センサリッチ携帯端末におけるミドルウェア開発
―状況依存型アプリケーションの開発効率向上―

1. 背景

無線通信技術の発達や計算機の小型･高性能化によって計算機やセンサは日常空間に遍在するようになり、日常生活で我々を取り巻くサービスは実空間の情報（コンテキスト）を取得することが可能になる。このような環境はユビキタスコンピューティング環境と呼ばれ、そこではコンテキストを利用してユーザの状況（ユーザコンテキスト）に適応したサービスを提供するコンテキストアウェア技術が重要となる。

また、PDA や携帯電話のようなパーソナルデバイスは、さまざまなコンテキストを知覚できるように拡張されつつある。パーソナルデバイスはユーザと密接に関係しているため、状況に応じ適切に振舞うことが期待される。しかし、パーソナルデバイスを用いたコンテキスト解析とセンサの適性に関する研究例や、解析したコンテキストを用いたアプリケーション例、解析結果の分解能まで踏み込んだ研究例が少ないなど、パーソナルデバイスを用いたコンテキスト取得に関する考察は多くない。そのため、センサを用いたコンテキストアウェアなアプリケーション開発の複雑さは増加傾向にある。

2. 目的

本プロジェクトでは、15 種類のセンサを搭載したセンサリッチ携帯端末 Muffin を用いてコンテキスト取得の方法を探る。そして、コンテキストアウェアなアプリケーションの開発者の負担を軽減するためのミドルウェアを開発することが本プロジェクトの目的である。

3. 開発の内容

ミドルウェアは大きく分けて 3 つのコンポーネントから構成される。センサデータを解析しコンテキストを生成する Worker、Database と Worker の通信を担い、アプリケーションにイベント通知を行う Daemon、そしてコンテキストを貯蓄する Database である。図 1 にミドルウェアのアーキテクチャを示し、各コンポーネントの詳細な開発内容を下記に記す。

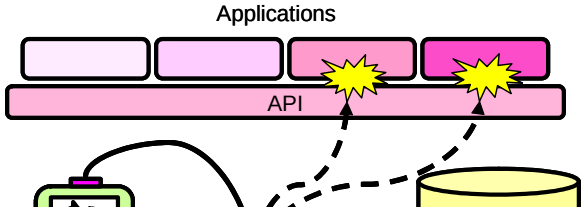

図 1 ミドルウェアのアーキテクチャ

解析モジュール(Worker)の開発

本プロジェクトを通じた経験から、解析 Worker を実装する複雑さを軽減するために、1 つの Worker が行う解析や出力結果をできるだけ単純化し、コンテキストの抽象化を段階的に行うことが好ましい。例えば、「ユーザがデバイスを見ているかどうか」を判定する Seeing Worker は、「ユーザがデバイスを持っているかどうか」を解析する Holding Worker と、「デバイスの向き」を解析する Topside Worker の解析結果からコンテキストを判定する。Seeing Worker 内で「ユーザがデバイスを持っているかどうか」と「デバイスの向き」という 2 つの解析を行うことは可能だが、本プロジェクトでは 2 つの Worker に機能分割した。機能単位ごとに Worker を分ける利点は、他の Worker からも解析結果を参照できるようにし、Worker の汎用性を高めている点である。このように、汎用性がある Worker を数多く実装することで、1 つあたりの Worker を実装する複雑さを軽減できると期待される。

本プロジェクトでは、コンテキストを解析する Worker を実際に 11 種類実装した。中でも、様々な状況・環境に応じ最適な解析方法を提供するために、「ユーザが歩いているかどうか」を取得する Worker を複数種類実装した。これらの Worker は、それぞれ異なるデバイスのセンサ・解析方法を用いて解析を行う。1 つのコンテキスト取得に対し、複数種類の Worker を実装した理由は、Worker それぞれに長所・短所があり、一概に 1 つの Worker が良いとは決められないからである。例えば、GPS を用いた解析 Worker は緯度・経度が得られるなど精度・分解能は高いが、ユーザが GPS 衛星を受信できる時のみ有効であり、GPS 衛星を受信できない場合は他のリソースを用いる解析 Worker が必要である。また、Muffin のみを用いた解析では、ユーザが Muffin から手を離した瞬間から解析結果に対する信頼性は低くなるため、ウェアラブルデバイスからセンサ値が入力される解析 Worker の重要性は増すが、ウェアラブルデバイスは、ユーザの様々な動きを捉らえてしまうため、扱うのは難しい。それぞれの長所・短所を考慮した上で、複数のデバイスを組み合わせたコンテキスト提供が有効であると考えられる。すなわち、1 種類のコンテキストに対し複数種

の Worker を実装することが有効である。

データベース作成･設計

データベースには組み込み用データベース SQLite を用いた。SQLite は C 言語で記述されており、サイズが小さいのが特徴である。SQLite は x86 アーキテクチャ用のバイナリを生成するように開発されていたので、Muffin 上で動作させるために ARM 用のバイナリを生成するように Makefile を修正した。

SQLite はファイルベースのデータベースになっており、複数の Worker が同時にデータベースに対して書き込みや読み込みを行おうとすると、データベースがロックしてしまうという問題が生じた。そこで、データベースに対するアクセスは全て Daemon を経由して行う設計にした。

イベント通知システム(Daemon)の開発

Daemon の役割は 2 つある。1 つ目はコンテキストに変化があった際にアプリケーションに通知を行うことである。イベント通知はコールバック関数により実装している。そして 2 つ目は Worker と Database 間の通信を制御することである。各 Worker および Daemon は独立したプロセスとして実装しているので、Worker および Daemon 間でデータを通信するにはプロセス間通信をする必要がある。本ミドルウェアではこのプロセス間通信に名前つきパイプを用いた。

次にミドルウェアがアプリケーションに提供する API を表 1 に示す。アプリケーションはこれらの API を通してコンテキストを取得することができる。

表 1 ミドルウェアの提供する API

返り値	関数の説明
context_t*	*get_context(const char* subject)* 引数 subject に該当するコンテキストを返す。 例えばユーザが歩いているかどうかを知りたいときは、 get_context(User:Activity:Walking); を実行する。
int	*add_event_handler(const char* subject, void (*handler)(char*))* 引数 subject に該当するコンテキストに変化があった際に、第 2 引数の関数ポインタのアドレスにイベントハンドラを実行するように登録を行う。
void	*remove_event_handler(void)* イベントハンドラを削除する。

4. 従来の技術（または機能）との相違

従来はこのようなミドルウェアがなかったため、センサを用いた状況依存型アプリケーションの開発者がセンサデータの解析などを行っていた。しかし、アプリケーションの開発の本質ではないセンサデータの解析には多くの難しさが存在し、センサを用いた状況依存型アプリケーション開発を困難にしていた。

5. 期待される効果

我々の開発したミドルウェアを用いることで、センサを用いた状況依存型アプリケーションの開発者はセンサデータの解析を意識せずに、抽象度の高いコンテキストを取得することができる。このことにより、センサを用いた状況依存型アプリケーションの開発は容易になると期待される。

6. 普及（または活用）の見通し

我々の開発したミドルウェアは Muffin だけでなく、他のセンサデバイスでも扱えるような機器非依存なミドルウェアとなっている。世の中の実験ソフトウェアとして提示できる程度のものは完成したが、開発を進めていく中で新たな課題も見つかり、実際に普及するにはまだまだ解決すべき問題が多いと感じた。例えば、Worker の増加に伴うシステムパフォーマンスの低下を防ぐための Worker の起動制御や、センサデータの解析の再検討および改善などが挙げられる。これらの課題を克服することで、広く世の中に普及するソフトウェアとなるのではないかと思う。

7. 開発者名（所属）

花岡 健介 （早稲田大学大学院 理工学研究科 修士課程 1 年）
高木 綾子 （早稲田大学大学院 理工学研究科 修士課程 1 年）
田村 真浩 （早稲田大学大学院 理工学研究科 修士課程 1 年）