論文の内容の要旨

高信頼性ソフトウェアシステムの実現のためのモデル修正技術に関する研究

熊澤 努

近年，様々なソフトウェアシステムが提供されるようになり，システムに欠陥や障害のない信頼性の高いシステムを開発することが求められている．しかしながら，現在のソフトウェアシステムは，大規模化だけでなく開放化，分散化も進んでおり，顧客からの多様な要求に応えつつシステムの信頼性を確保することは容易ではない．実際，不具合を含むシステムによって社会的に深刻な影響を与える事例が数多く報告されている．

この背景の下でシステムの品質の向上を図る開発手法がこれまで数多く提案されてきた．その一つが，標準的なシステム開発手順を定めた開発プロセスである．中でも落水型開発プロセスは，システムの開発運用過程を，要求分析，設計，実装，テスト，運用および保守の各工程に分解し，順に実施することで信頼性を確保することを目指すプロセスで，多くの開発プロジェクトで採用されている．

もう一つは，要求分析，設計工程でのモデルと呼ばれるシステムを抽象化した記述の活用である．モデルは，システムの構造を表す静的なモデルと，システムの振る舞いを表す動的なモデルに大きく分類される．状態遷移機械に代表される動的なモデルは，要求分析，設計工程において広く用いられる．特に，複数の状態遷移プロセスが相互作用しながら計算を行う並行システムは，しばしばシステム全体の振る舞いが複雑となるため，動的なモデルによるシステムの分析が必要である．

落水型プロセスは工程間で手戻りが発生しないことが前提なので，要求分析，設計工程で作成したモデルが所望の性質や仕様を満たしていなければ，それらを元に開発したシステムもまた誤りを含む．このような誤りはシステムの運用時に重大な不具合を引き起こすことがあるので，システムの信頼性を損なう要因となり，その結果，開発の手戻りを引き起こすこととなる．そのため，システム開発の要求分析，設計工程で，モデルの正しさを保証することがシステムの信頼性を確保するために必要である．

こうした状況に対して，システムに要求される性質を動的なモデルが満たすかどうかを形式的，自動的に検証するモデル検査技術やモデル検査器が広く研究されている．モデル検査器は，モデルが性質を満たさない場

合には，性質違反を示すモデル上の事象列（軌跡）である反例を提示する．この場合，モデルに誤りがあると考えられるので，正しいモデルを得るためには，開発者は性質を満たすようにモデルを修正する必要がある．しかし，反例はモデルの誤りやそれを正す方法を直接的には提供しないので，開発者は反例を手作業で分析して誤りを特定し，修正を行わなければならない．対象システムの振る舞いが複雑な場合には，この作業は開発者にとって大きな負担となり，信頼性の高いソフトウェアを開発する際の障害となる．

そこで，本論文においてわれわれはモデル検査に基づくモデル修正技術を提案し，高信頼性システムを実現する開発技術の確立に貢献する．本論文では，要求分析，設計工程でモデル検査が用いられる状況を想定し，動的なモデルの一種のラベル付き遷移系（LTS）によりシステムの振る舞いが関連する事象に基づいて記述されると仮定する．

本論文の 1 章では，以上のソフトウェア開発をめぐる背景と解決すべき課題について論じる．

2 章では，本論文で前提とするモデル検査技術を説明する．LTS に対しては，流動線形時相論理（FLTL）で記述された性質のモデル検査技術が提案されている．これは，FLTL 式が表す軌跡の集合を受理する Büchi オートマトンを自動的に構築できるという事実に基づく方法である．FLTL はシステムの振る舞いについての性質の記述に有用な論理体系である．FLTL 式で表される性質は一般に，システムにとって望ましくない事象が決して起こらないことを表す安全性と，望ましいことがいつか起こることを表す活性に分類される．性質を満たさない場合には，モデル検査器は，活性に対しては無限回反復をする閉路とそこへの有限長の接頭辞からなる無限長の軌跡を，安全性に対しては有限長の軌跡を反例として出力する．

3 章から 5 章で，提案するモデル修正技術を説明する．先行研究の多くは安全性のみを対象としているが，提案手法は安全性と活性の両者に適用できる．加えて，対象モデルにおいて公平性を仮定するモデル検査に対しても適用可能である．

3 章で提案する反復型モデル修正法を述べる．この手法は，安全性の集合で表現される対象領域に関する知識を用いて開発者に修正モデルの候補を提示する．まず，領域知識と反例をモデルに変換して，両者の合併モデルを求める．各遷移に対してその実行確度を追加した LTS の拡張記法で記述することで，合併モデルは領域知識を満たす軌跡を生起しうる遷移で表現し，そこに現れる反例を生起が禁止された遷移で表現する．その結果，合併モデルは，領域知識から反例を除いた軌跡の集合を表す修正モデルの候補と解釈される．そこで，開発者は，各遷移の実行確度を手掛かりに，システムに必要な遷移を選択して修正作業を実施する．最後に，修正モデルの検証を行い，モデルが性質を満たさなければ，新たに得られた反例を用いて上記の工程を繰返す．

本手法は既存のモデル検査技術を直接利用できる方法である．また，本手法のモデルの合併操作は自動化が可能であり，計算機による修正作業の支援ができる．本手法では，修正時に対象領域の知識と開発者による決定工程を導入することで，対象領域や開発者にとって無意味なモデルを生成する恐れが少ない．最後に，対象システムの正しい振る舞いが既知であることが多くの先行研究の前提だが，本手法は対象領域の知識を用いるので，この制約がない．

われわれは事例研究によって提案手法の有効性を確認した．

次に，モデル修正作業の効率化のために，提案手法の改善について議論する．これは，修正作業が修正対象モデルを元に行われるので，合併モデルや修正モデルは修正対象モデルと類似する，という考えに基づく．われわれは振る舞いに基づくモデル間の類似度を定義して，以下のように活用する．第一に，対象モデルと合併モデルとの類似度を算出することで，両者の関係を開発者に提示する．第二に，修正の結果得られるモデルと修正対象モデルとの類似度を算出し，修正結果の妥当性を測定する指標を開発者に提供する．

しかしながら，開発したモデル修正法において，反例に含まれる性質違反の原因は開発者が発見しなければならない．そこで，開発者によるモデル修正作業を支援するために，4 章でモデルの誤りを特定する方法であ

る LLL-F を提案する．LLL-F は，対象の性質を満たす軌跡（正例）のうち，反例との編集距離の近いものを抽出し，反例との差分を求める．この差分が，反例が正例から逸脱するモデル上の誤りの候補である．ただし，無限長の反例の編集距離を扱うことは困難なので，LLL-F は，有限長文字列の編集距離を扱うために反例を接頭辞と閉路に分割し，それぞれに対して編集操作を適用した軌跡の集合を表すモデルを構成する．これらを，正例の集合である性質の Büchi オートマトンとの積を求めることで，正例を探索する問題を有向グラフ上の最短路探索問題に帰着させる．

LLL-F は，先行研究が用いる SAT 求解器のような技術を前提とせず，既存のモデル検査器の出力を直接用いる簡便な方法である．加えて，先行研究の多くは，対象プログラムに性質や仕様を満たす実行列が存在することが前提である．よって，そのような実行列が存在しない場合には，従来手法により誤りを特定することが困難であるが，LLL-F では，性質を満たす軌跡の集合が Büchi オートマトンにより与えられるので，この制約がない．さらに，正例というモデルを修正する際の手掛かりとなる情報を開発者に提示できる．

われわれは LLL-F を自動化したプロトタイプツールを Java 言語で実装し，これを用いた事例研究を行った．7 つの事例に対して LLL-F を実行した結果，6 つの事例でモデルの誤りを正しく同定した．次に，正例の探索空間の規模を変化させたときの LLL-F の実行時間の変化を調べ，LLL-F は実用的な実行性能を持つことを確認した．

しかしながら，LLL-F は軌跡間の類似性を明確に定義していないので，反例の誤りを適切に修正した正例を生成できない場合がある．5 章ではこの点について LLL-F を精緻化した誤り特定手法として，LLL-S を提案する．LLL-S は，有限長の文字列間の編集距離を拡張した無限長の軌跡間の距離を導入する．そして，この距離が反例に対して最小の正例を有向グラフ上の最短路探索問題を解くことによって求める．反例と類似した正例を求める既存手法の多くは，有限長の反例と正例のみを扱うので，その類似度もまた有限長のプログラム実行列や事象列を前提とする．よって，無限長軌跡間の距離の計算には適用できない．

われわれは，LLL-S のプロトタイプツールを Java 言語で実装し，事例研究を LLL-F の場合と同様に実施した．その結果，7 つの事例のうち 6 つについて誤りを的確に発見した．また，残り 1 つの事例に関しても，反例の形状を手動で調整することで誤りを特定した．次に，正例の探索空間の規模を変化させたときの LLL-S の実行時間の変化を調べ，LLL-S が実用的な実行性能を持つことを確認した．

6 章において本論文で議論したことをまとめる．特に，提案手法を統合した統一的なモデル修正技術の確立に注目して議論する．加えて，統一的な方法論の確立のために残された研究課題について述べる．