スマートデバイスアプリケーションと Web アプリケーションの統一的開発に関する研究
―異なる開発環境を例として―

2011SE158 松下竜巳　2011SE259 田中真人　2011SE273 坪根祐治
指導教員：野呂昌満

1　はじめに

スマートデバイスや Web ブラウザの多様化に伴い，Web アプリケーションやスマートデバイスのネイティブアプリケーション (以下 Web and SD App. と呼ぶ) の実行時環境ならびに開発環境が多様化している．これらの開発環境や実行時環境では，実行時ライブラリの API や開発環境の UI がそれぞれ異なる．異なる UI や API をすべて把握する事は難しく，多くの環境で開発を進める事は，開発者に多大な負担を強いる事になる．この問題を解決するために，我々の研究室ではインタラクティブシステムのための共通アーキテクチャが提案されている．この共通アーキテクチャは高い抽象度で定義されたアーキテクチャであり，様々な環境におけるアプリケーションを説明可能なものである．

本研究の目的は，特定の環境から独立した Web and SD App. の開発支援である．Web and SD App. 開発において特定の環境から異なる環境における技法を別の環境で応用可能にすることを目的とする．

この目的を達成するために，共通アーキテクチャから一般的な開発プロセスである，抽象開発プロセスを定義します．また，共通アーキテクチャの抽象コンポーネントと特定の環境で開発されたアプリケーションのアーキテクチャの具象コンポーネントとの対応関係を整理します．それにより，共通アーキテクチャ基に定義された抽象開発プロセスを介して特定の環境の開発プロセスである具象開発プロセスを導出します．

我々は，ネイティブアプリケーション開発において多くの実行時環境に対応可能な PhoneGap と Web アプリケーション開発において多くのブラウザに対応可能な Rudy on Rails 間でそれぞれの環境で開発したアプリケーションの相互変換が可能であることを確認した．

2　共通アーキテクチャ

共通アーキテクチャはインタラクティブシステムを対象とし，様々な環境のアプリケーションアーキテクチャを説明可能としている．共通アーキテクチャに基づき一般的なアプリケーションの開発プロセスを定義する．

2.1　共通アーキテクチャのコンポーネントの説明

図 1 に共通アーキテクチャを示す．

以下に，図中の各コンポーネントについて説明する．

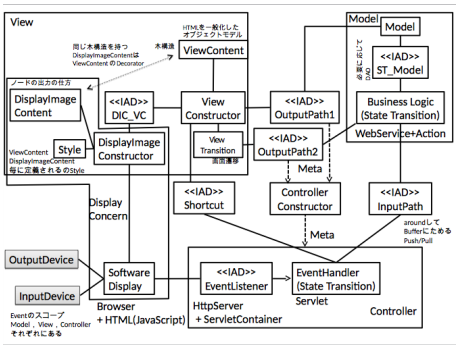

図 1　共通アーキテクチャ

Controller

- EventListener：イベントを内部形に変換し，アプリケーション使用者からのイベントを受け取り EventHandler に通知.
- EventHandler：自身の状態と EventListener からのイベントによって，Model または View にイベントを通知.

Model

- Business Logic：Model や View 更新のロジックを状態遷移機会として抽象化したもの.
- Model：Database.

View

- ViewTransition：画面遷移を抽象化したもの.
- ViewContent：View のデータ構造を定義.
- ViewConstructor：Model を参照し，ViewContent のインスタンス化を行う.
- DisplayImageContent：ViewConstructor で生成した View の各ノードの出力形式のデータ構造を定義.
- Style：外部表現 (色，サイズ等) を保持.
- DisplayImageConstructor：ViewConstructor で生成した View に外部表現と出力方式を結合し View を構築.

2.2　共通アーキテクチャによって規定される開発プロセスの定義

アプリケーションの開発プロセスを単純化すると，View の定義，Event の定義，Action の定義，Event に対する Action の定義，Event に対する View の定義の 5 つのス

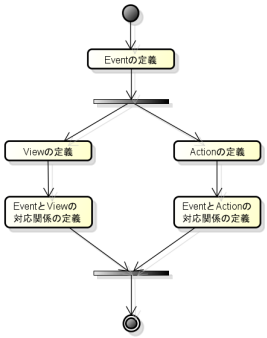

図 2 Web and SD App. の一般的な抽象開発プロセス

テップに分解出来る．図 2 に共通アーキテクチャに基づき定義した一般的な抽象開発プロセスを示す．

以下で，共通アーキテクチャに基づき定義した一般的な抽象開発プロセスを説明する．

- Event の定義：入力時に発生するイベント，アクション実行時に発生するイベントを定義．
- View の定義：画面構成の内部表現を定義．
- Action の定義：Business Logic を定義．
- Event と Action の対応関係の定義：イベントに対応して実行されるアクションを定義．
- Event と View の対応関係の定義：イベントに対応して表示される画面を定義．

3 背景技術

3.1 PhoneGap

多様なスマートデバイス向けのプラットフォーム上で動くネイティブアプリケーションの開発手法として，クロスプラットフォーム開発が提案されている．クロスプラットフォーム開発環境を用いることで，複数の実行時環境上でも同じサービスを提供できるアプリケーションをワンソースで開発が可能である．

クロスプラットフォーム開発環境は複数存在するが，PhoneGap は他のクロスプラットフォーム開発環境よりも多くのモバイル OS をサポートしている [1]．PhoneGap では，既存の Web アプリケーション開発技術を用いて開発を行う．JavaScript で記述された PhoneGap が提供するライブラリを使用する事によりデバイス独自の機能 (カメラ，GPS 等) が利用可能となる．[4]

PhoneGap のアーキテクチャのコンポーネントの説明

図 3 に，PhoneGap のアーキテクチャを示す．
以下に，図中の各コンポーネントについて説明する．

Controller

- EventHandler：通知されたイベントに応じて View and EventListener または BusinessLogic に通知する．

View，Controller

- View and EventListener：画面入力とユーザからのイベントの受け取り．外部表現として HTML を用いており View の役割と Controller の役割が含まれる．

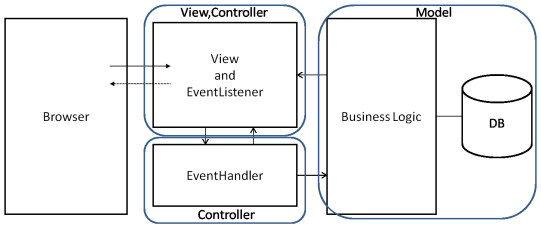

図 3 PhoneGap のアーキテクチャ

Model

- BusinessLogic：イベントに対応する DB アクセスや View 更新を記述．

3.2 Ruby on Rails

Ruby on Rails は，MVC アプリケーションアーキテクチャに基づいた Web アプリケーション開発のためのアプリケーションフレームワークである．Web の標準技術を前提とすることにより，異なる Web ブラウザ上でも同じ画面を表示可能としている．また，Ruby on Rails で作成したアプリケーションはサーバー上に配置され，ブラウザを介して実行される．

Ruby on Rails のアーキテクチャのコンポーネントの説明

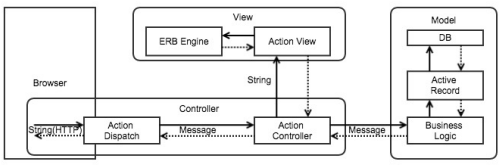

図 4 Ruby on Rails のアーキテクチャ

図 4 に Ruby on Rails のアーキテクチャを示す．
以下に，図中の各コンポーネントについて説明する．

Controller

- Action Dispatch : Browser から受け取った HTTP リクエストを解釈し，オブジェクトへのメッセージ変換を行う．
- Action Controller：自身の状態とイベントに応じて，Model または View にイベントを通知．

Model

- Active Record : DB の種類に関わらず，DB を操作することを可能とする．

View

- Action View：HTML を構築するためのテンプレート．
- ERB Engine：Action View から．erb(テンプレート) を受け取り，HTML を生成する．

4 抽象コンポーネントと具象コンポーネントの対応付け

我々は，共通アーキテクチャの抽象コンポーネントと PhoneGap，Ruby on Rails のアプリケーションアーキテクチャである具象コンポーネントを整理する．

4.1 PhoneGap の具象コンポーネントと抽象コンポーネントとの対応付け

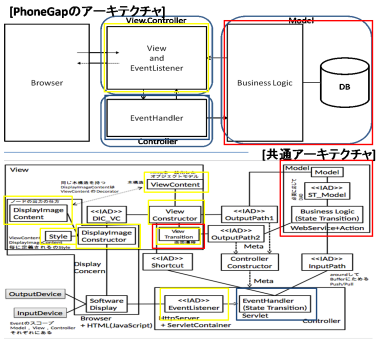

図 5 PhoneGap の具象コンポーネントと抽象コンポーネントとの対応付け

PhoneGap の具象コンポーネントと抽象コンポーネントとの対応付けを図 5 に示す．

- Controller：View and EventListener を用いる．EventListener でイベントを受け取り，EventHandler を指定して通知を行う．共通アーキテクチャでは，EventListener，EventHandler に対応．
- View：View and EventListener を用いる．View and EventListener 内で外部表現，内部表現を持ち，それらを基に View 構築を行う．共通アーキテクチャでは，ViewTransition，ViewContent，DisplayImageContent，ViewConstructor，Style，DisplaImageConstructor に対応．
- Model：BusinessLogic と DB により構成される．共通アーキテクチャでは，BusinessLogic と Model に対応する．共通アーキテクチャでは，ViewTransition，ViewContent，DisplayImageContent，ViewConstructor，Style，DisplaImageConstructor に対応．

4.2 Ruby on Rails の具象コンポーネントと抽象コンポーネントとの対応付け

Ruby on Rails の具象コンポーネントと抽象コンポーネントとの対応付けを図 6 に示す．

- Controller：ActionDispatch と ActionController を用いる．HTPP 通信をイベントとして ActionDispatch が受け取り，ActionController を指定して通知を行う．これにより，ActionDispatch は EventListener に，ActionController は EventHandler に対応付けられる．
- View：ActionView と ERB Engine を用いる．Ruby を埋め込んだ.erb(テンプレート) は View の構造を定義し，ERB Engine はテンプレートを解析し外部表現である HTML を生成する．これにより，ActionView は ViewContent と DisplayImageContent，ERB Engine は ViewContentor と DisplayImageContentor に対応付けられる．また Style には，見た目を記述した CSS が対応付く．
- Model：BusinessLogic，ActiveRecord，DB を用いる．BusinessLogic と ActiveRecord は BusinessLogic に，DB は Model に対応付けられる．

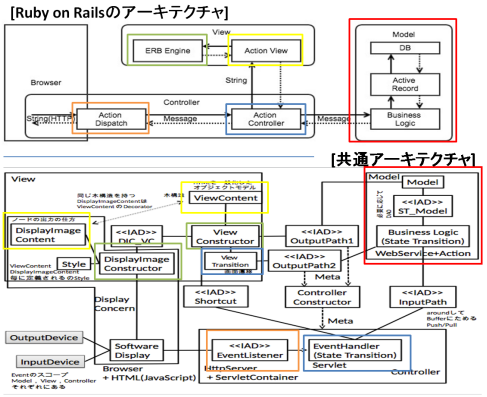

図 6 Ruby on Rails の具象コンポーネントと抽象コンポーネントとの対応付け

5 具象開発プロセス

抽象コンポーネントと具象コンポーネントの対応付けにより，共通アーキテクチャに基づいた具象開発プロセスを導出する．

5.1 PhoneGap の具象開発プロセス

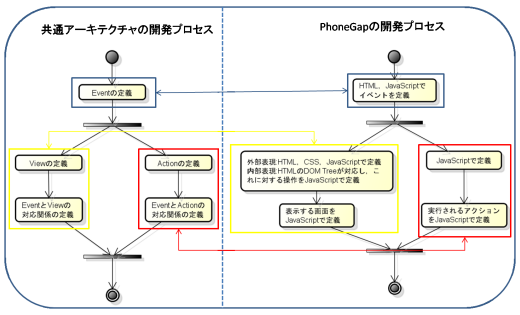

図 7 共通アーキテクチャに基づいた PhoneGap の具象開発プロセス

図 7 に共通アーキテクチャに基づいた PhoneGap の具

象開発プロセスを示す．以下に，PhoneGap の具象開発プロセスを説明する．

- Event の定義：PhoneGap はアプリケーションを HTML，JavaScript で定義することから，イベントを文字列によって定義．
- View の定義：外部表現を HTML，CSS，JavaScript を用いて定義．内部表現は HTML の DOM Tree が対応し，これに対する操作は JavaScript で定義．
- Action の定義：Application Logic を JavaScript を用いて定義．
- Event と Action の対応関係の定義，Event と View の対応関係の定義：イベントに対応して実行されるアクション，イベントに対応して表示される画面を JavaScript で定義．

5.2 Ruby on Rails の具象開発プロセス

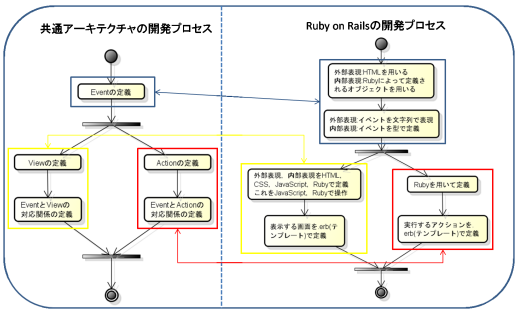

図 8 共通アーキテクチャに基づいた Ruby on Rails の具象開発プロセス

図 8 に共通アーキテクチャに基づいた Ruby on Rails の具象開発プロセスを示す．以下に，Ruby on Rails の具象開発プロセスを説明する．

- Event の定義：外部表現に HTML を用いる．内部表現に Ruby によって定義されるオブジェクトを用いる．外部表現として文字列で表現されるイベントと，内部表現として型で表現されるイベントを定義．また，外部表現のイベントと内部表現のイベントの対応関係を定義．
- View の定義：外部表現を HTML，CSS，JavaScript，Ruby を用いて定義．内部表現も HTML，CSS，JavaScript，Ruby が対応し，これに対する操作は JavaScript，Ruby で定義．
- Action の定義：Application Logic を Ruby を用いて定義．
- Event と Action の対応関係の定義，Event と View の対応関係の定義：イベントに対応して実行されるアクション，イベントに対応して表示される画面を．erb（テンプレート）で定義．

6 考察

共通アーキテクチャの抽象コンポーネントと特定のアプリケーションのアーキテクチャの具象コンポーネントの対応関係を整理した．これにより，具象コンポーネント間の対応関係が明確となった．本研究では，PhoneGap，Ruby on Rails のアプリケーションのコンポーネントと共通アーキテクチャのコンポーネントの対応関係を整理した．例として，PhoneGap の EventHandler と Ruby on Rails の ActionController は，共通アーキテクチャの EventHandler に対応付けられ，それぞれが互換である事を示した．共通アーキテクチャとの対応関係を整理することで，環境間のコンポーネントの対応関係を整理出来る．

具象コンポーネントと抽象コンポーネントの対応関係により，共通アーキテクチャを基に定義した抽象開発プロセスを介して PhoneGap，Ruby on Rails それぞれの具象開発プロセスを導出した．具象開発プロセスは抽象開発プロセスを介して導出されるので，具象開発プロセス間は対応付くといえる．これにより，ある環境の技法が別の環境で応用可能である．

具象コンポーネント間の対応関係と，具象プロセス間の対応関係が整理された事から，特定の開発プロセスで開発したアプリケーションを異なる環境のアプリケーションに自動変換が可能であると考える．すなわち，開発者は Ruby on Rails の開発手法により PhoneGap のアプリケーションを開発可能となる．

7 おわりに

本研究では，PhoneGap と Ruby on Rails で開発したアプリケーションアーキテクチャについて，共通アーキテクチャとの対応関係を整理し，具象プロセス間の互換性を整理し，具象プロセス間の相互変換性であると示したことで特定の環境から独立した開発支援になると考える．

参考文献

[1] Adam M.Christ, “Bridging the Mobile App Gap,” *Sigma*, vol. 11, no. 1, pp. 27-32, 2011.

[2] Andre Charland and Brian Leroux, “Mobile application development: web vs. native,” Communications of the ACM, vol. 54, no. 5, pp. 49-53, 2011.

[3] K. Sokolova et al., “Towards High Quality Mobil Applitions: Android Passive MVC Architecture,” International Journal on Advances in Software, Vol.7, No.1&2, 2014, http://www.iariajouenals.org/software.

[4] アシアル株式会社，PhoneGap 入門ガイド，翔泳社，2011.

[5] 久保田 光則，アシアル株式会社，HTML5 ハイブリッドアプリ開発 [実践] 入門，技術評論社，2014.