

KEPCO MPS

モデル検査のデバッグへの適用と モデル化作業支援

JaSST '06 in OSAKA 2006年5月11日

関西電力株式会社 電力技術研究所
篠崎孝一 太田 弘

メルコ・パワー・システムズ株式会社 技術統括部
星野 光勇 早水 公二

KEPCO MPS

はじめに

ソフトウェアシステムへの信頼性、安全性の要求

IEC61508 (JIS C 0508)
機能安全規格
SIL=4にて要求

情報システムの信頼性向上
に関するガイドライン(案)
2006.4.4 経済産業省

↓

数理論理学を使ってソフトウェアを厳密に検証
形式手法 (形式的仕様記述・形式検証)

モデル検査
(Modelchecking)

JaSST '06 in 大阪 2

KEPCO MPS

システムのモデル化

システムの動き → 状態集合と遷移関係 → 論理関数

4つの状態 A, B, C, D を2つの2進符号 p, q で表す

- {A} $\Leftrightarrow !p \& !q$
- {B} $\Leftrightarrow p \& !q$
- {C} $\Leftrightarrow !p \& q$
- {D} $\Leftrightarrow p \& q$

状態 B と状態 D の状態集合を表す
論理関数 $F_{B,D}$

$$\{B, D\} \Leftrightarrow p \& !q \vee p \& q = p$$

状態 A から状態 C への遷移関係を表す論理関数 $F_{A,C}$

$$(A, C) \Leftrightarrow !p \& !q \& !p \& q$$

ソフトウェアシステムの動き(処理の流れ)を状態遷移図で捉える

JaSST '06 in 大阪

KEPCO MPS

モデル検査器

対象システム (仕様書) (ソースコード) → 検査用モデル (専用言語でプログラム作成) (検査対象の模擬システム)

```

MODULE main
VAR
Home:boolean;
Bus:(on, off);
ASSIGN
init(PC) := ON
next(PC) := case

```

モデル検査器 (大学、研究グループのオープンソース)

論理関数に変換して、発生し得る全状態を展開する

検査項目(時相論理式)が、満たされているかを自動検査する

検査項目が満たされない場合、反例を出力する

検査項目 (時相論理式)

```

SPEC
EF(vend=COIN & pow = ON & can = OFF...
SPEC
AG(vend=COIN & pow = ON & can =OFF...

```

JaSST '06 in 大阪 4

KEPCO MPS

モデル検査の流れ

a.モデル化 (背景知識・顧客要求 → 仕様書・設計書・ソース → モデル)

b.コード化 (モデル → SMVコード (MODULE MAIN (モデルの記述) VAR ...))

c.検査 (SMVコード → モデル検査器 → 出力結果 (項目1 true, 項目2 true, 項目3 true, 項目4 false))

d.CTL作成 (検査内容 (項目1... 項目2... 項目3... 項目4...) → CTL (検証内容記述) → 日本語で記述)

e.検査内容抽出 (検査内容抽出)

f.解析 (出力結果 → 解析)

検査対象の修正 (解析 → 仕様書・設計書・ソース)

JaSST '06 in 大阪 5

KEPCO MPS

ソフトウェア開発工程への適用

上流工程への適用: 要求分析 → 設計 → プログラミング

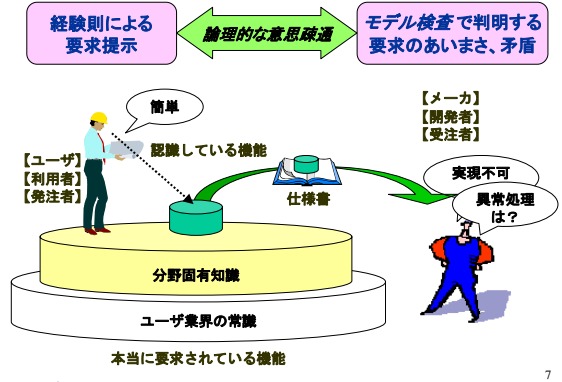
要求仕様書の検証 → 設計 → プログラム仕様書の検証

プログラムの検証 → デバッグへの適用 → 組込テスト → ソースコードの検証

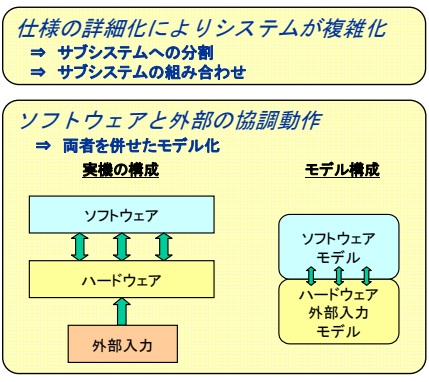
販売・保守

JaSST '06 in 大阪 6

仕様書作成の問題点



設計工程でのモデル検査



デバッグへのモデル検査適用

デバッグ

1. 不具合を再現する
2. 原因を発見する
3. 原因を修正する
4. 修正を確認する

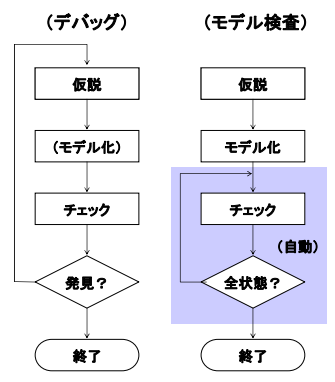
S.McConnel: Code Complete. 完全なプログラミングを目指して 日経BPソフトプレス

再現困難な場合には、原因予測(仮説)して確認する

- ・発生した不具合現象、ソフトウェア構造、経験から原因予想
- ・コード見直し、テストデータ入力、デバッグツール活用により確認

従来の方法では、デバッグの効率化が難しい

デバッグとモデル検査の流れ



モデル検査によるデバッグのメリット

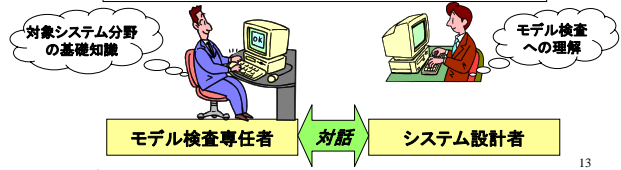
1. 網羅的な検査ができる
2. 不具合時のトレースが見れる
3. 不具合内容が判明しておりモデル化が容易
4. モデルが小さいと作業時間が短く、状態爆発も起きにくい

モデル検査とテストの比較

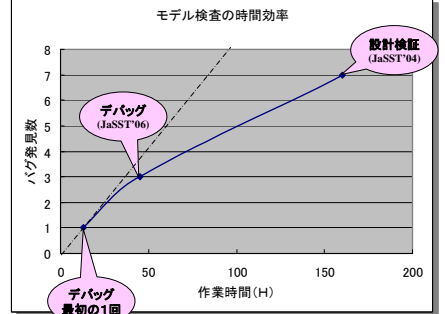
| | モデル検査 | テスト |
|----------------------------|-------------------|-----------------|
| Q.モデル化の時に不具合をどう推論するの? | モデル(状態遷移系)を検査する | コードを検査する |
| A.関係しそうな要素を漏れなくモデルに取り込むだけ。 | モデル作成が必要 | モデル作成不要 |
| | テストデータ不要(検査項目は必要) | テストデータが必要 |
| | 全状態を網羅的にチェック | 特定の状態だけをチェック |
| | 状態爆発すると答えが返らない | 答えは返るが、何度も検査が必要 |

モデル検査によるデバッグ手順

- ・検査対象の選定 (効果の大きいものを)
- ・検査メンバーの選定 (モデル検査が理解できる人)
- ・モデル化の準備 (必要な要素を全て抽出)
- ・モデルの作成 (ソースを織り込みモデル化)
- ・モデル検査の実施



デバッグの評価



デバッグでは、作業時間が飛躍的に短く、効率が良い

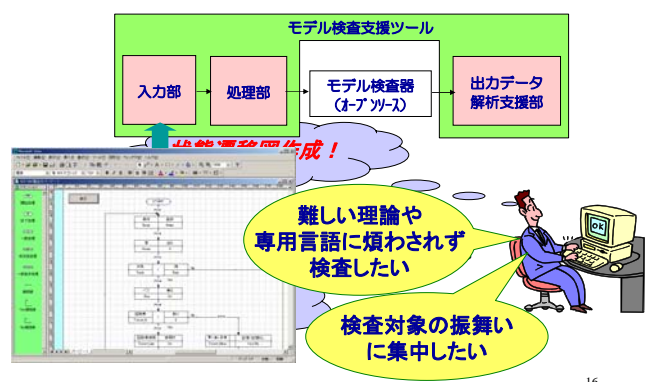
モデル検査で見つかるバグ例

組み込みソフトウェアの例 監視制御ソフトウェアの例

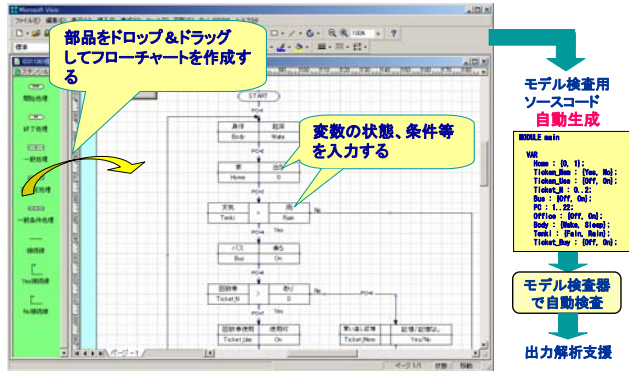
| | 仕様書の検証 | プログラム仕様書の検証 (~160H, 7件) | コードのデバッグ (~50H, 3件) |
|--------|----------------------------|--|----------------------------|
| 検査項目 | 要求仕様どおりにデータ書き デッドロックしない | 外部入力の変動に追従して切替できるか | 稀にデータ取得せず テスト時に一度だけ不正出力 |
| 発見したバグ | 想定外の書き手順 タイムアウト記述忘れ | 想定外の変数アクセス 想定外の割り込み 想定外の外部入力 割り込みのオーバーヘッド | ループ文の誤り コメントアウトの誤り |

仕様や動作の協調、場合分けや手順の正しさに関する不具合

モデル作成作業の支援



入力支援機能(フロー図)



今後の取り組み

【モデル検査によるソフトウェアテストの実践研究】
についてホームページを立ち上げました
<http://www.modelcheck.jp>

これまでのモデル検査によるソフトウェアテストの経験から、事例や実務適用に不可欠なノウハウの一部を公開しています

実践！ソフトウェアモデル検査 品質確認とバグ解決に向けて

ソフトウェア開発でお困りではありませんか？

高度情報化社会の進展によりソフトウェアはその重要性を高める品質に課する要求も高まる一方で、その一方、開発スピードは激化し開発期間も短くコストダウンの要求も高まることになりつつあります。設計手法やテスト手法に関する改善・習熟も求められています。

そして、このような課題に有効な技術として「モデル検査」がセミナー等で紹介されはじめました。実践的な検査、企業用に必要と思われるツール群の「モデル検査」ですが、実際にソフトウェア開発に適用を試みると、ちょっとしたことでも、なかなかうまく行かない場合が多いものです。

日々の報告も定期的な連絡で、「最初から知っていたら、取り返さなくて済んだらいいのに」と思うように、この程度のことでも日本語で紹介されていない？といった意見が多くなりました。

そこで、モデル検査を用いたソフトウェアテストの実用適用を目指す方々のために、独自の経験から開発した実践ノウハウを一冊を編纂し、モデル検査導入の質を高めるだけでなく、普及を促すために考えホームページを立ち上げました。

| | | | | | | |
|--------|-------|---------|------|------|------|-------|
| 取得済み内容 | トピックス | モデル検査とは | 適用事例 | 適用事例 | 適用事例 | モデル検査 |
|--------|-------|---------|------|------|------|-------|

【更新履歴】

2005/01/23 ホームページ公開