モデル駆動型開発の実践例

（代表著者）京都情報大学院大学 准教授　江見 圭司

MDD ロボットチャレンジとは

京都コンピュータ学院と京都情報大学院大学では，高度なモデル駆動型開発を身につけるために，株式会社ヒューマンエンジニアリングアンドロボティックスと産学連携で，情報処理学会の組込みシステム研究会が主催している MDD ロボットチャレンジという大会に参加した。この大会では，小型飛行船を制御するソフトウェアをモデル駆動開発に従って開発・実践・研究する機会と，組込み技術者育成の場を提供している大会である。MDD（Model Driven Development）に基づいた開発を行い，用いたモデルと飛行競技をコンテスト形式で競うのである。ところで MDD とは，明示的な論理にもとづいてソフトウェアモデルを変換することの繰り返しで最終ソフトウェアを生成する開発方法である。

審査基準はモデルと競技の２つからなる。

モデル図は，本プロジェクトでは UML を用いた。この場合，システムに要求する機能はユースケース，構造はクラス図（必要に応じてオブジェクト図），さらにシステムの構造をシーケンス図で表現する必要がある。また，作成したモデルからコーディングに至るまでのコード変換ルールを定めることも審査基準に含まれる。

あともう一点は，飛行競技である。”自動航法競技 ” であり，これは飛行船を出発地点から規定の通過点（Way Point 1，Way Point 2 といい，風船がそれぞれの地点に浮いている）を通過しながら目的地点に着陸させる一連の動作を自動制御で行うことを競うものである。離陸，第一通過点（Way Point 1），第二通過点（Way Point 2），着陸，ゴールへの着陸，制御画面の表示の６つの部分点の合計で競う。着陸とゴールへの着陸が異なるのは，ゴールへの着陸が困難であるため，とりあえず，規定時間内に着陸しただけでも部分点が取得できるように配慮されているのである。

我々のチームは 2007 年から 2010 年まで 4 回出場した。それで，2010 年には IBM 大和研究所のチームが総合一位で，我々は総合二位に輝いた。

ところが，この大会は残念ながら，2010 年で使命を終えて終了した。そこで本稿では，そのときに作成したモデル図をここに残すことにして，奥田茂人によるレポート「MDD ロボットチャレンジを終えて」を末尾に追加した。

◆ 1. はじめに

◆ 1.1. MDDロボットチャレンジへの参加

京都コンピュータ学院（KCG）・京都情報大学院大学（KCGI）・株式会社ヒューマンエンジニアリングアンドロボティックス（HERO 社）の共同チームは，2007 年，2008 年に続き，MDD ロボットチャレンジへ参加。チーム名は「京魂英雄（きょうこんひーろー）」とした。

◆ 1.2. プロジェクト運営，参加メンバと役割について

2007 年の大会では，短期間での準備・開発が課題となっていた。そこで，日本語プログラミング言語「ドリトル」を使用し，プログラミング言語習得に費やす時間と，開発工程の短縮を実現した。

2008 年は，2007 年の運営との比較を行うため，プロジェクトの管理方法や使用するプログラミング言語を C ＃として，ドリトルの比較を行った。

2009 年は，2007 年，2008 年の結果をふまえたプロジェクト運営を行った。2007 年で使用したドリトルからシリアルポート通信を強化した新バージョンが登場したので，それを採用した。

2010 年は開発言語には C# を用いた。2008 年でも C# を用いて開発したが，モデルの改善を行い，それに伴ってソフトウェアも改善されたと考えている。また，地上局の超音波センサとのやりとりの改善を行って位置測定の精度を上げることや，GUI の改善を行った。

また，モデリングには前年度までは JUDE（現 Astah*）を用いたが，モデル→コーディングまでのトレーサビリティを考慮した結果，Microsoft Visio 2010 と Visial Studio 2010 を利用した。

表 1 にメンバーの一覧と，それぞれの役割について示す。KCG・KCGI の学生は，プログラミングやモデリング，プロジェクトマネジメントなどを実践して学ぶことをこのコンテストの参加目的としている。HERO 社は，制御機器やファームウェアソフトウェアを担当した。また，2009 年と同様，資料の共有のために MS 社 Skydrive を採用したため，資料などをそのまま引き継ぐことができた。

プロジェクト運営において以前との違いは，開発開始時期や実際の開発スケジュールが変化したこと，プロジェクト開始以降，メンバーの脱退や入れ替わりが多く，途中段階で役割分担などの変更を要した。そのため，プロジェクト運営にあたっては臨機応変な対応が必要であった。しかし，過去のドキュメン

所属	名前	モデリング	コーディング（飛行船制御）	地上局開発	ファームウェア開発	機体作成	ドキュメント作成	アドバイザ
KCGI（准教授）	江見圭司（責任者）							
KCG（教員）	久保田英司					○		
KCGI（学生）	郭 輝平		○					
	高瀬えりか						ドキュメントまとめ	プロジェクト全般
	Vianh To	○						
	上川智弘							地上局開発
KCG（学生）	奥田茂人	○					モデル	
	米山哲平		○	○			飛行船制御	
	倉田 実功							ハードウェア等
HERO	西村憲二			○	○			
OB	甲斐信行							地上局開発

表 1 メンバーと役割

トの管理を徹底した結果，プロジェクトの引き継ぎは比較的スムーズに行われて，プロジェクトは継続された。

◆ 2. 機能

◆ 2.1. マインドマップとユースケース以前

ユースケース図を考えるにあたって，まずは MDD ロボットチャレンジ 2010 について，図 1 のマインドマップを作成した。これによって，MDD ロボットチャレンジの 2010 の全体像を掴むと同時に，ユースケース図の記述の参考になった。

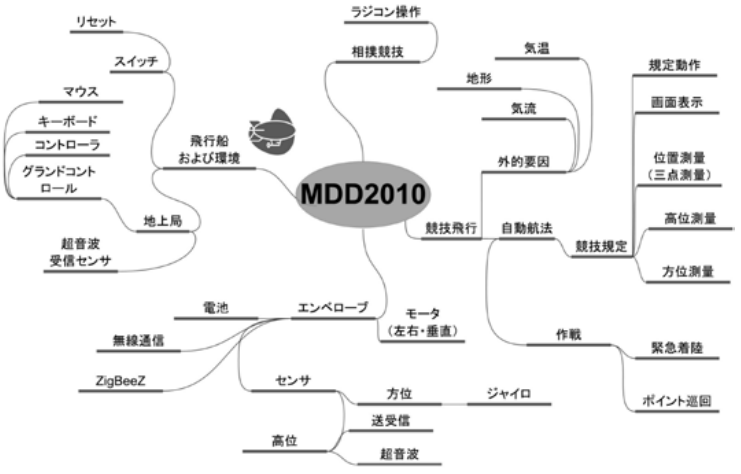


図 1 マインドマップ

◆ 2.2. ユースケース

規定動作と飛行競技についてのユースケースを図 2 に示す。「競技をやり直す」動作があるため，アクターは飛行船や地上局などのハードウェア以外に，「チャレンジャ」を加えた。チャレンジャは競技のやり直しの判断と，データ画面表示の測定結果の確認を行う。

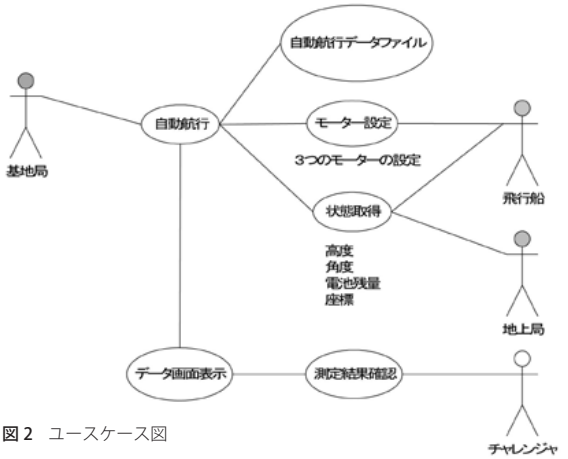


図 2 ユースケース図

規定動作・飛行競技のミスユースケースを図 3 に示す。ユースケース同様に，「チャレンジャ」アクターを記述し，人間（チャレンジャ）の手によって行われる「停止」と「やり直し」の動作も表現している。

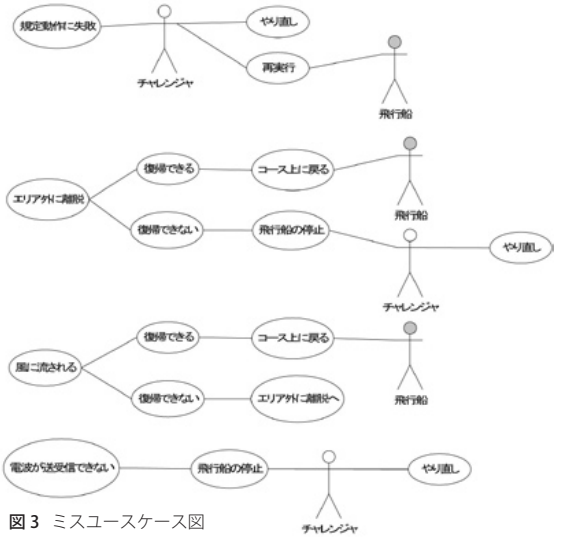


図 3 ミスユースケース図

以上のユースケースやミスユースケースは，飛行船だけでなく，ライトレースの走行体である競技でも応用できるように考えている。つまり再利用可能性は高い。

◆ 2.3. シナリオ（規定動作）

前述のユースケースを元に，ユースケース後のシナリオを明確にした。基本シナリオは飛行船の基本的な規定動作であり，例外シナリオは，基本シナリオの①～④がそれぞれ失敗した際の「競技のやり直し」の場合に発生する。なお，規定動作に関しては自動航行データファイルに記述してあるものを読み込み，実現するものとする。

基本シナリオ

①・飛行船が離陸する。
②・飛行船が規定通り上昇する。
③・チャレンジャがホバリングと，高さ，角度測定の申告をする。
・飛行船が規定通りホバリングする。
・飛行船が測定結果（高さ・角度）を基地局に送信する。
・基地局が，測定結果（高さ・角度）を画面に表示する。
・チャレンジャが測定結果（高さ・角度）を確認する。
④・飛行船が位置測定可能な場所にいる場合，チャレンジャが，位置測定の申告をする。
・地上局が，測定結果（位置）を基地局に送信する。
・基地局が，測定結果（位置）を画面に表示する。
・チャレンジャが測定結果（位置）を確認する。

例外シナリオ 1

①が失敗した場合，チャレンジャがやり直しの申告をし，
基本シナリオの①から開始する。

例外シナリオ 2

①は基本シナリオ通り
②・飛行船が上昇しない場合，チャレンジャがやり直しを申告し，基本シナリオの①→②→③→④の順に進める。

例外シナリオ 3

①，②は基本シナリオ通り
③・飛行船がホバリングしない，またはチャレンジャが測定結果を確認できない場合，チャレンジャがやり直しを申告し，基本シナリオの①→③→④の順に進める。

例外シナリオ 4-a

①，②，③は基本シナリオ通り。
④・飛行船が，位置測定不可能な位置にいた場合，飛行可能な位置に移動する。
・チャレンジャが，位置測定の申告をする。
・地上局が，測定結果（位置）を基地局に送信する。
・基地局が，測定結果（位置）を画面に表示する。
・チャレンジャが測定結果（位置）を確認する。

例外シナリオ 4-b

①，②，③は基本シナリオ通り。
④・飛行船が位置測定可能な場所にいるが，チャレンジャが測定結果を確認できない場合，チャレンジャがやり直しを申告し，基本シナリオの①→④の順に進める。

◆ 2.4. シナリオ（飛行競技）

前述のユースケースを元に，ユースケース後のシナリオを明確にした。以下の記述については「2.3 シナリオ (規定動作)」と同様である。

基本シナリオ

①飛行船が離陸する
②基地局が目的地を設定する
③飛行船が目的地に向かう
④飛行船が第 1 立ち寄り点を通過する
⑤飛行船が第 2 立ち寄り点を通過する
⑥飛行船が目的地に着陸する

例外シナリオ 1

飛行船がエリア外に離脱した場合，復帰できなければチャレンジャがやり直しの申告をし，
「2.3 シナリオ（規定動作)」の基本シナリオの①から開始する。

例外シナリオ 2

飛行船が風に流されてエリア外に離脱した場合，復帰できなければチャレンジャが
やり直しの申告をし，「2.3 シナリオ（規定動作)」の基本シナリオの①から開始する。

例外シナリオ 3

飛行船と電波の送受信ができない場合，チャレンジャがやり直しの申告をし，
「2.3 シナリオ（規定動作)」の基本シナリオの①から開始する。

◆ 2.5. 飛行戦略

前述のユースケースを元に，ユースケース後のシナリオを明確にした。

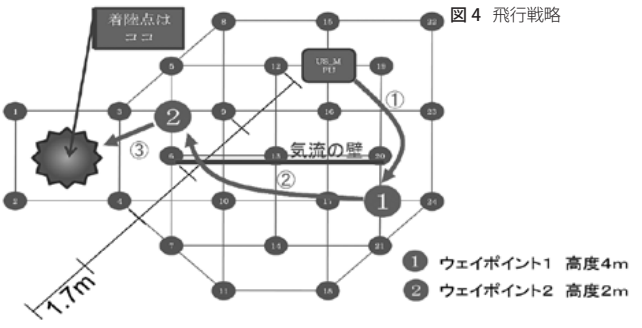


図 4 飛行戦略

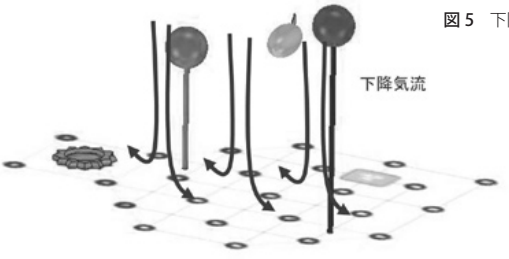


図 5 下降気流の予想

◆ 3. 構造

◆ 3.1. ユースケースからクラス図へ

ユースケース図および，ユースケース後のシナリオを作成した後，構造の段階として，クラス図が必要となる。ユースケースを実現するためのクラスを求めるために，ロバストネス図の記述方法を利用した。（図 6 参照）

バウンダリとして識別したユースケース：「データ画面表示」「状態取得」

「データ画面表示」は「飛行船データ表示 GUI」クラスとし，「状態取得」のユースケースにおいては，クラスを「飛行船状態」と「飛行船平面座標」に分割した。

コントロールとして識別したユースケース:「自動航行」「モーター設定」

「自動航行」は「飛行船モーター制御」クラスとし,「モーター設定」は「飛行船モーター状態」クラスとした。

エンティティとして識別したユースケース：「自動航行データファイル」

「自動航行データファイル」は，「自動航行データ」クラスとした。

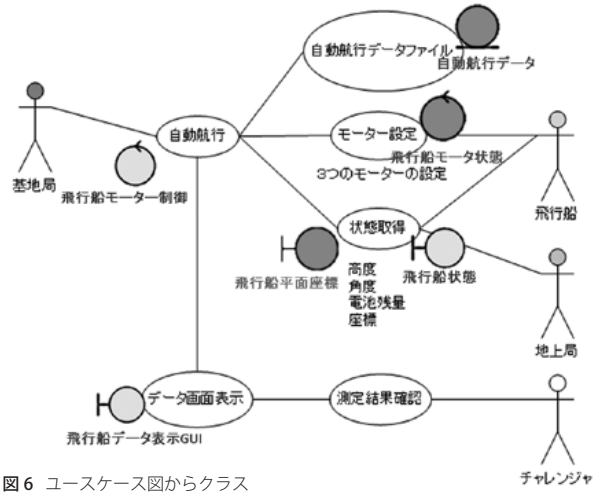


図 6 ユースケース図からクラス

◆ 3.2. クラス図

前節の分析から図 7 のようなクラスを分析レベルで設計した。

飛行船に関するクラス（うすい青色）として,「飛行船モーター状態」クラス,「飛行船状態」クラスがつくった。

地上局に関するクラス（うすい橙色）として,「飛行船平面座標」クラスをつくった。

飛行船制御に関するクラス（うすい黄緑色）として,「自動航行データ」クラス,「飛行船モーター制御」クラスをつくった。

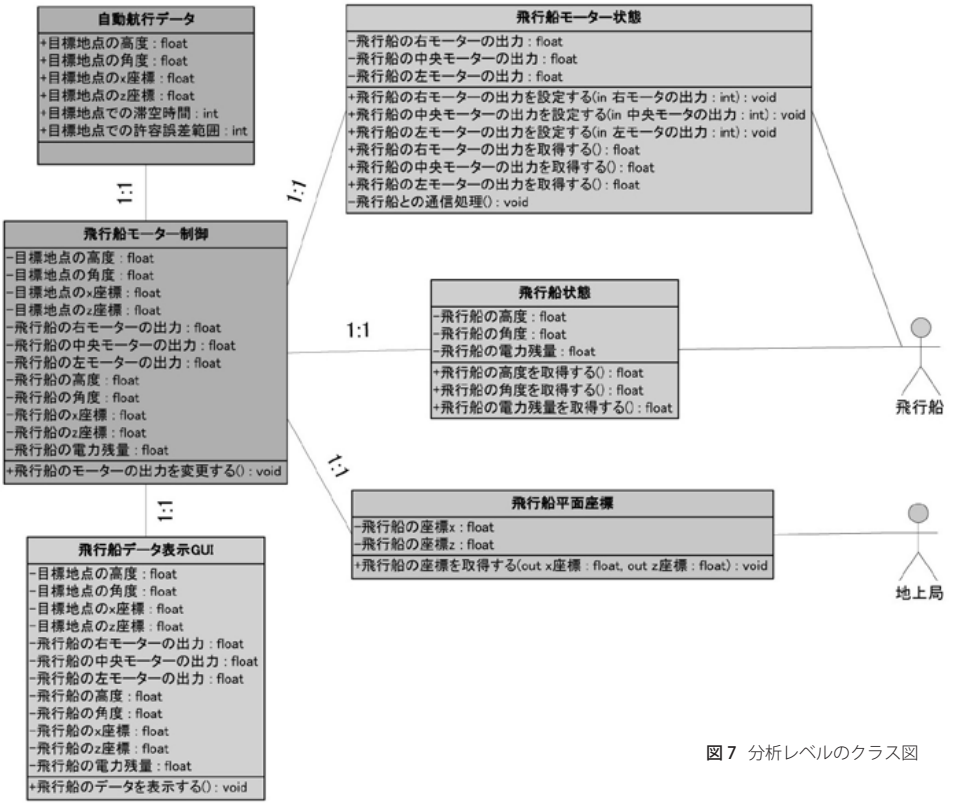


図 7 分析レベルのクラス図

表示部分に関しては「飛行船データ表示 GUI」クラスをつくった。

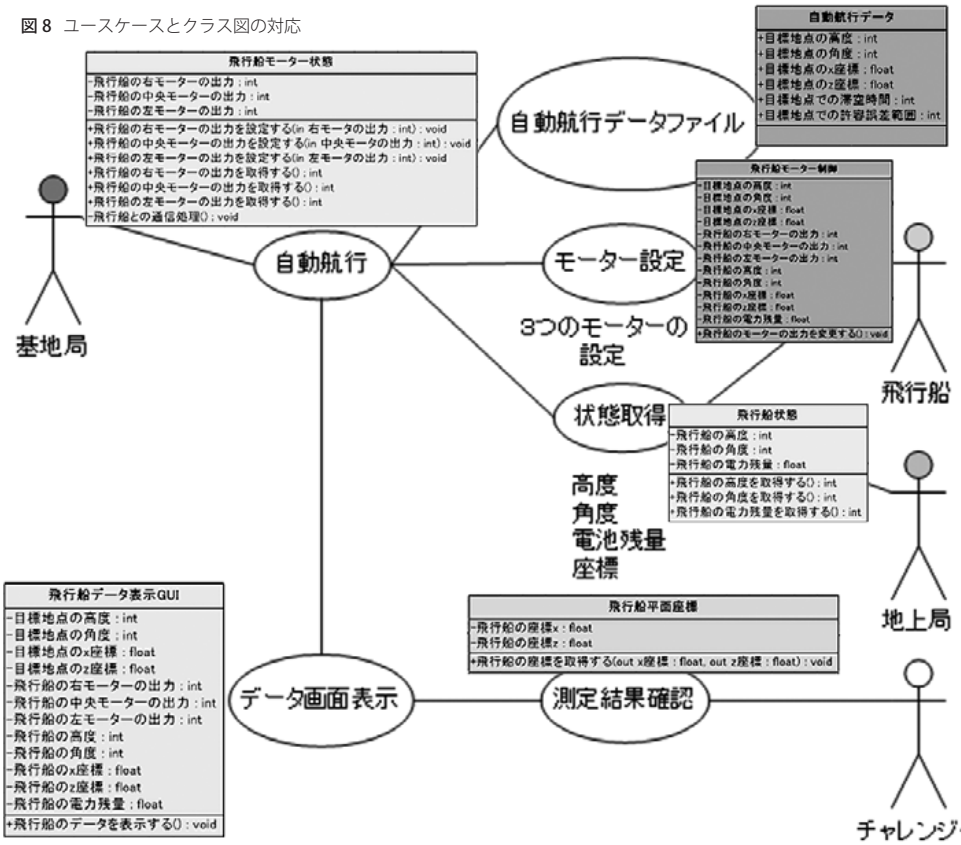
図 7 において以下の注意点がある。

・「飛行船モータ制御」クラスから「飛行船データ表示 GUI」クラスにデータを渡す際には C# の event を使用しデータをやり取りする。

・C# ではメンバー変数の値の取得 (get)・変更 (set) を行うためのメソッドはプロパティで表現できるのでそちらを使用する（次節参照）。

つぎに，機能要件を満たしているかどうか確かめるための図を添えておく。

図 8 ユースケースとクラス図の対応



◆ 3.3. UMLクラス図から C#への変換

メンバー変数の値の取得 (get)・変更 (set) を行うためのメソッドであるアクセサー (accessor) を必要とする。C++ や Java などの言語ではメンバー変数の数だけアクセサーが存在する場合もある。

クラス内部側ではメンバ変数の可視性はプライベートにしておいて，アクセサーというメソッドを通して取得 (get)・変更 (set) を行う方がよい。一方，クラス利用側からすると，メンバー変数に値を直接代入するほうが見た目がすっきりする。

このような理由から，C# では，クラス内部から見るとメソッドのように振る舞い，クラス利用側から見るとメンバー変数のように振る舞う「プロパティ」という機能がある。本プロジェクトでは，「プロパティ」という機能を前提にしているため，取得 (get)・変更 (set) などのアクセサーはクラス図から省いている。

以上のことを前提にして，分析レベルのクラス図から設計レベルのクラス図へ変換して，C# のコーディングを行った。

◆ 4. 振る舞い

◆ 4.1. シーケンス図とステートチャート図

シーケンス図について図 9 に示す。

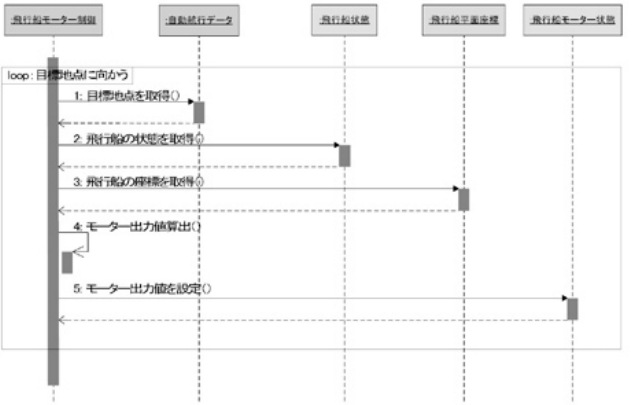


図 9 シーケンス図

シーケンス図に関する補足説明を追加する。

- 「loop: 目的地点に向かう」の目的地点とは，規定動作を行う地点，第 1，2 立ち寄り地点，着陸地点など，自動航行データファイルに設定された移動先のことを指す。
- モーター出力値算出では，目標地点情報と飛行船状態情報を元に設定値を算出する。この時，ミスユースケース状態が発生しても，正常状態に復帰する方法はユースケース状態のモーター出力値算出方法と同一である。よって，ミスユースケースはシーケンス図に表れない。

次に，飛行船モーター制御クラスのオブジェクトのステートチャート図を示す。

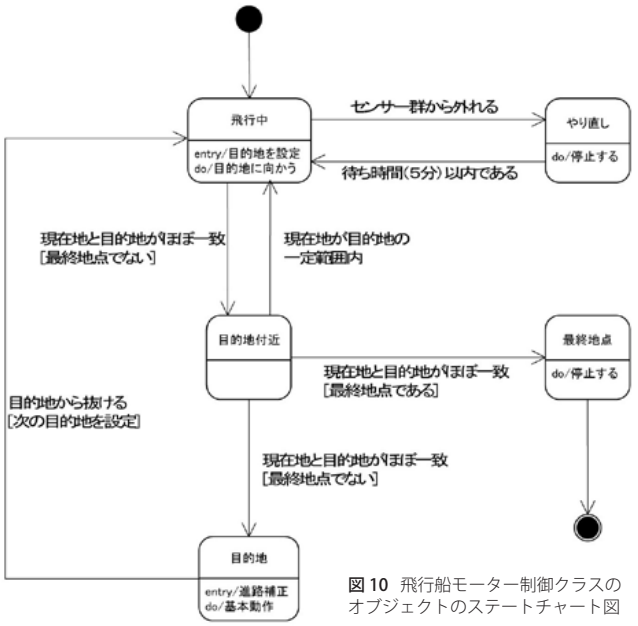


図 10 飛行船モーター制御クラスのオブジェクトのステートチャート図

◆ 4.2. UMLシーケンス図から C# への変換

シーケンス図から C# のソースコードへの変換方法の例を記載する。競技には直接関連がないが，UML からソースコードへ変換する際の手順の参考とする。

本シーケンス図においては自動航行データファイルに設定された目標地点に到達することを繰り返すことにより，最終的な着陸地点に到達するようになっている。そのため，プログラミング上では規定動作を行う地点，第 1，2 立ち寄り地点，着陸地点という概念はない。あくまでも設定された目標地点に向かうことを繰り返すのみである。以上から，各チェックポイントの位置が変更，あるいは増減した場合でもソースコード上の変更は一切必要ない。

また，ミスユースケース状態が発生した場合でも，飛行船の航行方法はユースケース状態の場合と同じため，プログラム上にミスユースケースのパターンは存在しない。これはどのような状態であっても，モーター出力値の計算方法は同一だからである。

◆ 5. ハードウェアのシステム構成とソフトウェア

◆ 5.1. 基地局

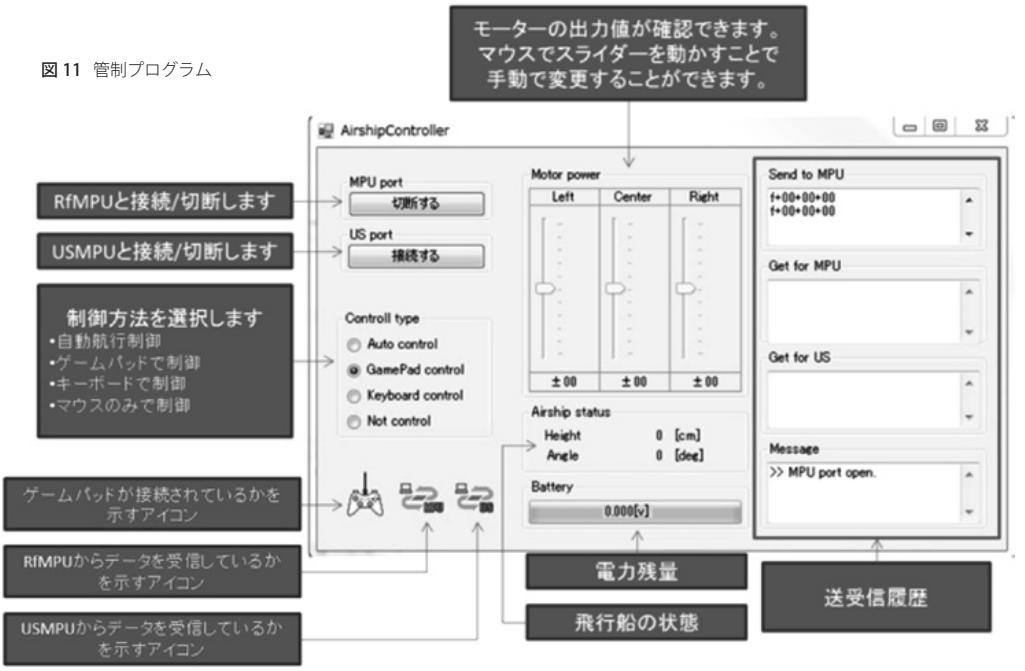
本システムはフロートデイで説明された飛行船ハードウェア構成概要を基本としている。以下，各構成物について，記述する。

H/W：PC（OS : Windows7 64bit），ゲームパッド

開 発 環 境：Microsoft Visual Studio 2010，Microsoft XNA4.0

主に Visual Studio 2010 及び C# 言語を使用し，飛行船の制御を行う。制御に必要なセンサ情報は，RfMPU-5H および US_MPU-5H で通信して取得する。

図 11 管制プログラム



ゲームパッドは手動制御を行う場合に使用する。

ゲームパッドを使用する際には Microsoft XNA 4.0 のクラスを使用する。

図 12 ゲームパッド (Xbox360 コントローラ)

◆ 5.2. Rf_MPU

飛行船と基地局間の通信を中継する。

H/W は配布されたものを使用し，データの送受信は Visual Studio の Serial Port コントローラクラスを用いて行う。

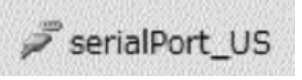
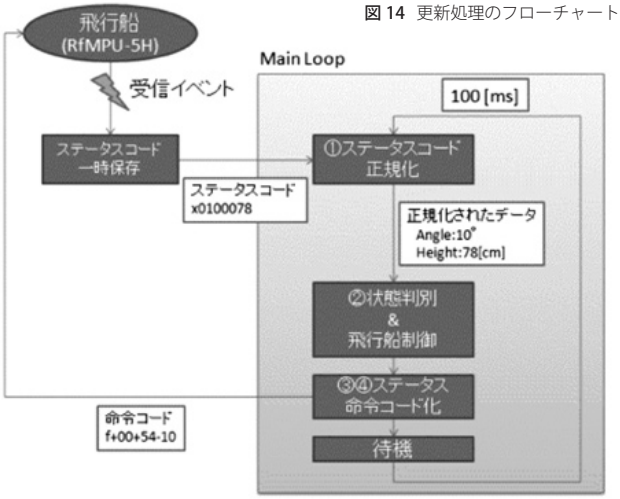


図 13 Serial Port コントローラクラス

データを受信すると受信イベントが発生するので，一旦データを格納する。
飛行船の状態の更新 (制御データの送信) は受信してすぐではなく，決まった間隔時間で定期的に実行されるようにしている。更新処理では以下のような処理を行っている。

図 14 更新処理のフローチャート



1. 格納したデータを正規化する。
2. 正規化されたデータから現在の状態を判断し，データ上のモータを制御する。
3. データ上のモーターの値を送信用データに変換。
4. 送信用データを RfMPU-5H に送信する。

データの送信漏れを考え，モーターの値の変更が無い場合でもデータを送信する。
正常にデータを受け取ったかどうかは，受信データから判別することができるが，それでは遅いのというのも理由のひとつである。

◆ 5.3. 飛行船

配布されたセット（Haluna-5H）を基本として使用。
プロペラの周りにフードを付ける，左右のプロペラの間の幅を拡げるなど，安全性，安定性などを向上させる改造を行っている。

図 15 改造部の写真

◆ 5.4. US_MPU

飛行船が現在どの辺りにいるのか判断するためのデータを送信する。
H/W は配布されたものを使用し，データの送受信は RfMPU-5H 同様，Visual Studio の Serial Port コントローラクラスを用いて行う。
地上局側が受信したデータはそのままでは使用することができないので，一旦格納する。
更新処理でデータを正規化し，過去データと比較して有効な現在の座標を決定する。

◆ 6. メトリックス

MDD ロボットチャレンジ 2010 では，2009 年までの資源を活用してプロジェクトを遂行した。図 16 に今回のプロジェクトの開発に関するガントチャートを示す。モデリングの知識は 4 ～ 7 月の間に KCGI で開講された「オブジェクト指向システム設計」の講義にて修得した。C# プログラミングの知識は，チーム内で勉強会を行い，修得に努めた。

また，ミーティングの日程と活動内容に関して図 17 に示す。2009 年に引き続き，Skype を利用したオンラインでのミーティングも利用した。これまでは活動開始時期が比較的遅く，夏期休暇中の 8 月に集中して活動が行われることが多かった。2010 年は活動開始時期が早かったが，実際に開発を手がけ始めたのは 2009 年と同じ時期の 6 月以降である。また，8 月に空白の期間が生じている。開始時期が早くある程度の準備が整っていたため，プロジェクトの遂行に致命的な支障をきたすことはなかったが，今後の改善点としたい。

	3月中旬	4月	5月	6月	7月	8月	9月上旬	9月下旬	10月上旬
ソフトウェア環境									
モデリング		L	L	L	L				
プログラミング	L	L	L						
ハードウェア環境									
飛行実験									

図 16 ガントチャート（L →勉強会や，講義を受けて知識を修得した）

日程	主な内容
3/12	引継ぎ・メンバー募集について
3/18、25、31 (On)	C#勉強会（ミーティングを兼ねる）・ソフトウェア環境調整
4/8(On)	C#勉強会（ミーティングを兼ねる）
4/16	メンバー顔合わせ・役割分担
5/13(On)	今後のスケジュール・役割調整
5/27(On)	C#勉強会（ミーティングを兼ねる）
6/19、28	ハードウェア調整および学内イベントの打ち合わせ
7/1、5、8、12、15	ハードウェア調整および開発・ソフトウェア環境調整
7/3、9、17、24	学内イベントでの展示（ミーティングを兼ねる）
9/4、8、12、23	モデリング・ハードウェア調整・飛行実験・その他打ち合わせ
9/25(On)	モデリング
9/26	モデリング・ドキュメント作成
10/1(On)	モデリング・ドキュメント作成
10/3、5、6	モデリング・ドキュメント作成・飛行実験

図 17 ミーティング日程（On → Skype を用いたオンラインミーティング）

	2009年		2010年	
モデル図	レビュー回数	修正回数	レビュー回数	修正回数
機能（マインドマップ、ユースケース）	3	2	3	2
機能（クラス図）	5	4	8	7
振舞い（ステートチャート図、シーケンス図）	4	3	4	3

図 18 モデル図レビュー・修正の回数 (MDD ロボットチャレンジ 2009 と 2010 の比較)

モデル図の作成に費やしたレビューおよび修正の回数を図 18 に示す。

参考文献

MDDロボットチャレンジ

[1] MDD ロボットチャレンジ
http://sdlab.sys.wakayama-u.ac.jp/mdd2009/

[2] 社団法人情報処理学会,「MDD ロボットチャレンジ 2004 産学連携による組み込みソフトウェア開発の実践」(社団法人情報処理学会, 2005)

[3] 社団法人情報処理学会,「MDD ロボットチャレンジ 2005 産学連携によるモデルベース組み込み開発の実践」(社団法人情報処理学会, 2006)

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

[5] 高橋修司, 中村州男, 江見圭司ほか,「ドリトルを用いたモデル化・シミュレーション・オブジェクト指向開発の自学自習実践」第 94 回コンピュータと教育研究発表会, (2008)

プロジェクトマネジメント

[6] 岡村正司,「徹底解説 ! プロジェクトマネジメント―国際標準を実践で活かす」(日経 BP 社, 2003)

[7] 金子則彦,「プロジェクトマネージャ完全教本 2009 年版」(日本経済新聞出版社, 2009)

[8] Project Management Institute,「プロジェクトマネジメント知識体系ガイド第 3 版」(Project Management Inst, 2005)

[9] Kim Heldman, PMI 東京支部 (翻訳),「PMP 教科書 Project Management Professional 第 3 版」(翔泳社 ,2006)

[10] Craudia Baca , PMI 東京支部 (編集),「PMP 教科書 問題集 Project Management Professional」(翔泳社, 2004)

モデリング

[11] 竹政昭利,「はじめて学ぶ UML 第 2 版」(ナツメ社, 2007)

[12] 青山幹雄, 中谷多哉子,「オブジェクト指向に強くなる―ソフトウェア開発の必須技術」(技術評論社, 2003)

[13] 山口淳一, 江見圭司,「オブジェクトモデリング (UML) を用いた組み込みソフトウェア開発技術者養成プロジェクト」, IPSJ Symp. vol.2004, pp.171-174 (2004)

[14] 江見圭司, 石井充, 矢島彰,「モデリングを中心にしたオブジェクト指向技術者養成カリキュラム」, IPSJ Symp. vol.2002, No.12, pp.127-132 (2002)

[15] 牛尾剛, 長瀬嘉秀,「オブジェクト脳のつくり方―Java・UML・EJB をマスターするための究極の基礎講座」(翔泳社, 2003)

[16] 平澤章,「オブジェクト指向でなぜつくるのか―知っておきたいプログラミング, UML, 設計の基礎知識―」(日経 BP 社, 2004)

[17] 株式会社テクノロジックアート,「UML 辞典」(翔泳社, 2004)

[18] 林田 幸司 , 國正 聡 ,「失敗事例から学ぶ UML システム開発の鍵」(ソフトリサーチセンター, 2008)

[19] 桐越 信一 , 国正 聡 , 竹政 昭利 , 照井 康真 , 橋本 大輔 ,「UML モデリング教科書 UML モデリング L2 第 2 版」(翔泳社, 2008)

[20] 竹政 昭利 ,「[改訂版] UML モデリング技能認定試験 < 入門レベル (L1)> 問題集 -UML2.0 対応」(技術評論社, 2007)

[21] 株式会社チェンジビジョン社 JUDE/Community
http://jude.change-vision.com/jude-web/index.html

C#, 言語関連

[22] Jay Hilyard,Stephen Teilhet, 鈴木 幸敏 (翻訳),「C# クックブック 第 3 版」(オライリージャパン, 2008)

[23] Jesse Liberty, 首藤 一幸 (翻訳), 鈴木 幸敏 (翻訳), 情報技研 (翻訳),「プログラミング C# 言語解説 第 3 版 」(オライリージャパン, 2004)

[24] UML モデルをどうやって C 言語に落とし込むか
http://monoist.atmarkit.co.jp/fembedded/robocon/etrobo06/road03/road03a.html

その他

[25] キミの設計に「トレーサビリティ」はあるか
http://www.atmarkit.co.jp/im/carc/serial/extend/11/11a.html

江見 圭司
Emi Keiji
経歴はジャーナル 28 ページに掲載。