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

柳田 拓人[*1*2]　野中 秀俊[*1]　栗原 正仁[*1]

Separation of Service Logics and Interfaces for User-prefered Modalities

Takuto Yanagida[*1*2], Hidetoshi Nonaka[*1] and Masahito Kurihara[*1]

Abstract – This paper presents a system architecture for user-prefered modalities with separation of service logics and interfaces. Nowadays a lot of web services like book shopping by form and information appliances like remote controllable VTRs or rice cookers are used. Most of their interfaces are GUIs regardless of users' properties in terms of users' physical characteristic, preferences and usage context. There are several solutions to deal with this modality problem. Some of them are multimodal interfaces development tools. However, It is hard to infer the properties. The purpose of our system is to provide the environment where users can use services with their own interfaces according to the situations. The interfaces are separeted from traditional services, and their connections are established on demand. The descriptions of interfaces must be highly abstracted to render multimodal interfaces. Thus we developed the language named Abstract Interaction Description Language (AIDL). To render actual interfaces from a description, the system relies on the intelligent information technology to reason about the user's preferences and renderer's capabilities.

Keywords : Abstract Interface Description Language (AIDL), User-prefered Modalities, Interface Client, Logic Server, Service Logic

1. はじめに

今日，交通機関の時刻案内やチケット予約，書籍やCD の通信販売といったウェブ・サービスが広く利用されるようになってきた．また，外出先から操作可能なビデオ・デッキや炊飯器などの情報家電製品が登場してきている．しかし，これら情報社会におけるサービスに用意されたインタフェースには，嗜好や利用環境といったユーザのさまざまな特性が考慮されることなく GUI のみしか提供されていないという不便な状況，すなわちモダリティの問題が見受けられる．

さらに情報サービスが広く社会で利用されるようになってきた今日アクセシビリティを忘れてはならない．従来，ユーザの身体的障碍に合わせてインタフェースを開発する場合，既存のマウスやキーボードの代替品を開発するというアプローチに限られる場合が多かった．個々に合わせるという手法がある一方，以前からユニバーサル・デザインという，ユーザの障碍の有無にかかわらずあらゆる人が利用可能にすべきであるという製品設計思想がある．しかしながらサービスにおけるインタフェースの利便性や快適性は，身体的特徴を含んだユーザの特性に強く依存する．そのため，あらゆるユーザの要望を満たす万能なインタフェースは作り得ないと言える．

*1: 北海道大学大学院 情報科学研究科
*2: takty@main.ist.hokudai.ac.jp
*1: Graduate School of Information Science and Technology, Hokkaido University

サービスのインタフェースにユーザの特性が反映されていないという原点に立ち返って考えると，ユーザの特性やそれに適したインタフェースを一番良く知っているのはユーザである．本研究の趣旨はこの考えと現状を踏まえ，サービスの提供者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モダリティのインタフェースを入手し使用できるようにすることである．そのために本研究では：

- さまざまなモダリティのインタフェースをいかに統合して記述し動的生成させるか，
- サービスからインタフェースをどのように分離し両者を連携させるか

を中心に研究開発を進めている．以下本稿では関連研究を示したあと，提案システムの概要と構成する手法について述べ，最後に実装例を示す．

2. 関連研究

関連研究は複数モダリティのインタフェース開発を支援しようとするものと，アプリケーションやアプライアンスを遠隔操作しようとするものに大別できる．前者では UIML [9] などの手法があり，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有効であると考える．また後者では 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 [7] があり，GUI 部品にセマンティクスが含まれないために適切な変換には困難が伴うことが予想されるが，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．UBI [5] [4] はサービス提供者に表示をコントロールする機会を与えている点に特定のインタフェースへの依存を強めてしまう懸念がある．本稿とよく似た研究として PUC [2] [3] があるが，対象をアプライアンスの遠隔操作に限定している．

3. システムの目的

本研究の目的とするコンセプトは次の 2 つである．まず 1 つ目は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さまざまな形態のインタフェースで利用するという「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」可能な環境の実現である．そして 2 つ目は，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である．

ここで，提案システムが社会に広く普及した場合に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．初めはエヌ氏が体験した出来事である：

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出来た．後日通帳を見ると切符代 590 円が引き落とされていた．

次はエム氏が体験した出来事である：

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出来た．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のは，切符販売サービスの提供者は，どのような形態のインタフェースによって利用されるかについて事前にまったく感知していなかったにもかかわらず，2 人が異なる形態のインタフェースによってサービスを利用できた点である．すなわちサービス提供者はユーザの特性を予測する必要がなかったのである．

2 人が使っていたような提案システム対応の機器はストーリーの上では彼らが店頭で購入したことになっている．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の関係に似ている．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，各会社から提供される携帯端末を利便性を基準に自由に選ぶことが出来る．提案システムはより広範囲のサービスとそのインタフェースに関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実現することを目指している．

しかしながら，上で述べたコンセプトやストーリーはその全てを実現するために数多くの課題を抱えている．そのため本研究では 1 章で述べたとおり研究対象を，インタフェースをモダリティ独立に記述しそこから動的にインタフェースを生成させる方法と，サービスからインタフェースを切り分け両者を連携させる方法に限定している．

4. 提案システム

4.1 システムの構造

提案システムは，ユーザが各々所有するインタフェース・クライアント（以下クライアントと略記）と，サービス提供者が用意するロジック・サーバ（以下サーバと略記）から成り立つ．両者は互いに独立して存在するが，ユーザがサービスを利用するときに抽象インタラクション記述言語（AIDL）文書を介してクライアント・サーバ関係で接続される（図 1）．

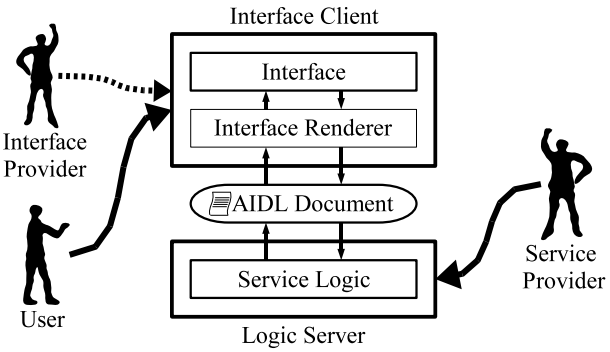

図 1　提案システムの概要
Fig. 1　Concept of the proposed system.

クライアントはインタフェース・レンダラ（以下レンダラと略記）とレンダラが接続時にサーバから受け取った AIDL 文書から生成したインタフェースによって構成される．そしてそのインタフェースの形態を希望するユーザによって所有される．レンダラはユーザによるインタフェースの操作を AIDL 文書に反映し，サーバによる AIDL 文書の書き換えをインタフェース

に反映する．

サーバはサービス・ロジックから構成され，サービス固有の AIDL 文書を持つ．ここで言うサービス・ロジックとは，一般的なサービスやアプリケーションにおける特定のインタフェース固有の処理を，AIDL の操作に置き換えたものである．

クライアントにおいて特定のサービスに固有な処理は必要とされないため，クライアント提供者は特定のサーバを考慮する必要がない．逆にサーバは AIDL 文書の操作という高度に抽象化されたインタフェース処理を行い，GUI や CUI といった具体的なインタフェースに固有な処理は行わない．このため，サービス提供者は具体的なインタフェースを考慮する必要がなくなる．よってサービスからインタフェースは明確に分離されたことになる．

ここで留意しなくてはならないのは，サービスをサーバとクライアントに分ける切り口は，従来のアプリケーション・プログラミングにおける MVC モデルや Document-View 構造などのインタフェースを分離する手法とは異なる点である．提案システムにおけるサーバは特定のインタフェースに固有なコードを含まないだけであり，それ以外のコード，すなわち抽象的なインタフェースのコードは含んでいる．

4.2 AIDL 文書による連携

クライアントとサーバは AIDL 文書を介して連携する．クライアントは接続時，サービスの内容に即した AIDL 文書をサーバより受け取る．AIDL 文書にはユーザとサービス・ロジックの間でやり取りされるインタラクションが記述してある．これを元にクライアントのレンダラが，実際にユーザが操作するインタフェースを生成する（図 2 上）．

ユーザが生成されたインタフェースを操作すると，その結果はインタフェース・レンダラによって，AIDL 文書に反映される．サーバはユーザによるインタフェースの操作をこの AIDL 文書の書き換え差分として受け取り，操作に沿ったサービスの処理を行う（図 2 下）．このように AIDL 文書はインタフェースの仕様書としての役割と，インタフェースの現在の状態を表す役割の 2 つを担っている．

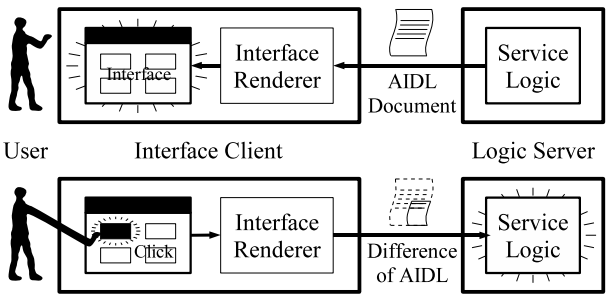

図 2　AIDL 文書からのインタフェースの生成（上）とインタフェース操作の反映（下）
Fig. 2　Generation of interface by AIDL document and reflection of interface operation.

AIDL 文書がクライアントから書き換えられることは上記したとおりだが，逆にサーバからも書き換えられる．このサーバによる書き換えによってインタラクションの状態遷移が実現される．GUI を提供するクライアントを例に挙げると，1 つの AIDL 文書は 1 つのダイアログ・ボックスに対応させることが出来る．その場合ウィザード形式のインタラクションは，次々と AIDL 文書が送られてくることで実現する．

5. 抽象インタラクション記述言語（AIDL）

AIDL はサービス・ロジックとユーザとの間でやり取りされるインタラクションを記述するための言語である（図 3）．しかしながら実際のインタラクションは非常に複雑であるため，それを単純化し提示選択モデルに合致するものとしてインタラクションを記述するようにした．

```
<aidl>
  <choice semantics="Station">
    <indicator>
      <caption>行き先</caption>
      <available type="image/png" data="to.png"/>
      <available type="audio/mpeg" data="to.mp3"/>
      <available type="movie/mpeg" data="to.mpeg"/>
      <detail>行き先の指定
        <detail>行き先を選択してください．</detail>
      </detail>
    </indicator>
    <option>
      <enum id="sapporo">
        <caption>札幌</caption>
      </enum>
      <enum id="otaru">
        <caption>小樽</caption>
      </enum>
      <enum id="hakodate">
        <caption>函館</caption>
      </enum>
    </option>
    <result understood="true">sapporo</result>
  </choice>
</aidl>
```

図 3　AIDL 文書
Fig. 3　AIDL Document.

5.1 提示選択モデル

提示選択モデルとは，インタフェースを媒体としてコンピュータとユーザとの間でかわされる情報のやり取り（インタラクション）をモデル化したものである（図 4）．このモデルにおいてサービス・ロジックとユーザとの間でインタフェースを介してやり取りされるインタラクションは．前者から後者への選択肢の提示と，後者から前者への選択結果の通知のみである．

5.2 記述言語の概要

AIDL が記述するインタラクションは木構造で表現され，その要素はロジックとユーザの対話の構造を表す構造要素，ユーザの行動を表す行動要素，サービス・ロジックからユーザへの情報を表す表示要素である．

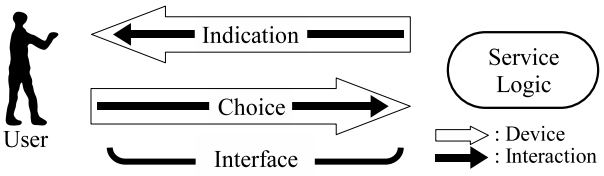

図 4　提示選択モデル
Fig. 4　Incidation and choide model.

これら要素はそれぞれの持つ意味によってさらに細かい要素に分けられる（表 1）．この木構造をインタラクション木と言う．インタラクション木と AIDL 文書の記述は同じものを表しているため，以降両者を区別しない．

インタラクション木は，要素の挿入と削除という操作によって生成され変化する．各要素は操作する主体がクライアント側であるかサーバ側であるかによって，挿入と削除のそれぞれが可能か否かが決められている．これにより AIDL 文書は常に意味的に妥当な状況に保たれる．また，インタラクション木におけるある要素とその兄弟要素との間には順序関係があり，要素の挿入時や削除時に決定される．ある要素の兄弟要素間における位置は他の要素を兄弟要素として挿入したり削除することによって移動する．

5.3　構造要素

構造要素はインタラクションの構造を表す．その各要素は意味上の包含関係を表し，インタフェース・レンダラはそのことがユーザによって理解されるように表現する．

5.4　行動要素

行動要素はサービス・ロジックがユーザに求める選択行動を表す．このときに選択の様相を示す選択型と，選択するデータの意味を示すデータ・セマンティクスという 2 つの概念を使用する．

選択型は選択肢の様相がユーザの行動を規定することに由来している．例を挙げると，料理を選ぶ時のような一定数の選択肢の中から選択する場合，ユーザはメニューに載った料理名から希望のものを指差す．また レストランの格付けをする時のような一定範囲の数値を選択する場合，マークシートを星の数だけ塗りつぶす．さらには自分の名前を求められたときのような選択肢の設けようのない選択の場合，空欄に名前を記入する．このような形態をそれぞれ：

- 選択肢からの選択（Option Choice）
- 範囲からの選択（Range Choice）
- 選択肢に依らない選択（Free Choice）

と名付ける（図 5）．選択肢からの選択はユーザがサービス提供者の用意した選択肢の中から選択することを表す．範囲からの選択はユーザが最小値と最大値の間にある数値を選択することを表す．選択肢に依らない選択はユーザが自由に入力できることを表す．これら選択型はユーザの行動を大まかに示すことに用いられる．

表 1　AIDL の要素
Table 1　Elements of AIDL.

分類	解　説
構造要素	Session（クライアントとサーバの接続），Interaction（意味上の大きなまとまり），Section（意味上のまとまり）
行動要素	Choice（選択），Option（選択肢のある選択），Range（範囲内の選択），Free（選択肢のない選択），Command（動作の実行）
表示要素	Indicator（表示），Caption（見出し），Detail（説明），Available（利用可能なメディア），Message（ユーザへのメッセージ）

図 5　　選択型
Fig. 5　Choice Types.

データ・セマンティクスは選択するデータの意味がユーザの行動を規定することに由来している．たとえば一般的に考えて日付を選ばせる場合カレンダーをユーザに表示し，また値段を決めさせる場合ユーザに電卓を操作させるのが普通である（図 6）．このように選択するデータのセマンティクスはインタフェースのメタファを決定する重要な役割を果たし，さらにメタファが持つアフォーダンスによってユーザの適切な行動を自然と促すことができる．

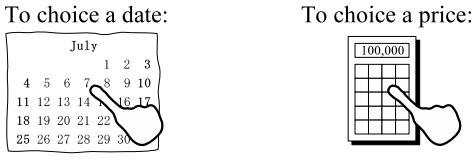

図 6　データ・セマンティクス
Fig. 6　Data Semantics.

なおデータ・セマンティクスは実世界の事象を扱っているため，闇雲に対象を広げては非常に種類が多くなってしまう．そこで一般的なサービスにおいてよく使われるであろうセマンティクスを限定し，データ・セマンティクス・テーブルとして AIDL の仕様に含めることを検討している．テーブルの項目はセマンティクス名，要素一覧（たとえば曜日というセマンティクスの場合，月から日までの各曜日），セマンティクスそれぞれの運用方法の解説といったものになる．

5.5 表示要素

表示要素は行動要素自身やそれが持つ選択肢が表示される際に必要となる情報を表す．その際の軸となるのが説明詳細度と呼ばれる説明の詳しさと，メディア提供能力と呼ばれるサーバが表示用としてクライアントに提供できるメディアの記述の2つである．なおここでの表示は視覚的なもの以外も含む．

説明詳細度はクライアントがユーザの希望に合わせて説明の詳しさを変化させられるように，あらかじめ複数の説明テキストを記述しておくものである．たとえばユーザによる名前の入力を受け付けるとき「名前」というテキストと「名前を入力してください」および「あなたのお名前をフルネームで入力してください」というテキストを併記しておく．するとユーザが簡潔な表示を好むときには「名前」のみが表示され，またユーザが初心者だったときにはなるべく詳しい説明があったほうが理解しやすいだろうから「あなたのお名前を〜」が表示されるということが可能になる．このように説明詳細度はクライアントの表示の自由度を高める．

メディア提供能力はサーバがクライアントにどのようなメディアが提供出来るかを記述するものである．メディア提供の例として，ショッピング・サービスにおける商品画像の提供や，音楽配信サービスにおける楽曲サンプルの提供などがある．しかしいずれの場合も注意すべきなのは，提供されるメディアはテキストを補完するものでなくてはならない点である．クライアントによっては画像に対応していないものや，音声に対応していないものも存在する．そのためメディアにはそれと同等の役割を果たすテキストが必須となる．

表示要素をどのように実際の表示に変換するかはクライアントの能力とユーザの好みによる．この両者をマッチングさせるための手法が必要となると思われるが，現在はまだ検討中である．

6. インタフェースの生成

AIDL文書からインタフェースを生成するのがクライアントに含まれるインタフェース・レンダラであることはすでに述べたが，ここでレンダラがどのようにAIDL文書を解釈するのかについて説明する．

クライアントはいかなる種類のインタフェースを提供するにしても，行動要素の3つの選択型と，表示要素におけるテキストの表示には対応している必要がある．行動要素にはデータ・セマンティクスの記述が含まれるが，クライアントは各セマンティクスすべてに対応している必要はない．受け取ったAIDL文書に記述されたセマンティクスに対応している場合，セマンティクス名から意味を判断し，意味に応じた様々な工夫が可能となる．ユーザの入力が適切であるかの判断がクライアントで可能なため，サーバでの判断は必要ない．一方対応していない場合，意味を持たないデータとして受け付け，ユーザの入力結果が適切であるかはサーバによって判断される．

選択肢からの選択，範囲からの選択，選択肢に依らない選択の各選択要素について，GUIを提供するクライアントがセマンティクスに対応していたときといなかったときの解釈例を示す．選択肢からの選択においてセマンティクスとして「都道府県」が指定してあったとき，対応しているクライアントでは日本地図が表示されクリッカブル・マップで入力を受け付けることが可能となる．対応していないクライアントではただのリスト・ボックスとして表現することが考えられる（図7）．

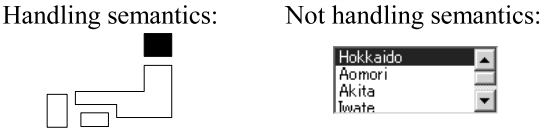

図7　選択肢のある選択の解釈例
Fig. 7　Interpretation of option choice.

範囲からの選択においてセマンティクスとして「角度」が指定してあったとき，対応しているクライアントでは分度器が表示され，それをクリックすることで角度の入力を受け付けることが可能となる．対応していないクライアントではスピン・ボックスとして表現することが考えられる（図8）．

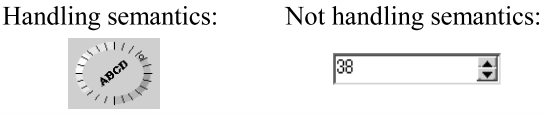

図8　範囲内選択の解釈例
Fig. 8　Interpretation of range choice.

選択肢に依らない選択においてセマンティクスとして名前が指定してあったとき，対応しているクライアントではアドレス帳から選べるようにすることが可能となる．対応していないクライアントでは，エディット・ボックスとして表現することが考えられる（図9）．

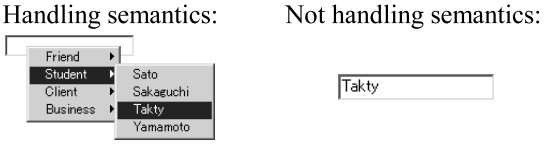

図9　選択肢のない選択の解釈例
Fig. 9　Interpretation of free choice.

7. 実装例

提案システムの実装例として，サーバとクライアントをそれぞれ示す．これらの実装には Borland C++Builder 3 を使用し，クライアント・サーバ間の通信方法として TCP/IP を使用した．

7.1 インタフェース・クライアントの実装例

GUI を提供するクライアントを実装し，AIDL 文書から GUI を生成させた（図 10）．選択肢からの選択を通常はチェック・ボックスの並びで，ウィンドウのサイズが小さくなりチェックボックスを並べて表示できなくなったとき，代わりにチェックリストボックスで表示するようにした．また，表示詳細度をユーザが自由に切り替えられるようにした．

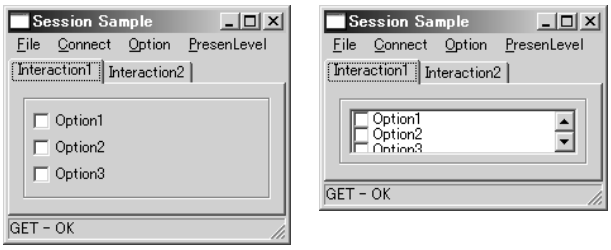

図 10 GUI 版クライアントの実装例の画面（左が通常時，右はウィンドウサイズが小さくなったとき）
Fig. 10 Implementation of client for GUI.

7.2 ロジック・サーバの実装例

提案システムのアプライアンスへの組み込み例として，卓上電気スタンドのシミュレーションとなる実装例を作成した．接続したクライアントからの電源の ON・OFF や明るさの切り替えが可能である（図 11）．

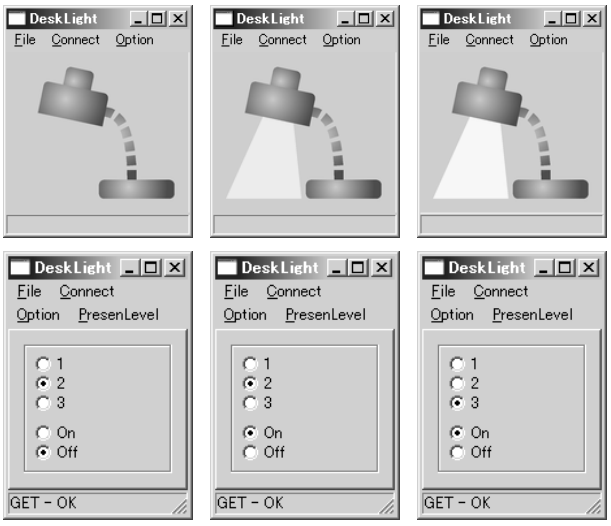

図 11 サーバ実装例と接続したクライアントの画面（上がサーバ，下がクライアント）
Fig. 11 Implementation of server and the client connected with it.

8. おわりに

本研究では現状のサービスにおける，ユーザの様々な使用環境が考慮されず GUI しか提供されていない問題を解決するために，サービスからインタフェースを分離する手法，およびそのための手段である抽象インタラクション記述言語（AIDL）を提案した．そしてロジック・サーバとインタフェース・クライアントによるシステムが，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．

今後の課題として，データ・セマンティクス・テーブルの決定，AIDL 文書の解釈アルゴリズムの作成，ユーザの嗜好とクライアントの能力をマッチングさせる手法の開発などがある．また，AIDL に実際のサービスを記述するだけの表現力があるかどうかの評価，さらには CUI 版や音声入出力版のクライアントを作成した上での GUI 版との比較，評価が挙げられる．

参考文献

[1] Göschka, K.M., Smeikal, R.: Interaction Markup Language–An Open Interface for Device Independent Interaction with E-Commerce Applications; Proc. the 34th Hawaii International Conference on Systems Sciences, IEEE Computer Society (2001).

[2] Nichols, J., Myers, B.A., Higgins, M., Hughes, J., Harris, T.K., Rosenfeld, R., Pignol, M.: Generating Remote Control Interfaces for Complex Appliances; UIST 2002, pp. 161-170 (2002).

[3] Nichols, J., Myers, B.A., Litwack, K.: Improving Automatic Interface Generation with Smart Templates; IUI 04, pp. 286-288 (2004).

[4] Nylander, S., Bylund, M.: The Ubiquitous Interactor–Universal Access to Mobile Services; Proc. HCII 2003 (2003).

[5] Nylander, S., Bylund, M., Waern, A.: The Ubiquitous Interactor–Device Independent Access to Mobile Services; Proc. CADUI'2004, pp. 274-287 (2004).

[6] Paterno', F., Mancini, C., Meniconi, S.: ConcurTaskTrees: A Diagrammatic Notation for Specifying Task Models; Proc. Interact'97, Chapman & Hall, pp. 362-369 (1997).

[7] 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，Vol.4, No.4,pp. 235-244 (2002).

[8] 荒木：ボイスウェブの可能性–VoiceXML 概説–；情報処理，Vol.44, No.10, pp. 1044-1051 (2003).

[9] Harmonia: Tutorial Booklet December 1997.

[10] World Wide Web Consortium: XForms – The Next Generation of Web Forms.
http://www.w3.org/MarkUp/Form

[11] Rivera, J.: XForms 入門; IBM.
http://www-6.ibm.com/jp/developerworks/xml/021115/j_x-xforms.html