first needs to discover and select an agent playing the required role. It may then migrate and/or ne-

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...\$5.00.

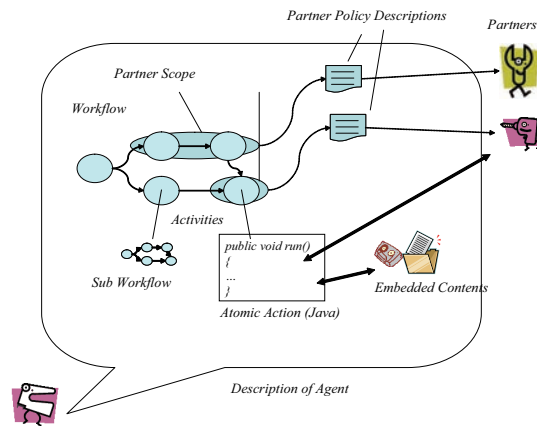


Figure 1: description Model for Active Contents

gotiate about service options such as licenses. The whole control flow woven with such behaviors for each partner agent is very complex and unmanageable. In response to the problem, this study proposes a framework named Freedria for development of agents encapsulating multimedia contents. It aims to enable developers to handle the complexity in development of agents including various aspects such as partner discovery, migration, and agreements on service options. The proposed Freedria framework allows developers to first specify the main interaction logic of an agent, that is, control and data flow between atomic actions including messaging actions. It is thus possible to start with development of agents that simply interact with their partners through remote messaging and without migration nor agreements. Developers can then incorporate policy descriptions that specify requirements and related behaviors for each interaction partner. The separated descriptions for each aspect facilitate incremental development and test, as well as later modification, of agents with complex behaviors.

2. FREEDRIA FRAMEWORK

2.1 Workflow and Partner Scopes

This section describes the proposed framework, Freedria, for development of agents that autonomously distributes and provides multimedia contents. The model for agent description in Freedria is shown in Figure 1. The main logic, including interaction with partner agents as well as internal behavior, is described as a workflow description. The workflow description is based on a common approach in process languages for Service-Oriented Computing [5, 6], that is, interaction with partner agents of required types is specified without concrete description for binding partners. Information used for discovery and selection of partners is given separately. The workflow description specifies control and data flow between actions in a graph-oriented way. The activities are implemented in Java and include messaging actions (sending or receiving messages in FIPA ACL), internal computation, and/or operation on contents the agent contains. The notation and operational semantics of workflow are based on Web Services Flow Language [5]. For each partner declared in the workflow description, policy descriptions are given to control behaviors related to the partner, including discovery, negotiation and selection for service agreements, and migration/synthesis (*Partner Policy* descriptions described in Session 2.2).

In definition of activities in a workflow, types of the partner agents the agent interacts with during execution of the activity are declared. Types of partner agents are called *roles* and denote interface of agents in terms of observable messaging actions. If an agent has a certain role, it means the agent can send or receive messages in a certain sequence specified in a certain interaction protocol. In definition of an activity in a workflow, a partner is declared by specifying a set of roles required for the partner.

As partners that the agent can interact with are declared in each activity, a set of activities where interaction with a partner occurs can be extracted. This set of activities is called a *partner scope*, and used by Freedria to determine when to activate initialization of a partnership, e.g., discovery and negotiation, or termination, e.g., disposal of service agreements. For each partner, initialization is activated when execution enters the scope for the partner, that is, when execution reaches an activity where interaction with the partner occurs. Termination is activated when execution exits of the scope for the partner, that is, by default when execution gets out of the outermost loop enclosing the partner scope (intuitively, when interaction with the partner turns out not to occur anymore). It is also possible to specify whether interaction is iterated in a loop with the same partner or switching the partner. Details of initialization and termination are described in the following.

2.2 Partner Policy Descriptions

2.2.1 Overview

For each partner declared in the workflow description, Partner Policy descriptions are given to specify requirements for and behaviors related to the partner. Currently, a partnership is built between two agents, and connection of partnerships between agent pairs leads to a collaborative group of agents. Preliminary interaction is held to build a partnership according to Partner Policy descriptions. In each agent pair for any protocol, one agent is classified as a *requester* and initially sends a request in preliminary interaction to build a partnership. The other is a *responder* and waits for a request for interaction. Below each item in Partner Policy descriptions is first described separately, and how to integrate them is then described.

2.2.2 Items in Partner Policy Descriptions

2.2.2.1 Agent Properties.

Requirements for properties of partner agents, such as organizations providing them and service quality, are given in Partner Policy descriptions. In the case of a requester, these requirements descriptions are used for discovery of agents, together with required roles declared in the workflow description. In the case of a responder, they are used to determine whether to accept a request for interaction or not.

2.2.2.2 Interaction Style.

Requirements for interaction styles with partners are specified in Partner Policy descriptions (based on the authors' previous study [4]). Here interaction styles include whether migration is conducted to be on the same host or not, and where the agent pair gathers if conducted. They also include whether an agent compound is established or not, where multiple agents work cooperatively and migrate as a unit (called agent synthesis), and what kinds of control are enforced in the agent compound (if established). For example, monitoring and operating on messages allows for encapsu-

lation of the contained agent, e.g., inhibiting the containing agent from providing services directly. It also allows for override of services by the contained agent, e.g., adding additional access control check or filtering. Each agent has requirements on these aspects in Partner Policy descriptions, and Freedra framework matches them and builds agreements on interaction styles if matched. The Freedra framework is also responsible to enforce the agreements, i.e., to activate migration of an agent, to monitor and pass messages to given handlers of containing agents, and so on.

2.2.2.3 Service Agreements.

Behaviors regarding agreements on service options, both interaction to build agreements and change to comply with the agreements, are specified in Partner Policy descriptions (based on the authors' previous study [3]).

For behaviors to build agreements, templates to implement Iterated Contract Net Protocol (CNP) [1] to build agreements are provided. Various existing or emerging techniques are thus incorporated such as iterated negotiation. The Iterated CNP protocol considers the most complex situation, iterated negotiation and selection between a consumer and multiple candidates for service providers. Each internal behavior in this process is implemented by using Java abstract classes provided by the Freedra framework. In Partner Policy descriptions, such components are specified and the framework conducts the negotiation and selection process by invoking the given components and messaging actions including time-out check according to the protocol.

For change in behaviors to comply with established agreements, Aspect-Oriented rule descriptions are provided. Some service options are strongly related to the workflow and originally expected to be customizable, e.g., bandwidth and size of movies. In the case such options are determined not in each request but in long-term service agreements, rule descriptions for setting parameters in the workflow can be adopted. Other service options are independent of the workflow and require additional actions, e.g., access control check and merger of value-added contents. In this case, rule descriptions that insert additional actions before/after activities or upon receiving/sending messages.

2.2.3 Putting Together

At runtime, initialization and termination are conducted for each partner at execution points determined according to partner scopes. Each item described above includes preliminary interaction for initialization, i.e., requirements matching and negotiation. At runtime, requirements matching for service properties and migration/synthesis is integrated into the given interaction for service agreements by Freedra. Partnerships are established only if matching for both aspects is succeeded and service agreements are successfully built. After agreements for interaction styles and for service options are established, migration/synthesis is conducted and rules for enforcing service agreements are activated (if necessary). When execution of workflow descriptions exits out of the partner scope, dissolve of synthesis and deactivation of rules for enforcing service agreements are conducted as finalization (if necessary).

Every item in Partner Policy descriptions does not always need to be given. For example, if requirements for service properties are not given, candidates for a partner are obtained by querying directory services only with roles. This default behavior, together with separation of concerns in the description model, facilitates incremental development, that is, first develop and test the interaction logic, and then give control descriptions for each aspect and test one by one.

2.3 Experiences

The proposed Freedra framework is currently implemented in Java. The framework includes a visual editor and debugger for workflow descriptions, as well as the infrastructure for interpretation of the workflow and Partner Policy descriptions and agent management.

Three applications developed using the proposed Freedra are described briefly. In the first one, agents encapsulating electronic medical charts are delivered to different organizations while obeying policies given by the original distributor. Second, agents encapsulating instruction manual of health supplements follow movement of human users, proactively showing instruction of the supplements timely and offer value-added services such as setting parameters of electronic exercise equipments. Last, agents encapsulating e-learning contents interact with each other to support experiences of English conversation between students while cooperating with agents given teachers' management policies. All the scenarios include aspects of partner discovery, migration/synthesis (e.g., following mobile users or override of services using the mechanisms for agent synthesis), and service agreements (authentication and authorization or learning level). These applications were developed incrementally. First only the interaction logic of agents was implemented, without any policies on coupling, negotiation, or synthesis, i.e., only workflow descriptions are given. Policies about discovery, cooperative migration, and service agreements were introduced and tested gradually.

3. SUMMARY

This study has considered encapsulation of multimedia contents as agents. This approach enables content providers to have their contents provided and distributed together with their own business logics regarding interaction with other agents that provide required services or contents, control logic for various options upon provision, and value-added services. For implementation of such agents, this study has proposed a framework that facilitates incremental development of agents with complex behavior including various aspects.

4. REFERENCES

- [1] FIPA. Fipa iterated contract net interaction protocol specification. <http://www.fipa.org/specs/fipa00030/SC00030H.html>, December 2002.
- [2] H. Haas. Web services. <http://www.w3.org/2002/ws/>, June 2004. (Access: May 2005).
- [3] F. Ishikawa, N. Yoshioka, and S. Honiden. Incorporating agreements on service options into bpm-based services. In *International Conference on Intelligent Agents, Web Technology and Internet Commerce (IAWTIC'2005)*, November 2005 (to appear).
- [4] F. Ishikawa, N. Yoshioka, Y. Tahara, and S. Honiden. Developing service-oriented compounds of cooperative mobile agents. In *Workshop on Service-Oriented Computing and Agent-Based Engineering (SOCABE'2005)*, July 2005.
- [5] F. Leymann. Web services flow language (wsfl1.0). <http://www-3.ibm.com/software/solutions/webservices/pdf/WSFL.pdf>, 2001.
- [6] S. Thatte et al. Business process execution language for web services, version 1.1. <http://www-106.ibm.com/developerworks/webservices/library/ws-bpel/>, May 2003.