SOFLの研究開発の経験による形式手法の課題の考察†

劉少英

法政大学・情報科学部

アブストラクト:形式手法が企業で普及していない主な原因の一つは低い実用性にあると考えられる。この問題点を解決するために、形式手法を企業に「導入」するではなく、既存のソフトウェア工学技術と「統合」することが重要である。本ポジション論文では、作者の SOFL 形式工学手法の研究開発、応用、および教育の経験に基づき、簡易性、明確性、および対話性を持つ実用性が高い形式工学手法を獲得する様々な統合方法を議論し、今後の課題を示す。

Study of Challenges to Formal Methods Based on Our Experiences in Research and Development of SOFL

Shaoying Liu

Faculty of Computer and Information Sciences, Hosei University

Abstract: One of the major reasons for formal methods not being widely used in industry lies in their poor practicality. An effective way to solve this problem is not to "introduce" formal methods to industry but to "integrate" them with commonly used software engineering techniques in practice. In this position paper, we discuss various approaches to integrating formal methods with existing software development techniques on the basis of our experiences gained from developing, applying, and teaching the SOFL formal engineering methods, and point out future research directions.

1. はじめに

形式手法は,数学や論理学に基づく形式的体系により、システムの記述や検証を厳密的に行うものである．８０年代後期から企業に導入する活発な展開が始まるが、９０年代の中ごろから形式手法の理論上の制限および実用性に関する問題点が徐々に認識されてきた [1,2,3]．数学的記述の多用により引き起こされる形式仕様の作成の難しさと読みにくいこと、実務者の抽象能力への要求、教育のコスト、可用性と有効性などに関する問題点は、形式手法の実用性に影響を与えている。

そこで、実用性が高い形式手法を目指す私たちが過去２５年間の研究によって開発した SOFL 形式工学手法は、形式手法を既存のソフトウェア工学技術に統合する技術を提案し、実用性が高い形式手法を創設する道筋を示している。

†本研究は、IPA 2012年度ソフトウェア工学分野の先導支援事業の委託に基づいて行われた。

2. SOFL 言語と形式工学手法

SOFL 形式工学手法は、SOFL (Structured Object-Oriented Formal Language) 形式仕様記述言語に基づき確立された次の三つの技術から構成された手法である。（１）SOFL 三段階形式仕様記述技術、（２）形式仕様に基づくプログラムのインスペクション技術、および（３）形式仕様に基づくプログラムのテスト技術。

2.1. SOFL 三段階形式仕様記述技術

SOFL は、企業に既に定着している要求分析と設計のデータフロー図, 形式仕様言語 VDM-SL、および Petri Nets を適切に統合することによって開発された構造化仕様記述およびオブジェクト指向の実装を支援する形式仕様記述言語である [4,5].

SOFL 三段階形式仕様記述は、要求分析によってシステムの形式要求仕様を SOFL 非形式仕様、半形式仕様の作成を通じて漸進的に作成する技術である。応用領域の資料や現実世界のシステムに関する知識などを検討することから始め,顧客とコミュニ

ケーションを取りながら，システムの非形式要求仕様を作成する．要求された*システム機能*，その機能を実現するために必要な*データリソース*，および機能またはデータリソースに対する*制約*（ビジネスルール、安全性、安心性など）が簡潔に記述される．

非形式仕様の内容をより明確に定義するためには，半形式仕様を作成する．半形式仕様は，SOFL モジュールの集合である。一つのモジュールには、必要な定数、型、状態変数、不変条件、およびプロセス（操作）仕様が含まれる。非形式仕様に記述された関連するシステム機能，データリソースおよび制約をモジュールにまとめる。全てのデータ型、状態変数、および全てのプロセスの入力変数、出力変数を形式的に宣言しながら、プロセスの振る舞いを構造化英語（または日本語）のような非形式言語で表現した事前条件と事後条件により定義する。

形式仕様は、半形式仕様で定義されたプロセス間のデータの依存関係により形成されたシステムのアーキテクチャを、CDFD （Condition Data Flow Diagram）と呼ばれる形式化された DFD によって定義した上で、各プロセスの振る舞いを形式的な事前条件と事後条件で定義する。

2.2. **形式仕様に基づく**インスペクション技術

厳密性が足りない伝統的なインスペクション技術の効率と効果を改善するために、形式仕様に定義された「機能シナリオ」を活用して、実装されたプログラムのパスに含まれるバグを発見することができる [6]。具体的には、形式仕様から全ての可能な機能シナリオを導出し、それらのシナリオから自動的にチェックリストを作成、対応するプログラムパスに結びつき、そのパスが対応する機能シナリオを正しく実装するかどうかをチェックする。このなかである作業は自動化できるため、インスペクションの効率を向上させることができる。

2.3. **形式仕様に基づくプログラムの**テスト技術

事前条件と事後条件で定義した SOFL プロセスの形式仕様に基づきテストケースを自動的に生成して、実装されたプログアムのテストを行うことができる [7]。具体的には、形式仕様に定義された機能シナリオを基に、様々なデータ型の原子述語式、論理積、論理和などを満たすテストケースを自動的に生成する。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、機能シナリオによってその実行結果を自動的に分析する。

3. 形式手法の移転に関する課題

SOFL 形式工学手法に関する研究開発、応用、および教育の経験によると、特別な強い理由がない限り、形式手法を企業にそのまま導入するのは困難だと感じる。但し、形式手法の基本技または原理を、企業でよく使われている開発技術とプロセスモデルに適切に統合して、より厳密的実用性が高い開発技術を確立することが可能である。その可能性を実現するために、教育を普及させる一方で、次の問題点の解決に向けて、更なる研究が必要である。

1） どのように実務者でも形式仕様を簡易に記述できるか。どのような記述プロセスを取れば、ユーザの要求を効率的、正確的、および完全に獲得できるか。
2） 確定された形式仕様をどのように活かして、高効率かつ強力なプログラムを作成できるか。
3） Hoare 論理に基づく形式検証技術とモデル検査技術は、どのようにソフトウェアレビュー、インスペクション、テストなどバグ発見技術を統合して、より有効な技術を確立できるか。
4） どのようにソフトウェア支援ツールを構築すると、統合して得た形式工学手法を有効に支援することができるか。

これらの課題を効果的に解決しない限り、形式手法を企業で活用することが困難だろう。

参考文献

[1] S. Liu and R. Adams. “Limitations of Formal Methods and An Approach to Improvement”, APSEC'95, Australia, 1995, pp. 498-507.

[2] J. C. Knight *et al*. “Why Are Formal Methods Not Used More Widely?”, NFM 2012, 1997, pp. 1-12.

[3] D. L. Parnas. “Really Rethinking Formal Methods”, The Computer, 43(1), 2010, pp. 28-34.

[4] S. Liu *et al*. “SOFL: a Formal Engineering Methodology for Industrial Applications”, IEEE TSE, 24(1), 1998, pp. 337-344.

[5] S. Liu. “Formal Engineering for Industrial Software Development using the SOFL Method”, Springer, 2004, 428 pages, ISBN 3-540-20602-7

[6] S. Liu *et al*. “Formal Specification-Based Inspection for Verification of Programs”, IEEE TSE, 38(5), 2012, pp.1100-1122.

[7] S. Liu and S. Nakajima. “A Decompositional Approach to Automatic Test Case Generation Based on Formal Specifications", SSIRI 2010, Singapore, June 9-11, 2010, pp. 147-155.