MDD ロボットチャレンジ 2004: モデル講評

鷲崎弘宜[†1] 満田成紀[†2] 小林靖英[†3]
渡辺博之[†4] 沢田篤史[†5] 二上貴夫[†6]

本稿では，オブジェクト指向に基づいて作成された組込みソフトウェアモデルの評価の枠組みを提案し，その枠組みに基づいて，MDD ロボットチャレンジ 2004 において 6 組の参加チームが開発した飛行船制御ソフトウェアのモデルを比較評価する．また，評価を通じて，モデル駆動開発におけるモデルの可能性を探る．

MDD Robot Challenge 2004: Comments on Developed Models

Hironori Washizaki,[†1] Naruki Mitsuda,[†2]
Yasuhide Kobayashi,[†3] Hiroyuki Watanabe,[†4] Atsushi Sawada[†5]
and Takao Futagami [†6]

In this paper, we propose a framework for evaluating object-oriented embedded software models, and evaluate six models developed at MDD robot challenge 2004 based on the proposed evaluation framework. Moreover, we discuss the future prospects of models in MDD.

1. はじめに

MDD ロボットチャレンジ 2004（以降，MDD2004 と略記）において 6 組の参加チームは，ソフトウェアモデルの作成と洗練を通じて，与えられた，もしくは自ら設計・作成した飛行船ロボットを自律的に制御するソフトウェアを作成し，飛行船の実際の航行によってソフトウェアとハードウェアの協調動作の正しさを実証した．参加したチーム名を以下に示す．以降では，チーム A を T_A，T_A が作成したモデルを M_A とする．

- A: ムンムン考房
- B: すりぃあみぃごず
- C: Mudskipper（プロフェッショナル賞）
- D: FC 専士（最優秀モデル賞）
- E: New Wave System
- F: cannot!!（学生賞）

我々審査員は，航行実験の前に参加チームが提出した各モデルについて，組込みシステムモデルとしての妥当さや，表記上の正しさ，モデル駆動開発（Model-Driven Development: MDD[1),2)]）に従ったモデリングの過程などについて審査を行った．MDD とは，問題領域の知識を着目する側面ごとに抽象化し幾つかのモデルとして表し，モデルを繰り返し変換していくことで実行可能なプログラムコードを導出する開発手法である．MDD の代表例がモデル駆動アーキテクチャ（Model-Driven Architecture: MDA）[3)] である．MDA では，最初にビジネスモデル・要件を表すコンピュータ化独立モデル（Computation Independent Model: CIM）を作成し，ツールなどによって CIM を変換してプラットフォーム独立モデル（Platform Independent Model: PIM）を得た後に，ツールによって PIM をプラットフォーム特定モデル（Platform Specific Model: PSM）に変換し，得られた PSM を PIM とみなして変換を繰り返し行い，最終的に，最後の PSM としてのプログラムコードを得る．

我々は短時間における集中的な議論によって，最初に全モデルについてそれぞれ特徴を抽出し，続いて，抽出した特徴を比較表にまとめて評価し，最終的に以下の各賞を決定した．

- 最優秀モデル賞（M_D）：もっとも優れたモデル．
- プロフェッショナル賞（M_C）：最優秀モデルと比較して劣るものの，実開発において実用的な技法を駆使したモデル．
- 学生賞（M_F）：最優秀／プロフェッショナルモデ

†1 国立情報学研究所, National Institute of Informatics
†2 和歌山大学, Wakayama University
†3（株）永和システムマネジメント, Eiwa System Management, Inc.
†4（株）オージス総研, Ogis-RI Co., Ltd.
†5 京都大学, Kyoto University
†6（株）東陽テクニカ, TOYO Corporation

ルと比較して劣るものの，学生のみによって編成されたチームによるモデルの中で，特に創意工夫されたモデル．

以降において最初に，モデル評価の枠組みを提案し，その枠組みに基づいて各モデルを比較評価する．また，MDD におけるモデルの品質について考察する．最後に，各モデルについて個別に講評する．

2. モデル評価の枠組み

提出されたモデルは全て，オブジェクト指向に基づく分析・設計を経て作成されている．そこで，これまでに提案されている種々のオブジェクト指向モデル評価の枠組み（目的に基づく分析モデル比較[4]，利用した技法に基づく分析モデル評価[5]）や，モデル品質測定法[6]，および，MDD に関する幾つかの独自の観点（マネジメントの観点など）を導入して，MDD とオブジェクト指向に基づく組込みソフトウェアのモデルが備えるべき品質特性の評価の枠組みを以下のように構築した☆．枠組みにおける品質特性の階層化には，国際標準ソフトウェア品質モデル ISO9126[7] を参考とした．

(1) 機能性: モデルが求められる機能を適切に持つ度合いを，以下の品質副特性の判定によって最終的に評価する．

- 合目的性: モデル中に，MDD2004 運営委員会より与えられた要求に基づいて記述すべき事柄が，分析・設計によって漏れなく妥当に記述されているかどうかを判定する．具体的には，オブジェクト抽出の妥当さ，静的側面の記述の適切さ，動的側面の記述の適切さ，並行性の検討の有無，非正常時の処理・対応の有無，および，モデリングの範囲の明確さを審査する．
- 正確性: MDD に従ってモデルから得られた最終プログラムコードが，モデルの表す通りの正しい結果をもたらしているかどうかを判定する．ただし，プログラムコードとモデルの間の具体的な対応付けについて事前に審査することは困難であったため，対応付けの間接的な評価法として，モデルからのコードの自動生成を可能とするツールの使用の有無，および，モデルの実装任意性の低さを審査する．
- 標準適合性: モデルの表記が，使用しているモデリング言語（UML など）や他の様々な記述法（例えば状態遷移表など）に正しく適合しているかどうかを判定する．

☆ 実際の審査時には，合意形成のための十分な時間が取れなかったため，各自が想定する審査基準に従って特徴を書き並べて比較評価するという Adhoc な方法を採用した．

(2) 使用性: モデルの使いやすさを，以下の品質副特性の判定によって最終的に評価する．

- 理解容易性: モデルが意図する事柄の理解しやすさを判定する．具体的には，物理モデル・飛行戦略の記述の有無，モデリング方針・手順（プロセス）の記述の有無，モデル中の個々のモデル要素単体の理解しやすさ，モデル要素の抽象化の妥当さ，モデル要素間の汎化関係の妥当さ，モデル要素間の結合関係の複雑さ（COF），および，複数のモデル図間の関係の理解しやすさを判定する．

(3) 保守性: モデルの将来にわたっての保守のしやすさを，以下の品質副特性の判定によって最終的に評価する．

- 管理容易性: プロジェクトマネジメントの観点から，成果物としての多くのモデルの管理が容易であることを判定する．具体的には，概念から設計にわたっての多様な抽象度のモデルの有無と適切さ，および，抽象度の異なるモデル間の整合性を審査する．
- 再利用性: モデル中のモデル要素やモデル図の再利用のしやすさを判定する．具体的には，モデル中のレイヤ間・パッケージ間の独立性や，モデル要素の抽象化と汎化関係の妥当さを判定する．
- 変更性・拡張性: モデル中のモデル要素やモデル図の変更と拡張のしやすさを判定する．具体的には，モデル中のレイヤ間・パッケージ間の独立性や，モデル要素間の結合関係の複雑さ（COF），および，変更性・拡張性の向上をもたらす分析・設計パターンの利用の有無を判定する．
- テスト容易性: モデルにおいて表すべきことが正確に表されていることを，検証しやすいものになっていることを判定する．具体的には，特定の実装環境に依存しないままでのアクション言語などを使用した動作検証可能な振る舞いの記述を判定する．また，モデル上でのテスト仕様書の有無も評価に加える．

3. モデルの比較評価

モデルの比較評価と総合評価の結果を以下に示す．

3.1 比較評価の結果

評価の枠組みに基づく評価結果の比較を表 1 に示す．**モデル要素間の結合関係の複雑さ**以外の定性的な審査基準については，我々の判断によって 3 段階に評価した．定性的な審査基準の属人性を排した評価法の確立は，今後の課題としたい．また，複数の品質特性に使用される審査基準は，初出の品質特性についてのみ記述した．以降において，各品質特性ごとに考察する．

(1) **機能性**

表 1 モデル評価の枠組みに基づくモデル比較（◎: 優れている，○: 良い，–: 必ずしもよくはない，×: 無い）

品質特性	品質副特性	審査基準	提出モデル					
			M_A	M_B	M_C	M_D	M_E	M_F
機能性	合目的性	カテゴリによるオブジェクト抽出	×	×	○	○	×	×
		問題領域概念のオブジェクト抽出	–	○	○	○	○	○
		静的側面（レイヤ，パッケージ，構造）	–	–	○	◎	○	–
		動的側面（相互作用，状態遷移表など）	–	○	○	◎	○	○
		並行性の検討	×	×	×	–	×	–
		非正常系処理・対応の検討	×	◎	×	○	×	○
		モデリング範囲の検討・明確さ	–	–	○	○	○	○
	正確性	コード生成ツールの使用	×	×	(○*)	○	×	×
		実装任意性の低さ	–	–	(○*)	○	–	–
	標準適合性	使用記述法への適合性	○	–	○	–	–	–
使用性	理解容易性	物理的モデルと飛行戦略の記述	◎	×	◎	×	○	×
		モデリング方針・手順の記述	×	×	◎	×	◎	–
		モデル要素の理解容易さ	–	–	◎	◎	○	–
		モデル要素の抽象化の妥当さ	–	–	○	○	○	–
		モデル要素間の汎化関係の妥当さ	×	×	×	–	–	×
		モデル要素間の結合関係の複雑さ	–	○	○	◎	◎	○
		（Couplig Factor（COF）の測定値）	0.67	0.20	0.22	0.04	0.07	0.15
		モデル図間の関係の理解容易さ	○	–	○	◎	–	–
保守性	管理容易性	概念モデル（多様な抽象度）	×	–	×	×	×	–
		分析モデル（多様な抽象度）	–	–	◎	○	○	–
		設計モデル（多様な抽象度）	–	×	–	–	×	×
		抽象度の異なるモデル間の整合性	○	–	○	○	×	–
	再利用性	レイヤ間・パッケージ間の独立性	×	×	–	◎	○	×
	変更性・拡張性	分析・設計パターンの利用	×	×	–	×	–	×
	テスト容易性	動作検証可能な振る舞いの記述	×	×	(○*)	○	×	×
		テスト仕様書	×	×	×	○	×	×

(*) 審査時に提出されたモデル文書中には，シミュレーションとコード生成を可能とするモデルは存在しなかった．

モデルの合目的性，正確性，および，標準的合成について以下に比較評価する．

- **カテゴリによるオブジェクト抽出:** 分析モデルを作成する際に，担当する責務のカテゴリに着目してオブジェクトを抽出することで，適切に抽出作業を行うことが可能となる．また，オブジェクト間の相互作用も早い段階で明確とすることができる．M_D では，パッケージ分割後の分析モデルについて，システムの処理・制御の単位を抽象化した”制御”（<<controll>>），一時的または永続的なデータの単位を抽象化した”エンティティ”（<<entity>>），物理的装置の単位を抽象化した”ハードウェア”（<<hardware>>），複数のハードウェアを集約する装置の単位を抽象化した”ハードウェア構成”（<<hardware_unit>>），および，他のパッケージを利用する手順と方法の単位を抽象化した”インタフェース”（<<interface>>）の 5 つのカテゴリから，漏れなくオブジェクトを抽出し，各カテゴリを UML のステレオタイプによって表している（図 1）．

一方 M_c では，ロバストネス分析によって，外部とのインタフェースを定義するバウンダリ，半永続的に管理するデータを定義するエンティティ，処理・制御を

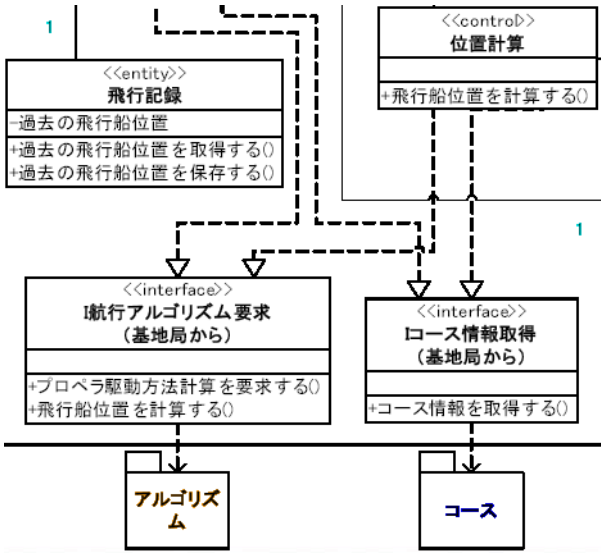

図 1 M_D: 分析モデル中の基地局パッケージの一部

定義するコントロールの 3 種（カテゴリ）からオブジェクトを抽出し，図中でのアイコンの使用によってそれらの種類を表している（図 2）．

- **問題領域概念のオブジェクト抽出:** 概念モデルや

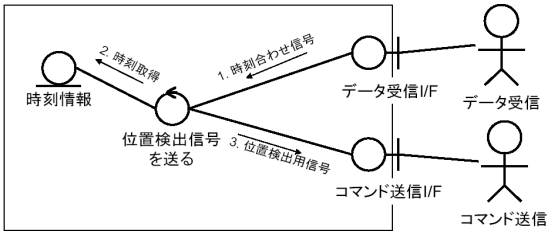

図 2 M_C: 位置検出信号の送信に関するロバストネス図

分析モデルでは，扱う問題領域に特有の概念をクラスとして識別しておくことで，問題領域の理解を明確なものとすることができる．MDD2004 においては，飛行船や基地局，送／受信器，プロペラといった主にハードウェア上の概念が特有なものとして要求仕様書で与えられている．各モデルとも概ね，適切に問題領域概念を識別している．

- **静的側面:** システムの静的側面を適切に捉えるためには，抽出したオブジェクト集合の構造を単純に記述することにとどまらず，システムの階層化とパッケージ分割（ドメイン分割）によって，要点となる構造・振る舞いを適切な単位に分割することが重要である．M_D では，システムを飛行船，基地局，コース，航行アルゴリズムの 4 つのドメインに分割し，さらに，飛行船と基地局についてそれぞれ主にハードウェアから構成されるメカニズムドメインを分割することにより，各ドメインを表すパッケージ単位での静的構造の徹底的なモデル化に成功している．
- **動的側面:** オブジェクト間の相互作用のモデル化には，状態遷移表や，UML におけるシーケンス図・コラボレーション図（コミュニケーション図）の活用が有効である．M_D では，分析モデルにおいてクラスの各操作単位でコラボレーション図を作成し，要点となる相互作用を丁寧に表している．また，同じく M_D では，状態を列，イベントを行にとる状態遷移表を作成して，状態図における記述の抜けや漏れを防いでいる（図 3）．
- **並行性:** 組込みソフトウェアシステムとは，実世界におけるハードウェア（センサ）からの入力に基づいて，何らかの処理を行い，その結果を別のハードウェアを通じて外部に表す系である．実世界はもともと並行性を持つ世界であるため，そのインタフェースとなるハードウェアデバイスと連係動作する組込みソフトウェアもまた，実装段階においてマルチタスク（複数の処理の同時実行）動作することが多い．そのため，分析あるいは設計段階でその並行性を検討し，モデル中で表すことが望ましい．M_D と M_E は，位置情報

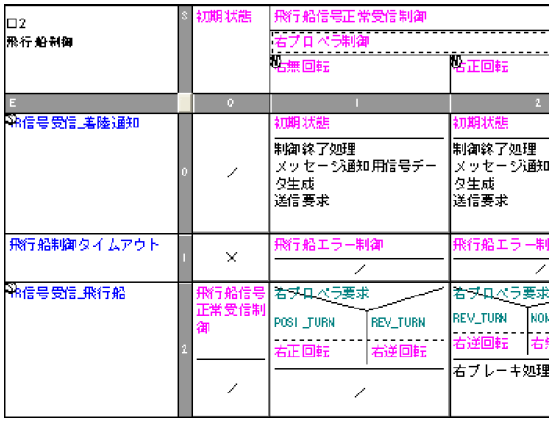

図 3 M_D: 飛行船制御の状態遷移表の一部

検出に関する状態図において，IR 信号と US 信号の両方を並行して送信・受信していることを幾らか記述している．

- **非正常処理・対応:** 組込みシステムのモデルは，ソフトウェアのみで閉じて表されるものではなく，ハードウェアを通じた実世界の現象も含めて表されるため，システムの稼動時に，期待していた（理想的な）現象とは異なる現象・状況が発生することがある．従って，組込みソフトウェアのモデルにおいて，実世界やハードウェアの不具合・想定外利用法などに起因する非正常・異常な状況（非正常系）をあらかじめ非正常シナリオとして抽出し，その対処処理法を検討し記述しておくことが重要である．M_B では，非正常系抽出法[8] の利用によって非正常系におけるイベントと状態の組み合わせを状態遷移表として網羅している．
- **モデリング範囲の検討・明確さ:** 組込みシステムは，ハードウェアとソフトウェアの協調系として表される．従って，分析・設計の段階で，ソフトウェアにおいて扱う範囲をモデル中で明記することが重要である．各モデルは範囲の明記について，ユースケース図におけるアクタと対象システムの境界付けを利用している．また，独自の工夫として，M_F ではハードウェア／ソフトウェア機能分担表（表 2）を，M_C ではロバストネス分析による境界を，M_E では機能割り当て方針の記述をそれぞれ利用している．

なお，MDD2004 では開発する部分を選択できたため，各モデルごとに扱う範囲には大きな違いがある．M_A, M_C では主に基地局のみを，M_B, M_E では主に飛行船のみを，M_D, M_F では飛行船と基地局の両方をそれぞれ扱っている．

- **コード生成ツール:** MDD はモデルとプログラムコードの接続（自動的な導出と追跡可能性）の実現を

表 2　M_F: ハードウェア／ソフトウェア機能分担表の一部

飛行船		
要求	ハードウェア	ソフトウェア
赤外線送信	○	○
赤外線受信	○	○
超音波受信	○	○
モーター回転	○	○
時間の計算	×	○
メッセージ投下	○	○
エラー処理	×	○

主眼とするため，モデルからのコード生成にコード自動生成ツールを使用していることが望ましい．M_D では，作成した設計モデルを PIM としてツール ZIPC[9] に与えてコードを自動生成している．M_C では，モデルの提出時にはコード生成にまで至らなかったものの，モデリングの当初よりモデルコンパイラ MC-3020 Model Compiler[10] の使用を明記し，また，モデルの提出後に，PIM としての設計モデルからコードを自動生成している．

- **実装任意性の低さ:** MDD におけるモデルは，変換後に対応する最終的な実装コードについて異なるバリエーションが存在する可能性が低いことが望ましい．M_C では，ロバストネス図中の構成要素に一対一に対応するクラスの作成と，アクション言語による振る舞いの厳密な記述により，設計モデルの実装任意性が低いと考えられる．
- **標準適合性:** モデルは，開発チーム内外における対話や，将来における第三者によるソフトウェアの保守に用いられるため，使用する表記法に厳格に従っていることが望ましい．全てのモデルは，使用した表記法（UML や状態遷移表の書き方）に概ね従っていた．ただし，M_B において本来オブジェクト図とすべき図がクラス図として描かれていること，M_D において状態図がフロー図となっていること，M_E においてシーケンス図中のオブジェクトの生存ラインが適切でないことといった軽微な不具合は見られた．

(2)　**使用性**

モデルの理解容易性について以下に比較評価する．

- **物理的モデルと飛行戦略の記述:** モデル中で最初に，組込みシステムとしての飛行船をどのように飛ばすのかという飛行戦略や，その背景にある物理モデルへの言及があれば，以降における分析・設計の流れが分かりやすいものとなる．例えば M_A では，要求仕様において抜けていた飛行船座標計算方法をその物理モデルとともに表すことで，飛行戦略を明確なものとしている（図 4）．
- **モデリング方針・手順の記述:** モデルにおいて，

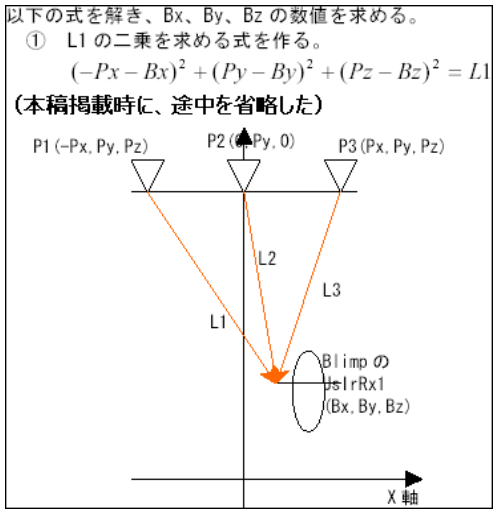

図 4　M_A: 飛行船の座標計算方式（抜粋）

モデリングの結果としてのモデル図のみではなく，モデリングの方針・手順（プロセス）が記述されていれば，そのモデルは第三者が見て分かりやすく，また，手順や方針を再利用しやすいものとなる．M_C では，モデル図を補足する文章において，MDD に従ったモデル作成からコード実装に至る手順と方針を明記している．M_E では，設計モデルの作成には至っていないものの，モデルの最初において分析・設計の方針を明記している．

- **モデル要素単体の理解容易さ:** モデルを構成する要素の分かりやすさには，要素名の妥当さや，持つ操作・属性の正確さなどが関係する．M_D では，分析モデルの段階で各クラスについて操作や属性が正確に明記され，また，要素名も妥当であるため，個々のクラスの意図や責務が明確なものとなっている．
- **モデル要素の抽象化の妥当さ:** モデル要素の過度の抽象化は，理解性の低下につながるため避けるべきである．適切な程度の抽象化によって，着目する本質のみを取り出してモデルの汎用性・利用性を高めることが重要となる．モデル要素がハードウェアを表す場合は適度な抽象化によって，モデル全体に対するそのハードウェアの変更影響を受けにくいものとする．例えば M_C では，モデル文書の最初において，赤外線などによる低レベルな通信処理を扱わないことを明示し，以降のモデルにおいて，”データ受信”や”コマンド送信”という幾らか抽象的なクラスを利用している（図 2）．この抽象化は，問題領域における枝葉末節（低レベルな通信処理）を取り除き，代わりにデータ／コマンドの送受信という本質の識別に成功している．
- **モデル要素間の汎化関係の妥当さ:** モデル要素間

の汎化関係は，構造を複雑にするため多用すべきではないが，明らかに共通するクラス集合の整理や，依存関係の適切な設定を行う目的で用いる．オブジェクト指向における依存関係逆転の原則（Dependency Inversion Principle: DIP）[11] に従い，モデル中の具体的な要素は抽象的な要素に依存し，その逆方向の依存関係は存在しないことが望ましい．例えば，モデルにおける信号送信機の抽象化についてみてみると，M_D では抽象度の高い”I 送信機”（インタフェース）と，それを実装した具体的な”送信機”を用意し，外部のデバイスとの依存性は送信機インタフェースに持たせることで，幾らか DIP を満たしている．

- **モデル要素間の結合関係の複雑さ（COF）:** モデルを構成する要素間の関係が複雑であるほど，モデル全体の理解容易性は低下する．要素間の関係の複雑さを測定する方法として，オブジェクト指向クラスモデルのための結合度測定法 Coupling Factor（COF[12]）がある．COF は，継承関係を除いた実際のクラス間結合関係の多さを測定する．関連を持ったクラス集合 S の複雑度 $COF(S)$ は次式で与えられる．

$$COF(S) = \frac{\text{(継承を除く有向参照辺の数)}}{|S|^2 - |S| - 2\text{(全クラスのサブクラスの数)}}$$

例えば，図 5 の例では，3 つのクラスから構成されるシステム S, S', S'' の COF はそれぞれ 1/4, 1/3, 6/6(=1) となり，有効参照辺によって完全グラフを形成している S'' が最も複雑と判断できる．

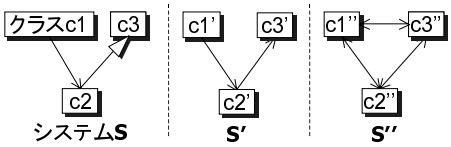

図 5 COF の測定例

COF を，分析モデル（M_A, M_B, M_D, M_E）中，設計モデル（M_C）中，または，概念モデル（M_F）中で構造を表すクラス図にそれぞれ適用した結果を表 1 に示す．測定にあたり，主に分析モデルにおける無向の関係については，両方向に有向参照辺が存在するものとした．測定値の比較から，結合関係の複雑さについて M_D が最も低く優れていることが分かる．この結果は，レイヤ化とパッケージ分割によってモデル要素間の独立性が保たれていたことによるものと考えられる．一方，M_A は最も複雑であると判定された．これは，分析モデルに登場するクラスが 4 つと少なく，かつ，それらのクラスが密に関連していたためである．測定法を適用したモデルの種類が統一されていないため，得られた測定値は単純に比較できるものではないが，COF は上述のようにモデリング上の複雑さを幾らか反映していると考えられる．COF による複雑さの測定は，モデリングコンテストにおける審査基準の定量化の 1 つの試みである．

- **モデル図間の関係の理解容易さ:** モデルが複数の図や文章から構成される場合に，それらの対応関係の明確さは，モデル全体の理解容易性に直結する．M_D では，徹底したパッケージ分割とパッケージ単位での詳細化により，モデル図間の対応関係を明確にしている．M_C では，モデル図を補足する文章において，モデル図間の対応・導出関係を丁寧に説明している．

(3) **保守性**

モデルの管理容易性，再利用性，変更性・拡張性，および，テスト容易性について以下に比較評価する．

- **概念モデルの有無と適切さ:** 概念モデルとは，問題領域中の概念と概念間の関係を主にクラス図で表したモデルである．概念モデルは，問題領域の範囲付けや整理，異なる利害関係者間での理解の統一を図るために，主に要求分析の初期段階で作成される．提出モデルの多くは，概念モデルを明示せずに分析・設計モデルから始まっている．
- **分析モデルの有無と適切さ:** 分析モデルとは，実装環境に依存しない形で，システムの構造や振る舞いを捉えたモデルである．分析モデルでは，要求や問題領域を的確に整理し，本質的な部分を正確に記述できていることが重要である．分析モデルの適切さについては，特に M_C および M_D が優れている．M_C では，分析法としてロバストネス分析を用いて，的確なオブジェクト抽出と構造・振る舞いの記述を行っている．M_D では，分析モデルにおけるハードウェア構成を考慮した階層化と，各階層における徹底的なパッケージ分割によって，システムを適切な単位に分割してモデル化し，将来の変更影響の局所化に成功している．
- **設計モデルの有無と適切さ:** 設計モデルとは，想定する実装環境において分析モデルで表した事柄をどのようにシステムとして実装するのかを表したモデルである．時間が不足がちであったことなどから，実装を意識した設計を明記したモデルは半分にとどまった．例えば図 6 は，M_A における C++言語による実装を意図した設計モデル中のシーケンス図を示す．
- **抽象度の異なるモデル間の整合性・追跡可能性:** モデル全体が複数の異なる抽象度のモデルから構成される場合に，それらの抽象度をまたいだモデル間の対応関係と整合性・追跡可能性が保たれていれば，マネ

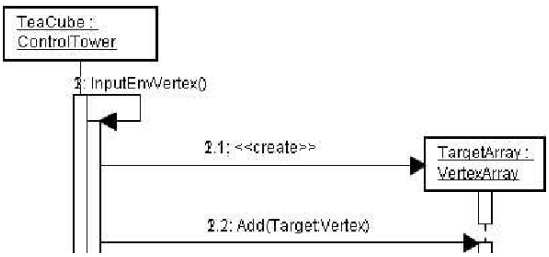

図 6 M_A: 設計モデル中の航行シーケンス図（抜粋）

ジメントの立場から全体を管理しやすいものとなる．また，MDD に沿った開発を目指すならば，手動または自動による変換前後における抽象度の異なるモデル間で対応関係が取れていなければならない．M_C では，ユースケースごとにロバストネス図を作成し，同図中の構成要素に一対一に対応するクラスを作成することで，抽象度をまたいだ整合性と追跡可能性を保つことに成功している．

- **レイヤ間・パッケージ間の独立性:** モデル内でレイヤ化・パッケージ分割の技法が使用されている場合に，レイヤ間・パッケージ間の依存性の低さは，モデル内の特定部分の再利用や，将来における変更影響の局所化などを促進する．それらの技法が使用されているモデルの中で，特に M_D では，パッケージをまたいだ参照関係にインタフェースを使用するなどの工夫により，各パッケージの独立性を高めている．
- **分析・設計パターンの利用:** 変更性や拡張性といった非機能特性の向上には，その向上に役立つことが経験によって実証済みの分析・設計パターンを利用することが効果的である．例えば M_C, M_E では共通に，経過点から構成されるコース情報のモデル記述に，通過点の接続に関する接続パターン[13] が暗黙に利用されている（図 7）．ただし，その利用をモデル中に明記していないため，利用したパターンやパターンの影響・意図を把握することが困難である．組込み分野では数多くの分析・設計パターンが公開されつつあるため，今後はモデル中での明示的な利用に期待したい．

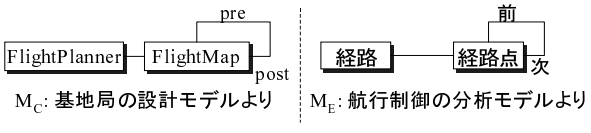

図 7 接続パターンの暗黙の利用

- **動作検証可能な振る舞いの記述:** MDD は，コードではない段階におけるモデルの振る舞いを，実装コードの振る舞いに結びつけることを目指す．そのためモデルに対して，実装環境に依存せずに検証可能なように振る舞いを記述する必要がある．M_C では，状態図についてアクション言語を用いて振る舞いを命令的に記述し，モデル動作の検証をツール BridgePoint[10] によって可能としている．M_D では，状態遷移表を記述し，状態遷移表のシミュレーションをツール ZIPC によって可能としている．
- **テスト仕様書:** モデルについて動作検証を行う際には，確認したい事柄をテストケースとして作成することがある．M_D では，状態遷移表のシミュレーション時に使用したテストケースを，シーケンス図を用いてモデル化している．

3.2 総 合 評 価

以上の評価に基づき，審査員の投票によって最優秀モデル賞は M_D に授与された．表 1 に示すように，M_D は機能性，使用性，保守性の全てに優れていた．また MDD の考え方に従って，設計モデル上でのシミュレーションの実施や，PIM としての設計モデルから実装コードを自動生成している点も評価できる．

プロフェッショナル賞は M_C に授与された．M_C は，提出時には振る舞いのモデルが不十分であった点が残念なものの，ロバストネス分析の活用による属人性を排除したクラス導出，アクション言語による振る舞いの記述とモデルシミュレーション，ツールによるコードの自動生成など，MDD にもっとも準拠したモデル作成プロセスを採用している点が評価できる．

学生賞は M_F に授与された．M_F は，実務家がチーム編成に加わっている上記の 2 つのモデル（M_C, M_D）に比べて劣るものの，分担表の作成によってハードウェア／ソフトウェアがそれぞれ担う役割を十分に分析しているなどの独自の工夫がなされており，学生主体の他のチームと比較して，モデル全体の完成度が幾らか高いことが評価された．

3.3 MDD らしさとは

MDD2004 の本来の主旨に照らし合わせれば，各モデルの”MDD らしさ”を評価することが望ましい．上述のモデル評価の枠組みは，主にオブジェクト指向と組込みシステムの観点に基づく．しかしながら同じ枠組みの中で，**コード生成ツールの利用**，**実装任意性の低さ**，**抽象度の異なるモデル間の整合性**，**動作検証可能な振る舞いの記述**，および，**テスト仕様書**の 5 つの審査基準について，特に MDD の観点から満たされることが望ましいことを示した．これらの審査基準は，MDD がモデルの多段階変換を経る開発手法であること，および，MDD では早い段階で動作検証可能なモデルを得ることに基づいている．特に後者の特徴は，

プログラムに残存する不具合が多大な損害をもたらす組込みソフトウェアの分野においてその重要性を増し，MDD を組込みソフトウェア開発に導入する大きな理由になると考えられる．

また本稿では，モデルの自動的な変換について，”モデル → コード”の変換についてのみ言及した．これは主に，MDD2004 運営委員会および参加チームが，飛行船制御ドメインにおけるメタモデルとモデル要素間の変換規則の定義にまで踏み込むことができなかったことに起因する．ただし，M_C におけるロバストネス図からクラス図への対応付けのように，手動的な”モデル →（コード以外の）モデル”変換の形跡は幾つか見受けられた．今後，MDD 処理系の発展と同ドメインにおけるモデルの蓄積を通じて，メタモデルとモデル自動変換について評価・議論されることを期待したい．

4. モデルの個別講評

本章では，各モデルの特徴を抜粋して解説する．節題には，モデルの講評を担当した審査員名を付記した．各モデルの詳細については，本稿が掲載される MDD2004 開催報告[14] における各チームの参加報告書を参照されたい．

4.1 T_A: ムンムン考房（講評: 小林靖英）

特徴として，空中にある飛行船の位置（座標）をどのように求めるかという物理的な要求を論理的に分析，詳細化していることが挙げられる．また，飛行戦略として，状態認識を汎用化することで，複雑なコース飛行を単純化してモデル化することにチャレンジしている．この 2 点は,「どのように飛ばすか」ということの論理がよく表現されたものである．

しかしながら，状態認識を汎用化したことによってか，分析，設計が曖昧なまま終わってしまったのは残念である．必要とされる技術要素，環境といったところの表現が少なく，設計における状態認識としても，通信状態におけるふるまいが定義されていないのは実装につながらないのではないかと考えられる．

開発チームの思考過程はよく表現されおり，悩みながらも工程を進めていったことがうかがえる．しかし，その思考過程が，各工程で必要とされる設計表現・モデル化につながっておらず，設計書としての物足りなさとなっている．言い換えれば,「モデルを読ませる」ことへの注力が不足していたとも言え，開発プロセスとそこで必要な要素のモデル化について再検討を期待する．それは例えば，この設計書からテストケースが抽出できるのかといったことからでもよいだろう．

とにかく,「状態を汎用化したい，さすればこのように計算し制御すればうまく飛ぶはず」という強い意志と解決への悩みが感じられるモデルである．状態認識の汎用化へのチャレンジもそうであるが，学生メインのチームで，模擬飛行体の製作，実験による開発へのフィードバックを行ったり，独自に通信環境を製作するなど，開発努力は素晴らしいものがある．次回は，思考過程の表現から，開発プロセスと必要なモデル化へと，さらなるエンジニアリングへの挑戦を期待する．

4.2 T_B: すりぃあみぃごず（講評: 渡辺博之）

ソフトウェアの開発にモデルを用いる目的はさまざまである．しかし，一番重要なことは，開発対象の本質を明らかにし，何を作ろうとしているかを誰にでも分かりやすく提示することであろう．本チームのモデルは，データフロー図，クラス図，ユースケース図，シーケンス図，状態図など，さまざまな図から構成され，多様な側面からモデル化していることが見て取れる．個々の図の完成度は決して高いとはいえないが，ソースコードよりは明らかに高い抽象度で記述されており，チームの開発現場において，コミュニケーションや設計意図の共有に，大いに貢献できたのではと推測する．そういった意味では，冒頭で述べたような，モデルが果たすべき役割は十分担うことができているモデルといえよう．

しかし，モデル開発における，最初のハードルは突破できたと思われる本チームであるが，本来 MDD で目指すべきモデルにはまだ近づけたとは言いがたい．MDD で使用されるモデルには，抽象度を適度に保ちながらも，その抽象度における表現は正確であり，かつ必要な情報をすべて網羅していることが求められる．この点において，本チームのモデルは，モデルとして表現すべき情報が大きく不足していた．特に，要素の定義が皆無（属性，関連名など）であることは，上述した動作検証をまったく不可能にしてしまった．

また，モデルをオブジェクト指向という視点から評価すると，クラス図等における「抽象化」が十分でない点も気になった．抽象化することで，一時的に理解が困難になる場合もあるが，モデルに含まれる個々の要素をシンプルにすることでモデルの汎用性も高まり，最終的な開発効率の向上につなげることが可能になる．次のステップとして，ぜひ挑戦してもらいたい．

まとめると，本チームのモデルは，アイディアの「スケッチ」レベル，つまり，本来モデルが持つべきであるコミュニケーション手段としてのモデルとしては，十分評価できるものであったといえる．しかしながら，今後，MDD としてのモデルを目指すためには，その

ような「スケッチ」レベルから「設計図」レベルへの進化が必要であろう．

4.3 T_C: Mudskipper（講評: 沢田篤史）

Mudskipper チームのモデルに対する印象を一言で述べると,「プロの仕業」である．この感想は審査員一同で一致したものであった．具体的にこの印象を与えた原因は次にあると考える．

- MDD の構築プロセスに素直に従いながら，モデル構築を行っている．
- 構築されたモデルの可読性が高い．
- モデルの間の関係が明瞭である．

MDA/MDD におけるモデル構築プロセス（CIM → PIM → PSM）の流れの中で，本チームでは，CIM としてユースケースを構築し，ユースケースの吟味にはロバストネス分析手法を用いロバストネス図を導いている．このロバストネス図はいわば PIM に相当するが，これからオブジェクト指向開発での PSM であるところのクラス図を導出する．ここではロバストネス図の要素および要素間関係が，機械的にオブジェクトクラスおよびクラス間関連に対応付いている．さらに，このクラス図に対し，振舞い仕様を状態遷移図（特定の動作環境に依存しない PIM）として与える作業が続き，最終的にはモデルコンパイラを用い，PSM 的位置づけであるところのソースコードが導出される．

残念ながら，後半の振舞い仕様を与える部分が審査時には間に合わなかったため，詳細な評価はできなかったが，前半部分の各プロセスの手順（設計判断基準）は明瞭に説明されており，結果としてモデル間の追跡性も高いものとなっている．

本チームのプロダクトを見ると「MDA/MDDって簡単，使えそう」という印象を受けるかも知れない．しかしながら，ロバストネス分析を用いたユースケースの吟味や状態遷移図によりオブジェクトの振舞いを与える部分などには，経験に裏打ちされたノウハウが込められているはずである．それを読者に感じさせず「簡単」と思わせてしまうところも「プロの仕業」と言えるかも知れない．いずれにせよ本チームのプロダクトは MDA/MDD の実践結果を示した好例であるといえよう．

4.4 T_D: FC 専士（講評: 二上貴夫）

初見で読みやすいモデルであると審査委員が共通の評価を下した．また，飛行船制御に必要な技術要素をよくカバーしているとの意見も多かった．

飛行船を制御するには，複数のプロセッサが関与する必要があり，モデルとして単純ではない．それら要求される問題領域を複数のドメインと捕らえて的確にモデリングしている．これは，ソフトウェア工学の原則によく一致している上に，分割も適切と評価された．同時に，開発工程ごとに必要とされるモデルが揃っていることも優れて実用的であるとみなされた．

また，当初のモデリング宣言で公開したとおりの UML と状態遷移表を組み合わせたモデルとなって初心をよく貫徹している．チャレンジ・チームリーダたちの運営成果といえる．しかしながら，詳細を見ると状態モデルが表現しているものが手続きのフロー図になっていることや，使われている用語の意味がドキュメント中に定義されていないなどモデルの完成度は組込みの MDD としてはモデル作成後の手作業工程が必要なレベルを超えるものではなかった．

また，搭載プロセッサ周りのハードウェアを製作するところまでテーマとして踏み込んだ挑戦は貴重なプロジェクト事例となった．ソフトウェアだけに留まらないシステム情報処理工学への試金石と言える．その努力とエンジニアリングへの熱意に敬意を表したい．

4.5 T_E: New Wave System（講評: 鷲崎弘宜）

本モデルは，飛行船を構成するハードウェアの制御方法と，飛行船全体の航行方法の関係を分かりやすく示している．本モデルでは，与えられた課題を満たすための航行制御アルゴリズムの概要を最初に明記することで，モデルの以降の部分が表す事柄を明らかとしている．また，設計方針として，責務を割り当てる際の複数のバリエーションを考案していることからも，思考過程が明確となっている．

分析モデルでは，上述の方針に基づいて，主にハードウェアに着目したクラス抽出とクラス間の構造および協調関係を記述している．プロペラや推進器といったハードウェアを幾らか具体的なクラスとして識別し，それらを制御する様子を具体的に分かりやすく示している．本モデルでは，モデリングの範囲として飛行船のみを扱っているため，それらのハードウェアの制御は本質的な関心事の一つであるといえる．従って，この具体さは抽象度として適切なように思われる．

ただし，審査時に提出された分析モデル中では垂直制御の方法が描かれていなかったため，MDD の考え方に従ってモデルの自動（または手動の）変換作業によりコードを導出することは困難であり，モデルの完成度としてはやや不十分といえる．また時間の都合から，モデルの作成が分析モデルにとどまり，設計モデルの作成にいたらなかった点も残念である．

結論として本モデルは，MDD に従ったモデルの段階的詳細化と実現可能性について不十分であるが，将来のコード実装を見据えたモデリングへの取り組み方

について良いものとなっている．次回は，今回の進行を踏まえた実現性のある開発計画を期待したい．

4.6 T_F: cannot!!（講評: 満田成紀）

学生チームであるためか，メンバーの経験とは関係なくモデル記述には UML を選択している．このような場合，往々にしてただ表記を真似るだけで，アプリケーションに合せた適切な使い方にまではいかないことが多い．しかし，このチームの場合は，組み込みシステムであることを十分に意識したモデル記述を行っている．

まず評価したいのは，要求分析の段階でいきなりユースケース図を書き始めるのではなく，その前に，仕様からあきらかなハードウェアとソフトウェアの機能分割を行っていることである．組み込みシステムの場合，ソフトウェアの都合でハードウェアに機能追加を行うことはまず在りえないため，要求レベルでハードウェアの機能をあきらかにしておくことは重要である．

続いて，ハードウェアとソフトウェアの機能分割にもとづいてユースケース図を作成し，個々のユースケースに対してシナリオ記述を行っている．ここでも組み込みシステムらしいアプローチがとられている．それは，思いつくかぎりの例外シナリオを記述することである．多くの組み込みシステムでは，例外発生時の対処が重要となっている．このチームは，例外シナリオとして例外発生の要因とその解決方法を記述することで，ソフトウェアによる例外解決機能を十分に認識している．

その後，設計・実装のフェーズでも UML の図をいくつか用いているが，モデル主導というよりは，自分たちの考えた設計や実装を図にしてみて問題がないか確認するために使っていたように感じられる．この辺りは，今後経験を積むことで，要求レベルのモデルと設計・実装レベルのモデルの間の関係性を積極的に活用した開発が可能になると思われる．

このチームは学部生 3 名と院生 1 名からなる学生チームであるが，院生がプロジェクトマネージャに徹することで，明確な分業体制を維持できたという話である．コンテスト翌日のワークショップにおいては，この辺りの知見についても興味深い議論が行われた．

5. おわりに

本稿では，開発方法として MDD を用いる際のオブジェクト指向に基づく組込みソフトウェアのモデル評価の枠組みを作成し，その枠組みを利用して，MDD2004 における飛行船制御ソフトウェアのモデルを比較評価した．一般に組込みソフトウェアは，規模や環境・要求が多様であるため，そのモデルを比較評価しにくい．本稿および各参加チームの参加報告資料が，同一ドメインにおけるモデルの比較を通じて，組込みソフトウェア開発および MDD における”良い”モデルのあり方を模索する一助となることを期待する．なお，提案したモデル評価の枠組みは，MDD に限らず，一般の開発方法に沿った組込みソフトウェアモデルの評価にも応用可能であると考えられる．

参 考 文 献

1) Bran Selic. The Pragmatics of Model-Driven Development, IEEE Software, Vol.20, No.5, pp.19–25, 2003. （邦訳）ブラン・セリック. モデル駆動開発の実用性, MDD ロボットチャレンジ 2004, 情報処理学会, 2005.
2) 山田正樹. モデル駆動開発とその周辺, 情報処理, Vol.45, No.1, 2004.
3) Object Management Group. MDA Guide Version 1.0, 2003. http://www.omg.org/mda/
4) 渡辺博之. パネルディスカッション: 組込みシステムのモデリングテクニック, オブジェクト指向シンポジウム 2000（OO2000）パネル資料, 2000.
5) 赤坂英彦, 組み込みシステムのモデリングテクニックを検証する, オブジェクトの広場, オージス総研, 2002. http://www.ogis-ri.co.jp/otc/hiroba/technical/Interface/
6) 小坂優, 渡辺博之, 井出幸次. UML モデル品質測定の指標, 組込みソフトウェアシンポジウム 2004.
7) ISO/IEC Standard ISO-9126. Software Product Evaluation-Quality Characteristics and Guidelines for Their Use, 1991.
8) 三瀬敏朗 他. 組込みソフトウェア仕様抽出のための非正常系分析マトリクス, 組込みソフトウェアシンポジウム 2004, 2004.
9) キャッツ株式会社. ZIPC, http://www.zipc.com/product/zipc/
10) Accelerated Technology. Nucleus BridgePoint Development Suite, http://www.acceleratedtechnology.com/
11) Robert C. Martin. 瀬谷啓介訳. アジャイルソフトウェア開発の奥義, ソフトバンクパブリッシング, 2004.
12) Fernando Brito e Abreu, et al. Toward the Design Quality Evaluation of Object-Oriented Software Systems, Porceedings of the 5th International Conference on Software Quality, 1995.
13) Leon Starr 著, 二上貴夫, 長瀬嘉秀 監訳. Executable UML 実践入門, CQ 出版社, 2004.
14) MDD ロボットチャレンジ編集委員会. MDD ロボットチャレンジ 2004: 産学連携による組込みソフトウェア開発の実践, 情報処理学会, 2005.