SOA に基づく監視システムのプロトタイピング
アスペクト指向に基づくサービス指向システムアーキテクチャの提案

M2010MM005 長谷川裕記
指導教員：野呂昌満

1　はじめに

システムの構成要素や機能の変更に柔軟に対応可能なアーキテクチャ技術として Service-Oriented Architecture(以下，SOA) がある．SOA システムのアーキテクチャを設計する際の方法論が提案されている [3]．この方法論では，システムに求められる機能特性をサービスとしてモジュール化する方法が提案されている．しかし，非機能特性のモジュール化については言及されておらず，機能特性のモジュール化と同様の方法で行なえるかも明らかではない．

Induruwana は，SOA に基づくシステムに対して横断的関心事の分離にアスペクト指向を用いることを提案している [1]．この研究では，階層アーキテクチャ上に Application Service Integration Layer(以下，ASIL) を設けることで二種類の関心事を分離できるとしている．一つは，仕様の異なるサービスを統合する際の統合ロジックの分離である．また，もう一つは，複数のサービスに横断するロギングやセキュリティなどといった関心事の分離である．ASIL では分離した横断的関心事をどのように記述するか定義している．しかし，SOA においてどのような横断的関心事が存在するか整理されていない．アーキテクチャはコンポーネントの意味を考慮したシステム構造を定義する，という立場に立てば，横断的関心事を整理分類することが必要となる．

本研究の目的は，SOA システムのアプリケーションアーキテクチャ設計を支援することである．Induruwana のアーキテクチャを参照アーキテクチャと位置づけ，SOA システムにおける横断的関心事を整理することで，非機能特性を考慮した SOA システムのシステムアーキテクチャを定義する．このシステムアーキテクチャに基づいてアプリケーションアーキテクチャを設計する手順を示す．

一方で我々は，SOA アプリケーションにおいての機能特性は状態遷移機械の集合として実現できることを確認している．これに基づき，非機能特性を付加することを考えた．非機能特性は Georgakopoulos らの研究 [2] を基に分類整理することで，横断的関心事として取り扱いアスペクトとしてモジュール化することにした．ATM 監視システムの設計に今回提案するシステムアーキテクチャを適用することで，システムアーキテクチャの有用性と，システムアーキテクチャを利用することによるアプリケーションアーキテクチャ設計の支援が可能なことを確認した．

2　背景技術と先行研究

本研究の背景技術である SOA とアスペクト指向について以下で説明する．また，Induruwana が提案している ASIL についても概要を説明する．

2.1　SOA

SOA はシステムを機能単位でモジュール化するアーキテクチャ技術である．この機能単位のモジュールをサービスといい，SOA システムは複数のサービスを連携することで構成される．連携するサービスを組み替えることでシステムの機能変更に柔軟に対応可能である．

2.2　アスペクト指向

アスペクト指向とは横断的関心事をアスペクトとして分離するモジュール化技術である．分割したモジュールを結合することをアスペクトを織り込むという．アスペクトを織り込む箇所はポイントカットで指定し，指定されたタイミングになるとアドバイスを実行する．

非機能特性は一般的にシステムを構成する機能特性に横断しているので，アスペクトととして分離することが適当である．非機能特性をアスペクトとして分離すると，実現したい非機能特性を変更する場合や，非機能特性の実現アルゴリズムを変更する場合は，変更箇所をアスペクトのみに局所化できる．このようにアスペクトを組み替えることでシステムの構成変更に柔軟に対応可能となる．

2.3　Application Service Integration Layer

Induruwana は，SOA システムにおける横断的関心事をアスペクト指向を用いることで分離することを提案している [1]．この研究では，階層アーキテクチャ上に横断的関心事を実現する ASIL を設けている (図 1)．

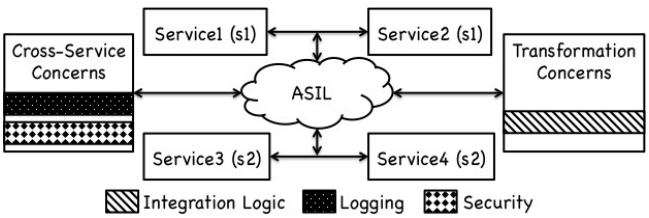

図 1　ASIL の概要 ([1] より引用)

2.4　SOA システムへの状態遷移機械の適用

本研究室では，組み込みシステムを設計する際に，システムの振る舞いを状態遷移機械の集合として実現している．SOA システムを設計する際も同様に，システムの振る舞いを状態遷移機械の集合として実現する方法を適用して，SOA システムを開発することに成功している．システムを構成する Web サービスや Web アプリケーションなどのコンポーネントは，すべて状態遷移機械で実現されており，状態遷移機械間の通信はすべてイベントとして表現される．各状態遷移機械はイベントキューを持っており，イベントを受理すると状態遷移を起こし指定の

アクションを実行する．システム内のコンポーネントへのアクセスはイベントを用いるのでインタフェースは統一されている．

3　非機能特性の整理分類

Georgakopoulos らの研究 [2] の定義に基づき，SOA システムに一般的に求められる非機能特性を整理しアスペクトとして抽出する．Georgakopoulos らの研究は一般的に認められているので，SOA システムに求められる非機能特性の根拠とする．この研究では，次の 11 つの非機能特性が SOA システムに求められていると定義している．

- Service Communication
- Dynamic Connectivity
- Endpoint Discovery with Quality of Service
- Integration
- Message Transformation
- Reliable Messaging
- Topic/Content-Based Routing
- Security
- Long-Running Process and Transaction
- Management and Monitoring
- Scalability

4　非機能特性の整理

Georgakopoulos らが定義する非機能特性を整理する．非機能特性ごとにどのような関心事が内在するか考慮して，アスペクトを抽出する際に参考にする．

Service Communication　この非機能特性は，通信プロトコルを意識せずに通信できることである．また，必要な際に異なる通信プロトコルに応じてサービスを変形できることである．通信プロトコルを意識せずに通信できることを Communication Protocol Aspect として抽出する．提案するシステムアーキテクチャに基づいたシステムでは，Communication Protocol Aspect が通信プロトコルへのアクセス方法を統一しているので，通信プロトコルに応じてサービスを変形する必要はない．

Dynamic Connectivity　この非機能特性は，サービスごとに個別の API やプロキシを使用することなく，動的に複数のサービスに接続できることである．提案するシステムアーキテクチャでは，サービスの呼び出しが状態遷移機械間のイベント通知と定義され，サービスのインタフェースが統一されるので，サービスごとの個別の API やプロキシを使用することはない．動的にサービスへの接続を可能にする．

Endpoint Discovery with Quality of Service　この非機能特性は，実行時にサービスを検索して最適のエンドポイントを選択することである．サービスの情報を保持し実行時の検索を可能にする Service Registry Aspect と，複数の候補から最適のエンドポイントを選択する Routing Aspect を抽出する．

Integration　この非機能特性は，異機種環境のサービスやレガシーシステム，パッケージアプリケーションなどと統合した際にうまく連携できることである．統合の段階は，アプリケーション統合，プロセス統合，情報統合，ポータル統合の 4 段階ある．アプリケーションやプロセスを統合する際は，統合する際のロジックやサービスを変形する Service Integration Aspect を抽出する．データフォーマットの統合は Data Integration Aspect がおこない，システム上に配置されたデータへのアクセスは Deployent Aspect を介しておこなう．またシステムの構成からユーザインタフェースを分離して User Interface Aspect を抽出する．

Message Transformation　この非機能特性は，異なるサービス間のメッセージモデルやデータ方式を変換可能にすることである．提案するシステムアーキテクチャに基づくシステムでは，サービスによってメッセージモデルやデータ方式は変わらない．メッセージの変換が必要となるのは，通信プロトコルを利用する際である．通信プロトコルに依存した形式にメッセージを変換する必要があり，この変換処理を Message Transformation Aspect として抽出する．提案するシステムアーキテクチャに基づかないシステムと通信する場合は Data Ingetration Aspect がメッセージの変換をおこなう．

Reliable Messaging　この非機能特性は，通知したメッセージが確実に相手に届けられることである．サービスの要求メッセージを待ち行列を用いることで確実に宛先に通知することができる．各状態遷移機械は Event Queue を持ち，アクション実行後に Event Queue から次のイベントを受理することによって動作する．この処理を Concurrent Aspect として抽出する．

Topic/Content-Based Routing　この非機能特性は，メッセージの配信先をトピックやメッセージの内容を基に動的に決定することである．トピックベースの配信は，配信希望者が関心のあるトピックを表明することで，メッセージを通知する際に配信希望者のみにメッセージが通知される．コンテンツベースの配信は，トピックを利用せずにメッセージの内容を基にメッセージの通知先を決定する．Routing Aspect の実現の一例として Topic-Based Routing Aspect と Content-Based Routing Aspect を抽出する．

Security　この非機能特性は，ユーザ認証やメッセージの暗号化をすることである．ユーザ認証することを Authentication Aspect として抽出し，メッセージの暗号化や復号化を Encrypted Message Aspect として抽出する．

Long-Running Process and Transaction　この非機能特性は，同時に発生したトランザクションの副作用を分離したり，プロセス障害からの復旧をサポートしたりすることである．

Management and Monitoring　この非機能特性は，管理機能としては動的な負荷の分散やシステムダウン時のフェイルオーバー，クライアントとサービスインスタ

ンス間の位相的，地理的な親和性を実現すること，また監視機能としてはサービス活動の追跡やサービス運用時の統計データを集計して提供することが挙げられる．付加分散を実現するメッセージの配信を Load Distribution Routing Aspect として抽出し，インスタンスの位相的，地理的な親和性の実現に Deployment Aspect を抽出する．また，サービス活動の追跡は Logging Aspect として抽出する．

Scalability　この非機能特性は，システムを拡張した際にも正常に動作することである．状態遷移機械をステートレス化することにより，システムの拡張に柔軟に対応可能になるので，Stateless Aspect を抽出する．

4.1　非機能特性の分類

整理した非機能特性とアスペクトの関連を表 1 にまとめる．

表 1　非機能特性とアスペクトの関連

非機能特性	アスペクト
Service Communication	Communication Protocol Aspect
Dynamic Connectivity	Service Registry Aspect
Endpoint Discovery with Quality of Service	Routing Aspect
Integration	Service Integration Aspect Data Integration Aspect User Interface Aspect
Message Transformation	Message Transformation Aspect
Reliable Messaging	Concurrent Aspect
Topic/Content-Based Routing	Topic-Based Routing Aspect Content-Based Routing Aspect
Security	Authentication Aspect Encrypted Message Aspect
Long-Running Process and Transaction	Error Handling Aspect
Management and Monitoring	Load Distribution Routing Aspect Deployment Aspect Logging Aspect
Scalability	Stateless Aspect

5　システムアーキテクチャ

システムアーキテクチャを状態遷移機械の集合に非機能特性をアスペクトとして織り込むことで定義する．

5.1　システムアーキテクチャの設計

4.1 節で整理した非機能特性を実現するアスペクトを基にシステムアーキテクチャを定義する．SOA システムに求められる機能特性の振る舞いを状態遷移機械の集合としてモデル化する．各アスペクトを織り込んだシステムアーキテクチャは図 2 のようになる．状態遷移機械は Push 型で通信をおこなうので，システムアーキテクチャの図はある状態遷移機械が他の状態遷移機械にイベントを通知する場合に前節で分類したアスペクトがどのように織り込まれるかを示している．また，Error Handling Aspect と Logging Asepct は自身以外のすべてのコンポーネントに対して織り込まれるので記述していない．

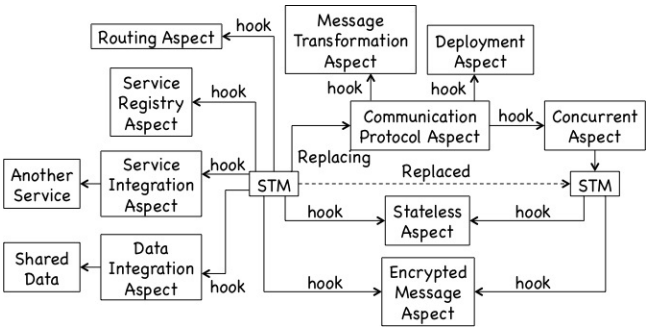

図 2　システムアーキテクチャ

5.2　コンポーネントの意味付け

5.1 節で定義したシステムアーキテクチャの各アスペクトの構造を定義する．主なアスペクトの責務や構成するコンポーネント，ポイントカットを説明する．

Communication Protocol Aspect　は，状態遷移機械間の通信に用いるプロトコルと，その通信プロトコルに依存するメッセージ形式への変換を行なう．このアスペクトの構成は図 3 のようになる．状態遷移機械が通信プロトコルを意識せずにメッセージを送信することができるようにインタフェースを統一する．ポイントカットはメッセージ送信である．

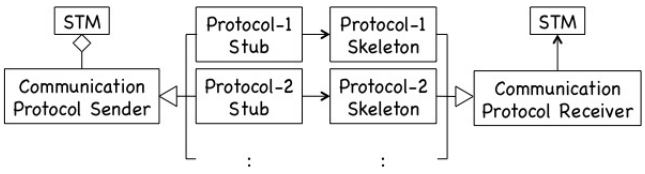

図 3　Communication Protocol Aspect の構造

Service Registry Aspect　Service Registry Aspect は，サービスの名前とサービスのインスタンスの関連を保持する Service Registry を管理する．このアスペクトの構成は図**??**のようになる．Service Registry はサービス名を基にサービスのインスタンスを返す．よってサービスを利用者はサービス名を知っているだけで，サービスのインスタンスを特定できるようになる．ポイントカットはインスタンスの特定である．

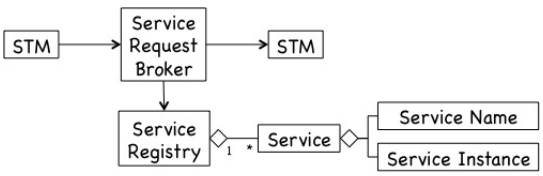

図 4　ServiceRegistry Aspect の構造

Routing Aspect

Dynamic Routing Aspect は，メッセージの通知先を動的に決定する．このアスペクトの構成は図 5 のようになる．状態遷移機械がイベントを通知する際に，Service Request Broker がサービスの情報を基に通知するサービスを特定する．この通知先特定のアルゴリズムは切り替え可能に

なっており，トピックベースやコンテンツベースの通知先決定アルゴリズムを選択できる．ポイントカットはメッセージ通知先の決定である．

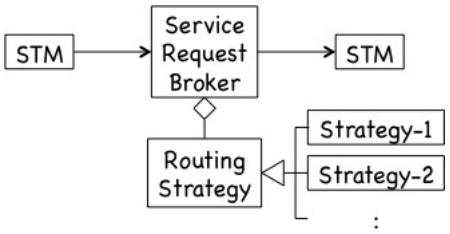

図 5 Routing Aspect の構造

Service Integration Aspect は，仕様の異なるサービスへのメッセージ通信を仲介する．Service Integration Adapter は異なるサービスへのメッセージ通信を仲介し，必要に応じてメッセージ形式の変換をおこなう．ポイントカットは他のサービスへのメッセージ送信である．

6 考察

6.1 システムアーキテクチャの有用性

システムアーキテクチャはシステムに求められる非機能特性によってアスペクトを組み替えることで，様々なシステムに対応可能である．また，アスペクトのアドバイスのアルゴリズムを変更することで，アルゴリズムの変更にも柔軟に対応可能である．よって，システムアーキテクチャは有用であると言える．

例えば Routing Aspect ではメッセージの通知先決定アルゴリズムをトピックベースやコンテンツベース，負荷分散を考慮したルーティングに切り替えることができる (図 6)．他にも Security Aspect の暗号化アルゴリズムや，Logging Aspect のログをとる箇所なども切り替えることが可能である．

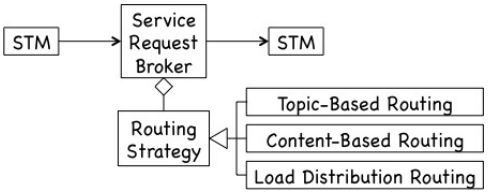

図 6 アルゴリズムの切り替えの例

6.2 状態遷移機械適用の有用性

システムアーキテクチャは機能特性を状態遷移機械の集合として定義している．アプリケーションアーキテクチャの機能特性を設計する際は，機能特性の実現に必要な状態遷移機械を設計するだけでよいので，アーキテクチャ設計が容易になると考える．

機能特性を状態遷移機械の集合としてモデル化できているので，モデル検査やコードの自動生成といった既存の技術を適用することが可能となる (図 7)．モデル検査は状態遷移機械のモデル上での誤りの検出を行なうことができ，コードの自動生成はコード記述の労力が削減できる．よってシステム開発自体の支援にもなると考える．

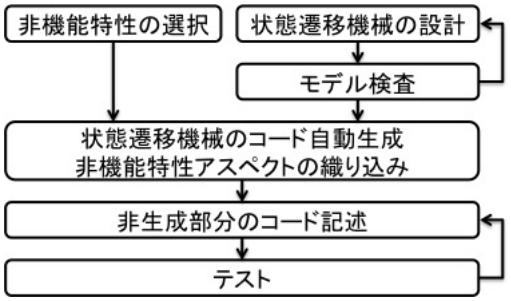

図 7 状態遷移機械適用の概要

6.3 アプリケーションアーキテクチャの設計支援の妥当性

今回提案するシステムアーキテクチャを基に ATM 監視システムのアプリケーションアーキテクチャを設計した．アプリケーションアーキテクチャを設計する際は，保証すべき非機能特性のアスペクトを選択，アドバイスのアルゴリズムの選択，状態遷移機械の設計などをおこなうのみでよい (図 8)．非機能特性について整理されているので，アプリケーションアーキテクチャ設計の支援がなされていると考える．

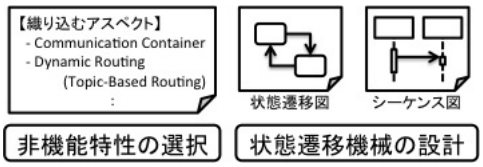

図 8 アプリケーションアーキテクチャ設計の概要

7 おわりに

本研究では SOA システムのアプリケーションアーキテクチャの設計を支援することを目的として，アスペクト指向に基づくシステムアーキテクチャを定義した．このシステムアーキテクチャでは，SOA システムにおける横断的関心事を整理することで，非機能特性をアスペクトとして抽出している．提案するシステムアーキテクチャはアスペクトを組み替えることで異なる非機能特性が求められる SOA システムに適用可能であると考えられる．

参考文献

[1] C. D. Induruwana, “Using an Aspect Oriented Layer in SOA for Enterprise Application Integration,” *ICSOC*, pp. 19-24, 2005.

[2] D. Georgakopoulos, and M. P. Papazoglou, *Service-Oriented Computing*, The MIT Press, 2009.

[3] M. P. Papazoglou, and W. Heuvel, “Service-Oriented Design and Development Methodlogy,” *IJWET*, no. 4, pp. 1-17, 2006.