ソフトウェア初学者へのモデリング教育におけるMDDの活用

赤山聖子[†1] 久保秋真[†2] 久住憲嗣[†3]
二上貴夫[†4] 北須賀輝明[†5]

モデリング教育は早期ほどよいと言われており，ソフトウェアエンジニアの育成においても，ソフトウェア初学者に対するモデリング教育の必要性が高い．しかし，モデリング教育では，学習者が「このモデルは合っているのだろうか？」という「理解性」や,「動くのはコードだからモデルを描く意味がない.」という「必要性」への疑問を持つこと，さらに上記に伴う「モチベーション」の低下が教育上の課題となっている．モデル駆動型開発（MDD）では，モデル上での検証とコードの自動生成ができるため，動作確認までの時間が短い，モデリングに集中して開発できるなどのメリットがある．MDD を初学者の教育に活用することで，上記の課題を軽減できると考えられる．

著者らは，ソフトウェア初学者に対する MDD を活用した教育プログラムを開発し，専門学校での実証講座を実施した．その結果，本プログラムにより，上記の課題を軽減することができ，モデリングスキルの向上を実現できることを確認した．

An Application of Model Driven Development (MDD) of Modeling Education for Software Beginners

Seiko Akayama,[†1] Shin Kuboaki,[†2] Kenji Hisazumi,[†3]
Takao Futagami[†4] and Teruaki Kitasuka [†5]

The modeling education is necessary for software engineers especially for beginners. However, the questions for understandability such as “ will this model be equivalent to the specification? ” and necessity such as “ this model will be meaningless since the execution process depends on the source code not on the model ” cause the decline of motivation of program development in the modeling educations. Model Driven Development (MDD) can verify the accuracy of models and generate the source codes leading that programmer can reduce the development time to check the software and can focus on the modeling process. The application of MDD for beginners can reduce the above described problems. We developed the education subject to utilize the MDD for beginners, and conducted the verification course on our College. As the results, we can confirm the improvement of modeling skills and reduce the above problems.

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)]．ソフトウェア開発では，UML を用いたオブジェクト指向モデリングを用いられることが多くなり，教育現場での教育の必要性も高まっている．しかし，モデル図はそのままでは動作しないため，理解性やモチベーションの低下が教育上の課題となっている．

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．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動作として確認することができるため，モデル図の評価手段の一つとして用いることができる．さらに，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．従って，MDD を初学者のソフトウェア開発教育に活用することで，モデリングに重点を置いた教育が行え，モデリングスキルの向上が図れると考えられる．

†1 九州技術教育専門学校
Kyushu Technical Education College
†2 株式会社アフレル
Afrel Co.,Ltd.
†3 九州大学システム LSI 研究センター
System LSI Research Center, Kyushu University
†4 株式会社東陽テクニカ
Toyo Corporation
†5 熊本大学大学院自然科学研究科
Graduate School of Science and Technology, Kumamoto University

MDD の教育への活用例としては，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[3)4)]．しかし，ソフトウェアエンジニア育成におけるオブジェクト指向モデリング及びソフトウェア開発におけるモデリング技術の活用方法の教育への活用はほとんど行われていない．

本稿では，ソフトウェア初学者に MDD を活用し，より短期間でのモデリングスキル及びモデリング技術の活用方法の習得ができるような教育プログラムの提案及び専門学校での実証講座の結果を考察する．

2. 教育目標

2.1 ソフトウェアモデリング教育の課題

組込みソフトウェアエンジニアの育成においてモデリング教育の必要性が高まり，ソフトウェアモデリングを重視したロボットコンテスト[5)6)] 等も実施されている．九州技術教育専門学校（以下，本校）では，組込みソフトウェア教育の一環として，2008 年から ET ロボコン[5)] に参加しており，組込みソフトウェアモデリング教育を行ってきた．ただし，モデリングの教育は難しく，学習者がモデリングを行う上で「このモデルは合っているのだろうか？」という疑問や，最終的に必要なのは動くソースコードなのだから「モデリングなんて意味がない.」という意見を口にし，学習途中で諦めてしまうことがある[2)]．本校の ET ロボコンの参加者も同様の傾向にあり，ET ロボコンの評価の対象になっているという理由で仕方なくモデル図を描いている状態で,「なぜ，モデリングが必要なのか？」という根本の理解させることが困難であった．このように，ソフトウェアモデリング教育においては,「理解性」,「必要性」,「モチベーション」に関する課題がある．

2.2 教育対象

モデリングは，細かい実装方法を知らない方が習得しやすいと言われていること[4)]，習得には時間がかかること[2)] などを踏まえ，ソフトウェア開発に関する学習の早い段階でモデルを用いたソフトウェア開発手法の教育を行うこととした．教育対象は，ソフトウェアエンジニアを目指している初学者，具体的には情報系の専門学校及び情報系学部に在籍する学生を対象と想定した．

2.3 教育プログラムの要件

上記の課題に対応する教育プログラムを開発するため，次のような要件を定義した．

- 一連の開発体験の中で開発におけるモデリングの利用方法を理解できること（理解性，必要性）
- モデルを作成したらすぐに動作を確認できる環境を提供すること（理解性）
- 理解度に合わせて，開発対象を限定できること（理解性）
- モデルを使って開発することはプログラミングと同等以上の価値があると実感できること（必要性）
- モチベーションの維持ができること（モチベーション）

上記の要件を踏まえて，本実践では，MDD を活用してモデリング教育を行うことを提案する．

2.4 教育目標

教育プログラムの要件を踏まえて，次のような教育目標を設定した．

- 静的・動的モデリングを行うための最低限のモデリング記法が分かる．
- MDD 手法を利用した開発手順が分かる．
- 静的モデリングにおいて，クラスに責務を割り当て，その責務が分かるクラス名をつけることができる．
- 動的モデリングにおいて，状態の変化と処理手順の違いが分かる．

3. MDD

3.1 MDD とは

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 の実現方法の一つとして，Executable UML[7)] がある．本プログラムでは，Executable UML を利用した MDD 教育を実施した．

3.2 Excutable UML

実行可能モデルと称される Executable UML とは，実行可能なセマンティクスとタイミング規則を UML 表記法のサブセットと組み合わせたものである．Executable UML における基本的な構成要素は，クラス図，ステートマシン図，アクション記述の 3 つである．構成要素の関係を**図 1** に示す．

Executable UML を用いてモデル化する際の手順を次に示す．

(1) システムの問題領域（ドメイン）を定義する．
(2) ドメインに対するクラス図を作成する．
(3) 各クラスのライフサイクルをステートマシン図で記述する．
(4) ステートマシン図の各状態のプロシージャをアクション言語で記述する．

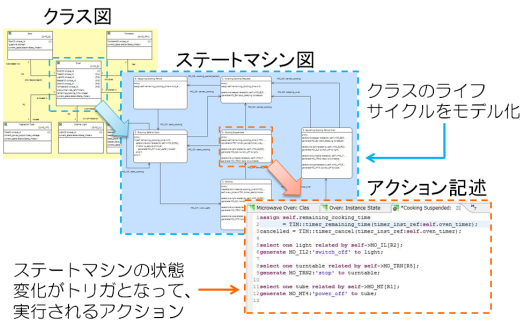

図 1 Executable UML の基本要素
Fig. 1 Fundamental elements of Executable UML.

(5) モデルを検証する.
(6) モデルをコンパイルし，ソースコードを生成する.

3.3 MDD のメリット

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) 実装と設計を分離できるため，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.

4. 教育プログラム

4.1 コンセプト

本プログラムでは，ソフトウェアの初学者に対し，モデリングのスキル向上やモデルを中心に置いた開発方法の理解を目的としている．2 章に述べた教育プログラムの要件を踏まえて，以下の 2 つを教育コンセプトとして，教育プログラムの開発を行った.

4.1.1 ソフトウェアモデリング教育に MDD を活用

MDD をモデリング教育に活用するメリットとしては，以下のようのものがある.

- モデリングに集中させることができる.
- 学習早期に一連の開発体験が行える.
- モチベーションの維持につながる.

MDD では，抽象度の高いモデリングの記述のみで，動作検証を実施することができるため，抽象度の変更に伴う混乱を招きにくい．さらに，開発対象以外の部分をブリッジという形で提供することで，学習者はアプリケーションドメインのみを開発すればよく，プログラミングやハードウェアの詳しい知識がなくても，一連の開発体験を行うことができる.

これらの特徴は，学習者のモチベーションの維持につながる.

4.1.2 要素技術と開発体験をスパイラル的に教育

教育項目を限定して最低限の要素技術を教育した後に，その技術を利用した一連の開発を実施させるという工程をスパイラル的に積み重ねることで，技術の必要性を理解させながら教育を行った．教育の概略を**図 2** に示す.

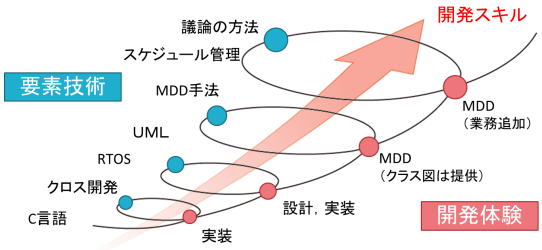

図 2 スパイラル教育の概略
Fig. 2 Spiral curriculum model.

この方法を用いることで，受講者のモチベーション維持につながるだけではなく，座学での講義により習得した技術の定着及びスキル化を図ることができる．開発体験においては，最初からすべての工程を開発することは困難であるため，受講者の開発対象領域を限定して，それ以外の部分は開発環境として提供し，徐々に開発対象領域を拡張していくことでスムーズに開発体験が行える工夫を行った.

4.2 教 育 項 目

本教育プログラムは，MDD 手法を活用することにより，効果的なモデリング教育を行うことを目的としており，主要な教育項目を UML の知識，MDD 手法，静的モデリング，動的モデリングとした．以下にそれぞれの教育項目の内容を示す.

4.2.1 UML の知識

UML には 13 種類の記法があるが，開発にすべてのモデルを利用するわけではなく，開発対象の分析や設計に必要なものを，開発者が適時利用する．UML は，様々な開発対象やプロセスに汎用的に利用できる反面，学習早期にはどの図を使ったらよいのだろうか？という疑問を持つことが多い．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.

本プログラムでは，Executable UML を開発言語として利用するため，クラス図，ステートマシン図を中心に教育を行った.

4.2.2 MDD 手法

MDD の基礎には，モデル変換という考え方がある.

変換ルールに基づいて，モデルからモデルまたは，モデルからコードに変換することにより，動くソフトウェアが出来上がるということを知ることで，モデルとコードがどのようにつながっているのか？ということを理解することができる．これにより，開発プロセスの中でモデルがどのように利用されるのかということへの理解が深まる．

4.2.3 静的モデリング

静的モデリングを行うクラス図は，モデルの骨格をなすものであり最も重要な図である．しかし，クラス図の作成には，開発対象の十分な理解と抽象化能力を必要とされるため初学者にとって作成が大変困難である．MDD 手法を用いることにより，実装にとらわれずクラス図の作成及びレビューに時間をかけることができる．本教育プログラムでは，特にクラスの責務分割を行えるようになること，クラスに責務の内容が分かる名前を付けるようになることを目標とした．

4.2.4 動的モデリング

動的モデリングには，ステートマシン図を用いた．ステートマシン図では,「状態」,「イベント」,「アクション」,「遷移」というキーワードと概念を理解させること，状態の変化と処理手順との違いを理解させることを目標とした．

4.3 教育プログラムの構成

作成した教材は、基礎編、応用編、PBL（Project Based Learning）編の 3 部である．主教育項目である MDD 手法やモデリング技術は，応用編で教育し，基礎編では組込みソフトウェア開発を行う上での基礎知識の習得を目的とした．PBL 編では，基礎編，応用編で習得した技術の定着と自律的にモデル中心の開発が行えるようになることを目標とした．

基礎編，応用編，PBL 編の各ステップで学習する基礎演習や総合演習の題材をできる限り同一のものにすることで，各ステップで学習する内容と事前のステップで学習した内容の関係を深めさせる工夫をした．各ステップでの教材間の関係を**図 3** に示す．

4.3.1 総合演習課題

基礎編及び応用編で実施した総合演習課題を紹介する．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 4**）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，それぞれに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．コース及び課題内容を**図 5** に，業務内容を以下に示す．

- 運搬業務
  - 配達先があるときは運搬業務を実施
  - 配達先へ運搬して，その後車庫へ回送
- 転送サービス
  - 配達先がないときは転送サービスを実施
  - 転送指示によって転送先へ運搬，その後車庫へ回送
- 回送業務
  - 荷物がないときは回送業務を実施
  - 配達先の有無に寄らず，車庫へ回送

図 4 自動搬送ロボット
Fig. 4 Auto transport robot.

4.3.2 基 礎 編

（1） 要素技術 A
組込みソフトウェアの一連の開発をする上で必要な，RTOS やクロス開発に関する知識の講義．

（2） 基礎演習 A
ロボットやセンサを動かすプログラムを組み合わせて，ライントレースなどを行う演習．

（3） 総合演習 A
基礎演習 A で学んだ，各センサの使うプログラムやライントレースのプログラムを組み合わせて，一連の業務をハンドコーディングで実装．

4.3.3 応 用 編

（1） 要素技術 B
UML の知識，MDD 手法，モデリング手法，モデルからのコード生成技法などの講義．

（2） 基礎演習 B
以下の 2 つの演習を行うことで，開発におけるモデルの利用方法及びモデルを利用して品質や開発効率が上がることを理解させる．

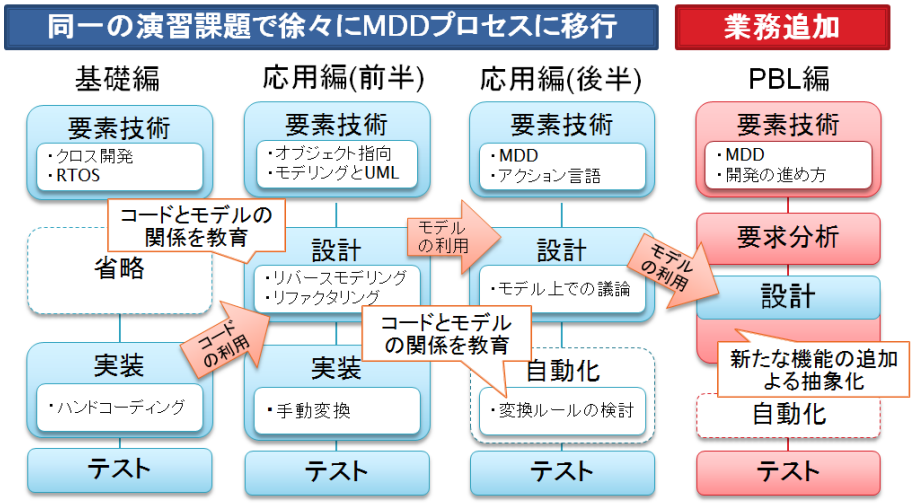

図 3 教材間の関係
Fig. 3 Relationship between educational materials.

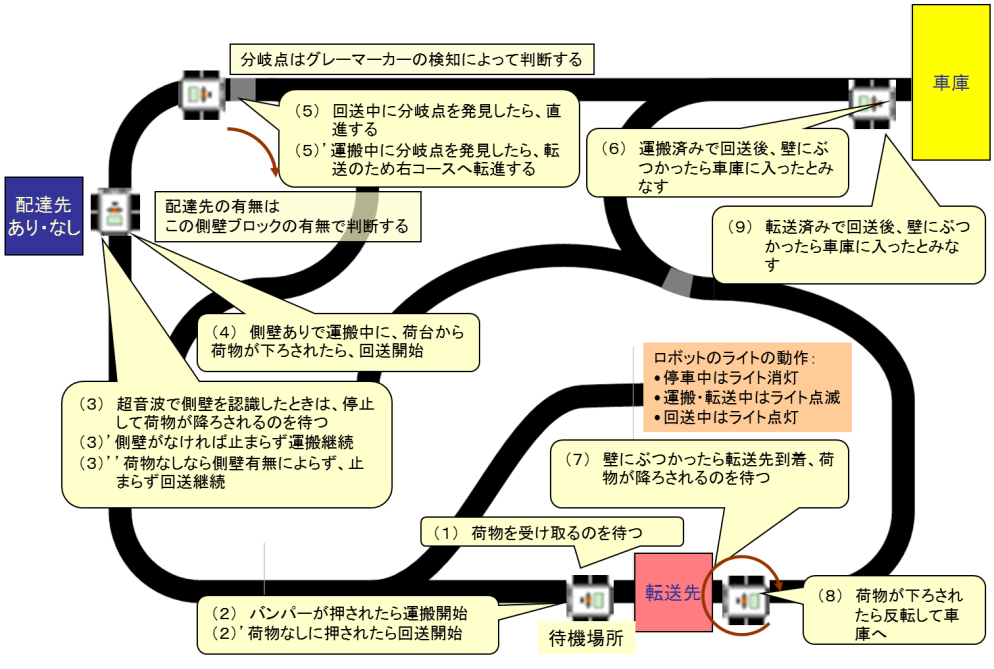

図 5 基礎編・応用編の総合演習課題のコース
Fig. 5 Course layout for basic and advanced classes.

(a) 基礎演習 A のプログラムをリバースモデリングする.

(b) モデルコンパイラになりきり，変換ルールに基づいてモデルからコードへの手動変換を行う.

(3) 総合演習 B

設計，実装をトップダウンで開発する演習であり，総合演習 A と同一の業務を MDD 手法で開発する．**図 6** に示すように，自動搬送システムの全体的な制御を行うクラス（AutoTranspoter），ラインに沿って走行するクラス（LineTracer）の 2 つ以外のクラス内のステートマシンは，実装済みのものを提供した．これにより，命名済みの 2 つのクラスの責務のみに着目して責務分割を検討する方法を学ぶことができる．また，各クラス内で起こる主要なイベントを定義した状態で配布し，各自が独自のイベントを定義する際に適切な抽象度で定義するための指

針となるようにした．さらに，MDD ツールを用いて，コードの自動生成を行うことで，モデル図のレビューを複数回行い，その度に動作確認をすることができるため，短期間でのモデリングスキルの向上が見込める．なお，MDD ツールには BridgePoint（Mentor Graphics）を用いた．

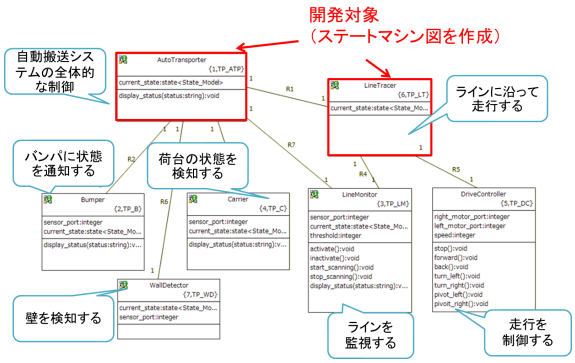

図 6 自動搬送システムのクラス図
Fig. 6 Auto transport system class diagram.

4.3.4 PBL 編

総合演習 B に対して業務を追加するソフトウェア開発案件を，MDD を用いてチーム開発を行う．これにより，モデルを利用して開発することで，仕様変更に対応しやすいことを実感してもらうことがねらいである．業務の追加，変更内容は，次の通りである．(1) タッチセンサを使用したバンパを廃止し，荷台を 2 つにする．同じデバイスでも別の利用方法をした場合の名前の付け方の工夫や多重度の変更を期待する．(2) 搬送コースを変更する．搬送コースが今後も変更する可能性があることを予想した設計になることを期待する．(3) 基地局と通信をして，搬送ルート及び搬送個数の変更を行う．通信機能を設けることで，新しいデバイスを利用することになるため，通信用ブリッジが必要になることを気づいてもらう．

なお，ブリッジの開発に関しては，教育対象外であるため，ブリッジが必要だと気づいたチームには，提供をする．自動搬送ロボットと基地局の通信には，bluetooth を用いた．通信の様子を**図 7** に示す．また，基地局は，PBL 期間中に実行モジュールを提供した．

本 PBL では，MDD の開発スキルを身に着けることを主目的としており，プロジェクトの進め方に関しては，できるだけガイドを設けて，スムーズにプロジェクトが進むように考慮した．なお，プロジェクトを円滑に進めるために，プロジェクトファシリテーション手法[8] を用いた．ツールとして，本 PBL 用にアレンジした下記の 2 つを利用した．

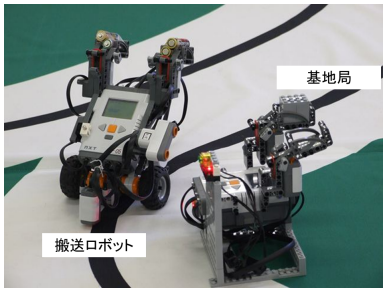

図 7 自動搬送ロボットと基地局の通信
Fig. 7 Comunication between auto transport robot and base station.

タイムボックスかんばん: チーム内での進捗状況管理に使う表で，1 日のチームの目標と各タイムスロット毎の個人単位の目標，ToDo，うまくいったこと，ダメだったところを記載する．

KPTT 表: 夕会時に，Keep（良かったこと），Problem（問題だと認識していること），Try（試してみたいこと）を記載し，その中から ToDo（次の日に試してみること）を決める．

また，1 日の始まりには朝会，終わりには夕会を行うこと，議論した内容を記録として残すことを義務付けた．

14 日間の PBL 期間の全体の流れを**表 1** に示す．MDD では，モデルからの動作検証により，機能面の検証はできる．しかし，責務が過大なクラスがある，クラス名がおかしいなどの質の悪いクラス図でも，その内部にステートマシンを記述することで，動くソフトウェアを作ることはできる．品質や再利用性に優れたモデル図になるよう PBL 期間中に中間レビューを入れることで対応した．なお，表 1 の提供資料は，チームが必要だと気づいて要求した場合に限り提供することとし，機能を作るにはどんなものが必要かをチーム内で考えるように誘導した．

5. プログラムの評価

5.1 実証講座

開発した教育プログラムの有効性を検証するために，専門学校生に対する実証講座を行った．前提知識の違いにより，習得の変化を見るために 1 年生及び 3 年生を対象に実証講座を実施した．それぞれ，講座 1，講座 2 と呼ぶこととする．

5.1.1 講 座 1

4.3 節に述べた基礎編，応用編，PBL 編の講座を専門学校の 1 年生を対象に実証講座を実施した．講座 1 の概要を**表 2** に示す．受講者は，入学後半年間 C 言語

表 1 PBL の流れ
Table 1 Schedule of PBL course.

項目	内容，到達目標，提供資料など
キックオフ 1 日目	PBL の説明 グループワークの進め方，ロールプレイ
キックオフ 2 日目	業務説明 ミニ PBL による環境構築
PBL2 日目	目標：走行部分の完成 応用編のモデル図を提供
PBL3 日目	通信ブリッジの提供
PBL4 日目	基地局プログラム（実行モジュール）の提供
PBL5 日目	目標：通信部分の完成
PBL7 日目	目標：基本の処理完成 中間レビュー
PBL9 日目	目標：業務完成
PBL10 日目	目標：システムテスト
成果発表準備	発表内容の説明 発表準備，リハーサル
成果発表	成果発表，デモ 講評，ふりかえり

及び IT スキル標準レベル 2 を目指すカリキュラムを受講した学生である．なお，基礎編の受講から応用編受講までの期間がおよそ 1ヶ月，応用編受講から PBL 編受講までの期間がおよそ 3ヶ月であった．

表 2 講座 1 の概要
Table 2 Outline of the No.1 trial.

基	受講者	専門学校の 1 年生 21 名
	前提知識	ポインタ，構造体を除く C 言語の文法
礎	演習形態	ペア（1 ペアに PC1 台，ロボット 1 台提供）
	期間	3 日間（1 日 7 時間）
編	テスト	なし
	アンケート	講座内容，知識・スキルに関して
	受講者	専門学校の 1 年生 20 名
応	前提知識	基礎編を受講したもの
	演習形態	ペア（1 ペアに PC1 台，ロボット 1 台提供）
用	期間	4 日間（1 日 7 時間）
	テスト	UML に関する知識（2 日目）
編		MDD に関する知識（4 日目）
	アンケート	講座内容に関して
P	受講者	専門学校の 1 年生 17 名
	前提知識	基礎編，応用編を受講したもの
B	演習形態	チームあたり 4，5 名の 5 チーム
	期間	14 日間 （キックオフ：2 日間，
L		成果発表：2 日間を含む）
	テスト	なし
編	アンケート	講座内容，知識・スキルに関して

5.1.2 講 座 2

専門学校の 3 年生で，C 言語，JAVA，SQL，UML のスキルを持っている，IT スキル標準レベル 2 程度の学生に対して，応用編の実証講座を行った．講座 2 の概要を**表 3** に示す．受講者 9 名中 8 名が ET ロボコンの経験者であり，ロボットの組込みプログラミングのスキルもあった．UML の前提知識はあったため，教育項目は 4.2 節から UML の知識を除いたものとした．講義時間は，講座 1 の応用編の 28 時間に対して，講座 2 は 20 時間とした．

表 3 講座 2 の概要
Table 3 Outline of the No.2 trial.

受講者	専門学校の 3 年生 9 名
前提知識	C 言語，JAVA，SQL，UML
演習形態	ペア（1 ペアに PC2 台，ロボット 1 台提供）
期間	4 日間（1 日 5 時間）
テスト	UML に関する知識（2 日目）
	MDD に関する知識（4 日目）
アンケート	講座内容に関して

5.2 評 価 項 目

モデリング教育の課題である，「理解性」，「必要性」，「モチベーション」の 3 つの側面においてプログラムの評価を行う．評価には，受講者のテスト及びアンケート結果，モデル図などの成果物，演習課題の達成率，講師のコメントを用いた．

5.2.1 理 解 性

理解性に関しては，4.2 節に定義した教育項目に関して評価を行う．応用編実証講座における UML の知識及び MDD 手法に関する確認テストを行った．UML の知識がまったくなかった講座 1 の受講者の UML の知識に関する正答率は，平均 65%程度であったが，演習で利用したクラス図，ステートマシン図に関する問いでは，平均 75%の正答率があり，MDD ツールを用いて動かしながら学べたことで理解度が向上したと考えられる．また，MDD 手法に関しては，講座 2 の受講者の平均正答率が 80%と高く，UML の前提知識があったため，MDD 手法及びモデルを中心にした開発方法の理解に注力することができたためであるといえる．

応用編総合演習では，10 個のマイルストーンを定義し，達成率を評価した．演習課題の達成率を**表 4** に示す．提供したクラス図の実装されていない 2 つのクラスの内部のステートマシンを作成することで，業務を実現するソフトウェアができる．ほとんどの受講者が 4 番まで達成できたが，5 番のマーカ検知でつまづく受講者が多かったため，それ以降の課題内容を実装できていない受講者が多かった．ただし，つまづいた原因は，マーカ検知のパラメータ調整の部分であり，動的モデリングスキルを身に着けることはできたといえる．

PBL 編では，仕様の変更に伴い，クラスの責務を整理してクラス図を作成するかがメインのテーマで

表 4 応用編総合演習課題達成率
Table 4 Achievement rate of the advance class exercise.

マイルストーン	講座 1	講座 2
1) ライントレース出来る	100 %	100 %
2) 側壁を検知して止まる	78 %	100 %
3) 荷下ろしを検知して回送する	78 %	100 %
4) 車庫に入って止まる	89 %	100 %
5) マーカを検知して振る舞いを変える	56 %	75 %
6) 右コースへ転進する	44 %	75 %
7) ラインエッジをトレースする	33 %	75 %
8) マーカを無視する	33 %	75 %
9) 反転する	33 %	50 %
10) 全てのパターンで完走する	11 %	50 %

あった．PBL の 7 日目に講師によるクラス図の中間レビュー，最終日に成果発表を行った．追加業務の内容を踏まえて，応用編のクラス図から変更を期待する項目を評価項目とし，達成率を評価した．評価結果を**表 5** に示す．

表 5 PBL 編実証講座のクラス図評価結果
Table 5 Evaluation Results of class diagrams in PBL trial.

評価項目	中間レビュー	成果発表
1) バンパクラスの削除	60%	80%
2) 荷台クラスの多重度の変更	0%	20%
3) 基地局との通信を行うクラスの追加	40%	80%
4) 搬送ルートを管理するクラスの追加	0%	60%
5) その他，責務に応じたクラスの追加	0%	80%

通信を行うクラスに関しては，Test という名前で実装済みのものを受講者に配布した．中間レビュー時には，Test という名前のまま追加されているチームが多かったが，その場合は，3) 番未達成とした．また，応用編でバンパとして使用していたタッチセンサを PBL 編では荷台として利用しているにも関わらず，バンパクラスが残っている，責務分割がうまくなされておらず，クラス図のステートマシンが大きなものになっているなどが特徴的であった．MDD では，名前の付け方や責務の割り当て方にかかわらず，クラスの動的ふるまいを規定することで，コードの自動生成・動作検証ができるため，課題の機能完成を急ぎ，クラス設計をせずに開発を進めがちになることが分かった．

中間レビューでは，責務の割り当て方やクラス名の付け方を中心に指導したところ，中間レビュー以降は，各班が積極的にモデルのチーム内レビューを実施しており，成果発表時には，期待したクラス図に近いものになった．達成率の低い荷台クラスの多重度に関しては，多重度を増やすべきと認識しているチームが複数あったが，それに伴いアクション記述をどのように記述したらよいかわからず，荷台クラスを 2 つ作ることで対応しているチームが多かった．この点に関しては，事前にアクション記述の教育する時間を設けることで，対応できると考える．クラス図レビューが終了 3 日前で時期的に遅かったが，その後クラス図の再検討，クラス内部の振る舞いの実装を行い，成果発表時には全チーム課題内容の 70%程度の業務が実現できており，MDD を用いたことで，レビューから改善のスピードが早くなったことを確認した．MDD 教育と適切なタイミングでのレビューとを組み合わせることで，短期間でのモデリングスキルの向上が図れることが分かった．

5.2.2 必 要 性

受講者のコメントでは,「モデル図を見ながらみんなで議論できたため，他の人がやっていることが理解しやすかった.」,「プログラミングよりも見やすくて作業効率が上がった.」,「次回も MDD で開発がしたい.」などの意見があり，モデルを使って開発することのよさを理解させることができた．さらに，PBL 編終了後のアンケートで，必要性に関する下記のアンケートを実施したが，ともに 90%以上の受講者が「そう思う.」と答えており，モデリングの必要性を理解できたといえる．

- MDD を用いることで開発効率は上がると思うか?
- MDD を用いることで品質は上がると思うか？

5.2.3 モチベーション

各講座ごとで受講者の取り組み姿勢に関するアンケートを実施した．その結果を**表 6** に示す．

表 6 取組み姿勢のアンケート結果
Table 6 Motivation survey results.

	講座 1			講座 2
	基礎	応用	PBL	
とても熱心	10 %	16 %	24 %	25 %
熱心	38 %	21 %	29 %	63 %
どちらかというと熱心	24 %	37 %	35 %	13 %
あまりやる気がなかった	29 %	26 %	12 %	0 %
やる気がなかった	0 %	0 %	0 %	0 %
全くやる気がなかった	0 %	0 %	0 %	0 %

基礎編の実証講座よりも MDD を教育した応用編，PBL 編の方が「とても熱心」に取り組んだ受講者が多く，MDD 教育によりペアやチームでモデル図を見ながら一緒に開発できたこと，MDD 手法という新しい方法を学べたことがモチベーションの維持につながったと考えられる．また，PBL の期間中は，放課後遅くまで残って，残業をやっているチームが多く見られた．MDD をモデリング教育に用いることにより，モ

デリング教育におけるモチベーション維持が難しいという課題を軽減できることを確認した．

5.3 考　　察

専門学校の 1 年生に対して実施した講座 1 では,「難易度が高かった」,「時間が短すぎた」という受講者のコメントが多かった．前提知識がほとんどない状態で，組込みソフトウェアの開発，モデリング，MDD 手法など幅広い分野を短期間の実証講座に盛り込みすぎたことが原因であった．また，講座を集中講義として実施したことで，知識の定着する期間がなかったといえる．ただし，短期間の実証講座期間に演習課題の一部の業務を，全受講者が実現できていたこと，モデル図をみんなで議論できていたこと，モデルの必要性を理解できるようになったことを踏まえると MDD による開発方法やモデリング手法を短期間で習得できるプログラムであったと考える．

一方，同一教材を用いて対象を変えて行った講座 2 では，受講者がプログラミング，モデリングの前提知識があったため，モデルを中心に開発する方法や，MDD 手法の理解が高かった．これらのことより，同一の演習課題であっても，受講対象に合わせた，情報提供をすることで，教育項目のバリエーションを増やすことができる．MDD では，ドメインを分割して開発することができるため，情報提供に変化を持たせやすいという利点が確認できた．

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．本プログラムでは，レビューを入れることで対応したが，今後さらにモデルの品質に関して意識を持たせる工夫が必要である．今後の運用方法としては，基礎編，応用編に関しては，通常の半期または通年のカリキュラムとして，毎週 1 コマ実施するなどの形態が望ましいと考える．ただし，PBL に関しては擬似的な開発プロジェクトを体験するために，集中での実施が望ましいと考える．

6. おわりに

本稿では，ソフトウェアモデリング教育における課題として,「理解性」,「必要性」,「モチベーション」をあげ，その課題の解決に MDD 教育を活用することを提案した．専門学校での実証講座では，MDD と適切なタイミングでのレビューを組み合わせることで，モデリングスキルの向上が図れることを確認した．また，受講者へ提供するモデル図や開発環境を変化させることで，受講者にレベルに合わせた教育が行えることも分かった．

今後は，実施期間，教育対象者と情報提供内容及び課題内容の組合せ，レビュー時期とレビュー項目の検討等を行い，さらにモデリングスキル向上が行える教育プログラムの開発を行う．

謝辞　本教育プログラムの開発は，文部科学省研究拠点形成費等補助金（産学連携による実践型人材育成事業）による助成のもとで行った．

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) 新井玲子: UML 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).
3) 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 (2010).
4) Starrett, C.: Teaching UML Modeling Before Programming at the High School Level, *Advanced Learning Technologies, IEEE International Conference on*, pp. 713–714 (2007).
5) ET ロボコン実行委員会: ET ロボコン 2011, `http://www.etrobo.jp/2011/`.
6) 二上貴夫: I 見聞録 : MDD ロボットチャレンジ 2010, 情報処理, Vol. 52, No. 4, pp. 572–576 (2011).
7) Mellor, S. J. and Balcer, M. J.: Excutable UML-MDA モデル駆動型アーキテクチャの基礎-, テクノロジックアート (2003).
8) 平鍋健児, 天野勝: プロジェクトファシリテーション価値と原則編, Vol. 8 (2011).