

米国におけるセマンティック Web サービスの現状と動向

Semantic Web Services in the USA

益岡 竜介
Ryusuke Masuoka

米国富士通研究所
Fujitsu Laboratories of America, Inc.
ryusuke.masuoka@us.fujitsu.com, <http://www.flacp.fujitsulabs.com/~rmasuoka/>

keywords: semantic Web services, ontology

1. はじめに

セマンティック Web サービスは「サービス」に豊かな記述を与えることにより、高度な、あるいは自動的なサービス発見、検索、合成、実行、監視などの機能を実現しようとするものである。

一般的な（「サービス」なしの）セマンティック Web の視点からみると、「セマンティック Web サービス」は「サービス」を記述するための枠組みやオントロジーを定義することと言える。

セマンティック Web サービスには現在 OWL-S, SWSO/FLOWS, WSMO, WSDL-S の 4 つの大きな流れがある。OWL-S [OWL-S], SWSO/FLOWS [SWSd] は DAML プロジェクト [DAM] のセマンティック Web サービスの活動の流れから出てきた。WSMO [WSM] [福田 05] は欧州を中心としたプロジェクトで作られてきた。WSDL-S はジョージア大学の METEOR-S [MET] プロジェクトから出てきた。いずれも W3C, OASIS などの何らかの標準化団体での標準化を目指している。

「サービス」や「サービスの意味（セマンティクス）」といっても、人それぞれとらえ方が異なり、最近の Web サービスの急速な普及によってもそのとらえ方が揺れ動いている。また各セマンティック Web サービス技術も変わり続けており、互いに影響しあっている。その状況の中でこれらのセマンティック Web サービスの技術をきれいに分類することはできないが、少なくとも以下に比較のためのいくつかの軸を提示し、各技術を大まかに位置づけてみる。

- 対象とする「サービス」
- サービスの実装とサービスセマンティクスの関係づけの枠組み
- サービスを記述するためのモデル

まず対象とするサービスであるが、明確に Web サービスであるとしているのが、WSMO と WSDL-S である。OWL-S と SWMO/FLOWS もメインは Web サービス

であるが、それ以外への応用も想定されている。

実装とセマンティクスの関係づけの枠組みについて、セマンティクスから実装への方向をとるのが、OWL-S, SWSO/FLOWS, WSMO である。これらではまずサービスのモデルを設定してその記述のためのオントロジーを定義する。そのオントロジーに基づきサービスの記述を与え、その一部として Web サービスなどのサービス実装との対応を定義するアプローチを取る。一方で実装からセマンティクスへの方向をとるのが WSDL-S である。WSDL-S では WSDL の拡張として WSDL 自体の中に WSDL の要素からセマンティックなモデルへのリンクという形でセマンティックな注釈を加えるアプローチを取る。

サービスを記述するためのモデルは、そのモデルに含まれる概念とその概念間の関係性から構成される。

OWL-S と WSDL-S の全て、および WSMO のうち予約語以外ではモデル内の個々の概念をさし示すのに URI が使われる。それらの場合 URI が各概念のユニーク ID となる。SWSO/FLOWS もその Web 版 (Web-variant) では URI を概念のユニーク ID として使う予定である。

複数サービス間のやり取りがあっても、それを全体としては 1 つのサービスとしての内部的な見方をとるのが OWL-S と SWMO/FLOWS である。OWL-S はサービスをプロセスとしてモデル化し、SWSO/FLOWS ではプロセスをさらに一階記述言語による公理系による「仕様」としてモデル化する。

WSMO は OWL-S や SWSO/FLOWS と同様なサービスの内部的な見方とともに、サービスをメッセージのやり取りをする主体として外から見た外部的な見方も持つ。また Web サービス連携のためのメタデータなどまで含めたオントロジー記述を可能にしている。

WSDL-S はサービス記述のための特定のモデルは規定しないという立場をとる。OWL, WSMO, UML など任意のオントロジー言語あるいはモデリング言語で記述されたモデルが独立に外部に与えられ、WSDL の要素をそ

のモデルの要素にリンクする。

本稿では、主に米国でのセマンティック Web サービスの動向として OWL-S, SWSMO/ FLOWS, WSDL-S の各技術について、それぞれ 2 章 から 4 章 で紹介する。WSMO は欧州での動向として本特集の [福田 05] で取り扱われる。5 章 ではセマンティック Web サービスのユビキタス・コンピューティングへの応用としての Task Computing とその中でセマンティック Web サービスの一つである OWL-S といかに関わってきたかを紹介し、最後に 6 章 でまとめる。

2. OWL-S

OWL Web Ontology Language for Services (OWL-S) [OWL-S] [Martin] は名前の通り、サービスの OWL *1 [OWL a] によるオントロジーである。OWL-S はセマンティック Web サービス技術の中でももっとも成熟度が高いものと言えるだろう。OWL-S と他の Web サービスやセマンティック Web サービスの関係は [Ankolekar] にまとめられている。

OWL-S は当初 DAML (DARPA Agent Markup Language) プロジェクト [DAM] [益岡 02] の中で、2001 年始め頃からセマンティック Web サービスに興味を持った研究者たちによって DAML プロジェクトの一部として始められた。当初は DAML オントロジー言語に基づいており DAML-S と呼ばれていた。2001 年 5 月に最初の DAML-S バージョン 0.5 がまとめられ、2001 年 12 月にはバージョン 0.6、2002 年 10 月にバージョン 0.7、2003 年 5 月にバージョン 0.9 と進化していった。サービス記述に使っていたオントロジー言語 DAML が OIL [OIL] と一緒になり W3C で OWL として標準化されるのに伴い、基礎となるオントロジー言語を DAML から OWL とし、2003 年 10 月のバージョン 1.0 より OWL-S と呼ばれるようになった。2004 年 11 月には OWL-S バージョン 1.1 が Member Submission [OWLd] として W3C に提出され、承認された。これにより今後 W3C がセマンティック Web サービスを標準化する際に OWL-S を使うことができるようになった。もうすぐ 2005 年秋には OWL-S バージョン 1.2 が出る予定である*2。OWL-S の標準化は、DAML プロジェクトに直接関わっていない研究者も多く参加して進められてきたが、バージョン 1.2 を出した後は、より多くの参加者に開かれた“オープン・ソース”形式で発展させていく予定である。

OWL-S は各バージョンを通じて内部の構造を変えていったが、総じて当初のいろいろなアイデアの寄せ集め

*1 正確には OWL の記述論理 (Description Logic) に基づく部分言語である OWL-DL による。

*2 バージョン 1.2 では XSLT スクリプトによるグラウンディング、および事前条件や影響などにて使われる表現式 (expressions) についての改善や拡張、例外指定についてのドラフトプロポーザルなどが含まれる予定である。

でまとまりの欠けたものから、より整合性があり、また分かりやすい形のものへと徐々になっていた。

OWL-S で目指しているのは、自動サービス発見、実行、合成などである。コンピュータによって解釈可能なセマンティックなサービス記述を与えることにより、プログラムやエージェントがなるべく人を煩わせずにサービスの発見、実行、合成などを可能にしようということである。

OWL-S では、一般にサービスにはアトミック (atomic) とコンポジット (composite) の二種類があると考え、OWL-S はその両方の種類のサービスをサポートすることを意図している。アトミック・サービスとは一つのメッセージを受け、一つのメッセージを返すワン・ステップのサービスである*3。コンポジット・サービスは、(そのそれぞれがまたコンポジット・サービスであるかもしれない) 複数のサービスを合成したものであり、一回だけではなく複数回にわたるメッセージのやり取りを持ちうる。

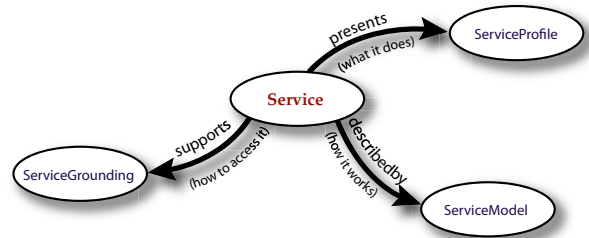


図 1 OWL-S の上位オントロジー。サービスをそのプロフィール (profile)、モデル (model)、グラウンディング (grounding) の三要素で表現する。 ([Martin] の Figure 1 より)

図 1 のように、OWL-S のサービス記述はプロフィール (profile)、モデル (model)*4、グラウンディング (grounding) の三要素からなる。

OWL-S のプロフィールはサービスが何をするか (what it does) の記述を、エージェントやプログラムがそのサービスに必要なものを適切に判断できるように与える。自動サービス発見やセマンティックなサービス合成を可能にするのがプロフィールの主な目的である。

プロフィールは具体的には、サービスの名前、説明、サービスに関する連絡先、サービスの IOPEs (入力 (input), 出力 (output), 事前条件 (precondition), 影響 (effect) *5) などの機能記述、サービスパラメータやサービスカテ

*3 ここで「一つのメッセージ」といっているが、それは複数の入力 (あるいは出力) を束ねたものであってよいし、全く入力 (あるいは出力) を含まないメッセージでもよい。

*4 現在のところ OWL-S のサービスモデルとしてはプロセスしかなくとも関わらず、プロセスでなくモデルと呼んでいるのは将来的に他のモデル (例えば関係データベースを内包するサービスのための RelationalDataServiceModel といったもの) を導入する可能性も想定しているからである。

*5 正確に言うと、影響は出力と関係するものとして結果 (result) の一部として表現されている。これはサービスの出力と影響が互いに関係していることをモデル化できるようにするためである。例えば、購入のためのサービスで、その実行が成功した場合には影響として商品の所有権が移ったということ、出力とし

ゴリなどのプロファイル属性 (profile attribute) などの記述を持つ。サービスパラメタは、その値として任意の OWL オントロジーで定義されたオブジェクト値をとるものを定義できる。

上に述べたようにプロファイルはサービス発見に使われ、例えばサービスのレジストリに OWL-S のプロファイルの部分だけ登録して、セマンティクスに基づいたサービスのより高度な検索に使われたりする ([Paolucci 02], [川村 05] 参照)。プログラムやエージェントは、そのレジストリを通じて、プロファイルの情報により必要なサービスを探すことができる。

現在 OWL-S ではサービスのモデル (model) をプロセス (process) として表す。OWL-S のプロセスは AI でのプランニング言語や PSL (Process Specification Language) [Schlenoff 00] [PSL] などのプロセスモデリングやワークフローの標準、その他関連した各種の研究成果に基づき、サービスがいかにか動くか (how it works) を記述する。コンピュータが解釈できる形でのサービスの挙動の記述を与えることにより、サービスを使う側がどのようにして、あるいはいつサービスとやり取りすればよいの分かる。OWL-S のプロセス記述はサービスの実行、プランニング、合成、監視などに使われる。プロセスにはアトミック (atomic)、単純 (simple)、コンポジット (composite) の三種類がある。アトミック・プロセスとコンポジットプロセスはこの章の最初に述べたアトミック・サービスとコンポジット・サービスの種類にそれぞれ対応するものである。単純プロセスはグラウンディングと関連付けられておらず、それ自体では実行可能ではない。その役割はアトミック・プロセスに特定の使い方のビューを与えたり、コンポジット・プロセスに簡単化された表現を与えるためのものである。

プロセスは具体的には、サービスの IOPEs (入力, 出力, 事前条件, 影響) *6, コントロール・フロー, およびデータフローの記述を持つ。

IOPEs のうち入力と出力は OWL クラスあるいはデータ・タイプの指定とそのタイプの値を持つパラメタとして定義される。事前条件と影響は表現式 (expression) で与えられる。その表現式を与える言語としては SWRL [Horrocks 03], KIF [KIF 98], PDDL [Ghallab 98] *7 などが挙げられている。特定の論理言語を決めていない点は標準として弱いところである。これは事前条件や影響などのサービスの要素を記述するのに必須なルールに関して、現時点で Web 上でルールを記述する言語の標準

が決まっていないことの反映である。今後 W3C でルール記述言語が標準化されれば、それが採用される可能性が高い。

コンポジット・プロセスに対しては、コントロール・フローを記述することができる。Sequence, If-Then-Else, Split, Iterate などの制御コンストラクト (control construct) で定義される。ただし注意しないといけないのは、これらの制御コンストラクトは通常のプログラミング言語のもののような印象を与えるが、サービスの挙動を指定しているわけではない点である。これらはコンポジット・プロセスを使う側から見た抽象的に統合されたビューであり、クライアントがコントロール・フローに従ったメッセージをやり取りすることにより始めて、コンポジット・プロセスの内容が実行される。

データ・フローはプロセスの各コンポーネントのどの入力が、その前のどの段階のどの出力からとられるのかを指定する。特別な場合以外、入力側でその入力がパラメタとしてどの出力とバインドするかを指定する形 (consumer-pull convention) で表現する。

OWL-S のグラウンディング (grounding) は、サービスにいかにかアクセスするか (how to access it) の記述を、プロトコル, メッセージ・フォーマット, シリアライゼーション (serialization), トランスポート (transport) などについて与える。プロファイルやプロセスなどの抽象的なレベルのサービス記述を、サービス実装と実際にやり取りできるレベルの具体的なサービス記述 (主に Web サービスの WSDL) に関係づけるのである。この関係づけは一番小さい単位であるアトミック・プロセスの入出力を通じて行なわれる。

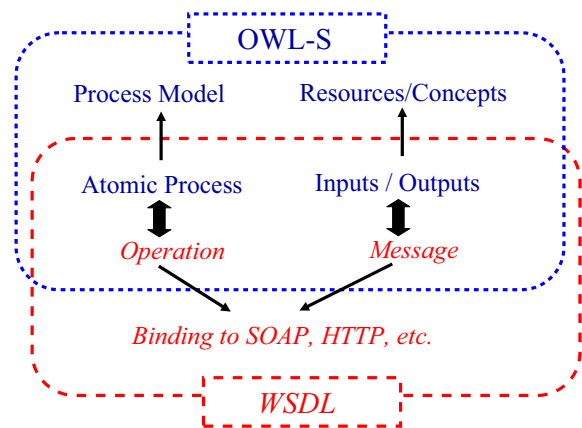


図 2 OWL-S と WSDL 間のマッピング ([Martin] の Figure 4 より)

OWL-S と WSDL 間のマッピングは図 2 のように、WSDL のオペレーションが OWL-S のアトミック・プロセスに、WSDL のメッセージ (message) が OWL-S のアトミック・プロセスの入出力に対応付けられる。この WSDL のメッセージと OWL-S アトミック・プロセ

ては確認番号になり、失敗した場合には影響はなく、出力はエラーメッセージとなるといったことを表現できる。

*6 IOPEs はプロファイルでも定義されていた。これはプロファイルとプロセスで全く異なる IOPEs を持つこともできるからである。ただし通常はプロファイルの IOPEs がプロセスの IOPEs の一部になっており、その場合、前者は後者の対応する要素へのリンクとして与えられる。

*7 OWL-S バージョン 1.2 ではさらに SPARQL [SPA], RDQL [RDQ], SWRL-FOL [SWR] が加えられる予定である。

スの入出力のマッピングは XSLT スクリプト [XSL] で指定することができる。

OWL-S のツールは, [OWLc] にまとめられている。OWL-S のエディターや, マッチメーカー, サービス・コンポーザなどのツールがリストされている。

OWL-S のよい点はやはりその成熟度であろう。長い時間をかけて来ており, またその間いろいろなアプリケーションで使われてきたことにより鍛えられ, かなり標準としてこなれている。

OWL-S の課題・問題としては, XML, RDF, RDFS, OWL といった多くのレイヤーの上の実現されているためにどうしても複雑になってしまっていることであろう。特にリストに関しては, Lisp のような再帰表現になり, また RDF と OWL-DL の整合性の問題もあり, 扱いが難しくなっている。

3. SWSO/FLOWS

Semantic Web Services Ontology (SWSO) [SWSd] は OWL-S と関係が深く, またその標準化に携わっている関係者も一部重なっているが, それらの間には特に公式な関係はない。2002 年 12 月に Innsbruck, Austria で第一回 SWSI (Semantic Web Services Initiative) [SWSb] が開かれた。SWSI に二つの部会, SWSA (SWS Architecture Committee) [SWSa] と SWSL (SWS Language Committee) [SWSl] が作られ, SWSL における活動の結果の一つとして SWSO が策定された。

SWSO は ISO18629 として標準化されている PSL (Process Specification Language) [PSL] [Schlenoff 00] をその基盤として*8, ワークフローを「仕様」として扱う*9。SWSO はその表現として一階述語論理に基づく公理化, FLOWS (First-order Logic Ontology for Web Services) を持つ*10。2005 年 5 月に SWSO バージョン 1.0 がまとまった。今後は FLOWS の Web 版 (Web-variant) や実行セマンティクスの策定, メッセージ・オントロジーの改良などを行なっていく予定である。

以下では “FLOWS” で SWSO のオントロジー内容とその表現の両方をさす。

FLOWS は, OWL-S と同様に一階述語論理や状況計算 (situation calculus) に基づきながらも, OWL-S を次のレベルに持っていく試みである。FLOWS も OWL-S のプロファイル, プロセス, グラウンディングに相当する, サービス記述子 (service descriptor), プロセスモデ

ル (process model), グラウンディング (grounding) を持っており, 他のセマンティック Web サービスと同様に, サービス発見, 実行, 合成, 監視などの自動化が FLOWS の目標に含まれる。しかし FLOWS の特徴はやはりサービス・セマンティクスに基づき, Web サービス同士あるいは Web サービスと “現実世界” がいかに相互に作用しあうかについての推論を可能とすることに重きをおく点であろう。

サービスを表現するのに Petri Net や FSM (Finite State Machine), Pi-Calculus などですでに使われてきたが, これらでは,

- 条件つき影響 (conditional effects)
- 非機能的制約 (non-functional constraints)
- ドメイン内のオブジェクト間の関係

といった側面をとらえきれない。そこですでにある Petri Net, FSM, Pi-Calculus などによるモデルを形式的に表現できることに加えて, さらに上にあげた点も表せるように, FLOWS では (準決定可能 (semi-decidable) でしかないこと, 至難 (intractable) であること, 人間にとって読みにくいことといった面で完全はでないが,) 一階述語論理が採用され, FLOWS の「仕様」の記述に用いられた (図 3 参照)。

OWL-S が使う OWL では, 本質的に動的なものであるプロセスの意味 (セマンティクス) をうまくとらえられなかった。OWL-S は OWL-DL (DL は記述論理 (Description Logic) の意味) の範囲内で記述されているが, OWL-DL を構文的に (syntactically) に使っているだけで, プロセスモデルに関して記述論理を使っただけの推論, 証明などはできない。

FLOWS では, OWL-S よりも豊かなセマンティクスを, プロセスに対する制約や公理の形で与える。これによりプロセスに対して, 「ある特定のアクションがいつも行われる」とか「ある条件がいつも成立する」といったことを推論することが可能になる。

さらに FLOWS により, 一般的な自動サービス発見, 実行, 合成, 監視に加えて, 例えば次のようなことも可能になる。

- 生成されたプランが FLOWS による仕様に合致しているかどうかを定理証明系 (theorem prover) を使って証明できる。
- FLOWS はサービスに関する一般的な形式化を与えるので, 例えば BPEL4WS [BPE] や WS-CDL [WSC] などの仕様を FLOWS にマップし, それらの間の同値性などの関係を見出すことができる。

FLOWS のよい点は, 理論的に厳密に構成されており, またサービスを表現するフレームワークとして非常に強力である点である。サービスに関するいろいろな記述, モデルの共通の基盤になりうる。

一方で FLOWS の課題・問題としては, 強力にするためにとった選択肢自体が, その適用範囲を狭めてしまった

*8 米国立標準技術研究所 (NIST) で PSL の標準化に中心的な役割を果たした Michael Gruinger がこの SWSO でも中心メンバーの一人として関わっている。

*9 例えば while や if-then-else も手続き要素ではなく制約として, 変数代入も認識論理の用語 (epistemic terms) によって指定される。

*10 ROWS (Rules Ontology for Web Services) あるいは SWSO-Rules と呼ばれる, 同じ公理の (一部の公理は部分的だが) ルールに基づく表現も提案され, SWSO に含まれている。

```

...
Service

Every service is associated with an activity.

forall ?s
  (service(?s) ==> exists ?a
    (service_activity(?s,?a))).

A service activity is an activity in the PSL
Ontology.

forall ?s,?a
  (service_activity(?s,?a) ==>
    activity(?a)).

A service occurrence is an occurrence of
the activity that is associated with the
service.

forall ?s,?o
  ((service_occurrence(?s,?o) and
    occurrence_of(?o,?a)) ==>
    service_activity(?s,?a)).
...
Split

An Split activity is equivalent to a
strong_poset activity in PSL. The activity
trees all have the same structure and each
activity tree consists of branches that are in
one-to-one correspondence with sequences of
subactivity occurrences that satisfy the
partial ordering constraints.

forall ?a
  (Split(?a) <==>
    (uniform(?a) and exists ?occ
      (occurrence_of(?occ,?a) and
        neg simple(?occ) and
        ordered(?occ) and
        strong_poset(?occ)))).
...

```

図 3 FLOWS 例: FLOWS ではサービスを一階述語論理を使ってこのように公理化する。Service と Split に関する公理の部分を示している。([FLO] より抜粋)

であろうということである。セマンティック Web サービスの一般的な目的とする自動サービス発見、実行、合成、監視などに使うには非実用的になってしまった。そのような一般的な目的を実現するには FLOWS が広く多くプログラマに使われる必要がある。一般のプログラマに対してはユーザ・インタフェースや表現力を抑えたハイレベル記述言語を通じて使ってもらうことを考えているようだが、それでも一般でのサービスのとらえ方の違いを埋めるのは難しいであろう。もう一つには一階述語論理の採用により、その記述レベルが OWL、特に OWL-DL の範疇を越えてしまったことである。特定の目的に応じてそれに対応した処理系を用意する必要がある。

中途半端に一般的な自動サービス発見、実行、合成、監視を目的にあげるよりは、むしろサービスに関する仕様を書く人たちに向けて、整合性のある正しい仕様を書くことをサポートする枠組み (いわばサービス仕様の BNF (Backus Naur Form) のようなもの) としてそれを前面にもっと強く押し出すべきであろう。

4. WSDL-S

WSDL-S [WSDb] は最初に ICWS 2003 [ICWS2003] で [Sivashanmugam 03] として発表された。この WSDL-S の着想は 2002 年秋頃にジョージア大学の METEOR-S プロジェクト [MET] のメンバが、OWL-S は扱いにくく、WSDL との間やそれ自体の中での冗長性が大きいと思った時点で遡る。そこから WSDL-S は次の原則に基づいて Web サービスのセマンティックな仕様として作られていった。

- (1) 既存の Web サービス標準の上で作る。
- (2) (OWL, UML など) セマンティック記述言語から独立であり、また一つの Web サービスに対して複数の異なる言語による注釈を同時に許す仕組みである。
- (3) XML Schema によるデータタイプを持つ Web サービスのセマンティックな注釈をサポートする。
- (4) Web サービスの入出力のスキーマタイプとオントロジーの間の多様な変換を仕組みをサポートする。

WSDL-S はその後さらに改良を加えられ、[WSD 04] が WSDL 2.0 コミッティーの一部のメンバーに 2004 年 6 月に渡された。その後 2004 年の秋に IBM の研究者たちがジョージア大学の METEOR-S グループに加わり活動を続け、WSDL-S バージョン 1.0 [Akkiraju 05] が 2005 年 4 月に公開された。

最近では WSMO と METEOR-S の研究者たちによる、WSDL-S を WSMO のグラウンディング (grounding) とする共同研究 [Verma 05] がある。今後企業の参加、スポンサーシップを受けて、W3C あるいは OASIS などでの標準化を目指している。

WSDL-S は OWL-S のプロファイルとグラウンディングに相当する部分を対象とする。OWL-S プロセスに

相当する部分は、Web サービスでいうと WSDL ではなく BPEL4WS [BPE] や WS-CDL [WSC] に対応する部分であり、WSDL を基にする WSDL-S の中では扱われない*11。

他の OWL-S, WSMO, SWSO/FLOWS がセマンティックな記述をサービスの実装とは独立に与えるのと異なり、WSDL-S では WSDL [WSDa] の拡張性要素 (extensibility elements) を使って WSDL ファイル自体の中に直接セマンティックな注釈 (annotation) を与える。

またセマンティクスを記述するために UML や OWL などの特定のモデリング言語やオントロジー言語に依存しないアプローチをとっている。したがってすでに UML や OWL などいずれかの言語で記述されたドメイン知識があれば、それを簡単に再利用することができる。また必要に応じて一つのサービスに複数の異なる言語による注釈を与えることも可能である。

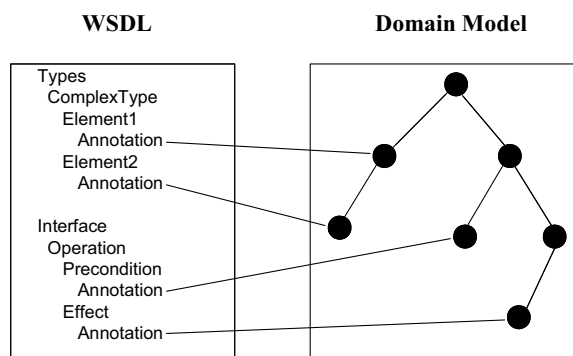


図 4 WSDL-S では WSDL の element へのセマンティックな注釈を、外部にあるドメインモデル (知識) へのリンクとして与える。 ([Akkiraju 05] Figure 1)

WSDL-S の具体的な仕組みは図 4 にあるように、WSDL のインターフェース (interface), 操作 (operation), およびメッセージ・コンストラクト (message construct) に対してドメイン知識へのリンク (URI 参照) をそのセマンティックな注釈として WSDL の拡張性要素を通じて与えるものである。注釈のために使われている WSDL の拡張性要素は以下のようなものである。

- WSDL のメッセージ・タイプ (message types) (XSD の simpleType や complexType) への注釈
 - modelReference: セマンティックな関連を与えるための拡張属性 (extension attribute)
 - schemaMapping: スキーマとデータ間のマッピングを与えるための拡張属性
- WSDL オペレーションへの注釈
 - precondition および effect: それぞれサービスの事前条件 (precondition) および影響 (effect) を与えるための拡張要素 (extension elements)

*11 プロセスモデルに対応するものとして、現在彼らは BPEL にセマンティックな注釈を与えることを検討している。その最初の成果は [Sivashanmugam 04] に報告されている。

- category: サービスのカテゴリを与えるためのインターフェース要素に対する拡張属性
- action: 操作要素 (operation element) に対する拡張要素*12

WSDL にこれらの要素を織り込むことにより、WSDL の要素に対するセマンティクスとして外部のドメイン知識へのリンクを WSDL 自体の中に埋め込む。上で述べたように、これらのリンク先のドメイン知識は OWL, UML などそのオントロジー言語を特定せず、また複数の異なるオントロジー言語によるドメイン知識へのリンクを与えることも可能である。

図 5 の例で見ると、サービスのカテゴリ (category) が “http://www.naics.com” において定義されている “naics:443112” のコードがついた “Electronics” という名前の項目に分類され、サービスの機能的セマンティクスが (action によって) RosettaNet の “RequestQuote” に対応し、その出力 (output) は POontology (発注 (Purchase Order) オントロジー) 中の “OrderConfirmation” という概念に対応付けられている。またこのサービスの事前条件 (precondition) は POontology の “AccountExists” という概念に対応し、影響 (effect) は POontology の “ItemReserved” という概念に対応することが記述されている。これらは記述は全てあくまでも単なるリンクによる参照であり、それぞれのセマンティクスはそのリンク先を含む外部のドメイン知識の中で定義されている。例えば事前条件の “AccountExists” は、POontology の中で「すでにアカウントが存在している状態」などとして WSDL の外で定義されている。

図 5 の例にはないが、complexType の入出力には schemaMapping を使って、OWL-S のグラウンディングと同様に XSLT [XSL] スクリプトや XQuery [XQu] などでセマンティックモデルとの対応を与えることができる。

WSDL-S をサポートする、あるいは採用するツールが [MET] のサイトからダウンロードできる。その中の Radiant は WSDL に WSDL-S によるセマンティックな注釈を与えるツールであり、Eclipse の Plug-in として実現されている。IBM alphaWorks の [IBM] から IBM の研究者による WSDL-S 関連のツールをダウンロードすることができる。

WSDL-S のよい点は、その非常に実際的なアプローチである点である。無理に WSDL の中でセマンティクスを与えることを避け、WSDL の拡張性要素を使って、外部のドメイン知識へのリンクだけを WSDL の中に埋め込む。WSDL の外部のそのリンク先にセマンティクスが記述されており、その記述はセマンティック記述言語を問わないといった点である。このためサービスのセマンティクスを記述する側にとっては、任意のセマンティック

*12 この action はまだ検討中のもので WSDL-S [Akkiraju 05] の仕様には入っていない。

```

...
<xs:element name="processPurchaseOrderResponse"
  type="xs:string"
  wssem:modelReference=
    "POOntology#OrderConfirmation"/>
</xs:schema>
</types>
<interface name="PurchaseOrder">
<wssem:category name="Electronics"
  taxonomyURI=http://www.naics.com/
  taxonomyCode="443112" />
<operation name="processPurchaseOrder"
  pattern=wsdl:in-out
  action="rosetta:#RequestQuote" >
<input messageLabel =
  "processPurchaseOrderRequest"
  element="tns:processPurchaseOrderRequest"/>
<output messageLabel =
  "processPurchaseOrderResponse"
  element="processPurchaseOrderResponse"/>
<!-- Precondition and effect are added as
  extensible elements on an operation -->
<wssem:precondition name="ExistingAcctPrecond"
  wssem:modelReference=
    "POOntology#AccountExists">
<wssem:effect name="ItemReservedEffect"
  wssem:modelReference=
    "POOntology#ItemReserved"/>
</operation>
</interface>
...

```

図 5 WSDL-S 例 (PurchaseOrder.wsdl): wssem の namespace で始まる要素が WSDL-S によるセマンティックな注釈である。 ([?] より)

記述言語を使って、いかにでもサービスを記述することができ非常に便利である。

しかし上であげた点が一方では WSDL-S の課題・問題となる。サービスのセマンティクスやモデルをどのように与えるかについては WSDL-S では全く規定しないため、その記述を処理する側にとってはそのセマンティクスがどのように与えられるかを予測できず、セマンティクスに基づく各種処理の実現が難しくなる。WSDL-S だけでは、本来セマンティック Web サービスが目的とするサービスに関するいろいろな処理の自動化や高度化に関しての (少なくとも直接的) 答えを与えられない。

5. Task Computing と OWL-S

Task Computing (TC) ([TCH], [Masuoka 03b], [Masuoka 03a], [湯原 04], [Song 04]) は、非常にダイナミックなユビキタス環境の中で一般の人がその場にあるデバイス、個人の端末上の各種アプリケーション、インターネット上の Web ページや Web サービスなどの機能を

自在に組み合わせて、各人がやりたいこととしての「タスク」を簡単に実行できるようにする枠組みである。

プロジェクトは 2002 年 6 月から、米国富士通研究所 (Fujitsu Laboratories of America - FLA) とメリーランド大学の Jim Hendler 教授の MINDSwap のグループ [MIN] との共同研究として始まり、2004 年からは日本の富士通研究所、富士通とともに進めている。

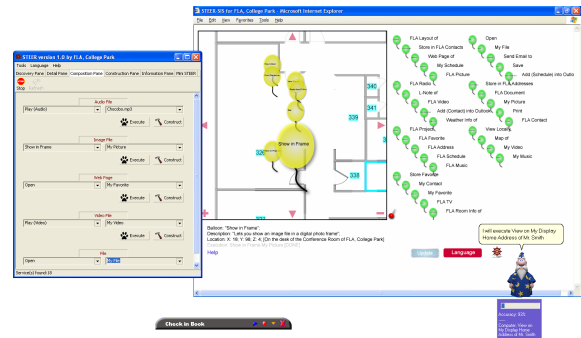


図 6 Task Computing クライアント: 左から時計回りに、テキストベース、グラフィカル、音声による Task Computing クライアントである。一番下のもは、タスクを OWL-S のコンポジット・サービスとして保存したファイルを直接実行するクライアントである。

Task Computing (TC) の仕組みを以下に簡単に説明する。TC クライアントはまずマシン・ローカルなサービス発見メカニズム, UPnP [UPn], あるいは Web サービスを使ったりリモートサービス発見メカニズムを使ってユーザの環境にあるいろいろな機能を「サービス」として発見する。次に発見したそれぞれのサービスの OWL-S によるセマンティックなサービス記述を入手する。TC クライアントはそのセマンティックな記述に基づき、その環境で可能な (サービス合成としての) タスクをユーザに提示したり、ユーザがサービスを簡単に組み合わせ、タスクを作り出せるようにする。ユーザはそれらのタスクを即その場で実行できる。

デバイス、自分のコンピュータ端末、インターネット上の Web サービスなど一つのタスクの基になっているサービスはどこからのものでも構わない。プロジェクトでは実際にデバイス, OS, アプリケーション, 実際の Web サービス (Amazon, Google, Yahoo!, グリッド・サービス, 他) などから 100 を越える種類のセマンティック Web サービスを実現した。そしてユビキタス環境でのいろいろなモダリティでそれらのサービスをその場で自在に使い、タスクを実行するために、テキストベース, グラフィカル, 音声, ジェスチャーなどによる TC クライアント (図 6 参照) を実現した。

上で述べたことを可能にするためには、システムが今いる動的な環境の中にどのような機能があるかを判断し、それをユーザのタスクと結びつける必要がある。そのためにはデバイス, OS, アプリケーション, Web ページ, Web

サービスなどのいろいろな機能がどのようなものであるかを統一的に記述し、扱える枠組みが必要であった。それらの機能を全て^{*13}をサービスとして抽象化し、TC ではそのセマンティクスを表すものとして、OWL-S (当初は DAML-S 0.6 であった) を採用した。Task Computing はそのプロジェクトのスタート時点から、OWL-S と一緒に育ってきたとも言える。

Task Computing (TC) をセマンティック Web サービスのアプリケーションとして見ると以下のような特徴がある。

ユーザのトータルな体験としての TC を実現するため、TC ではサービス発見、合成、実行などをそれぞれ別々のこととしてではなく、発見から、合成、実行、保存、さらにはユーザ自身やプログラムによる動的な生成までのサービスの全ライフ・サイクルを有機的なまとまりを持つものとして実現する。それをユビキタス環境の非常に動的な環境^{*14}で、セマンティック Web サービスとして抽象化されたサービスたちを対象にその場 (on-the-fly) ににおいて提供する。

またセマンティック Web サービスの目的としてよく自動的なサービス発見、合成、実行があげられるが、TC ではユーザを中心に据え、セマンティクスによる自動化よりは、セマンティクスによるユーザのサポートを目指した。一つには自動化が必ずしもユーザの望むことではないであろうとの考えがあった。勝手に何かが実行されてしまうよりも、最終的にどうするか判断はユーザが下したいと考えた。もしそれを自動化しようとする、非常に詳細なユーザのコンテキストやプリファレンス、サービスセマンティクスなどの提供にかなりのコストがかかり、それでも完全自動化は難しいであろう。それよりも OS に近い考え方で、セマンティクスを使った適切な「抽象化」により、ユーザが簡単にいろいろなタスクをできるようにサポートする方が有効であろうと判断した。“User in the loop”で、何をどうするかに関してユーザをサポートし、システムが難しい部分は逆にユーザに助けをもらえばよいという発想である。

Task Computing プロジェクトの当初は、MINDSwap のグループが OWL-S を扱うコアのエンジンと標準化を担当、FLA はそれらのユーザに徹し、アプリケーションとしての Task Computing を開発、その中でのエンジンや標準に関する問題・課題を MINDSwap グループにフィードバックし、一緒に解決していった。エンジンは OWL-S API [OWLb] として結実し、MINDSwap グループの Bijan Parsia と Evren Sirin は OWL-S 標準

*13 上記のような機能的なものだけでなく、TC ではファイル、連絡先、スケジュールといったオブジェクトさえも、オブジェクトを提供するサービスとして扱っている。

*14 例えば、サービスが急に現れたり、消えたりし、またサービスをホストするマシンやユーザの端末の IP アドレスも動的にアサインされる。いつでも FQDN (Fully Qualified Domain Name) や固定の IP アドレスでアクセスできるインターネット上にある Web サービスを扱うのとは別の難しさがある。

化で中心的な役割を果たしていった。

TC で OWL-S を使っていき過程ではいろいろな課題を乗り越えていった。他のセマンティック Web サービスのアプリケーションを作る際の参考のために以下にそれらの事柄を記す。

最初に OWL-S (DAML-S 0.6) を見たときに、そもそもなぜ一つのサービスに対するプロファイル、モデル (プロセス)、グラウンディングの三つに分かれているのか疑問に思った。論理的だけではなく、ファイルとしても独立になっているのである ([OWL-S] にある OWL-S の各バージョンの “Examples” を参照)。さらにはプロファイルとプロセスの両方に IOPEs があり、それらは何の関係もなくともよいという。

もちろんそれなりの理由があった。すなわち使われる目的が異なるからである。プロファイルの使われる主な目的は発見やセマンティックな合成であり、主に合成されたコンポジット・プロセスのモデルを表すのにプロセス、サービスの実行にはプロセスやグラウンディングが使われる。プロファイルを分けておけば、プロファイルだけをレジストりに登録してセマンティックなサービス発見をサポートすることが無駄なくできる。またサービス発見やセマンティックなレベルの合成をするためには、プロセスにある (通常プロファイルにあるものより複雑な) IOPEs の全てを、そのまま使うことはないというのが、プロファイルとプロセスの両方に IOPEs がある訳である。

一方で TC では最初から発見、合成、実行などを一つのものとして扱う統合された環境を実現しようと考えていたので、一つのサービスに対しプロファイル、プロセス、グラウンディングの三つ全てが必要であり、また TC の想定する動的な環境で見つけたそれぞれのサービスに関して、三つのファイルを別々に取りに行ったり、管理するのはどうにも非効率であった。この点は比較的簡単に解決し、三つの内容を一つの RDF のグラフとして統合して、それを一つのファイルにすることとした^{*15}。

プロファイルとプロセスの IOPEs に関しては、同じサービスに対して全く無関係の IOPEs があるのは標準としてとても分かりにくいと MINDSwap のメンバに主張してもらい、DAML-S 0.9 から OWL-S 1.0 の時に (プロファイルはまだ独立な IOPEs を持つこともできるが、関係があるときには) プロファイルからプロセスの IOPEs へのリンクとして表すようになった。ちなみに TC では OWL-S の意図に基づき、プロファイルの IOPEs をセマンティックなサービス合成を行なうときに、サービスの実行時にはプロセスの IOPEs と使い分けている。

もう一つ OWL-S を使い始めた頃に驚いたのは、グラフ

*15 この時は (また他にも何回か) OWL-S が RDF の上に構築されたメリットを強く感じた。何か難しいことをするわけではなく、スキーマの整合性なども考えることなく、ほぼファイルをつなぐだけでだけの操作で済んだのである。

ンディングに実際の WSDL による Web サービスを実行できるだけの情報を記述できなかった点である。OWL-S のサービスと WSDL との関係は記述できるが具体的にセマンティックなインスタンスを WSDL のメッセージにどのように変換するか (またはその逆) の十分な記述がなかったのである。サービスは実際に実行できてこそ大きな価値があり、入手した記述に基づいて実際の Web サービスや UPnP のサービスが実行ができなければ、TC は実現できない。

そこで MINDSwap のグループと XSLT によるグラウンディングのスクリプトを記述する枠組みを作り、OWL-S に提案した。また我々は UPnP によるサービスも使っていたので、それに対するグラウンディングの仕組みも実現した。さらにこの XSLT グラウンディング・スクリプトを使って、実際にはグラウンディングする Web サービスが存在しなくとも、OWL-S の記述だけで実行可能な仕組みも作り上げた。これにより、例えば簡単なオントロジー変換などを OWL-S ファイルだけで実現できるようになった。

あとずっと OWL-S を使っていく上で課題もあった。一つは、どうしても標準という性格上余計なものがいっぱいある、あるいは入ってくるといった点である。これは別に使わなくていい部分は使わないという方針で対応した。より難しかったのは、新しいバージョンで、すでに TC で使っている部分の表現の仕方が変わっていくことである。一般に新しいバージョンはより分かりやすく、整合性が取れており、よりいいのではあるが、新しいバージョンに対応しようとするシステムのいろいろな部分を変更する必要があった。コアのエンジンは MINDSwap グループが対応したが、それ以外の変更多く、対応はなかなか大変であった。いろいろ試行錯誤し、例えば最新版に内部的にマップするなどの方法も考えた。最終的には、システムに必要な情報を TC 固有の形のメタモデルとしてパースして持つようにすることで解決した。

上記のようにセマンティック Web サービスとして OWL-S を使う難しさもあったが、セマンティック Web サービスのレベルに到達してスムーズに動くように、下のレイヤーからしっかりと作り上げていくことも難しい作業であった。動的な環境の中、ネットワークのレベルから始まり、ファイルの符号化 (encoding)、XML、WSDL、XSLT、RDF、RDFS、OWL のそれぞれのレベル、あるいは複数のレベルにまたがったいろいろな課題にきっちりに対応していく必要があった。その上でセマンティクスを活用して徐々に高度な機能を実現していった。

6. ま と め

米国から DAML のグループが、欧州からは OIL のグループが連携 (coalition を形成) し、OWL をオントロジー言語の W3C 標準として実現したように、SWSI でも

当初米国と欧州が連携して進んでいた。しかしその遅々たる進展に痺れを切らし、欧州のグループが自分たちで WSMO プロジェクトを立ち上げた。

米国では現在セマンティック Web サービスに対する大きな研究ファンドがなく、一方現在欧州ではセマンティック Web サービスを含むセマンティック Web 一般に対する研究ファンドの方が潤沢なので、欧州の研究が一気に進む可能性がある。

このばらばらな状況をどうにかしようと、つい最近 (2005 年 7 月) W3C への今後のセマンティック Web サービスに関する方向性や活動の勧告を話し合うためのワークショップ、“Frameworks for Semantics in Web Services” (FSWS) [FSW 05] が開かれた。このワークショップではセマンティック Web サービスで活躍する米国、欧州の主な関係者が一同に会し、OWL-S、SWSO/FLOWS、WSMO、WSDL-S などの主な技術も紹介された。しかしそれらを統合して、具体的にどのように進めていくかの結論は出なかった。W3C で今年中にセマンティック Web サービスのワーキング・グループができることはなさそうである。ただ一方でセマンティック Web サービスの要求仕様 (requirements) をまとめたり、WSDL 2.0 を拡張して WSDL の中にセマンティックな注釈の仕組み (WSDL-S 的なアプローチ) を開発しようという話しが持ち上がりつつあるが、現在議論があり、どうなるか分からない。

今後セマンティック Web サービスの標準はどのように展開していくのか。現状は非常に流動的であり、また混沌としており、予測は難しい。私見ではあるが、この混沌の原因は研究の方からの押し出しが強すぎるのではないかと考えている。どこに向かうにしろ、よりビジネスからの必要性による引張りを強めていく必要がある。

研究では独自性が重要であり、それぞれ異なっているのである。ただ多くのリソースを使って標準を実現していくには、具体的なアプリケーションが定まって、企業が必要であるから、求められて決めるという過程でないと、收拾がつかなくなってしまう可能性が高い。もし企業からの必要性がないのなら、標準化はまだ時期尚早なのかも知れない。

一方で Web サービスの世界では数多くの標準、WS-* [Vinoski 04] [MSW] が出てきており、ある意味発散しつつある。その一部は Web サービスに新たなセマンティクスを与えるものと見ることができ、セマンティック Web サービスはそれらに対して基礎を与えることができるのではないだろうかと考える。そういう意味でもサービスはやがてはセマンティック Web の非常に重要な適用分野の一つとなるであろうと予想している。

最後にセマンティック Web のオントロジー言語、OWL の時もそうであったが、セマンティック Web サービスも米国と欧州で動向が決められてしまっているを見るのは少しさびしい。日本からももっといろいろなアイデアが

出て、その動向に大きな影響を与えるようになることを期待している。

謝 辞

本稿を執筆するにあたって、メリーランド大学の Bijan Parsia 氏, Evren Sirin 氏, Stanford Research Institute (SRI) の David Martin 博士, サウサンプトン大学の Terry Payne 講師, トロント大学の Sheila McIlraith 教授, ジョージア大学の Amit Sheth 教授, IBM T. J. Watson Research Center の Richard Goodwin 博士より情報や資料を提供頂いた。また富士通研究所の湯原 雅信氏に内容に関する有益な指摘をいただいた。ここに記して感謝の意を表する。

◇ 参 考 文 献 ◇

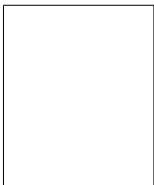
- [Akkiraju 05] Akkiraju, R., Farrell, J., J. Miller, , Nagarajan, M., Schmidt, M., Sheth, A., and Verma, K.: Web Service Semantics - WSDL-S, A joint UGA-IBM Technical Note, version 1.0, <http://lsdis.cs.uga.edu/library/download/WSDL-S-V1.pdf> (2005)
- [Ankolekar] Ankolekar, A., Martin, D., McGuinness, D., McIlraith, S., Paolucci, M., and Parsia, B.: OWL-S' Relationship to Selected Other Technologies, <http://www.daml.org/services/owl-s/1.1/related.html>
- [BPE] Business Process Execution Language for Web Services, <http://www.ibm.com/developerworks/library/specification/ws-bpel/>
- [DAM] The DARPA Agent Markup Language Homepage, <http://www.daml.org/>
- [FLO] Appendix B: Axiomatization of the FLOWS Process Model, <http://www.daml.org/services/swsf/1.0/swso/appendices/swso-fol.html>
- [FSW 05] W3C Workshop on Frameworks for Semantics in Web Services (FSWS), DERI, Innsbruck, Austria (2005), <http://www.w3.org/2005/04/FSWS/program.html>
- [Ghallab 98] Ghallab, M. and al., et : PDDL - The Planning Domain Definition Language V. 2, Technical Report, report CVC TR-98-003/DCS TR-1165, Yale Center for Computational Vision and Control (1998)
- [Horrocks 03] Horrocks, I., Patel-Schneider, P. F., Boley, H., Tabet, S., Grosz, B., and Dean, M.: Swrl: A semantic web rule language combining owl and ruleml, <http://www.daml.org/2003/11/swrl/> (2003)
- [IBM] Semantic Tools for Web Services, <http://www.alpha-works.ibm.com/tech/wssem>
- [KIF 98] KIF. Knowledge Interchange Format: Draft proposed American National Standard (dpans), Technical Report 2/98-004, ANS, Also at <http://logic.stanford.edu/kif/dpans.html>. (1998)
- [Martin] Martin, D., Burstein, M., Hobbs, J., Lassila, O., McDermott, D., McIlraith, S., Narayanan, S., Paolucci, M., Parsia, B., Payne, T., Sirin, E., Srinivasan, N., and Sycara, K.: OWL-S: Semantic Markup for Web Services, <http://www.daml.org/services/owl-s/1.1/overview/>
- [Masuoka 03a] Masuoka, R., Labrou, Y., Parsia, B., and Sirin, E.: Ontology-Enabled Pervasive Computing Applications, *IEEE Intelligent Systems*, Vol. 18, No. 5, pp. 68 – 72 (2003)
- [Masuoka 03b] Masuoka, R., Parsia, B., and Labrou, Y.: Task Computing - The Semantic Web meets Pervasive Computing -, in *The Semantic Web - ISWC 2003*, pp. 866 – 881, Sanibel Island, FL, USA (2003), Springer, LNCS No. 2870
- [MET] METEOR-S: Semantic Web Services and Processes, <http://lsdis.cs.uga.edu/projects/meteor-s/>
- [MIN] MINDSwap Homepage, <http://www.mindswap.org/>
- [MSW] Web Services Specifications, <http://msdn.microsoft.com/webservices/webservices/understanding/specs/>
- [OIL] Ontology Inference Layer (OIL), <http://www.ontoknowledge.org/oil/>
- [OWLa] Web Ontology Language (OWL), <http://www.w3.org/2004/OWL/>
- [OWLb] OWL-S API, <http://www.mindswap.org/2004/owl-s/api/>
- [OWLc] DAML Services - Tools, <http://www.daml.org/services/owl-s/tools.html>
- [OWLd] OWL Web Ontology Language for Services (OWL-S), <http://www.w3.org/Submission/2004/07/>
- [OWL-S] OWL-S, : DAML Services (DAML-S / OWL-S), <http://www.daml.org/services/owl-s/>
- [Paolucci 02] Paolucci, M., Kawamura, T., Payne, T. R., and Sycara, K.: Semantic Matching of Web Services Capabilities, pp. 333 – 347 (2002), LNCS No. 2342
- [PSL] Process Specification Language (PSL), <http://www.mel.nist.gov/psl/>
- [RDQ] RDQL - A Query Language for RDF, <http://www.w3.org/Submission/2004/SUBM-RDQL-20040109/>
- [Schlenoff 00] Schlenoff, C., Gruninger, M., Tissot, F., Valois, J., Lubell, J., and Lee, J.: The Process Specification Language (PSL): Overview and Version 1.0 Specification, *NISTIR 6459*, National Institute of Standards and Technology (2000), Gaithersburg, MD
- [Sivashanmugam 03] Sivashanmugam, K., Verma, K., Sheth, A., and Miller, J.: Adding Semantics to Web Services Standards, in *Proceedings of the 1st International Conference on Web Services (ICWS'03), Las Vegas, Nevada*, pp. 395 – 401 (2003), <http://lsdis.cs.uga.edu/lib/download/SVSM03-ICWS-final.pdf>
- [Sivashanmugam 04] Sivashanmugam, K., Miller, J., Sheth, A., and Verma, K.: Framework for Semantic Web Process Composition, *International Journal of Electronic Commerce (IJEC)*, Vol. 9, No. 2, pp. 71–106 (2004), http://lsdis.cs.uga.edu/library/download/SMSV_05.MWSCF_IJEC.pdf
- [Song 04] Song, Z., Labrou, Y., and Masuoka, R.: Dynamic Service Discovery and Management in Task Computing, in *MobiQuitous 2004*, pp. 310 – 318, Boston, USA (2004)
- [SPA] SPARQL Query Language for RDF, <http://www.w3.org/TR/rdf-sparql-query/>
- [SWR] SWRL FOL, <http://www.daml.org/2004/11/fol/>
- [SWSa] Semantic Web Services Initiative Architecture Committee (SWSA), <http://www.daml.org/services/swsa/>
- [SWSb] Semantic Web Services Initiative (SWSI), <http://www.swsi.org/>
- [SWSc] Semantic Web Services Language (SWSL) Committee, <http://www.daml.org/services/swsl/>
- [SWSd] Semantic Web Services Ontology (SWSO), <http://www.daml.org/services/swsf/1.0/swso/>
- [TCH] Task Computing Homepage, <http://taskcomputing.org/>
- [UPn] UPnP Forum, <http://www.upnp.org/>
- [Verma 05] Verma, K., Mochan, A., Zaremba, M., Sheth, A., Miller, J., and Bussler, C.: Linking Semantics Web service Efforts - Integrating WSMX and METEOR-S, in *Proceedings of SDWP 2005* (2005), <http://lsdis.cs.uga.edu/projects/meteor-s/wsd-s/WSMX-Meteor-S-interopability-final.pdf>
- [Vinoski 04] Vinoski, S.: WS-Nonexistent Standards, *IEEE Internet Computing*, Vol. 8, No. 6, pp. 94 – 96 (2004)
- [WSC] Web Services Choreography Working Group, <http://www.w3.org/2002/ws/chor/>
- [WSDa] W3C Web Services Description Working Group,

- <http://www.w3.org/2002/ws/desc/>
- [WSDb] Adding Semantics to Web Services (WSDL-S),
<http://lstdis.cs.uga.edu/projects/meteor-s/wsdl-s/>
- [WSD 04] WSDL-S: Adding Semantics to WSDL - White Paper,
<http://lstdis.cs.uga.edu/library/download/wsdl-s.pdf> (2004)
- [WSM] Web Service Modeling Ontology,
<http://www.wsmo.org/>
- [XQu] XQuery 1.0: An XML Query Language,
<http://www.w3.org/TR/2005/WD-xquery-20050915/>
- [XSL] XSL Transformations (XSLT) Version 2.0,
<http://www.w3.org/TR/2005/WD-xslt20-20050915/>
- [益岡 02] 益岡 竜介: DAML プロジェクトと Semantic Web - よりオントロジカルな世界へ -, 人工知能学会誌, Vol. 17, No. 4, pp. 392 - 399 (2002)
- [川村 05] 川村 隆浩, 長谷川 哲夫, 大須賀 昭彦, Paolucci, M., Sycara, K.: セマンティック Web サービスマッチメーカーの公開実験に基づく評価, 人工知能学会論文誌, Vol. 20, No. 6, pp. 426 - 436 (2005)
- [湯原 04] 湯原 雅信, 益岡 竜介: 情報機器を自由に連携させるタスクコンピューティング, *Computer & Network LAN*, No. 252, pp. 97 - 103 (2004)
- [福田 05] 福田 直樹, 山口 高平: 欧州におけるセマンティック Web サービスの現状と動向, 人工知能学会論文誌, Vol. 20, No. 6 (2005)

[担当委員 : × ×]

19YY 年 MM 月 DD 日 受理

著 者 紹 介



益岡 竜介

1962 年生。1985 年東京大学理学部数学科卒業。1987 年同大学修士課程卒業。2000 年同大学より数理科学博士号取得。数理科学博士。1988 年 (株) 富士通研究所入社。1991 年カーネギーメロン大学客員研究員。1993 年富士通研究所に戻り、2001 年より米国富士通研究所に移り、現在に至る。現在米国富士通研究所カレッジパーク研究所 主管研究員およびメリーランド大学 Adjunct Professor。ニューラル・ネットワーク、エージェントなどの研究を経て、現在セマンティック・ウェブ、パーベイシブ・コンピューティング、バイオインフォマティクス、タスク・コンピューティングなどに興味を持って研究を進めている。