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

M2006MM022 西山 遼平
指導教員　野呂 昌満

1　はじめに

部品化, 再利用などのソフトウェア工学の課題に対する, ソフトウェアアーキテクチャを中心とした解決策として, 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) が提案されている [1]. PLSE は, 共通の特徴を持つソフトウェアファミリに対し, 部品開発をおこない, 部品を基に製品であるソフトウェアを開発する. 部品とは, プログラムコード, ソフトウェアアーキテクチャおよび仕様などであり, これらを再利用することで, 開発期間の短縮やコストの削減がおこなえる.

PLSE の問題点として, 仕様モデルとソフトウェアアーキテクチャの関連が明確でないことが挙げられる. 仕様を整理して, 仕様とソフトウェアアーキテクチャの関連が明確になれば, 仕様が異なる製品の開発においてもアーキテクチャを再構築することが容易になる. しかし, 既存の開発方法論では, その関連について十分に考察されていない.

また, 本研究室では, 組込みソフトウェアにおける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] の提案をしている. E-AoSAS++ は, 組込みソフトウェアを並行に動作する状態遷移機械 (以下,CSTM) の集合として規定している.E-AoSAS++ は, 適切なモジュール化, 統一的な記述を規定する事により, 粒度の高い再利用性の保証を目指している. E-AoSAS++ に基づいた開発プロセスは, PLSE の概念に基づき, アーキテクチャをコア資産として再利用することを目的としている.

本研究の目的は, 仕様モデルとアーキテクチャの関連を明確にし, 対応付けることである. 既存の仕様モデルでは, 記述する仕様が曖昧であるという問題点があるので, その解決法を示す. 本研究では, 対象ドメインを組込みソフトウェアとし, 仕様モデルと E-AoSAS++ に基づくアーキテクチャとの関連を考察し, 自動的に対応できる部分を明らかにする. 対応関係について仮説をたて, 複数の組込みソフトウェアを事例として検証し, 対応関係の検証と一般化に関する考察をおこなう. 今回は, 組込みソフトウェアをドメインとしたが, 対応関係はドメインの特性に依存していないと考える. 本研究の成果として得られた対応関係を用いることで, 仕様に応じて自動的にアーキテクチャを決定することが出来る.

2　関連研究

2.1　E-AoSAS++

本研究室で提案している E-AoSAS++ は, 組込みソフトウェアを並行に動作する状態遷移機械 (CSTM) の集合として規定している. 各 CSTM が協調動作する事で組込みソフトウェアの機能を実現している. E-AoSAS++ に基づいた開発プロセスは,PLSE の概念に基づき, アーキテクチャをコア資産として再利用することを目的としている. E-AoSAS++ の開発プロセスにおいて構築するアーキテクチャは, 以下の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
  ソフトウェアファミリ全体の構造を表すアーキテクチャである. 機能・非機能をアスペクトとしてモジュール化し, 構築したコンポーネントの関連を示す. 新しく機能が追加されたら, 再構築する.
- プロダクトアーキテクチャ
  ソフトウェア開発をおこなう際, プロダクトラインアーキテクチャから必要なコンポーネントを選択し, 作成する. プロダクト毎に作成をおこなうが, 再利用できるアーキテクチャがあれば再利用する.

2.2　PLSE

PLSE では, ソフトウェア開発において再利用可能部品であるコア資産を開発し, 管理をおこなう. 以下の 3 つの活動を系統的におこなうように定義されている.

- ドメイン工学
  ソフトウェアを分析し, コア資産を開発する
- アプリケーション工学
  コア資産を用いてソフトウェア製品を開発する
- 管理
  コア資産および製品を管理する

2.3　Feature Oriented Reuse Method

Feature Oriented Reuse Method(以 下,FORM) は,PLSE におけるドメイン工学を支援する方法論である [2]. ドメイン工学の分析を基に, 開発者が実現可能なユーザ要求をフィーチャとして定義し, コア資産開発をおこなう.

3　仕様モデルの記述法

本研究の目的は, 仕様モデルとアーキテクチャの関連を明確にし, 対応付けることである. 既存の仕様モデルでは, 記述する仕様が曖昧であるという問題点があるので, その解決法を示す.

3.1　フィーチャ図

フィーチャ図は, 抽出されたフィーチャを明示的に整理する記述法である. フィーチャ図は, 以下の 4 階層を用いて表す.

- 特性層
  機能特性・非機能特性フィーチャを記述
- 操作環境層
  ハードウェアを記述

- ドメイン技術層
  ドメインに特化した技術を記述
- 実現技術層
  一般的なソフトウェアの実現に使用される技術を記述

また, フィーチャ図では, フィーチャを必須フィーチャの他に,1 つ以上の選択のオルタネイティブフィーチャ,1 つ選択のオアフィーチャ, および任意選択のオプショナルフィーチャに分類して表す. フィーチャ図の例としてエレベータ制御システムのフィーチャ図を図 1 に示す.

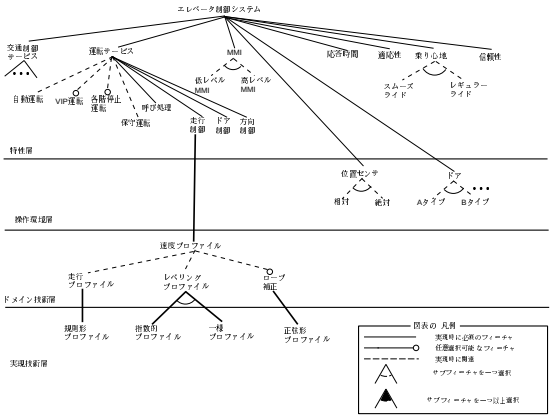

図 1　エレベータ制御システムのフィーチャ図

3.2　既存仕様モデルの問題点

フィーチャ図の特性層では, 機能・非機能特性を記述するが, フィーチャをどこまで詳細化して記述するのか明確に定義されていない. 記述されるフィーチャは, 開発者に依存している.

3.3　解決方法

問題点の解決方法を提案する.

3.3.1　特性層のフィーチャの記述

特性層の記述を特性層とアトミック特性層の 2 層に分けて記述する.2 層では, 以下の様に記述する.

- 特性層
  ユースケース図を作成し, ユースケースをフィーチャとして記述する
- アトミック特性層
  特性層で抽出されたフィーチャを実現する際に必要な機能特性または, 詳細化された非機能特性を記述する

2 層に分けて定義する事によって, 詳細レベルとして曖昧であった特性層のフィーチャの記述が明確になると考える.

また, 既存の仕様モデルでは, 非機能を抽象度の高い記述をしているが, 非機能をどの機能と関連するか理解できるまで詳細に記述すべきと考える.

4　仕様モデルとアーキテクチャの関連の仮説

3 節で再定義したフィーチャ図を仕様モデルとし, 仕様モデルとアーキテクチャとの対応関係について以下の仮説を考えた.

4.1　特性層, アトミック特性層

アーキテクチャと PLSE における仕様モデルであるフィーチャ図との関連の仮説を以下に述べる. E-AoSAS++ では, 機能・非機能をアスペクトとしてモジュール化し, コンポーネントとして記述するので, フィーチャとコンポーネントが 1 対 1 で対応すると考える [3]. よって, 以下の仮説を提案する.

・特性層, アトミック特性層に現れるフィーチャは, 全てアスペクトとして扱う.

4.1.1　操作環境層

操作環境層に現れるフィーチャは, ハードウェアを記述するので, アーキテクチャにおけるハードウェアコンポーネントと 1 対 1 で対応すると考えられる.

4.1.2　ドメイン技術層, 実現技術層

ドメイン技術層および実現技術層に現れるフィーチャは, 実現時に扱う技術を記述するので, アーキテクチャ上には現れないと考えられる.

5　事例を用いた仮説の検証

プリンタ制御ソフトウェアと携帯電話制御ソフトウェアを事例とし, 仮説の検証をおこなう.

5.1　プリンタ制御ソフトウェアの仕様モデル

プリンタ制御ソフトウェアの分析をおこなう. ユースケースを基にして, 考えられるプリンタ制御ソフトウェアの機能としては, 印刷, コピー, スキャン, 表示 (状態, エラー), および管理などが挙げられる. その機能を実現する機能特性として, 印字, 送紙, 印刷データ受信, データ読み取りなどが挙げられる. 非機能特性として, 使用性, 信頼性, 実時間性 (印刷速度, 送紙速度, 印字速度) が挙げられる. プリンタを構成するハードウェアは, 受信装置, プラテン, インクノズル, インクカートリッジ, スキャナーなどのハードウェアが挙げられる. 結果を基に作成したフィーチャ図を図 2 に示す.

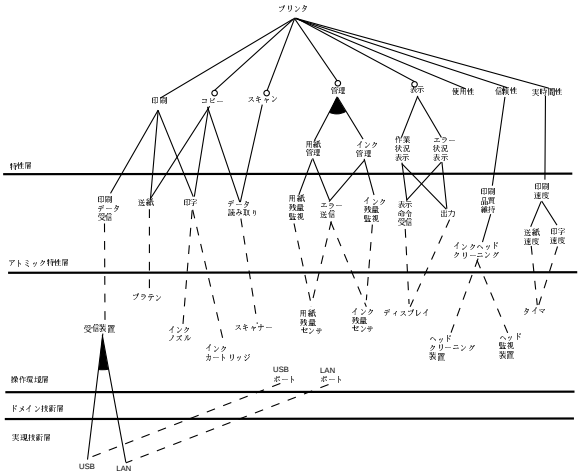

図 2　プリンタ制御ソフトウェアのフィーチャ図

5.2 携帯電話制御ソフトウェアの仕様モデル

プリンタ制御ソフトウェアと同様に携帯電話制御ソフトウェアの分析をおこない, 仕様モデルを作成する. 作成したフィーチャ図を図 3 に示す.

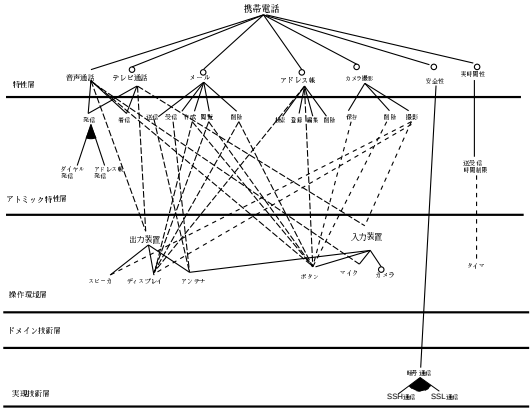

図 3　携帯電話制御ソフトウェアのフィーチャ図

5.3 仕様モデルとアーキテクチャの対応関係

図 2, 図 3 のフィーチャ図から仮説に基づきアーキテクチャを作成し, 仮説を検証する.

5.3.1 特性層, アトミック特性層

仮説で, 特性層, アトミック特性層に現れるフィーチャは, 全てアスペクトとして扱うとした. 全てのフィーチャをアスペクトとしてアーキテクチャを構築すると, フィーチャとアーキテクチャのコンポーネントは 1 対 1 に対応する. しかし, すべてをアスペクトとし, コンポーネントとしてしまうと, 横断的コンサーンではないフィーチャまでもアスペクトとしてモジュール化されてしまい, 構造が複雑化してしまう. フィーチャ図上で認識可能なアスペクトとして扱うフィーチャの提案をおこなう.

まず, フィーチャ図上の関連でフィーチャが複数のフィーチャを横断する場合, アスペクトとしてとらえ, モジュール化する必要があると考えられる. 図 2 の例では, 印刷が印刷データ受信, 送紙および印字を横断している. また, 印字は印刷とコピーを横断している. このようなフィーチャは, アスペクトとしてモジュール化する事が適切であると考える. 図 3 の携帯電話制御ソフトウェアにおいても, 音声通話が発信と着信に横断している事が認識できるので, プリンタ制御ソフトウェアと同様に適用可能であると考える.

選択を表すフィーチャをモジュール化することで, アーキテクチャ上でもコンポーネントの選択を明示できると考える. プロダクトラインアーキテクチャ上に明示することにより, プロダクトラインアーキテクチャ上でのコンポーネントの選択が可能になり, プロダクトアーキテクチャの決定が容易になると考えられる.

しかし, 他のフィーチャと同じ様にモジュール化した場合, 選択を表すフィーチャを, プロダクトラインアーキテクチャ上で明示する事ができない. そこで, 選択を表すフィーチャをアーキテクチャ上で示す手段として, 図 4 の様に選択を表すフィーチャのポリシーのステレオタイプとして $<< SpecificationModel >>$ を付加する方法を提案する. 選択のパターンをノートを用いて記述する. 図 4 では, オアフィーチャを例に示したが, 他の選択を表すフィーチャも同様の記述で表す事が可能である.

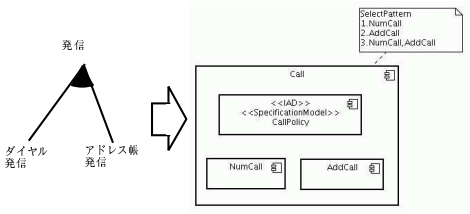

図 4　オアフィーチャの例

非機能フィーチャもアスペクトとしてモジュール化すべきフィーチャの候補である. 非機能特性として抽出されたフィーチャは, 他のフィーチャとの関連が現れるまで詳細化される. 他のフィーチャとの関連が認識でき, 非機能を実現する際に, 必要な機能およびハードウェアが現れた場合, アーキテクチャ上にコンポーネントとして記述する必要があると考える.

図 2 のように, 信頼性を詳細化した結果, インクヘッドクリーニングという機能特性の追加が必要となる. 非機能を実現するのに必要な機能フィーチャをアスペクトとしてモジュール化する必要があると考えられる. 図 3 では, 送受信時間制限を実現する際に, タイマというハードウェアが必要となる. このような場合, タイマをコンポーネントとして追加する必要があると考える. 図 2 の印字速度のような非機能は, 機能特性の印字と関連すると考えられるが, 印字に対応するコンポーネントの処理の変更で実現可能であると考えられる. このような場合は, アーキテクチャ上には, アスペクトとしてモジュール化する必要はないと考える.

以上より, 特性層, アトミック特性層でアスペクトとして扱うフィーチャは, 以下が挙げられる.

- 特性層, アトミック特性層で認識できる横断的コンサーン
- 非機能フィーチャの実現に必要な機能フィーチャ
- オア, オルタネイティブ, オプショナルフィーチャ

5.3.2 操作環境層

図 3 では, ボタン, スピーカおよびディスプレイなどのハードウェアがコンポーネントと対応し, 操作環境層に現れたフィーチャは全てコンポーネントと対応した. 同様に, 図 2 の例では, 受信装置, インクカートリッジおよびインクノズルなどのフィーチャは, コンポーネントとして対応した. 受信装置のオアフィーチャとしてドメイン技術層と関連している USB ポートと LAN ポートは, アーキテクチャ上で, 受信装置を詳細化した USB ポー

トをコンポーネントとした.

5.3.3 ドメイン技術層, 実現技術層

ドメイン技術層および実現技術層に現れるフィーチャは, 実現時に扱う技術を記述するので, アーキテクチャ上には現れない. 図 2 の実現時に USB や LAN を使用するの時, 関連する USB ポートや LAN ポートは, コンポーネントとして対応するが, 技術として記述されている USB や LAN は, アーキテクチャ上には現れない.

5.4 アーキテクチャの作成

以上の事により作成したプリンタ制御ソフトウェアのプロダクトラインアーキテクチャを図 5 に示す.

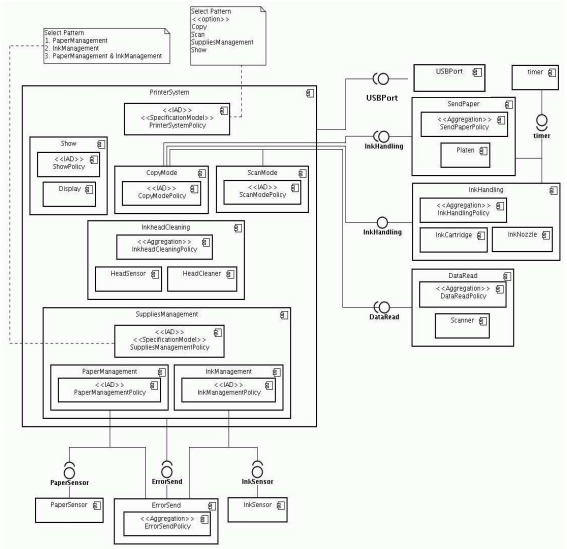

図 5　プロダクトラインアーキテクチャ

6 考察

6.1 オア, オルタネイティブ, オプショナルフィーチャに関する考察

本研究では, 特性層, アトミック特性層の選択を表すフィーチャをアスペクトとして扱い, モジュール化しプロダクトラインアーキテクチャ上で記述する方法を提案した. 操作環境層においても選択を表すフィーチャは存在するので, その記述法を考える必要がある. 図 2 の例では, 受信装置のオアフィーチャとして USB,LAN があり, これらのフィーチャと関連するハードウェアとして USB ポートと LAN ポートが存在する. 操作環境層で選択が現れる前に, 特性層, アトミック特性層でそれぞれのハードウェアに対応する詳細レベルまで記述することが考えられる. しかし, 機能特性は, 変わらずにハードウェアだけ変更する場合がでてくる可能性がある. よって, 操作環境層においても選択が現れた時は, 特性層, アトミック特性層と同様にモジュール化する必要があると考えられる.

6.2 対応関係の一般化について考察

本研究で提案した対応関係の一般化について考察をおこなう. 事例として用いたドメイン以外として, 図 1 のエレベータ制御システムとの対応関係を考える. 図 1 は, 既存の仕様モデルであり,3 節で提案した仕様モデルにしたがってないが, 特性層においては, 運転サービスの様な機能特性は, 各制御に横断しているのが認識できる. よって, 特性層においての対応は可能であると考える. 操作環境層は, 位置センサやドアなどのハードウェアを記述しているので提案した対応関係は適用可能であると考える. ドメイン技術層, 実現技術層は, 実現に使用する技術を記述していることから, 提案した対応関係が適用できると考える. 提案した対応関係により, 仕様モデルとプロダクトラインアーキテクチャとの対応関係は明確にできたと考える. プロダクトラインアーキテクチャからコンポーネントを選択することで, プロダクトアーキテクチャの決定が容易におこなえると考えられる.

7 おわりに

本研究では, 仕様モデルと E-AoSAS++ に基づくアーキテクチャとの関連について考察をおこなった. 仕様モデルとして,FORM で提案されているフィーチャ図を用いる事を考え, フィーチャ図の再定義をおこなった. フィーチャ図と E-AoSAS++ に基づくアーキテクチャとの関連の仮説をたて, 事例を用いて仮説の検証をおこなった. 今回, 複数の事例を用いて, 検証をおこなったが, 規模が小さく, 選択のパターンも少なかった. 規模の大きいソフトウェアを開発する際に, 今回の対応付けがうまくいくとは限らない. まだ, 検証をおこなう必要があると考えられる. また, 今回, 提案した対応付けを仕様モデルからアーキテクチャへの自動生成をおこなうツールの実現が挙げられる.

参考文献

[1] Linda M.Northrop, “SEI’s Software Product Line Tenets”, IEEE SOFTWARE, p.32, 2002.

[2] K.C.Kang, S.Kim, J.Lee, “FORM:A Feature-Oriented Reuse Method with Domain-Specific Reference Architectures”, POSTECH, p.28, 1998. Engineering”, IEEE SOFTWARE, p58,2002.

[3] K.Lee, K.C.Kang, M.Kim, S.Park, “Combining Feature-Oriented Analysis and Aspect-Oriented Programming for Product Line Asset Development”, IEEE SOFTWARE ,2006.

[4] Masami Noro, Atsushi Sawada, Yoshinari Hachisu, Masahide Banno, “E-AoSAS++ and its Software Development Environment”, Proc.of the 14th Asia-Pacific Software Engineering Conference(APSEC’07), p.206-213, Dec. 2007.