スマートデバイスアプリケーションのためのアーキテクチャに関する研究
―画面情報の動的取扱いに関する考察―

2012SE097 加藤朋也　2012SE108 川瀬裕介
指導教員：野呂昌満

1　はじめに

様々なスマートデバイスが存在しており，画面のサイズもプログラミング言語も異なるものが存在する．例を挙げると，iPhone6s，iPhone6，AQUOS，Xperia などは画面のサイズもそれぞれ異なる．画面サイズ毎のビューの構築が必要となる．レスポンシブ web デザインは，各画面サイズを静的に定義することが可能である．しかし開発者にとって，様々なプログラミング言語 (HTML，SWift，java) を理解し，全ての画面サイズに応じたビューの実装は困難である．

本研究室ではスマートデバイスアプリケーションのための共通アーキテクチャを提案している [2]．共通アーキテクチャを参照し開発を行なうことによって，異なる言語を統一的に扱うことが可能となる．レスポンシブ web デザインの技術では，様々なサイズに応じた画面を定義しなくてはならない．

スマートデバイスの多様な画面サイズや画面解像度により，適切なビュー表示ができない場合がある．例を挙げると，過度なスクロールやピンチイン・ピンチアウトを要するレイアウト，また不適切な文字や画像のサイズに関連するもの等がある．サイズおよびユーザ操作に応じたビューの動的変換が可能となれば，特定のビューを定義することでこれを多様なサイズのビューに対応することができる．

本研究の目的は，ビューの動的構築を可能にすることである．これにより，多様なサイズのビューに対応することができる．

本研究では，ユーザ操作に基づく最適な View との関係および，このビューの変換と共通アーキテクチャとの関係を考察する．アプローチとして，ユーザが画面に対して行なう操作を調査する．操作の意図からビューで行なうべき変換を整理する．それを用いてビューの変換とアーキテクチャ上のコンポーネントとの関係を整理する．そうすることで共通アーキテクチャ上のコンポーネント変換を抽象化することで変換パターンとして定義する．

2　背景技術

2.1　レスポンシブ Web デザイン

レスポンシブ Web デザインとは, 特定のブラウザでつくられる Web ページを，画面サイズに応じて表示させる技術である [1].

Web ページは，HTML(HyperText Markup Language) と CSS(Cascading Style Sheet) などの言語を用いて作成されている．HTML とは，Web 上の文書や構造を記述するための言語のことである．CSS とは，HTML で作成された文章や構造の表示やレイアウト，デザインなどを指定する言語のことである [4].

ブラウザの画面サイズを想定して作られている Web ページを，スマートフォンに対応させるには 2 つのパターンがある．1 つ目は，ブラウザ画面とスマートフォン画面両方に対応したページを作成するパターン．2 つ目はブラウザ用とは別に，スマートフォン専用ページを作成するパターンである．前者は，レスポンシブ Web デザインを利用する．後者は，HTML を実行時環境ごとに複数作成する必要がある．この場合，1 つの変更を行なうときに複数の HTML と CSS を変更しなければならない．

2.2　共通アーキテクチャ

スマートデバイスアプリケーションの開発は，その実行時環境の制約を受ける．本研究室ではスマートデバイス向けのアプリケーションのための共通アーキテクチャを提案している．これは，実行時環境に依存せず独立したものである．

スマートデバイスの多様化に伴い Web アプリケーションとネイティブアプリケーションいずれにおいても，Model，View，Controller の開発のさいに，多くの技術が利用されている．例えば，Web アプリケーションの View 定義技術として，HTML や CSS などが代表例として挙げられる．多様な技術について，すべてを把握することは困難である．さらに，提案されている技術はコンポーネントなどを置き換えても，同様に動作することはできない．技術の選択権は技術者にはない．複数の技術の中からどの技術を選択し用いるかは，利用者や市場の要求さらには開発部門の方針によって決定される．

開発環境が前提とする参照アーキテクチャに対して，共通の参照アーキテクチャを定義する．それぞれの参照アーキテクチャと共通参照アーキテクチャの関係を整理する．これより，実行時環境が前提とするアプリケーションアーキテクチャを定義し，それぞれのアプリケーションアーキテクチャと共通アプリケーションアーキテクチャの関係を整理する．これより実行時環境間の対応関係を明確にすることができる．

図 1 に共通アーキテクチャを示す．共通アーキテクチャは Model，View，Controller に基づいて分割されている．ViewModel はビューのオブジェクトモデルである．View のコンポーネントは 8 つに分類されており，その中で内

部表現を表しているコンポーネントは次の 3 つである．ViewContent は表示内容，DisplayImageContent は役割，Style は見栄えで定義している．また，アスペクト指向技術を利用し，アスペクト間記述を詳細に定義することが可能となった．このことから，実行環境に依存しない共通アーキテクチャの提案がされている．図 1 の IAD は，アスペクト間記述のことである．

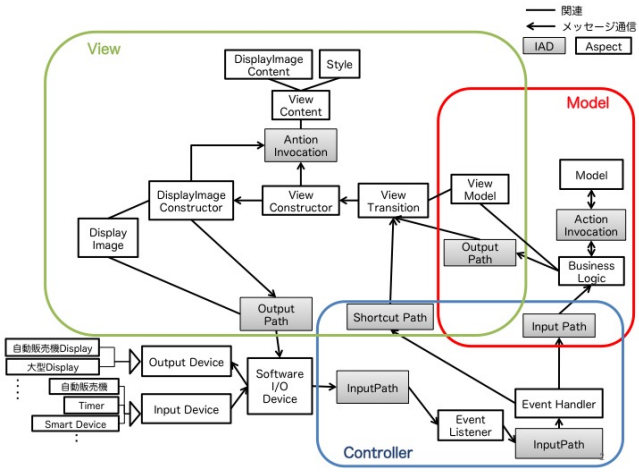

図 1　共通アーキテクチャ

3　View 変更が必要なユーザの操作

View 変更が必要なユーザの操作 (以下 View 操作という) とは，ユーザが使用するデバイスの画面に対して行う動作の回数や，頻度によって動的に振る舞いを変化させることである．

ユーザ操作の例を 4 つ挙げる．

- ピンチイン
  二本の指を画面上に載せて間隔を縮める動作
- ピンチアウト
  二本の指を画面上に載せて間隔を広げる動作
- スワイプ
  画面を前後に移動する動作
- タッチ
  画面を触れる動作

View 操作の例を 4 つ挙げる．

- 拡大縮小 (図 2)
  - ユーザが画面サイズ，文字列に対するピンチイン，ピンチアウト，ダブルタッチの頻度に応じて拡大縮小を行う．
  - テキストリンクの利用頻度に応じて拡大縮小を行う．
  - ページ作成者は利用頻度が多いものを大きく表示したい時に行う．

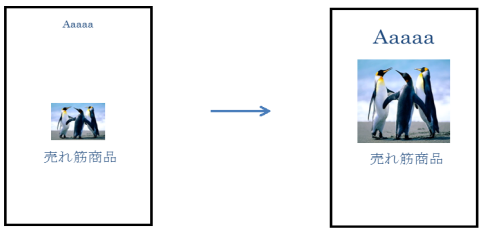

図 2　拡大縮小

- ショートカット (図 3)
  - 画面の閲覧回数, 目的の画面までの遷移回数に応じてショートカットを作成する．

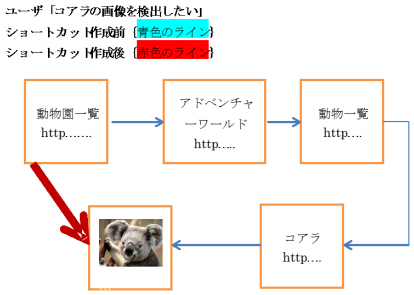

図 3　ショートカット

- 順序入れ替え (図 4)
  - リンクのリストに対して，利用頻度の高いリンクを上位に入れ替える．
  - ページ作成者は，売れ筋商品，人気商品を最初に表示したい．

図 4　順序入れ替え

- 要素変換 (図 5，図 6)
  - 右フリックが多い場合，縦並びもしくは省スペースの要素に変換をする

Title	Release Date	Ranking	Note
BabyBaby	2014/05/01	1	
銀河鉄道の夜	2014/05/02	5	発売中

図 5　要素変換

以上で述べた 4 つの操作と画面の変換対応の表を図 7 で示す．

Title	銀河鉄道の夜
Release Date	2014/05/02
Ranking	5
Note	発売中
Title	BabyBaby
Release Date	2014/05/01
Ranking	1

図 6　要素変換

拡大縮小	・ViewModelのインスタンス -子の要素の変更　○ -構造の変更　× ・ViewModelの型の変更　×
ショートカット	・ViewModelのインスタンス -子の要素の変更　× -構造の変更　○ ・ViewModelの型の変更　×
順序入れ替え	・ViewModelのインスタンス -子の要素の変更　× -構造の変更　○ ・ViewModelの型の変更　○
要素変換	・ViewModelのインスタンス -子の要素の変更　× -構造の変更　○ ・ViewModelの型の変更　○

図 7　操作と画面の変換対応表

4　共通アーキテクチャと View 操作の関係

- 拡大縮小
  ViewModel のインスタンス (内部表現:Style) のサイズ値を変更する事で，動的に px の値を変更することが可能である．内部表現で値を変更するため，レスポンシブ Web デザインとは異なり，ユーザが操作している時に動的に変更が可能になる．
- ショートカット
  ViewConstructor のインスタンス (内部表現:VC) に対するリンク要素の追加する事で，動的にショートカットを生成することが可能である．
- 順序入れ替え
  ViewConstructor のインスタンス (内部表現:VC) の型の構造を入れ替える．入れ替える事によって，頻繁にクリックされる画像やサイトへのリンクが構造的に上に表示する事が可能である．
- 要素変換
  ViewConstructor の型の変更を行う．内部表現の ViewConstructor 部分では，表の表示の仕方を変更し， DisplayImageConstructor では各開発環境での言語を変換し出力する．

5　事例

- 拡大縮小 (図 8)
  ViewModel のインスタンスに対する変更で自動生成可能となる．ピンチインが行なわれた場合，Size のコンポーネントが動的に元の値より大きくするように変更する．ピンチアウトが行なわれた場合，Size のコンポーネントが動的に元の値より小さくするように変更する．
- ショートカット (図 9)
  ViewModel のインスタンスに対する変更で自動生成可能となる．内部表現にリンクを追加し，型を定義する．よって，最短のリンクタグを生成することが可能となる．

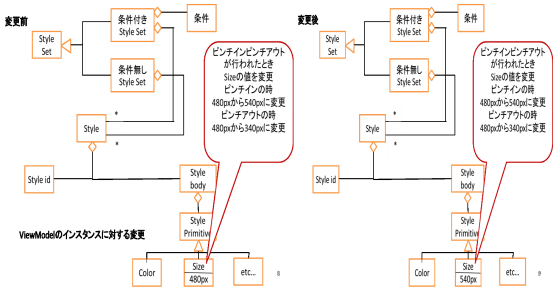

図 8　拡大縮小 変換前後

図 9　ショートカット 変換前後

- 順序入れ替え (図 10)
  ViewModel のインスタンスに対する変更で自動生成可能となる．閲覧頻度の高い文字列の一列目に順序を入れ替える．

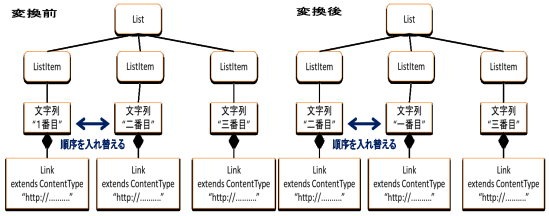

図 10　順序入れ替え 変換前後

- 要素変換 (図 11)
  ViewModel の型に対する変更で自動生成可能となる．任意の型を作成し，要素変換が可能である．

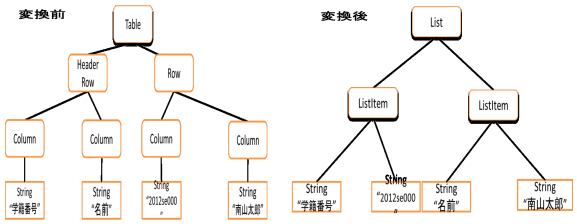

図 11　要素変換 変換前後

成果として，共通アーキテクチャと View 操作との対応関係を図 12 に示した．

拡大縮小は，CSS を用いると，Size のパラメータの部分を変更するため，共通アーキテクチャの ViewModel のインスタンスの要素を変更する．ショートカットは，HTML を用いると，リンクタグを新しく追加, 排除などがあるため共通アーキテクチャの ViewModel のインスタンスの構造を変更する．順序入れ替えは，HTML のリンクタグの構造や，順番を変更するため，共通アーキテクチャの ViewModel のインスタンスの構造と型を変更する．　要素変換は，表示方法を変更するため，共通アーキテクチャの ViewModel のインスタンスの型を変更する．

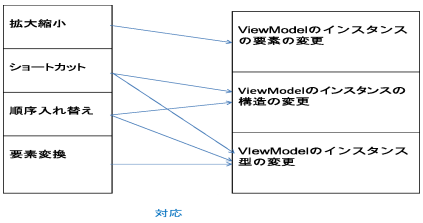

図 12　共通アーキテクチャと View 操作との対応関係

6　考察

多様な View サイズに適応する方法として 2 つある．レスポンシブ Web デザインのように静的に画面を定義する方法と，動的に自動生成する方法である．様々なデバイスに応じた View の定義は開発コストの増大につながる．動的な型の変更パターンを定義できれば，特定の View を様々な画面サイズに対応可能となるので View を定義をする必要がない．パターン化の例を図 13 に示す．二次元表から一次元表に変更を行なう図である．図 13 の上の図は変換前，下の図は変更後の表と木構造である．変更前と変更後の型と要素が変換されている．これを用いる事で型変更のパターン化が可能となる．

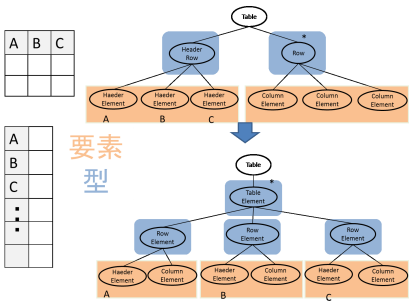

図 13　パターン化

パターンを用いて ViewConstractor で ViewModel を変更する．ViewConstractor でパターンを用いて，5 章でまとめたように ViewModel の型を変換と，ViewModel のインスタンスの変換を行なう．動的な型の変更にユーザの操作履歴を用いる．一定以上の履歴が蓄積されたらその View の変換を行なう．操作履歴とは，デバイス画面に対して行なわれた動作の事である．必要となる要素は，いつ・何に対して・どのようなタッチ操作を行なったのか，などがある．本研究では，履歴として蓄積すべき操作と View の変換の関係を整理した．共通アーキテクチャにおいてユーザの操作履歴を蓄積するために, TouchEventHistory, TouchEventHandler，ViewChangeHistory を追加する．

- TouchEventHistory ユーザの操作を登録，保管
- ViewChangeHistory 内部表現から外部表現に変換して表示
- TouchEventHandler　ユーザの操作を抽出，操作を追加

ユーザの操作履歴を検出し，操作を登録する事で，回数・頻度などで Display 表示を変更することが可能となる．

7　まとめ

スマートデバイスアプリケーションの開発は，その実行時環境の影響を受ける．1 人のソフトウェア技術者が様々な実行時環境を理解することが負担となる．また，それぞれの環境ごとに専門のソフトウェア技術者を確保することも困難である．これらの問題を解決するために，本研究室ではスマートデバイスアプリケーションの共通アーキテクチャが提案されている．また，スマートデバイスの多様化により，各端末に対して，様々な画面サイズや画面解像度が存在する．共通アーキテクチャを用いて，Web アプリケーションとネイティブアプリケーションにおけるレスポンシブ Web デザインの実現方法を，統一的に整理した．ユーザが使用するデバイスの画面に対して行なう動的な振る舞いである View 操作を整理した．

本研究では，例に挙げた拡大縮小，ショートカット，順序入れ替え，要素変換と共通アーキテクチャの対応関係を明らかした．よって動的に HTML のレイアウト部分を構築することが可能となった．ViewModel の型を決めた後に，要素変換を行なう．ViewModel のインスタンス生成された時, 拡大縮小, ショートカット, 順序入れ替えを行なう．これにより，動的に自動生成可能となった．定義された型をパターン化することで，自動生成可能であることが明らかになった．よって，様々なデバイスに対して適切な View 構築の支援を行ない，開発コストの削減につながる．

参考文献

[1] E. Harb, P. Kapellari, S. Luong, and N. Spot, “ *Responsive Web De sign*，” Dec. 2011.

[2] Viet, H.V,“ *Software engineering principles and practice. Wiley*，” Dec.2008.

[3] 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, ” Dec.2015.

[4] 山田晋平，矢野朱里, “ スマートデバイスアプリケーションのためのアーキテクチャに関する研究, ” 南山大学 2014 年度卒業論文 2015.