形式手法のソフトウェア開発利用の課題と事例

小川　秀人 †1　　市井　誠 †1　　新原　敦介 †1
鈴木　康文 †1　Phan Thi Thanh Huyen†1　坂井田　真也 †1

ソフトウェア開発での形式手法の活用が期待されている．活用事例が多く報告されているが，ソフトウェア開発に形式手法が受け入れられているとは言い難い．本稿では，モデルの断面という観点でソフトウェア開発と形式手法の相違について考察し，この差異を埋めることで形式手法の利用が促進すると考える．この考え方に基づき，プログラム解析をベースとした形式手法の活用事例を説明する．

Lessons learned from Formal Method Applications to Software Development

Hideto Ogawa,†1 Makoto Ichii,†1 Daisuke Shimbara,†1
Yasufumi Suzuki ,†1 Phan Thi Thanh Huyen†1
and Shinya Sakaida†1

The application of formal methods(FM) to software development is demanded. Although a lot of application examples were reported, FM is not widely used in practice. In this article, the difference between software development and FM will be discussed from a view point of cross sections of their models. Moreover, some applications FM are reported.

1. はじめに

複雑化が進むソフトウェア開発における形式手法への期待は言うまでもない．形式的仕様記述やそれに対する定理証明のほか，モデル検査手法や SAT/SMT ソルバを用いた検査，静的もしくは動的なプログラム解析に基づくプログラム検証まで，様々なアプローチでソフトウェア開発を支援する取り組みが行われている．本稿ではこれらを広く形式手法と捉える．

形式手法の開発適用事例の報告は多いが，ソフトウェア開発での形式手法利用が一般的とは言い難い．むしろ，機能安全規格などで形式手法の利用が推奨されるが，対応に苦慮している実情すらある．

本稿では，形式手法とソフトウェア開発の特性の相違に着目し，形式手法の活用の課題と利用方法を述べ，いくつかの事例を示す．

2. 形式仕様への期待と課題

ソフトウェア開発の形式手法への期待は，従来の開発者の経験に基づく手法から脱却し，形式的理論に基づく機械的かつ（ある種の）網羅的な検証による，高信頼かつ高効率な開発に移行することである．

しかし，形式手法はその形式性が普及を阻害している側面もある．形式仕様に用いる仕様記述の難しさがしばしば問題として挙げられる．しかし，本質的な問題はそこではない．仕様記述とは一種のモデル，すなわち世界を一側面から切り取った断面（もしくは一側面への写像）にすぎない．他方，ソフトウェア開発とは，複雑に絡み合う多面的な要求を総合的に勘案し，最適解を見いだす行為である．この多面性の解決こそが，人力では扱いきれないソフトウェア開発の課題であり形式手法に期待するところであるが，一断面をモデル化して扱う形式手法の特徴とは相いれない．

換言すれば，ソフトウェア開発で直面する課題に対し，各課題を特徴づける断面を適切に扱うことで，ソフトウェア開発への形式手法導入が期待される．

3. Program Oriented Modeling(POM)

POM は，プログラムを解析し，ユーザの目的に適合したモデルを抽出するコンセプトであり，それを実現するツール基盤でもある．POM に基づくアプリケーションの 1 つに POM/MC[1] がある．POM/MC は，ソースコードを入力としモデル検査用のモデル (Promela コード) を抽出する．POM/MC は，テス

†1 株式会社日立製作所 横浜研究所
Yokohama Laboratory, Hitachi Ltd.

ト等で発見された不具合が再現できず修正できない課題に対して，不具合に到達するパスをモデル検査で発見する．一般に「未知の不具合を見つける」という目的でのモデル検査はハードルが高く，POM/MC では，既知の不具合状態に至るパスを見つけるという断面を定め，ソフトウェア開発におけるモデル検査の利用を可能とした．不具合の原因を特定するための「試行錯誤」を実現するため，変換ルールの構成変更により抽出するモデルを変更し，開発作業を支援する．

他の POM アプリケーションとして，POM/EQ[2] がある．POM/EQ はリファクタリングを推進するための等価性検証技術である．度重なる改変によって構造が劣化したソフトウェアではリファクタリングが求められる．しかし，稼働実績のあるソフトウェアへの変更に対する抵抗感は大きい．そこで，リファクタリングで変更した箇所について，リファクタリング前と同等のプログラムであることを保証できれば，リファクタリングが促進されると考えた．POM/EQ では，リファクタリングにおける等価性検証という断面を定めることで，等価性検証において検証すべき性質を，(1) リファクタリングによる構造の置き換えが正しく行われているか，(2) 入力に対する出力が同等であるか，の 2 点に絞った．そこで，POM/EQ では，リファクタリングパターンに対する構造的置換性に基づく等価性検証と，入出力関係に対する記号実行に基づく等価性検証との組み合わせで，ソースコードの等価性を検証する．逆に言えば，時間に関する特性や，並行性に関する特性は，本等価性検証においては扱わない．

なお，POM/MC と POM/EQ の想定対象は，C/C++言語で書かれた組み込みソフトウェアである．

4. エンタプライズシステム向け等価性検証

POM/EQ の更なる応用として，エンタプライズシステム向け等価性検証技術を検討している．エンタプライズシステムでは，現行システムを新しい実行環境に移すなどのリプレースがある．現行システムは長期にわたる運用・保守がなされ，正確な仕様書が存在するとは限らない．また，リプレースの際には，実行環境だけでなくビジネスフローの変更が行われる場合がある．一方，業務の根幹となるロジック（例えば支払額計算）は，現行と同等でなければならない．

リプレース時のロジック等価性という断面では，リファクタリング時と異なりリファクタリングパターンが存在せず，構造変換に基づく等価性検証方法は利用しにくい．一方，計算ロジックの等価性という点では，記号実行による入出力関係の等価性検証が適する．

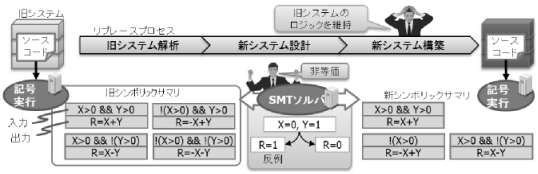

図 1 ロジック等価性検証
Fig. 1 Logic Equivalence Checker

我々は記号実行を用いたロジック等価性検証技術を開発している (**図 1**)．等価性検証の手順概要を示す．

(1) 新旧のソースコードから，計算ロジック部分を抽出した上で，それぞれを記号実行する．記号実行により，それぞれのシンボリックサマリが得られる．
(2) 記号実行結果の相違を示す論理式を生成する．
(3) 論理式の充足性を SMT ソルバで検査する．
(4) 論理式が充足されないとき，新旧ロジックは等価と判定する．論理式が充足されるとき，新旧ロジックで出力が異なる例が得られる．

Java 言語を対象とし，記号実行には JPF[⋆1] を，SMT-LIB2[⋆2] 形式の論理式の充足可能性検査には，SMT ソルバ STP[⋆3] を用いている．

5. ま と め

形式手法をソフトウェア開発に導入する上での課題を断面という観点から考察した．多断面からの課題を解決するソフトウェア開発と，特定断面をモデル化する形式手法の相違が形式手法実用化の課題の 1 つと考えた．ソフトウェア開発の各場面で着目する断面ごとに適した形式手法の適用方法を提供すべきと考える．これに基づき，POM によるモデル検査ならびに記号実行と SMT ソルバを用いた等価性検証の例を示した．

参 考 文 献

1) Ichii, M., Myojin, T., Nakagawa, Y., Chikahisa, M. and Ogawa, H.: A Rule-based Automated Approach for Extracting Models from Source Code, *Proc. of Working Conf. on Reverse Engineering (WCRE)*, IEEE, pp.308–317 (2012).
2) 市井　誠，新原敦介，鈴木康文，小川秀人：リファクタリング支援のためのプログラム等価性検証手法の提案，ソフトウェア工学の基礎 XX(FOSE 2013)，pp.303–304 (2013).

⋆1 http://javapathfinder.sourceforge.net/
⋆2 http://smtlib.cs.uiowa.edu/papers/smt-lib-reference-v2.0-r12.09.09.pdf
⋆3 https://sites.google.com/site/stpfastprover/