高信頼情報通信ソフトウェア開発における欠陥防止と検出の自動化に関する研究

Research on Automatic Techniques for Prevention and Detection of Defects in Development of Dependable Information Communication Software

劉 少英（Shaoying LIU, Ph. D.）

法政大学 情報科学部 教授

(Professor, Hosei University, Faculty of Computer and Information Sciences)

IEEE Computer Society　British Computer Society　日本ソフトウェア科学会

受賞：国際会議 ICECCS' 96 優秀論文賞 (1996)　Fellowship of British Computer Society (2010)

著書：Formal Engineering for Industrial Software Development Using the SOFL Method, Springer-Verlag, 2004

研究専門分野：ソフトウェア工学

はじめに　情報通信ソフトウェアシステムに潜む欠陥は、信頼性・安全性の問題を引き起し、延いてはシステムトラブル、ダウンによる莫大なエネルギー損失と、大きな環境問題、社会問題と発展しかねない。特にインターネット、スマートフォンなど急増する情報通信システムに使われる基盤ソフトウェア、OS および様々な応用ソフトウェア(オンライン銀行システム、航空券予約購入システムなど）には高い信頼性・安全性の確保が必要であり、開発期間とコストを大幅に減らすことは情報通信ソフトウェア産業にとって喫緊の重要課題であり、依然として解決していない。

そこで、本研究では、効率的な形式仕様記述技術および形式仕様に基づく自動テスト技術の研究を行った。前者では、ソフトウェアシステムへの要求と設計に関する考え漏れた機能および安全性に関わる欠陥を防ぐことができ、後者では、開発されたプログラムに埋め込まれた欠陥を効率的に検出することができる。

１．研究目的

数学に基づき開発されたソフトウェアの要求定義と機能検証を厳密的に行うことができる従来の形式手法（例えば、VDM[1]、B-Method[2]、CafeOBJ[3]など）の使い難い点と間違いやすい点を改善し、効率的な形式仕様作成技術およびソフトウェア欠陥の自動検出技術を提案することが、本研究の目的である。具体的には、様々な機能を明確に定義する形式仕様を表すテンプレートである仕様パターンを用いて、ソフトウェア機能の形式仕様を効率的に作成する技術、考え漏れた機能や安全性の制約を発見するために仕様の整合性と妥当性を厳密的に検査する手法と支援ツールおよび形式仕様に基づくプログラムの自動テスト技術の開発を目指す。

２．研究背景

集合論、論理学、代数などを含む離散数学に基づき開発された形式手法は、ソフトウェアの信頼性を向上させる最も厳密的なモデル駆動（model-driven）開発技術と考えられる。その中核となる形式仕様記述技術は、高信頼性と安全性が求められるシステム開発企業で採用されたこともある。しかし、形式仕様記述プロセスの中でどのような側面を考慮すれば、考え漏れた機能や安全性の制約を発見でき、埋め込まれた欠陥を自動的に検出できるかということには、課題がまだ残っている。更に、企業の開発現場で形式手法の使い難い点と間違いやすい点もあることが判明し、形式手法が企業で普及することはなかなか困難な局面になっている。

このような課題を解決するために、形式仕様パターン、形式仕様の検証および形式仕様に基づくプログラムの自動テストに関して、様々な研究が行われた。Stepney et al.は、形式仕様を作成するに使われる 6 種類の仕様パターンを提案し、小さな事例で仕様パターンの役割を説明した[4]。同じような考えで、Ackermann と Turowski は、Unified Modelling Language（UML）に使われている形式仕様記述言語 OCL に対して、様々な仕様パターンを提案した[5]。また、Leveson グループは、SpecTRM-RL という要求

仕様記述言語で形式仕様を記述することを支援するツール SpecTRM を研究開発した[6]。

ソフトウェア要件定義或いは設計の形式仕様の検証に関しては、Heyer が UML で作成した機能要求を表すユースケース図に含まれている操作に対して、事前条件と事後条件を作成した上でシステム設計を反映するシーケンス図の妥当性を検査する手法を提案した[7]。ここで、事前条件とは、操作の入力が満たすべき条件であり、事後条件とは、操作の出力が満たさなければならない条件である。Heitmeyer グループは、Software Cost Reduction（SCR）という表式言語で記述される形式仕様の整合性を自動的に検査する技術を提案した[8]。

形式仕様に基づくプログラム検査とテストの研究に関しては、Parnas グループはカナダの Darlington 原子発電所用ソフトウェアの検証するプロジェクトを通じ、Document Driven Inspection（DDI）という形式仕様に基づくプログラム検査手法を提案した[9]。Khurshid グループは、形式仕様言語 Alloy で記述した操作仕様の事前条件によってテストケースの自動生成技術を提案し、この技術の支援ツール TestEra を研究開発した [10]。

３．研究方法

実用性が高い形式仕様記述言語 SOFL（Structured Object-Oriented Formal Language）を基に、図 1 に示すソフトウェア開発プロセスと支援技術を研究開発した。

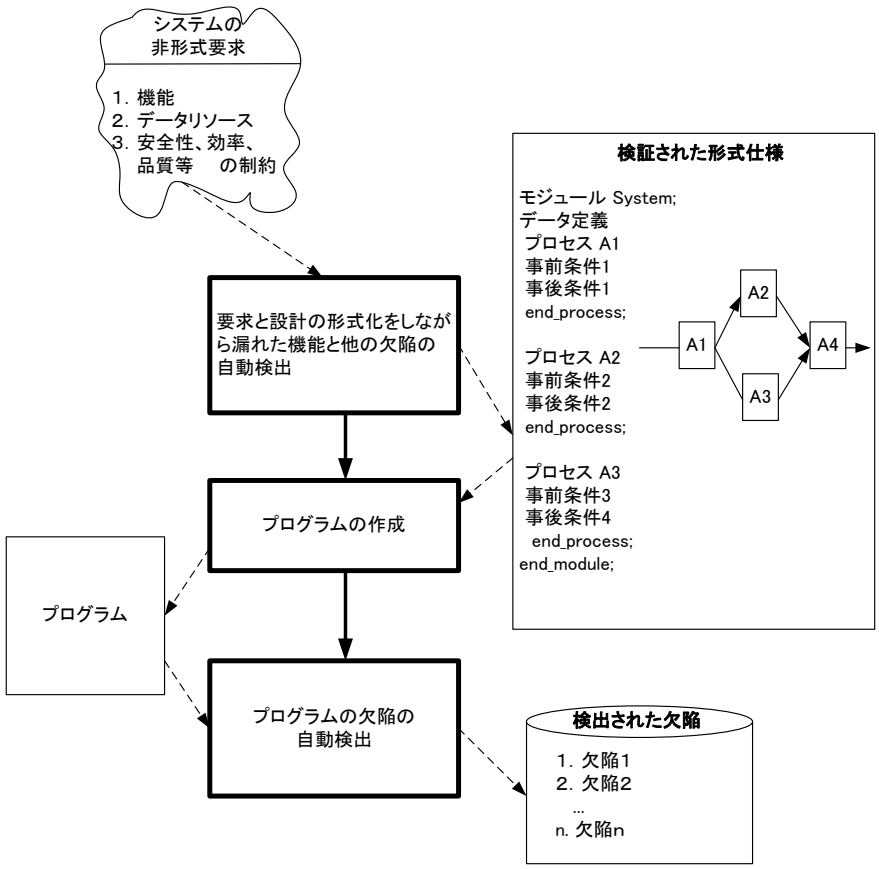

図１　SOFL でのソフトウェア開発プロセス

高信頼情報通信ソフトウェア開発における欠陥防止と検出の自動化に関する研究

Research on Automatic Techniques for Prevention and Detection of Defects in Development of Dependable Information Communication Software

最初に、自然言語でシステムの機能、データリソースおよび安全性等の制約を含む非形式要求仕様を簡潔に作成する。次に、実用性が高い SOFL 形式仕様記述言語で、その非形式仕様に基づき形式仕様を記述しながら、システムの様々な機能、データ構造および制約を細かく検討する。この過程を効率的に行うために、仕様パターンに基づく形式仕様を作成する手法を提案した[11]。一つの仕様パターンには、パターン名、パターンを使える条件、パターンの要素および形式表現のテンプレートが含まれている。仕様パターンが効率的に検索したり適用したりするために、全ての仕様パターンは木構造にまとめられる。コンピュータがこのような木構造を利用して、形式仕様を作成する開発者と交信しながら、機能要求の形式仕様を作成することができる。また、この過程の中で、開発者は支援ツールからのガイドを受け取りながら、機能と安全性の制約を完全に理解することができ、事前に考えられなかった機能と制約を引き出し、漏れた機能と制約を自動的に検出することも可能である。但し、現在の技術では、一つのプロセス（操作）の事前条件と事後条件を形式的に表現することができるが、仕様の他の部分を従来の方法で記述することが必要である。このように作成される SOFL 形式仕様の構造は、図 1 の中で右側の部分に示されている。

形式仕様の整合性と妥当性を保証するために、形式仕様のアニメーションによる厳密な検査技術を開発した[12]-[16]。形式仕様の中でシステムの構造を表す条件データフロー図（CDFD）に基づき、形式仕様に定義された機能シナリオを自動的に導出して、それぞれの機能シナリオで表す振る舞いのアニメーションを行いながら、その機能シナリオの整合性と妥当性を検査する。一つの機能シナリオは、入力データフローから出力データフローまでの操作シーケンスである。機能シナリオのアニメーションとは、そのシナリオに含まれている操作の振る舞いを入力データと出力データ間の関係を視覚的に表現することである。このようなアニメーションによって、形式仕様の整合性を検査することができ、妥当性を確認することもできる。全ての重要な性質を確認した上で、形式仕様によってプログラムを作成する。この仕様検査技術の支援ツールのスナップショットを図 2 に示す。

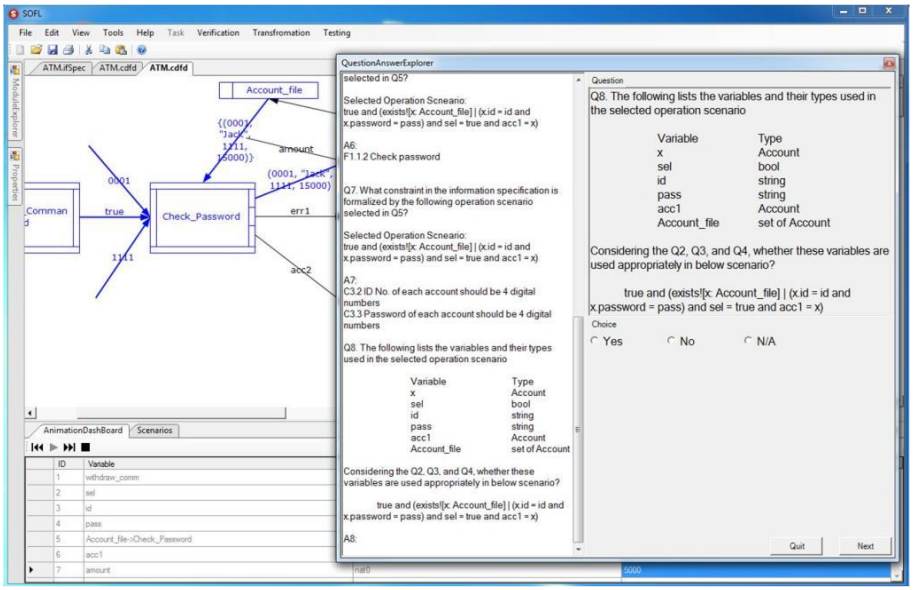

図２　形式仕様のアニメーションによる検査技術の支援ツール

図の左上部分は、条件データフロー図 CDFD およびそれにより抜き出された機能シナリオの一部を示す。図の右部分は、厳密な検査を行うためにツールとユーザの交信状況を示す。図の左下部分は、選択された機能シナリオのアニメーションを行うために必要なテストケースの生成に関するエリアである。

作成されたプログラムに含まれる欠陥を効率的に検出するために、形式仕様に基づくプログラムテスト技術を提案した[17],[18]。図 3 に示すように、この技術でテストを行うためには、三つの段階が必要である。第一段階で、形式仕様を基にテストケースを自動的に生成する。第二段階で、生成されたテストケースを用いてプログラムを実行させる。第三段階で、実行した結果を分析する。

本研究では、テストケースの生成を効率的に行うために、操作の機能を定義した形式仕様の事前条件と事後条件を生かして、テストケースを自動的に生成する方法とアルゴリズムを開発した。具体的には、事前条件と事後条件によってテストケースを生成するために必要な「テスト条件」を自動的に形成して、それに基づいてテストケースを自動的に生成する。テスト条件とは、事前条件と事後条件にあるガード条件の論理積であり、その中に操作の出力変数が含まれていない。テスト条件の原子論理式により、入力変数に代入するテストケースとした値を生成する。生成されたテストケースは、必ずその原子論理式を満たす。次は、原子論理式から構成された論理積という論理式を満たす入力変数のテストケースを生成する。このため、原子論理式間の入力変数の依存関係によって、論理積に含まれる全ての原子論理式を原子論理式集合に分ける。これらの集合を用いて、適切なアルゴリズムによって、テストケースを自動的に生成する。最後に、論理和によって、テスト条件を満たすテストケースの集合を生成する。また、テスト結果を分析するために、欠陥がプログラムに存在するかどうかを判定できるテストオラクルを明確に定義し、それを形式仕様から自動的に抜き出すアルゴリズムも確立した。これらの技術を支援するソフトウェアツールのプロトタイプも開発した。但し、本自動テスト技術では、欠陥がどのプログラムパスに含まれるかまでは確定できるが、そのパスの中で具体的な場所およびその欠陥の種類を確定することはできない。更に、テスト対象としたプログラムを実行させる「ドライブモジュール」の自動生成アルゴリズムは、自動テスト技術の実現にとって不可欠であるが、それはまだ開発していない。

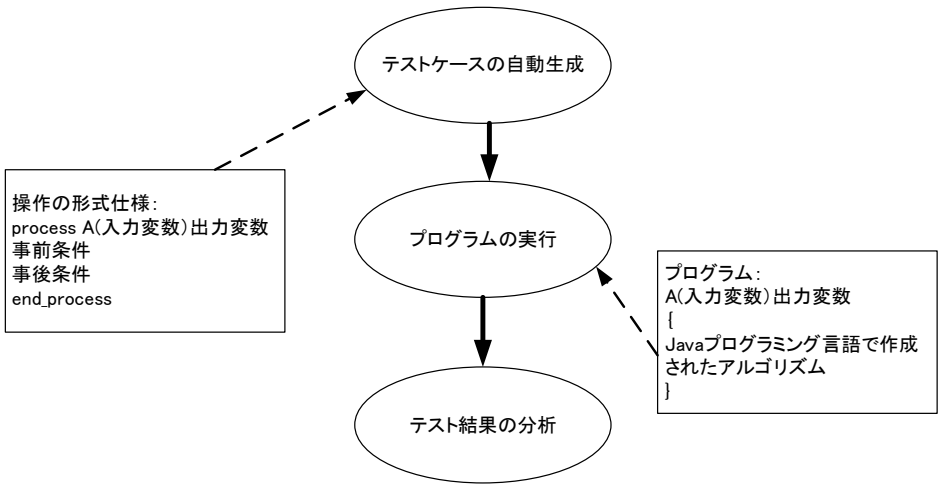

図３　形式仕様に基づくテスト技術の基本原理

４．将来展望

本研究で提案したソフトウェア開発における欠陥防止と欠陥検出の自動化技術は、高信頼情報通信ソフトウェアを含む幅広い応用領域のソフトウェア開発に適用できるが、完全に自動化を実現するには課題が残っている。主な課題は、ソフトウェアの要求仕様の品質を保証するために、仕様の完全性と様々な一致性という性質を仕様から自動的に抜き出すアルゴリズムの確立と、それらの性質を自動的にテストする技術の開発および効率的な支援ツールの開発にある。また、上述した形式仕様に基づくプログラムテストに関する課題も残っている。

これらの課題を系統的に解決するためには、論理式によるテストケースの自動生成技術と定理証明理論およびプログラムの正確性を証明する Hoare 論理と適切に統合して、形式仕様の様々な性質を自動的に検証でき、プログラムの欠陥の自動検出技術を開発することが重要である。具体的には、「自動テストに基づく形式検証」という新たな理論と方法および支援ツールの開発に集中することである。潜在的な課題としては、どのようにして自動テストを定理証明プロセスに統合して、実用性と有効性と共に高い形式検証方法を確立し、また、どのようにして十分なテストケースを効率的かつ自動的に生成できるかということである。このような課題を解決して確立する形式検証技術では、ソフトウェア仕様の様々な性質を表す定理の有効性の判定ができ、プログラムに含まれる欠陥をより効率的に検出することができる。これによって、ソフトウェアの生産性を向上させることができるだけではなく、ソフトウェア構築プロセスの中でエラー防止や開発されたソフトウェアの信頼性、安全性および堅牢性を向上させることもできる。

参考文献

[1] C. B. Jones. Systematic Software Development Using VDM. 2nd edition, Prentice Hall. 1990.

[2] J. R. Abrial. The B-Book: Assigning Programs to Meanings. Cambridge University Press. 1996.

[3] K. Futatsugi and A. Nakagawa. An Overview of CAFE Specification Environment: an Algebraic Approach for Creating, Verifying, and Maintaining Formal Specifications over Networks. Proceedings of the First International Conference on Formal Engineering Methods (ICFEM'97), IEEE Computer Society Press, pp. 170-181. 1997.

[4] S. Stepney, F. Polack, and I. Toyn. An Outline Pattern Language for Z: Five Illustrations and Two Tables. ZB 2003, LNCS 2651, Springer, pp. 2-19. 2003.

[5] J. Ackermann and K. Turowski. A Library of OCL Specification Patterns for Behavioral Specification of Software Components. 18th International Conference on Advanced Information Systems Engineering (CAiSE'06), Springer-Verlag, pp. 255-269. 2006.

[6] N. G. Leveson, J. D. Reese, and M. P. E. Heimdahl. SpecTRM: A CAD System for Digital Automation. Proceedings of 17th Digital Avionics System Conference (DASC 98), *The AIAA/IEEE/SAE*, vol. 1, pp. B52/1-B52/8. 2003.

[7] T. Heyer. Semantic Inspection of Early UML Designs. IEEE Transactions on Software Engineering, 28(4), pp. 413-430. 2002.

[8] C. L. Heitmeyer and R. D. Jeffords and B. G. Labaw. Automated Consistency Checking of Requirements Specifications. ACM Transactions on Software Engineering and Methodology, 5(3), pp. 231-261. 1996.

[9] D. L. Parnas and J. Madey and M. Iglewski. Precise Documentation of Well-Structured Programs. IEEE Transactions on Software Engineering, 20(12), pp. 948-976. 1994.

[10] S. Khurshid and D. Marinov. TestEra: Specification-based Testing of Java Programs using SAT. Journal of Automated Software Engineering, 11(4), pp. 403-434. 2004.

[11] X. Wang and S. Liu. An Approach to Declaring Data Types for Formal Specifications. 3rd International Workshop on SOFL + MSVL (SOFL+MSVL 2013), LNCS 8332, Springer, pp. 135-153. 2013.

[12] M. Li and S. Liu. Automated Functional Scenarios-based Formal Specification Animation. 19th Asia-Pacific Software Engineering Conference (APSEC 2012), pp. 107-115. 2012.

[13] M. Li and S. Liu. Reviewing Formal Specification for Validation Using Animation and Trace Links. Proceedings of 21th Asia-Pacific Software Engineering Conference (APSEC 2014), pp.286-293. 2014.

[14] M. Li and S. Liu. Traceability-Based Formal Specification Inspection. Proceedings of Eighth International Conference on Software Security and Reliability (SERE), pp.167-176. 2014.

[15] S. Liu, F. B. Zainuddin, and M. Li. Integrating Animation into Informal Specification Writing for Requirements Analysis. Proceedings of 3rd International Conference on Informatics Engineering and Information Science (ICIEIS 2014), SDIWC, pp. 136-143. 2014.

[16] M. Li and S. Liu. Tool Support for Rigorous Formal Specification Inspection. Proceedings of IEEE 17th International Conference on Computational Science and Engineering (CSE 2014), pp.729-734. 2014.

[17] S. Liu. Utilizing Hoare Logic to Strengthen Testing for Error Detection in Programs. The Turing Centenary Conference, Manchester, UK, EPiC Series, pp. 229-238. 2012.

[18] S. Liu and S. Nakajima. Combining Specification-Based Testing, Correctness Proof, and Inspection for Program Verification in Practice. Proceedings of 3rd International Workshop on SOFL+MSVL (SOFL+MSVL 2013), LNCS 8332, Springer, pp. 3-16. 2013.

この研究は、平成２３年度ＳＣＡＴ研究助成の対象として採用され、平成２４～２６年度に実施されたものです。