人工知能学会研究会資料
SIG-SWO-A404-P3

意味情報を活用した自動システム連携手法の提案と連携支援ツールの実装

中辻 真[1]　　三好 優[1]　　木村 辰幸[1]

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[1]

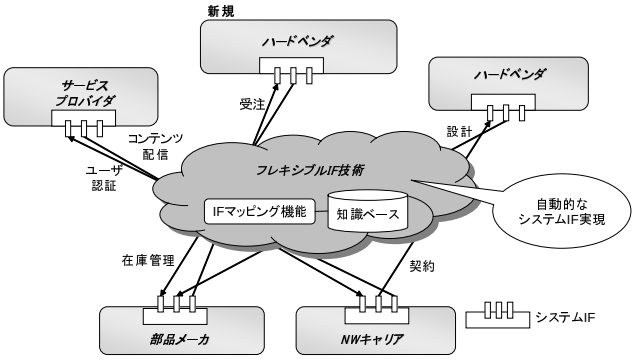

図 1　フレキシブル IF 技術イメージ

1　はじめに

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 [2]．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，提案手法に基づき連携自動化支援ツールを実装し，実運用 NW 管理システムへ適用・検証する．

2　既存システム連携技術と関連研究

既存システム連携技術である従来型技術 (図 2) や CORBA ，WS はそれぞれ適用領域が異なるため，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

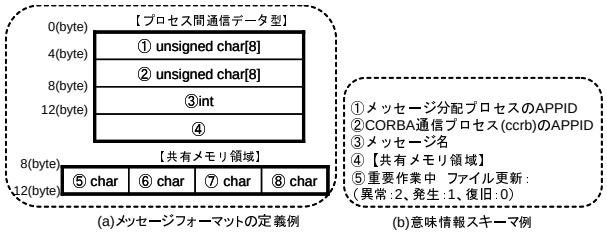

図 2　メッセージフォーマットにおける意味情報スキーマ欠如

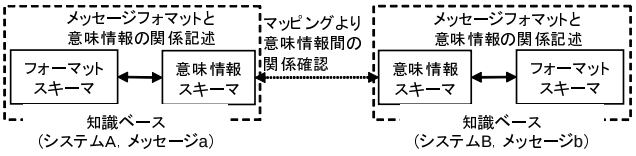

図 3　知識ベースの提案

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 Semantic Web Services(SWS)[1] では，セマンティック Web 技術を 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[3] を拡張した OWL-S[1] 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．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．

3　IF モデリング手法

3.1　知識ベースの提案

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．

表 1　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 owl:allValuesFrom	オブジェクトプロパティ rdfs:range	rdfs:domain	owl:sameAs
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

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・分析を行い，IF に必要な表現を記述ルールとして抽出した．

【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

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 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 1 に示す [3].

【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．

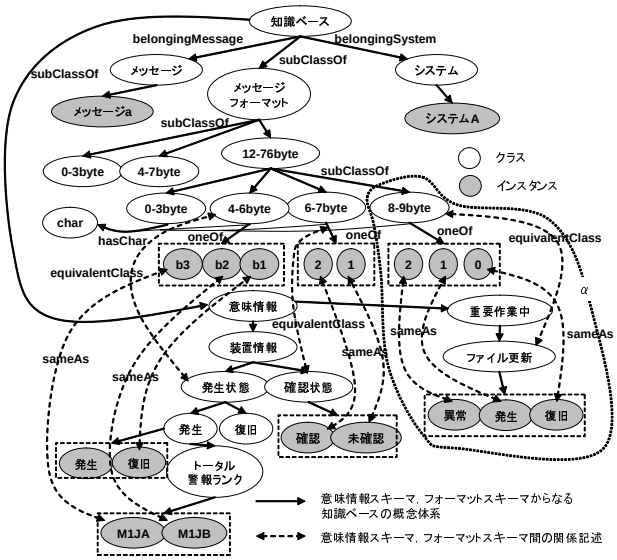

図 4　試作した NMS の知識ベースの一部

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[1] 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メッセージフォーマットが混在して記載され (図 2)，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 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 [1]が，uml backend plugin [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 4 に示す．

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．検討したマッピング方式 [2] のうち以下，方式 2-1，方式 2-3 について説明する．

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 [2]．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，自然言語処理技術を活用して計算し [2]，$S(N)_{ij}$ とする．

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 [2]．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．

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 $S(C_{ij}) = \kappa * T(S(N)_{ij}) + \lambda * T(S(U)_{ij}) + \mu * T(S(O)_{ij})$(式 (2)) により与える．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．

[1]http://protege.stanford.edu/
[2]http://protege.stanford.edu/plugins/uml/

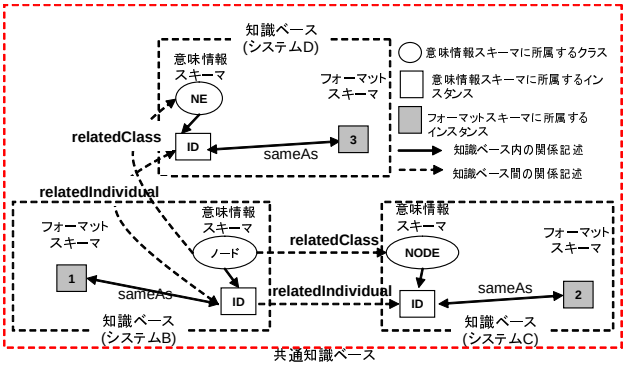

図 5　共通知識ベース具体例

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．

共通知識ベースの具体例を図 5 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，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．

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．

表 2　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 3　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

4.1　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．

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 2 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 3 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．

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 2)．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 3)．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 4 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 5 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．

表 4　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 5　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

5　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．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，デモを通じ実装ツールの有用性を広める．

参考文献

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

[2] 中辻真，三好優，木村辰幸: 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, *DEWS2005* (2005.3).

[3] 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).