Vol.2012-CE-117 No.8
2012/12/9

UMLモデリング教育における モデル駆動開発ツールの利用方法の検討

赤山 聖子[1,a)]　久住 憲嗣[2]　福田 晃[3]

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．UML 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 UML 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．

An Empirical Study of the Usage of Model-Driven Development Tool for a UML Modeling Education

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 don't use it . We clarify the advantages of each case with and without the use of the MDD tool. From these results, we propose how to the effective use of MDD tools in UML modeling education.

1. はじめに

近年，組込み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 を用いたモデリング教育を行う試みは数多くなされている．一方，産業界では，モデルを使う開発手法として，モデル駆動型開発（Model Driven Development:MDD）の実用化が進んでいる．UML モデリング教育に関する研究としては，教育向けの CASE ツールに関するものが多く [3][4]，モデル図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る．ただし，これらのツールは，UML の文法の習得には効果的であるが，モデリングスキルの習得は困難である．また，クラス図からインスタンス図を自動生成することで，クラス図の理解を深める試み [5] 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 [6] がなされている．しかし，これらの研究はクラス図やインスタンス図といった静的モデルに関するものが多く，動的な振舞い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教育に関するものとして，MDD を実現する手段の一つである，Executable UML を利用したものがある [7][8][9]．自動生成に関する研究としては，要求分析で得られたクラス図とアクティビティ図から Web ユーザインターフェースのプロトタイプやシナリ

[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
[2] 九州大学システム LSI 研究センター
System LSI Research Center, Kyushu University
[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
[a)] seiko@ktec.ac.jp

オなどを自動生成させるツールを要求分析の教育へ利用する取組みが行われている [10].

UML モデリングに関して筆者らは MDD を活用して UML モデリングスキルの向上を図る教育プログラムの開発を行っている [11]. MDD では，モデル上での検証とコードの自動生成が可能で，作成したモデル図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，開発対象のアプリケーションのモデリング作業に集中しできる，モデルの欠陥の検出から修正のフィードバックサイクルにかかる時間を短縮することで，モデルの洗練していく過程に時間をかけることができる，などの利点が考えられる．

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，受講者が 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である．本研究では，MDD ツールを用いた効果的な UML モデリング教育方法の検討を行う．なお，本論文では UML で記述されたモデル図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．

以下, 2 章では研究のアプローチとして研究の目的, MDD ツールの内容に関して述べる．3 章では実験内容，4 章では実験結果を示し，5 章では実験結果を踏まえた考察を述べる．

2. 研究のアプローチ

2.1 研究の目的

本研究の目的は，UML モデリングの教育を実施する際に MDD ツールを利用し，より効果的な教育方法を検討することである．MDD とは，設計段階で作成したモデルをツール等を使って動作シミュレーションを行うことで検証し，実際に動作するソフトウェアの実装コードを自動生成することを狙った開発手法である．MDD においては，モデルがソフトウェア・ライフサイクルのすべての局面において開発プロセスの中心となる [12]．MDD 手法をソフトウェア開発に用いることで，次のようなメリットがある．(1) 設計と実装を分離できるため，モデリングに注力して開発できる．(2) モデル上でのシミュレーションができるため，開発早期での検証が可能となる．(3) 実装工程を自動化することで，設計，テスト，改善のサイクルを短時間で繰り返すことができる．

2.2 対象とするモデル図

UML には 13 種類の図があり，開発工程や開発対象により開発者が適切な図を利用してシステムのモデル化を行う．本プログラムでは，モデリング手法を学ぶことを目的としているため，全ての図を習得する必要はない．通常，静的構造，動的振舞いの 2 種類の図を描くことで，システムは表現できる．本研究では，静的構造としてクラス図を動的振舞いとしてステートマシン図を教育対象とする．

2.3 MDD ツール

本講座では，MDD ツールとして Web ベースのソーシャル DSL プラットフォーム “clooca”[13] を用い，今回の演習課題用に設計した DSL を利用する．記述できるモデル図は下記の通りである．

- 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一つ記述することができ，クラスは自動的に一つずつインスタンス化される．アクション及びイベントはすでに登録済みのものを選択して利用する．なお，本研究で用いた DSL では，クラス図にメソッドを定義することができないため，他のクラスにメッセージを送る際は，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．送信される側のクラスでは，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.

2.4 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の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 “clooca” の付加機能として設けてある．学習者は，モデルを新規作成，追加・修正等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．

3. 実験

3.1 実験方法

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

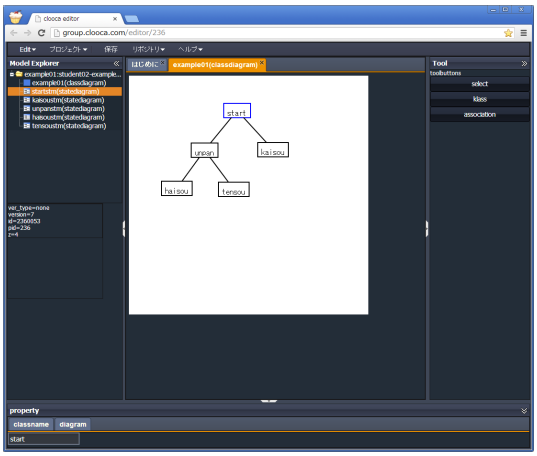

図 1 MDD ツールのエディタ画面
Fig. 1 A editor screen of the MDD tool.

言語である JAVA を習得している学生を対象とした．被験者を 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グループ，もう片方をコード生成及び動作確認ができないグループ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及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．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及び動作確認を行うことを許可した．

3.2 実験手順

実験に用いた講座の内容を**表 1** に示す．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 及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及び動作確認はできず，2 日目の最後に一度だけ動作確認ができることとした．

3.3 総合演習課題

本実験で主に評価の対象とするのが総合演習課題である．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，運搬業務，転送サービス，回送業務の 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．

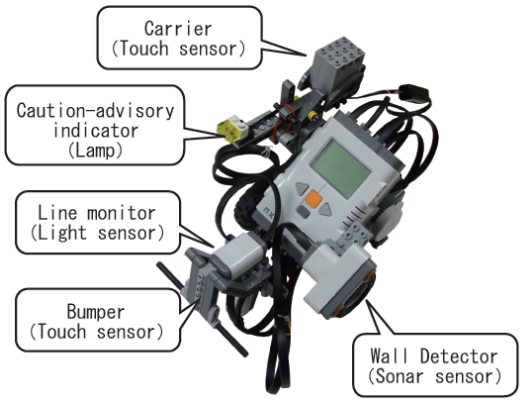

図 2 自動搬送ロボット
Fig. 2 Auto transport robot.

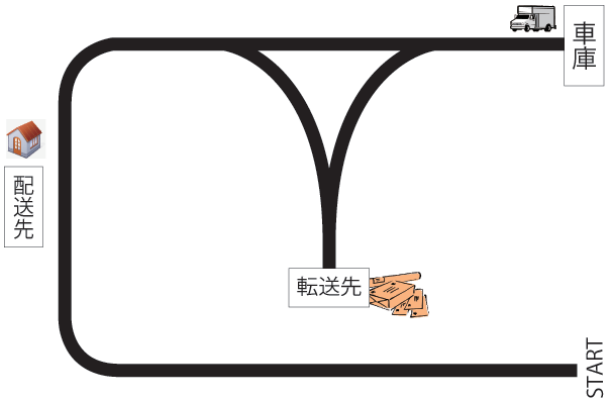

図 3 演習課題のコース
Fig. 3 Course layout for integrated exercise.

4. 実験結果

モデルの品質は，Syntactic, Semantic, Pragmatic の 3 種類の品質カテゴリに分類される [14]．各項目に関して結果整理を行う．各品質カテゴリに関する主な評価内容は次の通りである．

- Syntactic quality: 表記法の正確性
- Semantic quality: システム表現の正確性
- Pragmatic quality: 第三者が解釈する場合の正確性

なお，以下の評価は講座の最後に実施した総合演習課題に関するモデル図とする．

4.1 クラス図に関する結果

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 なし，MDD ありの各グループの被験者 6 名をそれぞれ A〜F とした時，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」，「業務内容に関するクラ

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

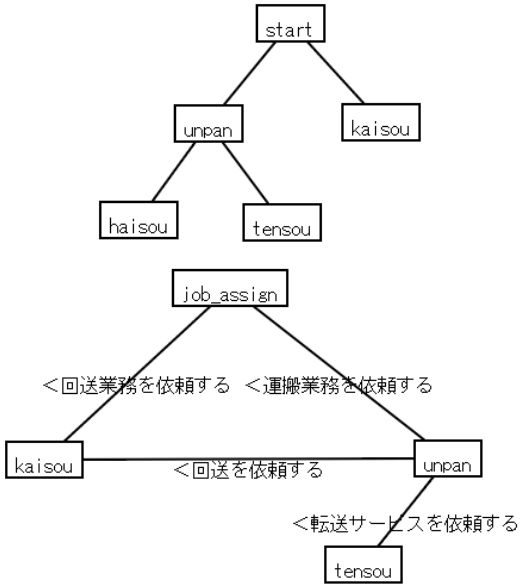

図 4 MDD なしグループのクラス図
Fig. 4 Class diagrams of the group with no MDD.

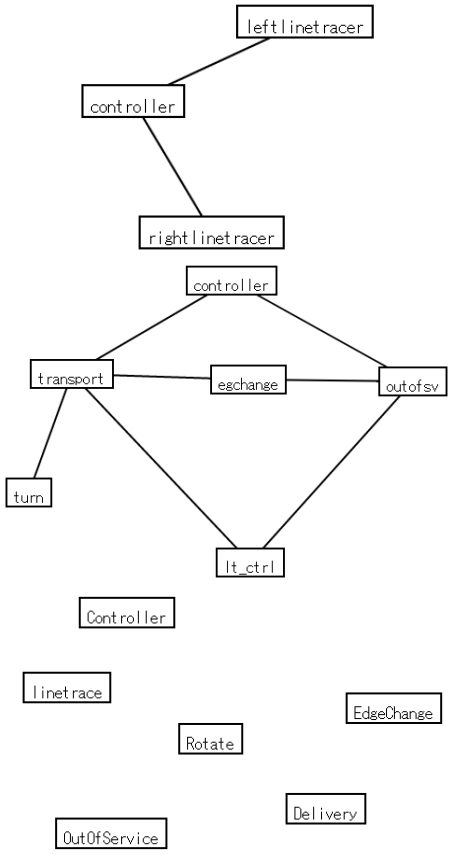

図 5 MDD ありグループのクラス図
Fig. 5 Class diagrams of the group with MDD.

ス」，「走り方に関するクラス」の 3 つに分類した結果及び総クラス数を**表 2** に示す．MDD なしと MDD ありグループの傾向を比較すると MDD ありグループでは，業務内容に関するクラス名のみであるのに対し，MDD なしグループでは走り方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．基礎演習 2 では，総合演習課題の内容の一部であるライントレースのモデルの作成を行っており，基礎演習 2 のモデルをそのまま利用しているまたは一部修正を加えて利用している数をモデルリポジトリから確認すると，MDD ありグループの 6 名中 4 名であった．

品質カテゴリに関するエラーを持つクラス数を**表 3** 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．

以上の結果より，MDD を利用しない場合は，開発する業務という抽象度の高い内容が思考の中心となり，走り方などの業務の実現方法には，意識がいきにくいため，複数のクラスに同一の機能が盛り込まれていることが多いと考えられる．一方，MDD を利用する場合には，機能ごとの単体テストを行いながら，機能の追加を行えるため，抽象度の低い業務の実現方法が思考の中心となり，全体としてどのような業務を行うクラス図なのかわかりにくいモデル図となっている．さらに，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 下）や，クラスが一つしかなく，適切な責務分割が行われていないモデルなど，品質に問題があるクラス図が多く見られた．

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 あり	全体を制御するクラスの数	1	1	1	1	1	1	1.0
	業務内容に関するクラスの数	4	4	3	3	4	3	3.5
	走り方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 なし	全体を制御するクラスの数	1	1	1	1	1	1	1.0
	業務内容に関するクラスの数	0	2	0	0	2	0	0.7
	走り方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

表 3 品質カテゴリに関するエラーを持つクラス数
Table 3 The total number of classes 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	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	不必要なクラス（振舞いが記述されていない）がある	0	1
Pragmatic	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
	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
	関連名が不適切	0	2

4.2 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，品質カテゴリーに関するエラーを持つ状態数を**表 4** 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図では，図 7 のように状態名がないモデルや状態名が a,b など状態の意味が読み取れないものなどが見られた．

Pragmatic 品質の一つに understandability がある [15]．文献 [16] によると，ステートマシン図の understandability と高い関係のあるメトリクスとして，Number of Activities(NA), Number of States(NS), Number of Transitions(NT) の 3 つを上げている．文献 [16] では，アクションを entry, do, exit の 3 つに分類して扱っているが，本研究で用いた DSL では entry アクションのみ記述できるため，本論文では NA を entry アクションの数とする．また，他のクラスにイベントを送信する状態が多いほど，モデルの understandability が低下すると考えられるため，Number of Send Event States(NSE) も計測した．計測したメトリクスの詳細は次の通りである．

- Number of Activities(NA)
  各クラスのステートマシン図それぞれの entry アクションの数のうち最大のもの．
- Number of States(NS)

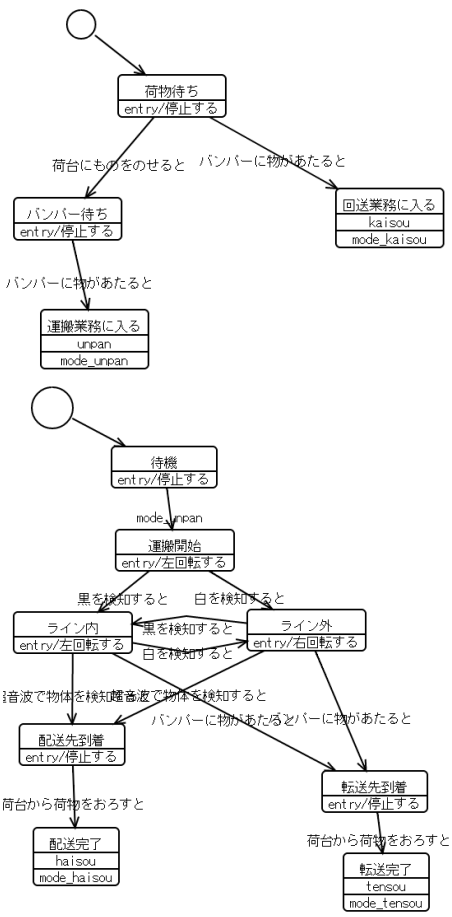

図 6 MDD なしグループのステートマシン図
Fig. 6 State machine diagrams of the group with no MDD.

表 4　品質カテゴリに関するエラーをもつ状態数
Table 4　The total number of stats 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	4	9
Pragmatic	状態名が不適切	0	7

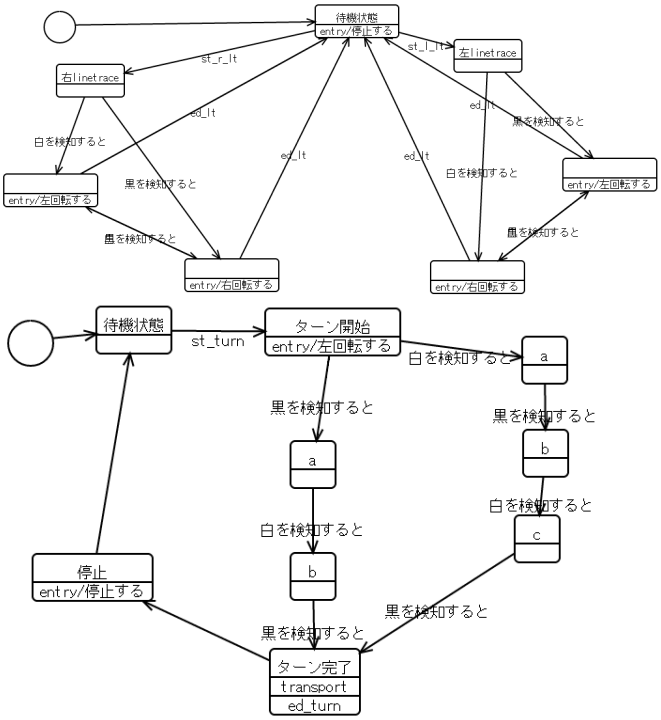

図 7　MDD ありグループのステートマシン図
Fig. 7　State machine diagrams of the group with MDD.

各クラスのステートマシン図それぞれの状態数のうち最大のもの.

- Number of Transitions(NT)
  各クラスのステートマシン図それぞれの遷移数のうち最大のもの．遷移数には，初期状態，通常の状態，イベント送信状態間すべての遷移を含める.
- Number of Send Event States(NSE)
  各クラスのステートマシン図それぞれのイベント送信状態数のうち最大のもの.

MDD なし，MDD ありグループそれぞれの被験者 A〜F 6 名のステートマシン図の understandability に関するメトリクスを**表 5** に示す.

表 5 によると，メトリクスの平均においては NA 以外のメトリクスはすべて MDD ありの方が高くなっている．個別の被験者の結果を見ると，MDD なしは各被験者のモデル間のばらつきが少ないのに対し，MDD ありではすべてのメトリクスにおいて高い数字になっているモデルがある．MDD ありの方が understandability が低いモデルが多く，複雑度が高いと考えられる.

表 5　ステートマシン図の understandability に関するメトリクス
Table 5　Metrics about understandability of state machine diagrams.

	MDD なし				MDD あり			
	NA	NS	NT	NSE	NA	NS	NT	NSE
A	9	9	11	2	7	19	20	7
B	6	6	8	2	5	12	15	7
C	8	8	11	2	4	6	8	3
D	8	8	10	2	5	9	10	3
E	7	7	9	2	5	7	12	7
F	7	8	12	2	12	15	26	0
平均	7.5	7.7	10.2	2.0	6.3	11.3	15.2	4.5

4.3　アンケート結果

講座終了後，次の受講者アンケートを実施した．Semantic カテゴリの一つとして，開発したシステムの達成率を評価するために，総合演習課題のマイルストーンを定義し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.

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%
全てのパターンで完走する	0%	17%

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 7** 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，ありの受講者伴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関わらず，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 8** に示す．MDD なしグループは全員が MDD ありグループは 67%が MDD を利用する方がモデリングしやすいと答えており，MDD 利用しない場合がよいと答えた受講生はいなかった．

表 7 取組み姿勢のアンケート結果
Table 7 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 8 モデリングしやすさのアンケート結果
Table 8 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%

5. 考察

MDD なし，ありの 2 つのグループに分けてモデリング演習を行った結果より，MDD の利用の有無それぞれの UML モデリング技術習得における利点を以下に示す．

- MDD なし
  - システムの業務に着目したモデリングを行える
  - モデリング記法に従い正確に表記できる
  - クラスや状態に適切な名前を付けられる
  - 読みやすいモデルになるように心がけられる
- MDD あり
  - システムの機能に着目したモデリングが行える
  - 短期間に機能を完成させることができる

これらのことより，UML モデリングの教育効果を上げるためには，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といえる．例えば，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及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．この時，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必要がある．その後に行う，業務システム（本稿では，自動搬送システムを用いた）開発の場合には，十分なクラス図の検討を行った後にコードの自動生成や動作確認を行うように指導するのが望ましいと考える．

6. おわりに

本論文では，UML モデリングの効果的な利用方法を検討するために，コードの自動生成及び動作確認を学習者の裁量で実施できる MDD ありグループとコードの自動生成及び動作確認を制限された MDD なしグループの 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図とアンケート結果を比較した結果，MDD なし，ありそれぞれに利点があり，どちらか片方の手法が優れているわけではないことが明らかになった．従って，UML モデリングの教育効果を上げるためには，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ことが分かった．今後は，上記の結果を踏まえた MDD ツールを用いた UML モデリング教育の検討及び講座の実施，開発させるシステムの課題の検討等を行いたい．

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進め [設計モデル編]，アイティメディア (2006).
[2] Chaudron, M. R. V., Heijstek, W. and Nugrohoi, A.: How effective is UML modeling ?, *Software and Systems Modeling*, Vol. 11, No. 4, pp. 571–580 (2012).
[3] Dranidis, D.: Evaluation of StudentUML: an Educational Tool for Consistent Modelling with UML, *Proc. Informatics Education Europe II Conference*, South-East European Research Center, pp. 248–256 (2007).
[4] 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).
[5] 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学習のためのインスタンス図自動生成システム，情報処理学会研究報告，Vol. 2011-CE-113, No. 8, pp. 1–9 (2011).
[6] Hasker, R. W.: UMLGrader: an automated class diagram grader, *J. Comput. Sci. Coll.*, Vol. 27, No. 1, pp. 47–54 (2011).
[7] Henry, S. F., Gardner, H. and Boughton, C.: Executable/Translatable UML in Computing Education, *Proc. Sixth Australasian Computing Education Conference* (2004).
[8] 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).
[9] Starrett, C.: Teaching UML Modeling Before Programming at the High School Level, *Proc. IEEE International Conference on Advanced Learning Technologies*, IEEE Computer Society, pp. 713–714 (2007).
[10] Ogata, S. and Matsuura, S.: Training of requirements analysis modeling with UML-based prototype generation tool, *Proc. 5th India Software Engineering Conference*, ACM, pp. 105–108 (2012).
[11] 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).

[12] Liggesmeyer, P. and Trapp, M.: Trends in Embedded Software Engineering, *IEEE Software*, Vol. 26, No. 3, pp. 19–25 (2009).

[13] Hiya, S.: clooca, Kyushu University (online), available from ⟨http://www.clooca.com⟩ (accessed 2012-11-12).

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

[15] Genero, M., Fernández-Sáez, A. M., Nelson, H. J., Poels, G. and Piattini, M.: Research Review: A Systematic Literature Review on the Quality of UML Models, *J. Database Manag.*, Vol. 22, No. 3, pp. 46–70 (2011).

[16] Miranda, D., Genero, M. and Piattini, M.: Empirical Validation of Metrics for UML Statechart Diagrams, *Enterprise Information Systems V* (Camp, O., Filipe, J., Hammoudi, S. and Piattini, M., eds.), Springer Netherlands, pp. 101–108 (2005).