5J-07

MDAアプローチを適用したスマートフォンアプリケーションプロトタイプ自動生成手法

チョウ　カンウ†　　岸知二†

早稲田大学　大学院経営システム工学研究科†

1 研究の背景と問題定義

効率的にスマートフォンアプリケーション開発を行うには、クロスプラットフォームが重要な課題になっている。この問題を解決するために、ハイブリッドアプローチやクロスコンパイラアプローチなどの方法が多く提案されている[1]。一方、モデル段階でプラットフォームの違いを吸収し、ソースコードを自動生成するモデル駆動開発を用いた開発手法も近年注目されている。

しかし、モデル駆動開発をスマートフォンアプリケーションの開発に適用するには、以下のような二つの面の問題点が存在している。

- GUI 設計:どのようなモデリング方法や変換ルールを用いて，高度な操作性やプラットフォーム特性に拝領した GUI 設計ができるか。
- 内部クラス構造及び処理ロジック:どのようなモデリング方法でプラットフォーム間ライブラリの違いを吸収できるか、また、どこまでモデリングすれば、全体の開発効率が高いか。

そこで、本研究では、従来手法[3]で提案されたモデル駆動開発手法を踏まえ、モデル変換で得られたモデルに対し、適用ルールを用いてモデルの特殊化を行うことで、スマートフォン特有な操作性やプラットフォーム特性に十分配慮した GUI 設計を導出するとともに、各 GUI コンポーネントをコントロールするクラス構造をパターンとして定義し、自動生成ルールとして作成することで、スマートフォンアプリケーションプロトタイプを自動生成する手法を提案する。

2 従来研究

上記の問題に関し、多くの研究が行われた。

Ayoub ら[2]は、GUI 情報のモデリングの問題に対して、プラットフォームから独立したスマートフォンアプリケーションメタモデルを作成し、作成されたメタモデルに基づいてオブジェクト図を作成することで各画面の画面設計を行う手法を提案した。しかし、この研究は GUI の静的な構造面に着目した研究であり、画面遷移などの動的な面については言及していない。また、各 GUI コンポーネントの位置、サイズ情報は画面サイズを固定し、絶対値座標を用いて指定したため、異なる画面サイズ、解像度に対応できない問題点が存在している。

G.botturi ら[3]は、UML2 プロファイルを定義し、プロファイルを基にオブジェクト図、クラス図、ステートマシン図を利用したモデリング手法を提案した。しかし、この研究では[2]と同様で各 GUI コンポーネントの位置、サイズ情報の精確性が足りない。また、画面遷移方法の定義が単純で、スマートフォン特有な操作性について言及されていない。さらに、各 GUI コンポーネントをコントロールする内部クラス構造の設計において、PIM 段階においてプロファイル要素で共通設計し、モデル変換段階で構造マッピングを行う手法を提案したが、外部ライブラリ利用によるクラス構造の変化について考慮されていない。そのほか、この研究、Android と WindowsPhone についての研究であり、iOS においての実現可能性が実証されていない。

3 提案手法

上記の先行研究を踏まえ、本研究では、まず、プラットフォームに依存しないスマートフォン共通の操作性を表現するために、スマートフォン GUI 設計に特化したメタモデル UML を拡張して定義する。定義されたメタモデルからモデルを作成し、プラットフォームごとの PSM モデルへ変換する。その後、共通化できない各 GUI コンポーネントに対して適用ルールを作成し、モデル変換後のモデルを特殊化することによって、スマートフォン特有の操作性を十分に配慮したモデルを導出し、得られたモデルから自動生成ルールによって、クラス構造を決定し、プロトタイプを自動生成する。以上を図 1 に示す.

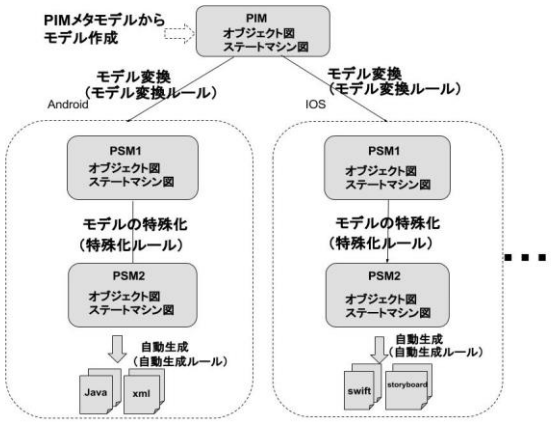

図 1.プロトタイプ自動生成手法の概要

3.2 メタモデルの定義

本研究では、まず、各プラットフォームが提供している標準ライブラリから GUI コンポーネントを抽出する。抽出された各 GUI コンポーネント及びコンポーネント間の関係から各プラットフォームのメタモデルを定義する。定義された PSM メタモデルから、「画面また画面に配置される共通機能をもつ GUI コンポーネント」という抽出基準を用いて、モデルの共通化を行い、共通 GUI コンポーネントを抽出する。抽出されたものを用いて PIM のメタモデルを定義する。

3.3 モデル作成

定義された PIM メタモデルを利用して、以下に示す PIM モデルを作成する

■ オブジェクト図：各画面に配置する GUI コンポーネントモデリングする。多種多様な画面サイズ、解像度に対応するために、各 GUI コンポーネントは画面全体と比較してパーセント指定を用いて、位置情報、サイズ情報を定義する。

■ ステートマシン図：オブジェクト図で定義された各画面の遷移順序と遷移方法を定義する。

3.4 モデル変換

3.3 で定義された PIM モデルに対し、MDA アプローチのモデル変換を適用し、モデル変換を行う。この段階では PIM で表現された共通の GUI コンポーネントを、対応する各プラットフォームの GUI コンポーネントに変換する。この段階のモデル変換はメタモデルレベルの変換で、3.2 のモデル共通化する際に定めた共通化ルールを基づいて変換を行う。

3.5 モデル特殊化

3.2 において共通化できない GUI コンポーネントに対して、適用ルールを定義する。この段階では 3.4 の変換で得られた 2 種類のインスタンスレベルのモデルを分析し、定義された適用ルールと比較して、一致すれば、モデルの自動変換と修正を行う。

各 GUI コンポーネントの適用ルールの定義方法は、まず、各プラットフォームの開発者ガイドからこの GUI コンポーネントの利用説明を確定する。この利用説明から、GUI コンポーネントの適用できるモデル構造を導出し、適用ルールとして定義する。

簡単な例を挙げる。iOS において、「UINavigationController」という特有な GUI コンポーネントが存在する。iOS の開発者ガイドによると、この GUI コンポーネントは「ある画面中のアイテムを選択されることによって、次の画面へ遷移する状況において、簡単に前の画面に戻ることができる」と説明されている。そこで、この「UINavigationController」に対して、「二つまたはそれ以上の画面が相互遷移関係であり、遷移方法画面中のアイテムを選択されること」という適用ルールを定義する。特殊化ツールは、モデル変換後のモデルを分析し、このルールに一致する状況を見つければ、この GUI コンポーネントを適用するようにモデルの修正を行う。

3.6 ソースコードの自動生成

得られたモデルから、自動生成ルールを適用してソースコードを自動生成する。自動生成ルールの定義方法は 3.2 で抽出された各プラットフォームの各 GUI コンポーネントに対し、コードの生成方法及び実装するためのクラス構造を定義する。これに基づいて、自動生成ルールを作成する。生成されたソースコードは各 GUI コンポーネントを表す外部の静的な GUI 設計情報と各画面及び画面内各 GUI コンポーネントをコントロールする内部クラスに関するコードが含まれている。

4 評価

プロトタイプの自動生成手法の実現可能性を確認するために、Android、iOS 二つのプラットフォームを対象として、実験を行った。メタモデルの定義、また、モデル変換ルール、モデル特殊化ルール、ソースコードを自動生成ルールの実装を行った。定義されたメタモデルからモデルを作成し、実装した変換ツールを用いて、「メモ帳アプリ」、「商品管理アプリ」という二つのアプリケーションのプロトタイプを自動生成することによって、実現可能性を確認した。

5 今後の課題

本研究では、MDA アプローチを適用したプロトタイプ自動生成手法を提案し、Android、iOS における実現可能性を確認した。一方、本研究で行われた評価実験は網羅できる GUI コンポーネントの数がまだ少なく。今後より多くの GUI コンポーネントに対応する必要がある。また、既存プロトタイピングツールとの連携性を考え、モデリング容易性を向上する必要がある。

参考文献

[1] Mounaim LATIF, Younes LAKHRISSI, El Habib NFAOUI, Najia ES-SBAI: “Cross platform approach for mobile application development: a survey”, IT4OD International Conference ,2016.

[2] Ayoub Sabraoui,Ismail khriss: “GUI code generation for Android applications using a MDA approach”, ICCS International Conference ,2012.

[3] G.Botturi,E.Ebeid,F.Fummi:“Model-driven design for the development of multi-platform smartphone applications”, Specification & Design Languages (FDL), 2013.