ソフトウェアモデル作成支援ツール実現のための状態遷移分析手法

福井 雅彦† 島津 仁晶†

State transition analysis method for software modeling support tool realization

Fukui Masahiko Shimazu Yoshiaki

ねらい 状態遷移モデルを導出する定型的な手法を確立し、ソフトウェアの品質向上のための形式手法モデル作成支援ツールを開発に繋げる。

キーワード 形式手法，B メソッド，モデリング，ステートマシン，状態遷移図，高信頼，ソフトウェア設計

Target: Establish a routine technique for deriving a state transition model, and lead to the development of formal methods model creation support tool for improving the quality of software.

Keywords: Formal methods, B-Method, Modeling, State machine, State machine diagram, High reliable, Software s w design

1. 想定する読者・聴衆

本テーマで想定する主な読者・聴衆は、形式手法開発を導入、検討または、関心のある技術者。特に VDM や Z、B 等モデル規範型の形式手法に於いて状態分析の手法に関心のある方を対象としている。

2. 背景

今日の組込み機器開発では、高い信頼／安全性、標準規格対応など品質を維持するための多岐にわたる対策が必要となっている。事実、欧米での自動車関連訴訟や製品リコールが企業業績やイメージに与える影響は甚大[1]である。原因の一つに機能やサービスの高度化に伴う、開発量の爆発的な増大[2]がある。これに伴い開発現場でも、テストや品質管理等の品質の維持コストが増大している。

この問題の解決策の一つに、高品質なソフトウェア開発技術として期待されているのが形式手法である。形式手法は数理表現に基づく形式的な記法を用いて仕様を記述することで、矛盾や曖昧さを排除して仕様や実装の誤りの低減を図る技術である。

特に B メソッド[3]は仕様の検証だけでなく仕様と実装の整合性検証やコード生成等の下流工程をカバーしている。反面、実装の正当性を証明するための仕様のモデル化が難しい。このため他の形式手法と比べ開発現場での活用が困難で、日本国内での普及が進んでいない。

3. 課題

B メソッドの技術情報はデータのモデル化に偏っており、ソフトウェアの振る舞いを記述する実践的な手法に関する情報が少ない。これは本研究で適用対象に定めている自動車をはじめとした制御系分野で大きな障害になっている。

理由は機器制御の場合、入力の大半が単純なスカラー値であるため、開発の力点が置かれるのはデータ構造ではなく振る舞いのモデル化だからである。

そこでソフトウェアの振る舞いモデルを構築するために状態遷移モデルを適用したが、以下の問題に直面した。

課題： 制御系の開発現場で B メソッドを活用するためのソフトウェアの振る舞いの形式的仕様記述手法が確立されていない。

このため、要求仕様の記述形式が状態遷移モデルと大きく異なるため仕様追跡が困難であるうえ、手作業で状態遷移モデルに変換するため仕様との適合性を保証する手段もない。

これらの問題を解決するため過去の研究事例及び既存のツールを調査したが、状態遷移モデルからのコード生成や[4]、導出可能な制御シーケンスを作成して状態遷移モデル生成する[5]手法は見つかったが、今回の目的に沿った事例は見つからなかった。

4. 提案・実験

前述の課題を解決するために、制御分野向け状態遷移分析手法を考案した。

- 実装を表す状態モデルを作成する前に、要求仕様に近いステートを意識しないモデルを作成する。以下、これを「仕様モデル」と呼ぶ。
- 明確なモジュール分割ルールを持ち、仕様モデルを適切な規模で維持できる。
- 実装を反映した状態モデルは、確立した手順で仕様モデルから機械的に導出する。以下、この導出さ

†アーク・システム・ソリューションズ株式会社 開発部 研究開発課

〒060-0001
北海道札幌市中央区北 1 条西 7 丁目 1 番地 15 あおいビル 5 階
TEL：(011)207-6460 FAX：(020)4622-5064
E-mail: fukui.masahiko@arcsys-sol.co.jp
E-mail: shimazu.yoshiaki@arcsys-sol.co.jp

れるモデルを「実装モデル」と呼ぶ。

本手法の作業の流れを図 1 に示す。

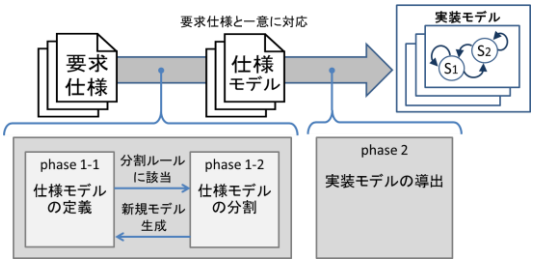

図 1. 作業の流れ

4. 1 仕様モデルの定義

仕様モデルは実装モデル導出を目的に作成するが、要求仕様をモデルに変換する際の誤り防止と、開発者に要求されるスキルを軽減するために、ステートを意識しない形で記述する。また。実装モデルを導出するために必要な変数、イベント等の情報を定義する。(図 2)

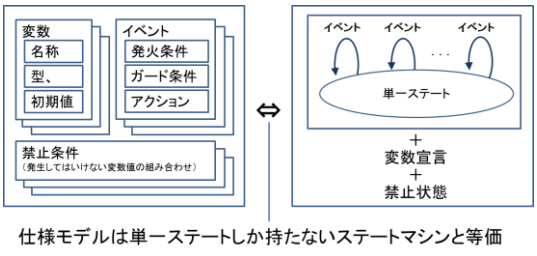

図 2. 仕様モデル

仕様モデルは要求仕様の項目と原則一意対応するため、作成や仕様追跡が容易である。これは 1 個のステートしか持たないため、同じ動作が複数のステートに分散しづらいためである。

4. 2 仕様モデルの分割

仕様モデルは実装モデルに 1 対 1 対応する。本手法では、個々のステートマシンに対応する仕様／実装モデルを 1 モジュールとする。

仕様モデルのイベント定義を行う際に以下に該当する場合はモジュールを分割する。

（1） ルール 1

イベント評価内で変数更新がある場合、モジュールを分割し、更新処理をアクションに記述する。(図 3)

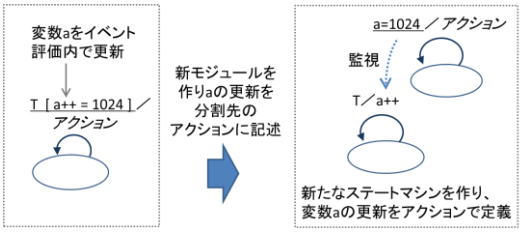

図 3. 下位モジュールに分割

（2）ルール 2

状態遷移に関わる変数間で相互依存がない場合は、モジュールの単純さを保つために分割する。

図 4 は分割できない例である。これ以外の依存関係なし or 一方的な依存関係の変数を持つモジュールは分割を行う。

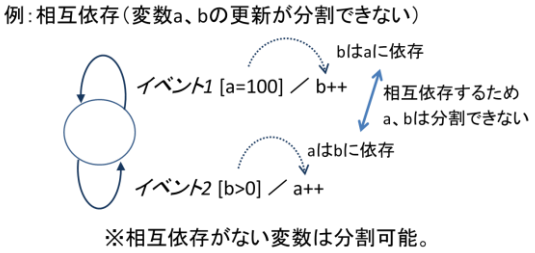

図 4. 分割できないケース

分割で新規に作成されたモジュールは、分割元と同様に仕様モデルの定義を行う。全てのモジュールで分割ルールの適用可能箇所がなくなったら仕様モデルの作成は完了する。

4. 3 実装モデルの導出

実装モデルは、実装を意識しているため複数のステートを持つ。本手法では、単一ステートで動作を記述した仕様モデルから実装モデルを導出する。(図 5)

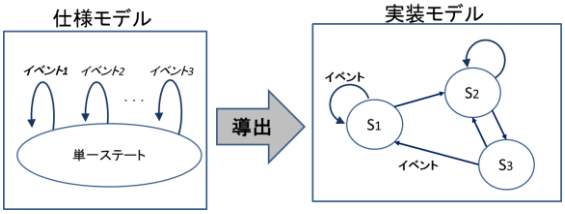

図 5. 実装モデルの導出

導出は自動化可能な数学的操作を元に行う。(図 6)

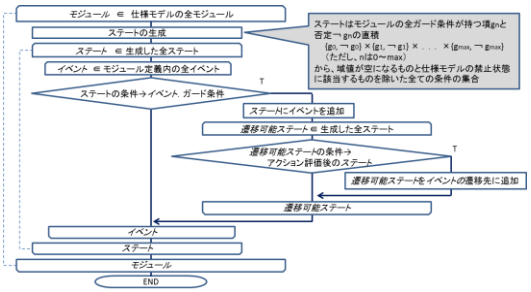

図 6. 実装モデル導出の概略フロー

5. 効果

本手法を実証するため簡単なサンプルをモデリングした。図 7 はサンプルのために作成した要求仕様。

図 7. サンプルの要求仕様

仕様モデルの作成は入力、出力、ロジックの３モジュールから開始する。入出力は要求仕様の入出力定義から、ロジックは入力による動作記述を元に変数やイベントを決定する。

モジュール分割とイベントの定義を全て終えた状態が図 8。（注：変数と禁止条件はスペースが足りないので省略する）

図 8. キッチンタイマの仕様モデル分析結果

更に分析結果を元にＢメソッドのモデルを作成した。仕様モデルは抽象モデル、実装モデルは具象モデルにそれぞれ対応して機械的に変換可能である。ただし、今回は実装モデルを手作業で作成したので、洗い出したステートやイベントを使って直接具象モデルを記述している。

実際に作成して証明を通したものが図 9 のツリーである。機械的に置き換えたモデルなので、点線で囲った部分の構成は図 8 の構成と完全に一致する。

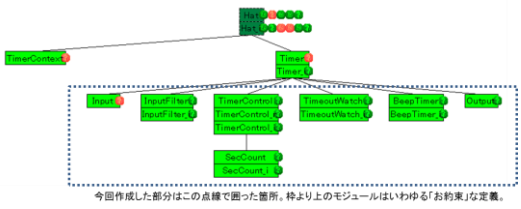

図 9. Ｂモデルで表現したモジュール構造

図 10 は以前に作ったほぼ同じ機能のモデルとの比較である。証明責務数を比較すると、今回は最大でも 96 件までであるのに対し、前回は最大で 442 件発生している。B では仕様や実装の記述が適切に分割できずにロジックが複雑化すると証明責務も増加する。即ち、今回作成したモデルは明確なルールに基づいた適切な分割がなされていることを示している。

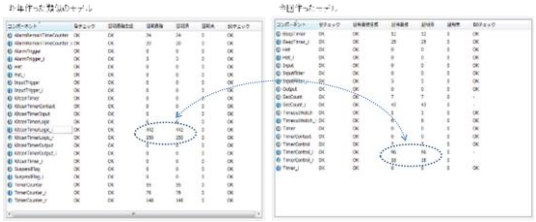

図 10. 証明責務数の比較

6. まとめ

ソフトウェアの振る舞いを要求仕様に近い仕様モデルで記述できるため、開発現場でありがちな属人的スキル依存が軽減できる。

仕様モデルからの実装モデル導出は数学的操作のみで行えるため、ソフトウェアツールによる自動作成支援の可能性が開けた。また、これは本来 B メソッド活用のために確立した手法ではあるが、状態遷移モデルを使う他の形式手法やソフトウェア設計にも応用が可能と思われる。

B メソッドの形式モデルに変換可能な仕様モデルと実装モデルの作成導出手法を確立したことにより、B メソッドを開発現場で活用し易くなった。

7. 用語・文献

文　　献

[1] 自動車リコール届出件数　平成 26 国土交通省 自動車局審査・リコール課公表資料より

[2] ET2002 TB-6 「組込みシステム開発における品質向上の施策」、 平山雅之(東芝 研究 開発センター)

[3] The B-Book: Assigning Programs to Meanings, Jean-Raymond Abrial, Cambridge University Press

[4] JaSST'08 Tokyo 「ModelChecking 技術の専門性の排除とその効用」 富士ゼロックス

[5] Generating Hierarchical State Machines from Use Case Charts