実行可能モデルを用いたモデル駆動開発実験

伊藤　邦彦†　　天川　美那‡　　松浦　佐江子†
芝浦工業大学　システム工学部　電子情報システム学科†
芝浦工業大学大学院　工学研究科　電気電子情報工学専攻‡

1．はじめに

ソフトウェア開発では，ソフトウェアの品質を保ち開発期間を短縮することが求められている．そのため，信頼性や保守性の高い既存のソフトウェア構成部品をいかに再利用するかということが重要である．また，多様な環境に適応するため，MDA (Model Driven Architecture)によるモデルの再利用が注目を集めている．

組込みシステム技術協会が主催する自律走行車を組込みソフトウェアで制御する競技である ET ロボコンに MDA を用いて取り組んでいる．本競技で規定されている車体に依存しない自律走行モデルを PIM(Platform Independent Model)として定義し，車体と制御プログラミング言語に依存した PSM(Platform Specific Model)を定義した．しかし，実機でテストを行った結果に基づき，コードを直接修正しながら走行の調整を行ったため，問題点を PIM 上で議論できずに，PIM とコードの対応が不明瞭になった．

本稿では，モデルの再利用性を確保するために，xUML(Executable UML)を用い，実行可能モデルを用いてテストを行う実験を行った．

2．ET ロボコンでの開発

ET ロボコンではレゴマインドストームを使いリアルタイムでライントレースする自律走行車を組込みソフトウェアで制御する競技である．車体は定められ，前方にラインセンサ，タッチセンサがあり，前方の操舵部分と後方の駆動部分の二つのモーターがある．

2．1 ET ロボコンでのコンセプト

コース全体を走りきることを目標とし，変化する多様な環境に対応するために MDA を用いた．PIM として自律運転する車のモデルを作成した．PIM では地図，運転方法の決定，車体という 3 つの領域に分割した．そして PSM として競技に用いられる Lego Mindstorms に特化したモデルを作成した．図 1 は地図の道路状況を見て，行くべき方向，速度を決定し，アクセル踏む・ハンドルをきるとしたものが左の PIM であり，右は PSM である．PSM で車体部分と運転方法を決定する部分の責務をまとめている．

3．Executable UML を用いた自律走行車の開発

実行可能モデルとは，モデルが実際に実行されるモデルである．実際に実行され動作することにより，システムを評価，検証することができる．PIM を実行可能モデルにすることによりモデルを実際に実行することにより正しい振る舞いであるかを検証することができる．

3．1 Executable UML

本稿では，Executable UML を用いて実行可能モデルを作成した．Executable UML は UML のプロファイル（拡張）であり，UML のサブセットに対する実行セマンティクスを定義する[1].Executable UML にはモデルコンパイラが存在する．モデルコンパイラはハードウェアとソフトウェア環境に関する決定群を使用して，モデルを変換

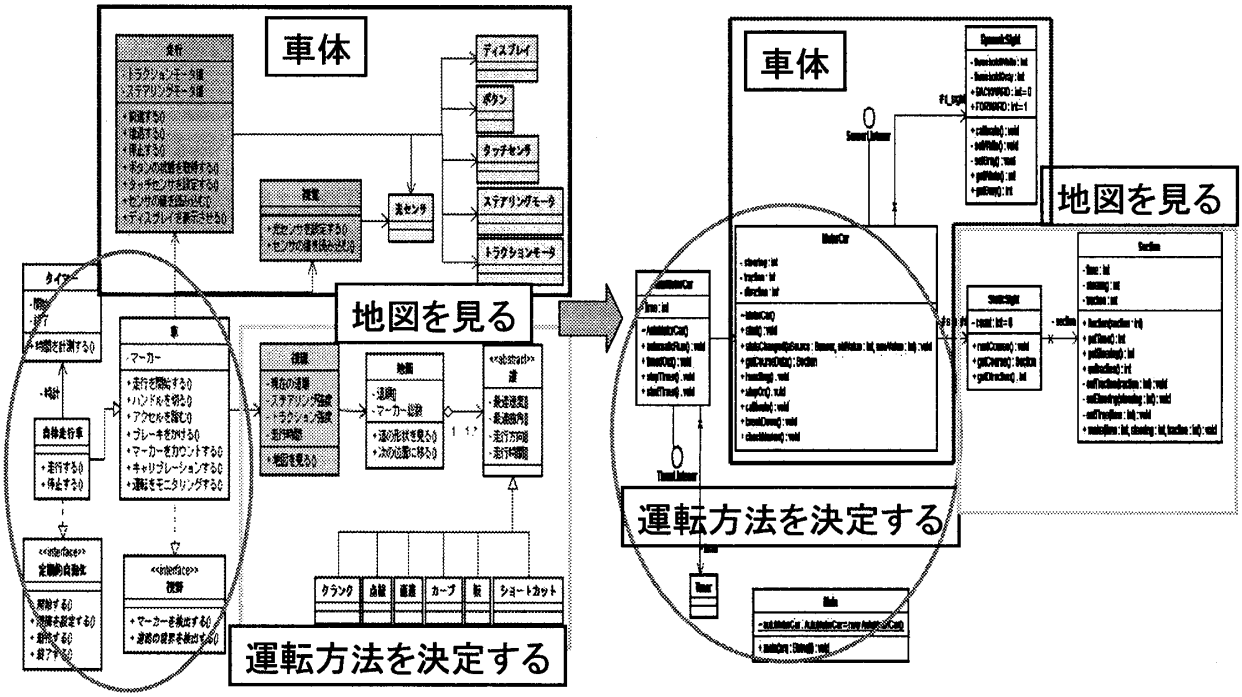

図 1　ET ロボコンで作成した PIM，PSM

A Model Driven Development Experiment with Executable Model.
† Kunihiko Ito　‡ Mina Amakawa　† Saeko Matsuura
† Shibaura Institute of Technology Department of Electronic and Information Systems
‡ Department of Electronic Engineering and Computer Science, Graduate School of Engineering, Shibaura Institute of Technology

する．Executable UML のアプローチとして以下の 3 つの基本的な視点がある[2].

① データを UML クラス図で定義する.
② 制御を UML ステートマシン図で定義する.
③ アルゴリズムをアクション言語で定義する.

クラス図はクラス，属性，関連，制約をモデル化し，ステートマシン図は状態，イベント，遷移，プロシージャをモデル化する．プロシージャはアクション群で構成される．アクションはシステムの基本的な計算処理を実行する．

3．2Executable UML を用いた開発

ＥＴロボコンで作成した PIM を Executable UML 化した．Executable UML を記述するにあたり Executable UML ツールとして iUMLite2.20 を利用した．

クラス図はシステム内のオブジェクトタイプと，それらの間に存在する各種の静的な関係を記述する．

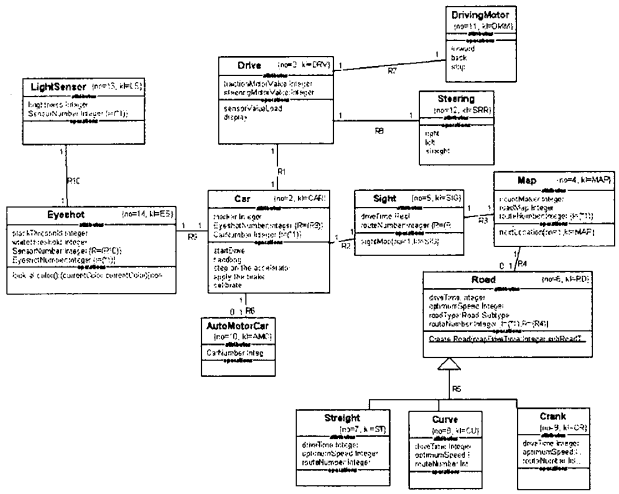

図２　ＥＴロボコンのクラス図

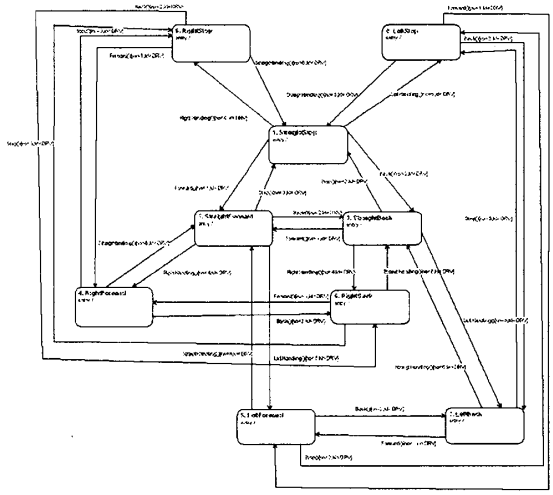

図３　Drive クラスのステートマシン図

図２はＥＴロボコンで作成した PIM を Executable UML 化したクラス図である．クラス間を結んでいる線を関連と呼びそれぞれ関連名を持つ．iUML では任意の関連名を記述することができないため「R1」というようなラベルとして関連作成順に番号が振られる．また，クラスが持っている操作に対してアクション言語を用いて操作を具体的に記述することも可能である．

図３では Drive クラスのステートマシンを示している．Drive クラスでは，車体の状態を駆動部が前進・後退・停止の３つの状態，操舵部が右にハンドルをきる・左にハンドルをきる・まっすぐにしている３つの状態，それらの組み合わせにより９つ定義した．

アクション言語により実装に依存せずに，振る舞いを明確にする．アクションはステートマシン図の状態の中に記述する．図４は Car クラスの地図を読み運転方法を決定する状態のアクションである．この状態では地図に定められた走行速度を決定し，次の状態へのシグナルを送っている．またアクション言語は iUML で使用可能な独自の ASL（Action Specification Language）を使用している．

```
1. Formulate a Driving
entry /
sightRoad = this -> R2 -> R3 -> R4
drive = this -> R1
speed = sightRoad.optimumSpeed
drive.tractionMotorValue = speed
drive.steeringMotorValue = 4
switch sightRoad.roadType
case 'Straight'
        generate CAR1 Straight_Driving() to this
case 'Curve'
        generate CAR2 Curve_Driving() to this
case 'Crank'

endswitch
```

図４　運転方法決定状態のアクション

4．Executable UML を用いた結果

実際に動作させる際，初めに作成した属性だけでは動作させることができなかった．現在のクラス構成では位置を正確に把握することができていないことが原因であった．これより，クラス構成の見直しを行う必要があることがわかった．その結果，現在位置を特定するため座標と速度の属性を Car クラスに付加した．ソースコードを修正した場合ではメソッド内を変更する傾向があったため責務の肥大化が見られた．モデルを実行し，モデルの修正を行うことでシステム全体を捉えることができる．また，自律走行車では走行路，車体の性能をテストデータとしてシステムとは分けて実験し定義することで実機に実装時と同じ状態を作り出すことができる．走行路および車体の変更が生じた場合はシミュレート時に変更することでシステムが正常に動作可能であるか検証できることができる．また，走行路，車体の性能を詳細に定義することでシステムの振る舞いを変更し，これらを再利用することが可能である．

5．まとめ

モデルを抽象度の高いレベルで定義し，実装技術やソフトウェアプラットフォームに依存させないことでシステムの再利用を高めることが可能になる．実装前にモデルをシミュレートさせることでモデル作成時にモデルを実行し修正することでコースから外れしまうなどのバクを早期に発見することができる．このようなバグは実機では正しく動いているかに見えるがモデルとして動作させた場合では意図せぬ動きをしていることが車体の実験を分けて行うことで分かった．

実行可能モデルはモデルコンパイラを用いることでソースコードへの変換可能であり，モデルとソースコードの乖離を防ぐことが可能である．

実行可能モデルを用いることでモデル駆動開発でのモデルの再利用性を高めることができると考える．

6．参考文献

[1] スティーブ J.メラー，ケンドール・スコット：MDA のエッセンス，翔泳社，2004.

[2] スティーブ J.メラー，マーク J.バルサー：Executable UML，翔泳社，2003.