モデル検査技術によるUML設計検証に関する考察

岸 知二
北陸先端科学技術大学院大学
情報科学研究科

野田 夏子
NEC
共通基盤ソフトウェア研究所

1. 背景

組込みソフトウェアの規模や複雑さの増大とともに、その信頼性向上が大きな課題となっている。我々は組込みソフトウェアの設計品質向上のために、モデル検査技術を用いたUML設計検証の手法と支援ツールの開発を行い、事例への適用を試みている[1]。ここでUMLでの設計モデルをモデル検査のための検証モデルにマッピングする際には、検証する性質を明確にし、合目的な変換を行うことが重要となる。本稿では、本ツールの概要を紹介するとともに、検証モデルの構築手法について議論する。

2. UML設計検証ツール

図1は我々が開発しているUML設計検証ツールの機能概要を示したものである。

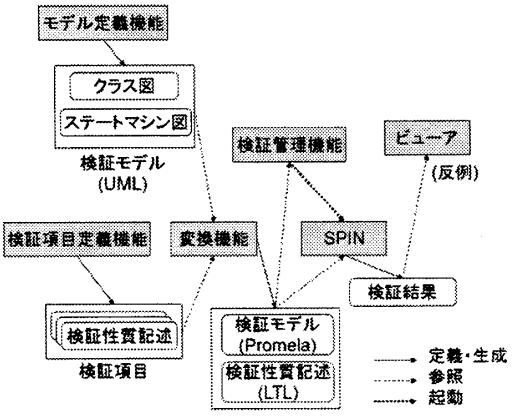

図1：UML設計検証ツールの機能概要

本ツールは、モデル定義機能によって定義されたUMLでの検証モデルと、検証項目定義機能で定義されたUML検証モデル上での性質を、変換機能でモデル検査ツールの理解できる形式にと変換し、検証管理機能によって制御されたモデル検査エンジン(SPINを利用)によって検証を行う。反例が示された場合には、ビューアによってUMLシーケンス図としてそれを表示する。

UML design verification using model checking techniques
Tomoji Kishi: Japan Advanced Institute of Science and Technology
Nattsuko Noda: NEC Corporation

3. 検証モデルの構築手法

以下、設計モデルと検証モデルの関係について考察する。

3.1 全体像

図2は、設計モデルと検証モデルとの関係を模式的に示したものである。

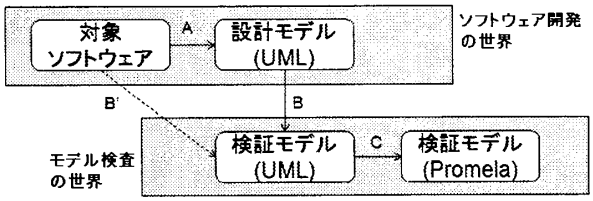

図2：設計モデルと検証モデルの関係

ソフトウェア開発の世界では、設計モデルは対象ソフトウェアの構造を抽象化したものと捉えられる。これは設計検証の観点から考えると、検証という目的に不要な部分を捨象したものと考えられる。したがって図2でのAは、開発者の設計意図に基づく抽象化作業である。

設計検証を行う方法としては、レビュー、スコアリング、シミュレーションなど様々なものが考えられ、検証内容に応じて適切な検証技術を選択することになる。モデル検査技術を用いた検証はタイミングの検証など組込みソフトウェアの設計にとって有効なものが多く、そうした側面の検証を行う際には、モデル検査の世界で活用できる検証モデルを構築しなければならない。本ツールでは検証モデルをUMLで表現するが、この作業(図2のBもしくはB')も設計意図に基づく抽象化作業である。一方UMLでの検証モデルからモデル検査ツールのための形式(SPINにおけるPromela)への形式変換(図2のC)はツールの機能によって実現している。

3.2 構築上のポイント

こうした検証モデル構築では、以下のような点が重要な考慮点になると考えられる。

- モデルの実行意味：タイミングなどの動的な側面の検証を行うためには、検証モデルの実行意味が重要となる。実際のソフトウェ

アは例えば特定の RTOS 等、なんらかの現実の実行メカニズム上で動作する。また UML 上での動作意味としては例えば MARTE など、モデル上での意味定義の提案がなされている。一方、SPIN の Promela などは独自の動作意味を持っている。そうした中、検証モデルをどのような動作意味上で構築するかが課題となる。もしも検証モデルが実際の検証対象と同等の複雑さを持つと、その検証モデル自体の正しさの確認が困難となる。したがって、検証モデルは簡素性を持つことが重要であると考えられ、現実の実行メカニズムを模擬することは一般には望ましくない。現実解としては、モデル検査エンジンの提供する実行意味の上に問題をマッピングするか、実行メカニズムのうち、検証の確認に必要な側面のみを抽象化して検証モデルの中でそれを実現するアプローチが妥当であると考えられる。

- モデルの抽象化：前述したように、検証モデルは検証目的のための対象の抽象である。一方、状態爆発などを回避するために抽象化を行うことも必要となりうる。特に後者の抽象化は、モデル検査技術特有の知識や技術が必要とされる場合があり、一般のソフトウェア技術者にとっては必ずしも容易な作業ではない。我々は実際の設計事例をモデル検査技術によって検証した際に、骨格構造への注目、パターン化、モデルの分割といったソフトウェア技術者が従来使ってきたモデル化技法が、状態爆発の回避に一定の有用性を持つことを確認した[2]。すなわち、検証モデルの構築に前だって、十分に適切な設計モデルを構築することが、検証目的のためだけでなく状態爆発への対応という観点からも有効だと考える。こういう意味からも、我々はいきなり検証モデルを構築するのではなく、いったん設計モデルを構築・吟味した上で検証モデルを作ることが望ましいと考える(図2のB)。
- 環境の定義：検証を行う際には、それが使われる想定や前提条件、あるいは検証を行うコンテキストやシナリオを明確にすることが重要である。こうした想定やコンテキストの明示化は、通常の設計においてはともすれば十分になされないことも多い。しかしながらすべての想定やコンテキストに対して正しさを検証することは一般に不可能であり、どのような条件下で検証を行うかを明らかにすることは極めて重要である。また状態爆発への対応として検証のコンテキストを制限することも有用である。我々は、想定やコンテキストの明確化のための想定モデリングとそこからの検証モデルの構築手法についても提案してきた[3]。

3.3 構築の手順

以上を踏まえ、検証モデルの構築の基本的な手順として、以下を提案する。

1. 設計モデルの構築：検証のためのモデル化の前に、設計目的やモデル化の視点を明確にして、ビューや抽象度に留意して設計モデルを構築する。
2. 検証項目の明示化：モデル検査技術を用いて検証したい検証項目を明確にする。
3. 想定やコンテキストの定義：想定モデリング等により、検証の想定やコンテキストを明確にする。具体的にはアクタの状態や検証のシナリオなどを明確にする。
4. 検証モデルの構築：上述の作業を踏まえて、検証モデルを作成する。必要に応じ、検証に必要な動的メカニズムの側面をモデル化する。
5. 検証・設計へのフィードバック：実際の検証を行い、結果を吟味して、設計や次の検証へのフィードバックとする。状態爆発に対しては、モデルの抽象化やシナリオの限定などでの対応を検討する。

4. おわりに

モデル検査技術の設計検証への適用は産業界でも大きく注目され、実用化の試みが行われている。こうした中、ソフトウェア技術者にとってわかりやすい手法や環境の整備が求められている。今後適用事例を増やしながら、そうした観点からの研究を進めていきたい。

参考文献

[1] Tomoji Kishi,, et.al.: Project Report: High Reliable Object-Oriented Embedded Software Design, The 2nd IEEE Workshop on Software Technology for Embedded and Ubiquitous Computing Systems (WSTFEUS'04) , 2004.

[2] 岸知二, 他:組込みソフトウェア設計検証へのモデル検査技術の適用と考察, IPA/SEC, SEC Journal 12 号, 2007.

[3] 岸知二, 他: プロダクトライン開発のための想定モデリング, システム制御情報学会組込みシステム研究分科会 情報処理学会組込みシステム研究グループ 合同研究会, pp39-46, 2006.