社団法人　情報処理学会　研究報告
IPSJ SIG Technical Report

2006－MBL－36（36）
2006－UBI－10（36）
2006／2／16

携帯端末における S/W アドインのための アプリケーション管理方式

清原　良三　　岡田　英明　　三井　聡

三菱電機（株）情報技術総合研究所

携帯電話の高機能化およびユビキタス時代の中心端末化において, 後から自由にソフトウエアをダウンロードしたいという要求がある．Java アプリケーションがその例である．しかし, Java は実行時間の課題と機能の制限が大きいという問題がある．そこで, ネイティブコードで書いたソフトウエアのアドインが必要と考える．しかし, アプリケーションの管理の面を考えると追加するプログラムに比べソフトウエアの更新量が大きくなるという心配がある．アプリケーションの管理の面では, 起動制御と状態管理およびアプリケーション間の連携の実装方式にソフトウエアの更新量は依存する．

本論文では, ソフトウエア更新量を抑えるためのソフトウエアのアドイン方式を提案する．

A Study of Software Add-in Method for Mobile Phones

Ryozo Kiyohara, Hideaki Okada and Satoshi Mii

MITSUBISHI ELECTRIC CORPORATION
INFORMATION TECHNOLOGY R & D CENTER

Due to increasing services of cellular phones(e.g. i-mode),it requires software add-inn functions. Java execution environment is one of the add-in functions. But Java execution environment has two problems those are execution time and limited functions. So, we think it needs to install the program which is written in C/C++ codes. But that has data size problem which depends on application manager.

In this paper, we propose new software add-in method.

1　はじめに

近年の携帯電話の普及に伴い，携帯電話はメガネ，腕時計に続いて違和感なく身に着けるようになったデバイスと言っても過言ではない．多くの人が身につけることから, 競争力強化のためにユーザが要求する機能をメーカが搭載することにより高機能, 多機能化が進んできている．

また携帯電話を近距離無線機能や，有線接続で他の機器と接続することにより, 携帯電話を鍵として利用したり, 周辺の機器から情報取得をするなど様々な利用方法も出現してきている．即ち, 今後ますますユビキタスコンピューティングの中心的なデバイスの一つに携帯電話がなると予想される．

ユビキタスコンピューティングの中心的なデバイスとして, 以下の 3 点をすべて満足することは現

状の携帯電話ではできないケースが多いが, 今後必須の機能と考えられる.

- 各種デバイスを有線または無線にて接続可能にすること.
- 各種機器を制御するためのアプリケーションが動作すること.
- 前記機能をユーザ自身が追加できること.

これらの機能はパーソナルコンピュータなどでは普通に実現できている機能であるため見過ごされやすい. しかし, 実際に多くのユーザ層を持つ携帯電話ではその実装は難しい.

本論文では, このようなソフトウエアの携帯電話発売後のアドインの課題を整理した上で, 研究動向を整理する, さらに, ソフトウエアのアドインの機能要件を定義し, 機能要件を満足させるための最も大きな課題と考えるアプリケーション管理方式に関して考察し, 要件を満足するアドイン方式を提案する.

2 携帯端末におけるソフトウェアアドイン

アドインソフトの代表として, Javaがある. Java仮想マシンをサポートする携帯電話 [1] では, Java言語で書いたアプリケーションプログラムに関してのみアドインが可能となるが, 制約条件も多くやりたいことが常にできるとは限らない. また, 特定のハードウエアを制御するようなドライバを記述するのもJavaでは困難である.

そこで, ユビキタス端末として携帯電話を活用するためにネイティブコード (C/C++) で開発したアプリケーションやドライバをインストール可能にする必要がある. 携帯電話のソフトウエアは組込みソフトウエアの一種であり, 組込みソフトウエア特有の問題がある.

ネイティブコードをユーザの手元で書き換えるサービスとして, ソフトウエア更新サービスが登場している [2]. このサービスは不具合の修正に限られているが, 不具合の修正でもコードのダウンロード時間と書き換え時間で相当な時間がかかるケースがあるため, ソフトウエアの更新時間を短くする必要がある.

また, 外部からのアプリケーションをインストールするためには, そのアプリケーションが利用するリソースの競合をどのように処理するのかという課題がある [3]. 後から追加するアプリケーションの開発者は, 他のアプリケーションがどのようにリソースを使うのかを把握して開発しているとは限らない. そのため, 競合制御が重要な課題となる.

次に, セキュリティの問題がある. セキュリティに関しては外部からプログラムを入れるという行為そのもので, 様々な脅威に対する弱点を作ることにつながる. そのため, ハードウエアプラットフォームから考えて行く必要がある [4]. また, コードをチェックすることにより証明書を付与し, 安全であることを示すための効率的な方式に関しても研究されている [5].

以下, アプリケーション管理に焦点を絞ってソフトウエアのアドインに関して検討する.

3 アプリケーション管理の課題

最近の携帯電話ではマルチタスク機能がサポートされた機種が出てきてはいるが, そのほとんどは1画面を1つのアプリケーションが占有するタイプの携帯電話である. 例えば音を出すようなアプリケーションがあるとする. このアプリケーションは, 起動中に電話がなってしまえば, たとえ先に起動されていようとも音を出してはいけない. このようにその状態によって動作を変える必要がある. このような競合管理には以下に示す課題がある.

起動制御 フォアグラウンドになれるアプリケーションは一つであり, しかもアプリケーションによってはメモリリソースを大量に消費するため, アプリケーション間での起動制御を行う必要性もある. また, ユーザの入力をトリガにしてアプリケーションが起動される場合もあれば, 外部からのイベントをトリガにする場合もあり, タイミングを考慮した設計が重要となる.

各アプリケーションは, 起動のときには自らアプリケーションマネージャに起動要求を出し, 起動条件が満足されていれば, 起動される. リソースのア

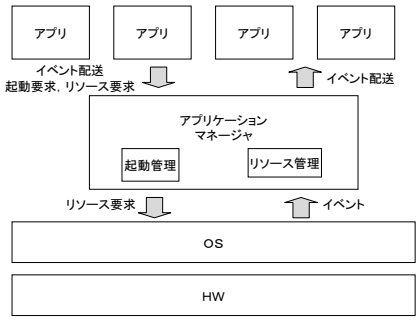

図 1: アプリケーションマネージャ

クセスに関しても, アプリケーションマネージャに要求を出し, アクセス条件が満足されていれば, リソースにアクセスできる. その例を図 1に示す. アプリケーションの数を n とすると, 単純な実装をすると, n 個のアプリケーションが自己の 2 重起動要求もも含めて考えると n 個のアプリの起動要求に対する動作を表現する必要があり, 計 n^2 の動作を示す必要がある. またアプリの状態によって起動の許可, 不許可をする場合もある. 起動に関連する状態数を p とすると, pn^2 の動作を示す必要がある. アプリケーションを m 個追加するとすれば, 起動制御を行うために次式に示す情報を追加する必要がある.

$$kidou(n,m,p) = p(n+m)^2 - p(n^2) \quad (1)$$
$$= 2pnm + zm^2 \quad (2)$$

アプリケーションの追加想定数や, メモリの容量も考えるとたかだか一桁であり, $n >> m$ と想定できる. 従って約 $kidou(n,m,p) = 2pnm$ 個の動作に関する追加がアプリケーション自身の他に必要になる.

アプリケーションの状態ごとのイベント制御 アプリケーションは多くの場合, 画面を持ち, 画面ごとにキーイベントを受け付ける, 受け付けないとか, 外部イベントに対してどう動作するべきかなど決めなければならない. このような動作は図 2のような状態遷移表で示される. これをプログラムコードに変えて実行する.

状態	非起動時	起動要求応答待ち	起動中	リソース獲得要求中	...
イベント					
起動要求	....	....	....	....	
起動要求応答	....	....	....	....	
リソース獲得要求	....	....	....	....	
....	....	....	....	....	
....	....	....	....	....	
電源キー長押し	....	....	....	....	

図 2: 状態表の例

一つのアプリケーションが持つ平均の状態数を s とする. また起こりうる平均のイベント数を e とすると, 次式で示す状態遷移の動作を示すコードがアプリケーションのコードに含まれる必要がある.

$$jousen(s,e) = se \quad (3)$$

例えば, アプリケーション数が 20 個あり, 起動の制御に関連する状態数の平均が 10 個であり, 2 個までアプリを追加可能とすると, 800 個の動作を記載できるようにしておく必要ある. 実際にはもっと多くの動作を管理しておかなければならない. また, アプリごとには, 状態数を 30 個程度としてもイベントが 20 あれば 600 となり, それがアプリの数あるため膨大となる.

アプリケーション間の連携 各アプリケーションの受けるイベントが別のアプリケーションからくるイベントの場合もある. このようなアプリケーション間の連携も状態遷移表で示した方が良い. しかし, 実質的にはアドインのソフトウエアに関してはインストール済みのソフトウエアが知ることは困難であるため, アドインソフトウエアに関しては許可できることを想定しておく必要があり, 開発時の課題と言うことができる.

4 関連研究

携帯電話のソフトウエアモジュールを発売後にユーザ自身の手でダウンロード, インストールする方法については, まず Java 実行環境が実現されて

いる．Java では JavaVM そのものを JAM というモジュールで制御し，Java プログラムからは JAM を制御できないようにするなど，機能を制限することによりセキュリティを確保し，アプリケーション競合に関しては予め決まった形でしか利用できなようにすることと，Java アプリケーションが同時には１つしか実行できないようにすることによりアプリケーション競合の問題を解決している [1].

しかしながら，複数の JavaVM を動作させることなどを想定すると，予め競合の問題を解いておくことができるわけではなく，ネイティブコードのアドインに関して Java と同様の手法を採用すると，ネイティブコードであるという意味が薄れてしまう．

ネイティブコードのダウンロード，書き換えという観点では，ソフトウエア更新サービスがある．後から追加するコードで如何に事前のコードに影響を与えないかというソフトウエア構成法 [6] もあるが，あくまでも不具合の修正であり，アプリケーション競合の問題の解決にはなっていない．

アプリケーション競合の問題が解けたとして，複数の既存のプログラムに修正を入れるという観点からは [6][7] は有効と考えるが，競合そのものの解決手法は示していない．

アプリケーションの競合問題を解く手法としては，リソースの利用方法をモデル化し，これによりリソースアクセス方法を統一して，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手法が提案されている [8].

これは一種の状態遷移表モデルであるが，開発の効率化という観点から提案された手法であり，後から追加するアプリケーションにとっては，他のアプリケーションとの関係などをすべてケアせねばならず，リソースを使う複数のアプリケーションが発売後インストールされるとなると，どちらのアプリケーションも意図通りに動作させられるとは限らないという問題がある．

また，アプリケーション単位にリソースに関する記述方法を用意するという提案もされている [3].

この手法も開発効率化の観点から考えられている．各アプリケーションが他のアプリケーションを意識することなく独立にリソースの使い方について記述できるのであれば有効な方法と言える． しかし，文献 [3] では，どのように記述すれば本課題を解決されるかが示されておらず，未解決である．

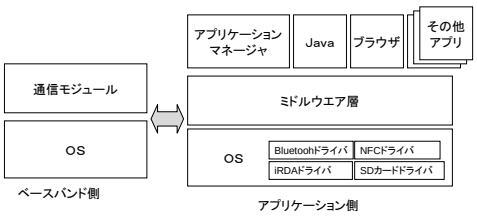

図 3: 想定ソフトウエア構成

本論文では，アプリケーションごとにリソースの使い方を定義する方法に基づいたアプリケーション管理方式で，ソフトウエアの更新量を抑える方式を検討する．

5 アドイン方式の提案

5.1 想定する携帯電話

想定する携帯電話機は，いわゆる 3G 携帯電話とする． 外部との通信機能としては，回線接続，パケット接続という広域を利用した接続機能の他に，FeliCa などの NFC 機能や，Bluetooth，無線 LAN，IrDA などの近距離無線である． さらに USB ポートや，外部メモリカードを保持する． アプリケーションとしては，Java などの安全なアドインソフトウエアと，携帯電話が持つ個人の情報にアクセスするためのソフトウエアおよび前記通信機能を利用するためのアプリケーションという一般的な携帯電話を想定する (図 3)．また，アプリケーションマネージャなど起動時から必要で利用頻度の高いものはフラッシュ ROM 上で直接実行する形式であり，一部のアプリケーションやデータは，フラッシュ ROM から RAM に実行時にロードして実行されると想定する．

5.2 アドインに関する機能要件

携帯電話のソフトウェアのアドインに関するユーザ側の機能要件を以下と考える.

- ユーザがアドイン実行時にリセット不要であること
- ユーザのダウンロード時間を極力小さくすること.
- 途中で電池切れを起こさない, または電池切れを起こしても復帰できること.

開発者側の機能要件は以下と考えられる.

- 既にインストール済みのソフトウェアを知らなくても開発できること

前節で述べた方式そのままでソフトウエアをアドインする場合は以下の情報を携帯電話にダウンロードし書き換える必要がある.

- 起動制御情報
  インストール済みのデータの更新が必要
- インストール済みアプリケーションプログラム
  インストールするアプリケーションからイベントをもらうようなアプリがある場合は, そのアプリの状態遷移関連のプログラムまたはデータの更新が必要
- アプリケーションプログラム

しかし, 通常インストール済みのアプリケーションに関して情報を持っていることは少ないと考える. また, プログラムの構成によってはアドイン時にダウンロードするデータ量が大きくなるという課題がある. また, 実行中のプログラムコードや定数的なデータの書き換えはフラッシュ ROM 上の直接実行を想定するとできない場合が多い.

5.3 提案方式

5.3.1 アプリケーション起動制御

提案する方式は, リセット不要にすることと, 書き換え量を少なくするために以下の特徴を持つ起動制御方式とする.

1. 起動制御関連の処理は起動制御に関しての情報をテーブル形式で保有し, プログラムはフラッシュ ROM, データはフラッシュ ROM から RAM に読み出す形とし, プログラム実行中にもその ROM 領域を書き換えられる領域とする.
2. テーブルは予め追加可能なアプリケーションの数に対応するフィールドを用意しておく形とし, 追加によるテーブルの大きさが変わらないこととする.
3. 追加するアプリの情報は情報なし即ち, アプリ追加時にフラッシュ ROM の消去不要のの状態としておく.

また, 電池切れなどの発生時にも問題を起こさないために, 以下の特徴を持たせる.

1. アドイン時にはダウンロードとインストールのフェーズを分離する. ダウンロード中の電池切れなどは再ダウンロードしたり, ダウンロードしきれなかった部分は途中から再ダウンロードするなどの方式とする. インストール時には, 電池の残容量をチェックしてからフラッシュ ROM の書き換えと RAM 上のデータの書き換えを行う.
2. 書き換えの成功, 不成功を示すフラグ情報を不揮発のメモリにおくことにより, 前記書き換えの終了判定を行う.

起動制御に関しては, 優先度の定義と使用リソースの宣言をファイル上で定義することにより, インストール時に起動制御のテーブルを書き換えるための情報を動的に生成することとする. この方式により, インストール済みのアプリケーションの情報は不要になるが, 制約条件も増えたことになる.

5.3.2 アプリケーションの状態遷移

各アプリケーションの状態遷移に関してはアプリケーションが抱え持つため, あまりフラッシュ ROM の書き換えを気にする必要はない. いかに書き換える情報を少なくするのかが課題である. 状態遷移表を作成するためのフレームワークの提供で詳

細を知らなくとも作成できるようにするのが良い. しかしながら, 未知のアプリケーションからのイベントの判断までは書くことができない. また, 既存のアプリケーションへの処理してもらいたいイベントの追加方法が明確にならない.

そこで, やはりアプリケーションの状態遷移も起動制御同様に, 状態遷移テーブルを RAM 上に持つことと, 予め追加可能なアプリケーションの数とイベントの数を想定したテーブルを各アプリが保持することにより対処する. 発売後の追加アプリとの間で全く連携を許可しない場合は, 余分なテーブルは不要とする.

アプリケーションの連携を要する場合は, 相手側のアプリケーションの詳細を知る必要があるが, ここも連携を許可するアプリケーションはその呼び出し方をおよびイベントの指定方法を定義しておき, その情報をダウンロードするのみで, 携帯電話機上でデータを生成し, データを書き換える方式とする. 多くの場合はアプリケーションの状態遷移の修正を必要とせず書き換え時間も短くすることができる.

6 まとめ

本論文では, ソフトウエアのアドインに関してアプリケーション管理の問題点を提起し, アプリケーション管理の点を含めてもアドインソフトウエアのダウンロード量を抑え, 開発上も他のアプリケーションを気にしなくても良い方式を提案した. 本方式ではリセットすることなく, 巨大な起動制御や状態遷移表といった情報を少量のデータのダウンロードで書き換えることができる.

しかし, 未実装であり, 今後実際の携帯電話ソフトウエアを対象に実装を行い, ダウンロード量が押さえられることと, 実行時のオーバヘッドが問題にならないことを検証していく予定である.

また, 起動制御に関して優先度を導入し, インストール済みアプリケーションの情報を知らなくとも良いこととした. その結果, トレードオフで一定の範囲の優先度しか実行できなくなる. アドインしたいソフトウエアとしての要求事項を明確にした上で, この制約があっても問題ないかを検証する予定である.

また, より深刻な問題であるセキュリティ面に関しても検討を進め, データのダウンロード量および書き換え時間の少ない安全なアドイン方式技術の確立を目指す.

参考文献

[1] 大関江利子, 山田和宏, ”携帯機組み込み Java の実用化”, NTT DoCoMo テクニカルジャーナル, Vol.8, no.1, pp.16-21, Apr. 2001

[2] S. Hoshi,A. Ichinose,Y. Nose,A. Hosokawa,M. Takeichi,and E. Yano, ”Software update system using wireless communication”, NTT DoCoMo Technical Journal, vol.5, no.4, pp.36-43. Mar. 2004.

[3] 奥山玄, 才田好則, 臼井和敏,”アプリケーション競合管理方式”, 情処 67 回全大, PP.(1-35)-(1-36). March. 2005.

[4] Sribaths Ravi and Anand Raghunathan,”Security in Embedded Systems:Design Challenges,” ACM trans. on Embedded Computing Systems, Vol.3, no.3, pp.461-491, 2004

[5] 太田賢、吉川貴、中川智尋、稲村浩, ”セキュア携帯機のためのアプリケーション自己監視方式”, 情報処理学会, DICOMO2005, pp.725-728, 2005

[6] 清原良三, 栗原まり子, 古宮章裕, 高橋清, 橘高大造, ”携帯電話ソフトウェアの更新方式, ”情報処理学会論文誌, vol.46, no.6, pp.1492-1500, June. 2005.

[7] 清原良三, 栗原まり子, 三井聡, 木野茂徳, ”携帯電話ソフトウェア更新のためののバージョン間差分表現方式”, 電子通信学会論文誌, Vol.J89-B, No.4, 2006(予定)

[8] 朝倉義晴, 奥山玄, 中山義孝, 臼井和敏, 中本幸一, ”携帯電話向けマルチアプリケーション実行環境”, 情処 93 回 OS 研究報告, Vol.19, PP.135-142. May. 2005.