社団法人 情報処理学会 研究報告
IPSJ SIG Technical Reports

2006－SLDM－127（1）
2006／11／28

仕様書から検証シナリオを生成する手法

大石 亮介　松田 明男　岩下 洋哲　高山 浩一郎

株式会社富士通研究所 〒211-8588 神奈川県川崎市上小田中 4-1-1
E-mail: roishi@labs.fujitsu.com

あらまし　. SoC や組み込みソフトウェアの設計において，バグの早期発見のためには，仕様設計工程で，検証を効率的に行う必要がある．従来の第三者検証技術では，自然言語で書かれた仕様書から検証シナリオを生成していたため，シナリオを人手で記述する必要があった．本研究では，モデルチェッキングの技術を利用して，操作とその事前条件，事後条件，不変条件，およびシナリオの制約条件から，検証シナリオを自動的かつ網羅的に生成する手法を提案する．入力は全て仕様の UML モデルから抽出する．実験の結果，十分に短い実行時間で仕様を網羅するような検証シナリオを生成できることが確認できた．

キーワード　仕様，検証，テスト，シナリオ，モデルチェッキング，UML

A methodology of generating verification scenarios from specification

Ryosuke OISHI,　Akio MATSUDA,　Hiroaki IWASHITA　and　Koichiro TAKAYAMA

† Fujitsu Laboratories LTD.　4-1-1 Kamikodanaka Yamada, Minato-ku, Tokyo, 105-0123 Japan
‡ R&D Division, Osaka Corporation　4-5-6 Kawada, Suita-shi, Osaka, 565-0456 Japan
E-mail: roishi@labs.fujitsu.com

Abstract　For designing SoC or embedded software, it is required that they have to be verified immediately in the level of specification design. By previous works, verifiers had to write scenarios without automation because the scenarios are generated from specification which is wrote in natural language. In this paper we propose a methodology how to generate verification scenarios fully and automatically, from pre-conditions and post-conditions and constraints in the scenarios by using model-checking tool. Experiment results show that verifications scenarios which fully cover specification are generated in short time.

Keyword　specification, verification, test, scenario, model checking, UML

1. はじめに

SoC や組み込みソフトウェアの設計において，バグの早期発見のためには，仕様設計工程で，検証を効率的に行う必要がある．従来の検証技術では，自然言語で書かれた仕様書から検証シナリオを生成していたため，シナリオを人手で記述する必要があった．本研究では，モデルチェッキングの技術を利用して，操作とその事前条件，事後条件，不変条件，およびシナリオの制約条件から，検証シナリオを自動的かつ網羅的に生成する手法を提案する．入力は全て仕様の UML モデルから抽出する．

2. 従来研究とその問題点

論理設計において検証にかかる工程は非常に大きく，回路規模にもよるが全体の 6 割以上を占めていると言われている．検証を効率的に行うには，仕様設計段階での効率的な検証が必要とされる．仕様設計から使える検証手法にはいくつかあるが，シミュレーションを用いた動的検証手法においては，以下のようなものがある．文献[1][2]では，仕様の UML モデルから検証シナリオおよび検証パラメタを抽出して検証を行う手法を提案している．この方法では仕様に対して少なくとも機能カバレッジを保証することができる．しかし機能カバレッジを確保するために必要な検証シナリオを生成する手法は，UML モデルに基づいているため系統的であるとはいえるが，人手に頼っている．

一方，文献[3]では仕様の UML モデルのうちシーケ

ンス図に相当するメッセージシーケンスチャートから検証シナリオを自動的に生成することができる．この方法は有力であり，シーケンス図に相当するモデルが与えられていれば，人手を介することなく検証シナリオを生成することができる．しかし，大規模なシステムにおいては，機能カバレッジを保証するのに必要なシーケンス図は膨大な量になる可能性が大きい．これらのシーケンス図は結局人手で作成しなければならない．

その他，検証シナリオを自動的に生成するツールはいくつか実用化されている．しかしその多くは実装のソースコードを元にして検証シナリオを生成している．この方法では実装がなければ検証シナリオの生成ができず，検証作業を進めることができない．仕様からのトップダウン設計フローの中で効率的に検証を行うためには，実装設計のフェーズに依存しない検証プロセスが必要である．また実装に含まれたバグがそのまま検証シナリオにまで継承され，そのシナリオではバグを発見できない可能性もある．すなわち，検証シナリオは実装に依存しないものが要求される．

3. UML モデルを用いたシナリオ分析手法

上記のような従来研究の成果およびふまえて，既存手法における問題点を可能な限り回避できるような検証シナリオ生成手法を考案した．概略を図 1に示す．

まず重要なことは，この手法の入力に使われる入力は，仕様書とシナリオの終了条件だけであり，検証シナリオが実装に依存しないことである．これには次のようなメリットがある．

- 実装のわかりづらい誤りやデッドコードが検証シナリオに含まれることを防ぎ，検証シナリオにバグが混入することを未然に防ぐ．
- 実装がなくても仕様書だけを入力として検証シナリオを生成することができ，設計の全体工程において検証作業を前倒しすることが可能になる．
- 設計者ではなくても検証シナリオを生成できるため，第三者検証に用いることができる．

次に本手法の重要な点として，分析の留度と出力が明示的に定められていることである．すなわち，分析の留度はユースケースの留度とアクティビティの留度が一致する時点であり，アクティビティを操作としたときに定義されるべき事前条件，事後条件および操作の種類の集合である．操作の種類は，次の 3 つに分類される

- 外部からの入力を必要とする(input)
- 外部への出力がある(output)
- 入力も出力もない(process)

入力と出力を同時に要求されるような操作は存在しないものとする．もしあれば，操作を分割して少なくとも input 操作と output 操作に分ける．

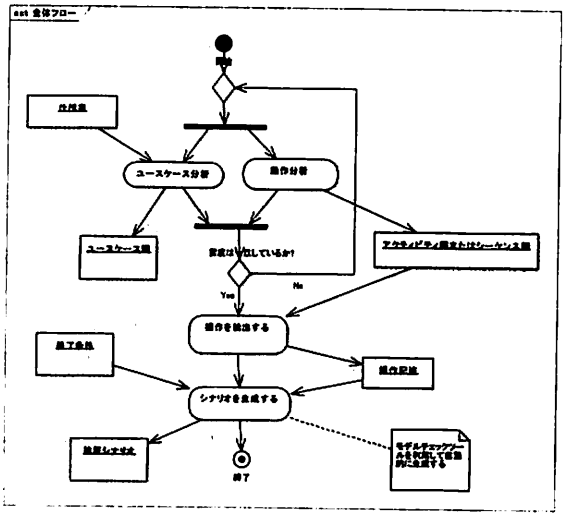

図 1 全体フロー

分析の具体的なフローは次のようになる．

1. ユースケース分析手法を用いて仕様書からユースケースを抽出し，ユースケース図を記述する．これは[1][2]の手法そのものである．
2. 動作分析手法を用いて仕様書からアクティビティを抽出し，アクティビティ図を記述する．これも[1][2]の手法そのものである．アクティビティ図の代わりにシーケンス図を用いてもよい．シーケンス図は原則として分岐を持たないがアクティビティ図には分岐を記述することができるため，1 つのアクティビティ図で複数のシーケンスを記述できる．本稿ではアクティビティ図を用いた場合のみについて説明する．
3. 抽出されたユースケースとアクティビティの留度が一致しているかどうかを確認する．留度が一致しない場合は，手順1または2に戻る．また一つのアクティビティで入力と出力が同時に起きていないことを確認する．
4. モデルから操作を抽出する．このとき，各操作に対して事前条件，事後条件および入力，出力を抽出する．また，各操作を入出力のタイプによって 3 種類に分類する．
5. BDD ソルバに操作記述と終了条件を与え，シナリオを生成させる．BDD ソルバはモデルチェッカ BINGO[4][5]で使用しているものを用いる．

この手法で得られた UML モデル，操作記述および生成された検証シナリオには次のような特徴がある．

- UML モデルはユースケース図とアクティビティ図で構成され，それ以外の図は必要ではない(あってもよいが，本手法では使用しない)．
- 一つのユースケースは必ず一つのアクティビティに対応している．
- あるユースケースが包含または拡張の関係によるサブユースケースを持っている場合，対応するアクティビティは必ずサブアクティビティを持つ．
- 最下層にあるユースケース，またはアクティビティは操作そのものである．
- 各操作は必ず事前条件，事後条件および入力・出力を持つ．事前条件を満たしているとき，その操作はいつでも実行可能であるものとし，操作が終了したときには必ず事後条件を満たしている．
- 各操作は，実質的に原因結果グラフ[7]であると言える．ここで原因は事前条件，結果は事後条件である．操作 A および操作 B について，もし操作 A の事後条件が操作 B の事前条件であるとき，操作 A と操作 B を繋げて一つのパスを生成できる．この手順を全ての操作に対して繰り返すと，シナリオを生成することができる．このアルゴリズムはモデルチェッキングのアルゴリズムと本質的に同一であり，モデルチェッキングでしばしば使われる BDD(Binary Decision Diagram)を用いたソルバを使って解くことが可能である．

本手法を用いて作成したユースケース図とアクティビティ図の例を図 2および図 3に示す．また，アクティビティ「直線を描画する」に相当する操作記述を表 1に示す．

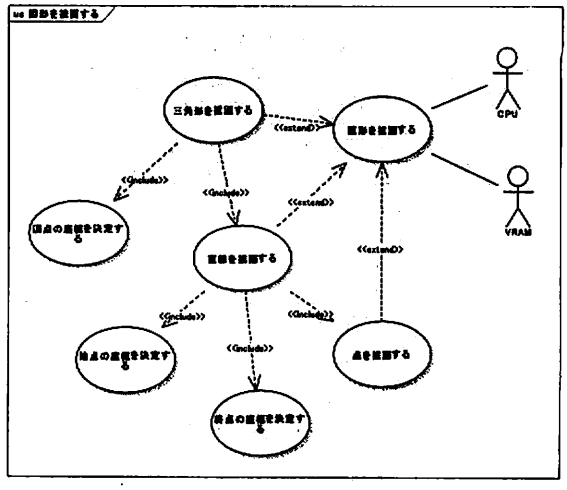

図 2 ユースケース図

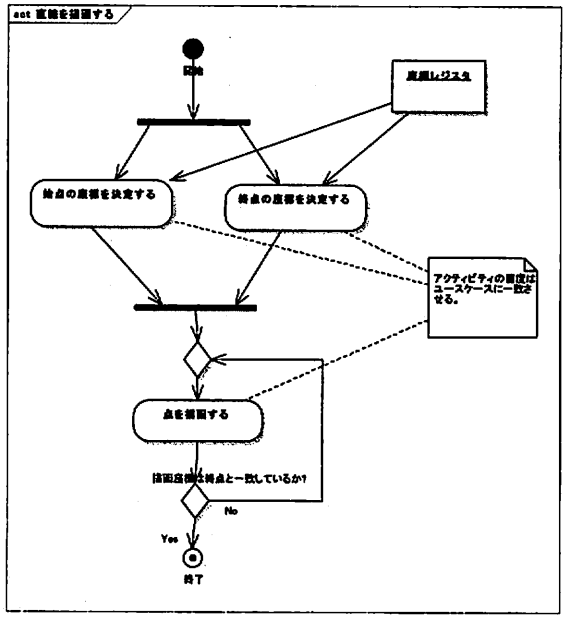

図 3 アクティビティ図

表 1 操作記述

操作名	直線を描画する
概要	VRAMに直線を描画する
事前条件	ステータスレジスタがline readyである
事後条件	ステータスレジスタがdot readyである
入力	始点座標 傾き 終点座標 status
出力	始点座標 終点座標 status

アクション	process 直線を描画する(始点座標, 傾き, status) {
	assert((終点座標.y - 終点座標.y) / (終点座標.x - 終点座標.x) == 傾き);
	assert(status == line_ready);
	{
	status = dot_ready;
	}

表 2 実験結果

入力	仕様書の量	31ページ
モデリング	ユースケース図枚数	1
	アクティビティ図枚数	6
	ユースケース数	26
	アクティビティ数	
	操作記述の全行数	368
ツール実行	生成シナリオ数	13
	実行時間	24.0sec.
	使用メモリ	60.8MB
	最大BDDノード数	774,790個

4. 適用事例

今回提案したシナリオ生成手法を無線通信システムのプロトコル制御ユニットの実装仕様書に対して適用し，下表の結果を得た．仕様書のモデル化およびUMLのユースケース図，アクティビティ図，操作記述の作成行った．さらに，操作記述から検証シナリオの生成の実験を行った．

適用は仕様設計者および実装設計者ではない第三者が検証者として行った．検証者は仕様書のみを入力としてモデル化を行い，実装のソースコードは全く参照しなかった．

実験結果を表 2に示す．ユースケース数，およびアクティビティ数(すなわち，操作の数)は 26 であり人手で記述するのに十分に小さい量である．操作記述の総行数も 368 行であり，決して多いとは言えない．BDDソルバの実行は 1 分以内に終了しており，実用上十分な速度であると言える．またこの検証シナリオが仕様を網羅していることを，レビューによって仕様設計者に確認した．

5. まとめ

本研究では仕様起点となる検証シナリオを自動的かつ網羅的に生成する手法を示し，そのために必要となる仕様のモデリング手法を提案した．

文　献

[1] Qiang Zhu, Ryosuke Oishi, Takashi Hasegawa, and Tsuneo Nakata. System-on-Chip Validation Using UML and CWL. Proc. 2nd IEEE/ACM/IFIP Intl Conf. on Hardware/Software Codesign and System Synthesis, pp.92-97, 2004.

[2] 大石 亮介, 朱 強, 中田 恒夫, 峰 正高, 黒木 健一郎, 遠藤 陽一, 長谷川 隆. UML を用いた上流検証手法. DA シンポジウム 2004, pp.79-84.

[3] Praveen K. Murthy, Sreerunga P. Rajan, and Koichiro Takayama. High level hardware validation using hierarchical message sequence charts. High-Level Design Validation and Test Workshop, pp.167-172, 2004.

[4] Hiroaki Iwashita and Tsuneo Nakata. Forward Model Checking Techniques Oriented to Buggy Designs. Proceedings of the IEEE/ACM International Conference on Computer-Aided Design, pp.400-404, 1997.

[5] 桑村 慎哉, 高山 浩一郎. 形式的検証ツール BINGO の実設計適用 ～モデル検査ツールを用いたシステム LSI の検証事例～. DA シンポジウム 98, 1998.

[6] 朱 強, 桑村 慎哉, 松田 明男, 庄司 稔, 中田 恒夫. UML を用いたシステムレベル設計手法の提案, DA シンポジウム 2002, pp.31-36, 2002.

[7] Glenford J. Myers, 松尾正信訳. 「ソフトウェア・テストの技法」, 近代科学社.

[8] E.Dustin, J.Rashka, J.Paul, 向井清訳.「自動ソフトウェアテスト」, ピアソンエデュケーション.