社団法人　情報処理学会研究報告
IPSJ SIG Technical Report

2008－MBL－44　(12)
2008－UBI－17　(12)
2008／3／5

スマートオブジェクトサービスのための時間的関係性を考慮したイベント記述方式の提案

米澤拓郎[1]，中澤仁[1]，榊原寛[1]，　國頭吾郎[2]，永田智大[2]，徳田英幸[1,3]
[1] 慶應義塾大学大学院 政策・メディア研究科
[2] NTT ドコモ 総合研究所
[3] 慶應義塾大学 環境情報学部

本研究ではセンサノードが取り付けられた日常物（以下スマートオブジェクト）を利用し，多様なサービスをユーザ自身が構築できる環境を実現するため，時間的関係性を考慮したイベント記述方式を提案する．スマートオブジェクトサービスを構成するモノに関するイベントの検知を行うためには，イベントをセンサの物理値に置き換えてモデリングする必要がある．従来の手法ではセンサの閾値設定及び単純な論理演算を用いたイベントモデリングが一般的であり，より複雑なイベントを記述するためには汎用プログラミング言語での作成をユーザに強いる必要があった．本研究ではイベント同士の具体的な時間的関係性を記述可能とすることにより，より多様なイベントをユーザ自身が容易に記述できる手法を提案し，エンドユーザ主導のサービス開発環境構築を目指す．

An Event Desciption Language Using Temporal Relation for Smart Object Services

Takuro Yonezawa[1],Jin Nakazawa[1], Hiroshi Sakakibara[1],
Goro Kunito[2], Tomohiro Nagata[2], Hideyuki Tokuda[1,3]

[1] Graduate School of Media and Governance, Keio University
[2] Research Laboratories, NTT DoCoMo, Inc.
[3] Faculty of Environmental Information, Keio University

This paper proposes event description language using temporal relation for creating various services with smart objects. By attaching sensor nodes to everyday objects, users can augment the objects digitally and take the objects into various services. When creating such smart object services, users should define event using physical value from sensors. The most common event is defined by simply describing threshold of sensor value and logical operation. Therefore, when end-users want to define more complex event, they need to use hard programming language such as C or Java. For defining such complex event easily without complex programming language, we propose a new event description language based on temporal relation between simple events.

1　はじめに

近年，複数のセンサが搭載され，小型化・高機能化された無線センサノードをモノに取り付け，モノ及びモノの周辺の環境情報を利用したサービス（以下スマートオブジェクトサービス）の研究開発が盛んである．例えば，センサノードをコップに取り付け，飲み物の温度を感知し冷める前にユーザに通知する MediaCup[1] や, センサノードを身の周りのものに取り付け，忘れ物があればユーザに通知する SPECs[2] などが挙げられる．我々の日常生活はモノとのインタラクションに溢れている．ユビキタスコンピューティングの目的である情報技術を利用したあらゆる場面での生活支援を実現するために，スマートオブジェクトサービスの重要性は高いと考えられる．

一方，家庭内でスマートオブジェクトサービスを実現するために，次の 2 点を考慮する必要がある．第 1 に，個々人は異なる数百のモノを所有しており，それらを対象とするサービスを構築するためには，各個人個人が自らモノにセンサノードを取り付け，環境情報を取得・処理可能なスマートオブジェクトとする必要がある．この問題を解決するため，我々は今までにモノとセンサノードの情報的な関連づけ手法を構築してきた [3]．第 2 は，個々人の生活スタイルが様々であるため，求められるサービスも多種多様となる点である．この問題を解決するためには，家庭のユーザの生活スタイルに応じたサービスを，そのユーザ自身が容易に構築できる環境が必要となる．

一般にユーザの要求は空間内の事象の変化により発生する．よって空間を構成するモノの状態の変化，ユーザとモノとのインタラクションをセンサを利用して取得することで，ユーザの要求を自動的に満たすサービスや，生活見守りサービス等が実現できると考えられる．上述した研究例も,「コップの温度が下がった」「モノとモノの距離が離れた」など，モノに関係する事象の変化（以下イベント）を利用して動作している．前川ら [4] は，モノを 1) 特徴的な繰り返し動作をもつモノ，2) モノのタイプに特徴的な動作がセンサの特徴的な出力の組み合わせからなるモノ，3) モノのタイプに特徴的な動作がない，またはセンサで特徴的な動作を検知できないモノ，の 3 つに分類し，3) に含まれるモノ（本研究が対象とする日用品の多くが含まれる）は，

[1] 本研究の一部は独立行政法人科学技術振興機構 CREST「マイクロユビキタスノード用ディペンダブル OS」プロジェクトにより行われています．

予めプログラマがモノの動作をイベントとして一意に定義できないことを述べている．すなわち，特徴的な動きがないモノを利用したサービスを実現するためには，ユーザ自身がモノの特定の状態をセンサノードが取得するセンサの物理量に置き換え，自ら利用したい形にモデリングする必要がある．

動きだけでなく，モノの温度やモノ周辺の照度に関するイベントを利用したサービスを構築する際にも，環境やそのモノを扱うユーザごとに同一のイベントでもセンサ値が異なる可能性が高いため，ユーザ自身がイベントの記述を行う必要性が高い．またユーザのイベントに対する感覚が異なるようなモノに関する事象も，ユーザ自身が定義せざるを得ない．例えば「飲み物が冷めた」という感覚はユーザごとによって異なっており，それをコップの温度が 30 度を下回ると記述するか，40 度を下回ると記述するかは，ユーザ自身が決定する必要がある．エンドユーザ自身がモノのイベントを定義する際には，取得したセンサ値に一定の閾値を設定することが一般的で最も簡単であると考えられるが，センサーノードによって検出されるイベントに基づくサービスを考えると，単一なイベントに基づくだけでなく，複数のイベントの組み合わせに基づいてサービスが提供されることの方がむしろ多いと考えられる．例えばあるモノを引き出しの中から取り出したというイベントを照度センサを利用して取得したい場合,「一定時間暗い」と「明るい」という 2 つのイベントを組み合わせる必要がある．この時，複数のイベントの時間的関係性が重要となる．本研究では複数のイベントの時間的関係性に着目し，ルールの記述に反映することを検討する．また，定義したイベントの対象となるモノを分離して記述することで，モノのイベントの共有・再利用性を高める特徴を有すイベント記述方式を構築することを目指す．

本稿は以下のように構成される．まず第 2 章でモノのイベントを記述するために必要な要素を整理し，時間的関係性を表現することの重要性を述べる．第 3 章で本研究が提案する時間的関係性を考慮したイベント記述方式を述べ，第 4 章でそのプロトタイプ実装について述べる．また第 5 章で関連研究を挙げ本研究の特徴を整理し，第 6 章で今後の課題について述べ，本稿をまとめる．

2　モノのイベント記述

時間軸を持つ実世界では，モノの状態は絶えず変化していく．我々がサービスを構築するためにイベントとして定義したいモノの特定の状態とは，この変化の一部分を切り取ったものである．プログラミングの知識の少ないユーザ自身がサービスを構築するためには，この変化の一部を容易にモデリングできる必要がある．センサデータの解析モデルとしては，人間の行動モデル・モノの状態遷移モデルを基にしたルールベース，パターンマッチング，隠れマルコフモデルなどの手法と，センサデータを構成事象とするニューラルネットワーク，ベイジアンネットワークなどの手法が挙げられる．これらの手法の中でも，ユーザが最も直感的に理解可能であるルールベースのサービス構築を支援するため，ルールベースにより入出力制御可能な小型デバイス [5] やサービスを容易に記述するためのビジュアルプログラミング環境 [6] が提案されている．本研究においても，ユーザ自身がルールベースで様々なサービスを記述できる環境の構築を目的とする．本章では，モノのイベントをルールベースで記述するために必要な要素を整理するため，ユーザがモノの状態をモノに取り付けられたセンサの物理値を利用していかに記述すべきか，以下，例を挙げて考察する．

1. **「温かい飲み物が冷めた」というイベント:**「温かい飲み物が冷めた」という状態をユーザ自身がスマートオブジェクトを利用して定義する例を考える．このイベントは,「冷めたら早くユーザに飲むように伝える」という，スマートカップサービスなどに利用可能である．今，ユーザが「冷めた」という状態を「コップの表面温度が 30 度を下回ったら」と記述したいとする．この場合，単純に $temperature < 30$ としただけではユーザの意図を正確には反映していない．これは,「30 度を下回る」というイベントが,「はじめは 30 度以上だったが，その後 30 度以下に下がる」ということを意図するからである.「30 度以下」という条件だけでは，コップに飲み物が入っているいないにも関わらず（例えば戸棚の中にしまわれていても）イベントが発火してしまうこととなり，問題である．よってこのイベントを記述するためには，$temperature >= 30$ と $temperature < 30$ という 2 つのイベント同士を,「前後」という時間的関係性を考慮して記述する必要がある．すなわち，if cup's $temperature >= 30$ and then $temperature < 30$. といったように記述する必要がある．
2. **「複数の椅子が同時に動いた」というイベント:**「複数の椅子が同時に動いた」もしくは「椅子のクッションに付けたが圧力センサが同時に反応した」といったイベントは,「複数ユーザが机の周りの椅子に座っている状態をミーティングと捉え，自動的にプロジェクターの電源をつける」などのスマートミーティングサービスなどに利用可能である．今，ユーザが「椅子が同時に動く」という状態を,「一定時間内に複数の椅子に取り付けられた加速度が一定以上の値を示した」と記述したいとする．この場合は，1. の例と異なり，複数の椅子が一定時間内に順不同に動いたという状態を記述する必要がある．すなわち，if more than 3 chairs' $movement > 0.15$ at least once in 10 sec. （3 つ以上の椅子が 10 秒以内に少なくとも一回以上 $0.15m/s^2$ の加速度を検知した）という形で記述する必要がある．

このように比較的単純に記述できそうなモノの状態でも，センサ値の閾値によるイベントを複数組み合わせる必要がある．なおかつ，実世界のイベントは時間的関係性を持って組み合わさっているので，ユーザがあるイベントの定義をどう行っていけばよいか，その指針を示す必要がある．C や Java などの汎用的プログ

ラミング言語を用いれば，上述したイベントを解析するプログラムを作成するのは可能であるが，一般ユーザには敷居が高いと考えられる．本研究ではこれらの問題の解決のアプローチとして，Allen によって定義された時区間関係表現 [7] に基づき，センサの閾値を利用しながらも，できるだけ実世界のイベントに近い形でユーザ自身がイベントを容易にモデリングするためのイベント記述方式を提案する．本方式は HTML に代表されるマークアップ形式を採用しているため，プログラミングに不慣れなユーザでも比較的容易に記述可能であると考えられる．

3 SOEUR: Smart Object Event Description Language using Temporal Relation

本章では我々が提案する，時間的関係性を利用したスマートオブジェクトサービスのためのイベント記述方式 SOEUR(Smart Object Event description language Using Temporal Relation) について述べる．SOEUR は XML 形式で記述され，センサの閾値により表現された単純なイベント同士を時間的関係性を考慮して組み合わせることで，より複雑なイベントとして定義可能とする．以下，SOEUR の詳細について述べる．

3.1 構成要素

SOEUR では，主に Event Template，Temporal Relation，Time Duration，Smart Object，Event の大きく 5 つの要素から構成される，単純なイベント記述方式である．以下，詳細について説明する．

3.1.1 Event Template 及び Temporal Relation

Event Template は，イベントの雛形となる要素である．Event Template は，Atomic Event Template と Composite Event Template の 2 種類が存在する．

- 単純イベントテンプレート（Atomic Event Template）

Atomic Event Template は，主にセンサの閾値を設定する．種類としては，*greater*，*greater-equal*，*equal*，*less*，*less-equal*，*between*，*except* の 7 つがある．それぞれについて，表に示す．以下は，μPart [8] というセンサノードが「温度が 30 度より低い」を示す，Atomic Event Template の例である．

```
<event_template type="atomic"
name="somethingCold">
 <node>
   <type>uPart</type>
   <sensor>temperature</sensor>
   <value exp="less">30</value>
  </node>
</event_template>
```

- 複合イベントテンプレート（Composite Event Template）

Composite Event Template は，Allen の時区間関係表現 [7] を用い，複数の Atomic Event Template で定義されたイベント雛形を組み合わせて構築する複合型のイベント雛形である．SOEUR では，Allen が定義した 13 の時区間関係表現に，any という種類の関係表現を加えた，計 14 個の関係性（図 1 参照）で示される複合イベントテンプレートを記述する．

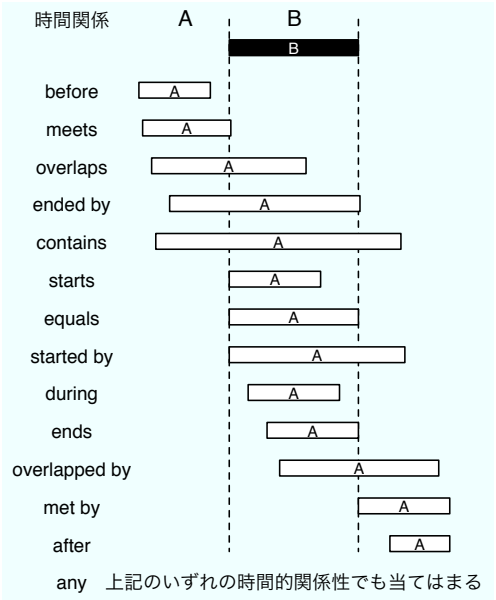

図 1: SOEUR で記述可能とする時間的関係性

今，somethingHot と名前がつけられた「30 度以上」を示す単純イベントテンプレート と，somethingCold という名前がつけられた「30 度より低い」を示す 単純イベントテンプレート の 2 つのイベントテンプレートを，「meets」の関係性で組み合わせ，somethingTurnsCold という名前の複合イベントテンプレートとしたいとする．この場合は，以下のように記述される．

```
<event_template type="composite"
name="somethingTurnsCold">
 <temporal_relation type="meets">
   <event_template ref="somethingHot"/>
   <event_template ref="somethingCold"/>
 </temporal_relation>
</event_template>
```

<temporal_relation> タグは，時間的関係性を type 属性としてとり，子要素として 2 つのイベントが含まれることとなる．なお特別な場合として，type 属性が any の場合は 2 つ以上のイベントが子要素として記述できるが，この詳細は後述する．

3.1.2 Time Duration

<time_duration> タグは,「１０秒間」などの時間間隔を表現するためのタグである．あるイベントを記述する際に，そのイベントに対してある時間間隔を用いたい場合がある．例えば,「１０秒間ずっと温度が 30 度より低い」といった時間間隔を持ったイベントが例として挙げられる．この場合，以下のように記述する．

```
<event_template type="composite"
name="somethingColdIn10Sec">
 <temporal_relation type="during">
   <time_duration>
      <sec>10</sec>
   </time_duration>
   <event_template ref="somethingCold"/>
 </temporal_relation>
</event_template>
```

SOEUR では,「10 秒間以上あるイベントが継続して発生する」といった具体的な時間間隔を持つイベントを，あるイベントと <time_duration> タグで表現される具体的な時間間隔とを <temporal_relation> によって両者の時間的関係性を定義することで表現する．上述した somethingColdIn10Sec の名前で示されるイベントテンプレートは，図 2 で示される関係性を持ち，somethingCold のイベントが 10 秒間継続して発生した時点で（実際には，10 秒間を超えた時点で）発火するイベントとなる．逆に 10 秒以内で終了したイベントを検知するためには，上述した例の <time_duration> と <event_template> の順番を逆にするか，Temporal Relation の種類として contains を記述すればよい．この場合は，somethingCold のイベントの開始（温度 <30）と終了（温度 >=30）の時間差が 10 秒以内であれば，イベントが発火することになる．

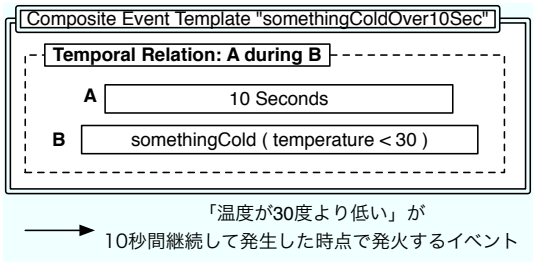

図 2: Time Duration と Temporal Relation を利用した具体的な時間間隔を持つイベントの例

3.1.3 Smart Object 及び Event

定義したイベントテンプレートに対しそのテンプレートの対象となるスマートオブジェクトを設定することで，実際のイベントとして取得することが可能となる．本研究の対象となるスマートオブジェクトは，ユーザによってセンサノードが取り付けられたモノである．スマートオブジェクトサービスを構築するためには，どのセンサノードがどのモノの情報を取得しているのか，両者の関連づけが必要となる．本研究では，センサノードとモノの関連付けに Spot & Snap [3] を利用する．Spot & Snap は，スポットライトが取り付けられたカメラデバイスを利用し，スポット光を照射しながらモノを撮影するインタラクションでセンサノードとモノを関連づける手法であり，関連づけられた情報は JPEG フォーマットの画像として保存される（モノ及びセンサノードの情報は，JPEG のユーザコメント領域に XML 形式で保存される（図 3 参照））．SOEUR ではこの関連づけられたスマートオブジェクトの情報をイベントテンプレートに対して設定することで，実体を持ったイベントとして定義される（図 4 参照）．

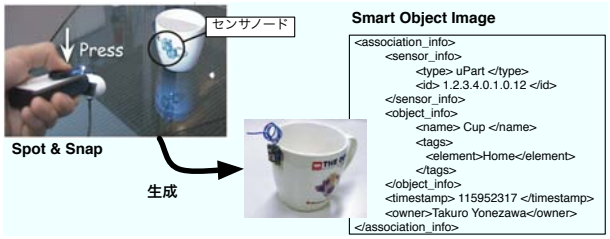

図 3: Spot & Snap によるモノとセンサノードの関連づけ

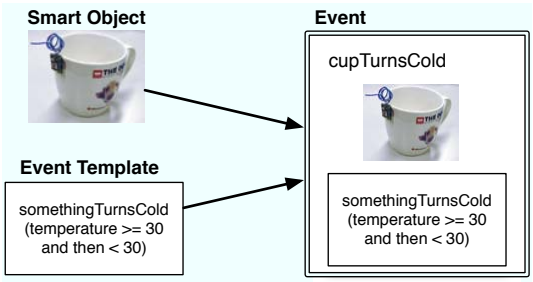

図 4: スマートオブジェクトと Event Template を関連づけ Event として定義

以下は，前述した somethingTurnsCold（30 度以上だったものが，30 度より低くなる）というイベントテンプレートの対象として，カップを設定している例である．Event も，Event Template と同様，Atomic Event と Composite Event の 2 種類に分けられる．Atomic Event は，構成要素として <smart_object> と <event_template> を対として持ち，Event Template で定義されたイベントの監視対象として，Smart Object を定義する．以下は，3.1.1 で定義した somethingTurnsCold というイベントテンプレートに対し，センサノードを取り付けたコップを対象として定義した，Atomic Event の例である．

```
<event type="atomic" name="cupTurnsCold">
 <smart_object>
   <sensor_info>
      <type>uPart</type>
      <id>1.2.3.4.0.1.0.12</id>
   </sensor_info>
   <object_info>
      <name>Cup</name>
   </object_info>
 </smart_object>
 <event_template ref="somethingTurnsCold">
</event>
```

Composite Event は Composite Event Template 同様，異なる 2 つ以上の Atomic Event からなるイベントを時間的関係性を考慮して定義する．Composite Event は複数の異なるスマートオブジェクトを利用した，複

雑なイベントを定義する際に使用する．図 5 に,「ミーティングが行われていること」を判別するために作成した,「4 つの椅子のうち，3 つ以上がいずれも 10 秒以内に動いた」というイベント（IsMeeting と名付けられている）を定義した Composite Event の例を記す．また，この Composite Event を構成する要素の関係性を可視化した図を，図 6 に示す．

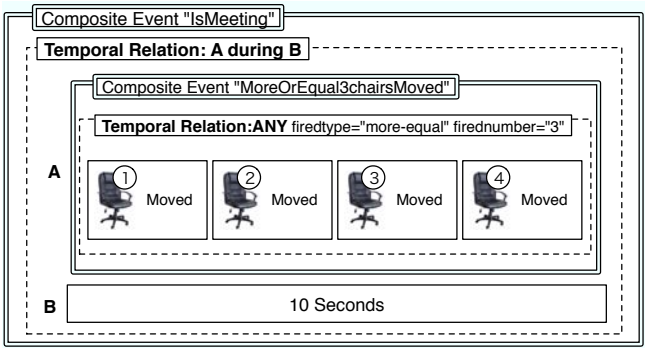

図 6: 可視化された図 5 の Composite Event

この Composite Event では，複数の Event が入れ子構造となって構成されている．まず MoreOrEqual3ChairsMoved という「3 つ以上椅子が動いた」を示す Event と,「10 秒間」を示す Time Duration が During という時間的関係性によって関係付けられている．これは，10 秒間以内に 3 つ以上椅子が動くと発火するイベントを示している．さらに MoreOrEqual3ChairsMoved という Event 自体も Composite Event となっており，chair1Moved（椅子 1 が動いた）から chair4Moved までの 4 つの Atomic Event が，**Any** という種類の時間的関係性によって関連付けられている．

Any という時間的関係性は，<temporal_relation> の構成要素となる Event が全て発火した際に，その発火のタイミング同士がいかなる時間的関係性でも全体として発火したとする，特殊な属性である．また，Any を属性として指定した Temporal Relation の追加属性として，firedtype と firedNumber が存在する．firedtype では *more, more-equal, equal, less-equal, less* が存在し，firedNumber で指定された数のイベントと組み合わさって,「子要素となるイベントのうち，firedNumber 数以上が発火したら」などの指定に利用することができる．Any という柔軟な時間的関係性を利用することで，SOEUR 自体に柔軟性を持ったイベント定義を可能とし，そのことによって複数のモノを扱うイベントが容易に定義できるようになっている．また，chair1Moved から chair4Moved の Atomic Event は，4 つの椅子に対し，それぞれ同じ Event Template を設定している．このように，テンプレートを用いることでモノとイベントを判断するロジックが分離できることで，あるイベントの再利用性・共有性を高めることができる．

3.2 特徴

本節では，SOEUR の特徴として，記述力，共有・再利用性，制限の 3 点について議論する．

3.2.1 記述力

SOEUR では，従来の単純な閾値設定によるイベント定義を踏襲した上で，イベント同士の時間的な関係性を考慮して組み合わせる手法を用い，より複雑なイベントを記述可能としている．更に，Any という任意の時間的関係性で発火する種類のイベント複合方式を用意することで，複数のモノを対象とした柔軟なイベント記述も行える．また一方で，できるだけ記述方式を簡素化するために，SOEUR には明示的に論理和（OR）・論理積（AND）を記述するタグは用意されていないが，Allen の 13 の時間的関係性及び本稿が提案した Any という関係性を利用することで，これら論理演算子を利用した際と同様のイベント記述が可能であるという点が特徴として挙げられる．例えば，<temporal_relation type=”any” firedtype=”equal” firednumber=”1”> という Temporal Relation タグで囲まれた複数のイベントを有する Composite Event は，それらイベントのうちいずれか一つでも発火すれば全体として発火が進むため，複数のイベントの論理和をとることと等しい．

3.2.2 共有・再利用性

SOEUR の最も基本的な構成要素である Atomic Event Template は単純なセンサの閾値を意味しているため，ユーザが記述したイベントはブラックボックス化されることがなく，他ユーザが理解可能となっている．よって，他ユーザが記述した SOEUR を自分の環境に合うようにユーザ自身が容易に改変することが可能であり，再利用性が高い．我々は，基本的にある環境でユーザ自身がモデリングしたイベントを，他の環境でそのまま使用できるとは想定していない．なぜなら環境やユーザが異なれば，そのモノが検知するセンサ値は異なるためである．よって他者が作成したイベントをテンプレートとして用い，自分の環境にあったイベントとして容易に改変できる単純さを持っているということは，重要な特徴である．また，一度定義した Event Template は対象となるスマートオブジェクトを入れ替えるだけで同様のイベントとして利用可能であるため，この点でも SOEUR は共有・再利用性を有していると考えられる．

3.2.3 制限

SOEUR は Event Template，Event，Temporal Relation，Time Duration，Smart Object の主に 5 つから構成される単純な構造となっているため，プログラミングの経験のないユーザでも比較的容易に利用できると考えられる．その一方で汎用プログラミング言語が備えている変数の利用や繰り返しの記述を有していないため，それらを利用したイベントの記述は不可能である．よって，あるイベントが 5 回発火したら全体として発火するようなイベントを記述したい際には，before や during などの時間的関係性を用い，全てのイベントを時系列にそって記述する必要がある．しかし，Event

```
<event type="composite" name="IsMeeting">
        <temporal_relation type="during">
                <event type="composite" name="MoreOrEqual3ChairsMoved">
                        <temporal_relation type="any" firedtype="more-equal" firednumber="3">
                                <event type="atomic" name="chair1Moved">
                                        <smart_object>
                                                <sensor_info>
                                                        <type>uPart</type>
                                                        <id>1.2.3.4.0.1.0.15</id>
                                                </sensor_info>
                                                <object_info>
                                                        <name>Chair1</name>
                                                </object_info>
                                        </smart_object>
                                        <event_template ref="somethingMoved"/>
                                </event>
                                <event type="atomic" name="chair2Moved">
                                        <smart_object>
                                                <sensor_info>
                                                        <type>uPart</type>
                                                        <id>1.2.3.4.0.1.0.204</id>
                                                </sensor_info>
                                                <object_info>
                                                        <name>Chair2</name>
                                                </object_info>
                                        </smart_object>
                                        <event_template ref="somethingMoved"/>
                                </event>
                                <event type="atomic" name="chair3Moved">
                                        <smart_object>
                                                <sensor_info>
                                                        <type>uPart</type>
                                                        <id>1.2.3.4.0.1.0.122</id>
                                                </sensor_info>
                                                <object_info>
                                                        <name>Chair3</name>
                                                </object_info>
                                        </smart_object>
                                        <event_template ref="somethingMoved"/>
                                </event>
                                <event type="atomic" name="chair4Moved">
                                        <smart_object>
                                                <sensor_info>
                                                        <type>uPart</type>
                                                        <id>1.2.3.4.0.3.0.96</id>
                                                </sensor_info>
                                                <object_info>
                                                        <name>Chair4</name>
                                                </object_info>
                                        </smart_object>
                                        <event_template ref="somethingMoved"/>
                                </event>
                        </temporal_relation>
                </event>
                <time_duration>
                        <sec> 10 </sec>
                </time_duration>
        </temporal_relation>
</event>
```

図 5:「4つの椅子のうち，3つ以上が 10 秒以内に動いた」というイベントを記述した Composite Event "IsMeeting"

Template を利用することで重複した記述はできるだけ避けられるようになっているため，多くのテンプレートを予め用意することでユーザの負荷はある程度減らすことが可能である．また，現状の無線センサノードは一定間隔のタイミングで無線ネットワークを通じてセンサデータをコンピュータに送信する．よって，センサノードのデータ送信間隔によっては，特定の時間的関係性は判別不可能である．例えば，異なる 2 つのモノのイベント同士が meets の関係であるかどうかや，あるモノのイベントが 10 秒という時間間隔と厳密に equal の関係であるかどうかは，判別不可能であると考えられる．これらの点については，次章及び今後の課題として後述する．

4 実装

本章では，我々がプロトタイプとして実装した SOEUR システムについて述べる．本システムは Java によって作成した．なお，システムが対象とするセンサノードとしては， uPart[8] を利用した．以下，XML から Java オブジェクトへのデータ変換モジュールとイベント判断モジュールの 2 点について述べる．

4.1 データバインディング

SOEUR のスキーマとして，XML Schema を利用して記述した．例として，<temporal_relation> を定義する XML Schema を以下に示す．図中 e_etBasicGroup や e_etAnyGroup といったグループ要素は，内部に出現回数に制限を持たせた <event> や <time_duration> 要素を定義しており，文書妥当性を向上させるよう工夫した．

```
<xsd:element name="temporal_relation">
 <xsd:complexType>
  <xsd:choice>
   <xsd:group ref="e_etBasicGroup"/>
   <xsd:group ref="e_tdBasicGroup"/>
   <xsd:group ref="et_tdBasicGroup"/>
   <xsd:group ref="e_etAnyGroup"/>
  </xsd:choice>
  <xsd:attribute name="type"
  type="temporalType" use="required" />
  <xsd:attributeGroup ref="fired"/>
 </xsd:complexType>
</xsd:element>
```

XML の解析ツールとしては，JAXB（The Java Architecture for XML Binding）[9] の参照実装である JWSDP（Java Web Services Developer Pack）を利用した．JAXB は XML で記述されたデータモデルを Java のオブジェクトモデルに変換することができ，生成された API を利用することでオブジェクトを操作した処理が可能である．また，反対にオブジェクトモデルを XML に変換することも可能であるため，今後のシステム拡張にも利用可能である．またユーザが作成した SOEUR をユーザが確認・理解しやすくするため，SOEUR の可視化ツールを作成した（図 7）．本ツールは SOEUR 内に記述された各要素一覧をユーザに提示すると同時に，ズーミングユーザインタフェースにより多重構造を探索的に閲覧・理解できるようになっている．

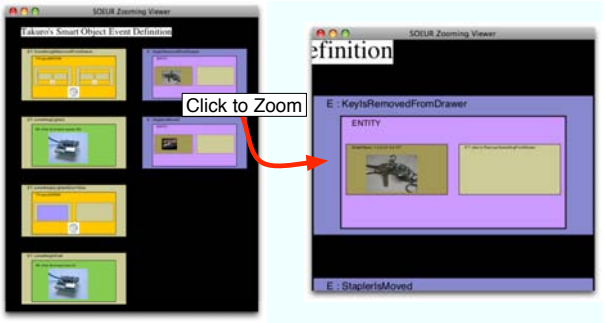

図 7: SOEUR 構造可視化ツール

4.2 イベント判断モジュール

本節では，複数のイベントの時間的関係性から構成される複合イベントの検知を行うイベント判断モジュールのプロトタイプ実装について述べる．イベント判断モジュールは個々のイベントが開始した時刻と終了した時刻を保持し，イベント同士の時間関係性が満たされたかどうかについて，それらの時刻を逐一比較することで評価する．すなわち，センサの閾値設定からなる単純イベント A の開始時刻を $t1$，終了時刻を $t2$，単純イベント B の開始時刻を $T1$，終了時刻を $T2$ とした時，A before B の関係性を評価するためには，$t1 < t2 < T1 < T2$ かどうかを評価する．同様に，複合イベント同士の比較する際も，それぞれの複合イベントの開始時刻と終了時刻を比較して評価を行う．

今回我々が利用したセンサノードはそれ自体が時刻情報を持っていないため，コンピュータ側でセンサデータを受信した時刻を利用した．センサデータは一定間隔で送信されるため，実際にイベントが開始・終了した時刻とセンサデータを受信した時刻の間にタイムラグが生じる．実装ではこのタイムラグを考慮し，センサデータの送信間隔毎にイベント発生・終了時刻ウィンドウを設定した．今回使用したセンサノードのデータ送信間隔は 500 ミリ秒と設定してあるため，500 ミリ秒以内に検知した全てのセンサデータの受信時刻を同一時刻として評価している．すなわち，本実装では 500 ミリ秒間隔の精度において，イベント間の時間的関係性を評価している．一般的に人間の視覚の時間分解能は 30 ミリ秒であると言われ，30 ミリ秒以内に発生した事象は同時であると眼は判断している．生活の中でイベントとして検知すべき「同時」が 500 ミリ秒間隔で良いかどうかは今後の評価を行う必要があるが，検知したいイベント毎に「何ミリ秒以内を同時とするか」を設定することで，より実用的な時間的関係性として利用できる可能性がある．

5 関連研究

Dey らは 20 人のプログラミング未経験者を対象に，あらゆる状況が検知可能なスマートルームで利用した

いアプリケーションについて調査を行い，その結果提案された 371 のアプリケーションのうち 80%が if-then ルールで記述可能であったと述べている [6]．これはユビキタスコンピューティング環境でのエンドユーザプログラミング形式として，if-then ルールの有効性を示している．またこの結果を受け，Dey らは if-then ルールを容易に記述するためのビジュアルプログラミング環境を構築しているが，記述可能なルールは AND，OR の単純な boolean ロジックを有したイベントに限られている．同様に単純な ECA（Event-Condition-Action）ルールを用いてサービスを記述する試みは多く，例えば寺田らは ECA ルールを用い，入出力制御可能な小型デバイスを用いたユビキタスコンピューティング環境の実現を提案している [5]．一方，Shankar らは従来の ECA ルールの欠点として，ECA ルール発火前後の条件指定ができないことを挙げ，事前事後ルールを追加した ECPAP ルールを導入し，更に複数ルールに対するペトリネットを構築することでルール適用可能数を増加させている [10]．しかし ECPAP ルールを記述する主体は部屋の管理者が想定されており，一般ユーザにとっては敷居が高い．

また Jung らは，ユーザの可読性を重視し，5W1H を用いたイベント記述方式を提案している [11]．しかし 5W1H の形式でユーザが記述するためにはセンサデータをコンテキストとして解釈可能な環境まで構築されている必要があり，本研究の想定する家庭環境には不適当である．一方，本研究と同様に時間的関係性を用いてイベントを検知する研究例として，前川らの Tag and Think[4] が挙げられる．Tag and Think では，モノに設置されたセンサノードが取得するデータからそのモノが何であるか推定するために，モノの状態とモノに起こりえる状態間の遷移の関係を時間的関係性を考慮して記述し，その状態遷移図と実測したデータから得られた状態とを照らし合わせ評価している．Tag and Think ではセンサのある時点での定常状態の変化量を用いてモノの状態を定義しているため，多様な環境に対応し高い精度でモノの推定を可能としている．一方本研究では同一のモノだけでなく，複数のモノを含んだイベントをユーザ自身がセンサノードの閾値を利用して定義できる環境に着目しており，また Any の関係性を記述可能とすることで柔軟なイベント定義が行える点が特徴である．

6　今後の課題とまとめ

ユーザの生活と密着するユビキタスコンピューティング環境を実現するためには，個々のユーザが自らの生活環境に対応したサービスを自分自身で容易に構築できる必要がある．本稿では，プログラミング未経験者でも容易に記述可能なイベント駆動型スマートオブジェクトサービスの多様性を高めるため，時間的関係性を考慮したイベント記述方式 SOEUR を提案した．SOEUR は XML 形式で記述され，センサデータの閾値設定による単純さ，イベントのテンプレート化による資産の共有・再利用性の高さ，Any という関係性の追加によるイベントの柔軟な記述力の 3 つの特徴を有する．本稿ではその具体的な記述方式及び現状のプロトタイプ実装について述べた．

今後の課題として，SOEUR の改良・評価以外の点で大きく以下の 3 つが挙げられる．第 1 に，イベントの検知対象としてより多くのセンサや情報家電の状態等と統合する点が挙げられる．例えば位置情報センサを用いることで,「ユーザ A がある部屋にいるというイベントの間（During），鍵が動いたら」といった，より多様なイベントが記述可能となる．第 2 に，イベント記述を支援するツールの考案が挙げられる．現状では記述した SOEUR の可視化ツールしか提供していないため，ユーザはソースコードをテキストエディタなどで直に記述する必要がある．ユーザが直感的にイベントを記述できるよう，ビジュアルプログラミング環境などのツールが必要である．第 3 に，アクションの記述方式の検討が挙げられる．イベントを検知した結果起動したいアクションをどう記述すればよいか，必要な要素を整理・実現していく必要がある．

参考文献

[1] Michael Beigl, Hans-W. Gellersen, and Albrecht Schmidt. Mediacups: experience with design and use of computer-augmented everyday artifacts. *Computer Networks (Amsterdam, Netherlands: 1999)*, 35(4):401–409, 2001.

[2] Mik Lamming and Denis Bohm. Specs: Another approach to human context and activity sensing research, using tiny peer-to-peer wireless computers. In *International Conference on Ubiquitous Computing(UbiComp)*, 2003.

[3] 米澤拓郎, 榊原寛, 中澤仁, 永田智大, 高汐一紀, and 徳田英幸. 日常物とセンサノードの関連付け手法の提案. In **第 *13* 回ユビキタスコンピューティングシステム研究会**, pages 179–186. 情報処理学会, 2006.

[4] 前川卓也, 柳沢豊, and 岡留剛. Tag and think: センサネットワークを前提としたモノ自身とその状態の推定. In **第 *13* 回ユビキタスコンピューティングシステム研究会**, pages 211–218. 情報処理学会, 2006.

[5] Tsutomu Terada, Masahiko Tsukamoto, Keisuke Hayakawa, Tomoki Yoshihisa, Yasue Kishino, Atsushi Kashitani, and Shojiro Nishio. Ubiquitous chip: A rule-based i/o control device for ubiquitous computing. In Alois Ferscha and Friedemann Mattern, editors, *Pervasive*, volume 3001 of *Lecture Notes in Computer Science*, pages 238–253. Springer, 2004.

[6] Anind K. Dey, Timothy Sohn, Sara Streng, and Justin Kodama. icap: Interactive prototyping of context-aware applications. In Kenneth P. Fishkin, Bernt Schiele, Paddy Nixon, and Aaron J. Quigley, editors, *Pervasive*, volume 3968 of *Lecture Notes in Computer Science*, pages 254–271. Springer, 2006.

[7] James F. Allen. Maintaining knowledge about temporal intervals. *Commun. ACM*, 26(11):832–843, 1983.

[8] Michael Beigl, Christian Decker, Albert Krohn, Till Riedel, and Tobias Zimmer. µ parts: Low cost sensor networks at scale. In *Ubicomp Demo Session*, 2005.

[9] jaxb: JAXB Reference Implementation. `https://jaxb.dev.java.net/`.

[10] Chetan Shiva Shankar, Anand Ranganathan, and Roy Campbell. An eca-p policy-based framework for managing ubiquitous computing environments. In *MOBIQUITOUS '05: Proceedings of the The Second Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services*, pages 33–44, Washington, DC, USA, 2005. IEEE Computer Society.

[11] Jae-Yoon Jung, Yoo-Suk Hong, Tae-Wan Kim, and Jinwoo Park. Human-centered event description for ubiquitous service computing. In *MUE*, pages 1153–1157. IEEE Computer Society, 2007.