抽象インタラクション記述言語を用いたユーザ嗜好インタフェースの設計

北海道大学 大学院情報科学研究科 コンピュータサイエンス専攻
数理計算科学講座 知能情報学研究室
柳田 拓人
平成 18 年 2 月 20 日

User-Preferred Interface Design with Abstract Interaction Description Language

Laboratory of Intelligent Information Systems, Research Group of Mathematical Information Science,
Division of Computer Science, Graduate School of Information Science and Technology, Hokkaido University
Takuto YANAGIDA
20 February 2006

1 ユーザ嗜好インタフェース

本研究はサービス・アーキテクチャである**インタフェース・クライアント/ロジック・サーバ構造（ICLS）**を提案する．ICLS は特定のインタフェースに固有な処理を行う**インタフェース・クライアント**と，すべてのインタフェース処理に共通な処理とサービス・ロジック処理を行う**ロジック・サーバ**から構成される（図 1）．そして両者は，ユーザとサービスとのやり取りを**抽象インタラクション記述言語（AIDL）**を用いて記述した**インタラクション・グラフ**によって，特定のインタフェースに依存せずに連携する．サービスに ICLS を採用することによって，インタフェースの容易な取り替えと自由度の高いカスタマイズ性からなる**ユーザ嗜好インタフェース**が実現する．本研究では，これをフレームワークとして実装することにより，そのようなサービスの開発を容易にする．

1.1 研究の背景

本研究の対象はコンピュータを接点とした情報社会におけるサービスである．たとえば，ウェブ・アプリケーションや情報家電製品，さらには，従来から存在する単体の PC 及び PC 上のあらゆるアプリケーションがそれに該当する．これらさまざまなサービスが日常生活に深く浸透することにより，個々のサービスに付随するインタフェースに接する機会が増加している．

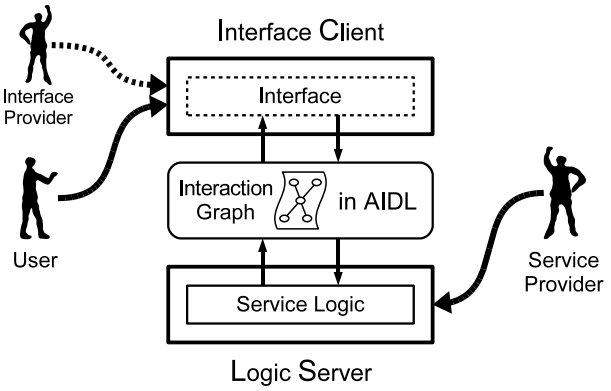

図 1 インタフェース・クライアント/ロジック・サーバ構造（ICLS）

このような現状において，サービスは多くの場合インタフェースとして単一の GUI のみを提供しており，利用環境や嗜好などの**ユーザ特性**を考慮していない．開発者にとってサービスそれぞれにおいて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出来ない．

本研究はサービスのインタフェースのユーザによる取り替えを可能とすることにより，この現状の打開を図る．

1.2 関連研究

関連研究には Ubiquitous Interactor（UBI） [1] や Personal Universal Controller（PUC） [2] がある．UBI は interaction acts を用いてインタラクションを記述し，デバイスに依存しないサービスの開発の手段を提供している．しかし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おらず，サービスが特定のデバイスへ依存してしまう懸念がある．PUC は対象を家電，事務機器などのアプライアンスを遠隔操作することに限定しており，インタラクションを状態変数として扱っている点が本研究との相違点である．また，Smart Template を提案することによってインタフェースの慣習的な表示に対応しているものの，その規格定義の仕方がはっきりしない．

2 インタラクションの記述

抽象インタラクション記述言語（AIDL）は，インタラクションを**提示指示モデル**によって抽象化し，グラフとして記述する．クライアントはこれ元に実際にユーザの操作するインタフェースを構築し，そのインタフェースに対する操作をグラフに反映させる．

2.1 提示指示モデル

提示指示モデルは**提示行為**（Presentation）と**指示行為**（Indication）の組からなる**選択行為**（Selection）としてインタラクションをモデル化したものである（図 2）．提示

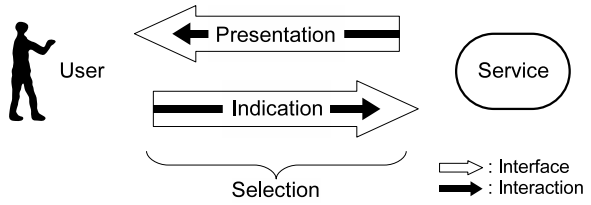

図 2　提示指示モデル

はサービスが提供可能な**機能**をユーザへ示す行為を，指示はその中からユーザがどれを求めるのかの**意図**をサービスへ示す行為を表す．機能と意図を**選択肢**として表現し，その集合を規定の仕方によって**列挙選択肢集合**，**範囲選択肢集合**，**全体選択肢集合**の 3 種類に分類する．さらに特定のインタフェースへ依存せずに記述の具体性を高めるために，どのような意味合いの選択行為なのかその意味を扱う．このモデルではインタラクションをこの選択行為として記述できるものに限定し，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．

2.2　抽象インタラクション記述言語

AIDL はセマンティック・ウェブ技術の 1 つである RDF（Resource Description Framework）の語彙として，インタラクションをグラフ表現する（図 3）．RDF が持つ URI によるリソース記述能力を，実世界におけるあらゆるものを選択肢にするための方法として利用し，クラスによるリソース分類を選択行為の意味として利用する．サービスによって選択行為の意味は異なるため，ICLS の利用例が増加するにつれて使用される意味も増加することが想定される．これに対して本研究ではさまざまなコミュニティや個人が必要に応じて意味を RDF のクラスとして定義できるため，意味の増加に対処出来る．

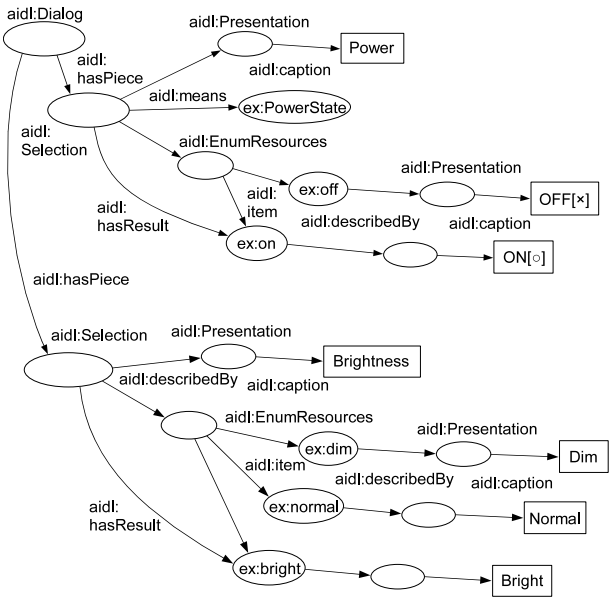

図 3　AIDL を用いたインタラクション・グラフの例

AIDL は主に aidl:Presentation（提示），aidl:Indication（指示），aidl:Selection（選択）リソースによってインタラクションを表現する．aidl:Group（グループ）リソースは，関連するリソースをまとめて，全体として木構造を形成する．aidl:Group を継承する aidl:Dialog（ダイアログ）リソースは木構造の親子関係においてルート要素となる．

aidl:Presentation リソースは提示行為を表し，他のリソースのユーザに対する説明などを表す．aidl:Indication リソースは指示行為としてユーザからサーバへ何らかの合図が必要なことを表し，これに対するユーザの応答は別途サーバへの通知の形態で伝達される．aidl:Selection リソースは選択行為として選択肢を表すために選択肢集合を表すリソースと連結し，aidl:hasResult プロパティによる選択肢の連結によってユーザによる選択結果を表す．aidl:Indication と aidl:Selection リソースには aidl:means プロパティによってそれぞれ意味が指定される．

2.3　状態の記述

インタラクション・グラフは単一のダイアログを含み，サービスのインタラクションは 1 つ以上のダイアログから成り立つため，1 つのダイアログでは完結しない複雑なサービスの場合，グラフを切り替えることにより状態遷移を表す．このような状態遷移を扱うのは AIDL 自身ではなく，外部からグラフを置き換えるサーバである．このためグラフ全体は，そのままユーザの現在におけるインタラクションの状態を表す．

また，**インタラクション・リレーション**として，インタラクション要素間の関係を表現出来るようにする．これによって 1 つのグラフの中で完結する単純な状態遷移や，要素間の意味的な関係を記述する．

3　提案アーキテクチャ

クライアントとサーバは互いの依存関係なしに，論理的に共有した 1 つのインタラクション・グラフを互いに変更しあうことによって連携する．実際にはクライアントとサーバとはそれぞれグラフを保持して互いにグラフを変更し，その**変更ログ**をやり取りすることにより構造を一致させる．このログの送受信によるグラフ構造の同期は**グラフ構造同期プロトコル**に従って行われる．

グラフを解釈して実際のインタフェースの生成を行うのは**インタフェース・コンポーネント**である．クライアントは自身が保持している，あるいはネットワークから入手するコンポーネントを複数組み合わせることによってインタフェースを構成する．コンポーネントは，グラフの選択行為や指示行為それぞれに対応して存在し，選択肢集合や選択行為の意味，指示行為の意味によってさまざまなものが存在する．ゆえに，どのコンポーネント

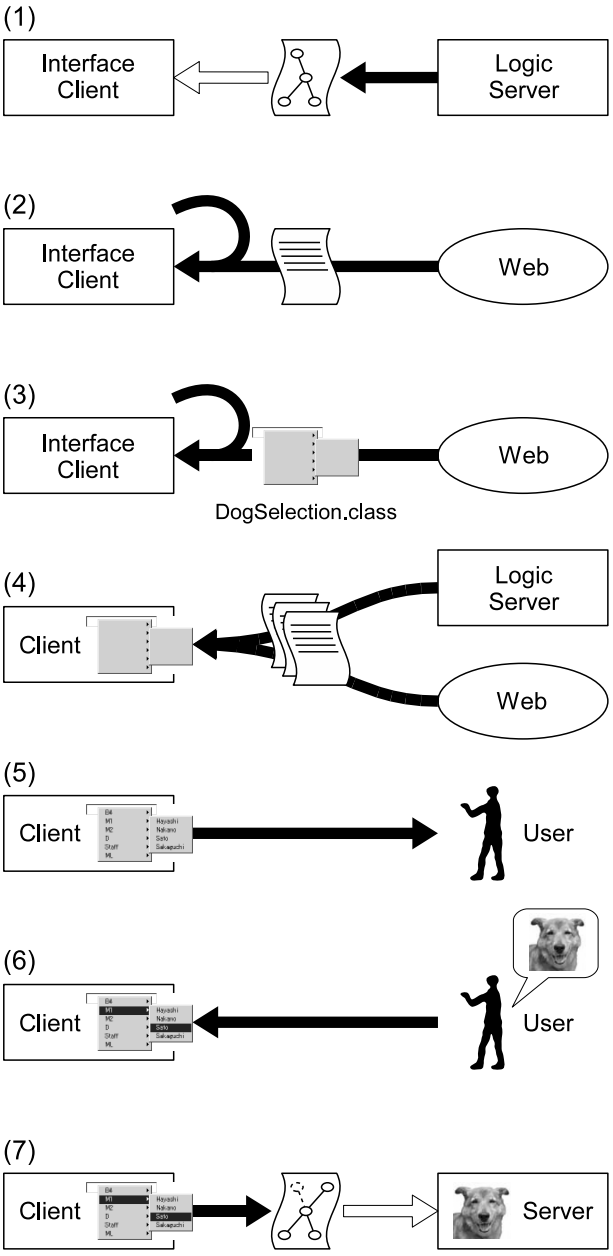

図 4　クライアントとサーバの連携プロセス

を用いるのか，用いることが出来るのかによって，クライアントはさまざまなインタフェースを構成可能である．

3.1　連携プロセス

ICLS を採用したサービスをユーザが利用するためには，初めにクライアントをサーバに接続する必要があり，接続後はグラフ構造同期プロトコルによって通信が行われる．ここで，クライアントがインタフェースを構成しユーザによる操作が可能になるまで，及びユーザの操作とそれに対するサーバの応答の流れを説明する（図 4）．

1. ユーザがクライアントをサーバに接続すると，クライアントとサーバはそのサービス用にサーバ開発者が用意したインタラクション・グラフを共有する．
2. クライアントは共有しているグラフに含まれる選択行為の意味についての情報をクライアント自身かネットワーク上から取得する．
3. クライアントは得た情報に基づいて対応するコンポーネントをクライアント自身から，またはネットワーク上から取得する．
4. 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グラフの一部を渡す．
5. コンポーネントは自分の持つグラフを元に自らを初期化，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表示する．
6. ユーザは表示されたインタフェースから何を選択するのかを理解した上で操作を開始し，コンポーネントはそのユーザの操作を受け付ける．
7. コンポーネントはユーザの選択結果が選択された状態になるように共有しているグラフを書き換え，サーバはそれに基づいてサービスを行う．

3.2　グラフ構造同期プロトコル

グラフ構造同期プロトコルは，RDF グラフの変更操作をクライアント・サーバ間で RDF/XML を用いて送受信し，両者のグラフを同一構造に保つための通信規約である．両者はそれぞれグラフと変更ログを保持し，グラフに対する操作を RDF ステートメントの追加と削除の 2 つとして保存する．そして，2 つの変更ログを任意のタイミングで互いにやり取りし相手のグラフに反映する．この変更ログの保存と反映を繰り返して両者のグラフを常に同一のものにする．通信の流れは以下のとおりである．

1. クライアントはサーバに接続し，サーバの持つグラフ全体を受信する．
2. クライアントとサーバは双方で各々自らのグラフを変更し，ログを保存する．
3. クライアントはログを送信し，サーバはそれを自らのグラフに適用する．サーバのグラフが大幅に変更されていた場合は 1 へ戻る．
4. サーバもログを送信し，クライアントはそれを自らのグラフに適用する．2 へ戻る．

更新には変更ログの送受信によるものと，クライアントのログ送信に対してサーバがグラフ全体を送信する場合の 2 つがある．クライアントが初めて接続したとき，あるいはサーバ側のグラフが大幅に変更された場合，サーバは保持するグラフの全体を送信する．それ以外はログによる差分情報のみの送受信であるので，常に全体を送受信するよりも通信量を削減できる．

4　フレームワークの実装

ICLS を適用したサービスやインタフェースの開発を容易にするために，フレームワークを提供する．これは

Java 言語によって書かれたクラス・ライブラリである．このフレームワークは，構成を従来の GUI ライブラリにならった設計とすることにより，開発者が同様の使い方で開発を行えるようにする．基本的な実装としてソケットを用いた TCP/IP 方式を採用する．

4.1 グラフの生成とイベント処理

本フレームワークでは Graph オブジェクトがグラフを表す．初めに Graph オブジェクトを生成し，それから取得する ElementFactory オブジェクトを用いて Dialog や Seleciton などのオブジェクトを生成する．生成したオブジェクトは各種設定を行った後，Dialog や Group オブジェクトに add メソッドによって追加する．これによって，Graph の保持するグラフに対応した RDF リソースが生成され必要なグラフが完成する．

フレームワークにおけるイベント処理も GUI に似た機構を採用する．IndicationListener，SelectionListener，SessionListener の 3 種類のイベント・リスナ・インタフェースをさまざまなイベント処理の際に用いる．それぞれの実装を Indication，Selection，Session に割り当てることによって，イベント発生時に適切なメソッドが呼び出される．

4.2 クライアントとサーバの開発

クライアントとして GUI 版と CUI 版，携帯端末型のシミュレータ，サーバとして卓上電気スタンドのシミュレータとスケジュール管理サービスを開発した．

GUI 版クライアントはグラフからダイアログ・ボックスを構築する（図 5-a）．コンポーネントの例として ex:PowerState と ex:Date に対応したものを開発した．CUI 版クライアントはグラフ構造をディレクトリ構造に見立ててユーザに示し，ユーザはコマンドを入力することによって操作する（図 5-b）．携帯端末型クライアントは，小さなスクリーンと 5 つのボタンを持ち，音楽プレーヤーのように操作する（図 5-c）．

卓上電気スタンド・シミュレータは電気スタンドのソフトウェアによるシミュレーションを遠隔操作するサービスである．家電製品を家庭内にて，あるいは外出先から操作するサービス形態の 1 例として開発した（図 5, 6-a）．スケジュール管理サービスは一般的な PC や PDA 上で動作するアプリケーションの 1 例である．クライアントを接続するとスケジュールの一覧が表示され，それらの修正や削除，新規追加が可能である（図 6-b）．

5 まとめと今後の課題

本研究では，サービスのインタフェースにまつわる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する，インタフェース・クライアント/ロジック・サーバ構造（ICLS）を提案した．また，RDF を用いた抽象インタラクション記述言語（AIDL）を開発し，インタラクションの記述手法，及び記述に基づくインタフェース構成手法を提案した．そして実装することによって，ICLS を採用したシステムが現状改善の手段となり得ることを示した．今後の課題には，コンポーネント自体とユーザの特性からの自動的なカスタマイズ手法の検討や，グラフを視覚的に記述できる開発ツールの提供などがある．

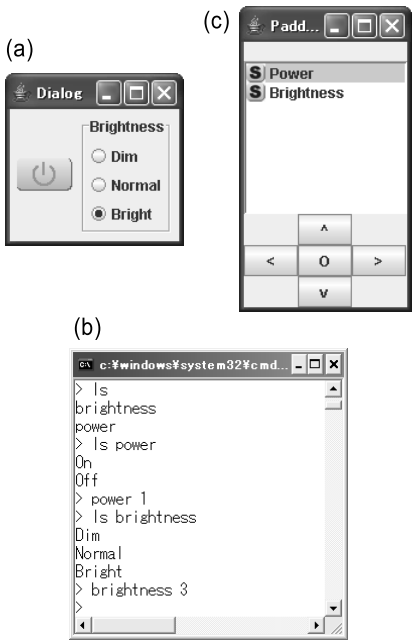

図 5　インタフェース・クライアントの動作画面

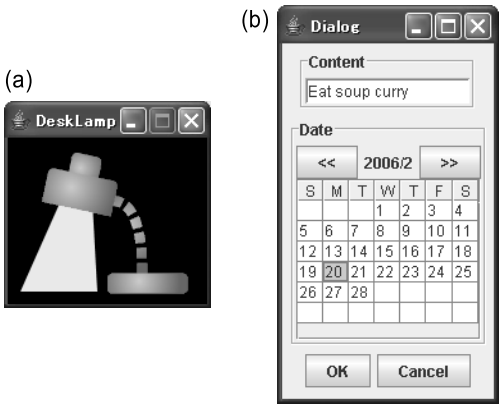

図 6　卓上電気スタンドとスケジュール管理サービスに接続したクライアントの動作画面

参考文献

[1] S. Nylander, M. Bylund and A. Waern: “The ubiquitous interactor–device independent access to mobile services”, CADUI’2004, pp. 274–287（2004）.

[2] 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）.