2007－SE－155 (12)
2007／3／22

モデル駆動によるテスト仕様の生成手法の提案

磯田 誠　　田村 直樹
三菱電機株式会社 情報技術総合研究所

我々は，高機能化・複雑化が進む組込み制御ソフトウェアへの，モデル駆動開発（MDD）の適用を検討している．MDD は開発対象ソフトウェアの動作仕様から実装コードを生成することを狙った開発手法であり，品質および生産性の向上が期待される．しかし，現時点では，実装コードを完全に生成する技術は確立していない．そのため，MDD を利用した開発においても，検証と妥当性確認を実施する必要がある．本稿では，動作仕様を記述する状態遷移モデルから検証と妥当性確認のためのテスト仕様を生成する手法を提案する．また，通信制御ソフトウェア開発への，本手法の適用例を示す．

An Approach to Define Test Specification Applying Model Driven Development

Makoto Isoda　　Naoki Tamura
Mitsubishi Electric Corporation　　Information Technology R&D Center

Model Driven Development approach is expected to help developing software in better productivity and quality, but still we are required to verify and validate the developed software. In this paper, we propose an approach to verify and validate embedded software applying MDD concept, and describe the process with example models.

1. はじめに

近年，組込み制御ソフトウェアの高機能化・複雑化が進んでいる．組込み制御ソフトウェアの特徴は、信頼性や効率性などの品質要求の比重が高いことにある．これらの品質要求に応えるため，ソフトウェアの実装では独自の方式を採用することが多い．同時に，要求される機能も高度化が進んでおり，結果的に開発規模も急増している．

このような組込み制御ソフトウェアの動作仕様を記述する手段として，一定の形式性を持ったUMLの利用が広がっている．また，CASEツールベンダからは，UMLを用いて記述した動作仕様から，実際に動作するソフトウェアの実装コードを半自動生成することを狙った開発手法として，モデル駆動開発（Model Driven Development，MDD）が提案されている．さらに，産官学共同での組込み制御ソフトウェア開発へのMDDの適用の取り組みも始まっている[1].

しかし，現時点では，動作仕様から実装コードを完全に生成する技術は確立していない．そのため，MDDを利用した開発においても，検証と妥当性確認のためのテストを実施し，品質を確認する必要がある．

本稿では，MDDにテスト仕様作成とテスト環境構築の作業を加え，動作仕様を記述する状態遷移モデルからテスト仕様を生成する手法を提案する．また，通信制御ソフトウェア開発への，本手法の適用例を示す．

2. モデル駆動開発の課題

モデル駆動開発の一般的な開発手順と，ソフトウェアの検証と妥当性確認についての課題を示す．

2.1. モデル駆動開発での開発手順

MDD とは，OMG が標準化を推進しているモデル駆動アーキテクチャ（Model Driven Architecture，MDA）[2][3]の体系および技術を取り入れ，近年，多くの CASE ツールベンダが提唱している新しい開発手法である．

MDA でのソフトウェア開発では，まず開発対象ソフトウェアの計算独立モデル（CIM）を作成し，モデル間のマッピングルールを用いてプラットフォーム独立モデル（PIM），プラットフォーム依存モデル（PSM），実装コードへと段階的にモデル変換を適用し，実装コードを半自動生成する．

各々のモデルでは，その振舞いをクラスに対応する状態遷移として記述し，詳細な動作はアクション言語を用いて定義する．アクション言語に関する標準仕様としては，Action Semantics[4]，OCL[5][6]が提案されている．ただし、アクションの構文を定義した標準仕様はまだ存在していないため，現時点では，多くの CASE ツールが独自のアクション言語やプラットフォーム依存のプログラミング言語を利用することが多い．

2.2. ソフトウェアの検証と妥当性確認

ソフトウェアの検証と妥当性確認（Verification and Validation）は，ソフトウェアの品質を確認し，保証するために重要な活動の一つである．特に，組込み制御ソフトウェアのように，高信頼性や高効率性の実現が要求される場合，これらの要求が確実に実現されたか確認する活動が重要となる．

検証の目的は，モデルで定義したとおりにソフトウェアが動作するかどうかを確認することである．また，妥当性確認の目的は，機器が実際に利用される外部環境（対向機器，ネットワーク，センサの監視対象物など）の中で，モデルが正しく動作するかどうかを確認するものである．このような検証と妥当性確認の手法として，表 1 に示す手法がよく利用されている[7][8]．

表 1 検証と妥当性確認の手法

検証手法	妥当性確認手法
ウォークスルー	アドホックテスト
チェック・リスト	ランダム・半ランダムテスト
同値分割／限界値分析	
状態遷移網羅	探索的テスト

現在の開発では，検証については，レビューでのウォークスルーの実施やチェック・リストの利用，状態遷移網羅に基づくテストなどの取り組みがなされている．一方，妥当性確認については，個人の経験に基づく部分が多いのが実情である．

MDD を利用した開発においても，現時点では標準のアクション言語が存在しないため，状態遷移の動作をコーディングする必要がある．そのため，この部分に対する検証は必要である．また，モデル自体の正しさを確認するために，従来の開発と同様に妥当性確認が必要である．

3. ソフトウェア開発への MDD 適用のアプローチ

2.1 で示した課題を解決するための我々のアプローチとして，状態遷移モデルに基づいた検証と，外部環境の動作をパターン化して定義する妥当性確認の手法を示す．

3.1. モデル駆動開発におけるテスト作業

2.1 に示した MDD の開発手順と組込み制御ソフトウェアの開発手順を，図 1 に示すように対応づける．標準文書や客先要求から作成するドメイン記述や要求仕様書（CIM）から，組込み制御ソフトウェアの仕様モデル（PIM），設計結果（PSM），さらに実装コードへと，段階的にモデルを詳細化する．

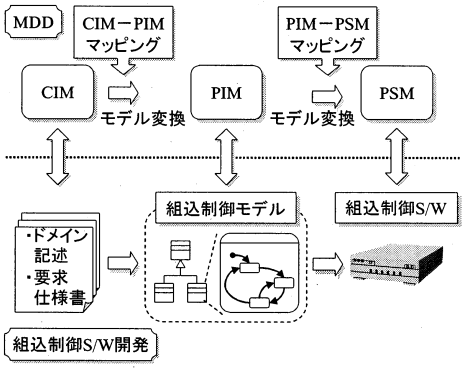

図 1 MDD と組込み制御 S/W 開発の対応

ここで，我々は，図 1 の開発手順に対してテスト仕様作成とテスト環境構築（テスト用ソフトウェアの開発）の作業を加えた開発手順（図 2）を，MDD を利用した組込み制御ソフトウェアの開発手順と定義する．

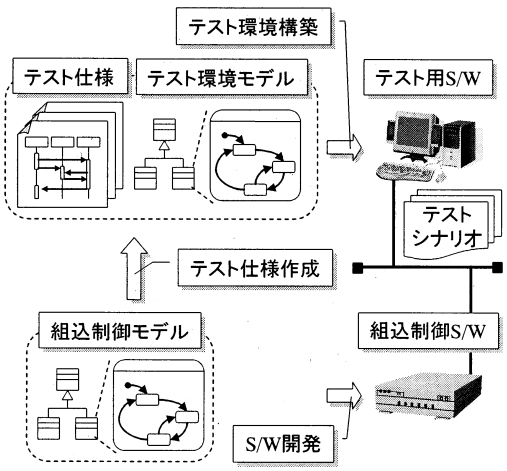

図 2 MDD を用いた組込み制御 S/W の開発

しかし，図 2 の開発手順の運用では，テスト仕様作成とテスト環境構築は手作業で行うことが多い．そのため，テスト仕様の抜けによる品質低下、テスト仕様作成とテスト環境構築の工数増加という課題が残る．

我々は，MDD を利用して状態遷移モデルからテスト仕様とテスト環境を自動生成するアプローチをとることで，網羅的なテストによる品質向上と，テスト作業の効率化を図ることを考えている．本稿では，状態遷移モデルから検証と妥当性確認のためのテスト仕様を生成する手法を提案する．

3.2. ソフトウェア検証のアプローチ

状態遷移モデルを中心に考えると，ソフトウェアの検証は，状態遷移モデルで定義した動作仕様を実現していないソフトウェアの動作を不具合として検出し，改修または制約事項とする活動である（図 3）．

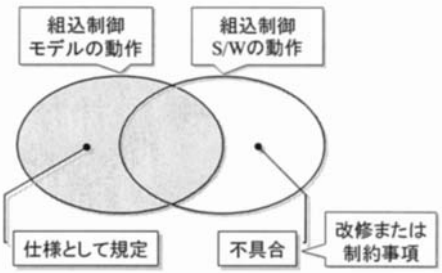

図 3 検証での不具合の検出

これまで，検証のためのテスト仕様を生成するため，状態遷移モデルの開始状態と停止状態を指定して経路探索ルールを適用し，遷移を網羅するパスを抽出する手法が提案されている[9]．この遷移網羅パスが，図 3 の「仕様として規定」された動作に相当する．

この手法では，状態遷移モデルの遷移パスを，状態を節点，遷移を枝とした木構造で表す．開始状態を表す節点 A から，その間の

任意の隣り合う節点 X から Y への遷移を再帰的に探索し，終了状態を表す節点 Z への経路を抽出する（図 4）.

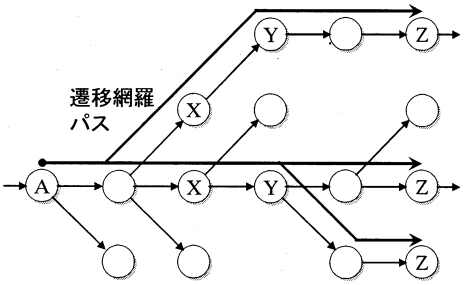

図 4 遷移網羅パスの抽出

抽出されたパスを元に，パスを構成する受信イベントを組込み制御ソフトウェアへの入力，遷移アクションを期待される出力や動作とし，検証のためのテスト仕様を生成する.

3.3. ソフトウェア妥当性確認のアプローチ

一方，ソフトウェアの妥当性確認は，状態遷移モデルを定義する際に仮定した外部環境の動作が起こったときの，ソフトウェアの想定外の動作を検出し，仕様変更・追加または制約事項とする活動である（図 5）.

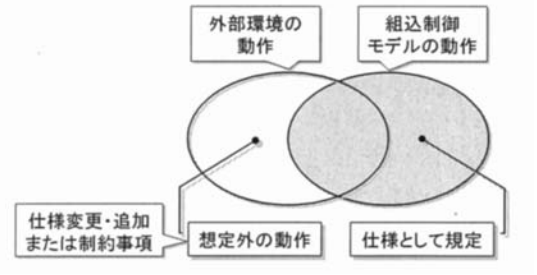

図 5 妥当性確認での想定外動作の検出

組込み制御ソフトウェアで重要視される，信頼性と効率性に関する妥当性確認のためのテストとして，信頼性テスト，回復テスト，性能テスト，容量テスト，ストレステスト，記憶域テストなどが実施される[7][10].

これらのテスト仕様の作成は，従来，テスト設計者の経験に頼ることが多かった．もし，テスト仕様を自動生成するなら，外部環境の動作をモデル化する必要がある．その方法として，以下の 3 つのアプローチが考えられる.

(a) 対向機器やネットワークなどの動作を状態遷移モデルで定義

(b) 組込み制御ソフトウェアの状態遷移モデルを元に，外部環境の動作を定義

(c) 外部環境からのイベント入力の順序やタイミングをパターン化して定義

(a)の方法は，外部環境の動作を厳密に定義できる．一方，テスト用に準備する外部環境のモデルが複雑で大規模になりやすい.

(b)の方法では，まず，状態遷移モデルを図 6 に示すような表形式で表現する（遷移アクションの記述は省略）.

State / Event	S0	S1	S2
E0	→S1	／	／
E1	／	→S2	／
E2	／	／	→S2
E3	／	／	→終了
E4	→終了	→終了	／

図 6 状態遷移モデルの表形式表現

図 6 では，未定義の動作を「／」で示している．例えば S1 状態での E0 受信などの動作であり，これは組込み制御モデルから見ると想定外の動作仕様となる．「／」を遷移なしとして，3.2 で示した経路探索ルールを適用すると，例えば図 6 では 4 通りの遷移網羅パスが抽出される．「／」を自己遷移とすると，16912 通りのパスが抽出される．よって，組込み制御モデルの想定内の動作が 4 通り，想定外の動作が 16908 通りとなる．また，パスを機械的に抽出するので，テスト仕様の有効性を判断しづらい.

(c)の方法では，状態遷移モデルから抽出した想定内の動作を表す遷移網羅パス（図 6 では 4 通り）と，イベント入力のパターンを組み合わせて，テスト仕様を生成する．これにより，テスト仕様の有効性が高くなる．

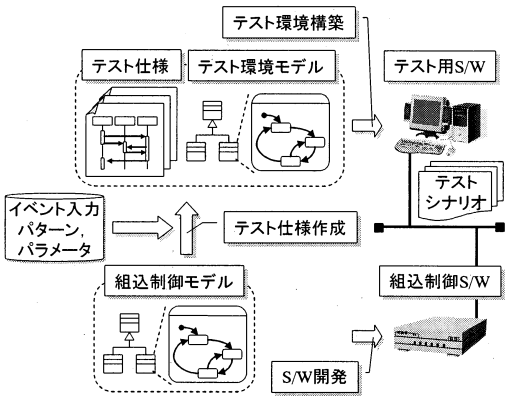

図 7 MDD を用いた妥当性確認のアプローチ

イベント入力のパターンを，入力順序をパターン化したイベント順序パターン，到着間隔など時間に関する性質をパターン化したイベント到着パターンに分類し，表 2 に示すように詳細に定義する．

表 2 イベント入力パターンの定義

分類	名称	説明
イベント順序	逆転	逆転して到着
	重複	重複して到着
	欠落	欠落して到着
イベント到着	周期的	ある周期で到着
	不規則	ある周期とジッタで到着
	制限	ある最小・最大間隔で到着
	バースト	ある密度で到着
	散発	ある内部・外部周期で到着

また，外部環境の動作を指定するパラメータとして，表 3，図 8 に示すイベント入力パラメータを定義する．

表 3 イベント入力パラメータ（順序）

名称	パラメータ
逆転	対象イベント数
重複	対象イベント数，重複数
欠落	対象イベント数

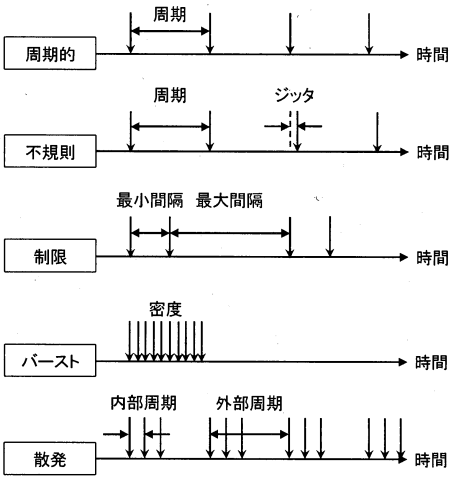

図 8 イベント入力パラメータ（到着）

今回は，外部環境モデルの規模とテスト仕様の有効性の観点から，(c)の方法をとる．

4. 通信制御プロトコルへの適用例

通信制御プロトコルを題材に，検証と妥当性確認のためのテスト仕様生成の例を示す．

4.1. 適用対象プロトコルの選定

ここでは，表 4 に示す機能を持った DHCP サーバ[11]を題材とする．

表 4 DHCP サーバの機能一覧

機能	説明
リース	パラメータ割り当て
リース延長	リース期間を延長
リリース	リースを終了
リース満了	リース延長要求がなく，満了
パラメータ取得	各種パラメータを取得

4.2. クラスおよび状態遷移モデルの作成

表 4 に示した機能を実現するクラスモデルと状態遷移モデルを作成する. 実際に作成したモデルの一部として, プロトコル処理の核となる Binding クラス（DHCP サーバ内で管理する DHCP クライアントに対応するクラス）のクラスモデルを図 9 に, 状態遷移モデルを図 10 に示す. ここで, 状態として, Init, Offering, Bound, Terminate, イベントとして evDiscover, evRequest, evRequestExt, evRequestOther, evRelease, evDecline, evInform, tm(OFFER), tm(LEASE)を定義する. OFFER, LEASE はそれぞれタイムアウトの秒数を表す.

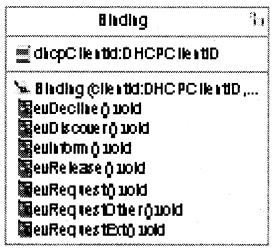

図 9 Binding クラスのクラスモデル

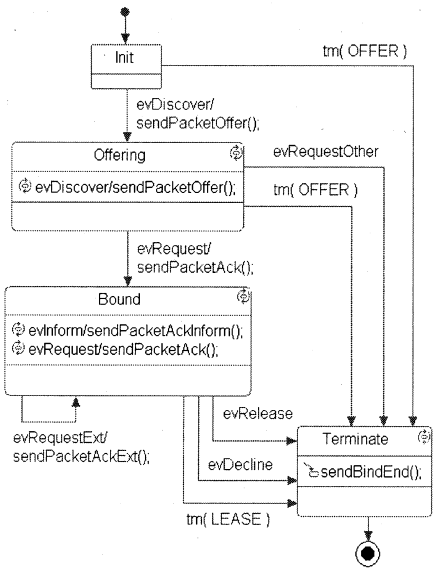

図 10 Binding クラスの状態遷移モデル

4.3. 遷移網羅パスからの検証テスト仕様の生成

図 10 の状態遷移モデルで定義している遷移には, 外部遷移, 自己遷移, 内部遷移がある. これらの遷移をすべて外部遷移で表現し, 経路探索ルールを適用して, 遷移網羅パスを抽出する. その結果, 全部で 101 通りのパスが抽出された. ただし, 自己遷移の繰り返し数は最大 1 回とした. 抽出されたパスの一例を図 11 に示す. このパスは「リース⇒リース延長⇒リリース」という動作を表す.

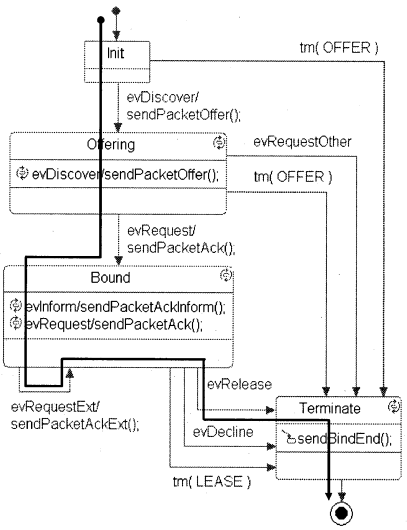

図 11 リース⇒リース延長⇒リリースの動作

状態遷移モデルから抽出されたパスを元に, 検証のためのテスト仕様を生成する. 図 11 に対応するテスト仕様を表 5 に示す.

表 5 リース⇒リース延長⇒リリースの検証テスト仕様

入力	期待する出力
Discover 送信	Offer 応答
Request 送信	Ack 応答
延長 Request 送信	延長 Ack 応答
Release 送信	応答なし

4.4. 外部環境動作のパターンを用いた妥当性確認テスト仕様の生成

4.3で状態遷移モデルから抽出した検証のためのテスト仕様に対して，我々が提案する手法を適用し，妥当性確認のためのテスト仕様を生成する．これにより，外部環境として仮定した動作が実際に起きたときの，通信制御ソフトウェアの想定外の動作を確認する．

具体的には，検証のためのテスト仕様に，表 2 のイベント入力パターンを組み合わせる．その際，表 3 のイベント入力パラメータを外部から与えるか，状態遷移モデルの定義から取り出し，外部環境の動作として指定する．以下では，逆転パターンおよび不規則パターンを利用して生成した妥当性確認テスト仕様の例を示す．

4.4.1. 逆転パターンを用いた妥当性確認

表 5 に対して逆転パターンを利用して生成した妥当性確認テスト仕様の一部を示す．延長 Request と Release，Request と Release の順番を逆転させている．

表 6 妥当性確認テスト仕様の一部（逆転）

入力	期待する出力
Discover 送信	Offer 応答
Request 送信	Ack 応答
Release 送信	**応答なし**
延長 Request 送信	**応答なし**
Discover 送信	Offer 応答
Release 送信	**応答なし**
Request 送信	**Ack 応答**
延長 Request 送信	延長 Ack 応答

4.4.2. 不規則パターンを用いた妥当性確認

表 5 に対して，イベント到着パターンの不規則パターンを利用する．ここで，図 10 の状態遷移モデルの時間に関する性質は，タイムアウトイベントで指定しているOFFER 秒，LEASE 秒が表している．これらはそれぞれ，Init／Offering 状態，Bound 状態に留まることができる時間を表す．

不規則パターンを用いて生成した妥当性確認テスト仕様の一部を示す．周期がOFFER 秒，ジッタが±1%の範囲でイベントが送信される．

表 7 妥当性確認テスト仕様の一部（不規則）

入力	期待する出力
(OFFER±0%)秒待機	
Discover 送信	Offer 応答
(OFFER−1%)秒待機	
Request 送信	**Ack 応答**
(OFFER＋1%)秒待機	
延長 Request 送信	延長 Ack 応答
(OFFER±0%)秒待機	
Release 送信	応答なし
(OFFER±0%)秒待機	
Discover 送信	Offer 応答
(OFFER＋1%)秒待機	
Request 送信	**タイムアウト発生**
(OFFER−1%)秒待機	
延長 Request 送信	応答なし
(OFFER±0%)秒待機	
Release 送信	応答なし

5. テスト仕様の生成手法の評価

状態遷移モデルに基づいて生成する検証のためのテスト仕様は，規定した仕様を実現しているかどうかを確認するためものであり，従来提案されているテスト手法と本質的には変わらない．

これに対し，今回提案した妥当性確認のためのテスト仕様は，組込み制御ソフトウェア

の状態遷移モデルとイベント入力パターンを組み合わせて定義した外部環境の動作仕様から生成することを狙っている．そのため，対向機器やネットワークなどの動作仕様をすべてモデル化する必要がなく，外部環境のモデルの規模を小さくすることができる．

また，今回定義したイベント入力パターンは，通信機器が対向機器とネットワークを介してメッセージ通信を行う状況でよく現れる性質のものである．近年の組込み制御機器は，ネットワークに接続する通信機能を持つことが多い．こうした通信機能を持つソフトウェアの品質を確認するために，パターンを利用して妥当性確認のためのテスト仕様を生成することは有効であると判断している．

6. まとめ

モデル駆動開発を実開発に適用するには，検証と妥当性確認を実施する必要がある．本稿では，その手法として，開発対象ソフトウェアの動作仕様を定義する状態遷移モデルから，検証と妥当性確認のためのテスト仕様を生成する手法を提案した．また，通信制御ソフトウェア開発を題材とした，本手法の適用例を示した．

今後の課題は，今回生成した妥当性確認のためのテスト仕様の有効性を，欠陥検出率などを用いて定量的に評価することである．また，今回利用しなかったイベント入力パターンについても，テスト仕様の生成とその有効性の評価を実施する予定である．

参考文献

[1] *MDD ロボットチャレンジ 2005 産学連携によるモデルベース組込み開発の実践*, 情報処理学会, 2006.

[2] *MDA Guide Version 1.0.1*, OMG omg/2003-06-01, OMG, June 2003.

[3] David S. Frankel, *MDA™ モデル駆動アーキテクチャ*, エスアイビー・アクセス, 2003.

[4] *Unified Modeling Language: Superstructure Version 2.0*, OMG formal/05-07-04, OMG, August 2005.

[5] *Object Constraint Language Version 2.0*, OMG formal/06-05-01, OMG, May 2006.

[6] Jos Warmer, Anneke Kleppe, *UML/MDA のためのオブジェクト制約言語 OCL*, エスアイビー・アクセス, 2004.

[7] Glenford J. Meyers, *ソフトウェア・テストの技法 第2版*, 近代科学社, 2006.

[8] Rick D.Craig, Stefan P.Jaskiel, *体系的ソフトウェアテスト入門*, 日経 BP, 2004.

[9] Ivan Bratko, *PrologとAI② AIプログラミング*, 近代科学社, 1996.

[10] Watts S. Humpyrey, *ソフトウェアプロセス成熟度の改善*, 日科技連, 1991.

[11] Ralph Droms, "Dynamic Host Configuration Protocol," *RFC2131*, IETF, March 1997.