モデルベース開発における大規模言語モデルを用いた自動プログラム修復

Automatic Program Repair Using Large Language Models in Model-Based Development

システム理工学専攻　　BP21005　安喰　蓮
ソフトウェア工学研究室　　指導教員　久住　憲嗣

１．研究背景

モデルベース開発とはシステムの動作や構造を表現するモデルを作成し、そのモデルを用いて設計、検証、実装を行う開発手法であり自動車産業や航空宇宙産業などの分野で活用されている。

現在のソフトウェア開発ではバグの修正や開発されたシステムのメンテナンスに対して多大な費用や人的資源などのコストを割いている。これらを自動化して行う技術を自動プログラム修復 (Automatic Program Repair; APR) と言う。これらの作業を自動化することでコストの削減につながると期待されている。コードベース開発と同様にモデルベース開発においてもバグは発生し、その修正にコストがかかる。

また近年大規模言語モデル (Large Language Models; LLM) を使用して APR を行う研究がいくつか行われている。プログラムコードを対象とした APR は様々な手法がある一方でモデルベース開発における APR の研究は分野によってはあまり研究が進んでいるとは言えない。

そこで本研究ではモデルベース開発における LLM を使用した APR のシステムを提案する。

２．方法

本研究ではモデルベース開発における LLM を用いた APR 手法をワークフローとして提案する。

初めにワークフローの適応前にモデルを LLM が処理可能な形式に変換する必要がある。そのためには先行研究 [1] に基づいてモデルの構成要素を解析しテキストデータとして抽出した。IF 関数に該当するような関数ブロックなどは条件式に該当する情報などが必要となるためブロックごとに特殊な対応が求められる。LLM が認識しやすいように構造を持ったテキストである JSON 形式での出力を行った。今後の修正結果等は JSON 形式での出力を行ったものである。

先行研究 [2] が示す通り APR プロセスはモデルの状態異常の検出やバグの特定を行う Fault localization(FL)、収集されたバグの情報からバグを修復するコードを作成する Patch Generation(PG)、作成されたコードが正しいかを検証する Patch Validation(PV) で構成されると定義されている。本研究では FL と PG の 2 種類を行うワークフローの実装を行った。これは JSON 形式での出力のためモデルのシミュレーションの実行を行うことができず検証プロセスを行うことが出来ないためである。

設計したワークフローの Phase1、2 では FL プロセスに該当する部分となっており、Phase1 では LLM を使用せずに機械的なモデルの状態異常の検出を行っている。当初 LLM のみでバグの検出を行っていたが十分な精度を得ることが出来なかった。バグの検出は本提案手法の基盤となる部分であり安定性を重視するために機械的なモデルの状態異常の検出を採用した。ただしこの段階でどのようなバグがあるかを検出するのではなくあくまでモデルがどのような状態であるかのみ (ex, 接続の切断など) を検出している。

Phase2 では Phase1 で検知したモデルの状態からどのようなバグがあるかを判断する。1 つだけ判断するのではなく 3 つ候補を生成しそれぞれ優先度をつけた出力を行うようにした。これは単一のバグでない想定や回答ごとに LLM の挙動のばらつきを抑えることで安定したシステムの構築のためである。

Phase3、Phase4 では PG に該当するプロセスと

なっており、Phase3では診断したバグの状態から修正案を生成することを行っている。修正案の修正シナリオを生成しPhase4にて使用する機械的修正プロセスに合致した修正案を生成している。

Phase4では機械的な修正プロセスを適用する形で修正案の適用を行う。LLMによるばらつきを防ぐために型にはめる機械的な修正プロセスを行っている。

３．結果

提案手法の有効性の検証のためワークフローに基づいた実装を行った。実験に使用したLLMモデルはGemini2.5-flashを利用した。実験対象は代表的なモデルベース開発環境であるSimulinkを使用し、開発元であるMathworks社の公式モデルにバグを挿入する形で実験を行った。修正の成否は差分検出を活用することで成否を診断している。バグ注入前と同一をS3、一部余分な修正をS2、一部のみ修正をS1、失敗をS0とした。成功率(Succ)はS2以上を対象として扱った。

４．考察

全体的にFL RATE(バグの発見率)は高く出ている。これはPhase1の機械的手法によるモデルの状態異常の検出がLLMの診断に好影響を与えていると推測できる。

パラメータ変更系のバグに関しては成功率が比較的高く出ているものの構造の変更のバグに関しては簡単なバグ以外では成功率が高く出ているとは言えない。またJSON形式での出力のため実際の実行環境での実装を行った際にエラーが出力される可能性や独特の挙動を行う可能性がある。isolated block(孤立ブロック)の削除に対する課題に関してLLMが当初ブロックの削除を最優先としたが再考時に仮想的な使用シナリオを生成し始めたケースがあった。このFaultはほかのブロックが正しく接続されているかを確認せずにLLMが自己生成したシナリオに惑わされたケースがあった。このケースはLLMの推論に対して削除という行動に何らかのバイアスがかかっているのではないかと考察する。全タスクに対してワークフロー全体の調整が必要であると考察する。

Fault Type	Total	FL Rate	Repair Scores	Succ.
data_type_change	25	100.0%	S3:20 S0:5	80.0%
delete_random_block	25	40.0%	S0:25	0.0%
disconnect	25	92.0%	S3:15 S0:10	60.0%
feedback_loop_injection	25	100.0%	S3:5 S2:13 S0:7	72.0%
line_branch_removal	25	96.0%	S3:8 S0:17	32.0%
multi_disconnect	25	100.0%	S3:5 S1:10 S0:10	20.0%
isolated_block	25	100.0%	S3:3 S0:22	12.0%
parameter_mutation	25	80.0%	S3:10 S2:5 S0:10	60.0%
sample_time_corruption	25	100.0%	S3:18 S2:6 S0:1	96.0%
signal_dimension_mutation	25	80.0%	S3:2 S2:15 S0:8	68.0%
OVERALL	250	88.8%	S3:86 S2:39 S1:10 S0:115	50.0%

表：実験結果

５．結論

本研究では、Simulinkを対象としたモデルベース開発における自動プログラム修復（Automatic Program Repair, APR）の実現を目的に、大規模言語モデル（Large Language Model, LLM）を活用したワークフローを提案し、実験を通じてその有効性を検証した。ワークフローを通じた実験結果ではタスクごとに成功率にばらつきがある点は今後の課題である。実際にSimulinkモデルを改善したパターンでの挙動を確認するといったケースやModel Context Protocol (MCP)を使用することでLLMネイティブなシステムを作成することで従来のJSON形式等を使用した場合と比較してさらなる精度向上や実行可能形式を整えることが目標である。本手法の発展によってモデルベース開発におけるAPRの手法の一つとして本手法であるLLMを用いたAPRが採用され、モデルベース開発における開発の効率化と精度向上に貢献できると考える。

参考文献

[1] Y. Funaguchi and K. Hisazumi. Generating bug fix proposals for simulink models using large language models. In *Proceedings of Asia Pacific Conference on Robot IoT System Development and Platform*, 2024.

[2] Yanming Yang, Xin Xia, David Lo, and John Grundy. A survey on deep learning for software engineering. *ACM Computing Surveys*, 54(10s), 2022.