設計モデル検証(応用編)

平成２４年度シラバス

２０１２年1月13日

国立情報学研究所

トップエスイープロジェクト

代表者　本位田　真一

1. 講座名

設計モデル検証(応用編)

2. 担当者

田原　康之、吉岡　信和

3. 本講座の目的

本講座では、ネットワーク家電の制御ソフトウェアを題材とした産業ソフトウェアの設計モデル検証問題を扱う。近年ますます大規模化・複雑化するソフトウェアの設計において、その動作の正しさを保証することがより困難になっているが、設計モデルの正しさを自動的に検証するモデル検査ツールを使用し、実際のシステム開発に適用する方法を習得する。しかし、現在多くのモデル検査ツールがあり、またモデル検査技術は進歩が早く、より高機能・高性能な新しいツールが現れてきている。したがって、それらのツールから適切なものを取捨選択し、また新しいツールへの移行を円滑に行うのは容易ではない。そこで本講座では、代表的な3つのモデル検査ツールであるSPIN、SMV、およびLTSAに対し、まず標準のモデル記述言語UMLを全面的に採用し、実際に３つのツールの利用を通して、ソフトウェア設計検証の難しさを体感した上で、設計モデルの特徴に合わせたモデル検査ツールの使い分け、およびノウハウを習得する。また、グループ討議を通じた例題演習により、議論を通してモデル検査技術の理解を深め、検証プロセスの実用的ノウハウを体得できる効果が期待できる。

4. 本講座のオリジナリティ

SPIN などのモデル検査ツールを使用した設計モデル検証については、従来多くの同様な講座が開講されている。しかしそれらの講座にはさまざまな問題点があったため、受講生が習得した内容を速やかに開発現場で適用するための障害となっていた。また、SPIN に絞った内容については、設計モデル検証(基礎編)講座において扱っている。しかし、本講座の目的である、さまざまなツールの使い分け、および新たなツールへの対応まではカバーしていない。本講座では、それらの問題点を解消し、本講座受講後各種モデル検査ツールを、速やかに開発現場で適用できるように配慮している。

表 1 に、既存の講座の問題点と、本講座における解を示す。

表 1 既存の講座の問題点と、本講座における解

既存の講座の問題点	本講座における解
複数のモデル検査ツールを扱っていても、それらの使い分けを、実習を通じて習得することがないので、受講生が各自の現場において、どのツールを使えばいいかの判断が難しい	実習では、3つのツールのいずれに対しても、共通の検証プロセスに従って、同じ例題に対して適用するため、各ツールの特徴を容易に実感可能
ツール毎に個別に講義・実習を行うことになるため、新たなツールへの対応力を習得するのが難しい	検証プロセスは、UML を共通の設計モデル記述言語として使用し、設計モデルの特徴にあわせてモデルを作成し、モデル検査ツールに依存しない共通の作業と、ツールに特化した作業とを明確に区別しているため、検証プロセスを習得することにより、新たなツールにも容易に対応可能

5. 本講座で扱う難しさ

近年、家電の制御ソフトウェアは大規模化、複雑化が進み、複数の家電機器がネットワークに接続され、相互に連携して動作することで 1 つの機能を実現する、ネットワーク家電市場が急速に立ち上がりつつある。ネットワーク家電では、従来の家電機器に比べて、他の機器との連携動作のためのプロトコルなどが必要になるため、制御ソフトウェアが非常に複雑になる。さらに、ホームネットワークでは、動的に変更される接続相手の識別や、不安定なネットワーク環境などを考慮する必要がある。このような環境に対して高信頼でかつ安全なソフトウェア設計を実施するために、全動作パターンを人手で検証する従来型の方法論は膨大な工数を必要とし、もはや現実的手段ではない。

たとえば、3 台のネットワーク家電機器、TV、DVD レコーダ、HD レコーダ一体型 TV から構成されたホームネットワークがあり、2 人のユーザがホームネットワーク上の異なる機器に対して操作を行うような場合、複数の資源獲得要求とそれに伴う処理を適切な順序で誤りなく実行するように設計されなければならないが、そのような正しい設計を行うのは容易ではない。特に、ネットワークで接続された分散システムの場合、ネットワークの状況により複数の機器から到着する要求の順序が変わるなどの状況が考えられたり、要求が消失したりするなどの様々な例外を考える必要があり、どのような環境で実行されるのかを考えつつ個々の機器の振る舞いを検証するのは容易ではない。さらに、複数の機能を組み合わせて一つの機器を構成する場合、高速化のため機能間はバスでつなぎ、クロック同期するシステムとして組み上げる。クロック同期するシステムの場合、設計時のモデル化方法が同期しない場合とまったく異なる。そのようにシステムの振る舞いの前提条件を考えながら設計モデルの正しさを検証するのも容易ではない。

また、設計モデル検証(基礎編)講座において、モデル検査ツール SPIN を用いた、自動検証技術を習得している。しかし、現在多くのモデル検査ツールがあり、各ツール毎に検証のためのモデル記述方や検証方法が異なり、モデル検査技術の適用においては、各時点でシステムの前提条件や検証項目も考慮し、適切なツールを取捨・選択しなければならない。

6. 本講座で習得する技術

モデル検査は、システム上で起こり得る状態を網羅的に調べることによって設計誤りを発見する自動検証手法の一種である。近年、高品質な検証ツールが開発され利用されて、モデル検査を実用システムに適用した多数の成功事例が報告されている。しかし、ソフトウェアの設計検証には、ソフトウェア開発プロセスと同様に、様々な角度から検証問題を捉えるという実践的なスキルと経験が求められる。

本講座では、設計検証に関わる活動をソフトウェアプロセスの 1 つと捉えた、検証プロセスについて学ぶ。検証プロセスは、以下に示す、モデル検査ツールに依存しない 2 ステップと、ツールに依存する 3 ステップの計 5 つのステップから構成され（図 1）、各ステップで必要な具体的作業の順序と内容とを体系的に習得する。

(1) 検証目的を明確化する検証要求分析ステップ
(2) 同期・非同期などの制約や外部環境をモデル化する検証モデル設計ステップ
(3) モデル検査ツールの入力言語による検証モデルの実装ステップ
(4) モデル検査ツールによる検証ステップ
(5) 検証エラーを分析して、設計誤りの原因を究明する設計誤り発見ステップ

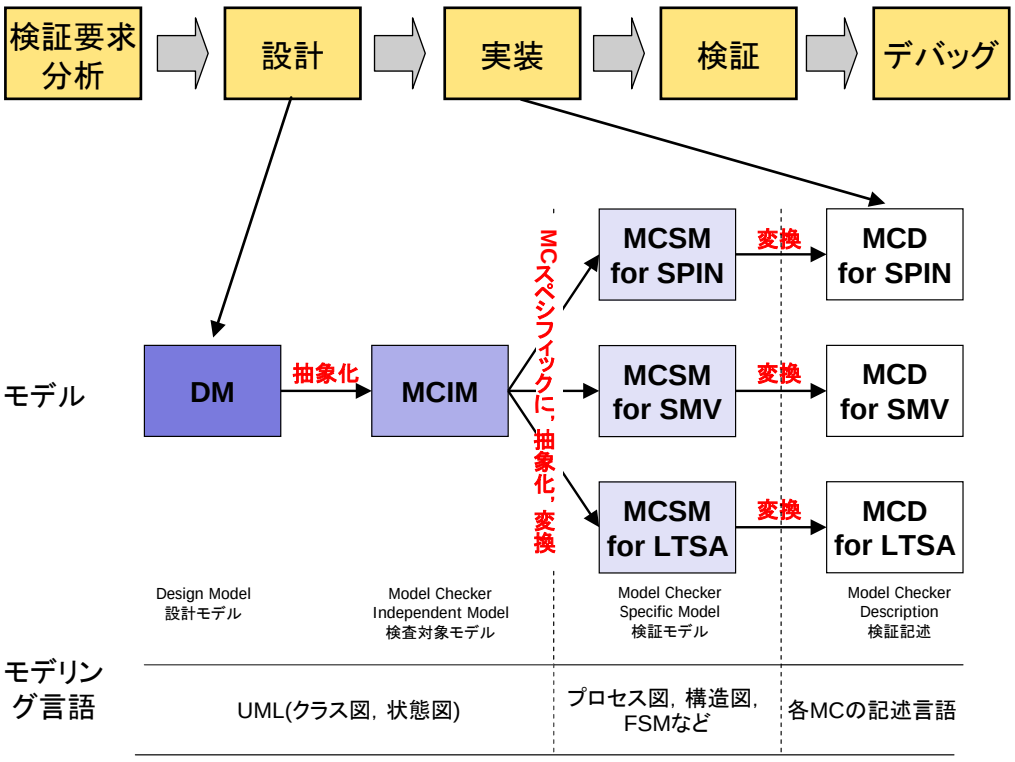

図 1　モデル検査を利用した設計検証プロセス

本講座では、具体的なモデル検査ツールとして、SPIN、SMV、および LTSA を使用する。
SPIN の特徴は次の通りである。

- C や Java などのプログラミング言語に近い言語 Promela (Protocol/Process Meta Language)により、検証対象となるモデルの記述が可能
- 検証機能が豊富(シミュレーション、デッドロック検証、線形時相論理(Linear Temporal Logic, LTL)式検証など)
- iSpin ツールによる GUI サポート(メニューとボタンによる操作、反例トレースからのシーケンス図生成など)
- 実適用事例が豊富(火星探査機などの宇宙システム、交換機システム、治水システムなど)
- オープンソースライセンス

SMV の特徴は次の通りである。

- 二分決定グラフ(Binary Decision Diagram, BDD)と呼ばれる、状態空間の効率的なデータ表現を扱うため、高性能な検証が可能
- GUI により、実行とレースの状態遷移の追跡が容易
- 実適用事例が豊富(プラント制御、IEEE 1394 プロトコルなど)

LTSA の特徴は次の通りである。

- GUI の機能が豊富(視覚的な検証モデル表示、シミュレーションなど)
- 有限状態プロセス(Finite State Process, FSP)と呼ばれる、簡明な言語によるモデル記述
- 検証項目記述が容易(デッドロック、進行性など)
- プラグインにより、拡張機能が豊富(メッセージシーケンス図によるモデル記述、BPEL4WS プラグインなど)

7. 前提知識

本講座の受講生は、以下の項目、および、「設計モデル検証(基礎編)」講座を習得済みであることが望ましい。

- 以下に関する基礎理論
  - 並行プログラム：非決定性、資源共有・排他制御、通信、安全性、生存性、公平性
  - オートマトン：意味論(形式言語理論)、通信
  - 時相論理：構文、意味論、パターン(安全性、生存性など)
- モデル検査技術の原理
  - 時相論理式のオートマトンへの変換
  - 受理言語探索
  - BDD (Binary Decision Tree)
  - On-the-fly 手法
  - 部分順序による状態数削減手法
- UML：特にステートチャートの意味論
- プログラム抽象解釈手法
- SPIN に関する基礎知識
  - iSpin 操作方法
  - Promela 言語

なお、これらの項目は、全て「基礎理論」講座、および「設計モデル検証(基礎編)」講座で習得可能である。

8. 講義計画

・ 概要

第 1 回：導入、基礎編復習、分散システムの難しさと検証
第 2 回：SPIN における分散システムの難しさの検証(1) – 同期・非同期モデルと検証
第 3 回：SPIN における分散システムの難しさの検証(2) – 環境モデルと検証
第 4 回：SMV 概論 – モデル記述、検証、反例分析、シミュレーション、設計モデル修正
第 5 回：SMV における分散システムの難しさの検証(1) – 同期・非同期モデルと検証
第 6 回：SMV における分散システムの難しさの検証(2) – 環境モデルと検証
第 7 回：LTSA 概論 – モデル記述、検証、反例分析、シミュレーション、設計モデル修正
第 8 回：LTSA における分散システムの難しさの検証(1) – 同期・非同期モデルと検証
第 9 回：LTSA における分散システムの難しさの検証(2) – 環境モデルと検証
第 10 回：検証のためのノウハウ詳論
第 11 回：演習(1) – 応用課題：設計モデルの特徴とツールの使い分け
第 12 回：演習(2) – 応用課題：設計モデルの特徴とツールの使い分け
第 13 回：演習(3) – 応用課題：設計モデルの特徴とツールの使い分け
第 14 回：演習(4) – 応用課題：設計モデルの特徴とツールの使い分け
第 15 回：議論とまとめ

・ 詳細

第 1 回：導入、基礎編復習、分散システムの難しさと検証

導入(座学)

- 基礎編で何を学んだか?
  - ✧ 分散システムの難しさ
  - ✧ 検証プロセスの概要
- 応用編では何を学ぶか?:
  分散システムの難しさと検証ツールの使い分け。
  - ✧ システムの制約：非同期・同期モデル
  - ✧ 環境のモデリング
- システム制約・条件を考慮した設計モデルの構築(個人実習)

グループ演習：

- 設計モデルの相互レビュー
- シナリオベースのシステムの検証と設計モデルの修正

宿題：検証する問題の設定：システムの制約・条件と検証項目を設定する

第 2 回：SPIN における分散システムの難しさの検証(1) – 同期・非同期モデルと検証

- SPIN による検証プロセス
- 同期・非同期モデルの検証モデルの作成と検証

- ✧ プロセスと通信チャネルの利用法
- ➢ 演習：トイ例題を用いた検証（個人実習とグループディスカッション）
- ➢ レポート：ＳＰＩＮにおける同期・非同期の検証ノウハウ

第 3 回：SPIN における分散システムの難しさの検証(2) - 環境モデルと検証

- ➢ 環境モデルの作成方法と検証
  - ✧ 検証式の記述の仕方
  - ✧ 検証のやり方
- ➢ 演習：トイ例題を用いた検証（個人実習とグループディスカッション）
- ➢ レポート：SPIN における環境モデルを考慮した検証ノウハウ

第 4 回：SMV 概論 - モデル記述、検証、反例分析、シミュレーション、設計モデル修正

- ➢ UML 設計モデルから検証モデル記述(個人実習)
  - ✧ 抽象化、検証モデル記述ノウハウを習得
- ➢ 検証モデルを検証、反例分析(個人実習)
- ➢ 検証モデルをシミュレーション(個人実習)
- ➢ CTL を使った検証式の作り方と検証
- ➢ 設計モデルを修正(個人実習)
- ➢ 演習：トイ例題を用いた検証、反例分析、設計モデル修正(バグがなくなるまで)(個人実習、宿題)

第 5 回：SMV における分散システムの難しさの検証(1) - 同期・非同期モデルと検証

- ➢ 同期・非同期モデルの検証モデルの作成と検証
  - ✧ 変数を使った同期・非同期の実現
- ➢ 演習：トイ例題を用いた検証（個人実習とグループディスカッション）
- ➢ レポート：SMV における同期・非同期の検証ノウハウと SPIN との違い

第 6 回：SMV における分散システムの難しさの検証(2) - 環境モデルと検証

- ➢ 環境モデルの作成方法と検証
  - ✧ 検証式の記述の仕方
  - ✧ 検証のやり方
- ➢ 演習：トイ例題を用いた検証（個人実習とグループディスカッション）
- ➢ レポート：SPIN における環境モデルを考慮した検証ノウハウと SPIN との違い

第 7 回：LTSA 概論 - モデル記述、検証、反例分析、シミュレーション、設計モデル修正

- ➢ UML 設計モデルから検証モデル記述(個人実習)
  - ✧ 抽象化、検証モデル記述ノウハウを習得
- ➢ 検証モデルを検証、反例分析(個人実習)
- ➢ 検証モデルをシミュレーション(個人実習)
- ➢ safety 機能、モニタプロセスを使った検証
- ➢ 設計モデルを修正(個人実習)

- 演習：トイ例題を用いた検証、反例分析、設計モデル修正(バグがなくなるまで)(個人実習、宿題)

第 8 回：LTSA における分散システムの難しさの検証(1) – 同期・非同期モデルと検証

- LTSA による検証プロセス
- 同期・非同期モデルの検証モデルの作成と検証
  - ✧ プロセスと通信チャネルの利用法
- 演習：トイ例題を用いた検証（個人実習とグループディスカッション）
- レポート：LTSA における同期・非同期の検証ノウハウと SPIN との違い

第 9 回：LTSA における分散システムの難しさの検証(2) – 環境モデルと検証

- 環境モデルの作成方法と検証
  - ✧ 検証式の記述の仕方
  - ✧ 検証のやり方
- 演習：トイ例題を用いた検証（個人実習とグループディスカッション）
- レポート：LTSA における環境モデルを考慮した検証ノウハウと SPIN との違い

第 10 回：検証のためのノウハウ詳論

- 検証モデルを作成するためのノウハウ
  - ✧ Partial Reduction、変数初期化
- 検証式を作成するためのノウハウ
  - ✧ 検証パターン
- 演習：トイ例題を用いた検証、反例分析、設計モデル修正(バグがなくなるまで)(グループ実習、宿題)
- レポート：新たな検証ノウハウ、ツールに依存・非依存のノウハウ

第 11 回：演習(1) – 応用課題：設計モデルの特徴とツールの使い分け

- グループ毎に設定した問題に対して設計およびその検証を行う
- グループで複数の場合を想定し、それぞれで２つ以上ツールを使って検証
- 演習：応用例題を用いた検証モデル記述、シミュレーション(グループ実習、宿題)

第 12 回：演習(2)

- グループ毎に設定した問題に対して設計およびその検証を行う
- グループで複数の場合を想定し、それぞれで２つ以上ツールを使って検証
- ツールをどのように使い分ければよいかを議論
- 各自が考えたノウハウを適用
- ツールの使い分けと新しいノウハウについて議論
- 宿題：SPIN、SMV、および LTSA の比較

第 13 回：演習(3)

- グループ毎に設定した問題に対して設計およびその検証を行う

- グループで複数の場合を想定し、それぞれで２つ以上ツールを使って検証
- ツールをどのように使い分ければよいかを議論
- 各自が考えたノウハウを適用
- ツールの使い分けと新しいノウハウについて議論
- 宿題：SPIN、SMV、および LTSA の比較

第 14 回：演習(4)

- グループ毎に設定した問題に対して設計およびその検証を行う
- グループで複数の場合を想定し、それぞれで２つ以上ツールを使って検証
- ツールをどのように使い分ければよいかを議論
- 各自が考えたノウハウを適用
- ツールの使い分けと新しいノウハウについて議論
- 宿題：SPIN、SMV、および LTSA の比較

第 15 回：議論とまとめ

- グループ発表：演習結果、およびグループ討議の内容
  - 特に 3 ツールの比較と使い分け
- 最後に講座全体のまとめ(座学)
  - 習得内容の総括
  - 関連技術
  - 次に履修が期待される講座
  - 宿題：講座で習得できたこと、3 ツールの比較、および検証プロセスとノウハウの有用性の評価

9. 教育効果

本講座を受講することにより、実際のシステム開発、特に組込みソフトウェアの開発に適用可能な、設計モデル検証プロセスを習得できる。その結果、開発現場において、モデル検査ツールを活用することにより、信頼性の高いシステムを、効率的に開発することができるようになる。しかも、習得した 3 つのモデル検査ツールから、システムの特徴や検証内容により適切なものを選択したり、また新たなツールを自分で習得して適用したりすることが可能になる。

10. 使用ツール

SPIN：モデル検査ツール

- 使用する上での難しさ
  - モデル記述言語 Promela、および LTL 検証式による記述が難しい
  - 検証の結果がエラーの場合、反例を設計モデル修正に反映するのが難しい
- 使用上必要なノウハウ
  - 検証モデル記述ノウハウ
    - 状態の表現
    - 遷移の表現
    - 通信の表現
  - 検証式記述ノウハウ
    - 検証項目の種類(安全性、生存性など)
    - 検証式パターン
  - 検証モデル抽象化
    - 内部動作の除去
    - データ領域の有限化
  - 検証効率化
    - 部分順序法
    - 状態削減
  - 反例分析
    - 変数値追跡
    - メッセージ追跡
- 選択理由、実用性：5 章に記載の SPIN の特徴を参照

SMV：モデル検査ツール

- 使用する上での難しさ
  - 検証モデル記述、および CTL 検証式による記述が難しい
  - 検証の結果がエラーの場合、反例を設計モデル修正に反映するのが難しい
- 使用上必要なノウハウ
  - 検証モデル記述ノウハウ
    - 状態の表現
    - 遷移の表現
    - 通信の表現
    - 非同期プロセス実行の表現
  - 検証式記述ノウハウ
    - 検証項目の種類(安全性、生存性など)
    - 検証式パターン

- 検証モデル抽象化
  - 内部動作の除去
  - データ領域の有限化
- 検証効率化
  - 部分順序法
  - 状態削減
- 反例分析
  - 変数値追跡

・ 選択理由、実用性：5 章に記載の SMV の特徴を参照

LTSA：モデル検査ツール

・ 使用する上での難しさ
- モデル記述言語 FSP、および検証項目の記述が難しい
- 検証の結果がエラーの場合、反例を設計モデル修正に反映するのが難しい

・ 使用上必要なノウハウ
- 検証モデル記述ノウハウ
  - 状態の表現
  - 遷移の表現
  - 通信の表現
- 検証項目記述ノウハウ
  - 論理式を用いずに記述
  - 検証項目の種類(安全性、生存性など)
- 検証モデル抽象化
  - 内部動作の除去
  - データ領域の有限化
- 検証効率化
  - 部分順序法
  - 状態削減
- 反例分析
  - 変数値追跡
  - メッセージ追跡

・ 選択理由、実用性：5 章に記載の LTSA の特徴を参照

11. 実験及び演習

2～3 名程度の少人数で構成されたグループ単位で演習課題の演習と議論を行い、設計検証の難しさを実際に体感しながら、設計検証プロセス、およびツールの使い分けを体得する。モデル検査ツールを利用してソフトウェア設計の検証を行い、設計誤りを検出できることを確認する。グループ内で複数のモデル検査ツールを適用した結果について比較評価を行い、検証プロセスの有用性と適用性、および 3 つのモデル検査ツールの相違を議論する。議論を通してモデル検査技術の理解を深める。

12. 評価

演習課題レポート、プレゼン発表、出席日数を総合して評価する。

13. 教科書/参考書

- B. Berard et al, “Systems and Software Verification: Model-Checking Techniques and Tools,” Springer Verlag, 2001.
  モデル検査手法を利用したソフトウェア検証について、入門から実践までの一通りが述べられており、この講義に最適である。

- E. M. Clarke et al, “Model Checking,”, MIT Press, 2000.
  モデル検査手法の原理を理論的に述べたものであり、上記教科書を補うのに最適である。

- G. J. Holzmann, “The SPIN Model Checker: Primer and Reference Manual,” Pearson Educational, 2003.
- K. L. McMillan, “Symbolic Model Checking,” Kluwer Academic Publishers, 1993.
- J. Magee and J. Kramer, “Concurrency: State Models & Java Programs,” John Wiley & Sons, 1999.
  それぞれ、各モデル検査ツール SPIN、SMV、LTSA に関する理論的背景とツールの利用法が述べられており、モデル検査ツールを用いた演習に最適である。