アスペクト指向に基づく飛行船制御ソフトウェア開発

2005MT008 江場 智文　　2005MT086 大石 早織
指導教員　沢田 篤史

1　はじめに

ソフトウェア開発では，開発効率や再利用性の面からソフトウェアアーキテクチャを定義し，それに基づいて製品を構築することが一般に行われる．アーキテクチャに基づくソフトウェア開発では，仕様モデルから系統的にアーキテクチャを構築することで開発効率を高める．しかし，仕様モデルとアーキテクチャの対応関係が不明確であり，系統的にアーキテクチャを構築することができない．また，アスペクト指向アーキテクチャ構築において，横断的コンサーンの系統的な抽出が困難である．

仕様モデルに対する分析方法の 1 つにロバストネス分析 [1] がある．ロバストネス分析は，仕様モデルから抽出したオブジェクトを 3 種類に分類することで，アーキテクチャ構築の基準を与える．また，アスペクト指向アーキテクチャでは，オブジェクト指向では分離できない横断的コンサーンをアスペクトとして記述し，コンサーンを分離することで，ソフトウェアの再利用性や保守性を高める．我々は，ロバストネス分析の結果から横断的コンサーンを発見することが可能であると考える．

本研究は仕様モデルとアスペクト指向アーキテクチャの対応関係を明確にすることを目的とする．ロバストネス分析を仕様モデルに対する分析とし，ロバストネス分析の結果とアスペクトの対応関係の仮説を立て，飛行船制御ソフトウェアを事例として仮説の検証を行う．また，仮説に基づいて設計したアーキテクチャに Model-View-Controller(MVC[2]) を適用したアスペクト指向アーキテクチャと，MVC を用いたオブジェクト指向アーキテクチャに基づいて設計したアスペクト指向アーキテクチャを比較することで，本手法の有効性を考察する．ロバストネス分析の結果とアスペクトの対応関係を明確にすることで，仕様モデルからアーキテクチャを設計する手順が明確になり，より系統的なアスペクト抽出が可能になる．また，アスペクト指向アーキテクチャの記述には，本研究室で提案する組込みソフトウェアのためのアスペクト指向アーキテクチャスタイル (以下，E-AoSAS++) を用いることとする．

2　背景技術

2.1　ロバストネス分析

ロバストネス分析とは，オブジェクト指向の分析方法であり，ユースケースに記述された機能を実現するために必要な要素とその役割を整理し，アーキテクチャ設計の基準を与えることを目的とする．ユースケース記述中のオブジェクト群を次の 3 種類に分類し，分類したオブジェクト間の関連を定義する．

- バウンダリオブジェクト
  システム外部との通信に使うオブジェクト
- エンティティオブジェクト
  システム内部で半永久的に管理するデータとその振る舞いを担うオブジェクト
- コントロールオブジェクト
  システムの振る舞いを担うオブジェクト

ユースケースごとに，3 種類のオブジェクトとユースケースに登場する外部実体 (アクター) で構成されるロバストネス図を作成する (図 1)．

図 1　ロバストネス図の記号

2.2　MVC

MVC はソフトウェアをモデル・ビュー・コントローラの 3 つに分割し設計・実現を行う手法である [2]．それぞれの責務は次のように定義されている．

- モデル
  アプリケーションの中核機能とデータに関する処理
- ビュー
  ユーザへの情報の表示
- コントローラ
  ユーザ入力をイベントとして受け取り Model，View へのリクエストに変換する

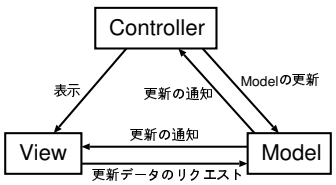

図 2　MVC の処理の流れ

MVC の処理の流れを図 2 に示す．Controller は外部からの入力を受け取り，Model を呼び出す．Model は内部データを更新し，View と Controller に対してデータ更新の通知を送る．View は更新データを Model にリクエストし，再表示を行う．この処理の流れを繰り返すことでソフトウェアの振る舞いを実現する．

2.3　E-AoSAS++

E-AoSAS++ では，組込みソフトウェアを並行動作する状態遷移機械 (CSTM) の集合と規定しており，互いにメッセージを送ることで協調動作を実現する [3]．図 3 に E-AoSAS++ で規定する組込みソフトウェアの概念を示す．

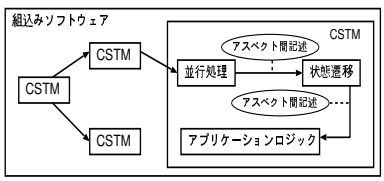

図 3　E-AoSAS++ が規定する組込みソフトウェアの概念

3　ロバストネス分析の結果とアスペクトの対応関係

仕様モデルに対する分析方法の 1 つであるロバストネス分析の結果と，アスペクトの対応関係を明確にすることによって，仕様モデルとアスペクト指向アーキテクチャの対応関係を明確にする．また，ロバストネス分析を用いた仕様モデルからのアスペクト抽出を可能にする．

ロバストネス分析の結果，図 4 の左に示すように，ロバストネス図において 1 つのバウンダリに対して複数のコントロールとの関連を持つ場合がある．これは，1 つのバウンダリに対する処理が複数存在し，バウンダリに関する処理が複数のコントロールに横断していると言える．図 4 の右に示すように，エンティティに関しても同様である．上記より我々は，ロバストネス分析の結果とアスペクトの対応関係に関して次の仮説を立てた．

> 1 つのバウンダリまたはエンティティに対して複数のコントロールと関連がある場合, そのバウンダリまたはエンティティに関する処理をアスペクトの候補とする

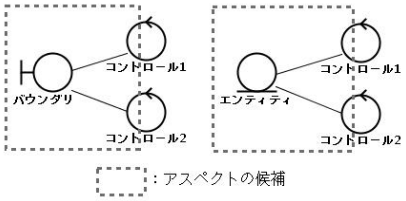

図 4　ロバストネス分析の結果とアスペクトの対応関係

4　事例検証

自動航行飛行船制御ソフトウェアを事例として仮説の検証を行う．

4.1　アーキテクチャ構築手順

飛行船制御ソフトウェアのアーキテクチャ構築手順を図 5 に示す．仕様モデル，それに対するロバストネス分析の結果に対して，仮説を用いて横断的コンサーンを発見し構築したアスペクト指向アーキテクチャ，仮説を用いず横断的コンサーンを発見し構築したアスペクト指向アーキテクチャの 2 通りを作成し，それらを比較することで，仮説の検証，妥当性の考察を行う．構築された 2 つのアーキテクチャの差を明確にするために，構築する 2 通りのアーキテクチャ両方に MVC を用いた．

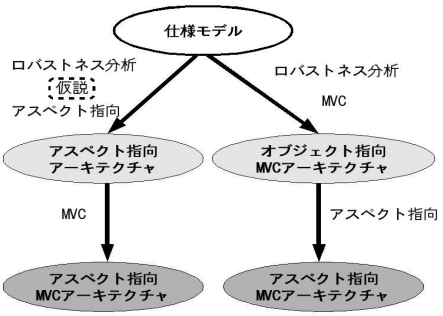

図 5　アーキテクチャ構築手順

4.2　仕様モデル

自動航行飛行船制御システムに対する要求分析を行う．システムへの要求として，航路の計算，推進，航行状態の表示，航行の中止が挙げられる．我々は自動航行飛行船制御システムを管制・航行・地上 US 測位システムの 3 つのサブシステムで実現する [4]．サブシステムへの要求を整理するために作成したユースケース図を図 6 に示す．

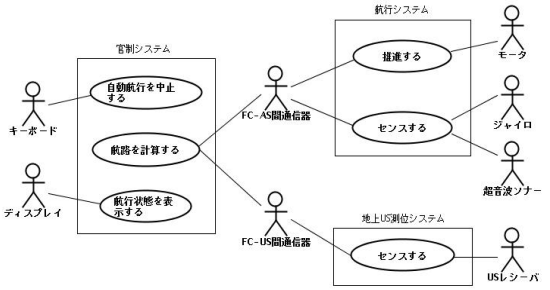

図 6　サブシステムに対する要求

4.3　ロバストネス分析

ユースケース図 (図 6) に対してロバストネス分析を行う．ユースケース記述からユースケースの実現に必要な 3 種類のオブジェクトを抽出し，それらの関連を定義する．図 7 はユースケース＜自動航行を中止する＞のロバストネス図である．バウンダリとしてキーボード，それに対するコントロールとして入力読み取りと目的地変更，さらにエンティティとして航路マップを抽出した．同様に，全てのユースケースに対してロバストネス分析を行い，作成したロバストネス図を 1 つに統合した．本来，ロバストネス分析では，類似した責務を持つコントロールは 1 つに統合することでモデルの洗練を行う．しかし，我々はアスペクトを意識しており，統合しない方が横断的コンサーンの存在が明らかになり，アスペクトを細かく抽出できると考えた．よって，今回はオブジェクトの統合を行わない．

次に，類似した責務を持つロバストネス図のオブジェクトを 1 つのモジュールとした．例として Keyboard モジュールを挙げる．図 7 のキーボードオブジェクト，入力読み取りオブジェクトはキーボード入力に関する責務が類似している．よって，これら 2 つのオブジェクトを Keyboard モジュールとして定義した．同様に全てのロバストネス図からモジュールを抽出した (図 8)．

図 7　ユースケース＜自動航行を中止する＞のロバストネス図

4.4　オブジェクト指向に基づくアスペクト指向アーキテクチャ構築

4.4.1　オブジェクト指向アーキテクチャ

図 8 によって抽出された各モジュールをクラスとし，オブジェクト指向アーキテクチャを構築した．さらに，各クラスに含まれるロバストネス図のオブジェクトの責務

より，クラスを MVC に分類した．その結果を図 9 に示す．各クラスを MVC に分類した結果，BTS_Zigbee クラス，Base_Transceiver_Station クラスについては，飛行船システムとのデータ送受信の責務を持つことから，ビューとコントローラの両方に分類された．

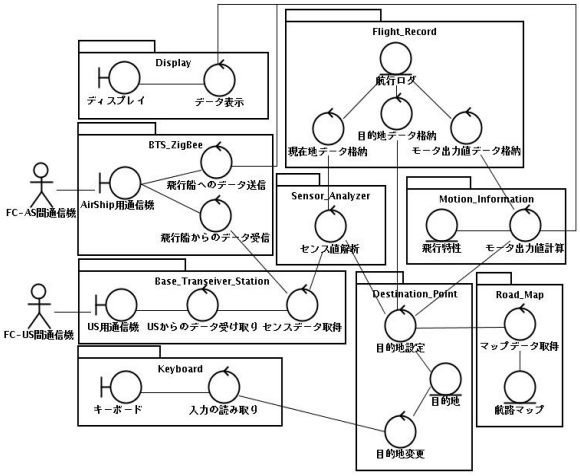

図 8　オブジェクトとモジュールの対応関係

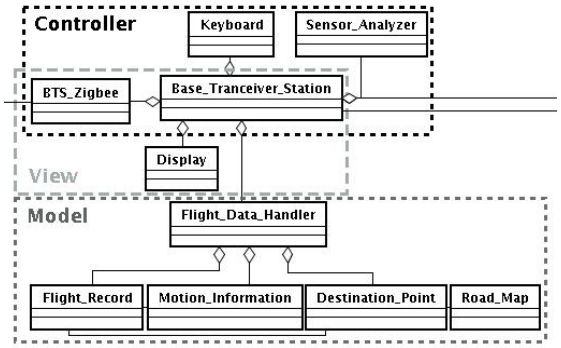

図 9　オブジェクト指向アーキテクチャと MVC の対応関係

4.4.2　アスペクト指向アーキテクチャ

オブジェクト指向アーキテクチャのクラスを MVC に分類した結果，ビューとコントローラの両方に分類されたクラスが存在した．我々は，ビューとコントローラ両方の側面を持つクラスをアスペクトとして抽出することが可能であると考えた．それらのクラスを新たに View_Controller アスペクトとして定義し，MVC とともにアスペクトとして抽出した．図 10 は E-AoSAS++ に基づいて設計したアスペクト指向アーキテクチャである．E-AoSAS++ に基づいて設計することで， View アスペクトと Controller アスペクトが View_Controller アスペクトを要求しているという関係を示した．

4.5　仮説に基づくアーキテクチャ構築

4.5.1　アスペクト指向アーキテクチャ

図 8 のロバストネス図において，仮説の形状に当てはまるオブジェクトを含むモジュールをアスペクトとして抽出する．例として Flight_Recorder モジュールを挙げる．Flight_Recorder モジュールに含まれる航行ログオブジェクトは，現在地データ格納オブジェクト，目的地データ格納オブジェクト，モータ出力値格納オブジェクトの 3 つのオブジェクトと関連しており，仮説の形状に当てはまる．Flight_Recorder モジュールをアスペクトとすることで，Sensor_Analyzer モジュール，Flight_Data_Handler モジュールが Flight_Recorder アスペクトを要求しているという関係を示した (図 11)．同様に仮説の形状に当てはまるオブジェクトを持つモジュールをアスペクトとし，アスペクト指向アーキテクチャを構築した．作成したアーキテクチャを図 11 に示す．仮説でアスペクトの候補としたモジュールをアスペクトとして扱うことの妥当性については 5.1 節にて述べる．

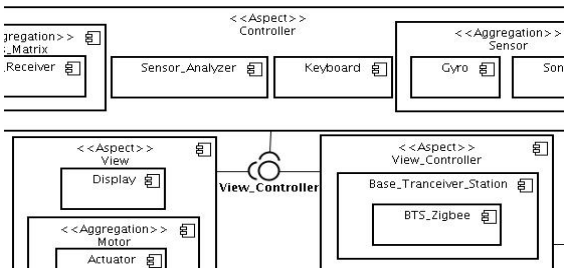

図 10　MVC に基づくアスペクト指向アーキテクチャ (静的構造図)

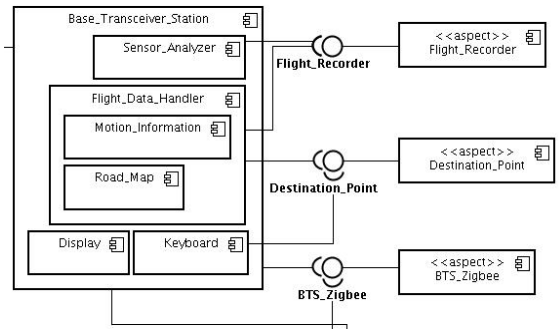

図 11　アスペクト指向アーキテクチャ (静的構造図)

4.5.2　MVC を用いたアスペクト指向アーキテクチャ

図 11 のアスペクト指向アーキテクチャの各モジュールを，責務より MVC に分類した (図 12)．図 10 と同様にビューとコントローラの両方に分類されたモジュールに関しては，View_Controller として定義し，MVC とともにアスペクトとして抽出した．また，Flight_Recorder アスペクト，Destination_Point アスペクトは共にモデルに分類されたが，4.5.1 節において処理が横断しているモジュールとして，すでにアスペクトとしているので，新たに Model_aspect を定義し，Flight_Recorder アスペクト，Destination_Point アスペクトを Model から抽出した．Airship_Zigbee アスペクト，BTS_Zigbee アスペクトについても，同様の理由より，新たに View_Controller_aspect を定義し，

Airship_Zigbee アスペクト，BTS_Zigbee アスペクトを View_Controller から抽出した．

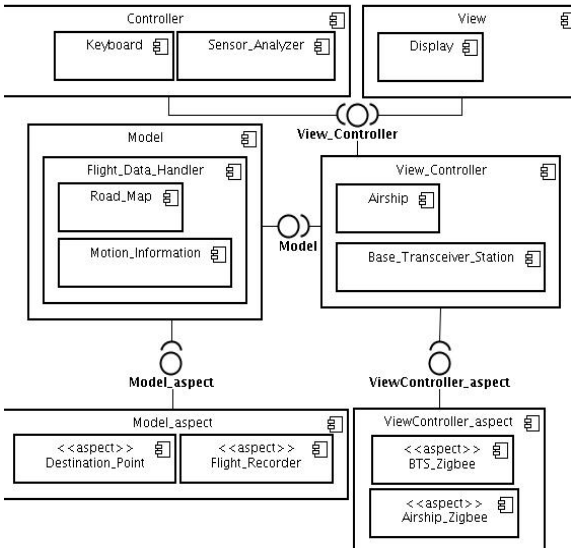

図 12　MVC に基づくアスペクト指向アーキテクチャ (静的構造図)

5　考察

5.1　ロバストネス分析の結果とアスペクトの対応関係に関する考察

4.5.1 節において，仮説に基づいて抽出したアスペクトの妥当性について考察する．例として図 11 の Flight_Recorder アスペクトを挙げる．Flight_Recorder コンポーネントは，含まれる航行ログオブジェクトと 3 つのコントロールオブジェクトとの関連 (図 8) と，そのコントロールオブジェクトと Sensor_Analyzer・Destination_Point・Motion_Information コンポーネントに含まれるオブジェクトとの関連からアスペクトとして抽出した．Flight_Recorder コンポーネントは Sensor_Analyzer・Destination_Point・Motion_Information コンポーネントにおいて算出されたデータを保持する責務を持ち，3 つのコンポーネントに Flight_Recorder コンポーネントを呼び出す処理が存在する．よって，Flight_Recorder コンポーネントをアスペクトとして抽出することは妥当であると言える．同様に仮説に基づいて抽出した他のアスペクトについても妥当であると言える．

5.2　アーキテクチャの比較

仮説に基づいて設計したアスペクト指向アーキテクチャ (図 12) と，仮説を用いず設計したアスペクト指向アーキテクチャ (図 10) を比較し，仮説を用いることで期待される結果と成果が一致するかを述べることで，本手法の有効性を考察する．図 10 では MVC がアスペクトとして抽出されているが，図 12 では MVC に加えて横断する処理がアスペクトとして抽出されている．本研究で提案する仮説は，横断する処理を横断的コンサーンとしていることから，横断する処理がアスペクトとして抽出されることが期待される．期待される結果と仮説に基づいて設計した成果が一致したことから，横断する処理を明確にしたい場合は本手法が有効であると言える．

今回，ユースケースに対してオブジェクト指向分析としてロバストネス分析を行った．ユースケースはシステムの機能を表しており，オブジェクト指向分析はデータ中心にモデル化を行っている．1 つのオブジェクトが複数の機能を持つ場合，オブジェクトと機能が横断すると言える．今回，ユースケースに対してロバストネス分析を行った結果，複数のユースケースに横断するオブジェクトが存在した．そのオブジェクトは複数の機能を持つことから，オブジェクトと機能が横断していると言える．この結果から，ロバストネス分析を用いることで，オブジェクトと機能が横断していることが明確になると言える．また，複数のユースケースに横断するオブジェクトは，全て仮説からアスペクトとして抽出された．上記のことから，仮説を用いて設計を行うことで横断的コンサーンを抽出することができ，ロバストネス分析の結果とアスペクトの対応関係を一般化できる可能性があると言える．今後，対応関係の一般性を他ドメインを用いて検証する必要がある．

6　おわりに

本研究では，仕様モデルからアーキテクチャを構築する際の問題点を挙げ，ロバストネス分析の結果とアスペクトの対応関係の仮説を立てた．飛行船制御ソフトウェアを事例に仮説の検証を行い，ロバストネス分析の結果とアスペクトの対応関係を考察した．また，仮説に基づいて設計したアーキテクチャと他手法を用いて設計したアーキテクチャを比較することで，本手法の有効性を検証した．その結果，ロバストネス分析を用いることで，仕様モデルからアスペクトが抽出可能であることが明確になった．今後の課題として，本研究で提案する対応関係の一般性について他ドメインを用いて検証することが挙げられる．

参考文献

[1] D. Rosenberg and K. Scott, *Use Case Driven Object Modeling with UML: A Practical Approach*, Addison-Wesley, 2001.

[2] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, and M. Stal, *Pattern-Oriented Software Architecture A System of Patterns*, John Wiley & Sons, Ltd., 1996.

[3] 太田将吾, “ソフトウェアアーキテクチャプログラムコードへの自動変換に関する研究,” 2007 年度南山大学修士論文, 2008.

[4] MDD ロボットチャレンジ 2008 実行委員会, MDD ロボットチャレンジ 2008 競技仕様書, Apr. 2008; www.ertl.jp/ESS/2008/mdd/file/MDDchallenge2008_system-requlation_sheet.pdf.