モデル検査技術による上流工程での効率的な設計検証

Efficient Design Verification Using Model Checking Techniques in Upstream Processes during Software Development

高田　沙都子　　森　奈実子　　村田　由香里
■TAKADA Satoko　■MORI Namiko　■MURATA Yukari

近年，ソフトウェアの複雑化により，設計の抜け漏れを人手によるレビューだけで防ぐことが難しくなってきている。設計の抜け漏れをソフトウェア開発の下流工程で発見し対策する場合，上流工程での対策と比較して品質コストが増大することが知られており，上流工程での設計検証の効率化が重要になっている。モデル検査技術は，システムの仕様が検証条件を満たすかどうかを自動的かつ網羅的に検証できる技術である。一方，モデル検査の適用は時間が掛かり，検査者の専門的知識やノウハウを要する点が課題であった。

東芝は，モデル検査に要する作業を自動化したモデル検査自動化ツールを開発することで，その課題を解消した。

The complexity of software in recent years makes it difficult to completely eliminate design failures and omissions by manual design reviews alone. When problems occur in downstream processes during software development, the increased costs of measures to assure product quality are generally higher than those of measures when problems occur in upstream processes. Therefore, efficient design verification in upstream processes is essential. Model checking makes it possible to automatically and exhaustively verify in upstream processes whether the performance satisfies requirements. However, in order to apply model checking techniques, reduction of the time required and the need for experts with technical know-how are significant issues.

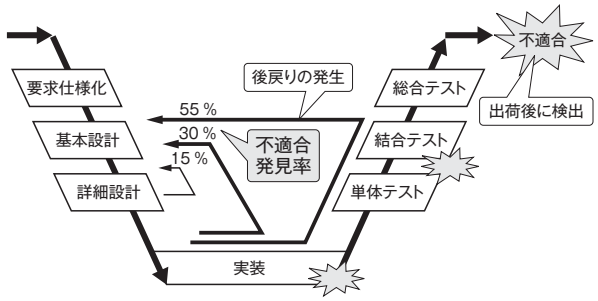
To address these issues, Toshiba has developed an integrated automatic model checking environment and confirmed its effectiveness by applying it to digital product design.

1　まえがき

近年，ソフトウェアは大規模・複雑化しており，仕様の全てのふるまいを網羅した検証を人手で行うことは困難になってきている。更に，システムの設計段階で不適合が混入すると，ほとんどの誤りがソフトウェア開発のテスト工程以降になるまで発見できず，設計工程までさかのぼる後戻りが発生するため，修正コストが増大する（**図1**）。

このような問題を解決する手法として，モデル検査技術が注目されている。モデル検査技術はソフトウェア開発の上流工程に適用可能で，システムのふるまいを網羅的に探索することで，システムの不適合を早期に発見できる技術である。

東芝では社会インフラや電子デバイスなどの分野に対し，上流工程でモデル検査技術を適用した実績がある。今回その実績を生かし，更に適用加速のためのモデル検査自動化ツールを開発した。ここでは，開発したツールの概要と，実際に開発ツールを用いたモデル検査の適用事例を通してモデル検査の適用効果について述べる。



＊不適合発見率は経済産業省「プロジェクト責任者向け調査」[1]を元に作成

図1．ソフトウェアの開発工程と後戻り ― 設計工程で混入した不適合は，多くは実装工程やテスト工程以降で検出されるため，後戻りが発生し品質コスト増大の要因となっている。

Software development process and occurrence of regression

2　モデル検査技術

2.1　モデル検査技術の適用プロセス

モデル検査技術では，動作仕様のふるまいをモデル化することで，モデル検査器を用いて検査を実施する。モデル化した動作仕様のふるまいの正しさを，システムが取りえる全ての状態に対し網羅的に探索することで検証できる。上流工程で適用が可能なため，不適合を早期に発見でき，後戻りを最小限に抑えることができる。

モデル検査適用時の処理の流れを**図2**に示す。モデル検査の入力は，動作仕様をモデル化した検査用モデルと，期待される動作を表わす制約条件を数学的に記述した検査式の二つである。対象システムの動作仕様は，例えばモデル検査

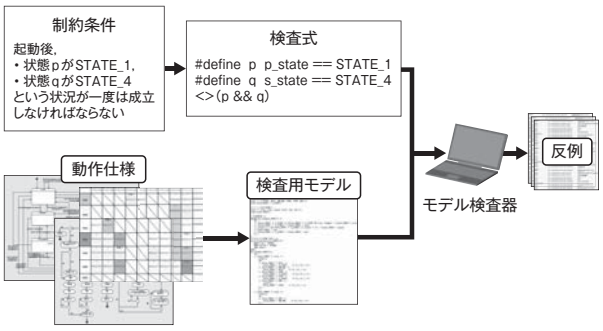


図2．モデル検査時の情報の入出力 ― 動作仕様と制約条件から検査用モデルと検査式を作成し，モデル検査を実施する。網羅的に探索し検査式を満たさないシーケンスを反例として出力する。

Input and output of model checking

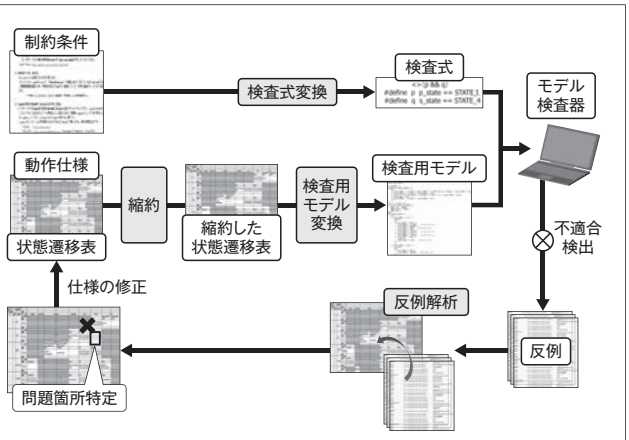


図3．モデル検査適用時に必要な手順 ― 実際の適用には動作仕様の縮約や効率的な反例解析が必要になる。これらモデル検査適用時に必要になる作業の自動化やサポートを行うモデル検査自動化ツールを開発した。

Workflow of model checking

器SPIN[2]では，SPIN独自の記述言語Promelaを用いて記述することでモデル化する。これを検査用モデルと呼ぶ。制約条件はSPINの場合，線形時相論理[3]（LTL：Linear Temporal Logic）を用いて数学的な式に置き換える。これを検査式と呼ぶ。

検査用モデルと検査式をモデル検査器に入力することで，検査式に違反する動作が検査用モデルに発生しないかを網羅的に探索する。検査の結果，検査式を満たさないケースを検出した場合，モデル検査器は，検査式を満たさない状況に至る動作シーケンスを反例として出力する。この反例を検査者が解析することによって，動作仕様の誤りを発見し修正する。検査者は反例が出なくなるまで動作仕様の修正と検査を繰り返すことで，最終的に正しい動作仕様を得ることができる。

2.2　モデル検査技術の適用上の課題

モデル検査を実際の開発で適用するには，次のような課題がある。

(1)　検査用モデルを記述するには独自の言語を習得する必要がある。

(2)　検査式の作成では数学的な知識が必要になる。

(3)　出力される反例は可読性が低く，膨大な量になることもあるため，解析にはノウハウが必要になる。

(4)　検証対象が複雑な場合，状態探索中に状態数が爆発的に増大することでメモリ不足が起きることがある。この状態爆発が発生すると検査が途中で打ち切られ，検査結果が得られないため，状態爆発を抑制する必要がある。

これらの課題を解決することで，モデル検査のより効率的な適用が期待できる。

3　モデル検査自動化ツール

モデル検査を適用する際に必要になる検査者の作業プロセスを**図3**に示す。2.2節で述べたモデル検査技術の課題を解決するために，検査者による動作仕様の縮約や反例の解析を効率的に行うといった作業が必要になる。

これらの作業手順を踏まえ，モデル検査の適用プロセスを自動化し，サポートを行うモデル検査自動化ツールを開発した。モデル検査器には2.1節で述べたSPINを採用している。このツールは，2.2節の課題に対し次に示す機能で解決を図っている。

課題(1)　検査用モデル生成機能

課題(2)　検査式生成機能

課題(3)　状態遷移表−反例連携表示機能

課題(4)　状態遷移表縮約機能

ここでは，それぞれの機能について述べる。

3.1　検査用モデル生成機能

検査用モデル生成機能は，SPIN独自の記述言語Promelaで記述された検査用モデルを自動生成する機能である。このツールでは動作仕様として状態遷移表を定義できるようになっており，他に検証対象となるシステムで扱っている変数や，アクション，ガード条件などを定義する表を備えている。変数定義表からは，変数定義用のPromelaが出力され，状態やイベント定義のPromelaは状態遷移表から出力される（**図4**）。Promelaのプロセスは状態遷移表の数に応じてアクションとガード条件の定義表を参照して出力される。ユーザーは状態遷移表と定義表を入力すれば，その入力を元にPromelaで記述された検査用モデルを生成できる。

3.2　検査式生成機能

検査式生成機能は，ツールの操作パネル上に表示される単語を用いて制約条件を日本語文で作成することで，LTLで記述された検査式を自動的に生成する機能である。ユーザーは制約条件を日本語文で記入することでLTLを意識せずに検査式を得ることができる（**図5**）。

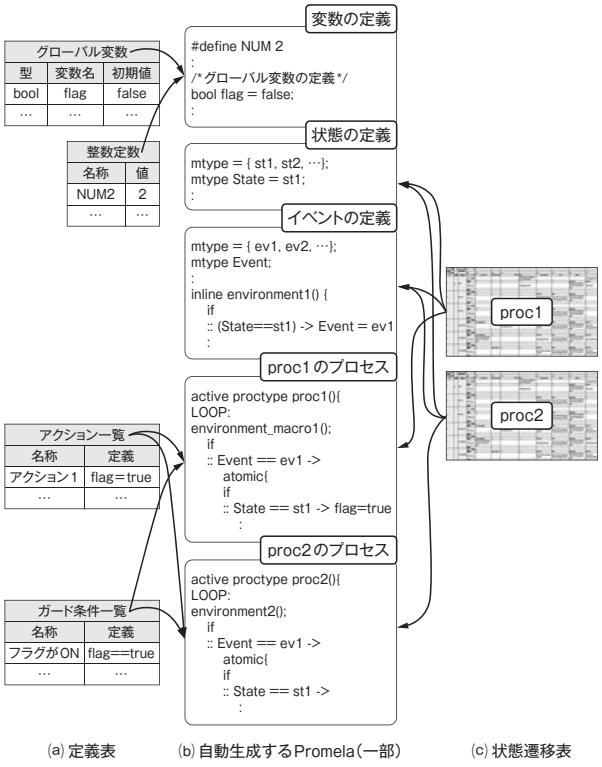


図4. 検査用モデル自動生成機能 — 状態遷移表と，変数や，アクション，ガード条件などの定義表を元に検査用モデルを自動生成する。

Automatic model generation function

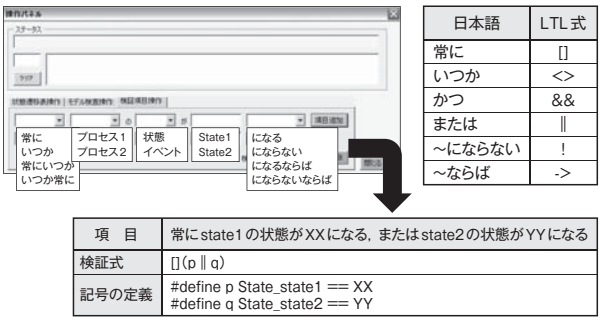


図5. 検査式自動生成機能 — プルダウン形式で単語を選択することで，一定のルールに従って作成された日本語文の制約条件から検査式を自動生成する。

Automatic linear temporal logic (LTL) formula generation function

3.3 状態遷移表−反例連携表示機能

状態遷移表−反例連携表示機能は，チャート化によって可読性を向上させた反例と，状態遷移表を連携して表示させる機能である。SPINが出力したテキスト形式の反例から，解析に必要な状態だけを抽出し，変数や各プロセスの状態やイベントの実行ログをチャート化することで可読性が向上する（**図6**）。

反例チャート上のあるステップを選択すると，選択したステップに対応した状態遷移表のセルを強調表示する（**図7**）。

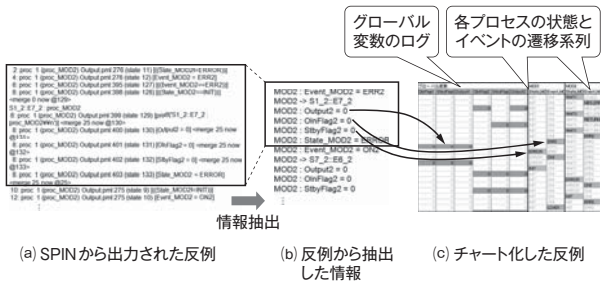


図6. チャート化した反例 — SPINから出力された反例は，そのままでは可読性が低いため，必要な情報だけを抽出し，チャート化して表示することで可読性を向上させる。

Chart diagram of counterexample

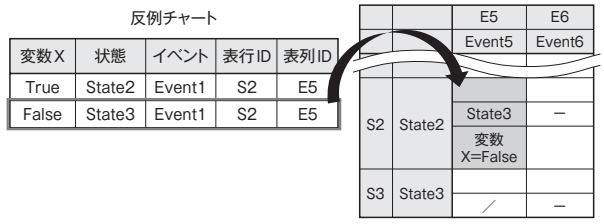


図7. 状態遷移表−反例連携表示 — 反例チャート上をクリックすると対応する状態遷移表のセルが強調表示され，状態遷移表と反例を比較しながら解析を行うことができる。

Counterexample display function

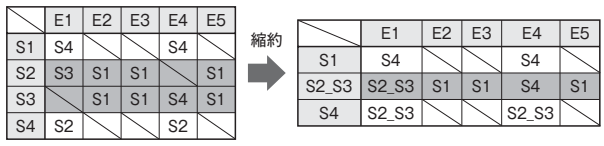


図8. 状態の縮約 — 状態S2，S3を同値とする状態群とみなした場合，状態S2，S3を一つの状態として扱い遷移先をまとめる。縮約前に状態S2，S3を遷移先としていた箇所は，同値化した状態S2_S3へ遷移するよう変更する。

Contraction of states

ユーザーは，この機能を用いて動作仕様と反例を比較表示しながら解析を進めることができる。

3.4 状態遷移表縮約機能

状態遷移表縮約機能は，検査に関連のあるイベントと状態を残し，状態遷移表を自動的に縮約する機能である。この機能では当社が開発した状態縮約技術[4]を活用している。開発者が検査のうえで冗長な中間状態などを同値の状態群とみなすことで，複数の状態を同一状態に縮約する（**図8**）。また，遷移元と遷移先が一致する状態遷移がある場合，一致する箇所を同一イベントとして統合する（**図9**）。

縮約された状態遷移表を用いて検査することで，探索時の状態数を抑制したモデルを作成できる。

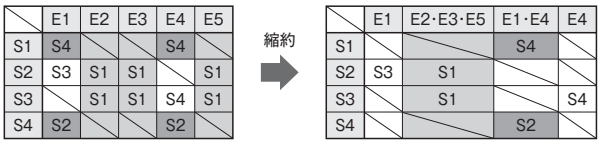


図9. イベントの縮約 — イベントE2，E3，E5は全ての状態で同じため，3イベントを一つのイベントに統合する。イベントE1，E4は遷移先が同一な状態S1，S4からの遷移だけを統合する。

Contraction of events

4 適用事例

ここでは，モデル検査自動化ツールを利用した設計品質向上の適用事例として，2011年に開発を進めていたデジタルテレビの録画機能の設計改善について述べる。

4.1 検証対象

この適用事例の検証対象はデジタルテレビの録画機能の実行管理モジュールである。実行管理モジュールは録画予約を管理する予約テーブルを持つメインタスクと，録画状態や録画実行を管理する複数の録画実行タスクから構成される（**図10**）。

実行管理モジュールが実現する録画機能として，例えば次のような二つの機能がある。

(1) 3H追従機能　　録画開始時に放送波に番組情報が見つからなかった場合，最大3時間予約した番組を探す機能

(2) イベントリレー機能　　録画中の番組が現在のチャンネルで放送が終了せず，他のチャンネルで放送が継続される場合に，引き続き録画する機能

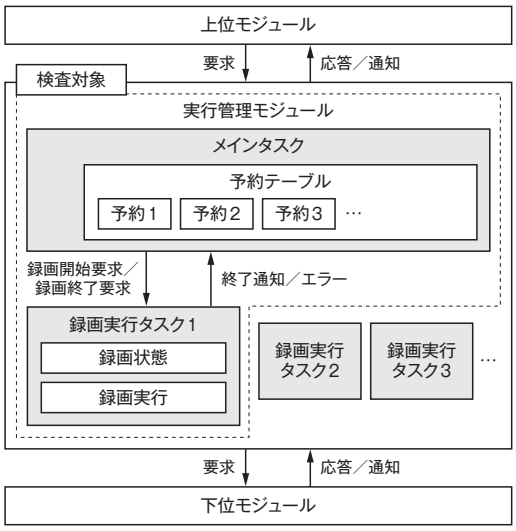


図10. 実行管理モジュールの概念図 — 実行管理モジュールは，録画予約を管理する予約テーブルを持つメインタスクと，録画状態や録画実行を管理する複数の録画実行タスクから構成される。

Configuration of execution management module

これら機能要求仕様に対する制約条件を，状態遷移設計として定義された動作仕様が満たしているかを検査した。

4.2 モデル検査自動化ツールの適用

この検証対象に対して，モデル検査自動化ツールを利用して，従来の開発の流れにモデル検査を組み込んだ場合のワークフローを**図11**に示す。従来の開発の流れでは，漏れのないレビューを行うことは人手では難しい。検査対象を例に挙げると，三つある状態遷移表の遷移数の組合せを121,716（=126×21×46）通りチェックする必要があった。モデル検査では更にアクションの実行やイベントの受付といった実行タイミングも網羅した探索が短時間で行える。制約条件にもよるが5秒～30分程度で探索を行い，不適合があった場合は反例を出力する。

4.3 検査結果

このツールを適用した結果，6件の不適合を検出し，不適合箇所を修正することで改善を行うことができた。ここでは代表例として，制約条件“タスク間の状態の不整合がないか”で検出された不適合について述べる（**図12**）。

録画実行タスクはメッセージキューを使ったタスク間通信を行っており，キューの先頭にあるメッセージから処理する。このとき，3H追従開始要求を受信する直前に録画実行タスクが定期イベントを受信していた場合に不適合が発生する反例があった。この定期イベントによる処理を実行して，“3H追従状態”である全ての予約の番組情報が放送波から検出されると，録画実行タスクは自身の状態を“3H追従なし状態”に遷移させる。そして，“3H追従状態”である予約は全て“予約状態”に遷移する。ところが，録画実行タスクはメッセージキューに残った3H追従開始要求の処理を実行して自身の状態を

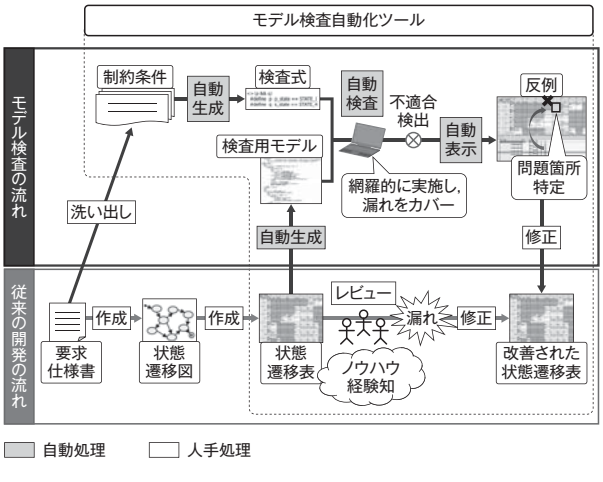


図11. モデル検査を組み込んだ開発の流れ — 従来の開発の流れにおいて，漏れのないレビューを行うことは人手では難しいため，モデル検査を組み込んで漏れをカバーする。

Workflow of software development applying model checking

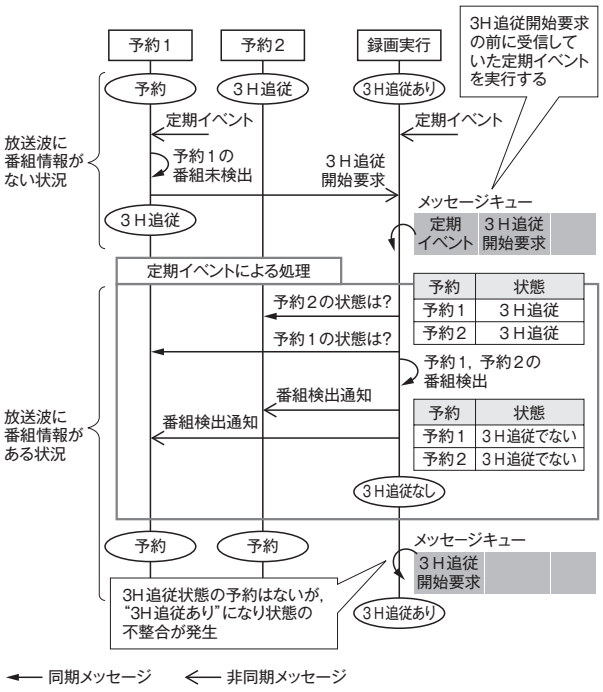


図12．反例シーケンス — 制約条件“タスク間の状態の不整合がないか”を満たすかモデル検査をした場合に，検出された反例の流れを示すシーケンスである。メッセージの処理タイミングによっては，予約の状態と録画実行タスクの状態とで不整合が起きてしまう。

Sequence of counterexample

“3H追従あり状態”に遷移させてしまい，予約の状態との不整合が起きてしまった。この不適合に対しては，3H追従開始要求を受信時に予約の状態を確認して整合性をとる処理を追加することで改善した。

このようなタイミングによって問題となる不適合をテスト工程で発見することは一般に困難であるが，モデル検査では検査式に違反する動作が発生しないかを網羅的に探索するため，この事例のように設計時点で不適合を検出できる。

4.4　モデル検査自動化ツールの効果

この適用事例では検査用モデルの生成と，一部検査の自動実行にモデル検査自動化ツールを利用したことで作業を効率化できた。この適用事例と同じモデルを手作業で作成した場合，およそ16人・時（5（人・分/遷移）×193遷移）掛かると想定されるが，モデル検査自動化ツールでは10秒程度で検査用モデルを作成できた。

また，状態遷移表から検査用モデルを生成し自動検査することで，モデル検査やSPINを意識することなく，状態遷移表ベースでの検査や反例解析を実施できた。

5　あとがき

モデル検査技術の時間的コストと専門知識やノウハウを要する課題を解決するため，モデル検査自動化ツールを開発した。実際の開発製品に適用することで，その効果を確認した。

今後は更に適用事例を増やすとともに，他の検証技術による上流工程での効率的な設計検証の実現を進めていく。

文　献

(1) 経済産業省．“プロジェクト責任者向け調査”．<http://www.meti.go.jp/policy/mono_info_service/joho/downloadfiles/2010software_research/index.htm>，（参照2012-10-16）．
(2) "ON-THE-FLY, LTL MODEL CHECKING with SPIN". SPIN Homepage. <http://spinroot.com/spin/whatispin.html>, (accessed 2012-10-15).
(3) 中島　震．SPINモデル検査−検証モデリング技法．東京，近代科学社，2008，238p．
(4) 進　博正 他．“商状態遷移系を用いたテストケース生成方法”．組込みシステムシンポジウム2008論文集．東京，2008-10，情報処理学会 組込みシステム研究会．情報処理学会，2008，p.167−174．
(5) Holzmann, G. J. THE SPIN MODEL CHECKER: Primer and Reference Manual. USA, Addison-Wesley Professional, 2003, 608p.
(6) 池田信之 他．モデル検査によるソフトウェア上流設計の品質向上技術．東芝レビュー．**64**，4，2009，p.40−43．

高田　沙都子　TAKADA Satoko
ソフトウェア技術センター　ソフトウェア設計技術開発担当主務。ソフトウェア上流設計・検証技術の開発に従事。情報処理学会会員。
Corporate Software Engineering Center

森　奈実子　MORI Namiko
ソフトウェア技術センター　ソフトウェア設計技術開発担当。ソフトウェア上流設計・検証技術の開発に従事。情報処理学会会員。
Corporate Software Engineering Center

村田　由香里　MURATA Yukari
ソフトウェア技術センター　ソフトウェア設計技術開発担当参事。ソフトウェア上流設計・検証技術の開発，及びソフトウェア開発プロジェクトの支援業務に従事。
Corporate Software Engineering Center