ユビキタスユーザインタフェースミドルウェア（1）
－構想とソフトウェアモデル－

石原　鑑† 中田　秀男† 轟木　伸俊† 大崎　雅代† 秋吉　政徳†
三菱電機（株）先端技術総合研究所†

1. はじめに

企業の業務効率化や業務改善を目的に、IT を活用した企業情報システムの開発が行われている。このようなシステムでは、ユーザは Web ブラウザがインストールされた PC を用いてシステムを利用するのが一般的であった。ところが、近年のネットワークのブロードバンド化や携帯情報端末、携帯電話等の普及に伴い、時と場所を選ばずに利用状況に応じて、ユーザが最も使い勝手の良い手段でシステムを利用したいという要望が高まってきた。このようなシステムを、情報端末（以下、情報端末を表示デバイスと呼ぶ）の種類ごとに個別開発していたのでは、生産効率の観点で問題となる。

また、システムが対象とする業務の範囲は複雑かつ高度化しており、企業内外の様々なアプリケーションやデータベースを容易に利用できることがより重要となっている。このため Web サービスの利用や EAI によるアプリケーション統合が注目を集めているが、複数種類の表示デバイスを用いたユーザインタフェース（以下 UI）の開発が支援されているわけではない。

さらには、日常的にインターネットを利用するユーザが増加しており、ユーザは視覚的効果の高い、いわゆるメディアリッチな UI を求める傾向が強まっている。そのためシステム開発における UI のデザイン作業の比重が高まっている。

以上の考察を踏まえ、本研究では、複数種類の表示デバイスに対応する企業情報システム（以下マルチデバイスアプリケーション）を単一のアーキテクチャで実現するミドルウェア、ユビキタスユーザインタフェースミドルウェア（Ubiquitous User Interface Middleware、以下 UIM）を提案する。UIM は、表示デバイス依存の開発作業を、画面と画面遷移の定義に局所化する。また、画面の定義は、UI のデザイナが市販の画面編集ツールを用いて作成できるようにする。表示デバイスに依存しない処理（システムのロジック定義）は、再利用可能なソフトウェア部品の組み合わせで行う。そして、画面定義と画面遷移定義を、ロジック定義に結びつける手段を提供する。本研究では、UIM の構想とソフトウェアモデルについて述べる。

Ubiquitous User Interface Middleware (1) – Concept and Software Model -
† Akira Ishihara, Hideo Nakata, Nobutoshi Todoroki, Masayo Osaki, Masanori Akiyoshi, Advanced Technology R&D Center Mitsubishi Electric Corporation

2. UIM の構想

図１は、UIM の基本構想を表している。UIM は、パソコン、携帯電話、カーナビ等の複数種類の表示デバイスからシステムを利用できるようにする（多デバイス対応）。また、表示デバイス間で、シームレスに業務を引き継げるようにする（セッション引継ぎ）。さらに、UIM は、Web サービス、RDB、EJB 等の、企業内外の既存サブシステムを呼び出してロジック定義を実行する（多分野対応）。

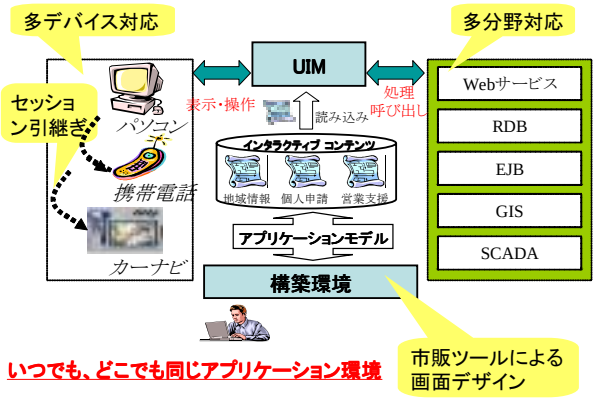

図 1　UIM の基本構想

UIM は、マルチデバイスアプリケーションの構築環境を提供する。そこでは、UI のデザイナが市販の画面編集ツールを用いて画面定義を作成できる。このときデザイナは、プログラミングの知識なしに画面定義を作成できる。

3. UIM のソフトウェアモデル

3.1. ソフトウェアモデル

まず、マルチデバイスアプリケーションを、以下の三つのモデルに分割する。

UI モデル　画面定義と画面遷移定義からなる。

データモデル　ロジック定義で利用される App（アプリケーション）変数定義、UI で利用され

るドキュメント変数定義からなる。

プロセスモデル　ロジック定義にあたるアクション定義、UI に渡すデータの生成処理を定義するバインディング定義からなる。バインディング定義は、ドキュメント変数と App 変数の間のフォーマット変換を行なう。

次に UIM は、ステートチャート（状態遷移図）とファンクションブロック図（FBD）の概念を応用して、これらのモデルを結合する。

ステートチャート　一つの画面が表示されている状態を、ステートチャートの状態とする。そうして、画面遷移定義をステートチャートとする。

FBD　アクション定義、バインディング定義を FBD で表現する。アクション定義においては、Web サービスや RDB の呼び出し処理がファンクションブロック（以下 FB）として表現される。FB の入出力となるのは、アプリケーション変数である。バインディング定義においては、フォーマット変換処理が FB として表現される。FB の入出力の一方は App 変数であり、他方はドキュメント変数である。

最後に UIM は、実行セマンティックスを定義する。ステートチャートの実行は、ECA(Event-Condition-Action)モデルに基づいて行う。状態遷移を、遷移前状態、ECA、遷移後状態の三つ組みで定義する。ステートチャートへイベント（Event）が入力されたときに、条件判定（Condition）がなされて、アクション（Action）が実行され、状態が変わる。UIM は、アクションの実行を以下のように拡張する。

・アクション定義を、入状アクション、退状アクション、遷移アクションに分類する。入状アクション、退状アクションは状態に関連付ける。遷移アクションは ECA に関連付ける。入状アクションは、ある状態に入るときに実行される。退状アクションは、ある状態から出るときに実行される。遷移アクションは、状態遷移時に実行される。

・バインディング定義を、入状バインディング、退状バインディングに分類する。これらを状態に関連付ける。入状バインディングは、ある状態に入るときに実行される。退状バインディングは、ある状態から出るときに実行される。

以下に、イベントが発生したときの典型的な実行系列を示す。

1. 発生イベントに該当する状態遷移を検索し、条件判定を行った上で、状態遷移を確定する。
2. 遷移前状態の退状バインディングを実行する。
3. 遷移前状態の退状アクションを実行する。
4. ECA の遷移アクションを実行する。
5. 遷移後状態の入状アクションを実行する。
6. 遷移後状態の入状バインディングを実行する。
7. 遷移後状態の画面定義に基づいて画面を表示する。

3.2. 構築環境

図 2 に、構築環境を用いた開発の様子を示す。UI のデザイナは、市販の画面編集ツールを用いて表示デバイスごとに画面定義を作成する。システム開発者は構築環境を用いて、アクション定義（図 2 の共通アプリケーション処理定義）を作成する。次にシステム開発者は画面定義を構築環境に取り込んで、ステートチャートを作成する。ステートチャートは表示デバイスごとに作成する。最後にシステム開発者は構築環境を用いて、バインディング定義を作成する。

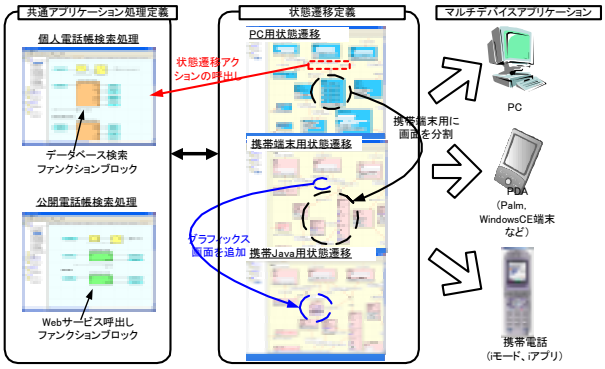

図 2　マルチデバイスアプリケーションの開発

4. おわりに

UIM は、マルチデバイスアプリケーションの開発を支援する構築環境[1]、プログラムの知識なしに UI のデザインを可能とする画面表示機能[2]、セッション引継ぎ機能[3]、等の特徴を持つ。これらの詳細については、参考文献で詳しく述べる。

参考文献

1) 中田他、ユビキタスユーザインタフェースミドルウェア(2)-構築環境ー，第 66 回情報処理学会全国大会論文集，2004

2) 寺岡他、ユビキタスユーザインタフェースミドルウェア(3)-表示画面の生成ー,第 66 回情報処理学会全国大会論文集，2004

3) 轟木他、ユビキタスユーザインタフェースミドルウェア(4)ー異種端末間での作業状態の引継ぎー，第 66 回情報処理学会全国大会論文集，2004