特集

# 状況に応じたサービスを提供するTask Computing

## Task Computing – Filling the Gap between Tasks and Services


### あらまし

ユーザのやりたいこととしての「タスク」と実行可能な機能としての「サービス」の間のギャップを埋める技術であるTask Computingを紹介する。

ユーザのコンピューティング環境の変化を三つの段階，パソコン環境，モバイル環境，ユビキタス環境としてとらえる。到来しつつあるユビキタス環境においては，「今，ここで」の機能を「サービス」として動的に発見し，ほかのサービスとともにユーザの「タスク」に構成することが最重要であり，それを可能とする技術がTask Computingである。

本稿では，Task Computingのアーキテクチャ，その技術要素であるサービス発見，合成，実行，生成，管理，およびユビキタス環境でのシナリオのTask Computingによる実現などについて解説する。

### Abstract

Task Computing fills the gap between the tasks to be done by the user and the services that offer feasible functions to the user. There are three stages in users' computing environments: PC, mobile, and ubiquitous. In the soon-to-be mainstream ubiquitous environment, it will be essential to dynamically find services in the environment and construct them into user tasks along with other services on the users' devices and remote environments. Task Computing based on Semantic Web technologies makes this possible. This paper describes the architecture of Task Computing and how to find, compose, execute, generate, and manage services that represent the functional resources of Task Computing. This paper also describes how Task Computing is used to implement service-task schemes in a ubiquitous environment.



**益岡竜介（ますおか　りゅうすけ）**

米国富士通研究所カレッジパーク研究所 所属
現在，Task Computingの研究開発に従事。



**湯原雅信（ゆはら　まさのぶ）**

ITコア研究所 所属
現在，ユビキタスミドルウェアの研究開発に従事。


 FUJITSU.55, 4, p.376-383 (07,2004)

## まえがき

Task Computing(1)-(3)とは，ユビキタス環境においてユーザのやりたいこととしてのタスクと，実際に実行可能な機能であるサービスとの間のギャップを埋める技術である。Task Computingは米国富士通研究所（FLA）カレッジパーク研究所と，セマンティックWeb(4)の研究で有名なメリーランド大学のMINDSwapグループ(5)との共同研究に始まり，現在では富士通研究所も参画するプロジェクト名でもある。

まずはユーザのコンピューティング環境の変化を**表-1**にあるように三つの段階に分けていくつかの面から概観したい。

まずパーソナルコンピューティングを実現したパソコン（以下，PC）環境がある。これにより，それまでのコンピュータは巨大で，特別な部屋に鎮座しているという概念を変え，より一歩ユーザに近づき，「一人に1台」という時代を実現した。作業には，個人やグループ用に設置されたPC上のOSやアプリケーションなどにより与えられる機能やサーバ上に用意された機能を利用することになる。ユーザは比較的大きな画面を前に，キーボードやマウスなどで，資料作成などのかなりの集中を必要とする作業を中心に行っていた。

表-1　ユーザのコンピューティング環境の変化

| | 標語 | 場所の制約 | ユーザインタフェース | 主な作業の種類 | 利用する機能 |
|---|---|---|---|---|---|
| PC環境 | 一人に1台 | 強い | 大画面<br>キーボード<br>マウス | 資料作成，コミュニケーション | PC上，サーバ上 |
| モバイル環境 | いつでも，どこでも | 弱い | 小画面<br>ペン<br>音声 | コミュニケーション | サーバ上，ユーザ端末上 |
| ユビキタス環境 | 今，ここで | 弱い | 同上 | デバイスコントロール | 環境内，ユーザ端末上 |

つぎにモバイル環境の時代がやってきた。小型のコンピュータを持ち歩き，ネットワークを経由してサーバに接続することにより，「いつでも，どこでも」同じ環境を使えるようになった。

モバイル環境でのユーザ端末のCPUパワー，メモリ，通信帯域などは次第にPC環境に追いつくであろうが，ユーザ端末の小型化のため，画面の絶対的なサイズは小さく，入力手段もペンかダイヤルキー程度であり，長い文書の入力は難しい。音声などの入出力手段も大きな位置を占めるであろう。PC環境に比べて利用する機能の比重はサーバに移るという違いはあるが，基本的にモバイル環境では場所による制約をなくし，「いつでも，どこでも」PC環境でできたことをできるようにすることを目指してきたと言える。

では到来しつつあるユビキタス環境とはどのようなものであろうか。そこではモバイル環境の「いつでも，どこでも」を前提としながらも，一歩進んで「今，ここに」いるからこそしたいことをできるようにすることと考えている。「ユーザが現在いる場所」という意味での「現場」をユーザの視点からとらえた「今，ここで」が重要になる。**図-1**のように自分の持ち物がリモート環境とだけではなく，周辺

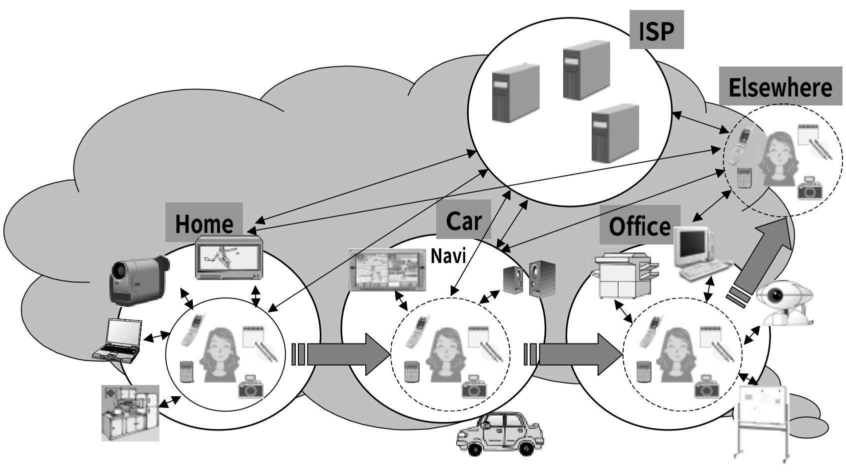


図-1　ユビキタス環境
Fig.1-Ubiquitous environments.

環境と互いに連携し，「今，ここで」の環境にダイナミックに適応していく。「今，ここで」使える機能を動的に見つけ，使う。作業の内容もPC環境やモバイル環境と質的に異なり，デバイスのコントロールなどに重点が移っていく。

ユビキタス環境で使えるインタフェースはモバイル環境と同様にかなり制限されたものである。しかし，作業の質や，それらの作業に利用する機能の数や，どこにある機能を使うかが変わることにより，ユーザインタフェースやコンピューティングモデルに，より本質的な変化をもたらし，それまでのものとは一線を画すものになると考えている。ユビキタス環境では，その名のとおりコンピューティング能力を持った様々な機器が至る所に存在する。従来の環境よりはるかに多い，しかも多くの場合事前には知らなかった機器やサービスを，そのままユーザに見せても，一般ユーザの手に余る。したがって，ユーザの状況（コンテキスト）に合わせて情報を絞り，ユーザが行いたいであろう「タスク」を推測してシステムの方から提示する必要がある。ユーザの作業は，その上でどのデバイスを操作対象とし，どう組み合わせ，実行し，操作するといった粒度の大きな指示が中心となってくる。このためPC環境やモバイル環境でのコンピューティングモデルをそのままユビキタス環境に持ち込んだのでは，うまくいかない。すなわちユビキタス環境に合った新しいコンピューティングモデルが必要であり，Task Computingこそがそれに応えるものであると著者らは考えている。

本稿では，まずユビキタス環境でのIT技術の要件を検討する。つぎにTask Computingのアーキテクチャを紹介し，システムがいかに動くかを，サービス発見，合成，実行の流れと，動的なサービスの生成，管理に分けて説明し，次章の最初で挙げるシナリオのいくつかがどのように実現されるかを最後に説明する。

## ユビキタス環境でのIT技術の要件

例えば著者らはユビキタス環境で自在に「今，ここで」のデバイスや機能を使って，以下のようなことができるようになるべきと考えている。

(1) 自宅で，DVDプレイヤ，TV，DVカメラ，ホームセキュリティカメラの映像を，リビングやキッチンのモニタに，あるいは2階の書斎のPCに，自在に表示させる。
(2) 自宅で，訪れている友人のコンピュータから提供されるゴルフ場の連絡先の情報と，インターネット上の天気予報の情報サービスを使い，ゴルフ場の天気予報を目の前のモニタに表示する。
(3) 車中で，自分が持ち込んだPDAにある住所や家のPCにある住所をカーナビゲーションシステムに与えて経路を表示させたり，自分のデジタルオーディオプレーヤにある音楽を車のカーステレオで再生したりする。
(4) 初めて訪れた会議室にあるプロジェクタに自分のマシン上の発表資料を表示し，その表示をコントロールする（もちろんビデオケーブルなどつながずに）。
(5) 初めて訪れた会議室にある電話で，自分のコンピュータ上のPIM（Personal Information Manager）にある電話番号を使い，電話をかける。
(6) 打合せで，あるプロジェクトのことが話題になり，早速，SE用にプロジェクト情報を提供するサービスからプロジェクト情報を取得し，そのプロジェクトのホームページを会議室のプロジェクタに表示し，さらにそのプロジェクトのマネージャと会議室の電話で電話会議を行う。後で使うために，この「プロジェクト」情報を自分のPC上に取っておく。

ユビキタスコンピューティング関連の研究では，一見よくありそうなシナリオである。しかし既存の研究では，システムをそのデモ専用に作り込んでおり，あらかじめ想定した使い方しかできないことが多い。そのような作りでは，新しい機能を持った機器を追加しようとするたびに，システムを作り直さなければならず，変化の激しいユビキタス環境には対応できない。では，ユビキタス環境での真のIT技術に求められる要件はどのようなものがあるか？著者らは以下のように考えている。

(1) その環境で使えるデバイスなどの機能の動的発見。すなわちユーザのクライアントデバイスが「今，ここで」の環境を認識する。
(2) ユーザ自身も「今，ここで」の環境の一部として，あるいは環境に働きかけ，環境を変える。

(3) それらの発見された機能とクライアントデバイス上の機能，リモート環境の機能を合わせ，それらの機能をユーザのやりたいこととしてのタスクとして再構成して，その現場，「今，ここで」を実行できる。

（1）は明確であろう。「今，ここで」何が使えるのかが分からなければ，始まらない。（2）は，ユーザが環境を媒体としてほかのユーザとやり取りすることを可能にする。（3）における「タスクとしての再構成」は，必ずしもシステムがすべて自動的に行うことは考えていない。むしろユーザが積極的に関与し，システムはその作業をサポートする形であるべきと考えている。ユビキタス環境での制限されたインタフェースを考えると，ユーザにいちいち細かい指示をさせるのではなく，ユーザには簡単な選択だけを行ってもらうようなインタフェースが求められると考えている。

## Task Computingのアーキテクチャ

前章の要件を実現するために必要なのは，徹底した機能の仮想化とできる限りの相互運用性（Interoperability）の実現である。「機能の仮想化」とは，それぞれの環境の機能をその実装と分離し，クライアント側で仮想的に機能を自在に扱えるようにすることである。「最大限の相互運用性」とは，標準など事前に必要となる了解事項を最小限に抑えつつ，できる限りいろいろな機能を使えるようにすることである。それらによって初めて，ユーザは現場での機能を使って自在にタスクを実行することができる。

Task Computingは，参考文献（6）や（7）などの論文にあるような，Web，エージェント，セマンティックWeb，Webサービス,[8] ユビキタスコンピューティングの研究を経て，ユビキタス環境での機能の仮想化と相互運用性の実現を突き詰めていったことに対する，著者らの一つの答えである。

機能の仮想化では相互運用性を高めるために共通の仮想化が必要である。Task Computingではすべての機能を「サービス」としてとらえ，そのサービスの記述自体を，そのサービス自身とほぼ同等に扱えるようにした。ここで「サービス」とは，ローカルにあるいはネットワークを通じて標準を使ってプログラムから実行可能な機能であり，Task Computingで主な対象としているWebサービス[7]でいう「サービス」とほぼ同じである。また相互運用性を実現するために必然的にTask Computingの中では「セマンティクス（Semantics）」が重要な役割を果たしている。「セマンティクス」とは情報あるいは機能のより高い相互運用性を実現するためのものである。すなわち情報ならより多くのシステムで使えるようにする。機能ならより多くのものとつなげられるようにするために使われるものである。ユビキタス環境での機能の仮想化と最大限の相互運用性の実現のため，Task Computingでは以下の二つの重要なアーキテクチャ上の決定,

(1) すべての機能をサービスとする

(2) セマンティックなサービス記述とサービス実装を完全に分離する

を行い，Task Computingを実現するためのアーキテクチャを**図**-2のようにした。以下に簡単にそれぞれの要素を記述する。

(1) ユーザ

Task Computingでは，いつもユーザが中心にいる。その意味は二つである。一つはユーザをサポートするためのシステムであり，何を行うかについての最終決定権はいつもユーザにある。もう一つは，難しければ無理に自動化する必要はなく，ユーザにシステムが助けてもらえばよい。

(2) HCI（Human-Computer Interface）レイヤ

いろいろなサービスをユーザに対してタスクという形に仮想化し，ユーザがタスクとやり取りすることを可能にする。

(3) TC機能レイヤ

発見，合成，実行，保存，および生成，消去，管理などの機能を実現するレイヤである。それらの機能はWebサービスによってアクセスすることが可能である。

(4) セマンティックサービスレイヤ

セマンティックなサービスとして仮想化されたサービスのレイヤである。それぞれのサービスのセマンティックなサービス記述がこのレイヤの要素である。

(5) サービス実現レイヤ

OS，アプリケーション，デバイス，Webサービス，Gridなど実際の機能をインプリメントしているレイヤである。

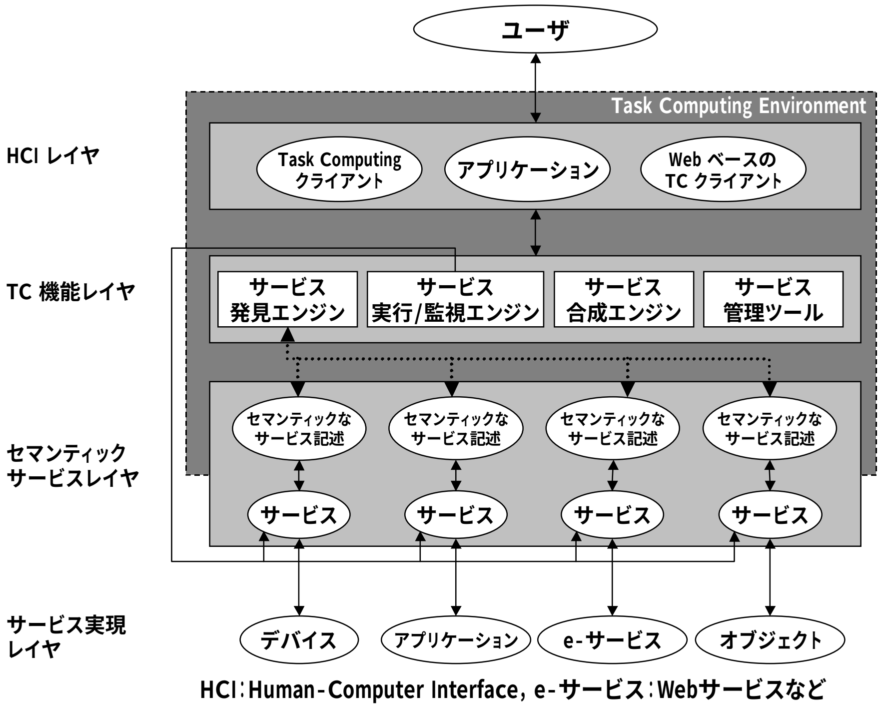


図-2　Task Computingのレイヤアーキテクチャ
Fig.2-Layered architecture of Task Computing environment.

前に述べたように，このアーキテクチャでは，OS，アプリケーション，デバイス，Webサービスなどが提供するすべての機能を「サービス」としている。この点は徹底していて，例えばシステム中に直接現れる「オブジェクト」はなく，「オブジェクト」が必要な部分には，「オブジェクト提供サービス」が使われている。もちろんすべてを「サービス」とすることによるオーバヘッドはあるが，すべてが「サービス」であることによりシステムや処理の整合性が非常に高くなり，そのオーバヘッドを上回るメリットがあった。

アーキテクチャ上の決定のもう一つ「セマンティックなサービス記述とサービス実装の完全な分離」とは，ユーザがTask Computingのクライアントを通じて，セマンティックなサービス記述だけでサービスに関する各種操作を行うことを可能にすることである。ここで「セマンティックなサービス記述」とは，具体的にはセマンティックWeb技術の標準の一つであるOWL-S[9]（Web Ontology Language for Services）によるサービスに関する説明である。その内容はサービスの名前や機能の説明，入出力と，内部プロセスの記述，さらにサービスの実装に関する情報（具体的な実装にどのように対応するのか）などである。

このセマンティックなサービス記述により「サービスの仮想化」が実現される。今までOS，アプリケーション，デバイス，Webサービスなどにとらわれていた機能がこのセマンティックなサービス記述によって，解放される。ユーザのクライアントがサービス発見やそのほかの手段で，サービスの「セマンティックなサービス記述」を手に入れさえすれば，ユーザはその「サービス」を自分の手元で自在に扱うことができる。これこそがTask Computingに大きな柔軟性と拡張性を与えたものと考えている。この「セマンティックなサービス記述」（以下，サービス記述）を扱い，上記の分離によるサービスの仮想化を実現する技術がTask Computingのコアになっている。サービス記述だけで，サービスの実行以外のサービスに関する各種の操作（発見，フィルタ，合成，保存，生成，管理など）を可能としている。そしてユーザが（合成した）サービスの実行を指示すると，そのサービス記述を使って，ユーザの介在なくサービス実行を行う。（もちろんユーザ入力が必要なサービスではそのための画面を表示し，ユーザ入力を受け付ける。）

またTask Computingで必要とする事前の了解事

項はユビキタス環境でサービス記述を見つけるためのプロトコルとその記述方法だけである。これだけの了解事項だけで，Task Computingにより，事前のプログラムなどなく，その場にあるデバイスなどの機能を動的に発見し，エンドユーザは数回のマウスやペン操作だけで，それらをユーザ端末上のアプリケーションやリモート環境のWebサービスと，これまた動的に組み合わせ，実行できる。さらには必要に応じて，いろいろな手段でサービスを生成したり，それらサービスの状態を管理し，自分がいる環境を変えたりすることができる。

## サービス発見，合成，実行

この章ではTask Computingにおけるサービス発見，合成，実行に関して説明する。

サービス発見の実現方法もいろいろあり得るが，サービス発見をセマンティックレイヤに抽象化すると，セマンティックなサービス記述を発見することになる。この意味で任意のサービス発見を実装することができるが，現在はプライベートサービス発見とユビキタスサービス発見の二つが実装されている。プライベートサービスとはユーザ端末上で走っているサービスであり，それらのサービスを発見するために，ファイルシステムとソケットでの通知によるプライベートサービス発見を実装している。ユビキタスサービスは周辺の機器に存在するサービスであり，そのサービス発見には標準であるUPnP(10)を使っている。サービスを見つけた段階でそれぞれ，プライベートの場合はローカルなファイルシステムから，ユビキタスの場合はUPnPのアクションを使い，OWL-Sによるサービスのセマンティックな記述を入手する。

サービスのサービス記述を見つけたら，サービスあるいはタスクの実行まで，Task Computingのクライアントはこのサービス記述だけを使ってユーザの各種操作を可能にする。

まずは発見されたサービスをリストアップしたり，各サービスの詳細を表示したりする。

しかし，それだけではユーザにとってその環境でどのようなこと（タスク）を行うことが可能なのか見えない。これらの機能は，その特定のユビキタスな環境で使える機能であるということでフィルタがかかってはいるが，それでもそれらの機能が意味なく羅列されたままでは役に立たない。（OSやアプリケーションの各種メニューの項目たちが脈絡なく並んでいる様子を想像してほしい。）それらの機能をエンドユーザのやりたいこととしての「タスク」として（ユーザも巻き込み）構成し，提示する必要がある。

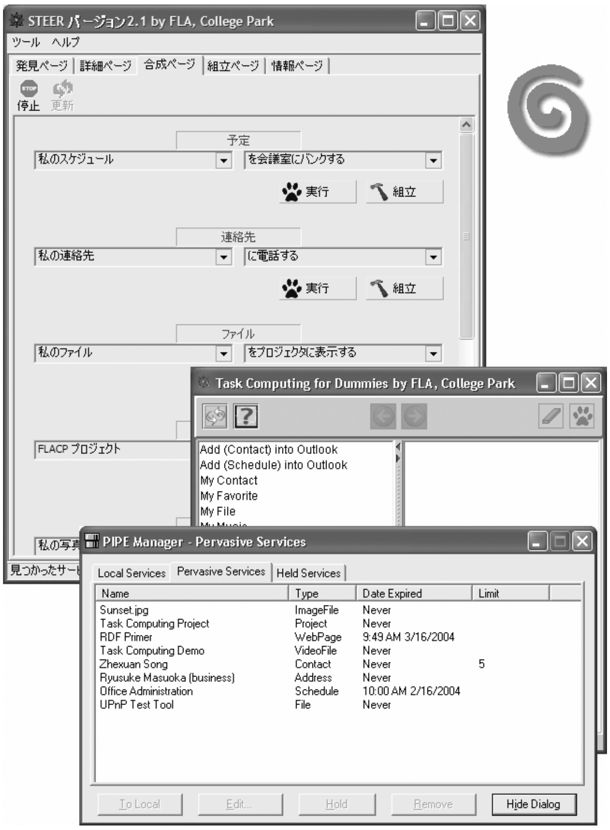


図-3　Task Computingのクライアントのデスクトップ例
Fig.3-Task Computing client desktop.

Task Computingのクライアントはいろいろな方法で，ユーザにそれらのサービスを合成したタスクを提示したり，ユーザがタスクを作ったりすることを可能にする。ここでセマンティクスが特に重要になってくる。それぞれのサービスのセマンティクスを見ることにより，どのサービス合成が意味のあるものか，ユーザにどのように（サービス合成としての）タスクを提示するか，あるいはユーザがいかにタスクを操作できるようにするかを決定することができる。

Task ComputingのクライアントであるSTEER（Semantic Task Execution EditoR）の画面例を**図-3**に示した。図-3上のSTEERは自動的に二つのサービスから合成したタスクを表示しており，ユー

ザが「実行」ボタンをクリックすれば，その組合せをすぐに実行できる。プルダウンメニューから合成可能なほかのサービスを選択し，新たなタスクを実行するのも簡単である。あるいは図-3の右中の別のクライアントインタフェースでは，ユーザがインタラクティブにサービス合成としてのタスクを簡単に定義できる。ユーザがタスクの実行を指示すると，タスクを構成する各サービスのセマンティックなサービス記述を参照し，セマンティックなレベルの入出力と具体的なサービスの実装（Webサービスや UPnPアクション）へのマッピングを知り，それを使って，つぎつぎとサービスを実行していく。

## 動的なサービスの生成，管理

実世界のものを，すべてがサービスであるTask Computingの世界に自由に持ち込めるようにするには，もの（オブジェクト）を簡単にサービス化できることが必要である。著者らはユーザにそのためのいろいろな手段を提供している。White Hole（図-3右上の渦巻き）はローカルなマシンのGUIを通じてのサービス生成を可能にするし，Bankサービスは実行されると渡されたオブジェクトを提供するサービスを生成し，また各種プライベートサービスはその実行時にユーザが選択したオブジェクトを提供するサービス生成が可能になっている。またユーザがデバイスを使うことによりサービスを生成することも可能である。（例えばスキャンした結果のイメージを提供するサービスを生成することができる。）

これらサービスの動的生成，管理のコアの部分はPIPE（Pervasive Instance Provision Environment）によって実現されている（図-3の一番下のウィンドウがPIPEのGUIである）。Task Computingの各種モジュールはこのPIPEのWebサービスインタフェースを使って，サービスの動的生成や管理を行っている。例えばWhite Holeに，OSのオブジェクト（ファイルやURL）やアプリケーションからのオブジェクト（PIMからの予定や連絡先など）をドラッグ＆ドロップすると，White Holeはまずそれらのオブジェクトを解析し，そのオブジェクトの内容に基づいたセマンティックなオブジェクトを生成する。このセマンティックなオブジェクトを使ってPIPEのWebサービスインタフェースを呼ぶと，PIPEはそのオブジェクトを提供するWebサービスを生成し，さらにそのWebサービスのセマンティックなサービス記述を生成し，プライベートやユビキタスなサービス発見の実装に対して新たなサービスとして登録する。

## シナリオの実現

現時点のTask Computingでは，英語音声によるものを含め7種類のクライアント，50種類以上のサービスを実装している。さらに国際化対応済で，8箇国語（中国語，英語，ギリシャ語，ヒンズー語，日本語，韓国語，トルコ語，スペイン語）の言語パッケージを用意した。さらにWindows上にIIS，.NET，Java実行環境があれば，このすべて（あるいは任意のコンポーネントの組合せ）をインストーラにより数分でインストールし，使い始めることができる。またTask Computingの各種機能にアクセスするためのWebサービスによるAPIも用意してあり，それを使えば，自分なりのクライアント，サービスを作成することも容易に可能である。

これらはもちろん実用化を目指すという目的もあるが，むしろユビキタス環境におけるIT技術の研究に必須であると考えている。まずは実験を始めるために，これらを用いて簡単に「いつでも，どこでも」数多くのサービスからなるユビキタス環境を実現できる。さらに，本当に使えるサービスが数多くある環境の中，ユーザがどのようにその環境とインタラクトできるのかをいろいろなクライントを実際に作って試すことができる。

このTask Computingのシステムを使うと前述のシナリオも簡単に実現される。例えば，「初めての会議室の電話で自分のPIMにある電話番号をかける」というものであれば，例えば図-3の左上のTask ComputingのクライアントであるSTEERのインタフェースに現れているように，ユーザがその会議室に入り，そのネットワークにアクセスした段階で，その会議室に提供されている「に電話をかける」サービスを発見し，それが自分のマシン上のサービス，「私の連絡先」と合成可能であることを認識し，実行可能なタスクとして「私の連絡先に電話をかける」があることをSTEERがユーザに提示する。ユーザはその行の下にある「実行」ボタンをクリックするだけで，そのサービス合成を実行し，

実際にPIMから連絡先を選び，その電話番号を会議室の電話機でかけることができる。

## む　す　び

「ユビキタス環境でのIT技術の要件」の章であげたシナリオはTask Computingですでに実現されていることのほんの一部である。

Task Computingはユビキタス環境だけでなく，OS，アプリケーション，Webサービスなどが提供する膨大な数の機能を使いこなさなければならなくなっているモバイル環境，あるいはPC環境でのIT技術にもやがて変革をもたらすと考えている。このTask Computing技術の重要な点は，富士通がかかわるであろう多くのアプリケーション領域，とくにユビキタス環境でのソリューションにおいて，いろいろなところにある各種機能を自在につなげることを可能にすることである。サービスの仮想化により，マシンやデバイスの境界をやすやすと越えて，エンドユーザ自身が機能をつなぎ合わせ，やりたいこととしてのタスクを実行することができる。これによりソリューションの提供の仕方も大きく変わってくるであろう。

このTask Computingによって，ユビキタス環境のIT技術に求められる要件の多くを実現，あるいは少なくともそのためのインフラを提供できたと考えている。今後の課題は，標準が決まりつつあるWebサービスのセキュリティをもとにしたユビキタス環境でのセキュアなサービス実行や，別の環境にあるサービスの発見や実行を実現することなどである。

## 参 考 文 献


(1) R. Masuoka et al.：Task Computing - the Semantic Web meets Pervasive Computing -. In Proceedings of the 2nd International Semantic Web Conference 2003 (ISWC 2003)，LNCS 2870，Florida，USA，2003，p.866-881.

(2) R. Masuoka et al.：Ontology-Enabled Pervasive Computing Applications. IEEE Intelligent Systems, vol.18，no.5，p.68-72 (Sep./Oct. 2003).

(3) Z. Song et al.：Dynamic Service Discovery and Management in Task Computing. MobiQuitous 2004.

(4) W3C Semantic Web.
http://www.w3c.org/2001/sw/

(5) MINDSwap.
http://owl.mindswap.org/

(6) 飯田一朗：XMLの新しい潮流－パーソナライゼーション技術－．*FUJITSU*，Vol.53，No.3，p.250-254（2002）.

(7) 湯原雅信ほか：Webサービスのダイナミック連携技術．*FUJITSU*，Vol.54，No.4，p.252-258（2003）.

(8) Web Service (W3C).
http://www.w3.org/2002/ws/

(9) OWL-S.
http://www.daml.org/services/

(10) UPnP Forum.
http://www.upnp.org/