ウェアラブルコンピューティングのための
イベント駆動型ミドルウェア開発

１．要約

本プロジェクトでは，ウェアラブルコンピューティング環境におけるソフトウェア開発の問題点である，装着ハードウェアの多様性や機能の動的な変更への対応，ユーザ状況の定義など，プログラマにとって実装が困難であった機能をミドルウェアおよび関連ツールとして提供し，いくつかのイベント駆動型ルールを記述するだけで手軽に高度なウェアラブルコンピューティング向けアプリケーションを構築できる基盤ソフトウェア・ツール群を開発した．

２．背景及び目的

コンピュータとセンサを装着することで体験記録や人の能力拡張，便利な道案内などさまざまなアプリケーションが実現できるウェアラブルコンピューティングへの期待が高まっている．ウェアラブルコンピューティングは図１に示すように，コンピュータや頭部装着型ディスプレイ（ヘッドマウントディスプレイ），ハンディマウスなどの入力機器を装着することが一般的なスタイルとなっている．ウェアラブルコンピューティングでは，一般のモバイルコンピューティングやデスクトップコンピューティングと異なり，(1)装着するセンサの種類や入力デバイスなど装着する機器が人によって異なり，アプリケーションがその多様性に対応する必要がある，(2)多様なデバイスを長時間バッテリ駆動で使用するため，電源 ON にするデバイスを動的に切り替えるなど省電力を考慮したアプリケーションを構築する必要がある，(3)どこに行っても自分専用のコンピュータを使ってサービスを受けるため，その場その場でアプリケーションの入替えや更新を行う必要がある，(4)装着センサ等の情報からユーザの状態を高度に認識する必要がある，といった特徴がある．これらの特徴を備えたアプリケーションを作成することは技術的にも作業負荷的にも困難であるため，本プロジェクトではこれらの機能を有する，ウェアラブルコンピュータのためのミドルウェアおよびツール群である Wearable Toolkit の開発及び一般公開を目的とする．Wearable Toolkit を用いることで，センサやアクチュエータを扱うウェアラブルシステムをプログラミング知識の乏しいユーザで

図1 ウェアラブルコンピューティングのスタイル

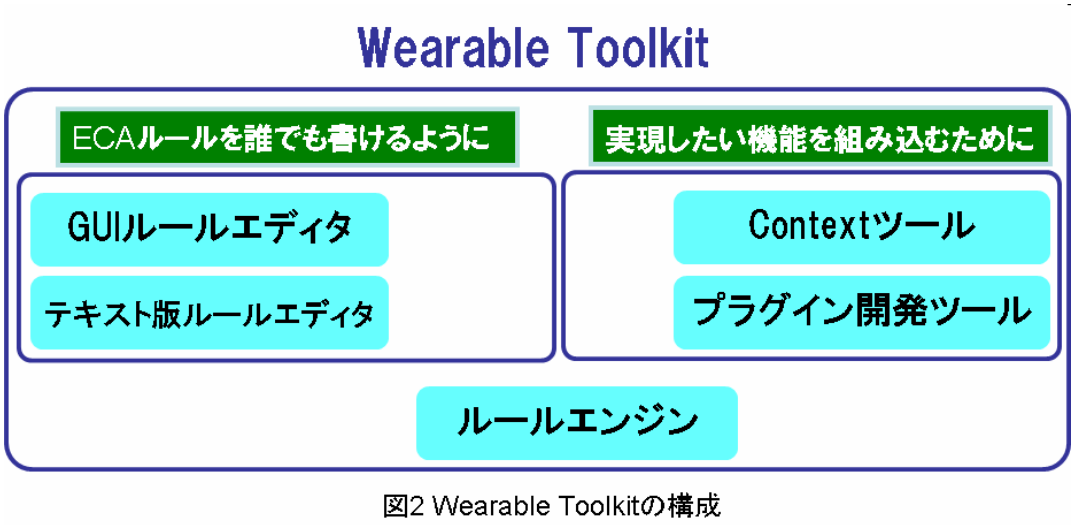

図2 Wearable Toolkitの構成

あっても容易に開発できる環境の提供を目指す．提案するミドルウェアは ECA ルールと呼ぶイベント駆動型ルールの集合でアプリケーションの動作を記述し，デバイス構成の変化や新たなサービスの受信をトリガとしてアプリケーションを記述できるため，上記の特徴を備えたウェアラブルシステムが容易に構築できる．

また，高度で特殊なシステム構築を行いたいユーザに対しては，システム自体がもつ機能を容易に拡張できるプラグインメカニズムを提供する．これらの仕様を一般に公開し，広く開発者を募ることで Wearable Toolkit の世界的な普及を目指す．

３．プロジェクト概要

本プロジェクトでは，前章で述べた目的を実現するために，図２に示すようなソフトウェアおよびツール群(総称して Wearable Toolkit)を開発した．

○**ルールエンジン**：アプリケーション動作の基盤となるエンジン．イベント駆動型ルールにより動作する Windows 用ミドルウェアである．ECA ルールと呼ぶシンプルなイベント駆動型ルールの集合でアプリケーションの動作を定義するため，プログラミングに慣れ親しんでいないユーザでも容易にウェアラブルシステムが構築できる．また，プラグインメカニズムを採用しているため，ルールの言語仕様自体が後に拡張可能である．

○**ツール群**：ウェアラブルシステムを構築するに当たって困難な点を解決するツール群を開発した．具体的には，センサ群からユーザの状況を知る機能をほぼ全自動で利用できるコンテキスト定義・認識ツール，ルールの動作を高度に解析できるデバッガ，ルールを容易に記述するためのルールエディタを開発した．

ルールエンジンに関しては，開発期間内に当初目標以上の機能をほぼ実現できた．また，GPS 処理や地磁気センサ処理などのプラグインを多数実装し，ルールを記述するだけでさまざまなアプリケーションが容易に実装可能であることを確認した．ツール群に関して，コンテキストツールおよびデバッガは，大阪万博公園における

スタンプラリーアプリケーション構築の際に利用し，実際にイベントで利用することでシステムが有効に動作していることを確認した．また，ルールエディタに関しては，初級者向けの GUI 版エディタおよび中・上級者向けのテキスト版ルールエディタを開発したが，GUI 版エディタは開発期間内に目標とする機能のうち一部のみを実現しており，実際にはテキスト版ルールエディタを用いてアプリケーションを実装した．

従来のウェアラブルコンピューティング向けツールキットは，センサ処理やネットワーク処理の API 関数群といった程度のものしかなく，イベント駆動型ルールを採用したもの，またプログラミング初級者でも高度なウェアラブルアプリケーションを開発させることを実現しているミドルウェアとしてはこの Wearable Toolkit が世界初であるといえる．

４．開発内容

前章に述べた Wearable Toolkit の開発の詳細に関して，ルールエンジン，プラグイン，ツール群(コンテキストツール，デバッガ，ルールエディタ)，作成したサンプルアプリケーションについて次節以降で詳細に解説し，最後に実際のアプリケーション構築の流れを具体的に解説する．

４．１．要求仕様

コンピュータを衣服のように装着するウェアラブルコンピューティングのアプリケーション例としては，常時ユーザの行動を測定して健康管理を行うシステム[1]や，バイクレースにおいて常時ユーザに情報を提示し続けるシステム[2]，状況の変化に応じて旅行者のためのナビゲーションを行うシステム[3][4]，看護婦の行動を認識してヒヤリハットをなくすためのシステム[5]などが挙げられる．例えば図 3 に示すバイクレース支援システムでは，バイクチームの監督がウェアラブルシステムを装着することで，戦略支援情報やトラブル情報を他人に見られることなく閲覧できる環境を実現している．ウェアラブルコンピュータを用いることで，従来の持ち歩き可能な機器と比較してユーザの生活により密着したサービスが提供できるようになりつつあり，ウェアラブルコンピューティング普及を考慮した新たなサービス提供モデルやプログラミングモデルが求められている．ウェアラブルコンピ

図3 バイクレース支援システムの画面

ュータを活用してサービスを提供する場合，たとえば「本屋に来て立ち読みをしたところ，この本を購入したいと思ったが手持ちのお金がない．次にこの本屋の前を通りかかったときにはこれを買うことを忘れないようにリマインダを提示させよう」といった要求が考えられる．このような要求は，その条件(次にこの本屋の前を通りかかる)や動作(リマインダを表示する)がその場で決定されるため，その場でそれらの要素をうまく記述し，システムに登録するメカニズムが必要となる．一方，このようなサービスの利用者・作成者は，一般ユーザを中心としてプログラミングに馴染んでいない者が多く，複雑な条件記述を行わせることは難しい．このような，ウェアラブルコンピューティング環境において特徴的なプログラミング形態である「その場でプログラミングする」例としては下記のようなものが挙げられる．

・本屋に来て立ち読みをしたところ，この本を購入したいと思ったが手持ちのお金がない．次にこの本屋の前を通りかかったときにはこれを買うことを忘れないようにリマインダを提示させることにした．

・迷いやすい場所に向かうので地図を表示したいが，常に表示されていると邪魔なので，立ち止まったときだけヘッドマウントディスプレイ上に地図を表示するようにした．

・ジョギングに出かけたが，医者に激しい運動を止められていることを思い出し，心拍が高かったり早く走っている期間が 1 分以上続いたりしたらアラートを出すようにした．

・今月はちょっと浪費気味なので，財布を出すたびに「無駄遣いするな」と表示することにした．

このようなサービスをウェアラブル環境においてプログラミングするためには，システムに下記の要素が必要となる．

(1) サービスを容易に記述できるプログラミングモデル
(2) システム稼働中の動的なプログラム追加機能
(3) コンテキストのその場定義機能

要求(1)は，ウェアラブル環境においてあらゆる場所でサービス定義を行うため，その場で簡単にサービス記述を行える必要があることを表す．上記のようなサービスは一般に，なんらかのイベント(本屋の前を通りかかる，立ち止まるなど)をきっかけとして，なんらかの動作(リマインダを提示するなど)を行うことが想定されているため，一般にイベント駆動型のプログラミングモデルが適している．また，複雑な構造をもつプログラミング言語はその場プログラミングには適しておらず，シンプルで簡単にサービス定義が可能な方式を採用する必要がある．要求(2)は，サ

ービス記述を行っている際にシステム自体を止めないことが重要であることを表している．ウェアラブルシステムでは多数のサービスが並行して動作しており，重要なロギングやセンシングを行っている場合も多い．したがって，頻繁に起こるであろうサービス追加のたびにシステムの再起動を行うようではいけない．要求(3)は，サービス開始のトリガとなる状況(以降コンテキスト)をその場で定義できる機能である．例で示したようなサービスにおいては，トリガとなる状況はバラエティに富んでおり，あらかじめ想定して用意しておくことは難しい．したがって，本屋の前を通りかかる，立ち止まる，といったコンテキストを現在の状況からその場で定義し，サービスのプログラミングにその場で活用できるような枠組みが必要となる．

本プロジェクトの目標は，この要求(1)～(3)を満たすウェアラブルシステムプラットフォームを構築することである．

４．２．ルールエンジンの開発

ルールエンジンは，提案するシステムプラットフォームの基盤となるソフトウェアであり，図４に示すように OS 上でミドルウェアとして動作する．ルールエンジンの特徴を下記に列挙する．

(1) イベント駆動型ルールによるアプリケーション開発

ウェアラブルコンピューティングのアプリケーションは「ユーザが交差点に近づいたら目的地に応じて曲がる方向の矢印を出す」といったような，簡単な動作の集合で表されることが多い．したがって，アプリケーションプログラミングは「交差点に近づく」といったイベント(発生事象)に対して，「目的地が設定されている」といったコンディション(条件)が満たされれば，「矢印を出す」といったアクション(動作)を実行するという ECA(Event-Condition-Action)ルールで記述することとしている．ECA ルールはデータベース技術のひとつであるアクティブデータベースにおいて古くから研究が進められてきた分野であり，ルールの実行連鎖の停止性検出やアクセス制御など信頼性の高いシステムを構築するための取り組みが多数行わ

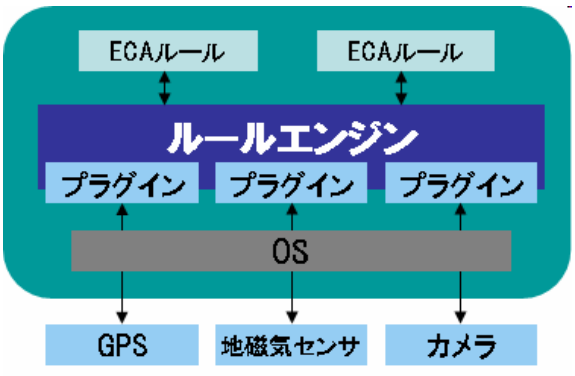

図4 ルールエンジンの動作イメージ

れている．本プロジェクトのリーダである寺田はもともとアクティブデータベース技術の研究者であり，ECA ルールを用いたアプリケーション開発手法がウェアラブルコンピューティング環境に適しているという着想を得て本プロジェクトを開始した．ECA ルールを用いることで，ユーザは「こういう状態になったとき」「こうしたい」という要求をそのままイベントとアクションに記述すればよく，機能の追加・削除・改変も，それぞれ ECA ルールを追加・削除・改変することで行える．また，これらの改変操作はシステム稼働中に行うことが可能であり，前章で示した用件(1)(2)を満たしたシステムとなっていると言える．このような部分的なシステム機能の入れ替えを行う場合，一般にルールの連鎖による無限実行や，悪意のあるルールによる攻撃が発生するが，提案システムでは無限実行の検出モデル[6]およびセキュリティ機構[7]の仕組みを提案しており，実利用において生ずる問題に関しても対応している．

(2) プラグインによる機能拡張

ウェアラブルコンピューティングにおいて使用するデバイスは次々と新しいものが登場している．そのような進歩や，新たなアプリケーション形態の登場に伴って，イベントやアクションに記述する内容も拡張できる必要がある．そこで，提案するミドルウェアではすべてのイベントやアクションを，プラグインにより定義するとしている．例えば GPS のプラグインをシステムに導入すると，イベントとして「GPSMOVE」が使えるようになり，位置が移動したことをイベントとしてアプリケーションが記述できる．このような形態を採用することで，必要な機能だけを利用するためシステムがコンパクトになる．また，プラグインは動的に追加・削除できるようになっており，場所に応じてプラグインをやり取りし，機能を変更することが可能である．本プロジェクトではすでに十個以上のプラグインを構築済みであり，必要に応じて開発して追加できるようになっている．プラグインは，C++，C#，VB.NET などさまざまな言語で記述できるようになっており，すでに存在するアプリケーションをプラグイン化することも容易である．

４．２．１．ルールエンジンで用いる ECA ルール

ECA ルールの定義イメージを図５に示す．また具体的な記述フォーマットは下記のとおりである．

DEFINE ルール ID [IN 所属グループ名のコンマ区切りリスト] [FOR ルール適用範囲]
[VAR 変数名 AS 変数の型]
WHEN イベントタイプ[(イベント対象)]
IF (左辺式演算子右辺式) (AND や OR を使用可能)
THEN
DO [戻り値格納先=] アクションタイプ(アクション内容)
FOREACH NEW.data
DO [戻り値格納先=] アクションタイプ(アクション内容)

END FOREACH

上記記述フォーマットで使用する語の詳細は以下のとおりである．

- **識別子**：ルール ID やグループ ID に使用する．英数字のほかに，いくつかの記号を使用できる．
- **シンボル**：ローカル変数名や CURRENT データなどに使用する．英数字とアンダースコアが使用できる．ただし，最初の文字に数字は使用できない．
- **数式**：整数や小数と+，-，*，/，%，^ といった演算子の組み合わせの総称である．数式は ECA ルールの処理中にシステムによって演算される．
- **数値**：整数や小数および#で挟んだ数式の総称である．#で挟んだ文字列はシステムによって演算され，整数や小数と同等に扱える．
- **文字**：文字列中で使用できる文字である．¥，%，’，#を使用する場合は¥ でエスケープしなくてはならない．また，¥r，¥n，¥t のように記述することで改行文字やタブを記述できる．
- **文字列**：文字の組合せの総称である．文字列中には，｛｝で挟むことによって変数を埋め込むことができる．
- **ローカル変数**：名前にはシンボルを使用する．ローカル変数の型が配列の場合はローカル変数名のあとに[1] のように 0 から始まるインデックスを指定するこ

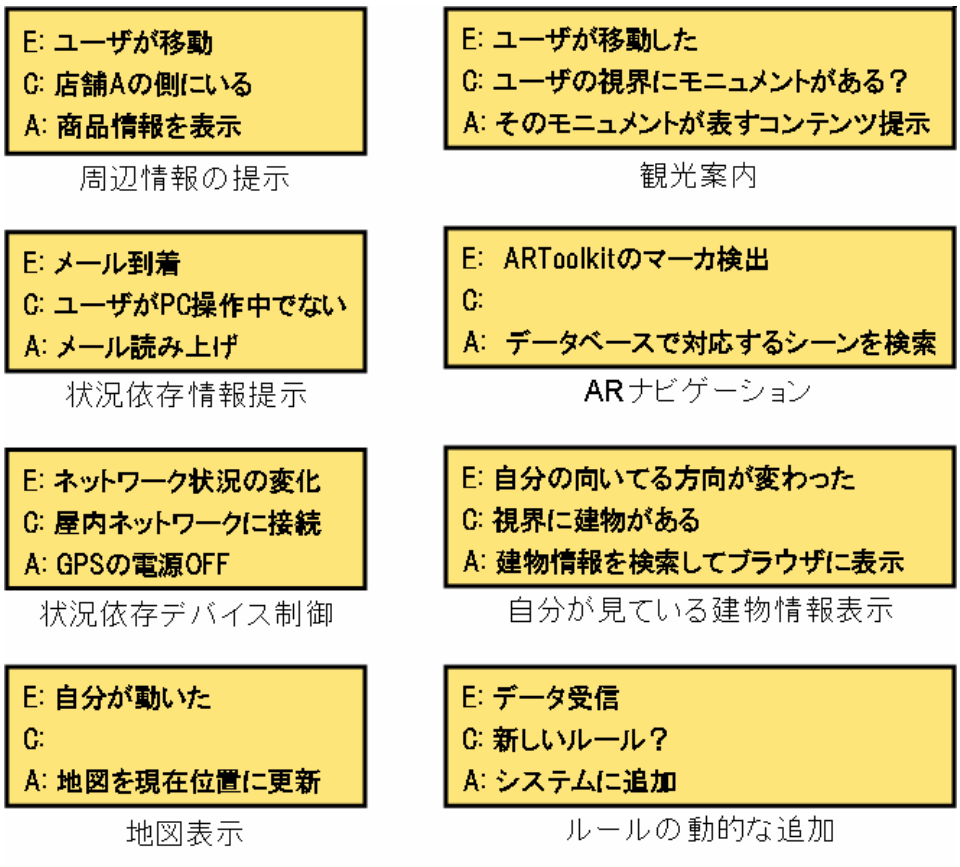

図5 ルール例

とで配列内のデータを参照・設定できる.

- **NEW データ・OLD データ**：NEW. または OLD. の後にシンボルである NEW データ・OLD データ名を記述することで参照できる. NEW データ・OLD データが配列の場合は, 後ろに[1] のように 0 から始まるインデックスを指定することで配列内のデータを参照できる.
- **引数**：アクションや CURRENT データでプラグインに引き渡す引数を示す. 変数をコンマで区切って指定する.
- **CURRENT データ**：CURRENT データの名前を示すシンボルの後にドットを記述し, さらにその後に文字列を記述することで参照・設定できる. ( )内に引数を記述することもできる. CURRENT_DATA.NAME は CURRENT_DATA(‘NAME’)のシンタックスシュガーであり, 同じ意味である.
  例えば CURRENT_DATA.NAME.TEXT(ARG1, ARG2)は
  CURRENT_DATA(‘NAME’, ‘TEXT’, ARG1, ARG2)とみなされる.
- **変数**：コンディションでの比較やアクションでの引数に使用する. 変数はローカル変数・NEW データ・OLD データ, CURRENT データ, 数値, ‘’で挟んだ文字列の総称である.

＊＊＊ BNF 定義一覧 ＊＊＊

<識別子> ::= (<英数字>| ‘_’ | ‘-’ | ‘/’ | ‘+’ | ‘.’ | ‘:’ | ‘;’ | ‘>’ | ‘<’ | ‘=’ | ‘@’ | ‘^’ | ‘#’ | ‘!’)*

<シンボル> ::= (<英字> | ‘_’) (<英数字> | ‘_’)*

<演算子> ::= ‘+’ | ‘-’ | ‘*’ | ‘/’ | ‘%’ | ‘^’

<数式> ::= <整数> | <小数> | ‘{’ <変数> ‘}’ | <数式> <演算子> <整数> | <数式> <演算子> <小数> | <数式> <演算子> ‘(’ <数式> ‘)’

<数値> ::= <整数> | <小数> | ‘#’ <数式> ‘#’

<文字> ::= <¥, %, ’, #以外>| ‘¥’ | ‘%’ | ‘’’ | ‘#’ | ‘¥r’ | ‘¥n’ | ‘¥t’

<文字列> ::= <文字>* | <文字列> ‘{’ <変数> ‘}’ <文字>*

<配列> ::= (‘[’ <数値> ‘]’)+

<ローカル変数> ::= <シンボル> <配列>?

<NEW データ> ::= ’NEW.’ <シンボル> <配列>?

<OLD データ> ::= ’OLD.’ <シンボル> <配列>?

<NEW・OLD データ> ::= <NEW データ> | <OLD データ>

<引数> ::= (<変数> (‘,’ <変数>)*)?

<CURRENT データ> ::= <シンボル> (‘.’ <シンボル>)* (‘(’ <引数> ‘)’)?

<変数> ::= <ローカル変数> | <NEW・OLD データ> | <CURRENT データ> | <数値> | ‘’’ <文字列> ‘’’

４．２．２．プラグインアーキテクチャ

すでに述べたように, 本システムはプラグインを追加することで機能の拡張が可能である. Wearable Toolkit のプラグインは, C++で記述することを想定した仕様となっているが, プラグインローダーと呼ばれるプラグインを用いることで, C#や

Java などさまざまな言語でプラグインを実装できる.

C++で記述するプラグインは, CreateWtkPlugin という関数をエクスポートし, IWtkPlugin1・IWtkPlugin2・IWtkPlugin3 等のインタフェースを実装した DLL（Dynamic Link Library）である. ルールエンジンは, アクションの実行などプラグインの機能を使用する場合は, 上記のインタフェースを通じてプラグインを呼び出す. また, ルールエンジンが公開する IWtkCore1 インタフェース等を用いて, プラグインはイベント発生の通知などをルールエンジンに対して行う.

C++以外の言語を用いてプラグインを開発する場合, そのプラグインを読み込むプラグインローダーの仕様に従って開発を行う.

Wearable Toolkit 開発プロジェクトでは, プロジェクトメンバにより下記のプラグインがすでに実装済みである.

- ECA ルールパーサープラグイン：ECA ルールを解析する
- 基本機能プラグイン:タイマなどの基本的名処理を実現する
- デバッガプラグイン：デバッグの機能を提供するン
- 基本 GUI プラグイン：ユーザが定義した GUI を使えるようにする
- QR コード読み取りプラグイン：QR コードを読み取る
- Web ブラウザプラグイン：Web ブラウザの表示や制御を行う
- シリアル通信プラグイン：シリアル通信を行う
- GPS プラグイン：GPS を用いて位置取得や移動検知を行う
- デバイス列挙プラグイン：システムに接続されているデバイスを列挙する
- データベースプラグイン：データベース操作, および値の変化検知を行う
- ネットワークプラグイン：TCP/IP, UDP/IP 通信を行う
- SMTP プラグイン：メールを送信する
- システム情報プラグイン：システム情報を取得する
- JOG ダイヤルプラグイン：JOG ダイヤルコントローラからの入力を受け取る
- カメラプラグイン：Web カメラからの入力を処理する
- Felica プラグイン：Felica カードの読み取りを処理する
- IRC プラグイン：IRC サーバに対してコマンドを送信する
- 地磁気センサプラグイン：地磁気センサにより方向を取得する

４．２．３．ルールエンジンの実装

ルールエンジンは, 図 6 に示すように, イベント管理部, ルール・グループ管理部, プラグイン管理部およびルール仮想マシンからなる. 以下, それぞれの機能について説明する.

・イベント管理部：プラグインによって通知されたイベントの管理を行う. 発生した

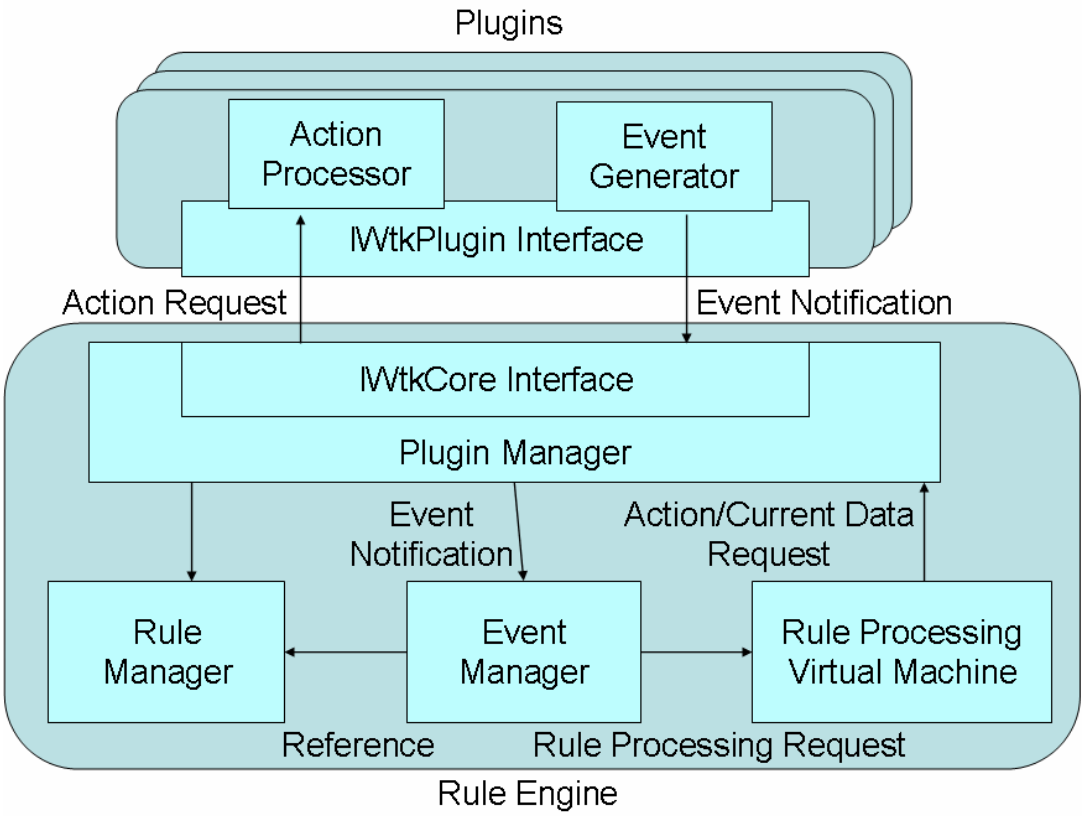

図6 ルールエンジンの構成

イベントはイベント管理部がもつイベントキューに追加される．イベント管理部はキューから順にイベントを取り出し，そのイベントに適合するルールをルール管理部から取り出してルール仮想マシンに渡し，ルールの処理を依頼する．

- ルール管理部：システム内のルールを管理する．プラグインからのルールの追加・削除要求およびグループの追加・削除要求などに応じてルールの処理を行う．さらに，イベント管理部からの要求に応じてルールを提供する．
- プラグイン管理部：プラグインのロードやアンロードを行い，さらにロードされているプラグインの制御を行う．ルール処理において発生するプラグインの呼び出しは，すべてプラグイン管理部を通して行われる．
- ルール仮想マシン：イベント管理部からの要求に応じてルールを処理し，コンディションが満たされているルールに対しては，ルールに記述されたアクションを処理するプラグインに対し，プラグイン管理部を通してアクションの実行を要求する．仮想マシンは独自の命令のみを実行可能であり，ルールをシステムに追加する際に仮想マシンが動作できるようにコンパイルすることでルールの判定処理を高速化する．ルールを仮想マシンが実行する命令にコンパイルしてから実行することで，ルールの動作に異常があった場合にルールの記述ミス・解析ミスであるかプラグインのエラーであるのかを容易に判定でき，原因の追求が容易になる．また，ルールをコンパイルして仮想マシンで実行することにより，特定のルール記述フォーマットに依存せずさまざまなルール記述フォーマットに対応できる．ルールのコンパイルはルール解析用プラグインが行うため，言語仕様の改良や新たな言語使用への対応を容易に行える．ルールの処理中には文字列中に変数を埋め込むなど，文字列の

結合処理を多く行うことが想定されるため，仮想マシンは文字列の結合処理を高速に行えるような構造・命令をもつ．

４．３．コンテキストツールの開発

要求仕様の節で述べたように，Wearable Toolkit において，(1)サービスを容易に記述できるプログラミングモデル，(2)システム稼働中の動的なプログラム追加機能，(3)コンテキストのその場定義機能，の 3 つを実現することがプロジェクトの目的である．このうち，Wearable Toolkit のルールエンジンを利用することで，要件(1)，(2)は実現できている．一方，要求仕様の節で挙げたような例を実現するためには，あらかじめ用意されたイベントだけでなく，容易にその場で新たなコンテキストを表現するイベントを定義できる必要がある．

多くの従来研究においても述べられているとおり，コンテキストとはなんらかの「状況」を表すものであり，「歩行中」「自転車に乗っている」「起きている」「メール着信」「ある時点から 10 分後」などといったようにさまざまな表現が可能である．そしてこれらのコンテキスト，特にウェアラブルコンピューティングにおけるサービスで利用されるコンテキストは単一または複数のセンサの特徴量から決定される場合が多い．ここでセンサとは，GPS や加速度センサなどハードウェア的なセンサに加え，メール着信を検出するメーラなど広い意味でのセンサを含む．特徴量とはセンサから出力されるそのままの値や，その値の一定時間における平均値・分散値などを表す．例えば「現在地が自宅」というコンテキストは具体的に，GPS から得られた緯度および経度の現在瞬間値が，あらかじめ登録されている自宅の緯度経度と一定値以上近いこと，と言い表せる．このように，あるコンテキストを定義するためには，装着しているセンサ群の中からいくつかのセンサを選び，コンテキストとして利用する特徴量およびその特徴量を計算する基となるデータの範囲を指定するというプロセスが必要となる．そこで，Wearable Toolkit では，図 7 に示すようなコンテキスト登録ツールを提供している．

このツールは，システム稼働中常にすべてのセンサから得られる値をモニタリングしている．ユーザは過去一定時間分のすべてのセンサデータ値を閲覧できるため，現在の状況をコンテキストとして登録したくなったときは，学習に用いる部分を選択し，「平均値」「変化量」などどの特徴量をコンテキストとして登録するかを選択する．複数センサや，単一センサにおける複数特徴量を組み合わせて登録できるため，柔軟にコンテキストを表現することが可能となっている．たとえばユーザが「次にここに来たときにアラートを出すために，ここという場所をコンテキストとして登録しよう」と考えた場合，このツールを呼び出して，GPS の X 値，Y 値における最新の部分の「現在値」を登録すればよいことになる．また，「走っているとき」というコンテキストを登録したい場合，しばらく走った後このツールを立ち上げて，走っている部分のデータから各加速度センサの X，Y，Z における適当な範囲を選択し，その「平均値」「分散値」両方を登録すればよいことになる．図 8 を用いて説

図7 コンテキストツールの概観

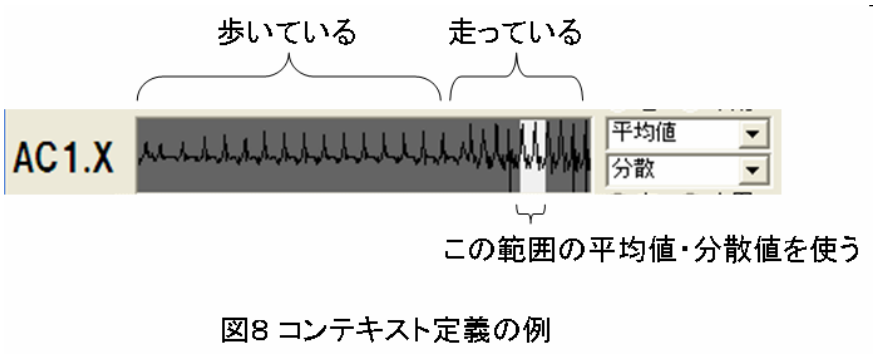

図8 コンテキスト定義の例

明すると，コンテキストツールを立ち上げた際，右腕につけた加速度センサの X 値は，歩いているときと走っているときで図のように変化していることがわかる．そこで，そのうち走っている部分のここでは 2 周期分を対象とし，その平均値と分散値を選択している．このとき，その 2 周期分は特徴量の計算に使われるだけでなく，実際に認識を行う際のウインドウサイズとしても利用される．また，不完全ではあるが，上記に説明したウインドウサイズや特徴量を自動的に決定する機能も備えている．

1 つ以上のコンテキストが登録されていた場合，このツールは特徴量の計算も同時に行い，コンテキスト発生を検知した場合にはルールエンジンに通知する．例えば，2 章に示した例のひとつである「次にこの本屋の前を通りかかったときにはこれを買うことを忘れないようにリマインダを提示させることにした」という機能を実現するためには，本屋にいる状態でコンテキストツールを呼び出し，前述のようにコンテキストを登録し，ルールエディタを用いて，イベントに先ほどのコンテキスト，

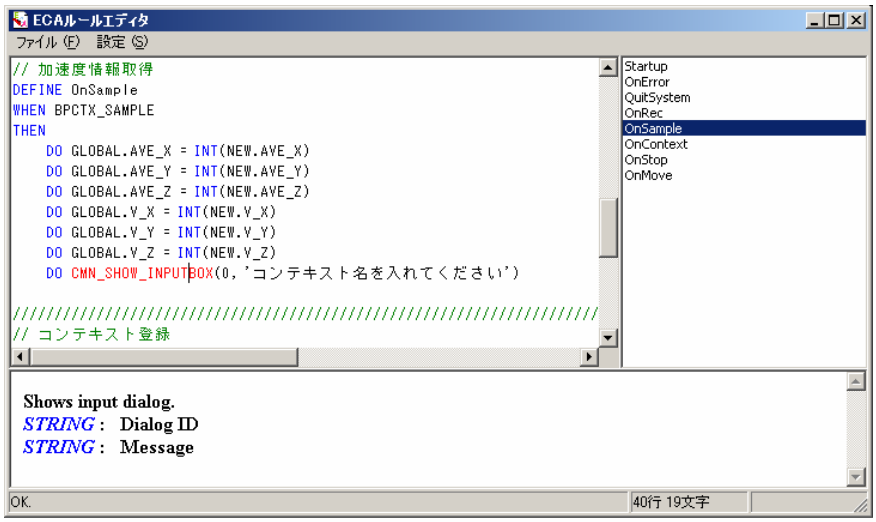

図9 ルールエディタ

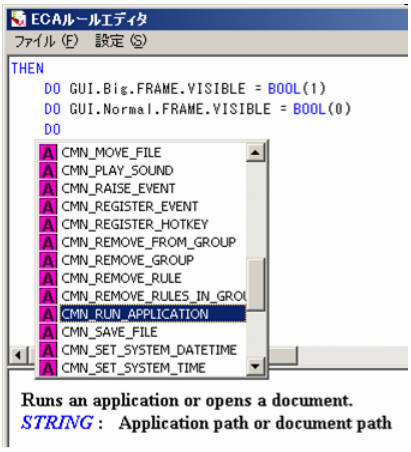

図10 入力候補の自動提示

アクションにメッセージ表示を設定すればよい．このように，ルールエンジンとコンテキストツールを用いることで，コンテキストの登録から実際のアプリケーション(ECAルール)の作成までを行うことができ，容易なプログラミングが実現できる．

４．４．ルールエディタの開発

Wearable Toolkit では，さまざまな ECA ルールを記述することで多様なアプリケーションを構築できるが，通常のテキストエディタ等を用いてルールを記述する場合，記述したルールを実際にルールエンジンに読み込ませるまで文法エラーがあるかどうかがわからないという問題がある．また，多くのプラグインを選択的に利用してアプリケーションを構築するため，利用できるイベントやアクションの把握が困難であるという問題がある．そこで，図 9 示すルールエディタを開発した．ルールエディタは以下の機能をもつ．

１．文法・イベント・アクションのヘルプ表示：ルール仕様で定められている予約語やイベント・アクションなどの入力候補を図 10 に示すようにポップアップ表示することで，ルールの記述効率を向上させる．また，ユーザが選択している項目

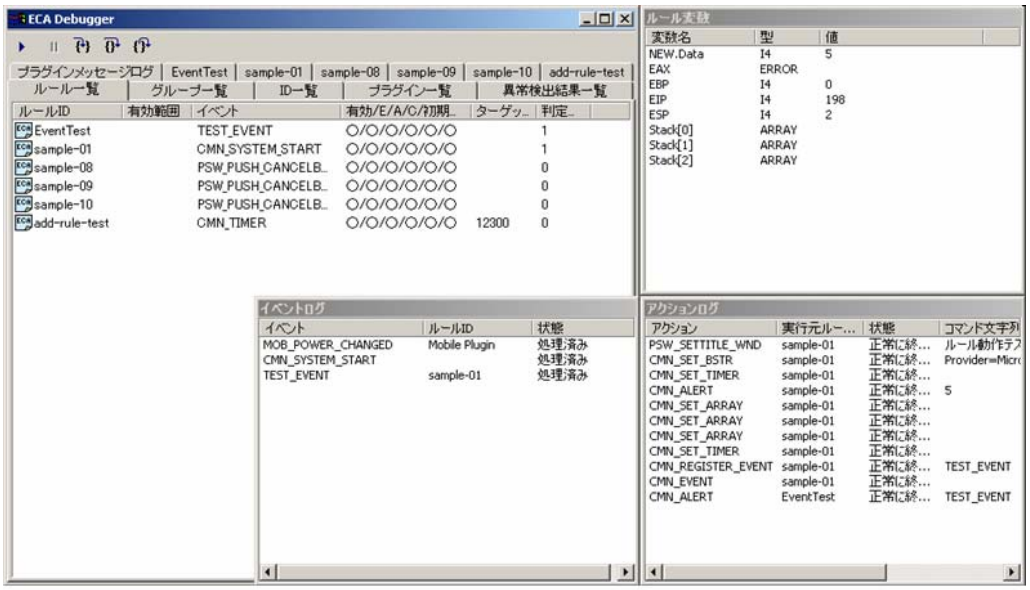

図11 デバッガ

の説明を表示し，ユーザがイベントやアクションの機能を覚えていなくてもルールを記述できるようにする．

Wearable Toolkit ではプラグインの追加によって新たなイベント・アクションが使用できるため，設定ファイルに使用するプラグインを記述することで，それらのプラグインが提供するイベント・アクションを自動的に取得し，適宜ポップアップで表示する．

2．ルールの文法エラーのリアルタイム表示：ルールエディタは入力中のルールが変更されるたびに構文を調べ，エラーがあればステータスバーに表示してユーザに警告する．

構築したルールエディタを用いることで，ECA ルールを容易に記述できる．

4．5．デバッガの開発

Wearable Toolkit を用いて効率的にアプリケーションを構築するためには，システムがどのようなルールやプラグインを読み込んでいるかなど，システムの内部状態を把握する機能が必要となる．そこで，図 11 に示すデバッガを構築した．デバッガはルールエンジンのプラグインとして動作し，以下の機能をもつ．

1．システムの内部状態の表示：システム内部の状態を把握するため，システム内に読み込まれているルール・プラグインの一覧およびシステムに登録されているイベント・アクションの一覧を表示する．

2．システム動作のトレース：発生したイベントおよび実行されたアクションの一覧を表示し，プラグインやルールが正常に動作しているか確認できるようにする．

また，個別ルールの任意の場所にブレークポイントを設定できるため，ルールの判定処理やプラグイン呼び出しの整合性を確認したり，エラーの原因を特定できるようになる．

構築したデバッガを用いることで，アプリケーション開発者はさまざまな情報を閲覧して，サービスのテストおよびデバッグを容易に行える．

４．６．アプリケーション構築例

本節では，実際にイベントで利用したウェアラブルシステムの実装を例に，Wearable Toolkit を使ったシステム構築の流れを説明する．

[対象となるイベント]

モバイルネイチャーラリーin 万博公園

このイベントは，大阪万博公園自然文化園内に設営した 15 のチェックポイントのうち，5 つのチェックポイントを回るラリーである．参加者は各地に設置された QR コードを携帯電話またはウェアラブルシステムに装着されたカメラで読み取り，Web にアクセスし，クイズに答えていく．5 つのチェックポイントを回り終え，ゴールへ行くとクイズの正誤等を記したラリー結果がもらえる．また，クイズの点数等によって決定する順位に応じた賞品ももらえる．

[STEP1 ウェアラブルシステムの機能設計]

ユーザはスタンプ台紙を持ち歩くのではなく，自分のウェアラブルシステムをスタート地点で登録してゲームに参加する．また，各スタンプポイントには QR コードが貼り付けられており，その画像を装着したカメラで撮影することで，(1)そのスタンプポイントの説明，(2)そのスタンプポイントに関連したクイズ，(3)次のオススメポイント，が順に提示される．ユーザはクイズに答えながらポイントを回り，5 箇所のポイントを回り終えたらゴールする．ゴールでは，図 12 に示すような終了認定章が配布され，歩いたコースや問題の正答率，消費カロリーなどの情報がわかるようになっている．このようなシステムを用いることで，システム側はユーザがいつどの地点に到着したかが判別できる．したがって，前のポイントから現在のポイントへ到着するまでにかかった時間を測ることで，ユーザの歩く速度や，疲れて歩くのが最初より遅くなっているかどうかといった情報が計算できる．これらの情報を用いることで，「ある範囲の時間でラリーを終わらせたい」「効率よくポイントを回らせることでゲームの完遂者を増やしたい」といった制御が可能になる．

以上を踏まえ，本イベントで用いるウェアラブルシステムに必要な機能は下記の通

	1	2	3	4	5
問題文	フランスの国のはたの色は何色の組み合わせでしょうか？	「ニュージーランド人」や「ニュージーランドの」という意味の名前をもつ鳥はなんでしょうか？	クウェートはどこにありますか？	近くの小川にある飛び石の数はいくつですか？	近くの池の名前は何でしょう？
回答	①赤と白と青	①キーウィ	②西アジア	③36こ	③どんぐり池
正答	①赤と白と青	①キーウィ	②西アジア	③36こ	③どんぐり池

図12 ゴール認定証

りである

- GPS データのログを取る機能：ユーザの歩いた距離を把握するため
- 歩いている時間と立ち止まっている時間を判断する機能：ユーザが休憩している時間をカウントしないため
- ジェスチャによりシステムを制御する機能：容易にシステムを利用できるようにするため
- ジョグダイアルを用いてシステムを制御する機能：容易にシステムを利用できるようにするため
- QR コードの読み取りにより該当のページを開く機能：スタンプポイントの処理を行うため.

[STEP2: ウェアラブルシステムの機器構成決定]

STEP1 で決定した機能を実現するために必要なデバイス群を決定する. 装着している様子を図 13 に示す.

- ウェアラブル PC(Sony VAIO type U)
- ヘッドマウントディスプレイ
- 装着型カメラ
- GPS
- 加速度センサ
- ジョグダイアル型コントローラ

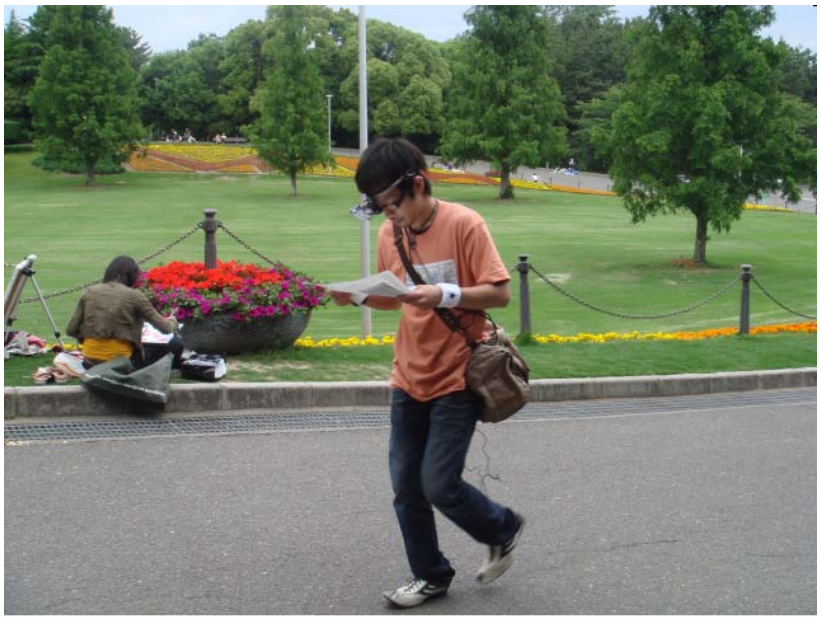

図13 ウェアラブルシステムを装着した様子

［STEP3：プラグインの決定］

STEP1 で決定した機能を実現するために必要なプラグインを集める．ないものは実装する．今回はすべて実装済みのプラグインを用いた．

- QR コード読み取りプラグイン
- ウェブブラウザプラグイン
- シリアル通信プラグイン
- ジョグダイアルプラグイン
- GPS プラグイン

［STEP4：ECA ルール記述］

ECA ルールを記述する．例として実際に利用したいくつかのルールを示す．

- QR コードを読み取ったときに該当する URL をブラウザに表示するルール

```
DEFINE QRJump
WHEN QR_DETECT
THEN
   DO CMN_SHOW_QUESTION ('jump','{NEW.CODE}へジャンプしますか？？')
   DO GLOBAL.QRurl = STRING('{NEW.CODE}')

DEFINE Yes_QRJump
WHEN CMN_ANSWER
   IF NEW.ID == 'jump' AND NEW.RESULT == -1
   THEN DO GUI.Main.Browser.URL = STRING('{GLOBAL.QRurl}')
```

・ ジョグダイアルに合わせてブラウザをスクロールさせるルール

```
DEFINE down
WHEN JOG_ROLLDOWN
THEN DO WEB_SCROLL_BY('Main', 'Browser', 0, -59)

DEFINE up
WHEN JOG_ROLLUP
THEN DO WEB_SCROLL_BY('Main', 'Browser', 0, 59)
```

・一定時間以上座っていたら注意を促すルール

```
DEFINE On_Sit_Timer
WHEN SERIAL_FORMED_DATA
THEN
  DO CMN_DEVIDE_STRING ('{NEW.DATA1}', 'sit', 1)
  DO CMN_SET_TIMER ('SIT_TIMER',60000)

DEFINE Off_Sit_Timer
WHEN SERIAL_FORMED_DATA
THEN
  DO CMN_DEVIDE_STRING ('{NEW.DATA1}', 'stand', 1)
  DO CMN_KILL_TIMER ('SIT_TIMER')

DEFINE Sit_Alert
WHEN CMN_TIMER (SIT_TIMER)
THEN DO CMN_ALERT ('そろそろ歩き出したらいかがでしょう？')
```

すべての機能を実現するために 25 個のルールが必要であった.

[STEP5：コンテキストの学習]
コンテキストツールを用い，システムが利用するコンテキスト(歩行中, 静止中，右手を上げている等)を学習させる．システム利用時にはコンテキストツールが状況認識を行い, ルールエンジンに仮想シリアルポート経由でコンテキストの変化を伝える.

[STEP6：デバッグ&テスト]
実際にシステムを利用して運用テストを行う. デバッガによってルールの発火状況や変数の変化を確認し, うまく動かない場合はルールをその場で修正して対処する.

以上の流れでシステムを構築した．実際にシステムを構築したのは，大阪大学情報科学研究科男子４回生１名であり，彼はほぼプログラミング経験が皆無であったにもかかわらず，上記システムを３日で完成させた．一般にはどのようなツールやミドルウェアを用いても，プログラミング経験の浅い者がこのようなシステムを実装するには多大な時間を要すると考えられ，Wearable Toolkit の有効性は明らかである．

５．開発成果の特徴

従来のウェアラブル向けツールキットは，センサ処理やネットワーク処理の API 関数群といった程度のものしかなく，イベント駆動型ルールを採用したもの，またプログラミング初級者でも高度なウェアラブルアプリケーションを開発させることを実現しているミドルウェアとしてはこの Wearable Toolkit が世界初である．また，状況認識のデータ選択や認識手法選択，ウインドウサイズの設定など，状況認識における各種の設定を自動化する機構を実現した初めてのシステムであるといえる．文献[8]によると，パラメータの設定によってその状況認識精度は数十％も変動するため，適切にパラメータ設定がされる仕組みは大変有用なものである．実際に，4.5 節で説明したアプリケーションの開発はプログラミング経験が皆無の学生１名によりほぼ１日で完了しており，コンテキストの学習や解析，システムプログラミングといった手間を省いた本ツールキットは高度なウェアラブルシステムを簡単に実現できることがわかる．

関連研究および関連システムとして，たとえば有名なツールキットとしては，米国 MIT で進められている MIThril プロジェクト[9]は，ウェアラブルコンピューティングにおけるハードウェア・ソフトウェアプラットフォームの構築を目指したものである．コンテキストアウェアシステムにおけるハードウェアの管理や，センサから抽出したデータの特徴量抽出およびコンテキスト認識を行う機能を API セットとして提供し，プログラマが容易にウェアラブルシステムを構築できる環境を目指している．また Context-toolkit[10]は，センサデータからコンテキストを得る処理を支援する．センサデータを直接取り扱いデータをカプセル化する context widget，複数の widget のデータを統合して抽象化する context aggregator，実際のコンテキスト計算を行う context interpreter の３層モデルを用いてコンテキスト認識を行うことで，使用するセンサの変化など末端のシステム変更の影響を吸収している．TEA System[11]は，TEA Board と呼ぶセンサが接続されたデバイスを持ち歩くことでデータを蓄積し，利用者の行動に応じてコンテキストのための閾値を自動生成する機構をもつ．LifePatterns[12]は，100 日間の自身の行動をカメラおよびジャイロセンサを搭載したウェアラブルコンピュータでモニタし，行動ログをある程度の数のコンテキストに自動的にクラスタリングする手法を提案した．R2 システム[13]では，コンテキストアウェアアプリケーション構築からセンサデバイスを隠蔽することが目的であるが，センサデータからコンテキストの定義が可能となっている．LifePatterns は，実験を行った段階であり，システムやプラットフ

ォームの提案は行われていない．また，そのほかのシステムは主な目的が，センサデータからコンテキストを抽出する部分の抽象化を行うことでアプリケーションプログラマの負荷を軽減することを目的としており，要求(1)～(3)は考慮されておらず，さまざまなユーザが手軽にウェアラブルシステムを構築できる枠組みはこれまで存在していなかった．

ウェアラブルコンピュータに関する研究は数多くなされており，プロトタイプシステムまで構築したものも多いが，そのほとんどは環境に特化されたシステムであり，一般にウェアラブルコンピュータが活用されているイメージとは遠い．一方，提案ミドルウェアを用いてシステムを構築したプロジェクトの場合，あくまでウェアラブルコンピュータを装着している一般ユーザが，サービスを受けるときにルールとプラグインを受け取り，その場のサービスを受けているというフレームワークになる．このような取り組みを活発化させることで，ウェアラブルコンピューティングを普段利用していると便利なのだ，という感覚が認知度を得て，本格的な普及へのステップとなることが期待される．

６．今後の課題、展望

現在，本プロジェクトにより開発した Wearable Toolkit のルールエンジンおよびツール群，ドキュメント等すべての情報は http://wearable-toolkit.com/にて公開されている．残る課題としては，さらなる実運用や他ユーザの使用をもとにエンジンおよび関連ツール群のブラッシュアップを行うことが挙げられる．また，作成したルールや開発したプラグインを自由にアップロード/共有できるリポジトリサイトの構築および運用も今後の課題となる．これらの課題を解決すれば，便利なプラグインが多数開発され，より便利なプラットフォームとなるだろう．

普及に向けての方策としては，上記ウェブサイトにて公開・アップデートを続け，多くの人々に使ってもらうことを目指す．具体的な広報活動としては，毎年 10 月に行われるウェアラブルコンピューティングの世界最大の国際会議および展示会である ISWC（International Symposium on Wearable Computers）にてデモンストレーションやチラシ配布を行う予定である．また，国内においても実際のシステム構築に Wearable Toolkit を利用し，積極的に広報する予定である．開発期間中に採り上げられた例としては，大阪万博公園にて QR コードを使ったスタンプラリーのシステムを開発し，そのシステム基盤として Wearable Toolkit を用いたものがある．このイベントは日本経済新聞および毎日コミュニケーションズに採り上げられ，本システムの普及の一助となるといえる．今後も数々のイベントでの実運用を予定しており，Wearable Toolkit の広報活動も継続的に行っていく予定である．さらに，Wearable Toolkit の内容を研究成果として論文発表するという取り組みも積極的に行っており，現時点で以下の 3 つの成果を挙げている．特に②は研究会の優秀論文賞を受賞した．

① 寺田 努, 宮前雅一, 福田登仁, ``コンテンツ再利用可能なイベント駆動型ナビゲーションシステムの開発と実運用,'' 情報処理学会研究報告(2007-EC-7), Vol. 2007, No. 37, pp. 31-38 (May 2007).

② 寺田 努, 宮前雅一, ``その場プログラミングの実現に向けて,'' 情報処理学会研究報告(2007-UBI-14), Vol. 2007, No. 46, pp. 1-8 (May 2007).

③ Tsutomu TERADA, Masakazu MIYAMAE, and Takahito Fukuda, ``An Event-driven Navigation System for Contents Reutilization,'' Proc. of the IADIS Interfaces and Human Computer Interaction (IHCI) 2007 conference, DVD-ROM (July 2007).

プロジェクト代表者の寺田は,特定非営利活動法人ウェアラブルコンピュータ研究開発機構において理事を務めており,数年間にわたってウェアラブルコンピューティングの普及・啓蒙活動に従事している. この活動を通してわかったことは, ITプロフェッショナルだけではなく, ファッションデザイナー, 音楽家, 舞踏家, 芸人などさまざまな分野の人々が,「ウェアラブルコンピューティングを活用したい」と考えていることである. 上記特定非営利活動法人では, 提案ミドルウェアを用いてこれらの人々に対しサポートを行っているが,マンパワーの不足から小さな活動にとどまっているのが現状である. ミドルウェア及び開発環境を広く公開し, 少しの知識でシステムが開発できるようになれば,あらゆる分野の人々がコンピュータを装着することでできる新たなコトにチャレンジするようになり,ウェアラブルコンピューティング普及への大きな足がかりとなるであろう. コンピュータの小型化や快適な装着といった技術分野は欧米人ではなく日本人こそが得意とする分野であるといえる. 日本発のミドルウェアとして, 提案するミドルウェアを世界に発信し, あらゆるところで使われるようになることが目的である.

７．開発者

代表者：寺田努（大阪大学講師）
共同開発者：宮前雅一（ATR 研究員）

関連Ｗｅｂサイト
Wearable Toolkit ウェブサイト：http://wearable-toolkit.com/
寺田努ウェブサイト：http://www-nishio.ise.eng.osaka-u.ac.jp/~tsutomu/