

KEPCO MPS

モデル検査のデバッグへの適用

JaSST '06 2006年1月30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

上流工程におけるモデル検査 (インデザインモデル検査) JaSST'04
組み込みシステムの設計書をモデル検査

2年間に少しずつ始めていたモデル検査

デバッグへの適用 JaSST'06

- ・ソフトウェア製作の現場で発生した問題を解決できる
- ・「時間」対「効果」が大きい
- ・導入が比較的容易

JaSST '06

KEPCO MPS

バグは無くならない

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.McConnell: 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

効率／非効率

- ・「原因予測可能」「再現性が高い」 ⇒ 比較的容易
- ・「原因予測不可能」「再現性が低い」 ⇒ 莫大な時間、困難

【調査範囲・内容の絞込みができる】

JaSST '06

3

KEPCO MPS

デバッグは仮説検証

原因を予測し、確認する

原因を予測し、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

全ての可能性を同時には追えない

- ・ひとつずつ仮説を立て、関係部分を抽出して確認している。
- ・仮説に基づく重点思考(試行?) ⇒ 見たい所だけを注視

無意識下のモデル化

JaSST '06

4

KEPCO MPS

モデル検査は全状態チェック

```

graph TD
    subgraph Debugging [Debugging]
        D1[仮説] --> D2[(モデル化)]
        D2 --> D3[チェック]
        D3 --> D4{発見?}
        D4 --> D5[終了]
    end
    subgraph ModelChecking [Model Checking]
        M1[仮説] --> M2[(モデル化)]
        M2 --> M3[チェック]
        M3 --> M4{全状態?}
        M4 --> M5{自動?}
        M5 --> M6[終了]
    end

```

JaSST '06

5

KEPCO MPS

モデル検査で不具合を再現

網羅的な検査ができる

- ・モデル化された範囲内で、全ての組み合わせを自動チェック
- ・モデル検査器(ソフトウェアツール)が公開されている

不具合時のトレースが見える

- ・モデルが検査項目を満足しない場合、反例(状態列)が出力される

JaSST '06

6

デバッグではモデル化が容易

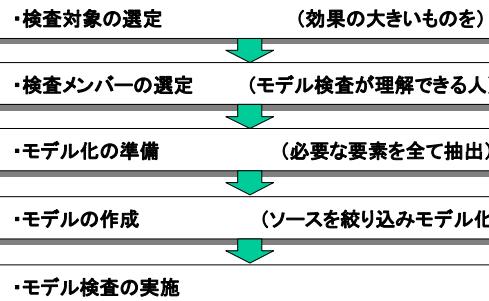
デバッグでは、不具合内容が判明している

- ・モデル化範囲を、不具合内容に関係するものに絞れる
- ・検査項目を、不具合内容から設定できる

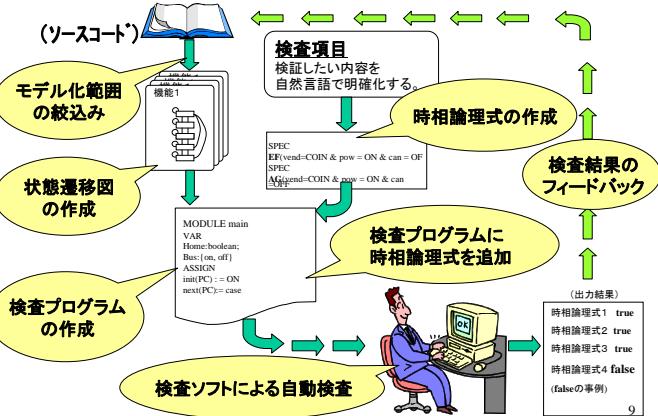
モデルが小さいと、何かと好都合

- ・モデル化のための作業時間が少なくて済む
- ・状態数も少なくなる ⇒ 状態数爆発が起き難い

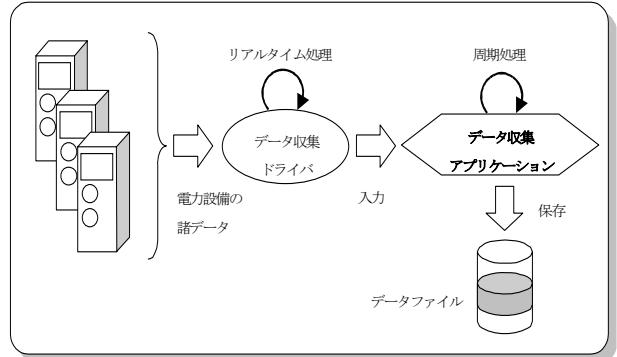
デバッグの手順



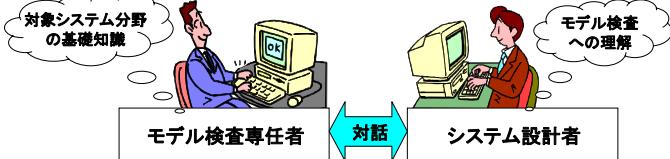
モデル検査の手順



モデル検査によるデバッグ事例



モデル検査メンバー



モデルの作成

不具合時内容: 稽にデータ収集アプリがデータを保存できない

モデル化範囲: データ収集アプリに絞込み 1730行

モデル化要素: データ収集・解析・保存に関わる部分に絞込み

モデル検査に要した作業時間

項目	時間(H)	
当初バグの モデル化	モデル化準備	4.0
	モデル作成	6.0
	モデル検査	3.0
改修内容の 確認	モデル改造	5.5
	再検査	1.5
合計		45

見つかったバグの例

```

1: for ( i=0; i<4000; i++ ) {
    データ変化点検出処理;
    if (データ変化点有) {
2:        if (変化点の数<規定の数) {
3:            変化点の数 += 1;
4:        } else{
5:            データの保存要求を出力;
6:        }
}

```

「変化点の数が規定の数に達するタイミング」と
「反復処理の最終回」が重複すると、
データ保存処理を行わない



13

見つかったバグの修正版

```

if (データ変化点有) {
    データ変化点の数 += 1;
}

```

```

if (変化点の数 => 規定の数) {
    データの保存要求を出力;
}

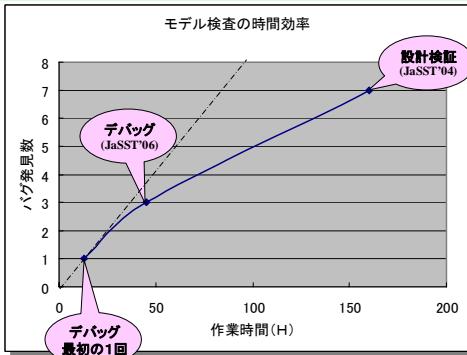
```

「変化点の数カウント」と「データ保存要求」を分割して修正

結果は単純なミスであったが、実際のコードでは
「ここまで分岐の深さ」「条件判断文におけるフラグの多さ」
から不具合検出は難しかった

14

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

15

モデル検査によるデバッグのFAQ(1)

Q. モデル化の際に、どのように推論(仮定)するのか？

A. 不具合に関係する要素、その関係を全てソースコードそのままにモデル化する

各要素が不具合発生時にどう組み合わさっているかを意識する必要はない

16

モデル検査によるデバッグのFAQ(2)

Q. モデル検査の専任者でないと、不可能か？

A. 現状では、まだまだ馴染みのないモデル検査であり、誰でも使えるわけではありません。しかし、システムの理解度から考えると設計者自身がモデル検査することが理想と考えます。そのため、モデル検査の敷居を低くする実践技術を開発しています。

17

モデル検査によるデバッグのFAQ(3)

Q. タイミング問題なら、何でも解決できるのか？

A. 適切なモデル化と検査項目が設定できれば、何でも見つけられるでしょう。但し、クロック数が大きくなると、システムが取り得る状態数が爆発的に増大するためにモデル検査器から答えが返ってきません。その場合、状態数を減らすためにモデルを抽象化する必要があります。

18

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品質に関する要求も高まる一方です。その一方、開発スピードは速くなり短納期化そしてコストダウンの要求も止まることはありません。設計手法やテスト手法に関する改良・改善もおこなわれていますがソフトウェア開発手法そのものが世界に近づいています。

そして、このように問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ています。網羅的な検査、計算機による自動検査といった能力満載の「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、「よつこくで済んだのに」とどうして、この程度のことが日本語で紹介されていないの？」といった意見が多く出ました。

そこで、モデル検査を用いたソフトウェアの実務適用を目指す方々のために、我々の経験から得られた実践ノウハウの一部を紹介し、モデル検査導入の壁をできるだけ低く下げ、普及を図りたいと考えホームページを立ち上げました。

contents menu

▶ 取り組み内容 ▶ トピックス ▶ モデル検査とは ▶ 活用事例 ▶ メンバー専用

【更新履歴】

2005/01/23 ホームページ公開