解　説

ユビキタスコンピューティングにおけるアプリケーション開発手法に関する研究動向

鄭 顕志　中川 博之　川俣 洋次郎　吉岡 信和
深澤 良彰　本位田 真一

ユビキタスコンピューティングにおけるアプリケーションは，実世界の状態を把握し，ユーザに対してより積極的なサポートを行わなければならない．このようなアプリケーションを効率良く開発するためには，従来とは異なる新たなアプリケーション開発手法が必要となり，数多くの研究が行われている．本論文では，ユビキタスコンピューティングにおける，アプリケーション開発手法に関する研究動向を調査し，傾向と問題点を整理する．

In ubiquitous computing scenarios, applications pro-actively support user activities by changing their behavior according to their contexts which for example represent the physical environment. Realizing such applications inherently calls for new development methodologies. In this paper, we present a survey on application development in ubiquitous computing and discuss future research directions in this field.

1　はじめに

無線通信能力の向上，オープンネットワーク技術の発達，計算能力の向上，無線センサー技術の発達，バッテリ技術の改善，柔軟なソフトウェアアーキテクチャの出現など，M. Weiser が提唱したユビキタスコンピューティング[41] に必要な要素技術が急速に発展してきている．ユビキタスコンピューティングはパーベイシブコンピューティングとも呼ばれている[34]．本論文では以後ユビキタスコンピューティングという表現に統一する．文献[18] では，ユビキタスコンピューティングとは，埋め込まれたコンピュータが，センサー等を通じて自分が埋め込まれた環境から情報を得て，その情報をコンピューティングモデルの動的な構築のために利用できるような形態を表す，と定義されている．また，文献[34] では，ユーザがコンピュータを利用しているという意識の度合いをできる限り低減させることが重要であると指摘している．従って，ユビキタスコンピューティングにおけるアプリケーション[37] は，環境に関する情報やユーザの意図を把握し，それらに基づいて最適な動作を選択，実行できなければならない．現状では，このようなアプリケーションは既存の開発手法を用いて作り込まれているため生産性が乏しく，ユビキタスコンピューティングの発展を妨げる大きな要因となっている．ユビキタスコンピューティングにおけるアプリケーションを効率良く開発するためには，従来とは異なるアプリケーション開発手法が必要となる．本論文ではユビキ

A Survey of Application Development in Ubiquitous Computing.
Kenji Tei, 早稲田大学, Waseda University/国立情報学研究所, National Institute of Informatics.
Hiroyuki Nakagawa, 電気通信大学, The University of Electro-Communications.
Yojiro Kawamata, 国立情報学研究所, National Institute of Informatics/東京大学, the University of Tokyo.
Nobukazu Yoshioka, 国立情報学研究所, National Institute of Informatics.
Yoshiaki Fukazawa, 早稲田大学, Waseda University.
Shinichi Honiden, 国立情報学研究所, National Institute of Informatics/東京大学, the University of Tokyo.
コンピュータソフトウェア, Vol.25, No.4 (2008), pp.121–132.
[解説論文] 2007 年 11 月 30 日受付.

タスコンピューティングにおけるアプリケーション開発手法に関する研究動向を調査し，傾向をまとめ，今後の研究動向を示す．

一方で，環境の変化に対し適応的に動作し，ユーザの意図に基づいて最適な動作を行うソフトウェアの開発方法は，エージェント指向開発方法論として研究が進められている．本論文では，ユビキタスコンピューティングにおける開発方法論とエージェント指向開発方法論の比較を行う．

本論文の構成を次に示す．本章ではユビキタスコンピューティングの概要について述べた．2 章ではユビキタスコンピューティングにおけるアプリケーション開発の特徴を示し，研究動向を示す．3 章では，既存研究の問題点を指摘，今後の研究動向を議論，エージェント指向開発方法論との比較を行い，4 章でまとめを述べる．

2 ユビキタスコンピューティングにおけるアプリケーション開発

通常のアプリケーション開発では，アプリケーションの機能的側面や構造的側面に関して分析，設計を行うことでプログラミングモデルを作成し，プログラミングモデルに基づき実装を行う．一方，ユビキタスコンピューティングではコンテキストモデル，嗜好モデル，プログラミングモデルの 3 つのモデルを分析，設計することが重要となる [12]．ユビキタスコンピューティングでは，アプリケーションは環境の情報に基づいて動作を変更するため，アプリケーション自体の機能的側面や構造的側面だけでなく，コンテキストモデルと呼ばれる，アプリケーションが関連する環境に関するモデルを分析，設計する必要がある．また，ユーザの指示を最小限にしつつユーザの意図に沿った処理を行うために，ユーザの嗜好モデルを分析，設計する必要がある．さらに，コンテキストモデルと嗜好モデルに基づき，アプリケーションの振る舞いを記述するプログラミングモデルを分析，設計する必要がある．本章では，ユビキタスコンピューティングにおけるコンテキストモデル，嗜好モデル，プログラミングモデルについての研究動向を述べる．また，それらのモデルを統合した開発方法論についての研究動向を述べる．

2.1 コンテキストモデル

A. Dey らの定義によると，コンテキストとは，アプリケーションの振る舞いに関連すると思われる人，位置，物体などの状況を特徴付ける情報である [1]．ユビキタスコンピューティングにおけるアプリケーション開発では，そのアプリケーションがどのコンテキストに対して適応的に動作し，またどのような影響を与えるかを分析しなければならない．コンテキストの重要性が認識されるのに従い，コンテキストの分析結果を表現できるコンテキストモデリング言語が必要となり，現実世界の多様な情報を表現できるよう記述能力を向上させるため，数多くの言語が提案されてきた．コンテキストモデリング言語の代表的なアプローチとしては，Key-Value，オントロジーベース，Object Role Model(ORM) ベースなどが挙げられる．

Key-Value アプローチは，コンテキストをキーと値のペアで表現する最もシンプルなアプローチであり，初期のユビキタスコンピューティングミドルウェアで多く採用されてきた [33] [8] [14] [2] [20] [6]．Key-Value アプローチでは，コンテキストの種類をキー，そのコンテキストの現在の状況をキーに対応した値として表現する．例えば，コンテキスト情報は，“(温度, 35 度)”,“(場所, 会議室)” などのように表現する．ContextToolkit [33]，Solar [8]，EgoSpaces [14]，TOTA [20]，Plan B [2] などでは任意のキーを記述できる．これらのコンテキストモデリング言語では，アプリケーション間で利用可能な，キーや各キーの取りうる値についての共通語彙が必要となる．Key-Value アプローチでは表現がシンプルで実装が容易な反面，記述能力が乏しい．例えば，Tei という人物が建物 A の 12 階に位置することを表現する場合，Tei が持つ *location* というキーに対し，“12 階” という文字列を値として持たせることで表現する．しかし，Tei は建物 A 内にも位置しているが，コンテキストとしてはそれを表現できない．location キーに複数の値を持たせることで表現力を増すことはできるが，“建物 A” という値と “12 階” という値の関係は暗黙的であり，同時に成り立ってよい値なのか背反

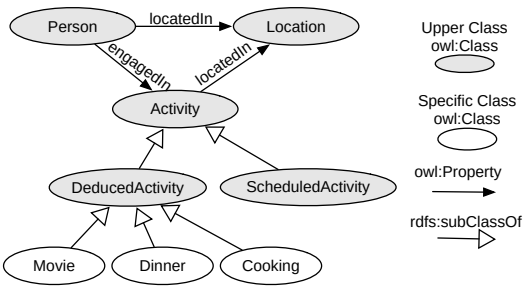

図 1 CONON の例

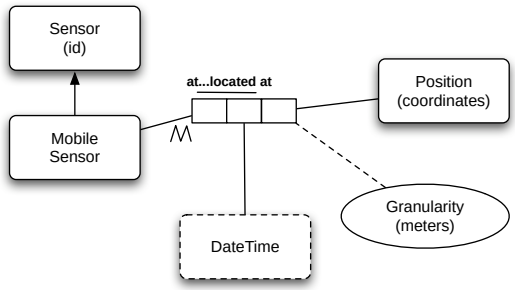

図 2 CML の例

する値なのかを明示的に表現できない．W4[6] では，使用するキーを限定することにより，既存の時間モデルや位置モデルを使用可能にしている．W4 では，Who，What，Where，When という 4 つのキーに限定し，位置を表す Where では PostGIS モデルを，時間を表す When では期間や日付表現に関しての一般的表現を用いている．しかし，キーと適用する既存のモデル間の関係は固定的であるため，予め定義されたキーでしかコンテキストを表現できない．このように Key-Value アプローチではコンテキスト間の関係をモデリングできないため，記述能力が乏しい．

コンテキスト間のモデリングを行うアプローチとしては，オントロジーベースアプローチがある．このアプローチでは，RDF，DAML+OIL，OWL などを用いてコンテキストをモデリングする．代表的なモデリング言語としては，CoOL[38]，SOUPA[9]，CONON[40] などがある．CoOL では，オントロジーを用いて，センサーなどから得られるデータなどの抽象度の低いコンテキスト表現から，計測方法などによらない抽象度の高い表現への変換を行っている．また，SOUPA や CONON では，位置モデルや時間モデル等のドメイン被依存の共通コンテキストをオントロジーを用いて定義し，ドメインに特化したコンテキストを定義する際の記述コストを低減している．図 1 では CONON の記述例を示している．灰色の楕円がドメイン被依存の共通コンテキスト，白色の楕円がドメインに特化したコンテキストを表している．人 (Person) が活動 (Activity) に従事し (engagedIn)，活動はある位置 (Location) で行われる (locatedIn)，活動には予定された活動 (ScheduledActivity) と推定される活動 (DeducedActivity) が定義され，ドメイン特化の DeducedActivity として Movie，Dinner，Cooking が定義されている．オントロジーベースアプローチは Key-Value アプローチと比べ，コンテキスト間の関係をモデル化できるため高い記述能力を持つが，RDF や OWL をベースとしているため，基本的に 2 項関係しかモデリングできない．従って，現実世界で多い n 項関係を表現する場合は独特のノウハウを用いて複数の 2 項関係に分割しなければならず，n 項関係を直接表現できない[24]．

n 項関係まで表現できるアプローチとしては，ORM に基づいた Context Modeling Language(CML)[11] がある．ORM では，オブジェクト，オブジェクト間の関係 (fact)，その関係においてオブジェクトが満たすべき役割 (role) を用いて対象をモデル化する．fact は複数のオブジェクト間の関連を記述できるため，ORM をベースにしている CML も n 項関係を記述できる．さらに CML では fact に対し可変性や生成方法，その fact の質を測るためのメトリクスなどの情報を記述することができ，コンテキスト間の多様な関係をモデリングすることができる．CML の記述例を図 2 に示す．図 2 の例では，MobileSensor がある時間 (DateTime) ある地点 (Position) に位置している関係を示している．MobileSensor，Sensor，Position はオブジェクトであり，DateTime は値を示している．また，MobileSensor は，id で識別される Sensor の一種であり，Position は座標で識別される．さらに，Position は，メートルを単位とする精度 (Granularity) で質が表される．

2.2 嗜好モデル

前述のように，ユビキタスコンピューティングでは，コンテキストとアプリケーションの動作を適切にマッピングする必要がある．しかし，あるコンテキストにおける適切な動作は，各ユーザの嗜好によって異なる．例えば，ユーザが動画を閲覧する際に，ユーザが持つ PDA で動画を表示させるか，ユーザの近くに位置する大型ディスプレイ上に表示させるかは，ユーザの嗜好によって異なる動作が選ばれる．また，同一のユーザであってもその嗜好は変化する．従って，アプリケーションから利用可能なように，ユーザの嗜好についても分析，設計を行い，モデル化する必要がある．ユーザの嗜好を記述するアプローチとしては，嗜好の記述対象によって，動作指定のアプローチとコンテキスト指定のアプローチに大別できる．

動作指定のアプローチでは，ユーザがあるコンテキストにおいて実行すべき動作を明示的に指定する．文献[25][36] では，Event-Condition-Action(ECA) モデルのポリシーを用いている．コンテキストを用いて Event や Condition を記述し，その条件下で実行すべき動作を記述する．

一方で，コンテキスト指定のアプローチでは，ユーザが希望するコンテキストに対する嗜好をモデリングする．Composite Capability/Preference Profiles(CC/PP)[16] は，使用する機器の画面サイズ，色数，CPU 速度，メモリ等のハードウェア情報や，OS，ブラウザ，処理可能データ等のソフトウェア情報に関して RDF を用いてユーザが使用したいデバイスに関する嗜好を記述する．しかし，CC/PP はコンテンツを視聴する用途に関する嗜好に特化しており，多様なコンテキストに対する嗜好は記述できない．文献[12] では，一階述語論理を用いてコンテキストの組み合せを抽象化し，それらに対して点数付けを行うことで嗜好を記述している．コンテキストの組み合せを用いることで，あるコンテキスト単体だけではなく，コンテキストの組み合せに対しての嗜好を記述することができる．また，0 から 1 までの点数付けや，義務 (obligation)，拒否 (veto)，無関心 (indifference)，未定義 (undefine) の特殊型を用いて，各嗜好に関する重要度が細かに指定できる．

2.3 プログラミングモデル

近年のユビキタスコンピューティングにおけるミドルウェアの成熟化に伴い，アプリケーション開発者に対して提供されるプログラミングモデルの研究が盛んになっている．ユビキタスコンピューティングにおけるアプリケーションはコンテキストに大きく依存するが，従来のプログラミングモデルではコンテキストの変化に対応するための記述をソースコードに直接書き込む必要がある．従って，このような環境においては従来のプログラミングモデルとは異なった，コンテキストの変化に対応する記述が分離，あるいは隠蔽されたプログラミング環境が期待される．本節では以降，ユビキタスコンピューティングへの適用を目的として提案されているプログラミングモデルを，コンテキスト変化への対応に関する能動性の観点から，イベント駆動型とノード発見・サービス記述型とに分類し，それぞれの動向を紹介する．

2.3.1 イベント駆動型

イベント駆動型のプログラミングモデルでは，コンテキスト変化に対し受動的に対応する．本モデルでは，コンテキストの変化をイベントで表現し，イベント通知によりアプリケーションの動作を変化させる．イベント駆動型のプログラミングモデルでは，対応すべきイベントの定義，取得方法に関して様々なモデルが提案されている．

TOTA[19][20] はタプルと呼ばれる情報の単位をノード間で伝播させることで環境情報やコンテキストの変化を周囲へ知らせ，アプリケーション側が受信したタプルを参照することでコンテキストの変化に対応するスタイルのミドルウェアである．TOTA において開発者は，伝播させるタプルとタプルを受け取るエージェントを実装することで，アプリケーションを構築する．しかし TOTA では補助的なサブクラスが提供されてはいるものの，依然としてタプルに対して通信経路や伝播条件など低レイヤのプログラム記述が要求されることから，開発者の負担が軽減されているとは言い難い．

Gaia[31] は Illinois 大学で研究開発されているユビキタスコンピューティングのためのミドルウェアであり，ユーザやデバイス，オブジェクト，位置，サービ

スなどの物理的制約を持った個体の集合をアクティブスペースと呼ばれる仮想空間上に定義し，それらのインタラクションによるアプリケーション構築環境を提供している．Gaia に関してもイベント駆動型プログラミングモデルとして，ポリシー記述によりアプリケーション動作を管理するプログラミングモデル[35] が提案されている．このモデルの特徴は，通常イベント，条件，動作 (Event-Condition-Action：ECA) として記述されるルールに，事前・事後条件を加えた，ECPAP(Event-Condition-Precondition-Action-Postcondition) 形式のルールを記述モデルとして用いることで，同一イベントに対して適用されるポリシールールの実行順序を制御するところにある．

イベント駆動型に分類されるプログラミングモデルとしてはその他，Java 言語をコンテキスト依存のアプリケーションを容易に開発できるよう拡張した JCAF(Java Context-Awareness Framework)[3] や，アプリケーション記述にコンテキストだけでなくユーザの嗜好に応じた意思決定の観点を導入した[12] など多くのモデルが存在する．

2.3.2 ノード発見・サービス記述型

ノード発見・サービス記述型のプログラミングモデルでは，プログラムの実行に適した条件を記述し，条件に合致した実行環境を発見，実行することで，コンテキストの変化に対し能動的に対応する．

本アプローチとしては，まず Spatial Programming[4] に代表されるノード発見型のプログラミングモデルが挙げられる．Spatial Programming では，開発者は Smart Message と呼ばれるモバイルエージェントに類似した移動可能なクラスを拡張し，関心のあるサービス (スペース) を参照するタグを条件として記述する．実装された Smart Message は該当するタグを持つノードへ移動し，タグにより表現された資源へアクセスすることで，アプリケーションの機能が実現されることから，物理的なネットワーク構成を隠蔽したプログラムが可能となる．

また，Gaia をミドルウェアとするノード発見型プログラミングモデルに Olympus[28] がある．Olympus は Virtual Entity と呼ばれる抽象オブジェクトを導入することで，実行時に制約に合致するエンティティを動的に割り当て，結果として抽象度の高いプログラム記述を可能としている． 図 3 に Olympus によるプログラム記述例を示す．この例はユーザ Sal に対するスライド表示サービスの提供を実現するものであるが，まず，空間の指定に as1 という変数を宣言し，Sal が存在するといった as1 の制約を記述することで具体的な場所と as1 とを対応付けている (1〜3 行目)．ここで as1 は Virtual Entity に該当する．Olympus では Virtual Entity に対して制約を記述することで対象を指定することが可能であり，実行時に具体的な対象が決定される．同様に 5〜8 行目で，先に指定した空間 as1 内でスライド表示に最も適したアプリケーションを Virtual Entity である app1 により特定するが，Virtual Entity を利用することで，実行時に指定するアプリケーション (スライド表示ソフトや PDF 閲覧ソフト，画像ビューアなど) の差異が吸収されている．このように Olympus では，サービスやアプリケーション，デバイス，場所，ユーザなどのエンティティに対して Virtual Entity という抽象的な概念を導入し，制約を記述することによる宣言的な指定方法を提供することで，プログラムとコンテキストとの依存関係を低減している．

```
1:  ActiveSpace as1;
2:  as1.hasProp("containsPerson",  "Sal");
3:  as1.instantiate();
4:
5:  Application app1;
6:  app1.hasProp("class",  "Slideshow");
7:  app1.hasProp("file",  "Olympus.ppt");
8:  app1.start(as1);
```

図 3 スライド表示プログラムの記述例 (Olympus[28])

さらに，STS Framework[22] では抽象位置という概念を導入することにより，適応動作の条件を記述することでアプリケーションの動作に望ましい時間・位置空間の指定を可能としている．具体的には，アスペクト指向プログラミング[15] における Pointcut[39] の概念を利用することで，タスクの各実行段階で位置に対する条件に適合するかを横断的にチェックし，適応動作を起動すべき状態を同定している．これにより，適応動作の処理プログラムの分散化が避けられ，結果としてソフトウェアの動的配置に関する開発者の

負荷を軽減している．

一方，近年のサービス指向アーキテクチャの普及に伴い，ユビキタスコンピューティングにおけるサービス発見・統合のためのサービス記述としてのプログラミングモデルも新たな傾向となりつつある．CA-SOA(Context Aware Service Oriented Architecture)[10] では，Web Services 記述言語である OWL-S [21] 上にコンテキストに応じたサービスと要求内容の記述を加えることで，コンテキストに適応したサービス発見手法を目指している．また，Service Ecosystem[27] は，異種デバイス・異種サービスが存在する環境下での，知識ベースを利用した動的サービス統合モデルを提唱している．Service Ecosystem では，実現デバイスやアプリケーションを意識しない XML 形式のサービス要求メッセージを記述するだけで，軽量エージェントとも言える Enzyme(酵素)[†1] が，所有する知識の範囲内で動的にサービスを発見，統合する．

ただし，CA-SOA，Service Ecosystem に代表されるサービス記述型プログラミングモデルの多くは，ミドルウェアの成熟に先んじて提案されたものが多く，サービス要求の適切な粒度への分解，サービスのプランニングなど，実現性の観点からまだまだ解決しなければならない課題が多い．

2.4 開発方法論

ユビキタスコンピューティングにおけるアプリケーション開発に関する研究は，2.1 節から 2.3 節で示したように，コンテキスト，嗜好，プログラミングの各モデリング手法に関する研究が主であった．これらのモデリング手法の成熟に従い，2006 年あたりからユビキタスコンピューティングにおける開発方法論について研究がなされ始めてきた．しかし，包括的な開発方法論を提案する研究は未だ数少なく，既存の開発プロセスをユビキタスコンピューティングに対し試験的に適用するのみに留まっている．文献 [23] [26] では，モデル駆動開発をユビキタスコンピューティングにおけるアプリケーション開発に適用し，有効性を議論している．しかし，これらの研究は，既存の開発方法論をユビキタスコンピューティングに単純に適用しており，コンテキストモデルや嗜好モデルといったユビキタスコンピューティング特有のモデリング要素を考慮していない．

コンテキストモデルや嗜好モデルを考慮した開発方法論を提案している研究としては文献 [12] がある．従来のオブジェクト指向開発方法論では分析，設計，実装，テストの各開発フェーズにおいてアプリケーションの動作を段階的に詳細化していくが，文献 [12] では，これらの各開発フェーズにおいてコンテキストモデルや嗜好モデルを扱うように拡張している．分析フェーズでは，従来のようなアプリケーション動作についての分析だけではなく，アプリケーションが関連するコンテキスト，アプリケーション動作とコンテキストの依存関係，想定される幾つかのユーザ像から得られる嗜好例を分析する．設計，実装フェーズでは，分析フェーズで抽出された，アプリケーション動作の要求，動作とコンテキストの依存関係，嗜好例からプログラムを設計，実装する．また，インフラカスタマイズフェーズという新たなフェーズを導入し，分析フェーズで分析されたコンテキストに対して，そのコンテキストの取得方法や，コンテキスト取得に必要な実行環境を分析する．また，テストフェーズに備え，アプリケーション実行に関連するコンテキストと嗜好例を生成する実行環境例を構築する．テストフェーズでは，実装フェーズで作成されたプログラムを，インフラカスタマイズフェーズで構築された実行環境例上で実行させ，デバッグを行う．さらに，テストフェーズでは，実際の実行環境上においてテストユーザを用いて実地テストを行い，デバッグを行う．

3 議論

本章では，2 章で示したコンテキストモデル，嗜好モデル，プログラミングモデル，開発方法論に関して，今後の研究動向について議論する．また，ユビキタスコンピューティングにおける開発方法論とエージェント指向開発方法論との関連を議論する．

†1 サービスを統合する触媒としての役割を果たすことから，Ecosystem(生態系) とも対比して，このようなメタファーが用いられている．

3.1 コンテキストモデルの研究動向

2.1 節で示した各コンテキストモデリングアプローチの比較を表 1 に示す．Key-Value アプローチシンプルなデータ構造のためコンテキストのデータ管理は容易である．また，そのシンプルなデータ構造のため，現状 Key-Value アプローチに対応したミドルウェアは数多い．しかし，記述能力は低いため，実世界の多様な事象をモデル化することが困難である．オントロジーベースアプローチは RDF や OWL に基づきサブクラス関係などの多彩な関係が記述でき，高い記述能力を備えている．また，オントロジー管理機構や記述ツール等の整備も進められており，導入の敷居は低くなってきている．ORM アプローチは 3 項関係まで記述可能な高い記述能力を有している．また，ORM はデータベースの分野で発展を遂げた言語であるため，記述ツールや関係データベースとの連携ツールも整備されている．しかし，未だ ORM アプローチに対応したミドルウェアは数少ない．

2.1 節で示したように，コンテキストモデリング言語は，実世界の記述対象を表現可能な高い記述能力が必要とされている．記述能力は，コンテキスト単体の表現から，コンテキスト間の 2 項関係の表現，さらには CML のような n 項関係まで表現できる高い記述能力を持った言語が提案されてきており，コンテキストの記述能力に関しては成熟してきている．プログラムモデルや嗜好モデルはコンテキストモデルに基づいて作成されるため．コンテキストモデルはユビキタスコンピューティングのアプリケーション開発において重要な役割を占めている．今後は，これらの高い記述能力を持ったコンテキストモデリング言語を前提として，プログラミングモデルや嗜好モデルの発展が起こるものと予想される．

コンテキストモデル自体は，記述時には高い抽象度での記述能力が求められるが，運用時には管理しやすい形式である必要がある．例えば，コンテキスト自体もデータであるため，多量のコンテキストを管理するにはデータ管理機構が必要となる．従って，現在主流である関係データベースで管理可能な形式にコンテキストを変換することが求められる．このように，コンテキストモデラー，コンテキストを利用するアプリケーション，コンテキスト生成，管理機構ごとに適した表現形式に変換可能でなければならず，モデル変換に関する研究が必要となる．CML では，初期の CML はグラフィック表現のみをサポートしていたが，CML から関係データベースへのマッピングルール [12] や，XML ベースの実行時表現 [30] など様々な形式への変換に関する研究がすすめられている．だが，センサーの計測値やプロファイルからコンテキストへの変換や，その際にセンサーの計測方法によって生じるコンテキストの質の変化に関する部分はまだ十分に議論されていない．今後は無線センサーネットワーク内でのデータ表現，計測方法とコンテキストモデル間での関係整理，変換に関する研究が行われるものと思われる．

表 1 コンテキストモデルの比較

モデリングアプローチ	記述能力	管理容易性
Key-Value	Poor	Good
オントロジー	Normal	Normal
ORM	Good	Normal

また，移動ロボットの分野でも世界モデル (World Model) [7] という実世界を表すモデルのモデリング手法に関する研究が進められている．これらのモデリング手法は実世界の物理的な側面に特化し，非常に高いモデリング能力を備えている．さらに，複数のロボット間で世界モデルを共有する手法についても研究が進められている [32]．世界モデルのモデリング手法と比較し，コンテキストモデルは実世界の事象の論理的な関係まで記述するよう設計されている．今後は，コンテキストモデルにも，世界モデルで培われた物理的側面の記述能力を取り入れていくことが期待される．

3.2 嗜好モデルの研究動向

2.2 節で示した嗜好モデリングアプローチの比較を表 2 に示す．嗜好モデルは動作指定とコンテキスト指定の 2 つのアプローチに大別できる．

動作指定のアプローチでは，ユーザが行ってほしい動作まで指定するため，詳細な指定が可能となる．従って，ユーザの意図を正確に反映することが可能となる．しかし，ユーザが，アプリケーションが実行可

表 2　嗜好モデルの比較

モデリングアプローチ	ユーザの負担	正確さ
動作指定	Much	Good
コンテキスト指定	Little	Poor

表 3　プログラミングモデルの比較

モデリングアプローチ	全体のフロー記述	変化への対応記述
イベント駆動	Poor	Good
ノード発見・サービス記述	Good	Poor

能な動作や，その動作が環境に対して与える影響を把握しなければならない．一方でコンテキスト指定のアプローチでは，ユーザは希望する環境の状態をコンテキストとして記述する．1 章で述べたように，ユビキタスコンピューティングではユーザからの指示を最小限にとどめなければならない．従って，嗜好モデルをユーザに指定させる場合，ユーザが細かなアプリケーション動作を意識しなくとも記述可能であるコンテキスト指定アプローチの方が適している．今後は，コンテキスト指定の嗜好モデリング言語が主流になると思われる．現状では，一階述語論理等を用いて，嗜好の記述対象であるコンテキストを抽象化しているが，コンテキストは時間と共に変化するため，時相論理などを用いた抽象化が必要になると思われる．また，一階述語論理や時相論理に基づいて一般のユーザが嗜好を記述することは困難であるため，よりユーザが直感的に指定できる嗜好記述モデルと，嗜好記述モデルと内部嗜好モデルとの変換ルールを整備する必要がある．

コンテキスト指定アプローチの場合，アプリケーションが指定されたコンテキストに到達するための動作を選択，実行する必要がある．そのためには，2.3.1 節で示した ECPAP のように，アプリケーション動作がコンテキストに与える影響を把握できなければならない．しかし，現状ではアプリケーション動作がコンテキストに与える影響に関して十分に議論がなされていない．今後は，動作とコンテキスト間の関係を分析する手法やその表現形式が必要となる．

3.3　プログラミングモデルの研究動向

プログラミングモデルに関して，2.3 節ではコンテキスト変化への対応に関する能動性の観点から，イベント駆動型とノード発見・サービス記述型に分類して紹介した．これらのアプローチ間の比較を表 3 に示す．プログラミングモデルでは，現在のコンテキストを考慮した上で，ユーザの意図する状態に達するための処理フローの記述容易性や，コンテキストやユーザの嗜好変化に対応する処理の記述容易性が問題となる．

イベント駆動型では，コンテキストやユーザの嗜好変化をイベントとして記述できるため，変化への対応記述に優れている．一方で，全体のフローを記述する場合は，アクションが環境に与える影響や，各イベントの条件を考慮して適切な個々のイベント対応処理を記述する必要がある．この場合，処理間の実行順序が見通しにくくなってしまう．

ノード発見・サービス記述型では，処理の流れをシーケンシャルに記述できるため，全体のフローを明示的に記述することができる．一方で，割り込み処理の記述が煩雑となり，変化への対応を記述する負荷が高くなってしまう．

実利用シーンを想定すると，例えばイベントによりユーザの移動を検知して，ユーザの現在の状況に相応しいサービスを発見・提供するなど，それぞれのモデルは相互補完の関係にある．従って，両モデルの成熟後は統合フレームワークを構築する際の問題点へと研究対象が移るものと思われる．実世界との連動に古くから取り組んでいるモバイルロボットの分野においても，サービス記述型とイベント駆動型の統合が検討されている．例えば，Object Management Group(OMG) で国際標準化が検討されているロボット構築ミドルウェアである RT ミドルウェアはビジネスロジック (サービス記述型) とイベント駆動型双方の利用が検討されている [17]．ユビキタスコンピューティングの分野でも，現時点で既にそれぞれの分類に対応するプログラミングモデルが Gaia 上で提案されていることからも，この流れは今後加速するものと考えられる．

しかし，プログラミングモデルから嗜好モデルへの参照については，現在，文献 [12] など一部の研究でしか言及されていない．ユーザの嗜好を考慮すること

で，複数の提供可能なサービスの中からユーザに最適なサービスを選択することが可能になるが，プログラミングモデルの観点からは，現在のコンテキスト，ユーザ嗜好によって期待される将来のコンテキスト，さらにそれらを結びつけるアクションとの関連を記述するための表現法を検討する必要があろう．

3.4 モデリングアプローチの選択

本論文では，コンテキスト，嗜好，プログラミングモデルの研究動向について論じてきた．実際にアプリケーションを構築する場合は，各モデルにおいて適切なモデリングアプローチを選択する必要がある．本節では，ユビキタスコンピューティングアプリケーションの種類ごとに各モデルにおいてどのモデリングアプローチが適切なのかを議論する．

文献[37] で示されているような初期のユビキタスコンピューティングアプリケーションは単一のアプリケーションのみが動作する閉じた環境を想定して作られていた．そのため，アプリケーションの規模や変化の度合い，他のアプリケーションの連携は考えられていなかった．従って，コンテキストモデルでは管理の容易さが，プログラミングモデルでは環境変化への対応が重視され，ユーザの嗜好への対応はあまり検討されていなかった．例えば，初期のユビキタスコンピューティングミドルウェアである TOTA では，嗜好モデルは導入されておらず，コンテキストモデルにはタプルベースの Key-Value を，プログラミングモデルにはタプルの変化に対するイベント駆動型を採用している．

複数のアプリケーションが共存するオープン環境での動作を考えた場合，アプリケーション規模の増大やアプリケーション間連携が重要となってくる．従ってコンテキストモデルでは，コンテキストを明確にモデル化し，他のアプリケーション間でのコンテキストの共有を可能にするために記述能力が，プログラミングモデルでは環境変化への対応に加え全体のフローを明示的に記述できる能力が求められてくる．例えば，Gaia ではコンテキストモデルにはオントロジー，プログラミングモデルにはイベント駆動とノード発見・サービス記述型を採用している．

さらに実運用を考えた場合，エンドユーザが意識しなくとも利用可能であり，意識する必然性がある場合でも指示の量を最小限に抑える必要がある†2．従って，今後は，記述能力の高いオントロジーベースや ORM ベースのコンテキストモデル，イベント駆動とノード発見・サービス記述型を連携させたプログラミングモデルに加え，ユーザの指示を最小限に抑えるコンテキスト指定の嗜好モデルを組み合わせることが重要となってくるものと考えられる．

3.5 開発方法論の研究動向とエージェント指向開発方法論との比較

コンテキスト，嗜好，プログラミングモデルを扱う要素技術が確立するに伴い文献[12] のように，コンテキスト，嗜好，プログラミングモデルを扱う開発方法論に関する研究が注目されていくと思われる．文献[12] では，開発方法論全体の概要についてしか示しておらず，各フェーズにおける分析内容や成果物に関しては厳密に定義されていない．今後は，各フェーズごとの分析内容や，連携についての研究が進むものと思われる．

分析フェーズについてはユーザの要求とドメイン情報からいかにコンテキストと嗜好を過不足なく抽出するかが焦点となる．また各モデルをどのような順番でモデリングするかなど，議論の余地は多い．設計フェーズに関しては，現段階ではアプリケーション実装に関する経験的な知識が少ないことから，デザインパターンのような経験に立脚したボトムアップな設計技術よりプログラミングモデルの理論に基づいたトップダウンな設計技術が注目されると思われる．実装フェーズについては 3.3 節で示したようなプログラミングモデルに基づき，設計からプログラムに落とし込む．実装の負担を低減するためには，設計からプログラムのひな形を自動生成する技術の適用が期待される．テストフェーズに関しては，コンテキストや嗜好など外的要因が介入することでテスト段階での動作保証が重要性を増す．よってテストの自動化やテストケースの自動生成による網羅的かつ効率的なテス

†2 文献[34] ではこの性質を Invisibility と呼んでいる．

ト技術だけでなく，モデル検証など定性的な動作保証技術が注目されると思われる．保守フェーズについては，コンテキストや嗜好などへの対応によるソフトウェアの複雑さの増大に伴い，ソフトウェアの改善点を効率的に抽出する技術が必要となると考えられる．

本節や 2.4 節では，PerCom 等のユビキタスコンピューティングコミュニティで議論されている開発方法論について述べたが，コンテキストや嗜好を明示的に扱う開発方法論としては，AAMAS 等のエージェントコミュニティで議論されているエージェント指向開発方法論がある．Gaia [43]†3，ROADMAP [13]，Tropos [5] といった代表的なエージェント指向開発方法論では，初期の開発フェーズからコンテキストや嗜好を明示的に分析し，コンテキスト変化やユーザの嗜好に対応したエージェントを開発する．Gaia や ROADMAP では，分析フェーズにおいてエージェントの実行環境に関するコンテキスト (Gaia や ROADMAP では Enviroment と呼ばれる) を分析し，分析されたコンテキストに基づいて各エージェントのロールや，ロール間のインタラクションを分析，設計する．Tropos では，分析フェーズにおいてゴール指向要求分析手法 i* [42] を用いており，アプリケーション動作に関連するコンテキストやユーザを系統的に分析することができる．さらに Tropos では，エージェントの実装アーキテクチャとして BDI アーキテクチャ [29] を採用している．BDI アーキテクチャでは，エージェントが認識している知識を Belief，達成すべき状態を Desire，現在実行中の目標を Intention として表現し，エージェントは Belief の変化に対応して Desire の中から達成すべき Intention を決定し，その Intension を達成するアクションを推論，実行する．環境変化に柔軟に対応するためにエージェントの動作は本質的に複雑になるが，BDI アーキテクチャでは人間のメタファからこの複雑性を低減するプログラミングモデルを備えている．また，コンテキストを知識で表現することにより，コンテキスト指定アプローチの嗜好モデルへの親和性も高い．

しかしながら，これまでのエージェント指向開発方法論では，コンテキストの分析，設計に関しては十分議論されていなかった．Gaia や ROADMAP では，コンテキストのモデリング言語や分析方法を規定していない．また，Tropos では i*の記述要素を用いてコンテキストを表現しているが，CML やオントロジーベースのコンテキストモデリング言語と比較して，その記述能力は乏しい．さらに，コンテキストの取得方法や実行環境についても扱っておらず，実環境との関連が特徴となるユビキタスコンピューティングにおける開発方法論としては不十分である．

このようにエージェントコミュニティで議論されている開発方法論では，プログラミングモデルや嗜好モデルの扱いを得意とし，ユビキタスコンピューティングコミュニティで議論されている開発方法論では，コンテキストの扱いを得意としている．今後は，これらの方法論を組み合わせることにより，ユビキタスコンピューティングに適した開発方法論が提案されるものと思われる．

4 まとめ

本論文ではユビキタスコンピューティングにおけるアプリケーション開発手法のサーベイを示し，今後の研究動向について論じた．環境の状態やユーザの意図によってアプリケーションの動作を適切に変更しなければならないユビキタスコンピューティングでは，プログラムモデルに加え，環境の状態を表すコンテキストモデルとユーザの意図を表す嗜好モデルの分析，設計が重要となる．現状では，プログラミングモデルは既存のアプローチを拡張する形で各モデルが提案され，そこから得られた知見に基づいて，コンテキスト，嗜好モデルの記述能力を向上させるよう研究が進められている．特にコンテキストモデルに関しては記述能力は成熟してきており，今後は，高い記述能力をもったコンテキストモデル，嗜好モデルに基づいてプログラミングモデルを改良するようになると予想される．また，このようなアプリケーションを開発する場合，従来のようなアプリケーションの機能的側面に基づいた開発方法論では対応しきれない．コンテキストモデルや嗜好モデルを加えた分析，設計，実装，

†3 2.3.1 節で示した Gaia と同名であるが，関連性はない．

保守に関する新しい方法論が必要となる．

参 考 文 献

[1] Abowd, G. D., et al.: Towards a Better Understanding of Context and Context-Awareness, in *Proc. of the 1st International Symposium on Handheld and Ubiquitous Computing* (*HUC*), Springer-Verlag, 1999, pp. 304–307.

[2] Ballesteros, F. J., et al.: Plan B: An Operating System for Ubiquitous Computing Environments, in *Proc. of the Fourth Annual IEEE International Conference on Pervasive Computing and Communications* (*PerCom*), IEEE Computer Society, 2006, pp. 126–135.

[3] Bardram, J. E.: The Java Context Awareness Framework (JCAF) - A Service Infrastructure and Programming Framework for Context-Aware Applications, *Pervasive Computing*, Vol. 3468(2005), pp. 98–115.

[4] Borcea, C., et al.: Spatial Programming Using Smart Messages: Design and Implementation, in *Proc. of the 24th International Conference on Distributed Computing Systems* (*ICDCS*), IEEE Computer Society, 2004, pp. 690–699.

[5] Bresciani, P., et al.: Tropos: An Agent-Oriented Software Development Methodology, *Autonomous Agents and Multi-Agent Systems*, Vol. 8, No. 3(2004), pp. 203–236.

[6] Castelli, G., et al.: A Simple Model and Infrastructure for Context-Aware Browsing of the World, in *Proc. of the Fifth IEEE International Conference on Pervasive Computing and Communications* (*PerCom*), IEEE Computer Society, 2007, pp. 229–238.

[7] Chatila, R. and Laumond, J. P.: Position referencing and consistent world modeling for mobile robots, in *Proc. of International Conference on Robotics and Automation* (*ICRA*), 1985, pp. 138–145.

[8] Chen, G., et al.: Design and Implementation of a Large-Scale Context Fusion Network, in *Proc. of the First Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services* (*MobiQuitous*), 2004, pp. 246–255.

[9] Chen, H., et al.: SOUPA: Standard Ontology for Ubiquitous and Pervasive Applications, in *Proc. of the First Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services* (*MobiQuitous*), 2004, pp. 258–267.

[10] Chen, I. Y. L., et al.: Ubiquitous Provision of Context Aware Web Services, in *Proc. of the IEEE International Conference on Services Computing* (*SCC*), Washington, DC, USA, IEEE Computer Society, 2006, pp. 60–68.

[11] Henricksen, K., et al.: Modelling Context Information with ORM, in *OTM Workshops*, Lecture Notes in Computer Science, Vol. 3762, Springer, 2005, pp. 626–635.

[12] Henricksen, K. and Indulska, J.: Developing Context-Aware Pervasive Computing Applications: Models and Approach, *Journal of Pervasive and Mobile Computing*, Vol. 2, No. 1(2006), pp. 37–64.

[13] Juan, T., et al.: ROADMAP: Extending the Gaia Methodology for Complex Open Systems, in *Proc. of the 1st International Joint Conference on Autonomous Agents and Multiagent Systems* (*AAMAS*), ACM, 2002, pp. 3–10.

[14] Julien, C. and Roman, G.-C.: EgoSpaces: Facilitating Rapid Development of Context-Aware Mobile Applications, *IEEE Transactions on Software Engineering*, Vol. 32, No. 5(2006), pp. 281–298.

[15] Kiczales, G., et al.: Aspect-Oriented Programming, in *Proc. of European Conference on Object-Oriented Programmin* (*ECOOP*), 1997, pp. 220–242.

[16] Kiss, C.: Composite Capability/Preference Profiles (CC/PP): Structure and Vocabularies 2.0, Technical report, W3C Working Draft, 2007.

[17] Kotoku, T., et al.: Robot Middleware and its Standardization in OMG, in *Proc. of International Conference on Ubiqutous Robots and Ambient Intelligence* (*URAmI*), 2005, pp. 116–119.

[18] Lyytinen, K. and Yoo, Y.: Introduction, *Communication of the ACM*, Vol. 45, No. 12(2002), pp. 62–65.

[19] Mamei, M. and Zambonelli, F.: Programming Stigmergic Coordination with the TOTA Middleware, in *Proc. of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems* (*AAMAS*), ACM Press, 2005, pp. 415–422.

[20] Mamei, M. and Zambonelli, F.: Programming Pervasive and Mobile Computing Applications with the TOTA Middleware, in *Proc. of the Second IEEE International Conference on Pervasive Computing and Communications* (*PerCom*), IEEE Computer Society, 2004, pp. 263–273.

[21] Martin, D., et al.: OWL-S, http://www.daml.org/services/owl-s/.

[22] Matsuzaki, K.: *On the Ease of Deployment of Applications for Ambient Environments*, PhD Thesis, The University of Tokyo, 2007.

[23] Munoz, J., et al.: Software Engineering for Pervasive Systems. Applying Models, Frameworks and Transformations, in *Proc. of International Workshops on Software Engineering of Pervasive Services*, 2006.

[24] Noy, N. and Rector, A.: Defining N-ary Relations on the Semantic Web: Use With Individuals, Technical report, W3C Working Draft, 2005.

[25] Patwardhan, A., et al.: Enforcing Policies in Pervasive Environments, in *Proc. of the First Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services* (*MobiQuitous*), 2004, pp. 299–308.

[26] Pham, H. N., et al.: Applying Model-Driven

Development to Pervasive System Engineering, in *Proc. of the 1st International Workshop on Software Engineering for Pervasive Computing Applications, Systems, and Environments*, IEEE Computer Society, 2007. p.7.

[27] Quitadamo, R., et al.: The Service Ecosystem: Dynamic Self-Aggregation of Pervasive Communication Services, in *Proc. of the 1st International Workshop on Software Engineering for Pervasive Computing Applications, Systems, and Environments*, IEEE Computer Society, 2007.

[28] Ranganathan, A., et al.: Olympus: A High-Level Programming Model for Pervasive Computing Environments, in *Proc. of the Third IEEE International Conference on Pervasive Computing and Communications* (*PerCom*), IEEE Computer Society, 2005, pp. 7–16.

[29] Rao, A. S. and Georgeff, M. P.: Modeling rational agents with a BDI-architecture, *Readings in agents*, Morgan Kaufmann Publishers Inc., 1998, pp. 317–328.

[30] Robinson, R., et al.: XCML: A Runtime Representation for the Context Modelling Language, in *Proc. of the Fifth IEEE International Conference on Pervasive Computing and Communications Workshops*, IEEE Computer Society, 2007, pp. 20–26.

[31] Román, M., et al.: A Middleware Infrastructure for Active Spaces, *IEEE Pervasive Computing*, Vol. 1, No. 4(2002), pp. 74–83.

[32] Roth, M., et al.: A Real-time World Model for Multi-Robot Teams with High-Latency Communication, in *Proc. of International Conference on Intelligent Robots and Systems* (*IROS*), 2003, pp. 2494–2499.

[33] Salber, D., et al.: The Context Toolkit: Aiding the Development of Context-Enabled Applications, in *Proc. of ACM SIGCHI Conference on Human Factors in Computing Systems* (*CHI*), Pittsburgh, 1999, pp. 434–441.

[34] Satyanarayanan, M.: Pervasive Computing: Vision and Challenges, *IEEE Personal Communications*, Vol. 8, No. 4(2001), pp. 10–17.

[35] Shankar, C. and Campbell, R. H.: Ordering Management Actions in Pervasive Systems using Specification-enhanced Policies, in *Proc. of the Fourth Annual IEEE International Conference on Pervasive Computing and Communications* (*PerCom*), IEEE Computer Society, 2006, pp. 234–238.

[36] Shankar, C., et al.: An ECA-P Policy-based Framework for Managing Ubiquitous Computing Environments, in *Proc. of the Second Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services* (*MobiQuitous*), IEEE Computer Society, 2005, pp. 33–44.

[37] Sommer, C., et al.: Tomorrow's Applications in Ubiquitous Computing: a Survey, 合同エージェントワークショップ&シンポジウム (JAWS), 2007.

[38] Strang, T., et al.: CoOL: A Context Ontology Language to enable Contextual Interoperability, in *Proc. of 4th International Conference on Distributed Applications and Interoperable Systems* (*DAIS*), Lecture Notes in Computer Science (LNCS), Vol. 2893, Springer Verlag, November 2003, pp. 236–247.

[39] Wand, M., et al.: A semantics for advice and dynamic join points in aspect-oriented programming, *ACM Transactions of Programming Language Systems*, Vol. 26, No. 5(2004), pp. 890–910.

[40] Wang, X. H., et al.: Ontology Based Context Modeling and Reasoning using OWL, in *Proc. of the Second IEEE Annual Conference on Pervasive Computing and Communications Workshops*, IEEE Computer Society, 2004, pp. 18–22.

[41] Weiser, M.: Some Computer Science Issues in Ubiquitous Computing, *Communication of the ACM*, Vol. 36, No. 7(1993), pp. 75–84.

[42] Yu, E. S. K.: Towards Modeling and Reasoning Support for Early-Phase Requirements Engineering, in *Proc. of the 3rd IEEE International Symposium on Requirements Engineering* (*RE*), IEEE Computer Society, 1997, p. 226.

[43] Zambonelli, F., et al.: Developing Multiagent Systems: The Gaia Methodology, *ACM Transactions on Software Engineering Methodology*, Vol. 12, No. 3(2003), pp. 317–370.