自動販売機制御ソフトウェアの再開発
—核資産の定義とコード生成ツールの設計と試作—

M2009MM008 伊藤英樹
指導教員：野呂昌満

1　はじめに

本稿では，富士電機リテイルシステムズ株式会社 (以下,FRS) と南山大学との共同研究および OJL 開発の事例報告をおこなう．この報告では，ベース技術として MDA とアスペクト指向技術，メタ技術として PLSE[1]，専用手法として新規環境のプラットフォームコードに対する PLSE の適用をおこなう．専用手法を用いて FRS における既存資産と新規環境に基づく部品が共存可能な環境を実コード上で実現し，新規環境への移行におけるレガシーコード対応の一方法について考察する．

自動販売機には，類似した機能を持つ製品が多数存在している．FRS では，類似機能を持つ自動販売機の製品群に対して，PLSE に基づく自動販売機制御ソフトウェアの開発を実践している．リアクティブシステムである組込みソフトウェアの開発では，一般的に並行に動作する状態遷移機械の集合によるシステムのモデル化がおこなわれており，FRS ではオブジェクト指向を用いた並行状態遷移機械の集合でシステムのモデル化をおこなってきた．この際，ハードウェアの種類に応じた例外処理や実時間処理などの横断的関心事が存在し，適切なモジュール化や部品化をおこなうことが困難となる．

本 OJL では，横断的関心事に伴う問題をアスペクト指向技術により解決し，アスペクト指向モデルからコードを生成するツール (以下, 生成ツール) を提案することで，PLSE に基づく開発支援の環境整備をおこなう．PLSE に基づく新規環境への移行を実践する上で既存資産を新規環境に移植可能にすることが移行コストを削減するために望まれる．本 OJL の目的は，生成ツールによって生成されるコードと FRS の既存資産が共存可能な環境を構築し，双方のソフトウェア部品を再利用可能にすることである．移行への一時的な環境にあたる共存環境をアスペクト指向モデルが対象とするプラットフォームコードの派生プロダクトと位置づけ，プラットフォームコードに対する PLSE と捉えることで共存環境の構築を図る．

我々の研究室では組込みシステム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 (以下，E-AoSAS++)[2] を提案している．E-AoSAS++では，組込みシステムを並行状態遷移機械の集合で規定し，横断的関心事による問題を複数視点で捉え，並行状態遷移機械の階層構造で統一的に記述するための枠組みを提案している．本 OJL ではアスペクト指向モデルの記述方法として E-AoSAS++を採用する．

本 OJL の成果として，FRS が提案する実行環境の並行処理を共存環境のプラットフォームコードに適応することにより，双方の部品の再利用を可能にする共存環境を構築した. 以上のことから，PLSE が本 OJL におけるメタ技術に位置づけられることを確認した．

2　E-AoSAS++と PLSE に基づく開発支援環境

E-AoSAS++は，組込みシステムを対象としたアスペクト指向ソフトウェアアーキテクチャスタイルである．E-AoSAS++の構成要素は，並行に動作する状態遷移機械 (以下, CSTM) であり，システムを CSTM の集合により規定し，キューを介して非同期のイベント通信により CSTM 間は協調動作する．

2.1　PLSE に基づく開発支援環境

E-AoSAS++では，PLSE に基づく開発支援環境の一つとして生成ツールを提案している．PLSE がアーキテクチャ中心開発であることからアーキテクチャに基づくプラットフォームコードにおけるフローズンスポットを自動生成するための技術として MDA を生成ツールに適用し，プログラム作成の労力を削減している．図 1 に提案する生成ツールの概要を示す．

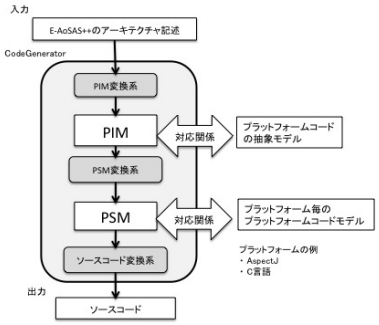

図 1　生成ツール概要

生成ツールの入力は，E-AoSAS++のアーキテクチャ記述である．出力は，対象とするプラットフォームのプラットフォームコードに基づくソースコードである．ツールは，MDA の概念に基づき入力となるアーキテクチャ記述からモデル変換によってソースコードを生成する．提案する生成ツールの特徴は，特定の実装技術に依存しないプラットフォームコードの抽象モデルの実現に必要な情報を抽象構文木として定義し，中間形として用いていることである．プラットフォームコードの抽象モデルは，E-AoSAS++の動的意味を基に設計されていることから，ツールは動的意味の拡張・変更に柔軟に対応可能となる．

2.2　プラットフォームコードの抽象モデル

E-AoSAS++プラットフォームコードの抽象モデルでは，CSTM の構造とインスタンスの関連をアスペクトとしてアスペクト指向設計している．図 2 にプラットフォームコードの抽象モデルを示す．

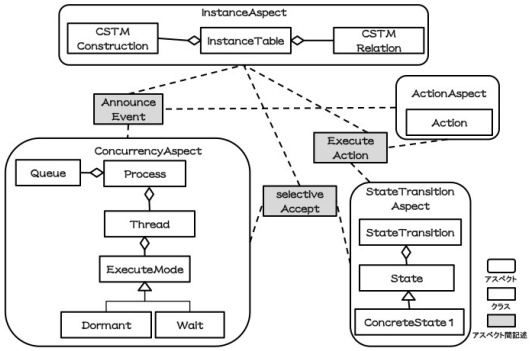

図 2 プラットフォームコードの抽象モデル

プラットフォームコードの抽象モデルでは，並行処理をコアコンサーンとして，CSTM が並行に動作するための処理を状態遷移機械から分離している．そして，アクションの部品化を目的に状態遷移と遷移に伴うアクションをそれぞれアスペクトとして分離することで CSTM の構造を設計している．CSTM のインスタンスの個数とそれらの関連を管理するインスタンス処理アスペクトと CSTM を構成するアスペクトが IAD によって結合され，E-AoSAS++で定義される動的振舞いを実現する．

3 FRS が提案する自動販売機制御ソフトウェア実行環境 ARTiC

Abstract RealTime Control(以下, ARTiC) は，FRS における自動販売機制御ソフトウェア開発のための実行環境である．本 OJL では，共存環境を構築する上での問題を明確にするために ARTiC のプラットフォームコードをモデル化し分析をおこなっている．図 3 に分析した ARTiC のプラットフォームコードモデルを示す．

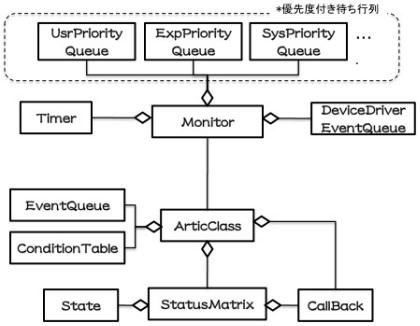

図 3 ARTiC プラットフォームコードモデル

ARTiC の実行単位は ARTiCClass のオブジェクトであり，状態遷移機械として振る舞う．ARTiC では，システム全体のスケジューリングを管理するモニタが存在する．モニタは，オブジェクトの優先度にあわせて複数の優先度付きキューを管理し，優先度の高いキューに格納されるオブジェクトから順に実行権を割り当てることで擬似並行に動作する．各オブジェクトの状態遷移機械としての振舞いは，状態遷移論理を表す状態表と特定の状態において特定のイベント受理の保留を指定する条件表によって実現される．

4 プラットフォームコードに対する PLSE の適用

4.1 共存環境構築における課題と解決アプローチ

本 OJL では，共存環境を E-AoSAS++のプラットフォームコードの派生プロダクトと捉え，プラットフォームコードに対して PLSE を適用する．プラットフォームコードに PLSE を適用し，特定の対象プラットフォームに求められる要求や制約に対する特定部分を明確にすることで，生成ツールは対象プラットフォームの変化に応じて生成コードを変更することで要求の変更に柔軟に対応可能になる．

E-AoSAS++と ARTiC のプラットフォームコードモデルの対応関係の整理をおこない，共存環境を構築する上で，並行処理に関する以下の三つの課題を特定した．

- スレッドのスケジューリングの差異
- キューのイベント消去，特急イベント送信などの ARTiC に特有の処理
- 状態遷移機械の実行状態の管理

本 OJL では，上記の課題に対して E-AoSAS++プラットフォームの並行処理アスペクトを ARTiC の擬似並行処理で置き換えることで共存環境を構築する．

ARTiC の擬似並行処理のスケジューリングと E-AoSAS++のスケジューリングを比較し，ARTiC の擬似並行処理が E-AoSAS++の並行処理として利用可能であることを示すことができれば，共存環境において E-AoSAS++の振舞いの一貫性を満たし，かつ ARTiC との共存が可能となる．さらに，ARTiC の擬似並行処理を利用することで，キューのイベント消去や特急イベント送信などの ARTiC 特有の概念に関しても利用可能となり，ARTiC の機構を再利用することで共存環境構築のためのコストを削減できる．

共存環境構築の課題として挙げた実行状態の管理は，E-AoSAS++では CSTM の実行状態を計算モデルで定義し並行処理アスペクト内で実現しているのに対して，ARTiC では実行状態に関する概念が存在せず，イベントの破棄や保留を状態遷移機械の状態遷移論理として実現していることである．共存環境では，並行処理アスペクトにおける実行状態の振舞いを ARTiC オブジェクトの状態遷移論理として置き換えることで実現をおこなう．

図 4 に共存環境構築の解決アプローチの概要を示す．

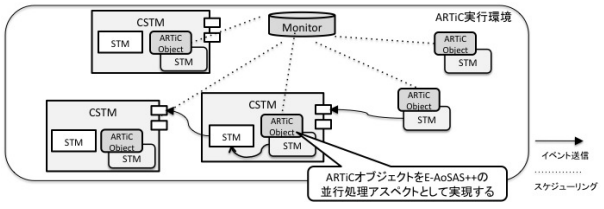

図 4 共存環境構築の解決アプローチ

ARTiC における並行処理実体は，ARTiC オブジェクトである．ARTiC の擬似並行処理では，ARTiC オブジェクトがモニタにスケジューリングされて動作することから，図 4 に示すように E-AoSAS++の構成要素である CSTM は内部に ARTiC オブジェクトを持つ構造となる．

4.2 E-AoSAS++と ARTiC の並行処理の比較

E-AoSAS++と ARTiC のスケジューリングを比較し，ARTiC の擬似並行処理のスケジューリングが E-AoSAS++の並行処理のスケジューリングの振舞いの一貫性を満たすことを示す．E-AoSAS++において各状態遷移機械は，Signal-Wait を用いて並行に動作する．図 5 に CSTM の Signal-Wait による同期処理を示す．

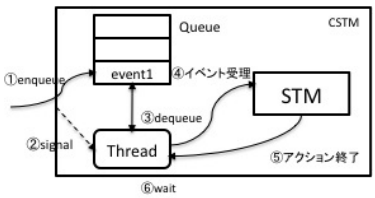

図 5 CSTM の signal-wait

CSTM は外部からイベントを受け取り，Signal 命令がかけられるとスレッドが起動されキューを参照する．キューの走査を CSTM の実行状態に応じておこない，受理可能イベントを選択する．状態遷移に伴うアクションを実行し，キューに受理可能イベントが存在しない場合 Wait する．E-AoSAS++では，Signal 命令を受けて実行可能なスレッドに実行権が割り当てられ並行に動作する．

ARTiC では，システム全体のスケジューリングを管理するモニタが各状態遷移機械の優先度にあわせたキューを管理し，イベント実行する状態遷移機械を決定する．図 6 に ARTiC の擬似並行実行の概要を示す．

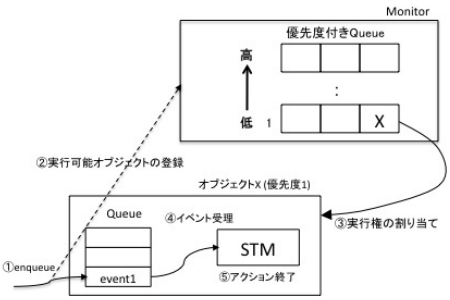

図 6 ARTiC の擬似並行実行

各 ARTiC オブジェクトは，外部からイベントを受信するとモニタに実行可能オブジェクトとして登録する．モニタは，オブジェクトの優先度に応じた優先度キューの末尾に実行可能オブジェクトを格納する．モニタは，スケジューリングをおこなう際に優先度の高いキューから順に参照し，実行可能オブジェクトが存在するキューの最も先頭のオブジェクトを取り出し実行権を割り当てる．実行権が割り当てられたオブジェクトはイベントキューを参照し，受理可能な最も先頭にあるイベントに対するアクションを実行する．

E-AoSAS++,ARTiC ともに状態遷移機械はイベント受理後，アクション実行が終了するまで次のイベント受理をおこなわない．ARTiC では，実行可能なオブジェクトがモニタによって到着順にスケジューリングされるのに対して，E-AoSAS++では実行可能な全ての状態遷移機械に実行権が割り当てられる可能性がある．つまり，ARTiC のスケジューリングは，E-AoSAS++が用いるスケジューリングのサブセットとして利用可能であると考える．

4.3 提案する共存環境のプラットフォームコードモデル

E-AoSAS++の並行処理に ARTiC の擬似並行処理を利用した際のプラットフォームコードの静的構造と振舞いについて述べる．提案する共存環境のプラットフォームコードモデルでは，並行処理アスペクトの設計と並行処理アスペクトと状態遷移アスペクトを繋ぐアスペクト間記述の変更をおこなうことで実現している．提案する共存環境のプラットフォームコードモデルを図 7 示す．

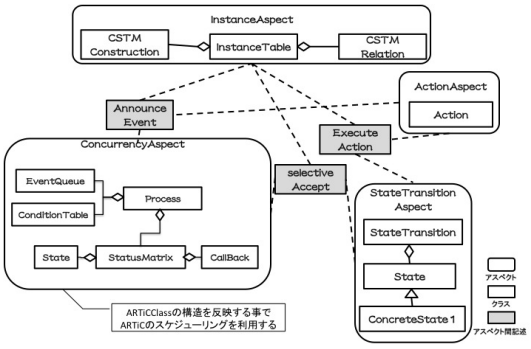

図 7 共存環境のプラットフォームコードモデル

本 OJL では ARTiC の擬似並行処理を E-AoSAS++の並行処理アスペクトとして利用していることから，CSTM の並行処理アスペクトの構造は ARTiC のプラットフォームコードモデルの構造を反映した設計となっている．ARTiC の擬似並行処理を CSTM の並行処理アスペクトとして実現することから，スケジューリングの差異と ARTiC に特有の処理に関しての対応をとることが可能になったが，実行状態に関しての振舞いを実現する必要がある．

ARTiC では，状態遷移論理によってイベントの破棄や保留を状態表と条件表によって実現していることから，共存環境では，CSTM の実行状態を ARTiC オブジェクトの状態遷移論理によって実現している．提案する共存環境では，図 8 に示すように状態遷移アスペクト内の各状態毎に受理可能なイベントを識別するための条件表の一部を保持し，状態遷移機械の状態が遷移した際に状態遷移アスペクトと並行処理アスペクトを繋ぐアスペクト間記述によって条件表を動的に変更することで実現している．

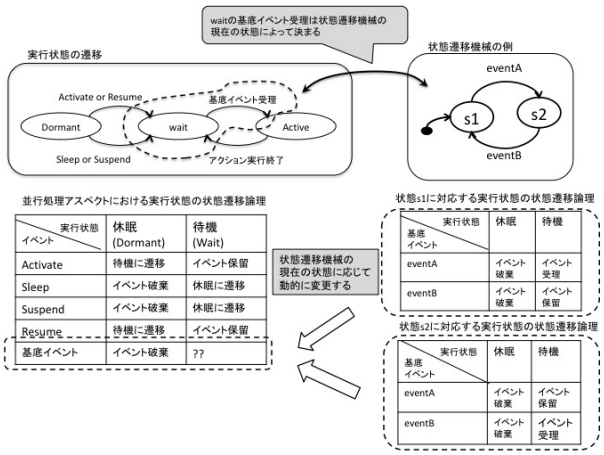

図 8 ARTiC オブジェクトの実行状態としての振舞い

5　考察

5.1　共存環境の妥当性

本 OJL では, 共存環境における E-AoSAS++の並行処理アスペクトを ARTiC の擬似並行処理を用いて実現している. 以下の三つの事柄を確認できたことから共存環境が E-AoSAS++の振舞いの一貫性を保証すると考える.

- ARTiC では割込みの概念が存在せず, キューからイベント受理しアクション実行を終えるまでは, 他のイベントを受理することがない
- ARTiC のスケジューリングが E-AoSAS++のスケジューリングのサブセットにあたる
- CSTM の実行状態の振舞いを状態表と条件表の組み合わせで実現可能である

さらに，提案する共存環境では ARTiC の擬似並行処理を利用することによって ARTiC 特有の処理を利用可能になっている．これにより，既存資産である ARTiC オブジェクトとのインタフェースの整合をとることが可能になり，サブシステム単位での段階的な E-AoSAS++への書き換えを容易におこなえ，移行コストを削減できると考える．以上のことから，提案する共存環境の設計が妥当であると考える．

また，本 OJL では共存環境を E-AoSAS++プラットフォームコードに対する PLSE と捉え実現している．E-AoSAS++のプラットフォームコードでは，アクションの部品化のためにアクションアスペクトを状態遷移論理から分離し，ユーザ記述箇所としている．共存環境では並行処理アスペクトを実現するコードと並行処理アスペクトに関連するアスペクト間記述のみを特定部分としていることから，ARTiC から E-AoSAS++に完全に移行が完了した後においてもアクションコードが再利用可能になる．これは，プラットフォームコードモデルをアスペクト指向設計することで，分離されたアスペクトがお互いに依存することなく変更可能になっているからである．本 OJL の成果から，PLSE に基づく再開発をおこなう上で核資産となるプラットフォームコードに対する PLSE の適用が環境移行における移行コストを削減するために有効であり，プラットフォームコードをアスペクト指向設計することで特定のプラットフォームに求められる要求や制約に柔軟に対応可能な設計となることを確認できたと考える．

5.2　PLSE における環境移行に関する考察

E-AoSAS++と ARTiC の共存環境を構築したことで，今後の環境移行において ARTiC に基づく既存資産を核資産として定義できると考える．また，環境移行における核資産を特定するために再開発プロセスを定義する．提案する生成ツールを用いた再開発プロセスを図 9 に示す. 定義した再開発プロセスでは，まず再開発の対象システムの分析を行い，外部システムとのイベント通信のインタフェースを特定するとともに，既存資産から対象とするサブシステムの仕様を復元する．このとき，内部仕様の分析として既存の ARTiC ソースコードを分析する

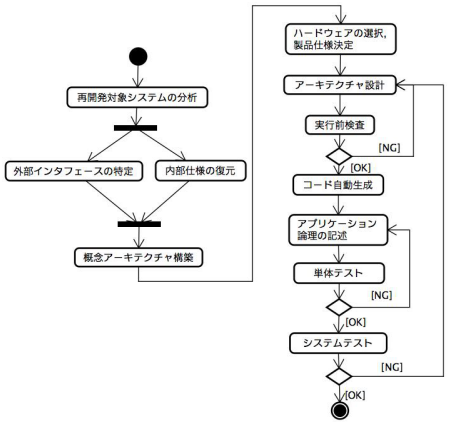

図 9　生成ツールを用いた再開発プロセス

際に，ARTiC オブジェクトのアクション実行に対応する Callback に着目することで，E-AoSAS++プラットフォームコードにおけるアクションコードの作成が容易になると考える．アクションコードはアプリケーション論理に関する処理を Command パターンを用いてオブジェクトとして分離していることから，ARTiC オブジェクトの CallBack に着目して分析することでアクションコードの核資産開発がおこなえるからである．以上のことから，PLSE を実践する上でプラットフォームコードの構造をアスペクト指向設計によって関心事毎に分離し，部品化をおこなうことは核資産開発における再利用部品の作成を容易にする上でも有効であることを確認できたと考える．

6　おわりに

本 OJL では，E-AoSAS++と ARTiC の共存環境を新規開発環境のプラットフォームコードの派生プロダクトと位置づけ，プラットフォームコードに対する PLSE と捉えることで実現をおこなった．共存環境の構築は，E-AoSAS++のプラットフォームコードにおける並行処理アスペクトを ARTiC の擬似並行処理を用いることで実現している．本 OJL のメタ技術は PLSE である．本 OJL では，MDA を用いた生成ツールによるプログラム作成労力の削減，アスペクト指向技術によるプラットフォームコードのコンポーネントの再利用性の向上を実現した．これらのベース技術を合成するメタ技術として PLSE が挙げられる．本 OJL の専用手法は，E-AoSAS++のプラットフォームコードに対する PLSE の適用である．

今後の課題は，実システムを用いた共存環境の動作検証と生成ツールの実現である．

参考文献

[1] K. Pohl, G. Bockle, and F. Linden, “ *Software Product Line Engineering Foundations, Principles, and Techniques,”* Springer-Verlag, 2005.

[2] 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.