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.

サービス指向ホームネットワークにおけるタイミング制約を用いたセンサ連携サービスの実装

丸尾　彰宏†　　まつ本真佑†　　中村　匡秀†

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

あらまし　我々の研究グループでは，ホームネットワークにおける様々なセンサデバイスを Web サービスでラップし，コンテキストアウェアサービスの開発を容易化するセンササービスフレームワークを研究している．先行研究では，コンテキスト間の時間的な制約を考慮するために，2 種類の時間的な制約 (タイミング制約) を導入し，高度なコンテキストを定義可能とした．しかし，タイミング制約を含むコンテキスト作成には，事前に既存コンテキストの詳細を知る必要があり，またコンテキストの成立を検出するために複雑なロジックを実装する必要があった．本研究では，既存コンテキストの詳細を一括管理するサービスを実装し，さらにこれを利用した複数のセンササービスを，2 種類の時間的な制約（逐次的タイミング制約，継続的タイミング制約）に基づいて連携するサービスを実装し，Web サービスとしてデプロイする．ケーススタディでは実装したサービスを用いて，入退室コンテキスト，家電つけっぱなしコンテキストを作成し，有効性の評価を行う．
キーワード　ホームネットワークシステム　コンテキストアウェア　サービス指向アーキテクチャ　タイミング制約

Implementing Sensor Integration Service with Timing Constraints in Service-Oriented Home Network

Akihiro MARUO†, Shinsuke MATSUMOTO†, and Masahide NAKAMURA†

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

Abstract　We have been studying the sensor service framework (SSF) in the home network, which wraps various sensor devices by Web services to achieve easy development of context-aware services. In order to specify time constraints in the contexts, importing two kinds of time constraints, we can define the high-level contexts. However, for creating high-level context with timing constraints, there is a need to implement complex logic to detect the context, and also will need to know the infomation of the pre-existing context. In this paper, we implement the services that collectively manage the infomation of an existing context. Using this service, we implement the sensor integration service to work on the basis of two types of time constraints. In case studies, using the service that implements, we create Enter-Leave context and TV left on context, and evaluate the effectiveness.
Key words　home network system, context-aware services, serivice-oriented architecture, timing constraints

1. はじめに

近年，様々なセンサを利用し，ユーザとその周囲の状況に即したサービスを実行するコンテキストアウェアアプリケーションの研究・開発が盛んである [1] [2]．コンテキストとは，ユーザや機器，場所などの多様な実体の状況を特徴付ける情報の集合を指す．コンテキストは一般にセンサから得られた値を利用した条件式で定義される．コンテキストアウェアアプリケーションは，あるコンテキストが成立したとき，その状況に応じた振舞いを実行するよう実装される．

我々は先行研究 [3] において，ホームネットワーク内の様々なセンサデバイスを標準的なサービスとして利用可能にする**センササービスフレームワーク** *(SSF)* を提案している．SSF はサービス指向アーキテクチャ(SOA) [4] の考えに基づき，温度センサや照度センサ等の各種センサを，デバイスやプラットフォームによらない Web サービスとすることで，アプリケー

ション-機器-センサ間の疎結合を実現する．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できる．

また SSF に加えて，時間的な制約を含んだより高度なコンテキストを定義・推定可能とするタイミング制約 [5] を提案している．具体的には，逐次的タイミング制約および継続的タイミング制約という 2 種類のタイミング制約を導入している．逐次的制約は，あるコンテキスト $C1$ が起こってから n 秒以内に別のコンテキスト $C2$ が起こるというもの，一方，継続的制約はあるコンテキスト C が m 秒間起こり続けるというものである．これらの制約を組み合わせることで，より高度なコンテキストを定義することができる．

しかしながら，タイミング制約の導入によって，より高度なコンテキストを定義可能になった一方で，コンテキスト作成には既存コンテキストの利用や作成作業に問題点がある．具体的には，単体もしくは複数の既存コンテキストを組み合わせるので，既存コンテキストの成立・不成立をどのセンササービスで検出可能なのか，どのような条件式で成立の判断をしているのか，そもそもコンテキスト自体が存在しているのか，といった詳細を知らなければ組み合わせることはできない．またコンテキスト間のタイミング制約を検出するためには，コンテキストが登録されているセンササービスや時間を測定する Web サービスといった複数の Web サービスを利用し，開発者がコンテキストの推定から時間的な制約のロジックまで実装しなければならず，作成作業が複雑化している．

そこで本研究では，コンテキストの詳細を統括・管理する ContextRegistry サービスを実装する．ContextRegistry サービスでは HNS 内すべてのセンササービスから登録されているコンテキストの詳細を取得し，統括・管理する．これによって開発者は既存のコンテキストを利用する際に，各センササービスを呼び出す必要はなく，一括して ContextRegistry サービスからコンテキストの詳細を取得可能となる．

また ContextRegistry サービスから既存コンテキストを検索・利用し，タイミング制約付きコンテキストの登録を容易に行える TimingContextSensor サービスを実装する．これにより開発者は高度コンテキストを容易に作成できるようなり，作成した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の開発も容易に行うことが可能になる．ケーススタディでは，我々が開発しているホームネットワーク [6] において，入室／退室の自動判別を行う入退室コンテキスト，および，TV が一定時間ついたままであることを検知する TV つけっぱなしコンテキストを実現し，提案手法の有効性を示す．

2. 先行研究：HNS のためのセンササービスフレームワーク

2.1 ホームネットワークシステム

ホームネットワークシステム *(HNS)* は，宅内の家電や設備機器をネットワークに収容して，付加価値サービスを実現するシステムである．TV や DVD，カーテン，エアコン，空気清浄機等の機器がネットワークに接続され，宅内外を問わない機器制御，消費電力振り返り [7]，複数機器の連携サービス [8] といった様々なサービス・アプリケーションが実現される．

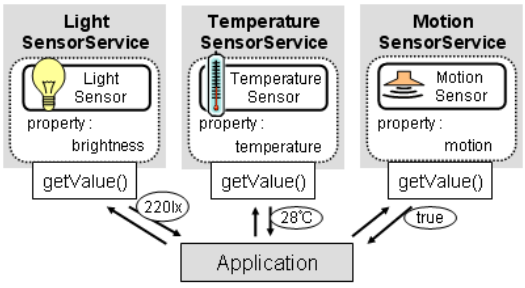

図 1　標準的なインタフェースによるセンサ値の取得

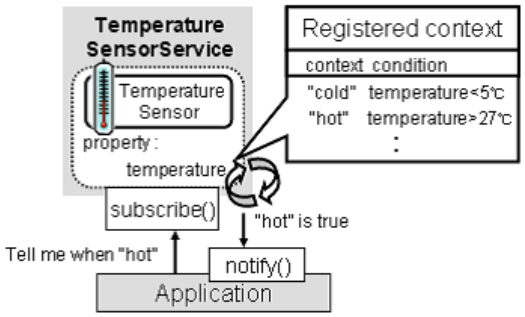

図 2　コンテキスト条件の登録とコンテキスト推定

我々の研究室では，サービス指向アーキテクチャ(SOA) [4] を HNS に適用し，各家電の機能を Web サービスとして利用できる HNS 環境 ”CS27-HNS”を開発している [6]．CS27-HNS では機器依存の制御方法や通信プロトコルを Web サービスでラップしており，全ての機器の機能を SOAP または REST 形式の Web サービスとして利用できる．例えば，テレビのチャンネルを 6ch にするには，`http://cs27-hns/TVService/setChannel?channel=6` のような URL にアクセスするだけで良い．

2.2 センササービスフレームワーク (SSF)

センササービスフレームワーク *(SSF)* [3] [9] は，様々なセンサデバイスを CS27-HNS 内の Web サービスとして容易に配備可能とするアプリケーションフレームワークとして開発された．

サービス化されたセンサ (以下，**センササービス**) 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．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．例えば，温度センサであれば temperature(℃) という摂氏温度を，照度センサであれば brightness(lux) という照度をプロパティとして持っており，その値をセンササービスの `getValue()` メソッドから取得できる（図 1 参照）．

また，各センササービスは自身のプロパティ値の変動を定期的に監視しており，登録された条件式 (**コンテキスト条件**と呼ぶ) に基づいてコンテキストの検出が行える（図 2 参照）．例えば，温度センサに `hot` という名前のコンテキストを `temperature >`

27 というコンテキスト条件で登録する (コンテキスト名と条件をつないで `[hot:temperature > 27]` と書くことにする)．温度センサは temperature を監視し続け，値が 27 ℃ より大きくなったときにコンテキスト `hot` を検出する．コンテキスト条件の登録は `register()` メソッドを利用して行う．登録されたコンテキストは `subscribe()` メソッドによって，任意の Web サービス呼び出しと関連付けることができる．この Web サービスに HNS の家電サービスを指定することで，**コンテキストアウェアサービス** を容易に構築できる．例えば，`subscribe(hot, http://cs27-hns/AirConditionerService/on)` とすれば，暑いときにエアコンを自動でつけるサービスを実現できる．

2.3 タイミング制約を含むコンテキスト

さらに我々は，単体あるいは複数のコンテキスト間で満たさなければならない時間的な制約 (**タイミング制約**) を提案し，高度なコンテキストの定義・推定を可能にした [5]．具体的には逐次的タイミング制約および継続的タイミング制約という 2 種類のタイミング制約を導入した．逐次的制約は，あるコンテキスト C_1 が起こってから n 秒以内に別のコンテキスト C_2 が起こるというもの，一方，継続的制約はあるコンテキスト C が m 秒間起こり続けるというものである．

例えば，部屋に入室したというコンテキストでは，玄関扉が開くというコンテキストが起こってから 5 秒以内に玄関ホールを人が通るというコンテキストが起こる，というように逐次的タイミング制約で定義できる．また，家電つけっぱなしにしているというコンテキストでは，家電の電源が ON というコンテキストが１０分間起こり続ける，といように継続的タイミング制約で定義できる．

これらの制約を組み合わせることで，より高度なコンテキストを定義することができる．

2.4 課　　題

タイミング制約の導入によって，より高度なコンテキストを定義可能になった一方で，コンテキスト作成には以下のような問題点がある．

P1:既存コンテキストがどのセンササービスに登録されているか，どのような条件式なのか，といったコンテキストの詳細を知らなければならない．

P2:タイミング制約を含むコンテキストの作成作業自体が複雑化している．

単体もしくは複数の既存コンテキストを組み合わせるので，既存コンテキストの成立・不成立をどのセンササービスで検出可能なのか，どのような条件式で成立の判断をしているのか，そもそもコンテキスト自体が存在しているのか，といった詳細を知らなければ組み合わせることはできない．またコンテキスト間のタイミング制約を検出するためには，コンテキストが登録されているセンササービスや時間を測定する Web サービスといった複数の Web サービスを利用し，開発者がコンテキストの推定から時間的な制約のロジックまで実装しなければならず，作成作業が複雑化している．

本論文では，これらのタイミング制約を含むコンテキスト作成の際の問題点を改善するために，コンテキストの詳細を統括・管理する ContextRegistry サービスを提案し，それを利用したタイミング制約を含むコンテキスト登録 Web サービスを実装する．

3. 提 案 手 法

3.1 キーアイデア

2.4 で述べた問題点を改善するために本論文では次の 2 つのキーアイデアを実現する．

K1:一つの Web サービスを呼び出すだけで，HNS 内すべてのコンテキストの詳細を取得できる．

K2:タイミング制約を含むコンテキストの登録を容易にする．

K1 を実現するためにコンテキストの詳細を統括・管理する ContextRegistry サービスを実装した．ContextRegistry サービスでは HNS 内すべてのセンササービスから登録されているコンテキストの詳細を取得し，統括・管理する．これによって開発者は既存のコンテキストを利用する際に，各センササービスを呼び出す必要はなく，一括して ContextRegistry サービスからコンテキストの詳細を取得可能となる．

また K2 を実現するためにタイミング制約を含むコンテキスト登録 Web サービス TimingContextSensor サービスを実装した．TimingContextSensor サービスでは，タイミング制約を含むコンテキストを登録するメソッドを公開する．ContextRegistry サービスで管理されている既存コンテキストとタイミング制約の時間を指定するだけで容易にコンテキストの登録が可能となる．

3.2，3.3 で ContextRegistry サービスの詳細とサービス利用の流れを，3.4 で TimingContextSensor サービスの詳細とタイミングコンテキストの登録について説明する．

3.2 ContextRegistry サービス

ContextRegistry サービスは，HNS 内のコンテキストの詳細を統括・管理する Web サービスである．ContextRegistry サービスが扱うコンテキストの情報は，コンテキスト名，コンテキスト成立/不成立を取得する呼び出し URL，コンテキスト条件式の 3 つである．HNS 内のコンテキストに対して，これら 3 種類の情報を取得するために，getContextList メソッドを公開する．さらにコンテキスト名から，各コンテキストの判定を行う getContextStatus メソッドを公開することで，コンテキストごとに各センササービスを直接呼び出す必要性をなくし，ContextRegistry サービスのみで各コンテキストの条件判定を可能にする．

またコンテキストは各センササービスに登録して管理されているため，ContextRegistry サービスでは HNS 内のセンササービスの情報も管理する．センササービスの情報は，センサ名，サービス URL の 2 つである．センササービスに対してもコンテキストと同様に情報を取得するため，getSensorList メソッドを公開する．

図 3 は，ContextRegistry サービスが提供する主要なクラス，メソッドのクラス図である．

各クラス，各メソッドの詳細を以下に示す．

Sensor クラス: センサ名，センササービス呼び出し URL，

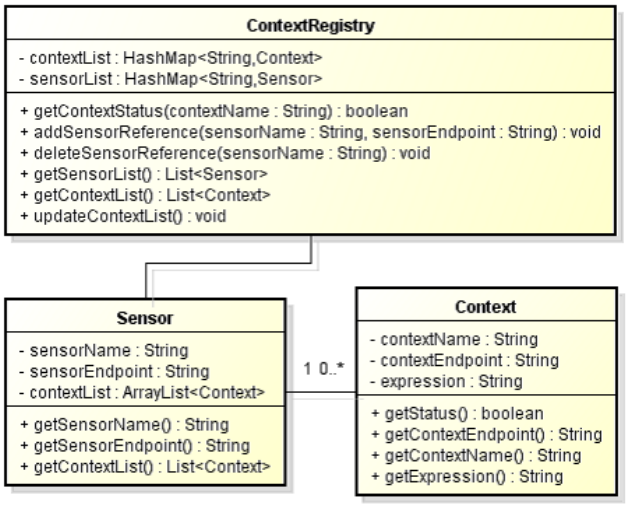

図 3　ContextRegistry サービスのクラス図

センサに登録されているコンテキストのリストをプロパティとして持ち，それぞれの値を取得する get メソッドを持つ．

Context クラス:　コンテキスト名，コンテキスト条件を判定する URL，コンテキスト条件式をプロパティとして持ち，それぞれの値を取得する get メソッドとコンテキストの成立/不成立を取得する getStatus メソッドを持つ．

getContextStatus:　コンテキストが成立しているかどうかを検出する．引数に「コンテキスト名」を指定する．

getSensorList:　ContextRegistry サービスに登録されているセンササービスのセンサ名，サービス URL をリストで取得する．

getContextList:　各センササービスに登録されているコンテキストのコンテキスト名，コンテキスト条件式，コンテキスト条件を判定する URL をリストで取得する．

updateContextList:　各センササービスから登録されているコンテキストの詳細を取得し，コンテキストリストを更新する．

addSensorReference:　ContextRegistry サービスにセンササービスを登録する．引数に「センサ名，サービス URL」を指定する．登録されたセンサからコンテキストの詳細を取得するようになる．

deleteSensorReference:　登録されたセンササービスを削除する．

3.3　ContextRegistry サービスの利用の流れ

実際の ContextRegistry サービスの利用の流れについて説明する．

まず初めに ContextRegistry サービスが管理するセンササービスを addSensorReference メソッドを利用し，登録する．引数にはセンサ名とセンササービスの URL を指定する．具体的に，ドアの開扉を検知するセンササービス (DoorSensorService) を登録するには以下の呼び出しとなる．

#DoorSensorService の登録

```
http://cs27-hns/ContextRegistryService/addSensorReference?name=DoorSensor&url=http://cs27-hns/DoorSensorService/
```

ContextRegistry サービスにセンササービスを登録することで，各センササービスに登録されているコンテキストの詳細をサービスが取得し，管理する．他のサービスからセンササービスやコンテキストの情報を利用したい場合，ContextRegistry サービスの getSensorList メソッドおよび getContextList メソッドから情報を取得可能であり，各センササービスにアクセスする必要がない．

またコンテキスト成立の検出に関しては getContextStatus メソッドを利用することで可能となる．具体的に，DoorSensorService に登録されているドア開扉コンテキスト (DoorOpen) を検知するには以下の呼び出しとなる．

#ドアの開扉コンテキストの検出

```
http://cs27-hns/ContextRegistryService/getContextStatus?name=DoorOpen
```

このようにコンテキストの検出を行う場合にも，各センササービスにアクセスする必要はなく，ContextRegistry サービスのみにアクセスすればよい．そのため，他のサービスから HNS 内のセンササービスやコンテキストを利用する際には，他のサービス自身がセンササービスの呼び出し URL などの情報を管理する必要はなく，ContextRegistry サービスから必要な情報をすべて取得することが可能となる．

3.4　TimingContextSensor

TimingContextSensor サービスは，既存コンテキストを組み合わせて，タイミング制約を含むコンテキストの登録を容易に行える Web サービスである．逐次的タイミング制約と継続的タイミング制約を含むコンテキストを登録するために，それぞれに対する register メソッド（registerSequentialContext, registerContiueousContext）を公開する．組み合わせるコンテキストは 3.2 で説明した ContextRegistry サービスから検索・利用する．

また従来の SSF と同様に subscribe メソッドを公開し，タイミング制約を含むコンテキストに基づくコンテキストアウェアアプリケーションの開発を容易にする．

図 4 は，TimingContextSensor サービスが提供する主要なクラス，メソッドのクラス図である．

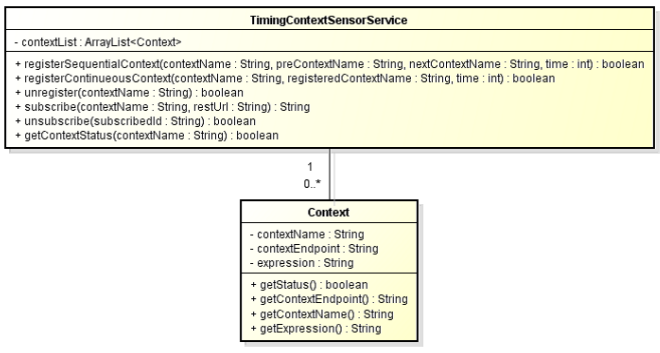

図 4　TimingContextSensor サービスのクラス図

Context クラスは 3.2 で説明した，ContextRegistry サービスと同様のクラスである．他の各メソッドの詳細は以下に示す．

registerSequentialContext: 逐次的タイミング制約を含むコンテキストを登録する．引数には「コンテキスト名，既存コンテキスト 1，既存コンテキスト 2，時間」を指定する．例えば，`registerSequentialContext(`C_S`,` C_1`,` C_2`,` n`)` とすると，あるコンテキスト C_1 が起こってから n 秒以内に別のコンテキスト C_2 が起こるというコンテキスト C_S が登録できる．また引数の既存コンテキストは ContextRegistry サービスに登録されているコンテキストから利用する．

registerContiueousContext: 継続的タイミング制約を含むコンテキストを登録する．引数には「コンテキスト名，既存コンテキスト，時間」を指定する．例えば，`registerContiueousContext(`C_C`,` C_3`,` m`)` とすると，あるコンテキスト C_3 が m 秒間起こり続けるというコンテキスト C_C が登録できる．また引数の既存コンテキストは ContextRegistry サービスに登録されているコンテキストから利用する．

getContextList: 登録されているタイミングコンテキストのリストを取得する．リストはコンテキスト名とコンテキスト条件式からなる．

getContextStatus: 引数に「コンテキスト名」を指定し，指定されたタイミングコンテキストが成立しているかどうかを取得する．

subscribe: 2.2 で説明した既存のセンササービスフレームワークと同様のメソッド．引数に「登録されたタイミングコンテキスト名，Web サービス呼び出し URL」と指定することで，任意の Web サービスと関連付けることができる．

4. ケーススタディ

CS27-HNS に配備済みのセンササービス，家電操作サービスと実装した TimingContextSensor サービス，ContextRegistry サービスを利用して，タイミング制約付きのコンテキストを登録し，そのコンテキストを利用したコンテキストアウェアアプリケーションを実際に作成する．各センサは，SSF によってセンササービスとして利用可能となっており，ContextRegistry サービスに登録されているものとする．

4.1 自動照明サービスの実現（逐次的タイミング制約の利用）

ドアの開閉を検知するドアセンササービスと人を検知する人感センササービスの 2 つのサービスを利用して,「入室」と「退室」を別個のコンテキストとして検知できるように登録し，それぞれのコンテキストに対して照明 (LightService) の on/off を実行する自動照明サービスの実現例を示す．

ドアセンササービスには，HNS 入り口ドアの開扉を検知するコンテキストを，DoorOpen という名前で登録している．人感センササービスには，HNS 入り口付近の人を検知するコンテキストを，HumanDetect という名前で登録している．

コンテキスト「入室」は，ドアの開扉⇒人を検知の順でコンテキストが成立したときに検知すると逐次的タイミング制約を用いて定義したい．TimingContextSensor サービスの registerSequeintialContext メソッドからコンテキストを登録し，subscribe メソッドで登録したコンテキストに対して Web サービスを関連付ける．以下に，入室に対して自動で照明が on になるコンテキストアウェアアプリケーションの登録するときの Web サービス呼び出し系列を示す．

#STEP1:入室コンテキストの登録

```
http://cs27-hns/TimingContextSensorService/registe
rSequeintialContext?name=Enter&context1=DoorOpen&c
ontext2=HumanDetect&time=5
```

#STEP2:入室コンテキストとライトの `on` を関連付け

```
http://cs27-hns/TimingContextSensorService/subscri
be?context=Enter&notify=http://cs27-hns/LightServi
ce/on
```

コンテキスト「退室」は，逆に人を検知⇒ドアの開扉の順でコンテキストが成立したときに検知すると逐次的タイミング制約を用いて定義したい．以下に，退室に対して自動で照明が off になるコンテキストアウェアアプリケーションの登録するときの Web サービス呼び出し系列を示す．

#STEP1:退室コンテキストの登録

```
http://cs27-hns/TimingContextSensorService/registe
rSequeintialContext?name=Leave&context1=HumanDetec
t&context2=DoorOpen&time=5
```

#STEP2:退室コンテキストとライトの `off` を関連付け

```
http://cs27-hns/TimingContextSensorService/subscri
be?context=Leave&notify=http://cs27-hns/LightServi
ce/off
```

以上の手順により入退室コンテキストそれぞれに対して照明の on/off を実行する，自動照明サービスが実現できた．

さらに登録したコンテキスト自体は以下の呼び出しで検出できる．

```
http://cs27-hns/TimingContextSensorService/getCont
extStatus?name=Enter
```

また登録したコンテキストは ContextRegistry サービスにも登録されるので以下の呼び出しでも検出できる．

```
http://cs27-hns/ContextRegistryService/getContextS
tatus?name=Enter
```

作成した 2 種類のコンテキストとコンテキストアウェアアプリケーションは，CS27-HNS で実際に稼動している．これらのコンテキストは ContextRegistry サービスに登録されているため，他のコンテキスト導出に再利用することもできる．

4.2 TV つけっぱなし通知サービスの実現（継続的タイミング制約の利用）

次に，家電操作サービスによる家電の状態取得から TV の電源が ON であるというコンテキストを利用して，家電がつけっぱなしであるというコンテキストを検知できるように登録し，そのコンテキストに対して HNS 内に通知行う TV つけっぱなし通知サービスの実現例を示す．

TV の電源が ON であるというコンテキストを，TVon という名前で登録している．

コンテキスト「TV つけっぱなし」は，TVon のコンテキス

トが一定時間成立し続けたときに検知すると継続的タイミング制約を用いて定義したい．TimingContextSensor サービスの registerContiueousContext メソッドからコンテキストを登録し，subscribe メソッドで登録したコンテキストに対して Web サービスを関連付ける．以下に，TV つけっぱなしに対して HNS 内への通知を行うコンテキストアウェアアプリケーションの登録するときの Web サービス呼び出し系列を示す．

#STEP1:TV つけっぱなしコンテキストの登録

```
http://cs27-hns/TimingContextSensorService/registe
rContiueousContext?name=TVLeftOn&context1=TVon&con
text2=HumanDetect&time=600
```

#STEP2:TV つけっぱなしコンテキストと HNS 内への通知を関連付け

```
http://cs27-hns/TimingContextSensorService/subscri
be?context=TVLeftOn&notify=http://cs27-hns/NotifyS
ervice/notify
```

さらに登録したコンテキスト自体は以下の呼び出しで検出できる．

```
http://cs27-hns/TimingContextSensorService/getCont
extStatus?name=TVLeftOn
```

また登録したコンテキストは ContextRegistry サービスにも登録されるので以下の呼び出しでも検出できる．

```
http://cs27-hns/ContextRegistryService/getContextS
tatus?name=TVLeftOn
```

作成したコンテキストとコンテキストアウェアアプリケーションは，CS27-HNS で実際に稼動している．このコンテキストは ContextRegistry サービスに登録されているため，他のコンテキスト導出に再利用することもできる．

5. ま と め

本論文では，ホームネットワークシステム (HNS) において，ContextRegistry サービスと TimingContextSensor サービスを実装することで 2.4 で述べた問題点 P1，P2 を改善した．

ContextRegistry サービスにより HNS 内のコンテキストは一括に管理され，ContextRegistry サービスを呼び出すだけでコンテキストの詳細を取得することが可能となった．タイミング制約を含むコンテキストを作成する際の既存コンテキストの詳細は，各センササービスごとに呼び出す必要がなく，ContextRegistry サービスのみを呼び出せばよい．

また TimingContextSensor サービスにより複雑化していたタイミング制約を含むコンテキストの登録作業を容易に行えるようになった．具体的には逐次的，継続的タイミング制約のそれぞれに対応した register メソッドを実装し，開発者は引数を指定するだけで登録が可能になった．さらに subscribe メソッドにより，作成したタイミングコンテキストを利用したコンテキストアウェアアプリケーションの開発も容易となった．

これらの提案手法に基づき，実際の HNS 環境である CS27HNS において 2 種類のコンテキストアウェアアプリケーションを実現し，提案法の有効性を示した．

提案手法によって，より高度なコンテキストの作成が容易になった一方で，コンテキスト条件自体が複雑化しており，登録されているコンテキスト間の競合を事前に検証するのが困難である．今後は，既存コンテキストの競合や矛盾などを考慮したコンテキスト作成支援の枠組みを開発していきたい．また，エンドユーザによるサービス作成支援 [10] も視野にいれ，作成した高度コンテキストを用いて誰でも簡単にサービスが作成できるようなユーザインタフェースの開発を目指したい．

文 献

[1] Bill N. Schilit and Norman Adams and Roy Want, “Context-Aware Computing Applications,” Proc. the 1st IEEE Workshop on Mobile Computing Systems and Applications (WMCSA), pp.85-90, 1994.

[2] Anind K. Dey and Gregory D. Abowd, “Towards a Better Understanding of context and context-awareness,” Proc. the 1st International Symposium on Handheld and Ubiquitous Computing (HUC), pp.304-307, 1999.

[3] 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.

[4] Thomas Erl, “Service-Oriented Architecture: Concepts, Technology and Design,” PRENTICE HALL 2008.

[5] 丸尾 彰宏, 松尾 周平, まつ本 真佑, 中村 匡秀, “サービス指向ホームネットワークにおける複数センサとタイミング制約を用いた高度コンテキストの抽出,” 電子情報通信学会 IN 研究会, vol.110, no.449, pp.187-192, March 2011.

[6] 田中章弘, 中村匡秀, 井垣宏, 松本健一, “Web サービスを用いた従来家電のホームネットワークへの適応,” 電子情報通信学会技術研究報告, Vol.105, No.628, pp.067-072, March 2006.

[7] 福田 将之, 瀬戸 秀晴, 坂本 寛幸, 井垣 宏, 中村 匡秀, “ホームネットワークシステムにおける電力消費振り返りサービスの提案,” 電子情報通信学会技術研究報告, vol.109, no.272, pp.029-034, November 2009.

[8] 井垣 宏, 中村 匡秀, 玉田 春昭, 松本 健一, “サービス指向アーキテクチャを用いたネットワーク家電連携サービスの開発,” 情報処理学会論文誌, Vol.46, No.2, pp.314–326, February 2005.

[9] 坂本 寛幸, 井垣 宏, 中村 匡秀, “SMuP: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,” ウィンターワークショップ 2010・イン・倉敷 論文集, vol.2010, no.3, pp.73-74, January 2010.

[10] 松尾 周平, 瀬戸 英晴, 坂本 寛幸, 井垣 宏, 中村 匡秀, “場所情報を用いたセンサ検索と類似条件提示によるコンテキスト構築支援環境 : Sensor Service Binder 2.0,” 電子情報通信学会技術報告, vol.109, no.327, pp.59-64, December 2009.