情報処理学会論文誌 Vol. 50 No. 6 1587–1597 (June 2009)

推薦論文

Wearable Toolkit：その場プログラミング環境実現のためのイベント駆動型ルール処理エンジンおよび関連ツール

寺田 努[†1] 宮前雅一[†2] 山下雅史[†3,*1]

近年の技術革新により，コンピュータを常時装着するウェアラブルコンピューティングが実用化されつつある．ウェアラブルコンピューティング環境ではユーザは高度に個人化されたサービスを求めるため，そのサービス定義をいつでもどこでも行いたいという要求が生じる．そこで筆者らの研究グループでは，ユーザがいつでもどこでも状況依存サービスを定義できる「その場プログラミング」を実現するために，Wearable Toolkit と呼ぶ枠組みを提案する．Wearable Toolkit はイベント駆動型ルールにより動作するルール処理エンジンおよび関連ツール群からなり，Wearable Toolkit を用いることでいつでもどこでもシステムの動作を止めることなくサービスの定義・削除・改変などの操作を行えるようになる．評価の結果，提案する枠組みは状況依存サービス提供のために有効に動作することが明らかとなった．

Wearable Toolkit: an Event-driven Rule Processing Engine and Tools for On-site Programming

Tsutomu Terada,[†1] Masakazu Miyamae[†2] and Masashi Yamashita[†3,*1]

Wearable computing environments, where a user wears a computer anytime and anywhere, are slowly becoming a reality because of the recent technological advancements. When a user acquires various services in wearable computing environments, the user wants to define a new service by himself. Therefore, in this research, we propose a new framework for constructing context-aware applications in wearable computing environments. Our framework is called *Wearable Toolkit*, which consists of an event-driven rule processing engine and tools for developing applications. By using our framework, we can define, delete, and customise services anytime and anywhere. From the result of evaluation, our system helps us to create context aware wearable services.

1. はじめに

近年の技術発展や機器の小型化により，人々は携帯電話などの情報機器を常時持ち歩くようになった．最近では，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に対する注目が高まりつつあり，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]，自転車などのメンテナンス作業を支援するシステム[6]などウェアラブルコンピュータを活用したさまざまなシステムが研究開発されている．たとえば筆者らが開発した，**図 1** 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にウェアラブルシステムを装着させることで，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るため，ウェアラブルコンピューティングのための新たなサービス提供モデルが求められている．

本研究では特に，ウェアラブルコンピューティング環境におけるエンドユーザプログラミングについて考える．ウェアラブルコンピ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

†1 神戸大学大学院工学研究科
Graduate School of Engineering, Kobe University
†2 ウエストユニティス株式会社
Westunitis Co., Ltd.
†3 大阪大学大学院情報科学研究科
Graduate School of Information Science and Technology, Osaka University
*1 現在，三菱スペース・ソフトウエア株式会社
Presently with MITSUBISHI SPACE SOFTWARE CO., LTD.
本論文の内容は 2008 年 7 月のマルチメディア，分散，協調とモバイル（DICOMO2008）シンポジウムにて報告され，同プログラム委員長により情報処理学会論文誌ジャーナルへの掲載が推薦された論文である．

図 1　ウェアラブルシステムのバイクレースでの活用
Fig. 1　A wearable system for supporting motorbike races.

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．

本研究では，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その場でプログラミングを行える環境の実現を目指し，Wearable Toolkit と呼ぶ枠組みを提案する．Wearable Toolkit はイベント駆動型ルールの処理エンジンおよび関連ツール群からなり，ユーザは状況依存サービスの知識を持たなくてもウェアラブルコンピューティングのサービス定義がいつでもどこでも行えるようになる．以下，2 章ではその場プログラミングに対する要求を具体的な例をあげながら説明し，関連する研究について述べる．3 章でその場プログラミングを実現するルールエンジンおよび関連ツールについて述べ，4 章では実運用および評価について説明する．最後に 5 章で本研究をまとめる．

2.　その場プログラミングに対する要求事項

ウェアラブルコンピューティング環境において,「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あげられる．

- 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，つね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．
- 座りっぱなしは体に悪いので，2 時間以上座っていたら体を動かすようにリマインダを表示させることにした．
- ウェアラブルコンピュータを用いて服の点滅を制御するダンスパフォーマンスの練習中，回転時に服を派手にすれば目立つと思い，回転時の点滅パターンを変更した．
- 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，心拍が高い期間が 1 分続いたらアラートを出すようにした．
- ある人からメールが届いてヘッドマウントディスプレイ上に表示されたが，この人からのメールは大事ではないので以降表示しないようにした．
- 今月は浪費気味なので，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．
- 妙にうなずく癖があると言われたので，1 日うなずく回数をカウントすることにした．

以降，上記各アイテムのようにアプリケーションの一部を構成する機能をサービスと呼ぶ．このようなサービスをウェアラブルコンピューティング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，複雑な構造を持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ならない．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．

本研究の目標は，この要求 (1)〜(3) を満たすシステムプラットフォームを構築することである．一方，ウェアラブルコンピューティング環境やユビキタスコンピューティング環境において，コンテキストアウェアネスアプリケーションを構築するためのツールキットに関する取り組みはこれまでにも数多く行われている．代表的なものとして，米国 MIT で進められている MIThril プロジェクト[7] 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[8] は，センサデータをカプセル化する context widget，複数の widget を統合する context aggeregator，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．LifePatterns[9] 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．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) は考慮されていない．例示によりコンテキストを定義することを狙った研究としては a CAPpella システムが提案されている[10]．しかし，このシステムでは例示を行った後に，その状況を登録するためのウインドウ選択やセンサ選択の処理が煩雑であり，要求 (3) を満たさない．また，状況定義後のプログラミングモデルや動的なプログラム更新などについては言及されておらず，要求 (1)，(2) も満たしていない．

ウェアラブル/ユビキタス以外の研究分野においてもコンテキストの記述は重要になりつつあり，特に Semantic Web[11] 関連のプロジェクトにおいて，コンテキストの抽象化や分割などが大きなトピックとなっているが，ウェアラブルコンピューティングにおけるエンドユーザプログラミングへの適性は考慮されておらず，要求 (1)〜(3) を満たしていない．

3. Wearable Toolkit

本研究では，前章で述べた 3 つの要求を満たすウェアラブルコンピューティングプラットフォームである Wearable Toolkit を提案する．Wearable Toolkit は**図 2** に示す構成となっており，機能の動的追加が可能なルール処理エンジンおよび，ルールを定義するためのルールエディタ，アプリケーション画面を Flash などを用いて作成する GUI 作成ツール，デバッガ，コンテキスト定義ツール，機能拡張のためのプラグイン SDK などのツール群からなる．

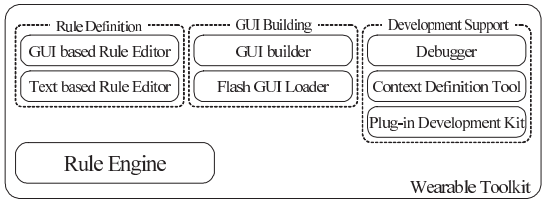

図 2　Wearable Toolkit の構成
Fig. 2　A structure of Wearable Toolkit.

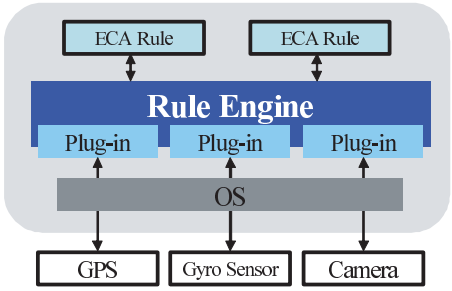

図 3　ルール処理エンジンの処理イメージ
Fig. 3　The processing image of Rule Engine.

一般ユーザがその場プログラミングを行うにあたっては，これらの要素のうちルール処理エンジン，コンテキスト定義ツール，ルールエディタを利用する．以下，詳細に説明する．

3.1 ルール処理エンジン

ルール処理エンジンは Wearable Toolkit の核となるモジュールである．**図 3** に示すように，ルール処理エンジンは WindowsOS 上でミドルウェアとして稼働し，プラグインを通して装着型センサや他のデバイスを制御する．その動作は**図 4** に示す構文に基づき，発生する事象（Event），実行条件（Condition），実行する動作（Action）の 3 つを一組とした ECA ルールにより記述する．ECA ルールを用いることで，ユーザは「ある状態になったとき」「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，

```
DEFINE Rule-ID
    [FOR Scope]
    [VAR Variable-name AS Variable-type]*
WHEN Event-type [ (Target-of-event)]
IF       Conditions
THEN DO Actions
```

図 4　ECA ルールのフォーマット
Fig. 4　Syntax of ECA rule.

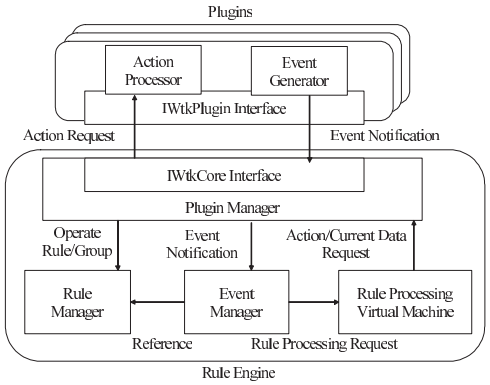

図 5　ルール処理エンジンの構成
Fig. 5　The structure of rule engine.

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，前章で示した要求 (1)，(2) を満たす．イベントやアクションに記述できる内容は，プラグインにより自由に拡張可能である．たとえば GPS 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，ユーザの移動をイベントとしてサービスが記述できる．**図 5** に詳細なルール処理エンジンの構成を示す．ルールエンジンでは，プラグインから通知されたイベントをイベント管理部が受け取り，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す．ルール仮想マシンではルール処理を行い，プラグイン管理部を通してプラグインにアクションの実行を依頼する．プラグイン管理部ではプラグインのロード/アンロードを行い，ロードされたプラグインを制御する．各プラグインは CreateWtkPlugin という関数をエクスポートし，IWtkPlugin などのインタフェースを実装した DLL（Dynamic Link Library）であり，イベントを発火させるルーチンおよびアクションに対応する関数が用意されている．

表 1　実装したプラグイン
Table 1　Implemented plug-ins.

名前	機能
Common	タイマなどの汎用機能
GUI	ユーザ定義 GUI の管理
Database	データベース処理
GPS	GPS による移動検知
System info	PC 状況の取得
Multimedia	動画や静止画の処理
Serial Com	シリアル通信
Camera	カメラ処理
Network	ネットワーク送受信
Mail	メール送受信
Browser	ウェブブラウザ制御
Map view	地図表示
VCode	ビジュアルマーカ認識
RF-ID	RF-ID 認識
Direction	方位取得
IRC	IRC 通信処理
Skype	Skype の制御

表 2　プラグインの詳細
Table 2　Details of some plug-ins.

Database Plug-in

種類	名前	内容
EVENT	SELECT	データ検索
	INSERT	データ挿入
	DELETE	データ削除
	UPDATE	データ更新
ACTION	QUERY	SQL 処理
CURRENT	DB.*table*	データ参照

Current Position (GPS) Plug-in

種類	名前	内容
EVENT	MOVE	移動
ACTION	N/A	N/A
CURRENT	LATITUDE	緯度
	LONGITUDE	経度

これまでに実装したプラグインの一部を**表 1** に，その中で，GPS プラグインおよびデータベース処理プラグインの詳細を**表 2** に示す．プラグインは，C++，C#，VB.NET など多様な言語で記述でき，既存の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．

図 6 にルール例を示す．この例は状況依存電話アプリケーションを実現しており，システムが電話（Skype）の着信を検出したとき，ユーザが立ち止まっていれば *StandingContext* ルールと *SkypeAccept* ルールが電話に出るかどうかをユーザに質問し，ユーザが走っている状態であれば *RunningContext* ルールが自動的に留守番電話の機能を実行する．

3.2　コンテキスト定義ツール

2 章で述べたようなサービスや図 6 に示すようなサービスを記述するためには，(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 つの要件がシステムに求められることを述べた．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) は実現した．一方，2 章であ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

```
DEFINE StandingContext
WHEN SKYPE_INCOMING
IF GLOBAL.CONTEXT == 'stand'
THEN DO CMN_SHOW_QUESTION(NEW.ID,
  'Call from NEW.DISPNAME. Connect?')

DEFINE SkypeAccept
WHEN CMN_ANSWER
IF ?NEW.RESULT
THEN DO SKYPE_HANDLE_INCOMING(NEW.ID, 'INPROGRESS')

DEFINE RunningContext
WHEN SKYPE_INCOMING
IF GLOBAL.CONTEXT == 'running'
THEN DO SKYPE_HANDLE_INCOMING(NEW.ID, 'RECORD')
```

図 6　ルール記述例
Fig. 6　An example of ECA Rule.

イベントだけでなく，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[8]，「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．たとえば「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，といい表せる．例として，**表 3** にいくつかのコンテキストを得る際に用いられるセンサおよびその特徴量を示す．表から分かるように，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．

表 3　コンテキストと用いるセンサおよび特徴量の例
Table 3　Contexts and characteristic values.

コンテキスト	必要なセンサと特徴量
ある場所にいる	GPS から取得した位置の現在値
移動中	GPS から取得した位置の変化量
回転している	地磁気センサから取得した方向の変化量
立っている	加速度センサの一定期間で
歩いている	の平均値・分散値
自転車に乗っている	
ある人からのメール着信	メーラから得られたイベントの現在値
財布をかばんから出す	RF-ID 読み取りの現在値
12 時ちょうど	時刻の現在値
ある時点から 10 分後	時刻の変化量
暑い	温度センサの値の現在値
暑くなってきた	温度センサの値の変化量

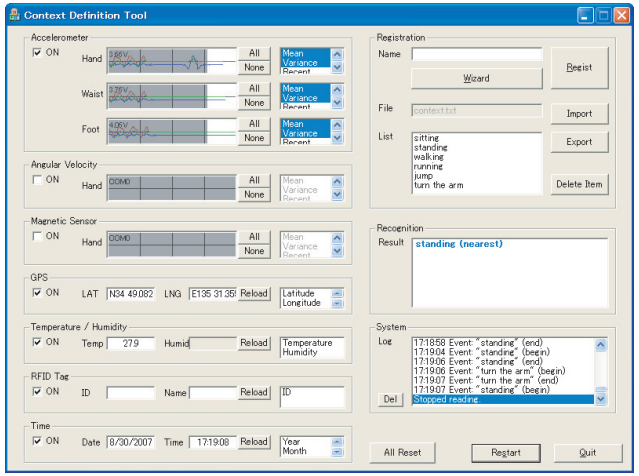

図 7　コンテキスト定義ツールの画面
Fig. 7　A snapshot of context definition tool.

そこで，Wearable Toolkit では，**図 7** に示すようなコンテキスト定義ツールを提供している．図の左部分には常時モニタリングしているセンサデータが表示されており，コンテキストを定義するときには，対象となるコンテキストを表すセンサデータの区間を直接指定し，利用するセンサ群および特徴量をそれぞれ選択すればよい．図の右部分はコンテキスト

定義のためのインタフェースおよびコンテキスト認識結果の表示部である．コンテキスト定義ツールはさまざまなセンサを同時に扱うことができ，センサの追加も容易である．図中では，3 個の 3 軸加速度センサ，方位センサ，角加速度センサ，GPS，温度センサ，湿度センサ，RF-ID タグリーダそれぞれの値が利用可能な状況であることを示している．

コンテキストに関する知識を持つユーザがこのツールを使ってコンテキストを定義する流れは下記のようになる．

(1)　一時停止ボタンを押し，現在から 10 秒程度前までのデータが表示された状態にする．
(2)　定義したいコンテキストを認識するのに適したセンサの組合せを選択する．
(3)　選択したセンサに対して，学習に用いる時間区間（ウインドウサイズ）および，その区間について計算する特徴量（平均，分散，FFT など）を手動で決定する．
(4)　登録ボタンを押し，コンテキストの名前をつける．

従来システムにおけるコンテキスト定義と異なり，ユーザはセンサやウインドウサイズ，特徴量を GUI を用いて直接的に決定でき，さらに定義したコンテキストはそのままルールエンジンのイベントとして登録できるため，対応する ECA ルールを記述するだけでコンテキスト依存サービスが容易に構築できる．一方，その場プログラミングを実現するためには，一般のユーザがコンテキストを定義する必要があるが，上記手順 (2) および (3) にあたる，一般ユーザがセンサの組合せや特徴量，ウインドウサイズを決定することは困難である．

そこで本研究では，コンテキスト定義ツールにセンサや特徴量，ウインドウサイズを自動選択する機構を搭載した．この機構を用いると，コンテキスト定義は下記の手順で行われる．

(1)　ストップボタンを押す．
(2)　システムが生成する簡単な質問に答える．
(3)　コンテキストに名前をつける．

ここで，手順 (2) で提示される質問は専門的なものではないため，一般人でも容易にコンテキストの定義が行える．この自動化は，以下で説明する分散に基づく状況クラスタリングと質問回答インタフェースを融合させることで実現した．

3.2.1　分散に基づく状況クラスタリング

コンテキスト定義を自動化するためには，システムはユーザが定義したいコンテキストを表しているセンサデータ区間を適切に認識する必要がある．ここで，加速度センサによる行動認識を例にとると，ユーザの状況は次の 3 つに分類できる．

(1)　立っていたり座っているなどの**静止状態**：センサデータの値は変化しない．したがって，認識に適した特徴量は平均値や瞬時値となる．
(2)　歩いたり走ったりなどの**定常運動状態**：センサデータは定常的なパターンで変化する．したがって，認識に適した特徴量は平均値や分散値となる．
(3)　ジャンプや腕を上げるなどの**ジェスチャ動作**：センサデータは不規則に変化する．したがって，適した特徴量は波形の形状となり，DP マッチングなどのパターン認識を用いることが有効となる．

これらの状況を考慮し，システムは適切なウインドウを下記の手順で取得する．

(1)　瞬間的な分散値 v を下記の式により計算する．

$$v(t, t-\delta) = \frac{1}{\delta}\sum_{k=t-\delta}^{t} x(k)^2 - \left(\frac{1}{\delta}\sum_{k=t-\delta}^{t} x(k)\right)^2$$

(2)　連続する 2 区間の分散値の変位 $d(t)$ を求める．$d(t)$ が閾値 T_d より大きければ，この点は状況が変化した点の候補となる．

$$d(t) = v(t+\delta, t) - v(t, t-\delta)$$

(3)　ウインドウサイズの閾値 T_w を下回らない最小の区間を求めるウインドウとする．

抽出したウインドウの特徴をもとに，次節に示す質問回答インタフェースを用いてその状況の認識に適した特徴量を自動的に決定する．

3.2.2　質問回答インタフェース

ジェスチャ動作や RF-ID の読み取りなど明示的な動作に関しては，前項のアルゴリズムにより自動認識可能である．一方，値がそれほど変化しない GPS や温度などの値は，ユーザがその値をコンテキストに含めたいのかどうかを判断することは困難である．具体的には，人が歩いているときにコンテキスト定義が開始された場合，歩いていることを登録したいのか，それとも GPS より得られるこの場所をコンテキストとして登録したいのかをデータのみから判断することは困難である．したがって，提案するツールでは質問回答インタフェースにより，そのような自動的に取得しにくい情報に限ってユーザに答えさせることで定義の精度を高めている．質問は「はい/いいえ」で回答する簡単なもののみに限定しており，一般人でも容易に回答できる．例として，加速度センサ，GPS，温度センサ，RF-ID リーダがシステムに接続されている場合の質問フローを**図 8** に示す．このような仕組みを用いることで，システムはコンテキストを自動定義できるようになる．

3.2.3　特徴量の決定

前項までに説明した分散に基づく状況クラスタリングおよび質問回答インタフェースの結果を用いた具体的な特徴量決定の手順は下記のとおりとなる．

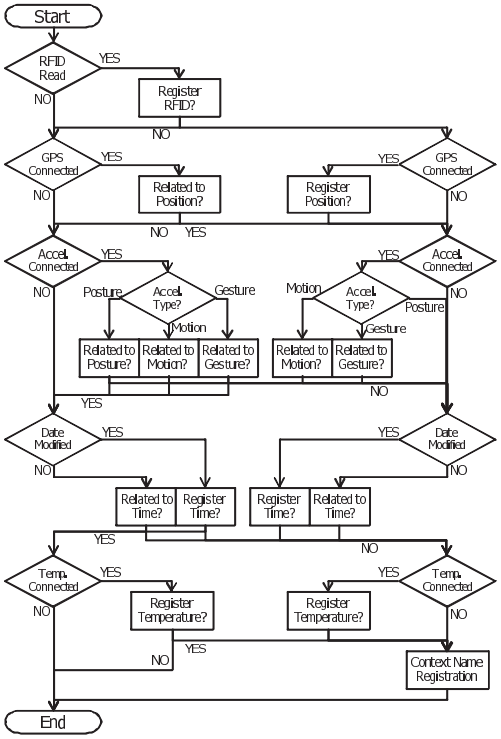

図 8　質問フローの例
Fig. 8　An example of question flow.

(1)　抽出ウインドウの長さが閾値 T_1 より長ければ静止・定常状態，短かければジェスチャと判定する．静止・定常状態の場合，さらにウインドウにおける分散値が閾値 T_2 より小さければ静止状態，大きければ定常状態とする．

(2)　状態に基づき基本特徴量を設定する．プロトタイプでは，静止状態の場合は平均値，分散値，瞬時値，定常状態の場合は平均値，分散値，積分値，ジェスチャの場合はDP マッチング差分，分散値，積分値，線形近似，が設定される．他のセンサに関してもそれぞれ基本特徴量が設定される．

(3)　複数の加速度センサを利用している場合，ウインドウ境界が他のものと大きく異なっている場合はそのセンサに関する特徴量を削除する．

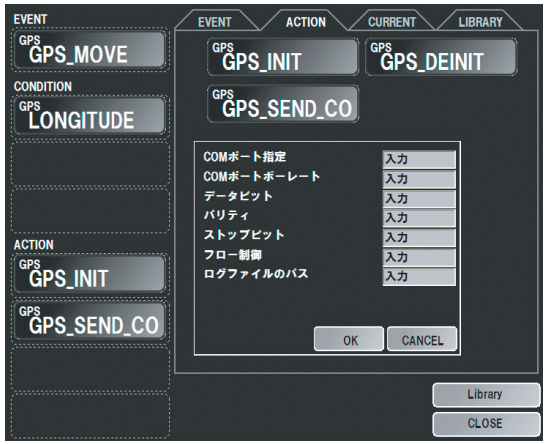

図 9　GUI 版ルールエディタ
Fig. 9　GUI-based rule editor.

(4)　質問の結果により関係しないとされた特徴量は削除する．逆に，質問の結果，関連するとされた特徴量を選択する．

(5)　選択されたセンサ数が少ない場合は基本特徴量をそのまま特徴量として登録する．センサ数が多い場合には，その数に応じて特徴量の選択数を減らしていく．これは特徴量が多すぎることで認識率が低下することを避けるためである．

このように，コンテキスト定義ツールを用いることで状況の定義が容易になる．ツールで利用するセンサはプラグイン形式で自由に追加可能であり，センサ値を用いて定義可能なあらゆる状況が登録可能である．

3.3　その他のツール

Wearable Toolkit はルールエンジンやコンテキスト定義ツール以外にもいくつかのツールから構成されている．**図 9** に示すルールエディタは，GUI を用いてイベントやアクションを選択するだけでルールを定義できるエディタである．また，アプリケーションの GUI を作成するための GUI ビルダを用意しており，一般のウインドウアプリケーションのようなシステムの構築や，Flash や HTML ページをインタフェースとして用いたアプリケーション制作が容易に行える．さらに，ルールの実行管理や動作解析を行うデバッガも提供している．デバッガはシステム内のルール・プラグインの一覧や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機能や，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てプラグインやルールが正常に動作しているかを確認する機能を提供している．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したりする助けになる．また，イベント駆動型ルールの連鎖実行による無限ループを回避するために，筆者らがこれまでに提案したモバイルトリガグラフ[14]によりルール群が安全に動作するかを事前に把握できる機能を持つ．

4. 実装および評価

提案するルールエンジンとツール群を実装した．実装は主に Windows2003 Server 上で Visual C++を用いて行い，ほとんどのツールはルールエンジンのプラグインとして作成した．ツールをプラグインとすることで，ツールからエンジンに対して直接イベントやアクションを発行できる．ツール群は Wearable Toolkit ウェブサイト[15]において一般公開しており，誰でも自由にダウンロードできるようにしている．

4.1 サービス開発例

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は，2007 年 6 月に開催された「モバイルネイチャーラリー in 万博公園」である．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などを記したラリー結果がもらえる．また，クイズの点数など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 カ所のポイントを回り終えたらゴールする．ゴールでは，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分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」「効率良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．これらの要求を満たすためにウェアラブルシステムに必要な機能は下記のとおりである．

- GPS データのログをとる機能：ユーザの歩いた距離を把握する．
- 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．
- ジェスチャによるシステム制御機能：容易にシステムを利用できるようにする．
- ジョグダイアルによるシステム制御機能：容易にシステムを利用できるようにする．
- 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．

STEP2　ウェアラブルシステムの機能設計

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．今回は，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を装着することとした．

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．利用したプラグインは，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 プラグインである．

STEP4　ECA ルール記述

求める機能を実現する ECA ルールを記述する．記述したルールの一部を**図 10** に示す．

STEP5　コンテキストの学習

コンテキスト定義ツールを用い，システムが利用するコンテキスト（歩行中，静止中，右手を上げているなど）を学習させる．システム利用時にはコンテキスト定義ツールが状況認識を行い，ルールエンジンにコンテキストの変化をイベントとして伝える．

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，大学 4 回生の男子 1

```
//Jump to web site according to detected visual code
DEFINE VisualCodeJump
WHEN QR_DETECT
THEN DO GUI.Main.Browser.URL = STRING('NEW.CODE')

//Browser control by gesture
DEFINE ScrollDown
WHEN CONTEXT_RECOGNIZED(RightHandUp)
THEN DO WEB_SCROLL_BY('Main', 'Browser', 0, -50)

//Send context information to the server
DEFINE SendContextInfo
WHEN CONTEXT_RECOGNIZED
THEN DO NET_SEND(['GLOBAL.SERVER', '%NEW.CONTEXT%'])
```

図 10　実装したルールの一部
Fig. 10　An example of implemented ECA Rules.

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 4 日間延べ 10 時間程度で完成させた．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．

また，本節では比較的簡単なアプリケーションの構築例を述べたが，Wearable Toolkit を用いて複雑で大規模なシステムを構築することも可能である．たとえば文献 2) では，Wearable Toolkit におけるルールエンジンの旧バージョンを用いて，バイクレースにおける監督・ピットクルーや観客を支援するウェアラブルシステムを実装し，3 年間にわたって運用した．システムは 127 個のルールからなり，ネットワークで接続された 30 台のコンピュータを用い，レース状況のリアルタイム更新やライダ交代時間予測，路面状況通知などレースを支援するさまざまな機能が実現されている．

4.2　コンテキスト定義ツールの評価

提案するツールキットの最も特徴的な機能の 1 つがコンテキストの自動定義機能である．一方，自動で定義されたコンテキストが十分な認識精度が得られなければこの機能の価値は少ない．そこで，コンテキスト自動定義機能の能力を評価した．評価はコンテキスト認識の研究者（専門家）5 名と，一般人 8 名に対して，**表 4** に示した 9 つのコンテキストを (1) コンテキスト定義ツールと自動定義機能を使った場合，(2) ツールの GUI のみを使って手動で操作した場合，それぞれについてコンテキスト定義を行わせ，その後定義した動作の認識精度を調査した．

表 4　評価に用いたコンテキスト
Table 4　Recognising contexts for evaluation.

コンテキスト	種別
立っている	静止
座っている	静止
歩いている	定常
走っている	定常
ジャンプする	ジェスチャ
手を回す	ジェスチャ
夕方に RF-ID を読む	混合
特定の場所を歩く	混合
特定の場所で手を回す	混合

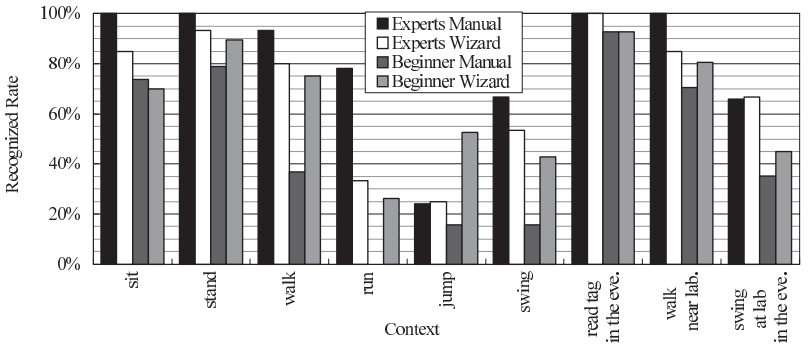

図 11　詳細な評価結果
Fig. 11　Evaluation result (detailed).

結果を**図 11**，**表 5** に示す．表は各条件における平均認識率および，リファレンスとなる専門家が手動設定した場合の特徴量選択との適合度を表す．また，図は各コンテキストごとの認識精度を示す．ここで適合度 R とは，そのコンテキストに対して専門家が選んだ特徴量の和集合 S_{ex} と，ユーザ n が選んだ特徴量集合 S_n を用いて，$R = \frac{|S_{ex} \cap S_n|}{|S_n|}$ と定義され，専門家が選んだ特徴量集合に含まれる特徴量のみを多く選ぶほど 1 に近づく値である．

評価結果より，一般人に関しては，自動設定を行うことで明らかに認識結果が上がっていることが確認できる．これは，自動設定を行うことで特徴量の適合度が上がっているという

表 5　評価結果の平均値
Table 5　Evaluation result (average).

組合せ	認識率	適合度
専門家 – 手動	0.81	1.0
専門家 – 自動	0.69	0.85
一般人 – 手動	0.47	0.62
一般人 – 自動	0.64	0.80

結果からも明らかである．つまり，一般人にとってはこの自動設定機能は有用であるといえる．一方，自動設定の性能はいまだに専門家が手動設定した場合に比べると性能が悪いため，さらなる自動設定精度の向上が求められることが分かる．

結果を詳細に検討すると，まず図 11 における *walk* や *run* では *beginner – manual* に対して *beginner – wizard* の結果が大きく改善されていることが分かる．これは，一般人は全般的に余計な特徴量を選択する傾向にあり，特に *walk* や *run* などでは波形マッチングを選択したことにより認識精度が大きく低下していたことが分かった．提案アルゴリズムを用いた場合，分散に基づく状況クラスタリングにより，これらの状況に関して波形マッチングが割り当てられず，さらに質問回答インタフェースによって不要な特徴量が除外されたため認識精度が高くなっている．一方，*sit* においては自動化した方が結果が悪くなっている．これは，あるユーザが自分が座った瞬間に登録処理を行ったため，座る際の衝撃を含めた短い期間をジェスチャとして登録してしまい，結果として再現時の認識精度が 0%になっていたことが原因であった．このように，分散に基づく状況クラスタリングが正しい場合には自動化により認識精度は大きく改善するが，失敗した場合にはかえって手動時より認識精度が悪化する場合があり，より精度の高い状況クラスタリングの実現が今後の課題となる．

jump や *swing* に関しては，専門家が手動で設定しても精度がそれほど高くない．これは，専門家はほぼ全員が波形マッチングを特徴量に含めていた一方，システムが波形比較を行う際の閾値の設定が厳しく，ジェスチャの認識自体の精度が悪かったことが原因であった．特に *jump* 動作に関しては，予備動作や着地時の衝撃が大きくしかもランダムであるため，*swing* と比べても波形マッチングによる認識精度が低くなっている．一方，*jump* において *begineer – wizard* の精度が大きく向上しているのは，*sit* の場合と同様にウインドウ認識に失敗し，波形マッチングではなく平均・分散が特徴量として選択されたことがかえって認識精度の向上を招いたことが原因であった．

また，複雑な組合せを持つ状況である図 11 の右 3 つの状況に関しても，自動化により認識精度の維持もしくは向上が実現できている．これは，質問回答インタフェースを用いることで過不足なく状況を表すセンサおよび特徴量が設定できたことを意味する．

5. 結　　論

本論文では，その場プログラミングの実現を目指し，Wearable Toolkit と呼ぶフレームワークを提案した．Wearable Toolkit のルールエンジンおよびコンテキスト定義ツールをはじめとするツール群を利用することで，一般ユーザでもコンテキスト依存サービスが容易に構築できることを示した．今後の課題としては，Wearable Toolkit を実際のイベントにおける運用を繰り返すことで機能向上および安定性の向上を図っていきたい．また，ECA ルールやプラグインのリポジトリを作成し，ウェアラブルコンピューティングのための基盤システムとして広く利用されることを目指す．

参 考 文 献

1) Ouchi, K., Suzuki, T. and Doi, M.: LifeMinder: A wearable Healthcare Assistant, *Proc. 2nd International Workshop on Smart Appliances and Wearable Computing* (*IWSAWC2002*), pp.791–792 (2002).
2) Miyamae, M., Terada, T., Tsukamoto, M., Hiraoka, K., Fukuda, T. and Nishio, S.: An Event-driven Wearable System for Supporting Motorbike Races, *Proc. 8th IEEE International Symposium on Wearable Computers* (*ISWC2004*), pp.70–76 (2004).
3) Cheverst, K., Davies, N., et al.: Developing a Context-aware Electronic Tourist Guide: Some Issues and Experiences, *Proc. 18th Conference on Human Factors in Computing Systems* (*CHI2000*), pp.17–24 (2000).
4) Miyamae, M., Terada, T., Kishino, Y., Tsukamoto, M. and Nishio, S.: An Event-driven Navigation Platform for Wearable Computing Environments, *Proc. IEEE International Symposium on Wearable Computers* (*ISWC2005*), pp.100–107 (2005).
5) Naya, F., Ohmura, R., Takayanagi, F., Noma, H. and Kogure, K.: Workers' Routine Activity Recognition using Body Movement and Location Information, *Proc. 10th IEEE Internation Symposium on Wearable Computers* (*ISWC2006*), pp.105–108 (2006).
6) Stiefmeier, T., Ogris, G., Junker, H., Lukowics, P. and Troster, G.: Combining Motion Sensors and Ultrasonic Hands Tracking for Continuous Activity Recognition in a Maintenance Scenario, *Proc. 10th IEEE International Symposium on Wearable Computers* (*ISWC2006*), pp.97–104 (2006).
7) MIThril Project. http://www.media.mit.edu/wearables/mithril/
8) Dey, A.K., Salber, D. and Abowd, G.D.: A Conceptual Framework and a Toolkit

for Supporting the Rapid Prototyping of Context-Aware Applications, *A Special Issue on Context-aware Computing in the Human-Computer Interaction*, Vol.16, No.2–4, pp. 97–166 (2001).
9) Clarkson, B.: Life Patterns: Structure from wearable sensors, Ph.D. thesis in Massachusetts Institute of Technology (2002).
10) Dey, A.K., Hamid, R., Beckmann, C., Li, I. and Hsu, D.: a CAPpella: Programming by Demonstration of Context-Aware Applications, *Proc. International Conference on Human Factors in Computing Systems* (*CHI2004*), pp.33–40 (2004).
11) Semantic Web. http://www.w3.org/2001/sw/
12) Miyaame, M., Terada, T., Tsukamoto, M. and Nishio, S.: Design and Implementation of an Extensible Rue Processing System for Wearable Computing, *Proc. 1st International Conference on Mobile and Ubiquitous Systems* (*MobiQuitous2004*), pp.392–400 (2004).
13) Widom, J. and Ceri, S.: *Active Database Systems*, Morgan Kaufmann Publishers Inc. (1996).
14) 寺田　努，塚本昌彦，西尾章治郎：移動体計算環境におけるアクティブデータベースの動的トリガグラフ構築機構の設計と実装，情報処理学会論文誌：データベース，Vol.43, No.SIG12(TOD16), pp.52–63 (2002).
15) Wearable Toolkit Website. http://wearable-toolkit.com/

(平成 20 年 10 月 17 日受付)
(平成 21 年 3 月 6 日採録)

推　薦　文

本論文は，ウェアラブルコンピューティング環境における PC ユーザがコンテキストアウェアなシステムを作る際の問題を明確にし，それぞれの問題に対して解決法を提示し，さらにそれらを実際に構築して評価している．これらの点で，ユビキタスコンピューティングの実用化を目指した研究事例として高く評価でき，論文誌の推薦論文としてふさわしい．
（マルチメディア，分散，協調とモバイル（DICOMO2008）シンポジウム
プログラム委員長　串間和彦）

寺田　　努（正会員）

1997 年大阪大学工学部情報システム工学科卒業．1999 年同大学院工学研究科博士前期課程修了．2000 年同大学院工学研究科博士後期課程退学．同年より大阪大学サイバーメディアセンター助手．2005 年より同講師．2007 年神戸大学大学院工学研究科准教授．現在に至る．2004 年より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理事，2005 年には同機構事務局長を兼務．2004 年には英国ランカスター大学客員研究員を兼務．博士（工学）．アクティブデータベース，ウェアラブルコンピューティング，ユビキタスコンピューティングの研究に従事．IEEE，電子情報通信学会，日本データベース学会，ヒューマンインタフェース学会の各会員．

宮前　雅一

2001 年大阪大学工学部電子情報エネルギー工学科情報システム工学科目卒業．2003 年同大学院工学研究科情報システム工学専攻博士前期課程修了．2006 年同大学院情報科学研究科マルチメディア工学専攻博士後期課程修了．博士（情報科学）．同年（株）国際電気通信基礎技術研究所知識科学研究所研究員．2008 年ウエストユニティス（株）入社，現在に至る．ウェアラブル・ユビキタスシステムの研究開発に従事．

山下　雅史

2006 年名古屋工業大学工学部電気情報工学科卒業．2008 年大阪大学情報科学研究科マルチメディア工学専攻博士前期過程終了．現在，三菱スペース・ソフトウエア株式会社に在籍．宇宙分野関連のシステム開発に従事．