2W-07

# オブジェクト指向設計仕様書の正確さと安全性が同時に検証できるツール

黄 玉蓮　　P.G. ウィジャヤラトナ　　前川 守

電気通信大学大学院情報システム学研究科

## 1 はじめに

ソフトウェアの品質特性の中で正確さは最も基本的で、しかも最も難しい問題である。またソフトウェアは原子力発電所、航空システムなどの safety-critical なシステムにも多く使われており、安全性 [1] も確保しなければならない。そこで、正確さと安全性の両方を検証する必要がある。

[2][3] は要求仕様書から Correctness 表を、設計仕様書から Design 表を、安全性に関する知識、経験などから Safety 表を生成した後、Design 表と Correctness 表を比較して正確さを、Design 表と Safety 表を比較して安全性をそれぞれ検証する。この手法は表を使うためユーザにも理解しやすいが、複数個前のイベントまで参照する場合は階層化により表が複雑になるという問題が生じる。Symbolic model checking[4][5] は system model と property をそれぞれ状態遷移図と時相論理で記述して検証するものである。しかし、時相論理は一般のユーザには非常に分かりにくい。

現実にはソフトウェアエンジニアが安全性についてよく理解していない、または安全性エンジニアがソフトウェアをよく理解していない場合が多い [1]。そこで、ソフトウェアエンジニアがシステムの機能的記述を、安全性エンジニアが安全性に関する記述を行うという前提でツールを作成する。本ツールでは 3 つのモードがあり、(1)GSL[6] モードで機能的記述、(2)fault tree モードで安全性の記述、(3) 状態遷移図モードで検証対象となる設計仕様書を記述させる。検証は、GSL と fault tree を時相論理に変換して行う (図 1)。

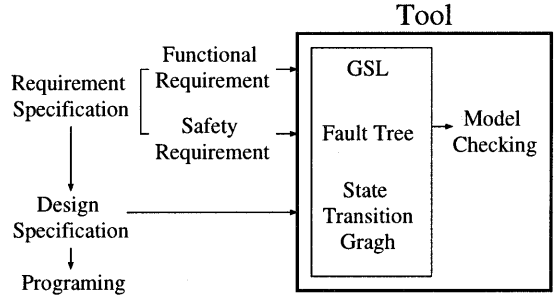


図 1: 本ツールの概要

## 2 Fault Tree

本研究では、fault tree(図 2[7]) を用いて安全性仕様を記述する。Fault tree では、統計的にも分析が可能になり、システムの信頼性が基準以内に収まるかどうかも判定できる。

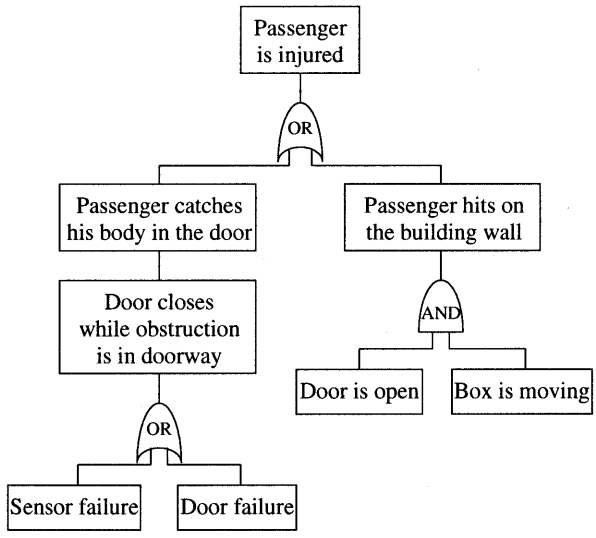


図 2: Fault tree の例 [7]

## 3 GSL

GSL[6] は形式的な仕様記述言語である。英語の構文に似ていて、一般ユーザにも非常に分かりやすい。本ツールではシステムの機能的要求仕様を GSL で記

---

A Correctness and Safety Verification Tool for Object-Oriented Design Specification

Yulian Huang, P.G.Wijayarathna and Mamoru Maekawa
Graduate School of Information Systems, University of Electro-Communications
1-5-1 Chofugaoka, Chofu, Tokyo 182-8585, Japan

述し、それを時相論理に変換する。GSL 記述と時相論理への変換の例をあげると、次のようになる。

| | |
|---|---|
| {**When** | Customer *arrive* Branch |
| | **While** Branch *close* |
| **If** | Branch *has* ATM |
| **Then_Do** | Customer *use* ATM} |

まず、Customer *arrive* Branch、Branch *close*、Branch *has* ATM、Customer *use* ATM などを一階述語論理の項 *arrive*(Customer,Branch), *close*(Branch), *has*(Branch,ATM),*use*(Customer,ATM) にそれぞれ変換する。そして、時間関係結合子を用いて時相論理式に変換すると 次のようになる。

$$((arrive(\text{Customer,Branch}) \amalg close(\text{Branch})) \wedge has(\text{Branch,ATM})) \Rightarrow use(\text{Customer,ATM})$$

ここで Π[8] は、時間関係結合子である。a Π b はイベント a が真であれば、そのすぐ後にイベント b が真となるという事を意味する。例えば Π を用いて、Allen の時間関係 [9] を表現すると、

$p$ before $q = q$ after $p = p \amalg (\neg p \wedge \neg q) \amalg q$
$p$ meets $q = q$ met-by $p = p \amalg q$
$p$ overlaps $q = q$ overlapped-by $p$
$= (p \wedge \neg q) \amalg (p \wedge q) \amalg (q \wedge \neg p)$
$p$ equals $q = p \wedge q$

## 4 検 証

本研究では symbolic model checking[4][5] の手法で検証を行う。GSL と fault tree を時相論理式に変換した後、検証対象の状態遷移図が時相論理式を満足するか symbolic model checking で検証を行う。

## 5 まとめと今後の課題

本ツールは機能的記述と安全性の記述に、それぞれユーザに分かりやすい GSL と fault tree を採用した。しかし、Fault tree は AND, OR, Priority-AND ゲートを用いて記述するが、それだけでは時間関係を十分に表現できない。時間関係をより正確に表現できるゲートを導入する必要がある。

## 参考文献


[1] N. G. Leveson: *Safeware: System Safety and Computers*, Addison Wesley, 1995.

[2] E. M. Kim, S. Kusumoto T. Tsuchiya and T. Kikuno: *An Approach to Safety Verification of Object-Oriented Design Specification for An Elevator Control System*, the third Workshop on Object-Oriented Real-Time Dependable System, February 5–7, 1997(WORDS '97).

[3] E. M. Kim, S. Kusumoto and T. Kikuno: *An Approach to Safety and Correctness Verification of Software Design Specification*, the 6th ISSRE, pp. 78–83, 1995.

[4] K. L. McMillan: *Getting started with SMV*, http://www-cad.eecs.berkeley.edu/ kenmcmil/.

[5] Edmund M. Clarke, Jr., Orna Grumberg, and Doron A. Peled: *Model Checking*, The MIT Press, Cambridge, Massachusetts, London, England.

[6] P. G. Wijayarathna, Y. Kawata A. Santosa, K. Isogai and M. Maekawa: *GSL: A Requirement Specification Language for End-User Intelligibility*, Software Practice and Experience, VOL.28(13), pp. 1387–1414, November 1998.

[7] T. Tsuchiya, H. Terada, S. Kusumoto, T. Kikuno and E. M. Kim: *Derivation of Safety Requirements for Safety Analysis of Object-Oriented Design Documents*, COMPSAC '97 - 21st International Computer Software and Applications Conference, August 11–15, 1997.

[8] P. G. Wijayarathna, A. Santosa, K. Isogai, Y. Kawata and M. Maekawa: *Representing Relative Temporal knowledge with TAND connective*, Eighth Ireland Conference on Artificial Intelligence(AI-97) Conference Proceedings, Vol.2, pp.80–87, September 1997.

[9] J. F. Allen: *Maintaining Knowledge about Temporal Intervals*, Communications of the ACM, Vol.26, pp. 832–843, 1983.