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
―画面サイズへの適応可能性に関する考察―

2011SE291 山田晋平　2011SE305 矢野朱理
指導教員：野呂昌満

1　はじめに

スマートデバイスの多様化によって，多くの画面サイズ・画面解像度が存在することで，ユーザインタフェースにおいて画面表示に関する問題が生じる．例えば，過度なスクロール・拡大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挙げられる．

Web における上記の問題は，レスポンシブ Web デザイン [1] アプリケーション開発で解決可能である．レスポンシブ Web デザインは，異なる実行時環境に応じて，適切なビューを構築する技術である．

ネイティブアプリケーションにおいては，上記の問題に対応する技術は確立されていない．多様な画面サイズ・画面解像度において，適切なビュー構築を行わなければならない．開発環境ごとに，適切なビュー構築のための仕組みは異なり，アプリケーション開発コストが増加する．

本研究の目的は，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の整理である．ここで整理とは，ネイティブアプリケーションにおいて，レスポンシブ Web デザインの技術要素と等価となるライブラリを定義することである．結果として，画面サイズ・画面解像度に応じたビューの構築を支援する．

レスポンシブ Web アプリケーションの抽象化を行い，レスポンシブネイティブアプリケーションを作成する．レスポンシブ Web デザインと対応するネイティブアプリケーションのライブラリを定義する．Web アプリケーションとネイティブアプリケーションをアーキテクチャレベルで統一的に整理する．その際に，ユーザインタフェースを持つアプリケーションに有効な MVC アーキテクチャを用いたい．本研究室で提案されている共通アーキテクチャは，抽象度の高い MVC アーキテクチャであり，クロスプラットフォーム開発の支援する．共通アーキテクチャにおいて，Web アプリケーション，ネイティブアプリケーションの画面サイズへの適応可能性を考慮することにより，適切なビュー構築の支援を行い，開発コストの削減を目指す．

2　背景技術

2.1　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には，実行時環境ごとに開発環境が存在する．開発者は，様々な開発環境を理解することが必須である．しかし，これらの開発環境を全て理解することは困難であり，コストがかかる．この問題を解決するために，本研究室では，共通アーキテクチャが提案されている．共通アーキテクチャは，複数のプラットフォームが混在する現状での開発支援を目的とし，様々な環境におけるアーキテクチャが説明可能である．共通アーキテクチャと様々な環境のアーキテクチャとの対応関係から，アプリケーション間の変換を可能とし，クロスプラットフォーム開発が可能となる．図 1 は提案されている共通アーキテクチャである．

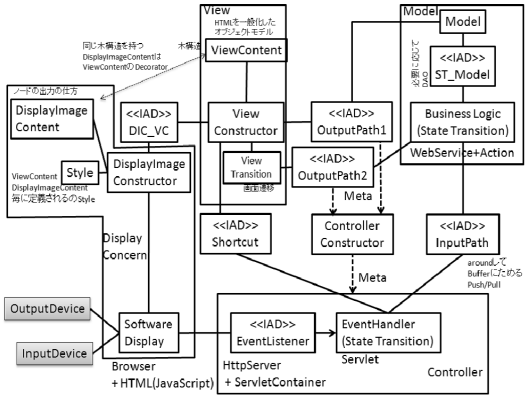

図 1　共通アーキテクチャ

2.2　レスポンシブ Web デザイン

レスポンシブ Web デザインとは，PC のディスプレイを想定して作られている Web ページを，スマートデバイスで適切に表示させるための技術である．

レスポンシブ Web デザインは，CSS(Cascading Style Sheet) の技術である．各種端末に合わせた外部表現を CSS を複数持つことで保持する．レスポンシブ Web デザインは，メディアクエリ（Media Query），フルードグリッド（Fluid Grid），フルードイメージ（Fluid Image）の 3 つの技術で構成されている．

メディアクエリ（Media Query）

表示領域の幅をブレイクポイントと呼ばれる任意の値で決定し，その値になった時点で，適応する CSS を切り替える．画面サイズやデバイスの画面解像度など，様々な条件に合わせた CSS の変更が可能である．図 2 は，スマートフォンとタブレット，ＰＣにそれぞれの CSS を振り分けた場合の図である．

フルードグリッド（Fluid Grid）

レイアウトの構成要素を，どのデバイスにおいても整えて表示することを可能とする．フルードグリッドは，グリッドデザインとリキッドレイアウトという

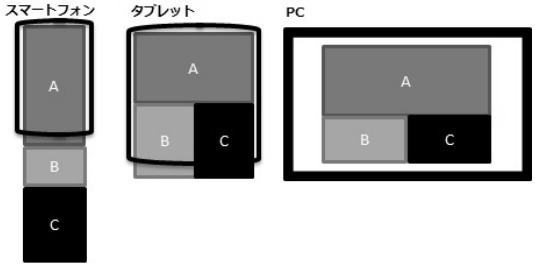

図 2　メディアクエリ

２つの技術を掛け合わせたものである．グリッドデザインでｐｘ固定でレイアウトをデザインし，それをリキッドレイアウトで画面に対する百分率指定に変換する．異なる表示領域における柔軟な対応を可能とする．図 3 は，グリッドデザインとリキッドレイアウトをそれぞれ表した図である．

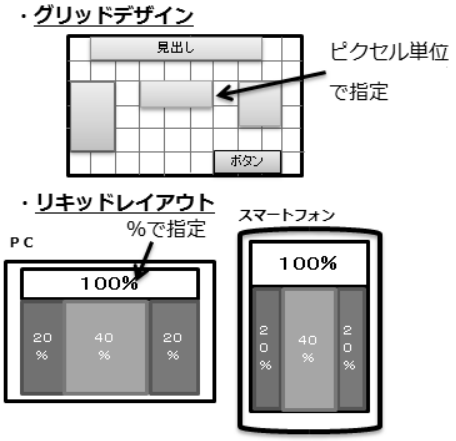

図 3　フルードグリッド

フルードイメージ（Fluid Image）

画像サイズの縦横比を保ったまま，各端末の画面サイズに画像の大きさを対応させる技術である．図 4 は，スクロールを要する画面の表示に，フルードイメージを適応させた際の図である．

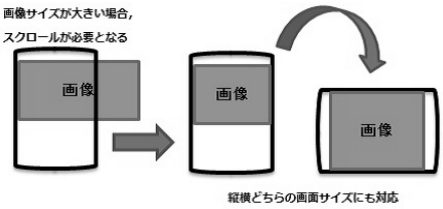

図 4　フルードイメージ

3　スマートデバイスアプリケーションのビューに関する問題点と解決策

本研究では，スマートデバイスアプリケーションにおけるビューに関する問題点を取り上げる．一般的に使用されているスマートデバイスには様々な画面サイズが存在しており，スマートフォンとタブレット端末の画面サイズは約 2 倍以上にもなる．例を挙げると，iPhone5S の画面サイズは 4 インチであるが，iPad Air は 9.7 インチである．スマートフォンだけに注目したとしても，多くの画面サイズが存在している．また，各端末によって画面解像度が異なる．画面解像度とは，ディスプレイにおける画面の総画素数を表している．画面サイズが少ししか変わらない場合であっても，画面解像度が異なると，表示領域が異なり，文字サイズやアイコン，ボタン，画像の大きさが変化する．

画面サイズと画面解像度の違いによって，各端末においてビューが異なるという問題点が生じる．ある端末では適切に表示されているビューだとしても，他の端末では，文字やアイコン，ボタン，画像のサイズが適切でなく，過度なスクロール量または拡大を要するレイアウトである可能性がある．この問題を解決するための開発が必要となる．

4　共通アーキテクチャを使用したレスポンシブ Web デザインの実現方法

Web とネイティブをアーキテクチャレベルで統一的に整理するために，本研究室で提案されている共通アーキテクチャを用いる．

4.1　Web アプリケーションと共通アーキテクチャの対応関係

Web アプリケーションの一般的なフレームワークとして，Ruby on Rails を対象とする．理由としては，他の Web アプリケーションフレームワークである，ASP.NET[2] や Java と比較しても，整理されており，より代表的であると考えるからである．以下，Ruby on Rails で実現した Web アプリケーションと，共通アーキテクチャのコンポーネントとの対応関係について記述する．

- DisplayImageConstructor：HTML
  - ファイル名：ビュー名.html.erb
- Style：CSS（レスポンシブ Web デザイン適応）
  - ファイル名：ビュー名.css.scss

図 5 は，Web アプリケーションと共通アーキテクチャとの対応関係を表している．

4.2　レスポンシブネイティブアプリケーションの実現方法の定義と，ネイティブアプリケーションと共通アーキテクチャの対応関係

ネイティブアプリケーションの一般的なフレームワークとして，Android と iOS を対象とする．理由としては，スマートフォンの OS 売り上げシェアは Android と iOS で 9 割を占めているからである．以下、レスポンシブ Web アプリケーションにおける，抽象化した技術要素を記述する．これに基づき，等価となるライブラリを定義する．

- メディアクエリ
  画面サイズを認識し，レイアウトを選択

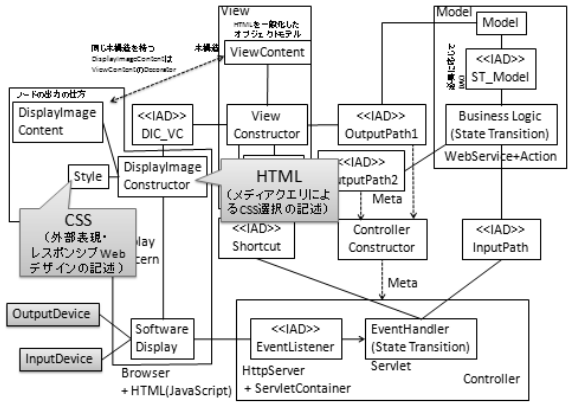

図 5 Ruby on Rails の共通アーキテクチャとの対応関係

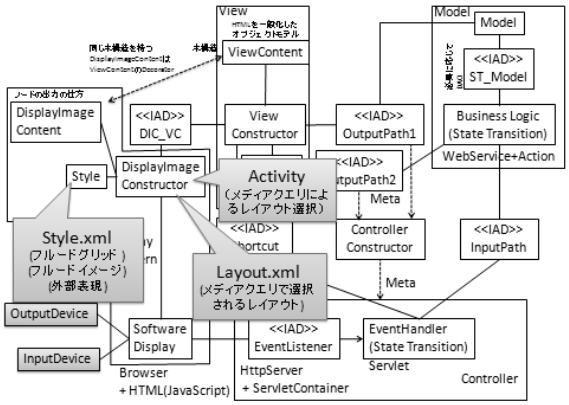

図 6 Android の共通アーキテクチャとの対応関係

- フルードグリッド
  画面サイズによって構成要素の縦横幅の伸縮
- フルードイメージ
  画像の大きさを画面サイズに適した大きさ（縦横比を維持し，画面から見切れない）に伸縮

4.2.1 Android

Android において，レスポンシブ Web デザインと対応するライブラリを定義する．

- メディアクエリ
  - Display クラス：画面サイズを取得
- フルードグリッド
  - LinearLayout クラス：ビューを一列に配置
  - LayoutParams クラス：ビューの幅を調節
- フルードイメージ
  - LayoutParams クラス：ビューの幅を調節
  - ImageView クラス：画像出力

また，これらのライブラリが実装される Android のコンポーネントは，以下のようになる．

- Activity
  メディアクエリのレイアウト選択部分．
- Layout.xml
  メディアクエリで選択されるレイアウト．
- Style.xml
  フルードグリッド，フルードイメージ，外部表現．

図 6 は，Android と共通アーキテクチャの対応関係を表している．

4.2.2 iOS

iOS において，レスポンシブ Web デザインと対応するライブラリを定義する．

- メディアクエリ
  - UIScreen クラス：画面サイズを取得
- フルードグリッド
  - CGRect クラス：対象オブジェクトの位置とサイズを管理
- フルードイメージ
  - UIView クラスの，UIViewContentModeScaleAspectFit：縦横の比率を維持しつつ，画像全体を表示

また，これらのライブラリが実装される iOS のコンポーネントは，以下のようになる．

- UIApplicationDelegate
  メディアクエリのレイアウト選択部分．また，外部表現．
- UIViewController，UIView
  メディアクエリで選択されるレイアウト．また，フルードグリッド，フルードイメージ．

図 7 は，iOS と共通アーキテクチャの対応関係を表している．

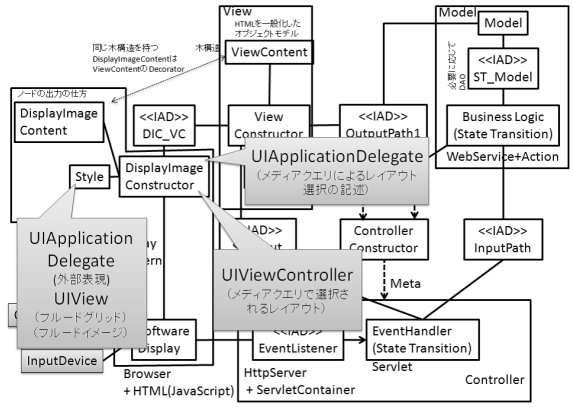

図 7 iOS の共通アーキテクチャとの対応関係

5 共通アーキテクチャの詳細化

前述の対応関係に基づき，共通アーキテクチャの詳細化を行う．詳細化については，共通アーキテクチャで定義されている Style，DisplayImageConstructor について行う．

5.1 Style

Style とは，見た目を定義している．図 8 は，Style で定義されているデータ構造と，それにおけるレスポンシブ Web デザインの適応部分を表している．

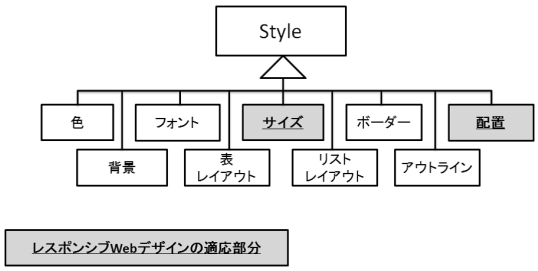

図 8 Style の詳細化

- サイズ

サイズの表現方法には，百分率・ピクセルがある．レスポンシブ Web デザインのフルードグリッド・フルードイメージは，百分率に適応されると定義した．

- 配置

配置とは，表によって画面幅また，それにおける画面要素がどの場所に配置されるかの情報を保持する．これはメディアクエリにおける画面幅とそれに対するレイアウトの変更に適応すると定義した．

5.2 DisplayImageConstructor

DisplayImageConstructor は，DisplayImageContent・ViewContent・Style で定義されたインスタンスに基づき，レイアウトを生成する．本研究では，レスポンシブ Web アプリケーションのサイズ・配置における生成時の振る舞いを定義した．

図 9 は，Web アプリケーションにおける DisplayImageConstructor の振る舞いについて表している．

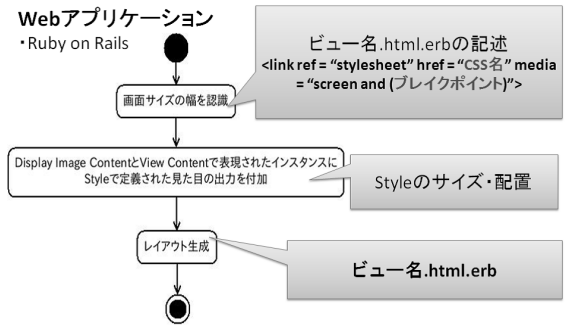

図 9 Web アプリケーションにおける DisplayImageConstructor の振る舞い

図 10 は，ネイティブアプリケーションにおける DisplayImageConstructor の振る舞いについて表している．

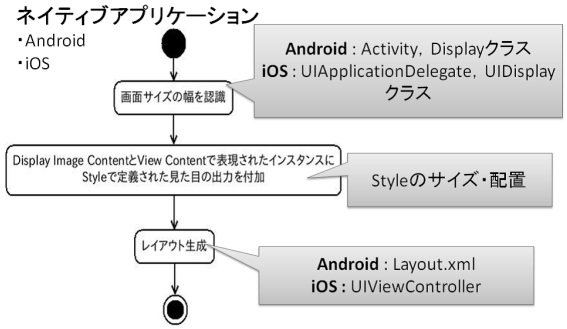

図 10 ネイティブアプリケーションにおける DisplayImageConstructor の振る舞い

6 考察

レスポンシブネイティブアプリケーションの実現方法を定義した．画面サイズ・画面解像度に応じたビュー構築を支援する．

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．よって，共通アーキテクチャにおいて画面サイズへの適応可能性が保証されたと考える．共通アーキテクチャにおいて，Style・Display Image Constructor の詳細化を行った．

共通アーキテクチャにおいて，レスポンシブ Web デザインの実現コンポーネントを明確化した．これより，開発環境ごとの適切なビュー構築の実現方法が対応付けられ，アプリケーション開発コストが削減される．また，実現コンポーネントを分けた事により，保守性・再利用性が向上したと考えられる．

7 おわりに

本研究では，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の整理を行った．Web とネイティブ共に，共通アーキテクチャにおいて，レスポンシブ Web デザインとの対応コンポーネントと実現方法を定義し，統一的なビュー設計の基礎ができた．今後，共通アーキテクチャとの対応関係に基づき，自動化を完成させることが課題である．また，メディアクエリのビュー選択部分は，アスペクトとして抽出し，共通アーキテクチャの保守性を高めることが課題である．

参考文献

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

[2] 山田祥寛，ASP.NET MVC 実践プログラミング .NET Framework による標準 Web 開発技法，株式会社秀和システム，2009 年 10 月 25 日.