Vol.2018-EMB-48 No.6
2018/6/29

ロボットを活用したモデル駆動開発教育の実践

久住憲嗣[†1]　久保秋真[†2]　細合晋太郎[†2]

概要：統一モデリング言語(Unified Modeling Language; UML)などのモデルを用いてソフトウェアの要求や設計を表現し、上流での検証やコード生成に活用する方法のひとつにモデル駆動開発(Model Driven Development; MDD)がある。MDD はソフトウェア開発の現場の一部においては活用されているものの、十分に普及している状況とは言えない。これは MDD が一般には理解されていない、知識としては知っていても導入方法が理解されていないためである。そこで本発表ではこれらの理解を促すための教育を提案する。本教育は主に修士の学生を対象とする。開発対象として掃除機型ロボットを採用し、ロボット制御を楽しみながら自然に MDD 関連技術を学ぶことができることを目指す。

キーワード：モデル駆動開発、ロボット制御、組込みシステム

1. はじめに

組込みソフトウェア開発現場からは開発効率と品質を向上させる開発技術が求められている。というのも、市場の要求を満たすためには、従来型の開発方法では対応しきれないほどの規模のソフトウェアを、高品質かつ短期に少人数で開発する必要があるからである。そのための開発方法の 1 つとしてモデル駆動開発（Model-Driven Development ; MDD）[1]がある。MDD において開発の初期段階での検証ができるようになるため、多くの品質問題を引き起こしていた開発の上流工程において問題の早期発見ができるようになる。さらには、詳細なモデルを記述することによりコードの生成ができるため生産性の向上やコーディング時の人為的ミスの削減を図ることができる。

MDD はソフトウェア開発の現場の一部においては活用されているものの、十分に普及している状況とは言えない。これは MDD が一般には理解されていない、知識としては知っていても導入方法が理解されていないためである。そこで本発表ではこれらの理解を促すための教育を提案する。本教育は主に修士の学生を対象とする。開発対象として掃除機型ロボットを採用し、ロボット制御を楽しみながら自然に MDD 関連技術を学ぶことができることを目指す。

本稿の構成は以下の通りである。2 節で MDD に関する教育の範囲と要求について議論する。3 節で教育内容について述べ、4 節で結言する。

2. 要求分析

2.1 技術に関するスコーピング

ソフトウェアの要求や設計を表現するモデルには様々な種類がありモデルベース開発(Model-Based Development; MBD)やモデル駆動開発(MDD)などと呼ばれる事が多い(図 1)。本稿では主に統一モデリング言語(Unified Modeling Language; UML)[2]などのグラフ形式でソフトウェアを表現する MDD について取り扱う。

MDD はさまざまな目的に使用される。開発の上流工程において、ソフトウェアの要求や設計などをモデル化し、目視によるレビューやシミュレーションによるテストに活用する。もしくは、設計図をもとにコード生成する、テストケースを生成するなどである。また、目的に応じて記述するモデルの抽象度を選択する。コミュニケーションのためにモデルを活用する場合は抽象度が高いモデルを記述するが、設計図として使用する場合にはさらに詳細なモデルを記述する必要があり、さらに検証やコード生成を行う場合にはかなり詳細なモデルを記述する。MDD においては、開発対象の性質に応じて使用するモデルの種類が選択される。ソフトウェアアーキテクチャの検討には UML クラス図など、離散的な振る舞いの表現には UML ステートマシン図が用いられる。

また、ランカスター大学の調査[3]などでは、既定のモデルを用いるだけでは生産性の向上が十分ではなく、社内、プロジェクト内などで独自のモデリング言語（Domain-Specific Modeling Language ; DSML）を定義し、活用する組織が多いことを指摘している。既存のモデリングと比較して、その問題領域を少ない工数で効率よく記述することができるためである。

本講義においてもこれらの項目を取り扱いたい。すなわち、UML を設計図として用いたり、プログラミング言語として用いたりしたい。また、既存の UML を用いるだけではなく、必要に応じて UML を拡張して使用したり、ドメイン特化モデリング言語を定義し記述できるようにしたりしたい。

2.2 想定受講生

教育対象は修士 1 年生を想定する。それまでに、受講生はプログラミング言語を理解し、オブジェクト指向言語の考え方をある程度は理解している。また、UML もすべての図について精通している必要はないが、クラス図、オブジ

†1 九州大学
Kyushu University
†2 チェンジビジョン(株)
Change Vision, Inc.

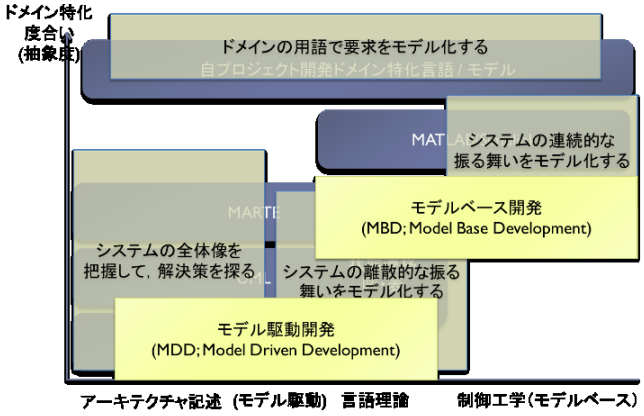

図 1 ソフトウェアを表現するモデルの種類

図 2 掃除機型ロボット Kobuki

ェクト図、ステートマシン図の文法を理解しており、簡単な対象であれば記述できる運用能力が求められる。

3. 授業の設計

本節では授業の設計について述べる。まず、教育目標について整理し、次に授業計画について述べる。

3.1 教育目標

本講義では MDD を実践する上で必要な知識と、基本的なスキルを身につけることを目標とする。具体的にはモデリング手法として xtUML (eXecutable Translatable UML)[4]を採用する。xtUML はその名の通り実行、変換可能な UML の記述方法、及び、方法論である。xtUML に従ってモデリングすることによって、上流でのシミュレーション検証やコード生成が可能になる。

受講生は xtUML におけるモデリング手法や、モデルを用いた上流での検証方法、モデルからのソースコードの自動生成のしくみ、生成されたコードと既存コードの連携方法について理解することを目標とする。

さらに、実際にツールを用いて簡単な題材において xtUML を実践することができることを目標とする。

3.2 演習環境

ET ロボットコンテストや ESS ロボットチャレンジに見られるように、ソフトウェア開発技術を学ぶ演習題材としてロボットを採用することで、受講生にとって魅力的な演習とすることができる。そこで、本講義においてもロボットの制御を題材として採用した。図 2 に演習教材として採用した Kobuki[5]を示す。

Kobuki は掃除機能をもたないが一般的な自動掃除ロボット型のロボットである。衝突検知のためのバンパセンサ、段差に落ちないための段差センサ、タイヤの回転角度を計測するエンコーダなどのセンサを持ち、左右 2 つのタイヤを駆動するモータを持つ。さらに、拡張が容易な構造になっており、デバイスを追加させる演習などにも対応できる。

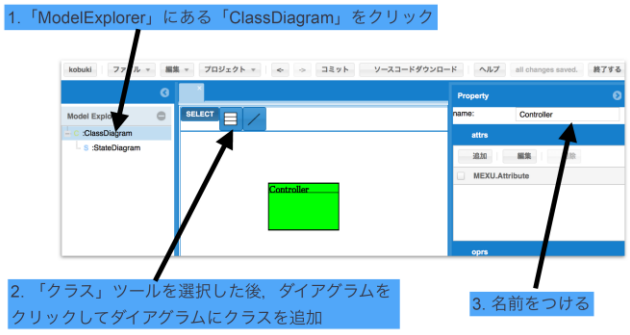

図 3 モデリングツール clooca

3.3 ツール

本講義ではモデル駆動開発を実施するツールとして clooca[6]を採用する。clooca は Web ブラウザ上で動作するドメイン特化モデリングツールである。インストール不要で Web ブラウザからサイトにアクセスするだけで使用できる。そのため、講義を準備する教員にとって頭の痛い、開発環境の配布が容易になる。本講義では clooca 上で動作する xtUML の簡易版開発環境を開発し、それを講義で用いる。

3.4 授業計画

次に授業計画について述べる。

(1) **概論**

初回は MDD について概説する。ソフトウェア開発方法論の必要性をコスト、品質、納期の面で説明し、特にモデルを使った開発方法を導入することによる開発の進め方の違いを説明する。また、受講生に意識付けをするために、企業における MDD の導入事例をいくつか紹介し、生産性や品質の向上に実際に資することを説明する。また、要求分析で述べたようなモデルを使った様々な手法について紹介し、本講義の範囲は UML を用いた MDD とするが、実際には目的に応じて適材適所で道具を使い分ける必要があることを意識付ける。

(2) **モデリング手法**

次に xtUML を用いたモデリング手法について解説する。

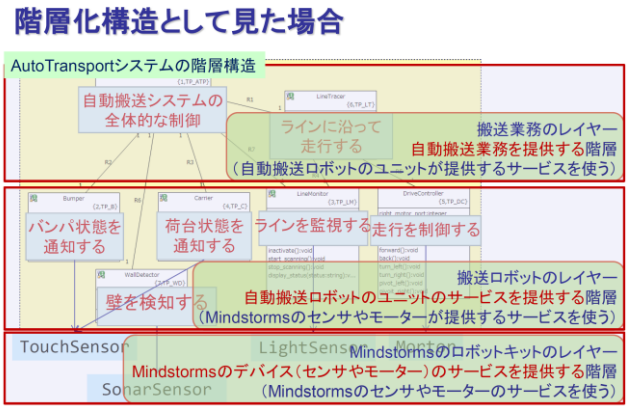

図 4 ドメイン分割の例

xtUML では構造をクラス図、振る舞いをステートマシン図及びアクション言語で与える。また、ステートマシンはイベントを送受信することができ、オブジェクト間で情報のやり取りができる。また、ライブラリ関数の呼び出し、外部デバイスからのイベントの受信ができる。これらのモデルの構成、及び、文法について説明する。

(3) モデリング演習

次に講義した xtUML のモデリング手法を実際に体験してもらうべく、ロボットを使用した開発を行う。さらに、モデル駆動開発を活用する際の設計内容について議論する。

著者らは 0 から xtUML を導入する場合と、すでに導入がなされた状態でモデルを記述するのとでは、開発の困難さが根本的に異なると考えている。そのため、まずはあらかじめ基本となるモデルが設定され準備された状態でモデリングをするところから始める。

まずは、外部入力に応じて回転するなどの、単純で具体的な演習から始め、徐々に複雑な制御を実施する演習にする。演習の詳細は後で述べる。

(4) 関心事の分離演習

次に、より複雑なシステムにモデル駆動開発を適用するための第 1 歩として、関心事を分離する方法について講義と演習を通して理解する。具体的には、あらかじめ与えられたモデルに、複雑な業務を遂行するためのモデルを段階的に追加させる。その過程で関心事の分離の重要性を体験させ、理解させる。

(5) ドメイン分割演習

関心事の分離をさらに進めて、大規模なシステムを構築する上で必要なドメイン分割の考え方を理解する。具体的にはデバイスとアプリケーションのドメイン分割の考え方や、ドメイン分割と階層化アーキテクチャ、ドメイン間インターフェースの役割について講義した後、ドメイン分割とインターフェース設計の演習を行う。

(6) モデル駆動開発の仕組み

次に、モデル駆動開発の背景にある仕組みを理解する。具体的には、メタモデルについて説明してモデルの表現方法、定義方法について理解する。次にメタモデルのインスタンスとして表現されたモデルを用いて、他のモデルに変換する方法や、コードへの変換方法を理解する。

さらに、今回使用しているツールでの、メタモデル、及び、コードへの変換の仕組みをウォークスルーすることで理解を深める。

(7) モデル駆動開発とコードの協調実行演習

モデル駆動開発においては、すべてをモデルで記述できるわけではなく、オペレーティングシステムや外部デバイスの機能を使用する必要がある。それらの機能をモデルで記述されたシステムから利用するための方法について学ぶ。具体的には既存のロボットに新しいデバイスや機能を追加させ、外部のシステムの呼び出し方法や、逆に外部からの処理の依頼の受付方法を体験する。

(8) モデルコンパイラカスタマイズ演習

次にモデルコンパイラの仕組みを実体験し、また、必要に応じてモデルコンパイラに改変を加えるスキルを身につけさせるべく、モデルコンパイラカスタマイズ演習を行う。本演習では、モデルコンパイラに、指定したクラスのインスタンスに所属するステートマシンが状態遷移したときのログを出力するという簡単な改変を行う。

(9) ドメイン特化モデリング言語、演習

最後にこれまでの講義の集大成として、自分独自のモデリング言語であるドメイン特化モデリング言語[7]の開発を行う。講義では、ドメイン特化モデリング言語の説明の前に、基本となる考え方であるソフトウェアプロダクトラインエンジニアリング[8]やフィーチャ図[9]の書き方を説明する。次にドメイン特化モデリング言語の事例を紹介する。さらに、ドメイン特化モデリング言語の開発手順を紹介する。

演習では簡単な題材において、フィーチャ図を用いた可変性の特定の方法や、モデルのアイデア出し、メタモデルによる定義を実施する。

3.5 段階的な演習

ここでは具体的な演習内容について紹介する。演習は段階的に設定しており、演習を順番に進めることで理解が促されるように工夫している。演習対象は前に紹介した Kobuki であり、多くの演習課題が Kobuki を動作させる内容になっている。

(1) 基礎的な制御

まずはモデルを用いて基本的な制御ソフトウェアを開発できることを目的とする。演習課題は以下の通りである:「ボタンを押したら微速で回転を始め、ボタンをもう一度押したら回転が停止する。ボタンを押したら直進を始め、2 秒後に停止する。センタバンパを押したら直進を始め、2 秒後に停止する。」

(2) 自動搬送システム(1)

次に複雑な問題を記述するために、関心事の分離を学ぶ

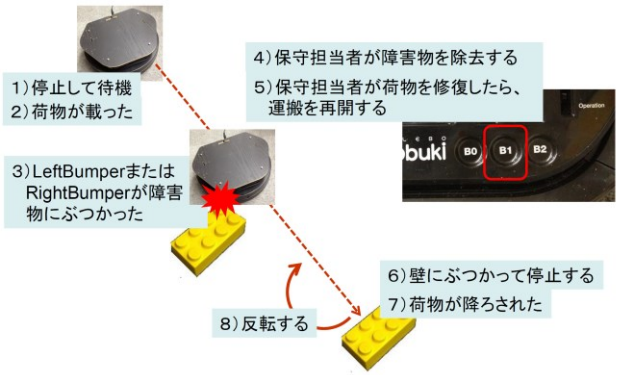

図 5 自動搬送システム演習例

ことができる演習にとりくむ。ここでは、応用分野に関するクラス群と、それをロボットの振る舞いに変換するクラス群とに分けて記述させる。応用分野の語彙を課題にいれており、これを用いてモデリングできることを求める。

演習課題は、シナリオに基づいており、荷物を自律的に配達する自動搬送ロボットの開発になる。本ロボットのシナリオは以下の通りである:「集配人は荷物を積む。集配人は荷物の配送を指示する。ロボットは荷物を配送する。配達先にロボットが到達したら、ロボットは自動的に停止する。集配人は荷物を下ろす。」

実際には荷物を載せる、降ろすはボタンによる指示であり、出発、配達先への到着はバンパセンサの反応を持って行うこととする。

(3) 自動搬送システム(2) (追加シナリオ)

ドメイン分割が適切に実施できているかを実体験するための追加演習をする。この演習では自動搬送システム(1)で実施した演習にシナリオを追加し、変更の少ないモデルになっているかを確認する。変更箇所が多い場合には、モデルをリファクタリングして、改良する:「ロボットは、運搬中に障害物が見つかったら停止して、エラー音を鳴らす。運搬中にバンパ「LeftBumper」またはバンパ「RightBumper」が押されたら、障害物が見つかったとみなす。保守担当者はエラー音を聞いて現場に急行し、障害物を除去する。保守担当者がボタンを押すと、ロボットは障害物の除去が完了したとみなし、運搬を再開する。障害物は複数個見つかることがある。」演習のイメージ図を図 5 に示す。

(4) 自動搬送システム(3) (追加デバイス)

モデルでの記述範囲外のシステムと、モデルで記述されたシステムとの連携方法を学ぶために、デバイスを追加する演習を行う。

本演習では、シナリオとして、壁に沿って走る機能を検討した上で、壁に沿って運搬できるようにするというシナリオとする。

壁との距離を計測するための測距センサが必要となる。そこで、距離比例電圧を出力する超音波距離センサを追加する。追加されたデバイスの制御を行うために、追加された超音波センサの値を、AD 変換値として取得し、さらに、取得した AD 変換値を距離に変換するプログラムを記述する。距離を取得する外部プログラムとモデルが連携して、測距機能を提供するユニットに仕立てる。

4. おわりに

本発表ではモデル駆動開発(Model-Driven Development; MDD)についての理解を促すべく、主に修士の学生を対象とした授業計画、及び、演習内容について提案した。本講義では開発対象として掃除機型ロボットを採用し、ロボット制御を楽しみながら自然に MDD 関連技術を学ぶことができることを目指した計画とした。本講義では、モデルを使った開発において自然に良い設計を採用できるようになるような工夫を加えた。また、モデリングを行えるようになるだけではなく、モデルからソースコードへ変換するモデルコンパイラの仕組みを理解することで、新しい実行環境に対応できる力をつけるのみならず、独自のモデリング言語を開発するための基礎を身に着けさせることを目標とした。

今後、実施結果を踏まえた分析、及び、改良について取り組んでいきたい。

参考文献

[1] スティーブ・メラー, ケンドール・スコット, MDA のエッセンス, 翔泳社, 2004.
[2] Object Management Group, UML: The Unified Modeling Language, http://www.uml.org/.
[3] John Hutchinson, Empirical assessment of MDE in industry, Proc. of Software Engineering(ICSE) 2011, IEEE, 2011.
[4] スティーブ・メラー, マーク・パルサー, Executable UML MDA モデル駆動型アーキテクチャの基礎, 翔泳社, 2003.
[5] Kobuki, http://kobuki.yujinrobot.com/about2/, 2018/6 アクセス.
[6] Shuhei Hiya, Kenji Hisazumi, Akira Fukuda, Tsuneo Nakanishi: clooca: Web based tool for Domain Specific Modeling, Proc. of ACM/IEEE 16th International Conference on Model Driven Engineering Languages and Systems (MoDELS 2013), pp.31-35, 2013.
[7] Juha-Pekka Tolvanen, Matti Rossi, MetaEdit+: defining and using domain-specific modeling languages and code generators, Proc pf the 18th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications, ACM, p. 92-93, 2003.
[8] Klaus Pohl, Günter Böckle, Frank J. van Der Linden, *Software product line engineering: foundations, principles and techniques*, Springer Science & Business Media, 2005.
[9] Kyo Kang, Sholem Cohen, James Hess, William Novak, A. Spencer Peterson, *Feature-oriented domain analysis (FODA) feasibility study*, No. CMU/SEI-90-TR-21. Carnegie-Mellon Univ Pittsburgh Pa Software Engineering Inst, 1990.