インタラクティブソフトウェアの動的変化に関する研究
—Controller を題材に—

2011SE174 森一晃　2011SE246 杉山京二廊
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り，様々な画面サイズを持ったデバイスが存在するようになった．特定の View を表示させようとした場合，画面サイズ毎に表示可能な領域が変わる．多様な画面サイズに対応するものを定義しようとした時，開発者はレスポンシブウェブデザインにみられるように静的に画面サイズ毎の表示形式を定義をしなければならない．ユーザ側からしても, 表示されていない情報に対して何度もスワイプを行なうなど余計な操作をする必要があり，利便性に欠ける．我々は，ユーザの操作履歴等を保持することによって，ユーザが求めるビューの自動構築を試みる．開発者がビューの変換パターンを定義するだけで，多様なデバイスの画面サイズ毎にビューを定義することなく，エンドユーザ毎に最適な画面表示を行うことに可能となることを目指す．

本研究室では，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を提案している．[2] 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，Web アプリケーションとネイティブアプリケーションの統一的な開発を可能にすることを目的として構築したものである．

本研究では，CD 販売一覧システムを題材とし，共通アーキテクチャに基づいて，ユーザの操作履歴に着目することで，動的なビュー変化に対応するためのコントローラの構造について考察する．アスペクト指向技術を適用し，既存のコントローラに対してアスペクトを織り込んだり，外したりすることで，操作イベントの検出と登録を行なう．これにより，ユーザの操作履歴を蓄積し，ユーザに応じたビューの再構築を行なう．また，状態遷移機械として実現することで，コントローラの仕様の可視化を行なう．これを通してコントローラ部の動的変化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．

図 1 に共通アーキテクチャを示し，以下に共通アーキテクチャの Controller 部について示す．

Controller は EventListner と EventHandler によって構成される．図 1 の右下に囲まれたコンポーネントは Controller に関連する．

以下が各コンポーネントの説明である．

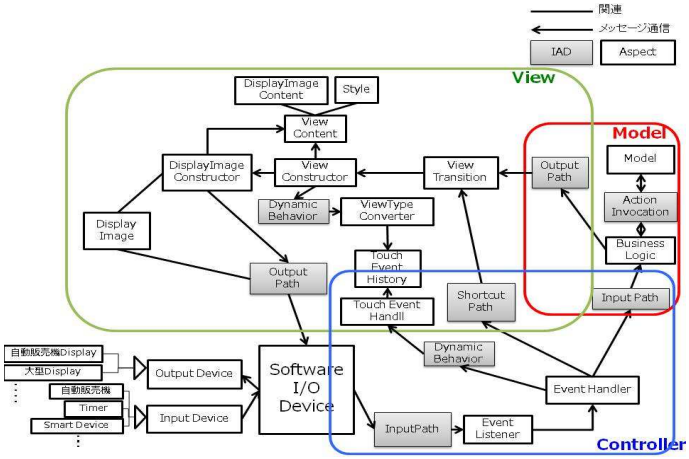

図 1　共通アーキテクチャ

- EventListner・・・SoftwareI/ODevice から受け取った外部イベントを内部イベントに変換し，EventHandler に通知する．
- EventHandler・・・EventListner から受け取った内部イベントを自身の状態によって Business-Logic(Model) か ViewTransition(View) に渡す．

外部イベントとは，SoftwareI/ODevice より通知されるイベントのことであり，内部イベントとはアプリケーション内部で通知されるイベントのことである，イベントを内部イベントと外部イベントに分けることにより，外部表現から独立してアプリケーションを記述可能としている．

3　画面操作に基づくビューの動的変化

我々は静的に多数のビューを定義しなければならない問題を，ビューの動的変更の問題とすることで，多数定義の省力化に資すると考えた．この節では，画面の動的変更に対応するためのコントローラの構造について考察する．

3.1　ビューの動的変化

画面の動的変化に対応していない画面を示したのが図 2 である．ユーザは表示されていない部分に対してスワイプを行なうことで続きの情報を手に入れることが出来るが，その分見えなくなる情報も存在する．

本研究では，画面操作に基づくビューの動的変更を実現する．画面の操作履歴による画面表示の変化を表したのが図 3 である．図 3 では，表示可能領域を越えたビューにスワイプ操作が頻繁に起こる場合，表示領域におさまるビューに変換する様子を示している．

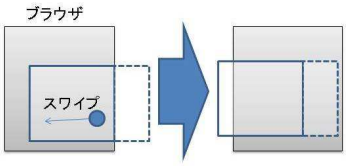

図 2　サイズに関わらない固定画面を表示したの例

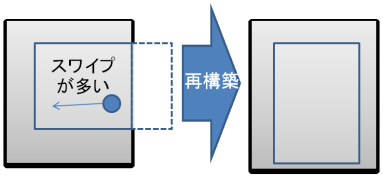

図 3　サイズに対応した画面を表示した例

3.2　動的変化に対応するためのコントローラの概要

画面操作に基づくビューの動的変化に対応するためのコントローラに関連した動的変化実現の概要を図 4 に示す．図 4 における A が既存のコントローラである．ビューの動的変化に対応するには，既存のコントローラに対してビューの動的変化に関連する処理を付加する．コントローラ A から通知された操作イベントは履歴登録され，蓄積される．自動構築処理は, 一定以上履歴にイベントが蓄積されると，履歴に応じて表示画面を再構築する．

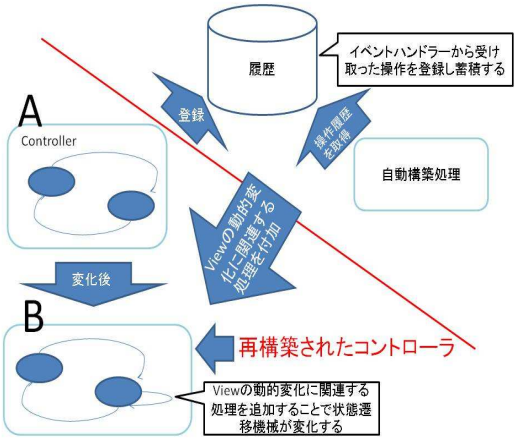

図 4　動的変化に対応するためのコントローラの概要

ビューの動的変化に対応するためにコントローラは画面操作を検出する必要がある．動的変化に対応していないコントローラの仕様を図 5 に示す．待機状態からトップページ表示というイベントを受け取ると，トップページ構築要求というアクションが起き，トップページイベント待ち状態となる．

動的変化に対応したコントローラの仕様を図 6 に示す．ビューの動的変化に対応する際に，コントローラでは図 5 から図 5 のように，動的に構造が変化する．6 にあるトップページ表示のイベントに対してトップページ構築要求と

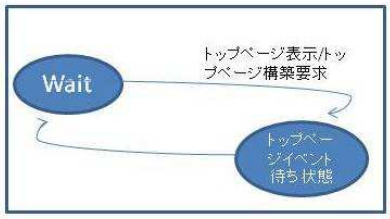

図 5　動的変化に対応していない状態遷移機械

いうアクションに加え，スワイプのイベントに対してイベントの回数を履歴として登録するアクションが存在する．

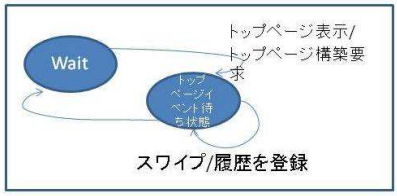

図 6　動的変化に対応した状態遷移機械

4　Controller 部の設計

4.1　設計指針

前述の通り，ビューの動的変化において，Controller が動的に対応していないものに対して，画面操作を検出可能にする必要がある．一定以上の履歴が蓄積されたら，Controller の仕様を変更する必要がある．この仕様の変更とは，例えば特定のリンクがビューに追加されるなどの変化があった場合に，Controller はこのリンク押下のイベントを受理可能となるように変化することである．

我々は，ビューの動的変化に対応するためにはプログラムの構造の動的変化を実現する必要があると考える．我々はプログラムの構造の動的変化も視野に入れて，Controller に対してアスペクト指向技術を適用する．横断的関心事とすることがモジュール化の観点から自然と考え，この横断的関心事によって規定される Aspect として MetaMachineAspect を特定した．アスペクト指向技術を応用することにより，既存の Controller を変更することなく，動的変換に対応可能となると考える．

Controller は状態遷移機械として実現することで仕様の可視化ならびに，仕様からプログラムへの自動生成が可能になると考える．この状態遷移機械に対する横断的関心事として，状態遷移コンサーン，アクションコンサーンを識別し，StateTransitionAspect，ActionAspect を特定した．

Aspect の実現には GoF デザインパターンを用いる．MetaMachineAspect の実現には，ActionAspect は手続きをモジュール化するために Command パターン，StateAspect は状態をモジュール化するために State パターンを適用した．

4.2 EventListner の設計

EventListner の機能は SoftwareI/ODevice から送られた外部イベントを内部イベントに変換して EventHandler に通知するものなので，外部イベント・内部イベント・イベント対応表を保持する必要がある．図 7 に示したように EventListner は，外部イベントと内部イベントの対応関係を保持している．

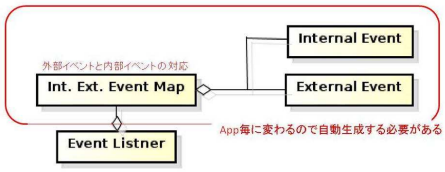

図 7 EventListner の設計

以下に図中の各クラスについて説明する．

- EventListner・・・SoftwareI/ODevice から受け取った外部イベントを Int.Ext.EventMap に渡し，Int.Ext.EventMap から受け取った内部イベントを EventHandler に通知．
- Int.Ext.EventMap・・・InternalEvent と ExternalEvent を持つ．EventListner から受け取った外部イベントを，対応表とイベントリストをもとに，内部イベントに変換し，EventListner に渡す．
- InternalEvent・・・内部イベント．
- ExternalEvent・・・外部イベント．

4.3 EventHandler の設計

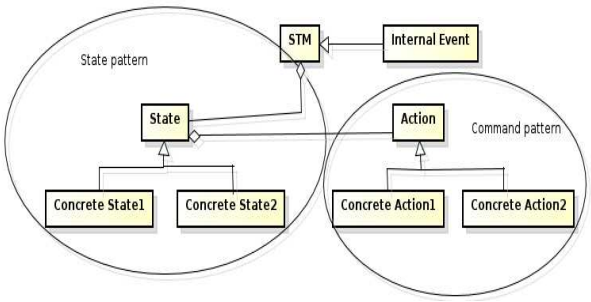

図 8 EventHandler の設計

EventHandler は，状態に応じて View または Model に内部イベントを通知する．したがって，状態遷移機械によるモデル化が適切である．

State パターンと Command パターンを用いて設計した EventHandler を図 8 に示し，以下に図中の各クラスについて説明する．

- 状態遷移機械・・・状態遷移機械であり，サブクラスとして InternalEvent がある．また，状態遷移機械は State を持ち，EventListner から通知された内部イベントを InternalEvent から受け取り，State と Action から受け取った情報から，BusinessLogic か ViewTransition に内部イベントを通知する．
- InternalEvent・・・内部イベント．
- State・・・Action クラスと状態を持っており，状態遷移機械に状態を通知する．サブクラスとして Concrete1・Concrete2 があり，サブクラスの数は状態の数だけ存在する．
- Action・・・アクションを持っており，State に状態を通知する．サブクラスとして ConcreteAction1・ConcreteAction2 があり，サブクラスの数はアクションの数だけ存在する．

5 事例検証

CD 販売一覧システムを題材として，View 再構築における Controller 部の事例検証を行う．

CD 販売一覧システムとは，あるアーティストが販売している CD が一覧で表示されるシステムのことである．画面の状態を図 9 に示す．図の左側が通常の画面であり，右側がスワイプイベントの履歴に応じて再構築した後の画面を示している．通常の画面ではアーティスト名が画面上側に表示され，その下に横長の表としてアーティストが販売している CD の情報が並べられている．項目としては，タイトル，発売日，ランキングがあり，画面におさまらないデバイスの場合，横にスワイプを行うことで続きの情報を見ることが出来る．スワイプの回数が多いと判断した場合，図 9 の右側のように，表を縦に再構築することで，表を見やすくすることを目的とし，共通アーキテクチャにおける Controller 部の設計を行う．

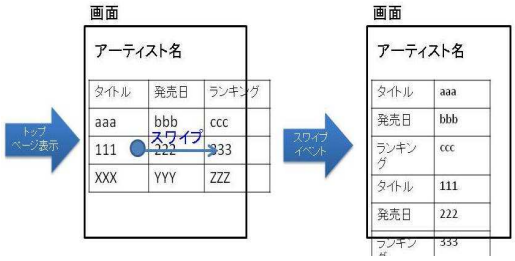

図 9 事例検証における View の再構築

5.1 EventListner

図 10 にスワイプ操作における外部イベントと内部イベントを示す．トップページ表示という外部イベントに対応する内部イベントは Start であり，スワイプ操作という外部イベントに対応する内部イベントは Swipe である．

5.2 EventHandler

図 11 にスワイプ操作における EventHandler の状態遷移機械とシーケンス図を示す．状態遷移としては，待機状態からトップページ表示の操作イベントを受け取ると，

図 10　スワイプ操作におけるイベントの対応表

操作履歴の登録を行い, 待機状態に戻る．本研究における事例検証に関しては，画面再構築後にイベントの追加が無いので，状態遷移機械も履歴の登録のみが追加される．シーケンス図では，EventHandler から EventHistory にスワイプ操作の履歴を登録する．confirmChange と DisplayRecordListPage ではスワイプ操作の回数を確認し，View の更なる変更が必要な場合は再表示を行う．

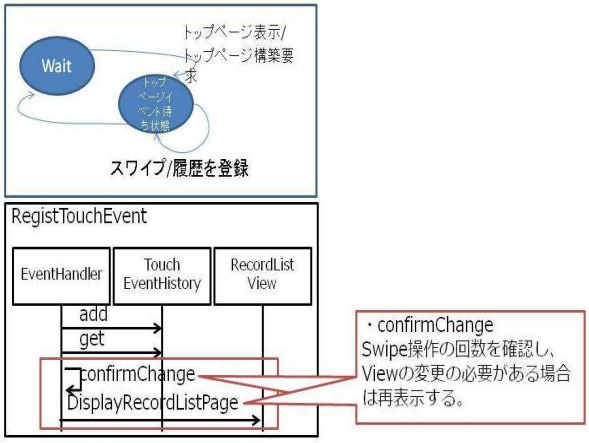

図 11　EventHandler の状態遷移機械とシーケンス図

6　考察

事例検証では，スワイプ履歴に応じてビューを再構築した際，コントローラの構造を動的変化させる必要がある．ビューを再構築した後のコントローラの状態遷移機械を表したのが図 5 である．ユーザの操作履歴から画面サイズを再構築することで画面サイズ内におさまりきるようになり，スワイプを検出する必要が無くなったことで，Aspect を取り除いた．これによりタッチ操作を検出して履歴を保存することが可能となり，これによりユーザ操作に応じた最適なビューの構築が可能となる．

本研究においてはスワイプ操作が必要な場合の表示画面の再構築のみを提案している．View の再構築以外で考えられる動的変化としてショートカットの作成や画面分割結合が挙げられる．ショートカット作成の例としては，メニュー画面から選択を行なうと，ある一覧が表示され，その一覧から選択を行うと，その詳細が表示されるアプリケーションがあるとする．この場合ユーザの操作履歴に応じて動的にメニュー画面に詳細表示のショートカットを作成することで，ユーザは一覧画面に遷移することなく詳細画面に移動することが出来る．その場合の状態遷移機械を図 12 に示す．図左側が動的変化に対応していない状態遷移機械であり，右側が動的変化に対応したものである．このようなアプリケーションの動的変化を考えた場合，図 12 のようなイベントの追加が想定されるので，状態遷移機械としてもイベントやアクションの追加が想定される．

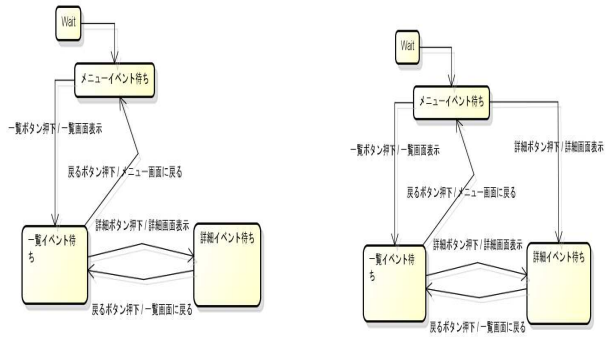

図 12　ショートカット作成における動的変化の状態遷移機械の例

7　まとめ

本研究では，CD 販売一覧システムを例として，共通アプリケーションアーキテクチャに基づき，Controller の構造について設計をし，イベントの対応表，状態遷移機械から，ユーザの操作履歴に応じた動的な View 変化に対応する Controller の構造を定義し，これらから Controller 部の動的変化への対応が可能なことを確認した．今後の課題として，自動生成系の実現が挙げられる．また画面の再構築だけでなく，ショートカットの作成や画面分割結合などの実現が挙げられる．この際は動的変化後にイベントが追加されるので，状態遷移機械の大きな変更が必要と想定される．

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.

[3] F. Patern, and G. Zichittella, “Desktop-to-Mobile Web Adaptation through Customizable Two-Dimensional Semantic Redesign,” *Human-Centred Software Engineering*, vol. 6409, pp. 79-94, 2010.