システムソフトウェアと
オペレーティング・システム 89－15
シ ス テ ム 評 価 2－15
（2002．2．15）

入出力デバイスに非依存なユーザインタフェースの動的生成機構

高橋 元[1] 門田 昌哉[1] 中澤 仁[2] 徳田 英幸[1 2]
[1] 慶應義塾大学 環境情報学部 [2] 慶應義塾大学大学院 政策・メディア研究科

本稿では，ホームネットワークにおける様々なサービスを，様々なエンドユーザ入出力機器から遠隔利用するための機構を提案する．既存の家電機器は，近年の高機能化，多機能化と共にネットワーク接続機能を持つにまでに至っている．ネットワーク接続機能を持つ家電機器を遠隔から利用する場合，遠隔利用のためのソフトウェアをエンドユーザ入出力機器上に配置する必要がある．従来の情報家電機器利用モデルでは，エンドユーザ入出力機器上に配置される遠隔利用のためのソフトウェアに記述される内容は，入出力機器個有の情報を含むものであった．入出力機器個有の情報を含むことによる弊害は，遠隔利用のためのソフトウェアの移植性を損なうことである．本稿で提示する機構では，サービス遠隔利用時にエンドユーザ入出力機器上で稼働するソフトウェアの仕様を入出力機器非依存に記述し，動的にエンドユーザ入出力機器上にユーザインタフェースを生成する．これにより，遠隔利用のためのソフトウェア開発者は，現存する入出力機器や将来登場するであろう様々な入出力機器の差異を意識せず，遠隔サービス利用アプリケーションを開発できる．

Dynamic generation of I/O device-independent user interface

Gen TAKAHASHI[1] Masaya KADOTA[1] Jin NAKAZAWA[1] Hideyuki TOKUDA[1 2]

[1]Faculty of Environmental Information, Keio University
[2]Graduate School of Media and Governance, Keio University

In this paper, we propose a mechanism for carrying out remote control of various appliances in a home network with various operation apparatus. The former have come to have network connectivity with the further advanced features and multi-functionalization in recent years. When operating a networked appliance with network connectivity, it is necessary to arrange the software for remote control on operation apparatus. In the conventional information appliance operation model, the contents described by the software for the remote control arranged on operation apparatus include information depending on operation apparatus. The evil by including the operation aparatus depending information is that the portability of the software for remote control is spoiled. We will show the mechanism in which a networked appliance is operated based on the same software describing the interaction between an user and a appliance from various operation apparatus.

1 はじめに

本論文では，ホームネットワーク上の様々なサービスを様々なエンドユーザ入出力機器から遠隔利用を実現する機構を提案する。

現在の居住環境では、汎用的デスクトップコンピュータ，特定用途のためのプリンタ，AV 機器などの情報家電機器，PDA や携帯電話などの移動端末及び各種センサ等がネットワークに接続され協調動作を行う環境が整いつつある．ユーザは上記環境の中で，入出力機器から目的に応じたサービスを遠隔利用できる．サービスの例として，汎用的用途のデスクトップ PC 上で稼働する電子メール送信プログラムや，AV 機器等の情報家電機器などエンドユーザに利便性を提供するものが挙げられる．エンドユーザ入出力機器は，ユーザがサービスを利用するためのユーザインタフェースを備えられる機器を指し，デスクトップコンピュータや携帯端末，音声コマンドを認識するペット型ロボットなどが例として挙げられる．

このような利用モデルを想定した場合，従来のサービス遠隔利用ソフトウェアの開発において，エンドユーザ入出力機器に配置されるソフトウェアは，入出力機器個有の情報を含むものであった．多様なサービスを多様な入出力機器から遠隔利用する場合，操作機器個有の情報を含むことによる弊害は，遠隔利用のためのソフトウェアの移植性を損なうことである．従って，ある入出力機器用のユーザインタフェースは，仕様の異なる他の入出力機器上では動作せず，ユーザインタフェース開発者は，それぞれの入出力機器のためのユーザインタフェースを個別に開発する必要がある．例えば，特定のツールキットを用いたグラフィカルユーザインタフェース (GUI) を開発した場合，そのユーザインタフェースは他の音声認識しか備えない入出力機器上で利用できない．

本稿では，サービス遠隔利用時に，ユーザインタフェースをエンドユーザ入出力機器上に動的に構築する機構を提案する．本機構において，ユーザインタフェースは，ユーザとサービスのインタラクションを状態遷移をもとに入出力機器に非依存に記述され，サービスと入出力機器間の通信プロトコルも下位通信機構に対して非依存に記述される．

これにより，遠隔利用のためのソフトウェア開発者は，既存の入出力機器や将来登場する様々な入出力機器，及び通信機構の差異を意識せず，遠隔サービス利用アプリケーションを開発でき，開発効率が増す．

2 従来のユーザインタフェース及び通信プロトコル開発の問題

本節では，既存のユーザインタフェース開発，及び通信プロトコル開発の問題点を述べる．

従来のユーザインタフェース開発では，ユーザインタフェース記述に入出力機器個有の情報を含む．入出力機器個有の情報とは，ユーザがどのような行為によって，サービスを利用するのか，及びどのようにそれが実現されるのかという情報である．従来の記述方法の優位な点は，入出力機器に最適な方法で，ユーザに行為を促すことができる点である．不利な点は，サービスまたはエンドユーザ入出力機器数の増加とともにユーザインタフェース記述の総数が増大する点である．サービス数 n，入出力機器数 m とすると，ユーザインタフェース記述の総数は $n \times m$ となる．

また，従来のサービスとエンドユーザ入出力機器間の通信プロトコル開発では，通信プロトコルに，下位の通信機構に個有の情報を含む．個有の情報を含むことによる不利な点は，サービスと入出力機器間の通信において，下位通信機構の変更があると，通信プロトコルは異なり，通信不能となる点である．従って，すべての通信機構上で，サービスと入出力機器間の通信を実現しようとすると，通信機構数 m，サービス数 n であれば，プロトコル総数は $n \times m$ となる．

本研究では，サービスの総数を a，ユーザインタフェース記述，及びプロトコル記述の総数を b とすると，$b = a$ となるシステム，及びその記述方法を提案する．これにより，多様なサービスと多様な入出力機器のシームレスな統合を実現する．

3 システムの要件

前節の目標を達成するため本システムの要件を以下に示す．

ユーザインタフェース記述の入出力機器に対する非依存性

サービス利用のためのエンドユーザ入出力機器は，GUI によるもの，音声コマンドによるもの，手話等を理解する画像認識によるもの，また，それらを組み合わせたものなどが挙げられる．入出力機器非依存なユーザインタフェース記述を実現すれば，入出力機器の差異を意識することなしに，サービス利用のためのユーザインタフェースを記述できる．

プロトコル記述の下位通信機構に対する非依存性

サービスの遠隔利用の場合，サービスとエンドユーザ入出力機器はネットワークを介して通信されるが，通信機構に，RMI や，CORBA[3] が利用されていたり，TCP/IP 上のソケット通信が利用されていたり様々である．非依存性を実現すれば，下位通信機構に対して，同一の記述から通信プロトコルスタックを動的に生成し，エンドユーザ入出力機器とサービス間で通信ができる．

アプリケーション層における通信プロトコルの表現性

下位通信機構に対する非依存性を実現するためには，アプリケーション層における通信プロトコルの表現性を考慮する必要がある．アプリケーション層における通信規約の表現性は，ステートフル，及びステートレスな通信のサポート，プッシュ型，及びプル型の通信のサポートに言い替えられる．

サービスとエンドユーザ入出力機器間の通信は，ステートレスなものとステートフルなものに分けられ，その両者に対応する必要がある．さらに，エンドユーザ入出力機器がサービスからデータを取得する場合，プル 型の通信とプッシュ型の通信に対応する必要がある．プル型の通信とは，エンドユーザ入出力機器からの要求の返答としてサービスはメッセージを送信する．プッシュ型の通信では，サービス側から入出力機器側にメッセージが非同期に送信される．

多様な実行環境への適応性

サービスの遠隔利用のために記述されたソフトウェア (ユーザインタフェースと通信プロトコル) は，多様なプログラミング言語，OS のもとで動作しなければならない．また，生成されるユーザインタフェースは，GUI，音声認識，及び画像認識などの様々な UI ツールキットに対応する必要がある．

4 ユーザインタフェース動的生成機構の設計

本機構のアーキテクチャ概念図を図 1 に示す．

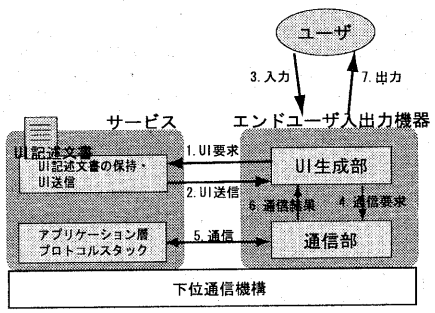

図 1: アーキテクチャ概念図

エンドユーザ入出力機器は，ユーザインタフェース生成部 (UI 生成部) と通信部から構成される．サービス側には，ユーザインタフェース記述言語 (Uniersal Interface Language, 以下 UIL と呼ぶ) が配置される．サービスは，サービス本来の機能の他に，UI 生成部から UI 要求があると，UI 記述文書を要求元に送信する機能，及びエンドユーザ入出力機器との通信機能を持つ．

以下に，ユーザインタフェースの動的生成及び，生成されたユーザインタフェースを用いて，ユーザがサービスを利用する流れを示す．

1. UI 生成部はサービスに対して UI 要求を出す．

2. サービスは，UI記述文書をエンドユーザ入出力機器へ送信する．
3. UI生成部と通信部は，取得したUI記述文書をもとに，UI生成部がユーザインタフェースを生成し，通信部が通信プロトコルスタックを生成する．ユーザは，生成されたユーザインタフェースに対して入力を行う．
4. UI生成部は，入力を保持し，通信部へ通信要求を発行する．
5. 通信部は，生成されたプロトコルスタックを用いて，サービスと通信を行う．
6. UI生成部は，サービスから取得したデータを表示するユーザインタフェースをユーザに提示する．

次に，UI記述文書，ユーザインタフェース生成部，通信部の設計，及びそのインタフェースについて述べる．

4.1 ユーザインタフェース生成部

ユーザインタフェース生成部は，エンドユーザ入出力機器にユーザインタフェース記述文書を読み込み，実際のユーザインタフェースを生成する．

ユーザインタフェース記述文書を読み込んだユーザインタフェース生成部を以下の特徴で定義する．

- 有限個のユーザとサービス間インタラクションの状態 K
- ユーザ及びサービスからの入力 Σ
- 有限個の遷移 Δ
- 開始状態 S (S は K の要素)

ユーザインタフェース生成部は，式 (1) に示すタプル M で記述できる．

$$M = (K, \Sigma, \Delta, S) \tag{1}$$

生成部は，有限個の状態をユーザ及びサービスからの入力と遷移規則によって遷移させることで，ユーザがサービスを対話的に利用するための実際のユーザインタフェース (PDA上のJava Swing Toolkitによるグラフィカルユーザインタフェースや音声認識を利用したユーザインタフェース) を生成する．

次に，生成部における K,Σ,Δ, S，及び遷移のタイミング (遷移イベント) について述べる．

ユーザ・サービス間インタラクションの状態 K

UILでは，ユーザインタフェース記述をユーザとサービス間インタラクションを状態遷移をもとに表す．各々の状態において，どのようなアクションをユーザに促すかをインタラクションとして記述する．また，状態は，ユーザからサービスに対する入力とサービスからユーザへの出力の2つに区分される．ユーザインタフェースはこの状態の集合として表される．以下にUILにおいて記述可能なインタラクション (入力と出力) を示す．

- 入力に関するインタラクション
  項目を選ぶ ユーザに選択を要求する．さらに，複数選択及び単一選択に分けられる．
  文字を入力する ユーザに文字列の入力を要求する．
  連続量を入力する ユーザに正規化された値の入力を要求する．
- 出力に関するインタラクション
  文字列を出力する

図2にCDプレーヤでCDを再生・停止するためのユーザインタフェースの状態，及びその遷移を示す．

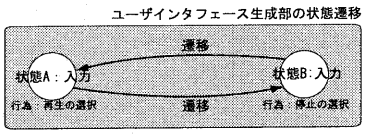

図2: CD Playerのユーザインタフェース状態，及びその遷移

入力状態Aでは，「選択」という行為をユーザに求める．選択項目として，「再生」がある．ユーザインタフェース生成部は，状態Aの時，ユーザに対してCDプレーヤの再生を促すユーザインタフェースを提示する．ユーザが，再生を選択し，実際のCDプレーヤが再生されると，ユーザインタフェースはユーザに停止を促す状態Bに遷移する．状態Bでは，状態Aと同様に「選択」という行為をユーザに求め，選択項目として，「停止」がある．

開始状態 S

Kは，かならず1つの開始状態を持ち，ユーザとサービス間のインタラクションは，この開始状態から始まる．

状態遷移 Δ

入出力機器及びサービスから状態を遷移させることで，ユーザはサービスを対話的に利用できる．遷移規則は，ユーザ及びサービスからの入力と状態の組み合わせであり，以下のように表わされる．

$\Delta[k, \sigma] \rightarrow k'$(ただし k は K の要素)

ある状態 k において，ユーザもしくはサービスからの入力が σ の場合，状態k'へ遷移する．また，状態から状態への遷移が静的な場合，以下のように表わす．

$\Delta[k] \rightarrow k'$

ユーザからの入力 Σ

ユーザの入出力機器に対する入力 Σ は，サービスに対する入力と状態遷移 Δ の一部となるもの，以下で述べる状態遷移イベントを発火するものに分けられる．

状態遷移イベント

状態の遷移は，状態遷移イベントが発火された場合に状態遷移規則をもとに行う．ユーザの入力を，状態遷移のための入力とサービスに対する入力とに区分し，さらに，状態遷移のための入力を，遷移イベントを発火するための入力と遷移規則の入力に区分する．また，遷移イベントは，ユーザの入力によって発火されるだけではなく，サービスからの入力によっても発火される．

状態遷移イベントを発火させるためのユーザによる入力はユーザインタフェース記述文書の記述内容とは

無関係である．ユーザインタフェース生成部に遷移のためのユーザインタフェースとして静的に備わっている．

4.2 通信部

本機構で表現される通信プロトコルは以下の特徴を持つ．

- メッセージパッシングによる通信
- ステートレス/ステートフルのサポート
- プッシュ型とプル型のサポート
- テキストプロトコル

プロトコル記述はメッセージパッシングに基づいて記述される．メッセージパッシングによる通信プロトコルを記述する利点は，下位の通信機構に依存した記述ではない点にある．従って，メッセージパッシングをもとに記述された通信規約は，RMI，CORBA，ソケット通信などの様々な通信機構上で実現可能である．

また，HTTP を代表とするステートレスな通信プロトコルが存在する一方，ステートフルなプロトコルも従来から存在する．これら双方を表現する必要がある．

サービスからエンドユーザ入出力機器が値を取得する方法は，プッシュ型とプル型に分けられる．これら双方を表現する．

通信部は，通信プロトコルが記述されたユーザインタフェース記述文書を読み込み，応用層プロトコルスタックを動的に生成する．

ユーザインタフェース記述文書を読み込んだ通信部の実行時イメージを以下の特徴で定義する．

- 有限個の通信状態の集合 L
- ユーザ及びサービスからの入力 Σ
- 有限個の遷移 Δ
- セッション開始状態 S (S は L の要素)
- セッション終了状態 E (E は L の要素)

通信部は，式 (2) に示すタプル N で記述できる．

$$N = (L, \Sigma, \Delta, S, E) \tag{2}$$

通信部は，有限個の通信状態をユーザ及びサービスからの入力と遷移規則によって遷移させることで，エンドユーザ入出力機器がサービスと遠隔通信するためのアプリケーションプロトコルを処理する．

次に，通信部における L,Σ,Δ 及び遷移のタイミング (遷移イベント) を述べる．

通信状態 L

本機構では，メッセージパッシングによる通信をサポートするが，ユーザインタフェース記述文書において，通信規約は通信状態の集合として記述される．通信状態は，メッセージ送信状態とメッセージ待ち受け状態の 2 つに区分される．通信規約は，これらの状態の集合として表される．これらの状態を遷移させることで，ステートフルな通信規約も，ステートレスな通信規約も処理できる．

送信状態では送信メッセージの内容を記述する．受信状態では，受信メッセージの型を記述する．型情報は，制御コードのバイト数，メッセージの終端コード，及び受信データの参照名である．受信データの参照名とは，項??で述べる共有空間に対してメッセージのヘッダ，及び本文が名前と値の組で書き込まれる場合の名前に相当する．

サービスからの入力 Σ

本機構では，メッセージパッシングによる通信をサポートし，サービスからの入力はメッセージとして扱われ，制御コード，本文，終端コードの構造を持つ．また，メッセージは，ユーザインタフェースに反映されるもの，遷移規則になるものに分けれらる．

状態遷移 Δ

サービス，エンドユーザ入出力機器間の通信は，通信部における通信状態が遷移していくことで行われる．遷移規則は，ユーザ及びサービスからの入力と状態の組み合わせであり，以下のように表わされる．

$\Delta[m, \gamma] \rightarrow m'$(ただし m は L の要素)

ある状態 m においてて，ユーザもしくはサービスからの入力が γ の場合，状態 m' へ遷移する．また，状態から状態への遷移が静的な場合，以下のように表わされる．

$\Delta[m] \rightarrow m'$

状態遷移イベント

状態の遷移は，状態遷移イベントが発火された場合に状態遷移規則をもとに行われる．実行時において，通信状態は通信中状態と完了状態の 2 つに区分され，完了状態になった場合，遷移イベントが発火される．送信状態では，メッセージの送信が完了した場合，受信状態では，メッセージの受信が完了した場合に，遷移イベントが発火される．

4.3 ユーザインタフェース記述文書

ユーザインタフェース記述文書とは，サービスの遠隔利用のための入出力機器非依なユーザインタフェース，及びサービスと遠隔通信するための通信機構非依存なアプリケーション層通信プロトコルを記述したものである．以下にユーザインタフェース記述文書におけるユーザインタフェース記述，及びプロトコル記述について述べる．

ユーザインタフェース記述

項 4.1 で述べた，K, Σ, Δ の記述は以下の XML タグで表す．

K <state>タグの集合として表し，その内部にユーザへ促す行為を記述する．

Σ ユーザの入出力機器に対する入力 Σ は，サービスに対する入力，遷移規則 Δ となる入力，または状態遷移イベントを発火する入力に分けられる．ユーザインタフェース記述には，サービスに対する入力と遷移規則となる入力を記述する．入力は，「参照名」と「値」の組として記述し，<var>タグによって表す．値が実行時に決定するものは，参照名のみを記述する．

Δ 遷移 Δ は，<rules>タグで表し，遷移規則は，<rule>によって表す．また，ユーザ，及びサービスからの入力によって遷移が動的に変わる場合，入力された値への参照名と値を記述する．

プロトコル記述

項 4.2 で述べた L,Σ,Δ の記述は，以下の XML タグで表す．

L <state>タグの集合として表す．L は，受信状態，または送信状態に区分される．よって，<send>，または<receive> によってその区別をする．

Σ サービスからの入力 Σ は，<message>タグによって表す．記述は，実際の入力に対する「参照名」を記述する．サービスからの入力は，メッセージであり，項 4.2 の構造を持つ．記述時に静的に決定する終端コード以外の参照名を記述する．

Δ 遷移 Δ は，<rules>タグで表し，遷移規則は，ユーザインタフェースの記述と同様に<rule>によって表す．

また，本記述による電灯制御サービスのユーザインタフェース記述文書例を第節 5 で述べる．

4.4 ユーザインタフェース生成部，通信部間インタフェース

本項では，ユーザインタフェース生成部から通信部対する通信要求の発行方法，ユーザインタフェース生成部の通信部が受信したメッセージに含まれる値の取得方法について述べる．ユーザインタフェース生成部及び，通信部は，互いの状態を遷移させる．例えば，ユーザインタフェース生成部がユーザの入力を受け取り，サービスと通信の必要があれば，通信部をある状態に遷移させることで通信要求を通信部に発行する．

通信部がサービスからメッセージを受け取った場合，通信部は，ユーザインタフェース生成部がある状態に遷移することでサービスから取得した値をユーザインタフェースに反映する．値の受け渡し方法は，ユーザによって入力された値をサービスが必要とする場合や，サービスから取得した値をユーザに提示する場合がある．それらの場合，値は，ユーザインタフェース生成部から通信部へ渡される必要があるが，値は直接渡されず，いったんユーザインタフェース生成部と通信部両モジュールの共有空間へ書き込まれる．共有空間へは，名前と値の組で書き込まれる．ユーザインタフェース生成部及び，通信部は，共有空間を参照し，ユーザが入力した値，及びサービスから取得した値を各々得ることができる．

また，遷移規則に関してもユーザからの入力やサービスから取得した値を必要とする場合 (ユーザの入力もしくは，サービスからの入力が遷移規則の一部である場合) もこの共有空間を介して値を取得する．

4.5 データ型式

本機構で，扱う値はすべてテキストデータである．ユーザからの入力およびサービスからの入力は文字列として扱われる．データは，ベクタ型とスカラ型に区分され，ベクタ型は，文字列を 1 次元配列に要素化でき，ユーザインタフェースにおいて反復的な表現が可能である．

5 UIL の応用システム実装例

本節では，入出力機器非依存なユーザインタフェース記述文書の実装として，Universal Access Language(UIL) について述べ，UIL を用いたサービス遠隔利用アプリケーション例として，電灯制御サービスの遠隔利用を示す．

5.1 実装環境

エンドユーザ入出力機器

GUI を備えた PDA 上で本機構を実現した．

実装言語，及びライブラリ

実装は，Java 言語によって行った．Java Virtual Machine は Sun microsystems J2SDK 1.3.1 を使用した．開発時には以下の外部ライブラリを利用した．

- JAXP(XML パーサ)
- Jakarta oro[6](正規表現処理)

下位通信機構

エンドユーザ入出力機器とサービス間は，TCP/IP ネットワークによって，接続されている．

電灯制御サービス

電灯制御サービスとは，室内の電灯の点灯，及び消灯を行うものであり，室内には，2 つの電灯が配置されているものとする．エンドユーザ入出力機器を用いて，ユーザは 2 つの電灯の消灯，および点灯を行う．

5.2 電灯制御サービスのプロトコル

電灯制御サービスは，TCP/IP ネットワーク上で図 3 のプロトコルを提供し，ステートレスな通信を行う．

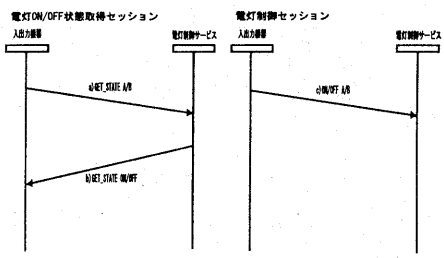

図 3: 電灯制御サービス

5.3 電灯制御サービスにおける状態遷移

電灯制御サービスに対するユーザインタフェース部，および通信部の状態遷移を図 4 に示す．

ユーザインタフェース状態 START(開始状態) では，ユーザに，制御する電灯の選択を促す．ユーザは電灯を選択し，状態を次に遷移させる (a)．通信状態 GET STATE では，ユーザによって選択された電灯の状態を

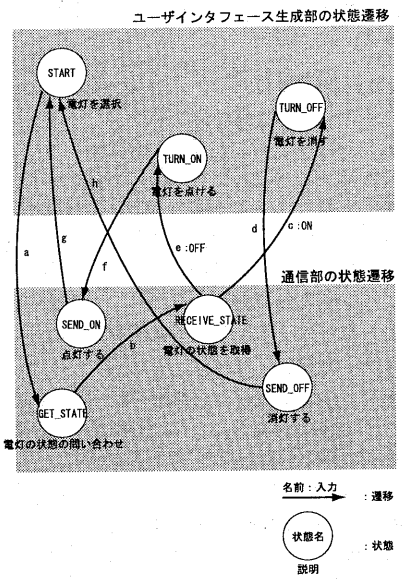

図 4: 電灯制御におけるユーザインタフェース部，および通信部の状態遷移

サービスに問い合わせる．メッセージの送信が完了すれば，RECEIVE STATE へ遷移 (b) し，サービスからの結果を待ち受ける．メッセージの受信が完了すれば，受信メッセージの種類に応じてユーザインタフェース状態 TURN ON，もしくは TURN OFF に遷移する (c および e)．ユーザに，電灯の点灯，もしくは消灯の確認を促す．さらに，ユーザの入力により，通信状態 SEND ON，もしくは，SEND OFF へ遷移 (f および d) する．通信状態 SEND ON，および SEND OFF では，サービスに，点灯，もしくは消灯のメッセージを送信する．送信が完了すれば，開始状態へと戻る (h および g)．

5.4 ユーザインタフェース状態記述

UIL による電灯制御サービスのユーザインタフェース記述を図 5 に示す．

図 4 における状態 (START, TURN ON, TURN OFF) がユーザインタフェース生成部の状態であり，実際に生成されるユーザインタフェースの状態である．図 5 において，<state>タグで囲まれる部分が，ユーザインタフェースの状態となる．この<state>タグの中で，ユーザに促す操作 (インタラクション) を記述する．状態 START では，ユーザに，点灯/消灯をする電灯の選択を促す．また，<var>タグによって，ユーザの入力を指定する．ユーザが，項目を選択した場合，共有空間 (項??) に名前と値の組で書き込まれる．

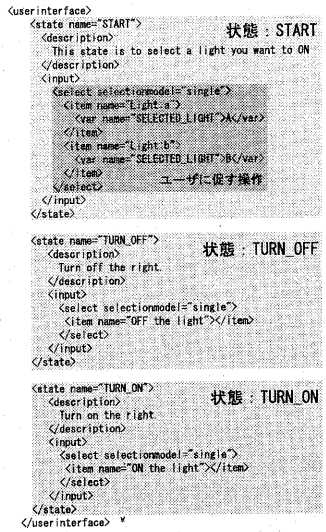

図 5: ユーザインタフェースの状態記述

5.5 プロトコル状態記述

UIL によるエンドユーザ入出力機器電灯制御サービス間通信のためのプロトコル記述を図 6 に示す．(サービスが提供する TCP/IP 上のプロトコルに関しては，図 3 を参照)

図 5 における，状態 (GET STATE, RECEIVE STATE, SEND ON, SEND OFF) が通信部の状態である．通信部は，この 5 つの状態を遷移させることでサービスと通信を行う．図 6 における<state>タグで囲まれる部分が，プロトコルスタックの状態となる．<send>タグでは，<message>タグによって，送信するメッセージを記述する．状態 GET STATE 内の<body>タグにおける${SELECTED LIGHT} の記述は，共有空間 (項??) 内の SELECTED LIGHT という名前に対応する置換されることを意味する．<receive>タグでは，予想される受信メッセージの型 (項 4.2) を<message>タグに記述する．

5.6 遷移記述

UIL によるユーザインタフェース，および通信規約の状態遷移規則を図 7 に示す．図 4 における，矢印 (a, b, c, d, e, f, g, h) が，遷移である．矢印は，図 7 における<rule>タグによって表現される．例えば，遷移規則が，

$[RECEIVESTATE, ON] \rightarrow TURN_OFF$

$[RECEIVESTATE, OFF] \rightarrow TURN_ON$

の場合は，図 7 における遷移 c，および遷移 e と記述される．実行時の入力値 (ON，または OFF) は，共有空間 (項??) の CURRENT STATE の値が参照される．

```
<protocol type="stateless">
 <state name="GET_STATE">          送信状態:GET_STATE
  <send>
   <message>
    <head chars="10">GET_STATE </head>
    <body>${SELECTED_LIGHT} </body>
   </message>
  </send>
 </state>

 <state name="SEND_ON">            送信状態:SEND_ON
  <send>
   <message>
    <head chars="10">ON         </head>
    <body>${SELECTED_LIGHT} </body>
   </message>
  </send>
 </state>

 <state name="SEND_OFF">           送信状態:SEND_OFF
  <send>
   <message>
    <head chars="10">OFF        </head>
    <body>${SELECTED_LIGHT} </body>
   </message>
  </send>
 </state>

 <state name="RECEIVE_STATE">      受信状態:RECEIVE_STATE
  <receive>
   <message>
    <head chars="10">HEAD       </head>
    <body>CURRENT_STATE</body>
    <end_code>.</end_code>
   </message>
  </receive>
 </state>
</protocol>
```

図 6: 通信プロトコルの状態記述

```
<transition_rules>
  <rule>                                  遷移 : a
    <state>START</state>
    <transition>GET_STATE</transition>
  </rule>

  <rule>                                  遷移 : b
    <state>GET_STATE</state>
    <transition>RECEIVE_STATE</transition>
  </rule>

  <rule in="CURRENT_STATE">               遷移 : c
    <state>RECEIVE_STATE</state>
    <in>ON</in>
    <transition>TURN_OFF</transition>
  </rule>

  <rule in="CURRENT_STATE">               遷移 : e
    <state>RECEIVE_STATE</state>
    <in>OFF</in>
    <transition>TURN_ON</transition>
  </rule>

  <rule>                                  遷移 : d
    <state>TURN_OFF</state>
    <transition>SEND_OFF</transition>
  </rule>

  <rule>                                  遷移 : f
    <state>TURN_ON</state>
    <transition>SEND_ON</transition>
  </rule>

  <rule>                                  遷移 : g
    <state>SEND_ON</state>
    <transition>START</transition>
  </rule>

  <rule>                                  遷移 : h
    <state>SEND_OFF</state>
    <transition>START</transition>
  </rule>
</transition_rules>
```

図 7: 状態遷移

6 関連研究

ICrafter[1] は，SDL をもとに，エンドユーザ入出力機器におけるユーザインタフェースを動的に生成する機構を実現している．SDL では，主に以下の内容を記述する．

- サービスの型
- オペレーション
- オペレーションのパラメータの型

ICrafter は，SDL に記述された内容を元に，実際のユーザインタフェースである HTML，VoiceXML[5]，MoDAL[4] 等を生成する．

また，UIML[2] では，入出力機器非依存なユーザインタフェースの記述を XML で行い，他のマークアップ言語 (HTML や VoiceXML など) やプログラミング言語のツールキットにスタイルシートを用いて対応させる．

次に，サービス記述によるユーザインタフェースの生成，および他の入出力機器非依存のユーザインタフェース記述との差異を述べる．

サービスの状態に適合したユーザインタフェースの提示

SDL などのサービス記述言語から，ユーザインタフェースを生成機構と，UIL による生成機構の差異は，ユーザにサービスの状態に適合したユーザインタフェースを提示できる点にある．また，本機構は，サービスの状態とユーザインタフェースの状態の整合性の維持を保証しない．従って，サービス提供者が整合性を維持のための通信プロトコルを提供し，ユーザインタフェース開発者が，整合性を保てるユーザインタフェース，およびプロトコル記述を行う必要がある．

CD プレーヤを操作するためのユーザインタフェースを例にとる．CD プレーヤは，停止状態と再生中状態をもち，CD を再生する機能と CD を停止する機能をユーザに提供しているとする．CD プレーヤが停止状態にあれば，ユーザインタフェース側では，ユーザに再生の機能を提示する必要はあるが，停止の機能を提示しても無意味である．状態遷移を元にユーザインタフェースを記述すれば，サービスの状態を反映したユーザインタフェースを生成することができる．

遠隔利用のための統合的記述

UIML 等の入出力機器非依存なユーザインタフェース記述は，入出力機器非依なユーザインタフェースの記述にその主眼がある．本研究では，サービスの遠隔利用ソフトウェアの開発という観点から，下位通信機構に非依存な通信プロトコルの記述の実現，およびそのユーザインタフェースとのシームレスな結合を行うことができる．

7 評価

本機構のプロトタイプ実装におけるユーザインタフェース生成，およびプロトコルスタック生成に関するオーバーヘッドの検証のための定量的な評価を行う．

7.1 評価環境

本機構を AMD Athlon Thandurbird 1GHz プロセッサ，Linux カーネル 2.4.14，JDK 1.3.1 上で稼働させ評

価を行う．ユーザインタフェース生成に関しては，Swing Toolkit で行った．通信プロトコルスタック生成に関しては，TCP/IP 上で行った．

7.2 UI 生成におけるオーバヘッドの検証

UIL の取得からユーザインタフェース生成までには，以下の過程がある．

1. DOM の生成
2. DOM 走査による DOM ノードから状態オブジェクトへのインスタンス化，および状態オブジェクトのコレクションオブジェクトへの追加
3. 開始状態のユーザインタフェースの生成
4. 描画，発話等

オーバヘッドの検証では，過程 1 から 3 における処理時間を検証する．読みこむ XML は，図 5，6，7 に示した XML を使用した．それぞれの過程に要する時間を 100 回計測した平均を表 7.2 に示す．(XML は，測定マシンのハードディスクに配置され，DOM の生成は，この読み込み時間も含む．また，XML は 2600 バイトである．)

表 1: 各過程の処理時間の平均値

	平均値 (ms)
DOM 生成	268.23
DOM Tree 走査	32.20
UI 生成	46.30

処理時間が最大となるのは DOM の生成であった．ユーザインタフェース生成時間の改善のために，DOM をシリアライズ，および入出力機器へのキャッシュの検討を考慮している．

ユーザインタフェース生成時間は，平均で 46.30 ms であった．これは 1 秒間に約 21 回ユーザインタフェースを表示できることを意味する．一般の情報家電サービス遠隔利用モデルにおいて，この数値は本機構が十分な性能を持つことを表す．

7.3 通信におけるオーバーヘッドの検証

電灯制御サービスに電灯の状態を問い合わせるメッセージ通信 (図 6) の過程を以下に示す．

1. 送信メッセージのパース
2. ソケットのオープン，およびメッセージの書き出し
3. メッセージの読み込み

通信におけるボルトネックの検証では，過程 1 から 3 における処理時間を検証する．読みこむ XML は，図 5，6，7 に示した XML を使用した．それぞれの過程に要する時間を 100 回計測した平均を表 7.3 に示す．(使用した XML 文書は，ユーザインタフェース生成におけるものと同様である)

全体の処理のほぼ半分に送信メッセージのパースとメッセージの読み込みが占めていた．送信メッセージのパース過程では，正規表現ライブラリ Jakarta Oro を使用した文字列パースを行っている．一連のメッセージの送受信に平均で 184 ms かかっており，1 秒間に約 5 回のメッセージ送受信が行えることを意味する．ユーザの操作によって，1 秒間に 5 回のメッセージ送受信を行えることは，情報家電機器遠隔利用において，本機構の性能は許容できるものと言える．

表 2: 各過程の処理時間の平均値

	平均値 (ms)
送信メッセージのパース	96.86
メッセージの書き出し	1.82
メッセージの読み込み	85.32

8 まとめと今後の課題

本稿では，多様なサービスを多様なエンドユーザ入出力機器から利用する機構を提案し，設計，実装および評価を行った．現段階では，家庭内ネットワークを想定してきたが，オフィスネットワークや組織内ネットワーク，工場内ネットワークなどより広範囲なスケールへ応用範囲を拡げることが考えられる．その場合，複数のサービスに対して統合なユーザインタフェースをユーザインタフェースの合成およびサービスの合成として考慮する必要がある．同型のサービスに対して，同様な制御を一度に行いたい場合や別種のサービスを平行的に利用したい場合に適したユーザインタフェース生成および通信を実現しなければならない．

また，利用モデルを単一のユーザによるものに限ったが，複数のユーザによるサービスの利用モデルも考え，既存の認証機構やアクセスコントロール機構を組み込む必要もある．

謝辞

本研究を進めるにあたり，慶應義塾大学徳田研究室の皆様に多大な御助言，御協力を頂きました．特に KMSF 研究グループの皆様には数多くの議論をして頂き，深く感謝致します．

参考文献

[1] Ponnekanti,S.R.,Lee,B.,Fox,A., Pat Hanrahan, and Winograd, T.: ICrafter:A Service Framework for Ubiquitous Computing Environments, Ubicomp (2001)

[2] UIML Project, http://www.uilm.org

[3] Object Management Group. Commom Object Request Broker Architecture, http://www.omg.org

[4] Eustice, K.F.,Lehman, J.T.,Morales A.,Muson, M.C.,Edlund,S.,Guillen, M.: A Universal Information Appliance, IBM System Journal(1999) http://www.almaden.ibm.com/cs/TSpaces/MoDAL

[5] Voice eXtensile Markup Language, http://www.w3.org/TR/voicexml

[6] Jakarta Project. Jakarta Oro, http://jakarta.apache.org/oro