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

丸尾　彰宏†　　松尾　周平†　　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,matsuo}@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，コンテキストは単一のセンサ値，もしくは複数のセンサ値の論理積，論理和と算術演算で定義可能であった．しかし，これらのコンテキストは現在の値から導出できるものに限られており，コンテキスト間の時間的な制約を表現できなかった．本研究ではセンササービスフレームワークを拡張し，2 種類の時間的な制約 (タイミング制約) を含むより高度なコンテキストへ対応させる．またタイミング制約を指定するためにタイマーサービスを実装し，実際のホームネットワーク内での高度コンテキストの登録・検出の手順を示す．

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

On deriving high-level contexts using multiple sensors and timing constraints in service-oriented home network

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

† Kobe University rokkoudaityou 1-1, nada-ku, Kobe, Hyogo, 657-8501, Japan
E-mail: †{maruo,matsuo}@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 the SSF, a context was defined by a condition over a single sensor, or a condition over multiple sensors derived by logical or arithmetic operations. However, the contexts were limited to the ones that can be defined by present values of the sensors. This paper extends the SSF so that it can specify two kinds of *timing constraints* in the contexts. Also, we present the timer service to implement the timing constraints within the SSF. We finally demonstlate how the high-level contexts with the timing constraints are registered and detected in a real home network system.

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，複数のセンササービスを組み合わせてより複雑なコンテキスト推定ができる**センサマッシュアップ**

ラットフォーム (SMuP) [5] を提案している．これにより例えば「温度が２８度以上かつ湿度が４０%以下」といった温度センサと湿度センサの条件を組み合わせたより高度なコンテキストを推定することができる．

しかしながら，SSF/SMuP で推定することのできるコンテキストは，単一もしくは複数のセンサ値の論理積，論理和と算術演算で構築されたものであり，センサの現在の値から導出できるコンテキストのみであった．例えば「玄関扉を開けて２秒後に玄関ホールを通った」や「ソファにずっと座っている」といったコンテキスト内の時間的な制約を扱うことができない．

そこで本研究では，従来の SSF/SMuP を拡張し，時間的な制約 (**タイミング制約**と呼ぶ) を含んだより高度なコンテキストを定義・推定可能とすることを目的とする．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する．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．

また既存の SSF/SMuP に新たにタイマーサービスを追加することで，タイミング制約付きコンテキストを推定できるセンササービスを実現できることを示す．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センサ，および，ソファで一定時間寝ていることを検知する仮眠センサを実現し，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．

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] 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）．

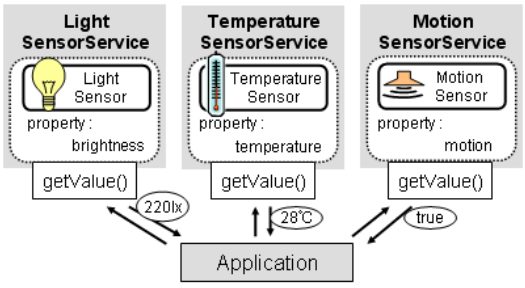

図 1　標準的なインタフェースによるセンサ値の取得

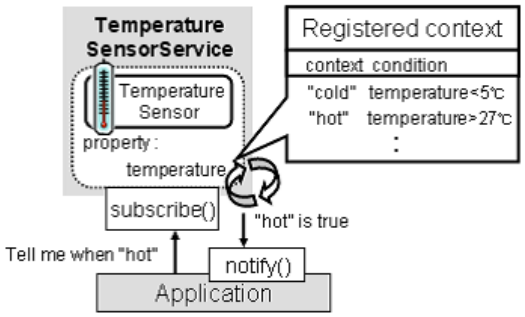

図 2　コンテキスト条件の登録とコンテキスト推定

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 センサマッシュアッププラットフォーム (SMuP)

さらに我々は，複数のセンササービスを組み合わせてより高度なセンササービスを構築できる**センサマッシュアッププラットフォーム** (SMuP) [5] を開発している．SMuP は，既存のセンササービスのプロパティやコンテキスト条件を算術演算や論理演算で結合した，仮想的なセンササービスを動的に作成できる．例えば，複数の照度センサを用いて部屋の明るさの平均値をとるセンサや，温度センサと湿度センサを組み合わせて不快

指数を算出するセンサ,「温度が２８度以上かつ湿度が４０%以下」の条件を判断するセンサ等が容易に作成でき，単一センサではできなかったコンテキストの検出が可能となる．

SMuP では，createSensor() メソッドで仮想センサを作成，addProperty() メソッドで既存のセンサを組み合わせた新しいプロパティを仮想センサに追加する．作成された仮想センサはセンササービスとして扱えるので，SSF のセンササービス同様，register() および subscribe() が利用できる．

2.4 課　　題

従来の SSF や SMuP では**現在のセンサ値**から導出できるコンテキスト条件のみを対象としており，コンテキスト間の時間的な制約を表現することができない．例えば，以下のようなコンテキストは SSF や SMuP で定義・検出できなかった．

(P1:帰宅した):「玄関扉を開けて２秒後に玄関ホールを通った」

(P2:休憩している):「ソファにずっと座っている」

(P3:休憩中にテレビを見始める):「ソファに座っていてテレビを見始めた」

本論文では，先行研究を拡張し上記のような時間的制約を含む高度なコンテキストを検出することを目的とする．

3. 提 案 手 法

3.1 キーアイデア

先行研究での課題を解決するため，本研究ではコンテキスト条件に**タイミング制約**を指定できるように拡張する．ここでタイミング制約とは，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を指す．例えば 2.4 で述べたコンテキストは，以下のようにタイミング制約を用いたコンテキスト条件で検出することができる．

(P1 検出条件):玄関扉のドアセンサが反応してから５秒以内に玄関ホールの人感センサが反応する．

(P2 検出条件):ソファの着座センサが１５秒間反応し続ける．

(P3 検出条件):ソファの着座センサが１５秒間反応してから１０秒以内にテレビの電源が ON になる

具体的なタイミング制約として，本研究では次の２種類の制約を導入することとする：(1) 逐次的タイミング制約，(2) 継続的タイミング制約．

3.2 逐次的タイミング制約

逐次的タイミング制約は，あるコンテキストが検出されてから，一定時間内に別のコンテキスト条件が検出されるという時間的な制約である．より厳密には「あるコンテキスト C1 が成立してから，n 秒以内にコンテキスト C2 が成立する」という新たなコンテキスト T1 を，以下のように記述する．

$$[T1 : \#n\,[C1, C2]]$$

例えば，ドアセンサが玄関の開扉を検知したコンテキストを [EntranceOpen:DoorSensor.isOpened==true]，玄関ホールの人感センサ (MotionSensor1) が人を検知したというコンテキストを [HumanDetect:MotionSensor1.motion==true] と定義する．このとき，3.1 の (P1 検出条件) は，

```
[ComingHome: #5[EntranceOpen, HumanDetect]]
```

と記述される．これにより，2.4 の (P1：帰宅した) を検出する．一方，上記２つのコンテキストの順番を逆にして，

```
[LeavingHome: #5[HumanDetect, EntranceOpen]]
```

とすることで,「外出した」というコンテキストも定義できる．

さらに３つ以上のコンテキストにも拡張可能である．例えば，洗面所の人感センサ (MotionSensor2) が人を検知したというコンテキストを，[WashRoom:MotionSensor2.motion==true] と定義する．これに先ほどの ComingHome と組み合わせ，

```
[WashHands: #60[ComingHome, WashRoom]]
```

とすることで，帰宅して 60 秒以内に洗面所に来た，すなわち「帰宅後手を洗いに来た」というコンテキストを定義できる．

3.3 継続的タイミング制約

継続的タイミング制約とは，あるコンテキストが一定の時間の間成立し続けるという時間的な制約である．より厳密には「あるコンテキスト C3 がｎ秒間成立し続ける」という新たなコンテキスト T2 を，以下のように記述する．

$$[T2 : @n\,[C3]]$$

例えば，着座センサがソファに人が座っていることを検知したコンテキストを [HumanSitting:SittingSensor.isSitting==true] と定義する．このとき，3.1 の (P2 検出条件) は，

```
[Rest: @15[HumanSitting]]
```

と記述される．これにより，2.4 の (P2：休憩している) を検出する．

さらに複数コンテキスト間の論理演算にも拡張可能である．例えば，リビングの人感センサ (MotionSensor3) が人を検知していないというコンテキストを [NotHumanDetect:MotionSensor3.motion==false]，暖房が付いているというコンテキストを [HeaterOn:Heater.power==true] と定義する．この２つのコンテキストを組み合わせ，

```
[WasteHeater: @600[NotHumanDetect && HeaterOn]]
```

とすることで，10 分間リビングに人がいないかつ暖房が付いている，すなわち「長時間リビングに人がいないのに暖房がついたままである」というコンテキストを定義できる．

3.4 逐次的制約と継続的制約の組み合わせ

また逐次的制約と継続的制約を組みわせることも可能である．テレビの電源が ON であるというコンテキストを [TVon:TV.power==true] と定義すると，前述の HumanSitting と組み合わせて，3.1 の (P3 検出条件) は，

```
[WatchTV: #10[@15[HumanSitting], TVon]]
```

と記述される．これにより，2.4 の (P3：休憩中にテレビを見始める) を検出する．

さらに複数コンテキスト間の論理演算を組み合わせることもできる．電気がついているというコンテキストを [LightOn:Light.power==true] と定義する．これに 3.2 で定義した LeavingHome と組み合わせて

```
[WasteLight: @600[LeavingHome && LightOn]]
```

とすることで，出かけていてかつ電気が付いている状態が 10 分間続いた，すなわち「長時間外出しているが電気が付いたま

まである」というコンテキストを定義できる．

3.5 タイミング制約評価のためのタイマーサービス

逐次的，継続的タイミング制約を含んだコンテキスト条件を評価するために，**タイマーサービス**を導入する．タイマーサービスは時間を測定するサービスであり，コンテキストが登録される SSF/SMuP からユーティリティとして利用される．

タイマーサービスは，`createTimer()`，`start()`，`stop()`，`isActive()` のメソッドを持つ．`createTimer()` は引数に制限時間をとりその時間を計測するタイマーを作成する．`start()` は作成したタイマーの計測を開始させ，`stop()` はタイマーの計測を止める．`isActive()` はタイマーの状態を取得する．

各タイマーは，`Waiting`，`Working`，`Expired` の 3 つの状態を持っている．`Waiting` はタイマーの計測が開始されていない状態，`Working` は計測が開始されてから制限時間内にある状態，`Expired` は時間切れの状態を表す．

3.6 逐次的タイミング制約の登録と検出

SSF/SMuP で作成されたセンササービス，および，タイマーサービスを利用して，逐次的タイミング制約を用いた高度コンテキストの登録，検出を行う手順を示す．

いま，SSF/SMuP に逐次的タイミング制約を用いたコンテキスト `[T1: #n [C1, C2]]` の登録要求が来たと仮定する．このとき SSF/SMuP は，コンテキスト T1 を検出する仮想センサを以下の手順で作成する．

STEP1： `createSensor()` で新しい仮想センサ S を作成．

STEP2： タイマーサービスを利用して，制限時間を n 秒のタイマー t を作成．

STEP3： C1 が登録されたセンササービスに対し，`subscribe(C1, t.start())` を実行．これにより，C1 成立時に t がスタートする．

STEP4： S の新しいプロパティとして `[T1:(C2==true && t.isActive()==Working)]` を定義，`addProperty()` で S に登録する．

上記の手順を行えば，T1 はタイマー t の制限時間内に C2 が成立する，すなわち「C1 成立から n 秒以内に C2 が成立する」という逐次的タイミング制約を表すプロパティとなり，S は T1 を検出する仮想センサとなる．

具体的に 3.2 で述べたコンテキスト `[ComingHome: #5[EntranceOpen, HumanDetect]]` の登録の流れを図 3 に示す．この図を利用して各 STEP を説明する．まず，STEP1 では `createSensor()` メソッドを用いて，新たな仮想センサ `ComingHomeSensor` を作成する．次に，STEP2 ではタイマーサービスの `createTimer()` メソッドを用いて，計測時間 5 秒のタイマー `CHTimer` を作成する．STEP3 ではコンテキスト `EntranceOpen` が登録された `DoorSensorService` に対し `subscribe()` メソッドを実行，`EntranceOpen` と `CHTimer.start()` を関連付ける．最後に STEP4 では，新しいプロパティ `[ComingHome: HumanDetect==true && CHTimer.isActive==Working` を作成し，`addProperty()` で仮想センサ `ComingHomeSensor` に追加する．`ComingHomeSensor` の `ComingHome` プロパティは，`EntranceOpen` が成立して 5 秒

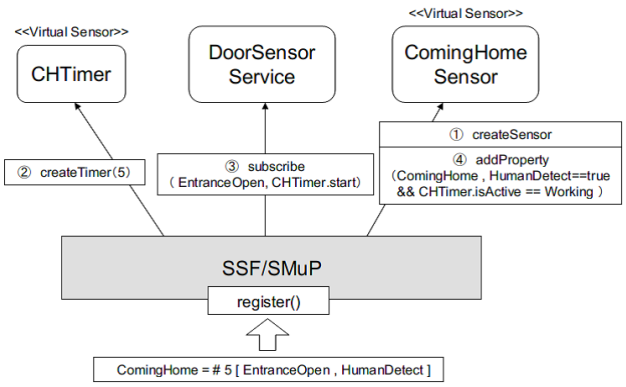

図 3 逐次的タイミング制約登録シーケンス

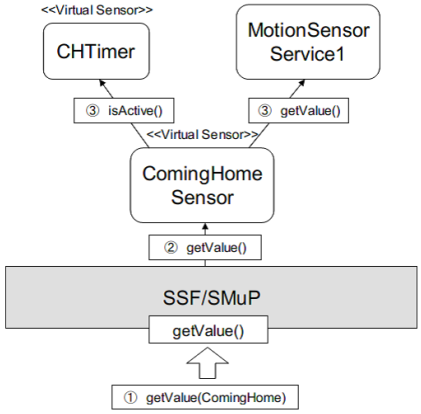

図 4 逐次的タイミング制約取得シーケンス

以内に `HumanDetect` が成立した時のみ真となる．

次に，コンテキスト `ComingHome` の検出の流れを図 4 を用いて説明する．コンテキスト成立の有無は，コンテキスト登録時に作成された `ComingHomeSensor` の `getValue()` メソッドを通じて取得できる．`ComingHomeSensor` からは，`HumanDetect` が登録されている `MotionSensorService1` と，`CHTimer` の `isActive()` メソッドにアクセスし，プロパティ `ComingHome` の真偽値を評価，コンテキスト成立の判断を通知する．

3.7 継続的タイミング制約の登録と検出

次に，SSF/SMuP に継続的タイミング制約を用いたコンテキスト `[T2: @n [C3]]` の登録要求が来たと仮定する．このとき SSF/SMuP は，コンテキスト T2 を検出する仮想センサを以下の手順で作成する．

STEP1： `createSensor()` で新しい仮想センサ S を作成．

STEP2： タイマーサービスを利用して，制限時間を n 秒のタイマー t を作成．

STEP3： C3 が登録されたセンササービスに対し，`subscribe(C3, t.start())` を実行．これにより，C3 成立時に t がスタートする．

STEP4： S の新しいプロパティとして `[T2:(C3==true && t.isActive()==Expired)]` を定義，`addProperty()` で S に登録する．

STEP5： さらに S の新しいプロパティとして `[AbortT2:(C3==false && t.isActive()==Working)]` を定義，`addProperty()`

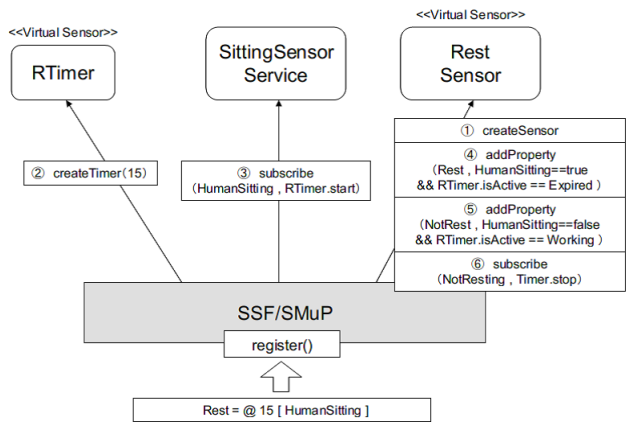

図 5　継続的タイミング制約登録シーケンス

で S に登録する．

STEP6： S に対し，subscribe(AbortT2, t.stop()) を実行．これにより，AbortT2 成立時に t がストップする．

上記の手順を行えば，T2 はタイマー t の制限時間中に C3 が成立し続ける，すなわち「C3 が n 秒間成立し続ける」という継続的タイミング制約を表すプロパティとなり，S は T2 を検出する仮想センサとなる．

具体的に 3.3 で述べたコンテキスト [Rest: @15[HumanSitting]] の登録の流れを図 5 に示す．この図を利用して各 STEP を説明する．

まず，STEP1 では createSensor() メソッドを用いて，新たな仮想センサ RestSensor を作成する．次に，STEP2 ではタイマーサービスの createTimer() メソッドを用いて，計測時間 15 秒のタイマー RTimer を作成する．STEP3 ではコンテキスト HumanSitting が登録された SittingSensorService に対し subscribe() メソッドを実行，HumanSitting と RTimer.start() を関連付ける．STEP4 では，新しいプロパティ[Rest: HumanSitting==true && RTimer.isActive==Expired] を作成し，addProperty() で仮想センサ RestSensor に追加する．RestSensor の Rest プロパティは，HumanSitting が 15 秒間成立し続けた時のみ真となる．STEP5 ではさらに新しいプロパティ[AbortRest: HumanSitting==false && RTimer.isActive==Working] を作成し，addProperty() で仮想センサ RestSensor に追加する．AbortRest プロパティは，15 秒間以内に HumanSitting が中断されたことを検知したとき真となる．STEP6 では RestSensor に対し subscribe() メソッドを実行，AbortRest と RTimer.stop() を関連付ける．

コンテキスト成立の有無はコンテキスト ComingHome と同様に，コンテキスト登録時に作成された RestSensor の getValue() メソッドを通じて取得できる．

4. 実装と評価

4.1 タイマーサービスの実装

タイミング制約を含んだ高度コンテキストの抽出のため，3.5 で述べたタイマーサービスを Web サービスとして実装し，我々の HNS 環境 CS27-HNS に配備した．実装に利用した環境と技術を下記に示す．

開発環境: Eclipse3.5.2

実装言語: Java(JRE1.5.0_18)

Web サーバ: Apache Tomcat 5.5

Web サービスエンジン: Apache Axis 2.1.3

4.2 ケーススタディ

CS27-HNS に配備済みのセンササービスと実装したタイマー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検出するセンサを実際に作成する．利用するセンサデバイスは Phidgets [9] 社の人感センサ（Motion Sensor 1111）2 つと圧力センサ（Force Sensor 1106）3 つである．各センサは，SSF によってセンサーサービスとして利用可能となっている．

入退室センサの実現

2 つの人感センササービス (MotionSensorService) 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入退室センサの実現例を示す．

MotionSensorService はプロパティとして motion を持ち，人を検知すると真，人がいない場合は偽の値となる．2 つのセンサデバイスは，CS27-HNS の入り口とリビング付近にそれぞれ配備され，サービスは玄関のほうが MotionSensorService1，リビングのほうが MotionSensorService2 となっている．それぞれのサービスにおいて，motion==true となるコンテキストを，HumanDetect1，HumanDetect2 という名前で登録している．

コンテキスト「入室」は，玄関⇒リビングの順で人の移動を感知したときに検出すると定義したい．よって逐次的タイミング制約を用いて，[Enter: #5[HumanDetect1, HumanDetect2]] と定義した．逆に，コンテキスト「退室」は，リビング⇒玄関の順で検出する．したがって，[Leave: #5[HumanDetect2, HumanDetect1]] と定義した．このコンテキストを SSF/SMuP に登録すると，3.6 で述べた手順でセンササービス，タイマーサービスへのアクセスが起こる．以下に，コンテキスト「入室」を登録するときの Web サービス呼び出し系列を示す．

```
#STEP1:仮想センサの作成
 http://cs27-hns/SMuP/createSensor?name=EnterLeaveS
 ensor
#STEP2:タイマーの作成
 http://cs27-hns/TimerService/createTimer?name=ETim
 er&time=5
#STEP3:MotionSensor1 とタイマーの関連付け
 http://cs27-hns/MotionSensorService1/subscribe?con
 text=HumanDetect1&notify=http://cs27-hns/ETimer/st
 art
#STEP4:HumanDetect2 とタイマーで仮想センサに登録
 http://cs27-hns/SMuP/EnterLeaveSensor/addProperty?
 name=Enter&property=HumanDetect2==true && ETimer==
 Working
```

登録したコンテキストは以下の呼び出しで検出できる．

```
 http://cs27-hns/SMuP/EnterLeaveSensor/getValue?pro
 perty=Enter
```

仮眠センサの実現

次に，CS27-HNS のリビングのソファに設置した 3 つの圧力センサを用いて,「ソファで仮眠をとっている」というコンテキストを検出するセンサを実現する．

圧力センササービス (ForceSensorService) は，ソファの左端，中央，右端に配備されている圧力センサにかかる圧力を，それぞれ SittingL, SittingC, SittingR という 3 つのプロパティで検出する．プロパティは，センサが圧力を検知する (センサの上に物体が乗る) と真の値となり，検知しないと偽の値となる．

いま，コンテキスト「ソファで仮眠をとっている」を，ソファに配備された 3 つのセンサがすべて圧力を検知し，その状態が一定時間の間続くことで検知したい．したがって，継続的タイミング制約を用いて，`[Sleeping: @60[SittingL && SittingC && SittingR]]` と定義する．

このコンテキストを SSF/SMuP に登録すると，3.7 で述べた手順でセンササービス，タイマーサービスへのアクセスが起こる．以下に，コンテキスト「ソファで仮眠をとっている」を登録する時の Web サービス呼び出し系列を示す．

#STEP1:仮想センサの作成

```
http://cs27-hns/SMuP/createSensor?name=SleepingSensor
```

#STEP2:タイマーの作成

```
http://cs27-hns/TimerService/createTimer?name=STimer&time=60
```

#STEP3:ForceSensor をまとめたセンサの作成とタイマーの関連付け

```
http://cs27-hns/SMuP/createSensor?name=AllForceSensor
http://cs27-hns/SMuP/AllForceSensor/addProperty?name=AllPress&property=SittingL==true && SittingC==true && SittingR==true
http://cs27-hns/SMuP/AllForceSensor/subscribe?context=AllPress&notify=http://cs27-hns/STimer/start
```

#STEP4:AllPress とタイマーで仮想センサに登録

```
http://cs27-hns/SMuP/SleepingSensor/addProperty?name=Sleeping&property=AllPress==true && ETimer==Expired
```

#STEP5:AbortSleeping を仮想センサに登録

```
http://cs27-hns/SMuP/SleepingSensor/addProperty?name=AbortSleeping&property=AllPress==false && ETimer==Working
```

#STEP6:AbortSleeping とタイマーの関連付け

```
http://cs27-hns/SMuP/SleepingSensor/subscribe?context=AbortSleeping&notify=http://cs27-hns/STimer/stop
```

登録したコンテキストは以下の呼び出しで検出できる．

```
http://cs27-hns/SMuP/SleepingSensor/getValue?property=Sleeping
```

作成した 2 種類の仮想センサは，CS27-HNS で実際に稼動している．これらのセンサはセンササービスとして登録されているため，他のコンテキスト導出に再利用したり，任意の Web サービスと関連付けてコンテキストアウェアサービスを実現することもできる．

5. ま と め

本論文では，ホームネットワークシステム (HNS) におけるセンササービスを対象として，時間制約を考慮したより高度なコンテキストを検出する手法を提案した．具体的には，我々の先行研究であるセンササービスフレームワーク (SSF) およびセンサマッシュアッププラットフォーム (SMuP) を拡張し，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 種類の時間制約をコンテキスト条件に指定できるようにした．さらに，タイマーサービスを導入して，これら 2 種類のタイミング制約付きのコンテキスト条件を評価する方法を提案した．提案手法に基づき，実際の HNS 環境である CS27-HNS において 2 種類のセンサを実現し，提案法の有効性を示した．

提案手法によって，より複雑なコンテキスト検出可能になった一方で，コンテキスト条件の作成作業自体が複雑化している．今後は，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] 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.

[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] Phidgets Inc. Unique and Easy to Use USB Interface, http://www.phidgets.com/

[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.