2B-5

組込みシステムへのモデル検査の適用

中川 雄一郎† 明神 智之† 小川 秀人†

株式会社日立製作所 中央研究所†

1. はじめに

近年、多くの分野の組込みシステムが複雑化、大規模化し、その開発コストの増大が深刻な問題となっている。そのような中で、機械的・網羅的に検証が可能なモデル検査技術[1]が注目を浴びている。

モデル検査技術は、検査対象が扱うモデルが大きくなると、検証に必要なメモリが数十ギガバイト～数百ギガバイトのオーダーになり実用に耐えられなくなる。モデル検査を実製品に適用する場合、検証範囲を限定し、検査すべき状態数を実用に耐えられる数に抑えるように、仕様・設計情報を抽象化するアプローチが取られている。しかし、検証範囲を限定することはモデルの正確性を下げることとなり、検証内容の正確さを妨げる要因となる。この２つのトレードオフを考慮した、検証対象の選定が必要となる。対象範囲の選定には、FTA（Fault Tree Analysis）を用いる手法などが考えられるが、影響波及分析が難しいソフトウェアにおいては、FTA を使用しての対象範囲の選択は困難である。

そこで、本研究では過去の不具合事例から検証範囲の選定と拡張を行うモデル検査の適用アプローチを考案した。複数の機器が通信を行い相互に情報の交換を行う組込みシステムを事例にとり、その有用性の検証を行った。

2. モデル検査適用アプローチ

本研究でのモデル検査適用プロセスを以下に示す（図 1）。

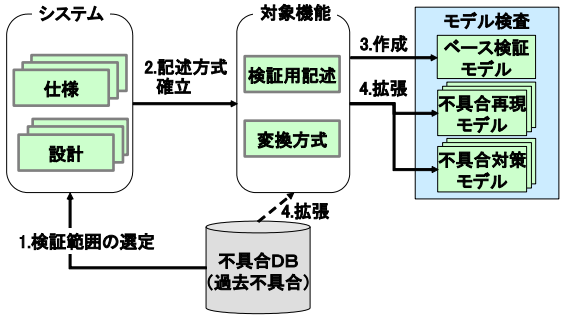

図 1 モデル検査適用プロセス

Application of Model checking process to Embedded system

† Yuichiro Nakagawa, Tomoyuki Myojin, Hideto Ogawa,

† Central Research Laboratory, Hitachi, Ltd.

2.1. 検証範囲の選定

過去不具合を分析しベース設計モデルの作成範囲を決定する。過去不具合を機能、不具合発生時の深刻度、複雑度（関連する機能の多さ、平行性の高さ）で分類し、深刻度と複雑度で優先順位付けを行い、最も優先度の高い機能を検証範囲とする。

2.2. モデル検査用記述方式の確立

上記で決定した機能に対し、PROMELA[2]等モデル検査用の言語で記述したモデル（以降、検証モデルと呼ぶ）作成のための中間モデルを作成する。中間モデルの記述方式、および検証モデルへの変換ルールを生成し、検証モデルを作成する。

仕様や設計の情報から直接検証モデルを作成するアプローチも考えられるが、不具合から随時検証範囲を拡張していく本プロセスでは、開発用に作成した仕様や設計の一部を検証モデルに変換するため、相互のトレーサビリティが取れなくなる恐れがある。そこで、検証モデル作成時の情報を中間モデルとして残しておくことで、変換のミスを防ぐ。

2.3. ベース設計モデル作成

検証範囲とした機能に対して、仕様や設計の情報から、過去不具合に関連する部分のみを抽出し、モデル検査用記述方式を行い、検査用変換ルールに従って検証モデルを作成する。ここで作成する検証モデルはベース検証モデルと呼ぶ。

2.4. 不具合再現モデル／不具合対策モデル作成

ベース検証モデルに対して、分析した過去不具合を再現できる不具合再現モデルを生成する。また、不具合を対策した不具合対策モデルも生成する。これを、複数の不具合に対して繰り返し実施することで、状態爆発を発生させずに、検証範囲の拡張を行うことが可能となる。

以下に、本プロセスの適用事例を示す。

3. モデル検査の適用試行

3.1. プロセス適用対象製品概要

今回、モデル検査の適用を行った組込みシス

テムは、以下の特徴を有している（図 2）。

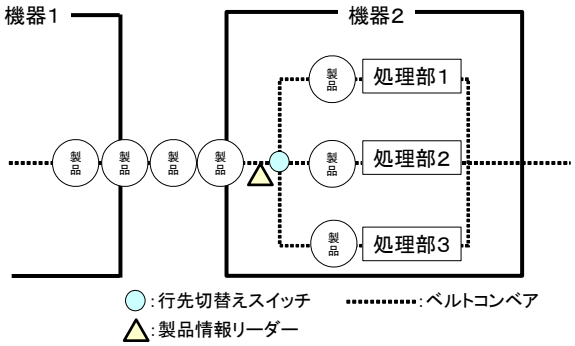

図 2 適用試行システム概要

- ベルトコンベアを用いて特定の製品を複数の機器に搬送する
- 複数の機器は通信を行い情報の交換を行う
- 搬送される製品は識別子を有しており、その識別子を参照してデータベース上にある製品情報の取得が可能となる
- ソフトウェアでは、製品情報リーダーを用いて製品情報を取得し、搬送経路を決定する

3.2. プロセスの適用試行

3.2.1. モデル検査用記述方式の確立

始めに過去不具合を分析し、深刻度と複雑度の観点から、搬送制御機能をモデル検査の適用対象と決定した。搬送制御機能は機器間の製品情報の受け渡しや、機器内の複数タスクが同時並行的に動作する。これにより、製品の識別子の読込みタイミングのずれが発生し、製品と識別子の情報不整合が発生していた。

3.2.2. ベース設計モデル作成

適用対象に対する記述方式、および変換ルールを生成した。記述例を図 3に示す。対象とする搬送制御機能では、検査対象は通信プロトコルを中心に設計されている。そこで、通信プロトコルを表現するのに適している UML2.0 のシーケンス図を作成のベースとし、シーケンス図をモデル検査用に拡張した搬送制御機能の記述方式とした。

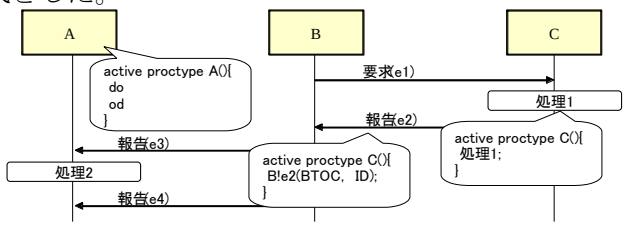

図 3 検証モデル用記述方式

3.2.3. ベース設計モデル作成、および不具合再現モデル／不具合対策モデル作成

上記の記述方式から変換ルールに従い、ベース検証モデルを生成し、識別子の読込みタイミングのズレ等の約 10 件の不具合が検出可能なモデルと、その対策モデルを作成し、関連する設計情報をモデルに追加した。

4. 評価

本研究でのモデル検査適用アプローチは、次のような効果が期待できる。

当初作成したベース設計モデルは、検証モデルで約 300step 相当であった。これを約 10 件の不具合を用いてモデルを拡張したところ、約 1500step まで拡張することが可能であった。

また、識別子の読込みタイミングのズレの検証では、5 つの不具合対策方針に対して、不具合再現モデルと、不具合対策モデルを用いることで最も有効な対策手段を選択することが可能であった。対策のうち 1 ケースでは不具合修正が不十分であることも検出できた。

これら対策案に関して、全てをテストで実施しようとすると、約 75000 件に相当する。これらを実機にてテスト行った場合、1 ケースにおよそ 10 分の時間がかかるため、全ケーステストするには約 74 人月の工数が必要となる。今回は、モデル検査の適用に約 6 人月程度の作業工数にて検証範囲の選定からモデル検査の実施、結果の検討までを行ったため、約 68 人月分の工数削減も効果が期待できる。

5. 結論

従来、モデル検査では検証対象が大きくなり、計算に必要なメモリ容量が数十ギガバイト～数百ギガバイトオーダーとなり、検査が最後まで終わらないケースがあった。しかし、不具合事例を元にすることで、状態爆発を発生させない範囲での検証範囲の拡張が可能となった。

また、不具合対策に対する妥当性の確認も同時に行うことができた。

参考文献

[1] Ranjit Jhala, "Software model checking", ACM Computing Surveys, 2009. Volume 41 Issue 4.

[2] Gerard J.Holzmann, “The SPIN Model Checker”, Addison-Wesley Professional , 2003