2007－DPS－133 (3)
2007／11／26

シチュエーションアウェアサービスを実現するための再利用可能な状況記述の手法

池田拓也[†] 制野孝幸[†] 程子学[††]

ユビキタステクノロジーにおいてコンテキストアウェアサービスに注目が集まっているが，本稿ではこの技術を拡張したシチュエーションアウェアサービスに着目する．また，サービスを提供するフレームワークを構築し，サービスの記述方法を提案する．この記述には，シンプルかつフォーマルな記述を可能にするために，SDML(Situation Description Markup Language)と呼ぶ XML ベースの状況理論を基にした記述方式を提案する．このSDML には，冗長な記述を低減するためにオブジェクト指向プログラミングによく利用される継承に似たコンセプトを導入する．ユーザのシチュエーションをいかに記述するかという課題がサービスの有用性を大きく左右することから，本稿ではシチュエーションの記述方法に主眼を置き，シチュエーションアウェアサービスの構築を行う．

Reusable Situation Description Method to Realize Situation-Aware Service

TAKUYA IKEDA[†] TAKAYUKI SEINO[†]
ZIXUE CHENG[††]

Context-aware technology has attracted researcher's attention in ubiquitous computing field. However, we propose situation-aware method which is an extension of context-aware system. We aim to develop a framework to describe situation-aware service definition. We propose SDML (Situation Description Markup Language) to define the description simply and formally. To realize the purpose, we construct this SDML according to Situation-Theory and describe it based on XML. And also, we introduce a method from object oriented programming such as extension to decrease cumbersome procedure of describing situations. In this paper, we focus on the method of situation description, because the solution of describing user's situations is a key to make the system worthful.

1. はじめに

コンピュータは小型化の一途をたどり，今では手のひらの大きさ程度のコンピュータも開発されている．小型化されたコンピュータの利用が見込まれるユビキタスコンピューティングの社会では，「コンテキストアウェア」と呼ばれる，個々のユーザに適したサービスを自動的に提供する仕組みが注目を集めている．

たとえば老人に対する駅構内のガイドを考える．既存のコンテキストアウェアサービスでは，ユーザの位置情報や，過去の行動履歴などから，駅構内の通路が危険であることをユーザにサービスとして提供することが考えられる．

このように，既存の一般的なコンテキストアウェアサービスは，すべてのユーザに対して同様にサービスを提供するか，ユーザの行動履歴等からサービスが必要か否かを判断する程度のものであった．

もし，これらの情報に加え，ユーザを取り巻く人や物の属性情報や，さらにそれらとユーザとの間にある関係性などを考慮することが可能であれば，さらに個々のユーザに対してパーソナライズすることが可能になると考えられる．

先の例で言えば，ユーザの近くに別のユーザが検出され，そのユーザが介護を必要としない人物であり，さらに介護者・被介護者といった関係や家族関係などが検出された場合には，サービスの提供を中止することが可能になるであろう．また，通路を歩く速度や，危険箇所の近くに来たときの行動パターンなどをセンシングし，現在の状況からサービスが重要ではないことを検出できれば，サービスをより軽いものに変更したり，あるいは中止したりすることも可能になる．

このように，ユーザの情報をより詳細にサービスに反映させるためには，ユーザをとりまくシチュエーションを考慮できる仕組みが重要になるであろう．なお，ここでいうシチュエーションとは，コンテキストアウェアサービスにおいて利用される位置情報などに加えて，ユーザ間や物との間にある属性情報や関係情報などを含めたひとまとまりの状況が，複数の時間帯にまたがって存在するような複合的な状況を指すものとする．そこで本稿では，ユーザのシチュエーションを考慮できる一歩進んだコンテキストアウェアサービスと位置づけられる「シチュエーションアウェアサービス」の構築を目標とする．

[1]では，家電制御のためのルールベース言語提案を行っている．しかし，ここで提案されているコンテキストの記述は，温度などのセンサデータとそれに基づく家電機器の操作という抽象度の低いシンプルなものである．そこで，時間と共に変化するシチュエーションの把握や，それに対するアクションの定義を可能にすることで，さらに有用なシステムにできる．また，利用者間や物事との関係性についての考慮がなされることで，さらに抽象度の高いルールが設定来る．

† 会津大学大学院 コンピュータ理工学研究科
Graduate School of Computer Science and Engineering, The University of Aizu
†† 会津大学 コンピュータ理工学部
School of Computer Science and Engineering, The University of Aizu

[2][3]では，サービス合成という手法を用いている．これは，ユーザの利用時に動的に必要なサービスを事前に定義された「意味的メタデータ」を基に自動収集し，個々のサービスを連結したひとつのサービスとして，利用者に提供する手法を提案している．サービス提供中に変化したコンテキストを再評価することで，動的に変化するコンテキストにも対応しているが，ユーザ個々のデータはいち時点での静的なデータである．そこで，時間と共にユーザのシチュエーションが動的に変化した際，どのように振舞うかに関して考慮することで，さらにユーザに対して高い抽象度のサービス定義でパーソナライズされたサービスを提供する仕組みが構築できると考えられる．

[4]では，ユーザの位置情報と RF-ID によって得られるユーザを取り巻く物の関係性に着目して，ユーザのシチュエーションを認識する手法を提案している．しかし，同じ場所や物が検出されたとしても，シチュエーションはひとつに限定できるとは限らない．ユーザのシチュエーションを把握するためには，現在に至る経緯や，物・人・場所などの間に存在する関係性についても考慮する必要があると考えられる．

日常世界において，人物のシチュエーションを表現する手段として，人間に一番なじみのある手法は自然言語による表現であろう．しかし，自然言語による表現のままでシチュエーションを把握するには，あいまい性が排除しきれない．そんな中，物事を体系的に表現できるような記述手法として，様々な理論が提唱されてきた．そのうちのひとつが状況理論[6]である．この状況理論では，対象者や対象者の情報と対象者を取り巻く人や物との関係性と時空間の情報を使うことで，対象者のシチュエーションを体系的に表現できるように考えられている．

[5]では，情報の表現における構成単位は，その一部もしくは全てが関係性から特定されるコンテキストによって決定付けられ，状況理論をベースにしたコンテキストを表現する方法を述べている．このアプローチは，コンテキストの表現が状況理論と親和性が高いことを示している．

状況理論では，シチュエーションを整理するための様々な枠組みが提唱されているが，本稿ではそれらの中から関係性や時空間などの要素と，それらを組み合わせて表現するシチュエーションに関する記述方法を抜き出して，シチュエーションの表現をコンピュータで処理可能なデータとして表現する手法を考察する．

本稿では，ユーザ固有の情報や位置情報，興味情報だけではなく，ユーザと物の関係性にも着目し，従来のコンテキストアウェアよりも柔軟なサービスの定義が出来るような仕組みの構築を目指す．そのために，シチュエーションとサービスの記述を可能にし，それらを適切に組み合わせて特定の条件が成立すると組み合わされたサービスが発火できるようなルール定義の枠組みを作成する必要がある．そこで本稿では，この定義を記述する枠組みとして Situation Description Markup Language（状況記述言語：以下 SDML と呼ぶ）を提案する．この SDML に，シチュエーションとサービス，さらにそれらを組み合わせるルールを記述して，シチュエーションアウェアを実現するための基盤として利用する．SDML についての詳細は，第 3 節で述べることにし，次節ではシチュエーションアウェアサービスを実現するために必要な表現能力と，SDML におけるそれらへの解決方法を考察する．

2. SDML の記述方法

2.1 SDML の記述に求められる表現能力

(a) 表現性

ユーザが望む多様なシチュエーションの記述に対応でき，複雑なシチュエーションの記述であっても，シンプルかつ統一された記述方式で，様々なシチュエーションの厳密な記述を実現できる能力が必要である．

(b) 一般性・流通性

新たに，ゼロから言語を作成するのではなく，現在広く利用されている汎用的な技術をベースに記述方式を構築し，シチュエーションの記述を行うために必要となる新たな知識を最小限に抑えることが望まれる．

また，ユーザが望むシチュエーションの記述には，細かい部分のみカスタマイズが必要であっても，大筋では似通ったシチュエーションの記述であることも起こり得る．そこで，既存の SDML を交換・流用できる仕組みがあると，利便性が大きく向上する．

(c) 冗長性の排除

SDML を流用する際に，類似したシチュエーションの記述をする必要に迫られることがある．こういったとき，毎回全ての記述をしていると，冗長な作業が多くなってしまい，記述に必要なコストが高くなってしまう．そこで，同じような記述が必要な場合，殆どの部分をそのまま流用し，一部分に変更を加えられるような仕組みがあると，記述に必要な労力が大きく軽減できる．また，複雑なシチュエーションの記述を行う際にも，いくつかの大きなパーツを組み合わせ，細かい部分のみを環境に合わせて変更できる雛形のような仕組みを作ることで，複雑な記述を避けシンプルに SDML の記述を行うことが可能になる．

(d) サービス発火条件の記述

ユーザに適切なユビキタスサービスを提供するためには，状況理論に沿った形でのシチュエーションの記述が適切に出来ること以外にも，どのようなサービスをユーザに提供するかを適切に記述できないと意味が無い．また，特定のシチュエーションが成立したときに，適切なサービスの発火(サービスをユーザに提供するために起動すること)が必要となる．シチュエーションの記述とサービスの記述は多対多である必要があり，複数のシチュエーション記述と複数のサービス記述をひとつのルールとして記述出来る能力も，SDML に求められる能力である．

(e) ルールへの変換ができること

自動的にサービスを提供する仕組みを作るために，シチュエーションの記述フレームワークに実装したルールベース推論システムが理解できるルール形式へ完全に変換できることが必要である．

2.2 SDML 設計方針

(a) 状況理論との対応関係

構成するタグの記述方式が，状況理論の 4 つのプリミティブ（個体，属性，関係，時空位置）と 3 つの要素（ToS，SoA，CoE）に対応していることで，状況理論に沿った表現が出来る．また，状況理論では，これら 4 つのプリミティブと 3 つの要素を利用し再帰的な定義を行うことで，複雑なシチュエーションの記述に対しても同じ記法のみで対応できるとしており，この考え方を踏襲することで，利用する記述方法がシンプルなものだけであっても複雑なシチュエーション表現をカバーできる．

また，自然言語では，表現者によって単語の意味に対する認識の違いなどから，ひとつの表現からいくつかの理解が生まれることがよくある．しかし，利用できる表現を限定することで，理解者によって認識が変化する相対的な表現を排除し絶対的な表現でシチュエーションの記述を構成する．結果として，あいまい性を排除した厳密な表現が出来るようになる．

(b) XMLに沿った構文

現在汎用的に利用されている XML の記述規則に沿った記述を行うことで，SDML の流通を実現出来る．つまり，既存のシチュエーション記述を，自分の環境にカスタマイズして利用することが出来るので，有用なシチュエーションの記述が作成されれば，広く一般に活用されることが期待できる．

(c) 再利用性の考慮

状況理論を基に構築された記述規則に則って記述された各要素に一意の名前をつけ，それを基に既存のシチュエーションに関する記述を引用可能にする．既存の定義をテンプレートのように利用できるので，SDML の記述に必要な労力を大きく軽減できる．

ただし，無秩序に引用を許すと，表現規則との矛盾や，ルールへの変換の際に不都合が生じる．そこで，引用可能な要素を限定し，同じ要素同士でないと引用できないように規則を定める．

また，引用の際に元の記述にある要素を，引用先においてどの程度まで効果を及ぼすことにするかというスコープに関する問題もある．これに対しては，後述するシチュエーション記述に必要な要素をツリー構造にして考えたときに，引用先の階層よりも下の記述全てに適用することにすると，自然な記述が出来る．つまり，状況理論における SoA に対して引用の記述を行った場合，SoA よりも下の階層にある ToS やその下の階層にある記述全てに適用することになる．また，必要であれば，スコープ内で有効になっている記述に対して，同じ要素で違う内容の記述を追加することで，その内容を上書きすることも可能とする．

この引用機能を利用して，第三者が作成した SDML 文書の一部を引用することも可能にする．つまり，記述されている内容がどの場所にあるのかを指定することで，他の場所にある記述(SDML ファイル)も参照可能にする．

(d) ルールの記述方法

ユーザに適切なサービスを提供するためには，シチュエーションの記述と共に，ルールの記述が必要となる．ここでいうルールとは，どのシチュエーションが成立したときに，どのサービスを発火させるかを定義したもので，1 つ以上のシチュエーションと，1 つ以上のサービスを対にして 1 つのルールを定義する．

(e) パーサの実装

SDML から推論システムのルール形式に変換できるパーサを実装する．ここでいう推論システムとは，ユビキタスコンピューティング環境に構築されたセンサネットワーク（いくつものセンサが組み合わせて設置されており，ユーザを取り巻く様々なシチュエーションをコンピュータで把握できることを目的として構築された環境）から得られたユーザに関する情報と，事前に設定された条件とを照らし合わせて，ユーザの置かれているシチュエーションを推論し，その推論結果を，ユビキタスサービスの提供を行うアプリケーションに伝えるためのシステムである．実世界に存在する様々な情報をセンサネットワークでセンシングし，ユーザのシチュエーションを推論システムが推論して，ユビキタスサービスを提供するアプリケーションに対して発火の必要があるサービスを伝えることによって，シチュエーションアウェアサービスを構築可能にする．ユーザは，特殊な構文や独自の制約が多い推論システム向けに設計されたルール形式の表現を意識せずにシチュエーションやアクションを記述することが可能になる．

2.3 状況理論の概要

状況理論は，Barwise らによって提唱された理論であり，シチュエーションは 4 つのプリミティブと，それらの組み合わせによって表現される．4 つのプリミティブとは，「個体(Individual)」「属性(Property)」「関係(Relation)」「時空位置(Time-Space Position)」である．そして，それらの個体の属性や，個体同士の関係を利用して表現する「状況型（Type of Situation）」がある．状況型(ToS)とは，時間や場所の表現を排除して，いくつ物状況間において不変の情報は，個体や関係だけではなく，それらからなる一連の集合も不変であることがあるという考え方に基づく表現である．そして，この状況型と時空位置を組み合わせて，不変の情報を現実世界の位置状況として特定するのが「事態(State of Affair)」である．そして，時空間軸に沿って事態(SoA)を集めた表現が，最終的なシチュエーションを表現する「出来事系列(Course of Event)」である．状況理論では，この出来事系列(CoE)を用いることで，体系的に日常における事象を表現できるとしている．

3. SDML 記述

第 3 節では，第 1 節と第 2 節で考察した状況理論に基づくシチュエーションの記述を行う際に必要となる SDML を提案する．3.1 では，SDML の概要について触れ，今回新たに構築する SDML を俯瞰する．3.2 と 3.3 では，SDML の書き方を詳しく説明していく．そして，最後に 3.4 で，ここまで述べてきた SDML の記述方法に関して例を交えて説明する．

3.1 SDML 記述の概要

前の節までに述べてきたように，SDML は状況理論に基づいたシチュエーションの記述を XML の記述規則に則って行う．そのためには，状況理論における各要素を XML で表現できるような構造化された構成を考える必要があるが，状況理論自体が既に十分構造化されているので，ここではそれらを整理するためにツリー構造で SDML の設計を考える．以降，このツリーを SDML ツリーと呼ぶ．

それぞれの要素に関しては，3.2 以降で詳細に説明することにし，ここでは図 1 を用いて，SDML ツリーの概要を説明する．

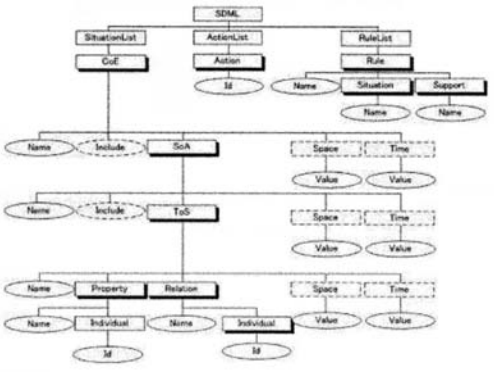

図 1　SDML ツリー

Figure 1　The SDML Tree

図 1 において，四角形は要素名を，円は属性名をそれぞれ表している．また，影付の要素は同一階層に複数記述することが出来る．細い実線の要素と属性は記述必須項目であり，省略を許可しない．細い点線の要素と属性は必要に応じて記述することが可能とする．

シチュエーションアウェアサービスのためのルールを記述する際には，大きく「シチュエーションの記述(SituationList)」「アクションの記述(ActionList)」「ルールの記述（シチュエーションとアクションの対応付け）」の 3 つの構造が必要になる．SituationList には，状況理論に基づいたシチュエーションを，ActionList には特定のシチュエーションが成立した場合に発火するアクションを，RuleList にはどのシチュエーションが成立したときにどのアクションが発火させるかを指定するルールを，それぞれ記述する．SDML ツリーの上位 2 つの階層で，この 3 つの構造を分別する．そしてこの 3 つの構造の下に位置するいくつかの階層で，それぞれの詳細な記述を行っていく．

3.2　各要素の XML 表記法

シチュエーションの記述には，3.1 で述べたように，大きく分けて 3 つの記述構造が存在する．以下にその 3 つの記述構造を説明し，その記述方法を明確にする．

シチュエーションの XML 表現

1 つ目の記述構造は，状況理論に基づく形でシチュエーションを表現するための XML 記述方法である．

(a)　SDML タグ

<SDML><SituationList>・・・</SituationList><ActionList>・・・</ActionList><RuleList>・・・</RuleList></SDML>

SDML の記述を行う際に記述するトップ階層のタグであり，これ以降の XML 文書が本稿で提案する SDML 文書であることを宣言する．このタグには，後述の SituationList，ActionList，RuleList の 3 つのタグを 1 つずつ合計 3 つのタグを記述する．

(b)　SituationList タグ

<SituationList><CoE name="*CourceName_1*">・・・</CoE>・・・<CoE name="*CourceName_n*">・・・</CoE></SituationList>

ユーザのシチュエーションを記述する部分．このタグの下の階層に状況理論に沿ったシチュエーションの記述を行う．SituationList タグはひとつの SDML にひとつだけ存在し，その下の階層には 1 つ以上の CoE タグを記述する．

(c)　CoE タグ・SoA タグ・ToS タグ

<△△△ name="*Name*"><□□□>・・・</□□□>・・・<□□□>・・・</□□□></△△△>

CoE，SoA，ToS の各タグは，それぞれ状況理論における「出来事系列(Course of Event)」「事態(State of Affair)」「状況型（Type of Situation）」に対応しており，上記例の△△△に入るタグによって，以降に続く□□□のタグが決定する．

△△△が CoE タグの場合，□□□には SoA タグが記述可能である．もしくは，必要に応じて Include タグも利用可．△△△が SoA タグの場合は，□□□には ToS タグと Include タグが記述出来る．△△△が ToS タグの場合は，□□□には Property タグ，Relation タグ，Individual タグ，Time タグ，Space タグ，Include タグが記述可能である．これらのタグに関する詳細な説明は紙面の都合上割愛する．詳細については，別途[7]を参照されたい．

(d)　Property タグ，Relation タグ

<Property name="*PropertyName*"><Individual name="*IndividualName*" /></Property>
<Relation name="*SituationName*"><Individual name="*IndividualName_1*" />・・・<Individual name="*IndividualName_n*" /></Relation>

状況理論における「属性(Property)」と「関係(Relation)」を表すタグ．状況理論では，属性や関係の中に含まれる要素は，個体であると述べられているが，ここでは，その考え方を拡張して，出来事型（ToS）の中にも時空位置を出来るように構成する．

Property タグや Relation タグの下の階層には，Individual タグを記述する．また，必要に応じて Time タグ，Space タグも指定可能．

(e)　Individual タグ

<Individual name="*IndividualName*" />

状況理論における「個体(Individual)」を表すタグ．登録済みのユーザや物体を指定する．Individual タグには終了タグは無く，内部に name 要素のみを持つタグで，上記 Property タグか Relation タグの下の階層だけに記述する．

(f)　Time タグ，Space タグ

<Time value="*TimeValue*" />
<Space value="SpacePositionValue" />

状況理論における「時空位置(Time-Space Position)」を表すタグ．本来の状況理論では SoA タグの下の階層にのみ記述出来る要素であるが，本研究ではこの考え方を拡張し，Property タグと Relation タグの下の階層にも記述出来るものとする．

他のタグと違い，Time タグと Space タグには，その記述が適用される要素が複数あり，それをスコープと呼ぶことにする．それぞれのタグのスコープは，そのタグが記述された場所よりも下の階層に存在するタグ全てである．たとえば，ある SoA タグに対して記述された場合，その SoA タグより下の階層に列挙された ToS タグ内に存在する Property と Relation 全てに該当タグの内容が適用される．もし，ひとつの Property や Relation にのみ適用させたい場合は，その Property や Relation 内に記述することで目的の結果を得ることができる．

アクションの XML 表現

2 つ目の記述構造は，記述された状況が成立したときにユーザに提示するアクションに関する XML 記述方法である．

(g)　ActionList タグ

<ActionList><Action name="*ActionName_1*">・・・</Action>・・・<Action name="*ActionName_n*">・・・</Action></ActionList>

ユーザに提供するアクションのリストを記述する．ActionList に記述するのは，後述の Action タグであり，必要に応じて Include タグも記述可能．これらの記述では，それぞれ後述のルールを作成するときに記述するアクションを指定する．

(h)　Action タグ

<Action name="*ActionName*" id=n />

特定の状況が成立したときに，ユーザに提示するアクションを記述する．ID 番号でアクションを指定する．

ルールの XML 表現

3 つ目の記述構造は，どの状況が成立したときに，どのアクションを提示するかを指定するための XML 記述方法である．

(i)　RuleList タグ

<RuleList><Rule name="*RuleName_1*">・・・</Rule>・・・<Rule name="*RuleName_n*">・・・</Rule></RuleList>

どの状況が成立したら，どのアクションを発火させるかのリストを記述する．RuleList タグの下の階層に記述するのは，ひとつまたは複数の Rule タグのみ．

(j)　Rule タグ

<Rule name="*RuleName_1*"><Situation name="*SituationName*"><Support name="*SupportName*"></Rule>

状況とアクションをペアにして記述することで，指定した状況が発生したときに，どのアクションを発火させるかを指定する．

(k) Situation タグ

<Situation name="*SituationName*">

SituationList にリストアップした状況の中から，後述のアクションを発火させるキーにしたい CoE の名前を指定する．複数指定することで，いくつかの CoE が同時に発生したときに，特定のアクションを発火させることも可能．

(l) Support タグ

<Support name="*SupportName*">

ActionList にリストアップしたアクションの中から，前述の状況が発生したときに発火させたいアクションの名前を指定する．複数指定することで，キーになる状況が発生したときに，複数のアクションを同時に発火させることも可能．

3.3 再利用のための XML 表記法

3.1 で述べた 3 つの概念以外に，本研究において重要な再利用性を実現するための表現を以下に説明する．

(m) Include タグ，include_uri 要素

引用は，既に定義されている別の場所にある記述を引用することで，再定義の冗長性を排除するための要素である．引用の記述方法には，以下で示すように，要素として記述する方法と，タグとして記述する方法がある．それぞれ引用を示すキーワードとして Include という単語を用いる．

<○○ name="NameOf○○" include_uri="sdml://*FileName*/*Path*/*To*/*Element*"><・・・>・・・</・・・></○○>

1 つ目のパターンは，既存のタグに include_uri 要素を追加で記述するものである．この記述を行った場合，引用された記述の内容は，該当する既存のタグ全体に対してスコープを持つ．この記述方法によって引用できるのは，include_uri を記述する要素と同じ要素のみである．たとえば，SoA タグに include_uri を記述する場合は，引用できる記述は SoA タグのみである．

<Include uri="sdml://*FileName*/*Path*/*To*/*Element*" />

2 つ目のパターンは，Include を独立したひとつのタグとして記述する方法である．この場合，uri 要素によって指定した，別の場所にある記述をそのまま引用する．この記述方法によって引用できるのは，Include タグを配置する階層と同じ階層に記述出来るタグのみである．たとえば，SoA タグ内に Include タグを記述する場合，SoA タグの下の階層に記述出来るタグである，ToS タグか Time タグと Space タグのみ引用可能である．

さらに，引用したタグ内に，自身より下の階層全てに対してスコープを持つタグが記述されている場合は，Include タグより下の全階層に対して，該当タグの影響が反映される．

3.4 SDML の例

ここまで述べてきた XML による状況記述の例を以下に示す．

これは，状況記述によって状況アウェアサービスが有効となる，老人への歩行経路案内のケースを例にしている．概要としては，歩行に障害を抱える老人が，階段とスロープのある駅構内の通路を移動しているというような状況を想定している．その上で，「十分早く歩行できる場合は，簡単なサービスだけを提示する」「歩くのが遅い場合には，周りにいる人物との関係によってサービスを提示するか否かを判断する」「手押し車を持っている場合にはスロープを通るように案内する」という 3 パターンの記述を行っている．

```
<?xml version="1.0" encoding="Shift_JIS" ?>
<SDML>
  <!-- 以下シチュエーションの定義 -->
  <SituationList>
    <CoE name="SymplifiedServiceSituation">
      <SoA name="AgedPersonNeedSimplifiedHelp">
        <ToS name="AgedPersonState">
          <Property name="is_aged">
            <Individual id="rfid0001" />
          </Property>
        </ToS>
        <ToS name="HaveStick">
          <Relation name="have">
            <Individual id="rfid0001" /><!-- 対象者 ID -->
            <Individual id="rfid0002" /><!-- 杖 ID -->
          </Relation>
        </ToS>
        <ToS name="WalkSpeed">
          <Relation name="walk_fast_to"><!-- 早く歩行 -->
            <Individual id="rfid0001" />
            <Individual id="rfid0003" /><!-- 登録地点 ID -->
          </Relation>
        </ToS>
      </SoA>
    </CoE>
    <CoE name="NormalServiceSituation">
      <SoA name="CloseToHelper">
        <ToS name="AgedPersonState">
          <Property name="is_aged">
            <Individual id="rfid0001" />
          </Property>
        </ToS>
        <ToS name="HaveStick">
          <Relation name="have">
            <Individual id="rfid0001" />
            <Individual id="rfid0002" />
          </Relation>
        </ToS>
        <ToS name="WalkSpeed">
          <Relation name="walk_slowly_to"><!-- 遅く歩行 -->
            <Individual id="rfid0001" />
            <Individual id="rfid0003" />
          </Relation>
        </ToS>
      </SoA>
    </CoE>
    <!-- 以下，引用(include)機能を利用した定義 -->
    <CoE name="AbortServiceSituation"
         include="sdml:///SymplifiedServiceSituation">
      <SoA name="CloseToHelper">
        <ToS name="Caretaker">
          <Relation name="have">
            <Individual id="rfid0001" />
            <Individual id="rfid0004" /><!-- 介護者 ID -->
          </Relation>
        </ToS>
      </SoA>
    </CoE>
    <CoE name="SlopeWayServiceSiktuation">
      <SoA name="SlopeWay">
        <ToS name="HaveCart">
          <Relation name="have ">
            <Individual id="rfid0001" />
            <Individual id="rfid0005" /><!-- 手押し車 ID -->
          </Relation>
        </ToS>
      </SoA>
    </CoE>
  </SituationList>
  <!-- 以下，アクションの定義 -->
  <ActionList>
    <Action name="SymplifyedService">
      <ActionId>act0001</ActionId>
```

```
    </Action>
    <Action name="NormalService">
      <ActionId>act0002</ActionId>
    </Action>
    <Action name="AbortService">
      <ActionId>act0003</ActionId>
    </Action>
    <Action name="SlopeWayGuide">
      <ActionId>act0004</ActionId>
    </Action>
  </ActionList>
  <!-- 以下，ルールの定義 -->
  <RuleList>
    <Rule>
      <Situation name="SymplifiedServiceSituation" />
      <Support name="SymplifyedService" />
    </Rule>
    <Rule>
      <Situation name="NormalServiceSituation" />
      <Support name="NormalService" />
    </Rule>
    <Rule>
      <Situation name="AbortServiceSituation" />
      <Support name="AbortService" />
    </Rule>
    <Rule>
      <Situation name="NoticeServiceSituation" />
      <Support name="SlopeWayGuide" />
    </Rule>
  </RuleList>
</SDML>
```

4. 実装

4.1 実装環境

ここまで考察してきた内容を実際に検証するため，Windows XP Professional Edition 上にインストールした Java 2 Standard Edition の Version 1.4.2 を利用して，SDML を扱うシステムを構築した．

4.2 パーサの実装と処理

本稿で特に重要となる SDML のパーサには，J2SE1.4 に含まれる JAXP を利用して CUI で構築した．

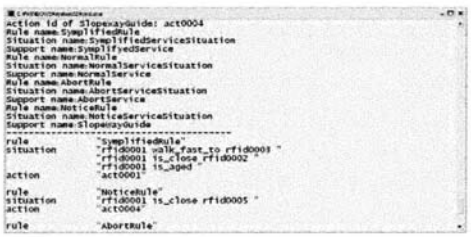

図 2　実装画面

Figure 2　An Implemented CUI

今回構築した SDML Parser では，まず，指定された SDML ファイルを読み込み，JAXP で処理できる文字列に内部処理を行う．つぎに，1 つ 1 つの SDML 要素を読み込んでメモリ上に DOM ツリーとして展開する．その後，記述内容をシチュエーション，アクション，ルールに分類しながらそれぞれのクラスに格納し，そのクラスに実装された推論ルール変換メソッドを利用して推論ルールとして出力する．

今回 SDML Parser で出力したルール言語による記述は，そのままシチュエーションアウェアサービスの構築時に利用するために実装された推論エンジンに入力可能であり，センサデータやユーザによる SDML 記述から，ユーザ状況の把握が可能になる．

5. まとめと今後の課題

このように，本稿では，状況理論に基づく記述をユビキタスコンピューティングの世界に適用可能にする拡張を行い，XML による記述の手法を提案した．また，今回提案された SDML による記述を，シチュエーションアウェアサービスの構築用に実装された推論エンジンに入力し，ユーザ状況の把握に利用できることが確認できた．

今後は，SDML 記述の発展を考慮に入れ，より柔軟かつ容易にユーザ状況の定義が出来るよう，SDML の記述ポリシーの推敲を進める．また，時間記述に関する議論を進め，一連のシチュエーション記述に必要な記述手法を考察していく．さらに，今回は主にユーザの状況を記述する手法について考察してきたが，サービスを記述するためには，ユーザ状況のほかにアクションの記述についても考察が必要である．SDML の利便性向上のためにも，容易に様々なアクションを記述する手法を考察しなければならない．

さらに，エンドユーザが SDML を利用して日常生活に必要なユビキタスサービスを定義する際に，簡単に利用できるテンプレートを充実させることで，更なる SDML の利便性の向上を目指し，有用なユビキタスサービスの実現を加速させることを目指す．

参考文献

1) 西垣弘二，安本慶一，柴田直樹，伊藤実：“コンテキストに基づいた情報家電の連携を実現するためのフレームワーク及びルールベース言語の提案，”情報処理学会研究報告，Vol.2004, No.112, pp.21-27, 2004

2) 山登庸次，中野雄介，横畑夕貴，浜田信，武本充治，須永宏，田中英里香，西木健哉，寺西裕一，下条信司，“買い物支援サービス実証実験を通したユビキタスサービス合成技術の検証，” 情報処理学会論文誌，Vol.48, No.2, pp. 754-769, 2007

3) 山登庸次，中辻真，須永宏，“ユビキタス環境で動的にサービスを実現するためのサービス合成技術，” 情報処理学会論文誌, Vol.48, No.2, pp. 562-577, 2007

4) NAOHARU YAMADA, KENJI SAKAMOTO, GORO KUNITO, YOSHINORI ISODA, KENICHI YAMAZAKI, and SATOSHI TANAKA, “Applying Ontology and Probabilistic Model to Human Activity Recognition from Surrounding Things,” 情報処理学会論文誌，Vol.48, No.8, pp. 2823-2834, 2007

5) VAROL AKMAN, MEHMET SURAV, “THE USE OF SITUATION THEORY IN CONTEXT MODELING,” Computational Intelligence, Vol. 13, No. 3, pp.427-438, 1997

6) J.Barwise and J.Perry, 1983 “Situation and Attitude,” MIT Press, Cambridge, MA.

7) http://himalayas.u-aizu.ac.jp/~m5091219/ThesisIkedaDPS133.doc

謝辞　本研究の一部は,日本学術振興会科学研究費補助金萌芽研究(18650251)の補助を受けたものである．