携帯電話制御ソフトウェアの構造に関する考察

2002MT032 加藤 麻希　　2002MT081 鈴木 麻紀子
指導教員　野呂 昌満

1　はじめに

多機能化が進み，構造が複雑化する携帯電話制御ソフトウェアの実現には，非機能特性としてコンフィギュレーションコントロール (以下，CC) を考慮する必要がある．携帯電話制御ソフトウェアは，構造の複雑化により，機能を複数のサブシステムに分割して管理している．携帯電話制御ソフトウェアは，起動するサブシステム毎にコンフィギュレーションを変える必要がある．CC は，携帯電話制御ソフトウェアのハード制約にも対応できる．

本研究室では，組み込みソフトウェアの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) を提案している．E-AOSAS では，組み込みソフトウェアは並行に動作する状態遷移機械の集合であるとする構造を規定している．状態遷移機械は並行処理アスペクト，状態遷移アスペクト，アプリケーションロジックアスペクトから構成されている．

E-AOSAS では，CC に関する処理をアスペクトとして分離する仕組みが提供されていないという問題点がある．E-AOSAS にしたがい，携帯電話制御ソフトウェアのアーキテクチャを構築し，携帯電話制御ソフトウェアを実現した．携帯電話制御ソフトウェアを実現したところ，CC に関する処理が複数のオブジェクトに横断的に関連していることを確認した．横断的に関連する処理が存在すると，追加・変更に関する柔軟性が低い．柔軟性を確保するために CC に関する処理をアスペクトとして分離する必要がある．他の組み込みソフトウェアにも CC に関する処理があると予想されるので，E-AOSAS に CC に関する処理をアスペクトとして分離する仕組みを取り入れる必要があると考えた．

本研究の目的は，CC を非機能特性として考慮した E-AOSAS+ を構築し，応用可能性について考察することである．E-AOSAS+ を構築することにより，他のドメインのアーキテクチャ構築時に，コンフィギュレーションを適正に管理することができる．

研究手順を以下に示す．

- E-AOSAS にもとづいた携帯電話制御ソフトウェアのアーキテクチャの構築
- 携帯電話制御ソフトウェアのアーキテクチャを再構築
- 再構築したアーキテクチャを一般化し，E-AOSAS+ を構築
- 構築した E-AOSAS+ の応用可能性について考察

本研究では，コンフィギュレーションを適正に管理することができる E-AOSAS+ を構築することができた．構築した E-AOSAS+ にもとづいて構築したアーキテクチャを用いることで，コンフィギュレーションを適正に管理するソフトウェアを実現することができる．

加藤は主に E-AOSAS+ の構築を，鈴木は主に CC の実現を担当した．

2　組み込みソフトウェアのアスペクト指向ソフトウェアアーキテクチャスタイル (E-AOSAS)

E-AOSAS では，組み込みソフトウェアのアーキテクチャは並行に動作する複数の状態遷移機械の集合であるとしている．複数の状態遷移機械は，互いにメッセージを送り合い，並行に動作する．並行処理実体は状態遷移機械である．状態遷移機械には待機状態および活性状態があり，システム起動時には待機状態にある．待機状態にある状態遷移機械は，受理できるメッセージが到着すると，活性状態になり，処理を開始する．処理を終了した状態遷移機械は，再び待機状態に戻る．

状態遷移機械は以下のコンサーンにもとづいて分離したアスペクトで構成されている．

- 主要コンサーン
  - 並行処理コンサーン
- 二次的コンサーン
  - 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

並行に動作する状態遷移機械は，並行処理に関する処理に状態遷移機械に関する処理が横断的に関連する．組み込みソフトウェアはハードウェアが並行に動作することから，並行処理を主要コンサーンとして，並行処理と状態遷移機械を分離した．状態遷移機械では，状態遷移に関する処理とアプリケーションロジックに関する処理が横断的に関連する．状態遷移に関する処理と，アプリケーションロジックに関する処理の再利用性を高めるために，状態遷移とアプリケーションロジックを分離した．

E-AOSAS を図 **1** に示す．アーキテクチャの記述には，本研究室で提案されているアスペクト指向ソフトウェアアーキテクチャの図式表現 [5] を用いた．

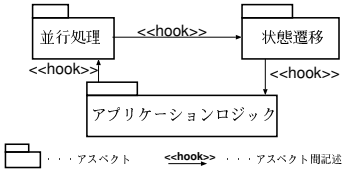

図 1　E-AOSAS

3　携帯電話制御ソフトウェアのアーキテクチャの構築

E-AOSAS にもとづいた携帯電話制御ソフトウェアのアーキテクチャを構築する．携帯電話制御ソフトウェアのボタンシステムは，メモリ制約や実時間制約により，

コンフィギュレーションを管理しなければならない典型的な例である．ボタンシステムからの入力による CC を例に取り，アーキテクチャを構築する．

3.1 コンフィギュレーションコントロール

コンフィギュレーションの切替えは，ボタンからのイベントがトリガーとなるので，状態遷移機械を使って CC を実現する．CC に関する状態遷移をメタ状態遷移，各サブシステムでの状態遷移をベース状態遷移と呼ぶ．

CC を実現する方法として，メタ状態遷移とベース状態遷移を一つの状態遷移機械で管理する方法を考える．この場合，メタ状態遷移とベース状態遷移が混在する構造となり，状態遷移の管理が困難になる (**図 2** 参照)．サブシステムを追加すると，状態遷移の変更が複雑になり，複数のクラスで変更が必要になるので，追加・変更に関する柔軟性が低い．

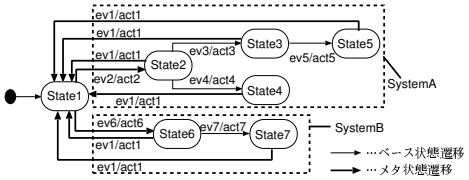

図 2　一つの状態遷移機械による CC の実現

CC を実現する方法として，メタ状態遷移とベース状態遷移を別の状態遷移機械に分け，メタ状態遷移機械でベース状態遷移機械を切替える方法を考える．状態遷移機械を分ける場合，メタ状態遷移とベース状態遷移が分割され，状態遷移の管理が容易になる (**図 3** 参照)．サブシステムを追加すると，状態遷移の変更は局所化される．状態遷移の変更による変更箇所も局所化されるので，追加・変更に関する柔軟性が高い．本研究では，状態遷移機械を分けてコンフィギュレーションを管理する方法を状態遷移機械の階層化を用いて図示する．

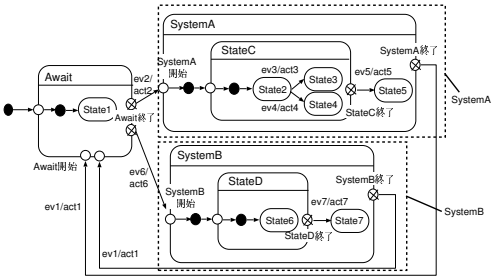

図 3　状態遷移機械の分割による CC の実現

本研究では，追加・変更に関する柔軟性の高い，メタ状態遷移とベース状態遷移を別の状態遷移機械に分ける方法で CC を実現する．

3.2 CC の実現

メタ状態遷移とベース状態遷移を別の状態遷移機械に分ける方法を用いて，CC を実現する．例として，携帯電話制御ソフトウェアのボタンシステムをあげる．ボタンには，複数のコンフィギュレーションが存在し，コンフィギュレーション毎に状態遷移機械が存在する．ボタンのコンフィギュレーションが切替わる様子を**図 4** に示す．

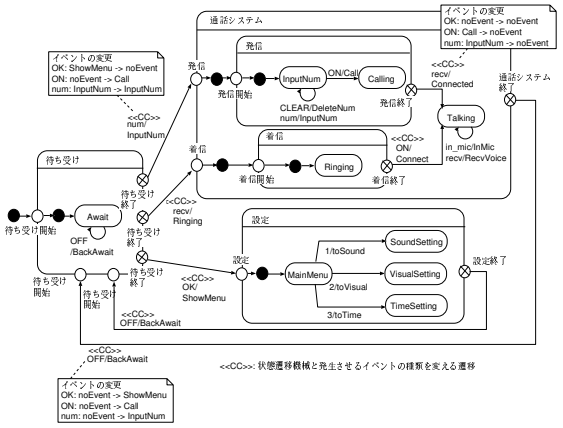

図 4　ボタンのステートマシン図

ボタンが待ち受け状態のさいに番号ボタンが押されると，ボタンのコンフィギュレーションが発信用に切替わる．状態遷移機械と発生させるイベントを発信用に切替えることで，ボタンのコンフィギュレーションを切替える．

3.3 E-AOSAS にもとづいた携帯電話制御ソフトウェアのアーキテクチャの構築

CC を取り入れた，携帯電話制御ソフトウェアのアーキテクチャを構築する．構築した携帯電話制御ソフトウェアのアーキテクチャを**図 5** に示す．

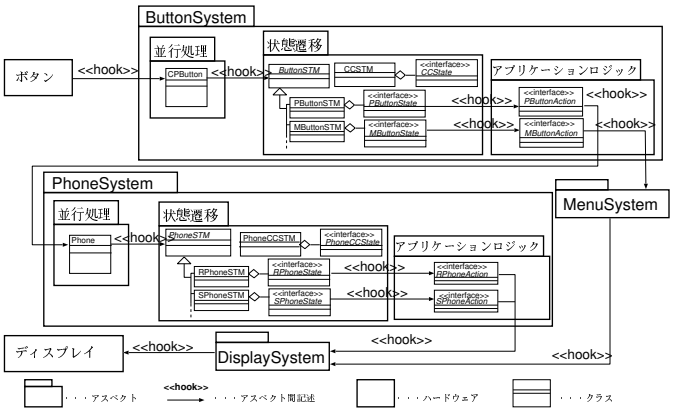

図 5　携帯電話制御ソフトウェアのアーキテクチャ

構築したアーキテクチャは通話システムなどのサブシステムと，ボタンやディスプレイなどのハードウェアから構成され，それらが並行処理実体となる．全てのサブシステムとハードウェアは，状態遷移機械を持ち，並行に動作する．

4 携帯電話制御ソフトウェアのアーキテクチャの再構築

4.1 構築したアーキテクチャを用いた実現

構築したアーキテクチャを用いて携帯電話制御ソフトウェアを実現した．実現した携帯電話制御ソフトウェアにおいて，CC に関する処理が横断的に関連していることを確認した．CC に関する処理が横断的に関連する様

子を図 **6** に示す．

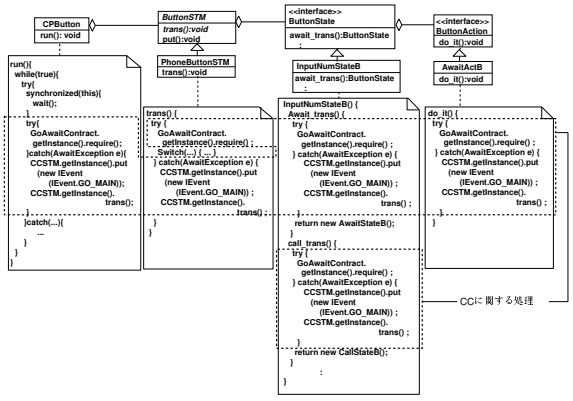

図 6　CC の横断問題

CC に関する処理が，CPButton および，ButtonSTM，ButtonState，ButtonAction の各サブクラスに横断的に関連している．

CPButton および，ButtonSTM，ButtonState，ButtonAction の各サブクラスから，横断的に関連している処理をアスペクトとして分離した．分離した CC に関する処理はアスペクト間記述に局所化した．実現したアスペクト間記述を図 **7** に示す．

```
public aspect ConfigAspect {

  pointcut atAwaitContract1() :
                execution(void PhoneButtonSTM.trans(..)) ;
                        :

  void around() : atAwaitContract1() {
     try {
        GoAwaitContract.getInstance().require() ;
        proceed() ;
     } catch(AwaitException e) {
        CCSTM.getInstance().put(new IEvent(IEvent.GO_MAIN)) ;
        CCSTM.getInstance().trans() ;
     }
  }
                        :
}
```

図 7　CC に関するアスペクト間記述

CC に関する処理をアスペクトとしてモジュール化した．

4.2　CC を考慮した携帯電話制御ソフトウェアのアーキテクチャの再構築

CC を非機能特性として考慮し，携帯電話制御ソフトウェアアーキテクチャを再構築した．再構築したアーキテクチャを図 **8** に示す．

携帯電話制御ソフトウェアは図 **8** のように CC に関する処理をアスペクトとして取り扱うことができる．CC に関する処理は全てのサブシステムとハードウェアで必要な処理である．CC に関する処理をアスペクトとすることで再構築したアーキテクチャを用いて実現した携帯電話制御ソフトウェアの CC に関する追加・変更の柔軟性が向上すると考えられる．

5　E-AOSAS+ の構築

再構築した携帯電話制御ソフトウェアのアーキテクチャを，アスペクト間の関連の共通部分を抜き出すことで一般化し，E-AOSAS+ を構築した．構築した E-AOSAS+ は，E-AOSAS に二次的コンサーンとして CC コンサーンを追加し，アスペクトとして分離した構造になる．構築した E-AOSAS+ を図 **9** に示す．

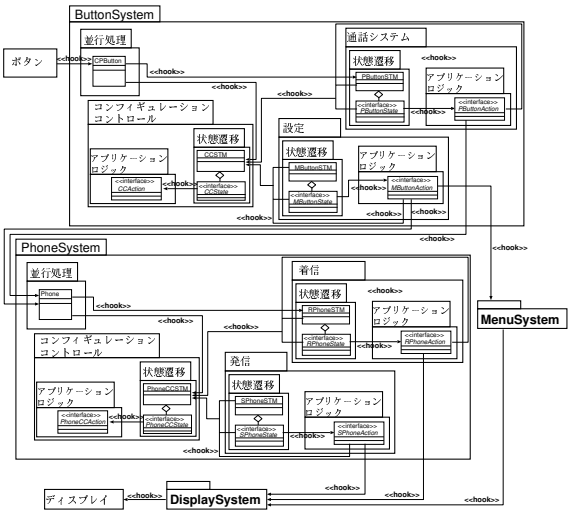

図 8　再構築した携帯電話制御ソフトウェアのアーキテクチャ

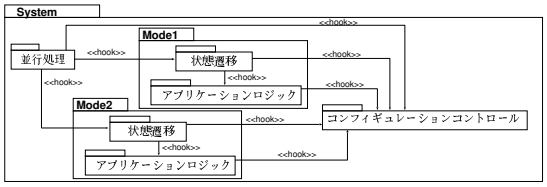

図 9　構築した E-AOSAS+

構築した E-AOSAS+ は，組み込みソフトウェアのアーキテクチャ構築において応用可能であると考える．具体的な適用例は考察で述べる．

6　考察

構築した E-AOSAS+ の応用可能性について以下の観点から考察する．

- 携帯電話制御ソフトウェアの他のシステムに E-AOSAS+ を適用
  - 通話受信用アンテナ (Receiver) を例にして
- 他の応用領域に E-AOSAS+ を適用
  - 腕時計制御ソフトウェアを例にして

6.1　携帯電話制御ソフトウェアの Receiver を例とした考察

本研究で構築した E-AOSAS+ は，携帯電話制御ソフトウェアの Receiver のアーキテクチャにも応用できることを確認した (図 **10** 参照)．Receiver とは，通話システム起動時にデータの受信をおこなうものである．Receiver が待ち受け状態の時にデータを受信した場合，Receiver は着信を知らせる役割りをする．通話中に Receiver がデータを受信した場合には，Receiver は通話相手の音声データ受信の役割りをする．Receiver のステートマシン図 [6] を図 **11** に示す．

E-AOSAS+ にもとづいた Receiver のアーキテクチャでは，CC に関する処理を適正に扱うことができた．構築した E-AOSAS+ は，携帯電話制御ソフトウェアの他のソフトウェアのアーキテクチャ構築においても応用可

能性が高いと考えられる．

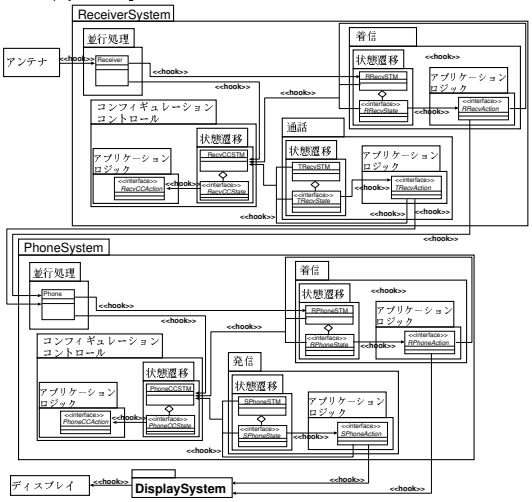

図 10　Receiver のアーキテクチャ

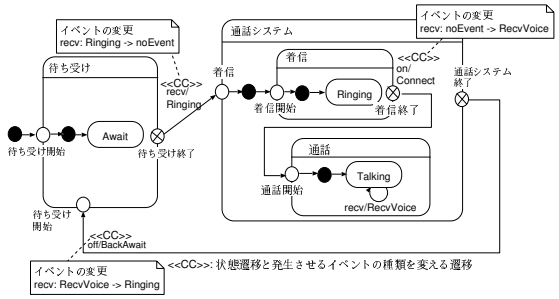

図 11　Receiver のステートマシン図

6.2　腕時計制御ソフトウェアを例とした考察

本研究で構築した E-AOSAS+ は，腕時計制御ソフトウェアのアーキテクチャにも応用できることを確認した．腕時計制御ソフトウェアとは，ストップウォッチ，アラームなどの機能を持つ腕時計を制御するソフトウェアである．腕時計制御ソフトウェアは，入力装置としてボタンシステムを，出力装置としてディスプレイシステムを管理する．サブシステムとしては，時計システム，ストップウォッチシステム，アラームシステムなどを管理する．

腕時計制御ソフトウェアのボタンは，ストップウォッチやアラームなどのコンフィギュレーションに対応する必要がある．ボタンはストップウォッチシステム起動中には計測開始や停止の役割りをし，アラームシステム起動時には時刻設定の役割りをする．構築した腕時計制御ソフトウェアのアーキテクチャを**図 12** に示す．
E-AOSAS+ にもとづいた腕時計制御ソフトウェアのアーキテクチャでは，CC に関する処理を適正に扱うことができた．構築した E-AOSAS+ は他のドメイン向けのアーキテクチャの構築においても応用可能性が高いと考えられる．

7　おわりに

本研究では，携帯電話制御ソフトウェアのアーキテクチャに CC アスペクトを取り入れ，E-AOSAS+ を構築した．構築した E-AOSAS+ の応用可能性について考察した．

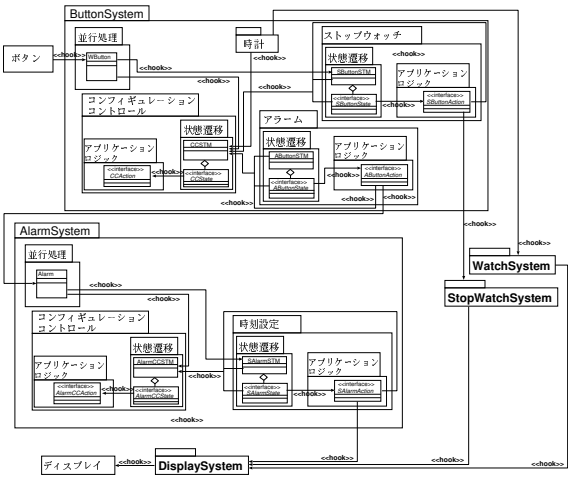

図 12　腕時計制御ソフトウェアのアーキテクチャ

携帯電話の通信処理ではエラーが発生しやすいことから，携帯電話制御ソフトウェアにはフォールトトレランスに関する処理が必要であると考えられる．今後の課題としては，携帯電話制御ソフトウェアのアーキテクチャにフォールトトレランスアスペクトを取り入れることがあげられる．

CC の実現を考える上で，コンフィギュレーションのコンテキストの保持を考える必要もある．例えば，メイル作成中に着信があった場合，コンフィギュレーションを着信用に切替え，通話終了後に再度コンフィギュレーションをメイル作成用に戻す必要がある．

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張漢明助教授，大学院生の石見知也さん，小久保佳将さん，八木晴信さん，石川智子さん，坂野将秀さん，久松康倫さん，本多克典さん，水野耕太さんに深く感謝いたします．

参考文献

[1] 安形 知美，本多 克典，伊藤 裕康，緒方 秀次，“携帯電話における通信ソフトウェアのアスペクト指向実現，“ 南山大学数理情報学部情報通信学科卒業論文要旨集，p．124-127，2004.

[2] AspectJ．http://eclipse.org/aspectj/, Dec．2005.

[3] E．Gamma, J．Vlissides, R．Helm, and R．Johnson, *Design Patterns Elements of Reusable Object-Oriented Software*, Addison-Wesley，p．249，p．325，1995.

[4] Java．http://java.sun.com/, Dec．2005.

[5] 森貴彦，“アスペクト指向ソフトウェアアーキテクチャの図式表現に関する研究，” 南山大学大学院経営学研究科経営学専攻修士論文，p．67，2004.

[6] UML．http.//www.uml.org/，Dec．2005.