組込みソフトウェア開発のイン-デザイン モデル検査

－設計工程における仕様書のモデル検査の提案－

篠崎 孝一† 水口 大知†† 石井 健志†††

†関西電力株式会社電力技術研究所 〒661-0974 兵庫県尼崎市若王子 3-11-20
††独立行政法人産業技術総合研究所システム検証研究ラボ 〒661-0974 兵庫県尼崎市若王子 3-11-46
†††東光精機株式会社NW機器開発部 〒566-8686 大阪府摂津市千里丘 3-14-40

E-mail: †K576789@kepco.co.jp, ††daichi.mizuguchi@aist.go.jp, †††TAKESHI_ISHII@toko-s.co.jp

あらまし ソフトウェア開発の設計工程において仕様書のモデル検査を行う「イン-デザイン モデル検査」を提案する．これにより，仕様書の不備の早期検出・修正を目指す．さらに，このイン-デザイン モデル検査を，実際の組込みソフトウェア開発に対して試行し，その効果，問題点，および実現性について検討した結果を報告する．

キーワード モデル検査，ソフトウェア開発，設計，組込み，仕様書

In-Design Model Checking for Embedded Software

－A Proposal of Model Checking for Specification in Design process－

Koichi SHINOZAKI† Daichi MIZUGUCHI†† and Takeshi ISHII†††

†Power Engineering R&D Center, Kansai Electric Power Company 3-11-20 Nakoji, Amagasaki-shi, Hyogo, 661-0974 Japan
††Laboratory for Verification and Semantics, National Institute of Advanced Industrial Science and Technology 3-11-46 Nakoji, Amagasaki-shi, Hyogo, 661-0974 Japan
†††Distribution Network Equipment Development Department, TOKO SEIKI CO.,LTD 3-14-40 Senrioka, Settsu, Osaka, 566-8686 Japan

E-mail: †K576789@kepco.co.jp, ††daichi.mizuguchi@aist.go.jp, †††TAKESHI_ISHII@toko-s.co.jp

Abstract We propose the In-Design Model Checking which incorporates model checking of specifications into the design process of software development, aiming at the early detection and correction of specification errors. We report on our trial application of the In-Design Model Checking to an actual embedded software development with the examination of its effectiveness, problems and feasibility.

Keyword Model Checking, Software Development, Design, Embedded, Specification

1. はじめに

高度情報化社会の進展に伴い，大小さまざまなシステムソフトウェアが生活に密着して稼働しており，その信頼性確保は必要不可欠な問題である．また，ソフトウェア開発者の立場からは，ソフトウェアのテスト・修正に費やされる莫大な作業時間が，開発コスト削減の障害となっている．

テスト工程でバグが発見された場合には，前工程に戻って原因究明を行い，修正作業を行う．特に発見されたバグが仕様書に起因する場合には，ソフトウェアの設計工程に戻って修正を行う必要がある．これにかかる労力は大きく，また新たなバグを生み出す可能性がある．

よって，仕様書に潜在している論理の矛盾や設計者にとって予想外の動作といったバグの原因を，プログラミングよりも前に検出して修正することができれば，ソフトウェアの信頼性向上と開発作業の効率化に大きな効果が期待できる．

本研究では，仕様書に対してモデル検査を適用することで，システムに起こりうる全ての状態を網羅的に検査し，仕様書の不備の早期検出・修正を行う手法を「イン-デザイン モデル検査」として提案する．

さらに，このイン-デザイン モデル検査を，実際の組込みソフトウェア開発に対して適用する体制を整えて試行し，その効果，問題点，および実現性について検討した結果を報告する．

2. イン-デザイン モデル検査

ソフトウェア開発は，顧客の要求仕様書から始まり，システム仕様書から機能仕様書，プログラム仕様書と

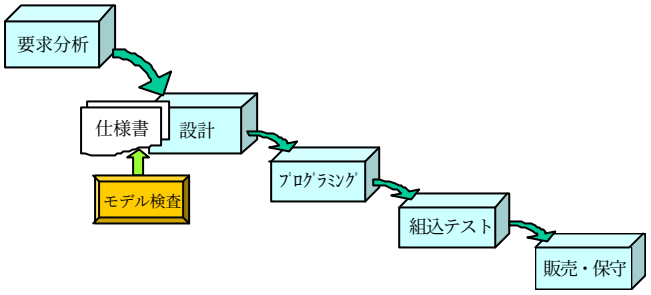

図１　イン-デザイン モデル検査

段階を追って機能が細分化されて行く．しかし，これらの仕様書には，論理の矛盾や作成者にとって予想外の動作といったバグの原因が潜在している．

こうした仕様書の不備を取り除くために，査読が行われるが，査読者の技能と経験に依存するため十分な実施が難しく，手戻りの原因となっている．

そこで，イン-デザイン モデル検査では，仕様を細分化して設計を行う工程の中で，仕様書に対してモデル検査を導入する（図１）．

モデル検査とは，状態遷移系として記述されたシステムが，時相論理式で表された検査項目を満足するかどうかを数理的に判定する手法である[1]．モデル検査では，システムの全動作を網羅的に検査することができるので，査読に比べて厳密な検査が可能である．

今日では，モデル検査を計算機上で自動的に実行するツール（モデル検査器）が公開されている[2]．イン-デザイン モデル検査では,モデル検査器を用いた自動的なモデル検査を，設計段階において仕様書に適用する．これにより，仕様書の不備の早期検出・修正を目指す．

イン-デザイン モデル検査の作業の流れを，モデル検査器ＳＭＶ（Symbolic Model Verifier）[3]を使った場合を例にとって説明する（図２）．まず，仕様書を状態遷移図としてモデル化し(a)，これをモデル化言語により記述してＳＭＶコードを作成する(b)．さらに仕様書もしくは，事前の知識に基づいて検査項目を抽出・設定(c)して，それらをＣＴＬ(Computation Tree Logic)と呼ばれる時相論理式によって記述する(d)．次に，これらを入力としてモデル検査器を実行する．モデル検査器は，入力したモデルが検査項目を満足するかどうかを自動的に計算して結果を回答する(e)．

モデルが検査項目を満足する場合には「true」が，満足しない場合には「false」と反例（検査項目を満足しない状態遷移の例）が出力される．「false」が回答された場合には，反例を解析することによりその原因を究明する(f)．モデル化の誤りや時相論理式の誤りが原因の場合には，それらを修正して再度，モデル検査を行う．そうでない場合には，仕様書そのものの不備が原因と考えられるので，その修正を行う．これらの作業を，「false」が回答されなくなるまで繰り返すことで，仕様書に潜在する不備を手直しする．

3. ソフトウェア開発工程への適用

研究者によるソフトウェア仕様書のモデル検査は，これまでにも発表されている[4][5]．今回は，イン-デザイン モデル検査を実際のソフトウェア開発工程に対して適用する試みとして，ソフトウェア開発者がモデル検査の教育を受けた上で研究者と協力してモデル検査を行なう体制を整えた．

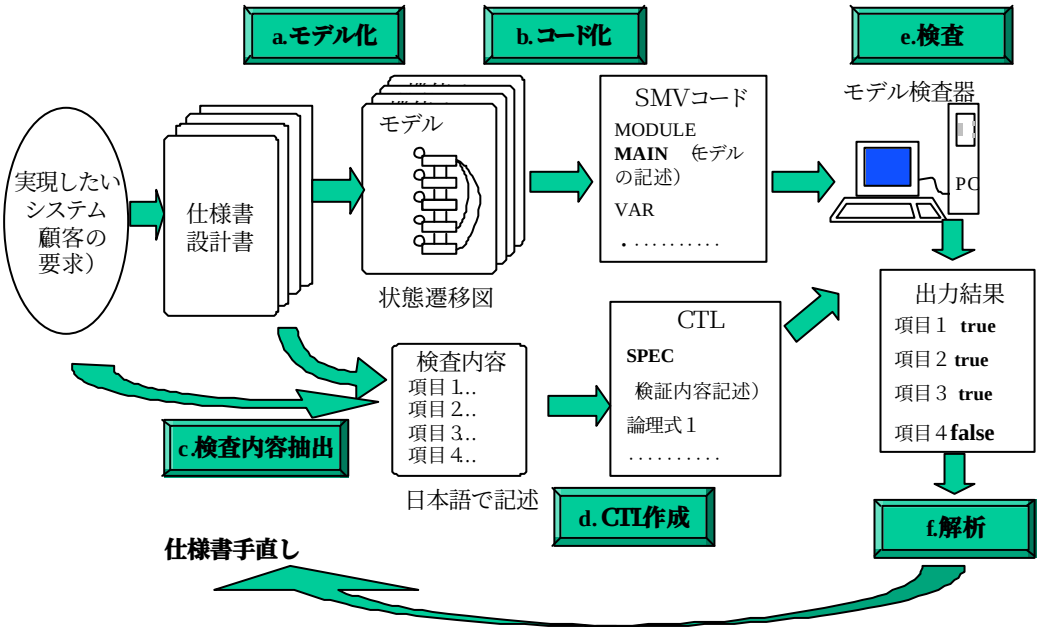

図２　イン-デザイン モデル検査の流れ

表１　モデル検査の教育内容

モデル検査の考え方	8 時間
モデル検査言語（ＳＭＶ）	30 時間
モデル検査例題演習	50 時間

開発者は，モデル検査について全くの初心者からスタートして，表１の教育を受講してから，ある組込みソフトウェアの仕様書のモデル検査を行った．モデル検査の経験がある研究者は，開発者から仕様書の説明を受けてソフトウェアの動作を理解しながら，開発者と協力してモデル検査を実施した．

インデザインモデル検査を試行したソフトウェアの概要を表２に示す．対象とした組込み機器は，新規開発品であり，ソフトウェアおよびハードウェアの併行開発のため，設計時点ではハードウェアが存在していなかった．また，仕様書全部ではなく，一部の機能についてのみ検査を行った．設計後のＣ言語によるプログラミングの結果，全体でおよそ 3500 行のシステムとなったが，そのうちおよそ 600 行に相当する部分を検査対象とした．

今回は，モデル検査とは別に，従来どおりのソフトウェア開発も併行して行っており，モデル検査の結果を適時，従来設計に反映する流れとした．

4. 仕様書のモデル化

一般に，上流の仕様書にはシステムの持つ機能が比較的大きな単位で記述されており，下流側の仕様書では機能が細分化され，プログラム言語に沿った形で処理が記述されている．今回は，開発者のニーズに基づき，プログラムに近いレベルでの詳細な仕様の不備を検出するため，最も下流側のプログラム製作仕様書を検査対象としてモデル化した．

今回使用したプログラム製作仕様書は，フローチャートを主体に記述された仕様書であった．開発者は，検査を行いたい箇所を抽出しモデル化を行った．

また，今回モデル化を行ったソフトウェアの中には，ハードウェアの割り込みにより起動する部分があったため，ハードウェアのモデル化も併せて行った．

さらに，ハードウェアの中には，外部からの入力を必要とするものがあったため，外部入力部分のモデル化も行った．これら３つのモデルを組み合わせて検査

表２　モデル検査対象

ＣＰＵ	8bit
ＯＳ	搭載せず
割り込み要因	8 種類
記述言語	Ｃ言語
モデル検査対象ステップ数	約 600 行 (約 3500 行のうち)

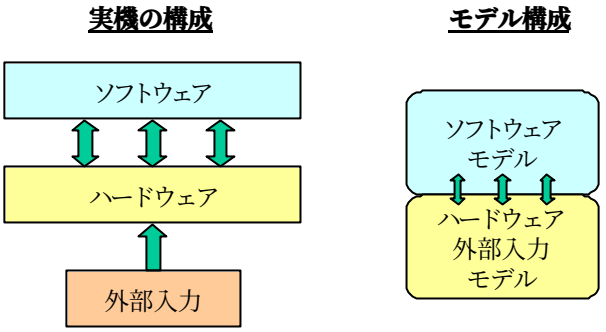

図３　モデル組み合わせの例

を実施した．この様子を図３に示す．

5. モデル検査の実施

今回の試行では，モデル検査器として前述のＳＭＶを用いた．ＳＭＶは状態遷移系の内部表現に効率的なデータ構造を用いており，比較的多くの状態数を扱うことができる．ＳＭＶを実行した計算機環境を表３に示す．

表３　使用した計算機環境

ＣＰＵ	32 bit
動作周波数	3 GHz
メモリー	4 GB

モデルから作成したＳＭＶコードの行数と状態数を表４に示す．Ｃ言語とＳＭＶのコード行数の違いは，文法の違いによるものである．

モデル検査に必要な検査項目は，プログラム製作仕様書に記述されたシステムの動作，上位の仕様書に記載されている動作，明示的に示されていない分野固有の知識等に基づいて，開発者が作成した．また，ハードウェア及び外部入力部分のモデルが期待通りに作成されていることを確かめるための検査項目も別途作成した．その結果，検査項目は，合わせて 99 個となった．こうして作成した検査項目をＣＴＬにより記述し，順次，検査モデル（ＳＭＶコード）に追加してモデル検査器で検査した．

今回，全検査項目について図２に示した a.モデル化から f.解析までの繰り返しに要した時間は，約 160 時間であった．作業時間の内訳を表５に示す．

表５の時間には，モデル化，コード化の誤りやＣＴＬ論理式の誤りを含んだ状態でＳＭＶを実行して結果

表４　ソースコードに対する SMV データ

項　　目		数　量
Ｃ言語のソースコード		約 600 行
ＳＭＶソースコードの行数	ハード部分	約 700 行
	ソフト部分	約 1200 行
ＳＭＶでの状態数		約 3000 万

表５　モデル検査に要した時間

作業内容	項　目	時間
仕様書のモデル化	ハード部分	4
	ソフト部分	2
モデルのコード化	ハード部分	2
	ソフト部分	4
モデル検査（SMVの実行時間）		132
検査結果「false」の反例解析		16

を解析した時間も含まれている．ＳＭＶの実行１回に要した時間は，数秒から数時間の範囲で大きくばらついている．ＳＭＶの実行により「false」が得られた場合の反例解析を，開発者と研究者が協力して実施してすることで，過去の例[5]と比較して効率良く解析作業を進めることができた．

全ての検査により，7 件の仕様書の不備が検出できた．そのうち，システムの動作として実際に支障となるものが3件,支障を発生しないものが4件であった．検出できた不備の要因を表６に示す．

システムをプログラム製作仕様書のレベルでモデル化していることから，ハードウェアに依存したソフトウェアの詳細な動作に関わる不備を検出できている．

6. 実施結果

今回の事例を基に，イン-デザイン モデル検査の費用対効果を検討する．教育に要した88 時間を除き，イン-デザイン モデル検査の実施に 160 時間を要して 7 件の仕様書の不備を検出したことを効果として算定するには，様々な方法が考えられる．今回は，イン-デザイン モデル検査の実施によって,テスト工程後のソフトウェア改修および再テストに要する時間（改修内容から経験で 80 時間と見積り）が発生しなかった，と考えて推定効果の最小値を 80 時間とする．

推定効果 80 時間からイン-デザイン モデル検査の実施時間 160 時間と差し引きすると, 80 時間の損失である．初めて実際の製品開発にモデル検査を導入する今回の事例では，実際に開発作業を効率化するには至っていない．しかし今回，理論研究やツールの進歩にもかかわらず，これまでは実用化に遠いと思われていたモデル検査が，実際の開発現場において教育も含めて現実的な作業時間で実施でき，成果が得られることが確認できた．対象ソフトウェアに高信頼度を要求される場合、新規開発で知見が不足する場合には，十分な効果が期待できるものと考える．

表６　検出できた不備の要因

不備の要因	件数
開発者が想定していない変数に対する不正なアクセス	3
開発者が想定していない割り込みの発生	1
開発者が想定していない外部入力	1
割込み処理のオーバーヘッド	1
仕様書の記入不足	1

表７　イン-デザイン モデル検査の課題

項　目	説　明
1.導入教育の効率化	専門書はあるが，実用面に重点を置いた教材がない．
2.モデル化のノウハウ蓄積	最初からノウハウを知っていれば，もっと効率的に実施できた．
3.検査手法の操作性向上	モデル検査器（ＳＭＶ）の操作性が悪い．

今回の取り組み結果を踏まえ, さらにイン-デザイン モデル検査の作業時間を短縮するための課題について, 作業を行った開発者と研究者の意見を元に取りまとめた結果を表７に示す．

7. 今後の取り組み

イン-デザイン モデル検査をソフトウェア開発手法として確立するために,以下の取り組みを進めていく．

- 開発者向けの教材開発を行い，効率よく習得するための機会を設ける．
- 様々なソフトウェア仕様書を対象にイン-デザイン モデル検査を行ない，モデル化のためのノウハウを得ると共に,これを再利用できる形で蓄積する．
- モデル検査器の操作性を改善する検討を行う．
- 検査を行う仕様書のレベル（上流／下流）について適切な指標を検討する．

なお，教育に関しては，既に産業技術総合研究所のシステム検証研究ラボを中心として一般企業向けセミナーの準備を進めている．

文　献

[1] E. M. Clarke, O. Grumberg, and D. A. Peled: Model Checking, The MIT Press, 1999.

[2] B. Berard, M. Bidoit, A. Finkel, F. Laroussinie, A. Petit, L. Petrucci, Ph. Schnoebelen, and P. Mckenzie: Systems and Software Verification; Model-Checking Techneques and Tools, Springer, 2001.

[3] A. Cimatti, E. M. Clarke, E. Giunchiglia, F. Giunchiglia, M. Pistore, M. Roveri, R. Sebastiani and A. Tacchella, “NuSMV 2: An OpenSource Tool for Symbolic Model Checking,”Proceeding of International Conference on Computer-Aided Verification (CAV 2002), pp. 27-31, Copenhagen, Denmark, July, 2002.

[4] 寺田博文，土屋達弘，菊野享,“記号モデル検査を用いたステートチャートの検証”，信学技報，SS98-41，pp17-24，Jan.1999.

[5] 早水公二，篠崎孝一，高橋孝一，渡邊宏,“モデル検査器を用いた自動検針システムの仕様検証”，産業技術総合研究所算譜科学グループ研究速報，AIST-PS-2003-005, June, 2003.