セマンティック･ウェブサービス用エージェント
―― 計画それとも解釈実行？ ――
Agent Architecture for Semantic Web Servicing
― Planning or Interpreting? ―

小出 誠二 Seiji Koide
島田紀一 Norikazu Shimada

株式会社ギャラクシーエクスプレス
Galaxy Express Corporation

Semantic Web Services coined with Semantic Webs and Web Services has become to be deemed a cornerstone for the next generation of IT. Autonomous agents that discover, composite, and invoke Web Services without human intervention will play an important role to fulfill the hype in IT industry, accompanied with task and domain ontology on everyday life. In this paper, the ascription and description of Semantic Web Service Agents to task and domain ontology is discussed, and a general architecture for Semantic Web Service Agents are investigated. The agents should be cognitive at not only annotated Web contents (domain ontology) and Web Services (task ontology) but also changes in the outer-world, be a situated planner, and be capable of learning from experience. We need an intelligent framework for the universal agent that performs task ontology with situated awareness and planning.

1. はじめに

文科省 IT プロジェクト「IT を活用した大規模運用システムの支援システムの構築」[小出 2002][Koide 2003]にて，ロケット打上げを題材として，大規模運用システムの支援システムを開発している．このプロジェクトでは，各サブシステムがウェブサービスとして実装され，インタフェースエージェントが全体システムを統合して，状況に対応して適切にウェブサービスを呼び出すことによりロケット打上運用の支援をすることを考えている．平成 14 年度研究「システム試作」では，実現性検討を目的に，作成した一つの不具合シナリオに基づいて各サブシステムをウェブサービスとして開発し，そのウェブサービスをプログラムから呼び出すことにより，不具合監視，検知，診断，対応操作導出，オントロジによるメッセージ生成，および運用者によるマルチメディアデータ検索のデモンストレーションを行った．ここではウェブサービスは J2EE あるいは.NET により構築され，ウェブサービスの呼出しは.NET により，静的にプログラムされた．

平成 15 年度研究「システム設計･製作」では，事例ベース監視機能とモデルベース診断機能融合アルゴリズム開発のため，木状に整理されたタスクオントロジの末端とウェブサービスをバインドし，STRIPS 風の計画プログラムによりタスクオントロジを用いた計画を実施することによりウェブサービスの動的呼出しを行って，性格の異なる二つの推論方式の効果的な融合アルゴリズムを探索した[島田 2003]．ただし，ここでタスクオントロジの記述と検索には Schank らの事例ベース推論[Riesbeck 1989]を用い，OWL-S 記述にはなっていない．

次世代ウェブ技術として，ウェブサービスとセマンティックウェブが注目されている．ティム･バーナーズ＝リーがセマンティックウェブを提唱した当初には，ウェブサービスがまだ現実的ではなく，エージェントがウェブサービスを利用するという考えは現れていなかったが[TBL 2001]，DAML-S あるいは OWL-S の登場とともに，今日ではウェブサービスとセマンティックウェブを融合させて，セマンティックウェブ技術によりウェブサービスを動的自動実行させることこそがキー技術と目されるようになってきた．我々は，事例ベース監視機能とモデルベース診断機能の融合の経験から，各サブシステムをウェブサービスとして実装する大規模運用システムの支援システムにおいては，事例ベース監視とモデルベース診断の融合プログラムを発展させて，ウェブサービス用エージェントとすることが進むべき方向であると考えるが，セマンティックウェブ技術の一つである OWL-S(ウェブサービスのためのオントロジ記述言語)の適用にあたって，いくつかの解決すべき課題があると考える．

たとえば，OWL-S では，OWL ウェブマークアップ言語によりウェブサービスを記述する．このサービスプロフィールには input, output, precondition, effect が記述され，プロセスモデルには Sequence, Split, Choice, If-Then-Else, Iterate などの制御構造が記述される．STRIPS 風計画プログラムを用いれば，input, output, precondition, effect の情報からエージェントが与えられたゴール達成のためのウェブサービス呼出し列を生成することができ，必ずしもプロセスモデルの制御構造を必要としない．一方，複合プロセス(CompositeProcess)とプロセスモデル制御構造(ControlConstruct)記述を用いれば，タスクフローを抽象レベルから具体レベルまで記述でき，それを順次展開実行することでウェブサービス自動実行が可能となり，計画機能を必要としない．それではこの二種類の情報の関連はどうなのか，それら情報はどのように使用されるべきなのか．本報告では，これらの疑問も含め，OWL-S 記述によるタスクオントロジによってエージェントがウェブサービスを動的に自動実行するときの諸問題について考察し，プロフィール記述や制御構造記述を用いてウェブサービスを自動実行するエージェントが備えるべき一般的アーキテクチュアについて考察する．

2. ロケット打上支援エージェント：予備的研究

ロケット打上支援において，地上設備を対象に打上準備作業を監視する事例ベースプラント監視モジュールと，不具合発生時に異常候補から異常原因を絞り込むモデルベースプラント診断モジュールをウェブサービスとして立ち上げ，これら二つの推論方式を融合するロケット打上支援エージェントを開発した[島

連絡先：小出誠二，(株) ギャラクシーエクスプレス，港区浜松町 1-18-16，Fax03-5733-7190，koide@galaxy-express.co.jp

Table 1 Web Services

サーバ	ウェブサービス	前提条件	入力	出力
CBR	CBR1:センサ名リスト取得	配管ｸｰﾙﾀﾞｳﾝﾓｰﾄﾞ	-	センサ名リスト
	CBR1:センサ定性値リスト取得	配管ｸｰﾙﾀﾞｳﾝﾓｰﾄﾞ	センサ名リスト	センサ定性値リスト
	CBR1:プラント監視	配管ｸｰﾙﾀﾞｳﾝﾓｰﾄﾞ	センサﾃﾞｰﾀﾘｽﾄ	プラント運転状態
	CBR1:事例検索	配管ｸｰﾙﾀﾞｳﾝﾓｰﾄﾞ	センサﾃﾞｰﾀﾘｽﾄ	事例
	CBR2:センサ名リスト取得	タンククｰﾙﾀﾞｳﾝﾓｰﾄﾞ	-	センサ名リスト
	CBR2:センサ定性値リスト取得	タンククｰﾙﾀﾞｳﾝﾓｰﾄﾞ	センサ名リスト	センサ定性値リスト
	CBR2:プラント監視	タンククｰﾙﾀﾞｳﾝﾓｰﾄﾞ	センサﾃﾞｰﾀﾘｽﾄ	プラント運転状態
	CBR2:事例検索	タンククｰﾙﾀﾞｳﾝﾓｰﾄﾞ	センサﾃﾞｰﾀﾘｽﾄ	事例
	CBR3:センサ名リスト取得	機体タンク充填	-	センサ名リスト
	CBR3:センサ定性値リスト取得	機体タンク充填	センサ名リスト	センサ定性値リスト
	CBR3:プラント監視	機体タンク充填	センサﾃﾞｰﾀﾘｽﾄ	プラント運転状態
	CBR3:事例検索	機体タンク充填	センサﾃﾞｰﾀﾘｽﾄ	事例
	異常原因事例検索	-	異常原因	事例
	センサ定性値リスト検索	-	事例	センサ定性値リスト
	センサ名リスト検索	-	事例	センサ名リスト
	センサデータリスト検索	-	事例	センサﾃﾞｰﾀﾘｽﾄ
	異常原因リスト検索	-	事例	異常原因リスト
	危険リスト検索	-	事例	危険リスト
	対応操作リスト検索	-	事例	対応操作リスト
DSV	運転モード取得	-	-	運転モード
	センサデータ取得	-	センサ名リスト	センサﾃﾞｰﾀﾘｽﾄ
MBR	MBR:センサ名リスト取得	-	運転モード	診断用センサ名リスト
	原因診断	-	センサ定性値リスト 運転モード	異常原因リスト
	危険予測診断	-	異常原因 運転モード	危険
	対応操作導出	-	危険 運転モード	対応操作

田 2003]. 打上支援エージェントはこれらウェブサービスを適時必要に応じて活用しつつ, 運転者に役立つ情報を提供できることを目標とした. そこではタスクオントロジの記述や検索に Schank らによる MOP(Memory Organization Package) [Riesbeck 1989]を用い, サービス前提条件(precondition)および入出力(input, output)から適切なサービスを自動的に検索選択するようにした. ただし, OWL-S サーバは存在せず, 環境中に存在するウェブサービス情報とタスクオントロジは事前にエージェントに登録されている. Table 1 に実現したウェブサービスを示す. 表中 CBR は事例ベース監視サーバ, DSV はデータサーバ, MBR はモデルベース診断サーバであり, 表中ウェブサービスは実装されたウェブサービス･メソッドを示す。ただし, 表中の日本語記述は実際にはこれらのウェブサービスがバインドされた, MOP 名称およびそのスロット名称(前提条件, 入出力)となっている.

これらのウェブサービスを MOP の要素として定義し, Figure 1 に示すように組織化した. ツリーの末端のタスクは各々具体的なウェブサービスにバインドされている. 末端のタスクには Table 1 に示した前提条件および入出力情報がスロット情報としてある.

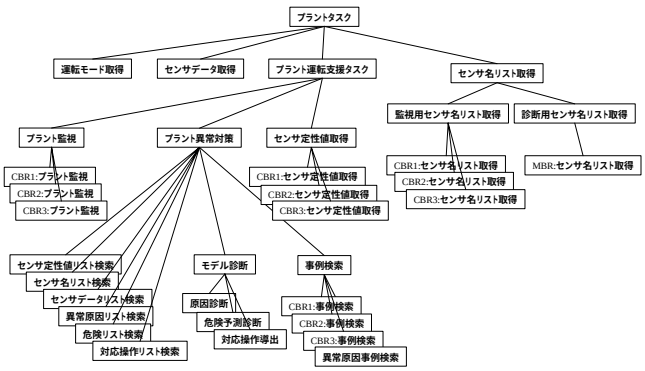

Figure 1 Plant Task Tree

エージェントが実現すべきゴールから必要なウェブサービスを呼び出す手続きを自動的に生成する手段として, 古典的な STRIPS 風の計画プログラム[Norvig 1992]を複数の計画立案可能なように拡張し, これをウェブサービスの呼出しに適応した. この計画プログラムは MOP 検索機能を用いてゴール達成可能なウェブサービス呼出し列を複数発見する. 発見された個々のウェブサービス呼出し列は, その時点での環境情報を持って個別に並列実行され, 実行結果を集めて比較熟考された.

この予備的研究において見出された問題点は, この計画システムがアジェンダを持たず, 複数の立案された計画が並列に実行されることにより, 無駄なウェブサービス呼出しが避けられず, また Susmann Anomaly のような計画要素が互いに干渉するような場合に対処できない, ということであった. ただし, この例題に限って言えば, Table 1 から分かるようにウェブサービスは副作用を持たず, またそのサービスの呼出しコストも低いものではあり実害はないが, エージェント一般のアーキテクチュアとしては機能不足である.

3. OWL-S

OWL-S によるウェブサービス記述には, プロフィール、プロセス、グラウンディングの三つの側面がある[DAML-S 2002]. ウェブサービスを利用するエージェントは, ウェブサービスの発見, 実行, 合成, 相互作用を遂行する. ここで, 発見にはサービスプロフィール, 実行にはサービスプロセスとグラウンディング, 合成と相互作用にはサービスプロセスが関係する. さらにサービス監視を記述する場合には, 監視モデルもサービスプロセスに記述されるであろう. 現在我々は OWL-S で記述されたコンテンツはタスクオントロジそのものであるという立場を取っているが, OWL-S をいざ実際に用いようとすると様々な疑問が生じる. 以下では, エージェントのウェブサービス発見, 実行, 合成, 相互作用, 監視のタスクの諸問題について考察する.

3.1 エージェントのウェブサービス発見

前章に述べた予備的研究では, Figure 1 に示したタスクオントロジがあらかじめエージェント内部にくみ上げられ, エージェント中のプラント監視スレッドにおいて, エージェントの手中にある具体的な前提条件(precondition)と入力(input)により「プラント監視」とか「監視用センサ名リスト取得」という中間抽象タスクから探索して, 適切な末端の具体タスクを見つけ出し, それにバインドされているウェブサービスを実行した. このウェブサービス発見の仕組みは OWL-S においても有効に用いることができる. すなわち, プロフィール記述には AtomicProcess や CompositeProcess のみならず, rdfs:subClassOf などを用いて Figure 1 に示したようなタスクタキソノミーを記述する. このタキソノミーは is-a 階層に相当し, composedOf や is-achieved-by ではない. その検索の仕組みとしてのポイントは, 抽象タスクが有する precondition や input は, 具体タスクの有するそれよりも抽象的でなければならないということである. こうすることにより, エージェントはタスク名ではなく, エージェントが現在手にしているウェブサービスの呼出し条件(precondition, input, output, effect)にマッチした具体的なサービスを抽象タスクレベルから検索することが可能になる. ここで precondition, input, output, effect のスロットに入れられるべき情報もドメインオントロジとして MOP によって定義されていることに注意されたい.

本格的研究では, タスクオントロジは OWL-S サーバに保存され, エージェントからの検索要求によって具体的条件にマッチした適切な AtomicProcess や CompositeProcess の情報を返すようにする.

3.2 エージェントのウェブサービス合成･実行

前章に述べた予備的研究では, タスクに関して CompositeProcess の記述はなく, 計画プログラムによってウェブサービスの実行シーケンスを求めた. ただし, その研究の目的が一般的なウェブサービス用エージェント研究ではなく, 事例に基づいて過去の経験から診断される結果とモデルに基づいて診断される結果の融合アルゴリズム開発であったため, 定型的

```
実行プラン1:((start)(execute CBR2:事例検索)(execute 異常原因リスト検索))
実行プラン2:((start)(execute 診断用センサ名リスト取得)(execute センサ定性値リスト取得)(execute 原因診断))
実行プラン3:((start)(execute CBR2:事例検索)(execute センサ定性値リスト検索)(execute 原因診断))
```

Figure 2 **計画プログラム出力例**

なプラント監視の部分，すなわち事例ベース監視機能により一定時間ごとにプラント状態を監視する部分は，メインスレッドとは別スレッドで「運転モード取得」，「監視用センサ名リスト取得」，「センサデータ取得」，「プラント監視」が順次定型的にプログラム実行された．ここで，プログラム上で実行指示されるのは抽象的なタスク名(MOP 名)であり，実際に起動されるべきウェブサービス決定の仕組みとして前項の検索機能が用いられた．定型的監視部分でウェブサービス呼出しの結果として異常が検知されると，計画プログラムが起動され，異常原因を求めるための計画が立案される．具体的な現在状態と抽象的な「異常原因リスト」が計画プログラムへの入力であり，計画の結果としてウェブサービスにバインドされた具体タスクのシーケンスが得られる．結果の一例を Figure 2 に示す．

ここで実行プラン1は過去の事例から類似事例を検索し，その事例の原因を出力する例，実行プラン2はモデルベース推論により原因診断する例，実行プラン3は過去の事例から類似事例を検索し，その類似事例のデータを用いて現在の原因診断をする例である．この実行プラン3も現在状態がまだ未成熟な異常状態や運転モードが異なる異常でデータに類似性がある場合には有効なものとなる．

この計画プログラムにおいては，具体的な現在状態のデータと抽象的なゴールから具体的なウェブサービス実行列が得られるが，そのための仕組みとしてやはり Figure 1 に示したタスクオントロジが用いられている．計画プログラムは最上位のタスクから出発して，オントロジツリーを探索しながら，望みの output をもたらすウェブサービスを発見し，後ろ向きに推論を重ねることで Figure 2 に示したような実行列を生成する．ここでエージェントの手中にある具体的なデータ（インスタンス）と input、output 定義にある抽象（クラス）との間でマッチングが起こっていることを注意しておく．

一方 OWL-S のプロセスモデルでは，CompositeProcess において Sequence や Repeat-Until のような ControlConstruct を記述することができる．これらの ProcessComponent はまた Process や ControlConstruct であり，様々な制御構造において入れ子の構造を作り出すことができる．これは言ってみればプログラムそのものであり，OWL-S で記述されたプログラムを解釈実行する処理系があってもよいのではと思わせる．そのような OWL マシンはおよそ以下のようなものであろう．

(1) プロセスが AtomicProcess なら，グラウンディング取り出して，ウェブサービス実行

(2) プロセスが SimpleProcess なら，realizedBy を取り出して(1)へ．さもなければ expand を取り出して(3)へ．

(3) プロセスが CompositeProcess なら，composedOf にある制御構造をし，制御構造に従って解釈実行する．

ここで，解釈実行は再帰的である．このような OWL マシンを製作し，試して見るのは興味深い．はたして，OWL-S 記述はプログラム言語としての性能を有しているのであろうか。ここで予想される一つの問題は入出力のバインドである．OWL-S 記述では入出力パラメータのバインドのための記述として，sameValues がある．これを用いれば，OWL マシン製作可能かも知れない．特に入計算関数型言語のように変数を有しないモデルであれば，多分実現可能であろう．しかし実用的な言語で変数を持たないものはない．変数の導入を考えたとたんに，スコープとエクステントの問題が生じる．はたして，現在の OWL-S 仕様で変数の導入が可能になるのか．OWL-S 仕様を拡張して，実用的な範囲でスコープとエクステントの問題を解決できるのであろうか．

前章で予備的研究において Susmann Anomaly のような計画要素が互いに干渉するような場合に対処できないことを述べた．ウェブサービスの連続呼出しあるいはウェブサービスの合成を考えたとき，別々の開発元リソースから得られたタスクオントロジの要素が互いに干渉する場合は大いに考えられる．計画問題の研究では Susmann Anomaly のような干渉問題を解決するために ABSTRIPS, SIPE, SHOP などで階層的プラニングが研究されてきた．すなわち，階層的プラニングはタスク分解によるタスク実行よりも，タスク合成によるタスク実行に必須であると考える．

3.3 エージェントのウェブサービス監視

現在の OWL-S 仕様で最も未定義な部分が，ウェブサービス監視である．現在の OWL-S 仕様ではトランザクションの概念がない．前述の予備的研究のように副作用のない場合には，問題とならないが，ユースケースとしてよく引用されるような，旅行手配のエージェントでは航空券とホテル予約のいずれかでも失敗すれば，そのサービス全体を失敗としてロールバックしなければならない．

ウェブサービス連携（コレオグラフィ）のトランザクション処理については，OASIS で標準化が進められている BPEL4WS と W3C で進められている WSCI において解決の努力が行われている．エージェントはこれらの標準化の努力を反映することになるであろう．

4. 関連研究

ウェブサービス発見については，それをハイパーグラフの最良被覆問題としてとらえて，最小のリクエストで最大の結果が得られるようにグラフ探索するものがある[Benatallah 2003]．グラフ探索の中には当然 rdf:type や rdfs:subClassOf のエッジも含まれると思われるが，RDFS や OWL のセマンティックスにおいては rdf:type の包摂性こそが重要であり，rdf:type や rdfs:subClassOf を他の一般のプロパティと区別しないわけにはいかない．その点で[Paolucci 2002]のウェブサービスのマッチメイキングの方法がより好ましい．2章の予備的研究で採用したマッチメイキングは，MOP の **slots->mop** という関数をそのまま利用したものであり，オントロジの上下概念や rdf:type の包摂性を利用しているという点で，本質的には[Paolucci 2002]と同じである．ただし，類似性の程度を計量する方法として，exact, plugIn，subsumes の三種類の測度を導入している点で，[Paolucci 2002]の方が柔軟性に富んでいると思われる．彼らの方法も **slots->mop** を使った方法も，サービス発見においてプロフィールに記述された precondition, input, output, effect に引用されたドメインオントロジがサービスの意味を規定するという点で本質的であり，セマンティックなウェブサービス発見の本来的なあり方であると言える．

ウェブサービスの合成では，SHOP2 という計画プログラムを利用して[TBL 2001]のシナリオを実現したものがある[Wu 2003]. SHOP2 は階層的タスクネットワーク(HTN)型計画システムで，CompositeProcess の入れ子構造と相性がよい．[Wu 2003]では DAML-S から SHOP2 へのトランスレータを開発し，AtomicProcess からなるフローを生成することを計画問題として捉えている．しかしこれははたして計画立案なのか，プログラム解釈なのではないか．

ここで，計画立案とプログラム解釈の違いは，決定論的であるかそうでないかの違いにあると考える．事前に変数のクラスや取り得る範囲が決定され，決定論的にエージェントのすべての行動を指定できれば，それは本質的にはコンパイル（自動プログラミング）と同じことであり，実行しながら展開していけばプログラム解釈であると考える．実際にエージェントが行動する場合にそのときでないと決定できない要素があり，時と場合によってエージェントの行動を動的に決定しなければならないとき，また前述したように，タスクの合成において計画要素が干渉するような場合に，計画立案が必要と考える．古典的計画問題では，初期状態からゴールに到達するまでの間に addlist や dellist 以外に状態は変化しないとの前提があるが，ロボットやソフトウェアエージェントの計画問題ではもはやそのような前提は成立せず，様々な計画手法が研究されてきている[三浦 2001][山田 2001]．そのような非古典的計画の考え方に立つと，実際[Wu 2003]において行われていることは，計画立案というよりも，自動プログラミングに近い．そこでは，Split，Split+Join は省略されており，Unordered も非並行的な場合のみが考慮されている（すなわち，マルチタスクではない．Choice は考慮されているが詳細な説明がない）．また，彼らは precondition が外部プログラムを呼び出すことを前提に，プランナの健全性と完全性のために，実行可能性（すべてのパラメータがグランドされている），終了可能性（有限な計算時間），再現性（同じ呼出しで同じ結果）を保証しなければならないと主張しているが，これは古典的計画問題の立場に立ったものであり，資源制約下での計画問題（実際はほとんどすべての問題が資源制約下にある）とか，エージェントが環境の変化に動的に対応していくことを目標とした場合には考慮にはあたらない．彼らはすべてのウェブサービスから提供される情報は，SHOP2 の計画時に変化しないということを前提としている（条件付きプランを今後研究する予定）．我々はより動的な環境変化に追随できる仕組みを次章で考察する．なお，外部プログラムの呼出し結果をアサインするための変数を SHOP2 のドメインに導入している．

DAML-S 記述のプロセスモデルを実行する DAML-S バーチャルマシン(DS-VM)なるものがある[Paolucci 2003]．一見単純な解釈実行系に見えるが実はそうではなく，たとえば choice は非決定的であるとされており，非決定的部分を実行時解決するために，エージェント推論システムが用意され，動的な意思決定がここで行われる．実際に Amazon.com のウェブサービスで実験が行われた．WSDL2DAML-S を用いて DAML-S を生成し，それらを実行するための制御フローが作成された．前述の SHOP2 では現実世界を変更することがないとして，計画時に precondition の計算にウェブサービス呼出しが行われることになっているが，Amazon のショッピングではそういうわけにはいかない．バーチャルマシンという言葉に反して，前述の SHOP2 よりも DS-VM のほうがより動的意志決定を行うエージェントに近い．なお，split は並列計算され，Split+Join は並列計算と同期が行われる．

5. ウェブサービス用エージェントの要件

5.1 ウェブサービス用エージェントの性質

ウェブサービス用エージェントの一般的要件について考察する．エージェントという言葉は非常に広い意味を持っているが，[Bradshaw 1997]は様々なエージェント研究の成果をまとめて，その本の第1章でエージェントの属性について詳細な分析を行った．ウェブサービス用エージェントも，[Bradshaw 1997]における分析結果と本質的に大きくかわることはないと思われるが，セマンティックウェブとウェブサービスに起因するところの特殊性と問題点について具体的検討が必要と考える．ユーザからのコマンドを受けて，ユーザの行為を代行してウェブサービスを呼び出すことにより，ユーザに利便を供給するウェブサービス用エージェントは，以下のような性質を持つであろう．

- 認知性：一連のウェブサービス実行を効果あらしめるために，サービス実行中に必要な範囲で外界の変化を選択的に認知する．
- 自律性：細かな指示を待つことなく，タスクオントロジとドメインオントロジをよりどころに，必要な行為を自ら実行する．
- 協調性：必要であれば他のエージェントとも協力する．
- コミュニケーション機能：他のエージェントとユーザである人間と，知識レベルのコミュニケーションができる．
- 推論機能：抽象的なタスクを現実に即して具体化したり，目標実現と現実の差異を見つけて問題解決したりできる．
- 持続性：長期間にわたって自己同一性がある．
- 適応性：学習により行動を改善できる．

5.2 ウェブサービス用エージェントとは何か

OWL-S 記述によるタスクオントロジと OWL 記述によるドメインオントロジにもとづいて，上記特徴を有するエージェントはユーザのおおざっぱな抽象的な指令を受けてそれをウェブサービスの呼出し列に具体化し，実行状態を監視しつつそれらウェブサービスを呼び出す．

タスクオントロジからウェブサービスの呼出し列を見いだすことは計画立案そのものであり，エージェントはプランナである．CompositeProcess の知識を利用するエージェントや AtomicProcess の合成を行うエージェントは階層的プランニングを行う．動的に変化する可能性のある情報を扱うエージェントは非古典的なプランナである．不完全かつ不正確な情報に対処するために，エージェントは条件付きプランニングあるいは実行モニタリングと再プランニングの機能を有しなければならない[Russell 1995]．条件付きプランニングでは帰結部実行の前に，条件部のチェックによって現実世界をプランニングに反映させようとし，再プランニングでは実行後に現実との差異を吸収する．

エージェントの感覚を外部世界と内部世界に分けたとき，エージェントの記憶の一部に外部世界を反映する部分が必要である．[Brooks 1999]のサブサンプションアーキテクチュアにおけるセンサと直結したメモリ部，Soar[Aasman 1995]のワーキングメモリのうち，特別な入出力機構によって非同期にセンサと直結した部分がそれに相当する．前述の予備的研究では，メインスレッドとは別スレッドで，一定周期で稼働する部分が外界へのセンサ部に相当し，「運転モード」，「プラント状態」などが現在状態として変数の値に反映されるようになっていた．メインスレッドでは

この変数をやはり一定周期でポーリングして,「プラント状態」が正常以外のときに異常原因調査のための計画を実行した.

外部世界の動的変化を前提に, OWL-S によるタスクオントロジを用いた計画問題を考える. したがって, 一般的には計画時に具体的なウェブサービス呼出しは実行するべきでない. すなわち, 条件付き計画であれば, 計画の結果として条件部チェックのコードが挿入され出力されるべきであって, チェックの結果のコードが出力されるべきではない. しかしこれで細部まで完結した計画を行おうとすると多分理論的には NP-完全問題になるであろう. OWL-S 記述のプログラムを一般のプログラム言語まで変換しようとすると, NP-完全の克服も含めて, 非常に困難な課題となるが (UML 図から完全なプログラムを生成するツールは, 理想的にはあってほしいが実際にはまだ存在していない), ウェブサービスという中粒度のサブシステムにグラウンディングすることで, OWL-S 記述のプログラム変換の実現性は高いと考える. 副作用なしコスト零のウェブサービス呼出しに限って計画時に呼び出してもよいように思えるが, 実行時に状態が変化していては意味がない. 初期状態からゴール達成までの計画立案ができない場合に備えて, エージェントはプラン完成前でも部分的にプランを実行しなければならない. そのようなエージェントは状況依存プランニングエージェント[Russell 1995]と呼ばれる.

5.3 エージェントのアーキテクチュア

エージェントの一般的フレームワークとして, 人間の認知メカニズムをモデルとした Soar [Rosenbloom 1991] [Aasman 1995] が参考になるかも知れない. Soar はプロダクションシステムであり, 長期記憶と作業記憶があるが, 扱う対象はエージェントの表層的な行動知識のみならず, 問題空間, ゴール・サブゴールなどを含み, 非常に汎用的な問題解決のアーキテクチュアとなっている. 長期記憶知識は認知メカニズムのためのプロダクション, 条件・行為のセットであり, 条件部・行為部間で変数を共有することにより, 条件部のマッチング機能によって変数の値を取り出し, 行為部でそれを利用することができる. 行為部にあって条件部と共有されない変数により, 新しいオブジェクトが生成される. 動的に変化するプロセス状態は作業記憶に保存される. ここには長期記憶によって引き出された情報が含まれ, Soar によって決定された結果情報, 外界から入力された情報, モータコマンドなどが保存される. このプロセス状態はある問題空間における単一の状態ではなく, システム全体の過渡的な状態であり, 問題空間の構成要素, 状態など, 問題解決の文脈におけるすべての要素が含まれる. 構造的には, 作業記憶はオブジェクトとオブジェクトに関する選好(preference)から成っており, 問題解決過程においては作業記憶の内容と長期記憶とのマッチングが並行的に網羅的に行われる. マッチしたプロダクションは並行的に発火され, この過程がすべてのプロダクションがマッチしなくなるまで繰返し行われる. 発火の度に作業記憶にはオブジェクトとオブジェクトに関する選好が書き込まれる. プロダクションが一つも発火しない静止状態になると, Soar アーキテクチュアにより, 作業記憶中の選好 (属性にはゴール, 問題空間, 状態 (以上変数) とロール:オペレータ, バリュー:5種類の選好のクラス (以上定数)) が参照されて, ある文脈でのロールに対するオブジェクトが選ばれてその特定されたオブジェクトがロールの現在の値となり, 再び問題解決過程が繰返される. 何らかの理由で選好も有効でないとき (袋小路), 状況を打開するためのサブゴールが自動的に生成される. 一つの問題空間における探索は, STRIPS のようなプランナで行うものと同等のレベルのものと考えられるが, このサブゴールを生成し, 推論を続行するための知識もある問題空間における知識として記述しなければならず, 汎用的かつ知的アーキテクチュアであると思われるが, そのプログラミングとデバッグには困難が予想される. Soar にはサブゴールの結果をまとめて, 複数のルールを要約した新しいルールを生成するというチャンキングによる学習機能があり, それまでの推論の経験から効率的な推論を実行できるようになる. 私見では, このチャンキングの機能は部分コンパイルに相当するのではないかと考える.

一方, 経験から学習をする機構として, Chef のような事例ベースの計画システムが考えられる[Riesbeck 1989]. Chef では, プラン記憶部, ゴールバリュー階層, 変更ルール部, プラン評価部, 失敗語彙部, 修正戦略部, 失敗予測部などがあり, ユーザから与えられたゴールは失敗予測部の知識を用いて Anticipator によって問題点が摘出され, ゴールと一緒に Retriever に渡される. Retriever は与えられたゴールを実行できそうな過去のプランを抽出し, Modifier がプラン評価部と修正ルール部の知識を用いて過去のプランを修正し, 修正されたプランは Storer によってプラン記憶部に蓄積される. 期待した効果が得られなかったプランは, 失敗語彙部と修正戦略部の知識を用いて Repairer によって修正され, その結果も Storer によってプラン記憶部に蓄積される. Soar のプロダクションにおいては属性情報のセットにより作業記憶部のオブジェクトと変数のマッチングが行われるが, オントロジのタキソノミはあまり重要な役割を果たしていない. 一方、Chef ではプログラム中に変数は出現せず, タキソノミが決定的に重要なものとなる.

SIPE[Wilkins 1988]では, Plan Critics and Solvers を有し, リソースのコンフリクトや, 実行の制御, プランの解釈などを行う. Execution Monitor はプランの監視を行い, Replanner と critics が既存のプランを修正する. 探索アルゴリズムで計画部の制御と解釈部を使用して計画作成を行う。SIPE の計画結果は, 条件部・実行部が連なった順序関係の有向グラフとして表現される.

オントロジ指向のプログラミングを考えると, Soar 的エージェントから出発するよりも, 事例ベース推論機構をエージェントの記憶とするエージェントから出発して, SIPE のような階層型プラニング機構を持つウェブサービス用エージェントのアーキテクチュアに発展させた方がよいように思われる. 今日の計画システムは, 単一のアーキテクチュアや機能というよりも, 階層型プラニング, 事例ベースプラニング, 混合イニシャティブなどの機能を統合した, ハイブリッドなものになっている[Munoz-Avila, 2001]. 今後の研究によって具体化する.

6. おわりに

我々は, 現在プラント運転支援用エージェントはプラント運転員のタスクのうち代行できる部分をプラント運転タスクオントロジから発見し, それを運転員と協力して実行し, 情報提供することで運転員の意志決定を支援することと定義している. そのようなタスクオントロジを OWL-S で記述したら, 万能エージェントがそのようなタスクオントロジを発見し, 実行することでプラント運転支援を実行することができるようになるであろうか. 言い換えれば, ユニバーサルマシンは計算機の動作モデルであるが, それと同様に, 万能エージェントが OWL-S によるタスクオントロジを身に付けることで, 個別エージェントになるようなアーキテクチュアを開発できるであろうか. OWL-S ではウェブサービスの発見, 合成, 実行, 監視がエージェントのタスクとされているが, そのタスクオントロジも OWL-S 表記すべきである. 万能エージェントはそのようなタスクオントロジを発見し, 部分コンパイルと状況依存プランニングを行って, ウェブサービスを実行することでウェブサービス用エージェントとなる. その実現に向かって努力する中で,

[TBL 2001]に述べられたオンオントロジ発見によって自らの能力をブーツアップするエージェントの姿が明らかになるものと考える.

Schank らの事例ベース推論システム MOP を用いた, ウェブサービス動的呼出し実装の経験から, OWL-S をタスクオントロジとしたときのウェブサービス用エージェントの要求仕様とあるべき姿について考察した. ロケット打上運用を題材として, 大規模運用システムの支援システムを開発中であるが, その中でウェブサービスを利用しそれによって支えられる人間支援用エージェントを開発する予定である.

7. 謝辞

本報告は, 文科省ITプログラム「ITを活用した大規模システムの運用支援システムの構築」, の一部として実施されたものである. 本プロジェクト実施では大須賀節雄東京大学名誉教授に技術評価委員長をお願いし, オントロジ構築に関して大阪大学溝口理一郎教授にご協力戴いている. 記して感謝の意を表する.

参考文献

[Aasman 1995] Aasman, J., Modelling Driver Behavior in Soar, Royal PTT, 1995.

[Bradshaw 1997] Bradshaw, J. M., Software Agents, AAAI, 1997.

[Benatallah 2003] Benatallah, B., et al., Request Rewriting-Based Web Service Discovery, ISWC2003, pp.242-257. 2003.

[Brooks 1999] Brooks, R. A., Cambrian Intelligence, MIT, 1999.

[DAML-S 2002] DAML-S Coalition, DAML-S: Web Service Description for the Semantic Web, ISWC2002, pp.348-363, 2002.

[小出 2002] 小出ほか,「ITを活用した大規模システムの運用支援システム」, SICE システムインテグレーション部門講演会, pp.399-400, 2002.

[Koide 2003] Koide, S., et al., “Operation-Support System for Large-Scale System Using Information Technology”, Proceedings of 5th Int. Conf. Enterprise Information Systems (ICEIS2003), Vol.4, pp.430-437, ESEO, Anger, 2003.

[Munoz-Avila, 2001] Munoz-Avila, et al., SiN: Integrating Case-based Reasoning with Task Decomposition, IJCAI-2001, Seattle, August 2001.

[三浦 2001] 三浦, ロボットにおけるプランニング, 人工知能学会誌, Vol.16, No.5, pp.617-622, 2001.

[Norvig 1992] Norvig, P., Paradigms of Artificial Intelligence Programming, Morgan Kaufmann, 1992.

[Paolucci 2002] Paolucci, M., et al., Semantic Matching of Web Services Capabilities, ISWC2002, pp.333-347, 2003.

[Paolucci 2003] Paolucci, et al., The DAML-S Virtual Machine, ISWC2003, pp.290-305, 2003.

[Riesbeck 1989] Riesbeck, C.K., and R.C. Schank, Inside Case-Based Reasoning, LEA, Hillsdale, 1989.

[Rosenbloom 1991] Rosenbloom, P.S., A. Newell, and J.E. Laird, Toward the Knowledge Level in Soar: The Role of the Architecture in the Use of Knowledge, Architectures for Intelligence (ed., K. Vanlehn), pp.75-111, 1991, LEA.

[Russell 1995] Russell, S. J. and Peter Norvig, Artificial Intelligence, Prentice-Hall, 1995.

[Schank 1994] Schank, R.C., A. Kass, and C.K. Riesbeck, Inside Case-Based Explanation, 1994, LEA.

[島田 2003] 島田ほか,「事例とモデルに基づくプラント運転支援, ―ロケット打上を題材として―」, 第 46 回自動制御連合講演会, pp.511-514, 2003.

[TBL 2001] Berners-Lee, J., T, Hendler, and O. Lassila, “The Semantic Web”, SCIENTIFIC AMERICAN, Scientific American, May 2001.

[Wilkins 1988] Wilkins, D. E., Practical Plannin: Extending the Classical AI Planning Paradigm, Morgan Kaufmann, 1988.

[Wu 2003] Wu, D., et al., Automating DAML-S Web Services Composition Using SHOP2, ISWC2003, pp.195-210, 2003.

[山田 2001] 山田, ソフトウェアエージェントにおけるプランニング, Vol.16, No.5, pp.623-628, 2001.