自動販売機制御ソフトウェアの再開発
～アスペクト指向を用いた組込みソフトウェアの実行前検査支援ツールの開発～

M2008MM020 宮田和季
指導教員：野呂昌満

1　はじめに

我々は OJL(On the Job Learning) として自動販売機制御ソフトウェアの再開発をおこなっている．再利用性の高い自動販売機制御ソフトウェアを実現するために，散在する関心事をアスペクトとしてモジュール化することで，ソフトウェアの再利用性の向上を目指している．

自動販売機制御ソフトウェアは組込みソフトウェアであり，並行ソフトウェアとして実現される．並行ソフトウェアの振る舞いを検証することは困難である．アスペクト指向ソフトウェアはアスペクト合成後の振る舞いを検証することが困難である．並行ソフトウェアの振る舞い，アスペクト合成後のソフトウェアの振る舞いを検証する手法として実行前検査技術がある．実行前検査は状態遷移モデルの動作パターンを網羅的に検査可能である [2, 7].

実行前検査では次の 2 つの問題がある．実行前検査で検出したフェーリア[1]からフォールトの特定が困難．大規模システムの実行前検査では，状態の組み合わせ爆発が発生し，検査が困難．この問題に対して，本研究室で提案している組込みソフトウェアのためのアスペクト指向ソフトウェアアーキテクチャスタイル (以後，E-AoSAS++)[4] に限定して，問題を解決する方法の提案をおこなう．

実行前検査で検出したフェーリアからフォールトの特定が困難であるという問題に対して，フォールトを検査する視点を整理し，段階的にフォールトを特定することで問題の解決を図る．E-AoSAS++で考えられるフォールトの整理をおこない，検査の視点を整理をおこなった．検査の視点の依存関係を考慮し，段階的にフォールトを検査する方法を提案し，自動販売機のプリンタシステムを用いて検証をおこなう．

大規模なシステムに対して実行前検査をおこなう場合，状態の組み合わせ爆発が発生し，実行前検査ができないという問題がある．この問題の対応として，E-AoSAS++ のコンポーネントの階層関係に着目し，詳細化関係を用いて段階的に実行前検査をおこなうことで問題の解決を図る．詳細関係を用いた実行前検査を自動販売機のプリンタシステムを用いて検証をおこなう．

提案した手法を用いた実行前検査を支援する実行前検査支援ツールの開発をおこなう．開発した実行前検査支援ツールを用いて，自動販売機のプリンタシステムを例に検証をおこない，提案する方法の有効性について考察をおこなった．

本 OJL のメタ技術はアスペクト指向である．ベース技術は実行前検査技術であり，専用手法は E-AoSAS++に基づく，アスペクト指向を用いた組込みソフトウェアに対する実行前検査の適用である．

[1] 一般的なフェーリアは，実行時における挙動と仕様との不一致を指すが，ここでは，実行前検査で検出できる，予測される実行時の挙動と，仕様との不一致を指す．

2　E-AoSAS++

E-AoSAS++に基づくソフトウェアの静的構造はコンポーネントの階層構造で表現される．E-AoSAS++におけるアーキテクチャの基本構成要素であるモジュールは並行状態遷移機械 (以下，CSTM) であり，複数の CSTM がメッセージ通信をおこない，協調動作することで機能を提供する．個々の CSTM の振舞いは状態遷移図により表現され，複数の CSTM 間にまたがる振舞いは CSTM 間のメッセージ通信とその系列により表現される．

2.1　E-AoSAS++の並行実行

E-AoSAS++では複数の CSTM が並行に動作している．CSTM は有限個の状態を持ち，イベントに対応するアクションを実行することで機能を提供する．CSTM のイベント送受信を図 1 に示す．

CSTM 間の通信はイベントキューを介して行われ，CSTM は各々イベントキューを 1 つ持つ．CSTM 間のメッセージ通信は非同期通信で行われ，メッセージ通信で送信されたイベントは，送信先の CSTM のイベントキューに格納される．CSTM のイベントの実行は，イベントキューに格納されているイベントの受信，アクションの実行，状態遷移の順で行われる．

2.2　アドバイスの織込み

E-AoSAS++でアドバイスを織込むことが可能な箇所は，CSTM のイベント受信であり，イベントに対応するアクションの実行前に実行されるアドバイスとアクション実行後に実行されるアドバイスを織込むことが可能である．アドバイスの実行タイミングを図 2 に示す．同一のジョインポイントに複数のアドバイスが織込まれる時のアドバイスの実行順序は非決定的となる．

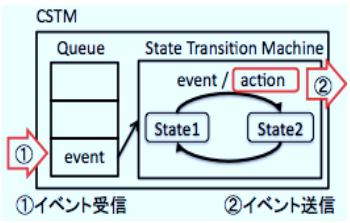

図 1 CSTM のイベント送受信

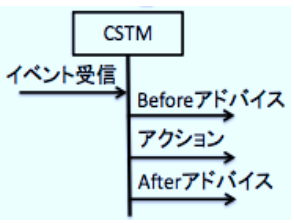

図 2 アドバイスの実行

3　実行前検査

E-AoSAS++を用いたシステムのフォールトは，並行処理を考慮せずにアーキテクチャの構築を行わなかった

ことに起因するフォールトとアスペクト干渉を考慮しなかったことに起因するフォールトが考えられる．これらのフォールトを検査する視点を整理し，検査方法を提案する．

並行処理に起因するフォールトは一般的に実行前検査で検査可能な項目の検査をおこなうことで対応する．アスペクト干渉に起因するフォールトは，アスペクトの織込み順序に関わらず，同一の動作をおこなうよう考慮してアーキテクチャを構築することで対応をおこなう．

3.1 E-AoSAS++のフォールトの検査視点の整理

実行前検査で検査したい項目を「決定的な振る舞い」，「非決定的な振る舞い」,「アスペクト干渉」の観点から分析した．

a 決定的な振る舞い

決定的な振る舞いとは，着目するイベントの系列が一意に決定する場合のフォールトを検査する．例えば，自動販売機において，コインを投入した後でボタンを押すと商品が排出されるといった一連の振る舞いを満たすかの検査をおこなう．

b 非決定的な振る舞い

非決定的な振る舞いとは，複数のイベントが並行実行することで，イベントの系列が一意に決定されない振る舞いのことである．非決定的な振る舞いの検査で，デッドロック，ライブロック等の一般的な並行実行で発生するフェーリアが発生しないかの検査をおこなう．

c アスペクト干渉

複数のアドバイス実行による干渉を以下に定義する．
衝突：ジョインポイントかつ，同一実行タイミングのアドバイスが複数存在する．
干渉 ： 衝突したアドバイスの実行順序の違いによって，実行結果に違いが生じる．

E-AoSAS++において，アドバイスの実行は CSTM に対するイベント送信である．E-AoSAS++におけるアスペクト干渉を以下のように定義する．

“ 衝突するアドバイスが同一の CSTM に対してイベントを送信する時，アドバイスが干渉することを考慮していないフォールトが実行時に顕在化し，フェーリアとなることである．”

3.2 段階的なフォールトの検査方法

決定的な振る舞い，非決定的な振る舞い，アスペクト干渉に着目して，段階的に検査をおこなう．

非決定的な振る舞いのフォールトが発生しないためには，並行実行する各プロセスが正しいことが前提である．よって，各プロセスは決定的な振る舞いの検査でフォールトが存在しないことを検査している必要がある．

アスペクト干渉は，各プロセスが正常に動作することは保証されているが，アスペクト干渉により，プロセスの振る舞いが開発者の意図しない振る舞いをする事で発生する．よって，非決定的な振る舞い，決定的な振る舞いの検査でフォールトが存在しない事を検査している必要がある．

各検査の関係から次の順番で段階的に実行前検査をおこなう．(1) 決定的な振る舞い (2) 非決定的な振る舞い (3) アスペクト干渉上記の順番で実行前検査をおこなうことにより，フォールトの原因を絞りこむことが可能である．

4 状態数を削減した実行前検査方法の提案

詳細化関係を適用して，検査対象のモデルの状態数を削減する方法の提案をおこなう．

詳細化関係は，基準となるプロセスを Spec(仕様)，基準となるプロセスを詳細化したプロセスを Impl(実現) とした場合，Spec ⊑ Impl の関係が成立することである．

例えば inp イベントを与えると out イベントを出力するシステムに対して，Spec を図 3，詳細化された System を図 4 とした場合．Spec ⊑ Impl が成立する時に詳細化関係が成立する．Spec では入力に対する内部動作は考慮せずに，入力に対する出力が必ずあることが満たされれば良い．よって，詳細化関係が成立する時，Spec は Impl よりも状態数を削減することが可能である．

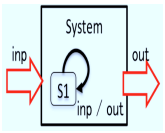

図 3 Spec の例

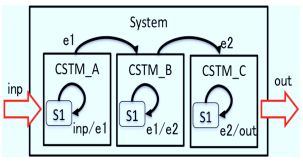

図 4 Impl の例

4.1 詳細化関係を用いた段階的な実行前検査

大規模なモデルに対して，モデル全体を一度に検証するのは状態の組み合わせ爆発が発生するので検査ができない．大規模なモデルで実行前検査をおこなう方法として，モデルの抽象化をおこなう．E-AoSAS++のコンポーネントの階層構造に着目し，モデルの抽象化をおこなった．E-AoSAS++のシステムの構造を図 5 に示す．

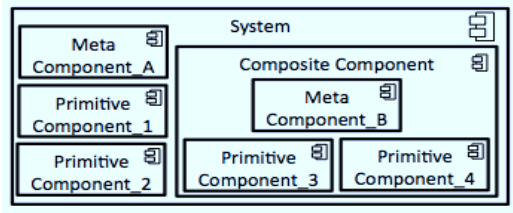

図 5 E-AoSAS++のシステムの静的構造

E-AoSAS++のシステムは複数のコンポーネントの集合により実現される．図 5 のシステムの仕様を S とした場合，システムは次の関係を満たすことを検証する．

$S[= MetaComponent\_A \parallel MetaComponent\_B \parallel PrimitiveComponent\_1 \parallel \cdots \parallel PrimitiveComponent\_4$

詳細化関係を適用する対象として CompositeComponent に着目し，適用をおこなう．CompositeComponent の仕様を M とした時，次の関係を満たす事を検証する．

$M [= MetaComponent\_B \parallel PrimitiveComponent\_3 \parallel PrimitiveComponent\_4$

上記の関係が成立するとき，System は次の関係を検証すれば良い．

$S[= MetaComponent\_A \parallel PrimitiveComponent\_1 \parallel PrimitiveComponent\_2 \parallel M$

5　自動販売機のプリンタシステム

自動販売機のプリンタシステムを用いて，段階的なフォールトの検査，詳細化関係を用いた実行前検査をおこなった．

自動販売機のプリンタシステムはスイッチの操作する順番によって，印刷するデータが選択され，対応するデータが記載されたシートが出力される．今回設計した自動販売機のプリンタシステムのコンポーネント図を図 6 に示す．

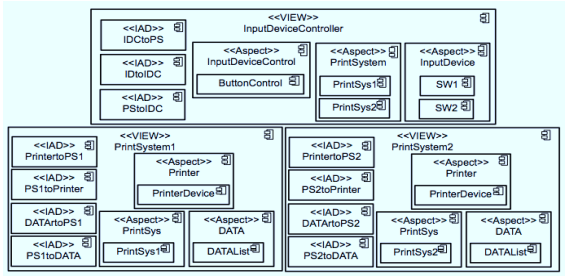

図 6　プリンタシステムのコンポーネント図

自動販売機のプリンタシステムはスイッチの入力を制御して，プリンタシステムに印刷命令を送信する InputDeviceController．印刷命令を受理したら，印刷シートを作成して，プリンタに印刷命令を出力する PrintSystem から構成される．PrintSystem は 2 つ存在しており，異なるデータが記載された印刷シートが出力される．

5.1　段階的なフォールトの検査の適用

a 決定的な振る舞いの検査

決定的な振る舞いの検査として，以下のようにイベント系列が決定的な場合に仕様を満たすかを検査する．

1) Switch を押す，対応する PrinterSystem から印刷命令がプリンタに出力される，プリンタから印刷終了命令を受理して終了．

2) Switch を押す．対応する PrinerSystem から印刷命令がプリンタに出力される．Switch を押す．印刷命令を出力した PrinterSystem から印刷キャンセル命令がプリンタに出力される．

b 非決定的な振る舞いの検査

非決定的な振る舞いの検査として，自動販売機のプリンタシステムに対する入力が非決定的に発生する場合の検査をおこなう．自動販売機のプリンタシステムに対する入力は，Switch1 を押す，Switch2 を押す，プリンタからの印刷終了命令である．これらの入力は非決定的に発生する場合に，仕様を満たすかの検査をおこなう．

c アスペクト干渉の検査

対象とした自動販売機のプリンタシステムでは，衝突するアドバイスのイベント送信先の CSTM が同一の場合が無いので考慮しない．

5.2　詳細化関係を用いた段階的な実行前検査の適用

プリンタシステムは 3 つの VIEW から構成される．InputDeviceController は PrinterSystem1，2 に対して印刷要求をおこなう．よって，PrinterSystem1，2 の VIEW に着目し，詳細化関係を適用した．

PrinterSystem1，2 の仕様は，印刷要求を受理したら印刷シートを作成して，プリンタに印刷要求を出力する．プリンタから印刷終了を受理したら印刷終了を出力し，印刷キャンセル要求を受理したら，プリンタに印刷キャンセル要求を出力する．

PrinterSystem1，2 の仕様を PS1_Spec，PS2_Spec，自動販売機のプリンタシステムの仕様を S とした時，次の関係を満たすことを検証する．

S [= InpuDeviceController ‖ PS1_Spec ‖ PS2_Spec

6　実行前検査支援ツール

実行前検査支援ツールの入力と出力を整理し，実行前検査支援ツールのプロトタイプの開発をおこなった．

実行前検査支援ツールは，アーキテクチャ記述から実行前検査器 FDR[3] の入力となる CSP 記述 [1] を生成する．E-AoSAS++の実行前検査は，アーキテクチャ記述をプロセス代数の一つである CSP で表現し，CSP 記述を入力とした実行前検査器 FDR を用いておこなう方法が提案されている [5, 6]．

6.1　実行前検査支援ツールの概要設計

実行前検査をおこなうために必要な情報として，ソフトウェアの詳細な動作を表すモデル，外部からの入出力を表す環境の記述，検査対象が満たすべき仕様の情報が必要である [2]．

E-AoSAS+においてシステムの詳細な動作を表すモデルはアーキテクチャ記述である．環境の記述，検査対象の仕様はステートマシン図で記述し，入力として与える．

実行前検査支援ツールでは，E-AoSAS++のアーキテクチャ記述から，インスタンスの情報と，E-AoSAS++のモデルを構築するのに必要な情報を取得する．E-AoSAS++のモデルとインスタンスの情報を参照し，開発者が着目しているコンポーネントの情報，検査対象の仕様，検査対象の環境の記述，CSP 記述のライブラリの情報を与えることで CSP 記述を生成する．

実行前検査支援ツールのプロトタイプとして，アーキテクチャ記述から，CSP 記述モデルに変換し，CSP 記述モデルから CSP 記述を生成するツールの実現をおこなった．図 7 にプロトタイプの実行前検査支援ツールの概要設計を示す．

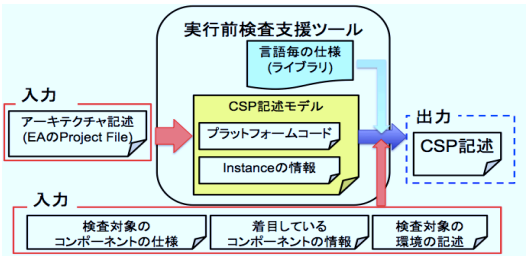

図 7　実行前検査支援ツールの概要設計 (プロトタイプ)

7　考察

7.1　段階的なフォールトの検査に対する考察

段階的なフォールトの検査に対する有効性を，自動販売機のプリンタシステムで発見できたフォールトを例に確認する．

非決定的な振る舞いの検査で検出したフェーリアが発生するまでのイベント列のシーケンス図を図 8 に示す.

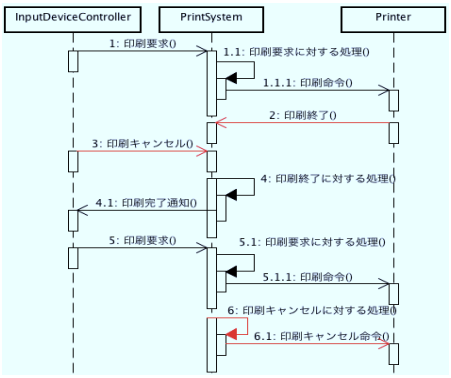

図 8 フェーリアが発生するまでのシーケンス図

今回フェーリアが発生した原因となるのは図 8 のシーケンス番号 2，3 の箇所である．PrintSystem に対して，印刷キャンセル，印刷終了イベントがほぼ同時に発生したときに，PrintSystem は先に Queue に格納された印刷キャンセルイベントに対する処理をおこなう．PrintSystem の Queue には，印刷終了イベントが格納された状態であり，その状態で PrintSystem に印刷要求イベントが発生すると，即座に印刷完了イベントに対する処理をおこない仕様を満たさなくなる．よって，今回のフェーリアは InputDeviceController とプリンタが並行に動作するときに，同時に PrintSystem にイベントが発生することを考慮しなかったというフォールトである．

決定的なイベント列の検査では仕様を満たすことを確認していたので，フェーリアが発生するまでのイベント列を確認し，決定的なイベント列の検査では発生しないイベントの発生に着目することで，フェーリアの発生箇所にあたりを付けることができた．段階的なフォールトの検査を用いいる事で，余分なフォールトを排除して考えることができ，フォールトの発見が容易になった．

今回発生したフェーリアは PrintSystem1，2 の VIEW に対して，詳細化関係を適用し，非決定的なイベント列の検査をおこなった際にも発見することができた．PrintSystem の VIEW に対して，段階的なフォールトの検査をおこなう過程で，同様のフォールトが発見できた．VIEW をひとつの検査の視点と捉え，検査の範囲を限定することでフォールトの特定がおこないやすくなる．

7.2 詳細化関係を用いた実行前検査に対する考察

PrintSystem1，2 に詳細化関係を適用した自動販売機のプリンタシステムの実行前検査では検出されたなかったフォールトが，詳細化関係を適用していない自動販売機のプリンタシステムの実行前検査で発見できた．

原因は PrintSystem1, 2 で共通して利用される，DATAList と PrinterDevice にアドバイスを織込むことで，IAD にイベントを送信し，処理を実現している．衝突するアドバイスが予期せぬタイミングで実行されることで，仕様と異なる動作をおこなったことが原因である．詳細化関係を適用することで，共有資源に対するアクセスは内部動作として隠蔽されてしまい，詳細化関係を適用した自動販売機のプリンタシステムの実行前検査では発見できなかった．

PrintSystem1，2 の段階的なフォールトの検査で発見できなかった原因は，PrintSystem の仕様のみを考え環境の記述を作成した結果，他の VIEW からの共有資源に対して発生するイベントについて考慮していなかったからである．この問題の対応として，PrintSystem1，2 に対する検査として，他の VIEW からの共有資源に対するイベントも考慮し，段階的なフォールトの検査をおこなった．結果，フォールトを検出することができた．

詳細化関係を適用するには，詳細化関係を適用する対象の仕様と，共有資源を考慮して段階的なフォールトの検査を行なう必要がある．

8 おわりに

本 OJL では，E-AoSAS++で発生するフォールトを分類し段階的にフォールトを特定する方法の提案をおこなった．詳細化関係を利用してシステムを構成するモジュールの状態数を削減する方法を提案し，自動販売機のプリンタシステムを例に用いて，提案した手法の有効性を示した．

本 OJL のメタ技術はアスペクト指向である．複数のフォールトから発生するフェーリアを，検査の視点を分離することにより，フォールトの特定をおこないやすくした．ベース技術は，実行前検査である．専用手法は E-AoSAS++に基づく，アスペクト指向を用いた組込みソフトウェアに対する実行前検査の適用である．

今後の課題は，アスペクト干渉が発生する例での段階的なフォールトの検査の考察である．

参考文献

[1] C.A.R Hoare, *Communicating Sequential Pro-cess,* Prentice-Hall, 1985.

[2] E. M. Clarke, O. Grumberg and D. A. Peled, *Model Checking*, The MIT Press, 2000.

[3] Formal Systems (Europe) Limited, “ FDR, ” http://www.fsel.com/index.html/, 2007.

[4] Noro, M., Sawada, A., Hachisu, Y. and Banno, M., “ E-AoSAS++ and its Software Development Environment,” *Proceedings of the 14th Asia-Pacific Software Engineering Conference(APSEC 2007)*, pp.206-213, 2007.

[5] 張漢明，蜂須吉成，沢田篤史，野呂昌満, “ アスペクト指向ソフトウェアアーキテクチャの振る舞い検証に関する考察，” ソフトウェア工学の基礎ワークショップ XVI， pp.267-274，2009.

[6] 加藤大地，蜂巣吉成，沢田篤史，野呂昌満, “ E-AoSAS++に基づく開発支援環境 実行前検査支援ツールの提案，” 情報処理学会研究報告，vol.2009，no.13，pp.113-120，2009.

[7] 鵜林尚靖，玉井哲雄, “ アスペクト指向プログラミングへの実行前検査方法の適用，” 情報処理学会論文，vol.43，no.6，pp.1598-1609，2002.