2008－EMB－10 (8)
2008／11／28

RESTアーキテクチャを用いた統合型機器連携手法の提案

植田健太† 小板隆浩†† 佐藤健哉†

近年ネットワークに対応した機器が増加しており，ネットワークに対応した機器を操作する研究や，ネットワークインタフェースを持たない機器をネットワークに対応させる研究などが行われている．今後，様々な機器やWebサービスが連携し，動作する環境が望まれる．また，ユーザにとっては，Webサービスや機器を自由に組み合わせることが望まれる．しかし，現在，様々な技術仕様が存在するため，機器を統一したインタフェースで連携させることは難しい．本論文では機器とWebサービスとの連携を考慮し，RESTアーキテクチャに従った機器連携手法を設計する．そして，この設計に従ったWebインタフェースを用いることで，ネットワークインタフェースを持たない機器，ネットワークに対応した機器，インターネット上に公開されているWebサービスそれぞれの技術仕様が互いに連携できる手法を提案し，その有用性を議論する．

Proposal of Integrated Device Cooperation Method with REST Architecture

KENTA UEDA,† TAKAHIRO KOITA†† and KENYA SATO†

This paper proposes a device cooperate system which provides web service interface using REST architecture. The proposed system has two characterisitcs. First is a device cooperation with not only a device but also web servivces. Second is the proposed system in designed based on REST architecture. The design can flexibly cooperate among devices and web services.

1. はじめに

近年，コンピュータの小型化，高性能化により機器に様々な機能を持たせることが可能となった．それに伴い，ネットワークに対応した機器が増加しており，場所にとらわれず機器の操作を行う環境が整いつつある．これにより，ネットワークに対応した機器同士を連携させる研究[1]や，ネットワークインタフェースを持たない機器をネットワークに対応させる研究[2]などが行われている．また，機器連携だけでなく，機器とWebサービスとの連携を考慮した研究[3)~5)]も行われている．これらの研究から，今後ネットワークインタフェースを持たない機器だけでなく，ネットワーク対応機器，Webサービスといった異なる技術仕様それぞれが互いにネットワークを通じて連携する環境の実現が望まれる．例えば，朝目覚まし時計が鳴ると，トースターがパンを焼きはじめ，テレビがつき，あらかじめWebのストレージに保存しておいた放送を再生する，といった機器の組み合わせをユーザが自由に設定できる環境である．

現在，機器同士連携し，動作させる方法はDLNA (Digital Living Network Alliance)[6]ガイドラインやDLNAガイドラインのベースとなるUPnP (Universal Plug and Play)[7]，Jini[8]，Havi[9]等の技術仕様を用いる方法や，独自の仕様でプログラムを作成する方法等がある．しかし，これらの技術仕様は，ネットワークに対応した機器を中心に考えられている．そのため，ネットワークインタフェースを持たない機器が数多く存在する現状では，異なる技術仕様それぞれが互いにネットワークを通じて連携する環境の実現は難しい．そして，独自の仕様でプログラムを作成する場合では，異なる技術仕様に対応させることや連携を考えると，独自の仕様では実現することは容易ではない．

Webサービスに関しては，Web API公開という形で公開されているサービスがあり，Web開発者はWeb API(Web Application Program Interface)を用いることで，様々なWebサービスを利用して新たなサービスを開発するといった連携が可能となっている．ま

† 同志社大学大学院工学研究科
Graduate School of Sciences and Engineering, Doshisha University
†† 同志社大学理工学部
Faculty of Sciences and Engineering, Doshisha University

た，家電をネットワーク化する際の技術の現状と標準化に向けた課題に関する調査[10]によると，様々な機器を連携するための方針として，インターネット標準を用いることと，サービス中心にプラットフォームを構築することが必要であると述べている．これより，様々な機器を連携する方法として，機器を Web サービスとして実装することが望ましい．

ユーザ側の立場で考えると，Irene Mavrommati[11]は，ユーザは開発者が意図した使用方法を越えた使い方を考え出すことがあると述べている．このことは，機器の使用方法についてもいえると考えられ，機器の使用方法を限定したインタフェースを決めてしまうのではなく，ユーザが使用時に使用方法を決定できることが望ましい．そのためには，機器操作のためのインタフェースをシンプルかつ柔軟に設計する必要がある．

本稿では，機器を REST (Representational State Transfer)[12),13)]に従ったアーキテクチャ (REST アーキテクチャ) を用いた Web サービスとして扱うことで，統一したインタフェースで，ネットワークインタフェースを持たない機器や，ネットワークに対応した機器 (以後特に区別する必要がある時以外は機器とする)，Web サービスなど異なる技術仕様を統合し互いに連携させ動作する手法を提案する．機器を REST アーキテクチャに従った Web サービスとして扱うことで，機器を Web API として利用し，機器や Web サービスとの連携を可能とする．これにより，異なる技術仕様それぞれが互いにネットワークを通じて連携する環境の実現や，機器操作のためのインタフェースをシンプルかつ柔軟に設計することが可能となる．REST はインターネット標準を用い，サービス中心に Web サービスのインタフェースをシンプルに実装することができる．本稿での機器連携とは，機器同士が通信するためのインタフェースを用いて通信，動作することとし，ユーザがそれぞれの機器に対して制御命令を送信して操作することとは区別する．本稿で提案する，機器を Web API として公開することで，機器同士や Web サービスとの連携を可能とし，組み合わせをユーザが自由に設定できる手法について述べ，その有用性について議論を行う．

2. 関連研究

本章では，既存の機器連携に関する研究と，機器操作や機器連携に関する既存の方式や標準化が進められている技術仕様についての特徴を述べ，機器や Web サービスそれぞれがお互いに連携し動作する環境を実現する際の問題点を明確にする．

2.1 既存の機器連携

ネットワークインタフェースを持たない機器をネットワークに対応させる研究は，増渕らの XML を用いた遠隔機器制御支援システム[2]がある．このシステムは独自の仕様を用いてネットワークインタフェースを持たない機器にプロトコル変換を行う機器を接続し，プロトコル変換を行う機器を通してネットワーク接続を可能とするものである．また，機器の情報や，機器操作の定義を XML を用いて表現することで，様々な機器に対応することが可能となっている．本稿で実現したい環境では，機器操作だけでなく，機器同士が連携することが必要となる．

機器と Web サービスとの連携を考慮した研究に，小澤らの CASTANET を用いて REST アーキテクチャ基づいたフレームワーク[3]がある．CASTANET とは情報家電などを相互に連携させたサービスを，簡単かつスケーラブルに構成することができるフレームワークである．このフレームワークでは，CASTANET を用いてセンサ情報を REST アーキテクチャで表現し，センサ情報をデータベースに格納する Web サービスとの連携を行っており，機器からは情報を取り出すことのみの連携となっている．本稿で実現したい環境では，機器から情報を取り出すことに加えて，操作，または Web サービスから情報を取得しての動作についても可能とする必要がある．

2.2 機器連携に関する問題点

2.2.1 機器操作の技術仕様の差異

機器操作のための技術仕様はそれぞれ互換性は無く各技術仕様間をまたいで機器を連携することはできない．複数の技術仕様が混在する環境において，それぞれの機器を連携して扱うには各技術仕様間の差異を吸収する仕組みが必要となる．

2.2.2 Web サービスとの連携

機器と Web サービスを連携させるには，どちらかが一方のインタフェースに合わせる必要がある．Web サービスをそれぞれの技術仕様に合わせることは柔軟性や，Web サービス自体の機能を果たさなくなる点から不適である．これより，それぞれの技術仕様が Web サービスに合わせる必要があり，それぞれの技術仕様の差異を吸収した上で Web サービスのインタフェースを持たせる必要がある．

3. REST アーキテクチャを用いた統合型機器連携手法

3.1 提案手法概要

本稿では，機器を REST に従ったアーキテクチャ

(REST アーキテクチャ) を用いた Web サービスとして扱うことで，統一したインタフェースで機器，Web サービスなど異なる技術仕様を統合し互いに連携させ動作する手法を提案する．本提案手法は，様々な技術仕様を用いた機器を REST アーキテクチャに従った設計の Web サービスとして扱い，機器同士や Web サービスと連携し動作させ，ユーザが機器や Web サービスの組み合わせを自由に設定することが可能となることを目的とする．

3.2 REST の特徴

REST は Roy Fielding が書いた博士論文[12] で初めて述べられたアーキテクチャスタイルで，HTTP や URI 等の Web 標準をどのように利用されるべきかを定めている．REST では，リソースへのアクセス方法を指定する際の情報として，メソッド情報とスコープ情報の 2 つの情報を定義しており必要な情報を全て HTTP メッセージに含める設計となっている．また，リソースに対しては一意な URI を割り当てる設計となっている．本稿ではリソースを，機器や Web サービスの中の 1 つのサービスを表すこととし，サービスの 1 機能をメソッドとする．HTTP メソッド情報とは，データの操作に関する情報である．現時点では GET, PUT, POST, HEAD, DELETE，OPTIONS の 6 つが一般的である．HTTP メソッド名の利点は標準化されていることである．スコープ情報とは，ユーザがサーバのどのデータを操作するのかを示す情報である．例えば，

http://www.examople.com/search?q=REST

という URI では，メソッド情報は「GET」で，スコープ情報は「/search?q=REST」となる．このように，REST ではメソッド情報とスコープ情報を用いてリソースへのアクセス方法を定義することで，統一インタフェースで，かつアドレス可能性を実現できる．本提案手法では，機器のリソースへのアクセス方法をメソッド情報とスコープ情報を用いて表現する．

3.3 提案手法のシステム構成

提案手法の構成例を**図 1** に示す． 図 1 では，一般家庭と，Web サービス提供者に分かれており，一般家庭では機器を扱い，Web サービス提供者では Web サービスを提供している．図では一般家庭は一つしか表示していないが，一般家庭はいくつあってもかまわない．一般家庭内では，機器連携サーバを設置し，機器連携サーバを通じて機器の連携を行う．ネットワーク対応機器は，それぞれが用いる通信の仕様を用いて他の機器と通信を行い，ネットワークインタフェースを持たない機器は，接続している PC や PLC を通じて他の機器と通信を行う．Web サービス提供者が公開する Web サービスが本提案手法と連携するためには，REST アーキテクチャに従った設計で作られている必要がある．機器連携サーバでは，ネットワーク対応機器を REST アーキテクチャに従ったインタフェースに置き換えて扱い，ネットワークインタフェースを持たない機器においては，接続している PC (機器連携サーバに接続しているならば機器連携サーバ) で REST アーキテクチャに従ったインタフェースで扱うこととする．

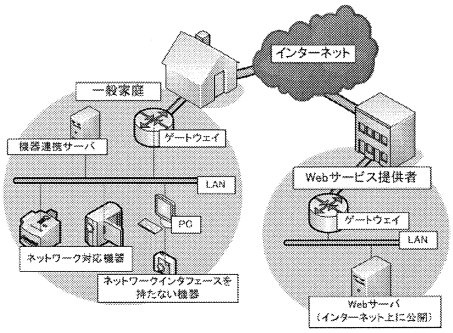

図 1 システム構成例
Fig. 1 System Overview

3.4 HTTP メッセージ

本提案手法での各メソッド情報についての役割を，HTTP の定義に基づき以下のように定める．

GET リソースを取得する
PUT リソースを変更する
DELETE リソースを削除する
HEAD メタデータ固有の表現を取得する
OPTIONS 特定のリソースがサポートする HTTP メソッドを調べる
POST リソースを新規作成する

この定義に基づいて HTTP メッセージを送信することで，インタフェースを統一することができる．また，URI については，次の規則を基本とする．

http://アドレス/名前/リソース名/メソッド/値

アドレスは，機器連携サーバや PC，Web サービスのサーバ等といった REST 設計に基づいてリソースを提供するコンピュータへアクセスするためのアドレスである．名前とは，機器連携サーバや PC，Web サーバの名前や建物の名前，サービス名など，識別しやすい名前であればよい．また，一意である必要はない．リソース名とは，提供する機器や Web サービスの名前である．メソッドとはリソースの 1 機能を表す．値はメソッドに渡す値とする．例を挙げると，REST ルームにあるライトを点灯・消灯させるという命令を送る

場合には，

http://www.restroom.com/RESTRoom/Light/switch

といったアドレスを PUT メソッドで送信する．また，機器同士連携させる場合には，

http://www.restroom.com/RESTRoom/Light/switch?method=put&next=http%3A%2F%2Fwww.restcastle.com%2FRESTCastle%2FLight%3Fswitch

といったように URI の最後に次の機器を操作するための URI と HTTP メソッドの種類を記述する．上記のように next=と記述した場合は機器の操作をしてから次の機器へ命令を送るといった流れになる．さらに，Web サービスと連携する場合には以下のように記述する．

http://www.restroom.com/RESTRoom/TV/Slideshow?method=get&get=http%3A%2F%2Fwww.restphoto.com%2FRESTPhoto%2Fkueda%3Falbum

この場合は，先に Web サービスから情報を取得し，その後その情報を機器に渡すことで，Web サービスの連携を実現する．認証が必要な Web サービスに関しては Web サービス側から認証情報を含んだ一時的な URI を生成し，そのアドレスに対してアクセスする方針をとる．

機器や Web サービスの機能によっては上記の規則では表現できない場合がある．そのため，上記の規則は任意とし，機器や Web サービスを HTTP メソッドや URI で表現することを優先する．

3.5 WADL による機器情報の定義

クライアントはそれぞれの機器連携サーバや Web サーバが提供するサービスや機能等を知る必要がある．そのために，それぞれの機器連携サーバや Web サービスがあらかじめ提供するサービスを記述したデータを保持しておく必要がある．そこで，WADL (Web Application Description Language)[14] を用いて REST の HTTP メソッドや URI，リソースの説明などを XML で記述し，クライアントが要求してきた際に WADL ファイルを返す．これにより，クライアントはどのような HTTP メッセージを送信すればよいかを知ることができる．REST ルームにあるライトの明るさの指定や状態を取得するサービスの WADL の記述例を**図 2** に示す．この WADL ファイルは，それぞれの機器連携サーバや Web サーバのトップディレクトリに置くこととする．クライアントは，“http://アドレス/WADL ファイル名” にアクセスすることで WADL ファイルを読み込む．

WADL ファイルのルートノードは application タグとなる．そして次にリソースの定義を行う．リソースの定義では，WADL があるサーバが扱うすべての機器や Web サービスの URI を定義している．機器や Web サービスの種類，機能，操作命令と徐々に定義の階層が深くなる．そして，各機能の定義では呼び出すメソッドを定義している．リソースの定義が終わると，次はメソッドの記述を行う．先ほど定義したメソッドがどの HTTP メソッドを受け取り，どのような値を返すかを定義する．最後に表現の定義を行う．表現の定義とは，メソッド呼び出しの後に返す値の定義や，メソッド呼び出しの際の値の定義などを行い，メディアタイプや，パラメータを定義する．この表現の部分

```
<?xml version="1.0" encoding="utf-8"?>

<application xmlns="http://research.sun.com/wadl/2006/10"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:light="https://www.restroom.com/LightState.xsd">

   <grammers>
   <include href="https://www.restroom.com/LightState.xsd" />
   </grammers>

   <!--リソース-->
   <resource base="https://www.restroom.com/">
   <doc xml:lang="ja" title="restroom API v1">
      REST ルームにある照明の調節をしてください.
   </doc>
   <resource path="RESTRoom">
      <param name="Authorization" style="header" required="true">
         <doc xml:lang="ja">
           REST ルームの API 呼び出しは全て HTTP の Basic 認証を必要とする
         </doc>
      </param>
      <resource path="Light">
         <resource path="Switch">
            <method href="#putState"/>
            <method href="#getState"/>
         </resource>
      </resource>
   </resource>
   </resource>

   <!--メソッド-->
   <method id="getState" name="GET">
   <doc xml:lang="ja" title="ライトの状態を返す" />
   <response>
      <representation href="#LightState" />
      <fault id="AuthrizationRequired" status="401" />
   </response>
   </method>
   <method id="putState" name="PUT">
   <doc xml:lang="ja" title="ライトの状態を変更する" />
   <request>
      <representation href="#LightState" />
   </request>
   <response>
      <representation href="#LightState" />
      <fault id="AuthorizatiinRequired" status="401" />
   </response>
   </method>

   <!--表現-->
   <representation id="LightState" mediaType="text/xml"
      element="light:LightState" />
</application>
```

図 2 WADL 記述例
Fig. 2 An example of WADL

```
<?xml version="1.0" encoding="utf-8" ?>

<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   elementFormDefault="qualified">

   <xsd:element name="LightState" type="Brightnaess" />

  <xsd:element name="Brightness">
  <xsd:simpleType>
     <xsd:restriction base="xs:int">
        <xsd:minInclusive value="0" />
        <xsd:minInclusive value="1000" />
     </xsd:restriction>
  </xsd:simpleType>
  </xsd:element>

</xsd:schema>
```

図 3 xsd 記述例
Fig. 3 An example of xsd

を XML Schema で記述することで，値を厳密に定義することが出来る．図 2 では XSD ファイルを指定し，XSD ファイルで定義を行っている．XSD ファイルの内容を**図 3** に示す．図 3 では，PUT メソッドで送信する値とレスポンスとして返ってくる値の定義を行っている．定義の内容として，ライトの明るさの上限と下限の範囲を定めている．

4. 設　　計

本章では，提案手法を実装する際に用いるサービスプラットフォームに OSGi フレームワーク[15] を用い，必要な機能の設計を OSGi フレームワークに基づいて行う．

4.1 OSGi

OSGi は遠隔から操作できる Java ベースのサービスプラットフォームを定義する標準化団体が取り決めている仕様である．OSGi ではフレームワークにバンドルと呼ばれるアプリケーションのライフサイクルを管理させ，遠隔からサービスを再起動せずにアプリケーションの機能を入れ替えることが可能となる．

OSGi フレームワークを用いることにより，バンドルという単位で必要な機能を実装することができ，それぞれのサーバに必要なバンドルを追加することでそれぞれのサーバに合った機能を実現することを可能とする．また，バンドルはサーバ上のサービスを立ち上げたまま追加したり削除したりすることが可能であるため，必要な機能があればサービスを立ち上げたまま機能を追加できるという利点がある．ネットワークに対応した機器と通信する技術仕様の際については，バンドルを用いて各技術仕様の差異を吸収する．それぞれの技術仕様の差異を Web のインタフェースに置き換えるため，新たな技術仕様が登場したとしても，新たにバンドルを実装することで対応が可能となる．それ以外の機器に対しては OSGi のバンドルを用いてネットワーク対応を実現する．

4.2 必要となるバンドル

本節では，提案手法に必要なバンドルを設計し，説明を行う．

4.2.1 REST インタフェース生成バンドル

Java で REST を実装するためのフレームワークである Restlet[16] を用いて，REST インタフェースを生成するバンドルである．REST インタフェースとは HTTP メッセージを受け，メッセージの内容を元に命令を実行するためのインタフェースである．このバンドルはどの状況においても必ず組み込まれている必要がある．

4.2.2 技術仕様 REST インタフェース変換バンドル

各技術仕様を REST インタフェースで扱えるように REST インタフェースと各技術士用間のインタフェースの変換を行うバンドルである．このバンドルは各技術仕様に対応した機器が LAN 内にある場合に機器連携サーバに組み込むこととする．

4.2.3 機器 REST インタフェース変換バンドル

ネットワークインタフェースを持たない機器を REST インタフェースに対応するためのバンドルである．本バンドルは，それぞれの機器が対応している技術仕様に基づいて実装を行う．そのため，本バンドルはネットワークインタフェースを持たない機器の数だけ用意する必要がある．このバンドルは，機器に接続する PC に組み込むこととする．

4.2.4 REST インタフェース対応クライアントバンドル

REST 設計に基づいた HTTP メッセージを送信するためのクライアントバンドルである．他の機器や Web サービスに対して HTTP メッセージを送信するバンドルである．このバンドルはどの状況においても必ず組み込まれている必要がある．

4.2.5 WADL 生成バンドル

REST インタフェースを公開する際に WADL を自動で生成するバンドルである．このバンドルはどの状況においても組み込まれている必要がある．

5. アプリケーション適応

本章では，実装に用いるアプリケーションの選定を行う．ネットワーク対応機器に，UPnP を用いたメディアプレイヤーを，ネットワーク非対応の機器として照明を用いる．Web サービスにはストレージサービスを用い，それぞれが連携できることを検証する．照明は

機器連携サーバに接続され，操作されることとする．

5.1 UPnP

UPnP は機器をネットワークに接続するだけで，その機器を利用することができる技術仕様である．UPnP はデバイス，サービス，コントロールポイントから構成される．デバイスは，1つ以上のサービスを持ち，それぞれの機器の機能を UPnP の仕様に従って提供するためのサービスコンテナである．サービスは機器が持つ機能を細分化した際の1単位であるサービスを制御するモデルとしてアクションが定義されている．コントロールポイントは他のデバイスを検出、制御を行うことができる．コントロールポイントには以下の機能がある．

- デバイスの情報（デバイスディスクリプション）を取得し，関連サービスのリストを取得する
- サービスを制御するアクションを呼び出す
- サービスのイベントソースをサブスクライブする

本提案手法を実装する際には，コントロールポイントのインタフェースを REST アーキテクチャのインタフェースに変換するバンドルを用いる．バンドルのは次の機能を持つ．

サービス発見 UPnP 機器発見を行い，UPnP 機器が見つかると，その UPnP 機器のデバイスディスクリプションを取得する

インタフェース変換 サービスとアクションを REST インタフェースに変換する

命令送信 HTTP メッセージを受け取った場合，そのメッセージ内容を SOAP メッセージに変換して UPnP 機器へ命令を送信する

5.2 提案手法による連携シナリオ

ユーザはストレージサービスに保存していた動画データを，メディアプレイヤーで再生することを考える．そして，メディアプレイヤーで再生を行うときにメディアプレイヤー側にある照明を消し，動画を見る環境を整えるといった連携を行う．

連携手法では，ユーザは機器連携サーバに対して，PUT メソッドを用いた HTTP メッセージにより，ストレージサービスに対して動画の要求，取得を行わせる．動画の取得が終わった際に，機器連携サーバはバンドルの機能である，インタフェース変換と，命令送信を用いてメディアプレイヤーに動画を送信し再生させる．次の命令としてメディアプレイヤーのそばにある照明に PUT メソッドを用いた HTTP メッセージを送信し，照明の消灯を行う．このようなシナリオで，ユーザが連携する機器を指定し，機器それぞれが通信を行うことで連携が可能となる．

6. 考　　察

6.1 技術仕様間の差異の吸収

本提案手法では，機器操作や機器の情報を取得するためのインタフェースを，それぞれの技術仕様で定義されている形式を REST アーキテクチャに基づいた Web インタフェースに変換することで，機器の操作や，機器の情報の取得を行うことを可能とする．それぞれの技術仕様で定義された形式を REST アーキテクチャに変換するためには，それぞれの技術仕様を REST アーキテクチャに変換するアプリケーションが必要となる．しかし，そのアプリケーションが再利用可能でない場合や，簡単に導入できない場合，あらゆる機器に対応することが難しいと考えられる．本提案手法を実現するための設計として，OSGi フレームワークを用いて，バンドルでそれぞれの技術仕様を REST アーキテクチャに変換する設計とした．また，インタフェースに REST アーキテクチャに基づいた Web インタフェースを用いることで Web サービスとの連携も統一したインタフェースで行うことが可能となる．この設計により，ネットワークを通じてバンドルの導入を行うことで，様々な技術仕様で定義されている形式を REST アーキテクチャに基づいた Web インタフェースに変換することを可能とした．これにより，機器操作や機器の情報を取得するための各技術仕様間の差異を REST アーキテクチャに基づいた Web インタフェースに統一することで吸収することが可能となる．

6.2 ユーザにおける機器の組み合わせの指定

ユーザは開発者が意図した使用方法を超えた使い方を思いつくことがあるという研究結果[11] より，ユーザが思いつく使用方法に対応出来るよう，あらかじめ開発者が定義した機器の組み合わせや Web サービスの組み合わせだけでなく，ユーザが思いついた組み合わせを指定できる環境が望ましい．本提案手法では，インタフェースの定義に WADL と XML Schema を用い，機器や Web サービスに対する命令やレスポンスの内容を定義する．これにより，クライアントプログラムはあらかじめ，どのような命令を送信すればどのようなレスポンスが返ってくるかが分かり，そのレスポンスに対応した機器や Web サービスを，ユーザは組み合わせて使用することができる．また，データの形式が異なる場合でも Web サービスを介することで，データの形式を適合させるよう変換を行うことも可能になると考えられる．これにより，ユーザは開発者が意図した以上に，様々な機器や Web サービスを

組み合わせることが可能となる．

7. おわりに

本稿では，REST アーキテクチャを用いた統合型機器連携手法を提案し，ネットワークに対応した機器やネットワークインタフェースを持たない機器同士，また，Web サービスとの連携を行うことを可能とするインタフェースの設計を行った．これによりユーザは機器や Web サービスを様々な利用状況において，それらの機器や Web サービスを自由に組み合わせて利用することが可能となる．現状では，ユーザが機器の組み合わせを設定するためにはスクリプトを書く必要がある．今後は，ユーザが自由に機器の組み合わせを設定できるフレームワークの実装やユーザ認証が必要な Web サービスに関しての対応の検討などを行う．

参考文献

1) 東海林祥一, 高橋秀幸, 北形　元, 菅沼拓夫, 白鳥則郎 :ユビキタス環境におけるコンテクストを考慮したマルチメディア通信システム, 情報処理学会研究報告. マルチメディア通信と分散処理研究会報告, No.58, pp.57-62, 2005.
2) 増淵　敬, 並木美太郎 :XML を用いた遠隔機器制御支援システム，情報処理学会論文誌, Vol.44, No.SIG 10, 2003.
3) 小澤政博, 川原圭博, 川西　直, 森川博之 :REST アーキテクチャスタイルに基づくコンテキストアウェアサービス連携フレームワークの設計, 電子情報通信学会総合大会, p.259, 2007.
4) 河口信夫, 春原雅志 :WebCodget：Web サーバに移動して動作する組み込み機器向け移動ソフトウェア, 情報処理学会研究報告, ユビキタスコンピューティングシステム, No.107, pp.41-44, 2005.
5) 岩崎陽平, 榎堀　優, 藤原茂雄, 田中宏一, 西尾信彦, 河口信夫 :REST に基づく異種スマート環境間のセキュアな連携基盤, マルチメディア, 分散, 協調とモバイル (DICOMO 2008) シンポジウム論文集 (2008), pp.185-194, 2008.
6) DLNA: http://www.dlna.org/.
7) UPnP: http://www.upnp.org/.
8) Jini Network Technology, Sun Microsystems: http://www.sun.com/software/jini/.
9) HAVI Home Pages: http://www.havi.org.
10) 杉原義得, 野村昌弘 :家電をネットワーク化する際の技術の現状と標準化に向けた課題に関する調査
http://www.ipa.go.jp/NBP/13nendo/reports/homeerct/ehomestd/ehomestd.pdf
11) Irene Mavrommati, Achilles Kameas: The evolution of objects into hyper-objects: will it be mostly harmless?, Personal and Ubiquitous Computing, ACM. vol.7, no.3-4. pp.176-181, 2003.
12) Fielding, R.T.: Architectual styles and the design of network-based software architectures, PhD Thesis, University of Californis, Irvine 2000.
13) Richardson, L. and Ruby, S. :Restful Web Services, Oreilly & Associates Inc 2007.
14) WADL: https://wadl.dev.java.net/.
15) OSGi Alliance: http://www.osgi.org/.
16) Restlet: http://www.restlet.org/.