モデル駆動開発を用いたオブジェクト指向モデリング教育に関する研究

赤山，聖子

https://doi.org/10.15017/1543994

出版情報：九州大学，2015，博士（工学），課程博士
バージョン：
権利関係：全文ファイル公表済

モデル駆動開発を用いたオブジェクト指向モデリング教育に関する研究

平成 27 年 8 月

赤山聖子

目次

図目次

表目次

概要

ソフトウェアの開発において，設計品質の向上が求められている．組込みソフトウェアはその特性上，高度な信頼性や安全性が求められるが，ソフトウェアの大規模化・複雑化によって，従来からの熟練者の勘と経験に頼った手法では，その設計品質を維持できなくなってきている．そうした中，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．

ソフトウェアシステムの設計・開発にはオブジェクト指向の考え方を用いられることが多くなっており，オブジェクト指向モデリング（Object-Oriented Modeling：OOM）のための標準化した仕様記述言語の 1 つである UML（Unified Modeling Language）を用いた OOM 教育を行う試みは数多くなされている．しかしながら，モデリングの教育においては，モデルの記述のみに閉じた教育だけでは，その妥当性の検証ができず，学習者が「このモデルは合っているのだろうか？」という懸念を抱くことが多く，その解決が教育上の大きな課題となっている．

一方，UML に基づくオブジェクト指向ソフトウェア開発プロセスとして広く採用されている Rational Unified Process （RUP）があり，そこでは，反復的開発を重視しており，正しい要件や設計をただ単に推測するのではなく，現実の構築とテストからフィードバックを得ることで，要件や設計を修正する機会が必要だとしている．モデリング教育においても，上記と同様にモデルの挙動を確認し，よりよいモデルへと改善／修正していく工程が行えれば，モデリング教育の課題を解決することができ，モデリング技術の習得に有益であると考えられる．

本研究では，上記のモデルの挙動を確認し，よりよいモデルへと改善／修正していく工程をスムーズに行う手段として産業界で実用化が進んでいるモデル駆動開発（Model Driven Development： MDD）をモデリング教育に活

用することを提案する．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 では，モデル上での検証とソース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という利点がある．

本論文では，上記を背景にして，モデリング教育への MDD の活用の有用性及び活用方法に関する下記の研究課題を明らかにする．なお，本論文では，モデルからソースコードを自動生成する機能をモデルコンパイラ，モデルの描画機能及びモデルコンパイラを備えたツールを MDD ツールと呼ぶこととする．

研究課題 1　MDD の活用

(1) **初学者の MDD 活用の可否：**　初学者に対して MDD を活用したモデリング教育が行えるのか？

(2) **モデルコンパイラの有用性：**　モデルコンパイラの利用することで，学習者が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，モデリングスキルの向上につながるのか？

研究課題 2　MDD の活用方法

(1) **モデルコンパイラの活用タイミング：**　モデルコンパイラの活用タイミングをどのようにしたらよいのか？

(2) **実行可能モデリング言語の選択：**　どのような、実行可能モデリング言語を利用した教育が効果的なのか？

初めに，研究課題 1-(1) 及び (2) の検討に寄与することを目的として，汎用的なモデリング言語（Executable UML）及び MDD ツール（BridgePoint）を用いた講座及び教育用のドメイン特化モデリング言語（Domain-Specific Modeling Language： DSML）を用いた講座を実施した．

次に，研究課題 2-(1) の検討のために，モデルコンパイラを学習者の裁量でいつでも利用できるグループ（MDD ありグループ）と演習の最後のみ利

用できるグループ（MDD なしグループ）の 2 種類に分けて講座を実施した．以上の講座の結果を踏まえ研究課題 2-(2) の考察を行う．

本論文の研究成果は，以下の通りである．

1. 初学者向けの OOM 教育として，MDD を活用することを提案した．MDD 教育は，プログラミングや UML モデリング等のソフトウェア開発教育の後に実施されることが一般的あり，初学者向けに MDD 教育を行われた事例は少ない．本論文では，MDD を活用した初学者向けの OOM 教育プログラムを開発し，実証講座を行うことで，プログラミング技術が十分にない初学者でも MDD による開発が行えること明らかにした．さらに，MDD におけるソースコードの自動生成機能を利用し，作成したモデルをすぐに実行して確認することで，モデルの不具合の一部を機械的に検出できるため，教員の負担軽減及び学習者の自己学習の促進を可能にした．
2. 教育用 DSML を開発し，よりスムーズな OOM 教育を行える環境の整備を行った．MDD に用いられる汎用的なモデリング言語及び実開発向けのツールでは，初学者が言語やツールに慣れるのに時間がかかり，主とする学習内容以外の部分に時間をとられるというデメリットがあった．本論文では，事前の実証講座の結果を踏まえ，UML の記法に合わせた教育用の DSML を開発することで，より短期間でのモデリングスキルの習得を可能にした．
3. MDD ツールを活用した OOM 教育を行う場合に，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の提案を行った．MDD ツールを活用した OOM 教育においては，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問題点がある．従って，モデリング教育において，コードの自動生成を行うモデルコンパイラをどのタイミングで利用するかが重要であり，本論文では，モデルコンパイラの利用のタイミングの異なる 2 つのグループの比較実験により，MDD ツールの利用がモデルに与える影響を明らかにすることで，より効果的な MDD のツールの活用方法の提案を行った．

第 1 章　はじめに

本章では，研究の背景として，ソフトウェア開発の現状と課題及びソフトウェア開発教育の課題に関して述べ，背景を踏まえた研究の位置付け及び目的を述べる．

1.1　背景

1.1.1　ソフトウェア開発の現状と課題

近年，エンタプライズ系ソフトウェアや組込みシステムソフトウェアは，生活や経済社会活動の基盤として適用を拡大しており，ソフトウェアが大規模・複雑化している．それに伴い，ソフトウェアの QCD（Quality, Cost, Delivery）の確保が困難になってきている．2012 年度「ソフトウェア産業の実態把握に関する調査」による，組込み系，エンタプライズ系各々の目標 QCD の不達成の原因を図 1.1, 図 1.2 に示す．

これによると，組込み系，エンタプライズ系ともに，「システム／ソフトウェア設計工程における問題」が上位になっている．次に，この問題の解決策を図 1.3, 図 1.4 に示す．組込み系，エンタプライズ系ともに，「技術者のスキル向上」が上位になっており，特に組込み系では，第一の解決策として挙げられている．

同調査における組込み系ソフトウェア開発の課題を図 1.5 に示す．組込みソフトウェア開発において最も改善の必要がある課題は，設計品質の向上である．さらに，2005 年〜2011 年の調査においても同様に設計品質の向上を第一番の課題としてあげられており，近年の組込みソフトウェア開発において最も解決の必要性がある課題であるといえる．

同調査における設計品質向上の課題の解決策を図 1.6 に示す．技術者のス

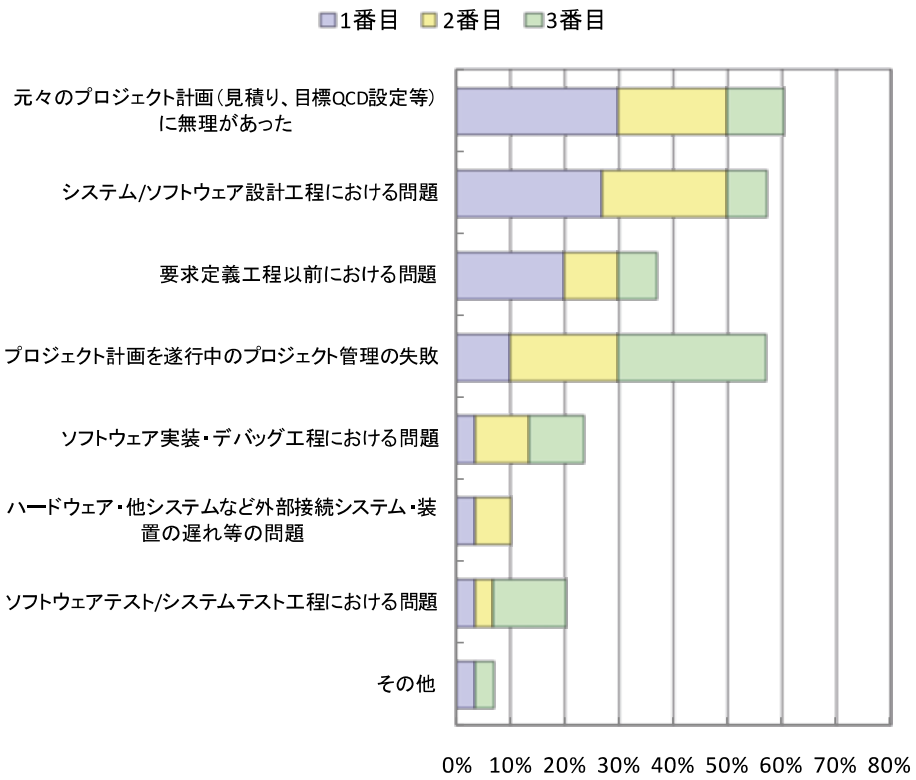


図 1.1　目標 QCD の不達成の原因：
組込み系 N=79（出展：“2012 年度「ソフトウェア産業の実態把握に関する調査」調査報告書”, 独立行政法人情報処理推進機構, 2013 [1)]）

キル向上を最も重視しており，スキル向上のための教育の必要性が高い．

組込みソフトウェアはその特性上，高度な信頼性や安全性が求められるが，ソフトウェアの大規模化・複雑化によって，従来からの熟練者の勘と経験に頼った手法では，その設計品質を維持できなくなってきている．そうした中，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．ソフトウェア開発にモデリング技術を活用には，下記のようなメリットがある [5)]．

- 正確な設計が可能となる．
- 設計が見えるようになり，レビューの効率が向上する．
- コミュニケーションが円滑になる．
- シミュレーションや検証をツール化できる．
- 標準化が促進され再利用が容易になる．

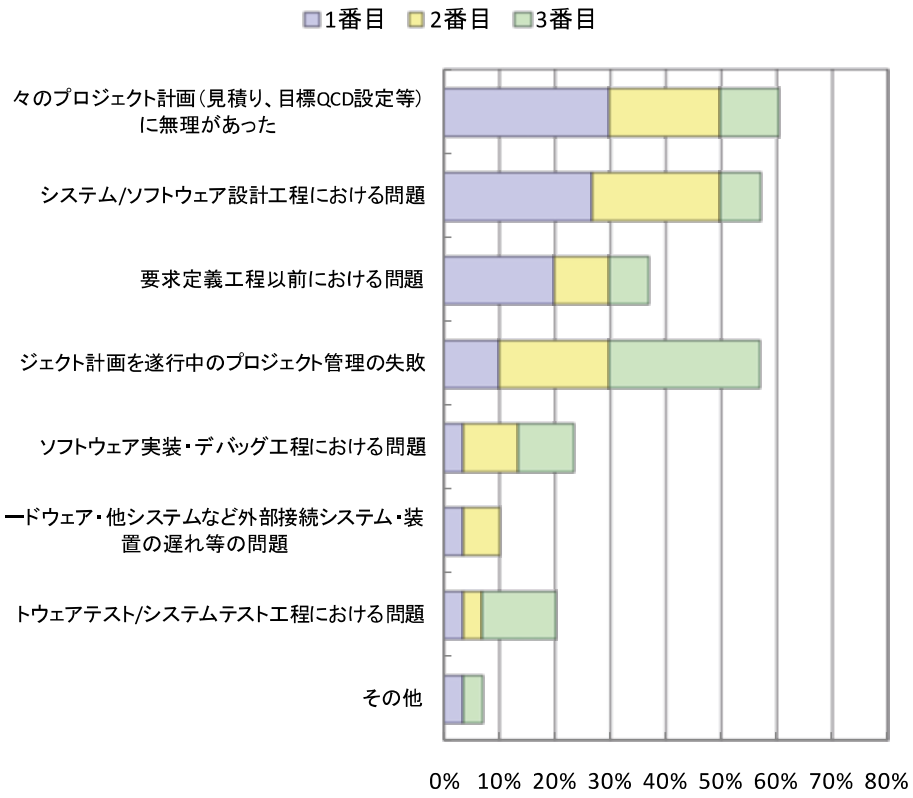


図 1.2　目標 QCD の不達成の原因：
エンタプライズ系 N=30 （出展：“2012 年度「ソフトウェア産業の実態把握に関する調査」調査報告書”, 独立行政法人情報処理推進機構, 2013 [1)]）

また，組込みシステムの開発力強化のためには、開発の上流工程における取り組み強化が求められており，特にモデルベース開発（モデルベース設計・モデルベース検証等）が開発力強化に有効であるとの指摘がある．

しかしながら，2010 年版組込みソフトウェア産業実態調査 [2)] によれば，わが国の組込みシステム開発においては，古典的なモデルベース手法である状態遷移図／表については 8 割程度のプロジェクトで利用されているのに対して，UML（Unified Modeling Language），制御モデルや外界モデル（プラントモデル）等の先端的モデルベース手法については，その適用は 1〜3 割程度にとどまっている（図 1.7）[6)]．

2010 年度から 2012 年度において，「自動コード生成」で使用した上位言

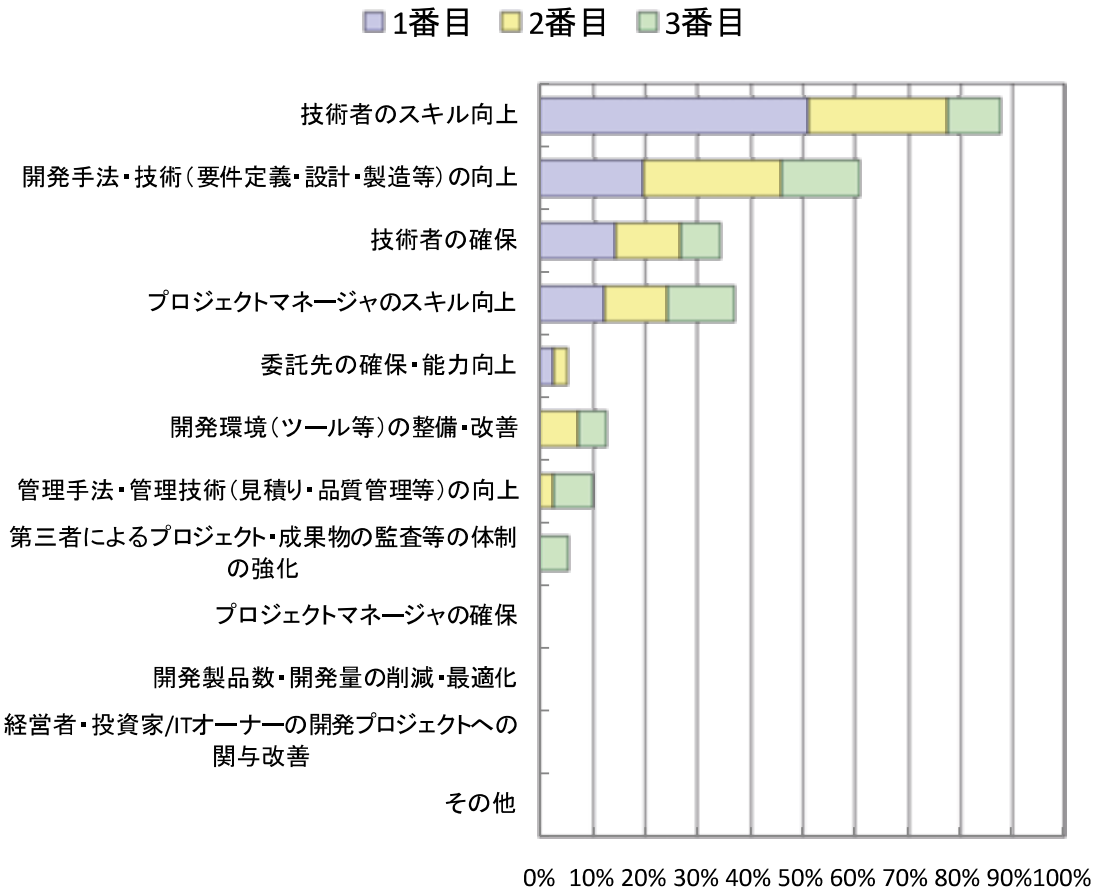


図 1.3 「システム/ソフトウェア設計工程における問題」に対する解決策：組込み系 N=41 （出展：“2012 年度「ソフトウェア産業の実態把握に関する調査」調査報告書”, 独立行政法人情報処理推進機構, 2013 [1)]）

語（モデルベース言語等）の比率を図 1.8 に示す [*1]．これにより，自動生成を前提とする場合の利用言語において，UML の利用率が増加傾向にあることが分かる．

モデルベース手法への適用にあたっては，組込みシステム技術者がモデルベース設計技術やモデルベース検証技術を扱うスキルを修得することが必要となるが，従来のソースコードベースの設計・検証技術と，基盤となる知識体系が異なるためスキルの修得を困難としており，このことがモデルベース手法の適用を難しくしている要因の一つと言われている [6)]．

[*1] 2010 年版組込みソフトウェア産業実態調査報告書及び 2011，2012 年度「ソフトウェア産業の実態把握に関する調査」調査報告書のデータを基にグラフを作成

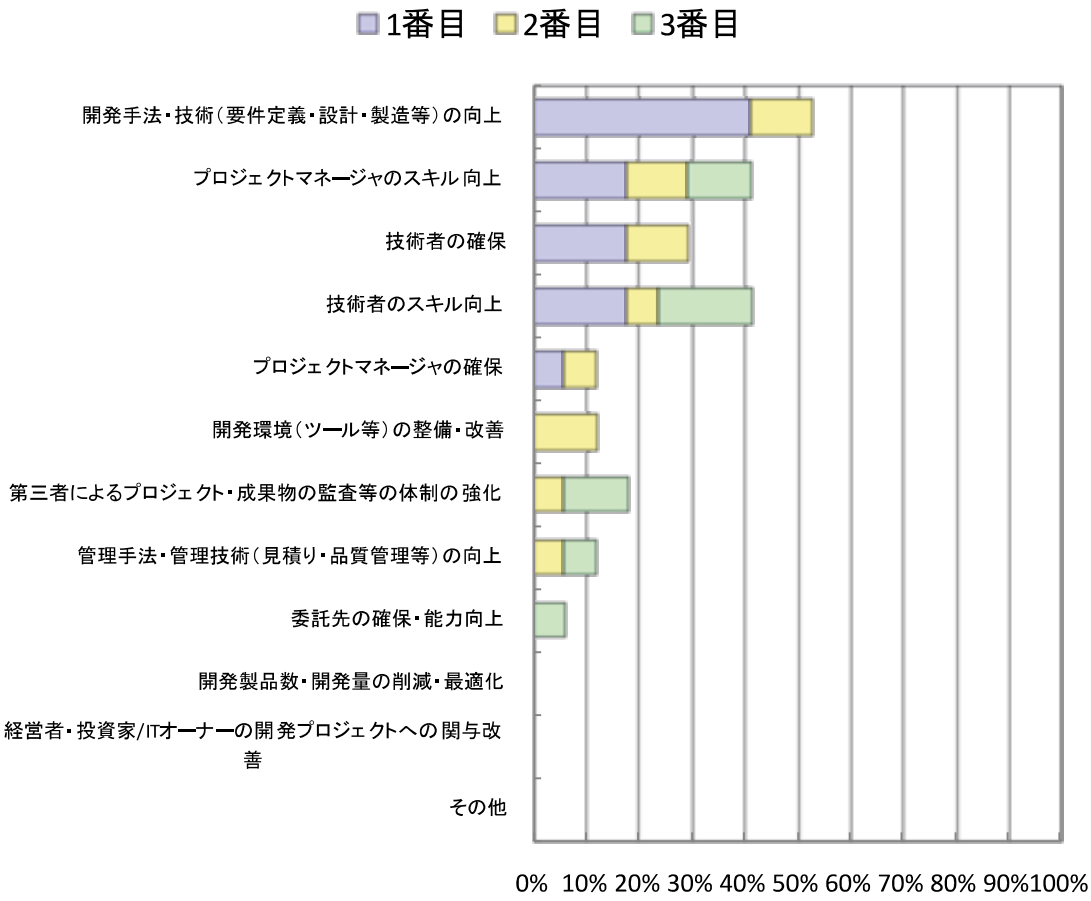


図 1.4 「システム/ソフトウェア設計工程における問題」に対する解決策：エンタプライズ系 N=17（出展：“2012 年度「ソフトウェア産業の実態把握に関する調査」調査報告書”, 独立行政法人情報処理推進機構, 2013 [1]）

以上のことから，組込みソフトウェア開発においてモデルを利用した開発を行える技術者の養成のための教育，特に UML を用いたオブジェクト指向モデリング（Object-Oriented Modeling：OOM）教育の必要性が高いといえる．

1.1.2 ソフトウェア開発教育の現状と課題

本論文では，ソフトウェア開発，特に組込みソフトウェア分野におけるモデリング技術の教育を対象とする．ここでは，組込みソフトウェア開発の現状及びソフトウェアモデリング教育の課題に関して述べる．

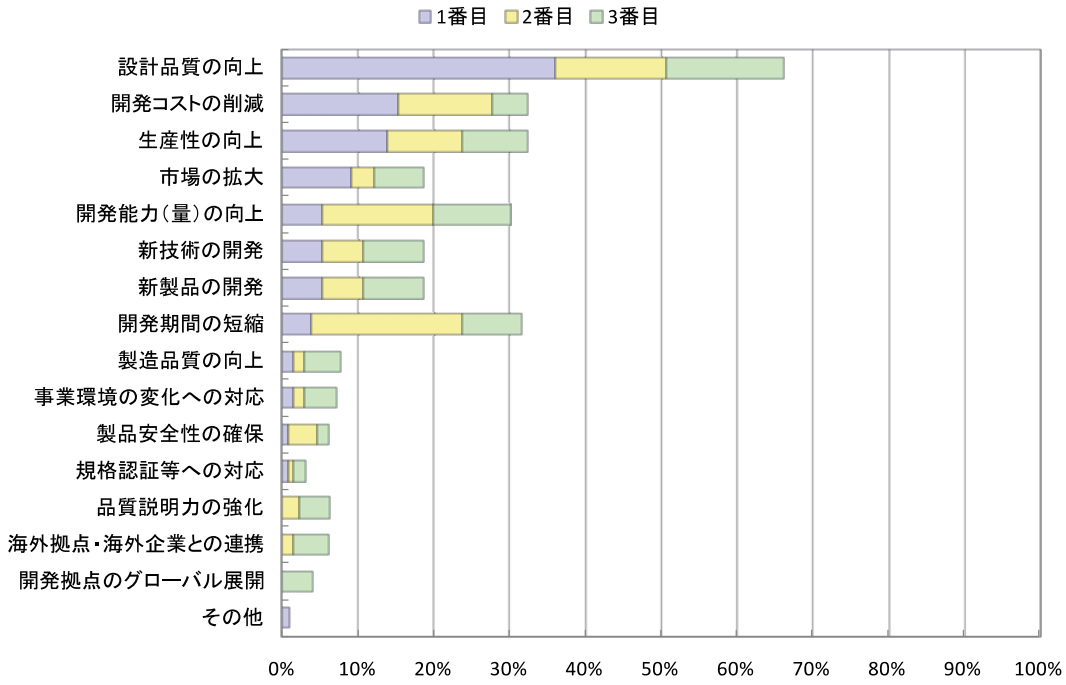


図 1.5　組込み系ソフトウェア開発の課題
（出展：“2012 年度「ソフトウェア産業の実態把握に関する調査」調査報告書”, 独立行政法人情報処理推進機構, 2013 [1)]）

組込みソフトウェア開発教育の現状と課題

組込みソフトウェア開発者の人材育成・活用に有用な「ものさし」（共通基準）として，組込みスキル標準（Embedded Technology Skill Standards：ETSS [7)]）が策定され，スキル基準が定義されている．スキル基準は組込ソフトウェア開発に必要な組込みスキルを以下の 3 つのスキルカテゴリに分類されている [8)]．

技術要素　組込みシステム自体に組み込まれ、システムの機能を実現する技術項目

開発要素　組込みシステムに各種技術要素を実装するために開発時に使用する技術項目

管理技術　組込みシステム開発を円滑かつ的確に進行させるために使用する技術項目

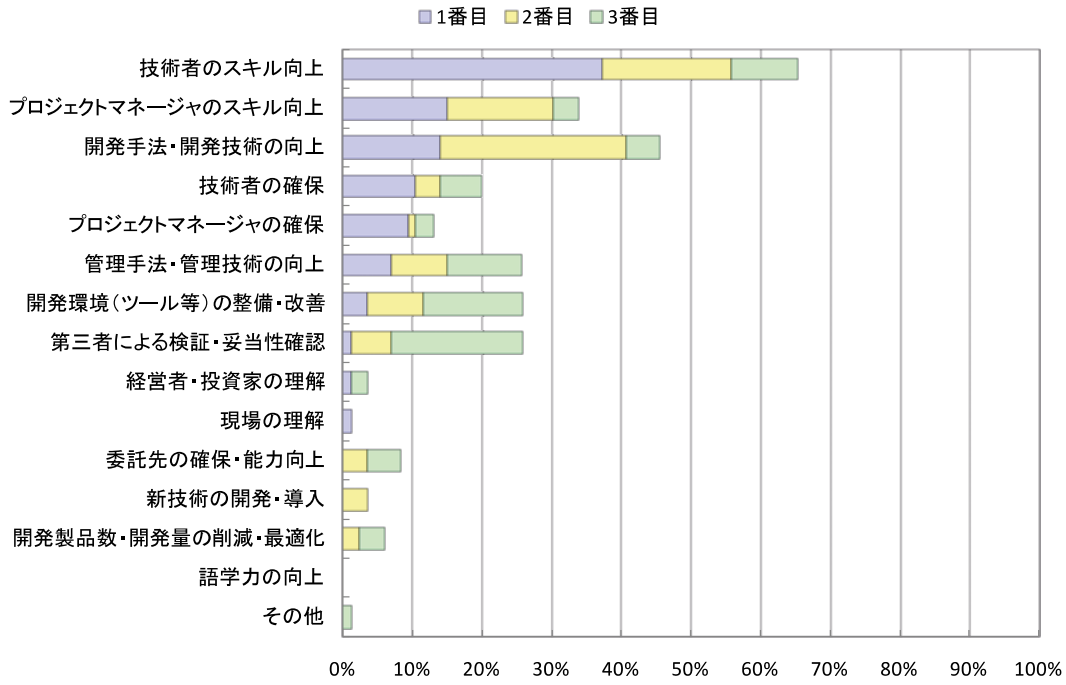


図 1.6　組込みソフトウェア開発の課題の解決策
（出展：“2012 年度「ソフトウェア産業の実態把握に関する調査」調査報告書”, 独立行政法人情報処理推進機構, 2013 [1)]）

上記のように，組込みソフトウェアの開発に必要な技術は多岐にわたる．本研究では，開発要素カテゴリのソフトウェア設計に関する教育を研究対象とする．

「組込みソフトウェアのエントリ人材教育に関する検討報告書」[9)] によると，未経験者向け教育カリキュラムの概要は，下記のようになっている．

組込みシステム技術　組込みソフトウェア技術者として必要な組込み基礎技術を習得する．

組込みプログラミング演習　組込みソフトウェア技術者として必要な C 言語を中心とするプログラミング技術を習得する．

組込みプロジェクト型演習　組込みシステム開発未経験者向け教育カリキュラムの総まとめとの位置づけとして，組込みソフトウェア開発に従事するために必要な技術や知識をプロジェクト型演習にて体験の上，習得する．

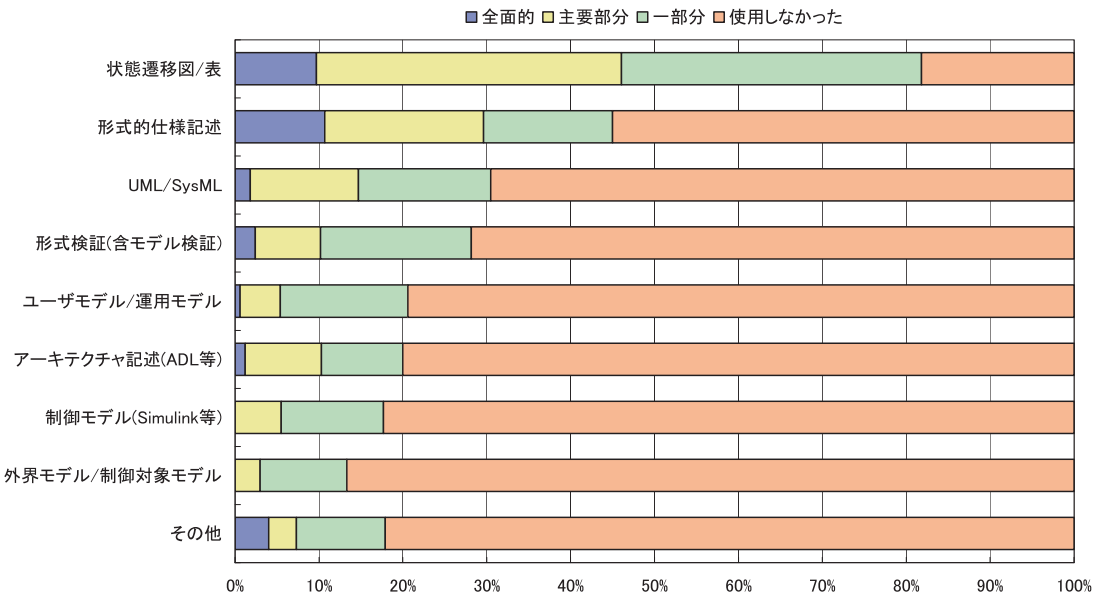


図 1.7　プロジェクトで利用した手法・技法
（出展：“2010 年版組込みソフトウェア産業実態調査報告書”, 経済産業省, 2010 [2)]）

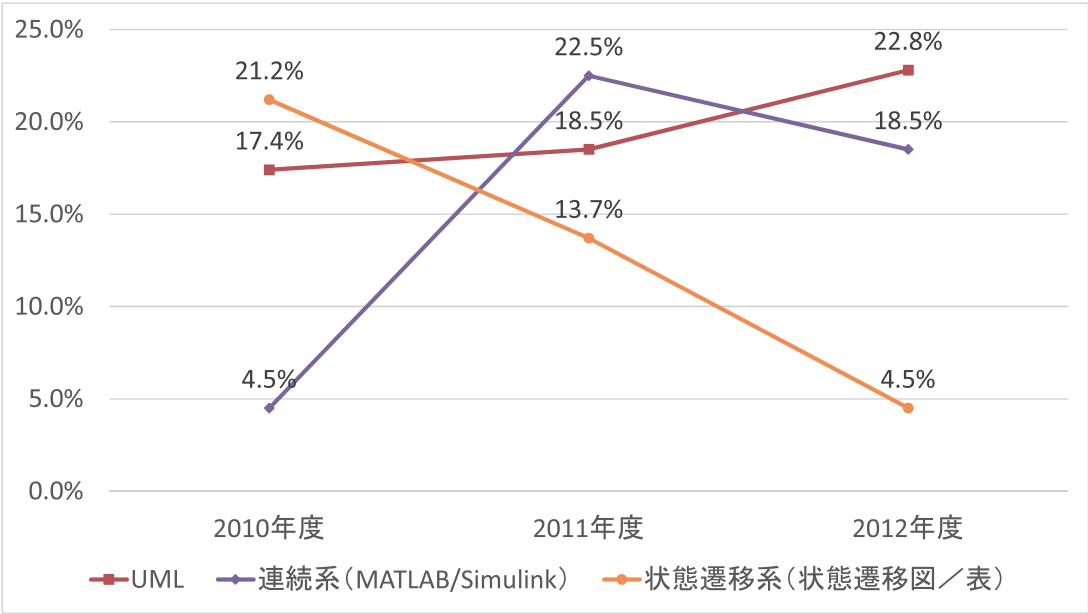


図 1.8　「自動コード生成」で使用したモデルベース言語等の比率

初学者向けのソフトウェア開発教育においては，プログラミング技術を始めとした実装教育に焦点が当てられた事例が多く，モデリング技術を用いた分析・設計といったより上流行程での開発スキル教育を目指したものは少ない．

ソフトウェアモデリング教育の現状と課題

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 [5]．1990 年代半ばからソフトウェアシステムの設計・開発にはオブジェクト指向の考え方を用いられることが多くなっており [10]，OOM のための標準化した仕様記述言語の 1 つである UML を用いた OOM 教育を行う試みは数多くなされている．

組込みソフトウェアにおいても同様に OOM 教育の必要性が高まり，モデリングを重視したロボットコンテスト [11][12] 等も実施されている．九州技術教育専門学校では，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 [*2][11] 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．

ET ロボコンを前提した，モデリング教育では，成果物であるモデルを動作として確認するまでに時間がかり，学習者は「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」への懸念を抱くこと，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た．

さらに，オブジェクト指向の学習曲線は図 1.9 に示すようなカーブを描き，学習の効果を得るのに時間がかかり，学習途中で諦めてしまうことがある [3]．図中の熟達レベルは，表 1.1 のような状態を指す [13]．従って，OOM の早期教育と学習に対するモチベーションを維持させる方法の確立が必要である．

大学等の高等教育機関におけるソフトウェアモデリング教育は，主にソフ

[*2] ET ロボコンは，オープン参加型のコンテストであり，2014 年は高校生から社会人までの 336 チームが参加している．

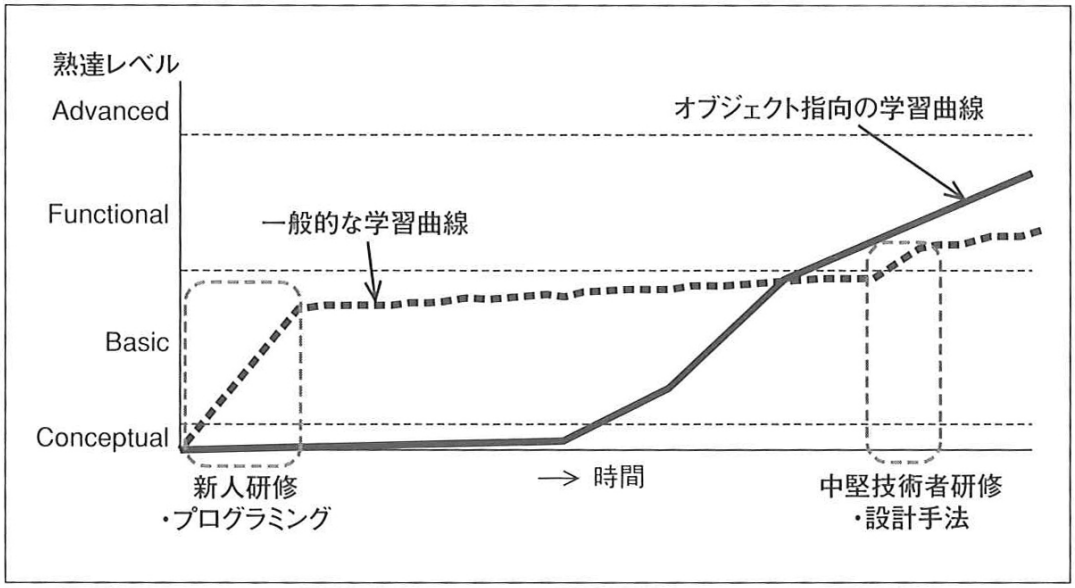


図 1.9　オブジェクト指向の学習曲線
（出展：“オブジェクト指向モデリングセルフレビューノート”, 荒井玲子, 2005 [3]）

トウェアエンジニアリング（SE）分野の 1 つとしてカリキュラムに導入されている．情報処理学会は，2007 年に情報専門学科におけるカリキュラム標準（J07）を作成した [14]．J07-SE は，大学などの高等教育機関の情報専門学科における SE 教育のためのカリキュラムモデルである．J07-SE のコンセプトの 1 つとして，SE において重要なのは技術の使い方よりも“ものの考え方”そのものであり，モデル化を習得することで「捉える力」や「考える力」,「表現する力」などを涵養することを指向している．

このように，モデリングの教育が重視されており，J07-SE でもモデリングに関わるカリキュラムが提示されている．J07-SE の年次進行の例を表 1.2 に示す．OOM を利用する科目は,「ソフトウェア設計」と「ソフトウェア開発実習」である．それぞれの科目の内容を下記に示す．

- ソフトウェア設計：ソフトウェア設計の概念を学んだ後，詳細設計の技法として，機能指向の設計，データ構造を中心とした設計，オブジェクト指向の設計，コンポーネント設計に関して学ぶ．そのうち，UML に関するものは 3 時間．

- ソフトウェア開発演習：エンタープライズソフトウェアまたは組込みソフトウェアに関するソフトウェア開発を行う．
  - エンタープライズソフトウェア：講師が提示する例題システムに対する要望を入力として，要求分析，アーキテクチャ設計，ソースコード作成/プログラミング，テストを実施する．グループでの演習を効率良く実施するために実習冒頭でグループのリーダ役を定め，リーダが中心となって，作業計画と作業目標を実習単位で明確にし，各実習終了前に計画の進捗度合いを確認し，計画に遅延が生じている場合は計画の再調整を実施する．
  - 組込みソフトウェア：講師が提示する例題システムに対する顧客の要望を入力として，組込みソフトウェアシステム開発における要求分析，アーキテクチャ設計，ソースコード作成/プログラミング，テストを実施する．グループでの演習を効率良く実施するために講義冒頭でグループのリーダ役を定め、リーダが中心となって，作業計画と作業目標を講義単位で明確にし，各講義終了前に計画の進捗度合いを確認し，計画に遅延が生じている場合は計画の再調整を実施する．

J07 に代表される既存のソフトウェアエンジニアリング教育では，プログラミング教育の後にモデリング教育を実施している．さらに，OOM に関する教育は，UML の文法と簡単なモデリング方法の教育の後，エンタープライズソフトウェアまたは組込みソフトウェアの比較的規模の大きなソフトウェアの一連の開発をウォータフォールプロセスで行っている事例が多い．プログラミング教育に比べ，モデリング教育に割り当てられている時間が短いとともに，モデリングの入門教育が不十分なまま，分析，設計，ソースコード作成，テストという一連の開発を行う実習を行うことが一般的である．このため，モデリングの重要性やモデリング方法の理解は，学習者本人の努力に委ねられている．

表 1.1　オブジェクト指向の熟達レベル

熟達レベル	状態
Advanced	エキスパートである．尋常ではない問題に対するアドバイスや解決を他人が求めてくるオーソリティである．このレベルの人は，しばしば技術分野に貢献する．このレベルは，熟練者として多数のプロジェクトに関わった人が長期間かけてやっと到達できる．
Functional	優れた業務知識と，有用なスキルを持っており，さらにこれらのスキルを細かに区分して説明する能力がある．このレベルに達すると，仕事を完全にやり遂げるためには，ほとんど指導者がいらない．たいていの場合に，生産的な商品開発はこのレベルの人がスタッフとして貢献する．このレベルは，訓練並びに多数のプロジェクトにおける直接的な経験の両方を通じて到達可能になる．
Basic	業務知識を持っており，何を知っていて何を知らないかを自分で認識している．このレベルの人は，小さい仕事にはアサインできるが，商品開発のクリティカルパスにはアサインできない．このレベルに達するには，一般的に導入訓練と単純な業務経験でよい．
Conceptual	入門的な概念の把握はしているが，業務知識を必要とするプロジェクトへの参加はできない．このレベルに達するには，フォーマルな訓練は不要で，本や雑誌を読めばよい．

1.2　本研究の位置付け

本研究では，OOM の教育方法を研究対象としている．本研究で対象とするモデルは，ソフトウェア開発プロセスにおける，分析・設計等に用いられ

表 1.2　J07-SE の年次進行の例

1 年前期	コンピュータとソフトウェアの基礎	モデリング に関する科目
1 年後期	確率・統計 離散数学 プログラミング基礎 プログラミング入門	
2 年前期	論理と計算理論 オペレーティングシステム基礎・データベース基礎 ソフトウェア構築	
2 年後期	ネットワーク基礎 モデル化と要求開発 ソフトウェア設計 プログラミング基礎実習	 ○ ○
3 年前期	ソフトウェアアーキテクチャ 検証と妥当性確認 ソフトウェアプロセスと品質 プログラミング応用実習	○
3 年夏期休暇	インターンシップ	
3 年後期	形式手法 開発マネジメント ヒューマンファクター ソフトウェア開発実習	 ○
4 年前期	工学基礎 卒業研究	
4 年後期	卒業研究	

る，UML を用いたオブジェクト指向モデルである．UML で描かれたモデルは，プログラミング技術を利用した実装工程を経てソフトウェア製品となる．従って，モデリング教育は，プログラミング教育との結びつきが非常に強い．プログラミングの入門教育においては，分岐や繰り返し処理などの文法の教育と合わせて，コンパイラを活用した実行確認を行い，プログラムの

動作を目で見て確認を行うことで，プログラミングの理解を深めることが一般的である．

一方，OOM 教育の場合は，教育対象となる言語は UML である．UML の使い方には，以下の 3 つのタイプがある [15]．

- スケッチとしての UML
- 設計図面としての UML
- プログラミング言語としての UML

この中で最も一般的なのは，スケッチとしての UML である [15]．スケッチとしての UML では，最終的な実装コードとの一貫性は求められておらず，前提条件としてメンバーに把握してほしいことや，プログラミングを始める前に視覚化しておきたい部分だけの説明に用いられる．対象的に，設計図面としての UML は完全性を重視される設計図面の役割は，プログラマがソースコードを作成するための詳細な設計を行うことである．

上記の実践的教育として，九州技術教育専門学校の UML 及び OOM 教育では，UML の文法教育として「スケッチとしての UML」の教育を行った後，ET ロボコンへの参加を通して「設計図面として UML」の教育を行っていた．J07 においても，「ソフトウェア設計」，「ソフトウェア開発演習」において各々，「スケッチとしての UML」，「設計図面としての UML」を活用している想定される．ただし，前節で述べたように，OOM 教育において，「理解性」，「必要性」，「モチベーション」に関する課題が生じた．

オブジェクト指向システムを構築するソフトウェア開発プロセスの代表的なものとして，Unified Process（UP）[16] がある．特に，UP に細かい工夫を施した Rational Unified Process （RUP）[17] は，広く採用されている [18]．RUP では，反復的開発を重視しており，正しい要件や設計をただ単に推測するのではなく，現実の構築とテストからフィードバックを得ることで，要件や設計を修正する機会が必要だとしている [18]．プログラミング教育と OOM 教育の大きな違いは，成果物として動作として確認できるかどうかである．OOM 教育においても，同様にモデルの挙動を確認し，よりよいモデルへと改善／修正していく工程が行えれば，モデリング教育の課題を解

決することができ，モデリング技術の習得に有益であると考えられる．

「プログラミングとしての UML」では，モデルからソースコードを自動生成する機能であるモデルコンパイラを利用することで，UML モデルが実行可能なコードに直接コンパイルされ，UML はソースコードになる．

従って，モデルからソースコードを自動生成するモデルコンパイラの活用により，プログラミング教育と同様の方法で OOM を教育することができると考えられる．

1.2.1 本教育のコンセプト

本研究では，上記のモデルの挙動を確認し，よりよいモデルへと改善／修正していく工程をスムーズに行う手段として産業界で実用化が進んでいるモデル駆動開発（Model Driven Development： MDD）をモデリング教育に活用することを提案する．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 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という利点がある．

1.2.2 モデル駆動開発

急速に複雑化している組込みソフトウェア開発において，品質の向上，開発時間の短縮，開発コストの軽減を目的として，MDD の活用されている[19]．MDD は，モデルからソースコードを自動生成することを狙った開発手法であるため，モデルがソフトウェア開発プロセスの中心となる．MDD の概要を図 1.10 に示す．

なお，本論文では，モデルからソースコードを自動生成する機能をモデルコンパイラ，モデルの描画機能及びモデルコンパイラを備えたツールを MDD ツールと呼ぶこととする．

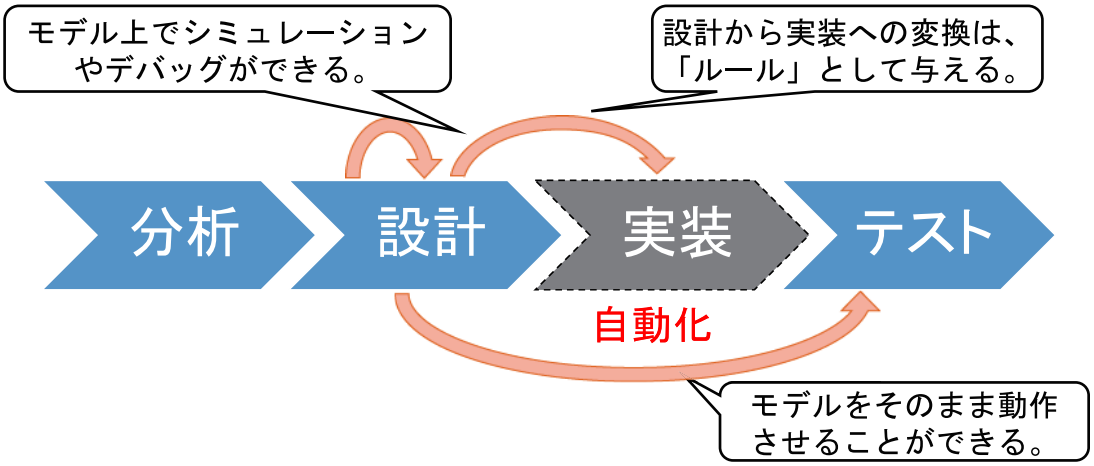


図 1.10　MDD の概要

UML を用いた MDD 手法の一つに OMG が提唱している MDA（Model Driven Architecture）がある [20]．MDA では，アプリケーションの表現に用いるプラットフォームに独立したモデル（Platform Independent Model：PIM）をツールで実行環境に依存したモデル（Platform Specific Model：PSM）に変換する．さらに，PSM をツールを用いてプラットフォーム用のコードを生成するという仕組みを定義している．PIM を作成するための言語として，Executable UML [21] がある．Executable UML は，UML の一部で表記法と共に実行モデルに対する意味論が定義されているため，モデルコンパイラを利用することで，動作確認を行うことができる．一方，UML を利用せず，ある開発対象領域に特化したモデリング言語であるドメイン特化モデリング言語（Domain-Specific Modeling Language：DSML）[4] を作り，それを利用して作成したモデルからソースコードを自動生成する手法もある．

MDD 手法をソフトウェア開発に用いるメリットは以下のようなものがあげられる．

- 開発対象領域の分析・設計部分に注力して開発できる．
- 開発早期に動作検証ができる．
- 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．

ソースコードは，モデルからモデルコンパイラを用いて自動生成されるため，開発の中心には分析・設計部分のモデリングになり，実装にとらわれることなく，分析・設計に時間とコストをかけることができる．また，Executable UML や DSML を用いて，厳密なモデルを作成することにより，モデル上で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．さらに，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．

Executable UML

Executable UML は, OMG によって FUML（Semantics Of A Foundational Subset For Executable UML Models）として，仕様の標準化が進んでいる [22]．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 1.11 に示す．動作可能なモデルとするために記述するアクション言語は，OMG により Action Language Foundational UML （ALF）として標準化されている [23]．

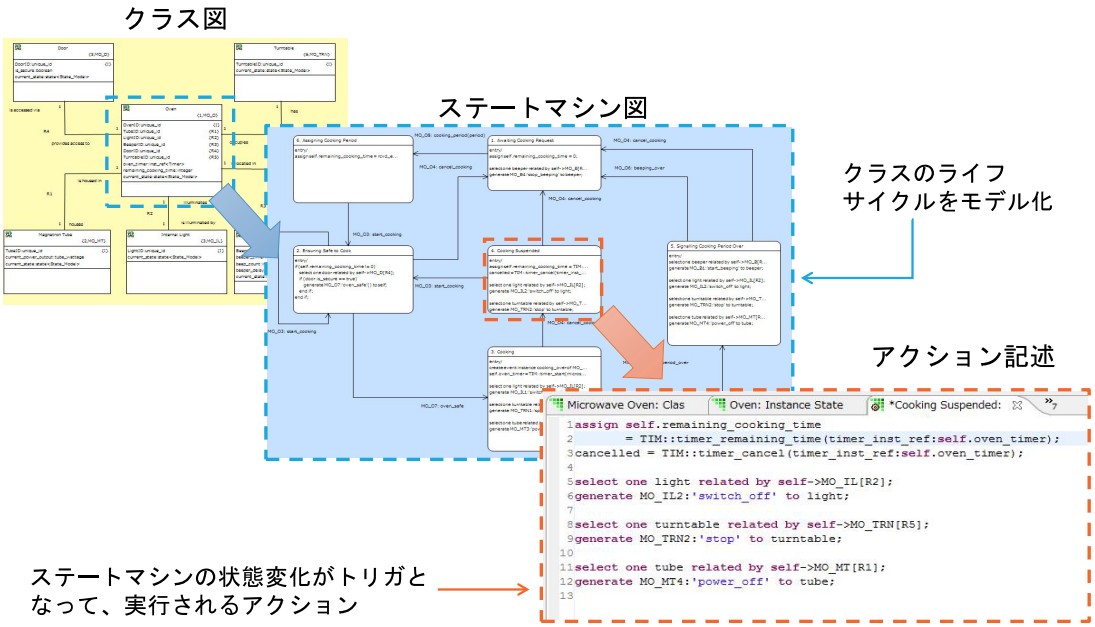


図 1.11　Executable UML の基本要素

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．
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．

DSML

MDD には，ある問題領域に特化した言語（Domain-Specific Language：DSL）及びコンパイラを開発し利用する方法がある．DSL の中でも特にモデリングを行う言語を DSML と呼ばれている．DSML は開発対象ドメインに特化したモデリング言語であり，DSML の主な目的は，以下の 2 つである[4)]．

1. 開発対象の問題領域に合わせたモデリング言語を作成することで抽象度を上げること
2. モデリング言語から開発対象に合わせたプログラミング言語を自動生成すること

図 1.12 にモデルの抽象度と生産性の関係を示す．DSML での開発では，UML を用いるよりも抽象度の高い言語での開発が可能となる．

1.3 研究の目的

本研究の目的は，モデリング技術を活用した組込みソフトウェアの開発において，OOM のスキル習得のための教育プログラム，教育方法及び教育環境を開発することである．具体的には，MDD を活用することで初学者におけるモデリング学習の敷居を下げること．さらに，教育用の実行可能モデリング言語及び MDD ツールの開発や MDD ツールの活用方法を工夫すること

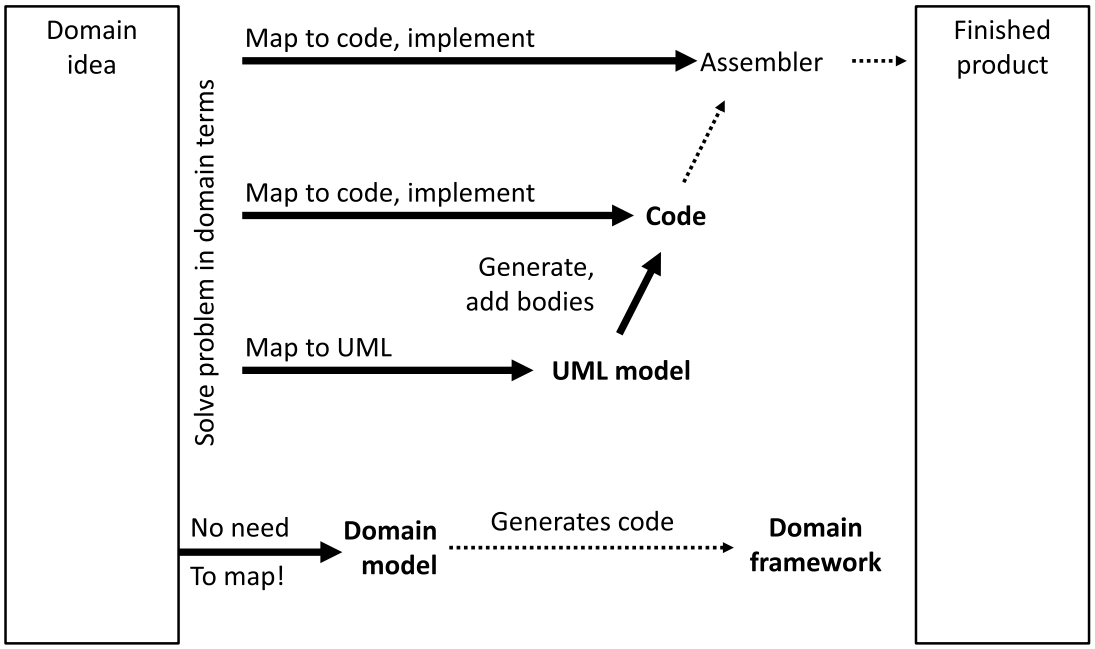


FIGURE 1.7 Bridging the abstraction gap of an idea in domain terms and its implemetation

図 1.12　モデルの抽象度と生産性
（出展：“Domain-Specific Modeling: Enabling Full Code Generation”, Kelly, Steven and Tolvanen, Juha-Pekka，2008 [4)]）

で，よりスムーズなモデリング技術の習得を目指す．

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考えられる．

- モデリングから動作確認までの時間を短縮でき，モデリング，動作確認というサイクルを繰り返すことにより，モデリング方法の理解を促進できる．
- ソースコードの自動生成機能により，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，学習早期に一連の開発体験を行うことができる．
- モデリングした結果をすぐに動作として確認できるため，学習者のモチベーションの向上につながる．

MDD では，モデリング，動作確認というサイクルを何度も繰り返すこと

ができる．さらに，開発対象以外のプログラムをモデル上から簡単に呼び出せる形式で提供することで，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向上にもつながる．

本研究では，組込みソフトウェア初学者に対して，MDD を活用したモデリング教育を行う教育プログラムの開発及び MDD ツールの効果的な活用方法の提案を行う．ただし，MDD を活用したモデリング教育の事例は少ないため，初めに，初学者にも MDD を活用した開発が行えるのか？ という懸念が生じる．次に，MDD の活用の中核であるモデルコンパイラ活用の有用性及び活用方法，特に自動生成を行うタイミングに関する疑問が生じる．さらに，MDD の実現には，実行可能モデリング言語の活用が不可欠であり，その選択が学習効果に大きな影響を与えると考えられる．

本論文では，上記を背景にして，モデリング教育への MDD の活用の有用性及び活用方法に関する下記の研究課題を明らかにする．

研究課題 1　MDD の活用

(1) **初学者の** MDD **活用の可否：**　初学者に対して MDD を活用した OOM 教育が行えるのか？

(2) **モデルコンパイラの有用性：**　モデルコンパイラの利用することで，学習者が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，モデリングスキルの向上につながるのか？

研究課題 2　MDD の活用方法

(1) **モデルコンパイラの活用タイミング：**　モデルコンパイラの活用タイミングをどのようにしたらよいのか？

(2) **実行可能モデリング言語の選択：**　どのような、実行可能モデリング言語を利用した教育が効果的なのか？

1.3.1 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 [24]．

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．

1.3.2 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．これを踏まえ本研究では，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．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 [25] に定義された Syntactic, Semantic, Pragmatic の 3 種類を用いる．各々，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．

組込みソフトウェア開発でよく使われる図を表 1.3 に示す．本研究で取扱

う UML の図は，構造モデルと振舞いモデルのそれぞれ 1 つずつとする．構造モデルは，表 1.3 より，組込みソフトウェア開発でよく使われているクラス図とし，振舞いモデルは，よく使われる図 2 つのうち，UML の利用以外でも状態遷移図のみの利用としてよく利用されている（図 1.7）ステートマシン図とした．

表 1.3　よく使われる主な UML の図

図の種類	UML の図	利用頻度
機能	ユースケース図	○
構造	クラス図	○
	コンポートネント図	×
	配置図	×
振舞い	ステートマシン図	○
	シーケンス図	○
	コミュニケーション図	△

各図の入門レベルとして想定される学習目標を下記の通りとした．入門レベルのため，詳細なモデルを描く上で必要となる項目（クラス図：多重度，属性，操作，可視性など，ステートマシン図：entry/do/exit アクション，スーパー状態，並行状態など）は，対象としない．

1. クラス図

   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　第三者に責務がわかるクラス名をつけることができる．

   関連　クラス間の関係を関連として記述できる．

2. ステートマシン図

   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つけることができる．

   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．

   振る舞い　各状態の振る舞いをアクションとして定義することがで

きる．

1.4 本論文の構成

本論文の構成は次の通りである．第 2 章では，モデリング教育，モデリング学習環境，MDD と教育に関する関連研究と本研究の位置づけについて述べる．第 3 章では，MDD を活用したモデリング教育プログラムの内容及び評価について述べ，初学者でも MDD を用いた開発を行うことの有用性を明らかにする．第 4 章では，開発した教育用 MDD ツールの説明及び MDD ツールを活用した教育及び実験内容，結果の説明を行う．第 5 章では，第 1 章に定義した研究の目的に関する考察及び今後の展開について説明する．

第 2 章　関連研究

本章では，オブジェクト指向モデリング（Object-Oriented Modeling：OOM）に関する研究及びモデル駆動開発（Model Driven Development：MDD）と教育に関する研究と本研究の関係について述べる．

2.1　オブジェクト指向モデリング教育に関する研究

Paige らは，モデリング教育において 13 項目の行ってはいけないことを挙げている [26]．その中で，本研究と関連のあるものには，次のようなものがある．

1. モデリング言語の仕様のみを教えること
2. 構文に注力し意味論の教育を疎かにすること
3. 演習に学習者に馴染みのない本格的な題材を選び，モチベーションを下げること
4. ツールを使うための学習は簡単であると考えること
5. モデリングの主要な目的はコード生成だと思わせること

モデリング言語は，本研究で対象とする UML をはじめ，SysML，フローチャート，拡張エンタープライズ・モデリング言語（Extended Enterprise Modeling Language：EEML），E-R 図（Entity Relationship Diagram：ERD）など様々な種類がある．上記の 1. 2. では，多くのモデリング言語の仕様やその構文の教育に注力した教育のみではなく，モデリング方法や意味論の教育の必要性が述べられている．本研究では，モデリング方法の理解を促進する教育にモデルからソースコードの自動生成を行える MDD ツールを活用することを目的としている．また，上記 3. に関しては，組込みシステムの車両型ロボットの開発を演習題材とし，学習者が楽しみながら演習を行えるように工夫した．上記 4. のツールに関しては，2 種類のモデリングツールを用い

た．1 つ目が，教育での利用事例 [27][28][29][30] が多く，高校生や大学初級学年での利用もある BridgePoint [*1] を用いた．2 つ目は DSL プラットフォーム “clooca” [32] を用い教育用の DSML を作成することで，使いやすいツールになるように工夫した．上記 5. のコード生成に関しては，本研究ではソースコードの自動生成を行うことは，モデリングスキルの向上につなげる手段であると考えており，モデリングの方法と成果物であるモデルの品質の向上に重点を置いた講義を行った．

これまでのツールを活用した OOM 教育事例では，初学者が使いやすいツールに工夫されたものや UML の誤り検知等に焦点を当てたものが多い．minimUML は，作成できる図をクラス図，扱う図の要素をクラス，インターフェース，関連，集約，実現のみに限定することで，初学者が扱いやすいツールを目指している．UMLint [33] は，ユースケース図とクラス図に関する一般的な誤りを提示するシステムである．誤りの例としては，「多重度の付加に関する誤り」や「継承において，継承元と継承先が逆になっている」，「属性や操作が不適切」などがある．UMLint を基にしたツールである UMLGrader [34][35] は，ある限定された問題に対するサンプルモデルを用意し，そのモデルと受講生が描いたモデルを比較することで，より詳細な欠陥を提示するシステムである．類似する研究として，Auxepaules らはクラス図の解答例と受講生のモデルを比較し，相違点を表示させる web ベースの e-learning システムを提案している [36]．Schramm らはアクティビティ図及びクラス図を対象として，同様に相違点を表示するシステムを提案している [37]．一方，完全なサンプルモデルではなく，想定されるモデルのルールのみを定義し，自動的なモデルチェックを行う方法も提案されている [38]．これらの研究においては，正解となるモデルがあることを前提としており，事前に正解を用意しておく必要があること，正解が一意に決まらないモデリング教育において受講生が正解を求める傾向に陥ってしまう恐れがあることなどの課題がある．

[*1] 講座で利用した時点では，Mentor Graphics の商品であったが，2014 年 11 月からオープンソースとして xtUML.org [31] のサイトで無償提供されている．

正解のないモデルの教育の有効な手段として，レビューがある．生徒間や教員のレビューを支援するツールとして ClassCompass [39] が提案されているが，個人学習ができない，レビューには教員の負担が大きいなどの課題がある．

また，モデルの正解に頼らない方法として，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，クラス図とオブジェクト図の関係や多重度等の学習を支援するシステムの提案も行われている [40]．また，UML のクラス図―オブジェクト図間の矛盾の指摘（一貫性診断）だけでなく，曖昧である箇所の指摘（明瞭性診断）を行うことが特徴としたシステムの開発も行われている [41]．

ただし，これは静的モデルのみを対象としたものであり，動的モデルを対象とした研究事例は少ない．

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の研究が行われている．StudentUML [42][43] は，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留まっている．

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つけるための学習の支援を行うことである．

2.2 モデル駆動開発と教育に関する研究

モデル駆動開発（Model Driven Development： MDD）と教育に関する研究では，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 [44]．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 [45]．

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 [30]．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伴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 [29]．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 [46]．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 ツールの活用を行なっており，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 [27][28]．文献 [47] 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．

第 3 章　モデリング教育プログラム評価実験

本プログラムの目的は，上記のモデリング教育に関する課題を軽減し，学習者が短時間にオブジェクト指向モデリングのノウハウやモデルを中心に置いた開発方法を習得できるようにすることである．

本章では，研究課題 1-(1)「初学者の MDD 活用の可否」及び研究課題 1-(2)「モデルコンパイラの有用性」の検討に寄与することを目的とした実験について述べる．さらに，研究課題 2-(2)「実行可能モデリング言語の選択」に関する検討材料として，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．

3.1　教育プログラムの目的

モデリング教育の課題の軽減のため，開発する教育プログラムの要件を下記のように設定した．

- 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）
- モチベーションを向上させられること（モチベーション）

3.2 教育プログラムのコンセプト

教育プログラムの要件を踏まえて，「スパイラル的教育」と「教育早期における PBL の活用」の 2 つを教育コンセプトとして，MDD を活用した教育プログラムの開発を行った．本プログラムでは，MDD の実現方法の一つである Executable UML[21]，ツールとして BridgePoint（Mentor Graphics）を利用した．

3.2.1 スパイラル的教育

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う．教育の概略を図 3.1 に示す．

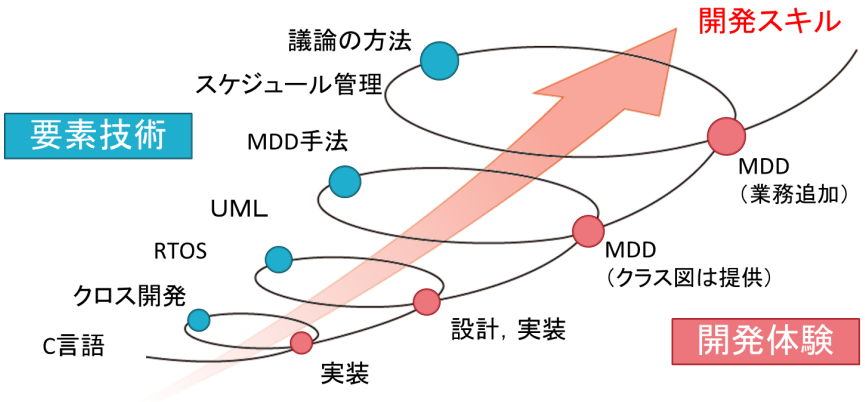


図 3.1 スパイラル的教育の概略

具体的には，座学による要素技術の講義と簡単なプログラムやモデルを作成する基礎演習，業務システムを開発する総合演習をセットで実施する．総合演習では，各ステップで同一の演習を開発方法を変えて開発をした後，業務が追加されたシステムの開発を行わせるにより，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ように工夫した．この方法を用いることにより，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．ただし，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ようにした．

3.2.2 教育早期における PBL の活用

PBL 手法では，総合的な技術力が必要なため，それまでに習得した知識や技術の定着が期待できる．また，PBL のもう一つの効果として，学習者自身が不足している知識や技術に気づくことができる点がある．その為，初級学年における PBL の経験は，次年度以降の講義や演習への取り組みに対する意欲の向上が期待でき，在学期間における教育効果の向上が見込める．ただし，PBL では，技術力だけではなく，プロジェクトマネジメント技術やコミュニケーション技術など，プロジェクトの進め方に関する知識も必要となる．先行実践においては，主に大学 3，4 年生や大学院生などの上級学年での利用が行われており [48)49)50)]，初学者に取り入れるのは難しい手法である．

本プログラムでは，ソフトウェア開発の経験がない初学者に対する PBL とするため，下記の 2 つを目的としたガイド付き短期 PBL とした．

(1) 開発手順を考える手助けをすること
(2) グループ内でのコミュニケーションの取り方やプロジェクトの進め方を自然と身に付けられるようにすること

上記（1）に関しては，総合演習課題に対する標準のシステムの開発プロセスを想定し，そのマイルストーンに合わせた演習問題を提供した．（2）に関しては，タイムボックス制を基本とした活動が行えるようにプロジェクトファシリテーションツールを活用した．

初学者向けの PBL であるため，プロジェクトファシリテーションツールを用いたガイド付き短期 PBL を実施することで，スムーズな PBL が行えるように工夫した．

3.3 教育内容

3.3.1 教育プログラムの構成

作成したプログラムは，基礎編，応用編，PBL 編の 3 部である．各教材の概要を以下に示す．

1. 基礎編
   組込みシステムの概要，組込みプログラミング，リアルタイム OS 等組込みシステム開発を行う上で基礎知識を習得する．
2. 応用編
   ソフトウェア品質，UML，オブジェクト指向モデリング，MDD など高品質な組込みソフトウェアを開発する上で必要となる知識・技術を習得する．
3. PBL 編
   チームによるソフトウェア開発演習を通して，基礎編，応用編で学んだ知識の定着を図り，MDD を実践できる技術を習得する．

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，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 3.2 に示す．

スパイラル的な教育を実現するため，基礎編の基礎演習 A において単純なライントレースなどのプログラムを作成させ，応用編の基礎編演習 B ではそのプログラムをリバースモデリング及びリファクタリングさせることで，モデルとソースコードの関係を教育する．さらに，総合演習において，より複雑な業務システムを開発するプログラムやモデルを作成させることで，各ステップで習得した知識を利用するスキルを身に付けさせる．基礎編では

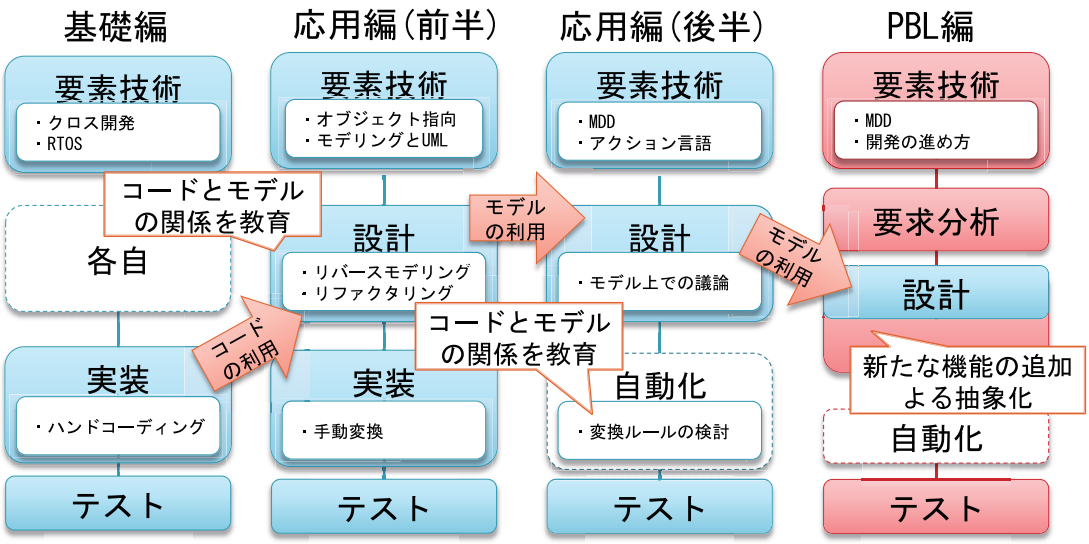
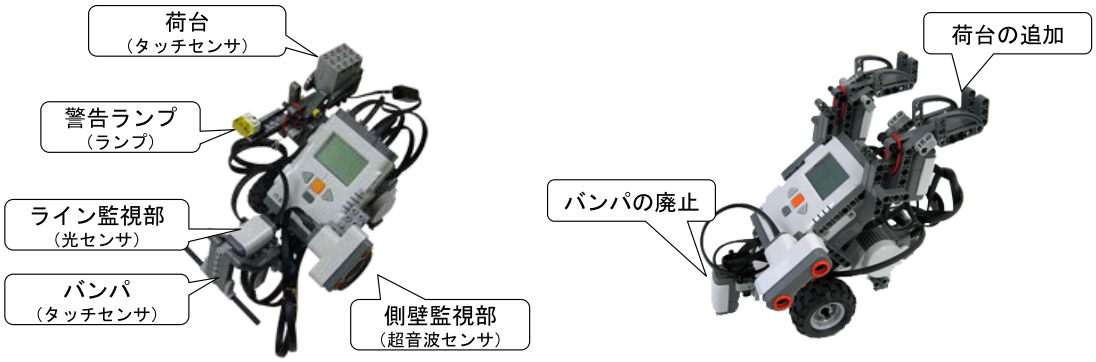


図 3.2　教材間の関係

実装のみ，応用編の前半では設計から実装までを手動変換で行い，後半では MDD ツールを利用した自動変換を行うという形で，徐々に抽象度をあげて開発させる．その後の 2 週間の PBL では，業務の追加に伴う仕様変更に対応したシステムの開発を MDD で開発させることで，MDD での開発手法の理解及びモデリングスキルの向上を図る．

基礎編及び応用編で実施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 3.3）である．総合演習 A,B 用 PBL 用を各々図 3.3(1),(2) に示す．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．演習のコースを図 3.4 に示す．



（1）総合演習A, B用　　（2）PBL用

図 3.3　自動搬送ロボット

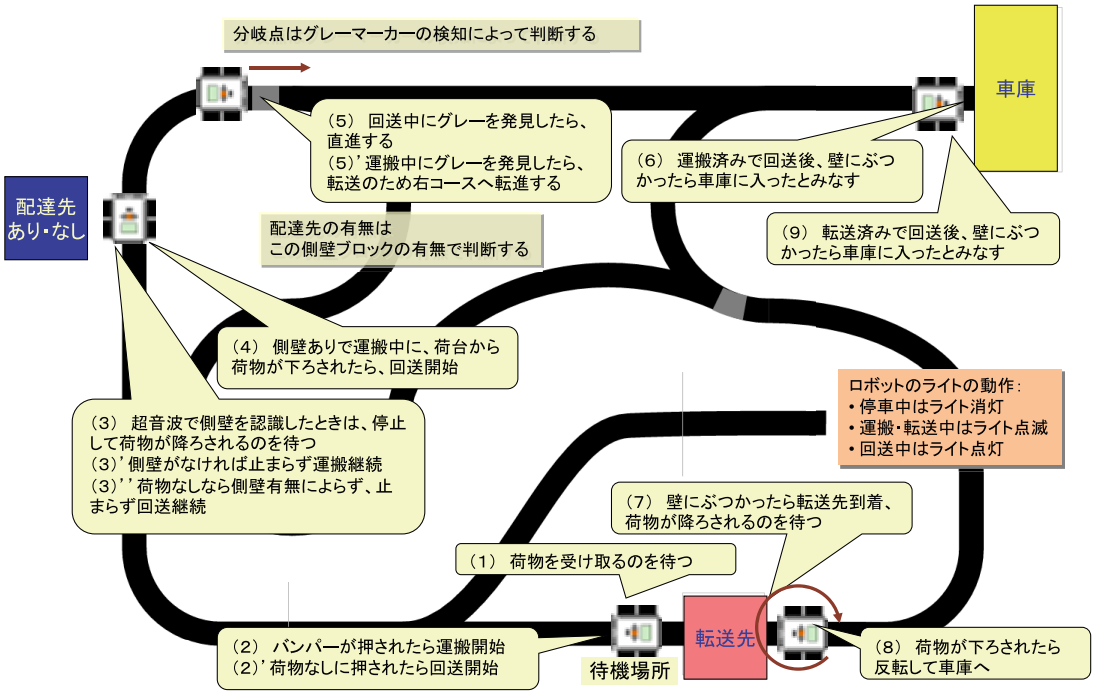


図 3.4　基礎編・応用編の総合演習課題のコース

3.3.2　基礎編

組込みソフトウェア開発の基礎知識の習得を目的としており，クロス開発やリアルタイム OS 等の組込みソフト開発独特の開発方法をロボットのモー

タやセンサを動作させる演習を通して学習する．基礎編の概要図を図 3.5 に示す．

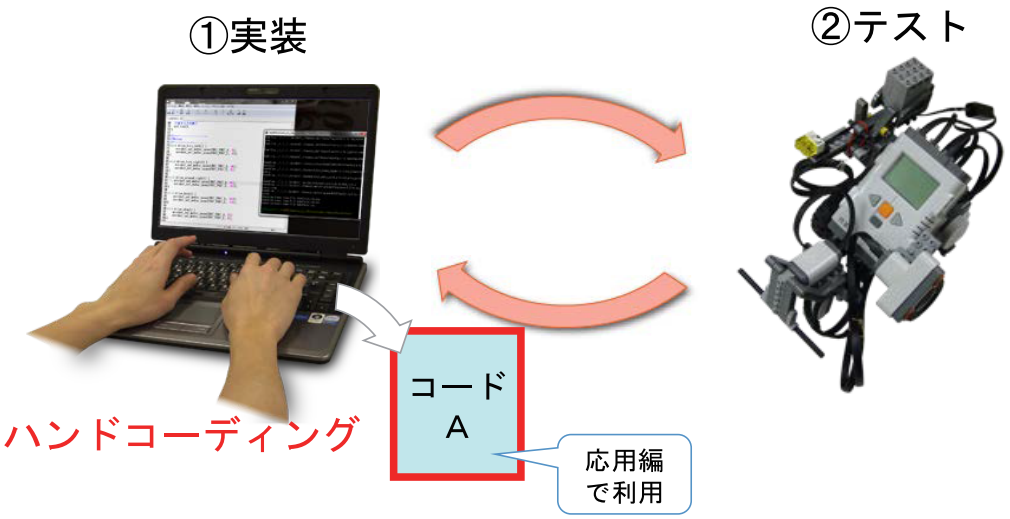


図 3.5　基礎編の概要

総合演習では，設計段階は省略し，実装のみを行うことで，基礎編で学んだセンサやモータを動かす方法や RTOS の使い方のスキルを獲得することを目的とする．具体的には，基礎演習で学んだ，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，ハンドコーディングで実装する．基礎編の教育項目を表 3.1 に示す．

表 3.1　基礎編の教育項目

要素技術 A	組込みシステムの基礎，RTOS，クロス開発
基礎演習 A	センサ，モータを動かすプログラム，ライントレース
総合演習 A	一連の業務をハンドコーディングで実装

3.3.3　応用編

MDD での開発に必要な要素技術であるモデリング手法やモデルからのソースコード生成技法，MDD ツールの使用方法等をロボットを開発対象として学習する．

総合演習では，設計，実装をトップダウンで開発する方法を習得することを目的とする．基礎編では，省略していた設計段階からはじめ，設計に合わせた実装を行うことで，質の高いソフトウェアを作る方法を習得する．応用編前半と後半の 2 回演習を行う．応用編前半と後半の概要を図 3.6, 図 3.7 に示す．

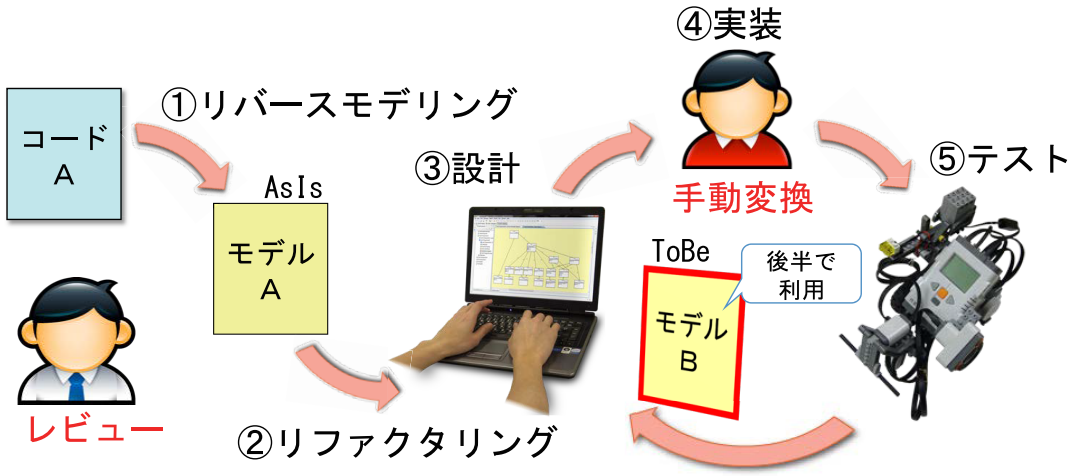


図 3.6　応用編前半の概要

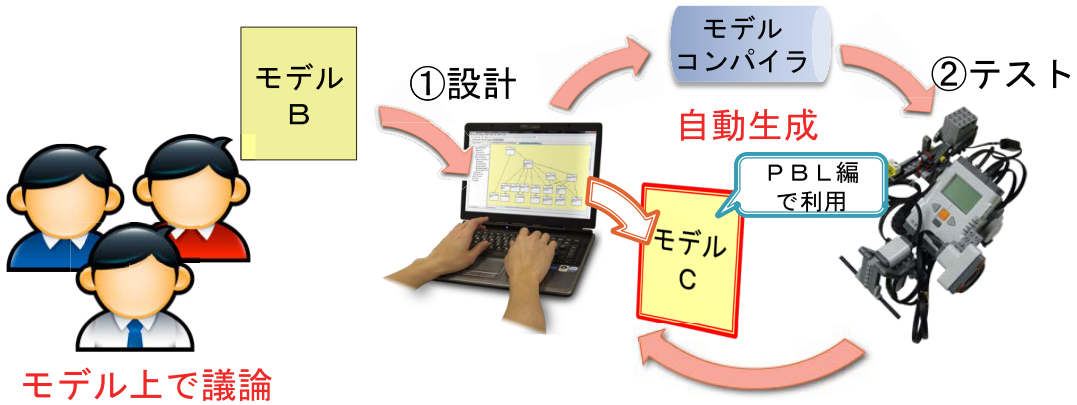


図 3.7　応用編後半の概要

前半では，応用編の基礎演習で作成したモデル図とプログラム，基礎編の総合演習で作成したプログラムをもとに，総合演習のモデル図を作成し，モデルからソースコードに手動変換する．前半では，一連の開発体験を経験することで，各工程での開発内容を習得することができる．後半では，

MDD ツールを用いて，ソース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．応用編の教育項目を表 3.2 に示す．

表 3.2　応用編の教育項目

要素技術 B	UML，MDD 手法，モデリング手法， モデルからソースコードの生成技法
基礎演習 B	基礎演習 A のリバースモデリング， 変換ルールに基づいてモデルからソースコードの手動変換
総合演習 B	総合演習 A と同一業務を MDD 手法で開発

総合演習課題を MDD で開発する応用編では，初学者がスムーズにモデリングに取り組めるように，いきなりクラス図を描かせるのではなく，クラス図を提供し，自動搬送システムの業務を実現する部分のみ開発してもらうこととした．提供したクラス図を図 3.8 に示す．なお，本クラス図は BridgePoint ツールで描いたもので，アクション記述を書く際に必要な項目が付加されている [*1]．

クラス図の下位層に位置する自動搬送ロボットを構成する，バンパ（Bumper）や荷台（Carrier）などのユニット層のクラスは，実行可能な状態として配布をする．主な課題内容は，自動搬送システムの全体的な制御を行うクラス（AutoTransporter），ラインに沿って走行するクラス（LineTracer）の 2 つの責務に着目し，各クラスの振舞いを表すステートマシン図を描くことである．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．

[*1] クラス名の右下にある英数字（例：TP_ATP）は，キーレターと呼ばれ，クラスへアクセスする記述を書く際に利用する．クラス間の関連に付加されている英数字（例：R1）は，関連指定子と呼ばれ，関連にアクセスする記述を書く際に利用する．

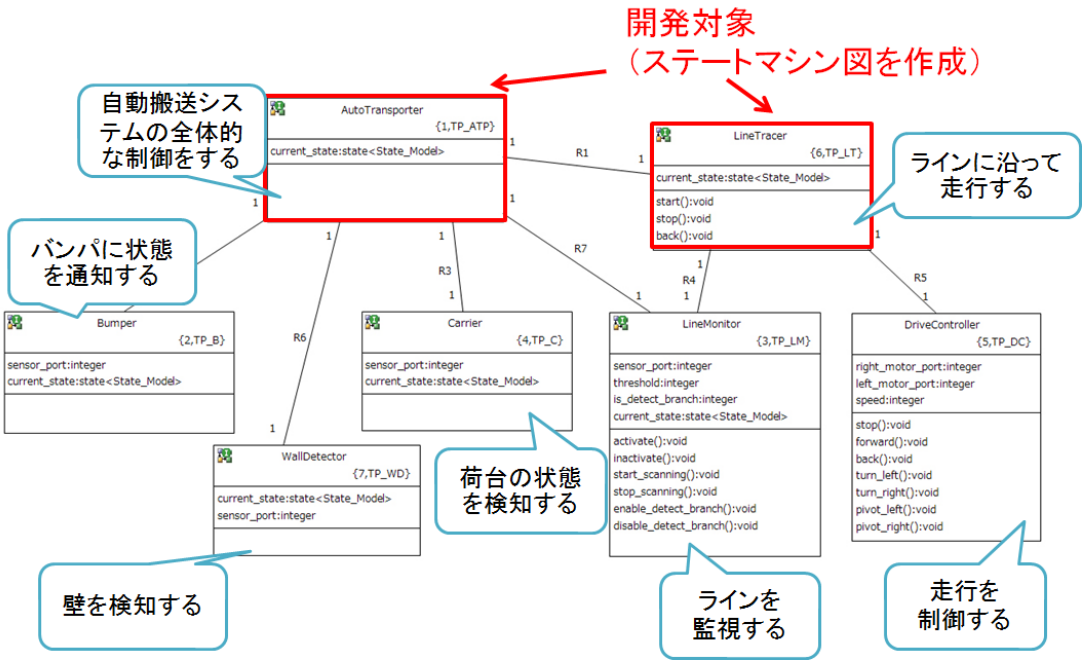


図 3.8　自動搬送システムのクラス図

3.3.4　PBL 編

PBL 編では，応用編で開発した業務システムの仕様変更に対応したシステムを MDD で開発させることで，MDD での開発手法の理解及びモデリングスキルの向上を図る．PBL 編の課題内容を以下に述べる．

課題内容

総合演習 B に業務を追加するソフトウェア開発案件に対して，MDD を用いてグループでの開発を行う．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．PBL で用いるロボットは前述した図 3.3（2）にである．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され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 [*2] 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時点で，提供をする．また，基地局は，PBL 期間中に実行モジュールを提供した．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 3.9 に示す.

図 3.9 自動搬送ロボットと基地局の通信の様子

[*2] 通信デバイスを動作させるモデル

3.3.5 PBL の進め方

14 日間の PBL 期間の全体の流れを表 3.3 に示す．最初の 2 日間をキックオフ，最後の 2 日間を成果報告及び準備にあて，実質の PBL 期間は 10 日間である．

表 3.3 PBL の流れ

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	通信用ブリッジの提供
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

3.3.6 プロジェクトファシリテーションツール

本 PBL では，初学者が時間管理及びチーム内での役割分担の方法，PDCA の回し方等を身に付け，円滑なチーム開発を行えるようになること目的とし，タイムボックス制を基本とした活動を行うように指導した．プロジェクトファシリテーションツール [51] として，本 PBL 用にアレンジした下記の 2 つを利用した．

タイムボックスかんばん

「タイムボックスかんばん（図 3.10）」と名付けた表の作成を義務付けた．各チーム 1 日 1 枚の模造紙に 1 日の目標，各タイムスロット（50 分）の目標や ToDo，うまくいったこと（OK），ダメだったところ（NG）を記載してチーム全員が見えるところに貼る．また，目標を緑，ToDo を黄色，OK をピンク，NG を水色の付箋紙で貼る．これにより，チームの進捗状況が一目でわかるため，チーム内でのコミュニケ―ションを促進することができる．さらに，貼られた付箋紙の色の違いにより，直観的にプロジェクトがうまくいっているのか？ 危ないのか？ を認識することが可能となる．また，指導教員も，別途日報の提出やヒアリングを実施せずに，各チームの進捗状況を確認できる．

図 3.10　タイムボックスかんばん

KPTT 表

1 日の終了時に実施する夕会のふりかえりを支援するツールとして，KPTT 表（図 3.11）を作成させた．KPTT 表は，次の手順で作成する．

1. テーマを決める.
   ふりかえりを行うテーマを決めて表に書く．特に思いつかなければ，「プロジェクトを円滑に進めるために」とする.
2. Keep を出す.
   付箋紙に「良かったこと」や「今やっていてこれからも続けたいこと」などを「〜する」という動詞形式で具体的に付箋紙に書き，表に貼る.
3. Problem を出す.
   「問題だと認識している」ことを付箋紙に書き，表に貼る.
4. Try を出す.
   「試してみたいこと」や「改善策」を付箋紙に書き，表に貼る.
5. ToDo を出す.
   Try の中から次の日に試してみたいことを選び，ToDo の覧に移す.

上記のように，Try に出したものの中から，チーム内でやれそうだと思う項目のみを ToDo に移すという過程を設けたことで，発言の敷居を下げる工夫をした.

テーマ：	
Keep ・良かったこと ・続けたいこと	ToDo ・試してみること
Problem ・問題だと認識していること	Try ・試してみたいこと ・改善策

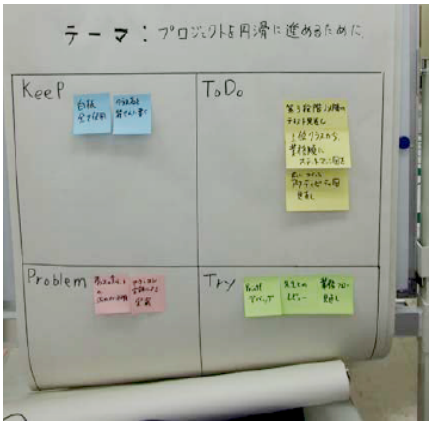


図 3.11　KPTT 表

3.3.7　PBL の 1 日の流れ

下記を目的として，1 日の始まりには朝会 [52]，終わりには夕会を行うこと義務付けた.

朝会

1. 朝会の目的
   - チーム全体が必要な情報を短い時間で共有すること.
   - 昨日やったことをふりかえり，今日やることを各自認識して積極的に役割を担えるようにすること.
   - 朝，気持ちよくスタートを切ること.
2. 夕会の目的
   - 一日の目標と作業内容のふりかえりを行い，開発の停滞を防ぐとこ.
   - 効果的な活動や改善策を共有し実行に移せるようにすること.
   - 次の日の活動をスムーズに進めること.

朝会，夕会のグランドルールとして「定時刻に行う」,「15 分以内に行う，全員が参加し」，全員が発言する」,「立ったまま行う」の 4 つを設定した.

PBL 期間の 1 日の流れは，次のようになる.

1. 朝会：15 分以内
   - 昨日やったこと，今日やること，問題点の報告
   - 一日の目標設定
   - タイムボックスかんばんの作成
2. 各タイムスロット（1 日あたり 6 タイムスロット）：50 分
   - 目標設定：5 分
   - 実行：40 分
   - ふりかえり：5 分
3. 夕会：15 分以内
   - 一日のふりかえり
   - KPTT 表の作成
4. 残業

   チーム内での合意のもと残業を行ってもよいこととした．なお，残業を行う際の注意点としては，タイムボックスかんばんに枠を追加し，

通常の開発時間と同様に 1 タイムスロット単位で実施することとし，時間管理に対する意識づけをさせた．

夕会の様子を図 3.12 に示す．

図 3.12　夕会の様子

3.4　実証講座

3.4.1　実証講座の内容

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．主教育対象への講座として，専門学校 1 年生を対象に実施した．受講者は，入学後半年間 C 言語及び IT スキル標準 [53] レベル 2 を目指すカリキュラムを受講した学生である．また，受講後の 1 年生のスキルレベルと比較評価することを目的として，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，プログラムの内容を短縮した講座を実施した．2 つの講座をそれぞれ，xUML 講座 1，xUML 講座 2 と呼ぶこととする．xUML 講座 1，xUML 講座 2 の概要を表 3.4，表 3.5 に示す．なお，xUML 講座 2 の受講生は，本プログラムの教育プログラムのうち，MDD 手法に関する部分のみス

キルを持っていなかった.

表 3.4　xUML 講座 1 の概要

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

3.4.2　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.

表 3.5　xUML 講座 2 の概要

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

理解性

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．テストの結果を表 3.6 に示す．

表 3.6　xUML 講座の確認テスト正答率

	xUML 講座 1	xUML 講座 2
UML の知識	65%	79%
MDD 手法	61%	80%

UML の知識に関するテストの正答率は，xUML 講座 1 の受講生が平均 65%，xUML 講座 2 の受講生が平均 75% であるが，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，xUML 講座 1 で平均 75% の正答率があり，xUML 講座 2 の受講者と同程度の正答率が得られた．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，xUML 講座 1 の受講者は，平均 61%，xUML 講座 2 の受講者は平均が 80% と開きが大きく，UML の前提知識がある場合には，本プログラムにより習得できるが，今回の講座のように集中講義で UML と MDD 手法を同時に習得させるのは難しいことが分かった．

また，xUML 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く，原因としては，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考えられる．これらの課題は，集中講義ではなく，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で実施することで，対策できると考える．

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 3.7 に示す．

表 3.7　xUML 講座応用編総合演習課題達成率

マイルストーン	xUML 講座 1	xUML 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 %

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ず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，つまずいた原因は，モデリングではなく，マーカ検知のパラメータ調整の部分であり，ステートマシン図を用いてシステムの状態の変化を記述することができるようになったといえる．

図 3.13 に応用編のステートマシン図の例を示す．「状態名」に関しては，状態名が未記入及び同じ名称の状態を複数記述しているものがあったが，不適切な状態名は少なく，状態名の検討を行なって記載していたといえる．

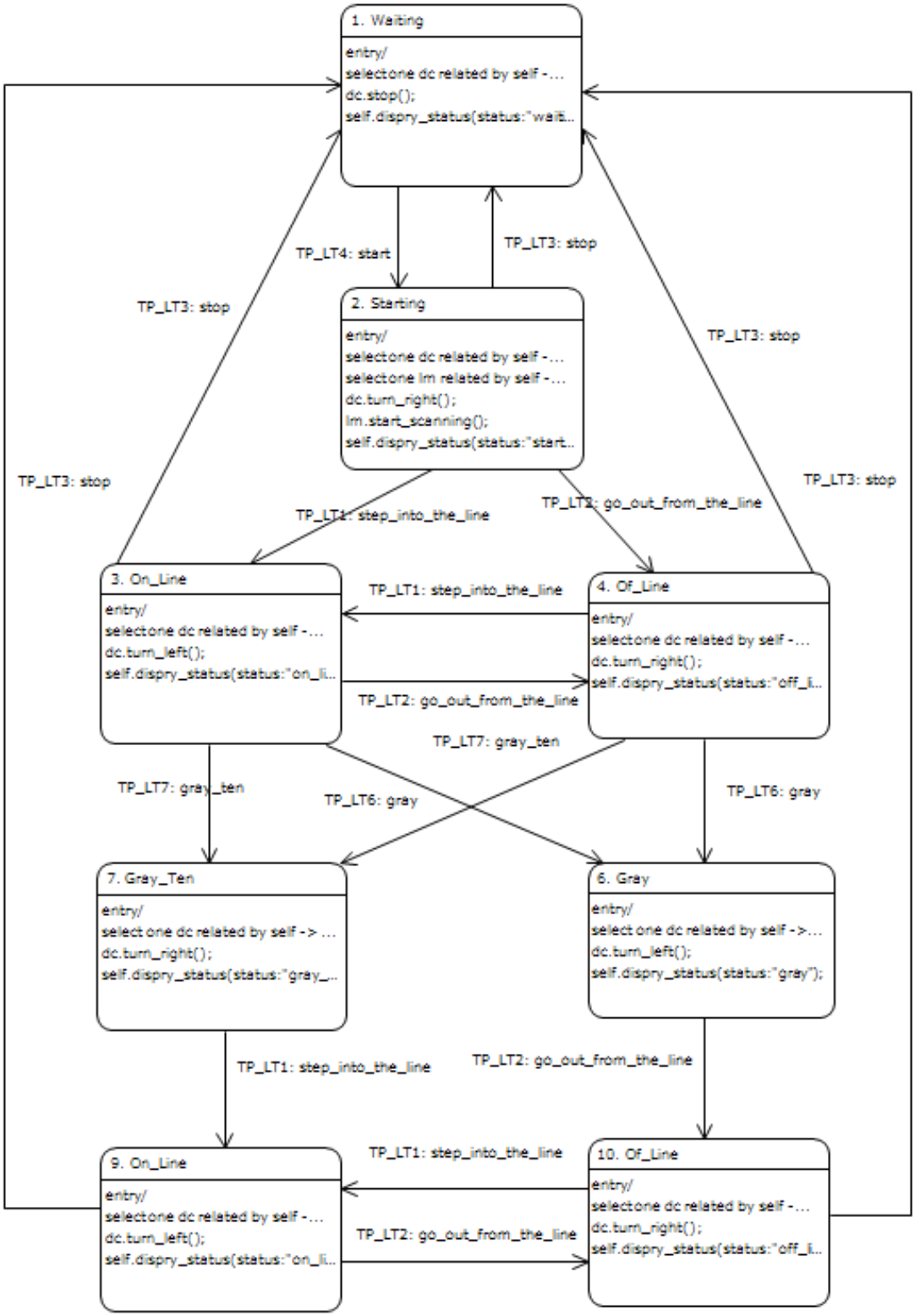


図 3.13　xUML 講座応用編のステートマシン図例

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．各チームのレビュー前後のクラス図を付録 A の図 A.1〜図 A.10 に示す．

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 3.8 に示す．

表 3.8　PBL 編実証講座のクラス図評価結果

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，クラスの動的振舞いを規定することで，ソースコードの自動生成・動作確認ができるため，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を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，クラス内部の振舞いの実装を行い，成果発表時には全チーム課題内容の 7 割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．

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？

モチベーション

各講座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 3.9 に示す．

xUML 講座 1 では，基礎編の実証講座と MDD を教育した応用編，MDD と PBL を組み合わせた PBL 編を比較すると，後半の講座ほど「とても熱心」に取り組んだ受講者が増加している一方，「熱心」と答えた受講生が減少し，「どちらかというと熱心」という受講者が増加している．理解できない部分があったことが，熱心だった受講生の減少につながったと考えられる．しかし，「とても熱心」の学生の数の増加していること，「あまりやる気

表 3.9　取り組み姿勢のアンケート結果

	xUML 講座 1			xUML 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 %

がなかった」という受講生が減少していること，xUML 講座 2 では，「やる気がなかった」などのマイナス意見がなかったこと等を考えると本プログラムで，モデリングにおいてモチベーションが高くないという課題は軽減されたと考える．また PBL では，残業時間がチーム平均 8.4 時間であり，コメントとしてももう少し長期間やりたいという意見が多く，モチベーション高く取り組んだ学生が多かったといえる．今後，講義回数や期間の延長等を行うことで，理解度が高まり，モチベーションの向上もできると考える．

3.4.3　実験の考察

専門学校の 1 年生に対して実施した xUML 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 xUML 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ことが確認できた．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．

協調作業をスムーズに実施するために導入した，タイムボックスかんばんと KPTT 表の利便性を評価するために，受講生のアンケートを分析した．受講者アンケートによると，タイムボックスかんばんを「良い」と答えた受講者は 77% であり，「一目で作業の進捗状況が分かるので計画の見直しができ，作業が進めやすかった.」，「チーム内のそれぞれの動きや成果などが分かりやすかった.」など，高い評価のコメントが多かった．

夕会時に作成する KPTT 表に関しては，全員「良い」と答えており，「一日の終わりに反省が出来て，次回に活かすことができるためテンポよく翌日の活動を行うことができた.」，「普段の活動では良かった面を書くことはなかったが，Keep を書くことが習慣になって，気持ちよく活動できた.」などの意見があり，プロジェクトを円滑に進めるツールとして，効果的であったといえる．

「チーム内の情報共有はできましたか？」という問いに対して，半数が「できた」と答えており，「どちらかというとできた」という学生を加えると 7 割の学生ができていた．タイムボックス制を基本としたガイド付き短期 PBL に関しては，9 割の学生がプロジェクトを円滑に進めるツールとして「効果的であった」と答えており，グループでの活動を支援することができたといえる．さらに，本講座で導入した，タイムボックス制を基本とした活動に関しては，95% の受講生が効果があったとしている．

PBL 編に参加した学生の中の希望者 8 名（1 チーム）が 1 年生の 3 月から 2 年生の 9 月までの約半年間，ET ソフトウェアデザインロボットコンテスト（ET ロボコン）[11] に向けた長期 PBL に参加した．ET ロボコンとは，組込みソフトウェア分野の技術教育をテーマにしたロボコンである．すべてのチームが同一のハードウェア（走行体）を使うことで，ソフトウェアの優劣を競うコンテストであり，UML 等で記述された、走行競技システムの分析・

設計モデル内容の評価を行う「モデル部門」と，自律型ライントレース・ロボットの走行競技による性能の評価を行う「競技部門」からなる．開発工程を学生に自由に設定させた長期 PBL の工程は，図 3.14 の通りである．月 1 回程度，ET ロボコン実行委員会が実施する教育や試走会などのイベントがあるため，それに合わせたスケジュールを 3 月に計画し，進捗状況により，適時スケジュール変更を行った．8 名の参加者のうち，主にモデル作成を担当するソフトウェア設計チームと走行戦略の立案とハードウェアの調整を担当する戦略設計チームに分かれて開発を進めた．なお，ET ロボコン用走行体でライントレースのみを行う BridgePoint モデルは，開始時に教員から配布し，改良して利用してもよいこととした．

また，長期 PBL では参加者に PBL ツールの利用を特に強制しなかったが，長期・短期のガントチャートを作成する，タイマーを用いて時間管理をする，目標に合わせた計画を立て実行し，KPTT 表を用いてふりかえりをするなどの活動を独自に実施していた．従って，本教育プログラムの PBL 編がプロジェクトの進め方や計画の立て方を短期間で学ぶ方法として有益であったと考える．

プログラム受講後に半年間の長期 PBL として ET ロボコン [11] に取り組んだチームと，2008 年〜2010 年過去 3 年間の本校からの ET ロボコン参加チームと比較して考察を行う．2008 年〜2011 年の ET ロボコンへの取り組みの結果を表 3.10 に示す．MDD を利用せず，長期 PBL のみであった，過去 3 年間の参加チームでは，4 名程度と比較的少人数であったが，その中でも 1，2 名の学生がほとんどの活動を行う形態となり，チーム内で分担した活動はできていなかった．さらに，モデルを描くモチベーションが低く，描く時期はモデル審査の直前であった．モデルは出来上がったソースコードにあうように描いていたため，モデルの効果的な利用ができていなかった．本プログラム受講後の学生で編成された 2011 年の取り組みでは，開発早期からモデル作成を行っており，モデルの有用性を理解しモデルを使った開発に抵抗なく取り組めていた．クラス図の作成においては，学生の提案によりクラス図を全員記述し，教員を交えたレビューの結果一番良いモデルを初期バージョンモデルに採用するという方式をとった．その後，各クラスごと

表 3.10　2008 年〜2011 年の ET ロボコンへの取り組み結果

年度	2008 年	2009 年	2010 年 -1	2010 年 -2	2011 年
チーム人数	2 人	4 人	4 人	4 人	8 人
実装言語	C 言語	C 言語	C 言語	C++	C 言語
モデル作成の時期	8 月	8 月	7 月，8 月	6 月〜8 月	5 月〜8 月
MDD の利用	なし	なし	なし	なし	あり
モデル審査評価	C	B	B	C+	B+

に，開発の分担を実施しており，特定のメンバーだけがモデリングを行っていた過去の大会と比べて，効果的なモデルの利用ができていた．

その結果，終了後の参加者アンケートから，参加者の 9 割が設計やモデリングのスキルが向上したと答えている．さらに，ET ロボコン九州地区大会では，モデルの正確性，理解性，設計品質や性能などに関しての評価が行われ，B+ 評価 (全ランク：A 〜 D の 10 段階) を得ることができ，企業や大学生のチームを含めた 35 チーム中 3 位で学生チームの中では 1 位と過去最高の成績を修めることができた．チーム内での作業分担がうまくいった要因としては，MDD を利用したことで，各クラス内の動的な振舞いを独立して作成することができたことや PBL 編（ガイド付き短期 PBL）でチームメンバー全員がプロジェクトの進め方や円滑に進めるためのツールの使い方を習得していたことがあげられる．

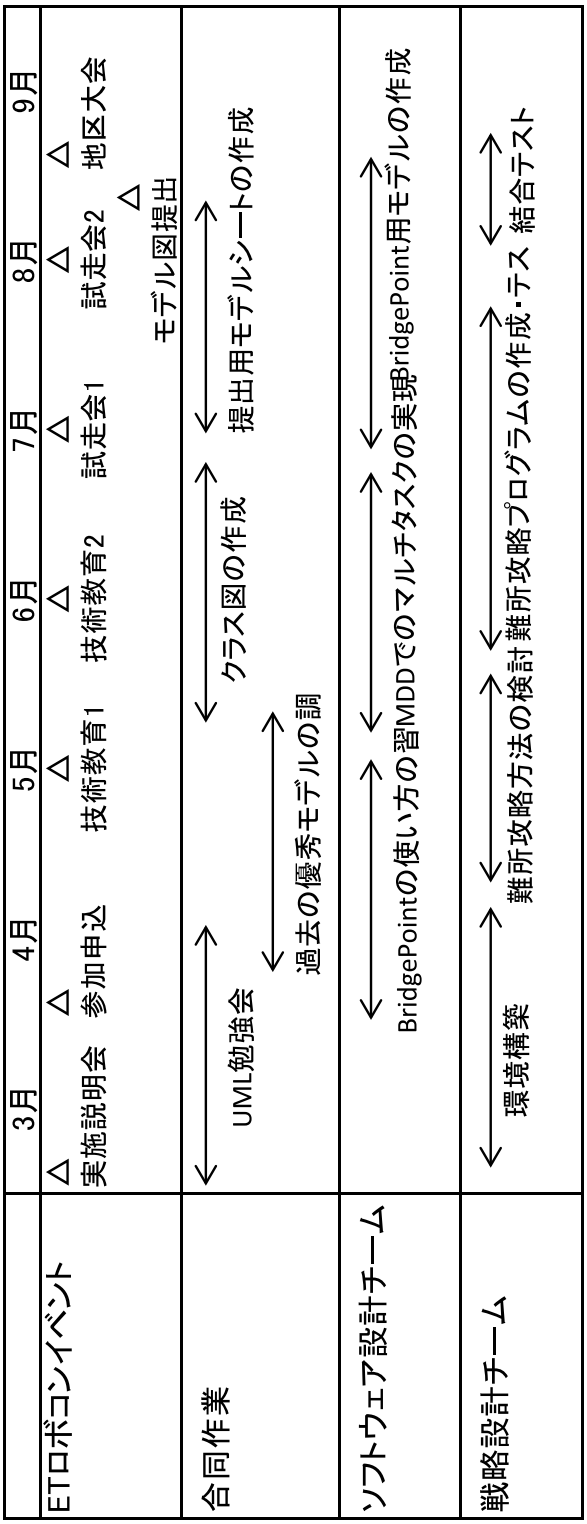


図 3.14　長期 PBL の実施工程

第 4 章　MDD の活用方法に関する実験

本章で行う実験の第一の目的は，研究課題 1-(2)「モデルコンパイラの有用性」及び研究課題 2-(1)「モデルコンパイラの活用タイミング」の検討に寄与する実験を行うことである．そのため，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．

第二の目的は，前章で用いた実行可能モデリング言語である Executable UML と本実験で利用する DSML での教育事例を比較し，研究課題 2-(2)「実行可能モデリング言語の選択」に関する検討材料とすることである．

4.1　MDD ツール

商用の MDD ツールとしては，Rational Rhapsody [54], BridgePoint [31] 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 [30][28]．ただし，前章の教育実践によれば，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 を開発することとした．

4.1.1 DSML

DSML の作成，利用には，メタモデルの定義やモデルの記述を行うツールである DSL ツールが必要である．本研究では，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”[32] 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通りである．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．

- 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，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 4.1 に示す．

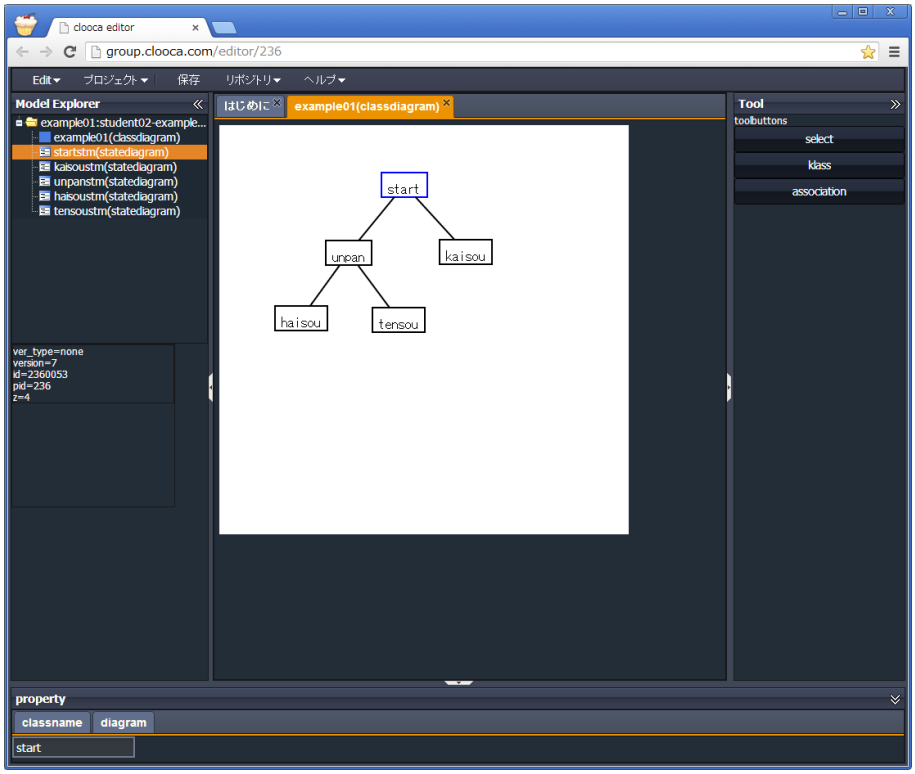


図 4.1　MDD ツール（clooca）のエディタ画面

4.1.2　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．リポジトリの表示の様子を図 4.2 に示す．

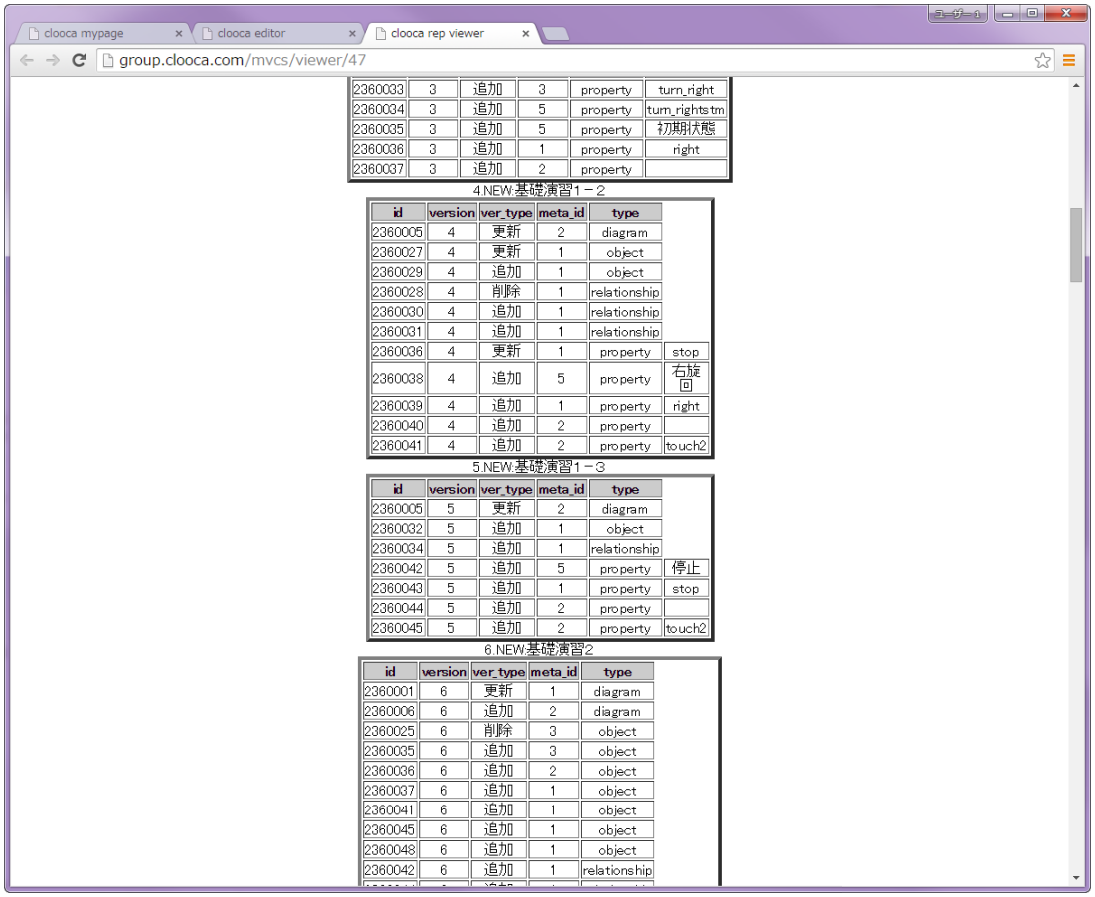


図 4.2　モデルリポジトリの表示の様子

4.2　実験

4.2.1　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ソースコード生成し，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，もう片方をソースコード生成及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ソース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．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，演習の最後のみソースコード生成及び動作確認を行うことを許可した．

4.2.2 評価内容

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．

モデル品質

前章の実証講座により，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課題の機能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 [55] 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 ＞

   - システムの振る舞いを正しく表現しているか．
   - 状態名は，その状態の意味がわ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ソースコード生成試行回数及びプログラムの転送回数を計測し，自由にソースコード生成，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．

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，モチベーションの向上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．

4.2.3 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 DSML 講座の内容を表 4.1 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 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ソースコード生成及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．

4.2.4 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 ツールの使い方及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前章で利用した課題を参考にし，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第 3 章で示した LEGO Mindstorms NXT で製作した自律型車両ロボット（図 3.3 (1)）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

表 4.1　DSML 講座の内容

	MDD なし	MDD あり
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
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

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 4.3 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及び，運搬，転送，回送などのシステムの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の動作（機能）の実現方法を定義する抽象度に低い「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．

1. クラス図

   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

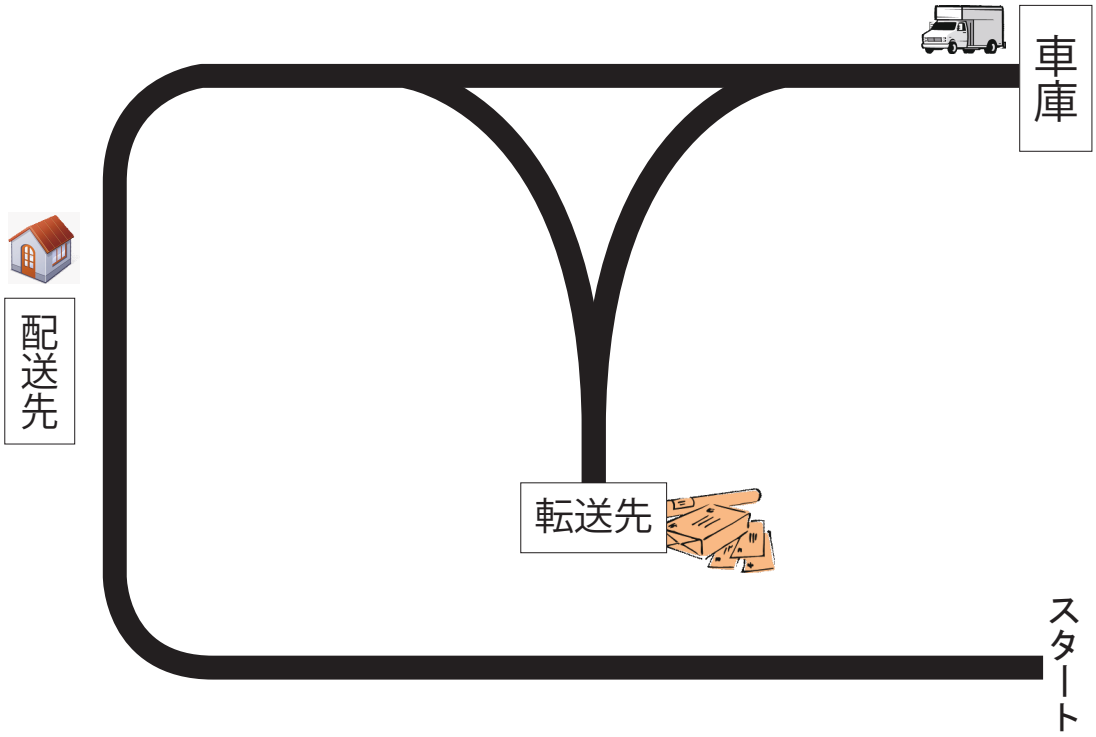


図 4.3　DSML 講座の総合演習課題のコース

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．

関連　クラス間の関係を関連として記述できる．

2. ステートマシン図

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．

振る舞い　各状態の振る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.

4.3 実験結果

4.3.1 モデル作成のアプローチに関する結果

全体の傾向

MDD なしグループのクラス図を図 4.4〜図 4.9 に，MDD ありグループのクラス図を図 4.10〜図 4.15 示す.

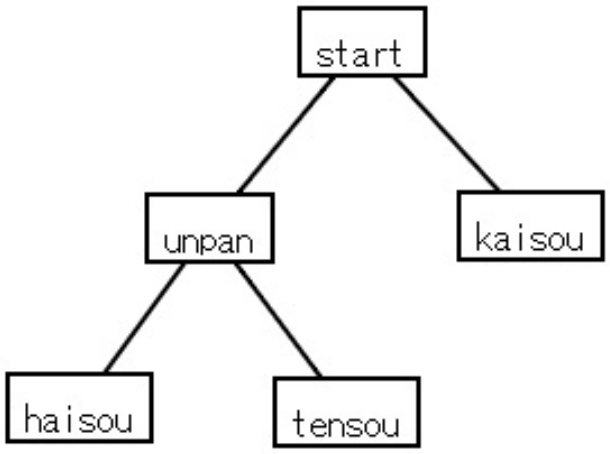


図 4.4 MDD なしグループ被験者 A のクラス図

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，2 種類のクラス図のどちらかだけであった.

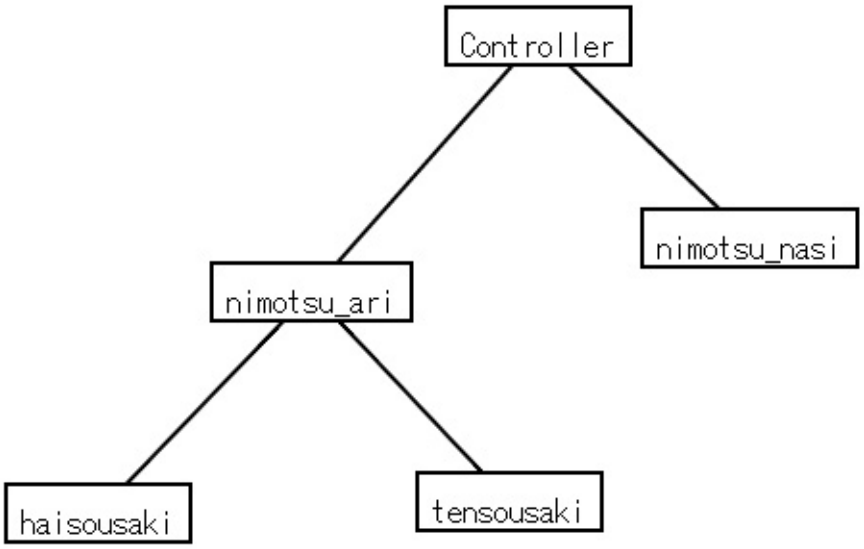


図 4.5　MDD なしグループ被験者 B のクラス図

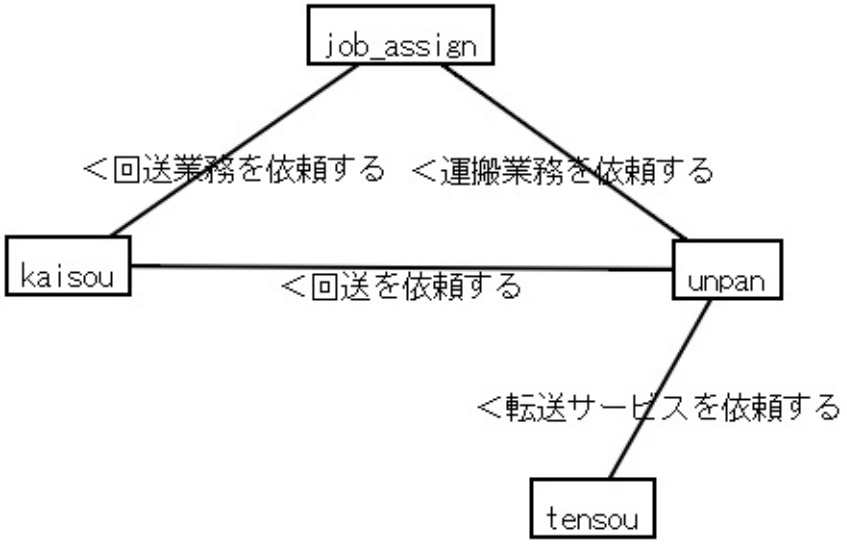


図 4.6　MDD なしグループ被験者 C のクラス図

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 つに分類した結果及び総クラス数を表 4.2 に示す.「全体を制御するクラス」以外に関して，MDD なし

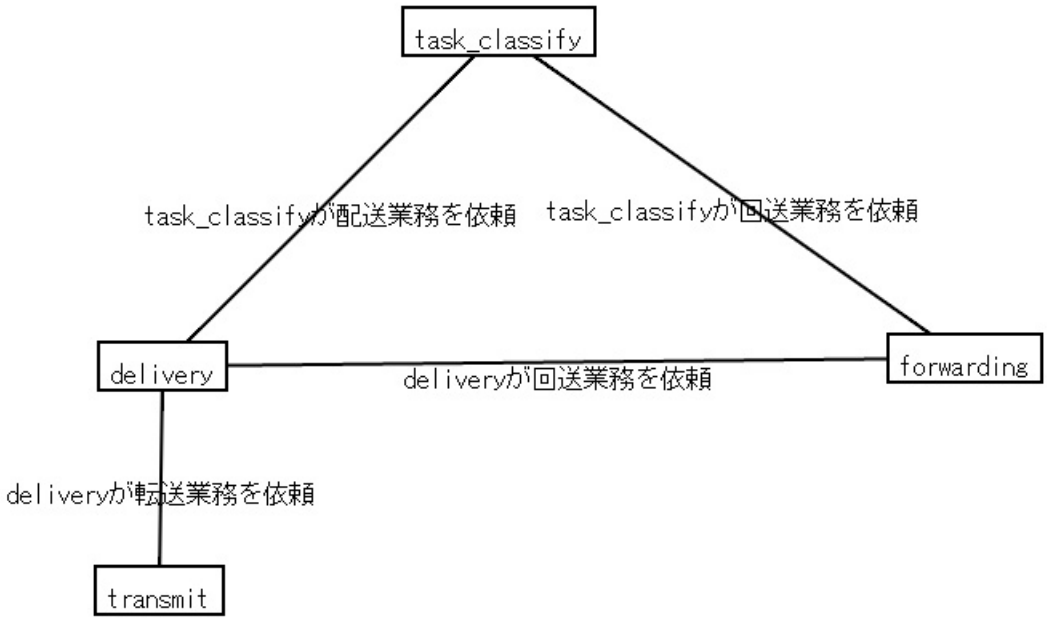


図 4.7　MDD なしグループ被験者 D のクラス図

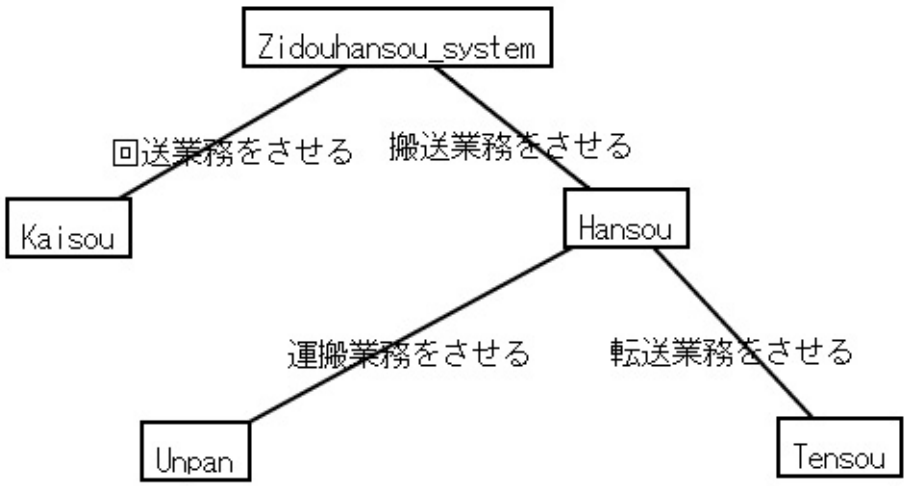


図 4.8　MDD なしグループ被験者 E のクラス図

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.

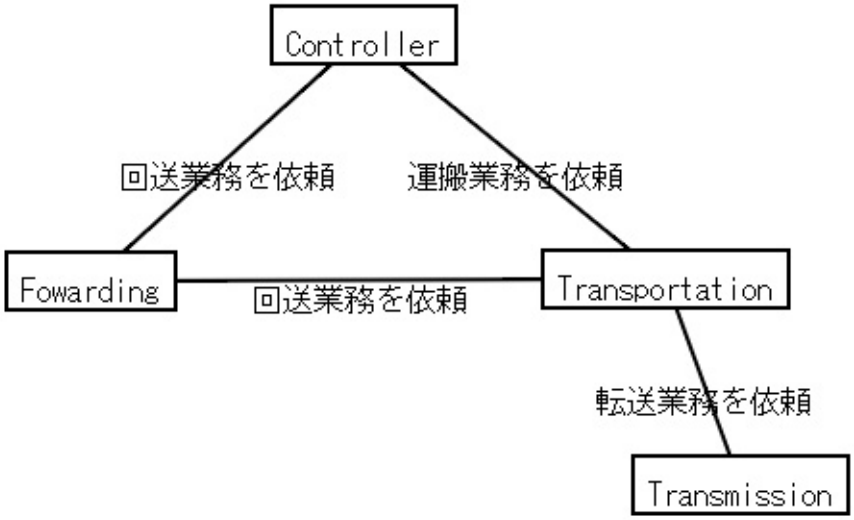


図 4.9　MDD なしグループ被験者 F のクラス図

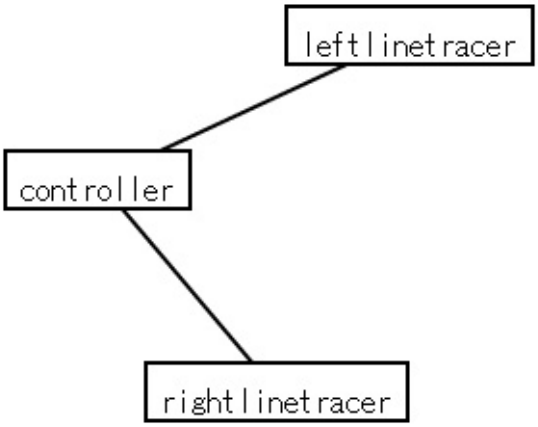


図 4.10　MDD ありグループ被験者 A のクラス図

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 人だった．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，クラス及び状態の追加，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 4.3 に示す．

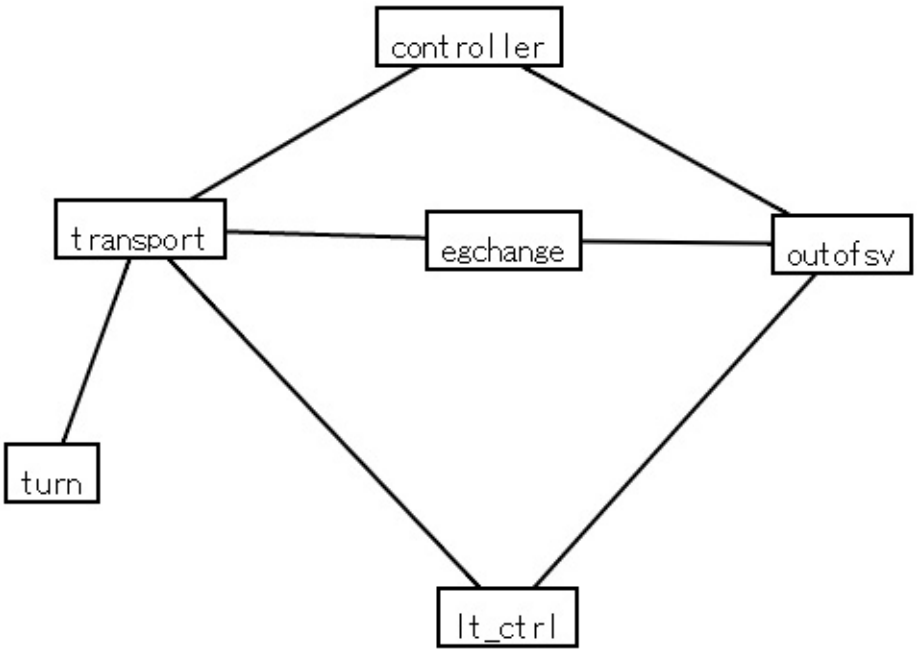


図 4.11　MDD ありグループ被験者 B のクラス図

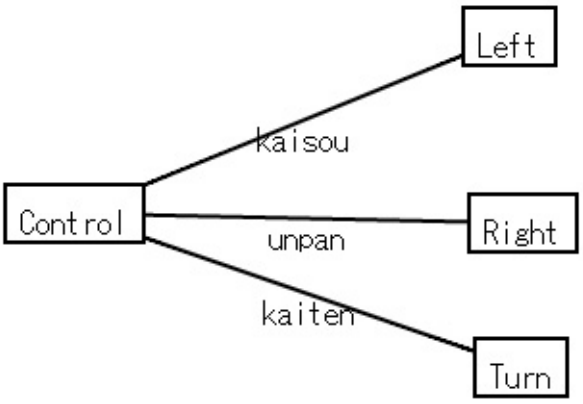


図 4.12　MDD ありグループ被験者 C のクラス図

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，動的振る舞いを規定していると考えられる．一方，

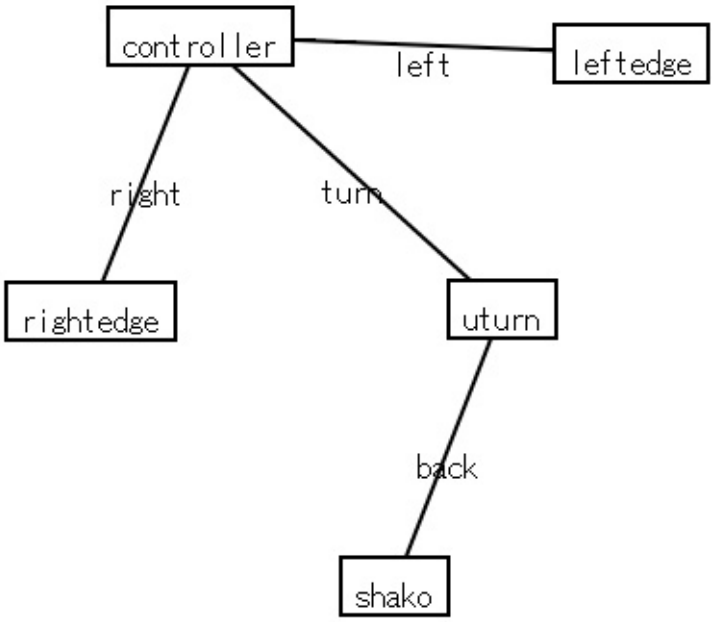


図 4.13　MDD ありグループ被験者 D のクラス図

表 4.2　DSML 講座総合演習課題のクラス名の分類

	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
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．従って，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．

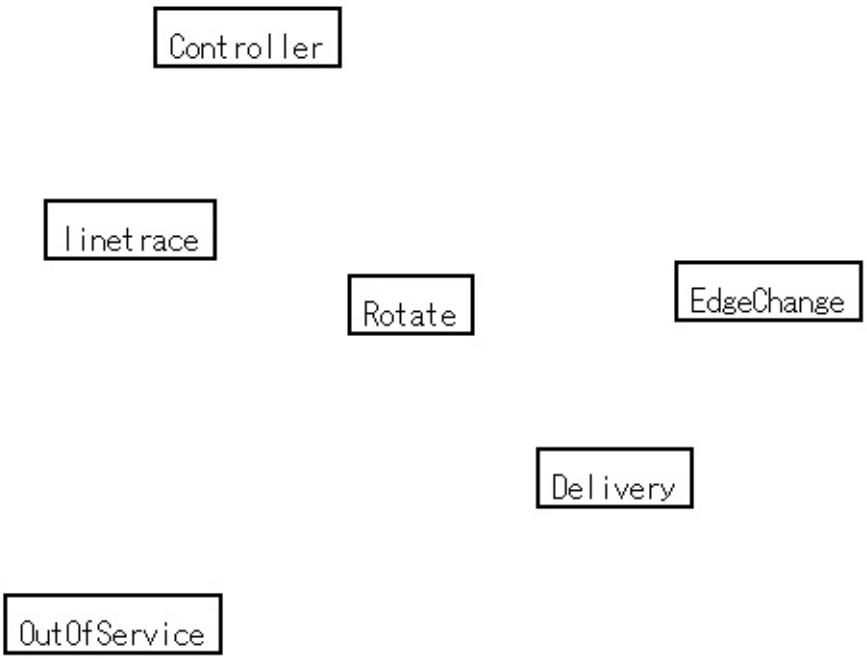


図 4.14　MDD ありグループ被験者 E のクラス図

図 4.15　MDD ありグループ被験者 F のクラス図

個別のモデル作成過程

4.3.1 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 人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. 全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　全体の業務及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

表 4.3　DSML 講座総合演習におけるモデル作成履歴

	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 %

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，「回送」,「運搬」,「転送」である．被験者 A は，同一の “controller” クラス内に振る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及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）機能の振る舞いモデルの記述，2）静的構造モデルの記述，3）業務の振る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わかる．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，全体の制御するクラス，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
edchange　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）全体を制御する振る舞いモデルの記述，3）業務の振る舞いモデルの記述，4）機能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．従って，被験者 B のコード生成試行回数は 8 回（後述の表 4.7 参照）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，振る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伺える．被験者 B は，全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．

4.3.2　モデル品質に関する結果

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 4.4 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．

表 4.4　品質カテゴリに関するエラーを持つクラス数

		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（振る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

ステートマシン図に関する結果

MDD なしグループのステートマシン図を付録の図 B.1〜図 B.11 に，MDD ありグループのステートマシン図を図 B.12〜図 B.23 示す．

品質カテゴリーに関するエラーを持つ状態数を表 4.5 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，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．

表 4.5　品質カテゴリに関するエラーをもつ状態数

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

4.2.2 に示した Semantic 品質の評価対象の 1 つとして，「システムの振る

表 4.6　DSML 講座総合演習課題達成率

	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%

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「全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 4.6 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，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 $10^{55)}$ 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．

4.3.3　モデル品質と動作確認の関係

表 4.7 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 4.4, 表 4.5 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 4.4 の「関連が引かれていない」,「関連名が記入されていない」, 表 4.5 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等，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.

表 4.7　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

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

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.

4.3.4　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 4.8 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 あり，なしの受講者ともに多くの受講生が熱心に取り組んでいた.

MDD の利用あり，なし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 4.9 に示す.

表 4.8　DSML 講座取り組み姿勢のアンケート結果

	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 4.9　DSML 講座モデリングしやすさのアンケート結果

	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%

4.4　実験の考察

本節では，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する．

4.3.1 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，動的振る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る舞いを作成する傾向にあった．MDD あり，なし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 全て

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（MDD あり被験者 E，F）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（MDD あり被験者 F）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 ツールのする導入だけでは，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，モチベーションの向上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 あり，なしの両方を体験した 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．

第 5 章　本研究の考察

本論文では，組込みソフトウェアの開発において，モデリング技術を活用した設計，開発をできる人材を育成する教育プログラム及び学習環境を開発することを目的とした．具体的には，初学者に対するモデリング教育への MDD 及び MDD ツール活用の有用性及び活用方法に関する内容に関して検討した．本章では，本研究での検証項目である以下の研究課題に関する考察を述べる．

研究課題 1　MDD の活用

(1) **初学者の MDD 活用の可否：**　初学者に対して MDD を活用した OOM 教育が行えるのか？

(2) **モデルコンパイラの有用性：**　モデルコンパイラの利用することで，学習者が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，モデリングスキルの向上につながるのか？

研究課題 2　MDD の活用方法

(1) **モデルコンパイラの活用タイミング：**　モデルコンパイラの活用タイミングをどのようにしたらよいのか？

(2) **実行可能モデリング言語の選択：**　どのような、実行可能モデリング言語を利用した教育が効果的なのか？

5.1　MDD の活用

5.1.1　初学者の MDD 活用の可否

第 3 章において提案した教育プログラムの評価のため，専門学校生に対して実証講座を実施した．教育プログラムは，基礎編，応用編，PBL 編の 3 部構成とし，実行可能モデリング言語として，汎用的な言語である Executable

UML を利用した．その後，第 4 章では，応用編のプログラムも実行可能モデリング言語を DSML 変えて実施した．

第 3 章の結果より，MDD を活用することで，学習早期での開発体験が行え，それに伴い学習早期の PBL を活用した教育が可能となった．これにより，モデリング教育における「モチベーション」の課題の軽減につながった．

本プログラムでは，応用編前半では手動変換による「モデルとソースコードの関係」の理解，応用編後半ではソースコードの自動生成による「モデリング方法」の理解を目的としていたが，時間的制約のため，応用編前半の内容を理解させることが困難であった．モデリング及び MDD 教育の場合，「モデリング」と「モデルからソースコードへの変換」は，求められる知識やスキルが異なるため，集中講義として同一の講義内で実施し，理解させるのは困難であった．

第 3 章の xUML 講座 1 の結果より，モデルからソースコードへの手動変換の内容の理解が不十分な場合でも，MDD ツールを用いた開発が行えることが明らかになったため，以後の講座では応用編前半の内容を実施せず，「モデリング方法」の理解に注力した教育プログラムとすることにした．

MDD を活用したモデリング教育を行う場合，オブジェクト指向の考え方，UML の文法，MDD とは何か？ という内容を必須と考え，演習前に最低限の教育を実施した．第 3 章，第 4 章の結果より，上記の内容の理解があれば，初学者でも MDD ツールを活用した開発が行えることがわかった．

5.1.2 モデルコンパイラの有用性

MDD ツールの骨格をなす機能がモデルコンパイラである．モデルコンパイラを利用することで，モデリング言語から実装コードへの自動変換が行える．この機能を用いることで，モデルをすぐ動作として確認することができるため，MDD をモデリング教育に活用するメリットとして下記のように考えていた．

(1) モデリングから動作確認までの時間を短縮でき，モデリング，動作確認というサイクルを繰り返すことにより，モデリング方法の理解を促

進できる．

(2) ソースコードの自動生成機能により，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，学習早期に一連の開発体験を行うことができる．

(3) モデリングした結果をすぐに動作として確認できるため，学習者のモチベーションの向上につながる．

上記 (1) に関しては，モデリングから動作確認の時間の短縮できることは確認できたが，単純に動作確認を何度も繰り返すだけでは，モデリング方法の理解を深めることはできなかった．モデルコンパイラを用いて検証ができるのは，走り方などの機能面のみであり，受講者が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という問題が生じた．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．(2) に関しては，第 3 章，第 4 章の実験では，ともに一連の開発体験が行える総合演習課題を実施しており，モデルコンパイラの活用により，学習早期の一連の開発体験が行えることを確認できた．(3) に関しては，第 3 章，第 4 章の実験ともに演習中のモチベーションの低下という課題は発生することがなく，モチベーションの向上につながることが確認できた．

5.2 MDD の活用方法

5.2.1 モデルコンパイラの活用タイミング

初学者のモデリング教育に対して，MDD を活用した事例が少ないため，初めに汎用的なモデリング言語である Executable UML 及び実開発向けの MDD ツール（BridgePoint）を導入した初学者向けの教育プログラムの検討及び実証講座を実施した．講座の結果，モデリング方法の理解及びモチベーションの向上につながったものの，プロが利用するツールの使い方やアクションを規定する専用言語の習得に時間がかかり，モデリング教育にかかる時間の短縮効果は低かった．次に，DSL プラットフォームを用いて，教育用の DSML を開発し，DSML を用いた教育を実施した．さらに，モデルコ

ンパイラを学習者の裁量でいつでも利用できるグループ（MDD ありグループ）と演習の最後のみ利用できるグループ（MDD なしグループ）の 2 種類に分けて，講座を実施した．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，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．

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 [56] 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．

本研究の対象とした，クラス図とステートマシン図の組み合わせは，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 [55]，開発は組込みシステムを対象としたものが多い [3]．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．

4.3.1 項の分析結果をもとに，動作確認のタイミングを考える．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（本稿では，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及びタイミングに関して検討する必要がある．

5.2.2 実行可能モデリング言語の選択

Executable UML と DSML において，モデリングのしやすさを比較する．モデリングのしやすさは，言語仕様によるものと，ツールの使いやすさのそれぞれに依存する．受講者コメント，モデル図等の受講者の成果物を基に，言語仕様によるものを中心に分析を行う．

Executable UML は，汎用的なモデリング言語であり，幅広い分野のモデルを記述することが可能である．ただし，初学者にとっては，アクション言語の習得が新たなプログラミング言語の習得のように捉えられ，アクション言語の習得，利用のハードルが，抽象度の高いモデリングに集中することを阻害していた．さらに，クラス名や状態名など，モデルに必要な名前をすべて英数字で記載する必要があり，本実証講座での受講者のほとんどは英語が得意ではなく，日本語から英語への変換にも多くの時間を必要とするとともに，モデリングを続けるモチベーションの低下につながっていた．また，英語のスペルミスや翻訳ミスがあり，モデルを第三者が見たときの理解性の低下にもつながっていた．

しかしながら，Executable UML を用いたモデリング方法の十分な理解があれば，様々なモデルを記述することができる．本講座で実施した PBL でも，仕様変更に伴う機能の追加に対して，ツールに手を加えることなくモデル記述し，コード生成，動作確認を行うことができた．

これに対し，DSML を用いた教育では，開発ドメインを限定して，事前にモデリング言語の作成を必要とする．本研究では，Executable UML での講

座の結果を踏まえて，DSML の設計を行った．細かな文法にとらわれることなく，モデリング作業に集中できることを目標とし設計を行ったため，自由度の低い言語となったが，本実証講座でのシステム開発においてはスムーズな開発が行えていた．xUML 講座での総合演習課題（応用編）の開発時間は 9 時間であったのに対し，DSML 講座での開発時間は 3 時間であり，課題の難易度の変更を考慮に入れても，DSML の方が短時間での開発を可能にするといえる．DSML 講座の受講者のコメントにおいても，「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を組めるので，楽しみながら行えた.」などがあり，モチベーション高く開発できるといえる．

次に各学習項目に関して，MDD 技術及び実行可能モデリング言語がどのように学習を支援できるかを検討する．

責務分割，クラス名，関連

クラス名の記述に関しては，Executable UML，DSML どちらも英数字での記載が必要であり，2 種類の言語に関して差異はない．また，関連に関しても関連名を書くことは，コードの自動生成の有無には関係がないため，MDD の活用が学習支援には寄与しない．

MDD を用いるメリットとしては，開発の後半にクラス構造やクラス名の変更を行った場合の影響が少ないことである．xUML 講座の PBL 編での中間レビュー時に，クラス構造の変更を指示したが，成果発表時には，多くのチームが課題を達成するモデルを作成しており，構造の変更から動作確認までの時間か短縮できることが確認できた．DSML 講座では，総合演習課題のモデル作成作業の後半 3 分の 1 の作業ログを確認したところ，作業のおよそ 1 割をクラスの追加，修正，削除の作業にあてており，作業後半でもクラスの変更ができていた．

モデリング教育においては，動作を確認しながらクラスを修正する作業を繰り返すことで，よりよいクラス構造を習得することに繋がると考えられるため，MDD 技術を活用するメリットは高いといえる．

状態名

Executable UML では，状態名の記述は英数字で記載する必要があるのに対し，本講座用に設計した DSML は，日本語での記述も可能にした．それにより，英語に翻訳する時間が短縮できよりスムーズに状態名を決定することができり，モデル作成の時間が短縮できることが確認できた．しかし，本講座の結果では，空白または意味がわからない状態名になっているモデルの割合は DSML 講座の方が高かった．DSML 講座では単独開発であったのに対し，xUML 講座はペアまたはチームでの開発であったため，お互いに状態名を確認しながら進めたことがこの違いの要因だと考えられる．

ただし，xUML 講座の受講者のアンケートではアクション言語や英語の難しさに関するものが複数あったこと，モデルを見てから内容を理解するのにかかる時間がかかることを考慮すると，特に初学者の教育においては，DSML を用いる方がよいと考えられる．

遷移，振る舞い

遷移，振る舞いを決定する，イベントやアクションはプルダウンメニューから選択をするだけで決定することができるため，ステートマシン図の作成時間の大幅な短縮につながっている．xUML 講座では，振る舞いをアクション言語で記述する必要がある．また，アクション言語では，振る舞いを定義するほか，インスタンスやリンクの生成，他のインスタンスにイベントを送信する際には，リンクをたどり送信先を指定するプログラムを記述する必要があり，作成に時間がかかる．さらに，xUML 講座と DSML 講座のステートマシン図を比較すると，Executable UML では，ステート内のアクション記述が長く，内容の理解に時間がかかるのに対し，DSML では，アクションは一行で定義できるため，どのような動作をするステートマシン図なのかがすぐ理解できる．

本講座で用いた DSML では，クラスのインスタンスが 1 つ自動的に生成される仕組みになっているため，インスタンス生成などのプログラムを書く必要がなく，作成や理解にかかる時間を短縮できている．一方，インスタ

ンスが 1 つしか生成できないため，多重度を定義することができない．多重度の教育に関しては，新たな DSML を作成する必要がある．このように，DSML では，自由度を少なくすることで，教育対象項目を絞り，理解度を上げる工夫ができる．モデリング技術における様々な教育項目ごとに DSML を作成したり，理解度が進むにつれて自由度の高い DSML を利用させることで，モデリングの教育効果が上がると考えられる．

第 6 章　おわりに

本章では，本論文のまとめと今後の課題について述べる．

6.1　まとめ

(1) 研究課題の設定と解決方法

本研究では，設計品質の向上が求められている組込みソフトウェア開発の初学者を対象とした OOM 教育の課題解決のために，産業界での実用化が進んでいる MDD を活用することを提案した．本論文では，モデリング教育への MDD の活用の有用性及び活用方法に関する下記の研究課題に関して言及した．

研究課題 1　MDD の活用

(1) **初学者の MDD 活用の可否：**　初学者に対して MDD を活用したモデリング教育が行えるのか？

(2) **モデルコンパイラの有用性：**　モデルコンパイラの利用することで，学習者が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，モデリングスキルの向上につながるのか？

研究課題 2　MDD の活用方法

(1) **モデルコンパイラの活用タイミング：**　モデルコンパイラの活用タイミングをどのようにしたらよいのか？

(2) **実行可能モデリング言語の選択：**　どのような、実行可能モデリング言語を利用した教育が効果的なのか？

上記の研究課題について，本論文で述べた回答を以下にまとめる．まず，上記の課題を明らかにするため，本論文では 2 種類のモデリング言語を用いた実証講座を実施した．講座を実施するにあたり，第 1 章にモデリング教育

の課題としてあげていた「理解性」,「必要性」,「モチベーション」の 3 点の軽減のために，教育プログラムの要件を設定し，教育プログラムを開発した．教育プログラムの要件の中核は，モデルを作成したらすぐに動作を確認できる環境の提供であり，その要件を達成するために MDD を活用することとした．作成した教育プログラムは基礎編，応用編，PBL 編の 3 部であり，段階的に知識やスキルを身に付けられるように，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方法を用いた．

初めに，教育プログラムの評価及び研究課題 1-(1) 及び (2) の検討に寄与することを目的として，汎用的なモデリング言語（Executable UML）及び実開発向けの MDD ツール（BridgePoint）を用いたモデリング教育プログラム評価実験を実施した．教育プログラムの評価結果の概要は下記の通りである．

理解性： MDD 教育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，短期間でのモデリングスキルの向上が図れる．

必要性： モデリングの必要性の理解につながった．

モチベーション： モチベーション高く取り組んだ学生が多かった．講義回数や期間の延長等を行うことで，より理解度が高まり，モチベーションの向上が期待できる．

次に，研究課題 2-(1) の検討のために，教育用の DSML 及び教育用 MDD ツールを用いて，モデルコンパイラを学習者の裁量でいつでも利用できるグループ（MDD ありグループ）と演習の最後のみ利用できるグループ（MDD なしグループ）の 2 種類に分けて MDD の活用方法に関する実験を実施した．

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，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．

以上の講座の結果を踏まえ研究課題 2-(2) の考察を行った．各研究課題に関する考察の概要は下記の通りである．

(2) 研究課題への回答

以上の結果を踏まえ，各研究課題に関して，本論文で示した回答を下記にまとめる．

研究課題 1-(1) 初学者の MDD 活用の可否：

オブジェクト指向の考え方，UML の文法，MDD とは何か？ という内容の理解があれば，初学者でも MDD での開発を行える．

研究課題 1-(2) モデルコンパイラの有用性：

モデルコンパイラの活用により，モデリングから動作確認の時間が短縮できること，学習早期に一連の開発体験ができること及びモチベーションの向上につながることを確認できた．ただし，単純に動作確認を何度も繰り返すだけでは，モデリング方法の理解を深めることができなかった．

研究課題 2-(1) モデルコンパイラの活用タイミング：

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（本稿では，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．

研究課題 2-(2) 実行可能モデリング言語の選択：

2 種類のモデリング言語を用いた講座の結果，DSML を用いた講座の方が短期間での演習課題の開発に取り組めており，モデリングの入門教育においては，教育対象の課題や教育項目に特化した DSML を作成し，アプリケーションを開発に取り組ませることが効果的である．

(3) 本論文の成果

本論文の研究成果は，以下の通りである．

1. 初学者向けの OOM 教育として，MDD を活用することを提案した．MDD 教育は，プログラミングや UML モデリング等のソフトウェア開発教育の後に実施されることが一般的あり，初学者向けに MDD 教育を行われた事例は少ない．本論文では，MDD を活用した初学者向けの OOM 教育プログラムを開発し，実証講座を行うことで，プログラミング技術が十分にない初学者でも MDD による開発が行えること明らかにした．さらに，MDD におけるコードの自動生成機能を利用し，作成したモデルをすぐに実行して確認することで，モデルの不具合の一部を機械的に検出できるため，学習者の自己学習及び教員の負担軽減の促進を可能にした．これらにより，OOM のスキル習得のための教育方法に関して新たな知見を与えていると考える．
2. 教育用 DSML を開発し，よりスムーズな OOM 教育を行える環境の整備を行った．MDD に用いられる汎用的なモデリング言語及び実開発向けのツールでは，初学者が言語やツールに慣れるのに時間がかかり，主とする学習内容以外の部分に時間をとられるというデメリットがあった．本論文では，事前の実証講座の結果を踏まえ，UML の記法に合わせた教育用の DSML を開発することで，より短期間でのモデリングスキルの習得を可能にした．さらに，学習者のスキルレベルに合わせた DSML を開発することで，より短期間での学習者のレベルアップを支援できる可能性を示唆し，DSML 設計指針を提案することで，OOM 教育環境に新たな知見を与えていると考える．
3. MDD ツールを活用した OOM 教育を行う場合に，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の提案を行った．MDD ツールを活用した OOM 教育においては，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問題点がある．従って，モデリング教育において，コードの自動生成を行うモデルコンパイラをどのタイミングで利用するかが重要であり，本論文では，モデルコ

ンパイラの利用のタイミングの異なる 2 つのグループの比較実験により，MDD ツールの利用がモデルに与える影響を明らかにすることで，より効果的な MDD のツールの活用方法の提案を行い，MDD を活用した OOM 教育プログラムを開発する際の指針を与えていると考える．

6.2 今後の課題

本研究で対象とした「OOM 教育方法」,「OOM 教育環境」,「OOM 教育プログラム」に関して今後の課題を述べる．

1. OOM 教育方法

   モデリング教育において，MDD を活用することに加えて，利用する MDD ツールに付加機能を設けることにより，学習者の自己学習及び教員の負担軽減をより促進できると考えている．

   付加機能の例として，文法的誤りやモデル品質に影響を与える可能性がある部分に対してエラーや警告を表示する機能などがある．上記のようなエラーを表示する機能を設けることで，学習者はエラー箇所を早く知ることができ，より早いフィードバックを与えることができる．これにより，モデリング学習の短期間化やモチベーションの向上につながると考える．

   さらに，これらの機能を設けることができれば，集団教育だけではなく，近年注目集めている MOOCS（Massive Open Online Courses）に代表される，オンラインによる遠隔教育への活用も期待できる．

2. OOM 教育環境

   本論文では，初学者の OOM 教育に MDD を活用することを提案した．実証講座の結果，教育対象や教育項目に特化した DSML を作成し，アプリケーションを開発に取り組ませることが効果的であることを明らかにした．

   今後は，各教育段階及び教育項目において，どのような DSML を設計し活用するとより効果的な教育ができるかを明らかにする必要があ

る．本論文で対象とした，初学者への入門教育においては，UML の図を静的モデル及び動的モデルの各 1 種類ずつと限定したが，中級者や上級者への教育においては，複数の図を描画できるツールにすることで，最適な図の選択や図の利用方法を習得できるような環境にすることが考えられる．また，リストからの選択式にしたアクション定義に関しても，モデルとソースコードのつながりを学習することを目的として，アクション言語で記述する，実装コードで記述するなども想定される．

3. OOM 教育プログラム

MDD をソフトウェアモデリング教育に活用した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．モデルの品質に意識を持たせる方法として，演習中にモデルコンパイラを利用する回数を制限し，学習者自信でモデルを確認するように促す方法が考えられる．しかしながら，本研究の結果からは，MDD を活用した OOM 教育における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，今後検証する必要がある．

謝辞

本研究を進めるにあたり，度重なるご指導とご助言を賜り，また多くのご支援を頂戴致しました九州大学 システム情報科学研究院 福田晃教授深く感謝の意を表します.

本論文を取りまとめるにあたり，ご助言とご指導を賜りました九州大学システム情報科学研究院 荒木啓二郎教授ならびに鵜林尚靖教授に深く感謝いたします.

本研究を進めるにあたり，度重なる議論の場を用意しご指導を頂きました九州大学 システムＬＳＩ研究センター 久住憲嗣准教授に深く感謝いたします.

研究を進めるにあたり，様々なご指導とご助言を頂きました名古屋大学大学院 情報連携統括本部 情報戦略室 山本修一郎教授に心より感謝の意を表します.

共同研究者として，多大なご支援を頂いた株式会社 Technical Rockstarts の部谷修平氏に深く感謝いたします.

教育プログラムの開発において多くのご支援ならびにご助言を頂戴致しました株式会社チェンジビジョン 久保秋真氏，株式東陽テクニカ 二上貴夫氏，熊本大学大学院 自然科学研究科 北須賀輝明准教授に深く感謝いたします.

九州技術教育専門学校ならびに福田・久住・アシル研究室の皆様には，貴重なご意見や様々なご支援をいただきましたことに心より感謝いたします.

実験への協力を頂きました徳山工業高等専門学校ならびに神戸電子専門学校の皆様に心より感謝いたします.

また，本研究ならびに本論文の執筆に際し，数多くの方々から多大なるご支援とご助言を賜りましたことを，ここに記して感謝いたします.

最後に私を支えてくれました父 武興に感謝いたします.

参考文献

[1] 独立行政法人情報処理推進機構：2012 年度「ソフトウェア産業の実態把握に関する調査」調査報告書 (2013).

[2] 経済産業省：2010 年度版組込みソフトウェア産業実態調査報告書 (2010).

[3] 荒井玲子：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).

[4] Kelly, S. and Tolvanen, J.-P.: *Domain-Specific Modeling: Enabling Full Code Generation*, Wiley-IEEE Computer Society Pr (2008).

[5] 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).

[6] 独立行政法人情報処理推進機構：「モデルベース設計検証技術者スキル体系化調査」調査報告書 (2012), available from ⟨http://www.ipa.go.jp/files/000004595.pdf⟩, (accessed 2015-07-22).

[7] 独立行政法人情報処理推進機構ソフトウェア・エンジニアリング・センター：【新版】組込みスキル標準 ETSS 概説書 (2009).

[8] 独立行政法人情報処理推進機構：組込みスキル標準「スキル基準」- 開発力を強化するスキルの最適活用に向けて- (2007).

[9] 経済産業省：組込みソフトウェアのエントリ人材教育に関する検討報告書〜実践的な組込み開発エントリ人材育成カリキュラム普及のために〜 (2007), available from ⟨http://www.ipa.go.jp/files/000023851.pdf⟩, (accessed 2015-07-22).

[10] 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).

[11] ET ロボコン実行委員会：ET ロボコン，組込みシステム技術協会

(online)，available from ⟨http://www.etrobo.jp⟩ (accessed 2015-07-22).

[12] 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).

[13] Goldberg, A. and Rubin, K. S.: *Succeeding with Objects*, Addison-Wesley Publishing Company (1995).

[14] 情報処理学会ソフトウェアエンジニアリング教育委員会：カリキュラムモデル J07-SE の概要 (2009).

[15] Fowler, M.: UML モデリングのエッセンス 第 3 版，翔泳社 (2005).

[16] Jacobson, I., B. G. and Rumbaugh, J.: UML による統一ソフトウェア開発プロセス—オブジェクト指向開発方法論，翔泳社 (2000).

[17] Kruchten, P.: ラショナル統一プロセス入門　第 3 版，アスキー (2004).

[18] Larman, C.: 実践 UML　第 2 版—パターンによる統一プロセスガイド，ピアソン・エデュケーション (2003).

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

[20] Mellor, S. J., e. a.: MDA のエッセンス-モデル駆動型ソフトウェア開発入門-，テクノロジックアート (2004).

[21] 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).

[22] OMG: *Semantics of a Foundational Subset for Executable UML Models (fUML), v1.1* (2013), available from ⟨http://www.omg.org/spec/FUML/⟩ (accessed 2015-07-22) .

[23] OMG: *Action Language for Foundational UML (Alf) Concrete Syntax for a UML Action Language Version 1.0.1* (2013), available from ⟨http://www.omg.org/spec/ALF/⟩ (accessed 2015-07-22) .

[24] UMTP/Japan: UMTP, 特定非営利活動法人 UML モデリング推進協議 (online)，available from ⟨http://www.umtp-japan.org/⟩ (accessed 2015-07-22).

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

[26] Paige, R., Polack, F., Kolovos, D., Matragkas, N. and Williams, R.: Bad Modelling Teaching Practices, *Proc. of the Educators' Symposium co-located with ACM/IEEE 17th International Conference on Model Driven Engineering Languages and Systems (MODELS 2014)* (2014).

[27] 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).

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

[29] Flint, S., Gardner, H. and Boughton, C.: Executable/Translatable UML in Computing Education, *Proc. Sixth Australasian Computing Education Conference*, Australian Computer Society, pp. 69–75 (2004).

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

[31] xUML.org: eXecutable Translatable UML Open Source Editor, xUML.org (online), available from ⟨https://www.xtuml.org/⟩ (accessed 2015-07-22).

[32] Hiya, S.: clooca, Technical Rockstars (online), available from ⟨http://www.clooca.com⟩ (accessed 2015-07-22).

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

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

[35] Hasker, R. W. and Shi, Y.: Teaching Basic Class Diagram Notation with UMLGrader, *Proc.121st ASEE Annual Conference and Exposition*, American Society for Engineering Education (2014).

[36] Auxepaules, L. and Py, D.: An Evaluation of Diagnosis in a Learning Environment for Object-Oriented Modeling, *Advanced Learning Technologies (ICALT), 2010 IEEE 10th International Conference on*, pp. 102–104 (2010).

[37] Schramm, J., Strickroth, S., Le, N.-T. and Pinkwart, N.: Teaching UML Skills to Novice Programmers Using a Sample Solution Based Intelligent Tutoring System, *FLAIRS Conference'12*, pp. –1–1 (2012).

[38] Striewe, M. and Goedicke, M.: Automated checks on UML diagrams, *Proc. of the 16th annual joint conference on Innovation and technology in computer science education*, ACM, pp. 38–42 (2011).

[39] Coelho, W. and Murphy, G.: ClassCompass: A Software Design Mentoring System, *Journal on Educational Resources in Computing (JERIC)*, Vol. 7, No. 2, pp. 1–18 (2007).

[40] 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).

[41] 光太郎野沢，芳昭松澤，三四郎酒井：一貫性・明瞭性診断による静的UML モデリング学習支援システムの設計と評価，情報処理学会論文誌，Vol. 55, No. 5, pp. 1471–1484 (2014).

[42] 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).

[43] Ramollari, E. and Dranidis, D.: StudentUML: An Educational Tool Supporting Object-Oriented Analysis and Design, *Proc. 11th Panhellenic Conference in Informatics*, pp. 363–373 (2007).

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

[45] Tekinerdogan, B.: Experiences in teaching a graduate course on model-

driven software development, *Computer Science Education*, Vol. 21, No. 4, pp. 363–387 (2011).

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

[47] 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).

[48] 沢田篤史，小林隆志，金子伸幸，中道 上ほか：飛行性制御を題材としたプロジェクト型ソフトウェア開発実習，情報処理学会論文誌，Vol. 50, No. 11, pp. 2677–2689 (2009).

[49] 松澤芳昭，塩見彰睦，秡川友宏，酒井三四郎：ソフトウェア開発の教員主導型 PBL における反復プロセスと EVM 導入の効果，情報処理学会研究報告，Vol. 2009-CE-99, No. 9, pp. 1–8 (2009).

[50] 金田重郎，井上明：実システム開発を通じた社会連携型 PBL の提案と実践，情報処理学会研究報告，Vol. 2009-IS-107, No. 32, pp. 185–192 (2009).

[51] 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 (2011), available from ⟨http://objectclub.jp/community/pf/⟩ (accessed 2015-07-22).

[52] 平鍋健児，天野勝：プロジェクトファシリテーション実践編朝会ガイド (2013), available from ⟨http://objectclub.jp/community/pf/⟩ (accessed 2015-07-22).

[53] 独立行政法人情報処理推進機構：IT スキル標準 (ITSS)，IPA（online），available from ⟨http://www.ipa.go.jp/jinzai/itss/index.html⟩ (accessed 2015-07-22).

[54] IBM: Rational Rhapsody family, IBM (online), available from ⟨http://www-03.ibm.com/software/products/ja/ratirhapfami/⟩ (ac-

cessed 2015-07-22).

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

[56] 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, Springer, pp. 40–47 (2012).

本研究に関する著者の発表論文

(1) 赤山 聖子，久保秋 真，久住 憲嗣，二上 貴夫，北須賀 輝明: “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”，組込みシステムシンポジウム 2011(ESS2011), pp.15-1-15-9, 2011 年 10 月

(2) Seiko Akayama, Shin Kuboaki, Kenji Hisazumi, Takao Futagami, and Teruaki Kitasuka: “Development of a Modeling Education Program for Novices using Model-Driven Development”, Proceedings of the 2012 Workshop on Embedded and Cyber-Physical Systems Education (WESE2012), Article No. 4, Oct. 2012.

(3) 赤山 聖子，久住 憲嗣，久保秋 真，部谷 修平，福田晃: “効果的なオブジェクト指向モデリング教育のための実行可能モデリング言語の比較評価”，情報教育シンポジウム 2013(SSS2013),pp.125-132, 2013 年 8 月

(4) Seiko Akayama, Kenji Hisazum, Syuhei Hiya, and Akira Fukuda: “Using Model-Driven Development Tools for Object-Oriented Modeling Education”, Proceedings of the Educators’ Symposium co-located with ACM/IEEE 16th International Conference on Model Driven Engineering Languages and Systems (MODELS 2013), online CEUR-WS.org/Vol-1134/paper2.pdf, Sep. 2013.

(5) Seiko Akayama, Birgit Demuth, Timothy C. Lethbridge, Marion Scholz, Perdita Stevens, and Dave R. Stikkolorum: “Tool Use in Software Modelling Education” Proceedings of the Educators’ Symposium co-located with ACM/IEEE 16th International Conference on Model

Driven Engineering Languages and Systems (MODELS 2013), online CEUR-WS.org/Vol-1134/paper6.pdf, Sep. 2013.

(6) 赤山 聖子，久住 憲嗣，部谷 修平，福田晃: “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”，情報処理学会論文誌，Vol.55, No.1, pp.72-84, 2014 年 1 月

(7) Seiko Akayama, Kenji Hisazumi, Shin Kuboaki, Shuhei Hiya, and Akira Fukuda: “Comparative Evaluation of Executable Modeling Languages for Object-Oriented Modeling Education” Proceedings of The 2014 International Conference on Software Engineering Research and Practice(SERP’14), pp.37-43, Jul. 2014.

付録 A - xUML 講座 1 のクラス図

第 3 章のモデリング教育プログラム評価実験の PBL 編講座では，中間レビューを行うことでモデル品質の確保を行った．BridgePoint ツールを用いて受講生のチームが描いたレビュー前後のクラス図を図 A.1〜図 A.10 に示す．

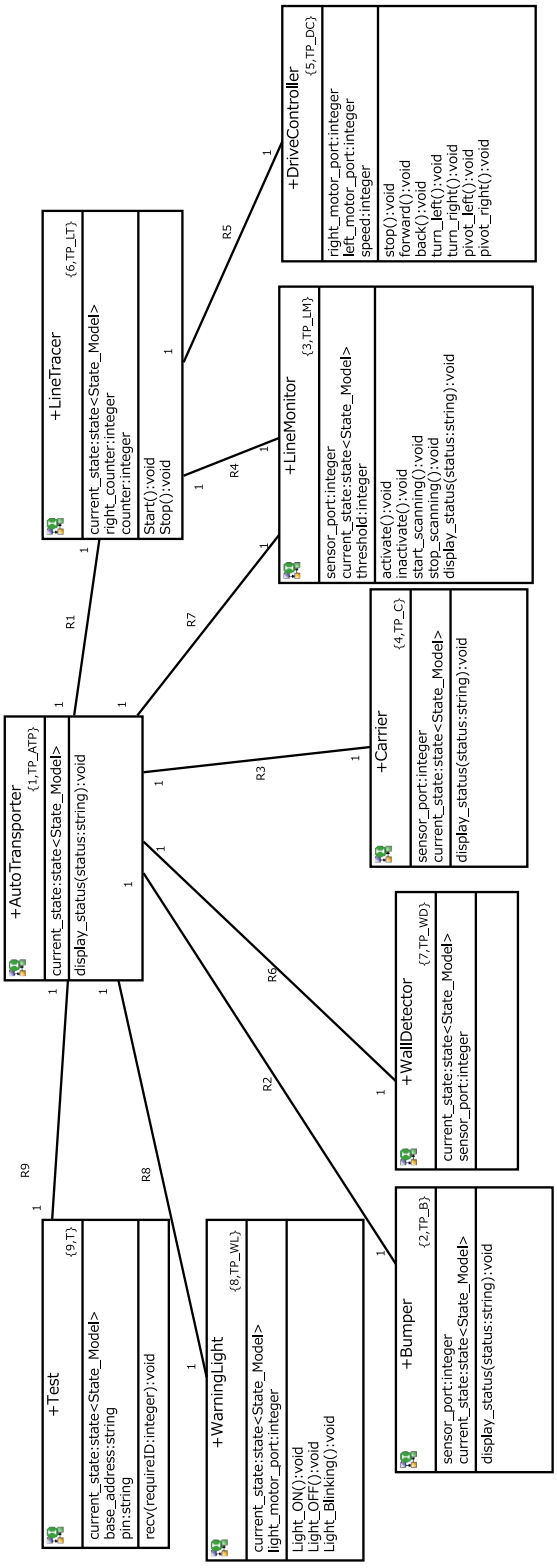


図 A.1　レビュー前のクラス図 A

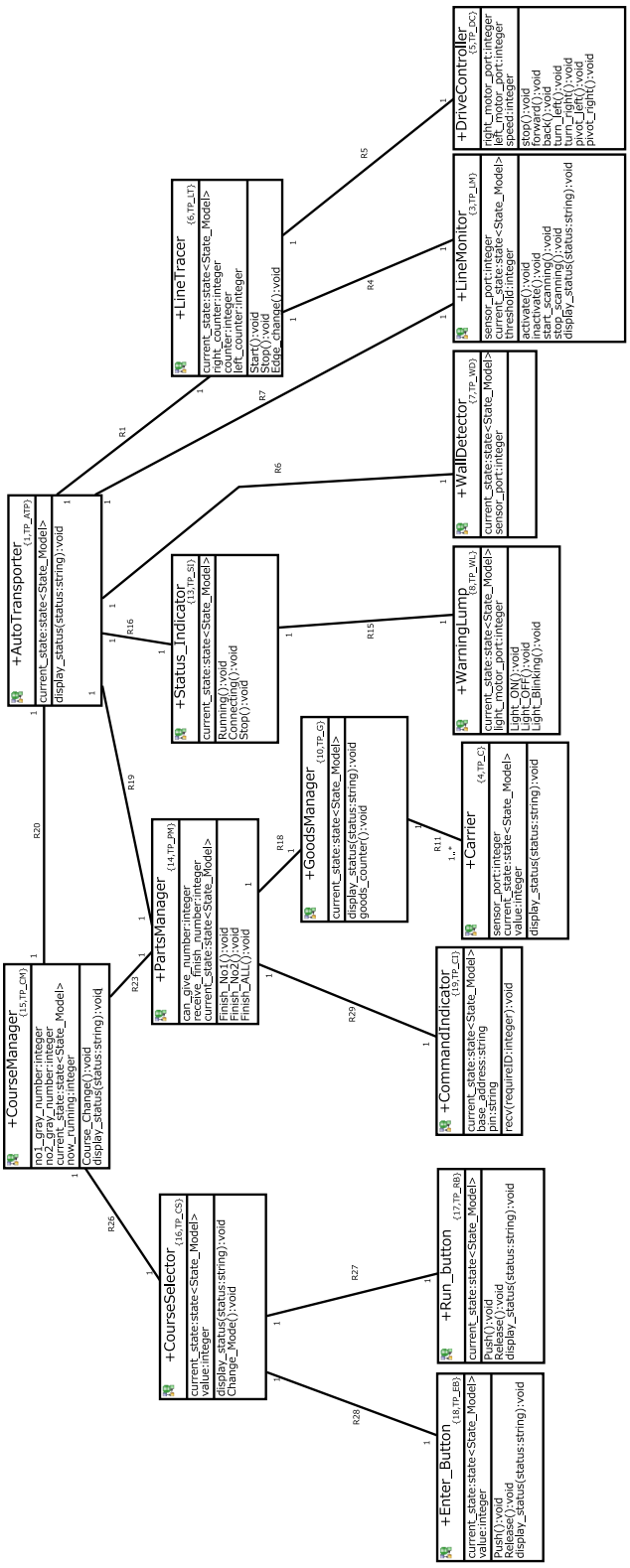


図 A.2　レビュー後のクラス図 A

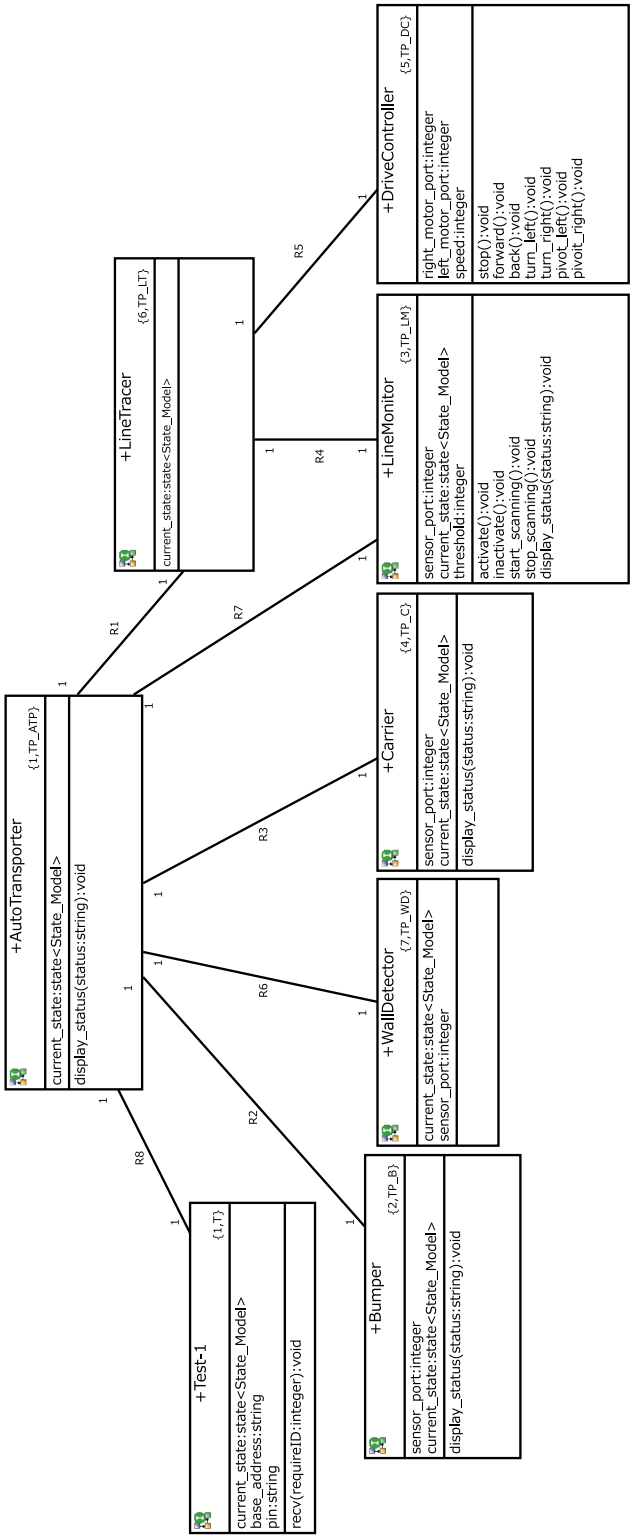


図 A.3　レビュー前のクラス図 B

図 A.4　レビュー後のクラス図 B

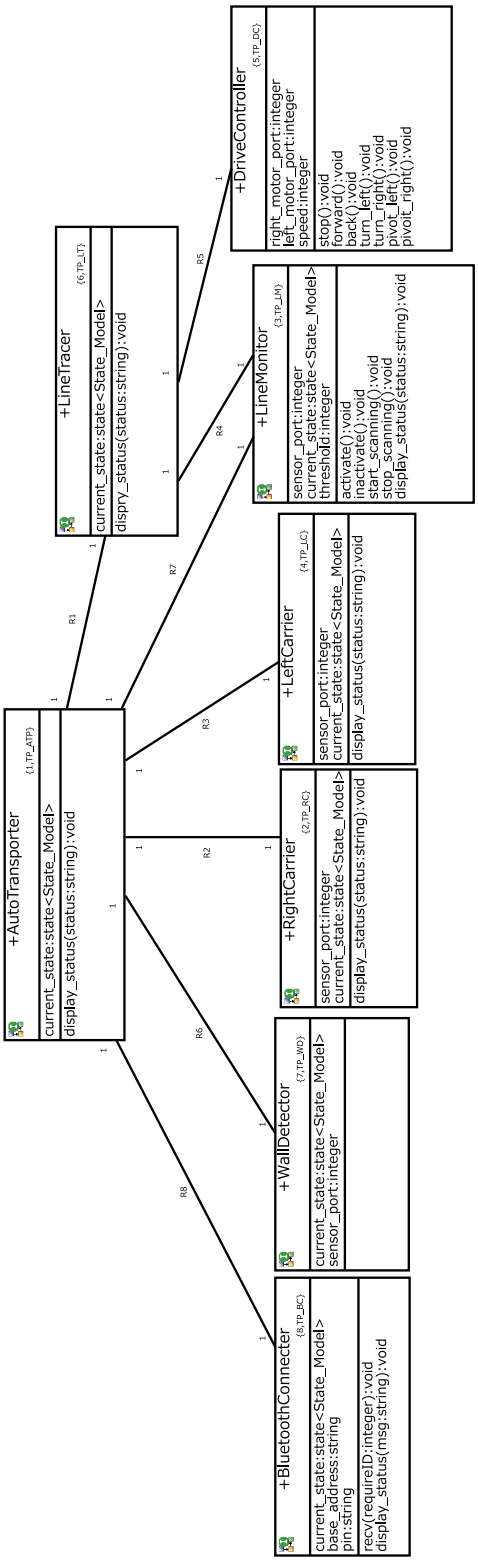


図 A.5　レビュー前のクラス図 C

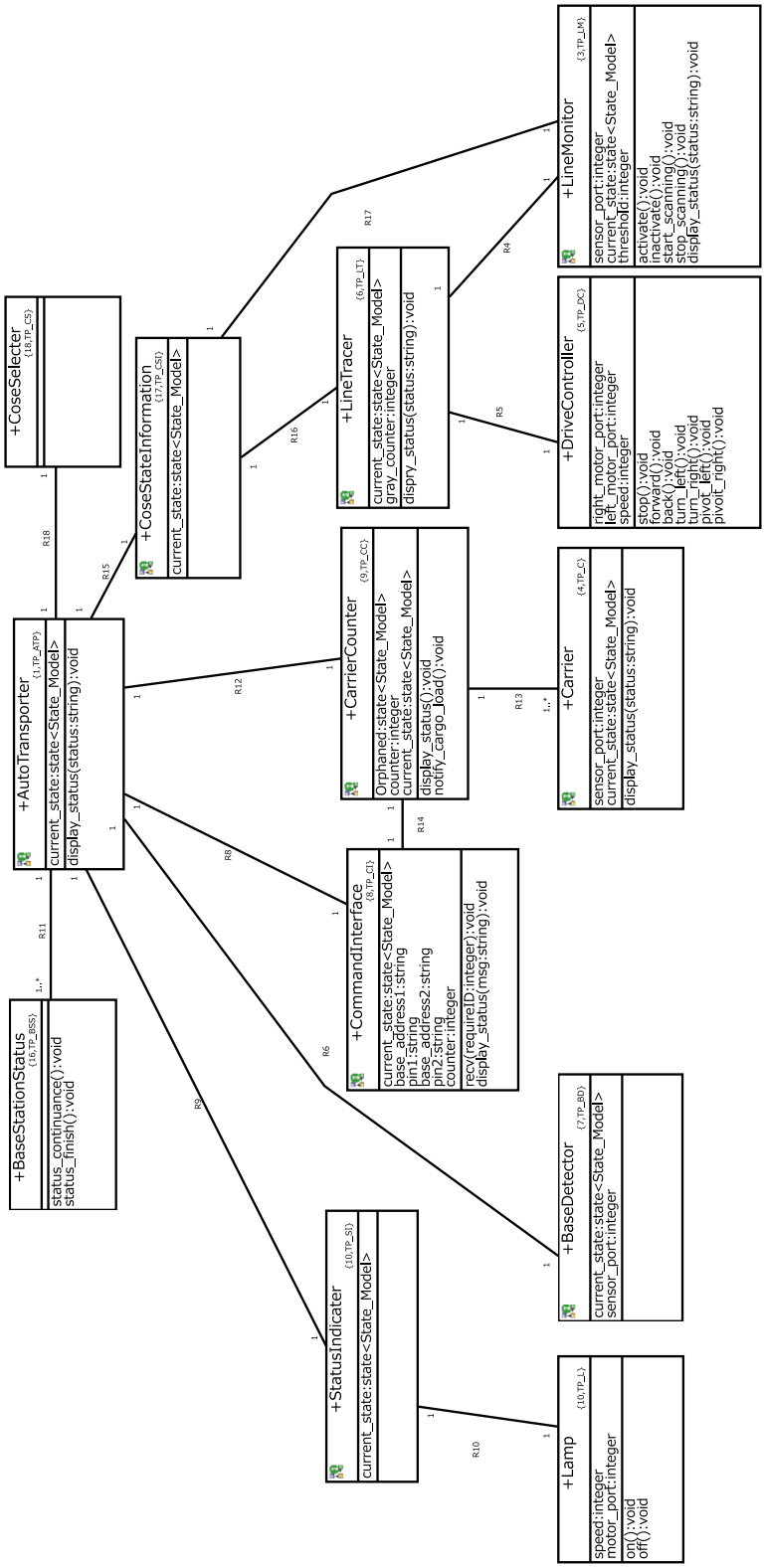


図 A.6　レビュー後のクラス図 C

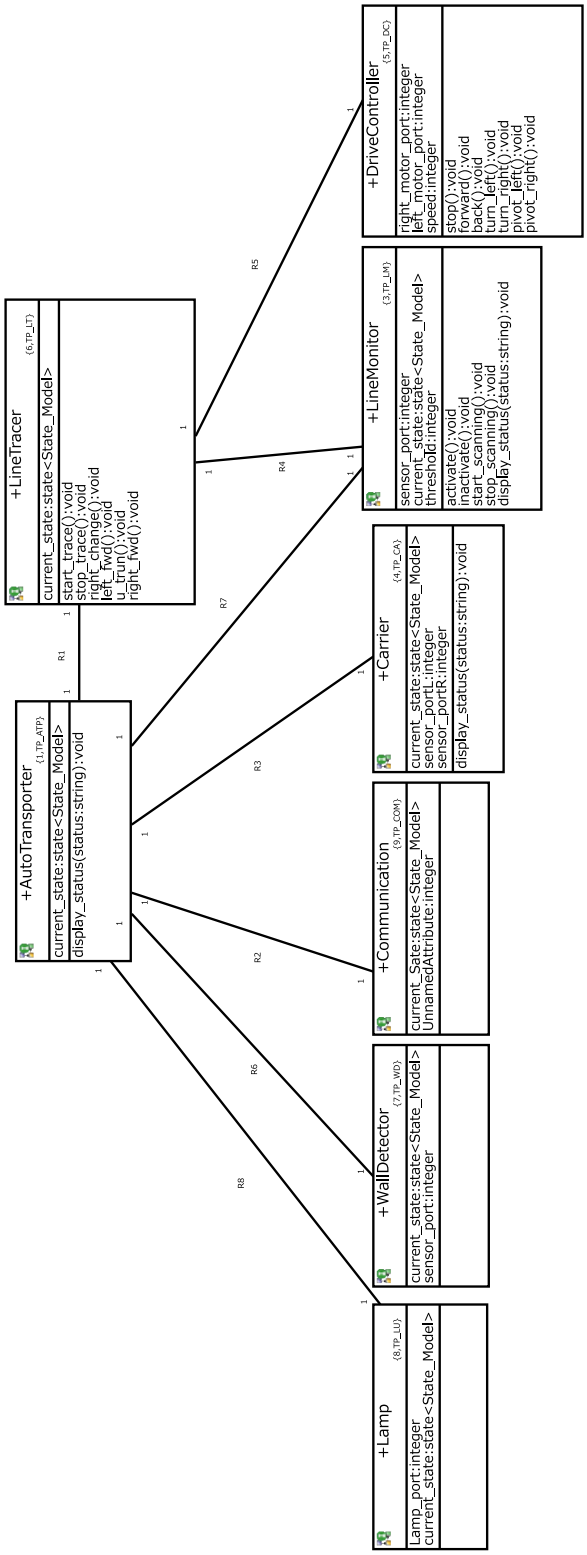


図 A.7　レビュー前のクラス図 D

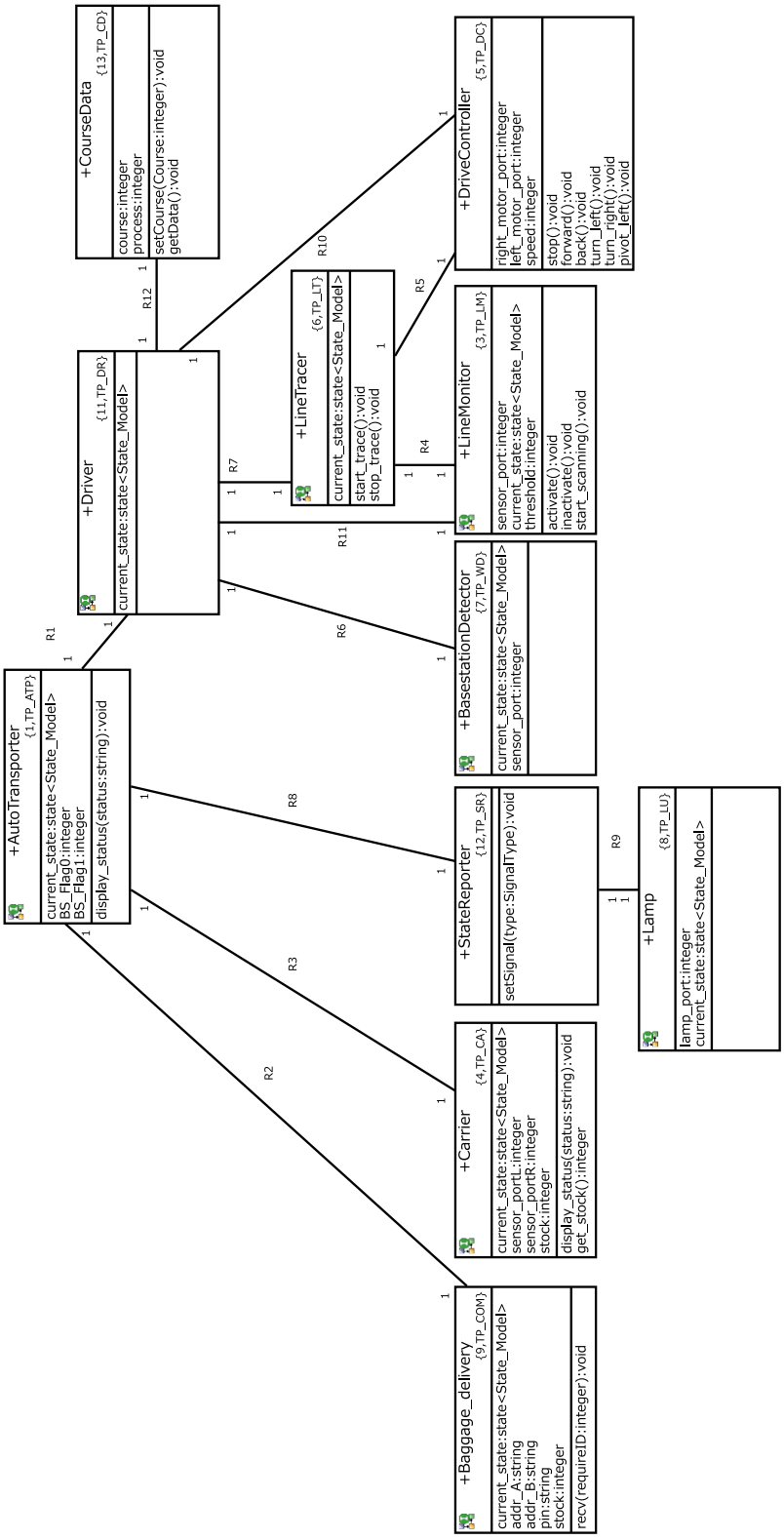


図 A.8　レビュー後のクラス図 D

図 A.9　レビュー前のクラス図 E

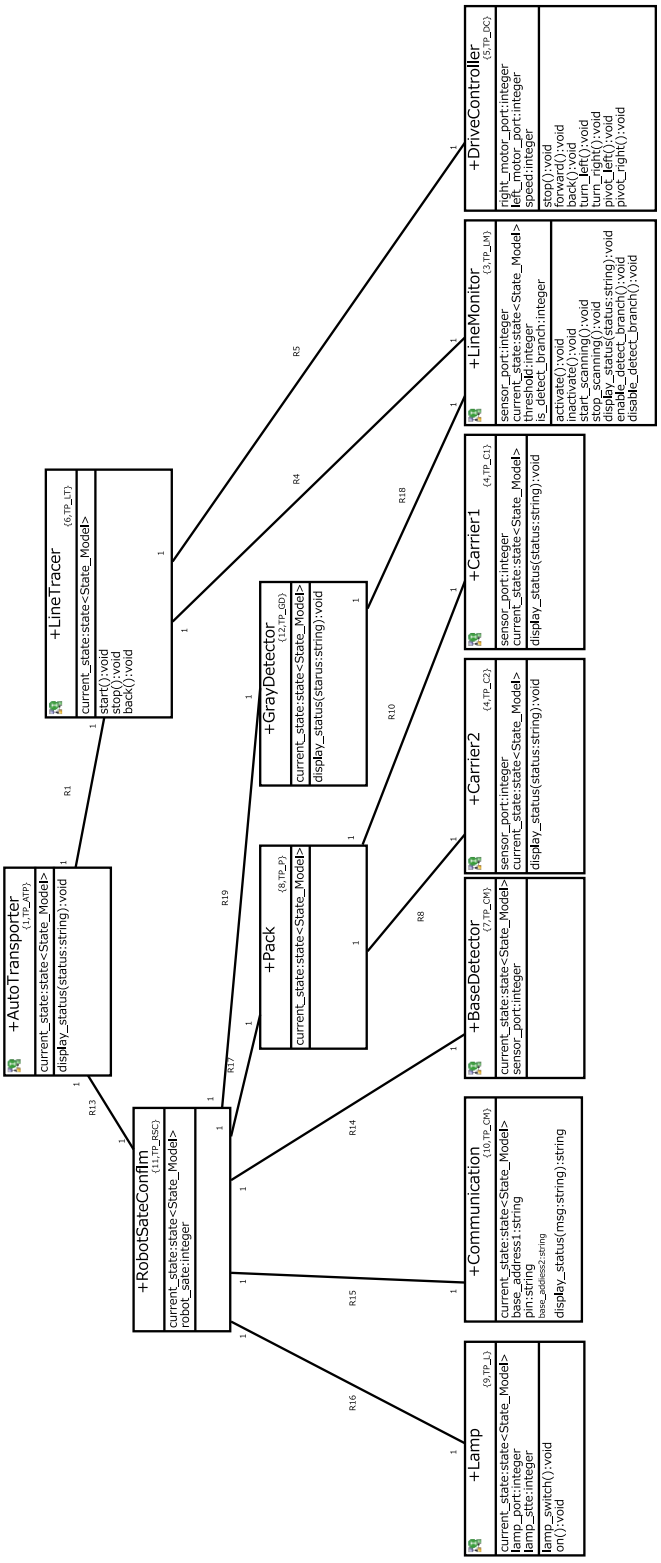


図 A.10　レビュー後のクラス図 E

付録 B - DSML 講座のステートマシン図

第 4 章の MDD の活用方法に関する実験の際の MDD なしグループのステートマシン図を図 B.1〜図 B.11 に，MDD ありグループのステートマシン図を図 B.12〜図 B.23 示す.

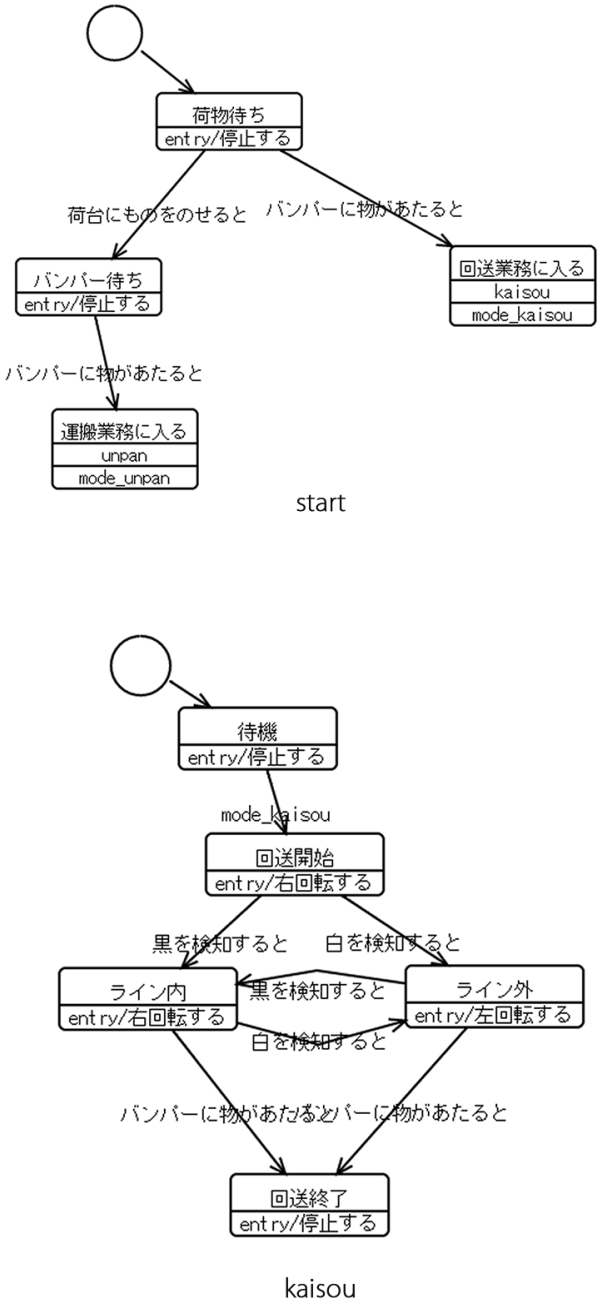


図 B.1　MDD なしグループ被験者 A のステートマシン図 1

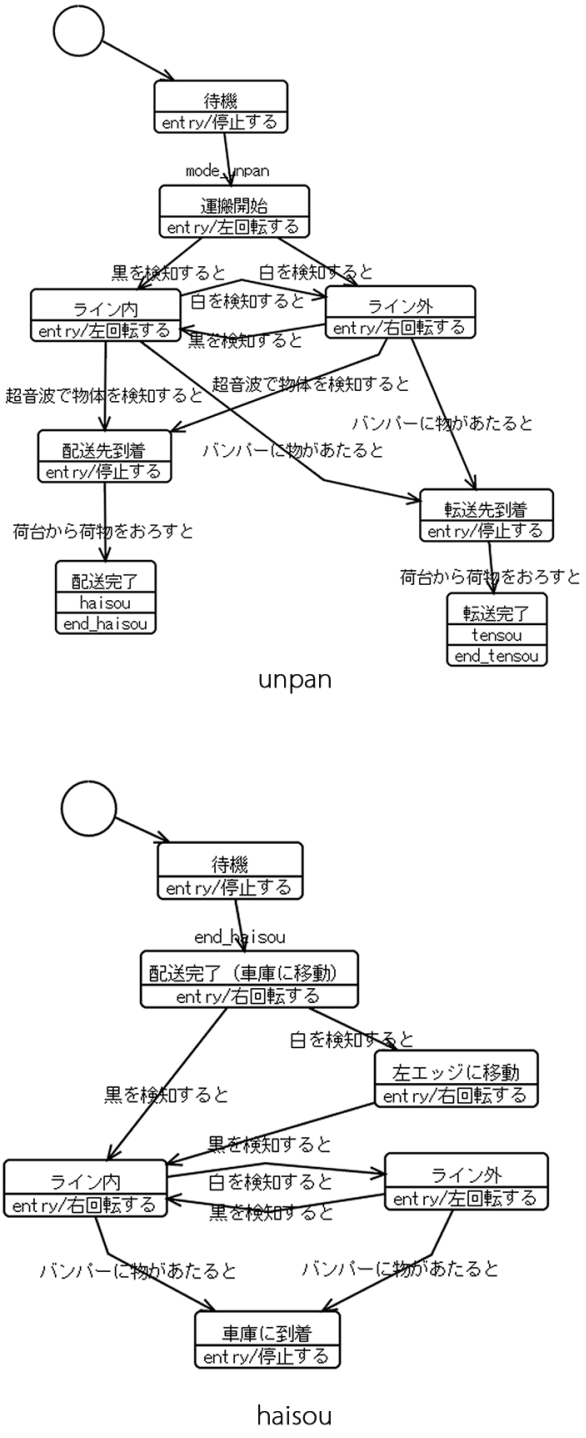


図 B.2　MDD なしグループ被験者 A のステートマシン図 2

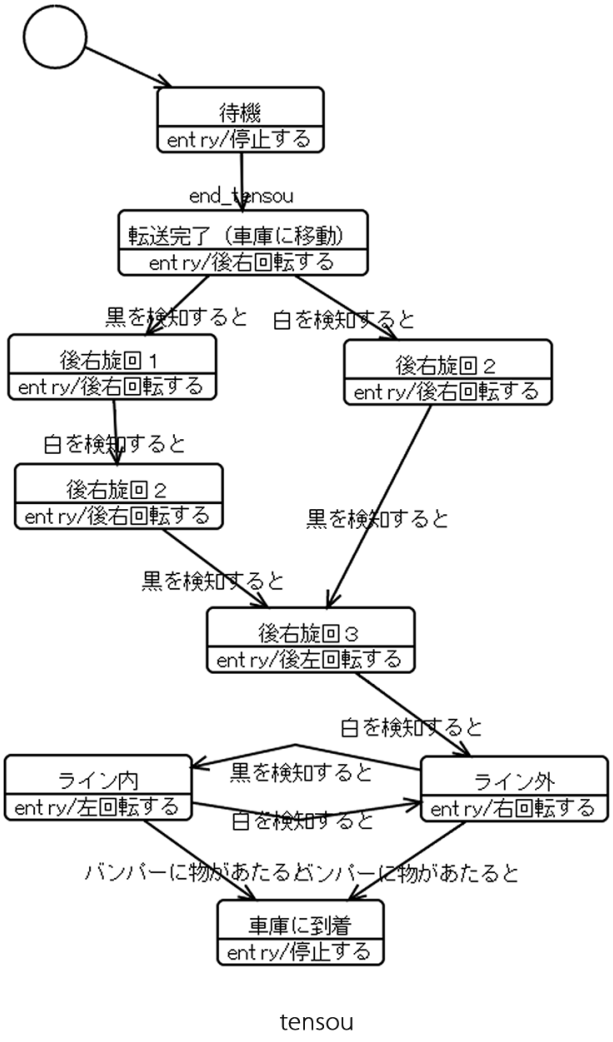


図 B.3　MDD なしグループ被験者 A のステートマシン図 3

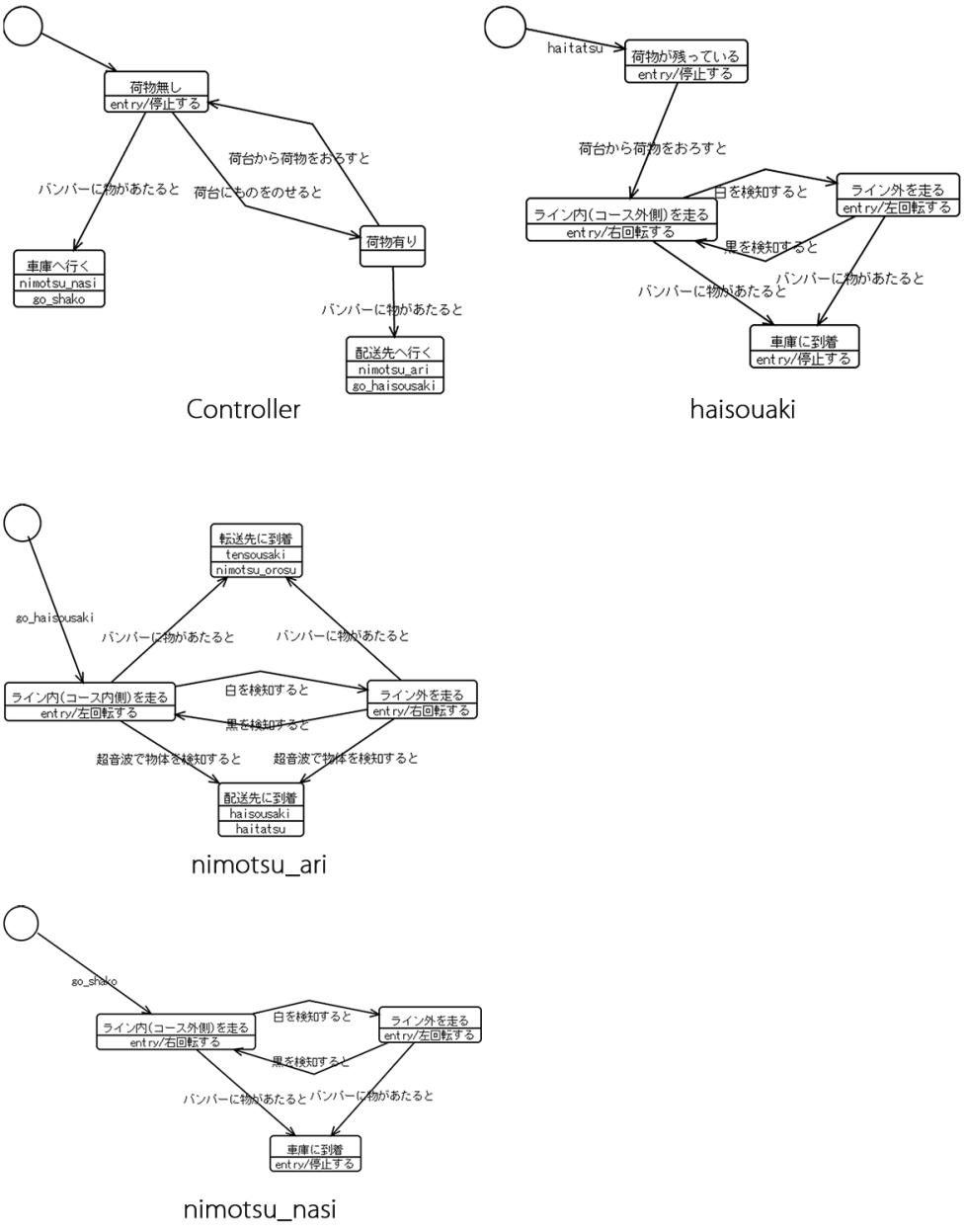


図 B.4　MDD なしグループ被験者 B のステートマシン図 1

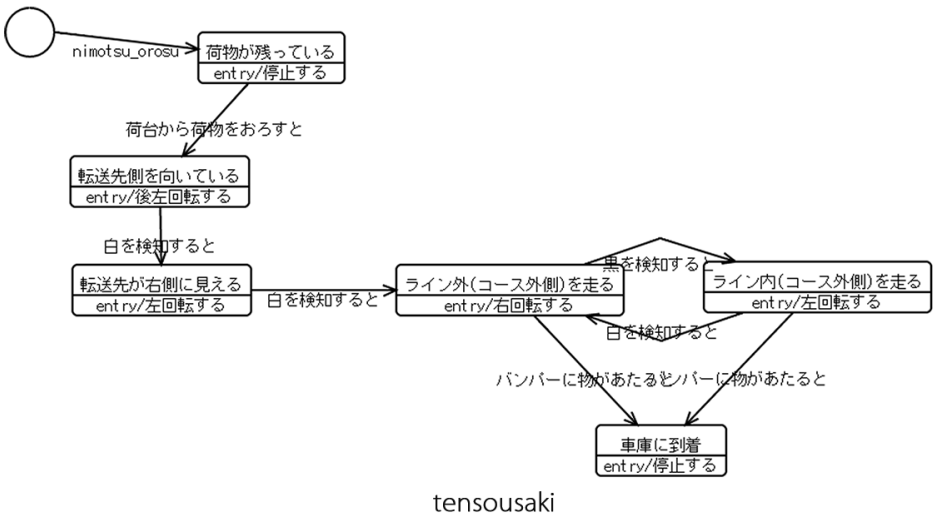
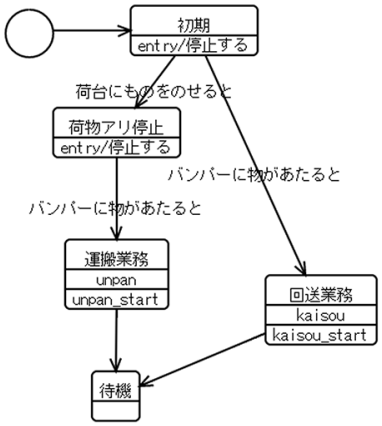
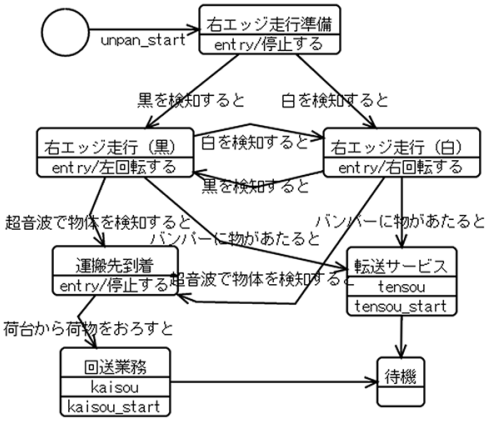


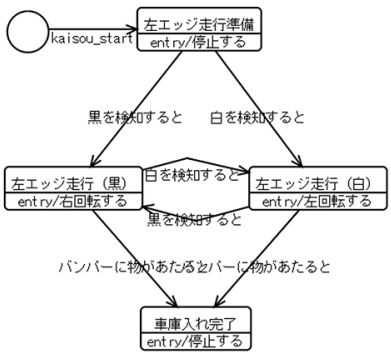
図 B.5　MDD なしグループ被験者 B のステートマシン図 2



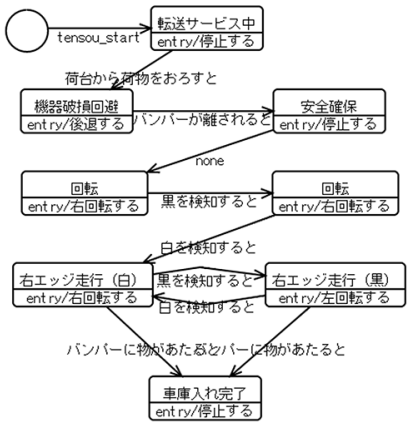
job_assign



unpan



kaisou



tensou

図 B.6　MDD なしグループ被験者 C のステートマシン図

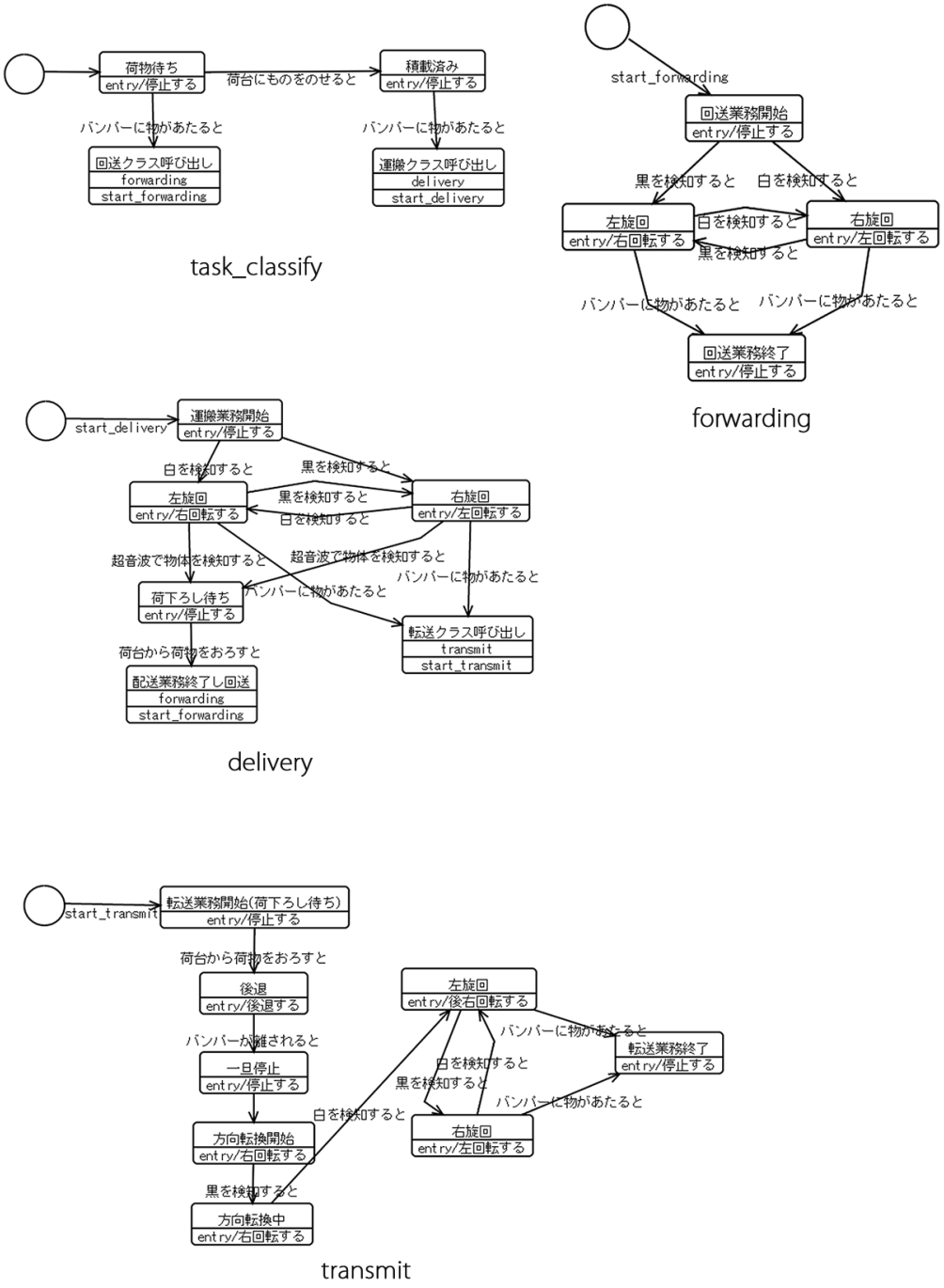


図 B.7　MDD なしグループ被験者 D のステートマシン図 1

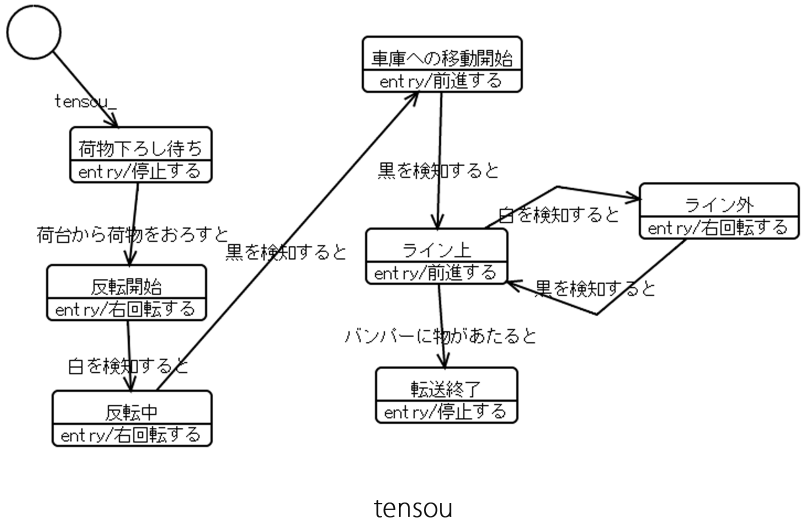


図 B.8　MDD なしグループ被験者 D のステートマシン図 2

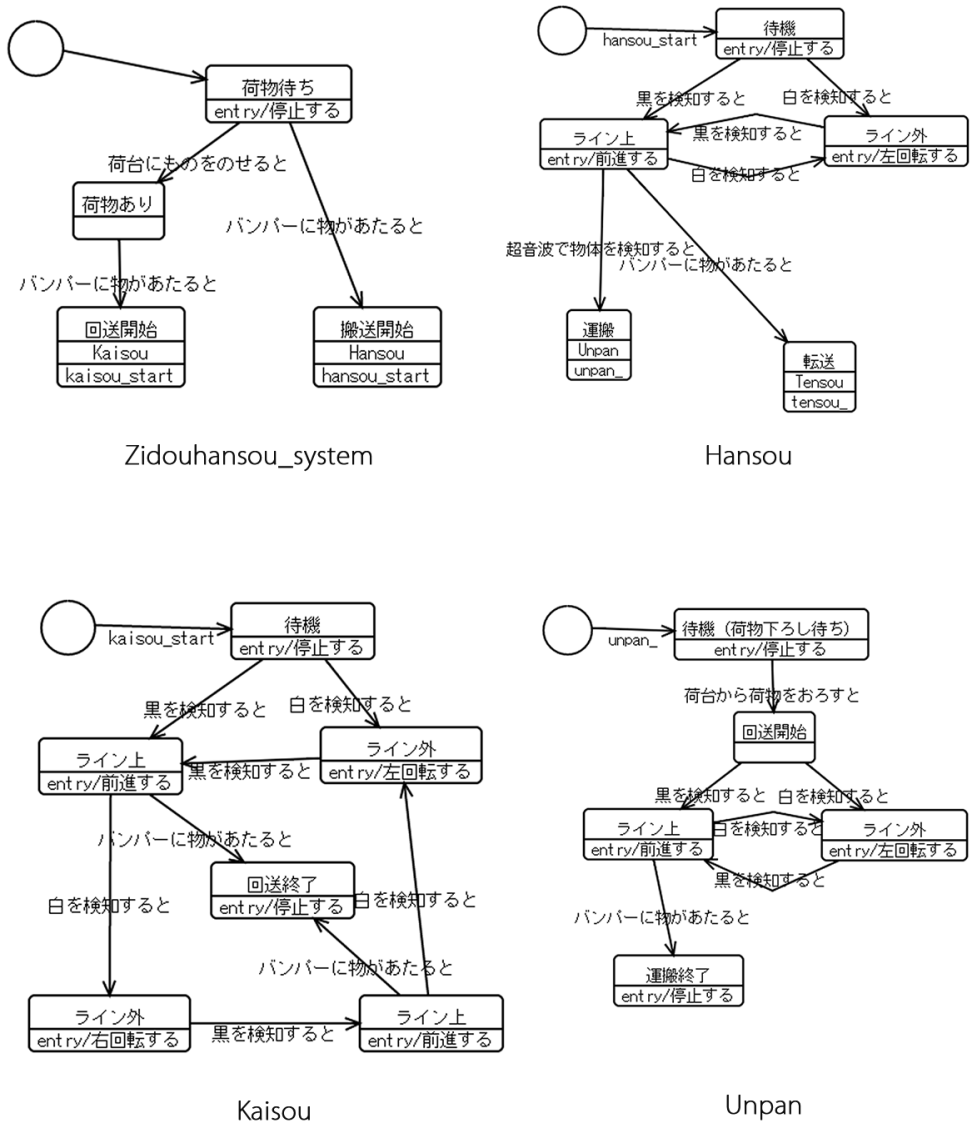


図 B.9　MDD なしグループ被験者 E のステートマシン図

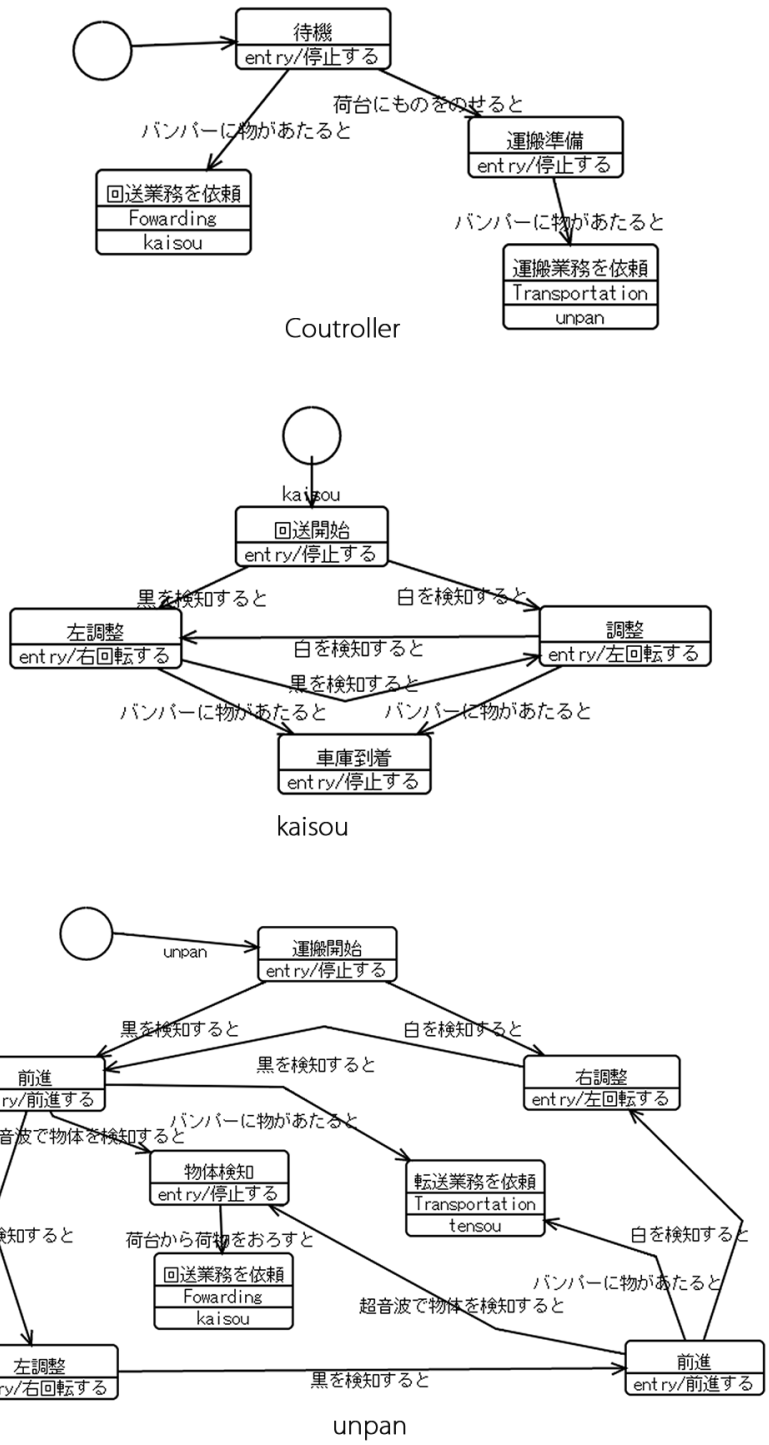


図 B.10　MDD なしグループ被験者 F のステートマシン図 1

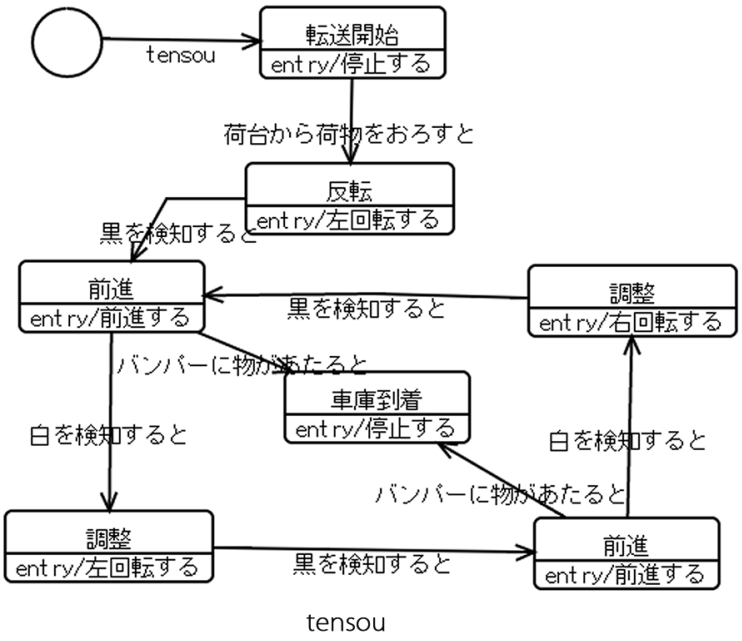


図 B.11　MDD なしグループ被験者 F のステートマシン図 2

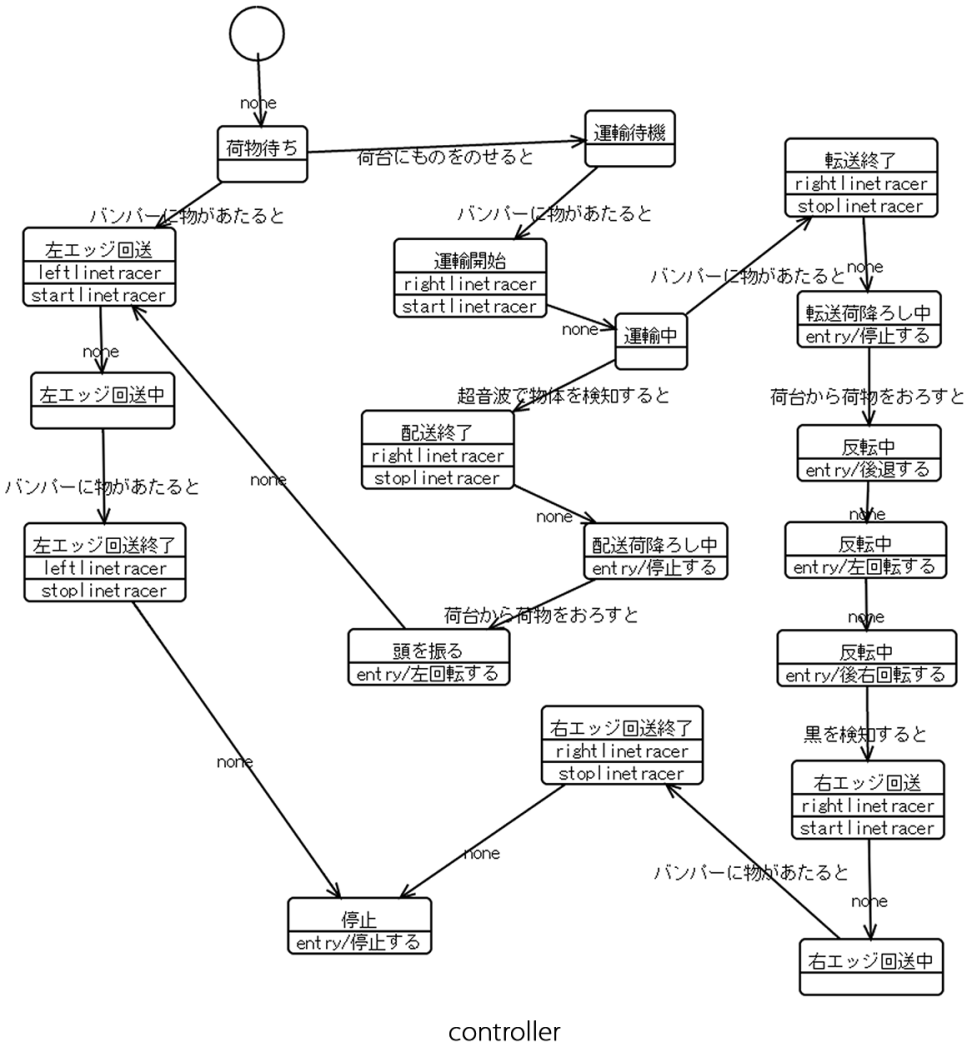
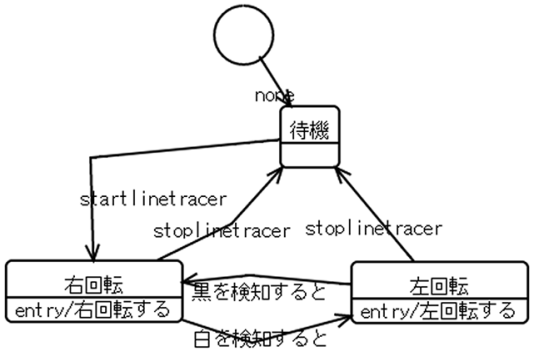
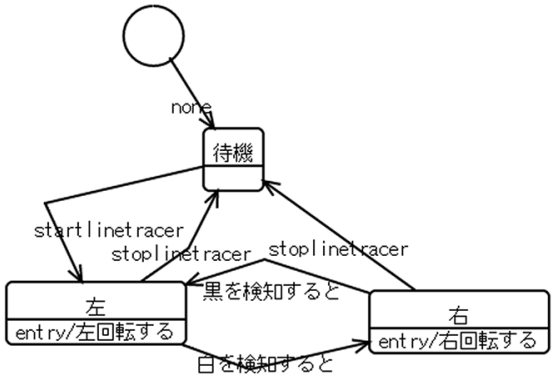


図 B.12　MDD ありグループ被験者 A のステートマシン図 1

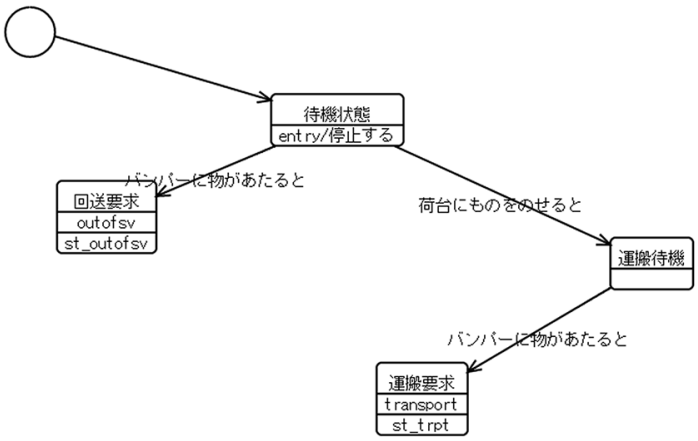


rightlinetracer



leftlinetracer

図 B.13　MDD ありグループ被験者 A のステートマシン図 2



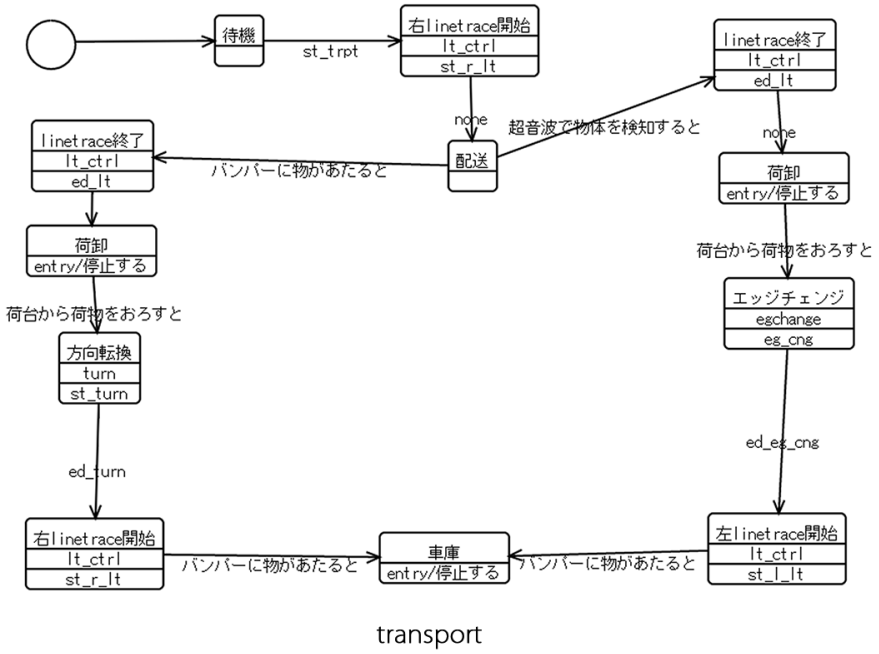
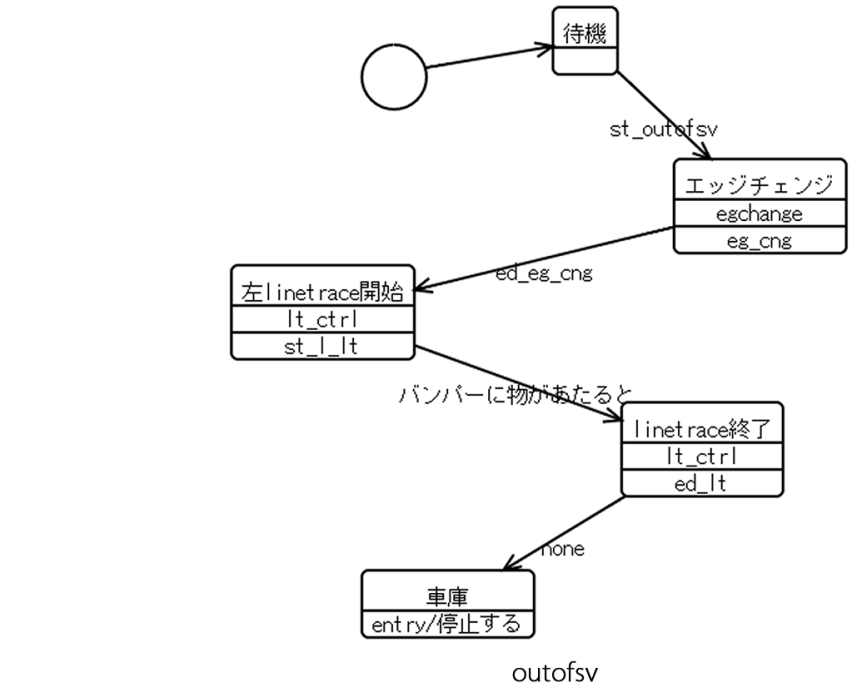


図 B.14　MDD ありグループ被験者 B のステートマシン図 1



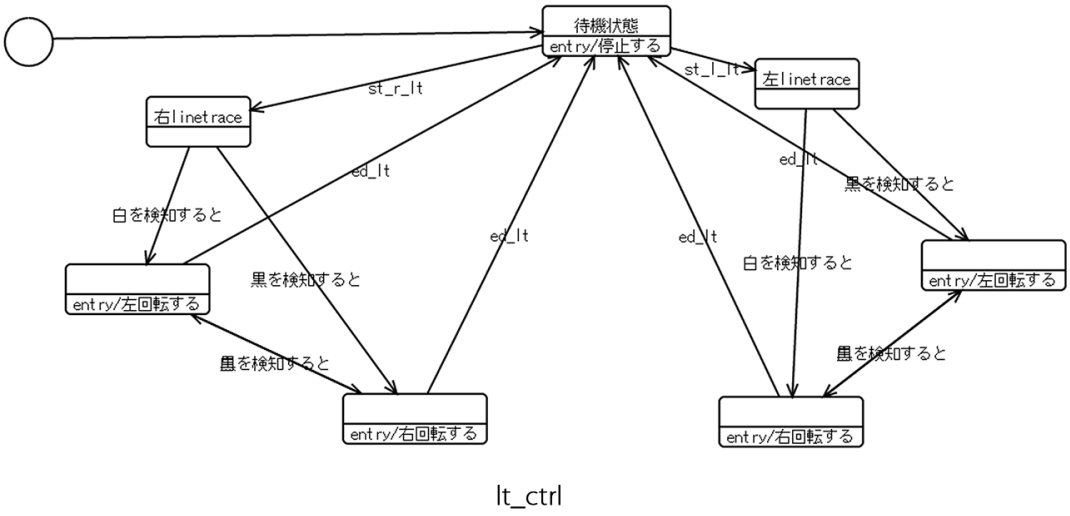
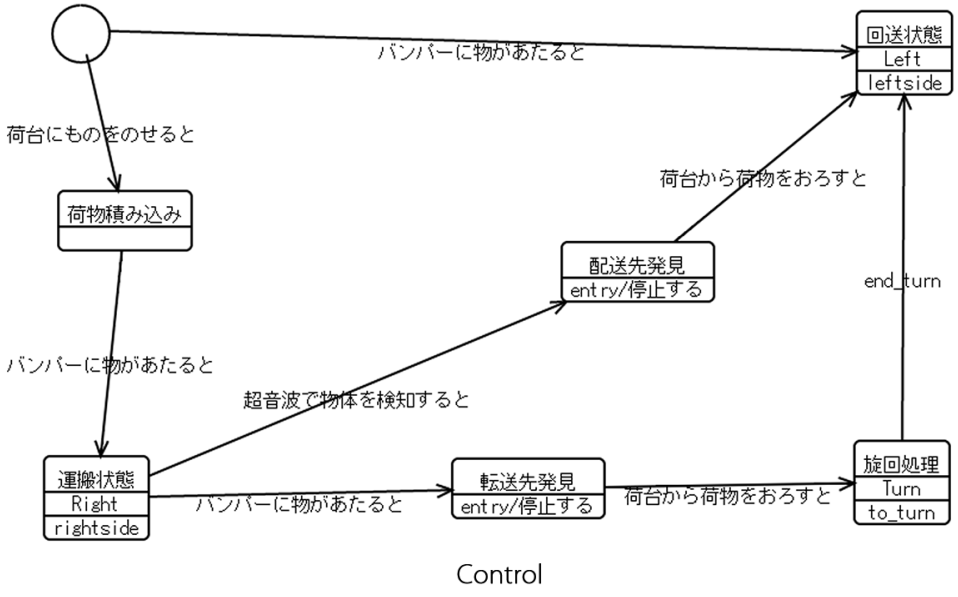


図 B.15　MDD ありグループ被験者 B のステートマシン図 2



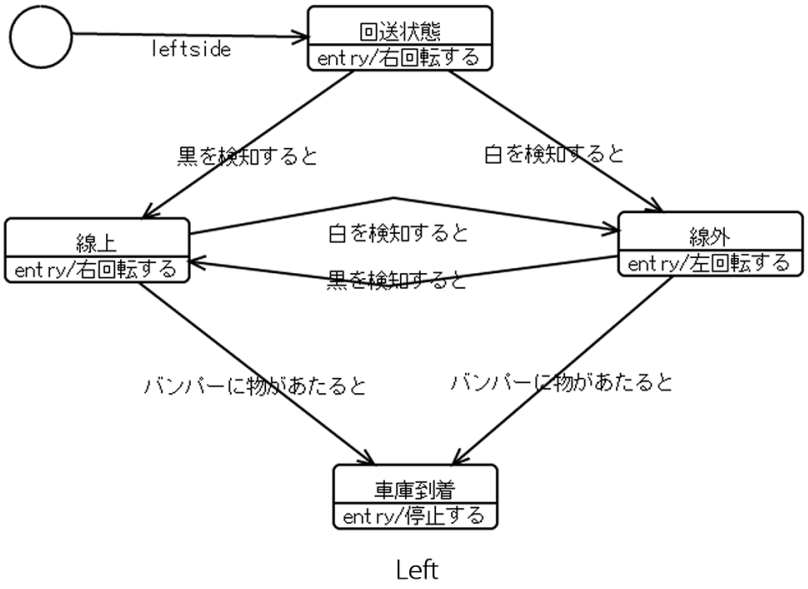
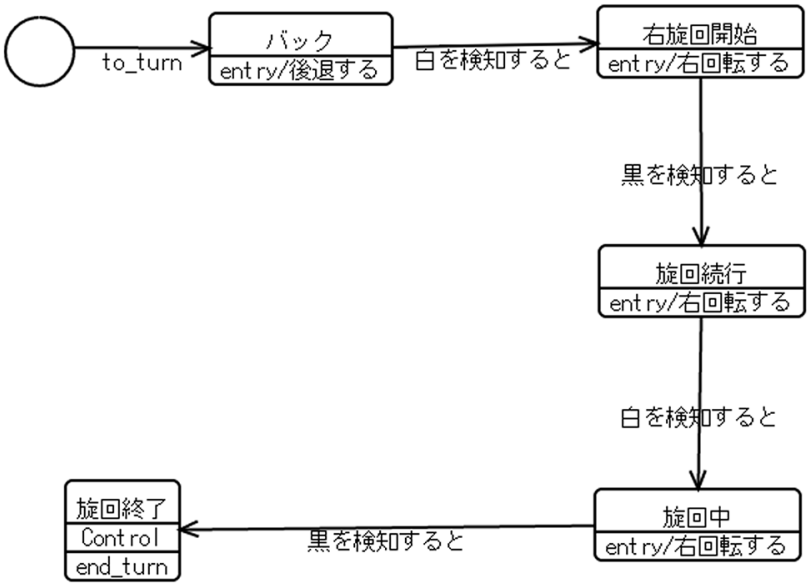


図 B.16　MDD ありグループ被験者 C のステートマシン図 1



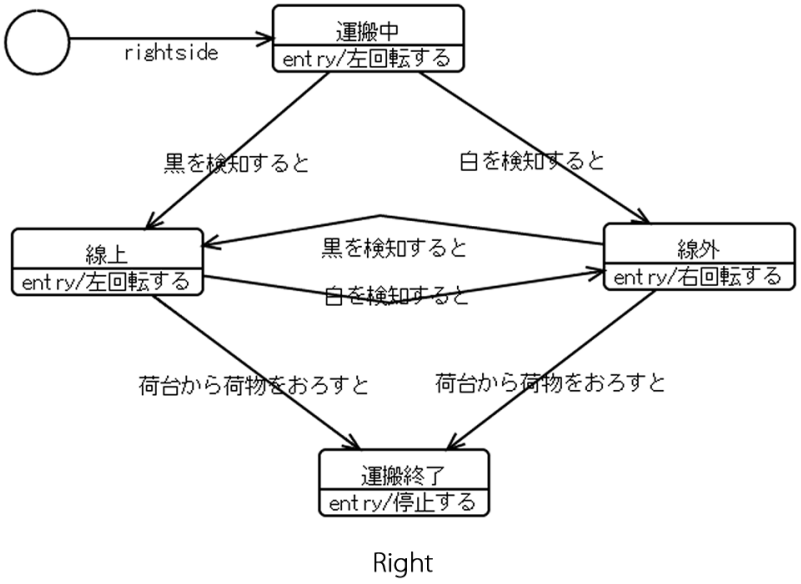
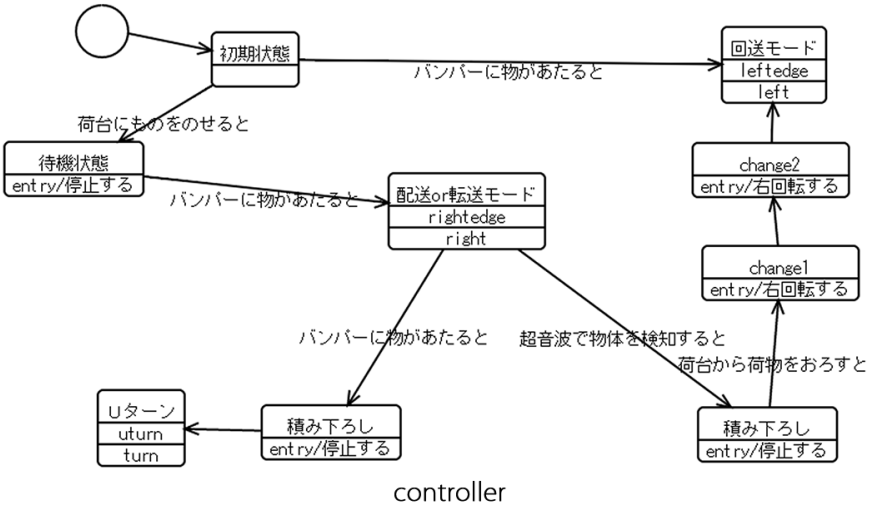
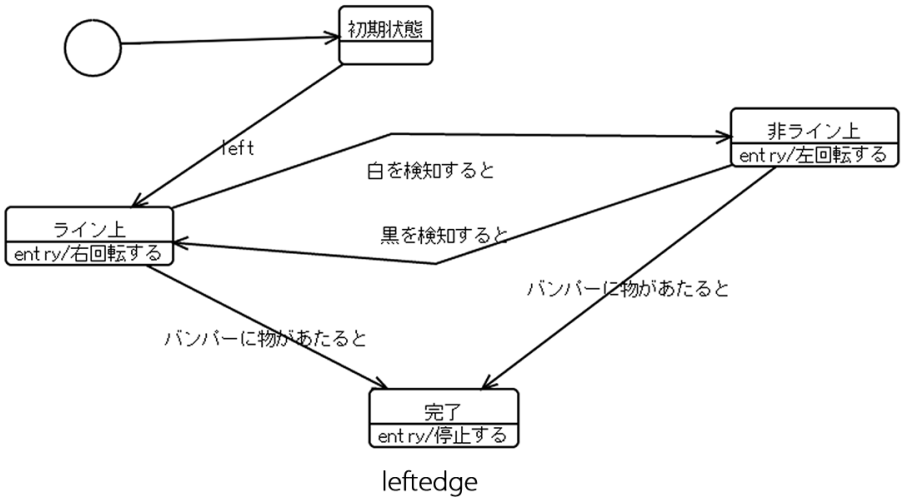


図 B.17　MDD ありグループ被験者 C のステートマシン図 2





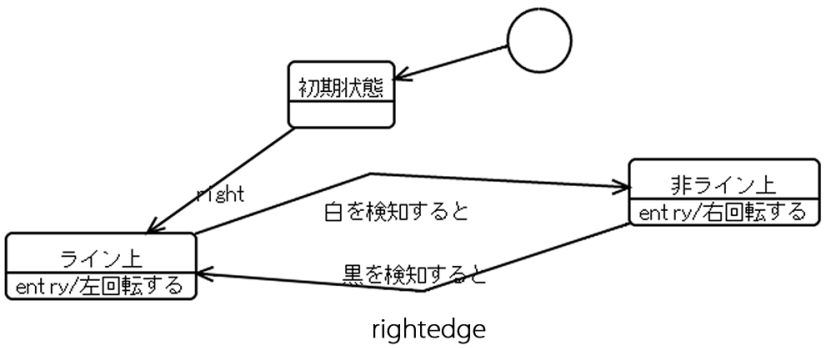


図 B.18　MDD ありグループ被験者 D のステートマシン図 1

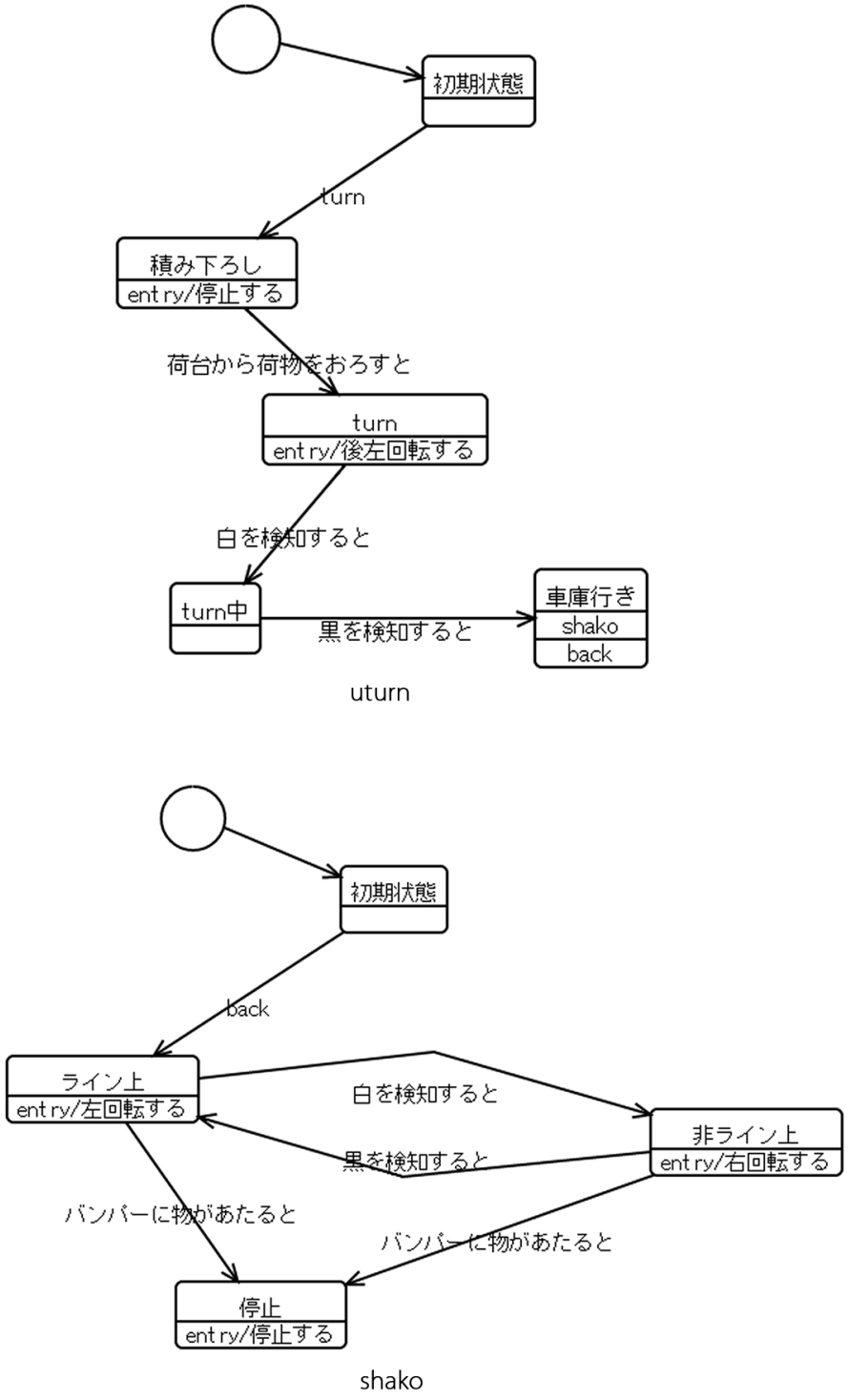


図 B.19　MDD ありグループ被験者 D のステートマシン図 2

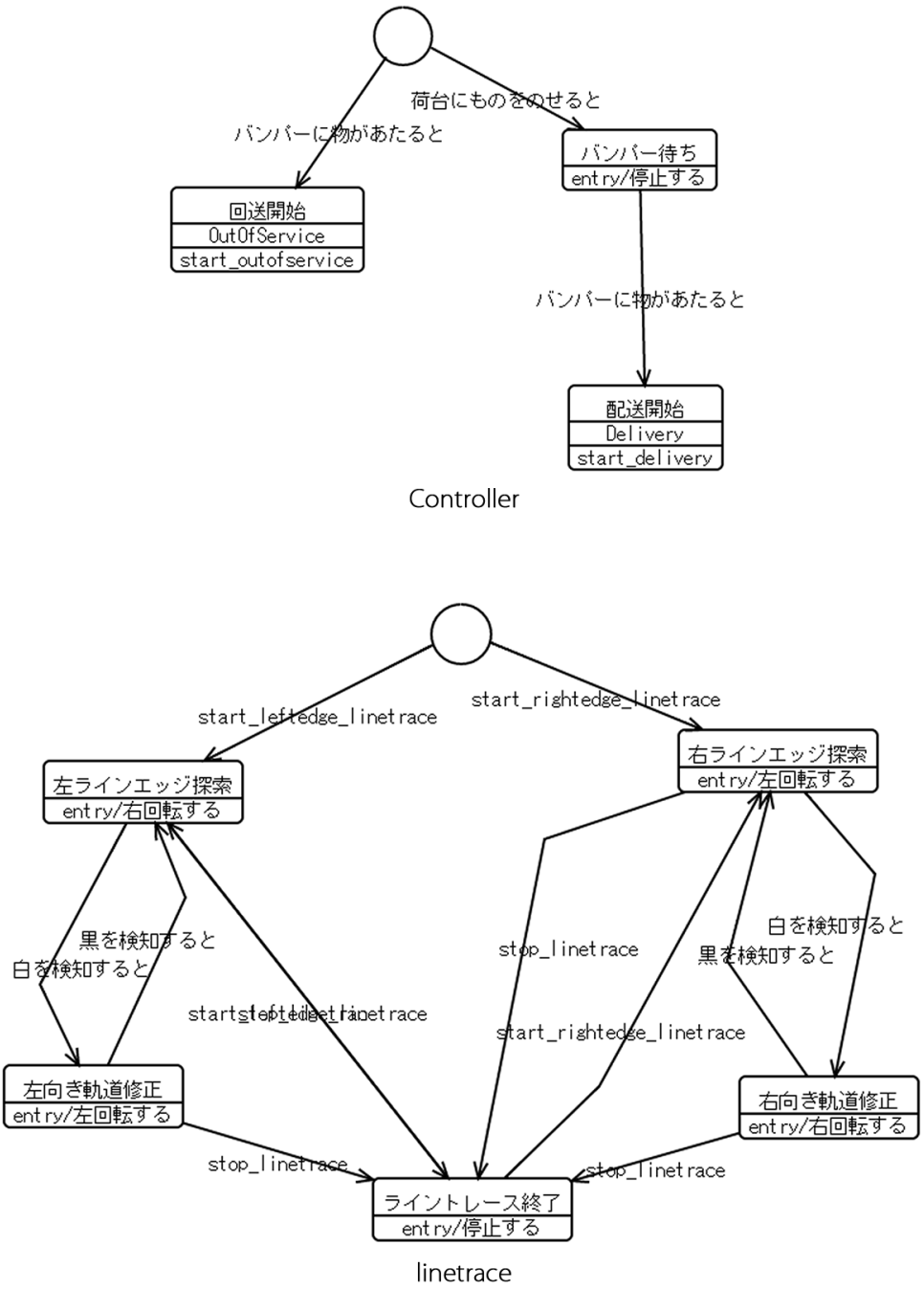


図 B.20　MDD ありグループ被験者 E のステートマシン図 1

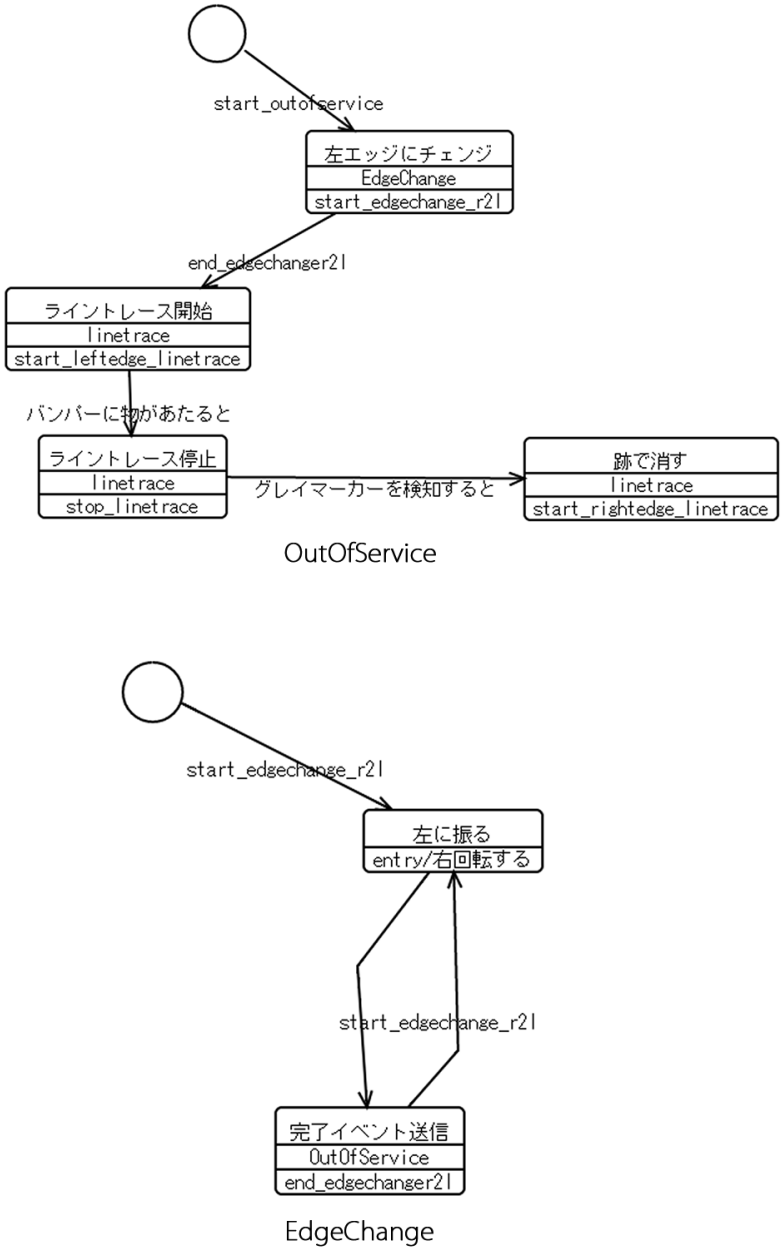
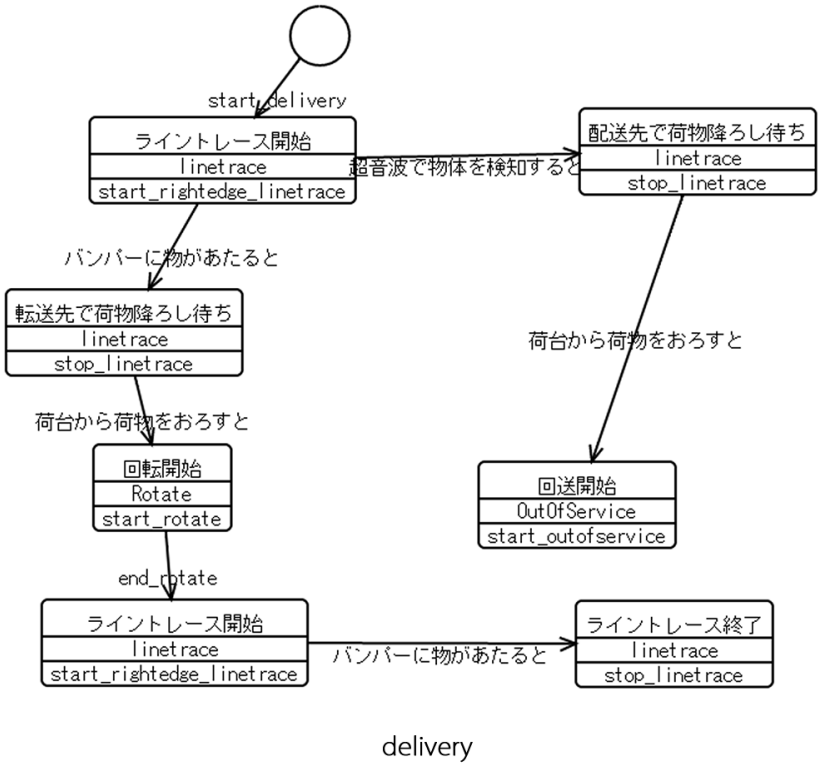


図 B.21　MDD ありグループ被験者 E のステートマシン図 2



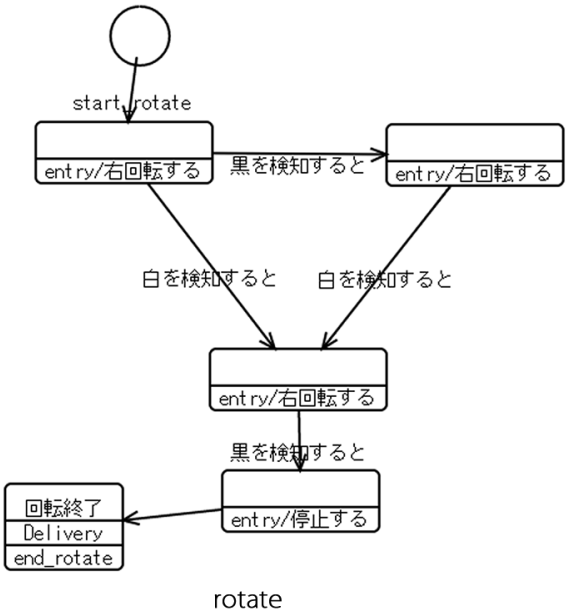


図 B.22　MDD ありグループ被験者 E のステートマシン図 3

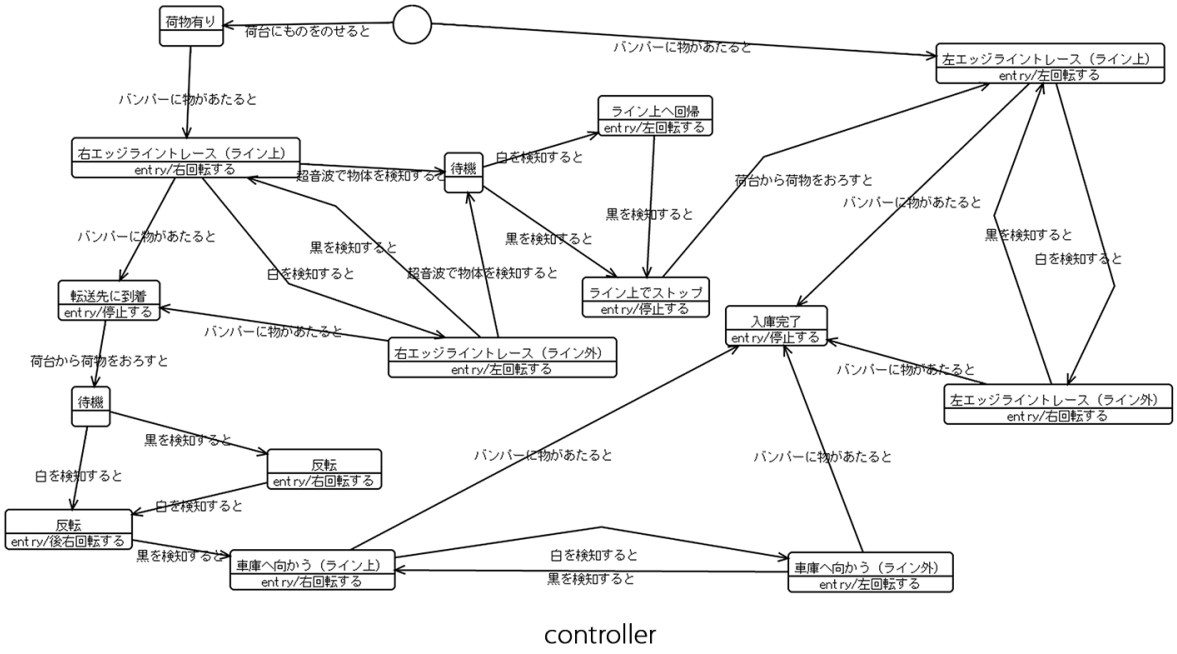


図 B.23　MDD ありグループ被験者 F のステートマシン図 1