意味情報に基づくインタフェースマッピングによるシステム連携手法の提案と評価

中辻　真† 三好　優† 木村　辰幸†

† 日本電信電話株式会社 NTT ネットワークサービスシステム研究所
〒 180-8585 東京都武蔵野市緑町 3-9-11
E-mail: †{nakatsuji.makoto,miyoshi.yu,kimura.tatsuyuki}@lab.ntt.co.jp

あらまし e ビジネスなどの企業活動は，ネットワーク (NW) 上の多様なシステム上で働くソフトウェア・コンポーネント (SC) の分散協調に基づいて実行される事が多くなっている．しかし，複数システム間のメッセージやプロセスといったインタフェース (IF) が，業務部門ごとに個別最適で設計されているため，IF 整合にかかるコストや導入までの開発期間が長く，ビジネスチャンスに即したサービス導入に問題が生じる．これに対し著者らは，急速に変化するビジネス環境でシステム設計者の目的に応じ，迅速かつ自動的にシステム連携を実現するフレキシブル IF 技術を確立するため研究を進めて来た．本稿では，オントロジ言語 OWL を用い，メッセージフォーマットとフォーマットの持つ意味情報の関係を知識ベースとしてモデル化する IF モデリング手法，及び様々なシステムの意味情報を半自動でマッピングする事でメッセージの差異を吸収するメッセージマッピング手法を提案する．そして，実運用 NW 管理システムの IF 仕様を題材とし提案手法の検証を実施する．更に，SC の再利用を特徴とするサービス指向アーキテクチャにならい，システムを構成する SC を自システムへ組み込み再利用するプロセスマッピング手法の提案も行う．
キーワード セマンティック Web サービス，オントロジ，OWL，サービス指向アーキテクチャ，NW オペレーション

A Proposal and Verification of Flexible Interface Mapping Technique for System Cooperation

Makoto NAKATSUJI†, Yu MIYOSHI†, and Tatsuyuki KIMURA†

† NTT Network Service Systems Laboratories, NTT Corporation
9-11 Midori-Cho 3-Chome, Musashino-Shi, Tokyo, 180-8585 Japan
E-mail: †{nakatsuji.makoto,miyoshi.yu,kimura.tatsuyuki}@lab.ntt.co.jp

Abstract Recently, more and more companies execute their business aims based on decentralized cooperation of software components (SCs) which work on various systems over NW. However, messages and processes between two or more systems are designed individually in each operations division. Therefore, the costs of system development for adjusting interfaces (IFs) are expensive, so the companies cannot introduce their services on dynamic business environment. For resolving such problems, we have advanced researches to establish Flexible IF Technology which cooperates the systems automatically on the purposes of system developers over dynamic business environment. In this paper, we propose IF Modeling Technique and Message Mapping Technique which model the relation between the message formats and semantics on the formats by using Web Ontology Language and execute the mapping between the message formats by using semantics. We evaluate our proposed methods based on the IF specifications of real NW management systems. Furthermore, we propose Process Mapping Technique which recycles SCs and puts them into the proces of developing systems, following the idea of Service Oriented Architecture.
Key words Semantic Web Services, Ontology, OWL, Service Oriented Architecture, NW Operation

1. はじめに

現在のシステムは，顧客データベースに代表されるように業務部門ごとに設計されているため，システムを連携する際にシステム間で交換されるメッセージが同じ意味で使用されていても，メッセージの持つメッセージ要素の識別 ID や型定義を定

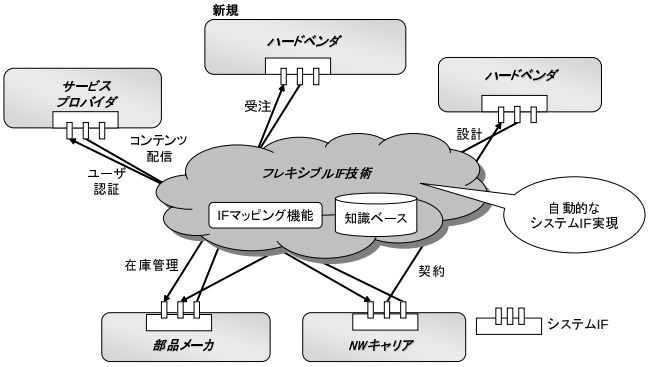

図 1　フレキシブル IF 技術イメージ

義するメッセージフォーマットが異なる事が多い．また，業務の実行手順を示すプロセスも，時々刻々と変化するビジネス環境に応じ各部門に固有な戦略に基づき設計されている [11]．こうした環境においては，既存業務に影響を与えないよう，各部門の戦略に沿ったメッセージやプロセスを維持しつつ適切な連携を行う必要があるが，例えばオペレーションシステムの設計に用いられているテーブル型のメッセージフォーマットを持つ従来型技術や CORBA [16]，Web Services(WS) [4] などのシステム連携技術では，システム固有のメッセージ及びプロセスの対応関係を人手で整合し連携を行う必要がある．そのため，インタフェース (IF) 整合にかかるコストやシステム導入までの開発期間に問題が生じる事が指摘されている [9,11,14].

上述したシステム連携に関する諸問題を解決し，ビジネスチャンスに即した企業活動やオペレーション業務に合致したサービス提供を行うため，著者らは，各システムが分散保持するサービス情報やネットワーク (NW) 管理情報，業務プロセスを設計者の目的に応じ自動連携する，フレキシブル IF 技術の確立を目指している (図 1 参照) [9,10]．そして，フレキシブル IF 技術により，複数システムの持つ知識を複合的に活用し，より高度な価値を創造する環境を提供する事を目的とする．

本稿では，フレキシブル IF 技術を確立するために必要となるメッセージとプロセスの整合を自動実現するための IF モデリング手法とメッセージマッピング手法，及びプロセスマッピング手法の提案を行う．IF モデリング手法では，システム間で交換されるメッセージフォーマットとメッセージフォーマットに割り当てられる意味情報の関係をオントロジ言語 OWL DL [13] を用い知識ベースとして機械識別可能にモデル化する．そしてメッセージマッピング手法では，複数知識ベースの意味情報間の関係を自然言語処理技術 [12] を活用しマッピングする事で，メッセージフォーマットのマッピングを自動実現する．特に，知識ベース間のマッピング結果を共通知識ベースとして蓄積し，過去のシステム設計知識として再利用する事でマッピングの自動化促進を試みる．そして得られたマッピング結果を用い IF を変換・再構築し，メッセージフォーマットの異なるシステム間でも連携を実現する．更に，従来型技術により構築されている実運用 NW 管理システム (NMS) の IF 仕様を題材としてメッセージマッピング手法のプロトタイプ実装を行い自動化の

表 1　既存システム連携技術の特徴

	従来型技術	CORBA	Webサービス
メッセージ	処理・サイズ小 記述形式固定	処理・サイズ小 記述形式固定 （形式は従来型技術と同様）	XMLベース 処理・サイズ大 記述形式固定
プロセス	記述なし	入出力メッセージのみ記述	入出力メッセージのみ記述
自動化の実現性	メッセージを人手で整合するため低い	メッセージ・プロセスを人手で整合するため低い	メッセージ・プロセスを人手で整合するため低い

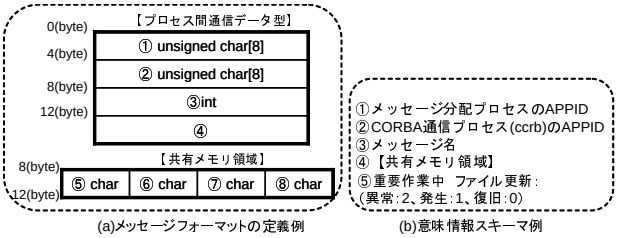

図 2　メッセージフォーマットにおける意味情報スキーマ欠如

検証を行う．またプロセスマッピング手法では，単一メッセージ間のマッピングではソフトウェア・コンポーネント (SC) 間でやりとりされるメッセージの順番や組合せといったプロセスを考慮した連携を実現できないため，サービス指向アーキテクチャ[14] にならい既存 SC を自システムへ組み込み再利用する．

以下，2. 章では，既存システム連携技術と関連研究について紹介を行い，3. 章では，IF モデリング手法を提案し，実運用 NMS の IF 仕様を題材とした知識ベースの試作結果を示す．そして，4. 章では，メッセージマッピング手法を提案し，実運用 NMS を対象としたシステム連携の自動化の検証実験結果を示す．さらに，5. 章において，プロセス再利用を実現するためメッセージマッピング手法を拡張したプロセスマッピング手法を提案し，最後に，6. 章の結論と将来の課題で結ぶ．

2.　既存システム連携技術と関連研究

本章では，既存システム連携技術として従来型技術，CORBA [16]，Web Services(WS) [4] の特徴を整理し，関連研究とし Semantic Web Services(SWS) [6]，オントロジマッピング技術 [7] およびサービス指向アーキテクチャ(SOA) [14] について述べる．

既存システム連携技術の特徴を自動化の観点より表 1 に整理する．従来型技術や CORBA はメッセージの処理コストやサイズが小さいため NW 管理など緊密な情報交換を行う領域に適している．一方 WS は，メッセージを XML ベースで記述しているため Web 上のエンドユーザ向けサービスなどテキスト形式の情報交換を行う領域に適している．このように各技術の適用領域は異なるため，WS のみでなく CORBA や従来型技術によるシステム構築も今後継続して行われると考えられる [16]．そのため著者らは，各技術に対し連携自動化を実現する事が重要であると考えているが，現状では各技術ともメッセージの記述形式が固定であり，連携を行う際には予めシステム間で交換されるメッセージフォーマットを人手整合する必要がある．

例えば，従来型技術におけるメッセージフォーマットでは，図 2-(a) に示すように，メッセージ要素の識別 ID はメッセージフォーマットを記述するファイル内の格納メモリ位置で定義さ

れ，それぞれのデータタイプは各メモリ位置に対するデータタイプ定義により別途表されている．しかし，図 2-(b) に示すようなメッセージフォーマットに割り当てられる意味情報スキーマとの関係が機械識別可能に記述されていない．そのため，メッセージフォーマットが異なるシステム間では，IF 仕様書等に記載されている情報を参照し，人手で意味を比較しメッセージフォーマットを整合しなければメッセージ交換ができない．

これに対し SWS [6] では，Web 上の情報に対しその背景となる知識を機械処理可能なメタデータとして記述する事で，様々なソフトウェアが自動で意味情報による処理を実行する事を目指すセマンティック Web 技術 [3] を WS におけるサービスの自動発見・選択・実行に適用する事を試みている．そのため WS で公開される IF に対し，オントロジ記述言語 OWL を拡張した OWL-S [6] を用い入出力メッセージ，サービスを行うための前提条件，サービスにより得られる効果の意味記述を行う．例えば，ユーザの問合せメッセージとサービスプロバイダの提供サービスに対する意味記述のマッチングを行い，意味情報に基づくサービスの自動発見・選択の実現を試みる研究がある [2,5]．しかし本研究の IF モデリング手法のように従来型技術や CORBA で構築されたシステムに対するセマンティック Web 技術の適用や，メッセージマッピング手法のようなシステム連携に必要となるメッセージフォーマット整合への提案が無い．

また，オントロジマッピングに関する研究として，PROMPT [7] では，概念 (クラス) 間のマッピングにおいて，クラスの名前やクラスに所属する実体値であるインスタンスを利用するだけでなく，クラスの性質を表す属性であるプロパティまで考慮する事でマッピングの精度向上を実現する事を試みている．しかし，全オントロジ間で整合性の取れる全体最適なクラスマッピングを計算する事を目的としており，システム連携のように時々のビジネス環境や個々のシステム設計者の目的によりマッピング結果が異なる領域には適用できない．これに対し本研究では，全オントロジ間で整合性を保証しなくても一部のオントロジの持つクラス間でマッピング出来れば，マッピング結果を共通知識ベースに蓄積する事で現在のマッピングへの再利用を促進し，連携の自動化を向上する事を図っている．

一方，システムを構成するソフトウェア・コンポーネント (SC) を再利用する事で，ビジネス戦略に従ったシステム構築と変更を迅速に実現する事を目指す SOA [14] が注目されているが，その実現のためには再利用可能な SC を抽出しシステムへ組み込む基盤技術を確立する必要がある．本研究では再利用可能な SC を抽出するプロセスマッピング手法を提案している．

3. IF モデリング手法

本研究ではメッセージフォーマットに意味情報を付与する事で自動的にメッセージフォーマットを整合する．そのため 3.1 節においてメッセージフォーマットと意味情報及び両者の関係を機械識別可能に記述する知識ベースを提案し，3.2 節で設計者が知識ベースを記述するための記述ルールの提案を行う．

3.1 知識ベースの提案

既存システム連携技術により構築されたシステムでは，メッセージフォーマットに意味情報が付与されていないため自動的に IF を整合する事ができない [9]．そこで本研究では，図 3 に示すようにメッセージフォーマットのスキーマであるフォーマットスキーマと意味情報スキーマを分離し両者の関係を機械識別可能に記述する知識ベースを提案する．これにより，意味情報に基づきシステムを検索・選択・実行・連携できる可能性がある．例えば連携に対しては，図 3 に示すように，意味情報スキーマ間のマッピングによりメッセージフォーマット間の自動マッピングができる．

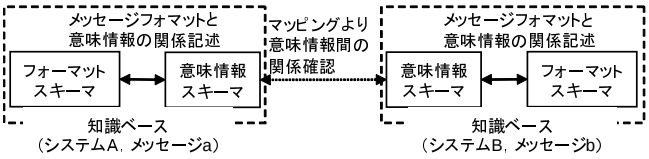

図 3　知識ベースの提案

3.2 記述ルールの提案

システム設計者が知識ベースを記述するためには，記述ルールの確立が必要である．そのため，NW 管理システム (NMS) やサービスシステムの様々な IF 仕様書を基に実運用システム IF の特徴の調査・分析を行った．そして，実際に知識ベースを試行錯誤しながら構築する事で，意味記述表現の中で特に従来型技術による IF に必要な表現を記述ルールとして抽出した．また Web Services(WS) の記述ルールは，OWL [13] で記述された意味情報スキーマと WS の IF 記述言語である WS Description Language (WSDL) で定義するメッセージフォーマットとの関係を記述できるため，OWL-S のグラウンディング [6] を利用する．

両スキーマを分離記述する事で，WS であっても従来型技術であっても意味情報スキーマはシステム設計者の目的に沿ったものとして同様の記述が可能となり，将来的に従来型技術を WS に置換しても構築された意味情報スキーマは再利用できる．

【知識ベース構築の要求条件】

システム間でメッセージを交換するには，実際にシステム間で交換されるデータであるメッセージ要素と，各メッセージ要素の識別 ID とデータタイプを識別する必要がある．またフォーマットスキーマと意味情報スキーマの関係を機械識別可能とするため，両者を分離記述した上で対応関係を記述する必要がある．そのため知識ベース構築には以下の要求条件を満たす必要がある．

（1） 意味情報スキーマとフォーマットスキーマ間の対応関係を機械識別可能とするため，両者を分離記述する事

（2） スキーマ記述ができる事

（3） 過去のマッピング結果を再利用するため，各クラス・インスタンスが所属するシステムやメッセージを識別できる事

（4） 連携の際，メッセージ要素を整合する必要があるためメッセージ要素を識別できる事

（5） 意味情報・フォーマットスキーマ間の関係を識別できる事

（6） 各システムの IF の持つ特性を表現するためメッセージ要素がどのようにシステム間で交換されるかを識別できる事

（7） データタイプの整合も実現するため，メッセージ要素の持つデータタイプを識別できる事

表 2　OWL DL による意味情報記述

クラス公理				プロパティ公理		インスタンスによる事実の記述
rdfs:subClassOf	owl:equivalentClass	owl:oneOf	プロパティの制約	オブジェクトプロパティ		owl:sameAs
			owl:allValuesFrom	rdfs:range	rdfs:domain	
参照クラスのサブクラスとして，主語クラスの必要条件を形成	参照クラスと同じインスタンスを持つクラスという，主語クラスの必要十分条件を形成	主語クラスのインスタンスを過不足なく列挙する	主語クラスの全てのインスタンスについて，プロパティの全ての値が目的語クラスのインスタンスであることを示す	プロパティの目的語は，参照クラスのインスタンスである	プロパティの主語は，参照クラスのインスタンスである	2つのインスタンスが同一であることを示す

なお，意味情報スキーマとフォーマットスキーマ間の対応関係を機械識別可能にする事が必要であるため，知識ベースの記述には，クラス間の関係を機械識別可能に記述できる OWL DL [13] を用いる．ここで，OWL DL による記述法則の中でも特に本稿における知識ベース構築に用いる法則を表 2 に示す [13]．OWL DL におけるクラスは，同様の性質を持つ個体をグループ化し，その性質を論理的に表現するための機能を提供する．クラスは，表 2 に示すようなクラスの持つ個体であるインスタンスの列挙などのクラス表現を用いクラスを定義する．また個体同士の関係 (オブジェクトプロパティ) や個体とデータ値の関係 (データタイププロパティ) を定義するプロパティは，RDF スキーマの rdfs:range や rdfs:domain で値域，定義域を記述できる．さらに，クラスのインスタンスに関する公理を用い，例えば owl:sameAs により 2 つのインスタンスが同値である事を記述できる．

【記述ルール】

OWL DL による知識ベース記述ルールを上記要求条件と対応した形で以下に示す．

（1） ルートクラスは “知識ベース” としサブクラス “ 意味情報” と “メッセージフォーマット” を持つ．意味情報スキーマの所属クラスは “意味情報” の下位クラスとし，フォーマットスキーマも同様とする．

（2） クラススキーマは，owl:subClassOf を用い表現する．

（3） 対象システムを主語クラス “知識ベース” のプロパティ“belongingSystem” の述語クラス “システム” のインスタンスより識別し，対象メッセージもプロパティ“belongingMessage” より識別する．

（4） メッセージ要素の識別 ID をクラスとし意味情報スキーマの対応オブジェクトもクラスとして記述する．同様にメッセージ要素をインスタンスとして記述し，意味情報スキーマの対応オブジェクトもインスタンスとして記述する．

（5） フォーマットスキーマと意味情報スキーマの関係は，クラス間の同値関係を示す owl:equivalentClassOf とインスタンス間の同値関係を示す owl:sameAs で記述する．

（6） 各クラスのインスタンスがインスタンス化する際の特性をクラスの構成要素記述により表現する．1 例とし，システム間でメッセージを交換する際，メッセージフォーマットのインスタンスの内 1 つがインスタンス化されるという特徴は，owl:oneOf で表現する．

（7） データタイプはフォーマットスキーマの所属クラスが持つプロパティ“hasType” の述語クラス “Type” のインスタンスより識別する．

本記述ルールにより，従来型技術や CORBA に対しても OWL-S で記述されたサービスの入出力メッセージパラメータを従来型技術のメッセージ要素と対応でき，現状 WSDL のみに対応している OWL-S を従来型技術へ適用できる．

3.3　NMS を対象とした知識ベースの試作

記述ルールの正当性確認のため，実際に運用されている NW 装置に対する 3 種類の NMS と 1 種類の情報取得システムの IF を対象に知識ベースを試作した．

NW オペレーション領域における意味情報は NW やノードなど管理対象の状態情報であり IF 仕様書はこうした情報を記載しているため，仕様書より意味情報を抽出し知識ベースを試作した．今回用いた NMS は従来型技術で構築されテーブル型のメッセージフォーマットを備えるが，仕様書には意味情報と

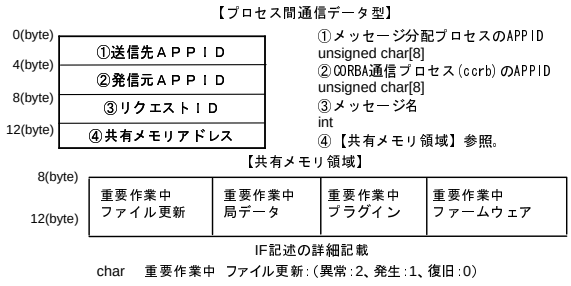

図 4　NMS の IF 仕様書例の一部

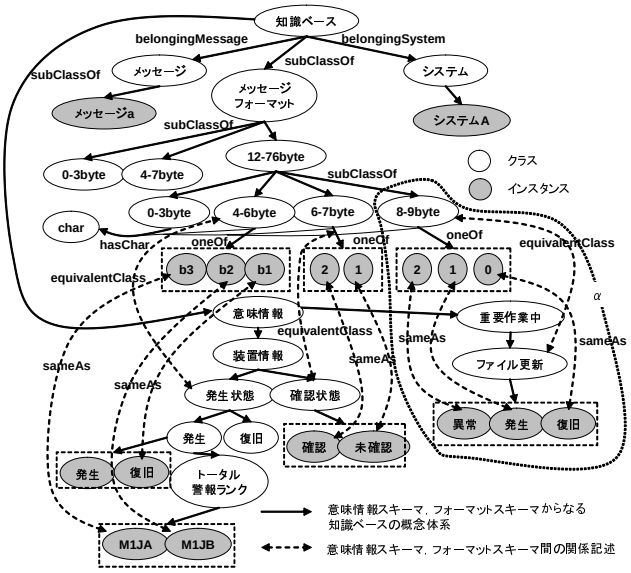

図 5　試作した NMS の知識ベースの一部

```
<owl:Class rdf:ID="ファイル更新">
 <owl:equivalentClass>
  <owl:Class>
   <owl:oneOf rdf:parseType="Collection">
    <ファイル更新 rdf:ID="復旧">
     <owl:sameAs>
      <アドレス8-9byte rdf:ID="data0">
       <owl:sameAs rdf:resource="#復旧"/>
      </アドレス8-9byte>
     </owl:sameAs>
    </ファイル更新>
    <ファイル更新 rdf:ID="発生">
     <owl:sameAs>
      <アドレス8-9byte rdf:ID="data1">
       <owl:sameAs rdf:resource="#発生"/>
      </アドレス8-9byte>
     </owl:sameAs>
    </ファイル更新>
    <ファイル更新 rdf:ID="異常">
     <owl:sameAs>
      <アドレス8-9byte rdf:ID="data2">
       <owl:sameAs rdf:resource="#異常"/>
      </アドレス8-9byte>
     </owl:sameAs>
    </ファイル更新>
   </owl:oneOf>
  </owl:Class>
 </owl:equivalentClass>
 <owl:equivalentClass>
  <owl:Class rdf:ID="アドレス8-9byte"/>
 </owl:equivalentClass>
 <rdfs:subClassOf rdf:resource="#重要作業中"/>
</owl:Class>
```

```
<owl:Class rdf:about="#知識ベース">
 <owl:equivalentClass>
  <owl:Restriction>
   <owl:allValuesFrom rdf:resource="#メッセージa"/>
   <owl:onProperty>
    <owl:ObjectProperty rdf:ID="BelongingMessage"/>
   </owl:onProperty>
  </owl:Restriction>
 </owl:equivalentClass>
 <owl:equivalentClass>
  <owl:Restriction>
   <owl:allValuesFrom rdf:resource="#システムA"/>
   <owl:onProperty>
    <owl:ObjectProperty rdf:ID="BelongingSystem"/>
   </owl:onProperty>
  </owl:Restriction>
 </owl:equivalentClass>
</owl:Class>
<owl:Class rdf:about="#アドレス8-9byte">
 <owl:equivalentClass>
  <owl:Restriction>
   <owl:allValuesFrom rdf:resource="#char"/>
   <owl:onProperty>
    <owl:ObjectProperty rdf:ID="haschar"/>
   </owl:onProperty>
  </owl:Restriction>
 </owl:equivalentClass>
</owl:Class>
```

図 6　試作した NMS の知識ベースの一部 (OWL 記述)

メッセージフォーマットが混在して記載され (図 4)，意味情報スキーマとフォーマットスキーマの関係を機械識別出来ない．また現在の仕様書は UML(Unified Modeling Language) [15] により記述される事が多くなっているため，IF 仕様書を提案する記述ルールに従い UML 記述してから OWL 記述による知識ベースへ変換できればメッセージマッピング手法は実用的になる．そこで知識ベースエディタ protege (注1) が，uml backend plugin (注2) を用いる事で UML 記述されたクラスダイアグラムを OWL 記述に自動変換出来るため protege により知識ベースを試作した．試作した知識ベースのクラス図を図 5 に示す．また，図 5 の内点線で囲まれた領域 α に対する OWL 記述を図 6 に示す．

図 5 に示すように，“意味情報” と “メッセージフォーマット” は “知識ベース” のサブクラスであり，“重要作業中” は “意味情報’ ’ のサブクラス，“8-9byte” は “ 12-76byte” のサブクラスである．図 6 に示すようにクラス “ファイル更新” は，<owl:equivalentClass>タグにより示される通り “ 8-9byte” と同じインスタンスを持っており，<rdfs:subClassOf>タグにより示される通り “重要作業中” のサブクラスである．また，<owl:oneOf>タグにより示される通り，“復旧”，“発生”，“ 異常” という 3 つのインスタンスを過不足なく保持し，各インスタンスは，<owl:sameAs>タグにより示される通りクラス “8-9byte” の持つインスタンス “data 0”，“data 1”，“data 2” と同値である．更に，プロパティ<owl:ObjectProperty rdf:ID=”BelongingSystem”/>に対しタグ<owl:allValuesFrom>で与えられている制約により “知識ベース’ ’ 以下のサブクラスに所属するインスタンスは全てクラス “システム” のインスタンスとなる．このように，意味情報スキーマとフォーマットスキーマの関係や知識ベースの所属システム，所属メッセージが機械識別可能に記述されている．

結果として，従来型システムの知識ベースを提案する記述ルールに従い試作し，知識ベースを機械計算可能な OWL DL で記述できる事，及び記述ルールの正当性を確認できた．今後は，UML 記述の IF 仕様書への本記述ルールの適応性の確認，及び NMS 以外のシステムに対する検証を進める．

4. メッセージマッピング手法の提案

本章では，連携自動化に必要となる (1) メッセージフォーマット間の自動マッピング，(2) マッピング結果を用いた IF 記述の変換を実現するメッセージマッピング手法について提案する．

4.1 メッセージフォーマット間の自動マッピング

本研究では，複数システムの意味情報スキーマ間を自動マッピングし，その結果からメッセージフォーマット間を自動マッピングする．マッピング方式として以下の方式を検討した．

1 名前のみによるマッピング
2-1 クラス諸属性利用方式
2-2 方式 2-1 に加えパターンマッピング
2-3 方式 2-1 に加え蓄積したマッピング結果を再利用
2-4 方式 2-1 に加え，方式 2-2，2-3 を両方実施

本研究では，提案する記述ルールに基づきメッセージフォーマットの特徴をクラスの持つインスタンス集合やクラス構成要素を用い記述する．そのため，インスタンス集合やクラス構成要素の近似度を利用した意味情報に基づくマッピングが実現でき精度を高める事が出来ると考え，単純な名前一致のみによるマッピングである方式 1 ではなく，方式 2 を選択する．以下，方式 2-1～ 方式 2-3 について説明する．方式 2-4 については，方式 2-1～ 方式 2-3 を組合せたものであるため説明は省略する．

方式 2-1：クラス諸属性利用方式

異なるシステムの意味情報スキーマに所属するクラス間の近似度をクラス名，インスタンス集合，クラス構成要素といったクラスの諸属性より求める [9]．諸属性をマッピングに利用する事で従来の名前のみのマッピングより精度の高いマッピングを実現できる．以下，ある意味情報スキーマ I のクラス C_i(ソースクラス) から見た，異なる意味情報スキーマ J のクラス C_j(ターゲットクラス) との間の近似度 $S(C_{ij})$ を計測する際の例を基に説明する．まず，クラス C_i から見た C_j のクラス名の近似度を $S(N)_{ij}$ とする．クラス名の近似度の計測には，クラス名が日本語である場合には，形態素解析により形態素に分割した上でベクトル空間手法を適用し名前属性の近似度を計算する [8]．例えば,「ファイル更新」というクラス名は形態素解析により「ファイル」と「更新」という形態素に分解できる．同じく,「ファイル更新情報」というクラス名は,「ファイル」と「更新」と「情報」という形態素に分解できる．ここで，形態素「ファイル」,「更新」,「情報」がクラス名に含まれる割合を要素とするベクトルを形成する．そうすると「ファイル更新」のベクトルは (1/2 1/2 0) となり,「ファイル更新情報」のベクトルは (1/3 1/3 1/3) となる．このベクトルの内積に基づきクラス名の近似度を計算し，上記例では近似度は 1/3 となる．

次に，クラスの持つインスタンス集合間の近似度を計測する．そのため，まず，ソースクラス C_i に所属するインスタンス I_i から見た，ターゲットクラス C_j に所属するインスタンス I_j の名前の近似度 $S(I)_{ij}$ を計測する．クラス名の近似度計測と同じく，ベクトル空間手法を適用し名前属性の近似度を計算する [8]．そして，ヒューリスティックな閾値 θ を用い，$S(I)_{ij} > \theta$(式 (1)) を満たすならば，インスタンス I_i と I_j は，対応すると考える．次に，クラス C_i と，クラス C_j の持つインスタンス集合を U_i，U_j とすると，インスタンス集合間の近似度 $S(U)_{ij}$ は $S(U)_{ij} = \frac{|U_i \cap U_j|}{|U_i \cup U_j|}$ と表される [1]．

続いて，クラスの構成要素間についても近似度を計測する．ここで，ソースクラス C_i から見たターゲットクラス C_j の構成要素間の近似度を $S(O)_{ij}$ とする．本研究では，3.2 節で述べたようにクラスに所属するインスタンスのうち 1 つのみがインスタンス化されるという特徴を owl:oneOf 要素で記述する．そこで，クラスの構成要素間の近似度を，クラス C_i と C_j 両方に owl:oneOf 要素がある場合，または無い場合は，$S(O)_{ij} = 1$，クラス C_i と C_j のどちらかにのみ owl:oneOf 要素がある場合は $S(O)_{ij} = 0$ とする．

次にクラス名，インスタンス集合，クラス構成要素に対し，クラス間の近似度へ与えるヒューリスティックな重み係数 κ，λ，μ を決定する．そして，クラス間の近似度 $S(C_{ij})$ を

(注1)：http://protege.stanford.edu/
(注2)：http://protege.stanford.edu/plugins/uml/

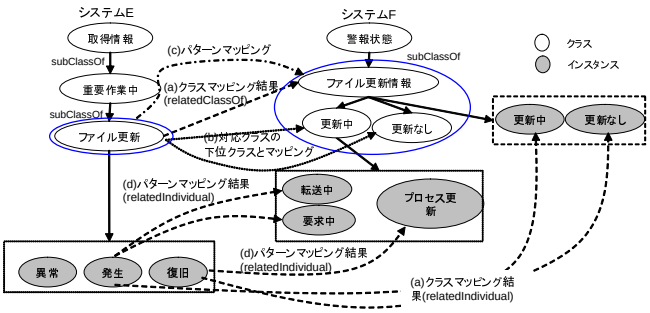

図 7　パターンマッピング例

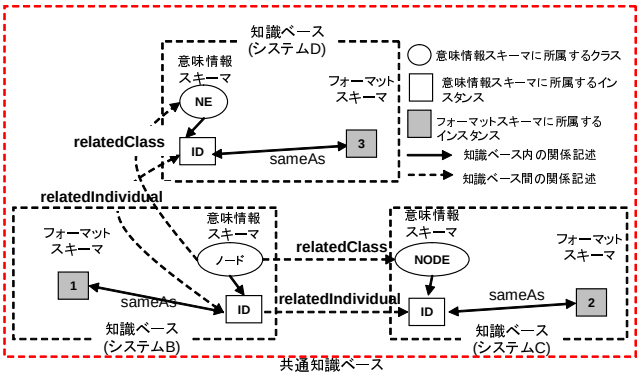

図 8　共通知識ベース具体例

$S(C_{ij}) = \kappa * T(S(N)_{ij}) + \lambda * T(S(U)_{ij}) + \mu * T(S(O)_{ij})$(式 (2)) により与える．ここで，重みを考慮する関数 T として，知識ベース J に所属するターゲットクラスを母集合とし，平均 $\overline{x}$，分散 ${\sigma_x}^2$ をもつ式，$T(x_i) = \frac{x_i - \overline{x}}{\sigma_x}$ を与える．そして，ターゲットクラスを $S(C_{ij})$ によりランキングし，クラス間の関係をクラスの持つプロパティ，インスタンスを含めてユーザへ提示する事で，ユーザはクラス同士およびクラスに所属するインスタンス同士をマッピング可能か判定する．

方式 2-2：パターンマッピング方式

設計者の目的等により，システム毎に取扱う情報の粒度が異なる事が多く，意味情報スキーマの抽象度には差異が生じる．この場合，方式 2-1 ではマッピング結果の再現率が低くなる．そのため本研究では，M 個のクラスと N 個のクラスをパターン化してマッピングするパターンマッピングを提案する．

図 7 において，方式 2-1 ではクラスマッピングによりシステム E の “ファイル更新” とシステム F の “ファイル更新情報” がマッピングされる．しかしこの結果では，システム F は “ファイル更新情報” の “更新中’ ’ 配下のインスタンスとマッピングが行えない．そこで方式 2-2 では，図 7 の手順 (b) で示すように対応クラスの親子クラスとの関係をユーザ確認させる．これにより (“ファイル更新”) という 1 クラスからなるサブオントロジと (“ファイル更新情報”，“更新中”，“更新なし”) からなるサブオントロジ間がパターンマッピングされ (手順 (c))，システム E のインスタンス “発生” とシステム F のインスタンス “転送中” 等がマッピングできる (手順 (d))．

方式 2-3：マッピング結果再利用方式

方式 2-1 におけるクラスマッピング結果を用い，インスタンス間のマッピングの人手の確認回数が削減できる．しかし自動化促進のために更に人手による確認回数を少なくする必要があるため，方式 2-1 に加え共通知識ベースを導入し，過去のマッピング結果を蓄積し以降のマッピングに再利用する．

```
<owl:Class rdf:ID="クラスa">
<owl:ObjectProperty rdf:ID="システムA"/>
<relatedClass>
  <Class rdf:ID="クラスb">
    <owl:ObjectProperty rdf:ID="システムB"/>
  </Class>
</relatedClass>
</owl:Class>
```

(a)クラス間の対応関係記述

```
<owl:Class rdf:ID="クラスa">
<owl:ObjectProperty rdf:ID="システムA"/>
<クラスa rdf:ID="インスタンス">
  <relatedIndividual>
    <owl:Class rdf:ID="クラスb">
      <owl:ObjectProperty rdf:ID="システムB"/>
      <クラスb rdf:ID="インスタンス"/>
    </owl:Class>
  </relatedIndividual>
</クラスa>
</owl:Class>
```

(b)インスタンス間の対応関係記述

図 9　クラス・インスタンス間の対応関係記述

共通知識ベースの具体例を図 8 に示す．本研究は動的なシステム連携を指向するため，共通知識ベースでは意味情報スキーマ間のマッピング結果を，あるシステム設計者から見て他システムのメッセージフォーマットが自システムのメッセージフォーマットと対応関係を示すが，その他のシステム設計者からはその対応関係を保障しない形で蓄積する．つまり，あるシステム設計者から見て他システムのクラスやインスタンスが狭域的に一致すれば，関連性を持つとして関係記述しシステム設計知識として蓄積する．そのため知識ベース間の関係は，整合性を保証しない言語 RDF により記述する．具体的には，クラス間の対応関係記述は，所属インスタンス全てが一致していなくても一部のインスタンスが一致していれば関係記述 (relatedClass 記述) を行い (図 9-(a))，インスタンス間の対応関係の記述は，共通知識ベースに所属する全システム間で広域的に同値でなく一部のシステム間で狭域的に同値であるだけでも関係記述 (relatedInstance 記述) を行い再利用性を高める (図 9-(b))．

例えば，図 8 ではシステム B の設計者から見てシステム B の (ノード- ID) と変換可能な候補は，システム C の (NODE - ID) やシステム D の (NE - ID) である事が関係記述より自動的に分かる．さらに，システム C の設計者から見て，システム C の (NODE - ID) とシステム B の (ノード- ID) やシステム D の (NE - ID) が変更可能性を持つ事が分かるが，システム連携に再利用するには設計者の確認が必要となる．

自動マッピングの評価方法

提案方式によるマッピング自動化の評価のためマッピング結果判定に要する人手回数とマッピング結果の精度を比較する．精度の尺度は，マッピング結果中の正解が全正解に占める割合 (再現率) とマッピング結果中の正解の割合 (適合率) を用いる．

4.2　マッピング結果を用いた IF 変換

新規サービスの為のシステム連携例を基にマッピング結果を用いた IF 変換を説明する．例として複数システムの警報情報を一括管理するサービスを想定し，図 10 に示すようなメッセージフォーマット間のマッピング結果より IF 変換の評価を行う．ここで図 10 において，クラス諸属性利用方式 (方式 2-1) に基づきクラス名の近似度より，“トータル警報ランク” と “トータル警報情報” というクラスがマッピングされたと仮定する．そして，人手によるインスタンス間のマッピングまたはマッピング

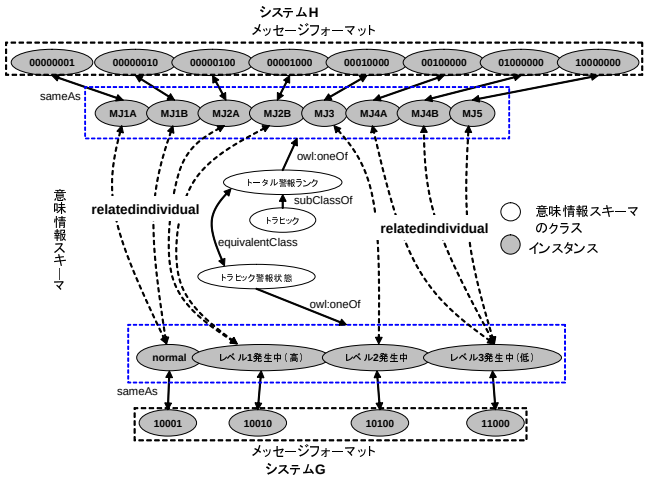

図 10　メッセージフォーマット間のマッピング

表 3　実験で用いた無駄語リスト

-	_	長	の	は	情報	ID	状態	データ	番号	を	現した	から	表す	ともに	が	する

結果再利用方式 (方式 2-3) より，“トラヒック警報状態” のインスタンス “レベル 2 発生中’ ’ と “トラヒック警報ランク” のインスタンス “MJ3” がマッピングできたと仮定する．このマッピング結果から意味情報スキーマとフォーマットスキーマの関係を機械計算し，インスタンス “10100” とインスタンス “00010000” がマッピング可能と自動判別でき，これより IF 変換則を生成する事でメッセージフォーマットが異なるシステムを連携できる．

4.3　NMS を対象としたマッピングシステムの実験

実運用 NMS の IF 仕様を題材としたユーザ対話型のマッピングツールの実装を行い，提案手法による連携自動化の検証を行った．現段階では 3.3 節で試作したうち 2 種類の NMS の持つ 15 個のソフトウェア・コンポーネント (SC) に対する知識ベースに提案手法を適用しクラス諸属性利用方式 (方式 2-1) とマッピング結果再利用方式 (方式 2-3) に対し評価している．なお実験で用いた IF 仕様書を用いマッピング判定を行った．

最終的なマッピング結果にマッピング誤りがあるとシステム連携として実用的でない．そこで実験では，クラス間の自動マッピング結果を近似度によりランキングしユーザ確認させマッピング誤りを除去するユーザ対話型のマッピングツールを実装した．以降で示す結果は，クラス名，インスタンス名の近似度計測の前準備として，表 3 に示すような無駄語を削除している．また 4.1 節の式 (2) において，$\kappa = 0.35$，$\lambda = 0.35$，$\mu = 0.3$ とした．これは，クラス名のみではマッピング結果として選択されない正解をインスタンス集合の近似性より選択できるケースが多く存在する事と，クラス構成要素に基づくマッピングによりメッセージ要素の交換形式の異なる結果を除去できる事を定性的に確認できた為である．またこの値は NW やノードの状態など管理対象の状態情報を交換する NMS 連携に対しては共通利用できる事を確認した．なお形態素解析には，Sen(注3) を利用している．

(注3)：http://ultimania.org/sen/

表 4　従来手法と提案手法の人手マッピング回数の比較 (方式 2-1)

	人手による確認回数
従来手法	25641
メッセージマッピング手法（ランキング2位）	468
メッセージマッピング手法（ランキング1位）	374

表 5　提案手法によるマッピングの再現率・適合率 (方式 2-1)

	クラスマッピング（レベル1）		クラスマッピング（レベル2）	
	ランキング2位	ランキング1位	ランキング2位	ランキング1位
再現率	143/169 ≒0.846	112/169 ≒0.662	229/290 ≒0.790	177/290 ≒0.610
適合率	143/468 ≒0.306	112/374 ≒0.299	229/468 ≒0.489	177/374 ≒0.473

表 6　マッピング結果の再利用による自動化の検証結果 (方式 2-3)

再利用回数	マッピング結果中の正解数（レベル2）	再利用回数 / マッピング結果中の正解数（レベル2）	再利用結果中のマッピング誤り数
138	229	138/229 ≒60.3%	0

表 7　インスタンス間の自動マッピング結果中の正解数の比較

属性利用（方式2-1）	共通知識ベース再利用（方式2-3）	増加数
328	359	31

メッセージフォーマットの差異整合における人手の確認回数を 2 システムの仕様書を人手で見比べる従来手法と，方式 2-1 により得られるクラス間の自動マッピング結果のみを人手整合する場合とで比較した結果を表 4 に示す．表ではユーザに対しランキングを 2 位まで確認させた場合と 1 位のみ確認させた場合を示す．また表 5 に，方式 2-1 により得られるクラスマッピングの再現率・適合率を示す．ここで 2 つのクラスの意味が等しい状態情報を取扱っており管理対象も同等である場合 (レベル 1) と，2 つのクラスが意味的に等しい状態情報を取扱っているが管理対象が異なる場合 (レベル 2) の 2 種類に分け再現率・適合率を算出した．レベル 2 の検証の理由は，管理対象が異なっていても状態情報の意味が一致すれば，状態情報を統合して扱う必要があると判断したためである．

従来手法と比較し方式 2-1 は人手による対応関係の確認回数を削減できる (表 4)．またランキングを 2 位まで確認すると確認回数は増えるが再現率が向上する．更に，適合率はほとんど変化がない (表 5)．これらより再現率を高くする事で IF の差異吸収にかかる人手の整合コストを小さくする必要がある一方，最終的なマッピング結果から誤りを除去する必要がある場合は，マッピング結果の上位をユーザ確認させる事が有効である．なおランキング 2 位と 3 位間の再現率向上は小さかった．

結果として NMS 間の IF 整合における人手マッピングと提案手法の比較を行い，提案手法がユーザ確認回数を 98.2% 程度削減できる事を確認した．更にクラスマッピングの適合率は 30% から 50% 程度であるが，結果をランキングしユーザ確認させる事で，高い再現率を持ちつつ適合率を補完する事ができた．

また方式 2-3 に対し同じ環境で実験を行った．表 6 に共通知識ベースによる自動化の検証結果を示す．複数 SC の知識ベー

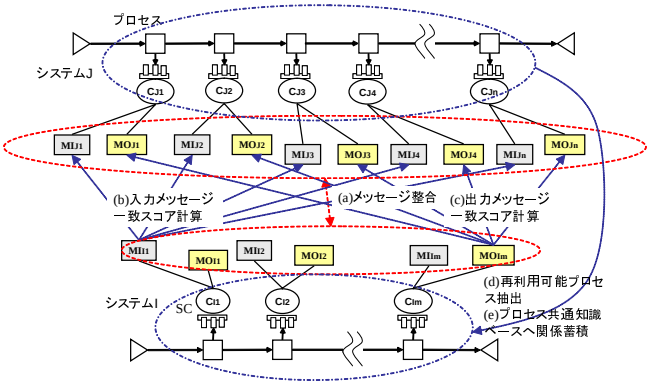

図 11　プロセスマッピング手法の手順

ス間でマッピングを繰り返し結果を蓄積・再利用する事で，最終的な再利用回数がマッピング結果中の正解数に占める割合は 60.3% にのぼり再利用結果に誤りが含まれていない事も分かった．これは，今回の実験は 2 システムの持つ 15 個の SC の知識ベース間でマッピングを行ったため，特に再利用されるマッピング結果が多い事もあるが，自動化に対し方式 2-3 が有効である事が分かる．また，表 7 は方式 2-1 と方式 2-3 で得られるインスタンス間の自動マッピング結果中に含まれる正解数を比較している．これにより方式 2-3 はインスタンス間のマッピング結果も蓄積・再利用し，方式 2-1 よりも約 9.5% 正解をユーザ提示できる事が分かる．これらより方式 2-3 によりマッピング自動化が促進できる事，マッピング精度向上が期待できる事が分かった．

5.　プロセスマッピング手法の提案

メッセージマッピング手法では，単一メッセージ間の自動マッピングを行うが，ソフトウェア・コンポーネント (SC) 間でやりとりされるメッセージの順番や組合せといったプロセスを考慮したシステム連携を実現できない．つまり，出力メッセージの持つメッセージフォーマットを整合できても，出力メッセージが次にどの SC の入力メッセージに対応するのかを判別できず，入出力を人手で判断し連携させる必要がある．そのため本研究では，SC の再利用を特徴とするサービス指向アーキテクチャ[14] にならい既存 SC を自動的に自システムへ組み込み再利用するプロセスマッピング手法の提案を行った．プロセスマッピングを実現する方式として以下の 2 つが考えられる．

（1）　入出力メッセージの近似度が高いプロセスを抽出

（2）　プロセスに効果・前提条件等の意味記述を与え，入出力メッセージ及び効果・前提条件の近似度が高いプロセスを抽出

Web Services(WS) などのプロセス記述には現状では効果・前提条件に対する記述は与えられていないため，本研究では，まずは方式 1 を選択しマッピングアルゴリズムの設計を行った．しかし，Semantic Web Services や WS Choreography Description Language 等では効果・前提条件を用いたプロセス記述を検討しているため，今後は方式 2 についても検討を進め方式 1 との比較実験を進める．以下に方式 1 のアルゴリズムを示す (図 11).

(a)　システム I がシステム J と連携を行うためメッセージマッピングを行い SC 間のメッセージフォーマットの差異を吸収

(b)　システム J の SC，C_{Ji} とシステム I の SC，C_{Ij} の組合せ全てに対し，C_{Ji} の入力メッセージ要素と C_{Ij} の入力メッセージ要素の和集合 U と積集合 N を計算し，C_{Ij} から見た C_{Ji} の SC の一致スコア $S_{IjJi} = \frac{|N|}{|U|}$ を計算

(c)　同じく，システム I のある SC，$C_{Ik}(1 \leqq k \leqq m)$ の出力メッセージ要素 MO_{Ik} と，システム J のある SC，$C_{Jl}(1 \leqq l \leqq n)$ の出力メッセージ要素 MO_{Jl} の和集合 U と積集合 N を計算し，SC の一致スコア $S_{IkJl} = \frac{|N|}{|U|}$ を計算

(d)　(b) と (c) の結果より，システム J の SC の内，入力メッセージ集合がシステム I のある SC の入力メッセージに包含される SC を起点とし，出力メッセージの一部がシステム I のある SC の出力メッセージの一部と一致する SC を終点とするサブプロセスを検索し，起点・終点両方の SC の一致スコアが大きい順に組み合わせ，ランキングし再利用性のあるプロセスとしてシステム設計者へ提示

(e)　設計者が再利用性があると決定したプロセスは，プロセス知識ベースへ蓄積し，以降のシステム設計へ再利用

本手法より，設計者はプロセス知識ベースへ問合せ (入出力メッセージ) を発行し再利用可能プロセスを獲得，自システムへ組込む事でシステム設計業務を自動化できる．

6.　結論と今後の課題

本稿では自動的なシステム連携を実現するため，フォーマットスキーマと意味情報スキーマの対応関係を知識ベースとしてモデル化する IF モデリング手法を提案し，実運用 NMS を対象とする試作により実現性を確認した．また，意味情報スキーマ間のマッピングによりメッセージフォーマット間の差異を自動吸収し IF の再構築を実現するメッセージマッピング手法を提案し，実運用 NMS を対象としたマッピングツールの実装により 2 種類の NMS 間での IF 整合において方式 2-1，方式 2-3 が自動化の有効性を持つ事を確認できた．さらに，システムを構成する SC を動的に自システムへ組み込み再利用するプロセスマッピング手法についても提案を行った．今後は今回試作した 4 種類のシステムの知識ベースに対し方式比較実験を進め再現率向上の検証やマッピング結果によるシステム IF の自動再構築の検証を進める．また，プロセスマッピング手法については，NW 管理業務の SO フローへ適用した実装により有効性の検証を進める．

文　　献

[1] Anderberg, M. R.: Cluster Analysys for Applications, *Academic Press* (1973).

[2] Benatallah, B., Casati, F., Toumani, F. and Hamadi, R.: Conceptual Modeling of Web Service Conversations, *Int'l conf. Advanced Information Systems Engineering*, pp. 449–467 (2003).

[3] Berners-Lee, T.: An attempt to give a high-level plan of the architecture of the Semantic Web (1998).

[4] Booth, D., Haas, H. et al.(eds.): *World Wide Web Consor-tium (W3C), Web Services Architecture*, http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/ (2004.2).

[5] M. Paolucci, N. S. and Sycara, K.: Expressing WSMO Mediators in OWL-S, *ISWC2004* (2004.11).

[6] Martin, D. et al.: OWL-s: Semantic Markup for Web Services, http://www.daml.org/services/owl-s/1.1/overview/ (2004.11).

[7] Noy, N. F. and Musen, M. A.: Anchor-PROMPT: Using Non-Local Context for Semantic Matching, *In Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)* (2001).

[8] Salton, G. and McGill, M. J.: Introduction to Modern Information

Retrieval, *McGraw Hill, New York, USA* (1983).
[9] 中辻真, 三好優, 木村辰幸: 柔軟なシステム連携を実現するためのインタフェースマッピングの提案と検証, 人工知能学会 SIG-SWO 研究会 (2004.11).
[10] 中辻真, 八巻洋一郎, 木村辰幸, 小池和郎: OpS-NE 間インタフェースへのオントロジマッピング適用に関する検討, 信学技報, Vol. 103, No. 572, pp. 25–30 (2004.1).
[11] 城田真琴: ユビキタスネットワーク時代のインテグレーション動向, NRI 技術創発 (2004).
[12] 人工知能学会: 特集:テキストマイニング, 人工知能学会誌, Vol. 16, No. 2, pp. 191–238 (2001).
[13] 神崎正英: セマンティック・ウェブのための RDF/OWL 入門, 森北出版株式会社 (2005).
[14] 安田正義: サービス指向アーキテクチャの未来を考察する, http://www.atmarkit.co.jp/fxml/tanpatsu/33soa/soa01.html (2004).
[15] 萩本順三: 【改訂版】初歩の UML, atmarkIT, http://www.atmarkit-.co.jp/fjava/devs/renew_uml01/renew_uml01.html (2003.2).
[16] 小野沢博文: いまなぜ CORBA なの?, atmarkIT, http://www.atmarkit-.co.jp/fjava/rensai/corba01/corba01.html (2001.3).