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
− 自動販売機制御ソフトウェアを例にして−

2004MT030 池内 伸彰 2004MT102 鈴木 健史
指導教員　野呂 昌満

1　はじめに

ソフトウェア開発におけるユーザ要求の多様化にともない, 機能の追加や変更が頻繁に行われ, ソフトウェアの複雑化と大規模化が進んでいる. 大規模化ソフトウェアの系統的な開発手法の一つとして, プロダクトラインソフトウェアエンジニアリング (以下, PLSE[2]) が提案されている. PLSE ではアーキテクチャをコア資産として再利用することができる. 多様化するユーザ要求を整理する手法としてフィーチャモデリングがフィーチャ指向プロダクトラインエンジニアリング (以下, FOPLE[1]) で提案されている.

本研究室では高い再利用性を保証する技術である組込みソフトウェア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++[3]) が提案されている. E-AoSAS++ を用いた開発に PLSE を適用することでアーキテクチャをコア資産とした開発を行うことができる. ユーザ要求の変化にともなうアーキテクチャの構造の変更を明確化できれば開発のコストを削減することができる.

PLSE にはユーザ要求からアーキテクチャを系統的に構築する手法が確立されていないという問題がある. またフィーチャ図を用いることでユーザ要求を整理することができるが, フィーチャを記述する基準が不明確であるという問題もある.

これらの問題を解決するために本研究では既存のフィーチャ図を再定義し, ユーザ要求とアーキテクチャの対応関係を明確にする. フィーチャ図とアーキテクチャの対応関係に仮説をたて, 自動販売機制御ソフトウェアを例に対応関係の検証をおこなう. また他のドメインの例題を用いることで対応関係の一般性について考察する.

2　研究の背景

2.1　PLSE

PLSE はソフトウェアを系統的に開発するためのソフトウェア開発手法である. PLSE ではコア資産を開発し, 抽出された部品の組合せによって製品開発を行う. コア資産とされるものには再利用可能な部品やアーキテクチャなどが挙げられる. PLSE は 3 つの活動からなっている.

- ドメイン工学
  -要求分析, コア資産開発
- アプリケーション工学
  -コア資産から製品を開発
- 管理
  -コア資産と開発プロセスの管理

2.2　E-AoSAS++

E-AoSAS++ は, 組込みソフトウェアのためのアーキテクチャスタイルであり, 組込みソフトウェアを並行状態遷移機械（以下, CSTM）の集合として記述する. 複数の CSTM は, 互いにメッセージを送り合うことで連動動作する. E-AoSAS++ ではアスペクトをコンポーネントとして扱う.

E-AoSAS++ に基づくアーキテクチャにはコンセプチュアルアーキテクチャ, プロダクトラインアーキテクチャ, プロダクトアーキテクチャがある.

プロダクトラインアーキテクチャ, プロダクトアーキテクチャは E-AoSAS++ で提案されているユニバーサルモジュール化パターン (以下, UMP) を用いて表される.

UMP

UMP は複数の CSTM による協調動作をポリシーをもちいてモジュール化する手法である. UMP 記述の例を図 1 に示す.

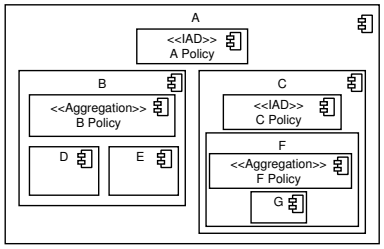

図 1　UMP 記述の例

コンセプチュアルアーキテクチャ

コンセプチュアルアーキテクチャは要求とシナリオから自然に認識することのできるハードウェア部品の関連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

プロダクトラインアーキテクチャ (以下 PLA) はソフトウェアファミリ全体を表したアーキテクチャである. PLA は UMP の記述法とコンポーネント図を用いて記述する.

プロダクトアーキテクチャ

プロダクトアーキテクチャ (以下 PA) は, ソフトウェアを開発する際に,PLA のアーキテクチャから実際のプロダクトを作成するために必要なコンポーネントを抽出し

て作成されるアーキテクチャである.

2.3 FOPLE

FOPLE では主に資産開発と商品開発の 2 つを主なプロセスとするものである. FOPLE ではユーザ要求を整理するフィーチャモデリングが定義されている. 次に FOPLE で定義されているフィーチャ, サブフィーチャ, フィーチャ間の関係, フィーチャ図の 4 階層, について次に示す.

- フィーチャ
  開発者の視点から実現可能なユーザ要求であり、製品の特徴、特性を表している.
- サブフィーチャ
  フィーチャを詳細化したフィーチャ群である. フィーチャとサブフィーチャは親と子の関係である.

フィーチャ間の関係を図 2 に示す. フィーチャ図は

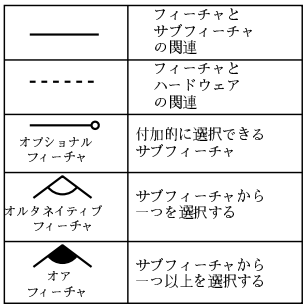

図 2 フィーチャ間の関係

フィーチャを各階層に整理した図である. 次にフィーチャ図の 4 階層について述べる.

- 特性層 (Capability Layer)
  機能特性と非機能特性
- 操作環境層 (Operating Environment Layer)
  ハードウェアに関する部分
- ドメイン技術層 (Domein Technology Layer)
  ドメインに特化した技術
- 実現技術層 (ImplementationTechniqueLayer)
  ソフトウェアの実現に利用される一般的な技術

3 フィーチャ図の再定義

フィーチャ図とソフトウェアアーキテクチャの対応関係について考えるにあたり, FOPLE で記述されるフィーチャ図の再定義をおこなう.

3.1 FOPLE におけるフィーチャ図の問題点

特性層に現れるフィーチャは機能特性と非機能特性とされているだけであり, どこまで詳細に記述するかが不明確である. 使用性や安全性などの非機能フィーチャは抽象度が高く, 他のフィーチャとの関連が不明確である.

3.2 問題点の解決策

特性層の整理

既存のフィーチャ図の特性層を分割し, 特性層とアトミック特性層とする. 特性層には機能要求の分析から得られるユースケースをフィーチャとして記述する, またアトミック特性層には特性層に現れるフィーチャの実現に必要な動作をフィーチャとして記述する. これにより既存の特性層に記述するフィーチャを明確にすることができる.

- 特性層に現れるフィーチャ
  要求整理の際にユースケース図を作成し, ユースケース図に現れるユースケースを特性層のフィーチャとする.
- アトミック特性層に現れるフィーチャ
  アトミック特性層に現れるフィーチャは特性層に現れるフィーチャに必要となるシステムの動作を記述する.

非機能フィーチャの記述

非機能フィーチャは特性層に記述する. 抽象度に関しては非機能を実現するのに必要なコンポーネントがアーキテクチャ上のどのコンポーネントと関連するかがわかるまで詳細化することとする. また非機能を実現するハードウェア, 技術があれば非機能と関連させ操作環境層もしくはドメイン技術層, 実現技術層に記述する.

操作環境層, ドメイン技術層, 実現技術層に現れるフィーチャに関しては FOPLE と同じである.

4 自動販売機制御ソフトウェアの分析

フィーチャ図とアーキテクチャとの対応関係を考えるにあたり, 自動販売機制御ソフトウェアの機能, 非機能, 構成部品について分析をおこなった. 機能分析としてユースケース図を作成する. ユースケースから考えることができる非機能として実時間性, 操作性, 耐故障性が挙げられる. 非機能の詳細化をおこない, 実時間性から購入時間制限が得られた. 構成部品の分析から自動販売機に必要な部品を得た. 分析を基に作成した自動販売機制御ソフトウェアのフィーチャ図を図 3 に示す.

5 フィーチャ図とアーキテクチャの対応関係

5.1 仮説

フィーチャ図と E-AoSAS++ に基づくアーキテクチャとの対応関係を自動販売機制御ソフトウェアを例にして考える. E-AoSAS++ ではアスペクトをモジュールとして取り扱うので, フィーチャ図からアスペクトとして取り扱うものが明確にできればアーキテクチャ上でコンポーネントとすべきものが明確にできる. この基本方針を反映し, 次の仮説を設定する.

- 特性層, アトミック特性層に現れるフィーチャはすべてアスペクトとして取り扱う.

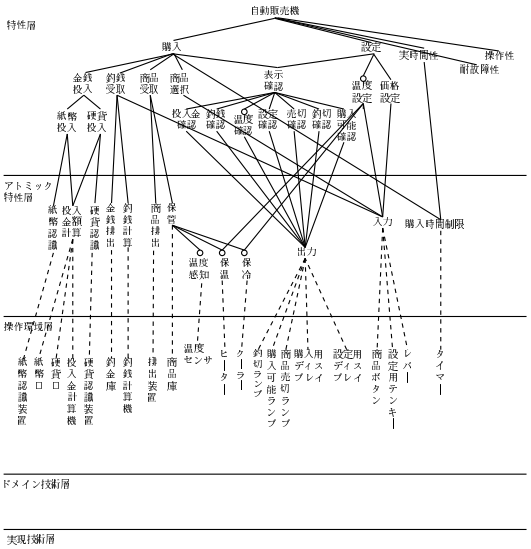

図 3　自動販売機制御ソフトウェアのフィーチャ図

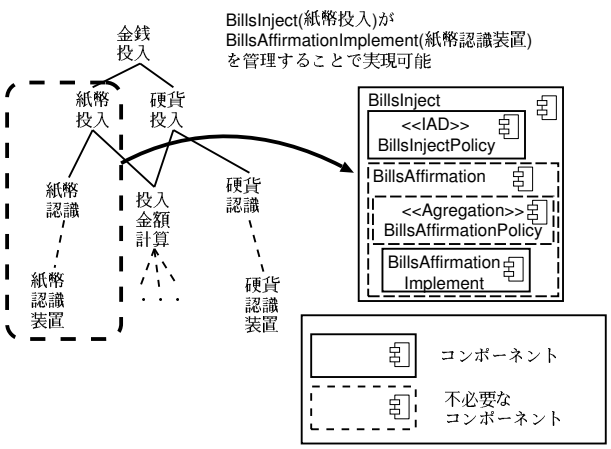

図 4　コンポーネントとする必要の無いフィーチャ

5.2　仮説に基づく対応関係

仮説に基づいてフィーチャ図の 5 階層とアーキテクチャの対応関係を述べる.

特性層, アトミック特性層

特性層, アトミック特性層に現れるフィーチャは仮説からアスペクトとして扱うので一つのフィーチャが一つのコンポーネントとして現れる.

操作環境層

操作環境層に現れるフィーチャは製品を構成する部品なのでハードウェアコンポーネントとしてアーキテクチャ上に現れる.

ドメイン技術層, 実現技術層

ドメイン技術層, 実現技術層に現れるフィーチャは機能フィーチャを実現する技術なのでコンポーネントとしてアーキテクチャ上には現れない.

5.3　仮説から得られた対応関係の考察

モジュール化されないフィーチャ

仮説から特性層, アトミック特性層のフィーチャを全てコンポーネントと対応させたが, フィーチャ図上からアーキテクチャと対応させる必要の無いフィーチャを発見した.

金銭投入に関するフィーチャをコンポーネントと対応させた図を図 4 に示す. 紙幣認識ポリシーが紙幣認識装置コンポーネントの切替えをおこなっているが, 紙幣投入ポリシーが紙幣認識装置コンポーネントの切替えをおこなうことで, 必要とされる機能は実現することができる. したがって, 紙幣認識フィーチャをコンポーネントとする必要は無い.

このように必須なサブフィーチャを一つだけもつフィーチャはアーキテクチャ上にコンポーネントとして現す必要はない.

非機能フィーチャの対応

図 3 に現れる実時間性や耐故障性などの非機能フィーチャは抽象度が高く様々な機能と関連するので, より詳細な非機能フィーチャとしてを記述する必要がある. 非機能フィーチャを詳細化することで非機能フィーチャと関連する機能フィーチャが明確になり, 関連する機能の処理の変更によって非機能を実現することができるので, 非機能フィーチャをコンポーネントとする必要は無い. また図 3 に現れるタイマーのように詳細化した非機能フィーチャを実現するハードウェアが存在する場合は機能と関連させることで必要な非機能要求を満たすことができる.

オア, オルタネイティブ, オプショナルフィーチャの対応

フィーチャ図に現れるオア, オルタネイティブ, オプショナルといったフィーチャの選択を PLA 上に表記することで PA を構築する際にコンポーネントの選択を容易に行うことができる. 選択を現すフィーチャは親フィーチャと対応するコンポーネントのポリシーにステレオタイプ`<<SpecificationModel>>` を付加することでこのポリシーがアーキテクチャの選択を兼ねることを表す. またポリシーに選択パターンを記述したノートを付けることで選択のパターンをアーキテクチャ上に表現する. フィーチャが選択された場合, その選択に合わせたノートのパターンを選択することでプロダクトアーキテクチャを構築することができる. これらの対応を図 5 に示す.

6　考察

自動販売機を例として提案した対応関係を, プリンタのドメインと比較することで一般性についての考察をおこなう.

6.1　特性層, アトミック特性層の考察

プリンタのフィーチャ図でも提案した対応関係と同様のことが言える. 図 6 に示す印刷データ受信, 送紙フィー

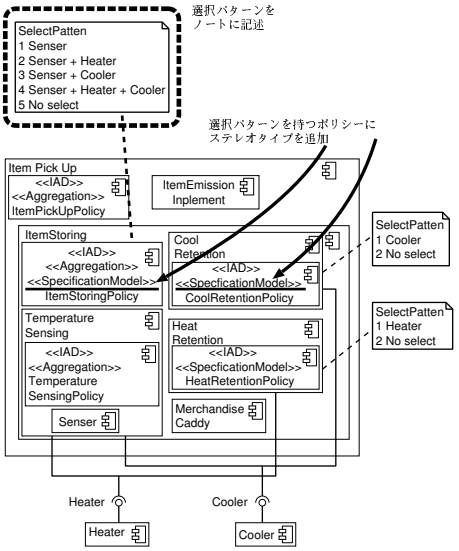

図 5　アーキテクチャ上への選択フィーチャの記述

チャは提案で示したようにコンポーネントとする必要は無い. よって特性層, アトミック特性層における提案した対応関係は他のドメインでも対応することができる.

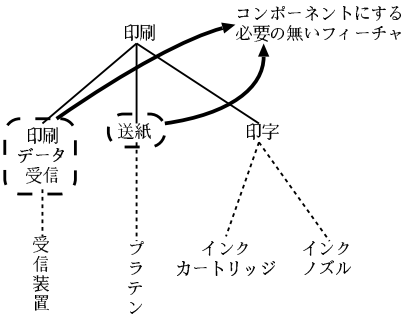

図 6　印刷に関するフィーチャ

6.2　操作環境層の考察

プリンタの操作環境層に現れるフィーチャも自動販売機と同様にドメインを構成する部品が現れるので, すべてハードウェアコンポーネントとしてアーキテクチャ上に現すことができる.

6.3　ドメイン技術層, 実現技術層の考察

プリンタでは受信装置のサブフィーチャとして USB と LAN のフィーチャが実現技術層に現れている. これらのフィーチャは受信装置の規格でありコンポーネントとして記述する必要は無い.

6.4　非機能フィーチャとアーキテクチャの対応の考察

プリンタに現れる非機能フィーチャの中にコンポーネントとすべきフィーチャを見付けることができた. プリンタのフィーチャ図に現れるインクヘッドクリーニングフィーチャでは, ヘッドクリーニング装置とヘッド監視装置により印刷品質維持という非機能フィーチャが実現される. 5 節で提案した対応関係では, 特性層, アトミック特性層に現れる非機能フィーチャはコンポーネントにする必要は無いとしたが, このような非機能を実現する機能があればコンポーネントとする必要がある. しかしどのフィーチャが非機能を実現する機能フィーチャであるかを判断するのは困難であり, 非機能フィーチャとアーキテクチャの対応関係については開発者の経験から判断するべきであると思われる.

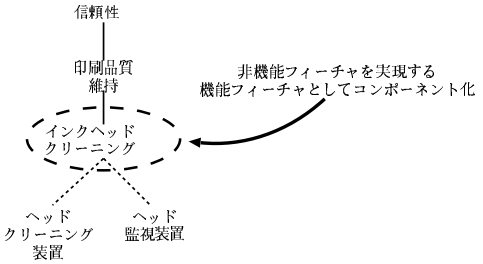

図 7　プリンタを例にした信頼性に関する非機能フィーチャ

7　おわりに

本研究では FOPLE のフィーチャ図を再定義, フィーチャ図とアーキテクチャの対応関係に仮説を立てた. 仮説を基にフィーチャ図の各層とアーキテクチャの対応関係を考え, 自動販売機制御ソフトウェアを例にして対応関係の提案をおこなった. 提案した対応関係を他のドメインに適用し考察することでフィーチャ図とアーキテクチャの対応関係に一般性を持たせることができた. 今後の課題としてより多くのドメインをもちいて対応関係をについて考察することとフィーチャを選択した際にアーキテクチャの構造変化を自動化するツールの作成が挙げられる.

謝辞

本研究を進めるにあたり, 二年間熱心な御指導いただいた野呂昌満教授, 有益なアドバイスを下さった沢田篤史教授, 蜂巣吉成講師, 大学院生の皆様に深く感謝いたします. また, ニ年間ともにがんばってきた研究室のみなさんにも感謝いたします.

参考文献

[1] Kyo C. Kang, Jaejoon Lee, Patrick Donohoe, “Feature-Oriented Product Line Engineering”, IEEE Software, Vol.19, No.4, pp.58-65, 2002

[2] Linda M. Northrop, “SEI’s Software Product Line Tenets”, IEEE Software, Vol.19, No.4, pp.32-40, 2002

[3] Masami Noro, Atsushi Sawada, Yoshinari Hachisu, Masahide Banno, “E-AoSAS++ and its Software Development Environment”, Proceedings of the 14th Asia-Pacific Software Engineering Conference(APSEC2007), pp.206-213, Dec. 2007