2004－UBI－3　(11)
2004／1／20

性能と信頼性を考慮したコンテキストアウェアミドルウェア CAMPUS

久住憲嗣[†1]　中西恒夫[†2,†3,†4]
北須賀輝明[†2]　福田晃[†2,†4]

ハードウェア技術の進化により，携帯電話，PDA 等の，常にユーザに携帯されユーザに様々なサービスを提供する携帯型情報端末が高性能化，高機能化している．これらの携帯型情報端末にはユーザの状況 (コンテキスト) を察知し自動的にユーザの補助を行う，いわゆるコンテキストアウェアネスが求められている．本稿では，携帯型情報端末向けコンテキストアウェアシステムには信頼性と高性能が必要であることを示し，コンテキストアウェアシステムの信頼性を向上させるコンテキスト導出モデルを提案する．さらに，コンテキスト導出モデルに従ってコンテキストを導出する，高性能な開発環境と実行環境を示す．

CAMPUS: A High Performance and Dependability Oriented Context-Aware Middleware

Kenji Hisazumi,[†1] Tsuneo Nakanishi,[†2,†3,†4] Teruaki Kitasuka[†2]
and Akira Fukuda[†2,†4]

In recent years, the mobile information terminal has more power and complexity. The mobile information terminal are required to have context-awareness that the computer aids our life automatically to apprehend user situation. In this paper, we describe that the context-aware system for mobile information terminal requires dependability and high performance. We propose the context deriving model which improves reliability of context-aware system and performance-oriented runtime environment.

1. はじめに

ハードウェア技術の進化により，携帯電話，PDA 等の，常にユーザに携帯されユーザに様々なサービスを提供する携帯型情報端末が高性能化，高機能化している．また，未来の携帯型情報端末であるウェアラブルコンピュータの研究も盛んに行われ，その実現の可能性が見えてきた．近い将来，人間が常に持ち歩くこれらの携帯型情報端末が，より広範囲にわたって人間の生活を補助するようになるものと期待される．

将来の携帯型情報端末には，従来のパーソナルコンピュータと同様に高機能が求められ，多種多様なアプリケーションが搭載される．また，パーソナルコンピュータとは違い，携帯型情報端末はユーザとともに常に移動するため，携帯型情報端末をとりまく周辺環境が著しく変化し，ユーザが必要とするサービスが時々刻々と変化する．さらに，ユーザは携帯型情報端末の操作に専念できない状況下におかれることが多々ある．そのため，携帯型情報端末には周辺の状況 (コンテキスト) に応じたサービスをユーザに提供する，いわゆるコンテキストアウェアネスが求められる．そこで，本研究ではコンテキストアウェアアプリケーションの実装を容易にするミドルウェア「CAMPUS」の研究，開発を行っている．

携帯型情報端末に搭載できる電源，計算機資源等の資源が限られる．搭載ソフトウェアには限りあるバッテリを節約するために，省電力，高性能が求められる．ユーザが常に携帯しサービスを受けるため，誤動作がユーザに危害を与える場合も考えられ，信頼性，安全性が非常に重要である．また，コンテキストアウェアアプリケーションには，リアルタイム性を必要とするアプリケーションが数多く存在する．携帯型情報端

†1 九州大学システム情報科学府
Graduate School of Information Science and Electrical Engineering, Kyushu University
†2 九州大学システム情報科学研究院
Faculty of Information Science and Electrical Engineering, Kyushu University
†3 システム LSI 研究センター
System LSI Research Center, Kyushu University
†4 九州大学情報基盤センター
Computing and Communications Center, Kyushu University

末を構築するためには，これらの要求を満たすソフトウェア基盤が必要不可欠である．そこで，本稿では信頼性の高いコンテキストアウェアシステムの構築を支援すべく，信頼性を考慮したコンテキスト導出モデルを提案する．さらに，コンテキスト導出モデルに従ってコンテキストを導出する，高性能な開発環境と実行環境を示す．

2. コンテキストアウェアシステムへの要求

2.1 高信頼性

常に人間の生活を補助するコンピュータである携帯型情報端末は，信頼性が非常に重要である．携帯型情報端末は従来のコンピュータとは違い，ユーザの実世界における行動に直接指示を与える場合が多くあるため，携帯型情報端末がユーザに間違った指示を与えた場合，携帯型情報端末はユーザに危害を与えうる．たとえば，盲人補助のための人ナビゲーションシステムが誤動作し，車が接近しているにもかかわらず「車道に進行」の指示を出すと，ユーザは事故に遭うことになる．ここで，ユーザが事故に遭わないためには，システム全体の信頼性，安全性が必要である．システム全体の信頼性の向上において，コンテキストアウェアミドルウェアに必要とされる性質は，ミドルウェアが導出したコンテキストをアプリケーションが信頼して自らの振る舞いを変更できる性質である．本稿では，この性質をコンテキストアウェアミドルウェアの信頼性と呼ぶ．安全性は信頼性を確保した上で，携帯型情報端末からの出力を議論すべきものであるため，安全性は本稿の議論の範囲を超える．

コンテキストアウェアミドルウェアの信頼性を向上する方法には以下の 2 種類があると考えられる．

- コンテキスト導出過程の検証を容易にし，テストの自動化や形式的検証等によりコンテキスト導出過程の信頼性を向上させる方法．
- 導出されたコンテキストがどの程度信頼できるか (信頼度) を評価し，アプリケーションにとって必要な信頼度を満たしている場合のみコンテキストを採用するこにより信頼性を向上させる方法．

2.2 高性能

実世界からの情報に基づき振る舞いを変えるコンテキストアウェアシステムでは，リアルタイム性が重要である．携帯型計算機を用いたコンテキストアウェアアプリケーションとして，HMD(ヘッドマウントディスプレイ) を用いたユーザに対して周辺状況の情報を表示するシステムがあるが[1]，このシステムの場合，ユーザの位置，視線に従って適切な情報を提示する必要がある．特に視線情報に従って適切に情報提示ができない場合，ユーザに不快感を与えることになるため，リアルタイム性が重要である．また，特にユーザが補助を必要する場合である火災発生等の危機的状況下では，補助システムが必要なリアルタイム制約を満たさない場合，適切な補助ができずユーザを死に至らしめることもあり得る．

3. 信頼性を向上させるコンテキスト導出モデル

3.1 コンテキスト導出過程の検証性

本節では，コンテキストアウェアミドルウェアの信頼性を向上させる要素の一つである，コンテキスト導出過程の検証性について議論する．

従来のコンテキストアウェアミドルウェアのコンテキスト導出手法を表 1 に示す．

表 1 コンテキスト導出手法

手法	ミドルウェア
コンポーネントの組み合わせによるコンテキスト導出	Context Toolkit[2]
単純な条件式記述によるコンテキスト導出	A-ware[3] RCMS[4]

コンポーネントの組み合わせによるコンテキスト導出は，コンポーネントとしてコンテキスト導出プログラムを通常の手続き的プログラムで記述し，そのコンポーネントを組み合わせることによってアプリケーションが必要とするコンテキストを導出する手法である．基本的に，手続き的プログラムの組合せであるため，拡張性があり，またプログラムコードとして記述できるコンテキスト導出手法であれば，どのような手法でも記述できる．すなわち，記述可能なコンテキスト導出手法が多岐にわたる．このことを記述性が高いと呼ぶ．しかし，手続き的プログラムコードの検証は困難であり，さらにそれをコンポーネントとして組み合わせるため，検証は非常に困難となる．

単純な条件式記述によるコンテキストを導出は，ミドルウェアが前提としているコンテキストモデルに基づいた条件式を与えることによりコンテキストを導出する．そのため，コンテキストモデルが想定していないコンテキスト導出ができないため，コンポーネント

の組み合わせによるコンテキスト導出に比べ記述性に劣る．しかし，比較的検証が容易である．

そこで，本研究では記述性とコンテキスト導出過程の検証容易性を両立させるため，形式的に定義された，しかも単純な条件式以上の記述性を持つコンテキスト導出モデルを提案する．

ここでの検証性は，コンテキスト導出が意味的に正しいかどうかではなく，コンテキストの導出がコンテキスト導出記述どおりに行われるかどうかの事である．

3.2 評価可能性

本節では，コンテキストアウェアミドルウェアの信頼性を向上させる要素の一つである，導出されたコンテキストの評価可能性について議論する．

コンテキストアウエアミドルウエアは，センサやデータベース等の様々なデータソースからコンテキストを算出し，アプリケーションに伝達する．このデータソースには誤差が含まれ，また時々刻々と変化し鮮度が低下する．さらに，生活環境上やインターネット上のデータソース等の，携帯型情報端末に直接装備されているデータソース以外のデータソースを利用する場合もある．そのような様々な性質を持つデータを組み合わせてコンテキストを導出するため，コンテキストをどの程度信頼して良いかわかりにくく，算出したコンテキストのクオリティ(Quality of Context: QoC) を統一的に扱う手法の必要性が指摘されている[5)6)]．QoC を扱う手法はいくつか提案されているが，コンテキストの算出法とともに詳しく議論されておらず，また，形式的に議論されているものはない．

Gray らは文献[5)] において扱うべき QoC の種類を議論しており，扱うべき QoC は, 有効範囲 (Coverage)，分解能 (Resolution)，精度 (Accuracy)，再現性 (Repeatability)，頻度 (Frequency)，適時性 (Timeliness) であると述べている．Henricksen らは文献[6)] において QoC をコンテキストモデルとともに整理する手法の提案を行っている. Gray らは扱うべき具体的な QoC を示してはいるが，いかに QoC を取扱うべきかの議論を行っておらず，また，形式的な定義を行っていない．Henricksen らは QoC を整理する手法を提案しているが，実装に触れていない．

本研究では実装を考慮した，QoC とコンテキスト情報を統一的に扱うことができるコンテキスト導出モデルを提案する．

3.3 コンテキスト型とそのインスタンス

CAMPUS は, 複数のデータソースからデータを取得し, それらを加工し, すなわちコンテキストを導出し, その結果をアプリケーションに伝達する. この一連の操作を行うには, コンテキストごとにシステム内で統一されたコンテキストデータの表現形式が必要である. また統一データ表現は, コンテキストデータや, それに付随する QoC 情報を統一的に取り扱うために, 構造化されたデータ表現である必要がある. たとえば, GPS 受信機からは, 時刻, 緯度, 経度, 高度, HDOP (Horizontal Geometric Dilution of Precision: 位置情報の精度の指標), その他の情報が取得できるので, これらを一つのデータ型として取り扱いたい. このデータ構造のことをコンテキスト型と呼ぶ. システム上に定義されたコンテキストの種類が n 個あるとすると, コンテキスト型を C_i $(1 \leq i \leq n)$ とする. また, コンテキスト型に基づいて生成されたコンテキスト情報を格納する実体をコンテキスト型のインスタンス, または単にインスタンスと呼ぶ. インスタンスがシステム上に m 個あるとすると, $c_j \in C_i$ $(1 \leq j \leq m)$ と表す.

構造化されたコンテキスト情報と複数の QoC 情報を統一的に扱えるシステム共通のデータ表現を定義するために, ASN.1[7)] を採用する. ASN.1 は情報の構造を定義するための言語であり, SNMP 等に採用されている. また, BER(Basic Encoding Rules), DER(Distinguished Encoding Rules) 等の, ASN.1 で定義されたデータ構造を可搬性のあるビット列に変換する規則が定義されており, ネットワーク経由でのデータ交換に都合がよい. CAMPUS では, ASN.1 でも特にデータ型を定義する記法を採用する. アプリケーションプログラマは, ASN.1 を用いて表現したいコンテキストとその QoC 情報を含む, データ構造を定義する. このデータ構造のことをコンテキスト型と呼ぶ. また, コンテキスト型にデータが格納されたものをインスタンスと呼ぶ. コンテキスト型はコンテキストの種類ごとに定義する必要がある.

3.4 ルールベースのコンテキスト導出

CAMPUS はデータソースからデータを取得し，アプリケーションが定義したルールに基づきコンテキストを導出する．このルールの記述法として，ECA ルールと呼ばれるアクティブデータベース[8)] で使用される概念を採用する．ECA ルールは下記の 3 つ組からなるルールである．

- 発生するイベント (Event)

ルールを評価する際にトリガとなるイベントを指定する．ここで指定されたイベントが発生した場合，ルールの発火条件を評価する．イベントはデータソースやコンテキスト型のインスタンスが変化したときに発生する．もしくは明示的に他のルールがイベントを明示的に発生させることも可能である．

- ルールの発火条件 (Condition)
  実行される操作 (Action) を実行するための条件を指定する．発火条件を満たした場合，次の実行される操作を実行する．Condition にはコンテキスト型のインスタンス c_i,c_j,... から真偽を判定する関数 $condition_k(c_i, c_j, \ldots)$ を記述する．関数 $condition_k$ に記述できる演算は後述する．
- 実行される操作 (Action)
  コンテキストの導出関数や，アプリケーションへのイベント通知を指定する．Action には，コンテキスト型のインスタンス c_i,c_j,... から c_k を導出する $c_k = f_k(c_i, c_j, \ldots)$，もしくは明示的なイベント通知を記述する．関数 f_l に記述できる演算は後述する．Action は複数記述することができる．

ECA ルールはシステム中に複数存在し，すべてのルールは並列に評価される．そのため複数の ECA ルールを組み合わせて複雑なルールを記述することができる．

GPS 受信装置からの位置情報を利用して，ユーザがある範囲に入った際に，適切なアプリケーションに通知を行うという場合，Event，Condition，Action は以下のようになる．

- Event: GPS 受信機からのデータが到着
- Condition: GPS 受信機から取得した位置情報が，ある範囲に入った
- Action: 適切なアプリケーションにイベントを通知する

ECA ルールは，ある事象が発生し (Event)，ある条件を満たしたとき (Condition)，ある操作 (Action) を実行するという，人間の考え方に即したルール記述法であるため，比較的理解が容易である．また，ECA ルール中に記述するコンテキスト導出演算に，数学的に定義された演算のみを使用すると，ECA ルールの振る舞いは形式的に定義できる．

ECA ルール中，特にルールの発火条件 ($condition_k$) や実行される操作 (f_l) には以下の演算を記述できる．

- 集合演算[9)]
- ファジィ集合演算[10)]
- 属性文法評価によるコンテキスト導出モデル[11)]

この 3 種類の演算は，それぞれの思想に基づいた独立したコンテキスト導出モデルである．アプリケーションプログラマは適切なコンテキスト導出モデルを選択し，使用する．異なるコンテキスト導出モデル間でインスタンスを利用する場合には，互換性のあるコンテキスト型に明示的に変換し使用する．

また，上記 3 種類の演算中に共通で使用できる基本的な演算は以下の通りである．

- インスタンスの変更
- 四則演算
- 比較演算

コンテキスト導出モデルの違いによる得失を表 2 に示し，以下に説明する．

表 2 コンテキスト導出モデルの違いによる得失

	性能	QoC 導出容易性	コンテキスト 変化記述性
集合演算	○	△	△
ファジィ集合演算	×	○	△
属性文法評価	○	×	○

集合演算を主に用いたコンテキスト導出手法は，コンテキストとその QoC 情報を変換する演算を，明示的にアプリケーションプログラマが与える方式であり，振る舞いに曖昧さはなく，また，高速に演算できる．しかしながら，コンテキストの演算に加え，QoC の演算内容も明示的に与える必要があるため，プログラマへの負担が大きい．

対照的に，ファジィ集合演算を用いたコンテキスト導出モデルは，コンテキストの演算のみに注力して演算を記述すればよく，QoC 情報はコンテキスト演算に基づき自動的に演算ができる．プログラマの負担は軽減されるが，実行性能が悪く，またその意味的な振る舞いはあいまいである．

前 2 種類のコンテキスト導出モデルは，現在のコンテキストのスナップショットを関数的演算により導出するモデルであるのに対し，属性文法評価によるコンテキスト導出モデルは，コンテキストの連続的変化に着目したコンテキスト導出モデルである．本モデルは主に他の 2 種類のコンテキスト導出モデルにより記号化されたコンテキストを使用し，そのコンテキストの連続的変化を直接的に記述可能である．

4. 性能指向実行環境

多くのルールベースシステムは，実行時にルールを文字列として与えそのルールを実行時に解釈し実行する．実行性能を向上させるために，ルールを開発環境上で中間表現にコンパイルすることにより，文字列解釈のコストを削減するシステムも存在する．CAMPUS ではより高性能を目指し，ひいてはリアルタイムシステムでも利用可能にすべく，開発環境上でコンテキスト導出ルールからコンテキスト導出を行う実行コードを生成する，コンテキスト記述コンパイラを導入する．CAMPUS の開発環境，コンテキスト記述コンパイラ及び実行環境の概略を図 1 に示す．アプリケーションプログラマは，CAMPUS に依頼したいコンテキスト導出ルール及びコンテキスト型定義を，コンテキスト記述コンパイラに渡す．コンテキスト記述コンパイラはそれらを元に，ターゲットプラットフォーム向けの実行コードを生成する．アプリケーションが生成された実行コードを実行環境に設定すると，実行環境は実行コードを内部で実行させ，コンテキスト導出を行う．本節では提案方式について議論する．

4.1 コンテキスト記述コンパイラ

ここでは，コンテキスト記述コンパイラについて議論する．コンテキスト記述コンパイラは，コンテキスト型記述やコンテキスト記述に基づいてコンテキストを導出する，コンテキスト導出コードを生成する．コンテキスト導出コードはターゲットプラットフォーム用の実行コード含む．コンテキスト導出コードは，コンテキスト型やそのインスタンスの情報，初期化コード，コンテキスト演算コード，使用するコンテキスト名リスト，使用するコンテキスト導出演算リストを含む．

コンテキスト記述コンパイラは，まず，コンテキスト型情報やインスタンスの確保を行うためのコードを生成する．コンパイラは ASN.1 記述されたコンテキスト型定義を元に，コンテキスト型名，コンテキスト型の要素名，コンテキストを格納するためのメモリイメージ情報を含む中間コードを生成する．また，インスタンス宣言を元に，インスタンスを格納する領域を確保するコードを生成する．

次に，コンテキスト記述コンパイラは，コンテキスト導出ルールに基づき，具体的なコンテキスト導出コードを生成する．前述したようにコンテキスト導出ルールは，Event，Condition，Action からなる．コンテキスト記述コンパイラは Event 部に記述されているイベント対象に，イベントが発生した際の自ルール評価を依頼する初期化コードを生成する．また，使用するコンテキスト名リストに登録する．また，Condition 部，Action 部には，前述した集合演算，ファジィ集合演算，属性文法コンテキスト記述を用いて記述されている．それらの記述に基づきコンテキスト演算コードを生成する．集合演算，ファジィ集合演算は，実行環境により関数として提供されるため，コンテキスト導出ルールに基づき適切に関数呼び出しを行うコードを生成する．属性文法コンテキスト記述は，コ ンテキストの認識のために属性言語の構文解析エンジンを必要とする．CAMPUS では，構文解析表を用いた構文解析アルゴリズ ムを採用する．そのため，コンパイラは属性文法コンテキスト記述に基づき構文解析表を生成する．ルール中で使用したコンテキスト導出演算リストをコンテキスト導出コードに埋め込む．

4.2 コンテキスト記述コンパイラによる最適化

ここでは，コンテキスト記述コンパイラによる最適化手法について議論する．まず，使用されるコンテキストの性質について考察し，考察結果に基づき最適化手法を示す．

- 比較的汎用的なコンテキスト:
  複数のアプリケーションから使用される一般的なコンテキストで，位置情報，時間等がある．この種類のコンテキストには，アプリケーション独自の解釈が入っていない．これらのコンテキストは汎用的であるため，導出するルールが変更されることはまれである．
- アプリケーションに固有なコンテキスト:
  1 つないしは少数のアプリケーションから使用されるコンテキストである．この種類のコンテキストにはアプリケーション独自の解釈が入っており，他のアプリケーションからは再利用しにくい．アプリケーションの開発時にコンテキスト導出ルールは記述され，コンテキスト導出ルールはアプリケーション実行時にシステムに追加，変更，削除される．

位置情報等のごく一部の一般的な概念に基づくコンテキストをのぞき，コンテキストは後者に分類される．これは，CAMPUS ではコンテキスト情報と QoC 情報を同時に取り扱うが，QoC 情報の解釈はアプリケーション独自の解釈であることが多く，そのようなアプ

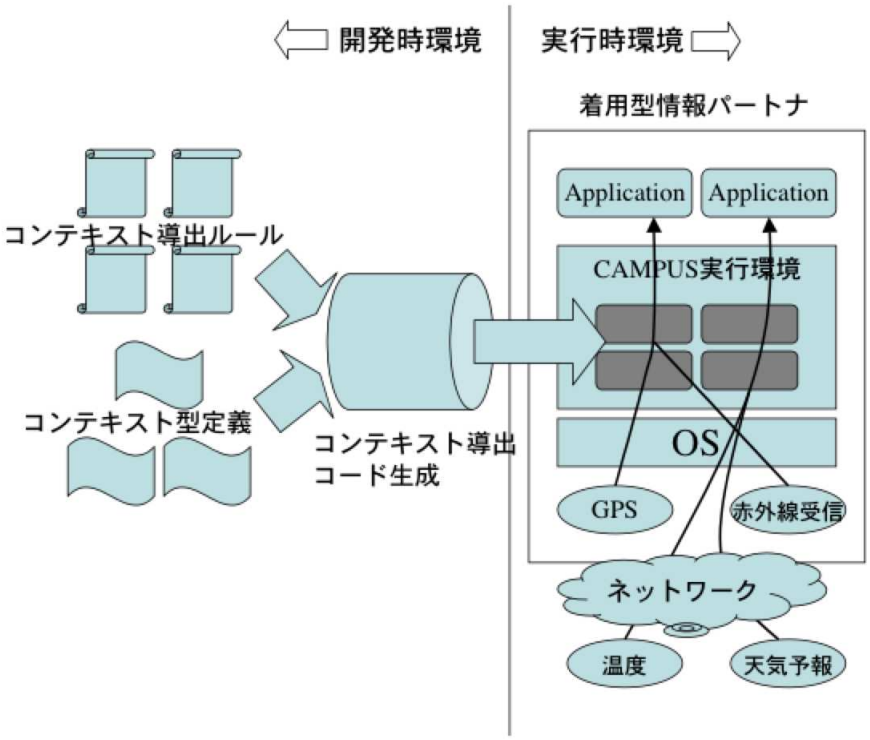

図 1 開発環境，コンテキスト記述コンパイラ，実行環境の概略図

リケーション独自解釈を含む情報は再利用しにくいためである．

アプリケーションに固有のコンテキストは他のアプリケーションから使用されることはない．また，CAMPUS のコンテキスト導出ルールは実行時には変更されることはない．そのため，コンテキスト導出ルールのアプリケーションに固有の部分は，開発時に静的な最適化が可能である．様々なルールの最適化手法が提案されており，CAMPUS ではそれらの最適化手法を実行時の性能低下なしに適用することができる．

4.3 CAMPUS 実行環境

ここでは CAMPUS の実行環境について述べる．CAMPUS 実行環境は，イベント管理部，コンテキスト管理部，ルール管理部，コンポーネント管理部から構成される (図 2)．コンテキスト導出コードを静的に CAMPUS 実行環境にリンクすると，CAMPUS 実行環境動作中に動的にアプリケーションの追加，削除ができない．また，コンテキスト導出コードをダイナミックリンクライブラリ化しアプリケーションを利用する際に実行環境に組み込むだけでは，インスタンスへのアクセスができなくなる等の問題がある．そのため CAMPUS 実行環境は，アプリケーションの実行時追加，削除を考慮し，設計する．

イベント管理部は，CAMPUS 実行環境で発生するイベントを管理し，コンテキスト導出コードにより設定された通りにコンテキスト導出コードを実行し，また，アプリケーションにイベントの伝達を行う．

コンテキスト管理部はコンテキスト型情報表とインスタンス表を内部に持ち，CAMPUS 実行環境内のコンテキスト型やインスタンスを管理する．コンテキスト演算コードは，コンテキスト管理部からコンテキスト名を用いてインスタンスを取得し，それらを加工することによりコンテキストを導出する．また，コンテキスト導出コードは，コンテキスト管理部からのコンテキスト型情報を用いて，インスタンス内部にアクセスする．動的にアプリケーションの追加，削除を行う場合，コンテキスト型やインスタンス情報があらかじめわからない場合があるため，コンテキスト管理部が必要となる．しかし，アプリケーション固有のコンテキスト型，もしくは CAMPUS が標準で提供する型を

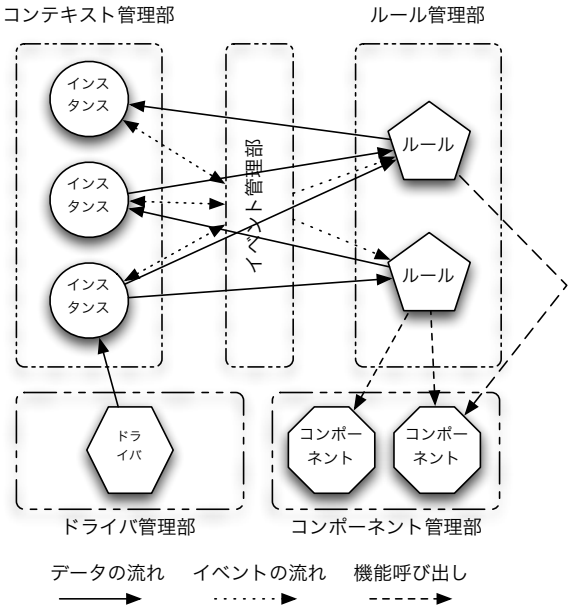

図 2 CAMPUS 実行環境アーキテクチャ

利用する場合は，あらかじめメモリイメージがわかっているので，直接アクセスすることができる．

ルール管理部は，コンテキスト導出コード表を持ち，コンテキスト導出コードの追加，削除を機能を提供する．コンテキスト導出コードを登録する際，コンテキスト導出コードが使用するイベント，コンテキスト，コンテキスト導出演算が適切に提供されているかを検査し，提供されていればコンテキスト導出コード表に登録し，コードの内容に従いイベント管理部にイベント設定を行う．提供されていない場合にはエラーになる．削除の際には，コンテキストが現在使用されていないか，コンテキスト導出コードの実行途中でないかを検査し，両条件を満たした場合にのみ削除を行う．見なさない場合はエラーになる．

コンポーネント管理部は，CAMPUS 実行環境に登録されているコンポーネントを管理する．CAMPUS では，実行環境サイズ削減のため，コンテキスト導出に使用する演算，その他の機能をコンポーネント化し，必要な機能のみを CAMPUS 実行環境に組み込む．標準で提供されるコンポーネントは，集合演算，ファジィ集合演算，属性文法評価によるコンテキスト導出実行環境，コンテキスト導出ルールインタプリタである．実行環境構築時にこれ以外のコンポーネントを CAMPUS 実行環境に組み込むことにより，CAMPUS の機能を拡張できる．さらに，コンテキスト導出コードに，コードが使用するコンポーネントを添付することにより，実行時に CAMPUS の機能を拡張することも可能である．機能を拡張する場合，コンポーネントが提供する機能が演算のみの場合，コンポーネントが提供する機能を関数として呼び出すことができるため，コンテキスト記述コンパイラを拡張する必要はない．属性文法評価によるコンテキスト導出のような，関数で記述できないコンテキスト導出手法の場合は，コンテキスト記述コンパイラを拡張する必要がある．

ドライバ管理部は，データソースからデータを取得し，CAMPUS 実行環境にデータを受け渡す機能を持つドライバを管理する．

5. ま と め

本稿では，将来のコンテキストアウェアシステムは高信頼性が必要であることを示し，その信頼性を向上させるためにはコンテキストアウェアミドルウェアが，形式的なコンテキスト導出モデルに基づきコンテキストの導出を行うこと，導出したコンテキストが評価可能であること，高性能であることの 3 条件を満たす必要があることを示した．そして，前 2 条件を満たすコンテキスト導出モデルを提案した．そのコンテキスト導出モデルは，コンテキスト導出ルールとして ECA ルールを採用し，コンテキスト導出ルール中に集合演算，ファジィ集合演算，属性文法評価によるコンテキスト導出記述が記述できる．集合演算，ファジィ集合演算を使用したコンテキスト導出結果は評価可能であり，また，3 方式すべてが形式的に定義される．さらに，高性能化を実現すべく，開発時にコンテキスト導出ルールをコンテキスト記述コンパイラによりコンパイルし，実行コードを生成し，その実行コードを実行環境上で実行させる手法を提案した．さらに，コンテキストの性質を整理し，コンテキスト記述をコンパイルする手法と組み合わせて，開発時にコンテキスト導出ルールに対する最適化が可能であることを示した．

CAMPUS の，実行環境上でコンテキスト導出ルールを解釈，実行する実装は完了しており，現在，コンテキスト記述コンパイラ，及び，実行環境の実装を行っている．今後，実装をさらに進め，両者の性能評価と比較を行っていきたい．

謝　　辞

本研究の一部は，科学技術振興機構 (JST) の戦略的創造研究推進事業 (CREST)「高度メディア社会の生活情報技術」プログラムの支援によるものである．

参 考 文 献

1) Tenmoku, R., Kanbara, M. and Yokoya, N.: A Wearable Augmented Reality System for Navigation Using Positioning Infrastructures and a Pedometer, *Proc. IEEE and ACM Int. Symp. on Mixed Augmented Reality(ISMAR 03)*, pp. 344–345 (2003).
2) Dey, A.K. and Abowd, G.D.: A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications, *HCI Journal 2001* (2001).
3) 宮前雅一, 中村聡史, 寺田努, 塚本昌彦, 西尾章治郎: ウェアラブルコンピューティングのための拡張可能なルール処理システム, 情報処理学会研究報告 (情報家電コンピューティング研究グループ 2002-IAC-3), pp. 41–46 (2002).
4) Yau, S., Karim, F., Wang, Y. and Gupta, S.: Reconfigurable Context-Sensitive Middleware for Pervasive Computing, *IEEE Pervasive Computing Magazine*, No. 03 (2002).
5) Gray, P. and Salber, D.: Modelling and Using Sensed Context Information in the Design of Interactive Applications, *Proc. of EHCI 2002*, pp. 317–335 (2001).
6) Henricksen, K., Indulska, J. and Rakotonirainy, A.: Modeling Context Information in Pervasive Computing Systems, *Proc. of Pervasive 2002*, pp. 167–180 (2002).
7) Dubuisson, O.: ASN.1 Reference Book. `http://www.oss.com/asn1/dubuisson.html`.
8) 石川博: アクティブデータベース, 情報処理, pp. 124–129 (1994).
9) 久住憲嗣, 北須賀輝明, 中西恒夫, 福田晃: ウェアラブル／移動情報端末におけるコンテキスト指向プロセス管理, 情処研報 2002-MBL-23 (2002).
10) 久住憲嗣, 北須賀輝明, 中西恒夫, 福田晃: ファジィ理論を応用したコンテキストのクオリティ表現, DICOMO2003 論文集, pp. 285 – 288 (2003).
11) 中西恒夫, 久住憲嗣, 北須賀輝明, 福田晃: コンテキストアウェアアプリケーションにおける文脈自由言語によるコンテキスト記述, 情処学会第 65 回全国大会論文集, pp. 1–(61–62) (2003).