組み込みソフトウェア開発のための仕様シミュレーション環境

伊藤 恵
公立はこだて未来大学

久保秋 真
キヤノンソフトウェア (株)

組み込みソフトウェア開発では、個々の製品寿命が短い一方で、似通った製品を多数開発することも多く、そういった状況に合わせた開発支援が必要とされている。さらに、ハードとソフトが並行して開発される組み込みシステムでは、通常、ハードウェアまで揃わないと最終的な動作確認ができないという問題がある。そこで、本環境では、個々の製品プラットフォームに依存しない論理的なモデル記述を、計算機上でシミュレーション実行することで、製品シリーズ全体の開発効率の向上を目指す。

A Specification-Level Simulation Environment for Embedded Software Development

Kei Itou
Future University - Hakodate

Shin Kuboaki
Canon Software Inc.

In software development for embedded systems, CASE tools support considering the situation that each embedded product is short-lived and resemble each other are needed. Especially in embedded systems, developping software cannot be checked against its expected behavior before partner hardware is supplied, because software and hardware of embedded systems are complexly involved.

This paper proposes an environment for embedded software development in which we execute and test logical specifications for the target system, not depending on each product platform, to improve efficiency and quality of developping a series of embedded systems.

1 はじめに

組み込みシステムのためのソフトウェア開発において、その個々の製品寿命が短い一方で、似通った製品を多数開発することも多くなり、あるプラットフォーム向けに開発したソフトウェアを、少し機能追加して別のプラットフォーム用に開発するなどといったことが、しばしば起こるようになってきている。こういった状況において、モデル駆動アーキテクチャ[4] のような考え方の導入により、実装プラットフォームに依存しない仕様記述を、依存する記述と分離し、プラットフォーム依存しない記述を再利用することで、開発効率や製品品質の向上を計ることが期待されている。

このような考え方の導入により、実装プラットフォームに依存しないモデル記述を分離した場合、そのモデルは再利用性は高くなるが、一般にそのようなモデル自体は実行することはできないため、記述対象の動作確認は、そこからプラットフォーム依存のモデルや実装コードに変換されるのを待たなければならない。プラットフォーム非依存のモデルを実行可能とするために ActionSemantics[5] 等の提案があり、そういったモデルのバリデータや実行ツールなども提供されているが、仕様のテストを行うツールとしての機能がまだ充分とは言えない。

組み込みシステムの場合、ハードウェアとソフトウェアが特に密接に関わるため、ハードウェアまで揃わないと、ソフトウェア側の動作確認さえ充分に出来ないという状況になることが多い。にも係わらず、開発期間の短縮のため、ハードウェアの完成前にもソフトウェアの開発を進める必要があり、そのため、ソフトウェア側のみでの早い段階での高度な動作テスト環境が強く求められている。

我々は、オブジェクト指向概念を含む状態遷移図ベースの仕様記述モデルとして ObTS を提案した。ObTS では、実装プラットフォームに依存しない抽象的な概念のみを用いてモデル記述を行うため、そこで用いる個々の概念を実際の開発対象となるソフトウェアのどの概念に対応付けるかによって、対象ソフトウェア全体の概略的な仕様記述や、ソフトウェア部品のより詳細な仕様記述を行うことができ、さらにはソフトウェアそのものだけでなく、関連するハードウェアを含めた実行時の周辺環境も含めてモデル化することも可能である。また、ObTS で記述された仕様は、関数型言語 Standard ML 上でシミュレーション実行することができる。単に仕様を実行できるというだけではなく、ML 言語の持つ高階関数や豊富なデータ型などの特徴を活かしたテストスクリプト等を作成可能であるため、仕様の高度なテストが可能となる。

そこで本研究では、仕様記述モデル ObTS を用いて、組み込みシステムの個々の製品プラットフォームに依存しない論理的なモデル記述を行い、それを計算機上でシミュレーション実行することによって仕様レベルのテストを行い、製品シリーズ全体の開発効率の向上と製品品質の維持を目的とした開発支援環境の提供を目指す。

以下、まず 2 で我々がすでに提案した仕様記述法である ObTS について説明する。3 では、ObTS を用いた仕様記述とそのテストについて、例題を用いて述べる。4 では、コード生成やテストパターン生成など、仕様シミュレーション環境から開発へのフィードバックについて述べ、5 で評価・検討を行う。

2 ObTS モデル

我々は状態遷移図を記述するための標準的記法である Statechart[1] を、オブジェクト指向の概念を用いて拡張したものとして、ObTS モデルを提案した [2]。ObTS モデルでは記述対象のシステム構造をオブジェクトの階層構造によって表現し、個々のオブジェクトの動作は状態遷移図で、システム全体の動作は個々のオブジェクトの内部動作と、オブジェクト間のイベント通信によって表現される。ObTS は特徴的なオブジェクトの階層化

を持っており、階層の親オブジェクトがその子となるオブジェクト (内部オブジェクトと呼ぶ) を自分自身の状態遷移図の一部として持っている。

個々のオブジェクトはその動作記述として状態遷移図、内部データとして属性を持ち、また動作委譲のために内部オブジェクトを持つことができる。内部オブジェクトも通常のオブジェクトであるが、親オブジェクトからは通常の状態と同様に遷移の対象として記述される。つまり、親オブジェクトが内部オブジェクトへの状態遷移をすることで、内部オブジェクトが起動されて動作を開始し、親オブジェクトが内部オブジェクトからの状態遷移をすることで、内部オブジェクトが動作を終了する。また、複数の内部オブジェクトを並行動作するものとして記述することができる。並行動作する内部オブジェクトの集まりは、単一で動作する内部オブジェクトと同様に親オブジェクトの状態遷移の対象であり、それらは親オブジェクトの状態遷移によって、同時に動作を開始/終了する (図 1)。

ObTS では個々のオブジェクトが非同期に独立して動作するという一般的な動作モデルを取らず、ステップというものに同期して、すべてのオブジェクトが同時に動作するという動作モデルを取っている。ステップとは、

1. 前のステップで発生したイベントをすべてのオブジェクトに配送する。
2. 個々のオブジェクトを、たかだか状態遷移 1 回ずつ、同時に実行する。
3. 状態遷移によって発生したイベントを回収する。

という一連の動作である。個々のオブジェクトの状態遷移は、入力イベントの受信、関数的属性計算、出力イベントの送信という一連の動作から成る。個々の状態遷移で入力として受信するイベントは、そのステップで受信可能な複数のイベントを指定でき、また状態遷移のトリガとして、受信できる単体もしくは複数のイベントだけでなく、イベント名を and や or で結合したイベント論理式として指定することができる。状態遷移で値が計算される属性は、そのオブジェクト自身の属性か、出力イベントに付加するイベント属性だけであり、属性計算で参照できる属性は入力イベントに付加されていたイベント属性か、オブジェクト自身の属性だけである。つまり、オブジェクト間作用はイベント通信によってのみ為され、属性そのものを介した通信は行われない。このような動作を繰り返すことで記述されたシステムを動作させていく。

2.1 ObCL 言語

我々は ObTS モデルをクラスベースで記述するための記述体系として ObCL 言語を提案した [3]。ObCL では ObTS モデルのオブジェクト、フィールド、イベント、属性をそれぞれクラスを用いて記述する。トップレベルのオブジェクト指定と、クラス間の参照関係に基づいて、ObCL 記述をインスタンス化したものが ObTS モデルになる。クラスの概念を用いることで基本的な記述の再利用を行うことができる。

2.2 シミュレーション環境 ObML

また、ObCL 言語によって記述された ObTS モデルを関数型言語 Standard ML 上でシミュレートするための環境 ObML を構築した (図 2)。この環境は ObCL 言語による記述を Standard ML 言語の記述に変換するコンバータと、Standard ML インタプリタ上で動作するシミュレーションエンジンから成る。シミュレーションエンジンは ML 関数とそれに関連する ML データ型として提供されており、シミュレータは一般の ML プログラムから通常の ML 関数呼び出しとして簡単に利用することができる。これにより、対象システムのためのテストスクリプトを ML プログラムとして作成することができ、単体テスト用スクリプトの全体テストでの利用、旧バージョン用テストスクリ

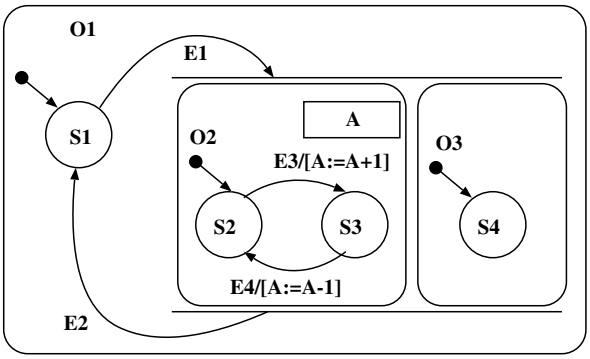

図 1: ObTS 図

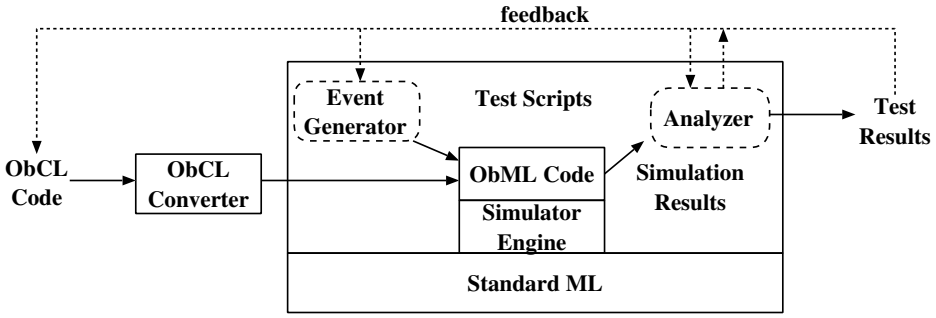

図 2: ObTS 支援環境

プトの新バージョン用への利用などといった再利用性の高いテストスクリプト群を構築できる。

3 ObTS モデルを用いた仕様記述と仕様テスト

ObTS モデルを用いた仕様記述と仕様テストの例として、迷路探索ロボットの制御プログラムを用いる。自走するロボットの制御プログラムでは、ソフトウェアから直接扱えるのは、センサーからの信号読み取りや、モーターの ON/OFF などであり、量産される家電製品の組み込み用ソフトウェアと同程度の複雑さやハードウェアとの密接度を持っていると考えられる。また、迷路探索ロボットの場合、探索アルゴリズムを実装しても、そのプログラムを組み込んだロボットが実際に迷路を意図通りに探索できるかどうかが予測しにくく、シミュレーションレベルで充分な仕様テストを行うには、ロボットの物理的な特性や、ロボットの外部環境のモデル化も必要となる。

3.1 仕様記述

記述例題として、迷路探索ロボットの制御プログラムを扱うにあたり、簡単化のため、迷路やロボットそのものについて、以下のような仮定を行う。

- 迷路の一マスは一定のサイズである
- 迷路は必ず 90 度に曲る
- 迷路上にはロボットの移動を妨げるものは何もない
- ロボットは前面にある接触センサーからの信号のみで壁の有無を判断する
- ロボットは一定時間のモータ制御により、誤差なく一定距離を前進 (または後進) する
- ロボットは一定時間のモータ制御により、誤差なく 90 度の左右回転を行う

さらに、仕様記述および仕様テストにあたってロボットの動作を抽象化するため、迷路一マス分の前進/後進や 90 度回転は atomic な動作として記述できるものとする。このような前提のもとで ObTS による仕様記述を行う。

図 3 は、このロボットの仕様を概略を示したもので、左列が ObTS による記述、右列が use-case 図である。

ObTS による記述では、ロボット (RobotSystem) を、実際に制御プログラムが動く制御部 (Controller) そのもののほか、タッチセンサー (TouchSensor)、左右 2 系統のモータ (Motor1, Motor2)、および、ロボットに動作開始/終了を指示するためのスタートボタン (StartButton) からなるものとしてモデル化する。制御部以外は、それぞれ対応するハードウェアをモデル化したもので、仕様レベルのシミュレーションを行うのに必要なだけの記述がされている。制御部は停止状態と走行状態があり、走行状態では操舵そのものを担うステアリング (Steering) と、次の操舵方向を判断する方向制御 (Decision) がある。方向制御はタッチセンサーからの信号を受け取って、ステアリングへの指示を送り、ステアリングはその状態に応じて、2 系統のモータに制御信号を送る。

3.2 仕様テスト

迷路探索ロボットの場合、実際にさまざまな迷路で走らせてみることが、その動作テストとなる。つまり、実装レベルでのテストは、その制御プログラムをロボットに組み込んで、実機でテストすることになる。一方、本環境における仕様テストでは、モデル化された仮想の迷路上を、仮想のロボットが移動できる。したがって、仕様記述段階での前提条件を満たす迷路を無作為に多数生成して与え、記述した探索アルゴリズムによってどの程度それを解くことができるのかを、シミュレーションによって容易に調べることができる (図 4)。

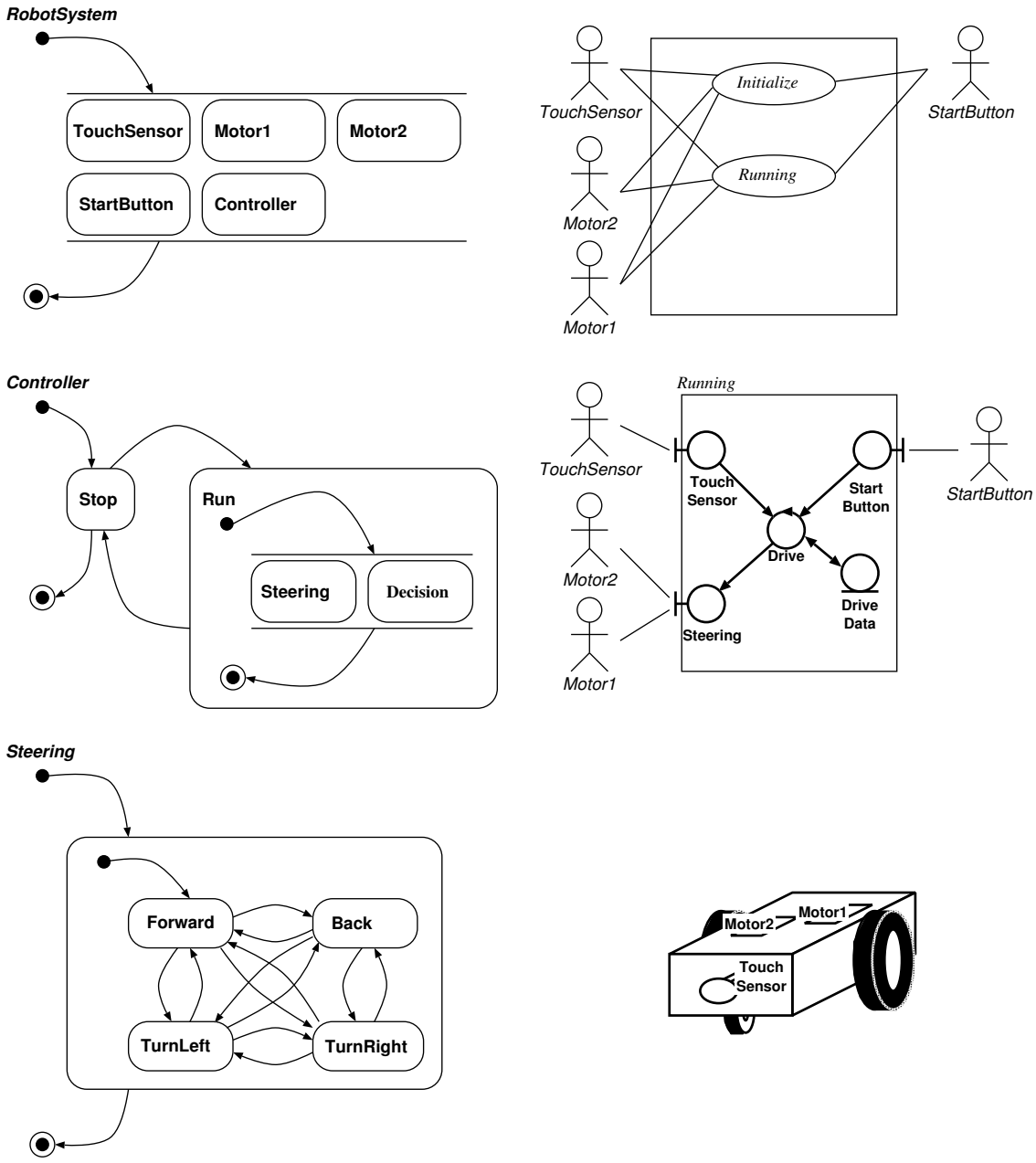

図 3: 迷路探索ロボットの仕様記述概略

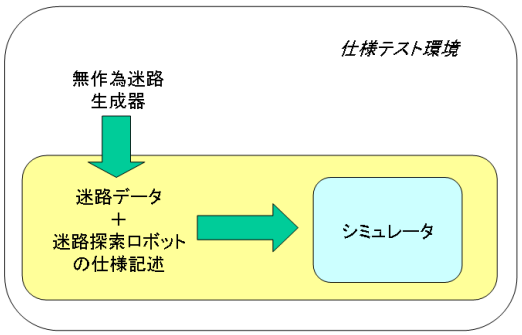

図 4: 迷路探索ロボットの仕様テスト

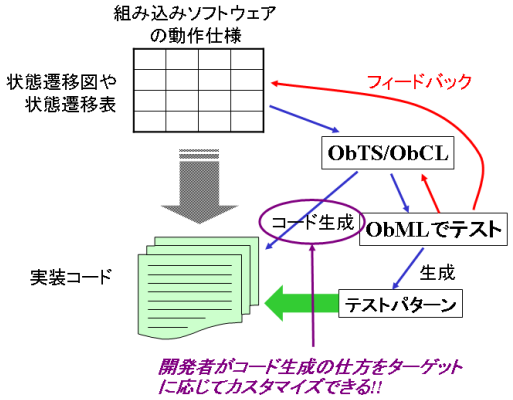

図 5: 仕様記述からコード生成まで

4 仕様シミュレーションから開発へのフィードバック

実装プラットフォームに依存しない仕様記述を単にシミュレートしたり、テストしたりするだけではなく、その結果から、以降の開発への支援となるような生成物を得ることも可能である (図 5)。

4.1 コード生成

組み込みシステムでは、実装プラットフォームのわずかな変更により、実装すべきコードが大きく変わることもあり、ツールによる自動的なコード生成では、そのすべてに対応することは難しい。しかし、実装プラットフォームについてのなんらかの情報をツールに与えることによって、生成するコードパターンを変更したり、その一部分に実装コードをあらかじめ埋め込むなどのカスタマイズは可能である。

また、最終的な実装コードの生成を目指すのではなく、実装コードのテンプレートや、実装コードの元になるような中間コードを、開発者が容易に変更可能であるような状態で出力することで、完全自動でない代わりに小回りのきくコードを生成することができる。

例えば、先に述べた迷路探索ロボットの制御プログラムを C 言語のプログラムとして生成することとし、センサーからの入力や、モータそのものの制御などは、ObTS モデルにより仕様記述上はイベントとして表現されるが、ハードウェアとのイベント送受信を関数やマクロの呼び出しとしてコード生成し、その関数自体は別途用意しておくことで、探索アルゴリズムそのものは仕様レベルのテストをパスしたものを、実装コードとして利用することが可能となる。

4.2 実装テストへのテストパターン生成

仕様レベルでのテストは、実装コードのテストを完全に置き換えるものにはならないが、実装コードのテストで重点的に調査すべき箇所を、仕様テストの結果から抽出することが可能な場合もある。

5 評価・検討

本環境の評価のため、実際に迷路探索ロボットとその外部環境をモデル化し、ObTS モデルを用

いて仕様記述を行った。また、無作為に生成した多数の迷路データを用いて、記述した探索ロボットの仕様レベルのテストを行った。

組み込みシステムのハードウェア部分や外部環境を適度に抽象化したモデルと共に記述することで、ハードウェア部分なしでも、組み込みソフトウェアの動作シミュレーションが可能となり、組み込みソフトウェア開発に対して、一定の有効性が確認できた。

さらに、記述した仕様からLEGO MindStorms[6]向けにコード生成を行った。出力言語は、MindStormsの制御プログラムをCライクな言語で記述できるnqcとし、ObTS上の概念から実装コード上のコードパターンへの変換を試みた。

記述した仕様からの、コード生成やテストパターン生成については、まだ充分な検討が出来ていないが、今後、複数の事例適用をすることで、現場のニーズに答えられるような生成物を提供できるかどうか検討を進めたい。

6 おわりに

本論文では、仕様記述モデルObTSを用いて、組み込みシステムの個々の製品プラットフォームに依存しない論理的なモデル記述を、シミュレーション実行することで仕様レベルのテストを可能とする組み込みソフトウェア開発支援環境を提案した。

この環境を用いて、小さな例題ながら実装プラットフォームに依存しないモデル記述と、そうでない記述から分離した上で、プラットフォーム非依存の記述を組み込みシステムのハードウェア部分の適度に抽象化されたモデル記述と共にシミュレーション実行とテストを行ったことで、ハードウェア部分が用意される前の段階での組み込みソフトウェアのシミュレーションテストの可能性を確認することができた。

参考文献

[1] D.Harel, A.Pnueli, J.P.Schmid and R.Sherman, “On the Formal Semantics of Statecharts”, Proc of 2nd IEEE Symposium on Logic in Computer Science, pp.54-64,1987

[2] 伊藤 恵, 片山 卓也, “オブジェクト指向方法論のための動的モデル ObTS”, コンピュータソフトウェア,Vol.14,No.2,pp.22-37,1997

[3] 久保秋 真, 伊藤 恵, 片山 卓也, “ObTS モデルに基づくオブジェクト指向仕様記述言語と支援環境”, 日本ソフトウェア科学会第 14 回大会論文集,pp.589-592,1997

[4] OMG Japan, “Model Driven Architecture”, http://www.omgj.org/technology/mda/

[5] Action Semantics Consortium, “Action Semantics for UML”, http://www.umlactionsemantics.org/

[6] LEGO Company, “LEGO MindStorms”, http://mindstorms.lego.com/