6N-2

モバイル JAVA アプリケーション用 共通記述仕様の提案とそのコードジェネレータの開発

中村　弘寿　　塚本　享治
東京工科大学メディア学部

1 はじめに

近年、モバイル端末における Java 環境、特に携帯電話用の Java 環境が急激に進歩し、身近な存在となった。しかし、携帯電話用の Java 環境はキャリアごとに仕様が違うため各キャリアにあわせ別々に開発が行われており、効率的な開発がなされていない。そこで、本稿ではひとつのソースファイルから各キャリアの仕様に見合ったソースを生成するためのモバイル Java アプリケーション用の共通記述仕様を提案し、その仕様に沿って記述したソースから DoJa 用と MIDP 用のコードを生成するジェネレータの開発結果について述べる。

2 モバイル Java アプリケーション開発の問題

2.1 モバイルツールの Java 環境

携帯電話のような小さな組み込み機器でのアプリケーションは、その機器向けの機能に絞って個別に開発される。そのため、多くの場合、機器間でアプリケーションの互換性はない。しかし、多様化するユーザの要望に対応するため、ユーザ独自で機能を選択できる仕掛けと、様々な開発者がアプリケーションを作れるように共通した仕様を持つ必要がある。

2.3 問題と解決策

(1)携帯電話 Java で用いられているプロファイルは DoJa と MIDP の二つの規格に分かれており、各規格の中でもキャリア、機種ごとに機能が拡張されており、仕様が統一されていない。
(2)規格、仕様が統一されていないため、開発が規格、仕様ごとに分業化されており、効率的な開発が行われていない。

開発が多岐にわたるこれらの問題は、開発対象ソースコードの記述をひとつにすることができ、そこからさまざまな規格や仕様に合わせて生成することができれば、開発対象をひとつにすることで解決できるのではないかと考えた。
図 1 は共通仕様をベースにした開発方法である。

"The proposal of the common specification for mobile Java applications, and development of its code generator." , by Hirotoshi NAKAMURA, Michiharu TSUKAMOTO, Tokyo University of Technology.

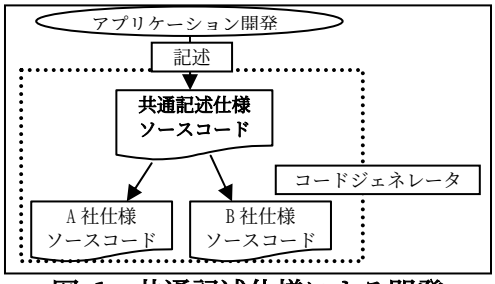

図 1　共通記述仕様による開発

3 モバイル Java アプリ用共通記述仕様

3.1 設計方針

共通仕様の設計にあたり、キャリア独自規格の部分を含めると違いが大きくなり、共通仕様が複雑化する。また、クライアント・サーバ型アプリケーションを対象とすると、通信を行う API の部分も含まれ、規格を拡張した部分に依存する部分が大きくなる。そのため、キャリアが独自で拡張した仕様は対象とせず、DoJa と MIDP の基本部分の規格とスタンドアロン型アプリケーションを対象とした。

XSL を用いてジェネレータを実装することを想定して、共通記述仕様を XML で設計した。

3.2 共通記述仕様の設計

モバイル Java アプリケーションのプログラムを次の 5 つの部分に分けた。

(1)基本部分
プログラムの名前や、インポート文などの基本的な情報を持つ。
(2)コンストラクタ部分
表示部分やキーイベント部分、スレッド部分で必要になる関数等の初期化を行う。
(3)表示部分
アプリケーションでグラフィックの描画に関する機能を担う。
(4)キーイベント部分
ユーザによるキーアクションにイベントを割り当てる部分。
(5)スレッド部分
スレッドを用いることにより、キーイベント待ち状態であってもほかのイベントを処理する機能を担う。

ただし、(1)(2)の部分に関しては、(3)(4)(5)の内容次第で自動的に出力可能であるため、(3)(4)(5)の動作についてのみ記述する。

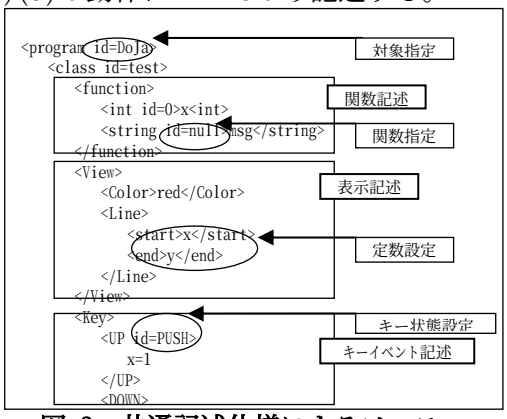

図 2　共通記述仕様によるソース

基本的なアプリケーションの場合、次のように記述する。
(1)関数記述部分では、これはプログラムの中で用いられる関数を記述する。属性値をもって初期値とする。
(2)表示記述部分ではスクリーンに表示する内容を記述する。直線や四角形などの図形の定数は入れ子のタグで表現する。
(3)キーイベント記述部分ではキーごとにイベントを記述する。属性値でイベントを発生させるキーの状態を示す。

このほかに、スレッドが存在する場合はスレッド記述部分が存在する。イベント記述の中では y=x+2;などの数式・関数はそのまま記述する。

この記述によって、入出力などの機種・規格に依存する部分と、関数・数式の機種に依存しない部分とを切り離して扱うことができるようになる。

4 コードジェネレータの開発

4.1 ジェネレータの構成と機能

ジェネレータは XSL で実装することにした。ジェネレータは図３のような構造になっており、以下のように動作する。
(1)基本仕様に基づいて、ユーザが作成したプログラムの定義ファイルを読み込む。
(2)XSLT 変換を行って、プログラムの構造をした中間ファイルを生成する。
(3)(2)で出力された中間ファイルを読み込み、XSLT 変換を行って、出力仕様にあわせたソースコードを生成する。

変換の対象を拡大したい場合は、(3)において、中間ファイルからそれぞれの仕様を生成する XSL ファイルを追加することで対応する。

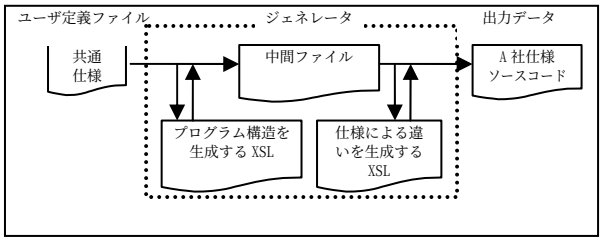

図 3　コードジェネレータの構造

4.2 実装した機能

ジェネレータで、構造化された中間ファイルから、各仕様に向けてソースコードを生成する際に実装した機能は下記の通りである。
(1)関数部分:int、string
(2)表示部分:色の指定、直線・四角形、テキストの描画、イメージファイルの表示、
(3)キーイベント部分：上、下、左、右、0～9 までの 14 のボタンに対してイベントを追加可能。
その他には、スレッドの利用や Class や Method の拡張性を実現した。

5 検証と考察

3 章で述べた共通記述仕様にそってジェネレータを開発した。その結果、上下左右を押して画像の表示エリアを移動し、数字ボタンによって、表示するマップを切り替える地図表示アプリのソースコードを生成することができた。

この開発においては、キーにイベントをあたえ、それに合わせて表示エリア、対象を変えるというアプリケーションを、モバイル Java ソース記述の知識がなくとも作成することができた。

実用度の高いアプリ生成に利用できたことにより、通常のスタンドアロン型のアプリケーション開発においても有用性が高いといえる。

6 おわりに

共通仕様を用いることで、開発効率の向上が期待できるものが開発できた。この方法による効果をあげていくためには、実装する機能、対象となる規格を拡張していかなくてはならない。今後は、対象となる規格を拡大していくため、機種に依存される表示など部分の互換性を高めていく方法と、通信を利用するアプリケーションの開発方法に改良を加えていきたい。

終わりに本研究に関して貴重な助言をいただいた本学橋本勉教授、棟上昭男教授、及び金澤典子講師に感謝いたします。

参考文献

[1]DoCoMoNet, http://www.nttdocomo.co.jp/
[2]イーバレー:“組み込み Java がわかる”,技術評論者,2003