携帯電話制御ソフトウェアのアスペクト指向実現

2001MT027 久松 康倫　　2001MT099 谷口 雄大
指導教員　野呂 昌満

1　はじめに

携帯電話制御ソフトウェア等の組込みソフトウェアは，製品の新規開発，改版が頻繁に行われるので，変更に対する柔軟性が要求される．一般のソフトウェア開発において，変更に柔軟に対応できる構造を構築する方法として，オブジェクト指向が注目されている．組込みソフトウェアは，オブジェクト指向を用いることで，柔軟な構造を構築できると考えた．

携帯電話制御ソフトウェアをオブジェクト指向で試作した結果，複数の横断的に関連する処理が存在することを確認した．組込みソフトウェアには，横断的な処理として，並行処理，およびイベント処理などがある．横断的に関連する処理が存在するソフトウェアは，変更に対する柔軟性が低い．

本研究の目的は，アスペクト指向 [5] で実現した組込みソフトウェアの変更に対する柔軟性を考察することである．組込みソフトウェアは複数のハードウェアを並行に処理する．携帯電話制御ソフトウェアは入力装置，出力装置など複数のハードウェアを並行に処理していると考え，組込みソフトウェアの典型的な例として題材とする．携帯電話制御ソフトウェアのアスペクト指向開発による，変更の柔軟性を考察することで，組込みソフトウェアの変更の柔軟性を考察できると考えた．

研究は以下のように進めた．

- 携帯電話制御ソフトウェアをオブジェクト指向で開発
- 実現したソフトウェアから横断的に関連した処理が存在することを確認
- 抽出した横断的に関連する処理をアスペクト指向を用いて分離
- アスペクト指向で実現した携帯電話制御ソフトウェアの，変更に対する柔軟性を考察

本研究室では，携帯電話制御ソフトウェアの各システムを分担して実現した．我々は入力装置，出力装置，および各グループが実現したサブシステムの統合を担当した．久松は主にサブシステムと状態遷移機械について，谷口は主に MVC について担当した．

2　携帯電話制御ソフトウェアのオブジェクト指向開発

横断的に関連する処理を確認するために，携帯電話制御ソフトウェアをオブジェクト指向により開発する．オブジェクト指向言語として，一般に広く使われている Java を使用した．

2.1　設計の指針

携帯電話は，複数のサブシステムが集まった構造をしている．我々の実現した携帯電話制御ソフトウェアのサブシステムを以下に列挙する．

- 通話システム
- メイルシステム
- Web 閲覧システム
- ファイルシステム
- アドレス帳システム
- カメラシステム
- テキスト編集システム
- 設定システム

携帯電話は入力装置と出力装置が存在する．入力装置はユーザ，サーバからの入力によってイベントを発生させる．イベントが発生すると，イベントに対する処理が行われ，出力装置からユーザ，サーバへの通知が起こる．

2.2　設計と実現

MVC モデル

GUI 構造を持つソフトウェアでは，MVC モデル [4] が有効だと言われている．携帯電話は入力装置，出力装置，およびアプリケーション実行部を持っている．携帯電話制御ソフトウェアは，MVC モデルを適用し，実現した．入力，出力，アプリケーション実行部は，分離され独立性が向上する．MVC の分類を表 1 に示す．

Controller	InputDevice
View	OutputDevice
Model	CellularPhone, 各サブシステム Timer，DataBase

表 1　携帯電話制御ソフトウェアの MVC モデル

サブシステム

サブシステムは共通の構造にすることで，開発を省力化できる．サブシステムには，複数の状態が存在する．状態によって異なる振舞いを行うので，状態遷移機械で実現した．状態遷移機械は，イベントに対して，状態の遷移と，状態によって異なる振舞いを行う．状態，および振舞いの管理を容易にするために，デザインパターン [2] のひとつである State パターン，および Command パターンを用いた．サブシステムは同時に起動するので，

並行状態遷移機械として実現する．サブシステムの管理は CellularPhone クラスで行う．

実現した携帯電話制御ソフトウェアのクラス図を図 1 に示す．

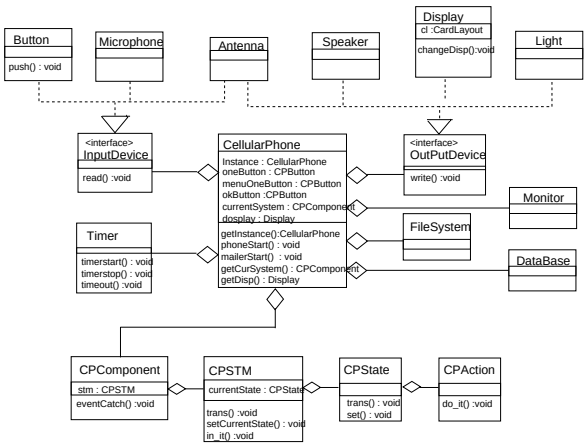

図 1　携帯電話制御ソフトウェアのクラス図

3　携帯電話制御ソフトウェアのアスペクト指向開発

オブジェクト指向で実現した携帯電話制御ソフトウェアからコンサーンを抽出し，アスペクト指向を適用して実現した．コンサーンアスペクト指向言語として java のアスペクト指向拡張言語 AspectJ を使用した．

3.1　コンサーンの抽出

オブジェクト指向で実現した携帯電話制御ソフトウェアから，横断的に関連する処理をコンサーンとして抽出した．一方で，携帯電話は協調場 [3] によって振舞いが異なる．協調場をコンフィグレーションコンサーンとして抽出することで，構造を整理できると考えた．

- コアコンサーン
  - 状態遷移コンサーン
  - アプリケーションロジックコンサーン
- イベント処理コンサーン
- 外部出力コンサーン
- 並行処理コンサーン
- 例外処理コンサーン
- 実時間処理コンサーン
- メモリストレージコンサーン
- データ永続性コンサーン
- 排他制御コンサーン
- カスタマイズコンサーン
- セキュリティ処理コンサーン
- インスタンス生成コンサーン
- コンフィグレーションコンサーン

3.2　コンサーンの詳細

全てのシステムに関係するコンサーンは，イベント処理，並行処理，外部出力，状態遷移，アプリケーションロジック, コンフィグレーションである．

コアコンサーン

MVC モデルを適用した携帯電話制御ソフトウェアにおいて，Model をコアコンサーンとした．コアコンサーンは各サブシステムを構成する状態遷移機械を含む．状態遷移機械は，以下の二つのコンサーンで分離できると考えた．

- **状態遷移コンサーン**
- **アプリケーションロジックコンサーン**

状態遷移機械は，イベントを受け取ったとき，状態の遷移と状態によって異なる振舞いを行う．状態遷移機械は，イベントの発生に対して，異なる二つの処理を行っていると考えられる．状態遷移を状態遷移コンサーン，振舞いをアプリケーションロジックコンサーンによって分離する．状態遷移と振舞いはモジュール化されるので，独立性は向上する．状態遷移機械の構造は整理される．

イベント処理コンサーン

イベント処理はイベントの発生するオブジェクトに横断的に関連している．MVC モデルを適用した携帯電話制御ソフトウェアにおいて，Contrlor はイベントが発生したとき，発生したイベントを Model に通知する必要がある．Controller が Model にイベントを通知する処理は，横断的に関連する処理として確認した．携帯電話制御ソフトウェアは，Controller である InputDevice 以外にも，Timer や状態遷移機械によって，イベントを発生させることが考えられる．Timer や状態遷移機械が，イベントを通知する処理は横断的に関連する処理として確認した．

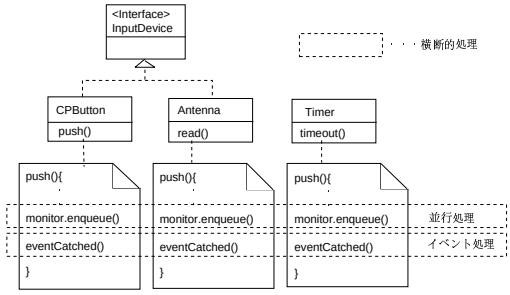

図 2　イベント処理と並行処理と横断的に関する処理

並行処理コンサーン

並行処理は，イベントを発生させるオブジェクトに横断的に関連していた．並行実行モニタは，ボタンからの入力と着信が同時に起こるなどの，イベントが複数発生したとき，イベントを順に処理する．イベントは優先度があり，着信やメイル受信のイベントは優先的に処理される．横断的に関連する処理を図 2 に示す．

外部出力コンサーン

外部出力処理はサブシステムの振舞いに横断的に関連している．MVC モデルを適用した携帯電話制御ソフトウェアにおいて，出力が必要なとき Model から View への，出力要求が必要となる．サブシステムの振舞いにおいて，画面の変更や効果音など，外部出力に関する処理が確認された (図 3 参照)．

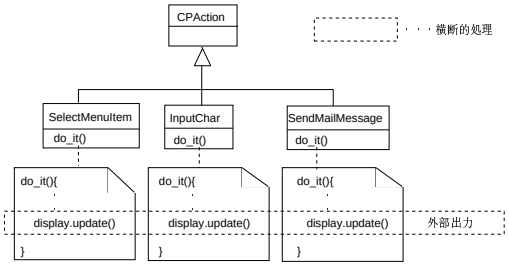

図 3　外部出力処理

アスペクトの構成するオブジェクトを図 4 に示す．

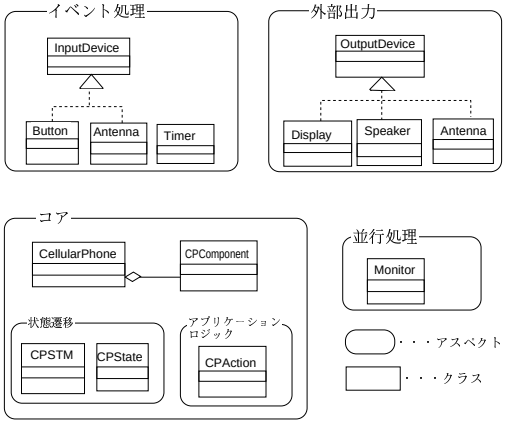

図 4　各コンサーンによるオブジェクトの分離

コンフィグレーションコンサーン

各サブシステムを構成する複数のオブジェクトは，協調場を定義するとものと考え，アスペクトとして分離する．ボタンや，ディスプレイは，サブシステムによって，役割が異なる．オブジェクトを役割ごとに分割し，その関係をモデル化したものを協調場と呼ぶ．協調場によって，サブシステムを分割することで，各サブシステムの独立性は向上する． 各サブシステムの構成要素を図 5 に示す．

4　実現と考察

オブジェクト指向とアスペクト指向で実現した携帯電話制御ソフトウェアを比較し，ソフトウェアの変更や追加に対する柔軟性について考察する．

4.1　MVC モデルに関する考察

今回オブジェクト指向実現した携帯電話制御ソフトウェアは，MVC にしたがった構造であるので，Controller から Model への通知であるメッセージ通信が必要である．本研究では InputDevice を Java のイベントモデルを適用し，InputDevice 自身をイベントリスナとして実現した．イベントモデルは，メッセージ先として Model を知る必要があり，InputDevice は CellularPhone へのメッセージ通信の記述が存在した． InputDevice であるボタンを取り換えによって，Button クラスが変更した場合を考える．変更された Button クラスは，CellularPhoen へのメッセージ通信を記述する必要がある (図 6 参照)．

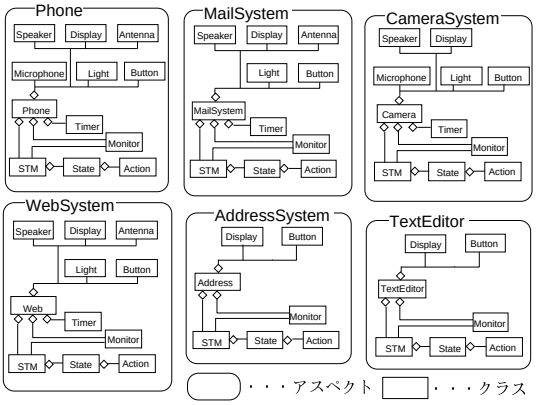

図 5　各サブシステムの構成

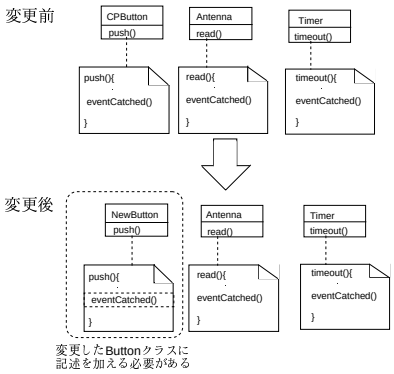

図 6　オブジェクト指向開発による InputDevice の追加

アスペクト指向実現により，Controller と Model 間の関係をアスペクト間記述で局所化することができた．Button クラスを変更した場合を考える．変更点はアスペクト間記述に局所化されている (図 7 参照)．オブジェクト指向と比較すると，変更に対する柔軟性が向上したと考えられる．

MVC モデルは，Model，View，および Controller 間の関係をアスペクト指向を用いることで，独立性が向上することが確認できた．Controller から Model への通知の関係をアスペクト間記述に局所化したのと同様にして，それぞれの関係を局所化することで，各モジュールの独立性は向上する．MVC モデルに適用した他の組込

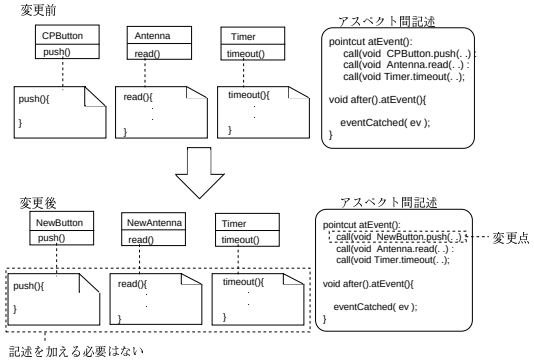

図 7　アスペクト指向開発による Inputdevice の追加

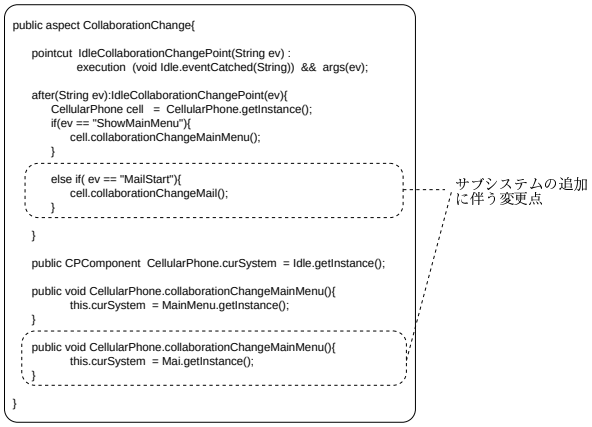

図 9　メイルコンフィグレーションの追加

みソフトウェアについても，アスペクト指向を適用することで，構造が整理できると考えた．

4.2　サブシステムに関する考察

サブシステムの追加がおこったときの，変更に対する柔軟性を考察する．オブジェクト指向開発では，現在起動しているサブシステムの管理は CellularPhone クラスで行っている．サブシステムの切替えは，現在起動しているサブシステムが，別のサブシステムに切り替わるための処理を持つ必要がある．新たにサブシステムを追加する場合は，関連を持つサブクラスと CellularPhone クラスに変更が及ぶ．サブシステム同士が関連し合うので，CellularPhone クラスで一元管理するのは，困難である (図 8 参照)．

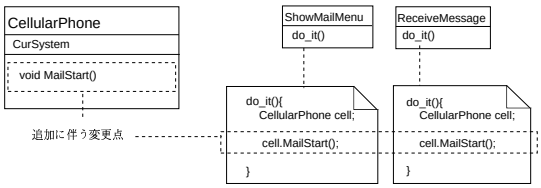

図 8　オブジェクト指向開発でのサブシステムの切換え

アスペクト指向開発では，サブシステムをアスペクトとしてモジュール分割して，それぞれ独立した構造とする．切り替えの処理は，アスペクト間記述で実現する．サブシステムの切り換えのための管理について，局所化できた．新たにサブシステムを追加する場合は，図 9 のように記述する．オブジェクト指向において，各サブシステム間に関連し合う処理がアスペクト指向実現で，分離，局所化できたので，柔軟性が向上したと言える．

5　おわりに

本研究では，オブジェクト指向を用いて携帯電話制御ソフトウェアを設計，実現した．実現したソフトウェアから横断的に関連する処理を確認し，アスペクト指向で分離した．システムの変更や追加に対する柔軟性の考察をおこなった．

携帯電話制御ソフトウェアは，入力装置，出力装置などの複数のハードウェアを並行に処理していた．組込みソフトウェアを，複数のハードウェアを同時に処理するものと考えると，携帯電話制御ソフトウェアは典型的な組込みソフトウェアであった．本研究で実現した携帯電話制御ソフトウェアを抽象化することで，組込みソフトウェアのアーキテクチャを実現できる考えた．以上を今後の課題としてあげる．

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熊崎敦司先生，大学院生の後藤修平さん，石見知也さん，小久保佳将さん，八木晴信さんに深く感謝いたします．また，いつも励まし合いがんばってきた野呂研究室のみなさんに感謝致します．

参考文献

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

[2] Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides : Design Patterns Elements of Reusable Object-Oriented Software, Addison-Wesley,(1995).

[3] T.Tamai: Evolvable Programming based on Collaboration-Field and Role Model, International Workshop on Principles of Software Evolution,(2002).

[4] 青木 淳:〜オブジェクト指向の基礎から解説する〜オブジェクト指向システム分析設計入門, ソフト・リサーチ・センター,(1993).

[5] COMMUNICATION of the ACM, Vol.44, No.10, P168,(2001).