情報処理学会論文誌 Vol.55 No.1 72–84 (Jan. 2014)

オブジェクト指向モデリング教育における モデル駆動開発ツールの活用方法の検討

赤山 聖子[1,a)] 久住 憲嗣[2,b)] 部谷 修平[1] 福田 晃[3]

受付日 2013年3月19日, 採録日 2013年10月9日

概要：モデル駆動開発（MDD）ツールを用いることで，UML モデルからコードの自動生成が可能になり，モデルの作成から動作検証までの時間を短縮することができる．オブジェクト指向モデリングの教育に MDD ツールを用いることは，学習者がモデリング作業に集中できるというメリットがあり，モデリングスキルの向上に役立つと考えられる．本研究では，MDD ツールを用いないグループと用いるグループに分けて UML モデリングを教育する講座を実施した．受講生の成果物や作成過程を比較することで，MDD ツールを用いる場合と用いない場合それぞれの利点を明らかにし，その結果を基にオブジェクト指向モデリング教育における効果的な MDD ツールの活用方法を検討する．

キーワード：UML，モデル駆動開発，オブジェクト指向，モデリング，学習支援

An Empirical Study of the Usage of Model-Driven Development Tool for an Object-Oriented Modeling Education

Seiko Akayama[1,a)] Kenji Hisazumi[2,b)] Shuhei Hiya[1] Akira Fukuda[3]

Received: March 19, 2013, Accepted: October 9, 2013

Abstract: Model-driven development (MDD) can verify the accuracy of models and generate the source code, which allows a programmer to reduce the development time required to check the software so he or she can focus on the modeling process. Thus, modeling should be taught with MDD because it allows students to acquire modeling skills in a short period of time. We conducted a course to educate UML modeling for the two groups. The first group members use a MDD tool, and the second group members do not use it. We clarify the advantages of each case with and without the use of the MDD tool. From these results, we propose the effective use of MDD tools in UML modeling education.

Keywords: UML (unified modeling language), MDD (model-driven development), object-orientation, modeling, learning support

1. はじめに

近年，ソフトウェアの設計品質の向上のため，設計にモデリング技術を活用することがソフトウェア開発における大きな流れになっている [1]．1990 年代半ばからソフトウェアシステムのモデリングに用いられる言語として UML（Unified Modeling Language）がデファクトスタンダードとなっており [2]，UML を用いたオブジェクト指向モデリング（以下，モデリング）教育を行う試みは数多くなされている．ただし，モデリングの教育コースの多くは，モデリング手法（品質の高いモデルを描く方法や手順）よりも UML の文法に焦点が置かれていることが多く，産業界の要求と教育機関での教育内容のミスマッチが言及されている [3]．モデリング教育に関する研究としても，教育向けの CASE ツールに関するものが多く [4], [5], [6], [7]，モデルの種類や機能を制限する，文法ミスやモデル間の一貫性の

[1] 九州大学大学院システム情報科学府
Graduate School of Information Science and Electrical Engineering, Kyushu University, Fukuoka 819–0395, Japan
[2] 九州大学システム LSI 研究センター
System LSI Research Center, Kyushu University, Fukuoka 819–0395, Japan
[3] 九州大学大学院システム情報科学研究院
Faculty of Information Science and Electrical Engineering, Kyushu University, Fukuoka 819–0395, Japan
a) seiko@ktec.ac.jp
b) nel@slrc.kyushu-u.ac.jp

欠如などに対しアラートを表示させる，などの機能が一般的であり，文法の習得に焦点をあてたものが多い．

モデリング手法の教育に関する研究として，クラス図からオブジェクト図を自動生成することで，クラス図の理解を深める試み [8] や，ある限定された問題に対して想定されるクラス図を事前に定義しておき，それと比較した結果を提示することで，モデリングスキルの向上を図る試み [5] がなされている．しかし，これらの研究はクラス図やオブジェクト図といった静的モデルに関するものが多く，動的モデルに関する教育の研究はほとんど行われていない．静的，動的両面のモデリングスキルの向上を目的として，モデリング教育にモデル駆動開発（Model Driven Development: MDD）および MDD ツールを利用する試みがある [9]．MDD では，モデル上での検証とコードの自動生成が可能で，作成したモデルをすぐに実行して確認することができるため，モデルの機能テストが可能である．また，設計と実装を完全に分離することができるため，モデリングに集中して開発できる．MDD をモデリング教育に活用することで，学習者はモデルの記述だけで動作検証を行うことができるため，開発対象のアプリケーションのモデリング作業に集中でき，学習効果をあげることができる．

一方，MDD をソフトウェアモデリング教育に活用する場合の問題点としては，学習者が MDD で評価できる機能面の完成に強く意識をとらわれることで，モデルの品質への意識が疎かになってしまうことがあげられている [9]．

本研究の目的は，モデリングを行う場合に MDD ツールが及ぼす影響を分析し，モデリングスキルの向上が図れる利用方法を検討することである．なお，本論文では UML で記述されたモデルからコードを自動生成できる機能を備えたツールを MDD ツールと呼ぶこととする．分析にあたり，MDD ツールを利用することができる実験群と利用することができない対照群に分け，モデルの作成演習を行う．被験者が作成したモデルおよびモデルの作成履歴を比較することで，MDD ツールを活用した開発の長所と短所を明らかにし，MDD ツールの効果的な活用方法を考察する．

以下，2 章では教育対象と学習目標，3 章では関連研究と本研究の関係を述べる．4 章では本研究で用いた MDD ツールの内容を述べ，5 章では実験内容，6 章では実験結果を示し，7 章では実験結果をふまえた考察を述べる．

2. 教育対象と学習目標

本研究では，MDD ツールを用いてモデリングの学習支援を行う方法を検討することであるが，本章では，学習支援を行う教育対象と学習目標を述べる．

2.1 教育対象

教育対象は，ソフトウェアエンジニアを目指しているモデリング初学者である．

UML モデリング推進協議会（UMTP）では，UML モデリングのスキル/知識体系を下記のように L1〜L4 の 4 段階定義している [10]．

L1 簡単な UML モデルの意味が分かる．

L2 UML モデルの読み書きが普通にできる（モデリングリテラシーがある）．

L3 実務でモデリングが実践できる．

L4 実践に基づいてモデリングを指導できる．

本研究で想定するモデリング初学者とは，UMTP の L1 程度のスキルレベルであり，UML の基本的な表記法の知識はあるが，ソフトウェア開発におけるモデリングを行った経験がない学習者とする．

2.2 学習目標

本研究では，UMTP の L2 程度のモデリングスキル習得に向けた入門教育を想定している．L2 のスキルレベルは「開発範囲の一部を担当し，モデリングができ，他者のモデルの意味を理解できる」とされている．これをふまえ本研究では，UML の静的・動的モデルを用いて，簡単なシステムをモデルとして表現できるスキルを身に付けることを学習目標とする．ただし，静的モデルとしてクラス図，動的モデルとしてステートマシン図を対象とする．モデリングスキルとは，品質の高いモデルを描くことができる能力であるとし，モデル品質は，文献 [11] に定義された Syntactic，Semantic，Pragmatic の 3 種類を用いる．それぞれ，表記法に正確に従っていること，要求に対してモデルの表現内容が正確であり一貫性があること，第三者が容易に理解できるモデルであることとされている．表記法に関しては，確認が容易であるため，特に Semantic，Pragmatic 品質に関する学習を行う．

具体的な学習目標は，下記のとおりとした．

責務分割 クラス図の作成において，クラスに責務を割り当てることができる．

クラス名 第三者に責務が分かるクラス名を付けることができる．

状態名 各クラスのオブジェクトのライフサイクルにおける各状態に適切な状態名を付けることができる．

遷移 状態の変化を引き起こすイベントを定義することで，状態の遷移を表現できる．

振舞い 各状態の振舞いをアクションとして定義することができる．

3. 関連研究

本章では，モデリング学習支援に関する研究および MDD と教育に関する関連研究を紹介し，本研究との関係を述べる．

3.1 モデリング学習支援に関する研究

モデリング学習支援に関する研究は，CASE ツールに関するものが多く，UML の文法をチェックしアラートを表示するツールとして UMLint [4]，サンプルパターンと比較することで文法の誤りをチェックするツールとして UMLGRADER [5] がある．また，クラス図からオブジェクト図を自動生成するシステムを用いて，静的モデリングの学習支援を行っている研究もある [8]．ただし，これらは静的モデルのみを対象としており，動的モデリングの教育は対象とされていない．

静的・動的モデリングの両面に関する教育支援として，モデルの一貫性をチェックするツール StudentUML [6] があるが，シーケンス図の作成において，クラス図にないクラス名を利用している場合にアラートを表示するなど，文法レベルのチェックにとどまっている．

本研究のテーマは，UML の文法ではなく，静的・動的両面からモデリングを行えるスキルを身に付けるための学習の支援を行うことである．

3.2 MDD と教育に関する研究

MDD と教育に関する研究では，MDD の仕組みやメタモデルなど MDD そのものの教育に関する研究，システム開発演習の中で MDD での開発を行う方法に関する研究，MDD を活用したモデリング教育を行う研究の 3 つに大別される．

MDD そのものの教育では，Gjøsater らは，メタモデルを用いた言語の設計・開発を行えるようにすることを目的とした教育コースの提案を行っている [12]．Tekinerdogan らは，3 年間大学院で MDD 教育を行った教育プログラムの内容とその結果を報告している [13]．

システム開発演習の中で MDD での開発を行う方法に関する研究として，Burden らは，学部生でも十分に MDD によるシステム開発が行えることを示している [14]．Flint らは，MDD を実現する方法の 1 つである Executable/Translatable UML を用いたシステム開発演習を行い，(1) 利用するモデルが少ない，(2) 厳密な定義ができる，(3) 関心の分離ができるなどの理由から，コードの自動生成をともなわない場合でも利用する有益性が高いことを示している [15]．Khmelevsky らは，大学生によるソフトウェアの開発や研究プロジェクトに MDD ツールを活用することの有益性を示している [16]．これらは，システム開発プロジェクトの開発環境の 1 つとして Executable/Translatable UML や MDD ツールの活用を行っており，プロジェクトの中で MDD をどのように活用するかに焦点があてられている．本研究では，より初期段階のモデリング教育を想定しており，モデリングの初学者に対して MDD ツールを活用したモデリング教育を対象とする．

MDD を活用したモデリング教育を行う研究としては，高校生や大学初級学年を対象としたプログラミングを前提としない，抽象思考の訓練があげられる [17], [18]．文献 [9] では，MDD を用いたモデリング教育を行う教育プログラムの開発を行っている．ただし，これらの研究においては，MDD ツールが学習者に与える影響は議論の対象になっておらず，MDD ツールの活用方法が明らかにされていない．本研究では，MDD ツールの活用方法に焦点をあてた分析を行う．

4. MDD ツール

商用の MDD ツールとしては，Rational Rhapsody [19]，BridgePoint [20] がある．MDD ツールを用いた教育事例は，BridgePoint を利用したものがいくつかある [9], [14], [18]．ただし，筆者らの経験によれば，BridgePoint でモデルを作成するにあたり，ステートのアクションを規定する際に必要となるアクション言語の習得に時間がかかり，本来の学習目標とするクラス図やステートマシン図の作成を集中して行えなくなることがあった．

本研究ではモデリング学習の初期段階に必要なスキルの習得を目的としており，習得させたいスキルに焦点をあてた学習ができるように通常の UML に比べて，記載できる範囲が制限された作成環境を用意するため教育用 DSML（Domain-Specific Modeling Language）を開発することとした．大きな特徴としては，アクション言語を用いず，定義されたアクションリストから選択できるように設計した．

本章では，開発した DSML およびツールの機能に関して述べる．

4.1 DSML の設計

本研究では，MDD ツールとして Web ベースのソーシャル DSL（Domain-Specific Language）プラットフォーム “clooca” [21] を用い，今回の演習課題用に DSML を設計した．記述できるモデルは下記のとおりである．

- クラス図
  - クラス（クラス名のみ）
  - 関連
- ステートマシン図
  - 初期状態
  - 状態
  - イベント送信状態（ほかのクラスのステートマシン図にイベントを送信できる）
  - イベント
  - 遷移
  - アクション

クラスのライフサイクルを表現するステートマシン図を各クラスに作成する．ただし，クラス図はシステムにただ 1 つ記述することができ，クラスは自動的に 1 つずつインスタンス化される．アクションおよびイベントはすでに登録

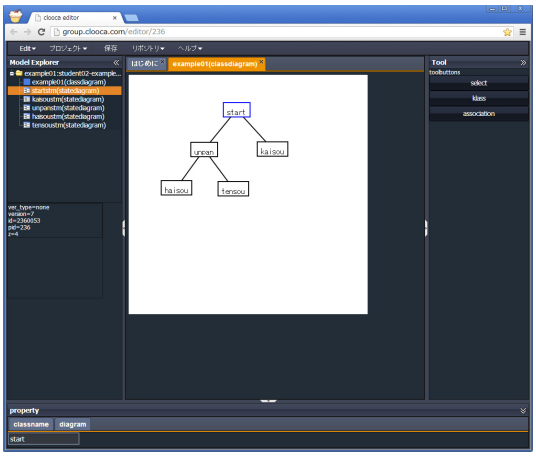

図 1 MDD ツールのエディタ画面
Fig. 1 An editor screen of the MDD tool.

済みのものを選択して利用する．なお，他のクラスにメッセージを送る際は，クラス図にメソッドを定義する代わりに，イベント送信状態を利用する．イベント送信状態の状態定義の際に，「状態名」，「送信先のクラス名」，「送信イベント名」を定義する．受信側のクラスでは，ステートマシン図中に送信イベント名と同一のイベント名を記述した遷移を作成することで，イベントを受信することができる．

MDD ツールのエディタ画面を**図 1** に示す．

4.2 モデルリポジトリ

学習者がモデルをどのように変更しながらモデルを作成していったかというモデルの変更履歴を収集するためにモデルリポジトリを作成した．モデルリポジトリは，MDD ツールの付加機能として設けた．学習者は，モデルを新規作成・追加・修正などを行った際にモデルリポジトリに格納することで，後から過去のバージョンを確認することができる．

5. 実験

5.1 実験目的

本実験の目的は，モデリング初学者が MDD ツールを活用することで，モデルを描くだけですぐに動作確認が行える環境にある場合，モデル作成のアプローチ，モデル品質，学習者のモチベーションにどのような影響を与えるかを明らかにすることである．

5.2 実験内容

MDD ツールを用いたモデリングの講義を行う際に，受講者を 2 グループに分け，片方のグループはモデルの記述後すぐにコード生成し，モデリング結果を動作として確認することができる実験群，もう片方をコード生成および動作確認ができない対照群として比較実験を行う．前者を MDD ありグループ，後者を MDD なしグループと呼ぶこととする．MDD ありグループには，MDD ツールとしてモデリング後コード生成および動作確認することができる環境を用意し，演習時間であれば学習者本人の裁量で自由に動作確認を行えることとした．これにより，MDD ありグループは，モデルを描くだけですぐに動作確認を行うことができる．一方，MDD なしグループは MDD ありグループと同様の環境を用意したが，演習の最後のみコード生成および動作確認を行うことを許可した．

5.3 評価内容

5.3.1 モデル作成のアプローチ

MDD ありでは，動作確認により機能面のテストができるため，評価可能な機能ごとのモデル作成を先に行い，その後，機能ごとのモデルを組み合わせることで，システム全体を構築していくアプローチ（ボトムアップ）をとり，MDD なしでは，仕様のみを頼りにしてモデリングを行うことになるため，仕様書に書かれている業務に関わる用語を用いたシステム全体のモデルを作成後，細かい部分のモデルの作成を行うアプローチ（トップダウン）をとると仮定される．

本実験では，モデルをどのような手順で作成し，作成したモデルにどのような影響がでるのかということを分析する．分析にあたり，下記の方法で記録を行う．

- モデルリポジトリによる，モデルのバージョン管理
- 実験中のコンピュータ操作のスクリーンキャプチャソフトによる録画記録
- ビデオカメラによる実験中の被験者の様子の録画記録

これらの結果をもとに，下記の内容を評価の対象とする．

(1) 演習終了後のクラス図の構造を分析し，MDD あり，なしがモデルに与える影響を明らかにする．
(2) クラス，状態の追加，修正，削除の割合を分析し，モデルの作り方にどのような違いが出るのかを明らかにする．

5.3.2 モデル品質

文献 [9] によると，モデリング教育に MDD ツールを用いた場合，学習者が動作確認により評価できる機能面の完成に強く意識をとらわれることで，モデル品質への意識が疎かになると指摘されている．本実験では，品質のどの部分に対して影響を与えるのかを明らかにする．なお，品質の評価には，モデルの品質の V&V（verification and validation）におけるチェック基準 [22] を基にし，本研究で対象としているモデルに対して行える品質の評価対象を下記とした．

(1) クラス図

〈Syntactic〉

- クラス，関連が表記法に従い記載されているか．

〈Semantic〉

- クラス名は，その責務が明らかな名前になっているか．
- クラスは，単一の概念を表現しているか．

- 関連名は，クラス間の関係を表したものになっているか．

〈Pragmatic〉

- クラスはバランスがとれたものになっているか．

(2) ステートマシン図

〈Syntactic〉

- 状態，遷移，アクションが表記法に従い記載されているか．

〈Semantic〉

- システムの振舞いを正しく表現しているか．
- 状態名は，その状態の意味が分かるものになっているか．

〈Pragmatic〉

- 状態の数が多すぎて複雑になっていないか．

さらに，MDD ありグループにおけるコード生成試行回数およびプログラムの転送回数を計測し，自由にコード生成，動作確認が行える状況の場合，どのくらいの頻度で動作確認を行うのか，またどのようなタイミングで動作確認を行うことが，よりよいモデルの作成につながるのかを明らかにする．

5.3.3 学習者のモチベーション

MDD ありの場合，すぐに動作確認を行える状況にあり，実際の動きを目で確認しながらモデリングが行えるため，モチベーションの維持につながると仮定される．講義終了後の被験者アンケートにより，MDD ツールの活用が学習者のモチベーションに影響するのかを明らかにする．

5.4 実験手順

被験者は，高専の 5 年生 2 名，専攻科 1 年生 10 名の計 12 名であり，すでに UML の記法およびオブジェクト指向言語である JAVA を習得している学生を対象とした．

実験に用いた講座の内容を**表 1** に示す．ただし，1 時間は 60 分とした．1 日目は，座学によるオブジェクト指向，UML の復習の後に利用するツールの使い方の説明を行った．演習課題は，基礎演習 1，2 および総合演習課題であり，MDD なしグループは基礎演習でのコード生成および動作確認はできず，2 日目の総合演習の最後に一度だけ動作確認ができることとした．

5.5 課題

本実験で主に評価の対象とするのが業務システムを開発する総合演習である．総合演習課題を開発する前に，簡単な機能を実現する基礎演習を実施する．この演習の目的は，MDD ツールの使い方および簡単な機能をモデル化する方法をマスターすることである．

総合演習課題として文献 [9] で利用されている課題を参考にし，4 時間程度の演習時間で作成できるものに改良した．総合演習課題は，架空の運輸会社の自動搬送ロボットを開発するという業務であり，開発対象ロボットは LEGO Mindstorms NXT で製作した自律型車両ロボット（**図 2**）である．自動化する業務は，「運搬」，「転送」，「回送」の 3 種類である．3 種類の業務は，配達先や荷物の有無により変更される．なお，配達先はロボット側面の側壁監視部（超音波センサ），転送先の検知はロボット前面のバンパ（タッチセンサ）で検知する．課題はロボットの前方にあるライン監視部（光センサ）でコースの黒いラインをトレースし，配達先または転送先，車庫で停止し，それぞれの地

図 2 自動搬送ロボット
Fig. 2 Auto transport robot.

表 1 講座内容
Table 1 Outline of the class.

		MDD なし	MDD あり
1 日目	1 時間目	オブジェクト指向，UML の復習，MDD とは？	
	2 時間目	MDD ツール（clooca）の使い方	
	3 時間目	基礎演習 1： 右旋回後停止のモデル作成	基礎演習 1： 右旋回後停止のモデル作成，コード生成，動作確認
	4 時間目	基礎演習 2： ライントレースのモデル作成	基礎演習 2： ライントレースのモデル作成，コード生成，動作確認
2 日目	5 時間目	基礎演習のレビュー，総合演習課題の説明	
	6 時間目 7 時間目 8 時間目	総合演習課題： 自動搬送システムのモデル作成 モデルの完成後，コード生成，動作確認	総合演習課題： 自動搬送システムのモデル作成，コード生成，動作確認

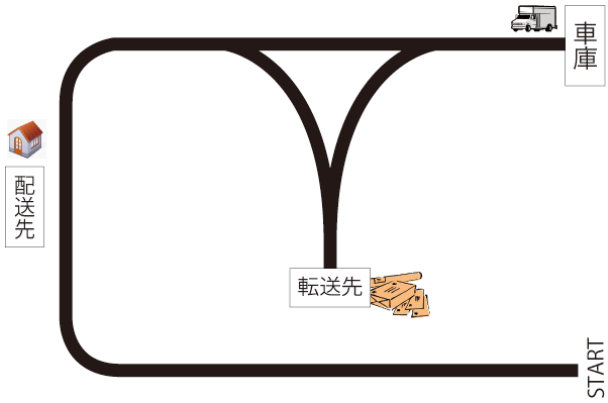

図 3 総合演習課題のコース
Fig. 3 Course layout for integrated exercise.

点において適切な動作を行い，所定の位置に荷物を届けることである．演習のコースを**図 3** に示す．

総合演習課題における想定するクラスは，全体を制御するクラスおよび，運搬，転送，回送などのシステム業務の実現方法を定義する抽象度の高い「業務」レイヤに属するクラスと，ライントレースや直進などのロボット動作（機能）の実現方法を定義する抽象度の低い「機能」レイヤに属するクラスに分けられる．課題と学習目標の関係を下記に示す．

責務分割 クラス図の作成において，クラスのレイヤを意識したクラス分けを行う．

クラス名 各レイヤのクラスでは，そのレイヤの抽象度に合わせた名前をつける．

状態名 クラスの抽象度に合わせた状態名をつける．

遷移 ロボットに課題を達成する動きをさせるためのイベントと状態の関係を表現する．

振舞い 各状態の振舞いをアクションとして定義し，課題を達成する．

課題の 3 種類の業務内容とそれぞれの業務達成に必要な機能を難易度の低い順に並べると下記のようになる．

(1) 回送
  (a) ライントレースできる．
  (b) 側壁を検知して止まる．
(2) 運搬
  (a) 荷下ろしを検知して回送する．
  (b) 車庫に入って止まる．
(3) 転送
  (a) エッジチェンジし，搬送コースを変える．
  (b) 転送先を検知し，反転する．
  (c) 反転後ライントレースを再開する．

6. 実験結果

6.1 モデル作成のアプローチに関する結果

6.1.1 全体の傾向

MDD なし，MDD ありグループの代表的なクラス図を**図 4**，**図 5** に示す．

MDD ありグループのクラス構造は全員バラバラであったのに対し，MDD なしグループは，図 4 に示す，2 つのクラス図のどちらかだけであった．

MDD なし，MDD ありの各グループの被験者 6 名をそれぞれ A〜F としたとき，作成したクラス図におけるクラス名を「全体を制御するクラス」,「業務に関するクラス」,「機能に関するクラス」の 3 つに分類した結果および総クラス数を**表 2** に示す．「全体を制御するクラス」以外に関して，MDD なしと MDD ありグループの傾向を比較すると，MDD なしグループでは，業務に関するクラスのみであるのに対し，MDD ありグループでは走り方などの機能に関するクラスの数の方が多く，業務に関するクラスを記述していない被験者も 4 名いた．

総合演習課題の内容を作成するにあたり，基礎演習で作成したモデルを再利用した人数は，MDD ありグループが 4 名だった．一方，MDD なしグループの被験者は総合演習課題の作成開始時に，全員，モデルを削除後，クラス図の新規作成を行っていた．

総合演習課題を作成する際に行った操作のうち，クラスおよび状態の追加，修正，削除を行った割合の被験者平均をモデル作成の前半，中間，後半に分類した結果を**表 3** に示す．MDD なしグループでは，後半になるにつれて，クラスの追加，削除の割合が減少しており，状態の追加は中間以降に多くなっている．これより，前半にクラスの追加，削除を行い，クラスの静的構造を固めた後，後半は状態の追加を行うことで，動的振舞いを規定していると考えられる．一方，MDD ありグループは，前半，中間でのクラスの追加の割合がほぼ同じであり，後半にもクラスの追加，削除が行われている．したがって，MDD ありグループは，後半までクラス構造を変更させていたといえる．

6.1.2 個別のモデル作成過程

6.1.1 項で述べた全体の傾向とは，少し異なった開発を行った 2 名の被験者の作成過程を分析することで，よりよいモデルを作成するためのモデルの動作確認の回数やタイミングの指針とする．

(1) すべての課題を達成したケース（MDD あり，被験者 A）
  被験者 A は，下記の 3 つのクラスを作成している．
  controller 全体の業務および走行を制御するクラス
  leftlinetracer 左エッジをライントレース走行するクラス
  rightlinetracer 右エッジをライントレース走行するクラス
  “leftlinetracer” と “rightlinetracer” は，基礎演習課題のモデルを再利用しており，最初にその部分のステートマシン図を追加・修正した後，“controller” クラスのステートマシン図の作成を行っている．本演習の業務は，大きく 3 つであり，難易度の低い順に，「回

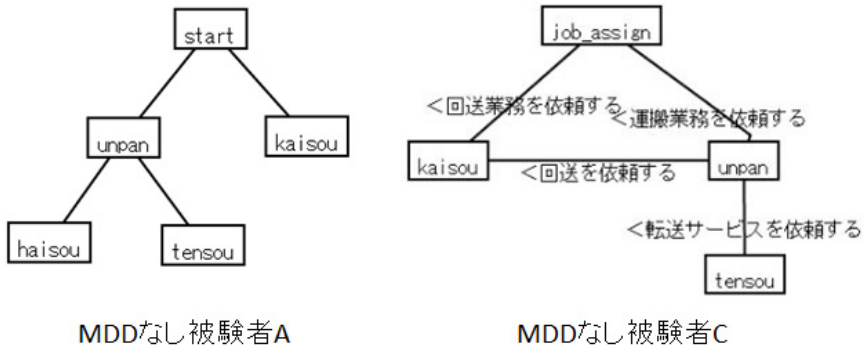

図 4 MDD なしグループのクラス図
Fig. 4 Class diagrams of the group without MDD.

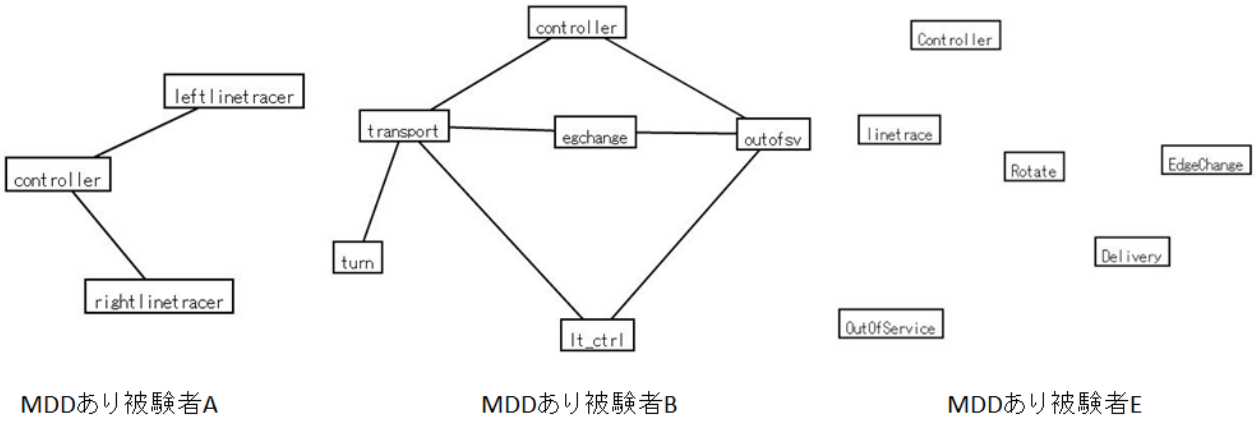

図 5 MDD ありグループのクラス図
Fig. 5 Class diagrams of the group with MDD.

表 2 総合演習課題のクラス名の分類
Table 2 Classification of a class name of the integrated exercise.

	クラスの分類項目	A	B	C	D	E	F	平均
MDD なし	全体を制御するクラスの数	1	1	1	1	1	1	1.0
	業務に関するクラスの数	4	4	3	3	4	3	3.5
	機能に関するクラスの数	0	0	0	0	0	0	0.0
	総クラス数	5	5	4	4	5	4	4.5
MDD あり	全体を制御するクラスの数	1	1	1	1	1	1	1.0
	業務に関するクラスの数	0	2	0	0	2	0	0.7
	機能に関するクラスの数	2	3	3	4	3	0	2.5
	総クラス数	3	6	4	5	6	1	4.2

送」,「運搬」,「転送」である．被験者 A は，同一の “controller” クラス内に振舞いを記述しているが，作成過程においては，難易度の低い順番に作成しており，それぞれの業務を追加後にコード生成および動作確認を行うことで，正しく記述できていることを確認している．さらに，各業務の完成のタイミングで，リポジトリにモデルを登録していた．被験者 A は，1. 機能の振舞いモデルの記述，2. 静的構造モデルの記述，3. 業務の振舞いモデルの記述という手順で作成しており，ボトムアップとトップダウンの両方のアプローチをとっていることが分かる．また，ある特定の機能や業務が完成したタイミングでテストを実施しており，全体としてスムーズなモデル作成が行えている．

(2) ほとんどの課題を達成できなかったケース（MDD あり，被験者 B）

被験者 B は，下記の 6 つのクラスを作成しており，全体を制御するクラス，業務に関するクラス，機能に関するクラスのバランスがとれており，責務分割を検討

表 3 総合演習におけるモデル作成履歴
Table 3 Model creation history of integrated exercise.

	前半		中間		後半	
	MDD なし	MDD あり	MDD なし	MDD あり	MDD なし	MDD あり
クラスの追加	17.3%	8.9%	8.1%	9.0%	0.0%	1.6%
クラスの修正	9.6%	7.8%	6.5%	3.0%	9.8%	3.9%
クラスの削除	3.9%	4.1%	1.8%	0.5%	0.0%	3.3%
状態の追加	44.0%	34.3%	69.9%	71.4%	68.4%	49.0%
状態の修正	10.0%	32.9%	6.5%	13.5%	20.9%	35.2%
状態の削除	15.2%	12.1%	7.3%	2.7%	1.0%	7.0%

表 4 品質カテゴリに関するエラーを持つクラス数
Table 4 The total number of classes with the error about the quality categories.

		MDD なし	MDD あり
Syntactic	関連が引かれていない	0	2
	関連名が記入されていない	2	4
Semantic	クラス名がそのクラスの責務が分かる名前になっていない	0	2
	関連名が不適切	0	2
Pragmatic	不必要なクラス（振舞いが記述されていない）がある	0	1
	1 つのクラス内に複数の責務が盛り込まれている	0	1
	複数クラスに同一の機能が盛り込まれている	6	0

表 5 品質カテゴリに関するエラーを持つ状態数
Table 5 The total number of states with the error about the quality categories.

品質カテゴリ	エラー項目	MDD なし	MDD あり
Syntactic	状態名が記入されていない	0	8
Semantic	同じ名称の状態を複数記述している	5	11
	状態名が不適切	0	7

したクラス図が作成できている．

controller 全体の業務を制御するクラス

transport 運搬・転送業務を行うクラス

outofsv 回送業務を行うクラス

ls_ctrl ライントレース走行を制御するクラス

egchange エッジチェンジを行うクラス

turn 転送先での方向転換を行うクラス

モデルの作成手順は，1. 静的構造モデルの記述，2. 全体を制御する振舞いモデルの記述，3. 業務の振舞いモデルの記述，4. 機能の振舞いモデルの記述となっており，トップダウンのアプローチをとっている．さらに，業務モデルの作成においては，難易度の高い「転送業務」から作成している．これにより，全体のシステムが完成しなければ，動作確認が行えない．したがって，被験者 B のコード生成試行回数は 8 回（**表 7**）と他の被験者に比べて，少なくなっている．また，モデル作成過程の半ばでクラス図の大幅な変更を行っており，振舞いを記述する過程で，同時によりよい構造を検討していた様子がうかがえる．被験者 B は，すべてのシステムを作成後に動作確認を行おうとしたため，どの部分に不具合があるのかを特定することが困難になり，業務の達成率が低くなった．演習後のアンケートでも，「ちょっとずつテストすべきだった」と記述している．

6.2 モデル品質に関する結果

6.2.1 クラス図に関する結果

評価項目として定義していた，3 種類の品質カテゴリに関するエラーを持つクラス数を**表 4** に示す．ただし，各グループ 6 名の受講生がクラス図を 1 つずつ作成しているため，最大数は 6 である．

エラー項目 7 項目のうち 6 項目においては，MDD ありグループのエラー数の方が多かったが，「複数クラスに同一の機能が盛り込まれている」という項目に関しては，MDD なしグループ全員があてはまっていた．

6.2.2 ステートマシン図に関する結果

MDD なし，MDD ありグループの代表的なステートマシン図を**図 6**，**図 7** に，品質カテゴリに関するエラーを持つ状態数を**表 5** に示す．エラーを持つ状態数を計測するにあたり，「状態名が記入されていない」，「状態名が不適切」に関しては，その状態の総数，「同じ名称の状態を複数記述している」に関しては同一ステートマシン図内に同一状態名があった場合を 1 カウントとして数えた．

MDD ありグループのモデルでは，図 7 のように状態名がないモデル（被験者 E）や状態名が a，b など状態の意味

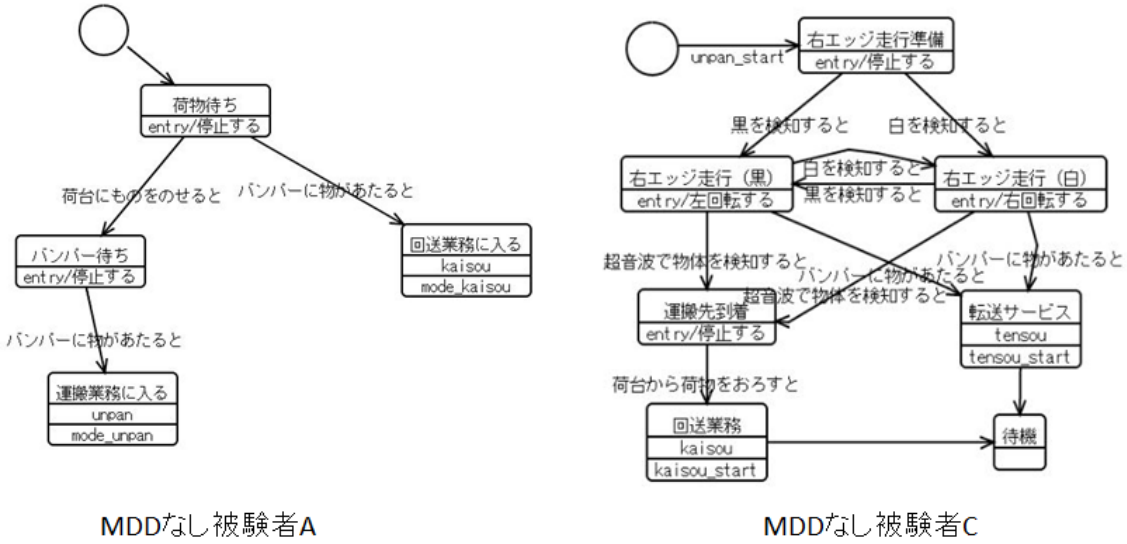

図 6 MDD なしグループのステートマシン図
Fig. 6 State machine diagrams of the group without MDD.

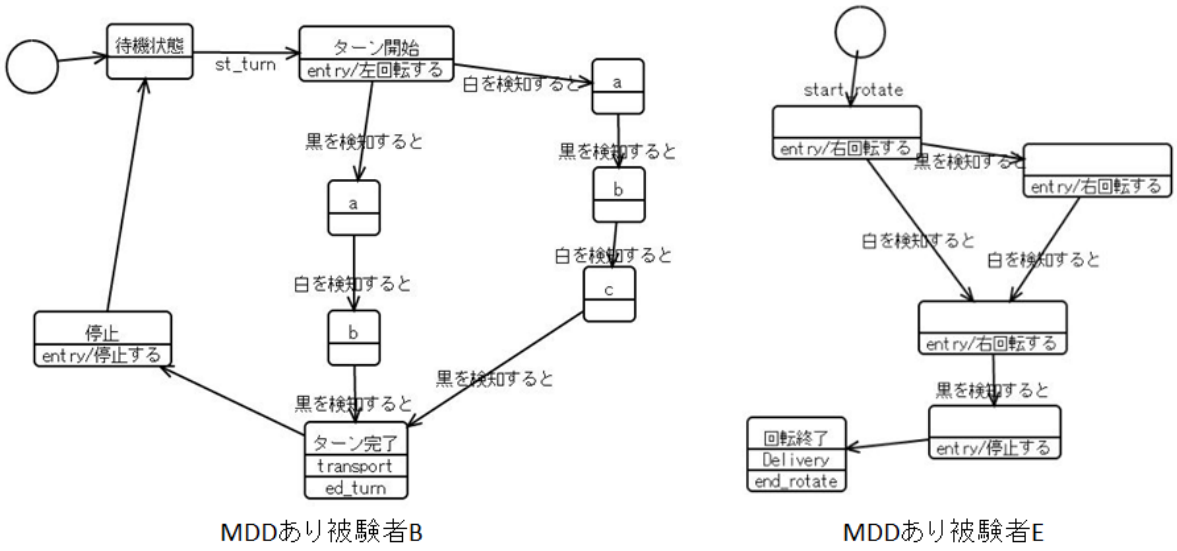

図 7 MDD ありグループのステートマシン図
Fig. 7 State machine diagrams of the group with MDD.

が読み取れないもの（被験者 B）などが見られた．

5.3.2 項に示した Semantic 品質の評価対象の 1 つである，「システムの振舞いを正しく表現しているか」という項目の評価として，開発したシステムの達成率の評価を行った．課題内容に示した，業務の達成に必要な機能に「すべてのパターンを完走する」という項目を加えて 8 つを評価項目として，達成した項目を自己申告してもらった．その結果を**表 6** に示す．MDD ありのグループは，動作確認ができるため機能に関する達成率は高くなっている．

Pragmatic 品質の評価として，状態の数の比較を行う．各グループの被験者が作成したモデルの状態の数は，MDD なしは 7.7，分散 1.1 であるのに対し，MDD ありの平均状態数は，11.3，分散が 25.1 であった．MDD なし，MDD ありの状態数において，t 検定を用いて両側棄却 5%の有意差検定を行ったが，有意差は見られなかった．ただし，

表 6 業務内容の達成率
Table 6 Achievement rate of the integrated exercise.

	MDD なし	MDD あり
ライントレースできる	83%	100%
側壁を検知して止まる	67%	67%
荷下ろしを検知して回送する	50%	83%
車庫に入って止まる	67%	67%
エッジチェンジし，搬送コースを変える	50%	83%
転送先を検知し，反転する	33%	67%
反転後ライントレースを再開する	0%	67%
すべてのパターンで完走する	0%	17%

表 7 MDD ありグループの被験者のモデル品質と動作確認の回数
Table 7 Model quality and the number of times of checking a software of the group with MDD.

被験者	品質スコア	課題の達成率	コード生成試行回数	転送回数
A	7	100%	20	14
B	6	13%	8	6
C	8	88%	8	3
D	6	75%	26	25
E	5	88%	20	11
F	4	50%	19	14

表 8 取組み姿勢のアンケート結果
Table 8 Motivation questionnaire results.

	MDD なし	MDD あり
1（やる気がなかった）	0%	0%
2	0%	0%
3	0%	17%
4	50%	50%
5（とても熱心）	50%	33%

表 9 モデリングしやすさのアンケート結果
Table 9 Questionnaire results about ease of modeling.

	MDD なし	MDD あり
MDD を利用する場合	100%	67%
MDD を利用しない場合	0%	0%
どちらともいえない | 0% | 33% |

MDD ありグループの状態数のバラつきが大きく，被験者の半数は，複雑さを増す状態数 10 [22] を超えた，12，15，19 であった．MDD なしグループは，最小 7，最大 9 であり，複雑なステートマシン図はなかった．

6.3 モデル品質と動作確認の関係

表 7 に，MDD ありグループの被験者 A〜F のモデル品質と動作確認の回数を示す．品質スコアは，減点方式としており，減点項目は表 4，表 5 に示す合計 10 項目とし，エラー項目数を 10 点から減算している．ただし，表 4 の「関連が引かれていない」，「関連名が記入されていない」，表 5 の「状態名が記入されていない」の 3 項目は，それぞれの名前が不適切だと想定される項目もエラーとしてカウントしている．「コード生成試行回数」は，MDD ツール上でコード生成ボタンを押した回数である．ただし，モデル上に未記入のアクションやイベントがある場合など，モデルに不具合がある場合には，コード生成の際にエラーとなる．また，コード生成を行えた後もコンパイルに失敗することもある．「転送回数」は，実際にロボットにプログラムを転送でき，動作確認を行えた回数である．

品質スコア，課題の達成率ともにコード生成試行回数や転送回数との相関関係がなく，この実験からは，最適な動作確認回数を特定することはできなかった．

6.4 学習者のモチベーションに関する結果

モチベーションの評価として，取組姿勢のアンケートを実施した．その結果を**表 8** に示す．

アンケートは 1〜5 の 5 段階評価とし，1 を「やる気がなかった」，5 を「とても熱心」とした．MDD なし，ありの受講者ともに多くの受講生が熱心に取り組んでいた．受講生から「コードが自動生成されることが大きな利点だったと思う．システム設計がイメージしやすくてモデリングできた」，「プログラムをたくさん書かなくてもいいので，モチベーションを保つことができた」，「かなり直感的にプログラムが組めるので，今回のように実際にロボットを動作させるようなプログラミングでは楽しみながら行えると感じた」などのコメントがあり，MDD なし，あり，どちらのグループに所属したかにかかわらず，MDD ツールを利用することに肯定的な意見が多かった．

さらに，MDD の利用なし，ありの場合どちらがモデリングをしやすいか尋ねた結果を**表 9** に示す．

MDD なし，ありグループともに，MDD ありの方がモデリングしやすいと感じると仮定していたが，アンケート結果によると MDD なしグループは全員 MDD を利用する方がモデリングしやすいと答えたのに対し，MDD ありグループはどちらともいえないと回答した受講生がいたため，67%にとどまった．

理由としては，受講生の裁量で動作確認をできた MDD ありグループは，動作確認を自由にできないという状況を体験できなかったため，比較ができずどちらともいえないと答えたと考える．ただし，演習の最後のみ動作確認が可能だった MDD なしグループは，少しではあるがモデルの動作確認を行える状況とできない状況の両方を体験でき，グループの全員が MDD ありの方がモデリングしやすいと答えた．これにより，MDD をうまく利用することで，モデリングしやすく，モチベーションの維持につながるのではないかと考える．

7. 考察

本章では，MDD ツールがモデル作成のアプローチ，モデル品質，学習者のモチベーションに与える影響を考察し，

実験結果をふまえたMDDツールの効果的な活用方法に関して述べる．

7.1 MDDツールがモデルや学習者に与える影響

6.1.1項より，MDDありグループは，完成している機能に関するモデルを修正した後，モデルの静的構造，動的振舞いともに修正を加えながら作成していくのに対し，MDDなしグループは，新規に静的モデルを作成後，それぞれに対応した動的振舞いを作成する傾向にあった．MDDなし，ありの被験者がそれぞれ6名と少人数であったが，MDDなしグループでは全員，MDDありグループでは4名が同様の傾向を示していた．これより，MDDを利用する場合には，機能に関するモデルを作成後に，機能に関するモデルを組み合わせて全体のシステムを作成するボトムアップ的な作成方法になり，MDDを利用しない場合には，仕様を基に，システム全体の静的構造を作成した後，各クラスの動的振舞いを作成するトップダウン的な作成方法となる傾向が確認できた．

クラス図のモデル品質においては，Syntactic，Semantic，Pragmaticすべての品質カテゴリにおいて，MDDツールの利用に制限を加えたMDDなしグループの品質が高くなった．ただし，MDDなしグループの全員が上手く機能分割できておらず，機能に関するクラスを記載している受講者は誰もいなかった．MDDなしグループ全員に同様の結果が見られたこと，MDDなしグループの業務の達成率も低かったことを考慮すると，MDDツールを用いず動作確認ができない状態では，抽象度の高い業務中心のモデルになり，走り方など抽象度の低い細部の機能のモデル化が難しくなると考えられる．

一方，MDDを利用する場合には，動作として確認することを優先しがちになるため，関連が引かれていないクラス（図5被験者E）や，クラスが1つしかなく，適切な責務分割が行われていないモデルなどがあった．ステートマシン図においても，状態名が記入されていないモデルや同じ名称の状態を複数記述しているモデルの割合が高く，単純にMDDツールを導入するだけでは，モデル品質の低下につながる恐れがあることを確認した．

MDDツールの利用は，モチベーションの維持にも効果があると仮定していたが，今回の少人数の実験における結果だけでは，明らかにすることができなかった．しかし，MDDなし，ありの両方を体験したMDDなしグループ全員がMDDありの方がモデリングしやすいと答えており，今後，MDDツールの使い方やMDDツール自身の改善により，モデリングしやすく，モチベーションを保てるのではないかと考える．

7.2 MDDツールの効果的な活用方法

現在，実施されているモデリング教育コースの多くが，トップダウン的な開発方法を採用している．一方，産業現場でMDDを成功させている例では，システム全体を一度にモデル化している例は少なく，ボトムアップ的に一部分をMDDとして開発し，MDD適応成功例を積み重ねることで，結果としてシステム全体のMDDへの移行を実現している．この事例を基に，文献[3]では，モデリング教育においてもボトムアップに進めるべきだとしている．本研究では，MDDツールを活用し，コードの自動生成や動作確認の回数を制限することで，トップダウン，ボトムアップ両方のアプローチによる開発を促すことが可能となることが確認できた．

本研究の対象とした，クラス図とステートマシン図の組合せは，リアルタイムシステムのモデリングに利用されることが多く[22]，開発は組込みシステムを対象としたものが多い[23]．クラス図とステートマシン図でモデル化できるシステムをモデリング演習の対象とした場合のMDDツールの活用方法を検討する．モデリングの教育効果をあげるためには，単純にMDDツールを用いた開発をさせるのではなく，コードの自動生成や動作確認の回数を制限した演習を実施する必要がある．

6.1.2項の分析結果をもとに，動作確認のタイミングを考える．たとえば，基礎的な演習で用いるライントレースなどの小さな機能の完成のタイミングでコードの自動生成および動作確認をさせて，機能を実現できているか確認させるというボトムアップ的なアプローチをとり，同時にクラス名や状態名のつけ方，モデルの表記法などを合わせてチェックさせるようにする．その後に行う，業務システム（本論文では，自動搬送システムを用いた）開発の場合には，トップダウン的なアプローチにより，十分なクラス図の検討を行った後，各機能を追加するごとに動作確認を行うように指導することが望ましいと考える．

ただし，今回の実験結果からは，動作確認回数とモデル品質や業務の達成率に相関関係がなく，最適な動作確認の回数を明らかにすることはできなかった．今後，より効果的なモデリング教育を行うためのモデル規模に対する動作確認の回数およびタイミングに関して検討する必要がある．

8. おわりに

本研究では，UMLモデリングの効果的な活用方法を検討するために，コードの自動生成および動作確認を学習者の裁量で実施できるMDDありグループ（実験群）とコードの自動生成および動作確認を制限されたMDDなしグループ（対照群）の2グループに分けて，自動搬送システムを開発させる実験を行った．

それぞれの被験者の作成したモデル，モデルの作成のアプローチを比較した結果，MDDなしの場合は，トップダウン的に開発する傾向とともに，モデリング記法に従い正確に表記したり，クラスや状態に適切な名前をつけるよう

に心がける傾向にあった．MDD ありの場合は，機能ごとの動作確認が行えるため，ボトムアップ的な開発傾向にあり，課題の業務達成率も高かった．

これらの結果より，MDD ツールを活用し，コードの自動生成や動作確認の回数を制限することで，トップダウンまたはボトムアップといった開発へのアプローチ方法の変化を促すことができると考える．

ただし，今回の結果からは，最適な動作確認の回数やタイミングなどを明らかにすることはできなかったため，今後の課題とする．

参考文献

[1] 独立行政法人情報処理推進機構：組込みソフトウェア開発における品質向上の勧め［設計モデリング編］，アイティメディア (2006).

[2] Chaudron, M.R.V., Heijstek, W. and Nugroho, A.: How effective is UML modeling?, *Software and Systems Modeling*, Vol.11, No.4, pp.571–580 (2012).

[3] Whittle, J. and Hutchinson, J.: Mismatches between Industry Practice and Teaching of Model-Driven Software Development, *Models in Software Engineering*, Lecture Notes in Computer Science, Vol.7167, pp.40–47 (2012).

[4] Hasker, R.W. and Rowe, M.: UMLint: Identifying Defects in UML Diagrams, *Proc. 118th Annual Conference of the American Society for Engineering Education* (2011).

[5] Hasker, R.W.: UMLGrader: An automated class diagram grader, *J. Comput. Sci. Coll.*, Vol.27, No.1, pp.47–54 (2011).

[6] Dranidis, D.: Evaluation of Student UML: An Educational Tool for Consistent Modelling with UML, *Proc. Informatics Education Europe II Conference*, pp.248–256, South-East European Research Center (2007).

[7] Turner, S.A., Pérez-Quiñones, M.A. and Edwards, S.H.: minimUML: A Minimalist Approach to UML Diagramming for Early Computer Science Education, *J. Educ. Resour. Comput.*, Vol.5, No.4, pp.1–28 (2005).

[8] 早川 勝，野沢光太郎，松澤芳昭，酒井三四郎：オブジェクト指向モデリング教育のためのオブジェクト図自動生成システムの設計と評価，情報処理学会論文誌，Vol.54, No.1, pp.66–79 (2013).

[9] Akayama, S., Kuboaki, S., Hisazumi, K., Futagami, T. and Kitasuka, T.: Development of a Modeling Education Program for Novices using Model-Driven Development, *Proc. 2012 Workshop on Embedded and Cyber-Physical Systems Education*, ACM (2012).

[10] UMTP/Japan：UMTP，特定非営利活動法人 UML モデリング推進協議（online），入手先 ⟨http://www.umtp-japan.org/⟩（参照 2013-09-26）.

[11] Lindland, O.I., Sindre, G. and Sølvberg, A.: Understanding Quality in Conceptual Modeling, *IEEE Softw.*, Vol.11, No.2, pp.42–49 (1994).

[12] Gjøsæter, T. and Prinz, A.: Teaching Model Driven Language Handling, *Electronic Communications of the EASST*, Vol.34, pp.1–10 (2010).

[13] Tekinerdogan, B.: Experiences in teaching a graduate course on model-driven software development, *Computer Science Education*, Vol.21, No.4, pp.363–387 (2011).

[14] Burden, H., Heldal, R. and Siljamäki, T.: Executable and Translatable UML – How Difficult Can it Be?, *Proc. 18th Asia-Pacific Software Engineering Conference*, pp.114–121, IEEE Computer Society (2011).

[15] Flint, S., Gardner, H. and Boughton, C.: Executable/Translatable UML in Computing Education, *Proc. 6th Australasian Computing Education Conference* (2004).

[16] Khmelevsky, Y., Hains, G. and Li, C.: Automatic code generation within student's software engineering projects, *Proc. 17th Western Canadian Conference on Computing Education*, pp.29–33, ACM (2012).

[17] 香山瑞恵, 二上貴夫, Starrett, C., 今野篤志：Model Driven Development に基づく抽象化概念教育の提案，日本情報科学教育学会第 3 回全国大会講演論文集 (2010).

[18] Starrett, C.: Teaching UML Modeling Before Programming at the High School Level, *Proc. 17th IEEE International Conference on Advanced Learning Technologies*, pp.713–714, IEEE Computer Society (2007).

[19] IBM: Rational Rhapsody (online), available from ⟨http://www-06.ibm.com/software/jp/rational/products/rhapsody/productline/⟩ (accessed 2013-09-26).

[20] Mentor Graphics: BridgePoint (online), available from ⟨http://www.mentorg.co.jp/products/sm/model_development/bridgepoint.html⟩ (accessed 2013-09-26).

[21] Hiya, S.: clooca, Technical Rockstars (online), available from ⟨http://www.clooca.com⟩ (accessed 2013-09-26).

[22] Unhelkar, B.: *Verification and Validation for Quality of UML 2.0 Models*, Wiley-Interscience (2005).

[23] 新井玲子：UML によるオブジェクト指向モデリング セルフレビューノート，ディー・アート (2005).

赤山 聖子 （学生会員）

1982 年生．2006 年熊本大学工学部数理情報システム工学科卒業．2007 年学校法人赤山学園副理事長就任．2008 年熊本大学大学院自然科学研究科情報電気電子工学専攻博士前期課程修了．2008 年から学校法人赤山学園九州技術教育専門学校校長補佐．2012 年九州大学大学院システム情報科学府博士後期課程入学，現在に至る．オブジェクト指向モデリングやモデル駆動開発の教育に関する研究に従事．IEEE 会員．

久住 憲嗣 （正会員）

2004 年九州大学大学院博士課程修了．2008 年から九州大学システム LSI 研究センター准教授．組込みソフトウェア開発方法論の教育，研究に携わる．特にモデル駆動開発や低消費電力化技術に関する研究に従事．ACM 会員．

部谷 修平

1988 年生．2011 年九州大学工学部電気情報工学科卒業．同年同大学大学院システム情報科学府に入学．2012 年株式会社 Technical Rockstars を設立．ドメイン特化型言語と周辺ツールに興味を持つ．

福田 晃 （フェロー）

1977 年九州大学工学部情報工学科卒業．1979 年同大学大学院工学研究科修士課程情報工学専攻修了．同年日本電信電話公社（現 NTT）武蔵野電気通信研究所入所．1983 年九州大学助手．1989 年同大学助教授．1994 年奈良先端科学技術大学院大学教授．2001 年九州大学大学院システム情報科学研究院教授，2008 年九州大学システム LSI 研究センター長（兼任），現在に至る．工学博士．組込みソフトウェア，ユビキタスコンピューティングに関する研究に従事．情報処理学会研究賞 (1990)，Best Author 賞 (1993) 等を受賞．電子情報通信学会，ACM，IEEE Computer Society，日本 OR 学会各会員，「NPO 法人九州組込みソフトウェアコンソーシアム（QUEST）」理事長．