ユビキタスコンピューティング環境に適した サービス提供アーキテクチャにおけるサービス合成方式とその実装

武本充治[†,††] 大石哲矢[†] 岩田哲弥[†]
山登庸次[†] 田中洋平[†] 徳元誠一[†]
島本憲夫[†] 黒川章[†]
須永宏[†] 小柳恵一[††]

新しいネットワーク技術と端末技術が開発されることにより，近い将来ユビキタスコンピューティング環境が確立されるが，その環境でユーザにサービスを提供する方法については，まだ十分な技術が開発されていない．我々は，ユビキタスコンピューティング環境に適したサービス提供方法を実現する Ubiquitous Service-Oriented Network（USON）アーキテクチャを提案した．USON アーキテクチャにおいては，サービス提供は，サービステンプレート（ST）に基づき，サービス構成要素（SE）を組み合わせることによるサービス合成と，SE と ST の使用履歴などから ST を生成するサービス創発から構成される．ST は，BPEL などのワークフロー型サービス連携技術をユビキタスコンピューティング環境に拡張したものであり，SE は，Semantic Web などと同様に，Semantics を持つものである．本稿では，主に，サービス合成を実現するために必要となる基本的な機能とその実装について述べ，それらを実際に利用してのアプリケーションを使っての機能検証を行う．本稿で提案する技術により，近い将来のユビキタスコンピューティング環境におけるサービス提供が可能となる．

Service-composition Method and Its Implementation in Service-provision Architecture for Ubiquitous Computing Environments

Michiharu Takemoto,[†,††] Tetsuya Oh-ishi,[†] Tetsuya Iwata,[†]
Yoji Yamato,[†] Yohei Tanaka,[†] Seiichi Tokumoto,[†]
Norio Shimamoto,[†] Akira Kurokawa,[†] Hiroshi Sunaga[†]
and Keiichi Koyanagi[††]

While new network technologies bring ubiquitous-computing environments ever closer, methods to provide suitable services within such environments remain immature. We describe the Ubiquitous Service-Oriented Network (USON) architecture, a new service-provision architecture, covering the basic concept, components and their roles, and the service-provision mechanism. A USON provides services in two phases, a service-composition phase in which service elements (SEs) are combined on the basis of service templates (STs), and a service-emergence phase in which a new ST is obtained on the basis of the history of usage of SEs and STs. An ST is an extention of workflow-stype service-coodrination description and an SE has the semantic information suitable for ubiquitous-computing environment. This paper mainly describes the functions of USON service-composition, their implemantations using P2P technologies and evaluation with actual applications. The establishment of USON technologies on networks will provide various services within ubiquitous-computing environments.

1. はじめに

現在，ホットスポットの量的・質的充実や携帯電話のデータ通信品質の向上によるモバイルネットワーク技術や，多種多様なセンサもネットワークを介してアクセスするセンサネットワーク技術などの新しいネットワーク技術が進展しつつある．また，ネットワークに接続される端末も，PC/PDA などは高機能化・高

† 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
NTT Network Service Systems Laboratories, NTT Corporation
†† 早稲田大学大学院情報生産システム研究科
Graduate School of Information, Production and Systems, Waseda University

性能化し，タグやセンサなどが軽量化・低消費電力化しつつある．これらのネットワークや端末の技術の進展により，近い将来ユビキタスコンピューティング環境[1] が実現される．

ユビキタスコンピューティング環境は，組み込み型コンピュータが実世界に遍在することを意味し，その環境では，端末機能やネットワークへのアクセス機能が向上するために，どのような場所からでも組み込み型以外の通常のコンピュータ上の機能を利用できるという特徴も持つ．したがって，ユビキタスコンピューティング環境において，コンピュータにより実現される機能をサービスとして，ユーザに提供する場合には，その機能の利用可能性が時刻や場所によって変化するという特徴がある．また，これらの機能（サービスを実現するための部品であるため，サービス部品と呼ぶ）が，DNS や SMTP といったプリミティブなサービスや航空券予約（文献 2) など）やホテル予約（文献 3), 4) など）といったコンピュータ上のみで実現されるサービスのほか，センサやアクチュエータにより実現される機能などといった実世界の機器と不可分なものも含まれることが，ユビキタスコンピューティング環境の特徴である．

ユビキタスコンピューティング環境でサービス提供を行うためには，BPEL[5] などの，既存の Web サービスの提供で採用されている，サービス部品を静的に決定し，結合する方法では，不十分である．たとえば，サービス部品の結合をプログラムで記述する場合，そのプログラムは，利用するサービス部品の組合せごとに実装する必要がある．ユビキタスコンピューティング環境においては，ユーザごとに利用できるサービス部品は異なるために，全ユーザが利用する全サービスについて，想定されるすべてのサービス部品の組合せに対するプログラムが必要となる．これはプログラムの維持管理も考慮すると，非現実的である．

つまり，ユビキタスコンピューティング環境でサービス提供を行うためには，サービス部品を，状況に適応して動的に結合する技術の確立が必要である．我々は，この状況を，ユーザの嗜好やユーザの状況（ユーザコンテクスト），サービス部品の状況（サービス部品コンテクスト），両者の独立な状況（サービス独立コンテクスト）とし，それらに適応してサービスを提供するためのアーキテクチャとして Ubiquitous Service-Oriented Network（USON）[6)~8)] を提案・実装し，サービス提供技術について研究している．USON の基本的なサービス提供モデルは，Web サービスビジネスで注目されているワークフロー型のサービス連携技術（BPEL など）を，Semantic Web[9] などで検討されている Semantic 情報をサービス部品に付与することで，ユビキタスコンピューティング環境に適するように拡張したものである．

本稿は，以下の構成をとる．まず，2 章で，ユビキタスコンピューティング環境に適したサービス提供のモデルについて議論をし，3 章で，そのモデルを実現する USON の全体のアーキテクチャを述べ，4 章で，P2P 技術を用いたサービス合成機能の実装の説明を行う．5 章では，アプリケーション実装を通じて USON のサービス合成の各機能についての検証を行う．6 章では，USON のサービス創発について述べる．7 章では，サービス部品を動的に結合する他の研究との関連を述べ，8 章で結論と今後の課題について述べる．

2. ユビキタスコンピューティング環境におけるサービス提供モデル

本章では，ユビキタスコンピューティング環境におけるサービス提供に必要となる条件を整理し，その条件をどのように解決するべきかの方針を述べる．

2.1 状況と状況適応

上述のように，ユビキタスコンピューティング環境では，状況に適応したサービス提供技術の確立が必要である．この状況には，それがどこに存在している情報に起因しているかにより，サービスを構成するサービス部品側のサービス部品コンテクスト，サービス享受者（ユーザ）側のユーザコンテクスト，両者に独立であるサービス独立コンテクストに分類することができる．

2.1.1 サービス部品コンテクスト

サービス部品コンテクストとは，そのサービスを提供するサービス部品の状態により表現されるものと定義する．たとえば，CPU パワーやネットワーク帯域の場合は，そのときの利用可能資源量であり，センサやアクチュエータにより実現される機能の場合は，その場その時の利用可能性である．

サービス部品コンテクストは，多様なサービス部品が利用可能な状況下では，頻繁に変化することが想定される．たとえば，他のユーザが利用中で利用不可能であった共用ディスプレイが，他のユーザの利用が終了したため，利用可能となるなどの，資源の占有・解放で，サービス部品コンテクストは変化する．そのような場合でも，ユーザには同じサービスを継続的に提供することが求められる．

つまり，ユビキタスコンピューティング環境におけるサービス提供では，「条件 1：サービス部品コンテク

ストに適応したサービス部品の選択・利用機能が必要である」.

2.1.2 ユーザコンテクスト

ユーザコンテクストとは，あるサービスを享受しようとする，もしくは，享受している特定の 1 ユーザに関する情報に基づく状況と定義する．ユーザの嗜好，ユーザの地理的な情報，ユーザ端末がネットワークに接続されている形態のほか，そのときに利用しているサービス部品に関する情報も含む．特に，最後の利用しているサービス部品に関する情報を含む理由は，複数のサービス部品から構成されるサービスをあるユーザが享受している場合，そのサービス部品間の依存関係は，その個別のユーザの端末や常時使っているサーバ（Web ブラウジングにおける Proxy サーバやメイルサーバなど）といった，ユーザに近いところでのみ取扱いが可能であるからである．

ユーザコンテクストに適応したサービス提供を行うことは，ユビキタスコンピューティング環境に限らず，一般のサービス提供において重要である．逆にいうと，適応しないサービスを提供することは，ユーザが明示的にも暗黙的にも求めていないサービスを提供することを意味する．

また，ユーザコンテクストの管理は，その定義からも，サービス享受者であるユーザごとに管理を行うことが容易である．また，そうすることにより，そのユーザコンテクストを利用してのサービス自体の選択やサービス部品の選択が容易に行えるという利点もある．この利点を利用してのサービス提供が重要である．つまり「条件 2：サービス部品連携を行う機能の実体は，ユーザコンテクスト・ユーザごとに存在すべきである」.

2.1.3 サービス独立コンテクスト

サービス独立コンテクストとは，ユーザやサービス部品を取り巻く状況のうち，各ユーザにもサービス部品にも独立で変化する情報であると定義する．この例としては，時刻や天候のほか，各ユーザやサービス部品から独立した，サーバなどに蓄積されている情報（特定のサービスに関する流行情報，あるサービス A を利用した人の大多数が別のサービス B を連続して利用するなどの情報[10]）などがある．

ユーザコンテクストやサービス独立コンテクストによっても，利用すべきサービス部品が変更になることはある．つまり，あるユーザが昼間のオフィスと夜の自宅で利用したい Web 検索サービスが異なることなどである．これらのことを合わせると，条件 1 はサービス部品コンテクストに限ることなく，状況に適応することが必要になる．つまり，条件 1 は,「条件 1'：状況に適応したサービス部品の選択・利用機能が必要である」，に置き換える．

2.2 多様なサービス部品

ユビキタスコンピューティング環境では，多様なサービス部品が利用可能である．この多様性を分類すると，物理的媒体やプロトコルなどの下位層のアクセス方法の多様性，利用するためのインタフェースの多様性（アプリケーションレベルの上位層のアクセス方法の多様性），利用する場合の粒度の多様性に分類することができる．これらのうち，下位層のアクセス方法の多様性については，本研究の直接的対象ではないので，特に述べない．

2.2.1 インタフェースの多様性

インタフェースの多様性とは，同一の機能を実現する，異なるサービス部品が，それらにアクセスするためのインタフェースとして異なるものを持つことを意味する．

ユビキタスコンピューティング環境では，同一の機能を実現するすべてのサービス部品のインタフェースが，1 つの統一的に規定されたものに従って実装されていることは仮定できない．また，ユビキタスコンピューティング環境でのサービス部品の実装は，頻繁に変更されるために，それらを利用するためのインタフェースも頻繁に変更になる．

前者の例としては，その場その時の動画像データを獲得したい場合に，そこで利用可能なコンピューティング機能が，ユーザがいる部屋に存在する PC に USB 接続されたカメラである場合と，その部屋に存在する監視カメラである場合に，まったく同じインタフェースで操作できることを仮定するシステムは，ユビキタスコンピューティング環境に適しているとはいいがたいということである．

したがって，インタフェースが異なる場合でも，可能な限り，その差を解決する機能が必要である．ユビキタスコンピューティング環境のサービス提供には,「条件 3：統一的な標準に従っていないインタフェースの解決機能を実現する」ことが重要である．

2.2.2 粒度の多様性

粒度の多様性とは，同一の機能を実現するサービス部品が，必ずしも同一レベルの機能に分解されることはないということを意味する．

ユビキタスコンピューティング環境では，その場その時に利用可能なサービス部品を利用するために，あるサービス部品はそれ以上分解不可能，また，別のサービス部品はさらに他のサービス部品から構成されてい

るということがある．たとえば，テキストデータを紙に印刷する機能を実現するものとして，ネットワークに接続されたプリンタを利用する場合と，ネットワークに接続された PC のローカルポートに接続されたプリンタを利用する場合と，テキストデータを FAX 情報に変換するサーバと FAX 機器の 2 つを連携して利用する場合がある☆．

したがって，状況に適応して，同一機能であっても，異なる粒度のサービス部品を利用できる機能が必要である．つまり,「条件 4：異粒度のサービス部品利用ができる」ことが，アーキテクチャや実装を検討するうえで重要である．

2.3 結合方針

上述のとおり，ユビキタスコンピューティング環境でのサービス提供のためには，ユーザ側に置かれた機能により，複数のコンピューティング要素を状況に応じて選択し，インタフェースを解決し，結合する必要がある．

ユビキタスコンピューティング環境では，あるサービスを提供するために必要な機能を有するサービス部品が 1 つないし複数ある場合に，いつ決定すべきかについては，Eager に行う方針と Lazy に行う方針がある．

Eager に行う方針は，あるサービスに関係するサービス部品をすべて選択し，それらのサービス部品のインタフェースはすべて解決してから，サービス提供を開始するというものである．逆にいうと，あるサービスが提供開始されている場合は，部分のサービス部品のインタフェースが解決できないためにサービス提供が停止することはない．

この点は，すでに提供されているネットワークサービスで守られているものである．実ユーザを獲得するためには，ユビキタスコンピューティング環境のサービス提供でも，守るべきものである．

一方，Lazy に行う方針は，最低限，サービス提供の開始時に必要なサービス部品についてのみ，選択とインタフェース解決を行うというものである．後は，必要に応じて，サービス部品の選択とインタフェース解決を行う．

Lazy に行う方針の長所は，サービス提供の途中で，新しい機能が必要になった場合や一部のサービス部品の利用可能性が変化した場合などに，対応できるという点である．また，インタフェースに関して，必要なサービス部品についてのみ解決できれば，サービス提供を開始できるという長所もある．

ユビキタスコンピューティング環境では，どのようなサービス部品が利用可能となるかは，事前には分からないため，Eager に行う方針を採用することは，困難である．

したがって，Lazy に行う方針を採用するものの，Eager に行う方針の持つ，提供を始めたサービスは，インタフェースが解決不能ということでサービス提供が停止することがないことを保証することが求められる．つまり，機能としては「条件 5：Lazy なサービス部品決定と動作保証を行う」ものとすることが求められる．

3. USON アーキテクチャ

3.1 基本要素：SE と ST

上述の 5 つの条件を満たすものとして，我々は USON アーキテクチャにおいては，サービスのメタ設計図であるサービステンプレート（Service Template，以下，ST）とサービス部品であるサービス要素（Service Element，以下，SE）を利用してのサービス提供モデルを採用した．

SE は，ユビキタス環境中に存在するサービス部品で，SE 記述により，その提供する機能が記述され，SE に共通のプロトコルで，ネットワーク中から発見され，メッセージを送信されることで，その提供する機能を実現するものとして定義される．また，サービス部品の決定は SE の発見と選択により実現される．また，サービス部品として同一の機能を持つ SE 群を指す概念として，サービスカテゴリ（Service Category，以下，SC）を定義する．

ST は，サービス提供のための SE を用いてのサービス連携の表現形式であり，具体的な SE ではなく SC により，ワークフロー表現されるものとして定義される．そして，サービス提供直前に，SE を発見・選択する．つまり，サービスを直接表現するものをそのサービスの設計図であるとすると，ST はサービスの設計図ではなく，設計図の設計図であるメタ設計図である（**図 1**）．また，USON アーキテクチャに基づくシステム中では，SE や ST の利用履歴などの情報を用いて，新たな ST を生成する機能も持つ．

なお，図 1 の例では，1 つの ST 中には 1 つのワークフローしか記述していないが，複数のワークフローやワークフローの分離・合流も記述できる．

SE や ST を取り扱うことによりサービスを提供する実体は,「条件 2：サービス部品連携を行う機能の実

☆ 最後の例のサービス部品の操作は，前の 2 つの例と異なるインタフェースを介して行うこともあり，これが 2.2.1 項のインタフェースの多様性を示す例である．

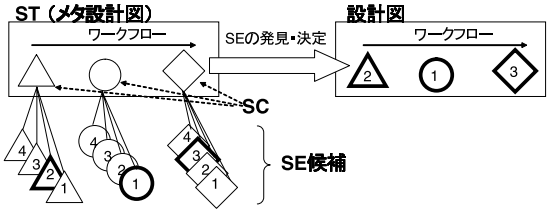

図 1 メタ設計図（ST）とサービス設計図
Fig. 1 Meta-design (ST) and service design.

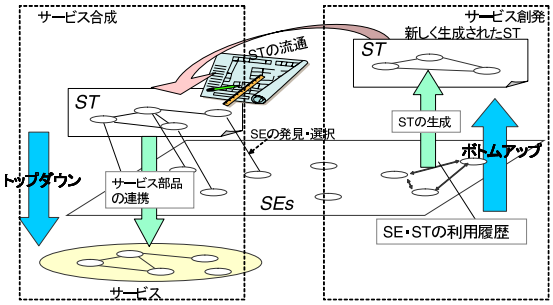

図 2 サービス提供
Fig. 2 Service provision.

体は，ユーザコンテクスト・ユーザごとに存在すべきである」を満たすために，ユーザごとに存在する．

つまり，Web サービスの連携のための BPEL 記述を実行するエンジンのように，複数のユーザのサービスの要求に応える 1 つの実態が存在するわけではない．したがって，ユーザ端末上や，ネットワーク上のユーザごとのポータル上に置かれる実装する方法をとることができる．これら SE や ST を発見・選択・利用する実体を USON エンジンと呼ぶ．

なお，人間が行う ST の実装については，SE の実装者が，自分の SE をユーザに利用してもらうことを意図して行うことも，ユーザが自分に適したサービスを利用することを意図して行うことも，第三者がサービス仲介を意図して行うこともある．このように，サービス部品提供者とサービス提供者を分離することが USON アーキテクチャの 1 つの特徴である．

3.2 サービス合成とサービス創発

USON アーキテクチャは，ユビキタスコンピューティング環境において，ユーザにその場その時に適切なサービスを提供するものである．**図 2** に示すように，このサービス提供は，サービス合成とサービス創発の 2 つの動作から構成される．

サービス合成は，ユーザが享受したい ST を用いて，USON システム中のサービス合成機能が，その場その時の状況に適した SE を発見・選択し，それらを ST 中の SC にあてはめることで，所望のサービスの設計図を獲得し，それらを実行することでユーザはサービスを享受するという，トップダウン的な動作を行うものである．

また，サービス創発は，サービス合成などにより利用された ST，SE の利用履歴を蓄積し，その蓄積された情報を元にして，新しい ST を生成するという，ボトムアップ的な動作を行うものである．この動作により生成された ST は，ネットワーク中を流通し，それを発見したユーザが利用可能である．

この 2 つの動作によりサービス提供を行うことにより，2 章の「条件 1'：状況に適応したサービス部品の選択・利用機能が必要である」を満たすシステムを実現することができる．つまり，サービス部品コンテクスト，ユーザコンテクスト，サービス独立コンテクスト（時刻や天候など）が変化した場合には，現在利用中の SE，もしくは，ST が最適なものではなくなっている可能性がある．その場合に，USON エンジンが，一部の SE のみを新たに発見・選択し，サービス合成を行うか，新たに ST を発見・選択して，必要な SE を発見・選択して，サービス合成を行う．これにより，状況に適応した SE を利用してのサービス提供が可能となる．

また，サービス独立コンテクストのうち，特定のサービスのみが頻繁に利用されるなどのネットワーク全体として利用されるサービスの種類などについても，それが変化した場合は，サービス創発動作である ST 生成により対処する．

3.3 SE と ST によるサービス合成

この節では，SE と ST によるサービス合成の詳細について説明する．サービス合成動作の実体である USON エンジンは，ST・SE を発見・選択し，SE の機能を呼び出すことで，サービス部品としての SE を利用する．基本動作は，ST を解釈実行していくととらえられる．

図 3 に，USON エンジンによりユーザがサービスを受けるときのユーザと USON エンジンのやりとりを示す．図 3 上が，基本パターンである．すなわち，ユーザがあるサービスを受けたいときに，USON エンジンに対して，そのサービスを実現する ST の発見要求を行い，USON エンジンはネットワーク中から適切な ST を発見してくる（実線矢印で表記）．候補となる ST が複数発見された場合は，その中から適切なものを選択するために，ユーザに問い合わせることもある．その後，選択された ST を解釈実行する．この解釈実行過程で，SE の発見や SE の選択が行われる．ST と同様に，候補となる SE が複数発見された

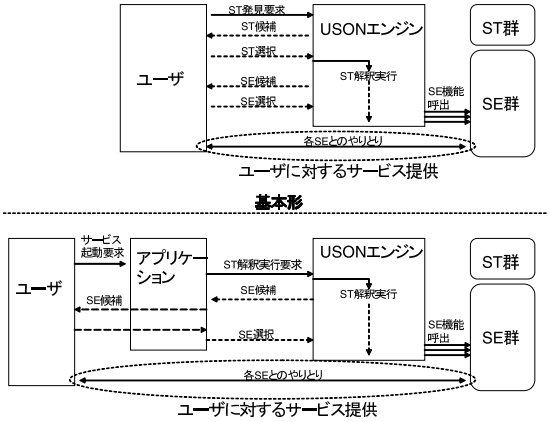

図 3 ユーザと USON エンジンのやりとり
Fig. 3 Interaction between user and USON engine.

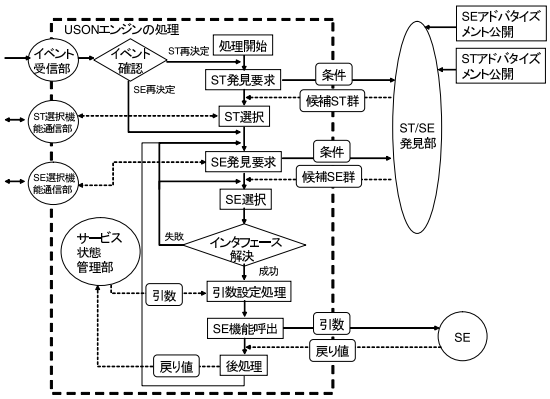

図 4 サービス合成の流れ
Fig. 4 Service-composition flow.

場合には，その中から適切なものを選択するために，ユーザに問い合わせることもある．これらのユーザへの問合せとその返答については破線矢印で表記している．USON エンジンにより機能を呼び出された各 SE とユーザのやりとりがサービス提供となる．

図 3 下のように，基本形の派生として，ST 選択の処理を省略，つまり，特定の ST にあらかじめ決定しておき，目的のサービスを提供するアプリケーションとして実装する方法もある．この方法でサービスを提供すると，3.2 節で述べた，ST を最適なものに保つことはできなくなる．しかしながら，特定の ST によるアプリケーションの動作確認をする場合には有用であるため，5 章で述べるアプリケーション例は，この方法で実装，動作確認している．

図 4 に，全体の流れを示す．なお，状況の変化は USON エンジン外部からのイベントとして通知され，必要に応じて，ST 発見要求，もしくは，SE 発見要求の処理を行う．

なお，USON アーキテクチャとしては，ST や SE の発見については，どのような方式で実現することも特に仮定はしていない．つまり，UDDI[11] などのサーバを用いてもよいし，P2P 的サーチ・ディスカバリプロトコルを用いてもよいし，あらかじめ定義され，ローカルに持っているリストの中から選択してもよい．

「条件 5：Lazy なサービス部品決定と動作保証を行う」もののうち，サービス部品の決定を Lazy に行うために，SE の決定と，そのインタフェースの解決は，1 つの ST の中のすべての SC について処理を行ってから，SE を呼び出すのではなく，ある SE 呼び出しについて，必要な情報がすべて解決した段階で，当該 SE を呼び出す方式を採用している．

また，条件 5 のうち，動作保証については，2 段階の方針で保証する．つまり，人手で記述された ST，もしくは，どこかで利用された実績のある ST については，適切な SE が確定する環境下では，動作することは保証できる．また，サービス創発過程で，システムにより生成される ST についても，その ST 生成過程で，適切な SE がある場合に，動作することが保証できるものを生成する．動作保証のある ST を生成するアルゴリズムについては，動作しているサービスを分解して，再合成する方法を中心に検討している[12]．

3.3.1 アドバタイズメントの公開

以下では，具体的な動作について述べる．まず，USON エンジンが動作を行う以前に，上述した SE 記述，ST をアドバタイズメントとしてネットワーク中に公開する．この動作は，SE や ST 側で行われる．つまり，SE や ST の実装者が手動で公開したり，SE や ST のアドバタイズメントを公開する別プログラムにより公開したりする．

なお，この公開されたアドバタイズメントは，SE の状態，つまり，サービス部品コンテクストが変化した場合は，SE 側で，変更し，再公開する．

3.3.2 ST の決定

ST の決定処理は，ユーザの明示的な要求のほか，ユーザの通常の振舞いやネットワークへの接続などの状況（コンテキスト）から，適切な ST をネットワーク中から発見・選択する処理である．この処理は，ネットワークに対する発見要求とその結果の選択の処理に分かれる．

ST 発見要求を処理する部分は，SE 発見要求を処理する部分と共通で，USON エンジンの外に，ST/SE 発見部として置かれる．

ST 発見要求の結果，複数の ST 候補が得られた場

合は，USON エンジンの外に設置される ST 選択機能により，選択される．

3.3.3 SE の決定

SE の決定処理は，決定された ST の中に記述してある 1 つの SC に対して，適切な SE をネットワーク中から発見・選択する処理である．この処理も，ネットワークに対する発見要求とその結果の選択の処理に分かれる．すなわち，発見要求は，USON エンジンの外にある ST/SE 発見部により処理され，複数の SE 候補が得られた場合は，同じく USON エンジンの外に設置される SE 選択機能により，選択される．

USON アーキテクチャでは，アドバタイズメントを共通化し，ST と SE の発見を同じ処理で行う．そのため，SE を発見しようとした場合にも，ST が得られる場合がある．その場合は，再び，その ST に対して，同様の処理を行う．これにより,「条件 4：異粒度のサービス部品利用ができる」ことを満たすことが可能となる．たとえば，ある ST が SC に「印刷機能カテゴリ」と指定してあり,「ユーザがいる部屋」という条件で発見要求をかけた場合，プリンタが発見されず，その代わりに「FAX サーバと FAX 受信機を用いてのサービス」が記述してある ST が見つかる場合があるということである．この場合，その ST について SE の決定処理を行うこととなる．

3.3.4 インタフェース解決機能

実行時にインタフェースを解決する機能が，ユビキタスコンピューティング環境におけるサービス提供には必要である.「条件 3: 統一的な標準に従っていないインタフェースの解決機能を実現する」ためには，SE のインタフェースは，統一的に標準を決められたものに従って実装されていることはない．しかしながら，インタフェースを解決するための何らかの情報は共有・流通する必要がある．この情報を語彙情報と呼ぶ.

語彙情報は，SE 記述と ST において記述され，ST 中の SE 機能呼び出し記述に従って，発見・選択された SE の機能を呼び出すために利用される．語彙情報は，各 SE や ST 中の記述における呼び出される機能の名前（機能名）と，その機能を呼び出すときに必要になる引数名と，他の語彙情報において，同等の機能名と引数名がどのように記述されるかを定義する情報，から構成される．

3.3.5 SE 機能呼び出し

SE 機能呼び出し処理は，上記までの処理で決定した SE に対して，適切な引数をメッセージ中に設定し，送信することで，サービス部品として SE が持っている機能を実現する処理である．この引数の設定は，引数名とその値の組で行われる．また，戻り値も，戻り変数名とその値の組として，呼び出し側が得る．

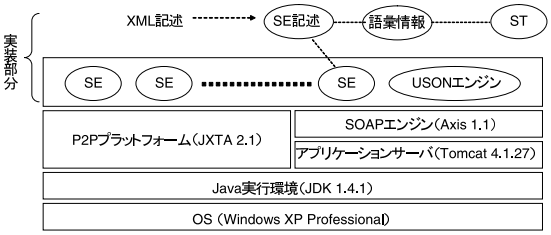

図 5 USON エンジン実装のソフトウェア構造
Fig. 5 Software structure of a USON engine implementation.

通常，ST の中に記載される複数の SC により決定される SE の間では，前の SE 呼び出しの結果を，次の SE 呼び出しの引数に用いるということがある．この場合に，前の SE で用いられている変数名やその値を，次の SE に引き継ぐ必要がある．そのために，値を保持するサービス状態管理部を USON エンジン中に持つ．

3.3.6 状況変化への対応

以上の動作で，最初の 1 SE の持つ機能を利用するところまでの処理が終わった．通常，1 つの ST には複数の SC が含まれるので，上述の動作を繰り返すことで，ユーザへサービスを提供する．そのサービス提供中に，状況が変化する場合がある．この状況の変化は，イベントして USON エンジンに伝わる．そのイベントにより，サービス部品としての SE の利用可能性が変化した場合は，SE 発見処理の再起動を，ユーザが別のサービスを必要とした場合は，ST 発見処理の再起動を，USON エンジンは行う．このときに，その直前まで動作していた SE に対する状態の引継ぎ処理も行う．

4. USON エンジンの実装

USON エンジンのプロトタイプ実装は，ネットワーク上のユーザごとのポータル上の USON エンジンとユーザ端末上の USON エンジンを想定し，Windows XP Professional のインストールされた PC の上に実装をしている．この実装を，ISP などのインターネット上に設置すれば，ポータル上の USON エンジンに，ユーザの持ち歩く端末上に設置すれば，ユーザ端末上の USON エンジンとなる．なお，PDA や携帯電話へのエンジンの実装は，現在検討中である．

Windows XP Professional の上に配置するソフトウェア構造を**図 5** に示す．また．SE 記述，ST，および，それらと関係する語彙情報は XML 記述として表

```
1: <USON xmlns:uson="http://uson.org">
2:   <serviceType>Service Element</serviceType>
3:   <serviceName>Pet Camera SE USB-CAM30S</serviceName>
4:   <serviceCategory>Camera</serviceCategory>
5:   <interface>
6:     <vocabulary>
7:       <import location="http://localhost:8080/axis/xml/vocaPetCameraSE_30S.xml"/>
8:     </vocabulary>
9:
10:    <wsdl:definitions targetNamespace="http://localhost:8080/axis/services/CameraSE30S"
11:                      xmlns="http://schemas.xmlsoap.org/wsdl/"
12:                      xmlns:apachesoap="http://xml.apache.org/xml-soap"
                    <省 略>
20:                      xmlns:xsd="http://www.w3.org/2001/XMLSchema">
21:      <wsdl:message name="getCameraURL_30SResponse">
22:      <wsdl:part name="getCameraURL_30SReturn" type="xsd:string"/>
23:      </wsdl:message>
                    <省 略>
30:    </wsdl:definitions>
31:  </interface>
32: </USON>
```

SEであることの記述
語彙情報
WSDL記述

図 6 SE 記述例（一部）
Fig. 6 Part of an SE description.

現した．Java 実行環境（JDK 1.4.1 を利用）の上に，以下のミドルウェアが搭載されている．

現在の実装では，SE の機能呼び出しは，SOAP メッセージにより実現している．そのため，SOAP メッセージを処理するためのミドルウェアとして，Axis 1.1 [13] とアプリケーションサーバとして Tomcat 4.1.27 [14] を利用した．

前述したように，USON アーキテクチャとしては，ST や SE の発見については，どのような方式で実現することも特に仮定はしていないが，現在の実装においては，サービス部品のその時々の状態を特定のサーバに登録する必要がないという利点から，P2P 的なサーチ・ディスカバリプロトコルを利用するものとし，JXTA プロトコル[15] を用いて実装した．利用したミドルウェアは JXTA 2.1 である．

4.1 SE 記述と ST

図 6 に SE の記述例を示す．この例は，5.2 節で述べる移動物体追跡サービスの中で使用しているカメラの SE 記述の一部である．現在の実装においては，SE は，サービスタイプ（図中，`serviceType`），サービス名（同，`serviceName`），SC（同，`serviceCategory`）と，インタフェース情報（同，`interface`）について規定する．

サービスタイプは，ST/SE 発見部により発見された ST/SE 記述が，ST であるか，SE であるかを示すものである．サービス名は，ST/SE 記述に対する名前である．

インタフェース情報の中には，「条件 3：統一的な標準に従っていないインタフェースの解決機能を実現する」ための語彙情報（同，`vocabulary`）と WSDL 記述[16] を含む．サービス部品として SE が提供する機能は WSDL 記述により表現される．

ST によるワークフロー記述も，STONE [17] のように，すべて USON 用に独自に定義するのではなく，ビ

```
1: <sequence>
2:   <input-param name="ServiceCategoryA.Operation1.X"/>
3:   <input-param name="ServiceCategoryA.Operation1.Y"/>
4:   <invoke category="ServiceCategoryA" operation="Operation1"/>
5:   <copy from="ServiceCategoryA.Operation1.Z"
6:         to="ServiceCategoryB.Operation2.X"/>
7:   <invoke category="ServiceCategoryB" operation="Operation2"/>
8:   <output-param name="ServiceCategoryB.Operation2.Y"/>
9: </sequence>
```

SCで機能を指定

図 7 ST 例（一部）
Fig. 7 Part of an ST description.

```
1: <invoke name="SC1Invoke"
2:         partner="sc1"
3:         xmlns:ns5="http://host001:8080/axis/services/ServiceElementA1"
4:         portType="ns5:ServiceElementA1" operation="serviceMethod"
5:         inputVariable="sc1request" outputVariable="sc1response">
6: </invoke>
7: <assign name="replyAssign">
8: <copy>
9:   <from variable="sc1response"/>
10:  <to variable="response"/>
11: </copy>
12: <invoke name="SC2Invoke"
13:        partner="sc2"
14:        xmlns:ns5="http://host001:8080/axis/services/ServiceElementB2
15:        portType="ns5:ServiceElementB2" operation="serviceMethod"
16:        inputVariable="sc2request" outputVariable="sc2response">
17: </invoke>
```

Webサービスに固有のオペレーションを指定

図 8 BPEL 記述例（一部）
Fig. 8 Part of a BPEL description.

ジネスの世界で標準になりつつある BPEL 記述[5] を拡張する形で定義している．その拡張点は，呼び出す機能の指定については，具体的な SE を記述するのではなく，SC により抽象化したもので記述するものである．

図 7 に ST の記述例を BPEL から拡張された部分について示す．この例では，SC として `ServiceCategoryA` に分類される SE のある機能（`Operation1`）を 2 引数で起動（4 行目）した戻り値を，SC として `ServiceCategoryB` に分類される SE の別の機能（`Operation2`）を起動（7 行目）するものを示す．この `Operation1` や `Operation2` は，SE 実装中のオペレーション名ではなく，SC レベルでの抽象化された表現である．また，引数も同様に，抽象化されている．なお，図 7 では記述していないが，図 6 と同様に，ST は `serviceType` タグを持ち，`Service Template` が記述してあることで，ST であることを示す．

比較のために，BPE L 記述で同様の処理を記述した例を**図 8** に示す．BPEL では，具体的なオペレーション名（`serviceMethod` など）や引数名（`sc1request`，`sc1response` など）を記述する．ST は，SC を用いた抽象的な記述であるために，BPEL 記述と比較すると，記述量は少ない．インタフェース解決のための語彙情報の記述は，SE の場合と同じく，ST 中には `import` タグを用いて指定する．SE の機能呼び出しは，インタフェースを解決しながら，実行されるため，

オーバヘッドが生じる．

なお，USON を利用する技術として，既存の Web サービスを USON の SE とするツールも開発中である．つまり，ある Web サービスの WSDL 記述から，SE 記述を生成するツールである．このツールにより，Web サービス開発者が開発した Web サービスを，SE としての公開が可能となる．

4.2 SE 記述/ST のアドバタイズメントとしての公開

SE 記述，ST のネットワークの公開の実装は，JXTA のアドバタイズメントを拡張し，SE 記述や ST そのものをアドバタイズメントとしている．図 4 では，ST/SE 発見部と表記しているが，JXTA で実装しているため，ST や SE が存在しているピアやランデブピアに情報があり，その発見動作も，1 カ所で処理されるわけではない．

4.3 SE 機能呼び出し

USON エンジン内で ST 中の SC と語彙情報により，呼び出す機能のオペレーション名や引数名が解決され，引数の値がサービス状態管理部により決定された後，SOAP メッセージの送受信の形で，SE の機能を呼び出す実装を，現在，採用している．

4.4 状況変化イベント

現在の実装では，利用中の SE が利用不可能になったことを通知する SOAP の例外メッセージと，ユーザの位置が変化したことを通知するメッセージを状況変化イベントとして処理している．後者については，位置情報取得機能を USON エンジン内に作りこむことも可能であったが，これはコンテクスト取得機能の 1 つでもあり，新しいコンテクストが把握可能になるたびに，USON エンジンを改造することを避けるために，1 つの SE として実装した．これは 5.2 節のアプリケーションで利用されている．

4.5 インタフェース解決機能

4.5.1 語彙情報と語彙情報ツリー

図 9 に示すように語彙情報は定義され，ツリー構造をとる．各語彙情報は，語彙情報名，ラベル，パラメータリストから構成される．ツリー構造を構成するポインタに，語彙情報名を使用する．ラベルは，その SE が属する SC 名と語彙情報名と親の語彙情報名を “:” で結合したものである．パラメータリストは，その SE が実際に使用する機能呼び出し時の引数名と戻り値と，それぞれが親の語彙中でどう記載されていたかを “()” で記述したものである．また，どの語彙情報も親として持たない場合は，ルートと呼び，ラベル中の語彙情報と親語彙情報には，それぞれ，“default” と “null” として記述する．

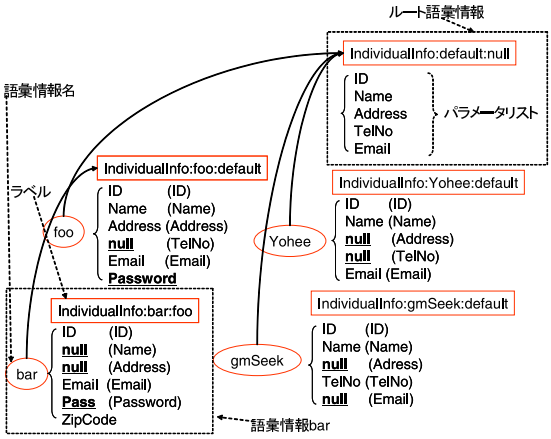

図 9 語彙情報ツリーの例
Fig. 9 An example of interface information tree.

図 9 の左下の例では，語彙情報 bar は，メソッド名 “IndividualInfo” であり，その親は，語彙情報 foo で定義されたものであるため，ラベルは “IndividualInfo:bar:foo” と記載されている．bar で定義されている変数 “ID” は，foo においても “ID” であることや，bar の “Pass” は foo においては “Password” であるといった対応が書いてある．

なお，語彙情報のツリー構造は，特定の管理者が管理するものではない．また，各語彙情報は 1 つ上の親の語彙情報を示すもののみ保持する．

4.5.2 語彙情報ツリーの利用・拡張と default/de facto

各 ST や SE の実装者は，既存のツリー中の語彙情報に基づく（利用ないしは拡張）か，自分で新たなツリーを構築する．前者の場合は，その語彙情報をそのまま利用するか，その語彙情報を拡張（つまり，その語彙情報を “:” を使って指定し，その子供の語彙情報とする）して，語彙を決定する．以下では，どの語彙情報を利用・拡張すべきかという判断基準とそれにより期待できることについて述べる．

同じルート語彙情報 $Intf_{root}$ から 1 回，ないしは，複数回拡張をして得られた異なる語彙情報 $Intf_1$ と $Intf_2$ があるものとする．また，$Intf_i$ $(i = 1, 2)$ を拡張して得られる語彙情報を $Intf_{i,k}$ と記述する．

全体の標準を統括，定義する運営者はいないため，ST や SE の作成者は，どの語彙情報を利用するかは，その作成者の責任である．しかし，各 SE 作成者は，自分が作成した SE はできるだけ利用してほしいという要求があるために，できるだけ多く利用機会があることを願うはずである．SE にとって利用機会が高い

ということは，利用機会が高い ST によく利用されることである．

$Intf_1$ を利用して作成された ST t_1 があり，$Intf_1$，もしくは，$Intf_{1,k}$（k：任意）を利用して作成された SE $e_{1,1}, e_{1,2}, \ldots$ は，その利用において，$Intf_1$ まで遡れば，必ずインタフェースは解決される．$Intf_2$ についても同様である．したがって，$Intf_1$ を利用して作成された ST の使用頻度が高くなってきた場合は，SE 製作者は，$Intf_1$ を利用すれば，自身が作成する SE の利用頻度を高めることができると期待できる．

たとえば，“Web サーチ” という語彙情報があり，それを拡張した “Google” [18] という語彙情報と，“XXX” という語彙情報があった場合，“Google” を利用する ST の利用頻度が高い場合，“Web サーチ” を拡張して SE を作成する人は少なく，多くは，“Google”，もしくは，“Google” を拡張して SE を作成することを期待している．この場合，“Google” はルート（default）ではないが，そこまでたどればほぼ確実に解決できる de facto になる．

これにより，将来，“Google” を拡張した “Google2” の利用頻度が高くなった場合は，“Google2” が新たに，de facto になることもある．このように，特定の団体が利用するものを決めなくても，(ST，SE 実装者の要求にことを発して）自律的に de facto が変更されるシステムが実現できる．

4.5.3 語彙情報の利用例

図 9 の例を用いて，SE 機能呼び出し時に必要な引数の決定について説明する．ST t の中で記述されている SC の語彙情報が，“foo” であり，発見・選択された SE e の語彙情報が “bar” であるとする．SE は SC により発見されるので，ラベル中に記載してある SC は一致する．これは，“foo” と “bar” が同じツリーのどこかに存在していることを保証する．つまり，名前は “foo” と “bar” と異なっているが，同等の機能を提供すると保証する．したがって，インタフェースの解決としては，引数名とその値を解決すればよい．

まず，USON エンジンは両者のラベルを比較する．この例の場合，ラベルが一致していないので，残りのインタフェース解決処理である引数名と値の解決を行う．

SE e の機能の呼び出し前に USON エンジン中には，ST t の “foo” として定義された値が存在する．つまり，図 4 中のサービス状態管理部に（“ID”，“100”）などとして，その値が保存されている．

それぞれの親語彙情報を見比べた場合，“foo” が “bar” の親であるので，SE e の引数として，“ID”，“Email” には，同じ変数名で保存されている値を用い，“Pass” には，“Password” と結び付けられて保存されている値を用いて，引数をセットする．

“ZipCode” については，保存されていないためインタフェース解決機能以外の機能を用いて決定する．この機能については，個別にユーザに問い合わせる方法など，様々な方法があるが，現在のところ，SE 中にあらかじめ用意された値を用いる方法を実装している．

SE 機能呼び出しの結果，戻り値がある場合は，呼び出しの場合の逆の処理を行い，サービス状態管理部に値を保存する．

5. アプリケーション実装例

USON アーキテクチャにより，ユビキタスコンピューティング環境でサービスが提供できるシステムが構築可能であることを示すために，具体的なアプリケーションを実装した．特に，実行時に決定する ST を用いてのサービス合成が可能であることと，実行時に最適な SE を選択し，インタフェースを解決してサービス合成に利用することが可能であることを示すことができるアプリケーションを実装した[19]．

5.1 アプリケーション 1：サービスナビゲーションサービス

本アプリケーション実装により，USON のサービス提供モデルの特徴の 1 つである，サービス連携記述（ワークフロー型である ST）を動的に決定しての，サービス提供というモデルが実現可能であることの確認ができる．ST を動的に決定する点が重要であるため，あらかじめ用意された複数の ST から選択するのではなく，ユーザからの要求によりアプリケーションが生成した ST を用いるものとした．

本アプリケーションは，ユーザがそのときに享受したいサービスの種類（宿泊，食事，観光など）と，それら順序と，予算と，場所（街）を指定すると，可能な組合せを地理的なナビゲーションも含めて提示するものである．

本アプリケーションの動作としては，まず，ユーザは **図 10** に示す GUI により，そのときに享受したいサービスを順序つきで入力する．この入力は USON においては，SC の指定となる．なお，現在の実装では選択できる SE は，Web サービスの形で存在し，かつ，利用する場合に場所が意味があるもの（つまり，ホテルやレストランなど）の予約サービスである．

アプリケーションはその入力を受け ST を生成する．この段階では，利用可能な SE が存在するかどうか，つまり，意味のある ST であるかどうかは確認しない．

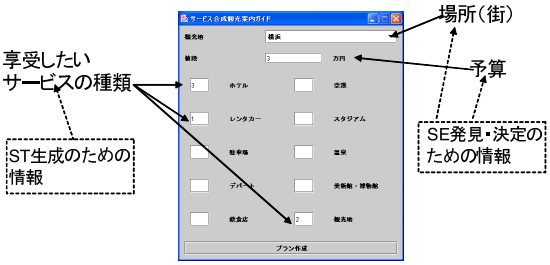

図 10 サービスナビゲーションサービス初期画面
Fig. 10 Start-up image of service-navigation service.

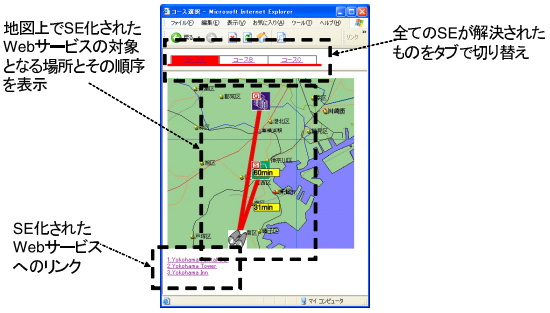

図 11 サービスナビゲーション画面
Fig. 11 Service navigation.

その後，ST 中の各 SC に適した SE をネットワーク中から発見する．この場合に与える条件は，図 10 で指定した金額と場所（街）である．

そして，それぞれの地理的情報を取得するために SE を呼び出す．その得られた地理情報と各 SE の予約ページの URL を**図 11** のように画面に生成する．通常，条件に合う SE は複数あるので，ブラウザ上ではタブとして，その複数の候補を表示している．

なお，図 11 の左下にある，SE 化された Web ページへのリンクをクリックすることで，各々の予約ページに移動でき，そこで，予約を行う．

本アプリケーションを実装することにより，BPEL や WSCI [20] などの既存のサービス連携技術にはない，サービス連携記述を動的に決定してのサービス提供が可能であることを示した．しかしながら，無秩序に ST を生成された場合は，当然のことながら，動作しない ST となることが多く，適切な SE が存在しないために，動作しない ST がある場合が多い．生成も含めた ST の適切な管理機構が必要であることが判明した．

5.2 アプリケーション 2：移動物体追跡サービス

本アプリケーション実装を通じて，USON のサービス提供モデルの特徴の 1 つである，状況が変化した場合に，サービス部品である SE を適切に切り替えて，サービス提供を続けることが可能であることの確認と，USON の機能的特徴である，サービス部品である SE のインタフェースを解決して，呼び出すことが可能であることの確認が行った．

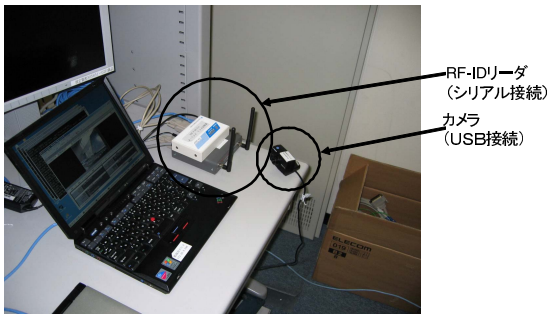

図 12 移動物体追跡サービスにおける「カメラ」
Fig. 12 “Camera” of moving object observation service.

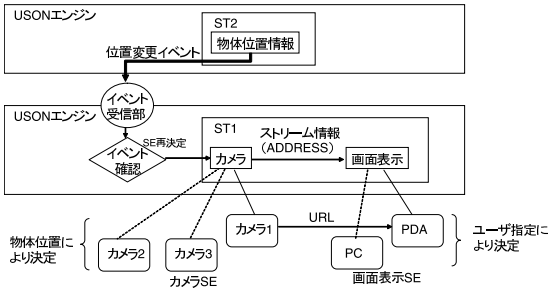

図 13 移動物体追跡サービスの動作
Fig. 13 Moving object observation service.

本アプリケーションは，RFID タグとそのリーダを用い，移動する物体の位置を検出し，その物体に最も近いところにあるカメラで撮影した画像をユーザ端末から見るものである．サービス部品としては，カメラと動画像を表示する機能を用いる．カメラとしては，異なるインタフェースを持つ，ネットワークに接続されたカメラとして，PC に USB 接続されたカメラを複数用いた．なお，これらを SE とした場合のインタフェースはすべて異なるものを用意した．**図 12** に PC に接続されたカメラの状況を示す．各 PC は RFID リーダとカメラを接続している．

また，動画像表示機能としては，Windows Media-aPlayer を PDA/PC 上で動作させるものとした．これを画面表示 SE として，システム中に組み込んだ．なお，起動時に画面表示 SE を選択することにより，PDA/PC の切替えを行う．

図 13 に，本アプリケーションの構成を，2 つの ST を中心に図示した．なお，ST1 と ST2 を実行している USON エンジンは同一でもかまわない．

ユーザがユーザの持つ端末から，アプリケーションを起動するときに，追跡したい物体の識別子を入力する．その物体の位置を常時追跡するのが ST2 である．

```
カメラ1のSE記述          (PetCameraSE_30S.xml)
<?xml version="1.0" encoding="UTF-8" ?>
<USON xmlns:uson="http://uson.org">
  <serviceType>Service Element</serviceType>
  <serviceName>Pet Camera SE USB-CAM30S</serviceName>
  <serviceCategory>Camera</serviceCategory>                語彙情報
  <interface>
    <vocabulary>
      <import location="http://localhost:8080/axis/xml/vocaPetCameraSE_30S.xml" />
    </vocabulary>
    <wsdl:definitions targetNamespace="http://localhost:8080/axis/services/CameraSE30S"
                          <省略>
      <wsdl:portType name="CameraSE30S">
        <wsdl:operation name="getCameraURL_30S" parameterOrder="reqXml">
          <wsdl:input message="impl:getCameraURL_30SRequest"
                      name="getCameraURL_30SRequest" />
          <wsdl:output message="impl:getCameraURL_30SResponse"
                      name="getCameraURL_30SResponse" />
                          <省略>
  </interface>
</USON>

カメラ1の語彙情報        (vocaPetCameraSE_30S.xml)
<?xml version="1.0" encoding="UTF-8" ?>
<USON xmlns:uson="http://uson.org" xmlns:stns="http://localhost:8080/st/">
  <vocabulary name="stns:Camera:30S"
            extends="http://HomeServer:8080/vocaPetCameraRegular.xml">
    <word name="tgtCam_30S" extend="TARGET" />
    <word name="status_30S" extend="STATUS" />                親語彙情報
    <word name="cameraURL_30S" extend="ADDRESS" />           変数名の対応
  </vocabulary>
</USON>
```

図 14 カメラ SE の SE 記述と語彙情報
Fig. 14 SE description and interface information of "Camera" SE.

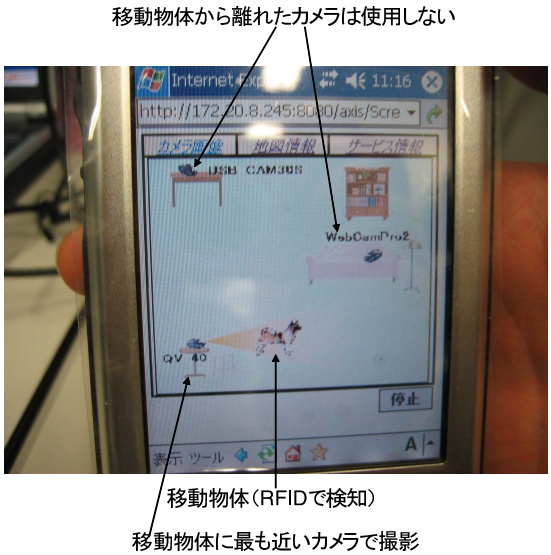

図 15 移動物体追跡サービスのシステム状態確認画面
Fig. 15 Screen image of moving object observation service.

その物体の位置が変更になった場合，ST1 を実行している USON エンジンにイベントの形で通知する．ST1 を実行している USON エンジンは，通知された物体の位置情報に基づき最適なカメラ SE を選択する．選択されたカメラ SE は，ユーザ指定の画面表示 SE に，ストリーム情報として，カメラが撮影している画像の URL を送信する．画面表示 SE は，その受け取った URL を表示する．

画面表示 SE が撮影画像を表示している間は，USON エンジンは，そのストリーム配信に何の関係もせずに，カメラと画面表示 SE の間で直接にストリーム配信される．

図 14 は，カメラ SE の SE 記述と語彙情報である．SE 記述は図 6 と同じものを示しているが，使用する語彙情報が`import` されており，語彙情報で，たとえば，親語彙（`vocaPetCameraRegular.xml`）で，`ADDRESS` と記述してある変数が，`cameraURL_30S` に対応する．

図 15 は，システムの状態を確認している画面である．最も近いカメラが反応している様子を本画面で確認することができる．

本アプリケーションを実装することにより，カメラ側の状況，つまり，サービス部品コンテクストが変化したことによるイベントにより，別のサービス部品に，インタフェースを解決した後に，切り替えることが可能であることが確認できた．本実装における，イベント検出から別のカメラへの切替えは，1 秒以内に行えるため，図 15 で示すような動作確認画面上は，まったくユーザにストレスを与えることがない．

しかしながら，カメラで撮影したストリームデータを再生する機能において，バッファリングの時間があるため，画像が瞬時に切り替わることは実現できなかった．ユビキタスコンピューティング環境においては，このようなストリームデータの取扱いも含めたサービス提供技術[21] が必要であることが判明した．

なお，今回は，RFID の電波を非常に弱く設定することで，RFID タグを認識できた PC に接続しているカメラが物体に最も近いカメラであるとした．カメラを接続してある PC は，電波を検出すると，ユーザ端末側にイベントを送っている．本来は，電波の強さから，位置を把握する機能を実装すべきであったが，物理的な位置把握を中心においた研究成果（文献 22) など）を利用する予定である．

5.3 実装からの知見

本節では，USON エンジンを用いたアプリケーションの実装により得られた，USON フレームワークの有用性と限界についての知見を述べる．

5.3.1 ST によるサービス合成

本実装により，ユーザが受けたいサービスを ST として確定できれば，適切な SE を USON エンジンが決定し，サービスとして提供できることは示した．「いつも家で使っているサービスを，会社でも」といったような，同じようなサービスを，異なる場面で享受したいというユーザの要求には応えることができる目処が立った．

しかしながら，ユビキタスコンピューティング環境

で，様々なサービス部品が利用可能となっている場面で，ユーザが具体的にサービス部品を指定して，それらを合成して，サービスとして利用したいというユーザの要求に応えるためには，USON フレームワークは不要な処理が多い．つまり，ユーザが具体的にサービス部品を指定しているため，USON エンジンにおける，与えられた ST から SE を決定する処理はすべて不要である．

また，USON フレームワークは，コンテクストに対応したサービス提供を目標としているため，あるユビキタスコンピューティング環境に存在する全ユーザに対して均等なサービスの実現にも不要な処理が多い．このようなサービスとしては，認証サービスなどがある．

これら USON フレームワークの中で定義されている処理の多くが不要となるようなサービスが，どの程度実際のユビキタスコンピューティング環境に必要とされるかは，今後，ユビキタスコンピューティング環境が整備されていく過程で明らかになる．それをふまえて，USON フレームワークの中から不要とされる処理を削除することも将来検討する可能性もある．

5.3.2 SE の実装

本実装では，SE は，その SE 記述をアドバタイズメントとして公開すること（実装としては，JXTA ピアであること）と機能呼び出しのためのメッセージを受信すること（実装としては，SOAP メッセージを受信すること）を求めている．この条件により，SE 化できるサービス部品が限定されることは，解決すべき課題である．

この課題の解決の方針としては，メッセージの受信の方法の変更も視野に入れた実装の軽量化と，JXME[23] のように，システムの機能の一部をネットワーク上の他のコンピュータ（Proxy）に再配置を検討する．

6. ST 生成とその ST によるサービス創発

2 章で述べたように，USON アーキテクチャによるシステムでは，状況に適したサービスを合成し，ユーザに対してサービス提供を行う．さらに，状況に適したサービスを生成するために，新しい ST を生成し，流通させる．このサービス創発の現在の研究状況について，本章で述べる．

この ST 生成アルゴリズムについては，GA（Genetic Algorithm）の応用であり，特定の SE に注目しての履歴を用いるアルゴリズム[12] が有力である．**図 16** に，その 1 形態を示す．図 16 は，利用頻度の高い SE（SE-b1）について，その SE を利用すること

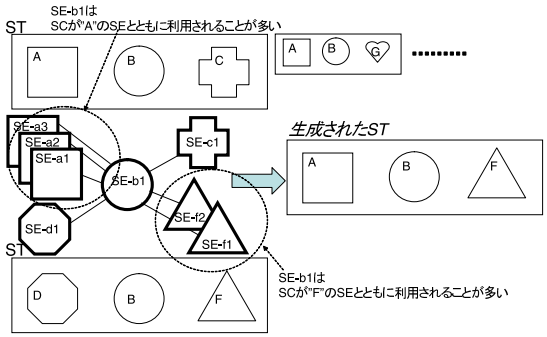

図 16 ST 生成アルゴリズムの例
Fig. 16 An example of ST-creation algorithm.

となった ST 群（A-B-C や D-B-F）から，よく利用されるものをその SE を基点に切断し，他のものと再接続する（A-B-F）というアルゴリズムを模式化したものである．この利用頻度をどのように解釈するか，すなわち，最大のものを採用するか，利用頻度率により傾斜を持たせたものとするかなどは，ST の運用に大きく依存するため，現在アルゴリズムから検討中である[12]．

この方法では，新たに生成された ST についても，その中に含まれる SC について必ず該当する SE が存在することが保証される．しかしながら，その ST がどの程度有用であるかは，利用する状況に依存する．現在，生成された ST の有用度がどのように変化するかをシミュレーションにより検討中である[12]．

サービス創発は，その提唱の 1 つである Ja-Net プロジェクト[24] のサービス創発と同じく，システム設計者（USON の場合は，SE 実装者と ST 実装者）が，事前に想定することのできなかったサービスを提供することを目指している．

7. 関 連 研 究

サービス部品として WSDL 記述で表現された状態のない Web サービスを用い，サービス連携記述としてワークフロー型の BPEL 記述を用いることは，実ビジネスの世界で発展している．また，サービス部品として状態のある Grid サービス[25] も共通のフレームワークで利用するための技術である WS-RF[26] も検討が進んでいる．これらのサービス連携技術においては，利用されるサービス部品に関する情報のほとんど☆を規定したワークフローを用いる．たとえば，BPEL 記述では，利用可能な Web サービスのインタフェースが WSDL で明示的に記述される．しかしながら，ユ

☆ Web サービスの URI は記述されない．

ビキタスコンピューティング環境でのサービス提供におけるサービス部品は，その場その時にならないと，利用可能性も，どのようなサービス部品がその状況に対して最適であるかも，決定できない．したがって，BPEL のみを利用したサービス連携では，サービス部品のインタフェースも含めての動的な決定機能がないため，そのままではユビキタスコンピューティング環境に適用できない．

Semantic Web[9] や Semantic Web サービス[27] などをはじめとするサービス連携技術は，各サービス要素に対して意味的な（Semantic）情報を付与し，発見・合成を自動で行う発想に基づいている．OWL-S[28] はそのための，サービス部品の記述方法の規定である．これらの技術は，上位のサービス設計図から下位のサービス部品の指定を Semantic 情報により指定するためのものであり，動的に切り替えることは，目的の対象外であり，Semantic 情報に指定されたサービス部品がさらにサービス設計図である場合も想定していない．また，これらの技術では，人はなるべく介在せずに，自動処理を行うことが発想の根本にあり，実装には AI 分野の基礎研究がこれからも多く必要である．したがって，そのままではユビキタスコンピューティング環境におけるサービス提供には適用できない．一方，USON では，語彙情報記述のように，実装者が他の語彙情報を見て，実装者が関連付けを記述するような発想を採用しているため，現在の AI 分野の技術でも実現可能である．しかし，USON アーキテクチャによる SE や ST を普及させるためには，W3C など準化団体で検討されているこれらの表現方式を採用することは重要であるため，できるだけ採用していく予定である．

USON のサービス連携記述である ST は，BPEL 連携技術の BPEL 記述と同じく，ワークフロー型を採用している．WSCI などのコレオグラフィー型は，サービス部品中に，他のサービス部品の動作に対応する動作を記述する必要がある．これは，サービス部品設計者が，想定した以外の目的での利用ができないことを意味しており，その場その時に利用可能なサービス部品をユーザの状況に合わせて利用すべき，ユビキタスコンピューティング環境でのサービス提供に提供することは困難がともなう．しかし，そのコレオグラフィー型自律分散型システムは，ユビキタスコンピューティング環境のサービス提供とも親和性は高いため，コレオグラフィー型の連携記述を自動生成する技術の研究成果を取り込んでいく予定である．

8. おわりに

本稿では，ユビキタスコンピューティング環境に適したサービス提供技術 USON におけるサービス合成技術について，それを実現するアーキテクチャとその実装，および，それによるアプリケーションによるそれらの有用性について述べた．USON においては，ユビキタスコンピューティング環境の特徴であるサービス部品がその場その時にならないと決定できない特徴に対応するため，サービスのメタ設計図を発見・選択し，それに適するサービス部品を発見・選択し，そのサービス部品のインタフェースを解決しての，サービス部品の呼び出しによるサービス提供を行う．

また，複数のアプリケーションを USON により実現することにより，提案モデルの正当性を実証した．現在の実行環境は主に PC を中心とした環境であるため，PDA や携帯電話などやセンサやアクチュエータも含めた実装を行う予定である．

今後は，ユビキタス環境でも有力なサービスと考えられる動画像などのストリーム情報を取り扱えるようにモデルを拡張し，その実装により検証を行う予定である．

また，状況に適したサービス部品やサービスメタ設計図の発見を，DHT[29)~31)] を用いて行う方式も検討中[32] である．また，サービス創発アルゴリズムに関しては，フィールド実験を通じて，適切なアルゴリズムを開発していく予定である．

謝辞 本研究の一部は，平成 15 年度総務省「ユビキタスネットワーク認証・エージェント技術の研究開発」の研究助成によるものである．

参考文献

1) Weiser, M.: The Computer for the 21st Century, *Scientific American*, pp.94–104 (1991).
2) 国内線.com. http://www.kokunaisen.com
3) 旅の窓口．http://www.mytrip.net
4) アルキカタ・ドット・コム．http://www.arukikata.com
5) Andrew, T., et al.: Business Process Execution Language for Web Services Ver 1.1 (2002). http://www-106.ibm.com/developerworks/webservices/library/ws-bpel.
6) Takemoto, M., et al.: The Ubiquitous Service-Oriented Network (USON) — An Approach for a Ubiquitous World Based on P2P Technology, *Proc. 2nd International Conference on Peer-to-Peer Computing*, pp.17–21 (2002).
7) Takemoto, M., et al.: Service Elements and

Service Templates for Adaptive Service Composition in a Ubiquitous Computing Environment, *Proc. Asia Pacific Conferencce on Communications* (*APCC*), pp.335–338 (2003).
8) Takemoto, M., et al.: A Service-Composition and Service-Emergence Framework for Ubiquitous-Computing Environments, *Proc. SAINT Service Oriented Computing Workshop*, pp.313–318 (2004).
9) Berners-Lee, T., et al.: The Semantic Web, *Scientific American* (2001).
10) 川村　亨ほか：Awareness サービス実現に向けた通信距離概念の具体化，電子情報通信学会 IN 研究会 (96-32) (1996).
11) OASIS UDDI. http://www.uddi.org
12) 山登庸次ほか：ユビキタス環境に適したサービス連携フレームワークにおけるサービステンプレート生成手法，電子情報通信学会 IN 研究会 (Jan. 2004).
13) Apache <Web Services /> Project. http://ws.apache.org/axis
14) The Apache Jakarta Project. http://jakarta.apache.org/tomcat
15) Project JXTA. http://www.jxta.org
16) W3C Note: Web Services Description Language (WSDL) 1.1 (2001). http://www.w3.org/TR/wsdl
17) 南　正輝ほか：動的でアドホックなネットワークサービスフレームワークの検討，*DICOMO2000* (2003).
18) Google. http://www.google.com
19) Takemoto, M., et al.: A Service-Coordination Framework for Application to Ubiquitous and Context-aware Computing Environments, 情報処理学会全国大会，pp.3–335–3–336 (2004).
20) W3C Note: Web Service Choreography Interface (WSCI) 1.0 (2002). http://www.w3.org/TR/wsci/
21) 岩田哲弥ほか：環境適応型サービス連携フレームワークにおけるストリーム指向アプローチの提案，情報処理学会全国大会，pp.3–341–3–342 (2004).
22) Balakrishnan, H., et al.: Scalable Location Aware Monitoring. http://nms.lcs.mit.edu/projects/slam
23) jxme: JXTA Platform Project. http://jxme.jxta.org.
24) Itao, T., et al.: Service Emergence Based on Relationship among Self-Organizing Entities, *Proc. Symposium on Applications and the Internet* (*SAINT*), pp.194–203 (2002).
25) Foster, I., et al.: Grid Services for Distributed System Integration, *IEEE Computer*, pp.37–46 (2002).
26) Czajkowski, K., et al.: The WS-Resource Framework (Ver. 1.0) (2004). http://www-106.ibm.com/developerworks/library/ws-resource/ws-wsrf.pdf
27) McIlraith, S.A., et al.: Semantic Web Services, *IEEE Intelligent Systems*, pp.46–53 (2001).
28) The OWL Services Coalition: OWL-S: Semantic Markup for Web Services. http://www.daml.org/services/owl-s/1.0/owl-s.html
29) Stoica, I., et al.: Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications, *Proc. ACM SIGCOMM* (2001).
30) Ratnasamy, S., et al.: A Scalable Content Addressable Network, *Proc. ACM SIGCOMM* (2001).
31) Project JXTA: JSRDI: JXTA Shared Resource Distributed Index Design Plan (2002). http://platform.jxta.org/java/workinprogress/srdi.html
32) 山登庸次ほか：ユビキタス環境におけるサービスコンポーネントの P2P 的検索手法，電子情報通信学会総合大会，B-6-197 (2004).

(平成 16 年 5 月 24 日受付)
(平成 16 年 12 月 1 日採録)

武本　充治（正会員）

1992 年東京大学理学部情報科学科卒業．1994 年東京大学大学院理学系研究科情報科学専攻修士課程修了．同年日本電信電話株式会社入社．以来，NTT ネットワークサービスシステム研究所，NTT 未来ねっと研究所等において，分散コンピューティングシステムの研究に従事．1999 年〜2000 年 Massachusetts Institute of Technology, Laboratory for Computer Science において Visiting Scientist として研究に従事．2003 年より，早稲田大学大学院情報生産システム研究科博士後期課程に在学中．1998 年電子情報通信学会学術奨励賞受賞．IEEE，電子情報通信学会各会員．

大石　哲矢

1993 年早稲田大学理工学部電子通信学科卒業．1995 年早稲田大学大学院理工学研究科修士課程電気工学専攻修了．同年日本電信電話株式会社入社．交換ソフトウェア，P2P，ユビキタスコンピューティング研究に従事．電子情報通信学会会員．

岩田　哲弥（正会員）

1994 年東京大学理学部卒業．1996 年東京大学大学院理学系研究科修士課程修了．同年日本電信電話株式会社入社．通信システムの基盤技術，大規模ソフトウェア開発技術，P2P 技術，ユビキタスコンピューティング研究等に従事．

山登　庸次（正会員）

2000 年東京大学理学部卒業．2002 年東京大学大学院理学系研究科修士課程修了．同年日本電信電話株式会社入社．P2P，ユビキタスコンピューティング研究に従事．電子情報通信学会会員．

田中　洋平（正会員）

2000 年八戸工業高等専門学校電気工学科卒業．同年東日本電信電話株式会社入社．同青森支店を経て，2003 年 4 月より，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に勤務．現在，P2P 技術とオントロジの研究に従事．

徳元　誠一

1987 年千葉工業大学工学部工業化学科卒業．同年日本電信電話株式会社入社．以来，通信設備，およびその運用系システムの研究開発，ユビキタスネットワークの実現技術の研究開発に従事．現在，NTT ネットワークサービスシステム研究所に勤務．

島本　憲夫

1987 年東京大学工学部卒業．1989 年東京大学大学院工学系研究科修士課程修了．同年日本電信電話株式会社入社．以来，交換機ソフトウェア，ユビキタスコンピューティング研究に従事．電子情報通信学会員．

黒川　　章

1982 年東京大学工学部卒業．1984 年東京大学大学院工学系研究科修士課程修了．同年日本電信電話公社（現 NTT）入社．以来，ISDN ディジタル交換機の研究開発，ISDN ユーザ網インタフェースの標準化，P2P，ユビキタスネットワーク等の研究に従事．現在は，次世代ネットワークの研究に従事．1991 年 3 月から翌年 2 月までモントリオール大学客員研究員．2002 年 TTC 表彰受賞．IEEE，電子情報通信学会各会員．

須永　　宏

1981 年東京工業大学工学部制御工学科卒業．1983 年東京工業大学大学院工学系研究科制御工学専攻修士課程修了．同年日本電信電話公社（現 NTT）入社．以来，情報通信ソフトウェア，次世代ネットワーク構成法，P2P，ユビキタスの研究に従事．ITU-T SG7 のラポータを務め，TMN，OSI の標準化推進に寄与．博士（情報科学，平成 14 年東北大学）．電気通信財団テレコム技術賞，日本 ITU 協会賞受賞．IEEE，電子情報通信学会各会員．

小柳　恵一（正会員）

1975 年慶應義塾大学工学部電気学科卒業．1977 年慶應義塾大学大学院修士課程修了．1998 年大阪大学大学院博士課程修了．1977 年日本電信電話公社（現 NTT）入社．工学博士．現在，早稲田大学大学院情報生産システム研究科教授．著書『C++オブジェクト指向設計』,『P2P インターネットの新世紀』等．電子情報通信学会会員，ACM 会員，IEEE Senior Member．