組込みシステムにおける仕様とアーキテクチャとの関係の考察

2006MI055 垣見 和也　　2006MI078 児玉 優太　　2006MI182 玉木 佑一
指導教員　野呂 昌満　沢田 篤史　蜂巣 吉成

1　はじめに

近年，ユーザの要求の多様化による組込みシステムの多機能化，大規模化に伴い，システム開発の生産性の向上が課題とされている．この課題を解決するソフトウェア開発方法論の 1 つとして, Product Line Software Engineering(以下，PLSE)[3] が提案されている．PLSE は製品系列で共通に開発する部品をコア資産として開発し，再利用することで，ソフトウェア開発の生産性の向上を実現する．部品とは，開発工程で開発する仕様やアーキテクチャ，プログラムコードなどである．

本研究室は組込みシステムのためのアスペクト指向ソフトウェアアーキテクチャスタイル (以下，E-AoSAS++)[4] を提案している．E-AoSAS++ に基づく開発体系は PLSE を開発の基礎としている．組込みシステムに散在する横断的関心事をアスペクトとして分離することで再利用性の高いアーキテクチャ設計が行なえる．

PLSE のドメイン工学におけるアーキテクチャ設計にはプロダクトラインアーキテクチャの作成を行なう場合と，作成した既存のプロダクトラインアーキテクチャのカスタマイズを行なう場合がある．どちらの場合においても，仕様モデルとアーキテクチャの対応関係が明確でないことから，アーキテクチャ設計を行なう際に多くの工数が伴う．対応関係を明確にするにあたり，先行研究 [6] から，仕様モデルのコンポーネントの意味を考慮せずに構文から対応関係を明確にすることは困難であることが確認できた．

本研究の目的は仕様モデルのコンポーネントが表す構文と意味から，アーキテクチャのコンポーネントとの対応関係を明確にすることである．仕様モデルとアーキテクチャの対応関係を明確にすることで，アーキテクチャ設計の系統的な手法の確立と省力化を目指す．

仕様モデルには Feature-Oriented Reuse Method(以下，FORM)[1] など，PLSE の開発において代表的な仕様モデルとして用いられているフィーチャモデルを用いる．アーキテクチャには E-AoSAS++ に基づくアーキテクチャを用いて，フィーチャモデルとの対応付けを行なう．本研究は，GoF デザインパターン [2] は横断的関心事の種類 (意味) とアーキテクチャ上の表現方法 (構文) の対応付けを行なっているものという認識から，フィーチャモデルから関心事を特定する際に GoF デザインパターンを利用する．携帯電話制御システムを事例とし，GoF デザインパターンに基づいてフィーチャと E-AoSAS++ アーキテクチャのコンポーネントの適切な対応付けが行なえるかを考察する．

2　背景技術

背景技術であるフィーチャモデル，デザインパターンについて説明する．

2.1　フィーチャモデル

フィーチャモデルとは，ユーザ要求を基に製品間の共通性と相違性の整理を目的とした仕様モデルである．フィーチャとは外部から認識できる製品の特徴または機能である．本研究で用いるフィーチャモデルの記述は FORM に従う．FORM はフィーチャを以下の 4 つのレイヤに分割する．

- Capability Layer
  ユーザの要求する機能，ユーザ要求を実現する製品の機能，非機能
- Operating Environment Layer
  ハードウェア，ソフトウェアの操作環境
- Domain Technology Layer
  特定のドメイン内で使用する技術
- Implementation Technique Layer
  複数のドメインで使用する一般的な技術

フィーチャは製品系列で必須 (Mandatory)，任意 (Optional)，選択 (Alternative) かに分類される．フィーチャ間の関係は Composed-of(全体-部分関係), Generalization/Specialization(汎化/特化関係)，Implemented-by(実現関係) の 3 種類で木構造を構成し表現する．

2.2　デザインパターン

デザインパターンとはソフトウェアの設計のパターンのカタログである．オブジェクト指向におけるデザインパターンとして，GoF デザインパターン [2] がよく知られている．GoF デザインパターンは目的別に生成，構造，振る舞いに関するパターンに分類される 23 種類のパターンを提案している．

3　E-AoSAS++

E-AoSAS++ とは，組込みシステムのためのアスペクト指向ソフトウェアアーキテクチャスタイルである．E-AoSAS++ はアーキテクチャを状態遷移機械 (以下，CSTM) の集合として定義している．CSTM がイベントを送信して協調動作を行うことにより，組込みシステムの機能を実現する．E-AoSAS++ ではシステムに横断する関心事をアスペクトとして抽出する．

3.1　E-AoSAS++ に基づくアーキテクチャ設計

以下に E-AoSAS++ に基づく開発体系で設計するアーキテクチャを示す．

- コンセプチュアルアーキテクチャ
  システムに横断する関心事を抽出したアーキテクチャ

- プロダクトラインアーキテクチャ
  製品系列全体の構成を表すアーキテクチャ
- プロダクトアーキテクチャ
  プロダクトラインアーキテクチャを基にプロダクトに必要なコンポーネントを抽出し，プロダクト毎に作成するアーキテクチャ

E-AoSAS++ に基づくアーキテクチャ記述は UML を用いる．E-AoSAS++ におけるアスペクトの表現はステレオタイプを用いて UML の意味を拡張して表す．

4 仕様モデルのコンポーネントが表す構文と意味からのアーキテクチャのコンポーネントとの対応付けの提案

本研究は仕様モデルのコンポーネントが表す構文と意味から，アーキテクチャのコンポーネントとの対応付けを提案する．仕様モデルのコンポーネントが表す関心事を，コンポーネントの構文と意味から特定する．特定した関心事がアーキテクチャではどのように実現されるかを整理することで，仕様モデルのコンポーネントとアーキテクチャのコンポーネントとの対応関係を明確にする．

4.1 フィーチャモデルと E-AoSAS++ アーキテクチャの対応関係

本研究はプロダクトラインフィーチャモデルと，E-AoSAS++ に基づくプロダクトラインアーキテクチャの対応関係を明確にする．

4.1.1 フィーチャと E-AoSAS++ アーキテクチャのコンポーネントの対応関係の仮説

プロダクトラインフィーチャモデルのフィーチャと E-AoSAS++ に基づくプロダクトライン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，次の仮説を立てる．

> プロダクトラインフィーチャモデルのフィーチャは，フィーチャが表す関心事を実現するプロダクトラインアーキテクチャのコンポーネントと対応する．

4.2 GoF デザインパターンを用いた仕様モデルのコンポーネントが表す関心事の特定

フィーチャとアーキテクチャのコンポーネントの対応関係を明確にするために，フィーチャが表す関心事を特定する．GoF デザインパターンはオブジェクト指向で発生する横断的関心事を解決するソフトウェアの設計のパターンである．本研究は，フィーチャが表す関心事を特定する際の基準として GoF デザインパターンを利用する．

4.2.1 GoF デザインパターンの整理

GoF デザインパターンが実現する関心事をフィーチャが表す構文と意味から特定するために，GoF デザインパターンの分類を行なう．

GoF デザインパターンが実現する関心事がフィーチャモデルではどのような構文で表されるのかを整理し，特定の構文で現れる GoF デザインパターンを 1 つのグループとして定義し分類を行う．表 1 に分類したグループの名前と，各グループが表すフィーチャの構文を示す．分類した結果，Adapter パターン，Command パターン，Composite パターンが実現する関心事はフィーチャモデルには現れないとわかった．

表 1　グループの名前とフィーチャの構文

グループの名前	フィーチャの構文
Composed	Capability Layer に配置され，全体/部分関係となるフィーチャ群
Generalization/ Specialization	Capability Layer に配置され，汎化/特化関係となるフィーチャ群
Primitive	Capability Layer に配置される個々のフィーチャ
Implemented	Capability Layer と Operating Environment Layer 間において実現関係にあるフィーチャ群
Strategy	Domain Technology Layer または Implementation Technique Layer に配置され，汎化/特化関係にあるフィーチャ群

GoF デザインパターンの分類と，各 GoF デザインパターンが実現する関心事はフィーチャもしくはフィーチャ群ではどのような意味で示されるのかを以下に示す．

■ Composed グループ

- Facade パターン：複数の子フィーチャの手続きによって親フィーチャの機能を実現している場合
- Observer パターン：子フィーチャ群が入力/演算/出力に分類されるいずれかの機能を表し，それらの協調動作によって親フィーチャの機能を実現している場合
- Mediator パターン：子フィーチャ群が表す機能同士が複雑に依存し，それらの協調動作によって親フィーチャの機能を実現している場合

■ Generalization/Specialization グループ

- Decorator パターン：親フィーチャが表す機能を機能拡張することによって，子フィーチャの機能が表されている場合
- Template Method パターン：子フィーチャ群が示す機能が共通の処理を持つ場合

■ Primitive グループ

- Proxy パターン：フィーチャの表す機能が実行効率，アクセス制御を表している場合
- Chain of Responsibility パターン：フィーチャの表す機能における処理が静的に決まらない場合
- Iterator パターン：フィーチャの機能がデータ構造へのアクセスを表している場合
- Interpreter パターン：フィーチャの機能が構文の解析，構文による処理を表している場合

- Flyweight パターン：フィーチャの表す機能がメモリ効率を求められている場合
- Memento パターン：フィーチャの表す機能が回復性を求められている場合
- State パターン：フィーチャの表す機能が状態に依存している場合
- Visitor パターン：フィーチャの表す機能がデータ構造に依存している場合
- Abstract Factory パターン，Builder パターン，Factory Method パターン，Singleton パターン，Prototype パターン：モード切替えに伴う制御対象の切替えを表現している場合

■ Implemented グループ

- Bridge パターン:親フィーチャが表す機能を, 子フィーチャであるハードウェア，ソフトウェアのフィーチャが実現している場合

■ Strategy グループ

- Strategy パターン：ある特定の機能を実現する親フィーチャの技術に対して複数の子フィーチャが表す技術が存在する場合

5 事例を用いた考察

携帯電話制御システムを事例として，フィーチャモデルのフィーチャとアーキテクチャのコンポーネントの対応付けと考察を行う．事例として用いる携帯電話制御システムの機能は通話機能，メール機能，カメラ機能とする．

5.1 プロダクトラインフィーチャモデル

図 1 に作成した携帯電話制御システムのプロダクトラインフィーチャモデルを示す．携帯電話制御システムに必須の機能を通話機能とメール機能とし，カメラ機能は任意の機能とした．非機能を実時間性，回復性，安全性とした．

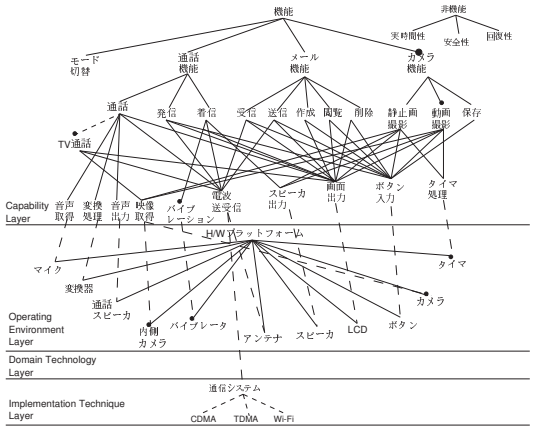

図 1　携帯電話制御システムのプロダクトラインフィーチャモデル

5.2 Composed グループに分類した関心事を表すフィーチャ群の対応関係の考察

Composed グループに分類した関心事は，E-AoSAS++ アーキテクチャではコンポーネントの配置，及びコンポーネント間の関係で実現される．

図 2 の左に示すフィーチャ群は構文と意味から Composed グループの Observer パターンが実現する関心事を表していることが特定できる．プロダクトラインフィーチャモデルに表れる Observer パターンが実現する関心事は，E-AoSAS++ アーキテクチャでは入力/演算/出力の機能を実現するコンポーネントと，それらの機能の協調動作によって実現される機能を担うコンポーネントとして実現される．子フィーチャ群は，子フィーチャが表す機能を実現するアスペクトとして設計する．親フィーチャは，それらのアスペクトをアスペクト間通信 (以下, IAD) を介して協調動作させることによって親フィーチャが表す機能を実現するコンポーネントとして設計する．

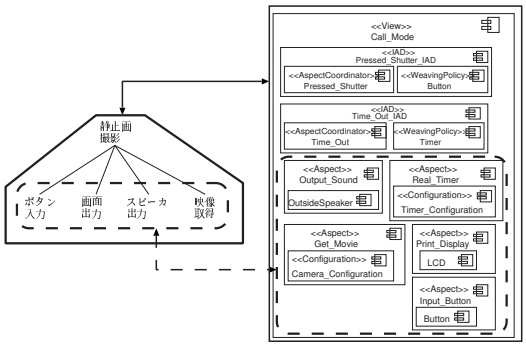

図 2　Observer パターンが実現する関心事を表すフィーチャ群とアーキテクチャの対応関係

5.3 Generalization/Specialization グループに分類した関心事を表すフィーチャ群の対応関係の考察

Generalization/Specialization グループに分類した関心事は，E-AoSAS++ アーキテクチャではコンポーネントの配置，及びコンポーネントの関係で実現される．

図 3 の上に示す通話のフィーチャ群は構文と意味から Generalization/Specialization グループの Decorator パターンが実現する関心事を表していることが特定できる．親フィーチャは，親フィーチャが表す機能を実現するコンポーネントとして設計し，子フィーチャは親フィーチャが表す機能に追加する機能を実現するアスペクトとして設計する．

5.4 Primitive グループに分類した関心事を表すフィーチャの対応関係の考察

Primitive グループに分類した関心事は，その関心事を表すフィーチャの機能を実現している E-AoSAS++ アーキテクチャのコンポーネントの構成要素，もしくはそのコンポーネントの実装工程で実現される．

図 1 に示す画面出力フィーチャは構文と意味から

Primitive グループに分類した Flyweight パターンが実現する関心事を表していることが特定できる．プロダクトラインフィーチャモデルが表す Flyweight パターンが実現する関心事は，フィーチャが表す機能を実現するアーキテクチャのコンポーネントの実装工程で実現される．

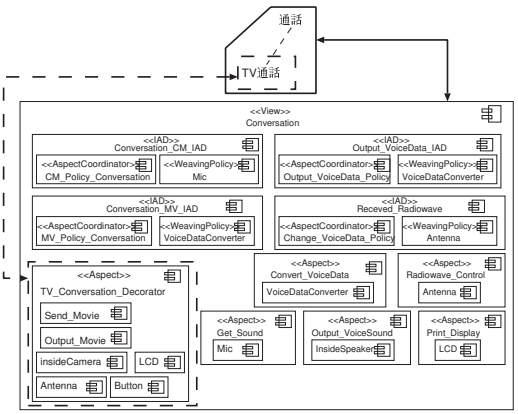

図 3　Decorator パターンが実現する関心事を表すフィーチャ群とアーキテクチャの対応関係

5.5　Implemented グループに分類した関心事を表すフィーチャ群の対応関係の考察

図 4 に示すボタン入力，及びボタンフィーチャは構文と意味から Implemented グループに分類した Bridge パターンが実現する関心事を表していることが特定できる．子フィーチャはハードウェアを制御する CSTM として設計する．親フィーチャは入力/演算/出力の機能を実現するアスペクトとして設計する．

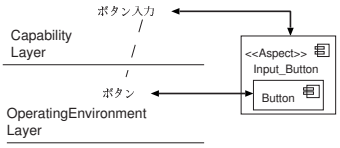

図 4　Bridge パターンが実現する関心事を表すフィーチャ群とアーキテクチャの対応関係

5.6　Strategy グループに分類した関心事を表すフィーチャ群の対応関係の考察

Strategy グループに分類した関心事は, Domain Technology，Implementation Technique の層に配置されるフィーチャ群で表される．技術層に配置されるフィーチャは実装工程で実現する技術を表していることから，Strategy グループに分類した関心事を表すフィーチャは実装工程で実現される．

図 1 に示す通信システム，CDMA，TDMA，Wi-Fi フィーチャは構文と意味から Strategy パターンが実現する関心事を表していることが特定できる．

5.7　提案した対応付けの一般性に関する考察

本研究で提案した対応付けの一般性について考察を行う．本研究は GoF デザインパターンを用い，フィーチャもしくはフィーチャ群が表す構文と意味から関心事を特定し E-AoSAS++ アーキテクチャのコンポーネントとの対応付けを行なった．GoF デザインパターンを用いることで，フィーチャモデル上の構文と意味から横断的関心事の特定が可能になる．横断的関心事を特定することにより，対応するアスペクト指向アーキテクチャのコンポーネントを明確にすることができる．このことから，本研究が提案する GoF デザインパターンを用いた対応関係は他のドメインにおいても適用できる可能性があると考える．今後，他のドメインに本研究が提案する対応付けを適用し，一般性を検証する必要がある．

6　おわりに

本研究では仕様モデルのコンポーネントの構文と意味から，アーキテクチャのコンポーネントとの対応関係を明確にする方法を提案した．GoF デザインパターンが実現する関心事をフィーチャの構文と意味から特定するために，GoF デザインパターンの整理を行なった．そして，携帯電話制御システムを事例とし，GoF デザインパターンが実現する関心事を表すフィーチャと E-AoSAS++ に基づくプロダクトラインアーキテクチャのコンポーネントの対応付け, 及び考察を行なった．今後の課題として，提案した対応付けの一般性の確認が挙げられる．

参考文献

[1] 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 Architechtures, ” *Annals of Software Engineering*, vol.5, no.1, pp.143-168, 1998.

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

[3] K.Pohl, G.Bockle, and F.Linden, *Software Product Line Enginnering: Foundations, Principles and Techniques* , Springer, 2005.

[4] 加藤大地，蜂巣吉成，沢田篤史，野呂昌満，“アスペクト指向に基づくソフトウェアアーキテクチャの文書化方式，” 知能ソフトウェア工学研究会 *(KBSE)*, vol.108, no.449, pp55-60, 2009.

[5] 中西恒夫, 久住憲嗣, 福田晃, “ソフトウェアアーキテクチャ事前設計を目的とするフィーチャモデルのガイドラインとアンチパターン，” 信学技報，vol.109, no.170, pp77-82, 2009.

[6] 西山遼平，“アスペクト指向ソフトウェアアーキテクチャに基づく PLSE に関する研究，” 南山大学大学院数理情報研究科 修士論文要旨集，2008.