知的人工物を用いた知的空間構築のための情報管理基盤

藤波香織[†] 中島達夫[†]

本論文では，知的空間をセンサやアクチュエータにより拡張した知的人工物により構築する際に，複数のアプリケーション間で共有すべき情報の管理機構（*Bazaar*）について述べる．*Bazaar* は，知的人工物の仕様のような静的な情報と，使用状況や位置情報といった動的な情報で構成される物理空間モデルを内部で保持し，上位のアプリケーションに統一的なアクセスを提供するものである．これにより開発者は必要な使用状況や人工物の種類，場所の名前といった高位でそれゆえアプリケーションで直接必要な情報を中心に扱うことが可能となる．本論文ではアーキテクチャについて述べるとともに，アプリケーション開発および基礎性能測定を通した評価を行う．

An Information Management Infrastructure for Sentient Artefact-based Smart Spaces

Kaori Fujinami[†] and Tatsuo Nakajima[†]

Physical world information should be shared with various applications in a dedicated space to realize “smart spaces”. In this paper, we propose a physical information management infrastructure, *Bazaar*, that organizes a collection of self-descriptive information obtained from sensor/actuator augmented daily objects called *Sentient Artefact*. *Bazaar* separates the structures of such information and ways of gathering from their utilization, so that an application developer can concentrate on his/her main task, i.e. application logic development. We present information for building a world model and a programming model to access the model that reflects our relationship with locations and objects in the physical environment. Moreover, we report its implementation, a basic performance evaluation, and its various applications.

1. はじめに

技術の発達により人間が扱わなければならない情報やサービスが爆発的に増加し複雑化する中で，人間やデバイスが置かれている状況や文脈を反映してこれを抑え，適切なメディアにより提供することが重要となる．このような課題に対しては，知的空間（スマートスペース）といったキーワードで研究がさかんに行われている[1),3),11),16)]．知的空間内では複数のアプリケーションが共通の情報を使用して独立に，あるいは連携して存在するために，共通の「世界観」が必要となる．この情報共有により，異なるアプリケーション間での情報交換が柔軟に行えるほか，状態取得手段とその利用を分離することが可能となり，開発効率の向上が期待される．しかしながら従来は，センサのみの抽象化や物体の識別子と位置情報の関係のみの共有化により十分な共有が図れていないことや，空間内のオブジェクトごとに詳細にモデル化を行っているがために，将来にわたっての空間の拡張が困難であったり，空間を記述する複雑なボキャブラリ（オントロジ）のみを定義して空間モデルの構築はすべて開発者に委ねるためにコストがかかったりするといった課題が存在する．

知的空間の構築方法に関しては，これまで物体の位置を取得するためのセンサを天井などに埋め込む技術が開発されているが，これらは設置や維持管理コストが大きく，コストを度外視できる場合をのぞき現実的ではない．そこで我々はイスやドアといった人工物にその使用状況を取得するセンサや通常の使用の延長線上で情報を提供するためのアクチュエータを付加することで，必要な機能単位で構築することを提案している[7)]．このような人工物はセンサから得られたデータを解析する機能を持ち，使用状況（使用・不使用，使用内容など）を判定することができるもので，「知的人工物」と呼んでいる．本論文では，知的人工物により知的空間を構築する際にアプリケーション間で共有すべき情報の管理機構（*Bazaar*）について述べる．

† 早稲田大学理工学術院コンピュータ・ネットワーク工学科
Department of Computer Science, Waseda University

Bazaar は，空間内に分散する知的人工物に関する情報（オブジェクトモデル）とそれらの存在場所に関する情報（ロケーションモデル）の 2 つのモデルで構成される物理空間モデルを内部に持ち，空間内のアプリケーションの動作を整合性がとれたものにする．そして，この物理空間モデルに対する統一的なアクセス手段（API）により，知的人工物の仕様のような静的な情報と，使用状況や位置情報といった動的な情報の取得とイベント処理を実現する．これにより開発者は必要な使用状況や人工物の種類，場所の名前といった高位でアプリケーションで直接必要な情報を中心に扱うことが可能となる．

以下では 2 章で知的人工物ベースの空間構築の概略を述べ研究の動機づけを行う．3 章で情報管理基盤の設計を行い，*Bazaar* で扱う物理空間モデルについて述べる．続く 4 章ではこのモデルにアクセスするための API を解説し，5 章では現状の実装について述べる．6 章ではアプリケーション開発経験と性能評価を通して提案内容の評価を行う．そして 7 章で関連研究について述べ，8 章で結論を述べる．

2. 知的人工物を用いた知的空間構築

以下では想定する知的空間のシナリオとその構築に知的人工物を用いる動機について述べる．

2.1 想定シナリオ

知的空間に一般的に求められる性質としては，情報提供や機器動作のタイミングとそれらの内容（コンテンツ）がその場にいるユーザに適したものであることがあげられる[1),3),11),16)]．本論文においてもこのような性質を持った屋内空間を「知的空間」と定義し，以下のようなシナリオを想定している．

『東京在住の花子は翌日の午後に横浜で友人と会う約束をした．携帯電話のスケジュール帳にそれを登録し，お気に入りのクレードルに置き，いつものように目覚し時計をセットして寝た．夜中に携帯電話に着信があったが，寝ているため鳴動も振動もしない．翌朝，天気が良いため予定より早く目が覚め，アラームを止めずに洗面所で洗顔と歯磨きを行った．すると目の前の鏡の表面に情報が表示され，横浜の天気の悪化と交通機関のトラブルを伝えていたので，洋服と交通手段の変更を決めた．部屋に戻ると携帯電話は鳴動しており着信があった旨を通知していた．その晩，妹が携帯電話のクレードルを交換したいというので，花子は妹のを使うことにした』

ここで花子の部屋（家）は，彼女が指示しなくても不要な鳴動や振動を避けたり，行動予定を反映した情報を適切なタイミングとメディアで表示したりすることで意思決定のスムーズな支援を実現している．よって，上述の一般的な性質を満足しているといえる．

2.2 動　　機

このような空間を構築するにあたり，ユーザや情報提供装置の位置をタグを用いた位置計測システムにより詳細に取得することで，ユーザの現在の状況をある程度推定することはできる．しかし精密な位置計測を行うためには大規模な基盤システムが必要となり，設置だけでなく維持管理も困難になる．よって，アプリケーションで必要なコンテクスト取得（入力）と情報提供（出力）の機能を持つ知的人工物を逐次追加していくことで初期導入の容易さと長きにわたる拡張を目指すのが，知的人工物を用いるシステム面での動機である．またユーザインタフェースの面からは，使用にともなって発揮される人工物が本来持っている役割や所有者といった情報を入力とし，本来の機能の延長線上で処理された情報を提供するため，その使用者にとっては装置としての使用感を低く抑えることができる．

このように知的人工物を知的空間構築のための部品と見なす場合には，共有による情報の整合性の向上と再利用によるアプリケーション開発者の生産性の向上のために，アプリケーションから分離して管理することが重要である．次章では管理機構（*Bazaar*）への要求条件と実現方針を述べて設計を行う．

3. 情報管理基盤の設計

3.1 要 求 条 件

以下に *Bazaar* に対する要求条件をあげる．

(1) 多様な情報の共有
(2) 共有情報の拡張性
(3) 低レベルの情報共有
(4) 容易な空間拡張
(5) 非知的人工物の透過的な統合

(1) は位置情報や使用状況といった動的な情報だけでなく，種別，所有者といった様々な情報がコンテクストとして意味を持ち，共有される必要があることを意味している．これは 2.1 節のシナリオにおいて，歯ブラシが，その他人と共有されないという性質から，所有者情報を活用することで使用者を推定することができる．また，空間が長きにわたり存続していく中で新たな情報共有が必要となった場合にそれを容易に実現できる必要がある (2)．(3) に関しては最低限のシンプルなモデルのみを共有し，アプリケーション依存の部分についてはそれを容易に構築可能になるような機構を提供すべきである．これは，たとえば「睡眠」と

いうユーザのコンテクストについてもアプリケーションごとに意味づけが異なるため，詳細で複雑な統一モデルはかえって共有を阻害することが考えられるからである．さらに，あらかじめ部屋や建物単位に大規模な位置計測システムを設置するアプローチと異なり，我々のアプローチは必要に応じて知的人工物を追加したり，その存在位置を特定するセンサを追加したりするためにこれが容易に行える必要がある (4)．そして最後に，計算機能を持たない非知的人工物であっても「タンスの近く」といったランドマークのような活用が見込めるためにその透過的な統合が必要である☆(5)．

3.2 *Bazaar* アーキテクチャ概要

3.2.1 前　　提

Bazaar ではすべての物理オブジェクトに対する外部で検出可能な識別子（ID）の付与，およびホームゲートウェイのような集中処理とインターネットアクセスを可能とする装置の存在を前提としている．なお，ID に関連づいた属性情報（後述のオブジェクトモデル，あるいは定義情報）はあらかじめ統一されたスキーマによりその製造元が提供することとする．

3.2.2 構 成 要 素

Bazaar は**図 1** に示す 6 つの要素により構成される．すなわち，1）低レベルのコンテクスト情報源である物理オブジェクト（Artefact），2）物理オブジェクトに付与された ID の検出器（IDDetector），3）任意の空間情報を管理し，そのアクセス手段を低レベル API により提供する Bazaar World Model Manager（BWMM），4）高レベルのコンテクスト情報を抽出するためのフレームワーク，5）アプリケーションの動作を表現するアプリケーションロジック，6）ID に対応した定義情報の格納場所を解決する IDResolver である．

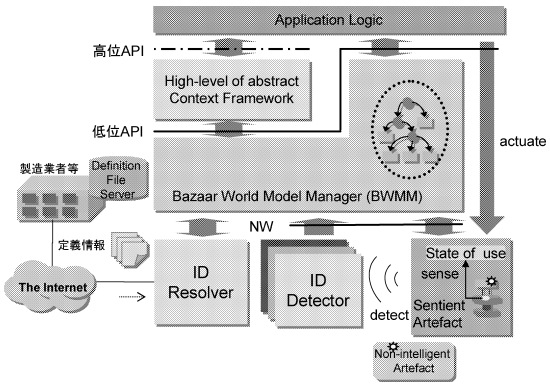


図 1 *Bazaar* 構成要素
Fig. 1 *Bazaar* overall architecture.

☆ 知的人工物と非知的人工物を同一に扱う際には，物理オブジェクトと称する．

3.3 物理空間モデル

物理空間モデルは，IDDetector の設置位置に関する情報を扱うロケーションモデルと物理オブジェクトに関する情報を扱うオブジェクトモデルで構成される．本物理空間モデルは，我々が提案する知的人工物を用いたサービス開発のために用いるものであり，特に使用状況（入力）に関する情報については，知的空間のための情報モデルを保持する他のアプローチ[4),16)]には見られない．その一方で，ユーザに関する情報は提案モデル中では持たない．これは，他の物理オブジェクト同様にユーザに ID を常時携帯させて位置検出することが非現実的であることと，知的人工物の使用状況から得られるユーザ行動に関する情報は，アプリケーションごとに解釈が異なるために共有すべきでないと判断したためである．以下の項では RDF による表現方法について述べたあと，これに基づく各モデルの設計を例を交えて述べる．

3.3.1 RDF による表現

Bazaar においては多様な属性を柔軟に扱うために，セマンティック Web の要素技術として注目されている RDF（Resource Description Framework）[13)] を用いる．RDF は本質的には「リソース」と呼ばれる任意の事物を定義するための（主語（Subject），述語（Predicate），目的語（Object））で表現される三つ組（Triples）である☆☆．このため再帰的に属性情報を詳細定義することが可能である．よって，追加削除が容易かつ局所的，断片的な情報を用いた柔軟な検索が可能といった特徴があり，多様な属性を取り扱うのに適していると考えた（要求条件 (1) および (2)）．また記述情報のパースや任意情報の検索，構造の可視化のためのツールが豊富に提供されているという点も選択理由としてあげられる．

3.3.2 ロケーションモデル

ロケーションモデルは，部屋や建物のような単位空間とそこに存在する玄関，台所，ドア周辺といったシンボリックな名称で識別される単位領域の関係を表すものである．単位領域は IDDetector ごとに設定されるために，IDDetector と単位空間の関係についても表現

☆☆ 後出の図 2 において主語（左端）と目的語（右端）は矢印で結ばれる．また，矢印の脇に書かれているのが述語である．また，楕円はリソースを表し矩形はリテラル（値）を表しており，リソースは他の三つ組の目的語になることができる．また，述語は通常，名前空間を示すための接頭辞が与えられるが本論文では簡便のためにこれを省略する．

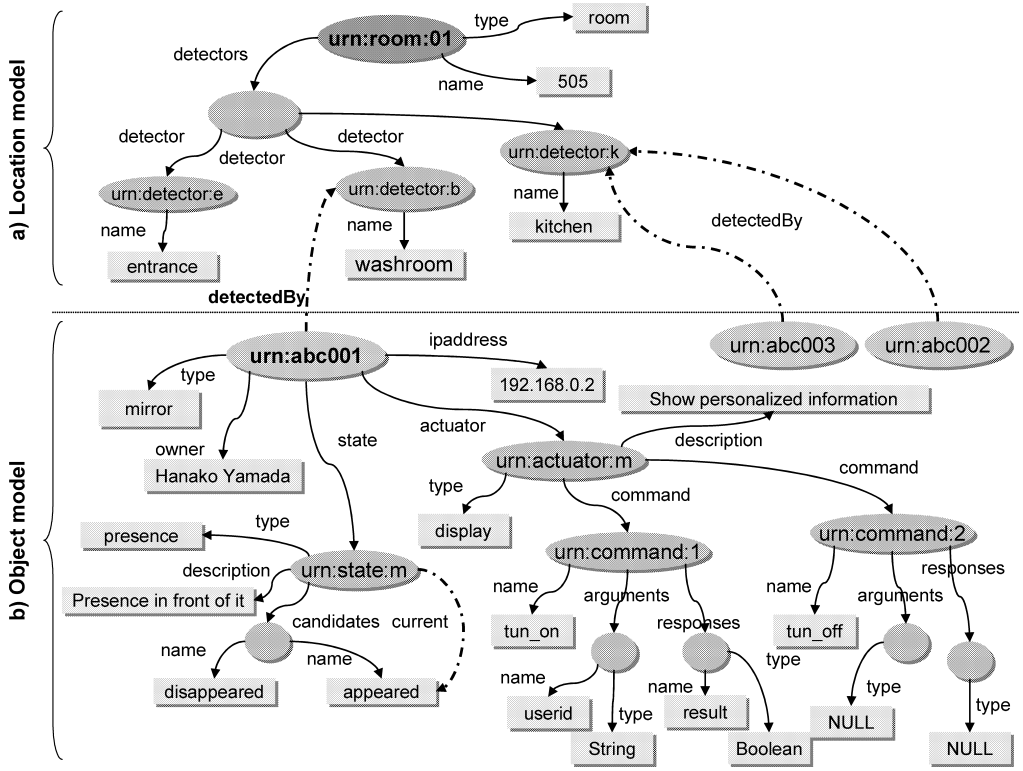


図 2 RDF グラフ表現による物理空間モデル：a) ロケーションモデル，b) オブジェクトモデル
Fig. 2 World model by RDF graph representation: a) Location model and b) Object model.

する．次節で述べるように ID の検出を契機に BWMM に登録される物理オブジェクトは，この IDDetector に関連づくことになる．**図 2** a) にロケーションモデルの RDF グラフ表現を示す．ここでは，単位空間 505 には 3 つの IDDetector（`detector`）が存在しており，それに対応する玄関，洗面所，台所の単位領域の存在を表している．また，`urn:abc001` で表される物理オブジェクトは洗面所に関連づいた IDDetector に検出され，`urn:abc002` および `urn:abc003` は台所で検出されていることを表現している．なお，単位領域間の隣接や単位空間間の包含関係のようなトポロジカルな関係については極力シンプルな構造とするために本ロケーションモデルの範疇外とした．これらはアプリケーションの一部として定義する．よって，1 つの BWMM は複数の単位空間を扱うことはできるが，それらの関係は外部（アプリケーション）で管理する．

3.3.3 オブジェクトモデル

オブジェクトモデルは物理オブジェクトの静的な仕様や動的な実行状態を表している．要求条件 (3) であげた低レベルの共有情報として，種別（`type`），使用状況（`state`），存在場所（`detectedBy`），アクチュエーション機能（`actuator`），所有者（`owner`），ネットワークアドレス（IP アドレス（`ipaddress`））を用いる．図 2 b) に本オブジェクトモデルの RDF グラフ表現を示す．

ここで，1 つの物理オブジェクトに対して複数の単位領域の関連づけを許す．これは，検出領域が重複している IDDetector により同時に ID が検出された場合には現実に起こりうることであり，複数箇所で検出されているという事実をそのまま表し，モデルを利用する側が解釈を加えるべきであるためである．また，使用状況についても複数持ちうることとする．これは，「着座」と「45°（北東）」という着座有無と背もたれの方角という 2 種類の分離可能な情報を同時に取得するイスなどが考えられるからである．またアクチュエーション機能に関しては，外部から同機能にアクセスする際のコマンド情報（`urn:command:`n 以下のノード，すなわち，コマンド名，引数名および型，返り値名および型）が設定されるが，このような機能を持たないものもあるためにすべての物理オブジェクトに必要なものではない．所有者情報は 2.1 節のシナリオ中に現れる歯ブラシのように他人と共有する可能性がきわめて低いものについて，使用者に近似することができる．ネットワークアドレスは，後述のように

アプリケーション側から知的人工物にコマンドとしてアクセスする際の特定に用いられる．これらのうち，変化の候補が存在する使用状況と存在場所については，使用や移動にともない自動的にリンクが張り替えられる（同図一点鎖線部分）が，ネットワークアドレスと所有者については随時書き換えられる．

なお同図は「urn:abc001 で識別される洗面所に存在する鏡の所有者は “Hanako Yamada” であり，現在だれかが使用している（前面にいる）．また，パーソナライズされた情報提示ができ，“turn_on” または “turn_off” というコマンドにより制御可能である．“turn_on” 時には文字列でユーザ ID を引数として渡し，論理型の返り値を受け取る」ことを表す．

3.4 空間の機能と規模の拡張

要求条件 (4) としてあげている容易な空間拡張について説明する．*Bazaar* の核となる考え方は自己記述的な物理オブジェクトの集合により物理空間を表現するところにある．新たなサービスを提供するにあたり，不足している情報（ユーザ状況）がある場合に，それを提供可能な物理オブジェクトを新規に追加することで，空間の機能を拡張することができる．サービス要件に従い使用する物理オブジェクトが決定され，各物理オブジェクトに付与された ID が IDDetector により検出されると，製造元から定義情報が取得され，その IDDetector の設置場所情報が当該物理オブジェクトの存在場所と解釈されて統合されることで，物理空間モデル（BWMM）の一部を構成することになる．たとえばユーザを特定する機能が必要になった場合には，それを実現可能な物理オブジェクトとして歯ブラシを購入するなどして対象となる空間に導入する．すると，歯ブラシに貼付された ID をもとに当該知的人工物自身あるいは製造業者の提供するインターネット上のサーバより定義情報がダウンロードされて BWMM に統合される．これは，図 2 においては物理オブジェクトを表すリソースである urn:abc*XXX* から IDDetector を表すリソースである urn:detector:*X* へのリンクの生成を意味する．

一方で，空間の規模も必要に応じて拡大していくことも求められる．*Bazaar* では単位領域ごとにそこへの物体の入退出を検出する IDDetector が存在し，この追加が規模を拡大していくことになる．IDDetector の制御プログラムは起動時に BWMM を発見し，自身の識別情報とともに設置者により割り当てられた単位領域の名称を登録することで，それ以降検出された物理オブジェクトを物理空間モデルの一員とすることが可能となる．これは，図 2 においては空間内の IDDetector の集合を表す無名リソースから，IDDetector を表すリソースである urn:detector:*X* へのリンクの生成を意味しており，非常に容易に追加することができるといえる．これにより ID 検出手段を問わずに空間拡張を行うことが可能となる．さらに，計算機能を持たない非知的人工物であってもこの枠組みで同様に物理空間モデルに取り込むことができる（要求条件 (5)）．このような IDDetector には，無線タグシステムや Active Badge システム[17] のように外部の検出器により ID を検出するシステムが該当する．

4. API の設計

アプリケーション開発者が *Bazaar* を扱ううえでの API について説明する．次節ではプログラミングモデルについて説明し，API についての要求を示す．

4.1 プログラミングモデル

これまで述べてきたように知的人工物によるアプローチでは，使用状況，所有者といった情報を用いることで位置情報以上のコンテクストを抽出すること，および適切なアクチュエーションにより情報を伝達することを目指している．よって，このような情報を *Bazaar* より取得したり，アクチュエーションを行ったりするための API が必要となる．この API の利用は，(1) *Bazaar* 契機の情報提供と (2) いわゆる getter メソッドによるアプリケーション契機での情報取得，の 2 つに分類できる．

4.1.1 *Bazaar* 契機

Bazaar 契機の利用では対象としている空間内での変化を関心を示しているアプリケーションに通知するもので，イベント通知型ということになる．このとき，3.3 節で述べたモデル内で比較的頻繁に変化する情報である「位置」と「使用状況」の変化をイベントとして扱う．また，位置に関しては視点を物理オブジェクトに置く場合と単位領域に置く場合の 2 つが考えられるためこの両方をサポートする．これはたとえば，特定の歯ブラシがある場所から別の場所に移動したことを通知するイベントと，洗面所という特定の場所に対するある物理オブジェクトの侵入（退出）というイベントを考えるということである．なお次に説明するように，通知に先立ちその発行元となるオブジェクトを取得してイベントに対する関心を登録する必要がある．

4.1.2 アプリケーション契機

アプリケーション内で *Bazaar* が提供する情報を処理するにあたり，BWMM より物理オブジェクトまたは位置を表すオブジェクトを取得する．これらは図 2 b) においてはそれぞれ urn:abc*XXX*，urn:detector:*X*

で表されるリソースに該当する．また，取得された物理オブジェクトから使用状況，アクチュエータといったリソースを次節で述べるオブジェクトとして取得する．これらは RDF のうえではリソースの属性値を取得する行為にほかならない．また，種別（type）や所有者（owner）といった属性値もそれを所有するオブジェクト（この場合は物理オブジェクト）から取得する．

4.2 API

オブジェクトモデルは図 2 b) にあるように共通のスキーマで表せるために，Chair や Mirror，SeatState や PresenceState といった独自のクラスを提供するのではなく，それぞれ Artefact，State というカテゴリごとに汎用的なクラスを提供し，実体を表す情報はこれらをインスタンス化したオブジェクト内で保持し，汎用的なメソッドによりアクセスできるようにした．これは．アクチュエータ Actuator，位置 Location についても同様である．これにより，将来にわたって無限に存在するこれらのカテゴリ内の種別を API 設計時に意識せずに済む（要求条件 (2)）．以下に，各カテゴリごとの代表的なメソッドを示す．なお各クラスにはオブジェクトの取得にあたり，World クラスに見られるような -ByAttrs をメソッド名の末尾に持ち Hashtable を引数にとるメソッドが存在するが，紙面の制約で省略する．これは Hashtable 内のエントリを論理和で結合した条件で検索した結果を取得することを意味しており，多様な属性の柔軟な検索を実現する．

4.2.1 単位空間（World）クラス

本クラスは単位空間を表す最上位のクラスであり，アプリケーションはここから場所と物理オブジェクトを表すオブジェクト（Location，Artefact）を取得する．なお，5 番目については状態の種別名（s）と値（v）を与えることで，ある使用状況下にある物理オブジェクトを取得するというものである．

```
Location[] getLocationsByAttrs(Hashtable a);
Location[] getLocationsByID(String id);
Artefact getArtefactByID(String id);
Artefact[] getArtefactsByLocation(Location l);
Artefact[] getArtefactsByState(String s, String v);
Artefact[] getArtefactsByAttrs(Hashtable a);
```

4.2.2 単位領域（Location）クラス

本クラスは単位領域を表すものであり，名称を取得するもののほかに，当該領域に対する入退出通知用のリスナ登録（削除）メソッドを持つ．入退出を検出すると検出した ID とともに（検出した IDDetector に関連づけられた）場所や時間といった情報を内包したイベント（DetectionEvent）をリスナに対して発行する．

```
void addDetectionListener(DetectionListener il);
```

4.2.3 物理オブジェクト（Artefact）クラス

本クラスは物理オブジェクトを表すものであり，内包する使用状況，存在場所，アクチュエータを表すオブジェクトを取得するためのメソッドを提供する．また，存在場所の変化を通知するためのリスナ登録（削除）メソッドを持つ．LocationChangeEvent として表現される存在場所の変化としては，旧領域からの退出と新領域への侵入という 2 つの情報が与えられ，利用側で適切な処理を記述することができる．さらに種別，所有者といった情報を取得するためのメソッドも提供される．

```
State getStateByType(String s);
Location getLocationByName(String n);
Actuator getActuatorByType(String t);
void addLocationListener(LocationListener ll);
String getType();
String getOwner();
```

4.2.4 使用状況（State）クラス

本クラスは使用状況を表すものであり，「（鏡の前での）存在状況」といった単位で存在する．そして，「存在（図 2 b) における appeared）」や「不在（同 disappeared）」といった「値」を StateChangeEvent に内包して通知する．そのためのリスナ登録（削除）用とともに，最新状態の問合せ用のメソッドも提供する．またこのイベントオブジェクトには変化の発生時間や発行元の物理オブジェクト ID も含まれる．

```
void addStateListener(StateListener sl);
String getCurrentState();
```

4.2.5 アクチュエータ（Actuator）クラス

本クラスはアクチュエーション機能を使用する際に利用されるものであり，実行する処理を表すコマンド名（n）とともに必要な引数（p）を与える．

```
Response execute(String n, Hashtable p);
```

4.3 記 述 例

API の使用例として，2.1 節で述べたシナリオに登場する鏡と歯ブラシの利用に関する記述を**図 3** に示す．これは，「洗面所にある鏡がその前面に人の存在を検出したときに，同じく洗面所で歯ブラシの使用を

```
1: boolean appeared, brushing, lsnrAdded;
2: Artefact mrr, tooth;

3: void init(){
4:   Location wr = new Location("washroom");
5:   wr.addDetectionListener(this);

6:   Artefact[] m = w.getArtefactByLocation(wr);
7:   for( int i=0;i<m.length;i++){
8:     String artefactName=m[i].getType();
9:     if(artefactName.equals("mirror")){
10:      mrr=m[i].clone();
11:      mrr.addStateListener(this);
12:      break;
13:    }
14:  }
15: }

16: void onEntered(LocationChangeEvent e){
17:   Artefact a=w.getArtefactByID(e.getID());
18:   String type=a.getType();
19:   if(type.equals("toothbrush")&&!lsnrAdded){
20:     tooth=a.clone();
21:     tooth.addStateListener(this);
22:     lsnrAdded=true;
23:   }
24: }

25: void onRemoved(LocationChangeEnvet e){
26: }

27: void onStateChanged(StateChangeEvent e){
28:   String s=e.getType();
29:   String v=e.getChangedState();
30:   if(s.equals("presence")){
31:     if(v.equals("appeared") appeared=true;
32:     else appeared=false;
33:   }
34:   if(s.equals("brushState"){
35:     if(v.equals("started") brushing=true;
36:   }
37:   if( appeared && brushing ){
38:     String owner=tooth.getOwner();
39:     showInformation(owner, true);
40:   }else{
41:     showInformation(null, false);
42:   }
43: }

44: void showInformation(String owner, boolean on){
45:   Actuator act=mrr.getActuatorByType("display");
46:   if(on){
47:     Hashtable arg=new Hashtable();
48:     arg.add("userid", owner);
49:     act.execute("turn_on", arg);
50:   }else{
51:     act.execute("turn_off",null);
52:   }
53: }
```

図 3 *Bazaar* 低レベル API を用いたコーディング例
Fig. 3 Sample code using *Bazaar* Low-level API.

検出した場合にはその所有者に関連づいた情報を表示するように鏡に指示する」というロジックを表現したものである．なお，プログラム中の文字列は図 2 の物理空間モデル中にみられる「述語（矢印の脇に記載）」または「目的語（矢印の先）」に対応している．また，`w` は単位空間 `World` への参照を表している．取得結果が 0 件であった場合や取得時に例外が発生した場合などの処理は省略している．

まず，初期化時に呼ばれる `init` メソッドでは洗面所を表す単位領域（`Location`）を取得し（4 行目），そこに対する入退出イベントの通知を希望する `DetectionListener` を登録する（5 行目）．なお，このリスナインタフェースのメソッドは 16 行目から始まる `onEntered` と 25 行目からの `onRemoved` である．その後，この単位領域を引数として `Artefact` オブジェクトを取得する（6 行目）．この場所には様々な種類のものが存在する可能性があるため，種別をチェックして鏡（`mirror`）であるときに使用状況変化イベントの通知を希望する `StateListener` を登録する（7–11 行目）．このリスナのメソッドは 27 行目からの `onStateChanged` メソッドである．このように関心のある場所や物理オブジェクトの状態を表すオブジェクトを取得し，イベント処理のためのリスナを登録するのが基本的な初期化時の記述である．

`onEntered` メソッドでは，洗面所を表す単位領域への侵入が検出されたときの処理を記述する．この例では侵入したのが歯ブラシ（`toothbrush`）であった場合には初回に限り `StateListener` を登録する．一方，`onRemove` メソッドでは登録したリスナの削除や表示の停止といった処理を記述するがここでは割愛する．

使用状況の変化を処理する `onStateChanged` メソッドでは，`StateChangeEvent` から得られた使用状況種別（`s`）とその値（`v`）から得られる鏡と歯ブラシの使用状況もとに 37 行目で情報表示可否を判定している．真の場合には歯ブラシの所有者を取得して（38 行目），`showInformation` メソッドを呼び，鏡のアクチュエータを取得し（45 行目），引数を設定し（48 行目），表示コマンドを発行している（49 行目）．

5. 実　　装

図 4 に，これまで説明してきた知的人工物，IDDetector，BWMM，AP ロジックの関連を示し，これを用いて実装について説明する．BWMM は Java で実現されており，前述のようにホームゲートウェイのような中央のホストマシン上で動作することを前提としている．空間内に分散する知的人工物は起動時のマルチキャストによるサービス発見により BWMM を特定し，以降は HTTP により通信を行う．よって知的人工物側の実装言語は問わない．また知的人工物はアクチュエーション要求受付のために自身のアドレスを

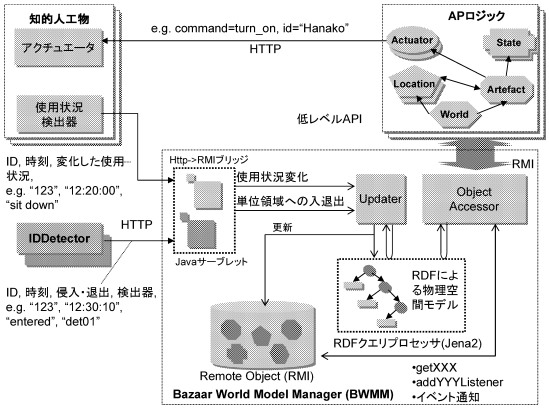
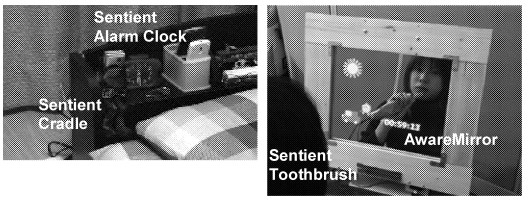


図 4 知的人工物，IDDetector，BWMM，アプリケーションの関係
Fig. 4 Relationship between sentient artefact, IDDetector, BWMM, and application.

通知する．これが図 2 b) 中の `ipaddress` 属性として管理されているものであり，`Actuator` オブジェクト内に格納されて使用される．また IDDetector も同様に BWMM を発見して，自身の情報（担当領域や自身の識別子）を登録する．実現にあたり RFCode 社のアクティブ無線タグシステム（Spider）を用いており，各物理オブジェクトに貼付された無線タグを半径 3 m 程度の検出分解能で検出する．

ロケーションモデルおよびオブジェクトモデルは XML により記述された RDF ファイルとして与えられるが，パースされた後に前述の 5 つの基本オブジェクトに属性を格納される．このうち，`World`，`Location`，`Artefact`，`State` については RMI におけるリモートオブジェクトとして実現されており，異なるホスト上に存在するアプリケーションロジックはこれらのリモートオブジェクトを介してイベント登録や受信，知的人工物の制御用オブジェクト（`Actuator`）を取得する．なお，RDF のパースやクエリのためのミドルウェアとして Jena2 [10] を用いている．

知的人工物側で検出された使用状況の変化は，その発生元の ID，発生時間，変化内容といった情報をともなって BWMM 側に渡される．そして，ID と変化内容をキーとして RMI レジストリ（`rmiregistry`）より取得された `State` オブジェクトは，そこに登録されているリスナを通じてこの変化をアプリケーション側に通知する．同時に RDF モデルにもこの変更を反映する．反映にあたっては，同様に ID と変化内容を用いて更新すべき属性値を決定する．たとえば図 2 において，ID が `abc001`，状況の種別が `presence`，変化した値が `disappeared` であった場合には，図中の `current` が指しているリンクが `disappeared` に変更となる．なお，変更箇所の特定にあたっては RDF Data Query Language（RDQL）[15] と呼ばれる SQL に似たクエリ言語を内部的に用いている．IDDetector から通知される単位領域への入退出イベントの場合も同様の手順で，リモートオブジェクトの取得と物理空間モデルへの反映を行う．この場合は，検出した ID と IDDetector の識別子が利用される．



(a) Unobtrusive Cradle (b) AwareMirror

図 5 アプリケーション構成：(a) Unobtrusive Cradle, (b) AwareMirror
Fig. 5 Sentient artefact deployment in the prototype applications: (a) Unobtrusive Cradle, and (b) AwareMirror.

また，4.1.2 項や 4.3 節にみられるようなアプリケーション側からのオブジェクト取得時には様々な属性をキーとして検索がなされるが，ここでも RDF のクエリを発行して得られた情報から該当するリモートオブジェクトを特定している．このように *Bazaar* では RDF を柔軟なクエリ処理のために用いている．

なお，ID に関連づいた定義情報の URL を解決する IDResolver は，現在はあらかじめテキストファイルに ID と URL のペアを記述しているが IDDetector が使用するロケーションシステムが EPC（Electronic Product Code）システム [5] に基づく場合には ONS（Object Naming System）を適用できると考える．

6. 評　　価

6.1 アプリケーション開発経験

2.1 節であげたシナリオを *Bazaar* を用いて実現した結果について知的人工物の様々な属性を共有したことの利点，開発にあたっての着眼点・注意点，API の有用性について述べる．

6.1.1 Unobtrusive Cradle

本アプリケーションは，シナリオに登場した携帯電話クレードルを就寝時以外にのみ機能させるために他の知的人工物の使用状況を使用するというものである（**図 5** (a)）．「就寝中」というコンテクストは，(1) 排他的コンテクストの利用，(2) 就寝時に使用する道具の使用状況の利用により抽出している．(1) については就寝時には使用しえないものの使用有無を用いると

いうもので，ここでは歯ブラシを用いる．また（2）では目覚し時計のアラームセットを就寝，解除を起床状態と見なす．

前述のシナリオの最後の一文にあるように交換により他人のクレードルを使用する可能性があるために，制御対象であるクレードルを特定するのに所有者情報を用いることはできない．よって，場所を中心に考えることにした．はじめに “Hanako の部屋” という名前を持つ単位領域を取得し，それをもとにこの領域内にあるクレードルを表す `Artefact` オブジェクトを取得する．目覚し時計についても同じ領域に存在するものを使用するという前提で，この単位領域を用いて取得をする．歯ブラシについては，今回は「その部屋の所有者は Hanako である」という開発者側の知識を用いて，所有者が Hanako であるものを取得している．しかしながら，リソース「検出器」に新たに「所有者（`owner`）」といった属性を定義することで部屋の所有者から歯ブラシオブジェクトの取得を自動的に行うことができる（要求条件 (1) および (2)）．

シナリオであげたようにクレードルのような持ち運びが容易なものは，長期の使用中に別の場所に移されることがある．特定の場所に存在することに意味がある場合には移動により役割を果たせなくなる．また後ほど，当初使っていたものと同種であるが異なる ID を持つものが再び移されて利用可能になること（交換）もある．このため，移動を検知した段階で登録してあるイベントリスナを削除する必要がある．そして，代替物の検索，同一のものが再び当該空間で検出するまで待機，当該空間の所有者に通知，といった処理を行うことになる．このため，特定の物理オブジェクトの移動に関するイベントを扱う `LocationListener` が役立つといえる．

本アプリケーションはクリティカルなものではないために，「就寝中」の定義についても上記のような 2 種類の情報を用いているが，より正確な状態が必要な場合にはベッドに埋設した体圧センサの変化や心電図や脳波などの情報を取得するための知的人工物を用いればよい．この場合には新たに品質や性能を表す属性を追加する必要があるが，これまで述べたように API を通してその種別やそれらの属性を容易に取得利用できるという点で要求条件 (3) を満たしているといえる．

6.1.2 AwareMirror

本アプリケーションは鏡を拡張したものであり，図 5 (b) にみられるように，その前に立っている人物にとって重要な情報を提供するというものである[6]．人物の特定には他人との共有がないという点で歯ブラシを選んでおり，所有者情報が用いられる．また，赤外線距離センサにより鏡の前面における物体（人物）の検出を行い，この情報と鏡付近の歯ブラシの使用を組み合わせることで，情報提供対象とタイミングを決定している．アプリケーションロジックとそれを実現したプログラムについては 4.3 節で説明している．

現状では歯ブラシの使用が前提となっているが，これはクシやひげ剃りのように他人との共有がないものであれば何でもかまわない．このようなロジックに対応するためには図 3 では `onEntered` メソッドに使用する可能性がある知的人工物の種別を列挙することで対応は可能である．しかしながら，将来を見越したうえですべてを列挙するのは不可能であるために，オブジェクトモデルに新たな属性を追加するのがよいと考える．これには，当該種別から発揮可能な「機能」と本来持つ「性質」の 2 種類が考えられる．前者は「個人の特定」や「物体の存在検知」といったものが該当するが，後者は「個人使用」や「共有」などが該当し，より低レベルな情報である．いずれを採用するかは共有情報としての価値の点から今後検討の余地があるが，どちらの場合でも `role` や `characteristic` といった属性を追加し，`getArtefactsByAttr` のような汎用的な引数を受け付けるメソッドの使用により *Bazaar* の API 側を変更せずに対応が可能であり，要求条件 (1) 多様な情報の共有，および (2) 共有情報の拡張性，に適合しているといえる．

なお，本アプリケーションで使用されている歯ブラシは上述の Unobtrusive Cradle において排他的コンテクスト取得手段として使用するものと同一である．Unobtrusive Cradle が先に存在していたために，歯ブラシは既存のものを活用することができ，洗面所への IDDetector と鏡の設置のみで済ませることができた（要求条件 (4)）．

6.2 基礎性能評価

前述のように，*Bazaar* ではホームゲートウェイのようなコンピュータ上で集中的に物理空間モデルの管理を行うために，中心機能である検索と状態更新に関する高負荷時の性能劣化傾向を測定した．

6.2.1 手　　法

検索および状態更新の評価ともに，異なる ID を持つ任意数（10，20，50，100，200，500，1,000，2,000）の BWMM 内の物理オブジェクトに対する検索および更新処理時間を測定した．その際，負荷状態との関係も明らかにするために人為的に一定間隔ごとに検索または更新処理を発行させておき，計測対象のメソッドを呼んだ．発行間隔は 200，500，1,000 ミリ秒の

表 1 測定環境
Table 1 Measurement environment.

マシン スペック	CPU：Pentium4 1.9 GHz メモリ：512 MB，HDD：40 GB
OS	WindowsXP
実行環境	Java2 SE 1.4.2_05

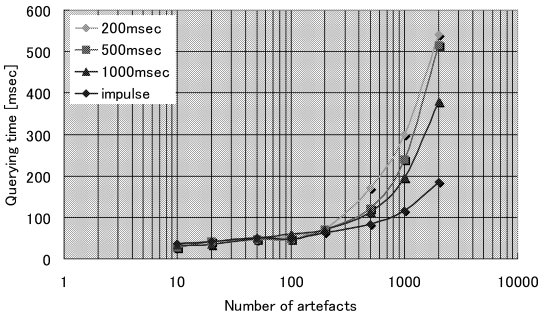


図 6 物理オブジェクト数と取得時間の関係
Fig. 6 Relationship between searching time and number of description files.

3 種類を用い，発行時に使用する ID はすべてランダムに選択した．また比較のために，単発の処理リクエストを発行した．なお，処理時間は Java 言語内で提供されている時刻取得メソッド（`System` クラスの `currentTimeMillis` メソッド）を当該処理の前後に記述し，その差分をとり算出した．**表 1** に測定環境を示す．

6.2.2 検　　索

検索処理は ID による `Artefact` オブジェクト取得（`World` クラスにおける `getArtefactByID` メソッド）を対象とした．**図 6** に示した片対数グラフにより，オブジェクト数が 200 を超えた時点での処理時間の差が明確になった．図中で「impulse」と書かれているものは上述した比較のための単一リクエストをさす．検索では物理オブジェクトが提供可能な使用状況定義やアクチュエータ定義といった情報も取得しているために，特定の値を更新する場合と異なり時間を要する．

また開発段階で様々な検索式を試験するために，柔軟に記述できる RDQL を BWMM 中で用いている．RDQL で記述された情報は Jena2 内部のインタプリタにより解釈されたのちに独自のオブジェクト構造に変換されており，このオーバヘッドが存在するものと考える．しかしながら，検索要求は ID が付与された物体の移動またはその使用状況の変化を契機として発生するため，使用者と存在する物理オブジェクト数が少ない比較的小規模な環境では，継続的に 500 ミリ秒以内の間隔で検索が発生している状況は考えにくいため大きな問題にはならないと考える．しかしながら，

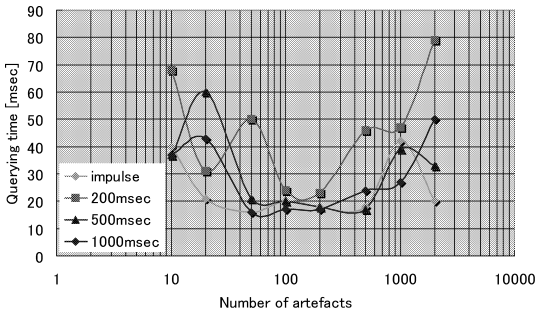


図 7 物理オブジェクト数と更新時間の関係
Fig. 7 Relationship between updating time and number of description files.

保有物理オブジェクト数が数千を超えるような場合には秒単位のリクエスト発行間隔でも遅延が大きくなることが予想されるため，単位空間（`World`）を分割する必要が生じる．現状では分割された単位空間どうしを関連づける術を提供していないため，ある単位領域がどの単位空間に属するかといった情報は外部で提供する必要がある．

6.2.3 状 態 更 新

状態更新においては `State` オブジェクトの値変更，すなわち知的人工物からの使用状況の更新通知の受信処理を対象とした．**図 7** に結果を片対数グラフとして示す．この図より，最大でも 60 ミリ秒程度の間隔に収まっており定義ファイル数および更新間隔の影響は小さいことが分かる．更新対象を決定するにあたり検索用 RDQL クエリを発行しているが処理時間が短い理由として，指定された ID の物理オブジェクトが持つ，指定された種別の使用状況を表す RDF リソースを検索するという単純なクエリであることが原因であると考えられる．

7. 関 連 研 究

本章では，1）知的空間の構築方法，2）基盤システムにおけるモデルの実現方法，3）API，について関連研究を述べる．

本提案では知的空間を実現する手段として知的人工物を用いている．知的空間実現のためのユーザ状況の取得に関しては，物体の位置検出[1),17)]や画像解析[3),12)]がよく知られている．しかしながら位置情報に頼るがゆえの高精度の位置検出を実現するためには，障害物の影響を考慮し天井などに綿密に練られた配置計画をもとに緊密にセンサを配置する必要があり，導入時ばかりでなく継続的なメンテナンスも必要となる[9)]．また，画像解析の場合には対象となる物体（ユーザを含む）は ID 情報を保持する必要がないものの，背景色

や形状の変化などに対する敏感さや高い計算機パワーを要すること，そしてプライバシの侵害感を与えるといった課題があり，一般家庭などでの使用には制約がある．一方で知的人工物を用いたアプローチでは使用状況や所有者といった多様な情報を利用可能であるため，位置情報に関しては精密な情報は必要ないといえる．このため，3.4 節で述べたように必要に応じて ID 検出器を設置すればよくなり，初期導入・維持管理コストを抑制することができると考える．Mediacup[2] とそれに続く SmartITs プロジェクト[8] ではセンサや計算機能による日常物拡張のプロトタイピング環境と様々な拡張例を示しており，我々の知的人工物の考え方に影響を与えている．これらは主として日常物の拡張という新しいアイデアの検証が目的であり，本提案で取り上げているような情報表現形式や知的人工物レイヤ，物理空間情報管理レイヤ，アプリケーションロジックレイヤで構成されるシステマティックな統合機構は提案されていない．

物理空間のモデリングに関しては，相互運用性と推論のような高度な知的処理を行うことを目的として，構成要素となる物理オブジェクトや人，場所の個々の定義や様々な関係をオントロジとして定義するアプローチがある．CoBrA[4] や Semantic Spaces[16] は知的空間を構築するための基盤システムであるが，オントロジはアプリケーションごとに使用される情報の意味を説明する「タグ」として用いられており，*Bazaar* の物理空間モデルのようなアプリケーションから独立した共有情報源を持たないという点で異なる．オントロジにより，3.2.1 項で前提としている定義情報の統一的なスキーマをより柔軟に定義することや，6.1.2 項で述べたような機能や性質の近さや置換可能性を定義することで，より柔軟に知的空間を構築することができると考える．この場合の推論機構は，*Bazaar* の低レベル API 上に構築される「高レベルコンテクスト抽出フレームワーク」として提供できると考える．また Satoh は包含関係に基づくロケーションモデルと分散型の基盤システムを提案している[14]．このモデルでは物体だけでなくサービスを表すソフトウェアやそれを実行する計算機の位置を透過的に扱えるという特徴を持っているが，位置変化を対象としているという点において様々な属性を使用する知的人工物ベースの空間構築を目的とした *Bazaar* とは異なる．

API に関しては，*Bazaar* ではクラスとして表現されるものは `Artefact` や `Location` といった概念であり，歯ブラシや玄関といった個々の種別ではない．一方で Sentient Computing プロジェクトでは，CORBA オブジェクトとして約 80 種類の人間やデバイスを表すオブジェクトが提供され，位置情報だけでなく CPU 使用率などのマシンリソース状態，ネットワーク帯域幅や遅延などの情報が扱われる[1]．このとき事前の対象ごとの注意深いモデル化が必要となるが，*Bazaar* では対象に関する情報は「定義情報」と呼ばれる部分のみに現れ，対象固有の情報にアクセスする汎用的なメソッドも提供されている．よって，基盤システムとしての導入後にアプリケーション要求に応じて発生する情報の種類の増加に対して頑健であると考える．

8. おわりに

本論文では，知的人工物を用いた知的空間の構築支援のためのミドルウェア（*Bazaar*）を提案した．*Bazaar* は位置検出システムにより検出された位置情報を中心に知的人工物の多様な情報を物理空間モデルとして管理し，様々な属性による検索とイベント通知を扱うための API を提供する．これにより，開発者はアプリケーションロジック中で使用する知的人工物の属性にのみ着目すればよく，その取得手段についての意識は不要になる．

またアプリケーション開発を通して，この API が知的人工物の移動や特定領域への入退出，使用状況の変化を用いた知的人工物ベースの空間構築に特有のプログラミングモデルを実現するのに適していることや属性の追加への柔軟性を確認した．さらに空間の機能や対象とする物理的な広さを容易に拡張し，知的人工物のコンセプトを補強可能であることを確認した．

最後に今後の課題をあげる．1 点目は単位空間の連携によるより広範囲な空間への柔軟な拡張機構である．これは 6.2 節であげた高負荷時の性能劣化への対策（負荷分散）であると同時に，ソフトウェアあるいはハードウェアの故障による影響を局所化する（危険分散）ことを目指している．1 つの候補として，携帯電話や PDA のようなパーソナルデバイス上でのいわば「my 物理空間モデル」により，ユーザ近傍やユーザが使用するアプリケーションで必要となるものだけを保持することを検討中である．2 点目は *Bazaar* を用いた開発のための統合開発環境である．ここでは物理空間モデル内の情報を活用したコード入力支援や試験，シミュレーション，デバッグといった広い意味での知的空間の開発支援に *Bazaar* を活用することを検討中である．

参 考 文 献

1) Addlesee, M., Curwen, R., Hodges, S., Newman, J., Ward, A. and Hopper, A.: Implementing a Sentient Computing System, *IEEE Computer Society*, pp.50–56 (2001).
2) Beigl, M., Gellersen, H.-W. and Schmidt, A.: MediaCups: Experience with Design and Use of Computer-Augmented Everyday Objects, *Computer Networks, Special Issue on Pervasive Computing*, Vol.35, No.4, pp.401–409 (2001).
3) Brumitt, B., Meyers, B., Krumm, J., Kern, A. and Shafer, S.: EasyLiving: Technologies for Intelligent Environments, *Proc. 2nd International Symposium on Handheld and Ubiquitous Computing* (*HUC2K*), pp.12–29 (2000).
4) Chen, H., Finin, T. and Joshi, A.: Using OWL in a Pervasive Computing Broker, *Proc. Workshop on Ontologies in Agent Systems* (*OAS2003*), pp.9–16 (2003).
5) EPCglobal Inc.: Web Site. http://www.epcglobalinc.org/
6) Fujinami, K., Kawsar, F. and Nakajima, T.: AwareMirror: A Personalized Display using a Mirror, *Proc. International Conference on Pervasive Computing, Pervasive2005*, LNCS 3468, pp.315–332 (2005).
7) Fujinami, K. and Nakajima, T.: Sentient Artefact: Acquiring User's Context Through Daily Objects, *Proc. 2nd International Symposium on Ubiquitous Intelligence and Smart Worlds* (*UISW2005*), LNCS 3823, pp.335–344 (2005).
8) Gellersen, H.-W., Kortuem, G., Beigl, M. and Schmidt, A.: Physical Prototyping with Smart-Its, *IEEE Pervasive Computing Magazine*, Vol.3, No.3, pp.74–82 (2004).
9) Harle, R.K. and Hopper, A.: Deploying and evaluating a location-aware system, *MobiSys*, pp.219–232 (2005).
10) HP Labs Semantic Web research group: Jena 2. http://jena.sourceforge.net
11) Kidd, C.D., Orr, R., Abowd, G.D., Atkeson, C.G., Essa, I.A., MacIntyre, B., Mynatt, E.D., Starner, T. and Newstetter, W.: The Aware Home: A Living Laboratory for Ubiquitous Computing Research, *Cooperative Buildings*, pp.191–198 (1999).
12) Krumm, J., Harris, S., Meyers, B., Brumitt, B., Hale, M. and Shafer, S.: Multi-camera Multi-person Tracking for EasyLiving, *Proc. 3rd IEEE Workshop on Visual Surveillance* (2000).
13) Lassila, O. and Swick, R.: Resource Description Framework (RDF) Model and Syntax Specification. http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/
14) Satoh, I.: A Location Model for Pervasive Computing Environments, *Proc. 3rd International Conference on Pervasive Computing and Communications* (*PerCom'05*), pp.215–224 (2005).
15) Seaborne, A.: RDQL — A Query Language for RDF. http://www.w3.org/Submission/2004/SUBM-RDQL-20040109/
16) Wang, X., Dong, J.S., Chin, C.Y. and Hettiarachchi, S.R.: Semantic Space: An Infrastructure for Smart Spaces, *IEEE Pervasive Computing*, Vol.3, No.3, pp.32–39 (2004).
17) Want, R., Hopper, A., Falcao, V. and Gibbons, J.: The Active Badge Location System, *ACM Trans. Information Systems*, Vol.10, No.1, pp.91–102 (1992).

(平成 18 年 1 月 26 日受付)
(平成 18 年 5 月 31 日採録)

藤波　香織（正会員）

平成 7 年早稲田大学大学院理工学研究科電気工学専攻修士課程修了．同年日本電信電話（株）入社．平成 9〜15 年 NTT コムウェア勤務．平成 17 年早稲田大学大学院理工学研究科情報科学専攻博士課程修了．博士（情報科学）．同年早稲田大学理工学術院客員講師（専任扱）．ユビキタスコンピューティングシステム，ヒューマン・コンピュータインタラクション等に興味を持つ．

中島　達夫（正会員）

平成 2 年慶應義塾大学大学院理工学研究科博士課程修了．平成 2〜4 年カーネギーメロン大学計算機科学科研究員．平成 5〜9 年北陸先端科学技術大学院大学情報科学研究科助教授を経て，現在，早稲田大学理工学術院コンピュータ・ネットワーク工学科教授．組み込みシステム，オペレーティングシステム，分散システム，ユビキタスコンピューティング，計算機ネットワーク，高信頼分散リアルタイムシステム等に興味を持つ．