自動販売機制御ソフトウェアの再開発
～自動販売機制御ソフトウェアのログ解析支援ツールの設計と実現～

M2007MM014 小島守道
指導教員：野呂昌満

1　はじめに

ソフトウェアの実行時に詳細な動作履歴をあらわすログを出力し，実行後にログを解析することにより，ソフトウェアの正常な動作，異常な動作を確認することができる．この方法は，高い信頼性が求められ，常時監視することが難しい Web アプリケーションや組み込みソフトウェアに対して有効である．しかし，近年のソフトウェアの高機能化に伴うログ出力の多様化が原因で，ソフトウェアのログ解析に要する工数は増加の一途を辿っている．多種多様な種類のログが出力されることから，解析手法は多岐に渡る．このような問題を解決するために，さまざまな解析手法を支援するツールが求められている．
文献 [6] では，ログのパターンを定義することでパターンマッチングをおこない，ログの検索をおこなうツールを提案している．文献 [5] では，性能デバッグするために実行情報を視覚的に表示するツールを提案している．しかし，本研究で対象とするログ解析では，さまざまな視点からログ解析をおこなうツールが求められる．文献 [6] で提案されているツールでは，パターンを与えるのが困難であると考えられ，文献 [5] のツールでは単一の視点からのログ解析に限られている．
本研究では，さまざまな視点からのログ解析を支援するログ解析支援ツールを実現することで，ログ解析における問題の解決を図る．我々は，ログ解析支援ツールの開発において，Product Line Software Engineering[3](以下，PLSE) に基づくアスペクト指向アーキテクチャ中心の開発プロセスを実施した．PLSE に基づく開発プロセスでは，製品系列の特徴を考慮したソフトウェアアーキテクチャの構築が鍵となる．
本研究の目的は，ログ解析支援ツールの特性を考慮したソフトウェアアーキテクチャを構築することにより，変更に柔軟なソフトウェアを実現することである．ログ解析支援ツールに対する要求として，画面表示の変更や解析手法の変更などの要求が頻繁に発生すると考えられる．ソフトウェアアーキテクチャとして，このような要求に対し柔軟に対応可能なアーキテクチャを構築する必要がある．本研究では，ソフトウェアアーキテクチャの比較をおこなうことで，適用したソフトウェアアーキテクチャの妥当性を考察する．
本研究で利用したベース技術を次に示す．

- ソフトウェアアーキテクチャ技術
  - ・アスペクト指向ソフトウェアアーキテクチャ
  - ・MVC アーキテクチャ
  - ・Pipes and Filters アーキテクチャ
  - ・デザインパターン

上記のベース技術を選択し，組み合わせて適用することにより，メタ技術として PLSE を習得する．

2　ログ解析支援ツール概要

本研究で開発したログ解析支援ツールの概要を図 1 に示す．

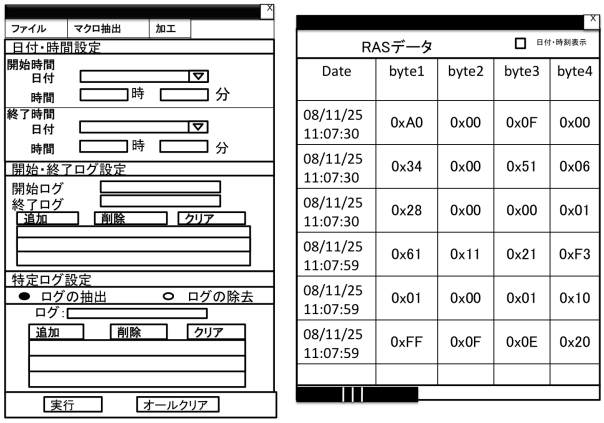

図 1　ログ解析支援ツール概要

我々は，ソフトウェアのログ解析手法を整理し，ログ解析を解析対象のログの変換をおこなうログ変換処理，指定した条件でログの選別をおこなう抽出処理，ログを異なる形式に変換する加工処理の三つに分類することができると考えた．また，ログ解析するさいには，抽出処理と加工処理を組み合わせて解析をおこなう．
開発するログ解析支援ツールでは，抽出の機能として，時間やログの種類を指定した抽出処理などの機能を提供する．加工の機能としては，指定した種類のログデータに対する色づけや金銭データを集計して表示するなどの機能を提供する．利用者はこれらの機能を組み合わせることでログの解析をおこなう．

3　ログ解析支援ツールのための開発プロセス

我々は，ログ解析支援ツールの開発において，PLSE に基づいた開発プロセスを実施した．対象の製品系列は，ログ解析支援ツールとする．実施した開発プロセスを図 2 に示す．
本 OJL では，ドメインの特徴とソフトウェアアーキテクチャをコア資産として蓄積する．ソフトウェアアーキテクチャの構築においては，ドメインの特徴からアスペクトを抽出し，概念的アスペクト指向アーキテクチャを構築する．概念的アスペクト指向アーキテクチャを詳細化することにより，構築するソフトウェアアーキテクチャを詳細アーキテクチャと定義する．
概念的アスペクト指向アーキテクチャを図 3 に示す．

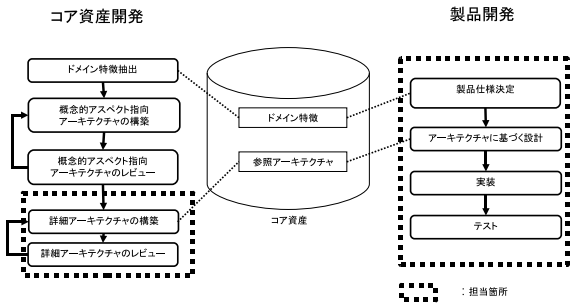

図 2 ログ解析支援ツールのための開発プロセス

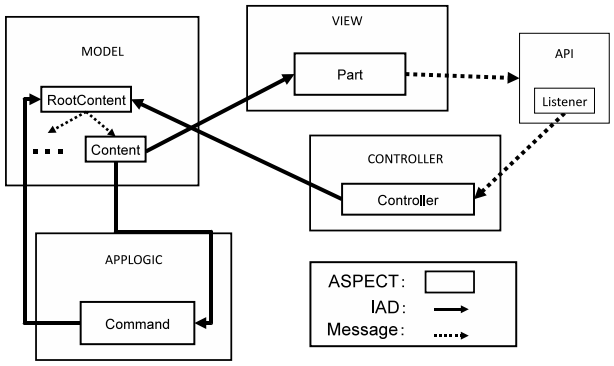

図 3 概念的アスペクト指向アーキテクチャ

概念的アスペクト指向アーキテクチャでは，ログ解析支援ツールの，表のデザインや文字の大きさなど，画面表示部品の見映えに関する変更が多いという特性を考慮し，MVC アーキテクチャを基にコンサーンの分離をおこなった．概念的アスペクト指向アーキテクチャは，画面表示部品の構成をあつかう ViewModel アスペクト，画面表示部品の見映えをあつかう View アスペクト，GUI からの入力をあつかう Controller アスペクト，アプリケーション固有の処理をあつかう Model アスペクトで構成される．概念的アスペクト指向アーキテクチャでは，Buschmann ら [2] が提案する MVC アーキテクチャの View を構成をあつかう ViewModel と見映えをあつかう View に分離することにより，見映えの変更に対する柔軟性を保証する．本研究では，概念的アスペクト指向アーキテクチャを基にした詳細アーキテクチャの構築について述べる．

4 詳細アーキテクチャ

本節では，コア資産開発で得られたドメインの特徴と概念的アスペクト指向アーキテクチャを基に，詳細アーキテクチャの構築をおこなう．詳細アーキテクチャの構築においては，ログ解析支援ツールの次の特性を考慮してアーキテクチャの構築をおこなう必要がある．

1. 画面の見映えの変更が多い
2. ログに対する処理の追加変更が多い
3. 出力されるログデータの形式の変更が多い

概念的アスペクト指向アーキテクチャでは，画面表示の変更を考慮し，MVC アーキテクチャを基に構築しているので，画面表示の変更に柔軟に対応可能である．

2，3 の特性は，ログ解析処理に関わる特性である．概念的アスペクト指向アーキテクチャでは，ログ解析処理は Model アスペクトとしてモジュール化できるが，Model アスペクトの設計方法は規定されておらず，場当たり的な設計がおこなわれる可能性がある．本研究では，詳細アーキテクチャ構築のさい，Model アスペクト内に 2，3 の特性を考慮したソフトウェアアーキテクチャを構築することで，変更に柔軟なソフトウェアを実現することができると考える．Model アスペクト内に構築するソフトウェアアーキテクチャをログ解析部分のソフトウェアアーキテクチャと定義する．

4.1 詳細アーキテクチャの構築

概念的アスペクト指向アーキテクチャを基に詳細アーキテクチャを構築する．詳細アーキテクチャの構造を図 4 に示す．

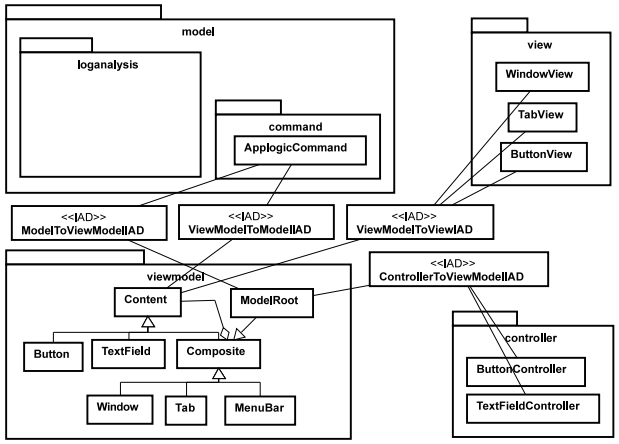

図 4 ログ解析支援ツールのソフトウェアアーキテクチャ

ViewModel アスペクトは，画面表示部品の構成をあつかう．ViewModel アスペクトの構造を図 5 に示す．

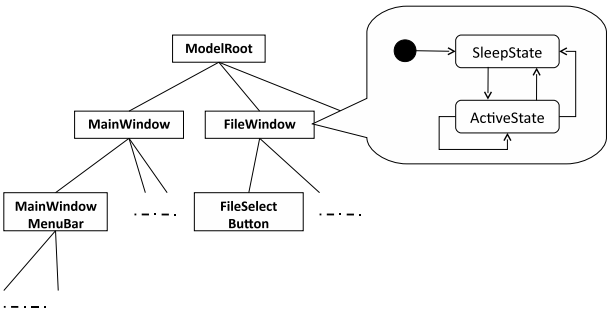

図 5 ViewModel アスペクト

ViewModel アスペクトの構造は，表示部品を木構造であらわした構造となる．図 4 に示すように，Composite パターンを用いて，表示部品の木構造のデータ構造を実現している．各表示部品をあらわすモジュールは，それぞれの構成を管理するための状態遷移機械をもつ．詳細アーキテクチャの動的側面を図 6 に示す．
Controller アスペクトが GUI のイベントを受け取った際

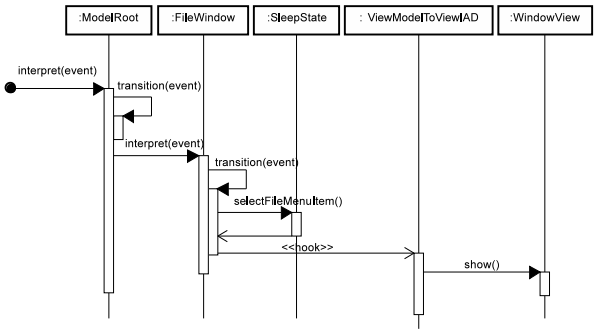

図 6 詳細アーキテクチャの動的側面

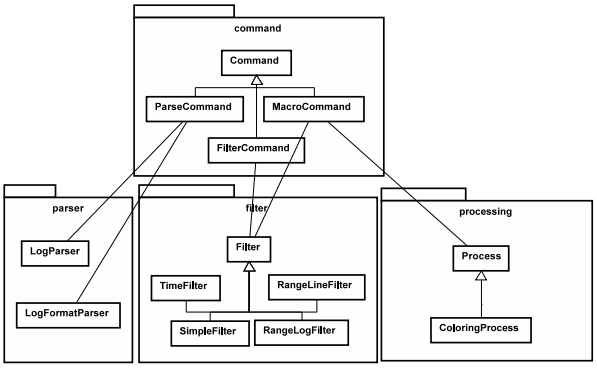

図 7 フィルタの構造

に，アスペクト間記述により，受け取ったイベントを適切なイベントに変換し，ViewModel の木構造のルートにイベントを送信する．ViewModel では，アスペクト間記述により変換されたイベントを受け取り，表示部品の構成をあらわす状態遷移機械の状態を切り替えながら表示部品の構成をあらわす木構造を辿る．状態遷移をおこなったさいに，アスペクト間記述により，View アスペクトを動作させて表示部品の見映えの変更をおこなうことや，Model アスペクトを動作させ，アプリケーションロジックの実行をおこなう．アプリケーションロジックの実行をおこなったさい，イベントを ViewModel の木構造のルートにイベントを送信する．

4.2 ログ解析部分のソフトウェアアーキテクチャ

概念的アスペクト指向アーキテクチャでは，ログ解析処理を Model アスペクトに分離することができる．しかし，概念的アーキテクチャではログ解析処理の特性に対し，十分に柔軟ではない．詳細アーキテクチャ構築のさいには，ログ解析処理の次の特性を考慮したログ解析部分のアーキテクチャを構築する．

- 出力されるログデータの形式の変更が多い
- 抽出処理・加工処理の追加変更が多い
- さまざまな処理の組み合わせが存在する

ログ解析部分のアーキテクチャは，概念的アーキテクチャの Model アスペクト内に構築される．
ログ解析部分のソフトウェアアーキテクチャとして，Pipes and Filters アーキテクチャ[2] を適用する．Pipes and Filters アーキテクチャでは，システムを複数の処理ステップの集合と捉え, 各処理ステップをフィルタコンポーネントとしてカプセル化する．Pipes and Filters アーキテクチャは，フィルタコンポーネントの追加や，フィルタコンポーネントの組み合わせに柔軟に対応可能であるという特徴をもつ．
ログ解析支援ツールでは，ログデータの解析処理，抽出処理，加工処理をフィルタと捉える．各フィルタの静的構造を図 7 に示す．
ログ解析支援ツールの抽出処理として，時刻指定による抽出処理，二つのログを指定し，その間のログデータの抽出をおこなう抽出処理，指定したログデータの抽出・除去をおこなう抽出処理などが考えられる．抽出処理をあわらす抽象クラスである Filter クラスを定義し，各抽出処理は Filter クラスのサブクラスとして実現する．
ログ解析支援ツールでは，抽出処理の追加がおこなわれることが多いので，抽出処理の追加に容易に対応が可能である必要がある．抽出処理の追加をおこなうさいには，追加する抽出処理を Filter クラスのサブクラスとして実現することで対応が可能である．
ログデータの解析処理と加工処理についても同様に追加，変更に対して対応が可能であると考える．

5 考察

本研究の考察として，次の 2 点を考察する．

- 詳細アーキテクチャの妥当性
- ログ解析部分のソフトウェアアーキテクチャの妥当性

5.1 詳細アーキテクチャの妥当性

ログ解析支援ツールでは，画面表示の変更に対する要求が頻繁に発生すると考えられる．詳細アーキテクチャの妥当性について，画面表示部品の見映えの変更に対する柔軟性の観点から考察をおこなう．
詳細アーキテクチャでは，画面表示部品の見映えをあらわす View と，表示部品の構成をあらわす ViewModel が分離されている．各表示部品の見映えを変更する振る舞いは，共通のインタフェースで統一されている．各表示部品の見映えを変更する場合，ViewModel と View 間のアスペクト間記述を見映えの異なる共通のインタフェースを実現したクラスにつなぎかえることで，変更に対応できると考える．

5.2 ログ解析部分のソフトウェアアーキテクチャの妥当性

ログ解析部分のソフトウェアアーキテクチャの妥当性について，次の観点から考察をおこなう．

- 出力されるログの変更に対する柔軟性
- 抽出処理，加工処理の追加変更に対する柔軟性
- 処理の組み合わせの追加に対する柔軟性

ログ解析部分のアーキテクチャの妥当性を考察するために，上記の観点から，Shaw ら [4] の提案するソフトウェアアーキテクチャから次のアーキテクチャを選択し，比

較をおこなう．

- Layered Systems アーキテクチャ
- Pipes and Filters アーキテクチャ
- Pipes and Filters と Shared Data を組み合わせたアーキテクチャ

Layered Systems アーキテクチャ

Layered Systems アーキテクチャでは，特定の抽象度の問題を取り扱う複数の階層によってシステムを構成する．ログ解析部分のソフトウェアアーキテクチャにこのアーキテクチャを適用すると，最下層から，各階層をログ変換処理，抽出処理，加工処理として捉えることができる．Layered Systems アーキテクチャの利点として，意味的に等価な Layer コンポーネントの交換が容易であることが挙げられる．しかし，加工処理では，各処理が出力するデータが異なることから，処理を追加するさい，Layer 間のプロトコルの変更を伴うと考えられる．また，Layered Systems アーキテクチャでは，処理の組み合わせを支援しない．

Pipes and Filters アーキテクチャ

Pipes and Filters アーキテクチャは，本 OJL で開発したログ解析支援ツールに適用したソフトウェアアーキテクチャである．ログ解析部分のソフトウェアアーキテクチャにこのアーキテクチャを適用すると，ログ変換処理，抽出処理，加工処理を Filter コンポーネントと捉えることができる．Pipes and Filters アーキテクチャの利点として，Filter コンポーネントの追加が容易であることや，コンポーネントの組み合わせが容易であることが挙げられる．欠点としては，Filter コンポーネントが処理をおこなう毎に段階的にデータストリームが複雑化することが挙げられる．

Pipes and Filters と Shared Data を組み合わせたアーキテクチャ

Pipes and Filters と Shared Data を組み合わせたアーキテクチャは，一般的にコンパイラ開発時に適用するソフトウェアアーキテクチャとして知られている．ログ解析部分のソフトウェアアーキテクチャにこのアーキテクチャを適用した場合，Pipes and Filters アーキテクチャと同様に，ログ変換処理，抽出処理，加工処理を Filter コンポーネントと捉えることができる．Pipes and Filters と Shared Data を組み合わせたアーキテクチャでは，Filter コンポーネント間のデータストリームの複雑化を軽減することができる．

比較結果

ソフトウェアアーキテクチャの比較結果を表 1 に示す．
上記の比較結果から，Layered Systems アーキテクチャはログ解析部分のソフトウェアアーキテクチャとして適切ではないと考える．Pipes and Filters アーキテクチャと Pipes and Filters と Shared Data を組み合わせたアーキテクチャは，処理の追加変更，処理の組み合わせに対して共に柔軟である．しかし，Pipes and Filters と Shared Data を組み合わせたアーキテクチャでは，データストリームの複雑化を軽減することと引き換えに，コンポーネント間で間接的な依存関係が発生する．Pipes and Filters アーキテクチャでは，コンポーネントが処理をおこなう毎に段階的にデータストリームが複雑化するこという問題があるが，ログ解析支援ツールで取り扱うデータがあまり複雑でないことから，この欠点が大きな問題とならないと考える．
ソフトウェアアーキテクチャの比較結果から，Pipes and Filters アーキテクチャを基にログ解析部分のソフトウェアアーキテクチャを構築したことは妥当であると考える．

表 1 ソフトウェアアーキテクチャの比較結果

	各処理の追加変更に対する柔軟性	各処理の組み合わせに対する柔軟性
Layered Systems	△	×
Pipes and Filters	○	○
Pipes and Filters + Shared Data	○	○

6 おわりに

本研究では，PLSE に基づくアーキテクチャを中心としたプロセスに基づきログ解析支援ツールの開発をおこなった．ログ解析支援ツールの設計と実現においては，ログ解析支援ツールの特性を考慮したソフトウェアアーキテクチャを構築することで，変更に柔軟に対応可能なツールを実現することができたと考える．
今後の課題として，本研究で開発したログ解析支援ツールの成果物のコア資産化が挙げられる．

参考文献

[1] AspectJ http://www.eclipse.org/aspectj/

[2] F.Buschmann,R.Meunier,H.Rohnert,P.Sommerlad ,M.Stal *Pattern-Oriented Software Architecture Volume 1: A System of Patterns*,John Wiley & Sons,1997

[3] L.M.Northrop “SEI’s Software Product Line Tenets”,*IEEE Software*,Vol19 No.4,pp.32-40,2007.

[4] M.Shaw,D.Garlan *Software Architecture: Perspectives on an Emerging Discipline*, Prentice-Hall, 1997

[5] 久野啓, 大澤範高, 弓場敏嗣 “並列プログラムのための視覚的性能デバッガ”, 電子情報通信学会技術研究報告.CPSY, コンピュータシステム,vol.94,No.383,pp.65-71.1994.

[6] 森本亮太, 若林隆行, 高田広章 “リアルタイムシステムのための正規表現を用いたログ検索システムの構築とその評価”, 情報処理学会研究報告.SLDM, システム LSI 設計技術,vol.2002,No.18,pp9-15,2002.