2003－OS－93 (19)
2003／5／9

携帯電話向けマルチアプリケーション実行環境

朝倉 義晴† 奥山 玄† 中山 義孝† 臼井 和敏† 中本 幸一†

現在の携帯電話はブラウザや Java 実行環境，GPS，カメラなどを備え，従来の携帯電話と比較して多様な使われ方がなされている．本稿では，複数のアプリケーションの並行動作を可能とする携帯電話向けマルチアプリケーションプラットフォームを提案する．このアーキテクチャは，OS に Linux，ウィンドウシステムに X Window System を用い，アプリケーションの実行制御，ウィンドウ操作の監視，リソースアクセスの管理，アプリケーションの属性の管理，アプリケーション間通信手段を提供する．この環境上において，アプリケーションは並行動作が可能となり，複数のアプリケーションによる連携動作が可能になる．

Multi-Application Platform for Mobile Phones

YOSHIHARU ASAKURA† , GEN OKUYAMA† , YOSHITAKA NAKAYAMA† ,
KAZUTOSHI USUI† and YUKIKAZU NAKAMOTO†

Current mobile phones support various functions such as a web browser and the Java Runtime Environment. These phones have been used for various purposes compared to former phones. We propose a multi-application platform for mobile phones that enables a mobile phone to have run multiple applications concurrently. This architecture, which is based on Linux and X Window System, provides functions that manage application behavior, operations of windows, access of resources. This architecture also provides a means of communication among applications. Applications on this architecture can run concurrently and work cooperatively.

1. はじめに

現在の携帯電話はブラウザや Java[1]実行環境を備え，これらを利用した様々なサービスが提供されている. 特に近年の携帯電話は, カメラ, GPS，赤外線インターフェース，Bluetooth[2]などのデバイスを備えた携帯電話も存在する．ソフトウェア面では上述のデバイスを活用するためのソフトウェアに加え，予定表，ToDo リストなど，PDA としての機能を備えるようになってきている．ユーザは機能の増加にともない，以前と比較してより多様な用途に携帯電話を利用するようになっている．そのため，携帯電話が備える複数の機能(アプリケーション)を切替えながら同時に利用できると便利な場合がある．このようなことができない携帯電話の場合，例えば，ブラウザによるブラウジング中にメーラに切替えてメールの読み書きを行うことはできない．メールを読み書きするためにメーラに切替えると，ブラウザは終了してしまう．再度ブラウザを起動した場合，ブラウザの初期メニューが表示され，メーラ起動前にブラウジングしていた WEB ページは表示されない. 一方, 複数のアプリケーションを切替えながら同時に利用できる携帯電話の場合，メーラに切替えた後もブラウザは動作可能であり，ブラウジング中の WEB ページは保持されたままである． 今後このような，複数のアプリケーションを切替えながら同時に利用する使われ方がますます望まれるようになると考えられる．

[1] Java は Sun Microsystems, Inc.の商標または登録商標である．
[2] Bluetooth は Bluetooth SIG, Inc.の商標である．

†日本電気(株) ネットワーク開発研究本部
Network Development Laboratories,
NEC Corporation

また携帯電話のアプリケーションプラットフォームという観点から見ると，単純に複数のアプリケーションを同時に利用可能とするだけでは不十分である．例えば，携帯電話が持つ排他的に利用するリソース(ネットワークやサウンドデバイス)へのアクセスを許可するアプリケーションを決めたり，優先的に画面に表示させるアプリケーションを決めたりする必要がある．そのため，これらの制御を行う仕組みも必要である．

本稿では複数のアプリケーションを切替えながら同時に利用可能とする携帯電話向けマルチアプリケーションプラットフォーム CMAP (Configurable Multi-Application Platform)の提案を行う．２章では本稿で提案するプラットフォームのモデルを述べ，３章ではアーキテクチャの構成を述べる．４章ではこのプラットフォームを複数 Java アプリケーションプラットフォームに適用した事例を述べ，５章でまとめを述べる．

2. CMAP モデル

本章では複数のアプリケーションを切替えながら同時に利用可能とする携帯電話向けアプリケーションプラットフォーム CMAP のモデルを述べる．CMAP を実現するうえで，携帯電話のアプリケーションプラットフォームという観点から，以下の問題がある．

(1) アプリケーションに対し，フォーカス割り当てや画面への表示を優先的に行う仕組みが必要
(2) アプリケーションに対し，排他的に利用されるリソースへのアクセスを制御する仕組みが必要

(1)を解決するためには，実行中のアプリケーションを制御し，複数のアプリケーションを協調して動作させる必要がある．そのため，アプリケーションの識別方法，振る舞い，画面への表示方法，アプリケーション間通信手段を決める必要がある．(2)を解決するためには，アプリケーションのリソースアクセスを管理する必要がある．

これらの問題点を解決するために，CMAP では以下の四点をモデル化している．

- アプリケーション
- アプリケーション表示
- アプリケーション間通信
- リソースアクセス

アプリケーションモデルは，CMAP 上で動作するアプリケーションの構成，識別子，状態をモデル化する．CMAP において，アプリケーションは他のアプリケーションと協調して動作する．そのため，他のアプリケーションを指定するための識別子やアプリケーションの状態制御が必要となる．アプリケーションのモデル化により，CMAP のアプリケーションの振る舞いを統一し，協調動作を可能とする．以後，CMAP 上で動作するアプリケーションを CMAP アプリケーションと呼ぶ．

アプリケーション表示モデルは, PC と比較して小型な携帯電話の画面に適したアプリケーションの表示方法をモデル化する．CMAP アプリケーションはアプリケーション表示モデルに従い，画面に表示される．アプリケーション表示のモデル化により，ユーザからのアプリケーションの見え方を統一する．

アプリケーション間通信モデルは，アプリケーション間の通信方法をモデル化する．複数のアプリケーションが連携して動作するため，アプリケーション間の通信が必要となる．アプリケーション間通信のモデル化により，CMAP アプリケーションが利用する通信手段を統一する．

リソースアクセスモデルはアプリケーション

のリソースの利用方法をモデル化する．リソースアクセスのモデル化により，CMAP アプリケーションのリソースアクセス方法を統一する．これにより，CMAP で個々のアプリケーションが利用するリソースの管理を可能とする．

2.1. OS，ウィンドウシステム

CMAP では，OS として Linux[3]を，ウィンドウシステムとして Linux で広く使われている X Window System[4]を前提としている．OS に Linux を用いる利点を以下に挙げる．

(1) アプリケーション単位のメモリ保護
(2) 動的なプロセスの生成と削除
(3) 豊富なミドルウェア，開発環境

(1)により，他のアプリケーションのバグによるメモリ破壊からアプリケーションのメモリを保護できる．(2)により，アプリケーションを必要なときのみ実行させることで，メモリの使用効率を改善できる．また，携帯電話出荷後に新規インストールしたアプリケーションの実行が可能になる．(3)により，Linux 上の豊富なミドルウェア，開発環境を利用することで開発効率の向上が図れる．

2.2. アプリケーションモデル

CMAP のアプリケーションモデルを図 1 に示す．CMAP アプリケーションは１つのプロセスから構成される．そして他のアプリケーション, ミドルウェア, ライブラリと通信や API コールを介して協調動作する．個々のアプリケーションは個別のメモリ空間を持つため，アプリケーション間のデータの共有，交換は 2.4 節で述べるアプリケーション間通信を利用する必要がある．CMAP アプリケーションは CMAP において一意なアプリケーション名を持つ．アプリケーション名はアプリケーション作成者が割り当て，アプリケーション間通信の通信先アプリケーションの指定に利用される．アプリケーション名は静的に割り当てられるため，アプリケーション作成時に通信先アプリケーションの指定が可能となる．そのため，携帯電話出荷後にユーザがダウンロードするアプリケーションと携帯電話にプレインストールされているアプリケーションとの間でアプリケーション間通信が可能となる．

CMAP アプリケーションは実行状態と停止状態の二つの状態を取りうる．実行状態はアプリケーションの通常の動作状態である．停止状態のアプリケーションは，ネットワークやサウンドデバイスなどのアプリケーション間で排他的なリソースの利用に制限がある．また，フォーカスを持つアプリケーションをアクティブアプリケーション，フォーカスを持たないアプリケーションをインアクティブアプリケーションと呼ぶ．アクティブアプリケーションは同時に最大一つしか存在しない．携帯電話のキー入力はアクティブアプリケーションに対して通知される．アクティブアプリケーションの変更を，アプリケーションの切替えと呼ぶ．

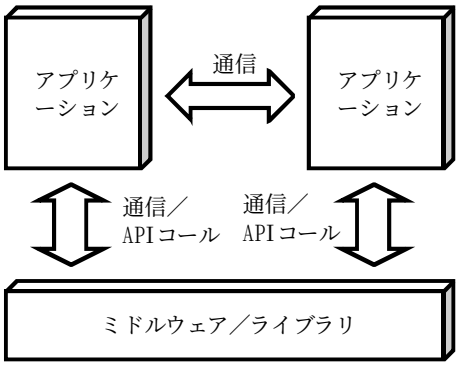

図 1 CMAP アプリケーションモデル

[3] Linux は Linus Torvalds の商標または登録商標である．
[4] X Window System は X Consortium Inc.の登録商標である．

2.3. アプリケーション表示モデル

CMAP では画面を一つ以上の矩形(表示領域)に分割し，図 2 に示すように一つの表示領域に一つのアプリケーションを割り当て，表示する．表示領域に割り当てられていないアプリケーションは表示されない．アプリケーションが表示領域に割り当てられている状態をフォアグラウンド，割り当てられていない状態をバックグラウンドと呼ぶ．また，フォアグラウンドのアプリケーションをフォアグラウンドアプリケーション，バックグラウンドのアプリケーションをバックグラウンドアプリケーションと呼ぶ．

表示領域の数はアプリケーション実行中も変更可能である．表示領域の数が増加した場合，新たに増加した表示領域の数と同数のバックグラウンドアプリケーションをフォアグラウンド状態に遷移させる．逆に表示領域の数が減少した場合，新たに減少した表示領域の数と同数のフォアグラウンドアプリケーションをバックグラウンド状態に遷移させる．

アプリケーションがユーザに情報提示やユーザインタラクションの目的でウィンドウを画面に表示する場合がある．このような目的で利用するウィンドウをダイアログと呼ぶ．ダイアログは例外的にバックグラウンドアプリケーションが画面に表示可能なウィンドウである．

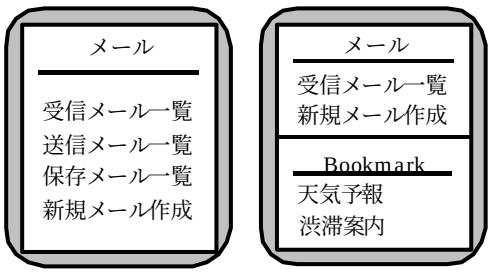

図 2 表示モデル

2.4. アプリケーション間通信モデル

アプリケーション間通信は，CMAP アプリケーション間のデータ共有，通信を実現する．アプリケーション間通信は複数のアプリケーションが協調動作をするために用いられる．アプリケーション間通信を実現する手段は複数存在する．個々のアプリケーションが独自の方法によりアプリケーション間通信を行うと，任意のアプリケーション間の通信が不可能になる．そのため，アプリケーション間通信方法を統一する必要がある．また，個々のアプリケーションが OS の提供するアプリケーション間通信手段を直接利用した場合，CMAP はアプリケーション間通信で利用するリソースを管理できない．携帯電話は PC や PDA と比較してリソースが少ないため，個々のアプリケーションが利用するリソースを制限しなければならない．そのため，CMAP アプリケーションが利用するアプリケーション間通信の実現手法を統一する．

CMAP では，以下に述べる三種類のアプリケーション間通信モデルを定義している．

- バス型
- 直接接続型
- 領域共有型

バス型は CMAP アプリケーションに接続されているアプリケーション通信バスを介して，指定したアプリケーションにデータを送信するモデルである．直接接続型は二つのアプリケーション間を通信路で直接接続し，データの送受信を行うモデルである．領域共有型は複数のアプリケーションが共有領域を介してデータの共有を行うモデルである．CMAP アプリケーションは用途に応じて，通信モデルの使い分けが可能である．

2.5. リソースアクセスモデル

CMAP アプリケーションは CMAP が提供する

リソースアクセス手段を用いてリソースアクセスを行う．これにより，CMAP は全ての CMAP アプリケーションのリソースアクセスを管理することが可能となる．空きのないリソースに対してアプリケーションのリソースアクセスが発生した場合，CMAP によりリソースアクセスの競合解決がはかられる．その結果，リソースアクセスが許可されたアプリケーションがリソースを利用できる．この時，新しくリソースアクセスをしたアプリケーションのリソースアクセスが許可された場合，競合解決以前にリソースを利用していたアプリケーションのアクセスは強制切断される．また，新しくリソースアクセスをしたアプリケーションのリソースアクセスが許可されなかった場合，競合解決以前にリソースを利用していたアプリケーションが引き続きリソースを利用できる．

3. CMAP アーキテクチャ

本章では 2 章で述べたモデルを実現するアーキテクチャを述べる．CMAP は携帯電話向けのマルチアプリケーションプラットフォームとして，以下の点を考慮しなければならない．

(1) システムにより強制的に別のアプリケーションに切替えられる場合がある．
(2) アプリケーションがダイアログを表示するときに，表示が抑止される場合がある．
(3) アプリケーションが携帯電話の排他的なリソースにアクセスしているときに，リソースアクセスが強制的に切断される場合がある．

(1)の具体例として，着信時にユーザが利用しているアプリケーションから着信処理アプリケーションに切替え，着信処理をする場合が挙げられる．(2)の具体例として，着信処理をするアプリケーションがフォアグラウンドのとき，バックグラウンドアプリケーションによる

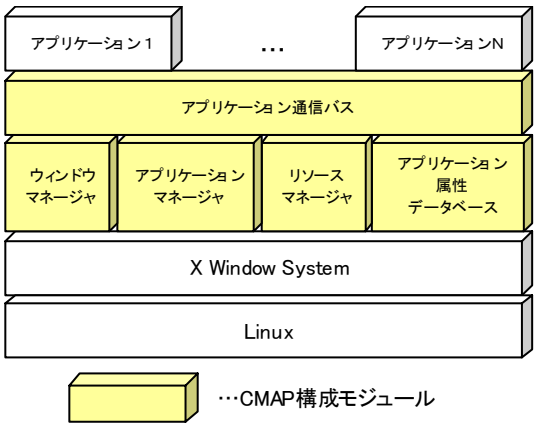

図 3 CMAP アーキテクチャ全体構成図

ダイアログ表示が抑止される場合が挙げられる．(3)の具体例として，着信時にゲームアプリケーションがサウンドデバイスを利用しているとき，ゲームアプリケーションによるサウンドデバイスへのアクセスが切断され，代わりに着信アプリケーションがサウンドデバイスを利用する場合が挙げられる．

3.1. CMAP アーキテクチャ全体構成

CMAP アーキテクチャの全体構成を図 3 に示す．CMAP は四つのモジュールとアプリケーション通信バスから構成されている．四つのモジュールと CMAP アプリケーションは，アプリケーション通信バスにより接続されている．

3.2. CMAP 構成モジュール

図 3 に挙げた CMAP を構成する四つのモジュールとアプリケーション通信バスについて述べる．

- ウィンドウマネージャ

  ウィンドウマネージャはアプリケーションのウィンドウ操作(ウィンドウの表示, 表示順序の変更, 移動, リサイズ, フォーカス取得, ダイアログの表示)を監視し, アプリケーションの不正なウィンドウ操作の無効化や適正な操作への変換を行う．またアプリケーショ

ンマネージャの指示に従い，表示領域へのアプリケーションの割り当てを行う．ウィンドウマネージャがアプリケーションのウィンドウ操作を監視することにより，CMAP として適切なアプリケーション表示状態を保つ．

- アプリケーションマネージャ

アプリケーションマネージャは CMAP として適切な実行環境を保つため，アプリケーションの実行制御を行う．アプリケーションマネージャは以下の実行制御を行う．

- アプリケーションの起動，終了
- アプリケーションの状態制御(実行状態，停止状態の遷移，アプリケーション切替え)
- ウィンドウマネージャへのアプリケーションの割り当て要求

CMAP アプリケーションはアプリケーションマネージャの制御下で動作し，個々のアプリケーションの判断による状態遷移はできない．アプリケーションマネージャは，ユーザからのアプリケーション切替え指示や着信などのアプリケーションマネージャ外部からのイベントに応じて，上述の実行制御を行う．これらの実行制御により，携帯電話としての実行環境を実現し，状況に応じて適切なアプリケーションへの切替え，表示領域への割り当て指示を行う．

- リソースマネージャ

リソースマネージャは CMAP アプリケーションによる排他的に利用されるリソースへのアクセスを管理し，CMAP アプリケーションに対しリソースアクセス手段を提供する．CMAP アプリケーションは，リソースマネージャが提供するリソースアクセス手段を用いてリソースを利用する．リソースマネージャがアプリケーションのリソース利用を管理することにより，複数のアプリケーションによるリソースアクセスの競合解決が可能となる．また，アプリケーションに対するリソースアクセスの禁止，リソースアクセスの強制切断，アプリケーション終了時の未解放リソースの解放も可能となる．

- アプリケーション属性データベース

アプリケーション属性データベースは CMAP アプリケーションの属性値を保存し，属性値の登録や問い合わせ手段をアプリケーションに提供する．アプリケーションの属性として，アプリケーション名などが定義されている．個々の CMAP アプリケーションは，アプリケーションの属性値をアプリケーション属性データベースに登録する．アプリケーション属性データベースの利用により，個々のアプリケーションによる属性値の保存や他のアプリケーションからの属性値の問い合わせ対応が不要になる．

- アプリケーション通信バス

アプリケーション通信バスはアプリケーション間通信時にデータの通信路として用いられる．全ての CMAP アプリケーションはアプリケーション通信バスに接続されており，任意のアプリケーション間でアプリケーション間通信が可能である．

3.3. アプリケーション間連携

CMAP アプリケーションは，アプリケーション間通信のバス型モデルを利用した同期処理，直接接続型モデルを利用した任意データの送受信，領域共有型モデルを利用した任意データのアプリケーション間共有が可能である．これらの同期処理，データ送受信，データ共有を利用することにより，複数アプリケーションの連携動作が可能となる．

4. 適用事例

本章では CMAP の適用事例として, 複数 Java アプリケーションプラットフォームを示す.

4.1. 携帯電話向け Java 実行環境

代表的な携帯電話向けの Java 実行環境には DoJa[5]([1])と MIDP(Mobile Information Device Profile)([2])が存在する. Java アプリケーションはこれらの Java 実行環境上で動作する. Java 実行環境は, Java アプリケーションを実行する仮想マシンと Java アプリケーションが利用するクラスライブラリから構成される. また, Java アプリケーションは状態を持ち, Java 実行環境が状態を管理する. Java アプリケーションが取りうる状態は DoJa と MIDP とで異なる. それぞれの実行環境が取りうる状態を表 1 に示す.

表 1 Java アプリケーションの取りうる状態

Java 実行環境	取りうる状態
DoJa	実行, サスペンド[6]
MIDP	Paused, Active, Destroyed

4.2. Java アプリケーションと CMAP アプリケーションの対応付け

一つの Java アプリケーションを一つの CMAP アプリケーションとして対応付ける. Java アプリケーションの実行には Java 実行環境が必要となるため, Java アプリケーションの CMAP アプリケーションとしての構成は図 4 のようになる. Java 実行制御モジュールは, CMAP アプリケーションとしての状態(実行, 停止状態)と表 1 に示す Java アプリケーションとしての状態との対応付けを行う. Java 実行制御モジュールは, CMAP アプリケーションとしての状態遷移に対応して, Java アプリケーションの状態を遷移させる. Java 実行制御モジュールにより, アプリケーションマネージャは Java アプリケーションを CMAP アプリケーションとして認識する. つまりアプリケーションマネージャは, Java アプリケーションを CMAP アプリケーションとして実行制御する.

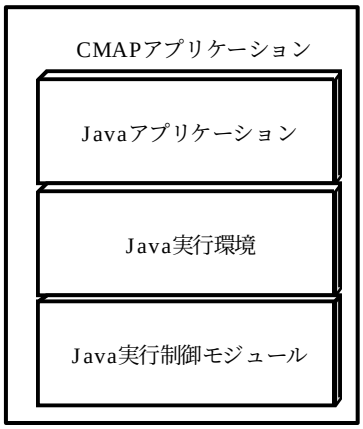

図 4 CMAP の Java アプリケーション

4.3. Java アプリケーションの管理

Java アプリケーションは携帯電話の外部から取得され, 携帯電話にインストールされる. 携帯電話にインストールされた Java アプリケーションは, Java 実行環境上で実行される. そして不要になった Java アプリケーションは携帯電話からアンインストールされる. CMAP にはこのような Java アプリケーションの管理を行う機能が存在しないため, Java アプリケーションを管理するモジュールが新たに必要となる. そこで JAM(Java Application Manager)を新規モジュールとして CMAP に追加する. JAM は Java アプリケーションのインストール, 実行, 終了, アンインストール機能をユーザに提供する.

[5] DoJa は NTT ドコモの商標または登録商標である.
[6] DoJa-2.x プロファイルで定義されている待ち受けアプリケーションの取りうる状態は除いている.

4.4. Java アプリケーション間連携

Java アプリケーション間や Java アプリケーションと非 Java アプリケーション間において，アプリケーション間通信を利用した連係動作が可能である．Java アプリケーションからアプリケーション間通信を利用するために，Java 実行環境が Java アプリケーションにアプリケーション間通信の機能(クラスライブラリ)を提供する必要がある．このアプリケーション間通信の機能を実現するクラスライブラリは，CMAP 独自のクラスライブラリとして定義される．Java アプリケーションがアプリケーション間通信を利用することにより，個々の Java アプリケーションが単独で動作するだけではなく，複数の Java アプリケーションによる連携動作も可能となる．

4.5. 主要アプリケーションの Java アプリケーション化

従来の携帯電話のブラウザやメーラは，一部の携帯電話を除いて，携帯電話搭載 OS のネイティブアプリケーションとして実現されていた．その理由は，従来の携帯電話では複数の Java アプリケーションを切替えながら同時に利用することができず，また，Java アプリケーション間の連係動作を実現する機能が存在しないためであった．そのためブラウザやメーラなど，他のアプリケーションとの連携動作が必要なアプリケーションを Java アプリケーションで実現することはできなかった．

CMAP では今まで述べてきたように，複数の Java アプリケーションを切替えながら同時に利用可能であり，また，Java アプリケーション間で連係動作が可能である．従って CMAP では，従来の携帯電話の環境では難しかったブラウザやメーラなどの携帯電話の主要なアプリケーションを，Java アプリケーションとして実現可能である．これらのアプリケーションを Java アプリケーションとして実現することにより，携帯電話出荷後のアプリケーションの入れ換えが可能となる．そのため，携帯電話出荷後のアプリケーションの機能拡張や新たに発見されたバグは，アプリケーションの入れ替えにより対処可能である．Java アプリケーションの入れ替えには，JAM が持つ Java アプリケーションのインストール，アンインストール機能を利用でき，入れ替えのために新たな仕組みを必要としない．

5. まとめ

本稿では携帯電話向けのマルチアプリケーションプラットフォーム CMAP について述べた．CMAP では複数のアプリケーションを切替えながら同時に利用可能であり，アプリケーション間連係動作も可能である．また適用事例として，複数 Java アプリケーションプラットフォームを示した．

現在 PC 上の Linux 環境において試作を行い，操作性や機能を検証中である．今後の予定として，検証結果を反映させた操作性，機能の追加，改良を行う．また携帯電話実機，もしくは，性能や使い勝手の近いデバイス上に CMAP を移植し，性能評価を行うことも必要である．

参考文献

[1] 株式会社 NTT ドコモ，“i アプリコンテンツ開発ガイド for 504i(詳細編) 第 2.0 版”，株式会社 NTT ドコモ (2003).

[2] Sun Microsystems，“Mobile Information Device Profile(JSR-37)”，Java Community Process，Java2 Platform，Micro Edition，1.0 (2000).