E-AoSAS++ に基づく飛行船制御ソフトウェア開発

2005MT021 服部 由衣　　2005MT037 伊藤 英樹
指導教員　野呂 昌満

1　はじめに

ハードウェアの高性能化に伴い，組込みシステムに対する要求が高度化し, ソフトウェアの規模や複雑さが増大している．規模や複雑さの増大に対してソフトウェアの保守性の高い設計を行うことが重要となっている．外界との密接な関係のある組込みソフトウェアではリアルタイム性や耐故障性といった様々な制約が存在し，ソフトウェア全体に散在する処理となってしまう．本研究室では，アスペクト指向技術を用いて保守性の高いソフトウェアを開発するために，組込みソフトウェアのため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++) が提案されている．E-AoSAS++ では，機能特性，非機能特性に対する記述方法を規定している．本研究室では, 事例として自動販売機や携帯電話等を E-AoSAS++ に基づき開発を行ってきた．しかしながら，実際の組込みソフトウェアに存在する複雑な制御を持つ事例に対する検証が行われていない．本研究の目的は，複雑な制御を持つ組込みソフトウェアにおける記述方法の妥当性の考察を行うことである．本研究では，複雑な制御を持つ組込みソフトウェアの事例として飛行船制御ソフトウェアを取り上げる．

本研究において，飛行船制御ソフトウェアを取り上げる理由として，飛行船制御にはモータやセンサの制御，通信処理などの様々な制御に関する要件が存在し，複雑な制御を持つソフトウェアの例として事例検証を行うことが可能であると考えられるからである．また，情報処理学会の組込みシステム研究会が企画する MDD ロボットチャレンジ 2008 において自動航行飛行船制御ソフトウェアが提案されており, 要求仕様が決定している組込みソフトウェアの開発の良い事例となるからである．

2　背景技術

2.1　E-AoSAS++

2.1.1　概要

E-AoSAS++ は，コンポーネントコネクタスタイルに基づく組込みシステムのソフトウェアアーキテクチャを構築するためのアーキテクチャスタイルである．組込みソフトウェアを並行に動作する並行状態遷移機械 (以下，CSTM) の集合として規定し，各 CSTM は，自分自身及び，外部からのイベントに対して状態を遷移させる．その際，自分自身及び，他の CSTM に対しイベントを通知することで協調動作を行う．

2.1.2　コンセプチュアルアーキテクチャとインプリメンテーションアーキテクチャ

E-AoSAS++ において作成するアーキテクチャは，ソフトウェアを概念的に表したコンセプチュアルアーキテクチャ，システムに散在する関心事をアスペクトとしてモジュール化したインプリメンテーションアーキテクチャである．インプリメンテーションアーキテクチャは，機能や非機能特性をアスペクトとしてモジュール化するアスペクト指向によって設計されるアーキテクチャである．コンセプチュアルアーキテクチャでは，静的構造を UML のクラス図により記述し動的振る舞いをシーケンス図，状態マシン図を用いて記述する．インプリメンテーションアーキテクチャでは，静的構造をコンポーネント図とクラス図により記述し動的振る舞いをシーケンス図，状態マシン図を用いて記述する．

2.1.3　グローバルコンサーンとローカルコンサーン

グローバルコンサーンは，組込みソフトウェア全体に散在する関心事であり，E-AoSAS++ では， 並行処理, 状態遷移, アクションをグローバルコンサーンとして規定し，アスペクトとしてモジュール化する．各 CSTM は，これらの関心事をモジュール化したアスペクトから構成され，各アスペクトの関連はアスペクト間記述である Inter Aspect Discription（IAD）に記述される．

ローカルコンサーンは，組込みソフトウェアの特定のコンポーネントに横断する関心事であり，E-AoSAS++ では，例外処理, 耐故障性, 実時間処理をローカルコンサーンとして規定する．

2.1.4　E-AoSAS++ に基づく開発支援環境の開発プロセス

E-AoSAS++ に基づく開発支援環境の開発プロセスでは, 要求分析, ドメイン分析を行い仕様モデルを決定し, 仕様モデルに基づきソフトウェアアーキテクチャを構築する．構築するアーキテクチャは, 開発するソフトウェアを概念的に表したオブジェクト指向モデル, 開発するプロダクトを表すアスペクト指向モデルである．

アーキテクチャの正しさを実行前検査により検証し, 構築されたアスペクト指向モデルのアーキテクチャ記述を基にコードを作成する．実行前検査, コード変換はツールを用いて実現する．

実行前検査, コード自動変換についてはツールの整備が

十分にされていないので本研究では行わない. コード変換はツールで用いる変換論理に基づき変換を行うことにより実現する.

2.2 MVC アーキテクチャ

MVC アーキテクチャとは，システムを Model，View，Controller の 3 つのコンポーネントに分類し，各コンポーネントの独立性を高めるアーキテクチャである．よって，保守性の高い設計が可能である．
各コンポーネントの責務や各コンポーネント間の関連を次に示す．システムをこれら 3 つのコンポーネントに分類し，システムを設計する．

- Model
  アプリケーションの処理の中核を成すデータとその処理をおこなうコンポーネント
- View
  表示，出力に関する責務を持つコンポーネント
- Controller
  外部からの入力にを受け取り制御を行うコンポーネント

各コンポーネント間の関連は，Controller が入力を受け取り Model に対し処理の依頼をする．Model からの更新通知を受け取った View が表示，出力の処理を行いアプリケーションの機能を実現する．

3 E-AoSAS++ に基づく自動航行飛行船制御ソフトウェア開発

E-AoSAS++ は, 組込みソフトウェアのアーキテクチャを作成するための系統的な記述方法を規定している．E-AoSAS++ を用いることで組込みソフトウェアを適切な粒度でモジュール化し，ソフトウェアの再利用部品を作成することが可能である． 本研究は以下の手順で進めた.

- 飛行船制御ソフトウェアの要求分析，ドメイン分析
- 飛行船制御ソフトウェア設計

3.1 要求概要

MDD ロボットチャレンジにおいて満たすべき要求を以下に示す.

- 飛行船を推進させ, 通過地点を二箇所通り, 目的地へ辿り着く
- 飛行船に搭載されたセンサと地上に配置されたセンサのセンス値を基に飛行船の現在地を計算する
- 航行は全て自動で行い飛行船の現在値から航路と推進力を計算し航行する
- 推進力と飛行船の現在地を画面に表示する

3.2 要求分析，ドメイン分析

要求分析にユースケース図を用いることで，システムの機能と要求の整理を行った．飛行船制御ソフトウェアの本質的な機能を表すことで，システムの全体像を把握することが可能になる．
今回対象とした飛行船制御システムは，ハードウェア構成についての仕様が MDD ロボットチャレンジの競技仕様として決定していることから，ユースケース図へこれらのハードウェア仕様を加えたユースケース図を作成した．作成したユースケース図を図 2 に示す．図 1 のユースケース図を仕様モデルとして飛行船制御ソフトウェア設計を行った.

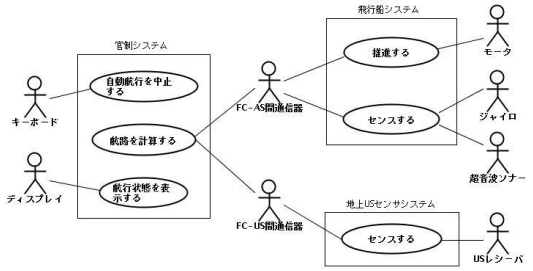

図 1　ユースケース図

また，ドメイン分析を行ったシステムの概要を図 2 に示す.

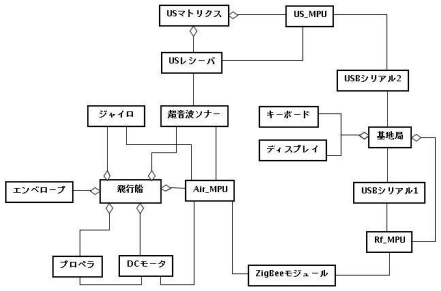

図 2　システムアーキテクチャ

システムアーキテクチャをクラス図を用いて作成することで，ハードウェアの関連を意識して構造を表すことができ，仕様を整理することができた．また，ハードウェア毎の責務を責務表を用いて記述した.

3.3 コンセプチュアルアーキテクチャ

コンセプチュアルアーキテクチャはソフトウェアの概念的な構造を示したオブジェクト指向モデルである．コンセプチュアルアーキテクチャもユースケース図と同様，飛行船制御ソフトウェアの本質的な構造と，システム構成を加味したコンセプチュアルアーキテクチャを作成する.
コンセプチュアルアーキテクチャを作成するにあたり，

ロバストネス分析を行い，オブジェクトの候補を抽出した．ロバストネス分析は，ユースケース記述を基にオブジェクトの候補を抽出するオブジェクト指向設計のためのアプローチである．ロバストネス分析に加え，MVC を適用しモジュール分割を行った．飛行船制御ソフトウェアにはセンサやキーボード，モータやディスプレイが外界とのインタフェースとして存在し，また位置情報や運動情報などのデータに関する処理が多く存在する．MVC を適用しそれらの責務を適切に分類することが可能になる．MVC を適用しモジュール分割を行ったコンセプチュアルアーキテクチャを図 3 に示す．

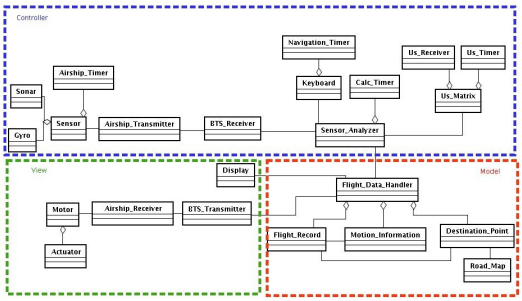

図 3　コンセプチュアルアーキテクチャ

今回の対象では，ハードウェアが決定していることから，ハードウェアの制約を考慮し設計を行った．その結果，ハードウェアの制約からセンス入力からモータ出力を行う振る舞いの中で飛行船クラス，基地局クラス，ZigBee クラス間に View と Controller の責務が横断することが分かった．責務が横断している箇所を図 4 に示す．

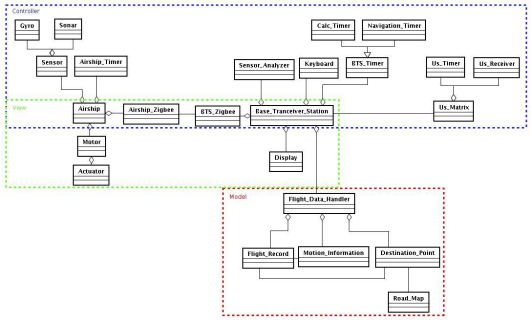

図 4　ハードウェア仕様を加味したコンセプチュアルアーキテクチャ

3.4　インプリメンテーションアーキテクチャ

インプリメンテーションアーキテクチャは，システムに横断する関心事をアスペクトとしてモジュール化したアスペクト指向モデルである．

MVC と E-AoSAS++ で規定するローカルコンサーンを関心事として関心事が横断している箇所をアスペクトとして定義した．ハードウェア仕様により View と Controller の責務が横断する箇所を View_Controller アスペクトとして定義を行い横断的関心事の分離を行った．更に，Sensor クラス，Sensor_Analyzer クラス，Us_Matrix クラスにリアルタイム処理が散在するので，リアルタイムアスペクトとして定義した．View_Controller アスペクト，リアルタイムアスペクトとして横断的関心事をモジュール化することで，各モジュールの責務を明確にすることができることに加えて，モジュール間の関係を疎にすることができ，ソフトウェアの保守性の向上につながると考えられる．作成したインプリメンテーションアーキテクチャを図 5 に示す．

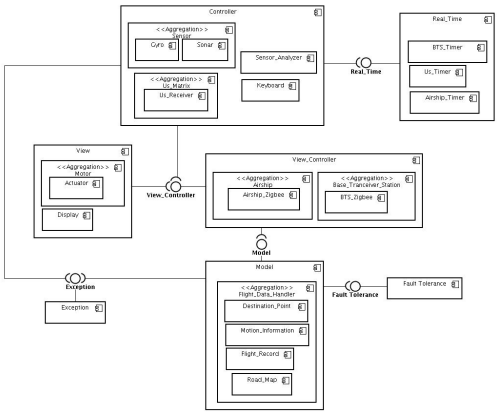

図 5　インプリメンテーションアーキテクチャ

4　本研究において作成したモデルの考察

4.1　非機能特性に関する考察

飛行船制御ソフトウェアの非機能特性としてセンシングや計算処理の実時間処理やハードウェアの故障，障害に対する処理が考えられた．これらの処理は，E-AoSAS++ におけるローカルコンサーンとして取り扱うことが可能であると考えられる．E-AoSAS++ では，ローカルコンサーンを UMP により実現する．UMP により通常処理と非機能処理を分離し階層的に記述することで静的構造において複数の視点の処理を一つにパッケージ化する．本事例のローカルコンサーンの例として実時間処理を挙げる．航行を行う振る舞いにおいて一定周期でセンス値を解析しモータを回転させる処理が存在する．要求分析の結果からセンス値解析を基地局が計算し，モータの回転を飛行船が制御を行うと考えられた．それぞれのハードウェアにおいて一定周期毎に計算，制御を行うことで振る舞いを行う．基地局と飛行船において別々のタイマが存在しそれらを制御することで実時間処理を実現する．これらの静的構造を E-AoSAS++ の記述方法で記述すると図 6 のようにな

る．図 6 は，コンセプチュアルアーキテクチャにおいて実時間処理を行うクラスとそれに関連するクラスを抜き出し，E-AoSAS++ におけるリアルタイムアスペクトの記述方法に基づき記述したものである．

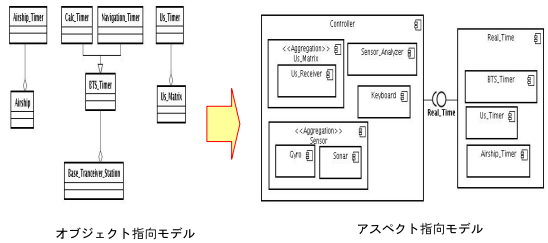

図 6　リアルタイム処理の記述方法

基地局のタイマを BTS_Timer が制御し，飛行船のタイマを Airship_Timer が制御する．基地局でモータ出力値の計算を行う際には，一定周期でセンス値を解析する必要がある．図 7 は，基地局においてセンス値解析の動的振る舞いを表したシーケンス図になる．基地局では，タイムアウト後に Controller アスペクトにタイムアウトイベントを通知しセンス値解析を行う．

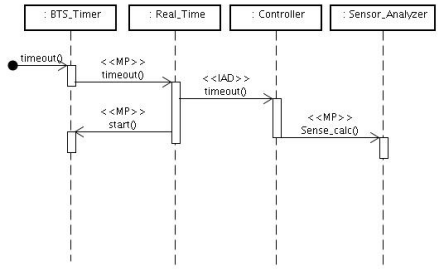

図 7　センス値解析の実時間処理

基地局, 飛行船, US マトリクスにおいて実時間性が横断しているので Realtime アスペクトを定義した．これにより Controller アスペクトと Realtime アスペクト間のイベント通知は，アスペクト間記述によって行われる．UMP において分離した非機能処理をアスペクトとして定義することで非機能処理を正常処理から分離することが可能になり非機能処理の追加や変更に対し正常処理に影響を及ぼすことがなくなり変更が容易になったと考えられる．一定周期で処理を行う実時間処理は，過去の事例で扱ったことがなく，本事例において E-AoSAS++ の記述方法に基づき動的振る舞いを記述できたことから E-AoSAS++ の記述が妥当であると考えられる．

4.2　複雑な制御に関する考察

飛行船制御ソフトウェアは，センサ入力に対しモータ出力値を計算しモータ出力を行う．モータ出力が次のセンサ入力に影響を与えセンサ入力とモータ出力を繰り返すことで自動航行を可能にする．E-AoSAS++ において過去に開発を行った組込みソフトウェアでは，状態遷移機械の振る舞いによって機能の大半を実現することが可能であった．自動販売機を例に挙げるとボタンの入力に対し缶を出力することで自動販売機の振る舞いを実現する．しかし本事例のような複雑な制御を持つ組込みソフトウェアは，状態遷移機械による振る舞いだけでなくデータの処理を記述することが必要になってくる．E-AoSAS++ では，状態遷移機械の振る舞いに関する記述は，アクションアスペクトに記述を行う．アクションアスペクトを構成するコンポーネントとして，状態遷移機械が処理する手続きを実行する Action と状態遷移機械が保持するデータ構造とデータ構造に対するアクセスを保持する ApplicationLogic が存在する．本研究では，組込みソフトウェアの入出力制御とデータ処理に関する制御の構造を MVC によって整理を行った．データ処理に関する制御を行うクラスを Model として定義したことで，Model で定義されたクラスのロジックを ApplicationLogic に記述することで複雑な制御に関する記述が可能であると考えられる．複雑な制御に関する記述をアクションアスペクトの ApplicationLogic に記述可能であることから E-AoSAS++ における記述方法が複雑な制御を持つ組込みソフトウェアの記述方法として妥当であると考えられる．

5　おわりに

本研究では E-AoSAS++ に基づいた飛行船制御ソフトウェアの開発を行い，作成したモデルについて E-AoSAS++ の記述方法の妥当性を考察した. 飛行船制御ソフトウェアを例に複雑な制御を持つ組込みソフトウェアに対し E-AoSAS++ の記述方法が妥当であることを検証し事例検証の蓄積を行った．今後の課題として同様の特徴を持つ他の組込みソフトウェアに対し本研究で用いた変換方法を基に E-AoSAS++ による記述が可能であるかの検証を行うことが挙げられる．

参考文献

[1] MDD ロボットチャレンジ競技仕様書，2008; www.ertl.jp/ESS/2008/mdd/file/MDDchallenge2008_system-requlation_sheet.pdf.

[2] M. Shaw and D. Garlan, *Software Architecture - Perspectives on an Emergin Discipline*, Prentice Hall, 1996.

[3] M. Noro, A. Sawada, Y. Hachisu, and M. Banno, “E-AoSAS++ and its Software Development Environment,” *Proceedings of the 14th Asia-Pacific Software Engineering Conference*, 2007, pp. 206-213.