Location Scope：ロケーションアウェアソフトウェア開発支援手法の提案

松崎和賢[†] 吉岡信和[††] 本位田真一[†,††]

ユビキタス環境において，ユーザの携帯端末上で動作するアプリケーションには，ユーザから空間的に近くに存在する周辺機器類により提供されるサービスを利用するというユースケースが考えられる．この機能を実現するために，ユーザとデバイスの間の位置関係などの条件を解析し，適切な対処をすることが必要となる．こうした位置関係の条件をアプリケーション開発時に扱う場合，アプリケーションを運用する単一の環境に特化して作成してしまう傾向にある．そのため違う機能の提供されているビルなどで動作させるためのポータビリティに欠けるという問題があり，再開発のコストがかかってしまう．本論文では，周辺機器類をそれらの提供する機能の種類に基づいて抽象化したうえで，アプリケーションやユーザとの位置関係トポロジに基づいたモデル化を行う．このモデルを用いて表現された位置条件に関する処理を，アスペクト指向開発に適用して実行時に結合する方式を適用することでポータビリティの実現を図る．関心事の組合せのために位置情報を用いるために Location Scope という概念を提案する．実際に Location Scope を実現する支援系を設計・実装しその評価を行った．

Location Scope: Development Support for Location-awareness Software

Kazutaka Matsuzaki,[†] Nobukazu Yoshioka[††] and Shinichi Honiden[†,††]

An application that is working in ubiquitous environment needs to satisfy some use cases about using services provided by service appliances considering distances among user and services. In order to achieve this functionality, analyses and reactions for location conditions are required. During developments of these applications, we found tendencies that the applications are made to run only for one environment. This causes the problem when we port the application into another environment, i.e. another building, which means the bad reusability of the application logic. In this paper, we use an abstract location model to describe relations among service appliances and users. In this model, service appliances are classified according to their services descriptions. We applied Aspect-Oriented Software Development (AOSD) with our model for binding main concerns and location related concerns of an application at deployment time in order to achieve portability. We propose to use a new concept, Location Scope for these bindings. The Location Scope is designed to make structured activities of the main concern, which allows handy description for reactive behaviors related with location issues. We implemented and evaluated the support system of Location Scope.

1. はじめに

近年，携帯端末や周辺機器の高性能化および無線環境の充実から，ユビキタス環境が普及してきている．ユビキタス環境の実現例として，様々な種類のサービスアプライアンス（3D プロジェクタ，壁面モニタ，プリンタなど）がネットワーク経由でサービスをユーザに提供し，さらにセンサ情報などにより位置情報の管理が行われる環境などが考えられる．こうした環境下では，サービスを位置に応じて利用する（Location-based Services: **LBS**）などモバイルユーザから様々なユースケースが存在する．これを実現するために，サービスアプライアンスはユースケースに合わせたサービス記述と位置情報をユーザに公開する必要がある☆．Printer Service Interface（PSI）[1] はプリンタにおけるサービス記述例である．本論文では特にサー

† 東京大学大学院情報理工学系研究科
The Graduate School of Information Science and Technology, The University of Tokyo
†† 国立情報学研究所
National Institute of Informatics

☆ 以下では特にビルなどの管理領域ごとに中央サーバがこれを管理するものとする．

ビスアプライアンスの提供するサービスが種類ごとにオントロジツリーとして一意に階層化され，事前に共有されている状況を想定する．

こうしたユビキタス環境下におけるモバイルユーザの活動を，ユーザの情報や位置情報を考慮して支援するロケーションアウェアアプリケーション（Location-Aware Application: **LAA**）が今後開発・利用されていくものと考えられる．動機としては主に 2 つ存在する．1 つ目は，オフィスビルなどで近くの共用壁面ディスプレイを用いるなどのユーザの利便性・生産性向上のためである．2 つ目は，ユーザの位置条件に応じた LBS を積極的に利用して，携帯端末のリソース制約（処理能力，表示能力，帯域，バッテリなど）を補うためである，LAA の動作モデルとして，特定の環境（組織の存在するビル内など）に入ってきた際にサービスとしてモバイルユーザの携帯デバイス上に配布される LBS 形式のアプリケーションを考える[2]．

各環境ごとに配備される LAA への要求として，短期・長期双方の変化への対応が求められる．短期の変化としてはユーザの移動にともなう周辺状況の変化や，利用可能なサービスの動的な変化が考えられる．長期的な変化として，新しいサービスの追加といった配備された環境の変化や，新しい組織への移植（異なる複数の環境に対して LAA を提供する状況）が考えられる．しかし，こうした環境の変化を LAA 開発者が予期して開発を行うことは通常困難である．開発の負担を可能な限り軽減するために，共通機能の再利用性と環境依存部分の変更容易性「ポータビリティ（Portability）」を考慮した LAA 開発手法への要求があげられる．

本論文では，適切な位置モデリングにより位置情報を用いる処理部分の環境への依存性を軽減することと，位置条件に関する処理をアスペクト（Aspect）としてとらえ，アスペクト指向開発を適用することを基本方針とした解法を提案する．これは，アプリケーションの別環境への移植・更新時に位置条件に関する処理を独立に変更することを可能にし，実行時（コンパイル時）になって具体的な環境依存性をアプリケーションに与えることにより，ポータビリティの向上を実現するというものである．この手法の実現のために，ロケーションスコープ（**Location Scope**）という概念の提案・利用をする．ロケーションスコープは位置条件に関する処理（関心事）の分離を支援し，位置条件の変化への対処動作記述の簡便性を高めることを支援する．また，アスペクト指向開発における関心事の組合せの際にロケーションスコープの情報に基づいて処理が行われ，環境への依存性が与えられる．

本論文は，2 章で本研究の対象とする問題について示し，3 章で本研究の提案手法を述べる．4 章で提案手法の実現に関して述べ，5 章で有効性に関する評価，議論を行う．6 章で関連研究との比較を行い，7 章でまとめと今後の展望について述べる．

2. ユビキタス環境におけるソフトウェア開発上の問題

この章では，まず LAA の一般的な構成を仮定し，LAA 開発におけるポータビリティ実現のために従来技術で妥当と考えられるアスペクト指向開発（Aspect-Oriented Software Development: **AOSD**）[3),4)] の適用を試みる．そのうえで，LAA 開発に残された問題点とその意義を明確にする．

2.1 LAA の一般的な構成

LAA の構成は機能（関心事）によって分けることができる．特に主要関心事とロケーションアウェア関心事という分類を行う．前者は，ユーザが利用するアプリケーションの主要機能となる部分の実装，または既存のアプリケーションの監視部分の実装が該当する．後者は，位置条件の変化に対処するための機能であり，ユーザやサービスの位置に関する情報を取得・解析したうえで対処するコードが含まれる．この機能が，既存のデスクトップ環境アプリケーションとの違いとなる．

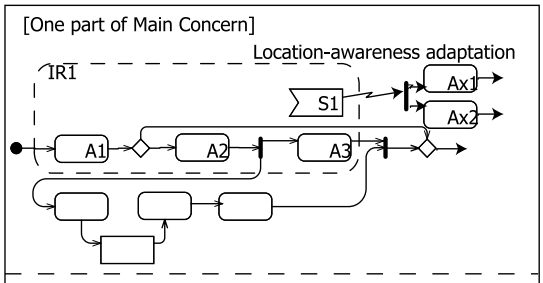
2.2 LAA のアスペクト指向開発

ユーザの移動にともなう位置条件の変化に対応するために，ロケーションアウェア関心事は主要関心事の至る所から呼び出される必要がある．たとえば，ユーザが移動したことにより最適なサービスを切り替えるための処理や，その処理を行う際にユーザの体感を落とさないように予測に基づいた処理を加えることなどが考えられる．これは関心事間の密結合につながり，LAA の長期的な変化への対応を妨げることとなる．

AOSD は横断的関心事の分離を進める手法であり，ここでは従来型の代表的な手法としてあげられる．一般的な AOSD の手順☆として以下のようなものが考えられる．

(1) 関心事の分離・設計：アプリケーションに対する要求は，主要関心事とロケーションアウェア関心事という単位で分離される．
(2) 関心事の実装：それぞれの関心事を独立に実装する．

☆ 一般的なアスペクト指向プログラミングツールで想定する手順




```
[Activity]
A1: receiving document data
A2: receiving related information of the document
A3: stream the data to output port
Ax1: change the output port to the wall display service (LBS)
(activity that is invoked only when S1 occurs insides the IR1
Ax2: get information of the URC of the service

[Signal]
S1: detect wall display service near the user
[InterruptibleActivityRegion]
IR1: Interruptible region for display services
```

図 1 位置条件の変化を意識した UML による設計の例．アプリケーションが本来実現したい主要関心事に加えて，位置条件の変化などへの対処動作（ロケーションアウェア関心事）をモデリングする

Fig. 1 An example of UML diagram that contains location-awareness behaviors. A core concern and reactions (location-awareness concerns) are modeled.

(3) 関心事の組合せ：実装した関心事どうしを組み合わせるルールを記述し，ツールを用いて組み合わせる．

本研究では LAA 開発が，AOSD に従うものとした場合を考える．また，設計・実装に関しては従来のオブジェクト指向開発で一般的に用いられている UML[5]，Java 言語を用いるものとする．

アプリケーションの例として，携帯端末上で動作するライブコミュニケーション用アプリケーションを想定する[6]．携帯端末で通信中の相手から動画などのデータが送信される際，周辺にユーザの携帯端末よりも高解像度なディスプレイサービス（LBS）が存在するようであれば，それを利用し，その際固有のリモコン画面をアプリケーションがユーザ端末上に描画するというものを考える．

図 1 に UML アクティビティ図を利用した設計例の一部を示す．データ表示を必要とする前のアクティビティ（A1，A2，A3）を実行している間（IR1☆）に，ディスプレイサービス（LBS）を発見することができたのであれば（S1: Signal），そのサービスを利用する（Ax1，Ax2）という動作を表現している．ここでの主要関心事は受信したデータをブラウズすることであり，他方ロケーションアウェア関心事はユーザの利便性を向上させるために，適切な条件を満たす際に LBS を利用することである．

この情報をもとに双方の関心事を組み合わせることで，ロケーションアウェア関心事を切り替えやすくなるため，ポータビリティの仕組みが得られる．しかし，依然として長期的な環境変化への対処に関する問題が残る．次の 2.3 節でその問題を明確にする．

2.3 LAA の開発における問題点

AOSD を適用したうえでも残る問題点は，LAA の位置条件の利用箇所に起因する．従来型の位置条件の指定方式はインフラよりの視点で論じられてきた結果，記述が具体的すぎて他の環境で動作しない状況が考えられる．現状として，取得できる位置情報の形式・種類が環境によって異なるうえに，既存のロケーションアウェアアプリケーション用のフレームワークも座標や恣意的なシンボルにより位置情報を扱うためである．

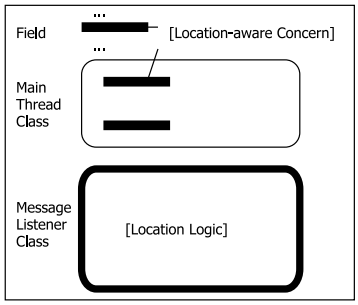
また，位置条件の変化への対処動作は既存のイベント駆動プログラミング（Publish/Subscribe 型[7),8)]）で記述することが考えられるが，位置条件と種類の増加にともない記述が環境に依存したものになる．こうした処理記述が主要関心事から横断的に呼ばれる際にも呼び出し元の条件が環境に依存した記述になっているため，関心事の組合せ記述もそのつど書き換える必要が発生し，AOSD を利用する負担が逆にかかってしまう．**図 2**，**図 3** にその状況を示す．図 2 では個々のアクティビティ単位で見た際に，主要関心事中にロケーションアウェア関心事が混在する様子を示している．また図 3 ではアプリケーション全体として見た際に，共通するロケーションアウェア関心事がアプリケーション中を横断する状況を示している．

ポータビリティのある LAA を作成するために，本研究ではロケーションスコープ（Location Scope）という概念に基づき，適切な位置情報の抽象化，ロケーションアウェア関心事の分離を実現するアプリケーションコンテナ，およびそれらを実現するためのインフラ機能について提案を行う．

以下の章では，これらの問題に対する提案手法に関して説明を行う．

☆ 文献 5)，Figure 259. Interruptible Region（IR）は複数のアクティビティノードを含む．トークン（token）がエッジから IR を出たとき，IR 内のすべてのトークンや振舞いは停止される．本研究では，停止するかどうかを別途指定することが可能な拡張をして用いる．

Activity model (A component unit of application)



[Field] contains condition variables

```
...
private Set<TopicSubscriber> subscriberSet_;
private boolean needVisualOutService_;
private boolean needSoundOutService_;
...
```

[Main Thread Class] contains interactions with infra.

```
public void run() {
   new Thread(new Runnable() {
   public void run() {
     // Main concern
     doSomething();
     // Location-aware concer
     needSoundOutService_ = false;
     doSomethingElse();}}).start();
     Context context = JNDIUtil.getInitialContext();
     TopicConnection topicConnection = JNDIUtil
               .getTopicConnection(context);
     TopicSession topicSession = JNDIUtil
               .getTopicSession(topicConnection);
     Topic topic = JNDIUtil.getTopic(context);
     subscriber_ = JNDIUtil.getTopicSubscriber(
        topicSession, topic,USER_PREFERENCE_MONITOR);
     subscriber_.setMessageListener(new MockListener());
     //add as much as needed
     ....
```

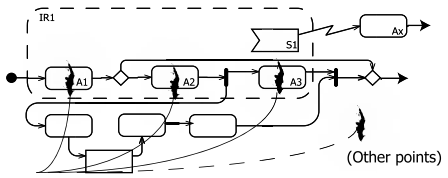
[Message Listener Class]

```
public class MockListener implements LocationMessageListener {
    public void onMessage(LocationMessage message) {
        try {
            String service = message.getStringProperty("service");
            String wsdl = message.getStringProperty("wsdl");
            String QoS = message.getStringProperty("QoS");
            if (needVisualOutService_ && needSoundOutService_) {
                if (wsdl.equals(VisualOut.class)) {
                    if (QoS.equals(USER_PREFERENCE_MONITOR)) {
                    // DynamicInvoker.invoke(wsdl, ...);
                        doReaction();
                    }
                }
            } else if (needSoundOutService_) {
                // ...
            } else {
                throw new Exception("Unhandled");
        ....
```

図 2 LAA における関心事の密結合．個々のアクティビティ単位で見た際に，主要関心事中にロケーションアウェア関心事が混在する．フィールド（Field）には位置条件の監視に必要な情報が含まれ，メインクラス（Main Thread Class）には対象とする位置条件の変更などの操作が含まれる

Fig. 2 These mock codes show how concerns in LAA are deeply coupled. Even in the core concern (Field, Main Thread Class), there are some location-related descriptions.

Bird' s eye view of the application model



Call of reactions crosscut the application model

図 3 LAA における関心事の密結合．アクティビティの集合を 1 つのアプリケーションとして見る際に，同じロケーションアウェア関心事が複数のアクティビティに横断して出現する状況が考えられる．これは主要関心事，ロケーションアウェア関心事とも更新が困難になる状況である

Fig. 3 This figure shows how concerns in LAA are deeply coupled. Each location-awareness reaction may appear several points in the activities. This makes it hard to maintain both the core and the location-awareness parts.

3. 提案手法：Location Scope によるロケーションアウェアアプリケーションの開発支援

この章では，本研究の提案するロケーションスコープ（Location Scope）というロケーションアウェア関心事の分離を支援するためのモデルの特徴，および利用手法に関して述べる．ロケーションスコープは，アスペクト指向開発における関心事の組合せの際に利用される情報を持つ．その特徴としてあげられる点は，抽象的な位置モデルを利用しアプリケーション（ユーザ端末）を取り巻く外界の動的な変化をアスペクト指向開発の概念（joinpoint：プログラムの実行時点）として含有する点である．

ロケーションスコープの概念は一般的なアスペクト指向開発に基づいた**図 4** に示す開発プロセスの中で利用される．このプロセスにおいて開発者は，関心事を主要関心事とロケーションアウェア関心事とに分離したうえで設計と実装を行う．UML アクティビティ図を用いて関心事の組合せに関するモデリングを行った場合，図 1 のように位置条件による影響を様々なケースで分析したものが中間生成物となる．図 1 における IR1☆などの情報は，XML 形式のロケーションスコープ記述（Location Scope Description: **LSD**）に変換され，関心事の組合せに必要な情報として保持される．この LSD が，実装段階の成果物とアプリケーション実行環境（Application Container）の提供する

☆ アプリケーションの実行単位（アクティビティ）のうちある特定の位置条件の変化の影響を受けるものを特定する．

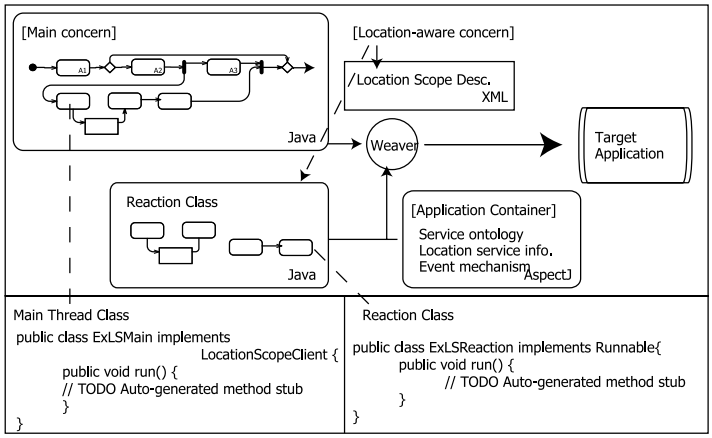


図 4 ロケーションスコープを導入した開発手順．［Core Logic］中の個々のアクティビティ，および［Location Logic］位置条件の対処動作は java.lang.Runnable の実装クラスである．ロケーションスコープ記述をもとに AspectJ（weaver）を用いてアプリケーションコンテナの機能をアプリケーションに組み込む

Fig. 4 The development process with the Location Scope. Each activity and reaction (Core Logic, Location Logic) implements java.lang.Runnable. AspectJ weaves these components and some functions from application container according to a location scope description file.

機能（位置情報，サービス情報など環境に依存する情報を取得する機能）とを結合することによって最終成果物であるアプリケーションを出力する．ロケーションスコープとアプリケーション実行環境の機能を利用することで，主要関心事とロケーションアウェア関心事は運用段階になるまで運用環境依存の情報を保持しない．これにより，他の環境への移植などで開発段階を再度反復するなど，長期的な変更の際の更新コストを下げることを実現する．以下では，ロケーションスコープの構成要素について説明する．

3.1 位置モデリング

3.1.1 トポロジに基づいた位置モデルの背景

ソフトウェアが位置情報を利用する際には位置情報を適切にモデル化する必要があり，様々な手法が提案されてきた[9),10)]．文献 10) では位置モデリングに関する汎用的な要求を 4 つあげているが，その中にトポロジ関係を取得できること，という項目が含まれている☆．これは，ある一定の領域内に対象とするオブジェクトが含まれているかどうか（containment），および 2 つのオブジェクトの点間の距離が一定範囲内かどうか（connection），という情報が LAA 実行時に要求されることを意味している．従来型位置モデルの多くは，部屋の座標やものの識別子などの特別な場所やものに束縛された，いわば絶対位置による位置モデルであり，1 つの環境にインフラからアプリケーションまですべてを作り込む形での開発を行う際に用いられてきたものである．これに対し，特定の場所への束縛の緩い，相対位置による位置モデルも考案されてきた[11)]．ここでの相対位置とは 2 つのノード間のトポロジ状態（containment，connection）を意味する．たとえば 2 つのノード間の距離が遠ざかり接続が認められなくなったときなどが相対位置の変化に該当する．本研究ではこの相対位置モデルの拡張を利用する．

3.1.2 サービス指向トポロジ位置モデル

本研究で提案し利用するモデルは，ユーザ端末ノード（アクタノード）と，それを取り巻く周辺機器の提供するサービス（サービスノード）とのトポロジを考慮したものである．ここでの接続性はアクタ/サービス両ノードが一定範囲内に存在する状態とする．これは，特定の「場所（座標）」への束縛を軽減することになり，ロケーションアウェアなアプリケーションのポータビリティ向上につながる．

従来モデルとの違いはトポロジの抽象度にある．従来のトポロジに基づいた位置モデルにおいては，個々のデバイス間の接続性が対象となっていたため異なるデバイスは同一視されることがなかった．これをアクタ/サービスノード間の接続と抽象化することで，以前のモデルでは別個のトポロジと分類されていたものが同一のトポロジと見なされることとなる．これは，特定の「もの（GUID）」への束縛を軽減し，ロケー

☆ 他の項目は，オブジェクトの位置取得，距離の取得，方向の取得，である．これらすべてを満たす必要があるというわけではなく，典型的なユースケースをカバーするためにこれらを組み合わせるというものである．

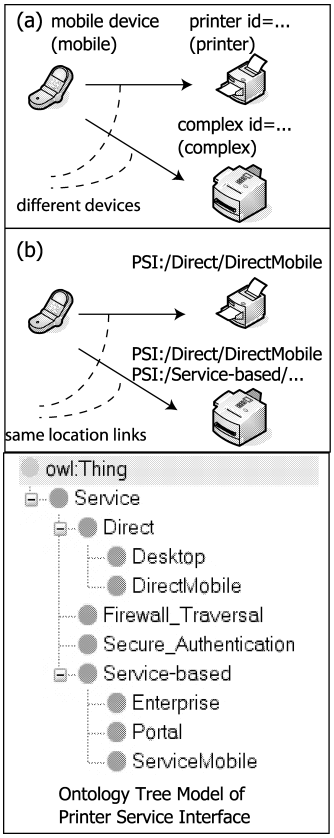


図 5 従来型トポロジモデルとサービス指向トポロジモデルの比較と PSI（Printer Service Interface）を例としたオントロジツリー

Fig. 5 The comparison between an ordinal topology model and the service oriented topology model with PSI (Printer Service Interface) ontology tree.

ションアウェアなアプリケーションのポータビリティに貢献すると考えられる．これは，運用環境に新しくサービスが追加された際に柔軟な対応をとることを支援する．**図 5** に，従来方式との比較を示す．ユーザの持つ携帯端末（mobile）と，プリンタ（printer），複合機（complex）という印刷機能を提供するデバイスが近くに存在することが位置情報サービスから取得されている状態を考える．図 5 (a) における従来利用されているモデルでは，mobile から見た printer，complex はそれぞれ別個の識別子を持つデバイスとして認識される．図 5 (b) は本研究の提案方式であり，指定されたサービスを実装しているノードは同一視される．printer，complex ともプリンタサービス階層中の DirectMobile（PSI の定義に基づいたサービス階層の中のノード）機能を実装しているため，これらのノードを位置モデルの観点で同一視できるというものである．また，指定したレベルよりもツリー上で下位のオントロジ概念に対応するサービスもトポロジの端点として認識されることになる．これはプログラマが抽象的にサービス選択記述をすることの支援となる．

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <locationScopeDescription>
3   <locationTopologyDescription name="ltd_example">
4      <targets>
5          <target>PSI:Direct/DirectMobile</target>
6      </targets>
7      <adjacent>
8          <base>urn:epc:id:IIJEFKL</base>
9          <qualifier>near</qualifier>
10     </adjacent>
11  </locationTopologyDescription>
12   <locationScopeElements>
13     <id>...</id>
14     <in>
15         <activity>DrawURC</activity>
16         <lifecycle>spawn</lifecycle>
17     </in>
18      <out>
19         <activity></activity>
20         <lifecycle></lifecycle>
21     </out>
22     <activities>
23         <activity>A1</activity>
24         <activity>A2</activity>
25     </activities>
26   </locationScopeElements>
27  </locationScopeDescription>
```

図 6 ロケーションスコープ記述（XML）

Fig. 6 Location Scope description (XML).

3.2 Location Scope

本研究の提案手法である，3.1 節で導入した位置モデルを用いて，主要関心事のポータビリティの実現を考慮した LAA 開発手法について述べる．ロケーションスコープは，アスペクト指向開発における関心事の組合せの際に利用される情報を持つ．横断的関心事（crosscutting concerns）としては，図 2 に示したようにロケーションアウェア関心事を想定しており，これは指定したノード間のトポロジの変化に対応して起動される処理である．関心事の組合せに必要な情報と，実際に XML 形式で表現された LSD（Location Scope Description）☆との関係を以下にあげる．

- 対象とするノード間のトポロジ：**図 6** の 3〜11 行目は監視する位置トポロジに関する情報を含む．4〜6 行目はトポロジの端点を担う対象となるサービスをサービス階層表現を用いて記述している．7〜10 行目では，基準ノード *base* と近接度合いについて記述している．
- ロケーションアウェア関心事：設計段階において，図 1 に示すように位置状態の変化へのリアクションは，UML2.0 の *InterruptibleActivityRegion* などを用いて表現される．14〜17 行目ではロケーションスコープに入るときの動作を記述する．そ

☆ 詳細定義は付録に記述する．

の際，呼び出されるアクティビティのクラスと，呼び出される側，呼び出す側の並列性定義（ライフサイクル定義）を含む．18〜21 行目では同様にロケーションスコープから出るときの動作を記述する．

- 主要関心事との関係：22〜25 行目において主要関心事のアクティビティ（*InterruptibleActivityRegion* に含めた要素）を記述することで主要関心事のどの状態の際にロケーションスコープが有効かを指定する．

これにより，ロケーションスコープは「アプリケーションが各実行遷移状態において理想的な位置条件を満たしている状態」という意味を持つ．ユビキタス環境の性質としてネットワークトポロジの動的な変化が考えられるため，ロケーションスコープから外れる状況が頻繁に起こることが考えられる．たとえば，ユーザの側の壁面モニタをプロセスの実行開始時に特定しても，ユーザが移動をしてしまうと最適なものではなくなってしまう．つまり，これらのアクティビティを含むプロセスにとって理想的な状況でなくなってしまう状況（out）が頻繁に起こりうるため，対処を明示的に扱う必要があると考えられる．同様に理想的な状態に状況が変化するケース（in）も考えられる．ロケーションスコープは，アプリケーションの実行情報，位置情報の双方を用いてフィルタリングされた状況の変化に対して，どのような対処をとるかを決定することになる．

対処動作の例としては，状況の好転を待機，ログレポート作成，ユーザへのメッセージ，スコープの条件を満たす場所に移動して実行を再開，などが考えられる．また，サービス固有の情報を受け入れて携帯端末に表示させる（URC，サービスの情報，他の利用者の情報など）ことも考えられる．これにより，ユーザの周辺にディスプレイサービスがあるときは利用し，ないときは携帯端末の画面を利用する，というようなロケーションアウェアな状況における様々なユースケースの実現を，主要関心事と分離した状態で進めることが可能となる．

4. ロケーションスコープ支援系の設計と実装

本研究で提案するロケーションスコープを用いたアスペクト指向開発を行う際には，アプリケーション実行環境をサービス側・ユーザ端末側双方に用意する必要がある．以下ではこうした支援系に要求される機能とその実装に関して述べる．

4.1 実現のための機能要求

4.1.1 位置情報提供サービス（インフラ側）

ロケーションスコープによる開発の実現のため，インフラに関しては以下の層の提供する機能が必要となる．

- サービス層：アプリケーションからのリクエストを受け付ける．ロケーションスコープ記述（LSD，図 6）を入力として受け付ける．LSD の中には，オントロジツリーの中から選択・指定されたサービスが含まれている．サービス層では，この指定されたサービスを提供可能なサービスノードと携帯端末との位置関係の監視を，下位層の機能で行い，条件を満たした時点にアプリケーションにメッセージを送信して通知する．
- メッセージ層：メッセージ層では，アプリケーションとサービス層間のメッセージ通信のためのインフラを提供する．アプリケーションは指定されたキュー（Queue）を利用して LSD を用いた要求を出し，そのときに指定されたトピック（Topic）を購読する☆．
- クエリ層：クエリ層は，位置トポロジの状態を問い合わせるためのインタフェースをサービス層に提供する．
- ノードグラフ層：ノードグラフ層は，位置モデルの基準点（例：RFID タグリーダ）の位置関係をグラフとして保持し，各ノード（例：RFID タグのついたデバイス）がどこの基準点の範囲に存在するか（認識されているか）に関してデータベース情報を動的に更新する．

4.1.2 アプリケーションコンテナ（クライアント側）

LAA を構築するうえで，サービスを利用するために環境ごとに切り替えなくてはならない機能や設定が存在する．たとえばアプリケーション中の各アクティビティのコードに記述するイベントリスナ，対処動作などが外部のサービスに依存している場合，その環境に応じた情報をそのつど与えなくてはならない．そのためアプリケーションコンテナは，以下のタスクをアプリケーション実行時（またはコンパイル時）に行う．

- 指定した位置条件の変化の購読（Subscribe）処理の挿入

☆ キュー，トピックはそれぞれ Java Message Service[12] で利用される用語である．キューは PTP メッセージングモデルに利用され，ここではアプリケーションが別の 1 つのアプリケーションにメッセージを送信できる．トピックは Pub/Sub メッセージングモデルで利用され，ここではアプリケーションが複数のアプリケーションにメッセージを送信できる．

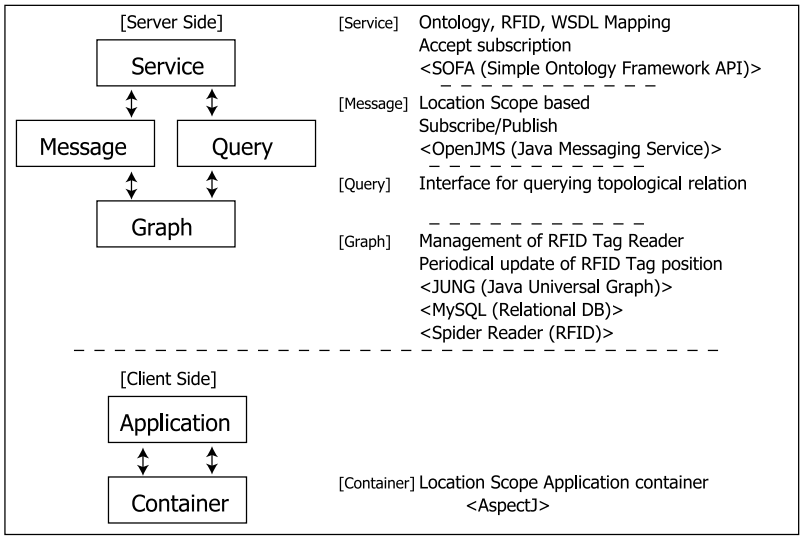


図 7 ロケーションスコープを実現するうえで必要となるサーバ側，クライアント側の機能スタック．<> 内はプロトタイプ実装に用いたライブラリなどの名称である

Fig. 7 The functional stack for a location scope system. The stack covers both client and server sides. The terms inside "<>" are the 3rd party tools that are used in our implementation.

- 指定されたアクティビティが位置条件の変化に対処するための機能の織り込み

これは，EJB の軽量コンテナなどで採用されるアスペクト指向プログラミングを携帯端末上においても，機能をドメインに特化したうえで実現したものであり，コードのメンテナンス性に貢献する．

4.2 実　　装

本研究では，**図 7** に示した各層のプロトタイプ実装を行った．これらの層の機能は**図 8** のような手順でロケーションスコープの情報をアプリケーションに伝えるためのものである．まず，アプリケーションからのロケーションスコープ購読要求に対して，サービス層では SOFA（Simple Ontology Framework API）[13] を用いてアプリケーションの要求するオントロジツリーで指定されたサービスを提供可能なノードの識別子（RFID タグの文字列）に変換している．これらの RFID タグの集合と基準点として指定された RFID タグとの位置関係を，クエリ層からノードグラフ層のデータベースに定期的に問い合わせる．ノードグラフ層，クエリ層の機能のために，文献 14) で著者らが用いた RFID による位置情報管理システムを用い，最新のトポロジ関係情報を得られる機能を追加した．その実現のために JUNG（Java Universal Graph）ライブラリを用いて**図 9** に示す RFID タグリーダの位置をグラフのデータ構造で管理している．またリーダから認識されている RFID タグの集合を動的にデータベースに更新する．ここでは，同じリーダから認識されている場合かグラフ上隣接するリーダから認識されている場合に *near* 集合とし，それ以外の場合を *elsewhere* 集合とする．このトポロジ関係を定期的に監視し，(1) *near* 集合が空集合から非空集合になったときには *LScopeIn* メッセージを，(2) *near* 集合が非空集合から空集合になったときには *LScopeOut* メッセージを，(3) *near* 集合の構成要素が変化したときには *LScopeUpdate* メッセージを，それぞれ JMS サーバ経由で通知する．

アプリケーション側では，コンテナからアプリケーションに織り込まれたアスペクトがこれらのメッセージを受信し，主要関心事の実行状態が購読条件を満たしている場合，対処動作クラスのコールバックメソッドである *LScopeIn()/Out()/Update()* をそれぞれ起動する．また，指定された *Lifecycle* に応じて主要関心事の実行を制御する．これらの実現には AspectJ ☆を利用した．

5. 評　　価

本提案手法の評価として，通常実行時と比較した際の実行時間上のオーバヘッドと，ポータビリティが得られるかどうかの検証とを行った．

☆ AspectJ は，Java でアスペクト指向プログラミングを実現するための言語仕様を追加したプログラミング言語である．

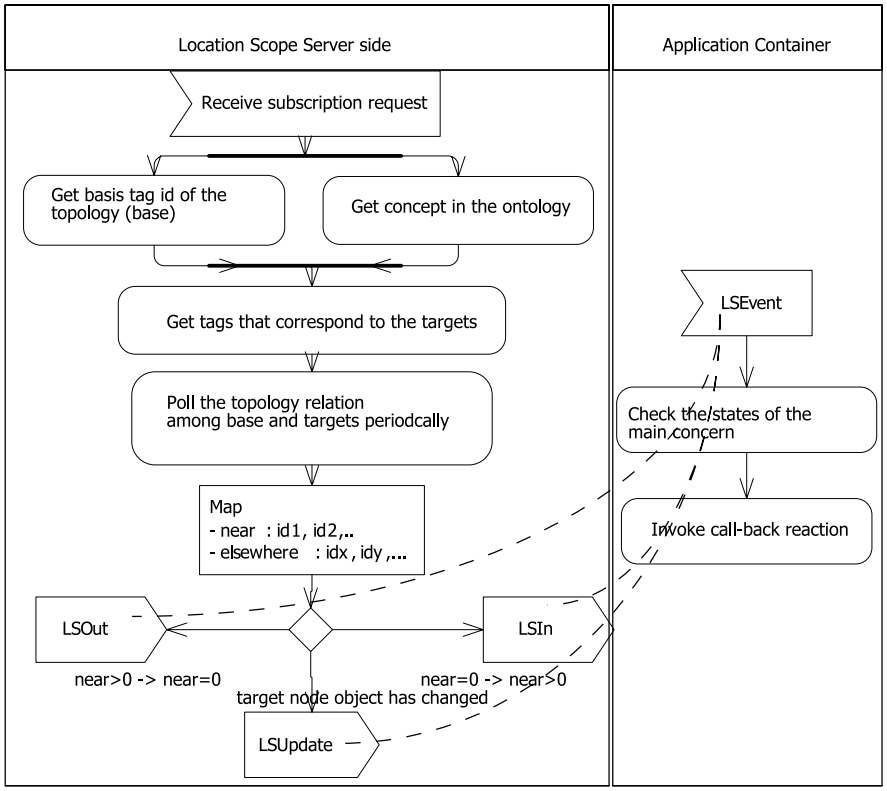


図 8 ロケーションスコープに基づいた対処動作を起動するまでのアクティビティ図
Fig. 8 This activity diagram shows a procedure for invoking location scope-based reaction.

```
<?xml version="1.0" encoding="UTF-8"?>
<graphml xmlns="http://graphml.graphdrawing.org/xmlns"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://graphml.graphdrawing.org/xmlns
 http://graphml.graphdrawing.org/xmlns/1.0/graphml.xsd">
  <graph id="G" edgedefault="directed">
    <node id="n1"/>
    <node id="n2"/>
    <node id="n3"/>
    <node id="n4"/>
    <node id="n5"/>
     ...
    <edge source="n1" target="n2"/>
    <edge source="n2" target="n3"/>
    <edge source="n2" target="n4"/>
    <edge source="n3" target="n4"/>
    <edge source="n2" target="n5"/>
     ....
  </graph>
</graphml>
```

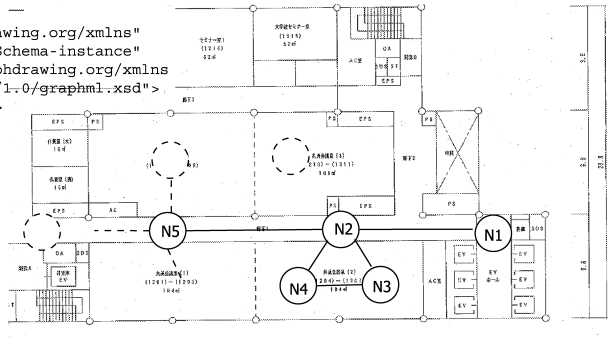


図 9 ノードグラフ層の実現イメージ．グラフは GraphML を利用して記述している
Fig. 9 An example of node graph layer implementation. GraphML is used to denote node positions.

5.1 ロケーションスコープ利用によるオーバヘッドの計測

ロケーションスコープを利用した際の処理は，(1) サービスに購読要求を出す，(2) サービス側で情報を登録する，(3) サービス側で情報を更新する，(4) 条件を満たした際に 4.2 節で示した手続きを経てアプリケーションの対処動作を呼び出す，という大まかな分類がなされる．

また，比較対象の従来型モデルとして，指定したサービスが利用可能・不可能間の変化への対処動作を実行する単純なものを考える．**図 10** の UML シーケンス図は，アプリケーションが対処動作を起こす必要がある状況を把握するまでの双方の実行シーケンスである．図 10 中の太線で示されるメッセージシーケンスは処

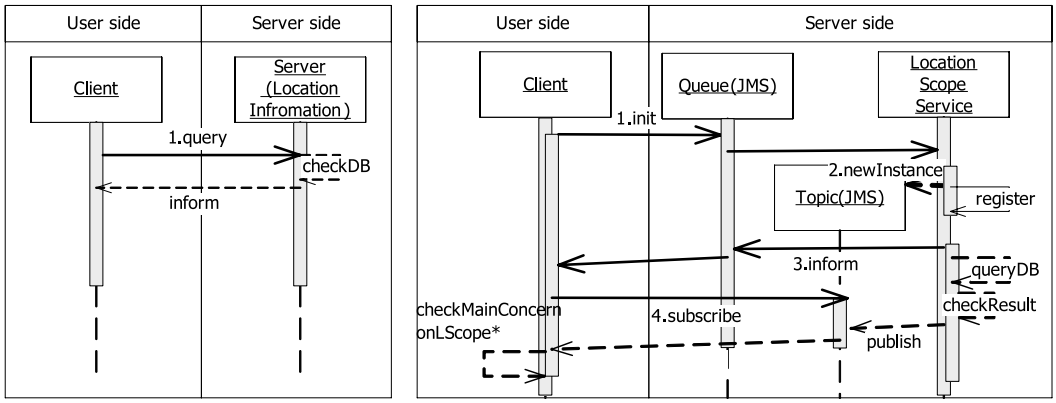


図 10 単純なクライアント・サーバモデル（左）と，ロケーションスコープモデル（右）との実行シーケンス比較．数字付きメッセージは位置条件を取得するために必要なプロトコルである

Fig. 10 A sequence diagram for comparison between simple client/server model (left) and location scope model (right). Numbered messages are needed to obtain location information.

表 2 オーバヘッドの計測（ロケーションスコープ型）単位はミリ秒

Table 2 Overhead measurement regarding to the location scope (msec).

	Calculation	Network	Database	(Numbered)	(All)
1.init	1,266	1,266		2,532	2,532
register			906		3,438
2.newInstance	16			2,548	3,454
3.inform		328		2,876	3,782
4.subscribe		94		2,970	3,876
queryDB, checkResult	314		110		4,300
publish		359			4,659
checkMainConcern, onLScope*	0				4,659
total	1,596	2,047	1,016	**2,970**	4,659

理 (1) に対応する．ロケーションスコープ型の場合アプリケーションごとに通知すべき情報が異なるため，初期化処理が増加する（OverHead-1）．図 10 中の点線で示されるメッセージシーケンスはアプリケーションの要求する状態を満たしているとした際に，その情報が通達されるまでの処理を示す．このときもロケーションスコープ型の場合，サーバ側とクライアント側（アスペクト）の双方でデータベースへの条件問合せなどのオーバヘッドが発生する（OverHead-2）．これは処理 (4) に対応する．処理 (2) ではアプリケーションに伝えるべき情報が即時に得られている場合に限りオーバヘッドとなる．処理 (3) の処理に関しては共通の処理であるためオーバヘッドは生じないものとする．以下では，OverHead-1，2 双方のコストを検証する．

実験環境としては，1 台の PC（CPU: Intel Pentium M 1 GHz，メモリ 1.24 GB）と 2 つの RFID リーダを用いてシミュレーション実験を行った．計測結果は**表 1**，**表 2** に示すとおりである．時間の内訳として

表 1 オーバヘッドの計測（従来型）単位はミリ秒

Table 1 Overhead measurement regarding to the normal sequence (msec).

	Calculation	Network	Database	(All)
1.query	1,047	375		**1,422**
checkDB			438	1,860
inform		422		2,282
total	1,047	797	438	2,282

計算コスト（Calculation），通信コスト（Network），DB コスト（Database）との分類を与えた．計測の段階は，図 10 のシーケンスに従う．

計測結果より，Overhead-1 に関しては約 1.5 秒，Overhead-2 に関しては約 0.8 秒の遅延が生じる☆．また，通信コストの占める割合はそれぞれ 84.8%，

☆ Overhead-1 に関しては 2,970 − 1,422 = 1,548 msec，Overhead-2 に関しては (4,659 − 2,970) − (2,282 − 1,422) = 829 msec

45.2%☆である．実際の環境でネットワーク遅延がある場合を考えたときには，この数値より大きくなることが想定されるが，すべてのメッセージサイズが小さいこと，想定するネットワークがビル内のイントラネットであること☆☆などから，実環境でもたかだか数倍の遅延と考えられる．Overhead-1 に関しては，アプリケーションの実行開始と同時に行われる処理であり，アプリケーションユーザの移動・携帯端末操作・データ蓄積などの作業と並行して行われることを想定するとき，致命的な遅延になることは少ないと考えられる．Overhead-2 に関しては，メッセージの伝達コストが問題となっているが，実際に運用するうえでは RFID タグなど位置情報の更新の間隔などと同レベルの程度であると考えられる．

以上より，基本的なシーケンスに関するオーバヘッドは，おおよその目安としてロケーションスコープによる開発が実用レベルのものであることを示した．

5.2 ポータビリティに関する議論

ここでは提案手法であるロケーションスコープがポータビリティ向上に寄与することを示す．

ロケーションアウェアアプリケーションがポータビリティを得るための解決策として，抽象的な位置条件の記述を許容することと，主要関心事と位置条件の変化への対処ロジックを独立して変更できることをあげた．特に本研究では，サービス指向トポロジ位置モデルとロケーションスコープを用いたアスペクト指向プログラミングの適用とによりこれらの実現を図った．

ロケーションスコープは，アプリケーションの実行時情報と実世界上の位置情報との双方を用いてプログラムのモジュール化を進める．これは，アスペクト指向開発の長所を利用するために適切な位置モデルを行ったものと考えられ，設計の変更などに対処しやすいものとなる．図 2，図 3 で指摘した主要関心事，ロケーションアウェア関心事のコードレベルにおける密結合は，ロケーションスコープで指定されたアクティビティと実行時に結合されるため回避できる．

たとえば，支社のビル用に作成したアプリケーションを本社ビルという新しい環境で動作させる際に，本社には支社になかった共用壁面ディスプレイが点在するようなときを考える．周辺に人がいないなどの条件下で，携帯端末の小さな画面に表示していたデータを大きなディスプレイに映すことで作業効率を上げるという変更を加える際に，ロケーションスコープを用いることで，ディスプレイを利用可能な状況，利用不可能な状況の処理を追加できる．その際，ロケーションスコープはサービスレベルの到達性に位置表現を抽象化して保持しており，サーバ側のサービスの存在を仮定することにはなるが，ポータビリティに寄与している．また，逆の例としてアプリケーションをよりインフラの整備されていない環境下に移植する際に，機能を削減する際にも本方式は適当な手段を提供すると考えられる．

5.3 適用範囲と限界について

本研究の提案手法は，携帯端末の処理能力を補う必要性のある作業（画面，処理力，ネットワーク，データサイズなどに関係）をともなうアプリケーションを開発する際に有効であると考えられる．

考えられる制約としては，各環境にロケーションスコープ支援環境の配備コストがあげられる．これは各環境で図 7 の層以下に存在するネットワーク，センサなどのインフラに多様性があり，クエリ層との間にその差を吸収するコンポーネントを用意する作業に関連する．これに関しては，一般的にロケーションアウェア環境に必要とされている位置トポロジを取得できる環境であれば，ロケーションスコープ支援環境の移植は可能である．本論文の実装には RFID を用いたが，超音波，画像認識など他のロケーション技術を用いた場合でも位置トポロジ取得は同様に要求仕様であると考えられるため，そのうえで様々なノードグラフ層の上にロケーションスコープ支援系は構築することが考えられる．実際に様々な位置情報管理技術を組み合わせた状況下でロケーションスコープ支援系を配備・評価することは今後の課題である．

2 章であげた具体的な位置記述によるポータビリティの低下という問題に対しては，サービス指向トポロジ位置モデル用いることで対処している．この位置モデルは，ユーザの周辺に遍く存在するデバイスを提供するサービスの種類で分類するというものであった．これに関しては，本研究の仮定である「サービスアプライアンスが LBS として利用される」，「サービスアプライアンスの提供するサービス機能がプリンタにおける PSI のように標準化され，オントロジツリーを構成する」，「ユーザは公開 LBS（プリンタ，ディスプレイ，スピーカ，カメラ，自販機など）を利用する」という状況が今後進展していくとき，本手法が現実的なものとして利用されうると考えている．また，それと同時にこうした機能の標準化も進んでゆくことが予想されるが，機器ごとにサービス品質の差異を残しなが

☆ Overhead-1，2 のうち Network 値の値を引いて再計算した場合との比較

☆☆ ping であれば IEEE802.11b 利用時でも 2〜3 msec で到達することが予期される．

ら進むことを想定している．

他の制約として，既存のデスクトップアプリケーションをロケーションアウェアな適応動作を含むものに変換する場合，元のアプリケーションの実現手法によっては本提案手法の適用のコストが逆に大きくなる場合も起こりうる．これに関しては本研究の提案する方法では対応することができず，現時点での限界として認識している．

6. 関 連 研 究

この章では，関連研究との比較を通して本研究の特徴を述べる．

位置モデリング

文献 9) においてロケーションモデルは Containment（包含関係），Positioning（座標表現）の 2 つに大別されている．本研究で用いている，サービスレベルのトポロジに基づいたロケーションモデルは，これらの分類で利用されるレベルの詳細な位置情報は利用可能であると仮定したうえでの抽象度の高いモデルと考えられる．また，幾何学的に 2 地点間におけるトポロジの関連は，共通部分に着目した disjoint，contains，within そして overlap が一般的に扱われる[15]．しかし，インフラ側の視点でなく，ロケーションアウェアアプリケーションにとって重要な情報は接続性であると考えられる[16]．LBS を利用するロケーションアウェアソフトウェアの開発という目的を持つ本研究においてもこの到達性がモデリングの際に重要となる．本研究では，共通なサービスを提供する，代替可能なデバイスを同一視して，接続性と組み合わせたモデルを用いることで開発者が扱わなくてはならない情報量を抑えることを主眼においている．

他方，文献 16) で与えられている詳細なセマンティック位置モデルなどにより，本研究で用いている位置モデルも定義することは可能であると考えられる．この場合，詳細な記述の許容と，実行環境に依存する機能をアプリケーションが含むことによるポータビリティの低下とが，トレードオフとして考えられる．また，Semantic Space [11] はトポロジベースのロケーションモデルで，Semantic Space 間の包含関係や Atom のプレゼンスを表現する．Space は簡単に Type 分けされており，たとえば近接度の高いプリンタだけを参照することなどを想定している．本研究ではこの考え方を拡張し，LBS の提供するサービスインタフェースの機能階層に着目したモデリングを行っている．これは，近接度の高い周辺機器を適宜利用するというユビキタス環境におけるユースケースを実現すること，サービスインタフェース多様化に対応することを考慮しているためである．高機能な LBS を頻繁に利用するアプリケーションを開発するうえで，より適したモデルであると考えられる．

本研究では，位置モデルをアプリケーション開発を直接駆動するために利用するということに主眼を置いており，関心事の組合せに必要な情報という観点でも位置情報をモデリングしている点で，先にあげた位置モデリングだけを取り扱う既存研究との有意な差があると考えている．

Location-aware Computing のソフトウェア工学的観点

本研究のモチベーションと同様，ロケーションモデリングに加えてソフトウェア開発者の支援をする研究として，文献 17) があげられる．文献 17) では，ORM モデルを拡張しグラフィカルにエンティティ間の関係をモデリングする CML（Context Modeling Language）が利用されており，プリファレンス，および分岐（*branching*）やトリガリング（*triggering*）という汎用的なプログラミングモデルと CML 表現された位置情報とを組み合わせることの提案を行っている．トリガモデルによって，*upon*，*when*，*do*，*always* などを利用して実現できる動作は本研究で提案するロケーションアウェア関心事の分離と同等の効果が得られると考えられる．また，CML はロケーションアウェア関心事の導出などで利用することも可能である．しかし，トリガの条件として *upon EnterFalse?*（*Occupied(APPLICATION_SPECIFIC_STRING)*）というように特定のアプリケーション，環境でのみ存在する具体的な情報を与えることを想定しており，本研究におけるアスペクト指向でポータビリティを得るための仕組みは特に考慮されていない．

Olympus [18] は，Gaia Location Middleware [19] 上で動作し，パーベイシブコンピューティングのための高水準プログラミングモデルを提供している．Active Spaces というパーベイシブコンピューティング環境を細分化した物理空間を表す概念を用い，Active Spaces に対する高水準な操作を与えている．たとえば，ユーザ A のいる Active Space を指定，インスタンス化し，その空間内のアプリケーションのうち所有者が A のものを指定，インスタンス化する．そのうえでアプリケーションへの操作（start，resume など）を実行するというものである．本研究との相違点としては，Olympus の実行モデルが能動的かつ静的なモデルに基づいている点があげられる．他方，本研究では位置条件に左右される受動的な動作に関して主に扱ってい

る．これはユビキタス環境における動的な状況の変化を想定しているためである．アプリケーションの実行される環境に対して管理権限などがあり，存在するアプリケーションなどの情報を持ち，十分にリソースが用意されているときに関しては，“そこにあるアプリケーションを使う”というOlympusのプログラミングモデルに問題はないと考えられる．しかし，そのような情報が与えられていないような状況での開発時には，本研究の手法を適用するなどの対処が必要になると考えられる．

アスペクト指向の応用

本研究では，ロケーションスコープをアスペクトの組合せルールとして用いる際に，JBoss[22]，Spring[23]などアスペクト指向の概念を取り入れた軽量コンテナで行われている処理を携帯端末上でも行う．Jadabs[24]は携帯端末への移植の試みであり，動的に最適なアプリケーションコンポーネントを配備することで，携帯端末のリソース制約に対応している．Jadabsは軽量コンテナの携帯端末上への移植そのものの実現に主眼を置いており，それにともなうメカニズムの変化などを提案しているわけではない．本研究ではロケーションアウェア関心事の支援という新規の事例と，そのためのメカニズムを示していることに意義があると考えている．

BPELのスコープ

BPEL（Business Process Execution Language for Web Services）[20]は，XMLベースのワークフロー記述言語であり，複数のWebサービスを連携させることで，複雑なプロセスフローを定義することができる．BPELのスコープは例外を処理するためにアクティビティを構造化する仕組みである．障害ハンドラおよび補正ハンドラをスコープに対して定義することにより，適切なエラー処理とロールバックを行う．xBPEL[21]のようにビジネスプロセスがユビキタス環境において利用されることを想定する際には，適切なスコープ定義として本研究のロケーションスコープを利用することが有効であると考えられる．

7. むすび

本論文では，ポータビリティのあるロケーションアウェアアプリケーション開発を支援するロケーションスコープの提案を行った．ロケーションスコープはアスペクト指向開発を取り入れ主要関心事と位置条件に関するロケーションアウェア関心事とを分離して開発することを支援する枠組みである．抽象的な位置モデルを与え，それをアスペクト指向開発に直結させることで，ポータビリティの実現を図るものである．こうした試みは従来なされてきていないが，今後ユビキタス環境が浸透してゆくにつれ必要になっていくと考えている．

今後の方針としては，LBSのQoSなど詳細な要素を考慮すること，BPELJなどWebサービス周りの標準化技術との連携などがあげられる．xBPELの発想のように，ビジネスプロセスに人を参加者として取り入れた際に，人がトークン（token）を持つユビキタス環境での局面に限り，本研究の想定していたアプリケーションはプラグインのように動作することが望まれる．こうした実用的なシステムと組み込んだ際の評価は今後の課題である．また，ロケーションスコープのイベントに対してアプリケーションがサービス切替えなどを円滑にできるような支援などの拡張も今後の課題として考えられる．

参考文献

1) The Printer Working Group: Print Service Interface Version 1.0 Working Draft (2004). ftp://ftp.pwg.org/pub/pwg/ps/wd/wd-psi10-20040113.doc
2) Syukur, E., Cooney, D., Loke, S.W. and Stanski, P.: Hanging Services: An Investigation of Context-Sensitivity and Mobile Code for Localised Services, *MDM '04: International Conference on Mobile Data Management* (2004).
3) Kiczales, G., Lamping, J., Menhdhekar, A., Chris Maeda, C.L., Loingtier, J.-M. and Irwin, J.: Aspect-Oriented Programming, *OOPSLA '97: Object-Oriented Programming 11th European Conference* (1997).
4) aosd.net: Aspect-Oriented Software Development Community and Conference. http://aosd.net/
5) OMG: UML 2.0 superstructure specification (2005). http://www.omg.org/cgi-bin/doc?formal/05-07-04
6) McFaddpen, T., Henricksen, K., Indulska, J. and Mascaro, P.: Applying a Disciplined Approach to the Development of a Context-Aware Communication Application, *PerCom '05: International Conference on Pervasive Computing and Communications*, pp.300–306 (2005).
7) Monson-Haefel, R. and Chappell, D.: *Java Message Service*, O'Reilly & Associates, Inc., Sebastopol, CA, USA (2000).
8) Eugster, P.T. and Guerraoui, R.: Distributed Programming with Typed Events, *IEEE Software*, Vol.21, No.2 (2004).
9) 高田敏弘，青柳滋己，栗原　聡，光来健一，清水

奨, 廣津登志夫, 福田健介, 菅原俊治：Proximity Mining：センサデータ履歴からの近接性の発見, 第 6 回プログラミングおよび応用のシステムに関するワークショップ（*SPA 2003*）(2003).
10) Becker, C. and Durr, F.: On location models for ubiquitous computing, *Personal Ubiquitous Comput.*, Vol.9, No.1, pp.20–31 (2005).
11) Brumitt, B. and Shafer, S.: Topological World Modeling Using Semantic Spaces, *Location Modeling for Ubiquitous Computing Workshop at Ubicomp 2001* (2001).
12) SUN: Java Message Service Specification (JMS). http://java.sun.com/products/jms/
13) SOFA project: SOFA (Simple Ontology Framework API). http://sofa.projects.semwebcentral.org
14) Matsuzaki, K., Yoshioka, N. and Honiden, S.: Development Methodology for Location-Aware Mobile Agent, *MATA '04: International Workshop on Mobility Aware Technologies and Applications* (2004).
15) Shekhar, S., Chawla, S., Ravada, S., Fetterer, A., Liu, X. and Tien Lu, C.: Spatial Databases-Accomplishments and Research Needs, *IEEE Trans. Knowledge and Data Engineering*, Vol.11, No.1, pp.45–55 (1999).
16) Hu, H. and Lee, D.-L.: Semantic Location Modeling for Location Navigation in Mobile Environment, *MDM '04: International Conference on Mobile Data Management* (2004).
17) Henricksen, K. and Indulska, J.: A Software Engineering Framework for Context-Aware Pervasive Computing, *PerCom '04: International Conference on Pervasive Computing and Communications* (2004).
18) Ranganathan, A., Chetan, S., Al-Muhtadi, J., Campbell, R.H. and Mickunas, M.D.: Olympus: A High-Level Programming Model for Pervasive Computing Environments, *PerCom '05: International Conference on Pervasive Computing and Communications* (2005).
19) Roman, M., Hess, C., Cerqueira, R., Ranganathan, A., Campbell, R.H. and Nahrstedt, K.: Gaia: A middleware platform for active spaces, *SIGMOBILE Mob. Comput. Commun. Rev.*, Vol.6, No.4, pp.65–67 (2002).
20) BPEL: Business Process Execution Language for Web Services version 1.1. http://www-128.ibm.com/developerworks/library/specification/ws-bpel/
21) Chakraborty, D. and Lei, H.: Pervasive Enablement of Business Processes, *PerCom '04: International Conference on Pervasive Computing and Communications* (2004).
22) JBoss.com: JBoss AOP. http://www.jboss.com/products/aop
23) Spring Framework. http://www.springframework.org/
24) Frei, A. and Alonso, G.: A Dynamic Lightweight Platform for Ad-Hoc Infrastructures, *PerCom '05: International Conference on Pervasive Computing and Communications*, pp.373–382 (2005).

付　　録

ロケーションスコープ記述の XML Schema 定義を**図 11**，**図 12** に示す.

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" />

<xsd:element name="locationScopeDescription">
  <xsd:annotation>
    <xsd:appinfo>Location Scope Description</xsd:appinfo>
  </xsd:annotation>
  <xsd:complexType>
    <xsd:sequence>
      <!-- Location Topology Description-->
      <xsd:element name="locationTopologyDescription"
        type="LTD_type" minOccurs="1" maxOccurs="unbounded" />
      <!--Location Scope Element-->
      <xsd:element name="locationScopeElements" type="LSE_type"
        minOccurs="1" maxOccurs="1" />
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>

<!-- Location Topology Description-->
<xsd:complexType name="LTD_type">
  <xsd:sequence>
    <xsd:attribute name="name" type="xsd:string" />
    <!--Specify end points in the topology-->
    <xsd:element name="targets">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element name="node" type="xsd:string" />
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
    <!--Specify proximity-->
    <xsd:element name="adjacent">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element name="base" type="xsd:string" />
          <xsd:element name="qualifier">
            <xsd:simpleType>
              <xsd:restriction base="xsd:string">
                <xsd:enumeration value="near" />
                <xsd:enumeration value="anywhere" />
              </xsd:restriction>
            </xsd:simpleType>
          </xsd:element>
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
  </xsd:sequence>
</xsd:complexType>
```

図 11　ロケーションスコープ記述の XML スキーマ
Fig. 11　XML schema for location scope description.

```
<!--Location Scope Element-->
<xsd:complexType name="LSE_type">
  <xsd:sequence>
    <xsd:element name="id" type="xsd:int" />
    <!--Entering the area where the location
                                  topology is satisfied-->
    <xsd:element name="in" maxOccurs="1">
      <xsd:complexType>
        <xsd:sequence>
          <!--Callee activity-->
          <xsd:element name="reaction" type="xsd:string"/>
          <!--Relationship with the caller activities-->
          <xsd:element name="lifecycle">
            <xsd:simpleType>
              <xsd:restriction base="xsd:string">
                <xsd:enumeration value="spawn" />
                <xsd:enumeration value="call" />
                <xsd:enumeration value="suspend" />
                <xsd:enumeration value="resume" />
                <xsd:enumeration value="euquire" />
                <xsd:enumeration value="terminate" />
              </xsd:restriction>
            </xsd:simpleType>
          </xsd:element>
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
    <xsd:element name="out" />
    <xsd:element name="update" />
  </xsd:sequence>
  <xsd:complexType name="activities"
                       minOccurs="1" maxOccurs="1">
    <xsd:sequence>
      <xsd:element name="activity" minOccurs="1" />
    </xsd:sequence>
  </xsd:complexType>
</xsd:complexType>
```

図 12 ロケーションスコープ記述の XML スキーマ
Fig. 12 XML schema for location scope description.

(平成 17 年 4 月 5 日受付)
(平成 17 年 10 月 11 日採録)

松崎 和賢

1979 年生．2002 年東京大学理学部情報科学科卒業，2004 年東京大学大学院情報理工学系研究科コンピュータ科学専攻修士課程修了．同年同博士課程進学，文部科学省国立情報学研究所知能システム研究系リサーチアシスタント，エージェント技術の研究に従事，現在に至る．日本ソフトウェア科学会学生会員．

吉岡 信和（正会員）

1971 年生．1993 年富山大学工学部電子情報工学科卒業．1998 年北陸先端科学技術大学院大学情報科学研究科博士後期課程修了．博士（情報科学）．同年（株）東芝入社．2002 年より文部科学省国立情報学研究所に勤務，2004 年より同研究所特任助教授，エージェント技術の研究，ソフトウェア工学の研究に従事，現在に至る．日本ソフトウェア科学会会員．

本位田真一（正会員）

1953 年生．1978 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．（株）東芝を経て 2000 年より文部科学省国立情報学研究所教授，2004 年より同研究所研究主幹を併任．2001 年より東京大学大学院情報理工学系研究科教授を併任，現在に至る．2002 年 5 月〜2003 年 1 月英国 UCL ならびに Imperial College 客員研究員（文部科学省在外研究員）．2005 年度パリ第 6 大学招聘教授．工学博士（早稲田大学）．1986 年度情報処理学会論文賞受賞．エージェント技術，オブジェクト指向技術，ソフトウェア工学の研究に従事．IEEE，ACM，日本ソフトウェア科学会等各会員．本学会理事．