アスペクト指向ソフトウェアアーキテクチャの文書化に関する研究

M2006MM035 安江 基規
指導教員　野呂 昌満

1　はじめに

今日のソフトウェア開発においてソフトウェアアーキテクチャの役割は重要である．PLSE[2] のようなアーキテクチャを中心としたソフトウェア開発が注目されている．一方，そのソフトウェアアーキテクチャの文書化に関しては次のような未解決の問題が存在する．

1. 多重視点による記述の妥当性
多重視点からソフトウェアを捉えることで，着目したい点を中心にソフトウェアアーキテクチャを記述することができるが，視点が多数存在することになり，ソフトウェアアーキテクチャ記述を複雑にしている．

2. 応用領域による文書化の変容
既存のアーキテクチャの記述は応用領域ごとに記述法を提案しており，開発コストの増加を引き起こす要因となる．記述方法を統一的に一般化することが必須であると考える．

3. 実用性
多くのアーキテクチャ記述方法が存在するが，ソフトウェア開発において実用的でなければ無意味である．

4. アスペクト指向への対応
アスペクト指向技術の普及により，アスペクト指向を用いたソフトウェア開発やアーキテクチャの研究が行われているが，アスペクト指向ソフトウェアアーキテクチャの記述方法に関しての考察は不十分である．

本研究の目的は，アスペクト指向ならびに PLSE の観点からソフトウェアアーキテクチャの多重視点を整理し，統一的かつ実用的なアーキテクチャの記述方法を提案することである．組込みシステムを応用領域の例として，アスペクト指向ソフトウェアアーキテクチャの記述方法を提案する．提案する記述方法を組込みシステム以外の応用領域への適応可能性を考察し，いくつかのソフトウェアのアーキテクチャを記述することにより，一般性を検証する．

本研究では，アーキテクチャ記述に UML[4] を用いる．UML はソフトウェアのモデリング言語として世界的に広く普及しているので，提案する記述方法が認知されやすいと考える．また，アーキテクチャ記述を用いたソフトウェア開発を考えたさい，ツールも充実しているので，実用的である．

2　関連研究

本説では関連研究である，IEEE 1471[1]，V&B[5] について説明する．

2.1　IEEE 1471

IEEE 1471 はソフトウェアアーキテクチャ記述の標準化を目指し，アーキテクチャ記述のためのメタモデルと定義を提供する．IEEE 1471 の目的はソフトウェアアーキテクチャの表現化とソフトウェアアーキテクチャを利用したソフトウェア開発におけるステイクホルダの意志疎通を促進することである．それにより，概念要素の標準化やソフトウェアアーキテクチャ記述を通して得られる品質およびコスト面での向上に役立つ基盤を与える．IEEE 1471 はアーキテクチャ記述の概念モデルを**図 1** のように定義している．

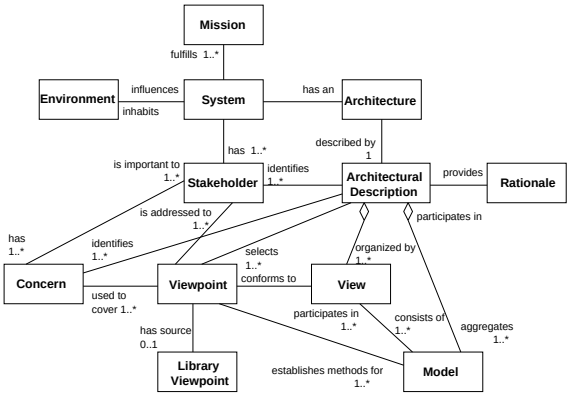

図 1　IEEE 1471:アーキテクチャ記述概念モデル

2.2　Documenting Software Architectures:Views and Beyond

V&B は，ソフトウェアアーキテクチャを捉える視点と各視点の相互作用であると定義している．ソフトウェアアーキテクチャを適切な視点から記述することで，明確に文書化することを目的にしている．複数の視点が混在していたり，視点が曖昧なままの文書は，的確にソフトウェアを表現することができない．ステイクホルダと視点の整理を行うことがソフトウェアアーキテクチャ文書にとって重要であると指摘している．
V&B ではソフトウェアを捉える視点として以下の 3 つの視点を必要としている．

- Module Viewtype
- Component and Connector Viewtype
- Allocation Viewtype

Module Viewtype は静的観点でシステムを捉えたもので，クラス，関数，インターフェースなどを扱い，実現するコードの単位を示す．Module Viewtype を記述することにより，ソフトウェアの再利用性を可視化する．
Component and Connector Viewtype は動的観点でシステムを捉えたもので，プロセス，データフロー，シーケンスなどを扱い，システムの動的な振舞いを示す．シ

ステムの可用性，パフォーマンス，信頼性などの分析に有効である．
Allocation Viewtype は物理的観点でシステムを捉えたもので，ソフトウェアが非ソフトウェアにどのように割り当てられるかを示す．非ソフトウェアとしてハードウェア，人，コストなどがあげられる．

3 アスペクト指向ソフトウェアアーキテクチャの記述方法の提案

本節では，自動販売機制御ソフトウェアを例として，アスペクト指向ソフトウェアアーキテクチャの記述方法の提案をおこなう．我々はアーキテクチャに基づくソフトウェア開発をおこなうことを前提としているので，はじめにソフトウェア開発プロセスを整理する．コンサーン，ビュー，ビュータイプを整理し，各プロセスでのアーキテクチャ記述方法を提案する．

3.1 アーキテクチャに基づくソフトウェア開発プロセス

アーキテクチャに基づくソフトウェア開発プロセスを整理する．我々は PLSE の観点に基づき，アーキテクチャをコア資産とし，再利用を目的としたソフトウェア開発を考えている．各プロセスでアーキテクチャ記述が必要となる．ソフトウェア開発プロセスの概要を以下に示す．

- 仕様モデル決定
- ソフトウェアアーキテクチャ構築
- 実行前検査
- コード作成

各プロセスでのステイクホルダを考え，ステイクホルダのコンサーンを特定することにより，ステイクホルダが協調する場面が特定できる．また，各場面でのアーキテクチャ記述が必要になることがわかる．ステイクホルダとして，ユーザ，ドメインエンジニア，アーキテクト，メンテナ，コーダが存在し，以下のステイクホルダが協調する場面が考えられる．

1. ユーザとドメインエンジニア
2. ドメインエンジニアとアーキテクト
3. アーキテクトとメンテナとコーダ

1 つ目は，ユーザのコンサーンである要求，機能性などを基に仕様モデルを作成する．ドメインエンジニアはユーザの要求からコア資産を参照し，開発するソフトウェアのドメインの概念モデルであるコンセプチャルアーキテクチャ (CA) を設計する．CA のビューはコンセプチャルビューである．

2 つ目は，CA を基にコア資産としてのアーキテクチャを参照してプロダクトラインアーキテクチャ (PLA) を設計する．PLA は CA を基に，PLSE の観点から，再利用部品であるアーキテクチャを利用して機能を追加したアーキテクチャである．また，機能を追加するさい，アスペクトを考慮してアーキテクチャ記述をおこなう．PLA におけるコンサーンは，再利用部品としてのアーキテクチャと CA との関連である．PLA のビューをプロダクトラインビューとする．3 つ目は，PLA と仕様モデルを基に実際に実現するアーキテクチャの記述が必要となる．このアーキテクチャをプロダクトアーキテクチャ (PA) とする．PA のコンサーンは，実現するアーキテクチャの構造となる．PA のビューはプロダクトビューである．

3.2 アーキテクチャのビュータイプの整理

アーキテクチャ記述は，捉える視点によって適切な記述をすることで，より明確な文書となる．アーキテクチャ記述には，以下のビュータイプから捉えた記述が必要であると考える．

- 静的観点
- 動的観点

静的観点では，システムの静的構造を表現し，Class や関数，インターフェースを扱うことで実現レベルでの再利用性を可視化する．静的観点を表現するために，UML のクラス図，コンポーネント図を用いる．クラス図は Class の静的構造と各クラス間の関連を示すことで，開発するソフトウェアの概念要素を記述することができる．コンポーネント図は Component の外部仕様と内部構造を表す図である．Component は提供インターフェースと供給インターフェースによって振舞いを定義することができる．また，Component は再利用可能な要素として表現することができる．

動的観点では，プロセスやシーケンスを扱い，システムの動的な振舞を示すことでシステムの可用性，パフォーマンス，信頼性などを分析する．動的観点を表現するために，UML のシーケンス図を用いる．シーケンス図は時間的な流れに沿って，オブジェクトやクラス間のメッセージのやり取りを記述し，システムの振舞いを表現する．

3.3 コンセプチャルアーキテクチャの記述方法

CA の記述方法を提案する．CA は要求分析の結果を基にした，ソフトウェアの論理的なアーキテクチャである．アスペクトは考慮せず，オブジェクトモデリングによって構築されたアーキテクチャである．開発対象となるソフトウェアの論理的なアーキテクチャを作成することにより，アスペクト導入の前段階でソフトウェアの構造を整理することができる．
静的観点により捉えた CA の記述方法には UML のクラス図を用いる．自動販売機制御ソフトウェアの CA の記述方法を**図 2** に示す．

3.4 UMP,PLA,PA の記述方法

UMP[3] は，アーキテクチャの階層構造や複数のモジュールによって構成管理されるモジュールを構造化のパターンを用いて記述することで，アーキテクチャ記述の一様性や容易性を保証する．アスペクト指向技術の本質は，アスペクトとして抽出した要素を，アスペクト間記述によってコンフィギュレーションの切替えをおこなうことであり，UMP を用いて記述する．構成要素を管理

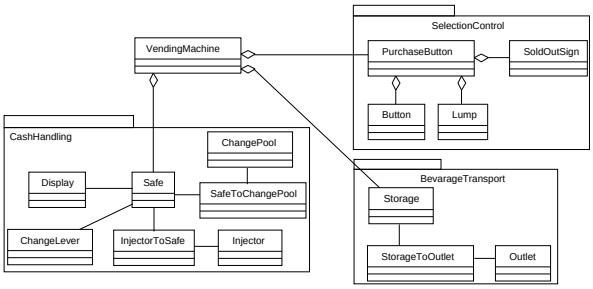

図 2　コンセプチャルアーキテクチャの記述方法

するモジュールが，どのモジュールにメッセージを通知するかを決定し，アスペクト間記述によって実現する．各アスペクトと各アスペクトを管理するモジュール (Policy)，アスペクト間記述 (IAD) をアーキテクチャ記述として記述すべきコンサーンとして捉え，それらの記述方法を考えることで，アスペクト指向ソフトウェアアーキテクチャの記述が可能になると考える．

アスペクトの記述は UML の Component を用いる．Component は分類子で構成される再利用可能部品を示す．
アスペクト間記述の記述は UML の Component にステレオタイプ $<< IAD >>$ を付加して定義する．
アスペクト間記述によって複数の構成を成す状態遷移機械を UML の Component にステレオタイプ $<< Aggregation >>$ を付加して定義する．
提供インターフェースと供給インターフェースをロリポップを用いて記述することで，複数の構成を成す状態遷移機械を表現することが可能となる．
PLA，PA は開発プロセスにおいて，PA は PLA から実現する機能を抽出したアーキテクチャなので記述方法は同じである．PA の記述方法を例示する．PA の静的観点による記述方法を**図 3** に示す．

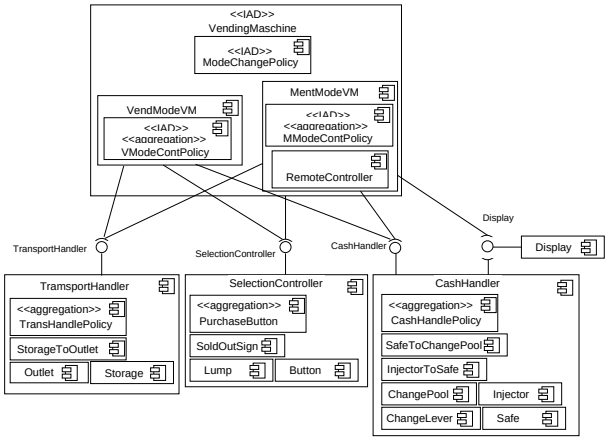

図 3　プロダクトアーキテクチャの記述方法

自動販売機制御ソフトウェアには，ユーザが商品を購入する VendModeVM と，管理者によってメンテナンスをする MentModeVM の 2 つのモードがある．PA では 2 つのモードを ModeChangePolicy によりコンフィギュレーションを切替えることで実現する．Policy が構成を管理するモジュールの内部関係をクラス図をもちいて記述する．Policy によって管理される各状態遷移機械の内部関係の例を**図 4** に示す．

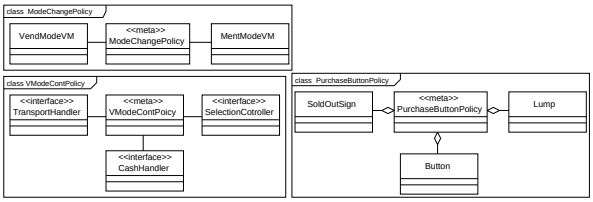

図 4　プロダクトアーキテクチャ:Policy と状態遷移機械の関連

PA の動的側面は UML のシーケンス図をもちいて記述する．シーケンス図をもちいてシステムの動的側面を表現する．また各状態遷移機械の状態遷移について，UML の状態マシン図をもちいて記述する．シーケンス図と状態マシン図により，システムの動的側面を不備なく表現することができる．PA の動的観点による記述方法を**図 5**，**図 6** に示す．

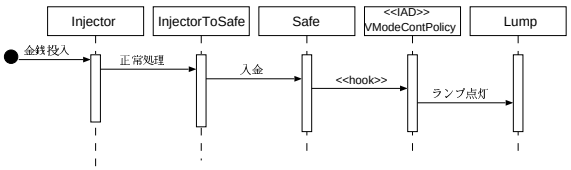

図 5　プロダクトアーキテクチャ:シーケンス図

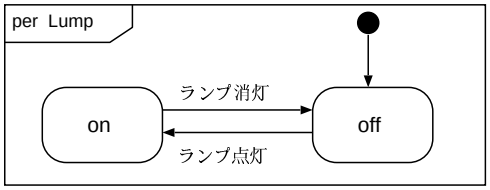

図 6　プロダクトアーキテクチャ:状態マシン図

4　考察

提案したソフトウェアアーキテクチャの記述方法について以下の観点から考察する．

- 視点の妥当性
- 組込み領域以外のソフトウェアへの適応について
- UML の妥当性

4.1　視点の妥当性

ソフトウェア開発プロセスの整理をおこなった結果，各プロセスでのビュー，コンサーンを整理することができた．ビュータイプとして，静的観点，動的観点のほかに物理的観点が挙げられる．物理的観点では，ソフトウェアのハードウェアへの配置を記述する．ソフトウェアのハードウェアへの配置はプラットフォームの設計・実現問題とする．ビュー，コンサーン，ビュータイプの整理

ができているので，提案するアーキテクチャ記述の視点は妥当であると考える．

4.2 組込み領域以外への適応について

本研究は，ソフトウェアアーキテクチャをアスペクト指向ならびに PLSE の観点から捉え，ビュー，ビュータイプ，ステイクホルダを整理し，アスペクト指向ソフトウェアアーキテクチャの記述方法を提案した．領域に依存する部分はドメイン特有のアスペクトとして取り扱う．領域に依存する部分をアスペクトとして扱うことで，組込み領域以外のドメインにも適応可能である．

4.3 UML の妥当性

ソフトウェアアーキテクチャを記述する方法としては UML の他に，アーキテクチャ記述言語 (ADL) を用いる方法や，独自に図式表現を作成する方法が考えられる．ADL は形式的に定義された文法と意味を備え持つ構文をコンポーネントに記述することによりアーキテクチャを記述する．ADL について様々な研究がおこなわれているが，現状の研究段階では標準規格がなく，広く使われていない．また，特定の分野に特化しているものがほとんどである．独自に新しいアーキテクチャの図式表現を作成することにより，記述したいようにアーキテクチャを記述することができるので，アーキテクチャ記述は容易になる．しかし，独自にアーキテクチャ記述を開発することは，労力がかかり，汎用的ではない．

UML はソフトウェアのモデリング言語として広く普及しているので，提案する記述方法が認知しやすいと考える．UML は UML プロファイルにより拡張機構が定義されているので，新たな要素を容易に追加することができる．UML はツールも充実しているので，実用的であると考える．UML を用いることが妥当である．

次に，本研究で用いた UML 図の妥当性を考察する．

静的観点

UML を用いた静的観点の図式表現として，以下の要素が考えられる．

- Package
- Component

UML の Package は任意のカテゴリによってソフトウェア構成要素のグループ化を表現する．Package はグループ要素すべてが含まれている場合は階層表現を記述可能であるが，ある一部の構成要素のみが複数の Package に所有される場合は，階層表現を記述することができないので，UMP 記述の，複数の管理モジュールによって管理されるモジュールを表現することができない．

UML の Component は複数のソフトウェア構成要素を再利用可能な単位にモジュール化する．Component は提供インターフェースと供給インターフェースをロリポップをもちいて記述することで，複数の構成を成すモジュールを表現することが可能となる．よってアスペクト指向ソフトウェアアーキテクチャの静的構造は Component を用いて表現することが妥当である．

動的観点

UML を用いた動的観点の図式表現として，シーケンス図，コミュニケーション図がある．シーケンス図はオブジェクトやクラスなどの分類子間のメッセージのやりとりを時系列にそって表現する．コミュニケーション図はクラス図やコンポーネント図のような記法でオブジェクト間のメッセージのやりとりを表現し，シーケンス番号によってメッセージの順序を表現する．コミュニケーション図は，静的構造にシーケンス番号を表記しており，視点が複数存在する．多重視点によるアーキテクチャ記述は曖昧な表現であり，我々が提案したいソフトウェアアーキテクチャの記述方法の考えに矛盾するので，動的観点からのソフトウェアアーキテクチャ記述はシーケンス図を用いて表現する．

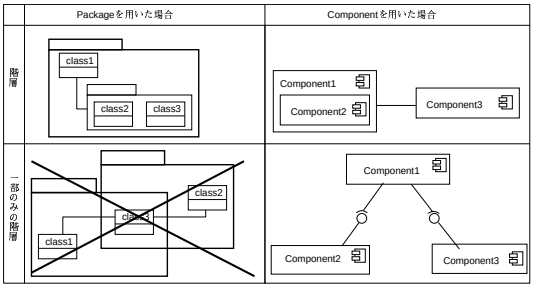

図 7 静的観点の UML 図の妥当性

5 おわりに

本研究ではアスペクト指向ならびに PLSE の観点からソフトウェアアーキテクチャの多重視点を整理し，統一的かつ実用的なアーキテクチャの記述方法を提案した．今後の課題としては，本研究は開発プロセスに基づいたアーキテクチャ記述なので，開発プロセスに依存しない場合の検証をおこなう必要がある．

参考文献

[1] IEEE 1471,"IEEE Recommended Practice for Architectural Description of Software-Intensive Systems," 2000.

[2] L.M.Northrop : "SEI's Software Product Line Tenets," IEEE Software, Vol.19 No.4, pp.32-40 (2002).

[3] 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(APSEC 2007),pp.206-213,Dec,2007.

[4] Object Management Group, "UML2.0 仕様書," 2006.

[5] P.Clements,F.Bachmann,L.Bass,D.Garlan,J.Ivers, R.Little,R.Nord,and J.Stafford,"Documenting Software Architectures: Views and Beyond," *Addison-Wesley*, 2002.