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

江坂 篤侍[1]　野呂 昌満[2]　沢田 篤史[2]

概要：スマートデバイスや Web ブラウザの多様化は，そのソフトウェアの実行時環境や開発環境の多様化を引き起こし，一人の技術者がこれら環境すべてを把握することは難しく，これが生産性向上の妨げとなっている．我々は，参照アーキテクチャは開発環境を規定し，アプリケーションアーキテクチャは実行時環境を定義するとの認識のもと，共通参照アーキテクチャを設計し，それを詳細化して共通アプリケーションアーキテクチャを定義した．これらを既存の参照アーキテクチャならびに実行時環境を規定する既存のアプリケーションアーキテクチャと比較し，それらの関係を考察した．さらに，任意の実行時環境で稼働するアプリケーションの任意の開発環境を用いた作成支援の可能性を考察した．

A Common Architecture for Interactive Software

Esaka Atsushi[1]　Noro Masami[2]　Sawada Atsushi[2]

Abstract: A number of software development environments and runtime environments for variety of smart devices increasingly coming out and that of Web browsers have different shapes one from the other. The variety and the increasing number of environments cause lower quality and/or productivity of interactive software running on the devices and the browsers. The inherent problem of this can be solved to define the problem as that of software architecture. That is, a reference architecture defines a development process and then the process prescribes a development environment. An application architecture is, in turn, reflected by a runtime environment. We have constructed the common architecture which is a set of the common reference architecture and the common application architecture for interactive applications. We also sorted the relationships between the common reference architecture and existing reference architectures. Correspondence between the common application architecture and existing application architectures is also considered. We concluded that there are possibilities for supporting the software production in a development environment for a runtime environment being for a different development environment. We discuss that how the relationships defined and sorted contribute the development support.

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，それらの上で稼働するインタラクティブソフトウェアの実行時環境や開発環境は多岐にわたるようになってきた．開発用のプログラミング言語を例にとっても，Java，Ruby，C#，Objective-C など，様々なものが利用されている．これらの環境を一人のソフトウェア技術者がその詳細にわたりすべて把握することは非常に難しい．さらに，これらの開発において，個別の環境ごとに専門の技術者を必要数確保することは困難である．

この問題に対して，ワンソース (One Source) 開発を可能にするクロスプラットフォーム開発環境 [1] や関連技術の標準化 [3], [5] 等の研究・開発が盛んに行なわれてはいるものの，依然，環境の差異は生産性向上の障壁となっている．これらは特定の実行時環境に対してのみワンソース開発を保証するものであり，さらには，独自の開発プロセスによって開発する必要がある．

特定の開発環境を用いて開発したソフトウェアを，別の開発環境が前提とする実行時環境上で稼働するソフトウェアに自動変換できれば，これらソフトウェアの生産性は向上する．開発環境はソフトウェアアーキテクチャにより規

[1] 南山大学大学院数理情報研究科
Graduate School of Mathematical Sciences and Information Engineering, Nanzan University
[2] 南山大学理工学部ソフトウェア工学科
Department of Software Engineering, Nanzan University

定され [2]，アプリケーションのアーキテクチャは実行時環境により規定される [8]．すなわち，環境の差異に起因する問題はソフトウェアアーキテクチャに関連する課題として定義できる．

本研究の目的は，インタラクティブソフトウェアの共通アーキテクチャを提案することである．複数の事例を基に共通アーキテクチャを共通参照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として定義し，アプリケーションフレームワークの自動生成の可能性および開発プロセスとの関連を考察する．

共通参照アーキテクチャの設計においては，MVC アーキテクチャとその派生である既存のアーキテクチャ [7] を調査し，それらのアーキテクチャが分離を試みている横断的関心事を特定することで，アスペクト指向アーキテクチャとして統合する．共通参照アーキテクチャを詳細化し共通アプリケーションアーキテクチャを設計する．共通アプリケーションアーキテクチャに定義される各個のアスペクトの開発ステップを特定することでアスペクト指向共通開発プロセスを設計する．さらに，このように設計した共通アーキテクチャを，既存の開発環境が前提とする参照アーキテクチャおよび既存の実行時環境によって規定されるアプリケーションアーキテクチャと比較することで，その構造と開発プロセスを洗練する．

結果として，共通アーキテクチャにより，要求に応じたアプリケーションフレームワークの自動生成と開発プロセスの特定が可能であることがわかった．すなわち，任意の実行時環境で稼働するアプリケーションを任意の開発環境を用いて作成する枠組みの基礎を築くことができた．

2. インタラクティブソフトウェアの開発の現状および問題点と解決策

インタラクティブソフトウェアは，Web アプリケーションとネイティブアプリケーションおよびこれらの組み合わせに分類される．近年，インタラクティブソフトウェアは MVC アーキテクチャに基づいて開発されるようになってきた [6]．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 および Controller の開発に際し，多岐にわたる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 が代表例として挙げられ，ネイティブアプリケーションの View 定義には Swing や AWT などのライブラリが利用される．AngularJS，Struts，Ruby on Rails，.Net や Spring 等のインタラクティブソフトウェアのためのアプリケーションフレームワークは，Web 関連技術やネイティブアプリケーション用ライブラリ群を取捨選択して MVC アーキテクチャに基づくアプリケーション構築を支援している．

インタラクティブシステムの開発では，上述のような多様な技術について詳細にわたりすべてを把握することが難しい．さらに，提案されている技術は互換性がないばかりか，通常，技術の選択権は技術者にはなく，どの技術を用いるかは，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．この問題に対してクロスプラットフォーム開発環境が実現，運用され，また Web 技術の標準化が行なわれている．クロスプラットフォーム開発環境は，異なる実行時環境を統一的に扱うために，複数の実行時環境の API を抽象化し，共通の実行時環境をブラックボックスフレームワークとして実現している．Web 関連技術の標準化は，異なるブラウザに対して同様の表示や動作を可能にすることを目的としている．しかし，クロスプラットフォーム開発環境が提供しているブラックボックスフレームワークは，特定のアプリケーションアーキテクチャを前提としていることから，すべての実行時環境を統一的に扱うことができない．また，Web 関連技術の標準は遵守されず，ブラウザベンダ毎の方言ができることが，これまで幾度も繰り返されてきている．要約すると，ある技術の組での開発に習熟した技術者は多数存在するが，多様な類似技術をすべて運用できる技術者は稀である．特定の環境上でのインタラクティブソフトウェアの開発および異なる環境への移行では，技術の多様性に起因して次の問題が起こり，結果として，生産性の低下を招いている．

- 要求に応じたフレームワークの構築が困難
- 開発プロセスの特定とその理解が困難

環境の差異に起因する問題は，ソフトウェアアーキテクチャに関連する課題として定義出来る．一般に，ソフトウェアアーキテクチャは開発プロセスを含意している [2]．開発環境はこの開発プロセスを支援する環境として定義される．アプリケーションフレームワークは，アプリケーションアーキテクチャに基づいて特定の実行時環境の使い方を説明したコードを定義している．すなわち，参照アーキテクチャと開発環境は関連し，アプリケーションアーキテクチャと実行時環境が関連している．参照アーキテクチャが含意するプロセスは開発環境を構成する基本となる．アプリケーションアーキテクチャはアプリケーションフレームワークを構成する基本となり，それが含意するプロセスは，アプリケーションフレームワークを用いた開発プロセスを構成する基本となる．開発環境が前提とする参照アーキテクチャについて，共通の参照アーキテクチャを定義し，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とで，開発環境で用いられる技術間の対応関係を明らかにできる．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についても，共通の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とで，実行時環境間の対応関係を明らかにできる．この様子を図 1 に示す．すな

わち，共通アーキテクチャと環境の技術間の関係が明らかになることから，特定の技術を用いたアプリケーションフレームワークの自動生成と開発プロセスの特定，共通アーキテクチャを介した技術転換の自動化の枠組みの基礎が与えられる．

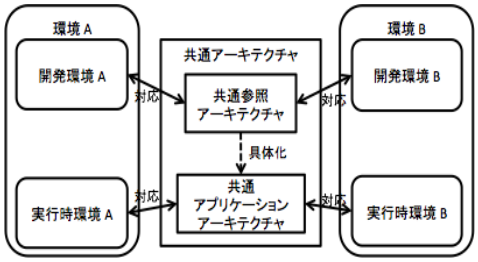

図 1　共通アーキテクチャと実行時環境および開発環境の関係

3. アーキテクチャ設計

ここでは，Clements らの複数のビューによるアーキテクチャ文書化に関する枠組み [4] に基づいてアーキテクチャを記述する．システムに対する関心事は複数存在し単一のビューだけで表現することは難しいことから，複数のビューによりアーキテクチャを記述する．本研究ではアーキテクチャの構造について議論しているので，共通アーキテクチャの記述にはモジュールビュータイプを用いる．共通参照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は，それぞれ抽象ビューと具体ビューとして記述する．

共通参照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設計は，

“複数の既存のアーキテクチャをすべて説明可能とする”

という設計思想に基づいて行なう．あるアーキテクチャを説明可能とは，アスペクト指向アーキテクチャとして定義した共通アーキテクチャにおいて，特定の横断的関心事のいくつかを指定して織込むことで，そのアーキテクチャを生成できることを指す．

3.1 MVC アーキテクチャとその派生である既存のアーキテクチャの調査

共通参照アーキテクチャの設計にあたり，MVC アーキテクチャとその派生として AM-MVC，HMVC，MVP，PAC，MVVM について調査した [7]．

MVC アーキテクチャ (図 2) は，プレゼンテーションロジックからビジネスロジックを分離し，それぞれの独立な変更を可能にする．画面に表示される視覚的要素を扱う View，データとビジネスロジックを含むアプリケーションのモデルを扱う Model，ユーザ操作によるイベントを扱う Controller によって構成される．View と Model の関連は，変更を Model へのプッシュ通信または Model からのプル通信により認知し，画面を更新する ClassicMVC と，View と Model の依存関係を無くし，Controller からのプッシュ通信を画面更新のきっかけとすることを前提とした PassiveMVC がある．

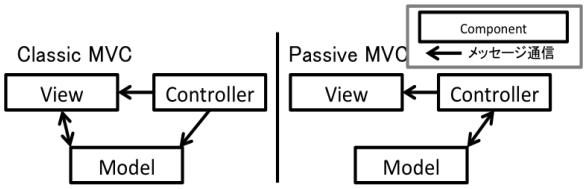

図 2　Model-View-Controller(MVC)

AM-MVC(図 3) は，MVC に ApplicationModel を追加している．ApplicationModel は画面表示用に加工された Model であり，View もしくは Controller に定義されるプレゼンテーションロジックは，この ApplicationModel を用いて View を操作する．

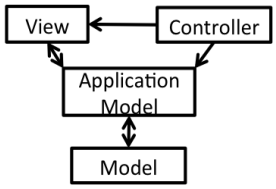

図 3　ApplicationModel-Model-View-Controller(AM-MVC)

HMVC(図 4) と PAC(図 5) は，階層構造を用いて記述することを目的としている．HMVC は，MVC の Controller 同士で階層間の協調を実現する．PAC は，その構成単位を特定の機能を実現する Agent としている．Agent は，画面出力とユーザ入力処理を扱う Presentation，Agent の機能とデータを扱う Abstraction，これらの間の協調と階層間の協調を実現する Control によって構成される．

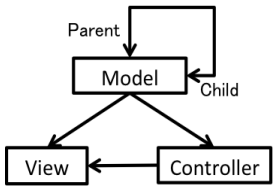

図 4　Hierarchical-Model-View-Controller(HMVC)

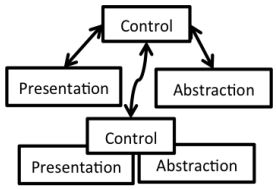

図 5　Presentation-Abstraction-Control(PAC)

MVP(図 6) は，リアクティブシステムに特化した派生である．画面に表示される視覚的要素とユーザ操作によるイベントを扱う View，プレゼンテーションロジックを扱う Presenter，アプリケーションのモデルを扱う Model によって構成される．MVP の View はユーザ操作によるイ

ベント処理と画面出力を行なうことから，MVC の View と Controller に対応する．MVP の Presenter は，MVC の View と Controller に定義されるプレゼンテーションロジックを分離したものである．Presenter と Model，View と Model 間はすべてイベント通知で協調する．これら要素間の関連も，MVC と同様に Classic タイプと Passive タイプがある．

MVVM(図 7) は，プレゼンテーションロジックとビジネスロジックの分離を目的とした，MVC とは異なる分割によるアーキテクチャである．画面に表示される視覚的要素とユーザ操作によるイベントを扱う View，アプリケーションのモデルを扱う Model，View と Model を関連づける View-Model によって構成される．

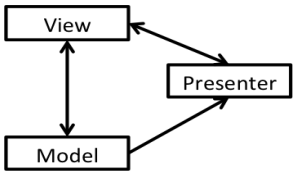

図 6 Model-View-Presenter (MVP)

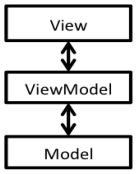

図 7 Model-View-ViewModel (MVVM)

3.2 共通参照アーキテクチャの設計

前節の調査結果に基づき，インタラクティブシステムのプライマリコンサーンをオブジェクト指向コンサーンとしたさいの横断的関心事として次の二つの関心事を識別した．

- MVC コンサーン
- UI(ユーザインタフェース) コンサーン

MVC コンサーンは，MVC，AM-MVC，HMVC に見られるように，画面に表示される視覚的要素を扱う View，アプリケーションのモデルを扱う Model，ユーザ操作によるイベントを扱う Controller アスペクトを規定する関心事である．UI コンサーンは，MVP，PAC，MVVM にみられるように，視覚的要素とユーザ処理によるイベントを扱う UIComponent，アプリケーションのモデルを扱う Model アスペクトを規定する関心事である．

さらに，MVC，UI コンサーンをプライマリコンサーンとしたさいの横断的関心事として次の四つの関心事を識別した．

- 表示ロジックコンサーン
- 表示モデルコンサーン
- 階層化コンサーン
- 通信コンサーン

表示ロジックコンサーンは，MVP にみられるように，View を構築するための Presentation Logic アスペクトを規定する関心事である．表示モデルコンサーンは，AM-MVC，MVVM にみられるように，画面表示用に加工された Model としての ViewModel アスペクトを規定する関心事である．階層化コンサーンは，HMVC，PAC にみられるように，システムの階層関係を規定する関心事である．通信コンサーンは，MVC，MVP にみられるように，要素間の通信の方向性を規定する関心事である．

識別した横断的関心事を分類した結果，次の二つの次元を定義した．

次元 1 :MVC コンサーン，UI コンサーン
次元 2 :表示ロジックコンサーン，表示モデルコンサーン

次元 1 には，オブジェクト指向をプライマリコンサーンとしたさいの横断的関心事が分類され，次元 2 には，次元 1 の横断的関心事をプライマリコンサーンとしたさいの横断的関心事が分類される．階層化コンサーンと通信コンサーンは，アスペクト間の関連を規定するものと捉える．MVC アーキテクチャおよびその派生は，次元 1 における一つの横断的関心事と，次元 2 におけるいくつかの横断的関心事によって規定されるアスペクトを分離している．例えば，図 3 の AM-MVC は，オブジェクト指向に対して MVC コンサーンによって規定されるアスペクトと，MVC コンサーンに対して表示ロジックコンサーンによって規定されるアスペクトを分離している．

二つの次元に分類した横断的関心事に基づき，共通参照アーキテクチャを設計した．設計にあたり，すでに述べたように各次元の横断的関心事を指定し，指定した横断的関心事によって規定されるアスペクトを織込むことで，MVC アーキテクチャおよびその派生のそれぞれを生成とすることを目的とした．

共通参照アーキテクチャを構成する各関心事のビューを図 8 と図 9 に示す．次元 2 に分類される関心事のビューに記述される <<any>> は，MVC もしくは UI ビューのいずれかに織込まれることを示す．階層化コンサーンと通信コンサーンは，いずれも要素間の関連を実現することから，アスペクト間記述 (IAD) として実現することが自然と考えた．

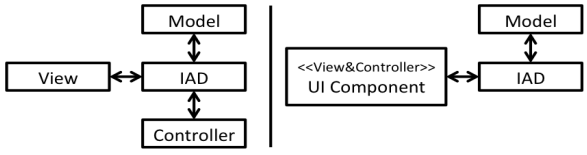

図 8 次元 1:(a)MVC コンサーンビュー，(b)UI コンサーンビュー

3.3 共通アプリケーションアーキテクチャの設計

2 章で述べた通り，参照アーキテクチャを詳細化したものがアプリケーションアーキテクチャである．共通アプリケーションアーキテクチャは，共通参照アーキテクチャを構成するモジュール群 (以下，アスペクトモジュール) に

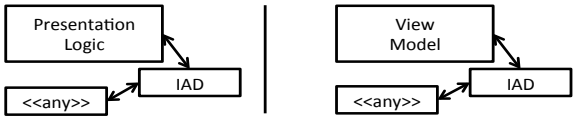

図 **9** 次元 2:(a) 表示ロジックコンサーンビュー，(b) 表示モデルコンサーンビュー

対する横断的関心事を識別することで設計する．識別した横断的関心事を表 1 に示す．

表 **1** アスペクトモジュールから識別した横断的関心事

アスペクト	識別した横断的関心事
Model アスペクト	ビジネスロジックコンサーン
	データアクセスコンサーン
Controller アスペクト	イベント管理コンサーン
Presentation Logic アスペクト	画面構築コンサーン
	画面遷移コンサーン
ViewModel アスペクト	表示コンサーン

MVC コンサーンや UI コンサーンによって規定される Model アスペクトのアスペクトモジュールに対する横断的関心事として，ビジネスロジックコンサーンとデータアクセスコンサーンを識別した．ビジネスロジックコンサーンは，アプリケーションの状態遷移に関する BusinessLogic アスペクトを規定する関心事であり，データアクセスコンサーンは，モジュール群の永続化に関する Data Access アスペクトを規定する関心事である．MVC コンサーンによって規定される Controller アスペクトのアスペクトモジュールに対する横断的関心事としてイベント管理コンサーンを識別した．イベント管理コンサーンは，コントローラの状態遷移と遷移時に起動するアクションに関する EventListener アスペクトと EventHandler アスペクトを規定する関心事である．

表示ロジックコンサーンによって規定される PresentationLogic アスペクトのアスペクトモジュールに対する横断的関心事として画面遷移コンサーンと画面構築コンサーンを識別した．画面遷移コンサーンは，画面遷移に関する ViewTransition アスペクトを規定する関心事であり，画面構築コンサーンは，遷移時に画面を構築する手続きに関する ViewConstructor アスペクトを規定する関心事である．

表示モデルコンサーンによって規定される ViewModel アスペクトのアスペクトモジュールに対する横断的関心事として表示コンサーンを識別した．表示コンサーンは，内容と役割と見栄えに関する ViewContent アスペクト，DisplayImageContent アスペクト，Style アスペクトを規定する関心事である．

識別した横断的関心事に基づき，共通アプリケーションアーキテクチャを設計した．それぞれのコンサーンに着目した各ビューを共通アプリケーションアーキテクチャとして記述する．本稿では，各々のビューについては，ページ数制限の都合上省略する．図 10 は MVC コンサーン，表示ロジックコンサーン，画面 Model コンサーンを分離した参照アーキテクチャを詳細化したアプリケーションアーキテクチャの例である．

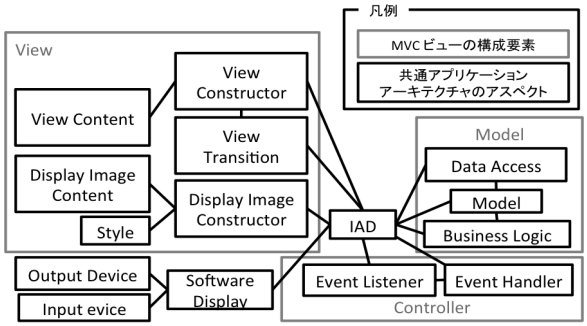

図 **10** 共通アプリケーションアーキテクチャ

3.4 共通開発プロセスの設計

インタラクティブシステムの開発において，開発者は Model，View，Controller に関連して次の 6 項目を定義する必要がある．

- アプリケーションのモデル，アプリケーションの状態遷移
- 画面レイアウト (内容，役割，見栄え)，画面遷移
- イベント，コントローラの状態遷移

実装は，この 6 項目を入力として，特定の開発技術に依存した一連の開発ステップ群によって行なわれる．前提とするアプリケーションアーキテクチャによって，特定の開発ステップ群の組み合わせが変わる．

共通アプリケーションアーキテクチャが含意する開発プロセスを明らかにすることで，共通開発プロセスを設計する．共通アプリケーションアーキテクチャはアスペクト指向アーキテクチャであることから，アスペクト毎にそのアスペクトモジュールを開発するための開発ステップ群が定義される (以下，開発ステップアスペクト)．共通アプリケーションアーキテクチャにおけるアスペクトの織込みは，開発プロセスの側面では，特定の開発ステップの実行をジョインポイントとし，after アドバイスまたは around アドバイスとして開発ステップアスペクトを織込むことで開発プロセスを生成する．

開発ステップアスペクトは，特定の開発技術を用いた開発ステップを抽象化することで，開発技術から独立した標準開発ステップとして定義する．開発者が各個の開発技術の開発ステップを全て把握することは困難であるが，この標準ステップを介して，特定の開発技術の開発ステップから，異なる開発技術の開発ステップを理解可能になる．

図 11 は，BusinessLogic アスペクトと DataAccess アス

ペクトの開発ステップアスペクトとその織込みを示す．ジョインポイントとアドバイスは，AspectC の記述を用いている．この各個の開発ステップアスペクトの実行は，指定した開発技術に特化して実行される．

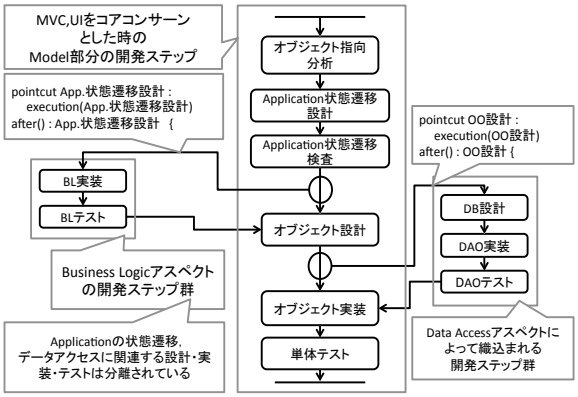

図 **11** 開発ステップアスペクトの織込み例

4. 共通アーキテクチャと既存の参照アーキテクチャおよび既存の環境の関係

共通参照アーキテクチャと MVC およびその派生，共通アーキテクチャと既存の環境の前提とする参照アーキテクチャおよびアプリケーションアーキテクチャそれぞれについて比較し，その関係を議論する．

4.1 既存の参照アーキテクチャとの関係

前述のとおり，共通参照アーキテクチャは，特定の横断的関心事のいくつかを指定し，織込むことで，MVC とその派生が生成される．4.2 節で説明する既存の環境が前提とする参照アーキテクチャである PassiveMVC コンサーンと表示ロジックコンサーン，MVVM コンサーンを指定することによる参照アーキテクチャの生成について例にあげる．

PassiveMVC コンサーンは，Controller を介して Model と View が協調する分割を規定する．表示ロジックコンサーンは，Controller もしくは View に記述されるプレゼンテーションロジックを規定する．図 12 は，図 8(a) と図 9(a) のアスペクトの織込みの結果を示す．

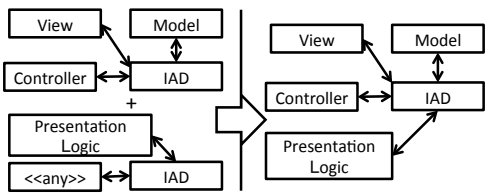

図 **12** PassiveMVC コンサーンと表示ロジックコンサーンの指定によって生成されるアーキテクチャ

MVVM は，UI コンサーンによって規定される分割に対して横断する表示モデルコンサーンを分離している．図 13 は，図 8(b) と図 9(b) のアスペクトの織込みの結果，MVVM アーキテクチャが生成されることを示している．

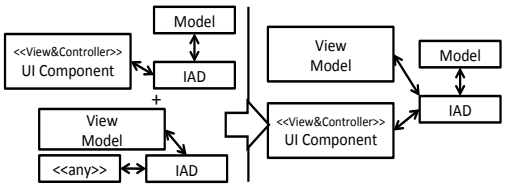

図 **13** UI コンサーンと表示モデルコンサーンの指定によって生成されるアーキテクチャ

4.2 既存の環境が前提とする参照アーキテクチャおよびアプリケーションアーキテクチャとの関係

一般にインタラクティブシステムは，ネイティブアプリケーションと Web アプリケーションに分類され，別の観点ではアプリケーションフレームワークを用いて実現されたアプリケーションとクロスプラットフォーム開発環境を用いて実現されたアプリケーションに分類される．共通アーキテクチャと環境との関係の例として，典型的に対称なものを挙げる．すなわち，Web アプリケーションの環境の例として Ruby on Rails，ネイティブアプリケーションの環境の例として PhoneGap をとり上げる．Ruby on Rails は，実行時環境として，Ruby インタプリタ，テンプレートエンジン，これらを実行可能な環境を前提としている．PhoneGap は，Android や iOS などの OS を対象とした統一実行時環境をブラックボックスフレームワークとして提供している．この実行時環境では HTML，CSS，JavaScript で実現されたアプリケーションを実行する．

4.2.1 Ruby on Rails との関係

参照アーキテクチャ間の関係

Ruby on Rails の開発環境が前提とする参照アーキテクチャは，PassiveMVC コンサーンと表示ロジックコンサーンを分離する．開発者はこの開発環境を用いて，Controller，Model，View のテンプレートを実装する．Model とテンプレートを用いた表示ロジック間の通信は，必ず Controller を経由する．

共通アーキテクチャは，前節で説明した通り，PassiveMVC コンサーンと表示ロジックコンサーン選択することで，これらを分離したアーキテクチャを生成できる (図 12)．したがって，共通参照アーキテクチャは Ruby on Rails の参照アーキテクチャを説明可能である．

アプリケーションアーキテクチャ間の関係

Ruby on Rails の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と共通アプリケーションアーキテクチャの関係について考察する．図 14 は，共通アプリケー

ションアーキテクチャから，横断的関心事を指定することで生成されたアプリケーションアーキテクチャである．このアプリケーションアーキテクチャは PresentationLogic アスペクトに横断する画面構築コンサーン，画面遷移コンサーン，Model アスペクトに横断するビジネスロジックコンサーンとデータアクセスコンサーン，View アスペクトに横断するイベント管理コンサーンを分離している．アプリケーションフレームワークの構成要素である ERB Engine はテンプレートを用いて画面を構築することから，画面構築コンサーンによって規定された要素である．ActionView は通知されるイベントに応じて利用するテンプレートを決定することにより画面遷移を実現しているので，画面遷移コンサーンによって規定された要素である．ActiveRecord は DB アクセスに関連する手続きを実現することから，データアクセスコンサーンによって規定された要素である．ActionDispatch は，ブラウザからの外部イベントを内部イベントに変換し，ActionController は，イベントに応じて BusinessLogic または ActionView へのルーティングを実現することから，イベント管理コンサーンによって規定された要素である．このように，アプリケーションフレームワークの構成要素と，生成されたアプリケーションアーキテクチャが対応付くことから，共通アプリケーションアーキテクチャは，Ruby on Rails のアプリケーションアーキテクチャを説明可能である．

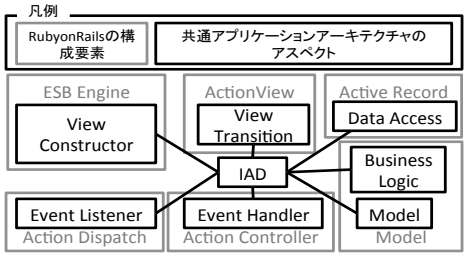

図 **14** 共通アプリケーションアーキテクチャと Ruby on Rails のアプリケーションアーキテクチャの関係

生成されたアプリケーションアーキテクチャの開発プロセスの側面では，Ruby on Rails の開発プロセスが生成されることを確認する．ページ数制限の都合上，ここでは画面構築コンサーンと画面遷移コンサーンによって規定される ViewConstructor アスペクトと ViewTransition アスペクトの開発ステップアスペクトの織込みを図 15 に示す．同様にその他の開発ステップアスペクトを織込むことで，Ruby on Rails の開発プロセスが生成されることを確認した．

4.2.2 PhoneGap との関係

参照アーキテクチャ間の関係

PhoneGap の開発環境は参照アーキテクチャとして MVVM を前提としている．開発者はこの開発環境を用いて，画面の入出力と，Model を実装する．

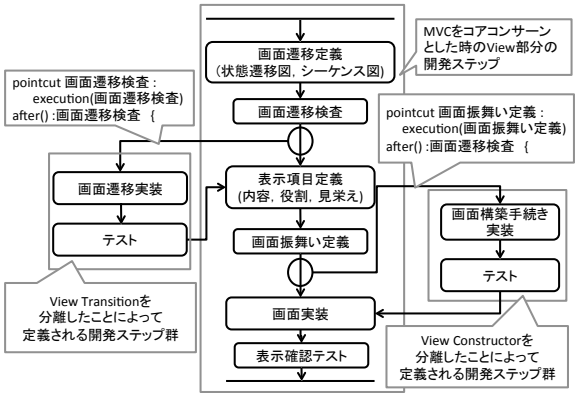

図 **15** ViewConstructor アスペクトと ViewTransition アスペクトの織込み例

共通参照アーキテクチャは，4.1 節で説明したとおり，UI コンサーンと表示モデルコンサーンを選択することで，これらを分離したアーキテクチャを生成できる (図 13)．したがって，共通参照アーキテクチャは PhoneGap の参照アーキテクチャを説明可能である．

アプリケーションアーキテクチャ間の関係

PhoneGap の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と共通アプリケーションアーキテクチャの関係について考察する．実行時環境は HTML，CSS，JavaScript で実現されたアプリケーションを実行することから，アプリケーションアーキテクチャは，HTML，CSS，JavaScript を用いて実装されるアプリケーションが前提とするものである．図 16 は，共通アプリケーションアーキテクチャから，横断的関心事を指定することで生成されたアプリケーションアーキテクチャである．このアプリケーションアーキテクチャは，Model アスペクトに横断するビジネスロジックコンサーンとデータアクセスコンサーン，ViewModel アスペクトに横断する表示コンサーンを分離している．PhoneGap では，JavaScript を用いて，UI コンサーンによって規定される Model アスペクト，ビジネスロジックコンサーンによって規定される BusinessLogic アスペクト，データアクセスコンサーンによって規定される DataAccess アスペクトが実装され，HTML，CSS を用いて，UI コンサーンによって規定される UIComponent アスペクトが実装される．このように PhoneGap で実装されるアプリケーションの構成要素と，生成されたアプリケーションアーキテクチャが対応付くことから，共通アプリケーションアーキテクチャは，PhoneGap のアプリケーションアーキテクチャを説明可能である．

生成されたアプリケーションアーキテクチャの開発プロセスの側面では，PhoneGap の開発プロセスが生成されることを確認する．3.4 節では，ビジネスロジックコンサーンとデータアクセスコンサーンによって規定される

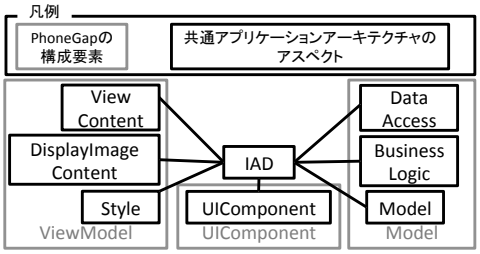

図 **16** 共通アプリケーションアーキテクチャと Phone Gap のアプリケーションアーキテクチャの関係

BusinessLogic アスペクトと DataAccess アスペクトの開発ステップアスペクトの織込みを示した．同様にその他の開発ステップアスペクトを織込むことで，PhoneGap の開発プロセスが生成されることを確認した．

5. 考察

アスペクト指向アーキテクチャとして共通アーキテクチャを設計することで，幾つかの既存の参照アーキテクチャと既存のアプリケーションアーキテクチャを説明可能にすることができた．プライマリコンサーンによって規定される分割に対して矛盾する関心事について，アスペクトとして分離することで，これらを矛盾なく説明可能となった．また，開発プロセスの側面では，アスペクトモジュールを開発するための開発ステップアスペクトを定義した．これにより，いくつかの事例において，アーキテクチャ上のアスペクトの織込みに応じて，開発ステップアスペクトが織込まれ，開発プロセスを特定することができた．

共通アーキテクチャにより，任意の開発環境を用いて任意の実行時環境上で稼働するアプリケーションの作成支援の枠組みの基礎を与えることができた．共通アーキテクチャをプロダクトラインアーキテクチャとした，インタラクティブシステムのプロダクトライン開発を実現することで，コア資産間の追跡性が確保される．仕様の決定によって，横断的関心事が指定され，既存の参照アーキテクチャおよび既存のアプリケーションアーキテクチャと開発プロセスを決定できる．また，MDA に基づくことで，PIM を既存のアーキテクチャ，PSM をフレームワークとし，入力される仕様に応じてアプリケーションフレームワークの自動生成が可能となる．

現状では，全ての事例について共通アーキテクチャが説明可能であることを確認していない．今後，様々な事例で確認することで共通アーキテクチャを洗練する必要がある．特定のアーキテクチャを共通アーキテクチャで説明できない場合は，本研究で提案された共通アーキテクチャ定義の枠組みに基づいて共通アーキテクチャを洗練する．

6. おわりに

インタラクティブソフトウェアの実行時環境と開発環境は多岐にわたることから，環境の差異が生産性向上の障壁となっている．本研究では，インタラクティブソフトウェアの共通アーキテクチャを提案することでこの問題の解決を試みた．共通アーキテクチャに基づき特定の実行時環境のためのフレームワークの自動生成および開発プロセスの決定を可能にする．共通参照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からなる共通アーキテクチャを定義し，既存の環境との関連付けを行なった．共通参照アーキテクチャの設計においては，MVC アーキテクチャとその派生である既存のアーキテクチャを調査し，それらのアーキテクチャが分離を試みている横断的関心事を特定することで，アスペクト指向アーキテクチャとして統合した．共通アプリケーションアーキテクチャは，共通参照アーキテクチャを詳細化することにより設計した．共通アプリケーションアーキテクチャの含意する共通開発プロセスを設計した．共通アーキテクチャは，特定の事例において，その環境の前提とする参照アーキテクチャおよびアプリケーションアーキテクチャと開発プロセスを説明可能であることを確認した．これにより，任意の実行時環境で稼働するアプリケーションの任意の開発環境を用いた作成支援を実現する基礎ができた．

今後の課題は，様々な環境との関係を考察することによる共通アーキテクチャの洗練と，共通アーキテクチャおよび開発ステップと開発技術間の関係を明らかにすることである．

謝辞 本研究の一部は，2015 年度南山大学パッヘ奨励金 I-A-2 の助成による．

参考文献

[1] Allen, S., Graupera, V., and Lundrigan, L. :*Pro smartphone cross-platform development: iPhone, blackberry, windows mobile and android development and distribution*. Apress (2010).
[2] Bass, L. :*Software architecture in practice*, Addison Wesley(2007).
[3] Bos, B., Celik, T., Hickson, I. and et al. :Cascading Style Sheets, W3C (online), available from `<http://www.w3.org/TR/CSS2/>`(accessed 2015-05-13)
[4] Clements, P., Bachmann, F., Bass, L. and et al. :*Documentiong Software Architectures Views and Beyond Second Edition*, Addison Wesley (2010).
[5] Hickson, L., Berjon, R., Faulkner, S. and et al. : HTML, WHATWG, W3C (online), available from `<http://www.w3.org/TR/html5/>`(accessed 2015-05-13).
[6] Krasner, G. E., and Pope, S. T. :*A description of the model-view-controller user interface paradigm in the smalltalk-80 system*. object oriented programming, Vol.1, No.3, pp. 26-49 (1988).
[7] Sokolova, K., Lemercier, M., and Garcia, L. :*Towards High Quality Mobile Applications: Android Passive MVC Architecture*. International Journal On Advances in Software, Vol. 7, No. 2, pp. 123-138(2014).
[8] Vliet, H. V. :*Software engineering: principles and practice.*, Wiley (2007).