ソフトウェアの実行前検査に関する研究

M2006MM001 安孫子 正康
指導教員　野呂 昌満

1　はじめに

並行ソフトウェアは資源を有効活用し，複雑なシステムであっても並行処理実体単位で設計できるという利点があるが，実用性には問題がある．並行処理実体間の振る舞い設計を誤ると，予期しない順序で処理が行われ，デッドロックやスタベーションを起こす可能性がある．プログラム上から設計の欠陥箇所を特定するのは困難なので，実行前検査で設計の欠陥箇所を特定し，再設計することが，並行ソフトウェアの開発に有効である．

本研究室では，組込みソフトウェアにおけるアスペクト指向アーキテクチャスタイル (以下，E-AoSAS++) を提案している．E-AoSAS++ では，ソフトウェアを並行に動作する状態機械の集合 (以下，CSTM) として定義する．E-AoSAS++ に基づくソフトウェアは，並行ソフトウェアといえるので，実行前検査により，設計の欠陥箇所を特定することで開発の支援ができる．

実行前検査を行なうためには，ソフトウェア設計の記述を，数学を用いて検証できる記述形式にする必要がある．ペトリネット，プロセス代数の CSP[1]，Promela の記述を，機械的に検証する検査ツールがあるが，ソフトウェア開発者には馴染みがない．ソフトウェア開発は一般に UML で設計が行われているので，検査ツールを利用するために，検査ツールが検証できる記述へ変換する．検査ツールは，検証に用いた記述の欠陥箇所を結果として返すので，設計の欠陥箇所を直接知ることはできない．設計の欠陥箇所を特定するためには，検査結果と設計との対応関係を調べる必要がある．ソフトウェア開発で実行前検査を行うためには，設計と検査について熟知する必要がある．

本研究の目的は並行ソフトウェア開発のための実行前検査支援ツールを設計・実現し，実行前検査の実用化について考察することである．本研究では，E-AoSAS++ に基づくソフトウェア開発への実行前検査の導入を考える．検査支援ツールは UML に基づくソフトウェアの設計を，検査ツールが検証できる記述に変換し，検査で得られた結果を UML 図式で表す．UML 図式から検査ツールが検証できる記述への変換を考えることで，対応関係を明らかにする．対応関係から，UML を用いて設計を行なうソフトウェア開発での，実行前検査に適した検査ツールが分かる．検査ツールのインタフェースとなる支援ツールを考え，実現することで，UML による設計を検査し，設計の欠陥箇所を UML 図式上で知ることができる．

2　E-AoSAS++

本研究室で提案している E-AoSAS++ のコンポーネントと，E-AoSAS++ に基づき，ソフトウェア開発のプロセスについて述べる．

2.1　コンポーネント

E-AoSAS++ に基づくソフトウェア開発は，ソフトウェアを CSTM や複合 CSTM の協調動作で設計する．複合 CSTM はポリシと CSTM で構成される．ポリシは複合 CSTM の外部と内部の CSTM のメッセージ中継を行ない，内部の CSTM を集合として機能させる．内部 CSTM の活性状態と休眠状態を変更することで，動的な構成変更を実現する．**図**1 に複合 CSTM を示す．

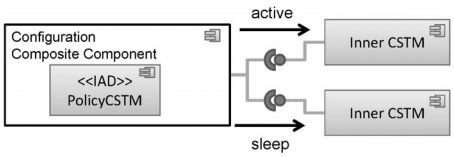

図 1　複合並行状態遷移機械

ポリシは内部の CSTM の構成を管理する．active または，sleep メッセージを CSTM に送ることで，活性状態と休眠状態を変更する．

2.2　E-AoSAS++ 開発プロセスと実行前検査

E-AoSAS++ に基づく開発は，プロダクトソフトウェアエンジニアリングの部品再利用の考えを適用した開発になる．開発手順を以下に示す．

1. ドメインとソフトウェア特性の選択
   開発するソフトウェアのドメインを決め，非機能特性や機能特性を選択する．
2. コンセプチュアルアーキテクチャ構築
   静的構造をオブジェクト指向設計し，システムの動作と共有資源を設計・検査する．
3. プロダクトアーキテクチャ構築
   静的構造をアスペクト指向設計し，システムの動作とポリシを設計・検査する．
4. プロダクトアーキテクチャに基づく設計
   全ての CSTM とシステムへの入力を設計・検査する．

設計について述べる．2 では，システムの動作をシーケンス図で設計し，共有資源を CSTM として設計する．CSTM は状態機械図とアクション時の動作をシーケンス図によって設計する．システムの動作と共有資源の設計の対応を検査する．3 では，システムの動作をポリシの挙動を含めシーケンス図で設計し，ポリシを CSTM

として設計する．システムの動作とポリシの設計の対応を検査する．4 では，システムを構成する全ての CSTM を設計し，システムへの入力をシーケンス図で設計する．システムへの入力と CSTM の設計の対応を検査する．

3 UML と検査言語との対応

本章では，UML 図式から検証可能な記述への変換を考えることで，UML 図式との対応関係を明らかにする．本研究では検査ツールが検証する記述形式を検査言語と呼び，検証でよく用いられるペトリネット，FDR，Promela への変換を考える．対応関係を基に，ソフトウェア開発での検査に適した検査ツールを発見する．

3.1 ペトリネット

状態機械図とペトリネット記述の対応を考える．ペトリネット記述はシステムをプレース，アークとトランジションの 3 つで表現する．**図**2 に，状態機械図と対応するペトリネット記述を示す．

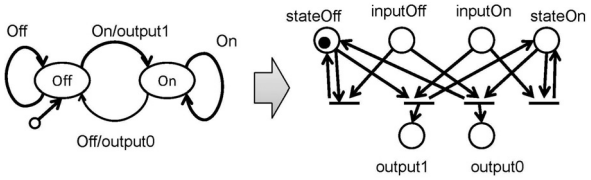

図 2　状態機械図とペトリネット記述

状態機械図に比べペトリネットによる記述は複雑になる．状態機械図からの変換を考えると，状態，イベントトリガとアクションは，ペトリネット記述ではどれもプレースとして表現される．検査結果でペトリネット上の欠陥が分かったとしても，状態機械図上の設計欠陥箇所を特定するのが困難である．

3.2 FDR

CSP に基づく検査ツール FDR の検査言語と状態機械図との対応を考える．**図**3 に状態機械図と対応する FDR 記述を示す．

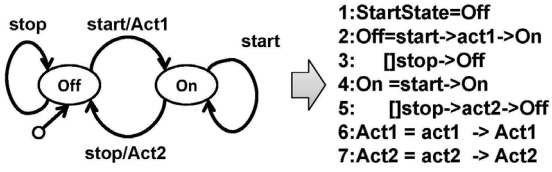

図 3　状態機械図と FDR 記述

表1 に状態機械図の要素と FDR 記述の対応を示す．状態機械図の要素と FDR 記述は 1 対 1 に対応づけることができる．**図**3 の状態機械図の Off 状態時に start イベントを受け取った際の遷移に対し，FDR 記述の 2 行目が対応する．Off プロセスは start イベントを観測し，アクションがあること示す act1 イベントを観測し，On プロセスにいたる．2 行目と 6 行目の act1 は同期しており，アクションで送信するメッセージは 6 行目の act1 の後に実現できる．

状態機械図の要素	FDR 記述
初期状態	StartState
イベントトリガ	start，stop
状態	On，Off
アクション	Act1，Act2

表 1　状態機械図の要素と FDR の要素の対応関係

シーケンス図と FDR 記述との対応関係を考える．**図**4 にシーケンス図と FDR 記述を示す．

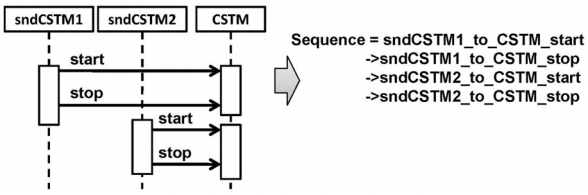

図 4　シーケンス図と FDR 記述

シーケンス図と FDR 記述では，シーケンス図のメッセージと FDR 記述のイベントが一対一に対応する．FDR 記述のイベント名を，sndCSTM1_to_CSTM_start のように，送信元と送信先の状態機械名とイベント名を組み合わせたものにする．CSP ではイベント名が同じ場合，同じ動作として扱うので，メッセージ送信元と送信先が異なるものは別のイベントとする．

3.3 Promela

検査ツール SPIN の専用言語 Promela と状態機械図との対応を考える．**図**5 に状態機械図と Promela 記述を示す．

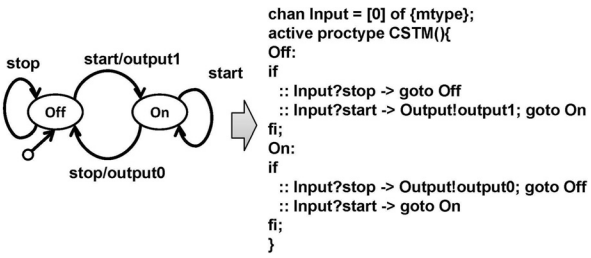

図 5　状態記述図と Promela 記述

Promela 記述と状態機械図の対応について述べる．Promela 記述は C 言語に近い構文であり，C 言語による状態機械の実現に近い記述が可能である．ラベルが状態を表し，if 文による条件分岐により，受け取ったイベントに応じたアクションでのメッセージ通信や，goto 文による遷移を表す．

Promela 記述はイベントを型として扱うので，シーケンス図との完全な対応付けは困難である．シーケンス図

はオブジェクトからオブジェクトへのメッセージ通信がどのような順序で行われるかを表す．Promela 記述ではメッセージ通信は送り先を指定するだけで，送信元を明示しないので，対応付けが難しい．メッセージの送信元が固定であれば，シーケンス図と対応付けができる．**図**5 にシーケンス図と Promela 記述を示す．

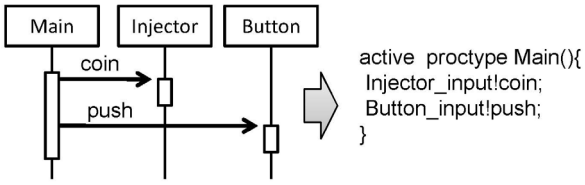

図 6　シーケンス図と Promela 記述

全てのメッセージの送信元が同一であるシーケンス設計と Promela 記述の対応付けができる．状態機械の状態遷移時のアクション設計や，システムへの入力設計をシーケンス図で表せば，Promela へ変換し，検査ができる．

3.4　検査言語の決定

実行前検査支援ツールは UML 図式との対応関係が明確な FDR と Promela の記述を扱うことにする．対応付けができた FDR と Promela の記述では，シーケンス図との対応付けに違いがあり，設計の異なった部分を検査できると考える．検査ツールである FDR と SPIN の双方の検査に対応した支援ツールを実現する．

4　実行前検査支援ツール

本章では，実行前検査支援ツールの入出力を考え，設計・実現を行なう．検査ツールから得られる情報を UML 図式に変換するだけでなく，設計の UML 情報を活用することを考える．設計の情報が活用できれば，設計の欠陥箇所を特定するための支援ができると考える．

4.1　実行前検査支援ツール概要

実行前検査支援ツールの概要について述べる．**図**7 に実行前検査支援ツールへの入出力を示す．

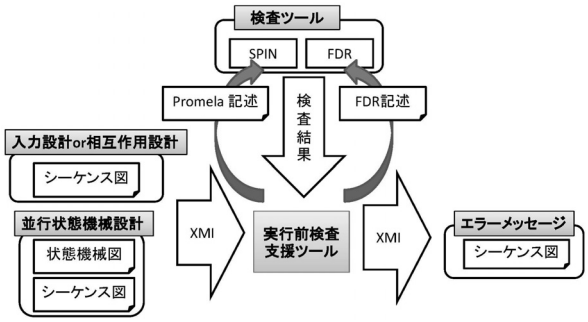

図 7　実行前検査支援ツールの入出力

実行前検査支援ツールの入出力を整理をした．入出力を XMI 形式で行なうことで，UML エディタで設計し，検査結果を確認できる．入出力を決めたので，支援ツールの内部を設計ができる．支援ツールの設計では，エラーメッセージ以外の情報による，設計の欠陥箇所特定の支援を考慮する．

4.2　実行前検査支援ツール設計・実現

UML 図式の検査言語への変換と，検査結果を UML 図式に変換する実行前検査支援ツールを実現する．検査支援ツール内部でのデータの扱いついて整理し，設計の欠陥箇所特定のための支援をどのように行なうかを考える．**図**8 に実行前検査支援ツールの内部構造を示す．

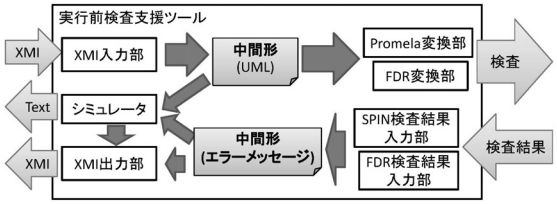

図 8　実行前検査支援ツール内部構造

設計の中間形と検査結果の中間形を生成し，目的とする形式に変換することで，支援ツールを実現する．設計の XMI から検査言語への変換と，検査結果を XMI に変換する必要があるので，変換のための中間形を生成する．SPIN と FDR 両方の検査ツールを利用したいので，中間形からそれぞれの検査言語に変換する．SPIN と FDR の検査結果は形式は違うが，システム停止までの動作順序を表すことは同じなので，同一の中間形で表現できると考える．検査結果は動作順序を表しているので，シーケンス図で表現できると考える．シミュレータでシステム停止時の情報を収集することで，設計の欠陥箇所特定のための情報が取得できると考える．

設計の情報を活用し，設計の欠陥箇所特定のための支援について考える．FDR は状態機械図とシステムのシーケンス図との対応を検査する．システムのシーケンス図上に検査結果を反映することで，設計の欠陥箇所特定の支援ができる．検査結果と設計の中間形を基に状態機械の動作をシミュレートし，デッドロックにより処理が完了できなかったアクションを調べ，アクション名をテキスト形式で出力する．アクションの動作と，検査結果を調べることで設計欠陥箇所の特定がきると考える．

FDR による検査結果を反映した設計の UML 図式について考える．**図**9 に FDR の検査結果と検査結果を反映したシーケンス図を示す．

FDR の検査結果とシーケンス図との対応，検査結果のシーケンス図への反映について述べる．FDR の検査結果 4 行目が，Main から Switch に対して on イベントを送るメッセージを表しており，UML のシーケンス図に対応付けできる．メッセージにステレオタイプを付加

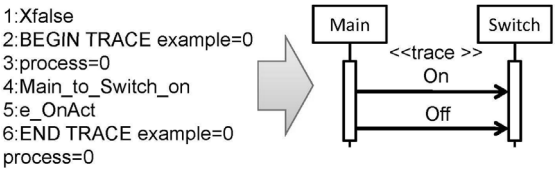

図 9　FDR 検査結果とシーケンス図

することで検査結果を反映する．ステレオタイプは多くの UML エディタで扱うことができるので，検査結果をシーケンス図へ反映し，停止までのメッセージと通常のメッセージを視覚的に区別できると考える．

5　考察

ソフトウェア開発に支援ツールを用いた場合の，実行前検査の実用性について以下の 2 点を考察する．

- 実行前検査の利用のしやすさ
- 設計の欠陥箇所発見のための情報支援

5.1　実行前検査の利用のしやすさ

ソフトウェア開発における実行前検査の実行前検査支援ツールの有効性を考察する．実現した実行前検査支援ツールを用いることで以下の利点がある．

- 検査ツールのための知識が不要
- 早期段階での検査が可能

前者は，UML 図式の知識があれば実行前検査ができるので，検査ツールに対する知識が必要なくなったといえる．ソフトウェア開発者は設計を UML で行ない，検査結果を UML 図式で知ることができる．

後者は，システムの設計が完了する前の段階で検査ができるので，早期段階で実行前検査ができるといえる．**図**10 に設計完了までのプロセスを示す．

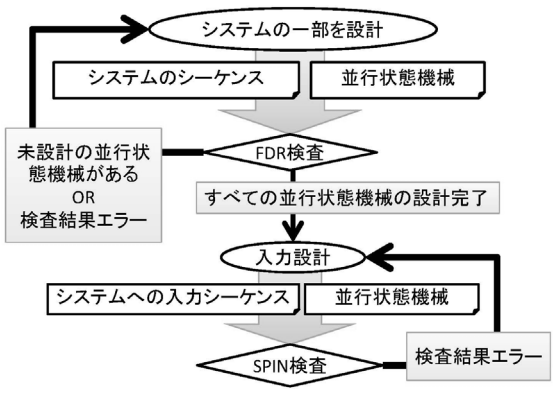

図 10　設計完了までのプロセス

FDR と SPIN の検査を使い分けることで, 早期段階での実行前検査を行う．FDR はシーケンス図と状態機械設計の検査に向いているので，システムのシーケンスと設計が完了した状態機械設計の対応の検査に使う．SPIN はシステムへの入力に対し，システムがデッドロックを起こさず動作することの検査に向いているので，入力設計と全ての状態機械設計との対応の検査に使う．CSTM の状態機械設計をひとつ終える度に FDR で検査をすれば，検査結果がエラーであっても，設計の欠陥に関係がある CSTM がすぐ分かる．

5.2　欠陥箇所発見のための情報支援

設計の欠陥箇所を発見するための支援情報の有効性を考察する．検査結果と設計の情報を基に各 CSTM の動作をシミュレートすることで，システム停止時における CSTM の情報を取得できる．システムの停止時にアクションが完了していない CSTM が存在していれば，その CSTM がデッドロックの原因に関係しているとわかる．完了していないアクションが実行される際の遷移がわかれば，設計の欠陥を修正する際の支援が行なえると考える．**図**11 に支援情報を付加した状態機械図を示す．

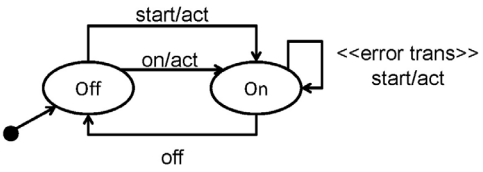

図 11　支援情報を付加した状態機械図

デッドロックに関係する CSTM の状態機械図の遷移にステレオタイプを付けることで問題のある遷移の可視化を行なう．同一アクションを複数の遷移で行う場合，アクション名をテキスト形式で出力するだけでは設計の欠陥箇所を特定するのが困難である．問題のある遷移を示すことで改善できる．状態機械の再設計を行なう際に，どの遷移が設計の欠陥に関係しているかがわかるので有効な支援であるといえる．

6　おわりに

本研究では，実行前検査支援ツールを実現し，E-AoSAS++ に基づくソフトウェア開発への実行前検査の実用性について考察した．UML 図式から検査言語への変換，得られた検査結果を UML 図式に変換する検査支援ツールを実現した．
今後の課題として，スタベーションの検査がある．

参考文献

[1] C. A. R. Hoare, “Communicating Sequential Processes.” *Communications of the ACM21(8)*, pp.666-677, 1978.
[2] Formal Systems (Europe) Ltd, http://www.fsel.com/, 2008.
[3] Spin - Formal Verification, http://spinroot.com/, 2008.