4H-6

アスペクト指向に基づくプラント監視制御フレームワーク

中川　晃一†

三菱電機株式会社　先端技術総合研究所

1. はじめに

水処理システムを代表とする産業システムのプラントは、リニューアル、増設、改造という形態が主となりつつある。また、プラントを新規に設置する場合においても、最初に全体を構築するのではなく、需要に対応して段階的にプラントを拡張していく形態が主流となると予測される。

しかし、システム構築を段階的に行うことにはいくつか問題がある。まず、数年後に同一の機器や開発環境が調達可能であるかどうかわからない。次に、開発期間中の機能拡張などの要求に容易に対応できないことが挙げられる。こうした機能拡張に関する要求はシステム内の多くのモジュールを縦断した実装が必要になり開発コストがかかる。また、実装のためにはシステムの十分な理解が必要になり熟練した技術者を長期間に渡って確保する必要がでてくる。

本稿では段階的なシステム拡張に対応したプラント監視制御システムを構築する方式について提案する。本方式ではアスペクト指向なスクリプト言語の開発実行環境をコントローラに内蔵することにより、保守性の向上やカスタマイズ性の向上を可能した。スクリプト言語はインタプリト方式で、コンパイラ・ライブラリなどの開発ツールが不要で、文法も理解しやすく開発の効率も向上する。また、アスペクト指向は非機能要求をモジュール化する技術として知られている。本スクリプト言語ではアスペクト指向なモジュールを定義することができ機能の修正や変更が容易に行える。

本稿では提案する本方式によるソフトウエアフレームワークの構成と概要について説明する。

2. 監視制御システムの実行モデル

プラント監視制御システムの実行モデルはイベント駆動が基本である。つまり、何分毎に処理を行う、ある条件が成立したときのみ処理を行うなど、時限や条件判断などに基づいて決まった処理が実行される。例えば、定期的にプラントの機器の現在値を収集しデータベースに蓄積する。機器の現在値を監視しながら制限値を超えた場合にアラームを発生させる処理などである。こうしたシステムでは同一のイベントで異なる処理が同時に実行したり、１つのイベントで複数の処理を連動して実行したりする場合が多くみられる。例えば決められた順序で排水ポンプを運転・停止させて排水を行う場合などがそうである。従来はこうした複数の処理は、１つのプログラムの複数のモジュールに跨って実装されるため、機能の拡張や変更、例えばポンプの台数や順序を変更したりする場合などには複数のモジュールを修正する必要があった。

A proposal of an aspect-oriented framework for industrial plant supervisory control systems

†Koichi Nakagawa,Mitsubishi Electric Advanced Technology R&D Center

3. アスペクト指向

アスペクト指向は通常のモジュール化では複数のモジュールに跨って実装されていた機能を１つのモジュールにカプセル化できるようにしたものである。アスペクト指向では JPM(Join Point Model)[1]が使われる。ＪＰＭはイベント駆動に似たモデルでイベントに相当するジョインポイントと、イベントハンドラに相当するアドバイス、また複数のジョインポイントの集合をポイントカットと呼び、ポイントカットとアドバイスをモジュールとしてカプセル化したものをアスペクトと呼ぶ。本方式では変数の更新や制限値越えなどのトリガをジョインポイントとし、スクリプトで記述した処理を実行することをアドバイスとし、一連のトリガと実行するスクリプトをまとめたものをアスペクトと定義した。

4. 本方式の特徴

4.1. イベントシーケンスの部品化

図１はポンプ起動に関する処理の例を示したものである。ここではトリガ発生、イベント発生条件の判定、発生したイベントによるアクションの起動を示している。アクションの起動に関しては(a)から(c)のように同じ処理でも異なるパターンが存在する。各パターンは一連のイベントと、イベントにより起動されるアクションで構成されている。従来はこうしたパターンはまとめて大きなモジュールで実装されていたため、ソフトウエアの再利用性が低かった。本方式ではこれらパターンをイベントシーケンスとアクションのそれぞれに分割して管理し、その組み合わせをアスペクトとしてモジュール化する。例えば(a)であれば、時間 t の時差で連続したイベントを発生させる１つのイベントシー

ケンスと、起動されるポンプ 1 運転、ポンプ２運転の２つのアクションの３つの要素で構成される１つの”ポンプの非同期起動”というアスペクトとして定義することができる。こうした結果、各アクションの独立性が高まるので、別の処理で同じアクションを使ったり、同じイベントシーケンスで別のアクションを起動したりするといったことが可能になり、ソフトウエアの再利用性を向上させることができる。こうした例には操作値をプラントの機器へ書き込む操作において、書込み操作を行ってから、その結果を読み出すことで処理を完了させるといった同期処理や、異常通報のように、ある制限値を超えた場合に複数の通報先に通報するといった並列処理、操作を行ってから一定時間内に応答がなければ処理を行うタイムアウト処理などがある。

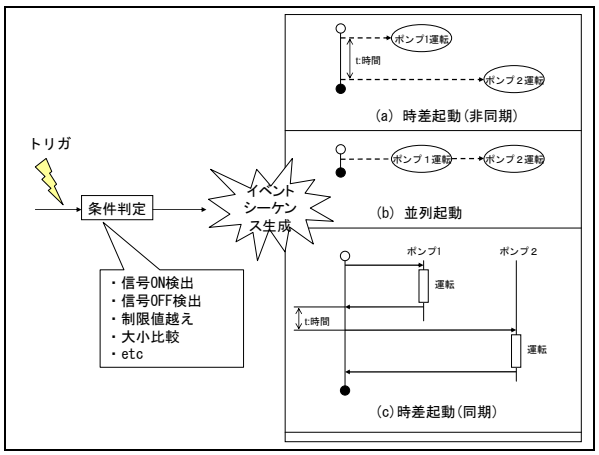

図 1　ポンプ起動の例

4.2.　スクリプトによるカスタマイズ

機能の拡張や変更はスクリプトにより行うことができる。スクリプトを使って起動するアクションやイベントシーケンス、これら２つを使ったアスペクトも定義することができる。そのためテキストエディタなど汎用ツールを使ってスクリプトを記述し、コントローラに転送、コントローラ内でスクリプトの再読み込みを行うことで機能の拡張・変更が行える。

5.　提案するシステムの構成

本システムの構成を図２に示す。本システムはコントローラと汎用ネットワークで繋がれた Web 端末で構成される。両者の間の通信はＷｅｂサーバとＷｅｂブラウザにより行われる。コントローラにはスクリプトを実行するスクリプトエンジンとプラントのデータを蓄積するプラントＤＢ（データベース）で構成される。スクリプトエンジンは定義されたスクリプトによってデータ収集やアラームの生成などを行う、また I/O 処理など機種依存の機能や専用命令などは拡張 API を使って実装することでスクリプトエンジンの機能を拡張できる。Web 端末は監視やエンジニアリングを行うために使われる。監視端末では Ajax を使った監視画面を実行することができ、プラント監視画面のような高度な監視画面も表示することができる。エンジニアリングはアクションの定義やアスペクトの定義などを行うが、実際にはスクリプトを編集するだけなので特別なツールは使わず、Web ブラウザとテキストエディタだけで実現することができる。

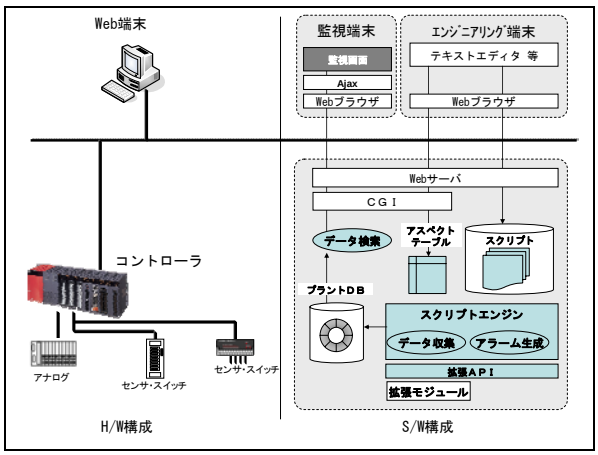

図 2　システム構成

6.　おわりに

本稿ではアスペクト指向に基づくプラント監視制御フレームワークの実装方式について提案した。本方式によれば、コンパイル不要で複雑な動作シーケンスをモジュール化する特徴をもつスクリプトエンジンをコントローラに内蔵することにより産業プラント特有の段階的な開発に低コストで対応することができる。今後は本フレームワークのデータ収集や通報処理などの基本機能について性能評価を行いながら、詳細な機能検証を行っていく予定である。

7.　参考文献

[1] Kiczales,G:The Fun Has Just Begun,Keynote Talk at International Conference on Aspect-Oriented Software Development(AOSD 2003)