インタラクティブソフトウェアの自動生成に関する研究

M2014SE008 吉川 翔平
指導教員：野呂 昌満

1 はじめに

インタラクティブソフトウェアの開発効率の向上を目的としてプログラムの自動生成が利用されている [4]．単一ソース開発を支援するクロスプラットフォーム開発環境においても自動生成技術は実用化されているが，これらは主に MVC におけるビューとモデルのアクション起動を対にして適応したものである [3]．我々は，多様な開発環境と多様な実行時環境のすべての可能な組み合わせに対する単一モデル開発を目的として，web アプリケーションとスマートデバイス等のネイティブアプリケーション (以下，まとめてインタラクティブソフトウェアと呼ぶ) の共通アーキテクチャを提案している [5].

本研究の目的は，提案する共通アーキテクチャ上のコンポーネントの自動生成について考察することである．すなわち，アーキテクチャの実現はアプリケーションフレームワークとの認識に立ち，自動生成可能なコンポーネントを同定し，その生成方法について考察する．

一般にアプリケーションフレームワークのホットスポットをカスタマイズするためのコードの自動生成は難しいとされているが，アプリケーションの仕様を与えて自動生成が可能なものも存在する．これらは，アプリケーションの仕様からプログラムコードを自動生成できるものと，仕様を与えてアプリケーションフレームワークを生成できるコンポーネントに分類できる (以下，生成されるものをサブアプリケーションフレームワークと呼ぶ)．すなわち，共通アーキテクチャのコンポーネントは，アプリケーションフレームワーク上で，

- ホットスポットとなるもの
- フローズンスポットとなるもの
- サブアプリケーションフレームワークとなるもの (以下，SAF コンポーネントと呼ぶ)

から構成されることが確認できた．共通アーキテクチャの目的は現存する任意の開発環境で作成したソフトウェアを任意の実行時環境で稼働させることなので，これら複数の環境要因を整理し自動生成について考察するために，モデル駆動アーキテクチャ [2] に着目する．

2 関連技術

2.1 インタラクティブソフトウェアの共通アーキテクチャ

インタラクティブソフトウェアの現存する参照アーキテクチャを調査し，プライマリーコンサーンを MVC とした場合，いくつかの横断的関心事があることを確認した．すべての横断的関心事を内包するアーキテクチャを定義することを目的として，共通アーキテクチャはアスペクト指向アーキテクチャとして定義した (図 1)．共通アーキテクチャは，1) MVC コンサーン，2) UI コンサーン，3) ビジネスロジックコンサーン，4) データアクセスコンサーン，5) イベント管理コンサーン，6) 画面構築コンサーン，7) 画面遷移コンサーン，8) 表示コンサーンの 8 つの横断的関心事によって規定されるアスペクトから構成されている．特定の横断的関心事のいくつかを指定して織込むことで，調査したアーキテクチャすべてを生成できる．

2.2 モデル駆動アーキテクチャ

モデル駆動アーキテクチャ (以下，MDA) はプラットフォームの仕様と PIM(Platform Independent Model) から，プラットフォームに依存した PSM(Platform Specific Model) を生成するためのシステムのアーキテクチャである．例えば，PIM としては UML 図で書かれたオブジェクトモデル等が想定されている．PSM は具体的なプログラミング言語，例えば Java 等で書かれたコードになる．このさい，プラットフォームは実現に使用するプログラミング言語になる．通常，複数の PSM に変換するためのアーキテクチャを定義したものである．

3 アプリケーションフレームワークの自動生成

3.1 共通アーキテクチャに基づくインタラクティブソフトウェアの開発

共通アーキテクチャに基づく開発では，まず，アプリケーションの仕様を与え，サブアプリケーションフレームワークを含むホットスポットコードの一部を生成する．つぎに，ホットスポットカスタマイズコードを記述する．モデルやビューは典型的なホットスポットであり，開発するアプリケーションに依存する．一方，イベントの授受やモデルからのビューの生成はイベント，モデル，ビュー，それぞれの表現形式，すなわち，アプリケーション上でのデータ型が標準化できれば定型コードとしてフローズンスポットとなる．他方，共通アーキテクチャでは，モデル，ビュー，コントローラ，それぞれを状態遷移機械としてモデル化する．これらの状態遷移機械は SAF コンポーネントであり，UML の状態機械図などから生成可能である．生成されたアプリケーションフレームワークでは，状態遷移はフローズンスポットとしてコード化され，状態遷移のさい起動されるアクションはホットスポットとなる．さらに，イベント変換に関しては外部イベントと内部イベントの対応を与えればコード生成可能である．この分類を図 1 に示す．

まとめると，共通アーキテクチャに基づく開発工程は以下の通りとなる．

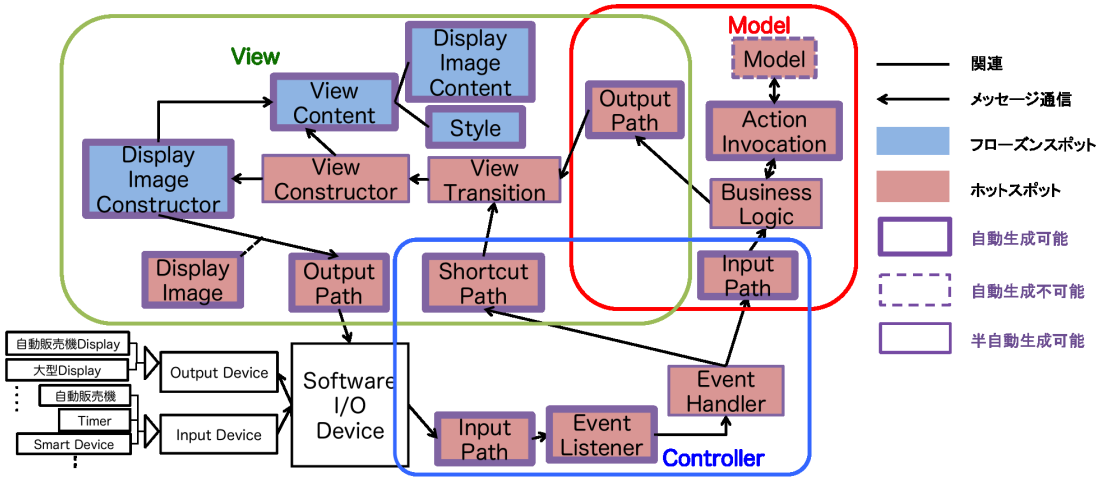

図 1　共通アーキテクチャのコンポーネントの分類

- 自動生成可能なホットスポットに与えるアプリケーション仕様の記述とカスタマイズコードの自動生成
- SAF コンポーネントの仕様記述からのサブアプリケーションフレームワークの自動生成
- サブアプリケーションフレームワークのホットスポットを含むホットスポットカスタマイズコードの記述

3.2　共通アーキテクチャのコンポーネントの自動生成

フローズンスポットとホットスポットの一部が自動生成可能であることは，これまでに述べた通りである．共通アーキテクチャの目的を考えた場合，開発環境や実行時環境に関連する要因群が MDA のプラットフォーム特定にいかに関係するかを考察することが鍵となる．SAF コンポーネントを含むホットスポットの一部については，これに加えて，生成系への入力形式も考慮しなけらばならないことがわかった．

3.3　フローズンスポットの自動生成

フローズンスポットコンポーネントと，そのコンポーネントの自動生成を実現するさいのプラットフォーム，PIM，PSM について整理し，表 1 に示す．

表 1　フローズンスポットの自動生成

フローズンスポット	プラットフォーム	PIM	PSM
Display Image Constructor	ビュー定義形式	ビジター、イテレータ、コマンドパターンを用いて定義された抽象クラスレベルのオブジェクトモデル ⇨	ビュー定義形式に関連する具象クラスを追加したオブジェクトモデル
	Display Image Constructorの実現に使用するプログラミング言語	ビュー定義形式に関連する具象クラスを追加したオブジェクトモデル ⇨	当該プログラミング言語で記述されたコード
View Model (View Content, Display Image Constructor, Style)	View Modelの実現に利用するプログラミング言語	コンポジット、デコレータ、ブリッジパターンを用いて定義されたオブジェクトモデル ⇨	当該プログラミング言語で記述されたコード

基本的にフローズンスポットはアプリケーションの仕様に依存せず，その記述プログラミング言語が決まれば実現できる．したがって，フローズンスポットコードの自動生成のプラットフォームはプログラミング言語，PIM はオブジェクトモデルとなり，PSM は当該プログラミング言語で記述されたコードとなる．共通アーキテクチャのコンポーネントのうち，Display Image Constructor を除くすべてのコンポーネントのプラットフォームはプログラミング言語となる．Display Image Constructor に関しては，Display Image Constructor が生成する Display Image のビュー定義形式に依存する．Display Image Constructor の自動生成に関しては，プラットフォームをビュー定義形式とプログラミング言語の 2 つとした MDA を考えなければならない．以下，プラットフォームが 1 つの MDA を OD-MDA(One-Dimentional-MDA，一次元 MDA)，プラットフォームが複数の MDA を MD-MDA(Multi-Dimentional-MDA，多次元 MDA) と呼ぶ．

3.4　ホットスポットの自動生成

サブアプリケーションフレームワークを含むホットスポットの自動生成を実現するさいのプラットフォーム，PIM，PSM について整理し，表 2 に記した．

表 2　サブアプリケーションフレームワークを含むホットスポットにおける自動生成

SAF コンポーネント	プラットフォーム	PIM	PSM
Event Listener	仕様の表現形式 (ex. 状態遷移表)	オブジェクトモデルの対応 ⇦	特定の表現形式で記述された仕様
	プログラミング言語	オブジェクトモデルの対応 ⇨	当該プログラミング言語で記述されたコード
STM (Business Logic, View Transition, Event Handler)	仕様の表現形式 (ex. イベントを対にした表)	ステート、コマンドパターンを用いて定義されたオブジェクトモデル ⇦	特定の表現形式で記述された仕様
	STMの実現に利用するプログラミング言語	ステート、コマンドパターンを用いて定義されたオブジェクトモデル ⇨	当該プログラミング言語で記述されたコード
View Constructor	View Constructorの実現に利用するプログラミング言語	ファクトリーメソッド、イテレータ、コマンドパターンを用いて定義されたオブジェクトモデル ⇨	当該プログラミング言語で記述されたコード

SAF コンポーネントである状態遷移機械からの自動生成について考える．状態遷移機械を自動生成するための仕様の入力は UML 状態機械図が適切であると一般的には考えられる．状態機械図を PIM，実現言語をプラットフォームとして MDA を定義するのが一般的であろう．一方で，Excel などを用いて状態遷移表を定義し，それを PIM とすることも可能である．しかし，状態遷移表は状態機械の一表現形式であり，むしろ PSM とすべきである．PSM から PIM への変換は設計またはアーキテクチャ回復と捉えることができる．生成系を設計するさいに使用者の使い勝手等を考えた場合，使用ドメインに特化した表現を用いる必要性は否定できない．この事情を考慮して，アーキテクチャ回復を逆モデル変換と捉えて考察する．状態遷移機械は MDA と RMDA(Reverse Model Driven Architecture, 逆モデル変換アーキテクチャ) の組み合わせで定義できる．以下，このアーキテクチャを RMDA-MDA と呼ぶ．

Event Listener も同様に，プラットフォームを定義形式 (例えば，Excel で記述したイベント対応表) とプログラミング言語の 2 つとした RMDA-MDA に基づき自動生成可能である．このさい，RMDA では対応表からイベントのオブジェクトモデルを回復し，MDA でそのオブジェクトモデルから特定のプログラミング言語で書かれたコードを生成する．

4　MDA に基づく自動生成の分類

共通アーキテクチャの各コンポーネントについて自動生成を考えた結果，MDA に基づく自動生成は以下の 3 種類に分類できることがわかった．

- OD-MDA
- MD-MDA
- RMDA-MDA

OD-MDA は PIM とプラットフォームの仕様を読み込み，PSM を自動生成するためのアーキテクチャである．

MD-MDA は PIM と複数のプラットフォームの仕様を読み込み，PSM を自動生成するためのアーキテクチャである．図 2 は，赤枠と青枠で囲まれた OD-MDA から構成される MD-MDA を示している．赤枠の MDA では，PIM と特定のプラットフォームの仕様を読み込み，PSM を生成する構造を示している構造を示している．青枠の MDA では，PIM(赤枠の MDA で生成された PSM) と異なるプラットフォームの仕様を読み込み，PSM を生成する構造を示している．多くの場合，MD-MDA では，プラットフォームはそれぞれ独立であり，生成の順番は，それを問わない．

RMDA は，PSM からプラットフォームを指定することにより PIM へのモデル変換を実現する逆モデル駆動アーキテクチャである (図 3 の赤枠)．前述のように，特定の設計や実装からアーキテクチャを復元するアーキテクチャ回復と位置付けられる．

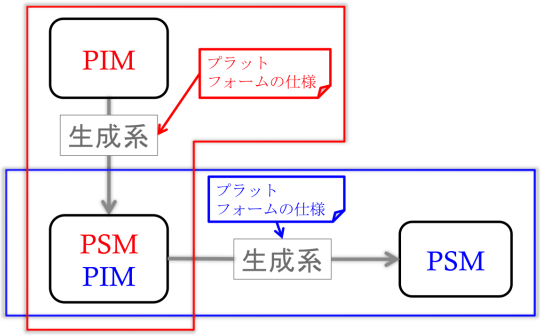

図 2　MD-MDA に基づく自動生成

図 3 に示す RMDA-MDA は，RMDA と OD-MDA の組み合わせである．図 3 では，赤枠では RMDA による生成を行ない，その後，この出力を PIM とした MDA に基づく自動生成が行なわれる．

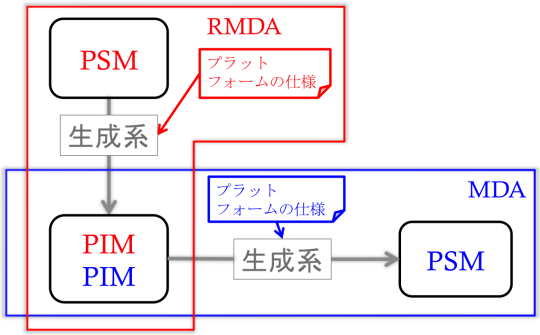

図 3　RMDA-MDA に基づく自動生成

5　考察

前節で示した 3 種類の MDA について，典型的なコンポーネントを例として，生成系の実現について考察する．

5.1　OD-MDA に基づく自動生成

フローズンスポット自動生成の典型的な例として，共通アーキテクチャの View Content 自動生成について述べる．View Content 生成のさいのプラットフォームはプログラミング言語であり，PIM はクラス図，PSM は当該プログラミング言語で記述されたクラス定義となる．プラットフォーム仕様は構造エディタで用いられた逆解析スキーマ (Unparsing Schema) を用いた具象構文情報付加の手法 [1] を応用できる．多くの UML ツールではオブジェクトモデルアクセスのための API を持っている．オブジェクトモデルをたどるためのイテレータが用意されている場合が多いので，これを利用してオブジェクトの木をたどりながら逆解析スキーマを参照してフローズンスポットのソースプログラムを生成できる．さらに，PIM をクラス群の静的構造と制御構造を定義したオブジェクトモデルとすることで，操作的意味のコード化が可能である．本稿では静的構造をクラス図，制御構造をコマンドパターンを用いたオ

ブジェクトモデル，コマンドごとの操作的意味をアクティビティ図で記述することを想定している．

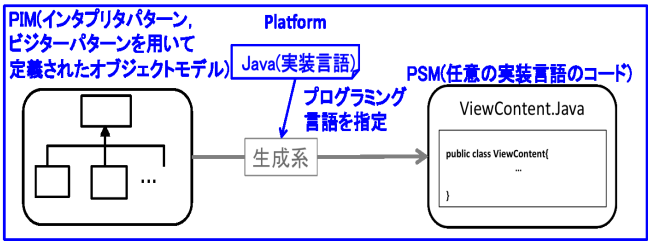

図 4　View Content の自動生成

5.2　MD-MDA に基づく自動生成

MD-MDA を適用すべき例として，Display Image Constructor 自動生成について述べる．プラットフォームはビュー定義形式とプログラミング言語とし，MD-MDA を定義すれば生成可能となる．

プラットフォームがビュー定義形式のさいの，PIM は多層型のコマンドパターンを使って定義したオブジェクトモデル，PSM は特定ビューに変換するためのコマンドサブクラスを付加したオブジェクトモデルとなる．プログラミング言語をプラットフォームとした場合の，PIM はビジターパターン，イテレータパターン，およびコマンドパータンを適用したオブジェクトモデルとなる．PSM はこれらパターンの持つ操作的意味を当該言語に翻訳したプログラムとなる．自動生成の概要を図 5 に記した．

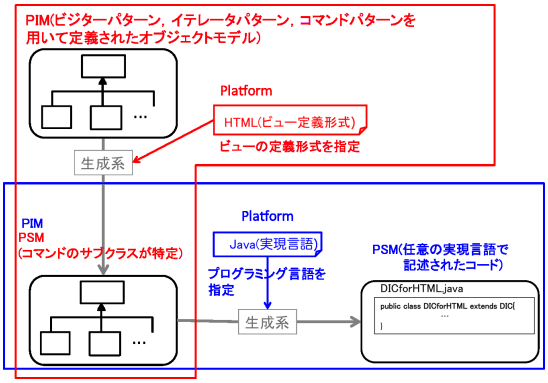

図 5　Display Image Constructor の自動生成

5.3　RMDA-MDA に基づく自動生成

SAF コンポーネントである Event Handler の生成について述べる．RMDA のプラットフォームは状態遷移機械の記述形式，PIM は UML 状態機械図で表現した状態遷移機械，PSM は特定の使用者を想定した表現形式，例えば Excel で書かれた状態遷移表など，とする．MDA のプラットフォームはプログラミング言語で，PIM は UML 状態機械図で表現した状態遷移機械，PSM は当該プログラミング言語で記述した状態遷移機械のプログラムコードとなる．自動生成の概要を図 6 に記した．

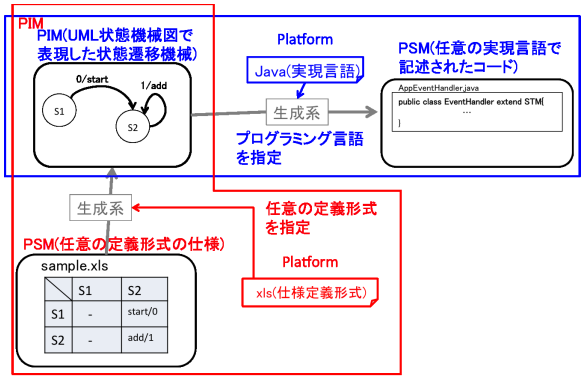

図 6　Event Handler の自動生成

6　おわりに

本稿では，我々の提案する共通アーキテクチャの実現について考察した．共通アーキテクチャのすべてのコンポーネントについて自動生成の可能性について議論した．アーキテクチャをアプリケーションフレームワークとして実現したさいに，すべてのフローズンスポットは自動生成可能であることが確認できた．ホットスポットについても，アプリケーションの仕様から自動生成または半自動生成可能なコンポーネントが存在することを確認した．共通アーキテクチャの目的である，“複数の開発環境と実行時環境の可能な任意組み合わせでアプリケーションが稼働すること”，に着目し，MDA に基づく自動生成について考察した．共通アーキテクチャのコンポーネントが OD-MDA，MD-MDA，または RMDA-MDA で生成可能であることを確認した．今後，これらのモデル駆動アーキテクチャに基づき，生成系を実現して行きたい．

参考文献

[1] D. Notkin, “ Gandalf: Software development environments,” *IEEE Trans. Soft. Eng.* , vol.12, no. 12, 1986, pp. 1117-1127.

[2] Object Management Group, “ *MDA*, ”http://www.omg.org/mda/, 2015.

[3] S. Allen *et al.* , *Pro Smartphone Cross-Platform Development: IPhone, Blackberry, Windows Mobile and Android Development and Distribution,* Appress, Berkely, CA, 2010.

[4] S. Lazetic *et al.* , “ A Generator of MVC-based Web Applications, ” *World of Comp. Sci. & Info. Tech. J.,* Vol. 2, No. 4, 2012, pp. 147-156.

[5] 江坂篤侍，野呂昌満，沢田篤史，“ インタラクティブソフトウェアの共通アーキテクチャの提案，” ソフトウェアエンジニアリングシンポジウム 2015 論文集，2015, pp. 137-144.