モデル検査の実務導入２つのアプローチ
ー適用事例14件の成功と失敗からー

JaSST ’06　2006年 1月30日

メルコ･パワー･システムズ株式会社　技術統括部
早水　公二　　星野　光勇

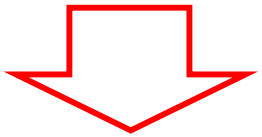
関西電力株式会社　電力技術研究所
篠崎　孝一

モデル検査の概要

＜モデル検査の原理＞

（現実のシステムから作った）仮想モデルが
（要求仕様である）検査項目を満たさない

→モデルに不具合がある



現実のシステムも
要求仕様を満たさないはず

→ソースコードにも不具合がある

モデル検査の概要

＜モデル検査のメリット＞

モデル検査を計算機上で自動的に行うモデル検査器がある

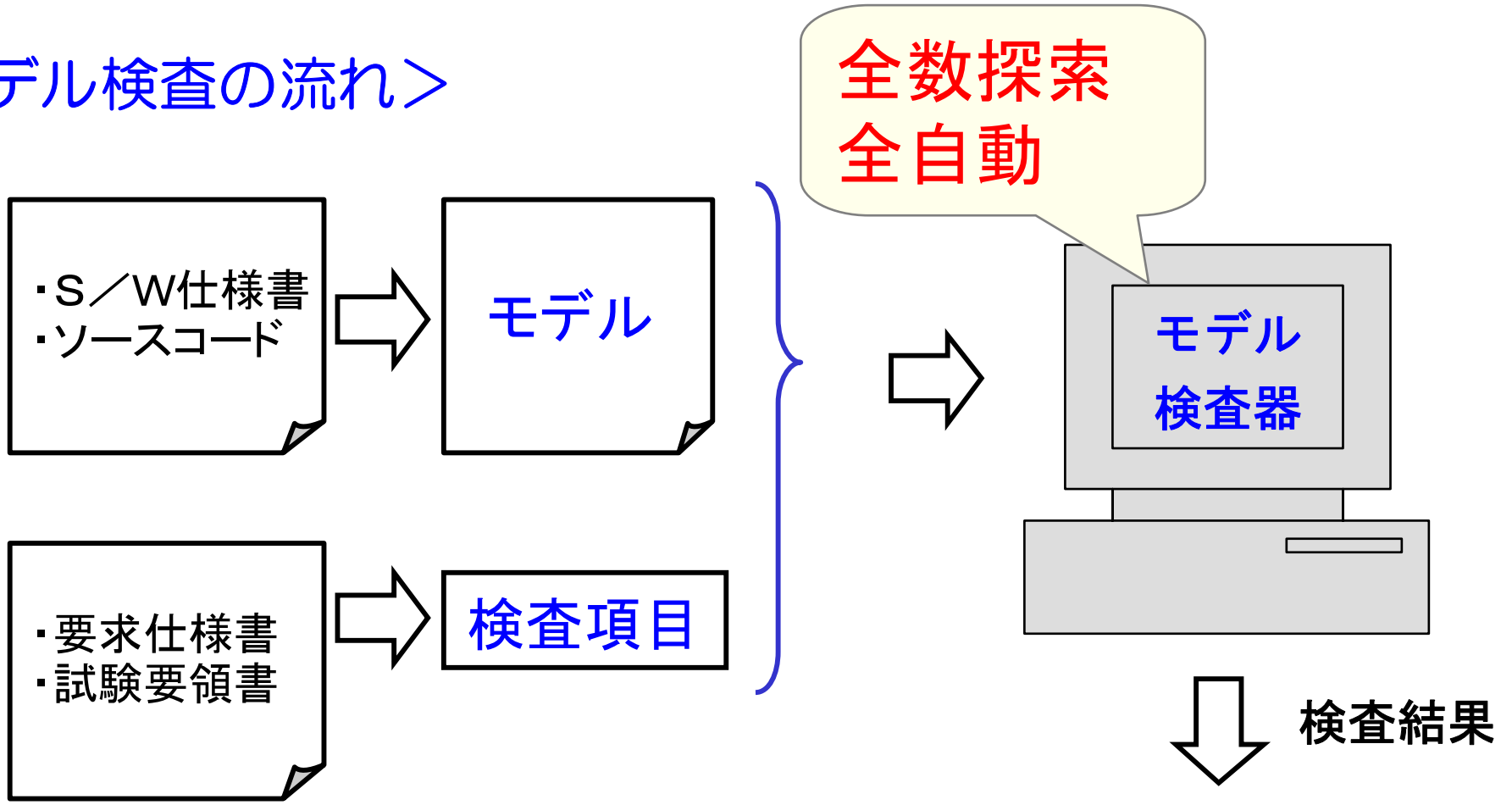
モデル検査器は

モデル（システム）が検査項目（要求仕様）を
満足するか否かを全数探索法を用いて、全自動で、
検査（試験）する。

要求仕様を満たす場合は「True」を出力する。
満たさない場合は「False」と、その証拠（反例）を出力する。

モデル検査の概要

＜モデル検査の流れ＞



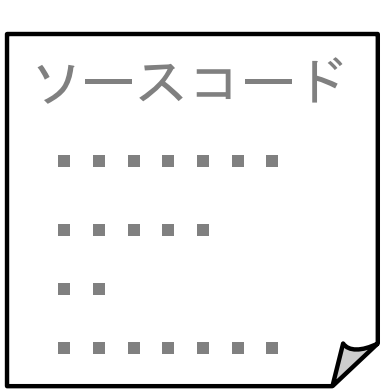
モデルが検査項目を満たす場合「True」
満たさない場合「False」＋「反例」

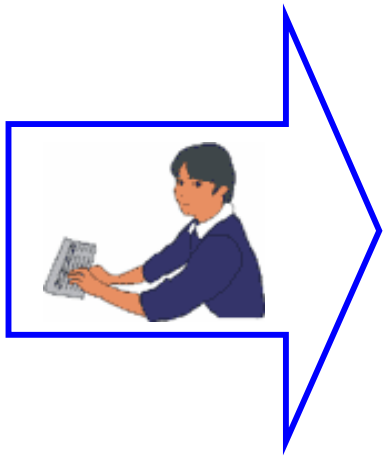
モデル検査の概要

＜モデルとは＞

ソースコードを専用言語で書き換えたもの（模擬システム）

「モデル化」





```
MODULE main  ～モデル検査の専用言語～

DEFINE
 Start_Task1 := Select_Task1 = 1;
 Start_Task2 := Select_Task1 = 0 & Se

VAR
 Select_Task1 : boolean;
 Select_Task2 : boolean;
 Task3 : task_L(Start_Task3,
Semaphore);

ASSIGN
 init(Select_Task1) := 0;
 next(Select_Task1) := case
  Select_Task1 = 0 : {0, 1};
  Select_Task1 = 1 & Task1.PC = 5 : 0;
                    esac;
```

これがモデル

モデル検査の概要

＜検査項目とは＞

要求仕様を論理式で書き換えたもの

例）

「トリガが発生したら、必ずファイル出力を行う」

→ AG (Trigger = ON -> AF (FileOut = ON))

「フラグAとフラグBが同時にONとなることはない」

→ ！EF (Flag_A = ON & Flag_B = ON)

「事象Qと事象Rの間で、事象Pが発生すれば必ず事象Sが発生する」

→ AG((Q & !R -> !E[!R U (!R & !((P -> A[!R U (S & !R)]) | AG(!R)))]))

モデル検査の概要

＜全数探索法とは？＞

条件文（if～else文）が５箇所あったら？
　２の５乗 ＝ ３２通りのパスが存在する

３０箇所あったら？
　２の３０乗 ＝ 約１１億通りのパスが存在する
　（１つのパスを１秒で試験すると、不眠不休で３４年間）

割込み処理があったら？　→　いつ割込みが発生するか．．．
ユーザによる入力があれば　→　ユーザはどんな値を入力するかわからない．．．

モデル検査器は全てのパスを検査（全数探索）する

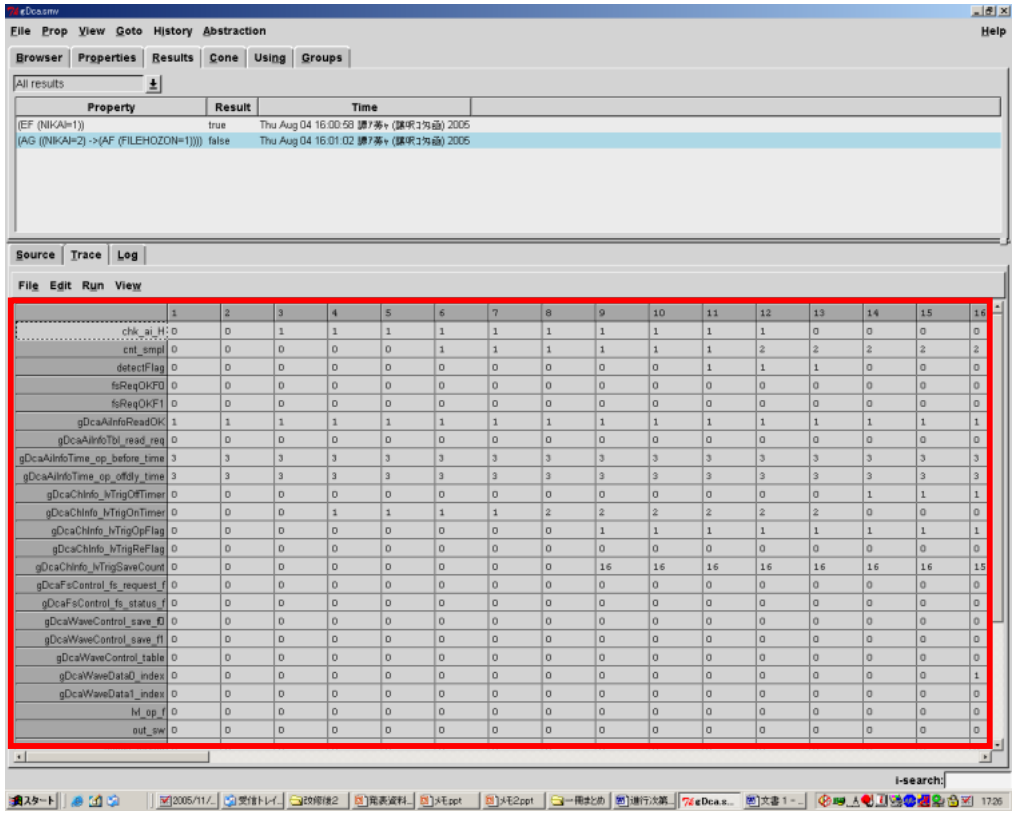
単純な総当たりではなく、
数学・論理学の規則や法則を利用して短時間で検査可能

モデル検査の概要

＜反例とは？＞

「どんなパスを通ると要求仕様を満たさないか」の証拠

形は変数の値の変化列



モデル検査の適用実績

１年６ヶ月間で自社開発した１４件のシステムにモデル検査を適用

システム名	適用結果
勤怠管理	入力操作のタイミングに起因する不具合の原因を究明
広域情報監視	非同期プロセスの割込み処理に起因する不具合の原因を究明
電力設備保全(Ver.1)	仕様の不備1件と不具合3件を発見
輸送運行情報監視	条件分岐処理の不具合2件を発見
画像認識（事象検出）	通信異常時の初回処理だけに発生する不具合の原因を究明
広域システム共通Lib	運用で用いるファイル定義書の不備1件を発見
電力設備監視	波形解析処理のアルゴリズムに起因する不具合の原因を究明
電力設備保全(Ver.2)	不具合の原因を究明し，さらに別の不具合2件を発見
＜その他＞基幹系システム5件　組み込み系システム１件	

モデル検査の実用化に向けた課題

1．状態数爆発により汎用計算機の処理能力を越えてしまう．．．

モデルが取り得る状態数が大き過ぎると、

・検査のための処理時間が長くなる

・メモリを大量に消費する（数分で数GB）

モデル検査器がフリーズし最後まで検査できない

→技術的対策（抽象化技術、H／Wの強化、モデル検査器の改良）

→運用上の対策

2．モデルの作成（ソースコードの書き換え）が大変．．．

→入力ツールあるいは変換ツールの開発

適用事例の分析

モデル検査を適用した１４件の事例を
適用時の形態に着目して分類

規則性を発見 ⇨ 定型的な適用手法

適用時の形態 = 設計者からの依頼形態

◆既に発生してしまった原因不明の不具合をモデル検査で解析してほしい（不具合解析）

◆不具合はないが、モデル検査の網羅的なデバッグで品質を向上させたい（品質向上）

適用事例の分析

分類結果は（代表的な好事例８件）

システム名	適用形態		適用結果
	不具合解析	品質向上	
勤怠管理	○		不具合の 原因究明
広域情報監視	○		不具合の 原因究明
電力設備保全(Ver.1)		○	不具合発見
輸送運行情報監視		○	不具合発見
画像認識（事象検出）	○		不具合の 原因究明
広域システム共通Lib		○	不具合発見
電力設備監視	○		不具合の 原因究明
電力設備保全(Ver.2)	○	○	原因究明 と 不具合発見

適用事例の分析

さらに形態毎に特徴を抽出すると

適用形態	システム名	適用時期	検査対象の範囲
品質向上	電力設備保全(Ver.1)	設計時から運用開始前	極力広い範囲
	電力設備保全(Ver.1)		
	輸送運行情報監視		
	広域システム共通Lib		
不具合解析	勤怠管理	本格運用前	特定の狭い範囲
	広域情報監視		
	画像認識（事象検出）		
	電力設備監視		

モデル検査の適用手法の提案

適用の際の２つのアプローチを提案

◆品質向上のアプローチ

開発初期から運用開始までの全般にわたって、広い範囲を検査し品質向上を目指す

→　従来手法と並行してモデル検査を実施

◆不具合解析のアプローチ

特定の狭い範囲に的を絞って、本格運用前など緊急時に発生した原因不明の不具合を早期に解析する

→　モデル検査に重点を置いて実施

適用時期・検査対象によって
モデル検査と従来手法を使い分ける

モデル検査の効率的な導入形態

設計段階

仕様書やフローチャートがあればモデル検査可能

⇩

モデル検査を単独で使う

Ｓ／Ｗ製作中

複雑でない不具合は従来手法でも発見できる

⇩

微妙なタイミング・複雑な条件で発生する不具合はモデル検査で発見できる

Ｓ／Ｗ試験中

システムの中枢部分だけモデル検査を行う

⇩

モデル検査と従来手法を併用する

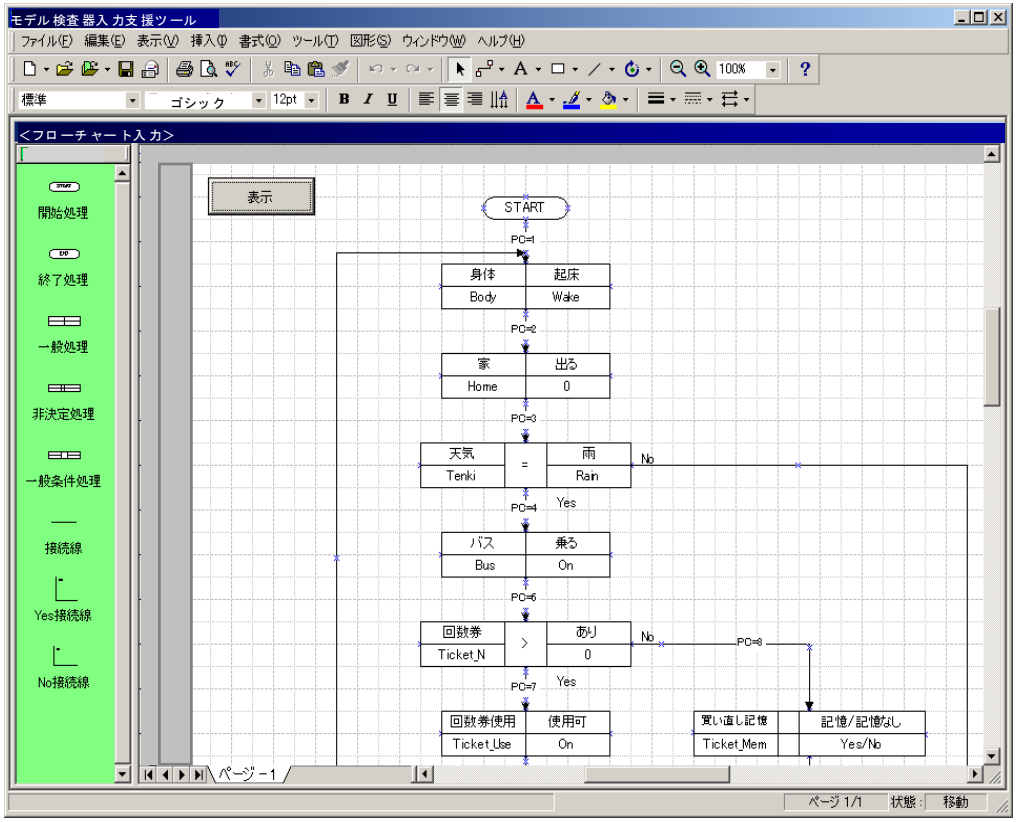
本格運用前

原因不明の不具合発生！　原因がわからない！
調査する時間がない！早く解決したい！

モデル検査を単独で使う
（デバッグへの適用 → 次の発表へ）

モデル検査器支援ツールの開発

フローチャートを入力



モデルを自動生成

```
MODULE main

  VAR
    Body : {Wake, Sleep};
    Home : boolean;
    Bus  : {ON, OFF};

  ASSIGN

    init(Body) := Sleep;
    init(Home) := 0;

    next(Body) := case
      PC = 1 : Wake;
      PC = 8 & time = 23 : Sleep;
      1 : Body;
    esac;
```

今後の取り組み

◆事例の積み重ね／ツールの開発を継続

◆事例の分析

→ モデル検査の適用基準策定

◆事例・ツールの紹介

→ モデル検査のＰＲ・導入指針