社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
TECHNICAL REPORT OF IEICE.

セマンティック・ウェブ技術を応用したユーザ嗜好インタフェースの実現

柳田　拓人†　　野中　秀俊†　　栗原　正仁†

† 北海道大学大学院情報科学研究科　〒 060–0814 北海道札幌市北区北 14 条西 9 丁目
E-mail: †{takty,nonaka,kurihara}@main.ist.hokudai.ac.jp

あらまし　本稿ではユーザ嗜好インタフェースの設計アーキテクチャを提案する．このアーキテクチャはサービスからインタフェースの特定のスタイルに依存する部分を切り離し，結果としてサービスのインタラクション仕様やユーザの嗜好に沿った，さまざまな構成のインタフェースが実現可能となる．これは我々が日頃経験する，情報社会におけるサービスが提供するインタフェースが GUI など特定の形態に限定されているという不便な状況を改善するものである．また，アーキテクチャはインタフェースの細部と無関係にサービスを開発することを可能にする．このインタラクションの記述とインタフェースの構成アルゴリズムはセマンティック・ウェブ技術に基づいている．本文ではシステム・アーキテクチャと実装例について示している．

キーワード　抽象インタラクション記述語彙（AIDL），ユーザ嗜好インタフェース，インタフェース・クライアント，ロジック・サーバ，リソース記述フレームワーク（RDF），セマンティック・ウェブ

User-preferred Interface Design Based on Semantic Web Technology

Takuto YANAGIDA†, Hidetoshi NONAKA†, and Masahito KURIHARA†

† Graduate School of Information Science and Technology, Hokkaido University　Kita 14, Nishi 9, Kita-ku, Sapporo, Hokkaido, 060–0814 Japan
E-mail: †{takty,nonaka,kurihara}@main.ist.hokudai.ac.jp

Abstract　This paper proposes design architecture of user-preferred interfaces. The architecture separates style-sensitive parts of interfaces from a service, and consequently diverse configurations of interfaces can be realized according to not only the interaction specification of the service but the user's preference. The system improves an inconvenient situation we experience recently that the offered interfaces of services in the information society are limited to a specific style such as GUI. Furthermore, it enables development of service without regard to the detail of the interface. The description of the interaction specification and the interface configuration algorithm are based on the semantic web technology. We present the system architecture and an example of implementations.

Key words　Abstract Interface Description vocabuLary (AIDL), User-preferred Interface, Interface Client, Logic Server, Resource Description Framework (RDF), Semantic Web

1. はじめに

1.1 背　　景

情報社会においてコンピュータを接点とした**サービス**には様々なものがあり，例えば，交通機関の時刻案内やチケット予約，書籍や CD の通信販売などのウェブ・アプリケーションや，外出先から操作可能なビデオ・デッキなどの情報家電製品がある．また，従来から存在する単体の PC，および PC 上のあらゆるアプリケーション・ソフトウェアや，コンピュータを内蔵した身の回りのすべての電気製品もこの中に含めることができる．このような様々なサービス（情報サービスやアプリケーション，アプライアンス）が日常生活に深く浸透することによって，個々のサービスに付随するヒューマン・インタフェースに接する機会が増加している．

サービスを日常生活のあらゆる局面で用いるようになった現状において，それらサービスはインタフェースとして多くの場合 GUI のみを提供している．すなわち，利用環境やインタフェースに対する嗜好といった様々なユーザの特性を考慮していない．さらにアクセシビリティの観点から，身体的障碍の有無に関係なくサービスを快適に利用できることが望ましいが，従来では既存のマウスやキーボードなどの代替品を開発するというアプローチを取る場合が多かった．また，ユニバーサル・デザイン手法では嗜好の個人差には対処していない．

一方でサービスの提供者や開発者の視点からは，サービスそ

れぞれが各ユーザの特性に合わせたインタフェースを提供することは，開発を困難にしコストを増大させるため容易には実現できない．しかも現状においてさえ複雑なインタフェースは，アプリケーション・コードに占めるインタフェース処理の割合を増加させ，その実装に多大な労力を要求している．

筆者らはこの現状を踏まえ，サービス提供者がインタフェースを用意するのではなく，それと無関係なインタフェース提供者から，ユーザ自らが最適な形態のインタフェースを入手し使用できるようにすること，さらにそのインタフェースの研究開発を既存の代替品の範疇にとらわれることなく可能にすることを提案した [1]．そこで本稿では，従来のアプリケーションの構造と比較することにより，ユーザが操作する具体的なインタフェースとサービス本体であるロジックとが連携する提案アーキテクチャ，**インタフェース・クライアント/ロジック・サーバ（IC/LS）構造**の詳細について述べる．また，セマンティック・ウェブ技術 [2], [3] を応用した，デバイスに依存しないインタフェースの記述法と，その記述からインタフェースを構成する手法を説明する．

1.2 関 連 研 究

これらの状況に対応する関連研究は，複数の形態のインタフェース開発を支援するものと，アプリケーションやアプライアンスを標準的なインタフェースで操作可能にするものとに大別できる．前者として UIML [4] があり，ユーザの特性をサービスの設計段階である程度予測できる場合において有効である．また後者として GUI-文字 UI 変換によるアプリケーション遠隔操作システムの研究 [5] があり，遠隔操作のインタフェースをメールもしくはブラウザに限定しているが既存のアプリケーションをそのまま利用できるという利点がある．本研究と類似した研究として，他に Ubiquitous Interactor（UBI）[6], [7] や Personal Universal Controller（PUC）[8], [9] が挙げられる．UBI は画面表示のカスタマイズ機能を完全にユーザ主導としていない点が本研究と異なり，PUC は対象をアプライアンスに限定している点が本研究と異なる．

2. サービスからのインタフェースの分離

2.1 提案アーキテクチャの概要

筆者らはインタフェース・クライアント/ロジック・サーバ（IC/LS）構造を提案した [1]．これは，サービスから特定のインタフェースに固有な処理を分離し，分離した処理を**インタフェース・クライアント**，サービスのロジック処理を**ロジック・サーバ**とし，それぞれが互いの依存関係なしに連携することを可能としたものである．この構造の利点は，ユーザによるインタフェースの選択や交換が可能なこと，サービス提供者によるインタフェース処理の実装の手間が軽減されることである．

インタフェースを交換可能にするアプリケーション・アーキテクチャとして，従来から良く知られているものに Model-View-Controller（MVC）やそれを簡略化した Document/View 構造がある．ここでは後者と IC/LS 構造を比較する．Document/View 構造では Document がアプリケーションのロジック部分に対応し，View がインタフェース部分に対応する．そのためこの構造は，Document を固定して View を交換することによってアプリケーションのインタフェースの切り替えが可能である．しかしながら View は Document の内部構造に依存しているために，他の Document との接続による正常動作が保証されない．

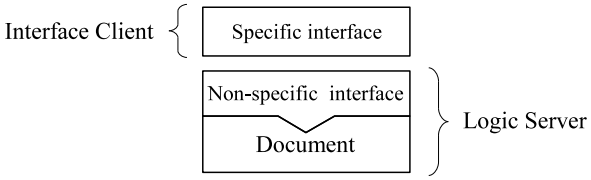

図 1　インタフェース・クライアント/ロジック・サーバ構造
Fig. 1　Interface client/Logic server architecture

IC/LS 構造は，Document/View 構造における View を特定のインタフェースに固有な処理を扱う層とそれ以外の抽象的な処理を扱う層とに分割したものとみなすことができ，前者がインタフェース・クライアントに，後者と Document を合わせたものがロジック・サーバに該当する（図 1）．特定インタフェース層は Document に依存せずに，GUI や音声入出力などの具体的なインタフェースに関係する処理を行い，一方の抽象インタフェース層は個々のインタフェースに依存することなく，Document に特化したインタフェース処理を行う．

2.2 クライアントとサーバの連携

インタフェース・クライアントとロジック・サーバは，インタラクションの**提示指示モデル**に合致したインタラクション情報の受け渡しによって連携する．提示指示モデルとは，インタフェースを媒体としてユーザとサービスとが互いに交わす情報（インタラクション）をモデル化したものである（図 2）．

本モデルではインタラクションを，サービスからユーザへの選択肢の**提示**と，ユーザからサービスへの選択結果の**指示**から成立する**選択行為**として記述できるものに限定する．これは，実世界におけるインタラクションの要素のうち，特定のインタフェース（あるいはコミュニケーション・チャネル）に固有のものを除外する必要があり，より高次の認知プロセスに関わるものはサービスの範疇に含めることが妥当と考えたからである．

IC/LS 構造におけるインタラクションの構成要素は，どのようなカテゴリの中から選択するのかを表す**選択行為の意味**，何を選択するのかを表す**選択肢**，そして選択行為がどのように組み合わさっているかを表す選択行為の構造の 3 点である．サーバはこのインタラクション情報の操作のみを行い，クライアントがその情報を元に，選択行為の意味に応じた**インタフェース・コンポーネント**を取得し，ユーザが操作する具体的なイン

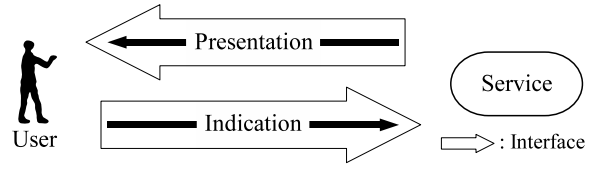

図 2　インタラクションの提示指示モデル
Fig. 2　Presentation and indication model of interaction

タフェースを構成する．そしてクライアントはこうして構成したインタフェースに対するユーザの操作を，ユーザが何を選択しているのかという情報としてサーバに伝達する．このような手法の採用により，インタフェースが具体的に何であるかに依存せずに，両者の連携が可能となる．

2.3 連携プロセス

IC/LS 構造のシステムでユーザがサービスを利用するためには，まずインタフェース・クライアントをロジック・サーバに接続する．その後，**抽象インタラクション記述語彙 AIDL**（Abstract Interaction Description vocabuLary）（3.1 を参照）を用いた RDF（Resource Description Framework）[10] 文書の送受信によりインタラクションが継続する．以降，接続後にユーザがインタフェースを操作可能になるまでの流れを説明する（図 3）．

（1） 初めに，クライアントはサーバからそのサービスを利用するに当たって必要なインタラクション情報を受け取る．ここでは例として「pal と koro と chiro の中から 1 匹の Dog を選択する」という記述を受け取ったとする．すなわち pal，koro，chiro は選択肢を，Dog は選択行為の意味を表す．

（2） 続けてクライアントは，受け取った記述にある選択行為の意味 Dog についての情報を取得する．取得するソースはクライアント自身が保持している場合とネットワーク上から取得する場合とがある．そうして得た情報から，Dog という意味の選択行為を表現できるインタフェース・コンポーネントが何で，どこにあるかを知る．例えば「Dog を表現するコンポーネントは DogSelection.class である」という情報を得たとする．

（3） 次にクライアントは，得た情報に基づいてコンポーネントを取得する．このときもコンポーネントの所在はクライアント自身の場合とネットワーク上の場合がある．クライアントがデバイスを有していてそれに対応するコンポーネントを保持している場合は前者になる．もし，Dog に利用可能なコンポーネントが複数存在した場合は，ユーザの特性を記述した**ユーザ・プロファイル**によって 1 つに絞り込む．

（4） 続いてクライアントは，取得したコンポーネントに対して，そのコンポーネントが必要とするインタラクション記述の一部を渡す．コンポーネントは受け取った記述を解析し，必要な情報が不足している場合はサーバ，もしくはネットワーク上から追加取得する．

（5） コンポーネントは自分の持つ情報を元に自らを初期化，あるいは実際にユーザの操作するインタフェースを生成しユーザに対して公開する．ユーザの操作はクライアントが持つインタラクション記述を変化させ，それをクライアントがサーバに伝えることによって，ユーザとサービスとのインタラクションが行われる．

3. インタラクション記述とインタフェース構成

3.1 AIDL によるインタラクション記述

ロジック・サーバのインタラクション情報は，AIDL を用いた RDF 文書として記述される．セマンティック・ウェブ技術の 1 つである RDF は，URI 参照を用いユニークにリソースを

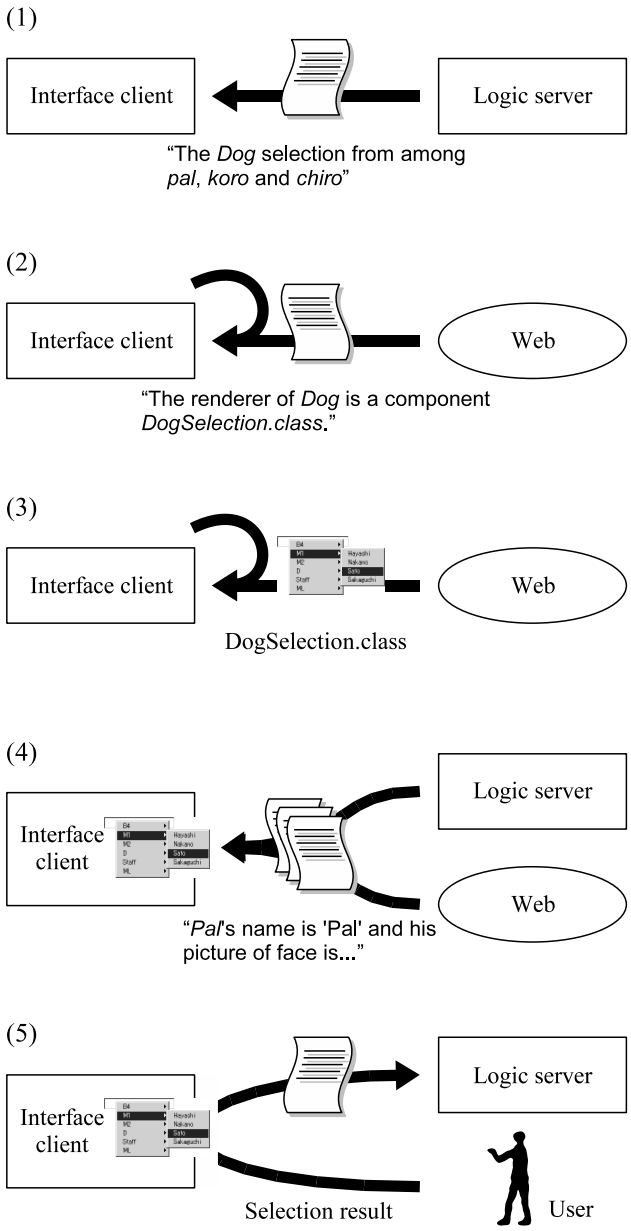

図 3 クライアントとサーバの連携プロセス
Fig. 3 Cooperation process of interface client with logic server

記述する方法，およびそれらリソースを分類する Class の概念を持っており，IC/LS 構造ではそれぞれを，選択肢と選択行為の意味を表現するために用いている．選択行為の意味は一般にサービスによって異なり，増加することが想定される．これに対して本システムでは様々なコミュニティや個人が必要に応じて RDF の Class として定義できる．

AIDL はインタラクションを記述するために筆者らが設計した語彙である（表 1）．Dublin Core [11] や FOAF [12] など他の語彙とともに用い，サービスを利用するのに必要な選択行為を表現する．AIDL の要素である aidl:Option(注1) クラスのイン

(注1)：各語彙の要素であることを意味する接頭辞として，RDF および RDF スキーマの場合は rdf:および rdfs:，Dublin Core の場合は df:，FOAF の場合は foaf:，AIDL の場合は aidl:をそれぞれ要素名の前に付ける．また，例示のための語彙を表す接頭辞として ex:を用いる．

表 1　AIDL の主要な要素
Table 1　Main elements of AIDL.

要素名	解　説
Selection	すべての選択行為を表すクラスの親クラス.
SingleSelection	すべての単一選択行為を表すクラスの親クラス.
Option	選択肢の中から選択する行為を表すクラス.
Range	範囲内から選択する行為を表すクラス.
Free	選択肢を指定できない選択行為を表すクラス.
SelectionGroup	複数の選択行為のまとまりを表すクラス.
Interaction	インタラクションを表すクラス.
means	選択行為が何を意味するのかを示すプロパティ.
hasChoice	Option の選択肢を示すプロパティ.
hasResult	選択行為が持つ結果を示すプロパティ.

スタンスが主に選択行為を表現し，そこから aidl:hasChoice プロパティによって連結したリソースが選択肢を表す．各リソースからはさらに別のプロパティによってノードが連結し，その選択肢の情報を表す．また，aidl:Option クラスのインスタンスは aidl:means プロパティを用いて rdfs:Class クラスのインスタンスと連結することによって，選択行為の意味を表現する（図 4）.

3.2　インタフェース・コンポーネント

インタフェース・コンポーネントはインタフェース・クライアントにおいて，インタラクション記述に現れる選択行為それぞれに対応して存在し，インタフェースを構成する．コンポーネントは，インタラクション記述（D）を受け取って解釈し，自身を初期化あるいはインタフェースを生成し，ユーザによる操作を反映したインタラクション記述（D'）を返す（図 5）．具体的には，主として aidl:Option クラスのインスタンスに，aidl:hasResult プロパティを用いて選択結果のリソースを連結することによって表現する．すなわち，コンポーネントはユーザによる選択行為を記述の変化に置き換える．

コンポーネントに必要な機能はユーザの選択行為を正しく記述の変化に結びつけることのみであるので，様々なコンポーネントを作成可能である．そのため新しいインタフェースを開発したとき，少ない手間でコンポーネント化できる．また，選択行為の意味に合わせて専用のコンポーネントを製作できるので，ユーザにとって分かりやすいインタフェースを提供することができる（図 6）.

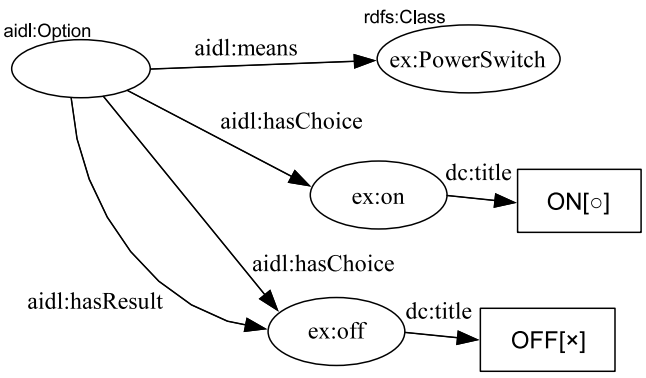

図 4　AIDL によるインタラクション記述の例
Fig. 4　An example of interaction description with AIDL

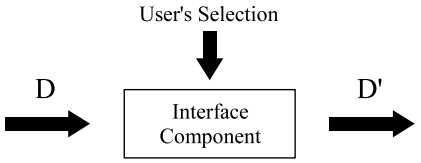

図 5　インタフェース・コンポーネント・モデル
Fig. 5　Interface component model

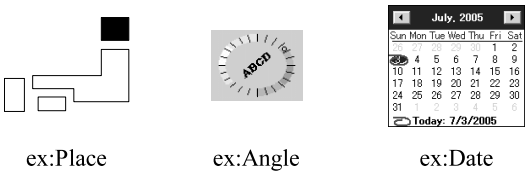

図 6　インタフェース・コンポーネントの例
Fig. 6　Examples of interface component

新規に開発するのではなく，既存のインタフェースをコンポーネント化する例として，GUI のウィジェットをコンポーネントとする場合がある．ユーザによるウィジェットの操作，すなわちラジオ・ボタンの切り替えやチェック・ボックスのオン，オフなどはそのまま aidl:hasResult プロパティの変化に結びつけることができる．GUI 以外では，例えば音声認識によるインタフェースとして，ユーザによる特定の単語の発話に対応し，その単語が意味するリソースを aidl:hasResult によって連結するコンポーネントなどが作成可能である．いずれの場合も選択行為の意味に対応させるにはさらに工夫が必要である．

3.3　インタフェース構成アルゴリズム

IC/LS 構造では RDF を用いてインタフェースの記述を行い，それを解釈しインタフェースを構成するインタフェース・コンポーネント自体も，インタフェース・クライアントが RDF による知識ネットワークを検索することによって取得する．さらにコンポーネントが具体的なインタフェースを生成するために必要な情報も RDF 文書として取得する．このようにクライアントは必要となる様々な情報をすべて RDF 文書として取得し，動的に**マージ**する（図 7）.

サーバから受け取ったインタラクション記述（1）には，aidl:means プロパティによる選択行為の意味の記述があり，それによってクライアントはサーバがユーザに何を選択させようとしているのかを知る．例えば，選択行為が ex:Dog を意味しているという記述があったとする．クライアントはこの記述から，ユーザに対してインタフェースを構成するためには ex:Dog に対応したコンポーネントが必要であると判断し，対応したコンポーネントを取得するために，ex:Dog を含む RDF 文書（2, 3）をローカル，またはネットワーク上のデータベースから取得し，インタラクション記述とマージする．コンポーネントの情報を含んだ RDF 文書にはコンポーネントに関する様々な属性が記述されている．その属性を元に，ユーザ・プロファイルを検索クエリとして対応するコンポーネントを検索する．クライアントは検索結果として得たコンポーネントに元のインタラ

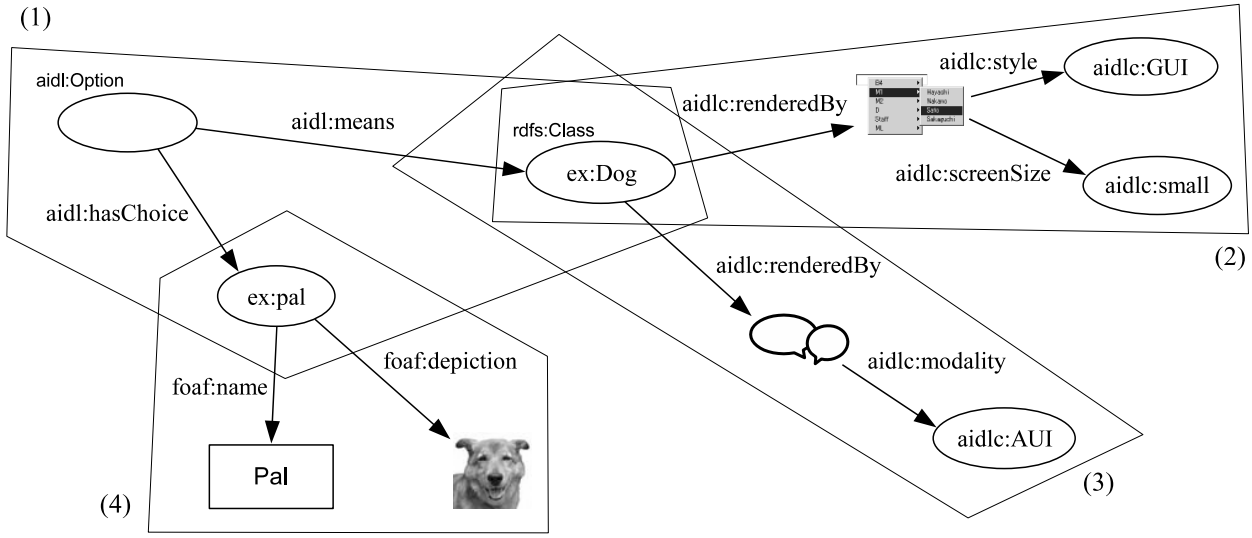

図 7　4 つの RDF 文書のマージによるネットワーク
Fig. 7　A network by merging 4 RDF documents

クション記述の一部を渡し，コンポーネントがその記述を解釈する．すなわちインタフェースを生成する．ここで不足する情報は別の RDF 文書（4）として取得，マージし，インタフェース生成に用いる．

選択行為の意味がクライアントにおいてすでに存在している場合，選択肢を記述しなくても意味だけからインタフェースを構成できる．例えばサーバから選択行為の意味として「ex:PowerSwitch」だけを受け取り，その選択肢の情報を受け取らなかったとしても，クライアント側に対応するコンポーネントが存在するならば，その選択肢が「ex:on」と「ex:off」とであることが自明であるため，インタフェースを生成できる．また，対応コンポーネントがないときでも，外部に追加情報を求め得られたなら，同様にインタフェースを生成することができる．このように RDF による知識ネットワークを組み合わせてインタフェースを構成する手法は，クライアントやコンポーネントの能力に合わせて柔軟に運用することができる．

4. 応用例と実装

提案アーキテクチャの適用範囲には，様々なバリエーションが存在する．1 つはアプリケーションの設計として，従来の MVC，あるいは Document/View 構造を置き換える応用方法である．この場合，従来の View に対してアーキテクチャを適用することで，インタフェース交換可能なアプリケーションを作成できる．ただしこれでは対応するインタフェース・クライアントもアプリケーション作成者が作成しなければならず，その手間は依然として存在し，アプリケーションの拡張性がある程度向上するという利点はあるものの問題の本質的な解決にはならないだろう．

4.1 インフラとしての応用

より有効な例として筆者らが検討しているのが，IC/LS 構造によるシステムを社会的なインタフェースのインフラとすることである．これにより，身の回りのあらゆる場所に埋め込まれたサービスを，ユーザが身に着けている様々な形態のインタフェースを用いて利用するという「ユビキタスなサービスにウェアラブルなインタフェースでアクセス」可能な環境を実現し，ユーザが遍在するサービスを利用する際に「お気に入りの使い勝手をいつでも，どこでも，何にでも」再現可能にすることができる．

ここで，システムが社会に広く普及した場合の生活スタイルがどのようになるのかをストーリー仕立てで紹介する．以下はある 2 人の人物が体験する出来事である：

（1） ある日エヌ氏は携帯音楽プレーヤで音楽を聞きながら，駅の構内に入ってきた．すると音楽が徐々に小さくなっていき「切符販売サービスです」という音が聞こえた．
（2） 続けて聞こえてきた「行き先はどこですか」の問いに対してエヌ氏は「○○行き」とつぶやいた．
（3）「590 円になります」という音声に対してエヌ氏は 「決定」とつぶやいた．すると改札をそのまま通ることができた．後日通帳を見ると切符代 590 円が引き落とされていた．

（1） ある日エム氏は携帯電話でメールを打ちながら駅の構内に差し掛かった．すると，携帯電話の画面の端に小さな字で「切符販売サービス」の文字が表示された．
（2） 画面に駅名がずらりと表示されたので，エム氏はそこから行き先を十字キーで選んだ．
（3）「590 円になります」という表示が出たのでエム氏は決定を意味する「1」ボタンを押した．すると改札をそのまま通ることができた．後日通帳を見ると切符代 590 円が引き落とされていた．

この例において重要なことは，切符販売サービスの提供者はユーザの利用形態には感知せず単一のサービスを提供しているにもかかわらず，2 人のユーザが異なる形態のインタフェース

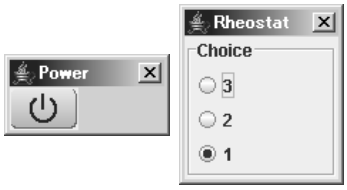

図 8　インタフェース・クライアントの動作画面
Fig. 8　A screenshot of an interface client

によってサービスを利用できた点である．すなわちユーザは自分の選択したインタフェースによってサービスを利用できたことになる．

このストーリー上では，2 人が使っていたようなシステム対応の機器を，ユーザが店頭で自由に選んで購入できるという状況を想定している．これは現在における携帯電話とそれで利用可能なサービスとの関係に類似している．携帯電話のユーザは同じ音声通話などのサービスを受けるために，キャリアとは別の各メーカが提供する携帯端末を，利便性を基準に自由に選ぶことができる．このインフラとしての IC/LS 構造システムの応用はより広範囲のサービスとそのインタフェースに対して，そのような環境，つまりサービス提供者とインタフェース提供者が独立に存在し，ユーザがそれぞれのサービスとインタフェースを自由に組み合わせて利用できる環境を可能にするだろう．

4.2 実　　装

IC/LS 構造を実装したシステムを開発した．プログラミング言語に Java を用い，セマンティック・ウェブ・ライブラリである Jena [13], [14] を使用した．インタフェース・クライアントはロジック・サーバと TCP/IP 接続し，Java で書かれたコンポーネントを取得した上でユーザにインタフェースを示し，操作結果を RDF 文書としてサーバに送ることができる．

GUI 版のクライアントを作成した（図 8）．サーバと接続すると受信した RDF 文書からサービスに必要な選択行為の意味を列挙し，あらかじめ用意しておいたコンポーネント情報を検索してネットワーク上にあるコンポーネントを取得し GUI を構成する．そしてユーザの操作を受け付け RDF 文書に反映し，その情報をサーバへと送信する．例として電源スイッチを意味する ex:PowerSwitch クラスを定義し，それに対応するコンポーネントを作成した．対応コンポーネントが見つからない場合に備えて，デフォルトのコンポーネントを内蔵する．

サーバとして仮想的な卓上スタンドを作成した（図 9）．クライアントと接続するとサービスに必要な RDF 文書を送信し，クライアントからの応答を待機する．クライアントから RDF 文書によるユーザの操作結果を受信し電気のオン，オフと明るさの切り替えを行う．

5. おわりに

本研究では，現状のサービスがユーザの様々な使用環境を考慮せずに，サービスごとに固定的なインタフェースしか提供していないという現状を改善するために，サービスからインタフェースを分離し両者を連携させるインタフェース・クライアント/ロジック・サーバ（IC/LS）構造を提案した．そして，セマンティック・ウェブ技術の 1 つである RDF を用い，抽象インタラクション記述語彙（AIDL）を開発した上で，インタラクションの記述手法，およびその記述からのインタフェース構成手法を提案した．そして IC/LS 構造によるシステムが，上記の問題を解決する手段となり得ること，提案手法が実装可能であることを示した．

図 9　ロジック・サーバの動作画面
Fig. 9　A screenshot of a logic server

今後の課題として，AIDL に対して様々なサービスやコンポーネントの情報を記述するのに必要な語彙を追加していくことや，同一あるいは異なる形態のコンポーネント同士を組み合わせるときに必要な全体を構成するアルゴリズムの作成，ユーザの嗜好をいかに記述するかなどがある．

文　　献

[1] 柳田, 野中, 栗原：“サービス・ロジックとインタフェースの分離によるユーザ嗜好モダリティの実現”, ヒューマンインタフェース学会研究報告集, **6**, 4, pp. 43–48 (2004).
[2] 神崎：“セマンティック・ウェブのための RDF/OWL 入門”, 森北出版 (2005).
[3] “W3C Semantic Web”. Available at http://www.w3.org/2001/sw/.
[4] Harmonia: “Tutorial booklet december” (1997).
[5] 岡田, 旭：“ユーザインタフェース変換に基づく PC 遠隔操作システムの開発”, ヒューマンインタフェース学会論文誌, **4**, 4, pp. 235–244 (2002).
[6] S. Nylander and M. Bylund: “The ubiquitous interactor–universal access to mobile services”, HCII 2003 (2003).
[7] S. Nylander, M. Bylund and A. Waern: “The ubiquitous interactor–device independent access to mobile services”, CADUI'2004, pp. 274–287 (2004).
[8] J. Nichols, B. A. Myers, M. Higgins, J. Hughes, T. K. Harris, R. Rosenfeld and M. Pignol: “Generating remote control interfaces for complex appliances”, UIST 2002, pp. 161–170 (2002).
[9] J. Nichols, B. A. Myers and K. Litwack: “Improving automatic interface generation with smart templates”, IUI 04, pp. 286–288 (2004).
[10] “Resource Description Framework (RDF)”. Available at http://www.w3.org/RDF/.
[11] “Dublin Core Metadata Initiative (DCMI)”. Available at http://dublincore.org/.
[12] “The Friend of a Friend (FOAF) project”. Available at http://www.foaf-project.org/.
[13] “Jena”. Available at http://www.hpl.hp.com/semweb/jena.htm.
[14] 上田, 和泉, 森田, 橋田：“Jena–セマンティック Web アプリケーション開発のための Java フレームワーク”, 人工知能学会誌, **19**, 3, pp. 325–333 (2004).