ポータブルミュージックプレーヤ制御ソフトウェアへの PLSE の適用

2003MT023 井戸田 典子　　2003MT057 水野 貴文　　2003MT076 成田 将友
指導教員　野呂 昌満

1　はじめに

メーカーの商品戦略によりポータブルミュージックプレーヤ (以下 PMP) の多種多様化が進み，新機能の追加や既存の機能の見直しが頻繁におこなわれている．PMP への機能の追加はハードウェアよりソフトウェアに依存した機器が多く，多様な要求に応えるには開発時間を短縮化しなければならない．

機能の追加や変更をともなうユーザ要求が頻繁に発生する．発生するたびに，初めからソフトウェアの開発をおこなうと開発時間が増加する．開発時間が増加することで，機能の追加や見直しを頻繁におこなうことが難しくなる．

開発時間の短縮化をはかるには製品に共通なアーキテクチャやコンポーネントをコア資産として再利用し，再利用部品をひとつの開発サイクルごとに洗練していくような系統的開発をおこなうことが必要と考える．系統的な開発を支援する技術のひとつに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)[1][2] がある．

本研究の目的は，PMP 制御ソフトウェアを事例として PLSE を適用し，フィーチャ図とアーキテクチャの関係を分析し，アーキテクチャの構築を容易にすることである．

本研究は以下のように進めた．

- PMP のユーザ要求を抽出し，フィーチャ図を作成
- オブジェクト指向をもちいて PMP 制御ソフトウェアのアーキテクチャを構築
- フィーチャ図とアーキテクチャの対応づけ
- 対応法則を PMP 制御ソフトウェアへ適用し，機能追加や変更を実現

2　PLSE

PLSE とはソフトウェア開発を系統的におこなうことを支援する技術のひとつである．その方法として，コア資産を開発し，共通のアーキテクチャのもと，それらを組み合わせて製品の開発をおこなう．コア資産とは，要求分析の結果，再利用可能部品およびアーキテクチャなどの開発に必要な全ての総称である．PLSE では以下の三点の考えを中心に開発をおこう．

- ドメイン工学
  －ユーザ要求の抽出，コア資産の開発
- アプリケーション工学
  －コア資産の統合による製品の開発
- 管理
  －コア資産の共有，管理

本研究ではフィーチャモデリングを適用してユーザ要求のモデル化をおこなう．フィーチャに基づく開発手法の代表的な考え方に FORM[2] がある．FORM のフィーチャとは，開発者がユーザ要求をシステムで実現可能なものと定義している．フィーチャ図は以下の 4 つの階層であらわしている．

- 特性層
  機能特性，非機能特性のつながりをあらわす
- 操作環境層
  ハードウェアに関する部分 (ボタンやディスプレイ) をあらわす
- ドメイン技術層
  ドメイン特有の技術をあらわす
- 実現技術層
  一般的で他のドメインでも利用できる技術をあらわす

3　PMP 制御ソフトウェアのユーザ要求分析

PMP 制御ソフトウェアへのユーザ要求の抽出をおこなう．ユーザ要求からフィーチャを抽出することができ，PMP 制御ソフトウェアを構成するフィーチャの関連をフィーチャ図を作成することでユーザ要求のモデル化をおこなう．

3.1　ユーザ要求の抽出

PMP へのユーザ要求の抽出をおこなった．必要不可欠な要求は音楽を携帯し再生することである．再生方法に対しても自分の好きな順番で再生したい，ランダムに再生したいなど要求も様々である．ハードウェアに対する要求も小型で携帯性に優れた機器がいい，記憶容量が多い機器がいいなどがある．

3.2　フィーチャ図の作成

ユーザ要求からフィーチャを抽出し，PMP 制御ソフトウェアを構成するフィーチャの関連をフィーチャ図をもちいて示す．フィーチャ図をもちいることでフィーチャの共通部と可変部を表すことができる．作成した PMP 制御ソフトウェアのフィーチャ図を**図 1** に示す．必須フィーチャを音楽関連機能とし，それ以外を付加的な機能とした．

4　PMP 制御ソフトウェアのアーキテクチャの構築

PMP 制御ソフトウェアをオブジェクト指向で開発することにより構造の整理をおこなう．開発する PMP の仕様を以下に示す．以下に示した機能が PMP 制御ソフト

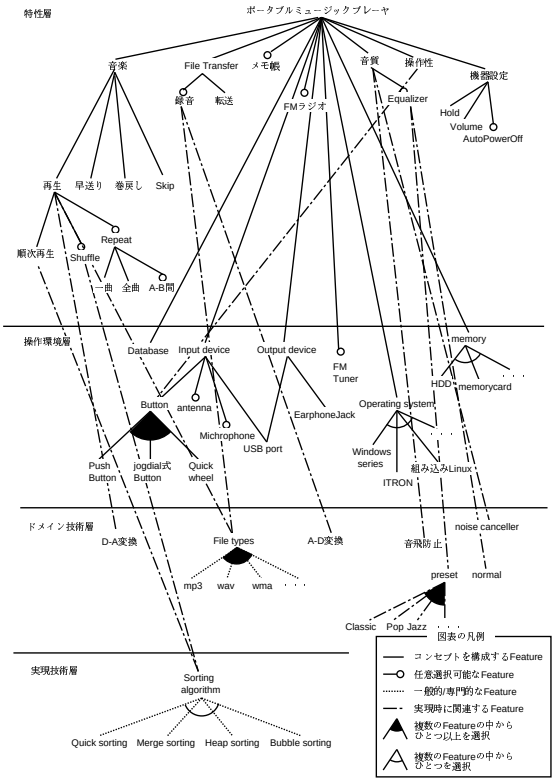

図 1: フィーチャ図

ウェアの必須機能である．

- 曲の再生，停止，早送り，巻戻し，スキップが可能
- 曲の転送には USB ポートを使用
- WAV ファイルの再生に対応

4.1 オブジェクト指向分析

PMP 制御ソフトウェアの構造を整理するために，オブジェクト指向分析をおこなった．PMP 制御ソフトウェアの必須機能を満たすには入力装置として Button，USBport を，出力装置として EarphoneJack，USBport が必要である．曲データを管理する Database，全体を管理する PMPlayer が必要と考えた．

4.2 オブジェクト指向設計

オブジェクト指向分析をもとに PMP 制御ソフトウェアのアーキテクチャを構築する．PMP 制御ソフトウェアのアーキテクチャの構築をおこなうさいに，以下のデザインパターンを適用した．

- Composite パターン
- Command パターン
- Visitor パターン

ファイルデータの検索方法にアーティスト名，アルバム名，およびタイトル名をキーワードとした．キーワードでの検索を迅速におこなうために，ファイルデータをアーティスト名，アルバム名，およびタイトル名と階層的に保持し，ファイルデータを木構造で管理した．再帰的な構造である木構造を容易に実現するためにデザインパターンのひとつである Composite パターンをもちいた．Command パターンによりデータベースへの要求をオブジェクトとしてカプセル化した．要求をカプセル化することで検索方法の追加や変更に柔軟に対応できると考えた．

PMP では種々のファイル形式を再生する事が可能であり，種々のファイル形式を扱うためにファイルを多態性で実現した．Visitor パターンをもちいてデータ構造と処理を分離した．分離することでファイルにおこなう処理の追加が容易になると考えた．

我々が構築したアーキテクチャを**図 2** に示す．**図 2** に示したアーキテクチャが PMP 制御ソフトウェアのコアアーキテクチャと考えた．

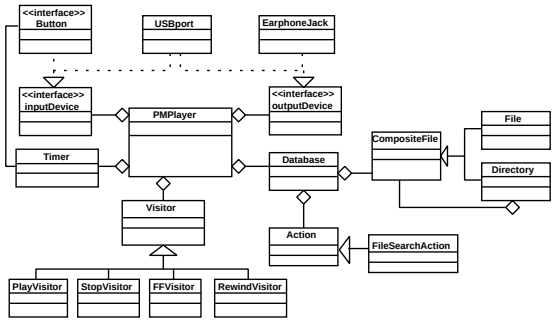

図 2: PMP 制御ソフトウェアのコアアーキテクチャ

5 アーキテクチャとフィーチャ図の対応関係

PMP 制御ソフトウェアのコアアーキテクチャとフィーチャ図の対応づけをおこなう．対応づけの結果からアーキテクチャとフィーチャ図の対応法則を考えた．

5.1 PMP 制御ソフトウェアのコアアーキテクチャとフィーチャ図の対応

オブジェクト指向分析をもとに構築した PMP 制御ソフトウェアのコアアーキテクチャとフィーチャ図の対応づけをおこなう．操作環境層に図示されている必須フィーチャは Operating System と memory を除き全てフィーチャ名とクラス名が 1 対 1 で対応している．例えばフィーチャ図の Input device のサブフィーチャはクラス図では Input Device クラスのサブクラスとして対応している．特性層の必須フィーチャである再生，早送り，巻戻しなどはそれぞれ Visitor クラスのサブクラスとして対応している．

5.2 アーキテクチャとフィーチャ図との対応法則

対応づけの結果をもとに PMP 制御ソフトウェアのアーキテクチャとフィーチャ図との対応関係に法則がないかを考えた．対応法則を考えるにあたり特性層のフィーチャと特性層以外のフィーチャでわけることができると考えた．

特性層の対応

特性層のフィーチャは機能特性，非機能特性を表しておりオブジェクト指向ではフィーチャとクラスは 1 対 1 で対応することができない．そこで我々はアスペクト指向を導入し特性層のフィーチャを機能アスペクトとして分離することを考えた．分離した機能アスペクトは**図 3** に

示すように，1 つの機能を実現するフィーチャアスペクトの集合と，それを管理するクラスを用意することで表すことができると考えた．新たにフィーチャを選択すると機能アスペクト内に選択したフィーチャアスペクトを追加し，管理するクラスのメソッドを変更することで対応する．PMP 制御ソフトウェアの機能アスペクトはデータベースに対する要求，データ構造に対する要求で分離できると考えられる．

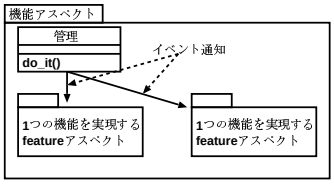

図 3: 特性層の機能追加の方法

特性層以外の対応

特性層以外のフィーチャの対応法則について考える．フィーチャ図では**図 4** にあるようにオプショナル，オルタネイティブ，およびオアの記号を使用するのでそれぞれについて考えた．

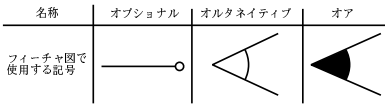

図 4: フィーチャ図で使用する記号

オプショナルフィーチャの追加は，追加するフィーチャと 1 対 1 で対応するひとつのクラスとして部品化できると考えた．

オルタネイティブフィーチャは**図 5** に示すように，フィーチャを実現するコンポーネントを入れ換えることで対応できると考えた．

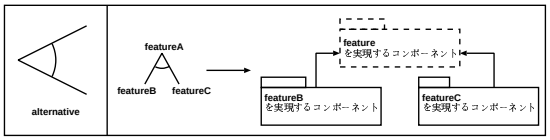

図 5: 特性層以外のオルタネイティブフィーチャの対応関係

オアフィーチャは**図 6** に示すように，ClassB を呼び出すのか，ClassC を呼び出すのか，それとも両方を呼び出すのか管理している Management クラスを用意する．選択されたフィーチャを実現するコンポーネントを Management クラスと has a の関係であるインターフェースのサブクラスとして追加していくことで対応できると考えた．

複数のフィーチャと関連があるフィーチャの対応

フィーチャ図に示したフィーチャを実現するさいに，**図 7** に示すように複数のフィーチャと関連があるフィーチャがある．フィーチャ A，フィーチャ B コンポーネントはフィーチャ C コンポーネントを共有している．一見するとフィーチャ A，フィーチャ B コンポーネントはそれぞれに独立してフィーチャ C コンポーネントがあ

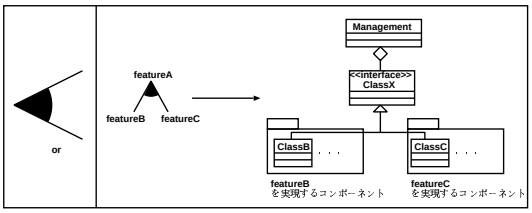

図 6: 特性層以外のオアフィーチャの対応関係

るように見える．実際はフィーチャ A，フィーチャ B コンポーネントが同時にフィーチャ C コンポーネントを共有している構造となっておりクロスカッティングしている．だからクロスカッティングしているフィーチャ C コンポーネントを抜き出すことでフィーチャとコンポーネントが 1 対 1 で対応できると考えた．

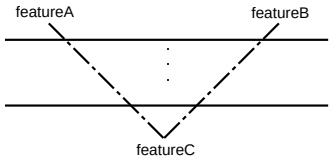

図 7: 上位フィーチャによる下位フィーチャの共有例

我々が考えた対応関係をオブジェクト指向設計した PMP 制御ソフトウェアのコアアーキテクチャに適用させる．適用したコアアーキテクチャを**図 8** に示す．

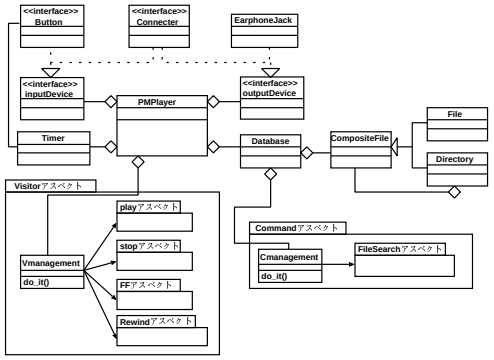

図 8: アーキテクチャの再構築

6 考察

PLSE を適用した PMP 制御ソフトウェアについて考察する．

6.1 機能の追加・変更についての考察

PMP 制御ソフトウェアへの機能追加・変更について考察する．機能アスペクトとして特性層のフィーチャを分離することにより各機能の追加時の変更箇所が少なくなる．変更箇所が少ないと，機能の追加・変更が容易であると考えられる．機能追加の例として Shuffle 機能の追加をあげる．Shuffle 機能はデータベースのファイル検索方法を新たに追加することで実現できると考えられる．追加・変更部分について**図 9** に示す．

Shuffle 機能の追加は**図 9** に示すように RandomeSearch アスペクトを追加し，Cmanagement クラスのメソッドの記述を変更することで追加ができる．機能を追加する

時に Cmanagement クラスを変更するだけで追加できるので，機能の追加が容易であると考えられる．

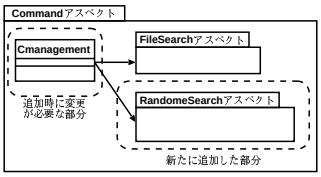

図 9: Shuffle 機能の追加によるアーキテクチャの変更部分

6.2 PLSE とアーキテクチャについての考察

我々が考えた対応法則が成り立つか対応法則をもとに PMP 制御ソフトウェアに機能追加をおこない，アーキテクチャを容易に構築することができるか考察する．オプショナルフィーチャ，オルタネイティブフィーチャ，およびオアフィーチャそれぞれについての考察をおこなう．

オプショナルフィーチャの例として，ハードウェアにディスプレイを追加し，ユーザが聞きたい曲をアーティスト名，アルバム名，およびタイトル名と階層的に選択することができる閲覧機能を追加する．新たなハードウェアの追加は特性層以外の対応法則に従って，ディスプレイを制御する Display クラスを Output Device クラスのサブクラスとして追加する．追加にともない，呼び出しコードを PMPlayer クラスに記述する．閲覧機能の追加はディスプレイに表示するデータの検索方法を追加することで実現できるので特性層の対応法則に従う．Command アスペクトに ReferenceDataSearch アスペクトを追加し，Cmanagement クラスのメソッドの記述を変更することで追加ができる．追加した様子を**図 10** に示す．オプショナルフィーチャの追加方法として，我々が考えた法則は成り立つ．

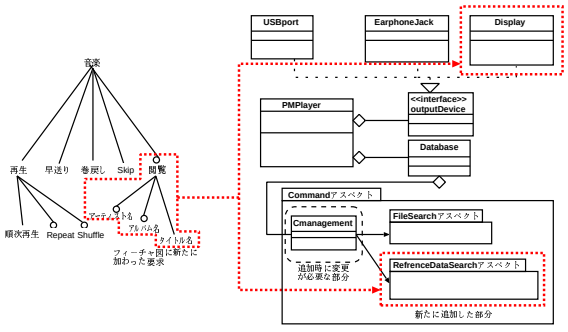

図 10: 閲覧機能の追加

オルタネイティブフィーチャの例として，Connecter の変更を考え，曲の転送を USB ポートからファイヤーワイヤ対応に変更する．Connecter の変更は特性層以外の対応法則に従って，USBport クラスを取り外し，取り外した箇所に FireWire クラスを取り付け，インスタンス生成をするコードを書き換える．変更した様子を**図 11** に示す．オルタネイティブフィーチャの変更方法も，我々が考えた法則は成り立つ．

オアフィーチャの例として，扱えるファイル形式を新たに追加する．ファイル形式を追加するには追加するファイル形式のクラスを用意し，File クラスのサブクラスとして追加していく．追加したファイル形式のクラスの再生方法などは Visitor アスペクト内の play アスペクト，stop アスペクトなどに記述する．追加した様子を**図 12** に示す．オアフィーチャの追加方法も，我々が考えた法則は成り立つ．

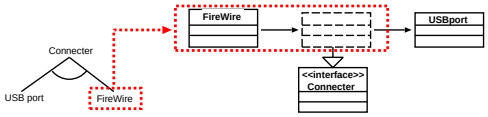

図 11: Connecter の変更

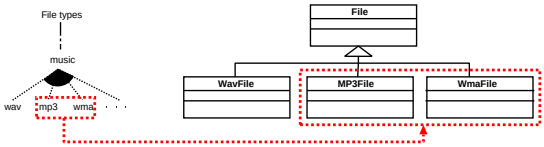

図 12: 新たなファイル形式の追加

機能追加をおこなった結果，我々が考えた対応法則は成り立つと予想できる．対応法則にもとづいてアーキテクチャを構築することができるので，新機能の追加や変更時にアーキテクチャの構築が容易になり，開発時間の短縮がはかれると考えられる．

7 おわりに

本研究ではフィーチャ図とコアアーキテクチャの対応づけの結果から対応法則を考えた．対応法則にもとづいて PMP 制御ソフトウェアに機能追加をおこない対応法則が成り立つことを確認し，アーキテクチャの構築が容易になるか考察した．

今後の課題として，本研究室で提案されている E-AOSAS++を適用しアーキテクチャの構築をすることで各部品の独立性や生産性をあげることがあげられる．

8 謝辞

本研究を進めるにあたり熱心な御指導をいただいた野呂昌満教授，有益なアドバイスを下さった張漢明先生，蜂巣吉成先生，大学院生のみなさまに深く感謝致します．また，いつも励まし合いがんばってきた野呂研究室，張研究室のみなさまに感謝致します．

参考文献

[1] Kyo C. Kang, Jaejoon Lee, and Patrick Donohoe, “Feature-Oriented Product Line Engineering”, *IEEE Software*, vol. 19, No. 4, pp. 58-62, 2002.

[2] Kyo C. Kang, Sajoong Kim, Jaejoon Lee, Kijoo Kim, Gerard Jounghyun Kim, and Euiseob Shin, “FORM: A Feature-Oriented Reuse Method with Domain-Specific Reference Architectures,” POSTECH, pp. 28, 1998.