通信制御ソフトウェア開発における
状態遷移設計の品質向上への取り組み
～ZIPC 状態遷移表のモデル検査適用への取り組み～

富士通株式会社　共通開発本部　ソフトウェア方式統括部
安岡　大知

1. はじめに

近年の通信ネットワーク(図 1)は、エンドユーザに提供するサービスの多様化、キャリア間の早期サービス提供競争が激化している。

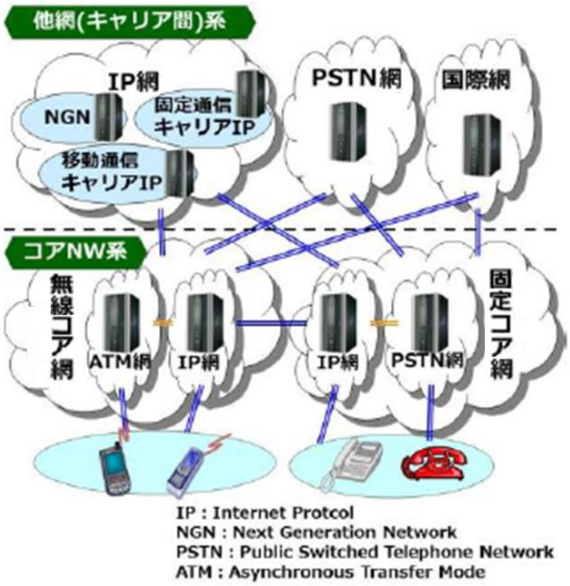

図１：通信ネットワークイメージ

その通信ネットワークを構築する通信制御ソフトウェア開発には厳しい条件が要求され続けている(図 2)。

図２：近年のキャリアからの要求条件

こういった現状に対し、通信制御ソフトウェア開発では、実績ある既存資産を流用し、機能追加・変更を行うことで対応している(図 3)。

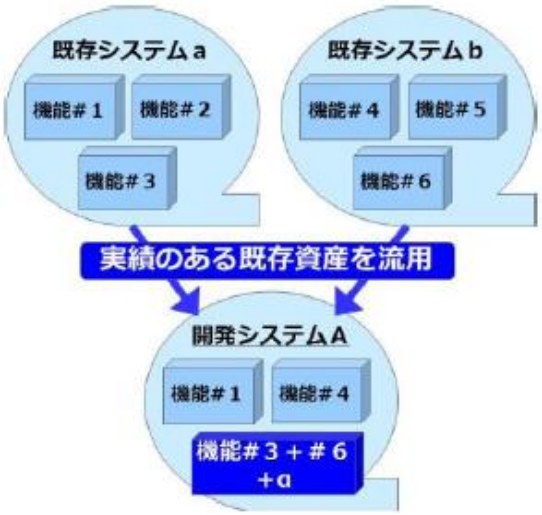

図３：既存資産流用イメージ

このような流用開発を主体とする中、より短期間で高品質なソフトウェアの提供を可能とする開発プロセス構築が必要であり、我々は以前から ZIPC を使って作成した状態遷移表に形式手法モデル検査を適用し、設計工程で状態遷移表を網羅的に検査し、品質を確保する取り組みを行ってきた。本稿では、「形式手法モデル検査ツール SPIN : Simple Promela Interpreter(以下、SPIN)」をプロジェクト適用する上での課題と解決への取り組みについてご紹介する。

2. 通信制御ソフトウェアの開発手法

通信制御ソフトウェアでは、複数の通信プロトコルの信号受信(イベント)を契機に状態を変化させ、状態毎に処理決定する「状態遷移設計」が重要である。

状態遷移設計において実現する状態制御を(図 4)に示す。

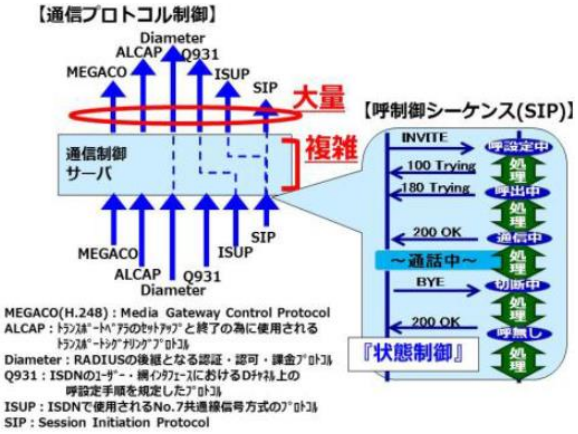

図 4：状態制御

「状態制御」は、多くの通信プロトコルを扱う、また、1 つの通信プロトコルを他の複数のプロトコルに変換するなど、「大量」、「複雑」である。

これを特徴とした通信制御ソフトウェアにおける状態遷移設計では、状態遷移表を用いて状態制御の網羅性担保を行ってきた。

3. 状態遷移設計における課題

状態遷移表は状態制御の網羅性を担保し易い反面、規模が大きくなりやすい。実際に我々が作成してきた状態遷移表は大きいものになると状態とイベントの組み合わせが数万以上となる。

この大規模な状態遷移表を設計工程で多くの時間をかけて新規作成、派生開発による追加・変更をしているが、(図 5) に示すように前状態に戻る遷移ルートを含めた確認ケースは膨大であるため、状態遷移表の検証(レビュー/試験)では確認対象の絞込み(精査)を行った結果、検証項目を漏らすことにより問題流出し、品質安定までに多くの時間を要していた。

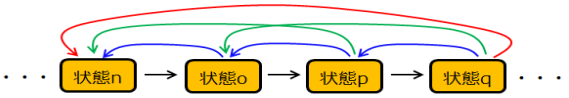

図 5：状態遷移ルートバリエーション

そこで、上流工程での品質確保による早期品質安定を目的とて形式手法モデル検査ツール SPIN のプロジェクト適用に取り組むこととした。

4. 形式手法モデル検査ツール SPIN

モデル検査はシステムから導出されたモデルが形式仕様を満足するかどうかをアルゴリズム的に検証する手法である。

モデル検査にも多くのツールが存在するが、我々は、状態遷移表との親和性や安定性、完成度の高さから「SPIN」を選定した。

SPIN では、状態遷移表について線形時相論理式(以下、LTL 式)による検査が可能である。LTL 式による検査とは、与えた条件が常に、あるいはいつか必ず真となるといった時間軸上で命題の真偽を記述できる論理を使用し、矛盾がないかどうかを確認することで状態遷移表の不備を検出することである。

5. SPIN プロジェクト適用の課題

SPIN による状態遷移表の検査作業と、各作業における課題を(図 6)に示す。

SPIN 検査では、大きく以下の作業が必要となる。

a)仕様から状態遷移表を作成し、その状態遷移表が実現する動作モデルを SPIN が解釈可能な Promela 言語へ変換する。

b)仕様から状態遷移表の検査観点を抽出し、検査観点から LTL 式を作成する。

c)作成した Promela 言語と LTL 式を SPIN に入力し、SPIN を実行する。

d)実行の結果、動作モデルが検査観点を満たさない場合、反例が出力され、反例を元に状態遷移ルートを解析し、不具合があれば状態遷移表にフィードバックする。

これら作業をプロジェクトで実行するうえで、以下の課題に直面した。

①Promela 言語/LTL 式/反例解析といった専門スキルの修得や SPIN の環境構築/操作方法の修得

②膨大な遷移ルート検査において検査不能となる状態爆発

③SPIN 適用における費用対効果が不明確

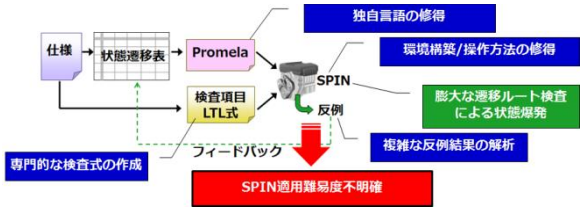

図 6：SPIN 検査作業と課題

これら課題について、①は「SPIN 検査支援ツールの作成」(6 章)、②は「状態爆発の防止策確立」(7 章)、③は「費用対効果の事前把握」(8 章)により、解決に取り組んだ。

6. SPIN 検査支援ツール

SPIN のプロジェクト適用において、専門スキルやノウハウの修得不要とする検査手順を確立することを目的として、SPIN 検査支援ツール(図 7)を作成した。以下に検査支援ツールの概要を示す。

●Promela 言語の自動生成

各プロジェクトの状態遷移表は、共通の記述ルールがないため、記述の仕方がそれぞれ異なり、Promela 言語への変換および SPIN 適用の阻害要因となっていた。そこで、自動生成のための処理簡素化として、記述レベル統一を目的として、ZIPC の状態遷移表を導入した。これにより、ZIPC による定型化した状態遷移表から Promela 言語の自動生成は現実的な工数で対応可能となり、Promela 言語のスキル修得を不要とした。

●LTL 式の自動生成

各種状態遷移処理や、変数の操作について LTL 式の雛形を作成した。具体的な状態定義や変数等のデータはプロジェクト毎に異なるため、検査時にはプロジェクト個々のデータ定義を入力することで、LTL 式の雛形にデータ定義が補完され、LTL式の自動生成を実現し、LTL 式のスキル修得を不要とした。

●反例解析支援

SPIN が出力する反例ログについて状態/イベントをインデックスにしてログを整理して CSV ファイルに出力することで、可読性が向上した。

●モデル検査 Web システム

SPIN によるモデル検査を開発プロセスに導入する仕組みとしてモデル検査 Web システムを構築した。

モデル検査 Web システムによって、ZIPC 状態遷移表とデータ定義を入力すると SPIN 検査実行から反例結果出力まで自動で行われ、Web アクセスにより、誰でも・いつでも簡単に検査が可能となった。

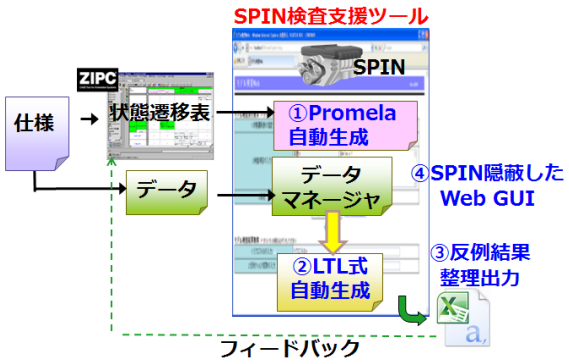

図 7：SPIN 検査支援ツール

7. 状態爆発の防止

状態爆発とは、例えば状態遷移表の検査において、探索すべき状態数が膨大な場合、コンピュータの記憶容量の限界によって検査不能となることである。

細かい粒度の状態遷移表をモデル検査すると状態爆発により検査不能となるため、対策として状態遷移表の抽象化によって、粗い粒度の状態遷移表をモデル検査することが必要となる。

しかし、我々の SPIN 適用事例では、抽象化によって状態爆発は回避されるが、肝心の状態遷移表の問題(反例)が見つからない傾向にあった。

そこで、SPIN をプロジェクト適用する上で、状態遷移表の問題検出するための抽象化手順を検討するにあたって、モデル検査ターゲットの決定に取り組んだ。

7. 1. モデル検査ターゲットの決定

モデル検査ターゲット決定においては、各種通信制御ソフトウェアの状態遷移設計に関する流出問題分析を行った結果、状態切替時のデータ操作誤り、特にデータの二重設定や二重解放、未設定データの参照といった問題が問題原因の大半であることが分かった。

このことから、我々は状態遷移表の記載すべてを検査するのではなく、特にデータ操作にターゲットを絞ることにした。これにより、状態遷移表の記載内容・粒度が定まり、状態爆発の防止と状態遷移表の問題検出の両立が可能となった。

7. 2. 状態爆発の防止例

モデル検査ターゲットの決定によりデータ操作に着目したモデル検査においても状態爆発は完全に回避することは出来ない。そこで、状態爆発発生を防止する解決策の一例を以下にご紹介する。

●状態遷移表の分割

データ操作を検査するにあたって、1 つのシステムの全てのデータ操作を 1 度に検査するのではなく、状態遷移表をノードや機能ブロック等に分割して記述し、検査することで、状態爆発が回避出来る場合がある。

例えば、(図 8)に示すように、1 システムの全てのデータ操作を 1 回で検査するのではなく、ノード単位で 2 回、あるいは、機能ブロック単位で 4 回、コンポーネント単位で 8 回に分けて検査する。

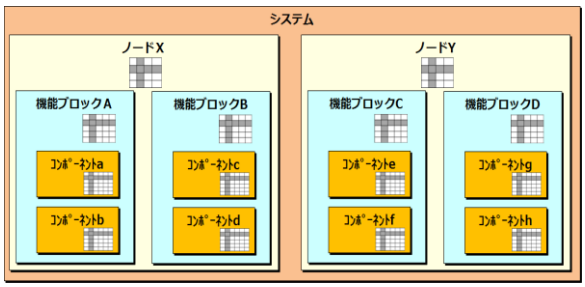

図 8：状態遷移表の分割

●データ種別による分割

状態遷移表を分割しても状態爆発が回避出来ない場合、分割した状態遷移表をデータ種別毎に記述し、データ種別毎に記載した状態遷移表を検査することで、状態爆発が回避出来る場合がある。

例えば、(図 9)に示すように、1 つの状態遷移表のデータ操作を「タイマ」、「リソース」、「受信イベント情報」に分類し、分類したデータ毎に状態遷移表を記述し、検査することで、状態爆発が回避出来る場合がある。本例では、1 つの状態遷移表の全てのデータ操作を 1 回で検査するのではなく、「タイマ」、「リソース」、「受信イベント情報」の 3 回に分けて検査する。

図 9：データ操作の分割イメージ

8. 費用対効果の事前把握

SPIN を初めて適用するにあたって、適用により発生する工数を還元できる効果が得られるかは大きな関心事である。

そこで、我々は状態遷移表の各種パラメータとモデル検査サーバの処理性能、過去の SPIN 検査実績を元に、検査対象の状態遷移表の複雑度を点数化するツールを作成した。この点数にて、SPIN 適用時の費用対効果を定量的に判断可能とした。判断例を(表 1)に示す。ツールによる点数計算例を(図 10)に示す。

点数	適用難易度	状態爆発対策	導入コスト	導入効果
30点以下	低	不要	低	低
31点～65点	中	不要	中	有
66点以上	高	必須	高	有

表 1：費用対効果判断例

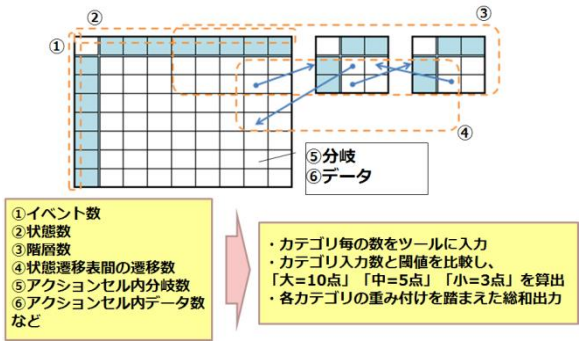

図 10：複雑度点数計算例

9. プロジェクト適用事例

あるプロジェクトでは、用件からデータ操作としてタイマ設定/停止の妥当性確認に SPIN を活用して検証を行った。この適用結果を(図 11)に示す。SPIN を適用しなかった場合、設計工程で問題が潜在し、下流工程以降に流出していたが、これを未然に取り除くことが出来、設計工程での SPIN 検査適用の有効性を確認することが出来た。

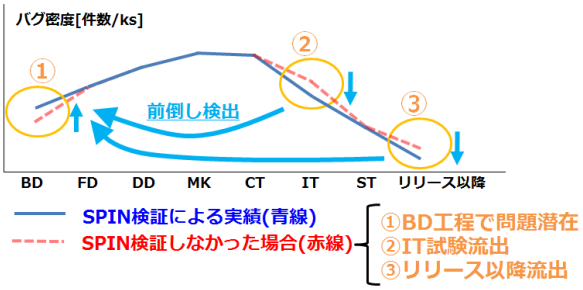

図 11：SPIN 適用効果

10. まとめ

モデル検査を適用するにあたっては専門スキルを不要とする手順確立だけでは不十分で、状態爆発への対策が必要と考える。

状態爆発は、単純に状態遷移表を抽象化するのではなく、明確なモデル検査ターゲットを元に実施することが望ましいと考える。

これらをクリア出来た場合、モデル検査適用効果(品質向上/コスト削減)は高いと考える。

一度モデル検査の仕組みを作ると繰り返して活用可能であるため、今後はリグレッション試験やアジャイル開発に用途を拡大する予定である。

参考資料

[１] 富士通株式会社　松下亮太郎、ZIPC WATCHERS Vol.15 通信制御ソフトウェア開発における状態遷移表の実装効率化への取組み

[２] 富士通株式会社　三橋崇、ZIPC WATCHERS Vol.15 通信制御ソフトウェア開発における状態遷移表の実装効率化への取組み ～ ZIPC によるコード生成自動化への取組み ～