組込みソフトウェア開発環境に関する研究
– ソフトウェアアーキテクチャの設計支援 –

2008MI075 伊藤 和輝　　2008MI082 岩田 繁　　2008MI213 澤田 康太
指導教員　野呂 昌満

1　はじめに

近年，ソフトウェア開発方法論である Product Line Software Engineering(以下，PLSE)[4] が注目されている．PLSE では，製品系列の開発において共通の部品をコア資産として開発し再利用することで，生産性の向上が期待できる．PLSE におけるアプリケーション開発は，アーキテクチャを特定しなければ開始することができないので，仕様モデルとアーキテクチャの追跡性を確保することが重要である．組込みシステムはユーザ要求の複雑化に伴い多様化しているので，コア資産を用いる PLSE に基づく開発が適している．組込みシステムには非機能特性が複雑に絡んでおり，横断的関心事として散在している．この問題を解決する技術として，アスペクト指向技術がある．システムに散在する横断的関心事をアスペクトとして抽出することでモジュールの独立性が保証される．非機能特性とモジュールの関係が単純化され，追跡性が保証される．PLSE の開発体系にアスペクト指向技術を適用することで，コア資産の独立性と追跡性が保証され，再利用性を高めることができる．

PLSE のドメイン開発において，仕様モデルからアーキテクチャへの追跡性が定義されていない．追跡性が定義されていない場合，仕様からアーキテクチャを設計する際に技術者の経験や知識に依存する．結果として PLSE の目的である生産性の向上が実現できないので，仕様モデルからアーキテクチャへの追跡性を定義しなければならない．

本研究の目的は，仕様モデルとアスペクト指向アーキテクチャの対応関係をアーキテクチャ設計のためのガイドラインとして提案することである．ガイドラインに従って仕様モデルからアーキテクチャを設計する手順を定義することにより，仕様モデルからアーキテクチャへの追跡性を確保し，アーキテクチャ設計の際の省力化を目指す．図 1 に，定義する手順の概要を示す．

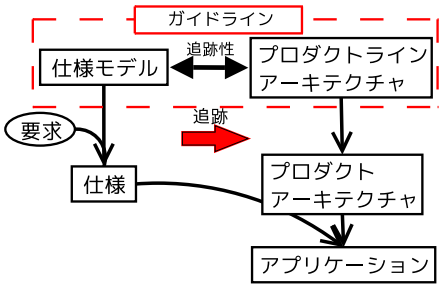

図 1　定義する手順の概要

本研究では，非機能特性はアスペクトコードになる可能性があるという認識の基に，非機能特性に着目し対応関係を明確にした．組込みシステムに必要な非機能特性を ISO9126 の品質特性 [1] を基に整理し，整理した非機能特性と関心事の関係を整理した．仕様モデルにおける非機能特性を表すコンポーネントを明示的に表現するために，仕様モデルの定義方法を拡張した．仕様モデルにおける可変部分をアーキテクチャレベルで表現するために，プロダクトラインアーキテクチャを提案した．仕様モデルとアスペクト指向アーキテクチャの非機能特性を表すコンポーネントの対応関係を提案した．二つの事例を用いてフィーチャモデルとプロダクトラインアーキテクチャを作成し，対応関係を考察した．

仕様モデルには，PLSE で代表的に用いられている Feature-Oriented Reuse Method(以下，FORM)[3] に基づくフィーチャモデルを用いる．アーキテクチャには，本研究室が提案している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++)[6] に基づくアーキテクチャを用いる．

本研究の成果として，仕様モデルとアーキテクチャの対応関係をアーキテクチャ設計のためのガイドラインとして提案することで，追跡性の確保ができた．

2　背景技術・先行研究

本章では，本研究の背景技術であるフィーチャモデルと, 先行研究である Can Aspects Model Product Lines? 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，PLSE の開発で代表的に用いられている仕様モデルである．フィーチャモデルは，構成要素であるフィーチャを木構造で表し，製品間における共通部分と可変部分の情報の整理を目的とする．フィーチャとは，ステークホルダから認識可能な機能や品質，特徴であり，プロダクトラインの製品間で必須 (Mandatory)，任意 (Optional)，選択 (Alternative) の三つの種類に分類される．FORM に基づくフィーチャモデルは，フィーチャの役割を明示的に表現するために，四つの層に分離し記述する．

2.2　Can Aspects Model Product Lines ?

Oldevik[2] は，PLSE の開発工程において，アスペクトがアーキテクチャ上のフィーチャの表現や，横断的な構造や振舞いをモジュール化するために有用であることを示している．フィーチャと，アスペクトによりモジュール化されたアーキテクチャの対応関係を定義しておくことで，フィーチャモデルの構成に従ってアーキテ

クチャを容易に構築することができる．しかし着目対象が機能のみなので，本研究では非機能特性を対象とし同様の対応付けが行なえるか検証する．

3 E-AoSAS++

E-AoSAS++ は本研究室で提案している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++ では，システムを並行に動作する状態遷移機械 (以下，CSTM) の集合として定義している．CSTM は受信したイベントに応じて状態を遷移し，状態遷移時にアクションとして CSTM の処理を実行する．状態遷移時に他の CSTM にイベントを送信する．この各 CSTM の協調動作によって，組込みシステムの機能を実現する．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現する．

4 仕様モデルとアーキテクチャの対応関係の提案

本章では，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の対応関係を示し，アーキテクチャ設計のためのガイドラインとして提案する．

4.1 仕様モデルとアーキテクチャの対応関係の仮説

本研究ではプロダクトラインフィーチャモデルのフィーチャと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関係について，以下の仮説を立てた．

> プロダクトラインフィーチャモデルの非機能特性を表すコンポーネントは，その非機能特性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．

これは仕様モデルに表れる非機能特性はアーキテクチャにも反映されることを示す．

4.2 非機能特性と関心事の関係の整理

非機能特性を表すコンポーネントの雛型を構築するために，組込みシステムに必要な非機能特性と関連する情報を整理する．本研究では，非機能特性は関心事の実現によって満たされると考える．組込みシステムを対象としているので，関心事は専用のコンポーネントを用いて実現されると考える．

非機能特性に関連する情報は以下の三つと考えた．

- 非機能特性を満たすために実現すべき関心事
- 関心事の実現手法
- 関心事の実現に用いるコンポーネント

非機能特性と関心事の実現手法の関係を調査し，表に整理する．

4.2.1 組込みソフトウェアに必要な非機能特性の整理

非機能特性と関心事の関係を整理するにあたり，組込みシステムに必要な非機能特性を整理する必要がある．ISO9126 の品質特性はソフトウェア品質の評価に関する国際基準である．ISO9126 を基に組込みシステムが機能する上で必要な非機能特性に着目し，整理した．

今回は以下の五つの非機能特性を対象する．

- セキュリティ (Security)
- 障害許容性 (Fault Tolerance)
- 時間効率性 (Time Behaviour)
- 資源効率性 (Resource Behaviour)
- 解析性 (Analyzability)

4.2.2 非機能特性と関心事の実現手法の関係

非機能特性と関心事の実現手法の関係を表に整理する．表 1 に非機能特性と関心事の実現手法と関心事の実現に用いるコンポーネントの一例を整理した表を示す．

表 1　非機能特性と関心事の実現手法の関係

非機能特性	関心事	実現手法	コンポーネント
セキュリティ	アクセス制御	認証	認証機器
	アクセス解析	アクセス解析ロギング	データロガー
	データ機密性保持	暗号化	暗号化機器
障害許容性	障害検知	例外処理	障害検知センサ
	耐故障処理	冗長化	代替機器
時間効率性	実時間処理	レスポンスタイム	タイマー
		ポーリング間隔時間	
解析性	障害解析	障害解析ロギング	データロガー
資源効率性	メモリ管理	参照回数	メモリ管理ユニット

4.3 仕様モデルの定義方法の拡張

FORM におけるフィーチャモデルを用いた仕様モデルの定義方法の問題点を明らかにし，仕様モデルの定義方法を拡張する．FORM における仕様モデルの定義方法には以下の二つの問題点がある．

- 非機能フィーチャがどの機能フィーチャと関連を持っているかが不明確
- どのような関心事が非機能特性を実現しているか不明確

上記の問題点を解決するために仕様モデルの定義方法の拡張を行なう．Boskovic ら [5] はシステムを複数の関心事に分割し，関心事毎にフィーチャダイアグラムを作成している．本研究では，この考え方を基に本来の FORM に基づくフィーチャダイアグラムとは別に非機能特性を実現する関心事毎にフィーチャダイアグラムを作成する．関心事毎に作成するフィーチャダイアグラムでは関心事フィーチャと機能フィーチャを明確に分離し，その間の関係を定義する．図 2 に関心事毎に作成するフィーチャダイアグラムの記述例を示す．

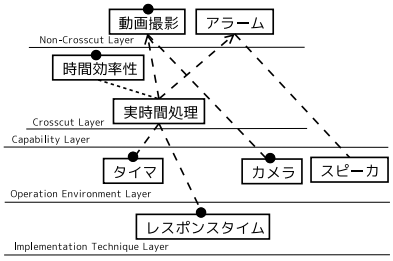

図 2　実時間処理に着目したフィーチャダイアグラムの例

4.4 プロダクトラインアーキテクチャの提案

本研究では E-AoSAS++ に基づくプロダクトラインアーキテクチャを提案する．プロダクトラインアーキテクチャとは，製品系列全体の構成を表したアーキテクチャである．プロダクトラインアーキテクチャでは，フィーチャモデルにおける可変部分を表現する必要がある．可変部分を表現するにあたって，アーキテクチャレベルで任意フィーチャと選択フィーチャを表現する方法を提案した．図 3 に，アーキテクチャレベルで任意フィーチャを表現する方法を示す．

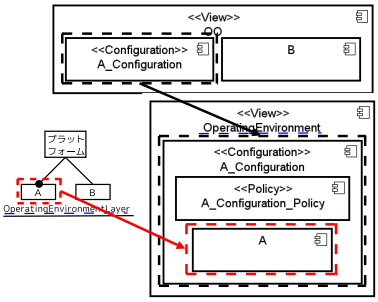

図 3　任意フィーチャの表現方法

可変部分はアーキテクチャ上でステレオタイプ<<Configuration>>が付加された Configuration コンポーネント内に配置される．可変部分の選択は，同じコンポーネント内のステレオタイプ<<Policy>>が付加された Policy コンポーネントにより行なわれる．

4.5 フィーチャモデルと E-AoSAS++ アーキテクチャの対応関係

拡張した仕様モデルの定義方法における関心事に着目したフィーチャダイアグラムのコンポーネントと，アーキテクチャのコンポーネントの対応関係を提案する．図 5 に，提案する対応関係の一例を示す．この対応関係について順に説明する．

関心事と実現手法の関係の対応関係

関心事と実現手法の関係は，その関心事をどのように実現するかを表す．AspectCoordinator はアーキテクチャ上でどのような実現手法により関心事を実現するかを表す．したがってフィーチャダイアグラムにおける実現手法とアーキテクチャにおける AspectCoordinator が対応する．図 5 の場合，フィーチャダイアグラムの「レスポンスタイム」とアーキテクチャの「Response-Time_AC」が対応する．

アスペクトの織込みに関する対応関係

フィーチャダイアグラムでは，破線矢印により関心事がどの機能フィーチャに横断するかを示している．したがってフィーチャダイアグラムにおける関心事が横断する機能フィーチャと実現関係にある Operating Environment Layer のフィーチャと，アーキテクチャにおけるアスペクトの織込み先が対応する．図 5 の場合，フィーチャダイアグラムの「スピーカ」とアーキテクチャの「Speaker」などが対応する．

関心事を実現するコンポーネントに関する対応関係

関心事を実現関係にある Operating Environment Layer のフィーチャは，関心事の実現手法に用いるコンポーネントを表す．したがってそのフィーチャとアーキテクチャにおけるアスペクトを実現するコンポーネントが対応する．図 5 の場合，フィーチャダイアグラムの「タイマ」とアーキテクチャの「Timer」が対応する．

5 事例検証

本章では，事例を用いてフィーチャモデルを作成し，4 章で提案した対応関係を基に，プロダクトラインアーキテクチャを設計する．本研究では，携帯電話制御システムとプリンタ制御システムの二つの事例を用いる．ここでは携帯電話制御システムの事例について記述する．携帯電話制御システムにおける非機能特性を満たす関心事を耐故障処理，メモリ管理，実時間処理の 3 つとした．対応する実現手法，関心事を実現するコンポーネントは表 1 を参考にし，本研究で提案した仕様モデルの定義方法に基づいて携帯電話制御システムのフィーチャモデルを作成した．図 4 に携帯電話制御システムのフィーチャダイアグラムを示す．

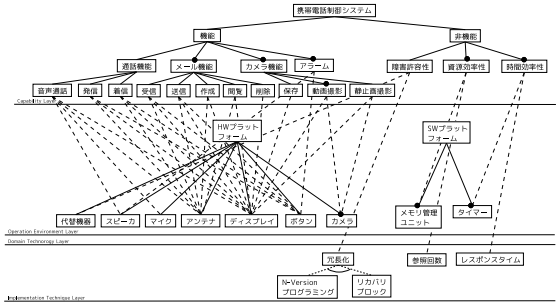

図 4　携帯電話制御システムのフィーチャダイアグラム

図 2 は事例における実時間処理に着目したフィーチャダイアグラムである．図 2 のフィーチャダイアグラムのコンポーネントから 4 章で提案した対応関係を基に，プロダクトラインアーキテクチャのコンポーネントを設計した．事例における時間効率性を表すフィーチャ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の対応関係を図 5 に示す．

設計したプロダクトラインアーキテクチャを基にプロダクトアーキテクチャを構成し，E-AoSAS++ に基づく一連の開発プロセスを行なった．構成したプロダクトアーキテクチャを基にアプリケーション設計を行ない，設計したアプリケーションアーキテクチャの欠陥を取り除くために実行前検査を行なった．完成したアプリケーションアーキテクチャを基に，コード自動生成ツールを用いてプログラムコードを自動生成することができた．

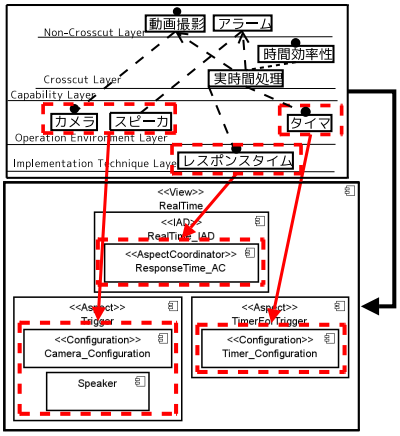

図 5　実時間処理における対応関係

6　考察

整理した非機能特性と関心事の関係と，提案したプロダクトラインアーキテクチャと，仕様モデルとアーキテクチャの対応関係の妥当性について考察する．

6.1　整理した非機能特性と関心事の関係の妥当性

非機能特性を表すコンポーネントの雛型に必要な情報に，非機能特性を満たす関心事，関心事の実現手法，コンポーネントが必要であると考えた．ISO9126 に基づいた非機能特性を用いて，情報を整理することで，非機能特性を表すコンポーネントの雛型を構築できた．事例検証の結果，整理した非機能特性と関心事の関係を基に，携帯電話制御システムとプリンタ制御システムにおいて，非機能特性を表すコンポーネントの雛型から非機能特性を実現するコンポーネントを構築することができたことから，妥当性の確認ができた．

6.2　プロダクトラインアーキテクチャの妥当性

二つの事例を用いてプロダクトラインフィーチャモデルからプロダクトラインアーキテクチャを設計した．プロダクトフィーチャモデルの構成に従って，プロダクトラインアーキテクチャの可変部分を決定することでプロダクトアーキテクチャの基本構造を構成することができた．構成されたプロダクトアーキテクチャを基に，E-AoSAS++ に基づく一連のソフトウェア開発プロセスを行なうことができることが確認できた．以上のことからプロダクトラインアーキテクチャから構成されたプロダクトアーキテクチャの妥当性が確認でき，プロダクトラインアーキテクチャの妥当性が確認できた．

6.3　仕様モデルとアーキテクチャの対応関係の妥当性

図 5 から，提案した対応関係に基づいてフィーチャモデルの時間効率性を表すコンポーネントから，時間効率性を実現するアーキテクチャのコンポーネントを設計することができた．携帯電話制御システム，プリンタ制御システムで，障害許容性，資源効率性などの非機能特性についても同様にアーキテクチャのコンポーネントを設計することができた．このことから対象とする非機能特性や製品系列に依存せず，提案した対応関係に基づいて仕様モデルにおける非機能特性から系統的にアーキテクチャを設計できることが確認でき，提案した対応関係の妥当性の確認ができた．以上のことから，4.1 章で立てた仮説が正しいことがいくつかの例で確認できた．したがってプロダクトラインフィーチャモデルの非機能特性を表すコンポーネントと，その非機能特性を実現する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ことができた．

7　おわりに

本研究では仕様モデルとアスペクト指向アーキテクチャの非機能特性を表すコンポーネントの対応関係をアーキテクチャ設計のためのガイドラインとして提案した．非機能特性を表すコンポーネントを明示的に表現できるように仕様モデルの定義方法を拡張した．E-AoSAS++ に基づくプロダクトラインアーキテクチャを提案し，拡張したフィーチャモデルとの対応関係を明確にした．事例を用いてプロダクトラインフィーチャモデルとプロダクトラインアーキテクチャを作成し，対応関係について考察を行なった．本研究の成果として，仕様モデルとアスペクト指向アーキテクチャの対応関係を明確にしアーキテクチャ設計のためのガイドラインとして提案することで，追跡性の確保ができた．今後の課題として，本研究で考慮した非機能特性以外についても同様な対応関係が表れるか確認する必要がある．

参考文献

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

[2] J. Oldevik, “ Can aspects model product lines?, ” *Internal Conference on Aspect-Oriented Software Development*, vol. 2, pp. 1-8, 2008.

[3] 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.

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

[5] M. Boskovic, G. Mussbacher, E. Bagheri, D. Amyot, D. Gasevic, and M. Hatala, “ Aspect-Oriented Feature Models, ” *Models in Software Engineering*, vol. 6627, pp. 110-124, 2011.

[6] M. Noro, A. Sawada, Y. Hachisu, and M. Banno, “ E-AoSAS++ and its Software Development Environment, ” *Proceedings of the 14th Asia-Pacific Software Engineering Conference(APSEC2007)*, pp. 206-213, 2007.