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.

# センササービスのマッシュアップを実現するサービス指向基盤の提案

坂本　寛幸† 　井垣　宏† 　中村　匡秀†

† 神戸大学　〒 657–8501 神戸市灘区六甲台町 1–1
E-mail: †sakamoto@ws.cs.kobe-u.ac.jp, ††{igaki,masa-n}@cs.kobe-u.ac.jp

**あらまし**　近年，ホームネットワーク上のセンサから機器やユーザの状態をコンテキストとして推定し，コンテキストに応じたサービスを提供するコンテキストアウェアアプリケーションの開発が進みつつある．我々の研究グループでは，温度センサ，照度センサといったセンサデバイスをサービス化するためのセンササービス基盤を提案している．サービス化されたセンサ（センササービス）を利用することで，アプリケーション開発者は任意の機器とセンサを容易に連携させることができる．しかしながら，センササービスだけでは単一のセンサしか扱えず，複数センサを必要とするような複雑なコンテキスト (マッシュアップコンテキスト) 推定を行うには，開発者自身が複雑なロジックを実装する必要があった．そこで本論文では，複数センサを用いた複雑なコンテキスト推定を行うことのできるプラットフォーム SMuP（Sensor Mashup Platform）を提案する．SMuP は既存のセンササービスのマッシュアップによる仮想センササービスの構築とマッシュアップコンテキスト推定のための条件登録機能を公開している．ケーススタディでは，SMuP を利用して実際に複雑なマッシュアップコンテキストを利用するアプリケーションの例を示す．
**キーワード**　web サービス，コンテキストアウェア，センサ駆動，ホームネットワーク

# A service-oriented platform for sensor service mashups

Hiroyuki SAKAMOTO†, Hiroshi IGAKI†, and Masahide NAKAMURA†

† Kobe University　rokkoudaityou 1–1, nada-ku, Kobe, Hyogo, 657–8501 Japan
E-mail: †sakamoto@ws.cs.kobe-u.ac.jp, ††{igaki,masa-n}@cs.kobe-u.ac.jp

**Abstract**　Context-aware applications are widely developed in home network system. Such applications estimate contexts with sensor devices. Generally, it is not easy to modify applications which incorporated an appliance and a sensor after deployment, because of tightly-coupled between the appliance and the sensor. In our precedence research, we proposed a sensor service framework to encapsulate sensor devices. Sensor services with our framework expose standardized interfaces which enable user to acquire sensor data and register conditions to estimate contexts. With using such interfaces, application developers can make arbitrary appliances and sensors integrated easily. However, our sensor service framework includes only a single sensor device. In this paper, we propose a SMuP(Sensor Mashup Platform) which supports estimating mashup contexts using various sensor data. In SMuP, we can create virtual sensor services including mashup sensor properties by combining multiple sensor data. With using such virtual and real sensor services, we can register complex contexts(mashup contexts) which contain multiple sensor data. As case studies, we present that developers can implement applications easily with the result of estimated contexts by SMuP.
**Key words**　Sensor driven, Home Network, Web Service, Context Aware

## 1. はじめに

近年，多様なセンサを利用し，ユーザとその周囲の状況に即した機器制御を提供するコンテキストアウェアアプリケーションの開発・研究が進められている．ここでコンテキストとは，ユーザや機器，場所等の多様な実体の状況を特徴付けるのに用いられる情報の集合を指す．コンテキストは一般に条件式 (**コンテキスト条件**) として定義され，コンテキストアウェアアプリケーションでは，検出されたコンテキストに即した振る舞いを行うように実装されている．

例えば室温が 28 度以上になると，クーラーの冷房を起動するといったユーザや機器の状態に即したサービスを提供するこ

とができる．実際に，人感センサと照明機器や水道の蛇口，自動ドアやセキュリティなどの多様な機器とセンサを連携させたアプリケーションが既に販売されている [1] [2] [3].

これらの既存のコンテキストアウェアアプリケーションでは，その構成要素である機器やセンサが密に結合されているものが多く，新たなサービスの開発やコンテキスト条件のカスタマイズなどが非常に困難であった．

我々は先行研究 [6] [7] において，コンテキストアウェアアプリケーションにおけるセンサをサービス化するための**センササービス基盤**を提案した．アプリケーションで利用される温度センサや照度センサ等の各種センサ類を基盤を用いてサービス化することで，アプリケーション-機器-センサ間の疎結合を実現した．サービス化されたセンサ (**センササービス**) はセンサ値の取得やコンテキスト条件の登録といった機器との連携のための標準インタフェースを公開している．コンテキストアウェアアプリケーションの開発者はセンササービスを利用することで，任意の機器とセンサを容易に連携させることが可能となった．

一方で，先行研究におけるセンササービスには特定の単一センサしか扱えないという制約があった．例えば温度センササービスに対して「室温が 28 ℃以上」といったコンテキスト条件を登録することは可能であったが,「リビングに在室していてかつ室温が 28 度以上」といったような複数のセンサを含む複雑なコンテキスト条件 (**マッシュアップコンテキスト条件**) の登録は出来なかった．結果として，開発者自身が個々のセンササービスから値を取得し，複雑なコンテキスト推定のための実装を行わなければならなかった．

本論文では，このような複雑なコンテキストを利用するコンテキストアウェアアプリケーション開発を支援するために，複数センサのマッシュアップを容易化するプラットフォーム (SMuP:Sensor Mashup Platform) を提案する．

SMuP では，既存のセンササービス (Single Sensor Service) を import することで，マッシュアップコンテキスト条件式の登録が可能である．また，SMuP では，import されている任意のセンササービスをマッシュアップすることで新たな付加価値の高いセンサを作成することができる．例えば，部屋に複数設置してある照度センササービスを組み合わせて，部屋の平均の照度を取得できるセンサを作成することができる．既存のセンササービスとマッシュアップにより作成された新たなセンサを用いることでより高度なコンテキストの推定ができ，コンテキストアウェアアプリケーションを容易に開発することができる．

以降では，SMuP について詳述し，ケーススタディとしてコンテキストアウェアアプリケーション開発例を紹介する．

## 2. 準　　備

### 2.1 コンテキストアウェアアプリケーション

近年，ユビキタスコンピューティング技術の発展に伴い，ホームネットワーク上のセンサデバイスを利用したアプリケーションの開発が行われている．コンテキストアウェアアプリケーションは，多様なセンサデバイスから環境情報やユーザ，機器の状態をコンテキストとして推定し，その内容に応じたサービスを提供する [4]．コンテキスト推定のための情報はセンサ等から得られる．例えば「部屋の温度センサが 28 度」という情報からは「暑い」というコンテキストが推定される．アプリケーションはこの「暑い」というコンテキストをもとに「エアコンの冷房運転を開始する」といったサービスを提供する．

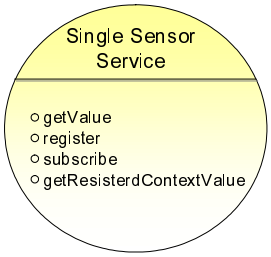


図 1　センササービスの持つインタフェース

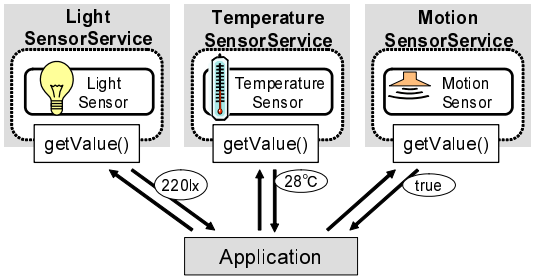


図 2　標準的なインタフェースによるセンサ値の取得

より高度なコンテキストアウェアアプリケーションを実現するためには，多くのセンサを利用し，複雑なコンテキストを推定することが必要である．

### 2.2 センササービス基盤

我々の研究グループでは，これまでに温度センサ，照度センサなどのセンサデバイスをサービスとして公開するためのセンササービス基盤 [6] を提案した．サービス化されたセンサ (以下，センササービスという) はセンサ固有の制御ロジックを内部に隠蔽し，標準的なインタフェース (API) を公開することで，アプリケーション-機器-センサ間の疎結合化を実現している．図 1 にセンササービスの持つ主要な標準インタフェースを示した．このインタフェースを利用することで，センサ値の取得やコンテキスト条件登録が利用できる．

アプリケーション開発者のセンササービス利用イメージを図 2，図 3 に示した．各センサは自身が測定可能な**プロパティ**を保持している．getValue インタフェースはそれらのセンサプロパティ値を標準的な方法で取得可能にする．また，コンテキスト条件の登録は register インタフェースが利用できる．例えば，センササービスに *hot* というコンテキスト名で $temperature > 27$ ℃ というコンテキスト条件を登録できる．このように登録されたコンテキストは，subscribe インタフェースや getRegisteredContextValue インタフェースから利用可能である．getRegisteredContextValue インタフェースでは，開発者は登録された特定のコンテキスト条件の現在の値を取得できる．subscribe インタフェースでは Publish/Subscribe 型のメッセージ交換パターン [5] にもとづき，任意のコンテキスト

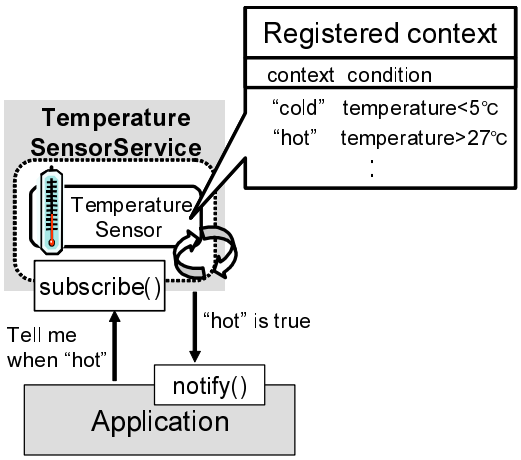


図 3　コンテキスト条件の登録と publish/subscribe によるコンテキスト推定

条件の値が真になったときに notify を行うサービスを指定することができる．

このように，センササービスを利用することで，アプリケーション開発者はコンテキスト条件が真になったときの振る舞いのみを実装すればよい．

### 2.3　複雑なコンテキスト推定実現のための課題

センササービスでは，例えば温度センササービスに対して「温度が 28 度以上であれば」といったような自身のプロパティを利用した単純なコンテキスト条件を登録することができた．しかし，複数のセンサを組み合わせた複雑なコンテキスト推定をセンササービスのみで行うことはできない．例えば，部屋に設置されている複数の照度センサが示す値の平均値や室温と湿度から構成される不快指数の取得をセンササービスのみで行うことはできない．複数のセンサプロパティを組み合わせたコンテキスト条件を特定のセンササービスに登録することも対応できない．そのため，従来のセンササービスを利用して複雑なコンテキスト推定を行うためには，開発者が個別のセンサプロパティ値の取得からコンテキスト推定までの全てのロジックを実装する必要があった．

よって本論文では，以下の要求を満たすような，複数センサを利用したコンテキストの推定ができるプラットフォーム (SMuP:Sensor Mashup Platform) を提案する．

**要求 R1**: マッシュアッププロパティの作成

**要求 R2**: マッシュアップコンテキスト条件の登録

マッシュアッププロパティとは，複数の照度センサの平均値や特定の温度センサと湿度センサから算出される不快指数，といった既存のセンサプロパティのマッシュアップによる新しいプロパティのことをいう．R1 では，マッシュアッププロパティの登録により，登録されたプロパティの再利用や複雑なコンテキスト推定のための条件式構築が可能となる．

マッシュアップコンテキストとは，複数センサを利用したコンテキストのことであり，マッシュアップコンテキストを推定するための条件式をマッシュアップコンテキスト条件という．R2 では，R1 で作成されたマッシュアッププロパティを含む多様なセンサプロパティを用いたマッシュアップコンテキスト条件の構築・登録を目的とする．これまでの特定のセンササービス内部に閉じていたコンテキスト条件登録の幅を広げることで，より多様で複雑なコンテキスト推定を実現する．

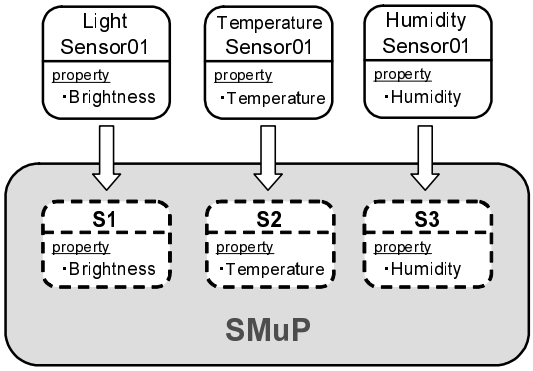


図 4　SMuP へのセンササービスの登録

表 1　保持するセンササービス情報の例

| SensorName | wsdl |
|---|---|
| S1 | http://myHNS/LightSensor01.wsdl |
| S2 | http://myHNS/TemperatureSensor01.wsdl |
| S3 | http://myHNS/HumiditySensor01.wsdl |
| S4 | http://myHNS/MotionSensor01.wsdl |
| S5 | http://myHNS/LightSensor02.wsdl |
| S6 | http://myHNS/MotionSensor02.wsdl |

アプリケーション開発者は，R1,R2 を満たす SMuP を利用することで，センサの制御ロジックを気にすることなく，コンテキストが推定された際の振る舞いのみを実装すればよいので，容易にアプリケーションを開発できるようになる．

## 3.　Sensor Mashup Platform の提案

### 3.1　キーアイデア

2.3 で述べた要求を満たすために，SMuP では次の 2 つのキーアイデアを実現する．

(K1):**マッシュアッププロパティを持つ仮想センササービスの登録**

(K2):**マッシュアップコンテキストの登録**

SMuP では，仮想センササービスを利用してマッシュアッププロパティが管理される．アプリケーション開発者は，既存のセンササービスと同様に，仮想センササービスを通してマッシュアッププロパティにアクセスすることが出来る．

また，SMuP はマッシュアップコンテキストを登録するためのインタフェースを公開する．既存のセンササービスでは，センササービス個別に持っていたコンテキスト登録インタフェースを SMuP に持たせることで，多様なセンサプロパティを利用することが可能となる．

これらのキーアイデアについて，次節以降で説明する．

### 3.2　仮想センササービスの登録

仮想センササービスはマッシュアッププロパティを持つセンササービスである．SMuP は仮想センササービスを用いてマッシュアッププロパティの管理を行う．ユーザはまず SMuP 内に

仮想センササービスを作成し，その仮想センササービスに対してマッシュアッププロパティの登録を行う．具体的な STEP を以下に示した．

Step1: 既存のセンササービスを SMuP に登録する．

Step2: 仮想センササービスを作成する．

Step3: Step1 で登録されたセンササービスの持つプロパティをマッシュアップし，マッシュアッププロパティを作成する．

Step4: マッシュアッププロパティを Step2 で作成した仮想センササービスに登録する．

上記の STEP では，SMuP が持つ *importSensor*, *createSensor*, *addProperty* の各インタフェースが利用される．

具体的なセンササービスの例を表 1 に示す．以降ではこの例を利用して各 Step について説明する．Step1 では *importSensor* インタフェースを利用して既存のセンササービスの SMuP への登録が行われる．例えば図 4 の例では，LightSensor01 に S1 という別名を付けて SMuP に登録を行っている．SMuP 内部では S1.Brightness を指定することで，LightSensor01 の持つ Brightness プロパティにアクセスすることが可能である．

*createSensor* は Step2 で述べた仮想センササービスの作成を行う．SMuP では，マッシュアッププロパティは必ず特定の仮想センササービスに属さなければならない．

Step3 で作成されたマッシュアッププロパティは Step4 において任意の仮想センササービスに SMuP の *addProperty* インタフェースを利用して登録される．ここで，マッシュアッププロパティの構築には，四則演算子，比較演算子，論理演算子の利用が可能である．マッシュアッププロパティは，SMuP 内に import もしくは create された全てのセンサが持つプロパティを変数として利用する式として表される．

図 5 の例では，NewSensor01 が createSensor によって作成される．さらに，NewSensor01 に対する addProperty を実行することで，Avg_Brightness という名前を持つマッシュアッププロパティが登録される．ここで Avg_Brightness は $(S1.Brightness + S5.Brightness)/2$ という S1 と S5 の二つのセンササービスが持つプロパティを含むプロパティ式として表現されている．

このようにして作成された仮想センササービスからは SMuP の getValue を通じてマッシュアッププロパティの値を取得することが可能である．SMuP ユーザは容易にマッシュアッププロパティの値を再利用し，アプリケーションを開発することが出来る．

### 3.3 マッシュアップコンテキストの登録

SMuP は，複数のセンサが持つプロパティを横断的に扱うことを目的として，SMuP 自身が *register* および *subscribe*, *getRegisteredContextValue* インタフェースを公開する．

*register* インタフェースでは，マッシュアップコンテキスト名およびマッシュアップコンテキスト条件の登録を行う．図 6 の例では，mashup context01 として，$NewSensor02.Human\_enter == true \&\& S2.Temperature > 28$ が *register* インタフェースを利用して登録されている．このマッシュアップコンテキストは，NewSensor02 の $Human\_enter$ プロパティが真で，かつ，S2 の $Temperature$ プロパティが 28 ℃よりも大きくなったときに，全体が真になる．ここで登録されたマッシュアップコンテキストは Registered Mashup Context において管理されており，SMuP の *getRegisteredContextValue* を利用してその値を取得することができる．登録されるマッシュアップコンテキスト条件は既存のコンテキスト条件と同様に，値として真偽値のみを取る．

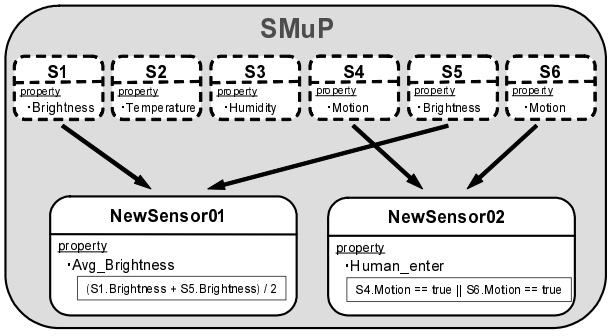


図 5 マッシュアップによる仮想センササービスの作成

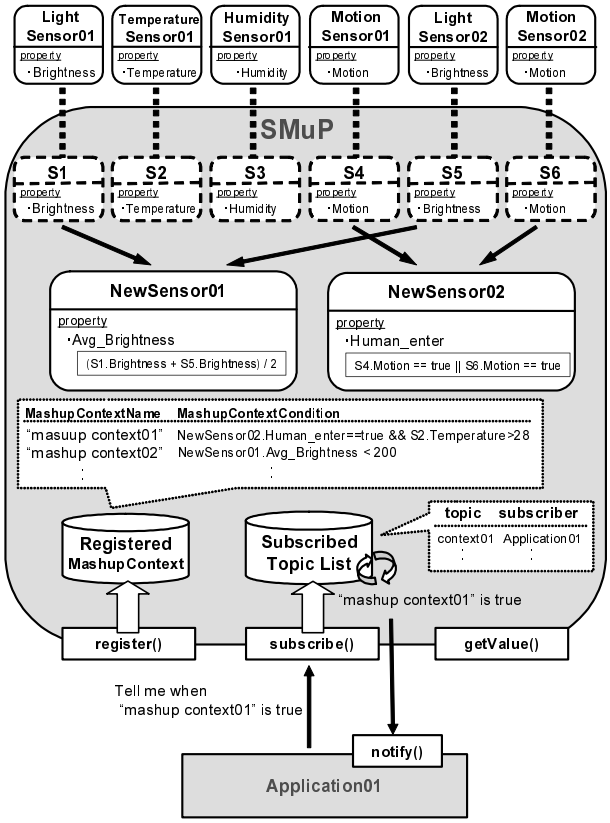


図 6 SMuP 全体のアーキテクチャ

*subscribe* インタフェースでは，登録されたマッシュアップコンテキストを Publish/Subscribe 型メッセージ交換におけるトピックとして，購読処理を行うことが出来る．*subscribe* インタフェースによって，マッシュアップコンテキストと通知先アドレスを指定することで，対象のマッシュアップコンテキストが真になったときのみ notify をさせることが可能となる．

このように，SMuP を利用することで複数のセンサ間にまたがる横断的なマッシュアップコンテキストの登録・利用が容易

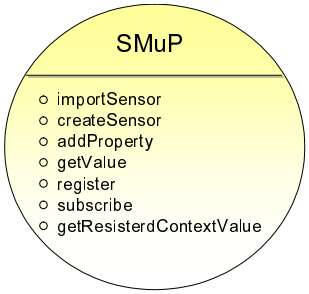


図 7　SMuP の持つインタフェース

に行える．

### 3.4　SMuP の設計

図 7 は，SMuP が提供する主要なインタフェースの一覧である．各インタフェースの詳細を以下に示す．

**importSensor**:　SMuP に新たに既存のセンササービスを登録する．引数にセンササービスの WSDL とセンサ名を指定する．

**createSensor**:　SMuP 上に仮想センササービスを作成する．引数にセンサ名を指定する．作成されるセンサ名は SMuP 内でユニークなものでなければならない．

**addProperty**:　createSensor インタフェースで作成されたセンササービスに対して，マッシュアッププロパティの追加を行う．引数に追加する対象となるセンササービス名，マッシュアッププロパティ名，マッシュアッププロパティ式を指定する．式では，変数として各センサ (import もしくは create されたもの）のプロパティを利用できる．このとき，S1.Brightness のように「センササービス名. プロパティ名」という形式で各プロパティは指定される．

**getValue**　SMuP に登録されているセンササービスのプロパティ値を取得する．引数に「センササービス名. プロパティ」を指定する．

**register**:　SMuP にマッシュアップコンテキストを登録する．引数にマッシュアップコンテキスト名とマッシュアップコンテキスト条件を指定する．

**subscribe**:　register インタフェースにより登録されているマッシュアップコンテキスト名とマッシュアップコンテキスト条件が真になった際の通知先アプリケーションを引数に指定することで，マッシュアップコンテキスト条件の判定を始める．また，対象のマッシュアップコンテキスト条件が真になった時はアプリケーションに通知を行う．

## 4.　ケーススタディ

本章では，SMuP を利用したコンテキストアウェアアプリケーション開発例をケーススタディとして示す．以下のケーススタディにおいて，表 1 で示される各センササービスが SMuP に登録されているものとする．

### 4.1　不快指数を利用した冷房サービス

本節では，不快指数に応じて冷房の設定温度を自動調節する冷房サービスの開発例を示す．不快指数とは，温度と湿度から計算される蒸し暑さの指標であり，温度を T，湿度を H とすると以下の計算式で与えられる．

$$0.81T + 0.01H(0.99T - 14.3) + 46.3$$

一般に，不快指数が 75〜80 であればやや不快で，80 より高ければとても不快であることが知られている．ここでは，不快指数が 75〜80 であれば冷房を設定温度 28 度で運転し，80 より高ければ冷房を設定温度 26 度で運転するサービスを開発する．

まず，表 1 における S2 と S3 のセンササービスをマッシュアップして，SMuP 上に不快指数を測定できる新たなセンササービス (RoomEnvironment センサ) を作成する．さらに，作成した RoomEnvironment センサを対象として，不快指数 (discomfort_index) をセンサプロパティとして登録する．不快指数は上記の計算式を利用する．ここで用いられる SMuP インタフェースは createSensor，addProperty である．呼び出しの詳細を以下に示す．

**RoomEnvironment センサの作成**:　http://myHNS/SMuP/createSensor?name=RoomEnvironment

**マッシュアッププロパティ (不快指数) の登録**:　http://myHNS/SMuP/addProperty?sensor=RoomEnvironment&property=discomfort_index&expression=0.81*s2.Temperature+0.01*s3.Humidity*(0.99*s2.Temperature-14.3)+46.3

createSensor インタフェースによって RoomEnvironment センササービスが作成され，addProperty インタフェースにより「discomfort_index」というプロパティが RoomEnvironment センササービスに追加される．

次に，コンテキスト名，コンテキスト条件の登録を行う．ここでは，不快指数を利用して,「不快 (uncomfortable)」と「非常に不快 (very unconfortable)」の 2 つのマッシュアップコンテキストを登録する．呼び出される SMuP インタフェース register の詳細を以下に示す．

**コンテキスト (不快) の登録**:　http://myHNS/SMuP/register?context=uncomfortable&condition=RoomEnvironment.discomfort_index>75&& NewS01.discomfort_index<80

**コンテキスト (とても不快) の登録**:　http://myHNS/SMuP/regster?context=very_uncomfortable&condition=RoomEnvironment.discomfort_index>=80

以上の処理により,「uncomfortable」と「very_uncomfortable」の 2 つのコンテキストが登録された．

最後に，subscribe インタフェースにより登録したマッシュ

```
public class CoolerService{
   public boolean notify(String context){
      if(context.equals("uncomfortable")){
            cooler.on(preset=28)
      }else if(context.equals("very_uncomfortable")){
            cooler.on(preset=26)
      }
   }
}
```

図 8　CoolerService のソースコード

アップコンテキスト名とマッシュアップコンテキストが真になった際の通知先アプリケーションを指定することで，冷房サービスが開始される．

**コンテキスト (不快) の購読**: http://myHNS/SMuP/subscribe?context=uncomfortable&notify=http://myHNS/CoolerService.wsdl

**コンテキスト (とても不快) の購読**: http://myHNS/SMuP/subscribe?context=very_uncomfortable&notify=http://myHNS/CoolerService.wsdl

ここで，subscribe の際に指定する通知先アプリケーションは Web サービスとして公開されているものとし，notify インタフェースを持つものとする．subscribe によって条件判定を開始した SMuP は,「uncomfortable」というコンテキストを推定すると，coolerService の notify インタフェースを以下の URL によって呼び出す．

http://myHNS/coolerService/notify?context=uncomfortable

推定されたコンテキストが notify インタフェースの引数として与えられることで，coolerService はコンテキストを知ることができる．図 8 は CoolerService のソースコード例である．notify メソッド内にて,「uncomfortable」が推定されれば設定温度 28 度で冷房運転を,「very_uncomfortable」が推定されれば設定温度 26℃ で冷房運転を開始するロジックを記述しておくことで，不快指数を利用した冷房サービスが実現される．

### 4.2 エコ照明サービス

本節では，部屋の内外の照度差を利用して部屋の照明を制御するエコ照明サービスを開発する．エコ照明サービスは，部屋の外の方が十分に明るければカーテンを明け，そうでなければ照明をつけることで部屋を明るくするサービスである．

表 1 で示される S2 が部屋内に設置された照度センサ，S5 が部屋の外 (屋外) に設置された照度センサであるとすると，部屋の内外の照度差は以下の式で与えられる．

$$S5.Brightness - S2.Brightness$$

まず，addProperty インタフェースを用いて，4.1 で作成した SMuP 上の RoomEnvironment センササービスに新たなマッシュアッププロパティ (Brightness_difference) を登録する．

**マッシュアッププロパティ (内外の照度差) の登録**: http://myHNS/SMuP/addProperty?sensor=RoomEnvironment&property=Brightness_difference&expression=S5.Brightness-S2.Brightness

次に，マッシュアップコンテキストの登録を行う．Brightness_difference の値が 100 ルクスよりも大きければ，部屋の外が十分に明るく (*bright*)，100 ルクスよりも小さければ部屋の外は暗い (*dark*) ものとして登録する．

**コンテキスト (bright) の登録**: http://myHNS/SMuP/register?context=bright&condition=RoomEnvironment.Brightness_difference>=100

**コンテキスト (dark) の登録**: http://myHNS/SMuP/register?context=dark&condition=RoomEnvironment.Brightness_difference<100

最後に，subscribe インタフェースにより上記の各コンテキスト登録したコンテキスト名とコンテキストが推定された際の通知先アプリケーションを指定する．

**コンテキスト (bright) の購読**: http://myHNS/SMuP/subscribe?context=bright&notify=http://myHNS/ecoLightService.wsdl

**コンテキスト (dark) の購読**: http://myHNS/SMuP/subscribe?context=dark&notify=http://myHNS/ecoLightService.wsdl

4.1 と同様に，SMuP によってコンテキストが推定されると ecoLightService の notify インタフェースが呼び出される．ecoLightService の notify メソッド内にて,「bright」が推定されればカーテンを開け,「dark」が推定されれば照明をつけるロジックを記述しておくことでエコ照明サービスは実現される．

## 5. おわりに

SMuP によって，仮想センササービスとマッシュアップコンテキストの管理を行うことで，2.3 で述べた要求 R1,R2 を実現した．一度登録された仮想センササービスおよびマッシュアッププロパティ，マッシュアップコンテキストは SMuP を通して再利用が可能である．ケーススタディで示したように，SMuP に登録された仮想センササービスおよびマッシュアップコンテキストを利用することで，開発者は図 8 に示す推定されたコンテキストに応じた振る舞いを実装するだけで良い．結果として，アプリケーション開発者はコンテキスト推定に関する全てを SMuP に委託し，推定結果にもとづくロジックのみに注力することが可能となった．

一方で，SMuP によってこれまでより複雑なコンテキスト条件の構築が可能となった結果，マッシュアップコンテキストの構築そのものが複雑になりつつある．今後，マッシュアップコンテキスト開発エディタのようなコンテキスト構築支援の枠組みをサービスとして提供していきたい．

**謝辞** この研究の一部は，科学技術研究費 (若手研究 B 18700062，20700027)，およびパナソニック電工株式会社の助成を受けて行われている．

## 文　献

[1] Matsushita Electric Works Ltd. Katteni switch, http://biz.national.jp/Ebox/katte_sw/.

[2] Nabtesco Corp. Automatic entrance system. http://nabco.nabtesco.com/english/door_index.asp.

[3] Daikin Industries Ltd. Air conditioner, http://www.daikin.com/global_ac/.

[4] Dey, A. K. and Abowd,G. D.,"Toward a Better Understanding of Context and Context-Awareness", In Proceedings of the CHI2000 Workshop on The What, Who, Where, and How of Context-Awareness, 2000.

[5] G. Muhl, "Generic Constraints for Content-Based Publish/Subscribe." In Proc. of the 9th Int'l Conf. on Cooperative Information Systems(CooplS '01), pp. 211-225, 2001.

[6] 坂本 寛幸, 井垣 宏, 中村 匡秀, "コンテキストアウェアアプリケーションの開発を容易化するセンササービス基盤," 電子情報通信学会技術研究報告, vol.108, no.458, pp.381-386, March 2009.

[7] Yoji Onishi, Hiroshi Igaki, Masahide Nakamura, and Kenichi Matsumoto, "A Scalable Sensor Application Framework Based on Hierarchical Load-Balancing Architecture," In Proc. of the IASTED International Conference on Software Engineering (IASTED SE 2008), pp.37-42, February 2008. (Innsbruck, Austria)