インタラクティブソフトウェアのビューのオブジェクトモデルの設計

2011SE156 松本杏 2011SE197 西川卓磨 2011SE238 清水大志
指導教員：野呂昌満

1　はじめに

近年，インタラクティブソフトウェアのビューの定義において，HTML5 と CSS3 が主流となっている．一方，実行時環境で異なる言語を用いた開発も行われている．一人の開発者やページデザイナが，すべての言語を把握することは困難である．画面ごとに適したビューを表現するための技術として，レスポンシブ Web デザインがある．レスポンシブ Web デザインでは，画面のサイズに合わせて静的に画面を定義している．しかし，すべての画面について定義することは困難である．結果として，考慮されていないサイズに対してユーザはスワイプやピンチアウト，ピンチインなどの余計な操作が必要になる．

我々はすべてのビュー定義形式を説明可能な抽象度の高いモデルを定義する必要があると考えた．説明可能とは，それぞれを生成可能にすることである．これにより，すべてのビュー定義形式を統一的に扱うことが可能である．さらにユーザ操作などのコンテキストから動的変換可能な枠組みを提案する必要があると考えた．この解決策として，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．図 1 に共通アーキテクチャを示す．共通アーキテクチャでは，ビューのオブジェクトモデル (以下，ViewModel) を定義している．ViewModel は，ViewContent，DisplayImageContent，Style の三つの要素で構成されている．ViewContent は画面上の表示内容，DisplayImageContent は画面上の役割と見え方，Style は画面上の見栄えを表している．

本研究の目的は，インタラクティブソフトウェアのビューのオブジェクトモデルの設計である．ビューのオブジェクトモデルの設計とは，共通アーキテクチャ上で定義されている ViewModel の ViewContent，DisplayImage-Content，Style の構造を明らかにすることである．さらにビューの動的変換に対応するように ViewModel を拡張することである．我々は HTML5 と CSS3 を参考にそれらを抽象化し，既存の技術すべてで説明可能な抽象度の高い ViewModel の設計を行った．共通アーキテクチャではビューの動的変換を実現する枠組みとして画面変換におけるパターンの適用を提案している．本研究では，すでに ViewModel の型は生成されていることを前提とし，特定の ViewModel の型をパターンによって生成された型に変換可能とするために ViewModel を拡張した．

2　ViewModel の設計方針

ViewModel の要素である Viewcontent，DisplayImage-Content，Style はそれぞれ概念的に木構造になっている．ViewModel と画面との関係を図 2 に示す．

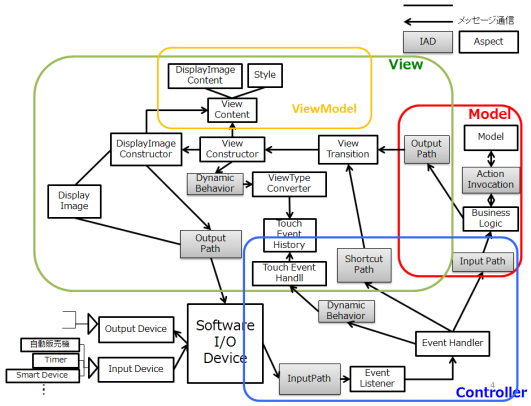

図 1　共通アーキテクチャ

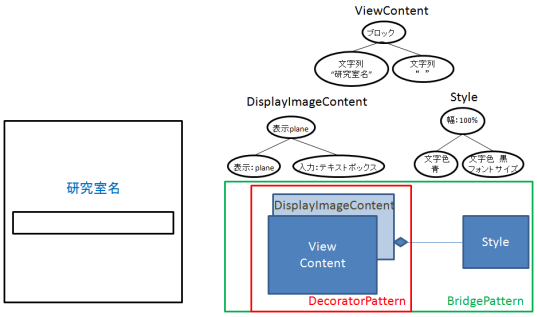

図 2　画面と ViewContent,DisplayImageCintent,Style との関係

ViewContent，DisplayImageContent，Style の対応関係を HTML5，CSS3 を参考に調査した．図 3 に HTML5 と CSS3 の例を示す．この図は，研究室名簿システムの検索と登録をするためのページの記述である．CSS3 では表示画面の色等の見栄えを記述している．図 3 のコンポーネ

図 3　HTML5 と CSS3 の記述

ントの説明を以下に示す．

- ViewContent：画面の表示内容を表す．図 3 にあるように，’ トップページ’ や’ 研究室名簿システム’ の文字

列や章，画像，動画等を表す．

- DisplayImageContent：画面の役割を表す．図 3 にあるように，’button’ のボタンやリスト，表等を表す．
- Style:画面の見栄えを表す．図 3 にあるように，’color’ の色やフォント，サイズ等を表す．

3 ViewModel の設計

3.1 ViewContent の設計

HTML5 を参考に HTML 要素と ViewContent の対応関係を整理した．その結果，本質的に木構造で表現可能であることがわかった．図 4 に HTML 要素と ViewContent の対応関係を示す．対応関係から <header> や <section> のタグは文書のまとまりを表すので木構造の節にあたる．同様にテキストや <script> は木構造の葉になる．図 5 に ViewContent の設計を示す．ViewContent は画面上の表示内容を表わしている．CompositeViewContent は章，主題，区分等を持っているコンポーネントである．また，ここに定義されていない ViewContent の型を表現するために，CompositeViewContent の型にブロックを定義した．ブロックはすべての ViewContent の型のインスタンスを子要素にもつことができる．PrimitiveViewContent は文字列や数値，画像，動画等を持っているコンポーネントである．それらの関係は，HTML 記述では章，主題，区分等の中に章や主題，文字列や数値，画像や動画等が記述される．よって再帰的な構造の取扱いを容易にするためにコンポジットパターンを適用した．また PrimitiveViewContent にそれがインラインなのかリンクなのかの表示内容を付加するためにブリッジパターンを適用した．

HTML 要素	View Content
テキスト	文字列
<figure>	図表
<script>	プログラミング
<header>	補足
<main>	メインコンテンツ
<section>	セクション
<footer>	フッタ
<article>	記事
	区分

図 4 HTML 要素と ViewContent の対応関係

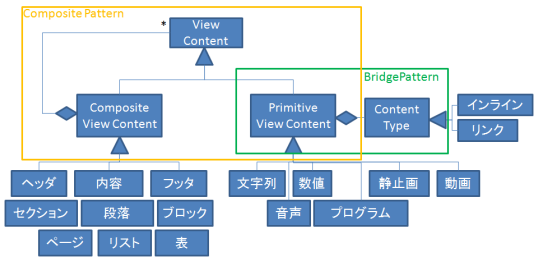

図 5 ViewContent の設計

3.2 DisplayImageContent の設計

図 6 に HTML 要素と DisplayImageContent の対応関係を示す．
DisplayImageContent は役割と見え方を表している．役割と見え方とは，画面部分としての振舞や機能であり，内容の表現方法である．更に HTML5 を参考に HTML 要素と DisplayImageContent の対応関係を整理して洗練した．DisplayImageContent の設計を図 7 に示す．

HTML 要素	Display Image
<table>	表
<tr>	行
<td>	セル
	序列ありリスト
	序列なしリスト
	リスト
テキスト	文字列
<textfield>	テキストフィールド
<form>	フォーム
<input type = button	ボタン
<input type = select	セレクトボックス
<input type = submit	送信ボタン
<input type = checkbox	チェックボックス

図 6 HTML 要素と DisplayImageContent の対応関係

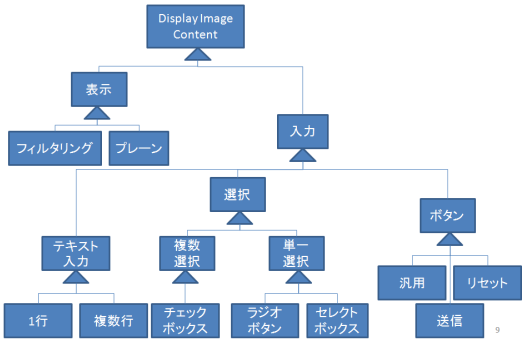

図 7 DisplayImageContent の設計

3.3 Style の設計

図 8 に HTML 要素と Style の対応関係を示す．Style は見栄えを表している．見栄えとは装飾に関する情報である．CSS3 の記述では，要素に装飾する場合，ひとつの id と複数の見栄えを持つことができる．これにより Style は Style id と Style Body からできており，Style Body は多数の Style Primitive を持っている．また，Style Set は任意の条件が付いている Style の集合である．条件付き Style Set と条件のない条件なし Style Set から構成されている．Style の設計を図 9 に示す．

3.4 ViewContent と DisplayImageContent と Style の合成

ViewContent と DisplayImageContent と Style は概念的に同じ木構造をしているので実現するために相互に対応付ける必要がある．各々が異なる型階層なので具

HTML 要素	Style
background	背景
font	フォント
line-Height	行の高さ
margin	マージン
padding	パディング
table-layout	表レイアウト

図 8　HTML 要素と Style の対応関係

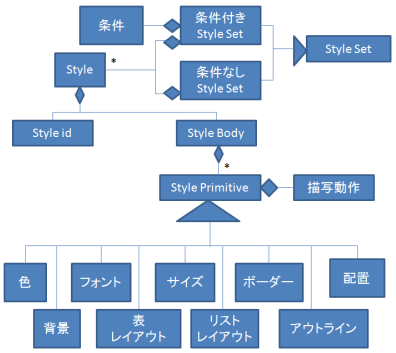

図 9　Style の設計

象クラスのインスタンスの組み合わせを表現する必要がある．DisplayImageContent は主となる木構造である ViewContent に装飾するためにデコレータパターンを適用した．Style は多重 has-a 関係にあるのでブリッジパターンを適用し付加した．またプロトタイプパターンは，ViewModel の複製をするために適用した．これを図 10 に示す．

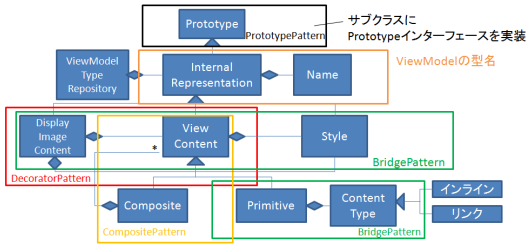

図 10　ViewContent と DisplayImageContent と Style の合成

4　ViewModel の拡張

4.1　動的取扱いの概要

本研究では，ViewModel の型を変換することにより様々な画面サイズに合わせたビューを定義する．この概要を図 11 に示す．変換パターンを定義することであらゆる ViewModel について同一の変換を可能にする．ViewModel はユーザの操作履歴に応じて新たに生成される ViewModel の型について名前をつけて登録する．それ以降は，登録された型の名前から部品を取得し，ViewModel を構築する．これを図 12 に示す．

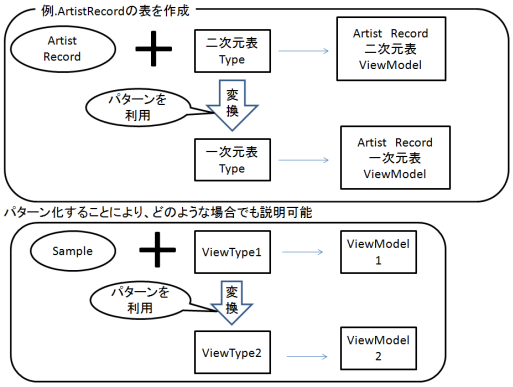

図 11　動的取扱いの概要

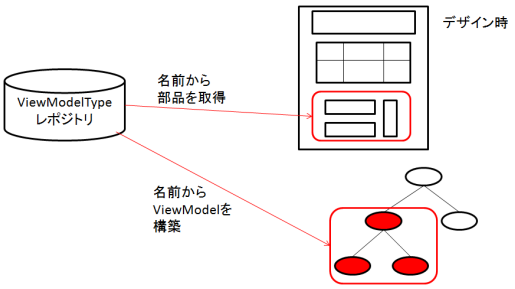

図 12　ViewModel の型の利用

5　実例による考察

本研究では，銀杏というアーティストの CD のタイトルと発売日の情報を持った表と説明文を表示する画面を実例とする．アーティストの CD の情報を表で表す場合の ViewModel を一次元の表と二次元の表でそれぞれ選んだときに画面が表現できることを確認した．図 13 にその実例を示す．

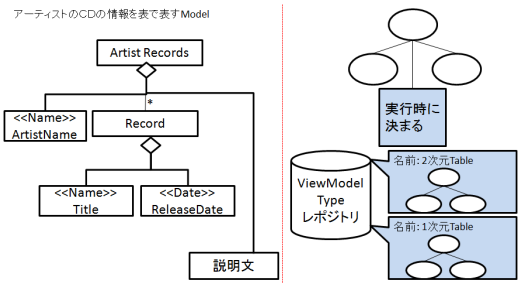

図 13　実例による ViewModel

5.1　二次元表を用いた ViewModel

図 14 に二次元表を用いた画面を示す．図 15 に二次元表を用いた ViewModel を示す．図 15 の線の枠内の部分が二次元の表を表す部分である．この表の ViewModel を二次元表と名前を付けて ViewModelType レポジトリに登録する．二次元表を利用することで図 14 の画面が表示できることを確認した．

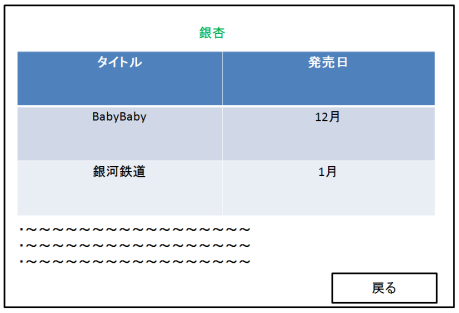

図 14　二次元表を用いた画面

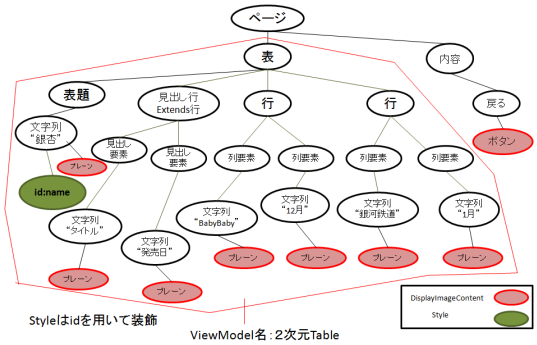

図 15　二次元表を用いた ViewModel

5.2　一次元表を用いた ViewModel

二次元表を用いた画面でスワイプが多く行われた場合，横長の二次元表から縦長の一次元表へ変換する．図 16 に一次元表を用いた画面を示す．図 17 に一次元表を用いた ViewModel を示す．図 17 の線の枠内の部分が表を表しており，この表の ViewModel を一次元表と名前を付けて ViewModelType レポジトリに登録する．一次元表を利用することで図 16 の画面が表示できることを確認した．

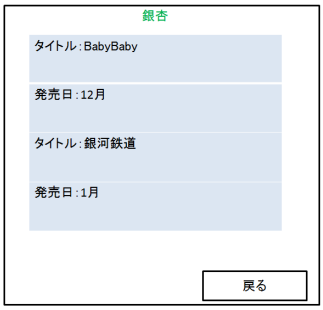

図 16　一次元表を用いた画面

5.3　プロトタイプパターンを使った一次元表の生成工程

ViewModel にプロトタイプパターンを適用することで拡張した．これにより，事前に登録した型の複製を取得し，これを利用して型変換を可能とした．木構造の ViewModel の根に対し，clone メッセージを送ると，自身の複製と子の要素に対する clone メッセージを送ることで，木構造の複製を生成する．この概要を図 18 に示す．

事例では，登録した型を利用することで二次元表の型を一次元表の型に変換が可能であることを確認した．事前に型が登録されていれば他の ViewModel についても同一の変換が可能であると考えている．

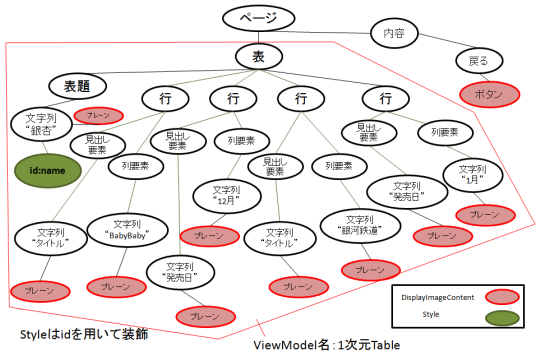

図 17　一次元表を用いた ViewModel

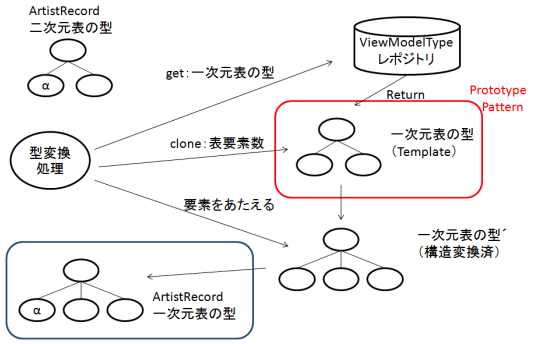

図 18　プロトタイプパターンを使った一次元表の生成工程

6　まとめ

本研究では，インタラクティブソフトウェアのビューのオブジェクトモデルの設計を目的として，HTML エディタで画面を定義し HTML5，CSS3 を参考に作成して ViewModel を設計した．ネイティブアプリケーションでも矛盾がないように ViewModel を設計しビュー定義形式を定義した．今後の課題として，異なる開発言語を用いてビュー定義形式の要素と ViewModel の対応関係を整理する必要がある．

参考文献

[1] E. Gamma, J. Vlissides, R. Helm, and R. John- son, *Design Patterns: Elements of Reusable Object- Oriented Software*, Addison-Wesley, 1995.

[2] 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，”情報処理学会研究報告．ソフトウェア工学報告，vol.2015-SE-187, no. 32，pp. 1-8，2015-03-05.