XML Consortium

SOAのパターン -ESBを中心に-

XMLコンソーシアムSOA部会

日本IBM　天野富夫
JIEC　坂下秀彦
日本オラクル　森本信次
ウルシステムズ　林浩一

XML Consortium

アジェンダ

- はじめに
- SOAパターンとESB
- Enterprise Integration PatternsとESB
- SOAのソフトウェア構成とESB
- まとめ

XML Consortium

© XML Consortium

はじめに

- SOA(Service-Oriented Architecture)
  - システムを，業務視点の機能（サービス）の集合と捉え，ビジネス環境の変化に対して，迅速にサービスを組み換えることで柔軟に対応するシステム構築方針
- ESB(Enterprise Service Bus)
  - Webサービス、メッセージングミドルウェア、インテリジェントなルーティングと変換の機能を利用した新しいアーキテクチャ(Gartner 2002)

SOA部会のアプローチ

- 抽象的な定義からSOAの具体的なイメージを導き出すには？
  - 作ってみるのが一番だが..
    - 時間とスキルが要る
    - 他者に説明するには細かすぎる記述
  - SOA実現で使われるアーキテクチャやデザインのパターンのレベルで議論する
    - 細かすぎず粗すぎず議論するのに適切なレベル
    - ESBもSOA実現で用いるパターンの一種として扱う

SOAパターンとESB

XML Consortium

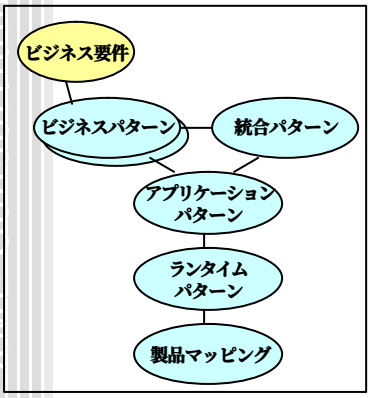
SOAのパターン(前回発表)

- パターンとは
  - システムを実現する上で繰り返し発生する要件や課題に対して解決策をまとめたもの
  - システムのどのレイヤーに注目するかによってさまざまな粒度のパターンが存在する
  - **あるパターンの実現に別のパターンが使われる**

© XML Consortium

SOAと関係するパターン

- IBMのPatterns for e-business



■**ビジネスパターン**
ユーザー、企業、データ間のインタラクションを定義

■**統合パターン**
複数のビジネスパターンを結びつける

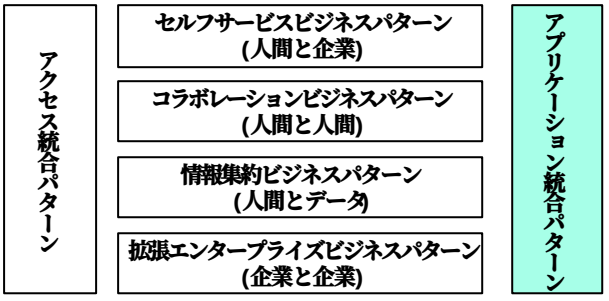
■**アプリケーションパターン**
ビジネスパターンや統合パターンの中のコンポーネントやデータがどのようにインタラクションを行なうかを記述した概念的なレイアウト

■**ランタイムパターン**
アプリケーションパターンをサポートする論理的なミドルウェアの構造

■**製品マッピング**
各ランタイムパターンの検証済みテスト済みのソフトウェア実装

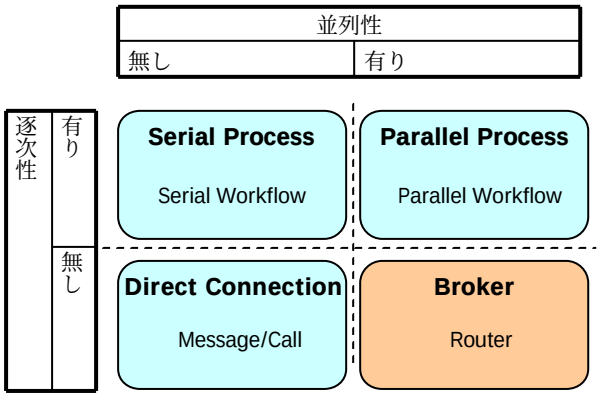
アプリケーション統合パターン

- SOAと関わるPatterns4e-Businessパターン
  - ビジネスパターンの統合のためのコンポーネントやコンポーネント間のインタラクションや論理的な配置を定義



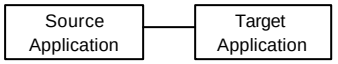
アプリケーション統合パターンの分類

- インタラクション４つのパターン

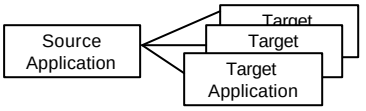


参考 :インタラクションの種類

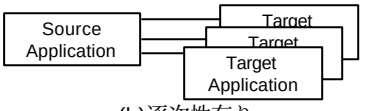
- Source Application(呼出し元)とTarget Application(呼出し先)の関係による



(a)インタラクションの基本形式



(b)並列性有り



(b)逐次性有り

Brokerパターン

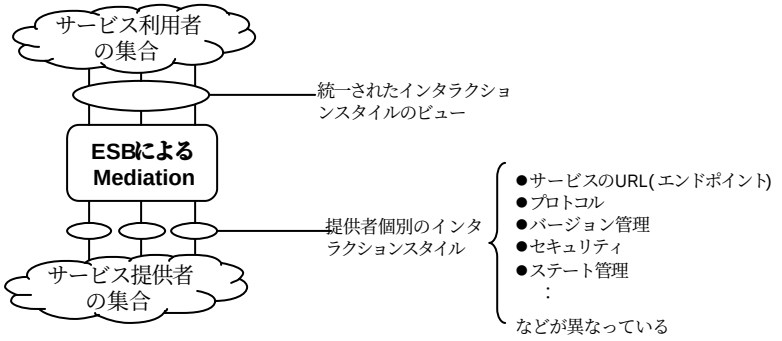
- 1対多の1回限りのインタラクションによる統合
  - メッセージのルーティングや分解/合成を行なうBrokerコンポーネント
  - 複数のターゲットのうち1個だけにメッセージを送るRouter variationがある
- Brokerパターンの実現とESBパターンは関係が深い

ESBパターン

- 課題：サービス同士をスムーズに連携させたい
  - サービス利用者と提供者の間の不整合/ギャップの解消
    - 個別の"利用者-提供者"の組ごとの実装は手間がかかる
  - ロケーションやメッセージフォーマットなど一方が変化したとき相手に影響を与えないようにする
- 解決：サービス提供者と利用者を"バス"経由で接続する
  - ギャップの解消,ルーティング(1対1、1対多)の機能の機能をバスに集約する
  - プログラミングではなく設定変更で変化に対応する

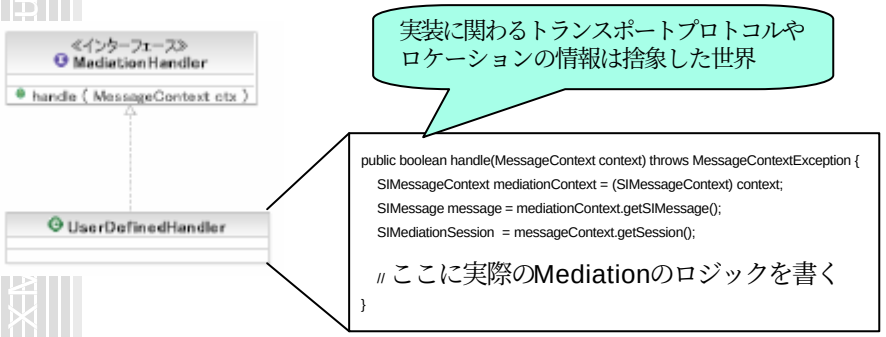
ESBパターンが提供するMediation

- サービスの使い勝手の統一
  - 利用者と提供者の間のギャップ/不整合の解消
  - 一方が変化したとき相手に影響を与えない
- プログラミングではなく設定変更で変化に対応する



ESBパターンの実現：Mediationフレームワーク

- ESBはさまざまなMediationロジックを作るための
  フレームワークを提供する



Mediationのメリット

- サービスのロケーション(エンドポイント)の仮想化
  - サービスのアドレスはプログラムから切り離す
- 外部の設定情報によるルーティング(経路決定)
  - サービスの実ロケーションの情報を一括管理
- メッセージ内容によるルーティング
  - e.g メッセージフォーマットのバージョンをチェックして送り先を変える
- メッセージ内容の変換サービス
- メッセージの分配(Disaggregation)と集約(Aggregation)
  - e.g 送り先が1個増えたとき

XML Consortium

Enterprise Integration PatternsとESB

© XML Consortium

Enterprise Integration Patterns

- G.Hohpe & B.Woolf, Enterprise Integration Patterns, Addison-Wesley
- 非同期メッセージングによるアプリケーション(サービス)統合のためのパターン集
  - Messaging Systems 6パターン
  - Messaging Channels 9パターン
  - Message Construction 9パターン
  - Message Routing 12パターン
  - Message Transfomation 6パターン
  - Messaging Endpoints 11パターン
  - System Management 8パターン

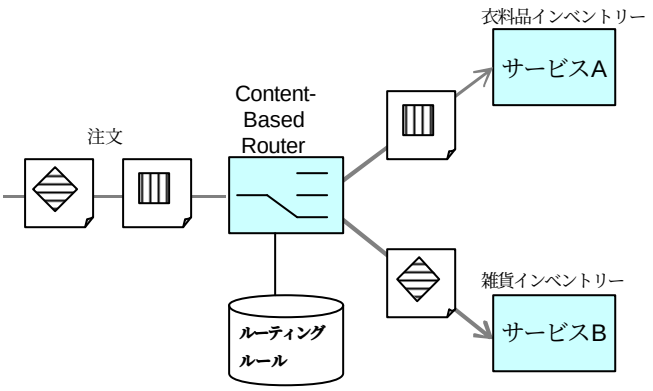
Why Enterprise Integration Patterns?

- SOAとはサービスの統合で物事を実現する
- サービスはメッセージのやりとりによって起動される
- メッセージのやりとりを柔軟に制御する必要がある
- メッセージのやりとりを柔軟かつ安全に設計するためのパターンがEnterprise Integration Patterns
- ESBはEnterprise Integration Patternsの特にMessage Routing関連のパターン(Patterns4e-businessのBroker/Routerパターン)の実現に役立つ

Message Routingパターン(1)

名称	課題
	解答
	SOA/Webサービスによる実現
Content-Based Router	論理的に単一のロジックの実装が複数の物理的システムに分散している状況をどう扱うか?
	Content-Based Routerが内容に応じてメッセージを正しい受信者(サービス)にルーティングする。
	•ESBのMediationフレームワークによる実装 •XMLによるメッセージの記述/XPathによる参照
Message Filter	サービスが関係ないメッセージを受信するのを避けるには?
	Message Filterがある基準に従って関係の無いメッセージをチャネルから除去する。
	•ESBのMediationフレームワークによる実装 •XMLによるメッセージの記述/XPathによる参照

Content-Based Routerパターン



Message Routingパターン(2)

名称	課題
	解答
	SOA/Webサービスによる実現
Dynamic Router	ルーティングの決定が宛先となるサービスの状態に依存している
	ルーティング対象のサービスから設定用メッセージを受信してルーティングルールを再構成する。
	•ESBのMediationフレームワークによる実装 •XMLによるメッセージの記述/XPathによる参照 •ルーティング設定用Webサービスの提供
Recipient List	動的な受領者リストにあるサービスにのみメッセージをルーティングしたい
	受領者リストを生成またはメッセージから読みこんで対象者にのみルーティングする。
	•ESBのMediationフレームワークによる実装 •XMLによるメッセージの記述/XPathによる参照

Message Routingパターン(3)

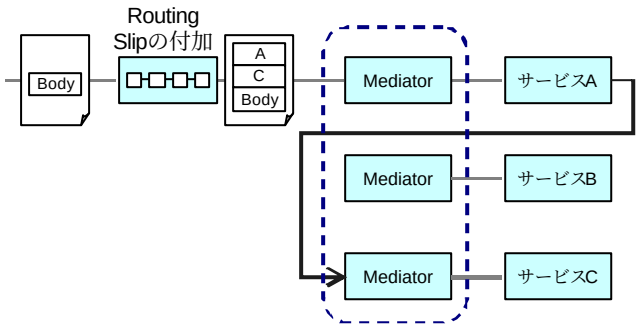
名称	課題
	解答
	SOA/Webサービスによる実現
Splitter	別々のやりかたで処理されるべき複数の要素を含んでいるメッセージをどのように扱うか？
	メッセージを個々の要素ごとに複数のメッセージに分割する。
	•ESBのMediationフレームワークによる実装 •XMLによるメッセージの記述/XPathによる参照 •XSLT/DOM/SDO によるメッセージの生成
Aggregator	個別のしかし関連があるメッセージ群をまとめて処理するにはどうするか？
	ステートフルなフィルタを用意してメッセージ群を1つにまとめる。
	•ESBのMediationフレームワークによる実装 •XMLによるメッセージの記述/XPathによる参照 •XSLT/DOM/SDO によるメッセージの生成

Message Routingパターン(4)

名称	課題
	解答
	SOA/Webサービスによる実現
Routing Slip	複数の処理過程を連続して通るメッセージの処理順が実行に一部変わる場合の対応
	処理順を記述したRouting Slipをメッセージに付与し、受信したサービスがSlipを参照して処理のスキップ等を行なう。
	•ESBのMediationフレームワークによる実装 •WS-RoutingによるRoting Slipの記述 •XSLT/DOMによるメッセージの変更 •XMLによるメッセージの記述/XPathによる参照
Process Manager	複数の処理過程が設計時に決まっておらずシーケンシャルでもない場合。
	次に処理すべきサービスを決定しプロセスの状態を管理するProcess Managerを導入する。
	•BPEL4WSエンジンによる実装

Routing Slipパターン

- 直列に実行される処理の順序を実行時の条件に応じて変更する



●Routing Slipの内容に従って経路を決定

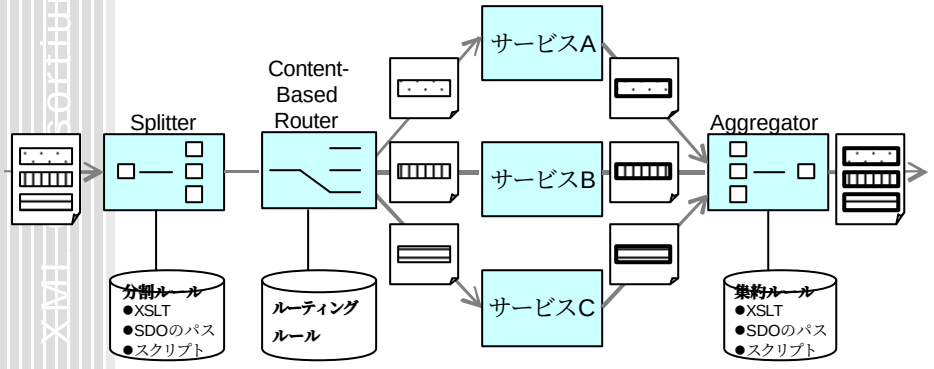
●実体は一つで汎用的に使える

Message Routingパターン(5)

名称	課題
	解答
	SOA/Webサービスによる実現
Composed Message Processor	メッセージが別々に処理されるべき複数の要素を持っているときの全体のメッセージの流れをどのように制御するか？
	メッセージを分割し、それぞれを適切な宛先にルーティングし、応答を1つにまとめる。
	•Splitter,Content-Based Router, Aggregatorの組合せによる。
Scatter-Gather	メッセージが複数のサービスに送られそれぞれが応答してくるとき全体のメッセージの流れをどのように制御するか？
	メッセージを複数のサービスにブロードキャストし応答を1つのメッセージに集約する。
	•Recipient ListまたはPublish-Subscribe ChannelとAggregatorの組合せによる。

Composed Message Processorパターン

- Splitter、Router、Aggregatorの組合せ



Message Routingパターン(6)

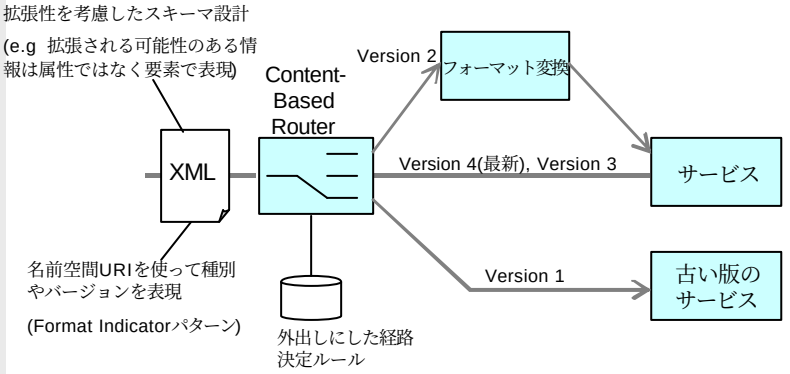
名称	課題
	解答
	SOA/Webサービスによる実現
Resequencer	関連するしかし順番の狂ったメッセージ群を正しく並べなおすには？
	ステートフルなフィルタを用意してメッセージを並べなおす。
	•ESBのMediationフレームワークによる実装 •XMLによるメッセージの記述/XPathによる参照
Message Broker	メッセージの宛先をサービスから分離し集中管理したい。
	メッセージの中継を行なうブローカーを用意する(Patterns4e-businessのブローカー)。
	•ESBの抽象化されたendpoint(destination)を用いて宛先を指定する。

Message Transformationパターン(一部)

名称	課題
	解答
	SOA/Webサービスによる実現
Envelope Wrapper	ヘッダのフォーマットなどが決まっているメッセージングのやりとりに既存のシステムを参加させるには？
	既存システムのメッセージをエンベロープでラッピングして流す。
	•ラッパー/アンラッパーをESBのMediationフレームワークにより実装 •XMLによるメッセージの記述/名前空間を用いたボキャブラリの混在 •XPathによる参照/XSLTによるデータの抽出や変換
Cannonical Data Model	連携対象サービスが異なるメッセージフォーマットを採用しているときお互いの間の依存性を最小化するには？
	特定のサービスから独立したCannonical Data Modelに基づくフォーマットを設計し、個別フォーマットとの間で変換を行なう(変換はN2乗ではなくNのオーダーで済む)。
	•Cannonical Data Modelとの変換はESBのMediationフレームワークにより実装 •XSLTによる変換ルールの記述 •Cannonical Data Model 設計時のXMLスキーマのデザインパターンや共通部品(UBLなど)の利用

パターンの適切な組み合わせが変化への対応能力を実現する

- 複数バージョンのサービスへの対応



Enterprise Integration PatternsとESB

- Enterprise Integration PatternsのMessage Routingパターン, Message Transformationパターンの多くはESBのMediationのフレームワークによって実現することができる
- その他XML/Webサービスの技術やベストプラクティスもEnterprise Integration Patternsの実現に役立つ

SOAの ソフトウェア構成とESB

XML Consortium

SOA実現に必要なソフトウェア

- いくつかの観点がある
  - ビジネスプロセスモデリング
  - サービスの設計 ・開発
  - サービスの実装
  - サービスの統合/連携
  - ビジネスプロセスの実行
  - 実行状況の監視

XML Consortium

各社から製品は出揃いつつある

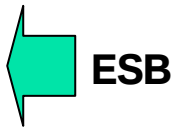
- SOA部会メンバーによる分類(中間結果)
- 今回はサービスの統合/連携に注目

ITベンダー	ビジネスプロセスモデリング	サービスの設計・開発	サービスの実装	サービスの統合/連携	ビジネスプロセスの実行	実行状況の監視
Microsoft	Visual Studio 2005	Visual Studio .NET,.NET Framework,	Windows Server 2003	BizTalk Server 2004	BizTalk Server 2004, Host Integration Server 2004, iWay Adapters for BizTalk 2004(iWay社)	Operations Manager 2005
Oracle	Oracle BPEL Designer/Oracle JDeveloper	Oracle BPEL Designer/Oracle JDeveloper	Oracle Application Server 10g、および J2EE AP Server	Oracle Application Server 10g、および J2EE AP Server OracleAS Integration Adapters	Oracle BPEL Process Manager (BPELエンジン)	BPEL Console

ITベンダー	ビジネスプロセスモデリング	サービスの設計・開発	サービスの実装	サービスの統合/連携	ビジネスプロセスの実行	実行状況の監視
IBM	WBI(WebSphere Business Integration) Modeler	WeSphere Studio Application Developer Integration Edition	WebSphere Application Server	WebSphere Application Server (SI-Bus)、WebSphere Business Integration Adapter	WBI-Server Foundation	WBI-Monitor
BEA	WebLogic Workshop, WebLogic Integration	WebLogic Workshop	WebLogic Server	WebLogic Integration, WebLogic Adapters	WebLogic Integration	WebLogic Integration
Sonic Software	Sonic Workbench	Sonic Workbench	Sonic ESB	Sonic ESB, Sonic Orchestration Server, Adapters for Sonic ESB	Sonic ESB, Sonic Orchestration Server	Sonic Orchestration Server

サービスの統合/連携

- サービス同士を接続・連携させるための基盤
  - 信頼できるトランスポートプロトコル
  - メッセージの仲介、ルーティング、変換
  - 管理・設定機能



- 既存アプリケーションをESBに接続するためのアダプター群

ESBとMOM(Message Oriented Middleware)

- パターンとしてのESB
  - サービス間のMediationを行なう仕掛け
- 製品/Middlewareとしての ESB
  - メッセージ仲介/変換機能
  - 信頼できるトランスポートプロトコルの提供
  - 管理/監査機能

機能の実現にあたっては既存の「Hub&Spoke」の MOM技術をベースにしていることが多い

ESBとは「Hub & Spoke」のMOMのことか

- ESBでは物理的にただ1個のHubにメッセージが集中するわけではない
- 管理・設定は集中制御する
- RoutingやメッセージXML変換をXMLやWS-**などのオープンな標準技術を用いて実現する

まとめ

- ESBパターンの中心はMediationの概念である
- Eneterprise Integration Patternsの定評あるパターンの多くがMediationによって実現可能である
- 「変化に強い」等のSOAのメリットはESBを中心とするさまざまなパターンの組合せで実現される
- 製品としてのESBはその他にもさまざまな機能を持つ。ベースとしているMOMによって実現されているものもある