Regular Paper

Flexible Interface Mapping for System Cooperation and Its Evaluation

Makoto Nakatsuji,† Yu Miyoshi,† Yoshihiro Otsuka† and Miki Hirano†

We model the relationships between the message formats of a business system and their semantics in a machine-processable knowledge base. We describe a message-mapping technique that extracts the relationships between the message formats of several systems semiautomatically by using the class characteristics of the semantics and stores these relationships as past system design knowledge. In addition, we propose process-mapping, which is a technique that discovers suitable software components for system cooperation. We evaluate these methods using the interface specifications of actual service management systems and show that the frequency of interface adjustment can be reduced.

1. Introduction

Many companies have tried to introduce the concept of a service oriented architecture (SOA) in order to develop services agilely and operate them efficiently. In SOA, new services are constructed by incorporating services constructed in the past into the business process of new services. We consider this SOA concept to be important because it enables companies to provide new services agilely with an effective operating environment by combining the business processes of distributed services.

However, existing systems for executing various services, such as systems for customer relationship management (CRM), are designed individually for their own purposes in each operating division. Even if all services are developed using a common communication protocol such as Web Services or socket-type communication, it is difficult to reuse services by applying SOA concepts in such environments for two reasons.

- Message formats for defining the addresses and types of definitions of message elements are frequently different even if they are used for the same meaning.
- Processes that define service execution procedures are designed individually based on the strategies of each division in a dynamic business environment.

In this paper, we define *system cooperation* as a method of developing systems by extracting reusable interface specifications of existing systems from the viewpoint of semantics and incorporating the extracted specifications into the business processes of newly constructed systems. We then try to provide an environment that enables system designers to develop new systems according to their own purposes by reusing distributed information within the processes of distributed services in a dynamic business environment.

To achieve these purposes, we express the interface specifications of systems semantically as a hierarchy of classes. Then, we present *interface mapping*[8),9),11)], which calculates the relationships between messages and processes of several systems by using semantic mapping of the interface specifications between different systems. Interface mapping presents these relationships to the designers of newly cooperating systems, so they can extract systems for incorporation into newly coordinated systems and design the interface specifications of newly coordinated systems by referring to and reusing these relationships.

The specific contributions of this paper are as follows.

- We describe an interface modeling of the relationships between message formats and the semantics of those formats in a machine-processable way using OWL (web ontology language) Description Logic (OWL DL)[7)] for systems developed by socket-type communication. Using OWL DL, we describe the characteristics of interface specifications from the viewpoints of how to exchange messages between systems. Service management systems (SMSs), especially ones that have to manage a lot of message transactions submitted by several service systems on a network, will continue to be developed using the socket-type communication tech-

† NTT Network Service Systems Laboratories

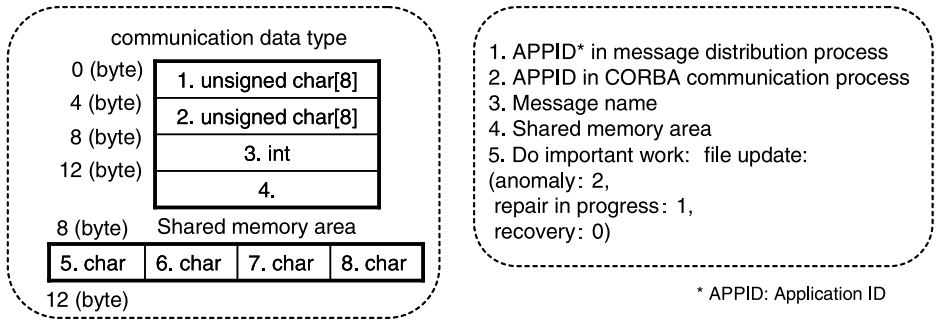

Fig. 1 Semantic schemas lacking information in message formats.

nique, even though system coordination techniques based on web services have been proposed in many research studies. Thus, providing semantics to message formats of socket-type communication is important.

- We introduce message-mapping, which automatically calculates the correspondences of message elements of different systems based on semantic mapping results. Message-mapping stores the mapping results in a knowledge base of relationships about interface specifications of different systems (KBR) as past system-design knowledge. By referring to these relationships in the KBR, system designers can adjust the interface specifications of newly constructed systems effectively. In addition, we propose process-mapping, which extracts the interface specifications of existing systems in order to incorporate those systems into existing system specifications. Process-mapping manages message parameters that have a low probability of being reused in a KBR. This improves the accuracy of process-mapping considering that services that commonly have these low-probability parameters treat similar information and are relevant to each other.
- We describe a comprehensive set of experiments performed using actual system interface specifications. In previous experiments on system coordination, such as Semantic Web Services (SWS) [3),4),6),12),14),15)], semantic mapping of the interface specifications for system coordination mainly used ontology mapping techniques [2),5),16)]. However, those ontology mapping techniques used only test data provided for research purposes on the web [10)] and did not focus on the mapping of real system interface specifications. We applied our technique to the adjustment of interface specifications of real SMSs. Furthermore, we compared our technique with the previous experiments from the viewpoint of system cooperation.

This paper is organized as follows. Section 2 discusses the problems of existing system cooperation and introduces related work. Section 3 provides the semantics of message formats. Section 4 describes message mapping, and Section 5 introduces our process mapping. Section 6 shows how system designers can extract interface conversion rules from mapping results. Section 7 describes our experimental study. Section 8 concludes with a short summary of the main points.

2. Related Work

In systems that cooperate using existing technologies such as socket-type communication and Web Services, we must manually adjust the message formats of different systems beforehand.

For instance, as shown in **Fig. 1**, in the message formats defined in operations support systems (OSSs), message elements are identified at the storage memory position in the message formats, and each data type is shown separately according to the data type definition in each memory position. However, the relationships between the schemas of the semantics and message format are not written in a machine-processable way. Therefore, in cooperating systems, we should refer to information described in interface specifications and manually adjust the message formats by comparing the semantics of the formats.

To resolve such problems, researchers of Semantic Web Services (SWS) [3),4),6),12),14),15)] use

Semantic Web technologies [1], which use various types of software to process their tasks automatically by describing the background knowledge of web resources as computer-interpretable metadata and using that metadata as semantics. For example, to achieve automatic service discovery, selection, and composition, some researchers built OWL-S [6] (formerly DAML-S [4]) which is a semantic markup language for Web Services. By using OWL-S, they gave semantics to Web Service interfaces, such as I/O (input/output) messages, preconditions required by the services, and effects that result from the execution of the services. On the other hand, some researchers tried to establish WSMO (Web Services Modeling Ontology) [3], which models a semantic mediator between services by using ontology language, and enables automatic interoperation of distributed services.

As described above, SWS provides an environment for describing the semantics of Web Services. For example, OWL-S grounding [6] provides semantics to the message formats of Web Services. Thus, we can use OWL-S grounding to model the relationships between the semantics and message formats of Web Services. However, SWS does not consider systems built using socket-type communication though systems will continue to be developed using socket-type communication, especially in situations where they must manage a lot of message transactions like SMS in a telecom network. Furthermore, most SWS research studies rely on semantic mapping to adjust messages and processes necessary for system coordination in ontology mapping techniques [2),5),16)]. They usually do not create semantic ontologies that are extracted from system interface specifications of real SMSs and do not evaluate system coordination based on the system interface specifications of real SMSs.

In this study, we utilize the concept of SWS to coordinate actual systems considering the following two points. (1) We utilize an interface-modeling technique that provides semantics to the message elements in message parameters, especially for socket-type communication. (2) We propose an ontology mapping technique suitable for system coordination by storing the relationships between interface specifications of different systems in a KBR as past system design knowledge. System designers semiautomatically extract the correspondences of message elements and reusable processes between different systems for system cooperation based on semantic mapping results. We evaluated our technique based on the semantics of the interface specifications of real SMSs that use socket-type communication.

However, our technique does not provide implementational techniques such as converting interface specifications of the systems based on socket-type communication to those of Web Services. Readers should note that such implementations in cooperating systems are tasks for the system designers.

Serin, et al. [15] proposed SDG, which contains the relationships between services based on the relationships between input and output messages of different systems. System designers can find reusable services by referring to the relationships in SDG. However, this semantic mapping technique [15] is a simple one that only uses the relationships between terms in a free thesaurus/dictionary such as Wordnet. If the relationships among words used in the interface specifications of various systems are not included in Wordnet, this technique cannot extract the relationships between interface specifications of different systems. In our technique, users can create new relationships between words used in the semantic descriptions of interface specifications in various systems by interactively constructing the relationships between the name attribute values of classes or instances based on our semantic mapping results. Therefore, the KBR in our technique stores a much larger amount of past system design knowledge than SDG does.

In a research study on ontology mapping and schema matching [2),5),16)], Glue tried to improve the accuracy of the mapping results by making use of the name, instances, and properties of the class. Furthermore, a corpus that stores the direct relationship between one schema and other schemas is introduced in Ref. 5). By using this corpus, Madhaven, et al. [5] tried to improve the accuracy of schema matching. There are two main differences between our message-mapping and their schema-matching [5]. (1) Our KBR is a description of the set of relationships between the messages of two or more systems. Therefore, our message-mapping enables system designers to find the direct relationships between systems and also indirect relationships. The use of these indirect relationships, gives our technique much greater potential to present the cor-

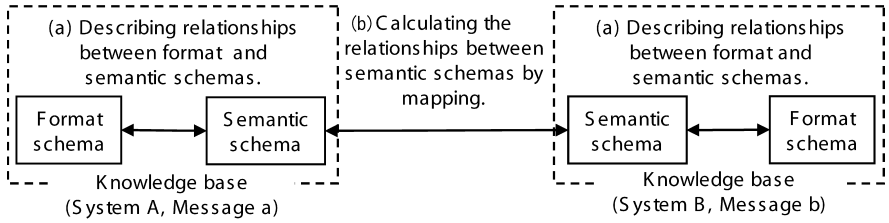

Fig. 2 Overview of knowledge base.

respondences between interface specifications of different systems. (2) We can improve the accuracy of process mapping by making use of the reuse counts of mapping results managed in the KBR.

In Ref. 17), we also applied our interface-mapping to the coordination of systems developed using Web Services, though we focus on the coordination of systems developed using socket-type communication in this study.

3. Interface Modeling

First, we explain the interface modeling that gives semantics to the message formats. For automatically cooperating systems, we model the relationship between semantics and interface specifications considering the following two points.

(1) System designers need to confirm the relationships between message elements in system message parameters when designing the interface specifications of a new system that is being developed through the cooperation of existing systems. We provide the relationships between message formats and semantics (**Fig. 2** (a)) based on interface modeling. Thus, our technique can automatically calculate the relationships between message elements in message parameters of systems based on semantic mapping results of message formats between systems (Fig. 2 (b)).

(2) System designers need to understand how the message elements will be exchanged between systems. Thus, we describe the semantics of message elements as characteristics of the interfaces. For example, the characteristic that only one of the instances of a class is exchanged between systems at the same time is described by the owl:oneOf component.

To satisfy the above points, we chose to use OWL DL, which can define the characteristics of a semantic class appropriately in a machine-processable way.

Table 1 OWL Description Logic.

Class axiom	rdfs:subClassOf	SubClassOf relates a more specific class to a more general class.
	owl:equivalentClass	The property owl:equivalentClass is used to indicate that two classes have precisely the same instances.
	owl:oneOf	The owl:oneOf specifies a class via a direct enumeration of its members.
Fact description by instances	owl:sameAs	The owl:sameAs can be used to state that seemingly different individuals are actually the same.

Parts of the OWL DL are shown in **Table 1**. The class in the OWL DL makes a group of similar individuals and offers functions to express their characteristics logically. The class axiom defines the characteristics of the class by using class expressions such as the enumeration of instances. In addition, by using the axiom concerning the instances, system designers can describe that two instances are the same using owl:sameAs.

To satisfy point (1) above, we model the relationships between message parameters and their semantics in a machine-processable way. We define the correspondences between message parameters and semantic classes by using owl:equivalentClass and those between message elements and semantic instances by using owl:sameAs. This is important for applying ontology mapping to system cooperation because system designers need to adjust not only message parameters but also message elements in cooperating systems. In **Fig. 3**, the semantic class "File update" is given to "Parameter A", and semantic instance "repair in progress" is given to message element "1".

To satisfy point (2), we express the characteristics of exchanging message elements between systems by using class axioms in OWL DL. In Fig. 3, only one message element among all the enumerated elements of the class "File update" is exchanged in one transmission between systems. We define this characteristic of a class

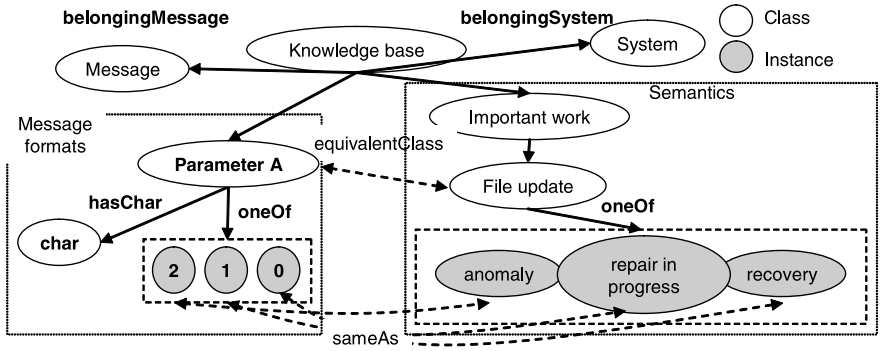

Fig. 3 Overview of interface modeling.

by using the owl:oneOf description. Moreover, we define class characteristics such as information about data types and systems to which the class belongs by using the property restrictions of a class.

4. Message Mapping

First, we apply an existing ontology mapping technique such as Glue to message mapping [5),9),11)] to semiautomatically adjust the message formats of different systems based on semantics. Then, we introduce the KBR to improve the accuracy of mappings by reusing past mapping results.

4.1 Introducing Ontology Mapping to Message Mapping

First, we define some terminology. We provide a source system, which is the component of the system that the designer wants to develop, and a target system, which is the component of the system that may be incorporated into the source system. We define source classes C_is as classes in semantic schema about the message formats of the source system and target classes C_js as classes in semantic schema about the message formats of the target system.

4.1.1 Measuring Similarity of Name Attributes

Message mapping calculates the similarity $S(N)_{C_iC_j}$ between name attribute values of C_i and C_j. First, message mapping divides name attribute values of classes into a morphological analysis tool such as Sen [☆]. Then, applying the vector space model to the frequency of each morpheme in name attributes, message mapping evaluates the score of the name attribute.

4.1.2 Measuring Similarity of Instances

Message mapping also calculates the similarity $S(N)_{I_iI_j}$ between name attribute values of instance I_i in C_i and instance I_j in C_j based on frequency of morphemes. If I_i and I_j satisfy equation $S(N)_{I_iI_j} > \theta$, message mapping considers that they have the same meaning. The threshold value θ in the above equation is heuristically determined. Finally, by using U_i, which is an instance set of class C_i, and U_j, which is an instance set of class C_j, message mapping expresses the score of instance set attribute $S(U)_{C_iC_j}$ as equation $S(U)_{C_iC_j} = \frac{|U_i \cap U_j|}{|U_i \cup U_j|}$.

4.1.3 Measuring Similarity of Component Attributes

Message mapping evaluates the score of component attribute $S(O)_{C_iC_j}$ from C_i to C_j. In this paper, the feature that only one of the instances in the class is exchanged between systems at the same time is described by the owl:oneOf component. Therefore, when both C_i and C_j have or do not have the owl:oneOf component, $S(O)_{C_iC_j} = 1$. When only one of them, either C_i or C_j, has it, $S(O)_{C_iC_j} = 0$.

Then, message mapping evaluates the score $S(C_{ij})$ from source class C_i to target class C_j by using the similarity of class attributes such as that of *name*, *instance set*, and *component description* [5),9),11)]. Then, message mapping ranks the target classes on the basis of the score $S(C_{ij})$ with instances and properties. This ranking enables a system interface designer to determine the mappings between classes and between instances.

4.2 Knowledge Base of Relationships about Interface Specifications between Different Systems

We need a support tool for semiautomatically adjusting the interface of newly constructed systems by reusing existing systems as compo-

[☆] http://ultimania.org/sen/

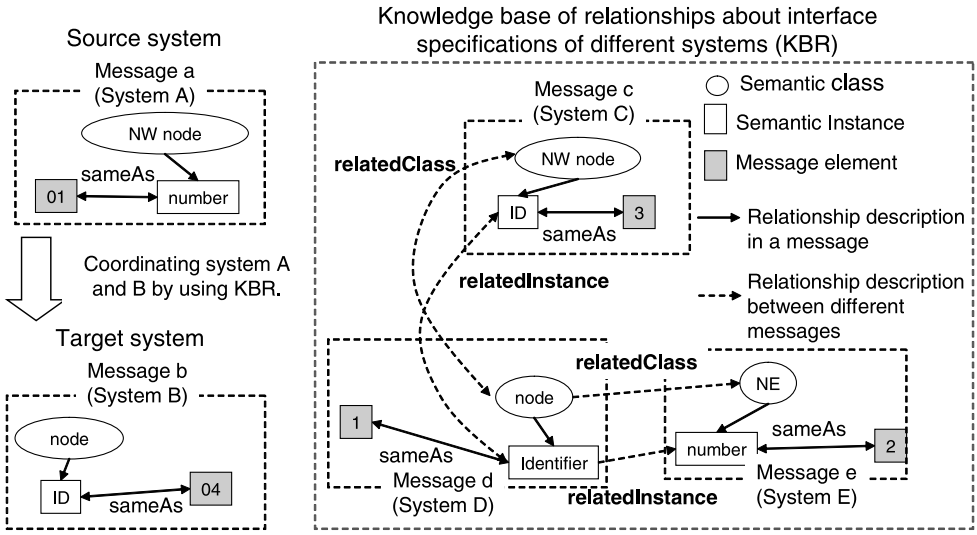

Fig. 4 Overview of KBR.

nents according to the SOA idea. Therefore, we store the past mapping results in the KBR to improve the reusability based on the following ideas.

(1) Adjusting message formats of two or more systems by one mapping

(2) Improving the accuracy of mapping results by using past mapping knowledge

4.2.1 Design Policies for Constructing KBR

An overview of a KBR is shown in **Fig. 4**. To be applicable to the purposes of each system designer in a dynamic business environment, the KBR stores mapping results even if they are not guaranteed to be consistent in all systems. In other words, from the viewpoint of a certain system designer, the message format of his/her system has a relationship to the message format of another system. However, this relationship is not always applicable to other system designers. Thus, the designer needs to check the proposed mapping when reusing past mapping results for current system coordination.

Therefore, relationships between classes in different systems are described in the language RDF (Resource Description Framework) [13], which does not guarantee the processability of the relationships consistently. Specifically, to reuse the past mapping results, our message mapping describes the relationships between classes even if some instances do not correspond between classes, so we introduce the relatedClass description. For the same reason, message mapping describes the relationship of instances even if they are equivalent between only some of the systems in the KBR, so we introduce the relatedInstance description [9),11)].

4.2.2 Storing Mapping Results in KBR

Message mapping stores the mapping results in the KBR after the system designer has confirmed that they are correct results. In this storing process, message mapping gives the system and message information as class properties. Thus, system designers can search the KBR for reusable mappings for future system coordination tasks.

4.2.3 Reusing Mapping Results

We describe the algorithm for reusing past mapping results from the KBR as follows. In searching the KBR, message mapping checks the relationship of the relatedClass description. We define a hop as the movement from one class to the next class following the relationships of the relatedClass description. We set the maximum number of hops so that the accuracy of the mapping results is not decreased by having a lot of indirect relationships. We also consider the duplication of mapping results by using class properties for the information about systems and messages.

We explain the algorithm when system designers coordinate source system A with target system B by using Fig. 4. Message a of A has a class "NW node" (network node) with an instance "number", and message b of B has a class "node" with an instance "ID" (identification).

(1) Message mapping searches the KBR for

classes that have the same name attribute value as the classes in the source system (source classes). We call these the starting classes. In Fig. 4, message mapping finds the starting class "NW node" in system C.

(2) If the KBR has starting classes, then message mapping searches for candidate classes by following the relatedClass description obtained from the starting classes. It finds "node" and "NE" (network element) as candidate classes for the source class "NW node", as shown in Fig. 4.

(3) When message mapping finds starting or candidate classes, it checks instances of those classes to see if they have the same name attribute values as instances in source classes (source instances). If it finds any such instances, it stores them as candidate instances by following the relatedInstance description. In Fig. 4, message mapping can store candidate instances "Identifier" and "ID", which are used in different systems.

(4) When the number of hops is equal to the maximum number of hops, message mapping stops checking the relatedClass description further and searches for classes with instances in the target system that have the same name attribute value as those of candidates. If such classes exist, message mapping provides them to system designers as mapping results. In Fig. 4, the convertible candidate with the (class, instance) relationship (NW node, number) of message a is (node, ID) of message b.

(5) The designers judge the mapping results. If they consider them to be true, the reuse counts R that shows how many times designers have reused the mapping results is incremented. Message mapping uses R as a parameter that indicates the reliability of the mapping. In judging the mapping results, the designers can create new relationships between instances in candidate classes and those in source classes.

The KBR enables message mapping to find candidate mapping results of two or more messages at once. For example, message mapping can extract candidate classes from messages c, d, and e at once. Furthermore, message mapping can help system designers to extract the corresponding message parameters of the source and target systems automatically.

5. Process Mapping

To design new systems by incorporating existing systems into them, we should consider the following points.

- Extract software components (SCs) with interface specifications that meet the purpose of a newly constructed system.
- Adjust message formats of a newly constructed system and existing SCs to reuse these SCs.

First, we model the SC as a process that has I/O messages, following the idea of OWL-S. Then, we extract reusable SCs based on the results of message mapping and show those extracted SCs to system designers with message mapping results including semantic information about messages. In our method, we consider the following ideas.

- Extract SCs with message parameters that are mutually related to each other based on message mapping. (process mapping algorithm 1: PMA1).
- When message parameters with low reusability in the KBR correspond between SCs, we consider that these SCs are related because they have low-probability parameters that reflect the characteristics of systems. Reusability is calculated by checking the reuse counts R in the KBR, as described in Section 4.2.3. (process mapping algorithm 2: PMA2)

5.1 PMA1

In **Fig. 5**, system A's SCs are C_{Ai} $(1 \leq i \leq m)$ and system B's SCs are C_{Bj} $(1 \leq j \leq n)$. We define MI_{Ai} as an input message of C_{Ai} and define MO_{Ai} as an output message of C_{Ai}.

(1) PMA1 extracts the correspondences of message parameters based on message mapping for incorporating system A into system B.

(2) Then, PMA1 calculates the number of correspondence classes N between MI_{Ai} and MI_{Bj} and the number of classes U in MI_{Ai}. We express the agreement score of input message $S(I)_{AiBj}$ as $\frac{|N|}{|U|}$.

(3) PMA1 also calculates the number of correspondence classes N between MO_{Ak} and MO_{Bl} and the number U of message classes in MO_{Ak}. We express the agreement score of output message $S(O)_{AkBl}$ as $\frac{|N|}{|U|}$.

(4) PMA1 evaluate the similarity score $S(P_{A_{ik}B_{jl}})$ between processes as $T(S(I)_{AiBj}) + T(S(O)_{AkBl})^{\alpha}$. The system designer checks the results based on the score of $S(P)$ and decides whether

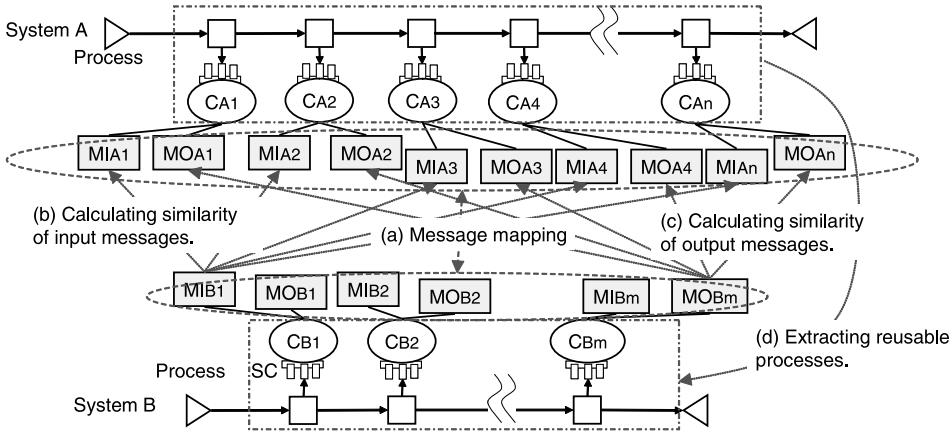

Fig. 5 Process mapping procedure.

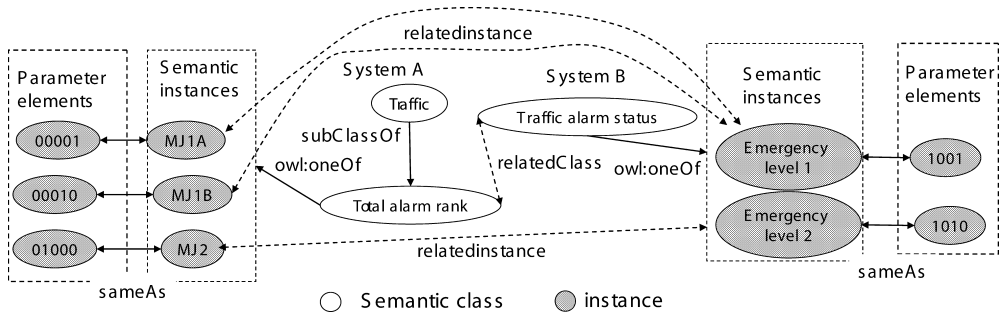

Fig. 6 Overview of extraction of interface conversion rules.

to incorporate the corresponding system into the newly constructed system. Here, we define a function T that gives the balance of the score for the mapping results by using average $\overline{x}$ and dispersion ${\sigma_x}^2$. We use constant α for a user to decide whether the input or output is important.

5.2 PMA2

In calculating the number of corresponding classes N in procedures (2) and (3) of PMA1, PMA2 checks the reuse counts R in the KBR, as described in Section 4.2.3, for each corresponding class. If R satisfies the equation $R \leq \beta$, PMA2 considers that these corresponding classes are actually related because they correspond via low-probability parameters that reflect the characteristics of systems. The threshold value β in the above equation is determined heuristically.

For example, the message element about "node ID in a network" is expected to appear in a lot of messages in various SMSs. Thus, R for this message has a high value. However, the message elements reflecting the characteristics of each SC such as "alarm status of node ID" are expected to appear a few messages in SMSs, and R for these message have low values.

6. Supporting the Extraction of Interface Conversion Rules Based on Mapping Results

Here, we explain the extraction of interface conversion rules for system cooperation. In **Fig. 6**, we assume for example that system designers try to develop a new system that manages all the alarm information of all the distributed systems.

Applying the message mapping, system designers acquire the mapping relationship between semantic classes. In Fig. 6, "Traffic alarm status" is assumed to have a relationship to "Total alarm rank". Moreover, reusing mapping results in the KBR, the instance "emergency level 2" of the class "Traffic alarm status" is assumed to have a relationship to the instance "MJ2" of "Total alarm rank". From these mapping results, system designers can extract interface conversion rules and implement these rules in cooperating systems. For exam-

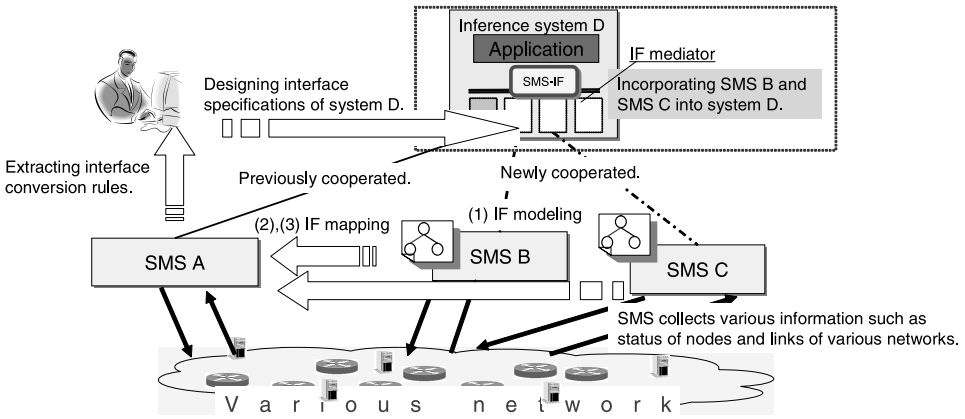

Fig. 7 System cooperation in our experiment.

Table 2 Data sets in our experiments.

	(SMS A & system D)	SMS B	SMS C
Number of SCs in experiments.	7	6	6
Average number of classes in SCs.	41.67	43.33	31.50
Number of SCs in a system.	179	165	228

ple, they could implement a interface conversion rule that changes message element "01000" in system A to "1010" in system B in their interface adjustment programs.

However, interface mapping technique only supports system designers to design interface specification of a newly cooperated system effectively by referring the relationships between interface specifications of past systems in KBR. The extraction and implementation of interface conversion rules in cooperating systems are tasks for the system designers.

7. Experimental Results

We now present experimental results that demonstrate the performance of message and process mapping.

7.1 Datasets and Methodology

We evaluated our interface mapping method using the interface specifications of actual service management systems (SMSs). These SMSs manage various kinds of information such as the status of nodes or links of several networks like IMT or PDC wireless networks and those of fixed telephony networks. In **Fig. 7**, SMSs A, B, and C are composed of a lot of SCs, as shown in **Table 2**. For example, SMS A has SCs developed for purposes such as confirming the alarm status of network links, executing important file updates of network nodes, and checking information about customers using these network nodes.

Moreover, these systems were constructed in different company divisions and at different times for different purposes. Thus, it is difficult to adjust the messages and processes in order to coordinate the systems.

By adjusting the interface specifications between the coordinated system (SMS A & system D) and SMSs B and C, we tried to incorporate systems B and C into the inference system D that infers restoration scenarios for responding to trouble caused by two or more problems in various networks. The procedure of our experiments was as follows.

(1) We constructed a knowledge base of SCs in SMSs by giving semantics to the SC message formats based on IF (interface) modeling. As shown in Table 2, we applied IF modeling to 3 SMSs of 19 software components among 267 software components. The interface specification documents describe semantic information such as status information of managed objects and procedures for messages. Therefore, we extracted semantics from these documents.

(2) We evaluated message mapping between the interface specifications of (SMS A & system D), SMS B, and SMS C.

(3) We evaluated the process mapping based on the message mapping results.

To improve the practicality, we implemented our mapping technique as a user-interactive tool. The system designer can confirm mapping results and delete mapping mistakes. The

Table 3 Example of useless word list used by experiment.

-, _, length, information, ID, status, data, sequence

mapping results help system designers to generate interface conversion rules.

To evaluate the accuracy, we defined two kinds of correct answers: positive and negative correct answer.

(1) Positive correct answers agree between classes and between instances. For example, when message parameters with the meaning of "alarm status" agree and message instances with the meaning of "normal alarm level" also agree, we call this mapping a positive correct answer. We can perform system cooperation by considering the value of status information by using positive correct answers.

(2) Negative correct answers are agreements between only classes. For example, when message parameters with the meaning of "node objective" agree, but message instances with the meaning of "node number 5" do not agree, we call this mapping a negative correct answer. We can perform system cooperation by integrating the status information of different services by using negative corrects.

We evaluated the accuracy of mapping results by checking the rate of correct answers in mapping results among all correct answers (recall) and the rate of correct answers in mapping results (precision).

In setting the parameters in our message mapping and in Glue, as described in Section 4, we prepared test data having the same type but half the size of the one used in our evaluation. For this test data, we deleted useless words (a partial list is shown in **Table 3**) as preparation. The words shown in Table 3 are English translations of the useless words that were originally in Japanese. Furthermore, we set the parameter θ to 0.7. We also applied almost the same weights to the approximation level of the class name, instances, and properties. The reason for choosing these values is that we extract correct answers selected not only by class name attributes but also by class instance set and class property attributes through the experiment. The difference between our message mapping and Glue is that our message mapping uses the KBR for calculating mapping results. System designers only set the appropriate value for the maximum number of hops for KBR, as

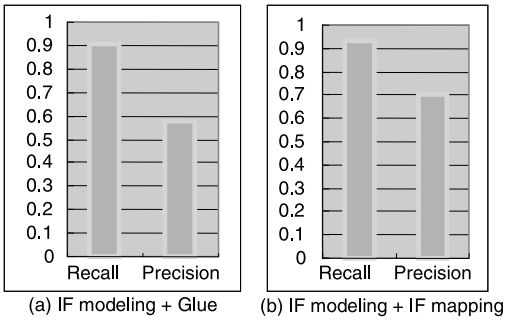

Fig. 8 Comparison of recall and precision.

mentioned in Section 4.2.3. We set the maximum number of hops to three in the evaluation and achieved mapping results with high accuracy in our evaluation in Section 7.2. Thus, we consider that the system designers should set a maximum value with a low value such as two or three when beginning to use message mapping and heuristically increase it in their mapping tasks.

7.2 Evaluating Message Mapping

We compared 1) Glue mapping based on our IF modeling and 2) our IF mapping with our IF modeling. We used our IF modeling to create the relationships between message formats and the semantics of these formats. By using our IF modeling, both Glue and our IF mapping could calculate the relationships between the semantics of message formats of different systems.

The average of results of 120 ($= 7 \times 6 + 7 \times 6 + 6 \times 6$) combinations of SCs in Table 2 are depicted in **Fig. 8**. In this evaluation, we checked the mapping results using the top six scores. The evaluation results show that our IF mapping improved the precision by about 14% compared with IF modeling + Glue and reduced the frequency of adjusting mapping results. This is because IF mapping reduced the number of mapping mistakes by reusing correct mappings of past coordination work from the KBR. We also improved the recall because the similarity in the instance set between classes was improved through the increase in instance-mapping knowledge in the KBR (see Fig. 10).

Then, we evaluated the reuse counts of the mapping results through the following three-step procedure. Previous techniques used these three steps, but our technique adds the new steps (1') and (2') of storing results from the first two steps.

Step (1) We evaluated message mapping when we set system A as the source system and

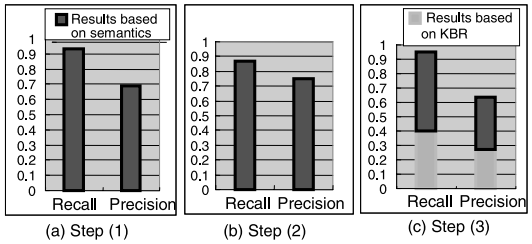

Fig. 9 An automation result by reusing mapping results.

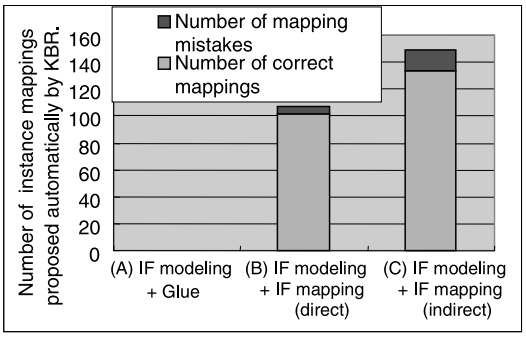

Fig. 10 Increase in correct answers in instance mapping results.

system B as the target system.

Step (1') IF mapping stored mapping results from steps (1) in the KBR.

Step (2) Then, we evaluated message mapping when we set system A as source system and system C as target system. We can reuse direct mapping results stored in step (1') for the message mapping in step (2).

Step (2') IF mapping stored 518 mapping results from steps (1) and (2) in the KBR.

Step (3) Next, we evaluated message mapping when we set system B as the source system and system C as the target system. We can reuse direct mapping and indirect mapping results stored in step (1') and (2') for the message mapping in step (3).

Graphs of the precision and recall of class mapping results are shown in **Fig. 9**. In step (3), we automatically extracted about 45% of the mapping results by reusing the indirect mapping results in the KBR. From these results, we conclude that we satisfied the two ideas for introducing the KBR described in Section 4.2.

The comparison between instance mappings of the previous technique (IF modeling + Glue) and our IF mapping is shown in **Fig. 10**. In step (2), our technique could reuse direct mapping results stored in KBR in step (1'). We show the result in step (2) in Fig. 10 (B). Our technique automatically extracted about 100 more correct mappings (about 9.5%) than the previous techniques. In step (3), our technique could reuse direct and indirect mapping results stored after step (2). We show the result in step (3) in Fig. 10 (C). Our technique extracted about 132 correct mappings (about 13.8%) in step (3), though the result of (3) include some mapping mistakes. This is because IF mapping reused past instance-mapping results that were confirmed as correct answers by the user in previous coordination work.

7.3 Evaluation of Process Mapping

We evaluated the process mapping. In the evaluation, we defined 22 correct mappings selected by an SMS designer out of 120 patterns in Table 2. We also evaluated the accuracy of the mapping results by using recall and precision. The system designer checked the results with the highest scores in the ranking and selected SCs with an interface specification applicable to the coordinated system.

In this evaluation, we set parameter α in Section 5.1 to 1 because this enabled us to get good results by using the correspondence of both input and output messages. Thus, we set this parameter to 1 to achieve good results on average.

We compared the recall between PMA1 and PMA2 by changing the number of results in the ranking that the user checks, as shown in **Fig. 11**. These results indicate that the recall of PMA2 is higher than that of PMA1 when the user checks the results with high scores. We also observed the recall when changing parameter β in Section 5.2. We found that, when the reuse frequency R was larger, the recall was higher: the recall was about 95%, as seen in the results for the 30 highest results, and was 100%, as seen in the results for the 35 highest results. Furthermore, the precision was also high because we could extract a lot of correct answers in the results with high scores. From this result, we also understand that the accuracy is higher when the reuse frequency R is somewhat large than when the reuse frequency R is too small. Thus, our process mapping is effective for extracting SCs for incorporation into a coordinated system.

8. Conclusion

We described IF modeling, which models the relationship between semantics and message

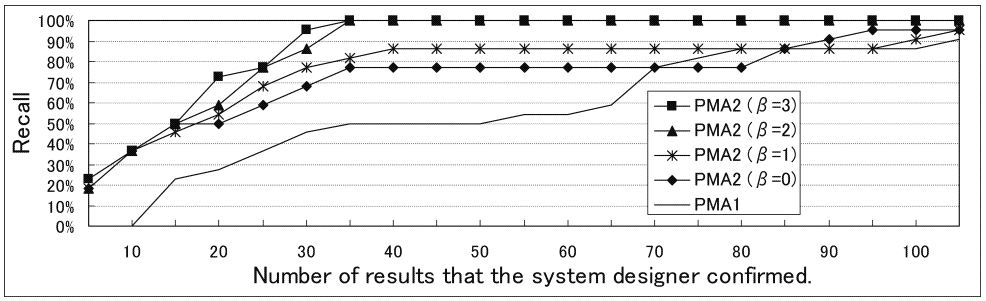

Fig. 11 Change in recall in process mapping.

formats in a machine-processable way. We proposed message mapping that executes the mapping between message formats by using semantic mapping results and extracted interface conversion rules. Then, we devised process mapping, which extracts software components with an interface specification applicable to a coordinated system. We confirmed the effectiveness of the interface mapping by using interface specifications of actual service management systems.

We will evaluate our interface-mapping technique in business environments that are more dynamic based on cooperation among a larger number of distributed software components. Furthermore, we will improve the adaptation of our technique to the cooperation of actual systems and help system designers adjust interface specifications of coordinated systems.

References

1) Berners-Lee, T.: An attempt to give a high-level plan of the architecture of the Semantic Web (1998).
2) Doan, A., Madhavan, J., Domingos, P. and Halevy, A.Y.: Learning to Map between Ontologies on the Semantic Web, *ACM World Wide Web Conference*, pp.662–673 (2002).
3) Feier, C. and Domingue, J.: WSMO Primer, *DERI*, http://www.wsmo.org/TR/d3/d3.1/v0.1/ (2005).
4) Liang, Q., Chakarapani, L.N., Su, S.Y.W., Chikkamagalur, R.N. and Lam, H.: A Semi-Automatic Approach to Composite Web Services Discovery, Description and Invocation, *International Journal of Web Services Research*, Vol.1, No.4, pp.64–89 (2004).
5) Madhavan, J., Bernstein, P.A., Doan, A. and Halevy, A.: Corpus-based Schema Matching, *International Conference on Data Engineering* (*ICDE*), pp.57–68 (2005).
6) Martin, D., et al.: OWL-S: Semantic Markup for Web Services, http://www.daml.org/services/owl-s/1.1/overview/ (Nov. 2004).
7) McGuinness, D.L. and v. Harmelen, F.: Web Ontology Language (OWL): Overview, W3C Recommendation, http://www.w3.org/TR/owlfeatures/ (2004).
8) Nakatsuji, M., Miyoshi, Y. and Kimura, T.: Proposal and verification of flexible interface mapping technique for automatic system cooperation based on semantics, *IEEE/WIC/ACM International Conference on Web Intelligence 2005*, pp.812–813 (2005).
9) Nakatsuji, M., Miyoshi, Y. and Kimura, T.: Verification of Message Mapping Technique based on Semantics for Flexible System Cooperation (in Japanese), *DBSJ Letters*, Vol.4, No.1, pp.37–40 (2005).
10) Ontology Alignment Evaluation Initiative web site. http://oaei.ontologymatching.org/
11) Otsuka, Y., Kimura, T., Miyoshi, Y., Nakatsuji, M. and Fukuda, A.: Proposal of Flexible Interface Technology for multidomain/multivendor Network Management, *Asia-Pacific Conference on Communications* (*APCC 2006*), CD-ROM (Sep. 2006).
12) Paolucci, M., Kawamura, T., Payne, T.R. and Sycara, K.: Semantic Matching of Web Services Capabilities, *Int'l Semantic Web Conf.*, pp.333–347 (2002).
13) RDF Core Working Group: *Resource Description Framework home page*, http://www.w3.org/RDF/ (2004).
14) Semantic Web Services web site. http://www.daml.org/services/
15) Sirin, E., Hendler, J. and Parsia, B.: Semi-automatic Composition of Web Services using Semantic Descriptions, *Proc. 1st Workshop on Web Services: Modeling, Architecture and Infrastructure in conjunction with ICEIS2003* (2002).
16) Xu, L. and Embley, D.W.: Discovering Direct and Indirect Matches for Schema Elements, *8th International Conference on Database Systems for Advanced Applications* (*DASFAA '03*), pp.39–46 (2003).
17) Yamato, Y., Nakatsuji, M. and Sunaga,

H.: Ubiquitous Service Composition Technology for Ubiquitous Network Environments (in Japanese), *IPSJ Journal*, Vol.48, No.2, pp.562–577 (2007).

(Received March 19, 2007)
(Accepted July 5, 2007)

(Editor in Charge: *Naohiko Uramoto*)

Makoto Nakatsuji graduated in 2001 in Applied Mathematics at Faculty of Engineering, Kyoto University. He completed Master's Degree in 2003 in Systems Science at Graduate School of Informatics, Kyoto University. He is currently working at the NTT Network Service Systems Laboratories. He is interested in Web mining, semantic-based search systems, and context-aware ubiquitous networks research. He received the JSAI SIG Research Award in 2007. He is a member of The Institute of Electronics, Information and Communication Engineers (IEICE) of Japan, The Japanese Society for Artificial Intelligence (JSAI) and The Database Society of Japan (DBSJ).

Yu Miyoshi received the B.E. and M.E. degrees in electronics, information, and communication engineering from Waseda University, Tokyo in 1998 and 2000, respectively. In 2000, he joined NTT Network Service Systems Laboratories, Tokyo, where he was engaged in R&D of network and service operation systems. He received the 2004 IEICE Young Researchers' Award. He is now studying automation settings for network elements. He is a member of the Institute of Electronics, Information and Communication Engineers (IEICE) and the Information Processing Society of Japan (IPSJ).

Yoshihiro Otsuka is Senior Research Engineer, Supervisor, Network Software Service Project, NTT Network Service Systems Laboratories. He received the M.E. degree in physical electronics from Tokyo Institute of Technology, Tokyo in 1985. He joined NTT in 1985. Since then, he has mainly been engaged in research on broadband switching systems and network management systems. He is currently working on the modeling of network management functions and multi-layered network management systems. He is a member of IEICE.

Miki Hirano received the B.E., M.E., and Ph.D. degrees in electronics engineering from Kyushu University in 1983, 1985, and 2002, respectively. Since joining the Musashino Electrical Communication Laboratories of NTT in 1985, she has been engaged in the development of switching system architectures, traffic management systems for broadband communication networks, and public ATM switching systems. She is currently involved in research and development of next-generation network systems and new network services.