携帯端末プラットフォームにおける アプリケーションアーキテクチャ設計に関する研究

M2012MM047 山本幸法
指導教員：野呂昌満

1 はじめに

携帯端末プラットフォームに関する技術は多様化・複雑化している．それぞれのプラットフォームで提供されている API や，アーキテクチャは異なっており，アップデートによる追加・変更の可能性がある．開発者はプラットフォームで提供される API やアーキテクチャを理解し，プラットフォームごとにアプリケーションの開発を行なう必要がある．

携帯アプリケーション開発の問題として，アプリケーションの開発がプラットフォームに依存しており，統一的な開発を行なうことができない点がある．多様化・複雑化するプラットフォームに関する技術を，プラットフォームごとにすべて習得することは困難である．開発者は技術を習得した上で，同じ機能を持つアプリケーションであっても，プラットフォームに応じて個別に開発を行なわなければならない．

本研究の目的は，プラットフォームに依存しない統一的な開発を可能にするアプリケーションアーキテクチャの設計である．コンポーネントの再利用とプラットフォームに依存しないモデル (以下，PIM) を入力とした自動生成により，プラットフォームに関する技術に依存しない開発を目指す．

本研究では，携帯端末プラットフォームの差異を吸収するアプリケーションプラットフォーム (以下，App.PF) のアーキテクチャを設計し開発プロセスを明確にする．App.PF では，統一的な開発を行なうための携帯アプリケーションの共通構造を提供し，特定のアプリケーションに依存しない機能と非機能を実現する．

App.PF の開発は，PLSE (Product-Line Software Engineering)[5] を適用したフレームワークを用いて行なう．PLSE を適用する理由として，共通部分と変動部分を明確化することで，コンポーネントの再利用による開発が可能になる点がある．

App.PF のプロダクトラインアーキテクチャ(以下，PLA) は，MVC に基づいたアスペクト指向アーキテクチャとして設計する．MVC に基づく理由として，アプリケーションに依存した機能を提供するコンポーネントを Model 要素として分離できる点がある．アスペクト指向設計を行なう理由として，非機能に関係する横断的関心事をアスペクトとして分離し，疎結合を実現できる点がある．これにより，コンポーネントを独立にすることができ，コンポーネントの再利用による開発を支援できる，

プラットフォームの技術に依存しない系統的な App.PF 開発のために，伴らの研究 [7] に基づき仕様モデルとアーキテクチャの変動部分の対応関係を明確にする．これにより，PLA からプロダクトアーキテクチャ(以下，PA) を，プラットフォームの技術に依存することなく構築可能となる．

2 研究の概要

本研究では，携帯端末プラットフォームの技術に依存しない開発を可能にするアプリケーションアーキテクチャの設計を行なう．アプリケーションアーキテクチャを設計するためには，以下の 2 点を達成する必要がある．

- プラットフォームの技術に依存しない開発プロセスの構築
- プラットフォームの差異を吸収し，コンポーネントベースの開発を可能にする共通アーキテクチャの設計

共通アーキテクチャの設計と，共通アーキテクチャに基づいた技術に依存しない開発を実現するために，PLSE を適用した App.PF の開発を行なう．プラットフォームの差異を吸収する App.PF を製品系列として定義し，共通部分と変動部分を明確化することで，コンポーネントの再利用による開発を支援できる．App.PF が対応する携帯端末プラットフォームの対象は，Android OS と iOS とする．本研究では，App.PF を製品系列としたフレームワークをプラットフォームフレームワーク (以下，PFW) と定義する．

プラットフォームに関する技術に依存しない開発を実現するために，伴らの研究 [7] を参考にして，仕様の変動部分とアーキテクチャの変動部分の対応付けを行なう．Bachmann ら [2] の定義した変動性の分類から製品系列の変動部分を分析し，FORM[4] の記法に基づいて仕様モデルを構築する．

各プラットフォームに対応した App.PF の共通アーキテクチャ(PLA) として，MVC に基づいたアスペクト指向アーキテクチャの設計を行なう．MVC に基づいたアスペクト指向設計を行なう利点として，特定のアプリケーションに依存する機能と非機能に関係する関心事をアスペクトとして分離できる．これにより，コンポーネントベースの開発を支援できる．本研究では，PLA を以下の視点で記述する．

- アスペクト間の関係
- アスペクトを構成するコンポーネント間の関係と標準インターフェイス
- アスペクトを構成するコンポーネントの変動性

アスペクト間の関係の記述は，App.PF に存在する関心事を分析し，プリミティブな関心事をアスペクトとして定義することで行なう．対象とする携帯端末プラットフォームのフレームワークと，ISO9126 [3] をから関心事を分析する．分析した関心事を MVC の関心事において識別することでプリミティブな関心事を特定し，アスペ

クト間の関係を記述した．

アスペクト間の関係の記述から，アスペクトを構成するコンポーネントの関係を明確化し，共通の構造を記述する．それぞれのコンポーネントの標準インターフェイスを定義することで，プラットフォームに依存した実現技術を隠蔽する Wrapper コンポーネントを生成できる．

PLA の変動性の記述として，仕様の変動部分を実現する候補値と，アーキテクチャ上の変動部分を実現するコンポーネントの対応関係を明確にする．対応関係の記述についても，伴らの研究を参考にして行なった．仕様とアーキテクチャの対応関係を明確にすることで，仕様の変動に対して一意に PA を構築可能となり，プラットフォームに依存しない開発が可能となる．

App.PF の実装は PFW では行なわず，App.PF を利用するアプリケーションを製品系列とするアプリケーションフレームワーク (以下，AFW) で行なう．App.PF の実装は特定のアプリケーションに依存していることから，特定のアプリケーションの PIM を入力として App.PF のコンポーネントを生成する．PLA のアスペクトを構成するコンポーネントの標準インターフェイスを定義することで，PIM の設計を支援できる．

3　App.PF に存在する関心事の分析

各プラットフォームのフレームワークと ISO9126 で定義されている非機能特性から，App.PF に存在する関心事の分析を行なった．フレームワークと ISO9126 から識別した App.PF に存在する関心事を表 1 に示す．

表 1　App.PF に存在する関心事

分析した基準	App.PFに存在する関心事	
各プラットフォームのフレームワーク	EventListening	StateManagement
	ViewManagement	Persistence
ISO9126	Concurrency	Persistence
	ExceptionHandling	StaticsAnalysis
	FaultTolerance	

フレームワークから関心事の分析を行なう際には，それぞれのプラットフォームで提供されている API から行なった．ViewManagement の関心事を例に挙げると，この関心事は Android OS と iOS ではそれぞれ，Activity クラスと UIViewController クラスに存在する．ISO9126 から分析する際には，定義されているすべての非機能特性に対して，App.PF の製品系列に必要な関心事を整理することで行なった．

4　製品系列の変動部分の分析

伴らの研究を参考にして，App.PF の変動部分の分析を行なった．Bachmann[2] らの定義した変動性の分類に基づいて，変動部分と実現候補を表 2 に示す．

表 2　整理した App.PF の変動部分

変動性の分類	変動部分(variation point)	実現候補	依存関係にある変動部分
Function	処理するイベント	クリック，ロングタップ，etc.	−
	画面表示方法	ウインドウ，ポップアップ，etc.	−
	並列処理	並列処理の有無	−
	永続化	データ永続化の有無	−
Data	画面部品	ボタン，ラベル，etc.	−
	使用するレイアウト	リスト，グリッド，etc.	−
Control Flow	アスペクト間記述の実現方法	Aspect，デザインパターン	言語
Technology	OS	Android OS, iOS	−
	実現技術	すべてのコンポーネントの実現方法	OS
	永続化方法	XML, SQLite	−
Quality Goal	Maturity	FaultTolerance, ExceptionHandling	−
	FaultTolerance	FaultTolerance, ExceptionHandling	−
	Recoverability	ExceptionHandling, Persistence	−
	Analyzability	StaticsAnalysis	−
	Testability	StaticsAnalysis	−
Environment	言語	Java+AspectJ, Objective−C	OS

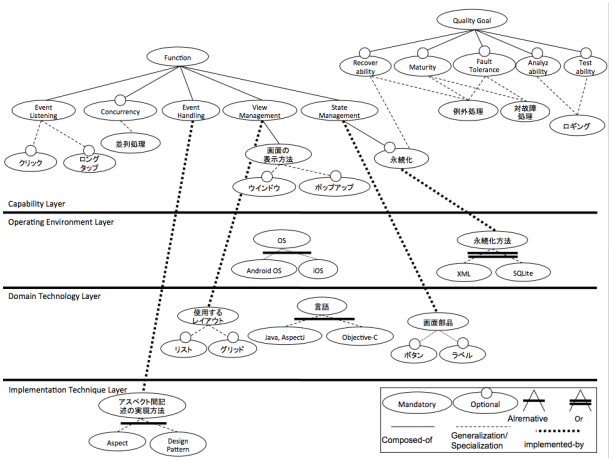

図 1　App.PF のフィーチャモデル

5　仕様モデルの構築

整理した関心事と変動部分から，仕様モデルの構築を行なった．仕様モデルの記述には，FORM[4] の記法を用いた．特定した変動部分を，伴らの研究で整理した変動性の分類と FORM のレイヤの対応付けにしたがって記述することで，仕様モデルを構築した．構築した仕様モデルを図 1 に示す．

6　PLA 設計

整理した関心事と，仕様モデルから PLA の設計を行なった．App.PF の PLA は，MVC のアーキテクチャスタイルに基づいて構築した．PLA は以下の視点でそれぞれ記述した．

- アスペクト間の関係
- アスペクトを構成するコンポーネント間の関係と標準インターフェイス
- アスペクトを構成するコンポーネントの変動性

6.1　アスペクト間の関係

アスペクト間の関係について記述した PLA の設計を行なった．アスペクト間の関係を記述するために，MVC の関心事から整理した関心事を分類し，プリミティブな関心事を特定する．特定したプリミティブな関心事をアスペクト指向アーキテクチャにおけるアスペクトとして定義し，アスペクト間の関係を記述する．

MVC の関心事から整理した関心事を分類し，包含関係を明確化した．MVC と整理した関心事の関係を図 2 に示す．MVC の関心事は，Model・View・Controller・Model と View 間の EventHandling が存在する．

特定したプリミティブな関心事をアスペクト指向アーキテクチャにおけるアスペクトとして定義し，アスペクト間の関係を記述した．アスペクト間の関係記述は，Clements ら [6] が定義したアーキテクチャ記述法の Component & Connector View Type のスタイルに基づいて記述した．記述したアスペクト間の関係を図 3 に示す．

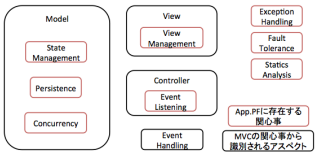

図 2　MVC と整理した関心事の関係

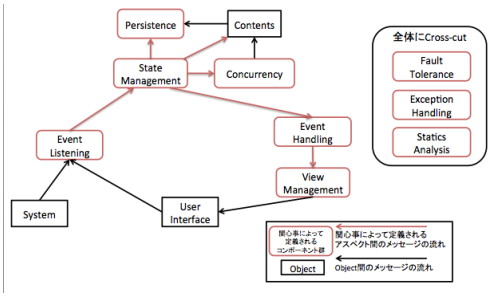

図 3　アスペクト間の関係

それぞれのアスペクトは複数のコンポーネントで構成されており，仕様の変動点を実現する候補によってアスペクトを構成するコンポーネントは変化する．App.PF の共通の構造を定義する上で，アスペクトを構成する共通のコンポーネントの関係と，共通のコンポーネント単位の変動性を記述する必要があると考える．

6.2　アスペクトを構成するコンポーネント間の関係

App.PF の製品系列における共通構造として，アスペクトを構成する共通コンポーネントの関係を記述した．記述は，Component & Connector View Type のスタイルに基づき UML の記法で行なった．アスペクト間記述 (以下，IAD) が横断するコンポーネントの記述には，Aspect Style を組み合わせることで記述した．

記述したアスペクトを構成するコンポーネント間の関係を図 4 に示す．図 4 に示したコンポーネントに対して，標準インターフェイスを定義することで，実現技術を意識しない開発が可能となる．

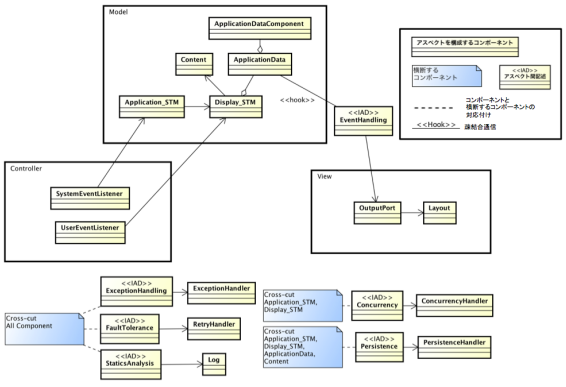

図 4　アスペクトを構成するコンポーネント間の関係

6.3　アスペクトを構成するコンポーネントの変動性

構築した仕様モデルの変動部分の候補と，アーキテクチャ上の変動部分となるコンポーネントの実現候補を対応付けることで，変動性を記述した．変動性を記述した一例として，ApplicationDataComponent の変動性を表 3 に示す．

App.PF の製品系列においては，それぞれのコンポーネントを実現するプラットフォームに依存した技術もアーキテクチャ上の変動点であると考える．仕様の変動部分である OS の候補の値に応じて，アーキテクチャ上の各コンポーネントの実現技術を赤いノートで記述した．

表 3　ApplicationDataComponent の変動性

仕様上の variation point	仕様上のvariant \ コンポーネントの実現の値	Button / Button Class	Label / TextView Class	Button / UIButton Class	Label / UILabel Class
OS	Android OS	○	○	-	
	iOS	-	-	○	○
画面部品	ボタン	○	-	○	-
	ラベル	-	○	-	○

ApplicationDataComponent は仕様において選択される画面部品によって変動することを示している．各プラットフォームの画面部品を抽象化したコンポーネントが変動の候補値となっており，仕様において選択される画面部品に応じて変動する．選択された OS に応じて，抽象化したコンポーネントの実現技術を対応付けた．

7　PA 構築

仕様において iOS 選択し，選択フィーチャをすべて選択したと仮定した場合の PA の例を示す．PLA の変動性の記述に基づいて構築した PA を図 5 に示す．

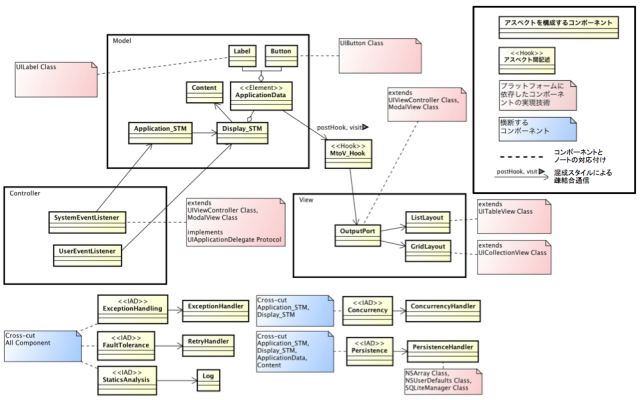

図 5　iOS 選択時の PA

仕様上で OS の候補値として iOS が選択された場合，アスペクト指向言語を使用できないことから，デザインパターンを用いることで IAD を実現する．デザインパターン [1] である Hook Operation Pattern と Visitor Pattern を組み合わせ，混成スタイルとして定義する．

混成スタイルには，Element と Hook のコンポーネントが存在し，アーキテクチャ上ではステレオタイプで表現する．Element コンポーネントは，Hook Operation Pattern における Hook へのメッセージ送信を行なうコンポーネントであり，Visitor Pattern における Element である．Hook コンポーネントは，Hook Operation Pattern にお

ける Hook であり，Visitor Pattern における Visitor である．Element コンポーネントに Template Method として，Element 自身を引数に持つ Hook へのメッセージ送信を定義する．Hook コンポーネントは，Visitor Pattern おける Visitor と同様に Element に応じた処理を行なう．これにより，Element の記述を変更することなく，アスペクト指向に基づいて IAD を実現できる．

Android OS が選択された場合には，アスペクト指向言語を使用できることから，AspectJ における Aspect で IAD を実現できる．Aspect 指向言語によって疎結合通信が実現されることから，構造の変更は行われず PLA と同じ構造となる．

8 考察

8.1 PLSE を適用した App.PF の開発

携帯端末プラットフォームの差異を吸収する App.PF を製品系列として定義し，PLSE を適用した開発を行なった．伴らの研究に基づいて，仕様モデルとアーキテクチャの対応関係を明確化することで，プラットフォームの技術に依存しない PA の構築が可能となった．これにより，携帯端末アプリケーションのための App.PF のドメインにおいても，仕様とアーキテクチャの対応付けによる PA 構築が可能であることを確認できた．構築した PA に対して，再利用コンポーネントや，ジェネレータを実装することで，App.PF の実現技術を隠蔽する Wrapper コンポーネントを生成できると考える．

8.2 App.PF に存在する関心事の分析

App.PF に存在する関心事を，携帯端末プラットフォームのフレームワークで使用される API と ISO9126 で定義される非機能特性から分析した．フレームワークで使用される API から分析することで，携帯アプリケーションに共通の基本的な非機能を分析した．しかし，それぞれの携帯端末プラットフォームで提供されている API を網羅的に調査することはできておらず，他の関心事が存在する可能性があると考える．ISO9126 で定義される非機能特性については，網羅的に分析を行なう事ができたと考える．

8.3 設計した PLA

各携帯端末プラットフォームに対応する App.PF を製品系列として定義し，MVC に基づいた PLA のアスペクト指向設計を行なった．MVC に基づいたアスペクト指向設計を行なうことで，関心事をアスペクトとして分離し，コンポーネントの再利用による開発が容易になった．また，Model 要素として機能を実現するコンポーネントを分離することで，App.PF を変更することなく，機能を実現するコンポーネントを再利用することができる．このことから，MVC に基づいたアスペクト指向設計を行なったことは妥当であると考える．

アスペクトを構成するコンポーネント間の関係を明確化することで，差異を吸収する App.PF の共通構造を記述することができた．共通構造におけるコンポーネントの標準インターフェイスを定義することで，プラットフォームに依存しないアプリケーションの設計が可能になると考える．

携帯アプリケーションのための App.PF のドメインでは，共通構造におけるコンポーネントの実現技術もアーキテクチャ上の変動部分として記述した．共通の構造に対して，プラットフォームに依存した実現技術を対応付けることで，実現技術を隠蔽する Wrapper コンポーネントの生成を支援できる．よって，実現技術を変動部分として扱う事は妥当であると考える．

9 おわりに

本研究では，プラットフォームに依存しない統一的な開発を可能にするアプリケーションアーキテクチャの設計を行なった．プラットフォームの差異を吸収する App.PF の共通アーキテクチャと，仕様とアーキテクチャの対応関係から，プラットフォームの技術に依存せず，PA を構築できる．

今後の課題として，次のことを行なう必要がある．

- AFW において App.PF の Wrapper コンポーネントを生成する際に入力となる PIM の明確化
- Wrapper コンポーネントの生成規則
- 再利用コンポーネント開発の支援

参考文献

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

[2] F. Bachmann, L. Bass, “ Managing Variability in Software Architectures, ” *Proceedings of the 2001 symposium on Software reusability*, pp. 126-132, 2001.

[3] ISO/IEC, *Software engineering Product quality - Part 1: Quality model,* 2001.

[4] K. C. Kang, S. Kim, J. Lee, K. Kim, G. J. Kim, and E. Shin,“ FORM: A Feature-Oriented Reuse Method with Domain-Specific Reference Architectures, ” *Annals of Software Engineering*, vol. 5, no. 1, pp. 143-168, 1998.

[5] K. Pohl, G. Bockle, and F. Linden, *Software Product Line Engineering Foundations, Principles, and Techniques*, Springer-Verlag, 2005.

[6] P. Clements, F. Bachmann, L. Bass, D. Garkan, J. Ivers, R. Little, P. Merson, R. Nord, and J.Stafford, *Documenting Software Architectures Views and Beyond Second Edition*, Addison-Wesley, 2010.

[7] 伴昌樹，吉川翔平,“ SOA アプリケーションプラットフォームのプロダクトライン化に関する研究 –アスペクト指向に基づくプロダクトアーキテクチャの自動生成–, ”南山大学情報理工学部ソフトウェア工学科, 2014.