軽量サービス指向アーキテクチャ設計方法の提案と評価

M2007MM009　池﨑　崇

指導教員：青山　幹雄

1. はじめに

サービス指向アーキテクチャ(SOA)の技術基盤である Web サービスのメッセージ交換は，送信情報が少量のメッセージ交換が多いため，REST(REpresentational State Transfer)[1]の技術要素を適用した軽量メッセージ交換が提案されている[2].

しかし，軽量メッセージ交換の構成要素とその組み合わせによるアーキテクチャの特性(属性)は多様であるため，軽量メッセージ交換に基づくサービス指向アーキテクチャの設計が困難となっている.

本稿では，軽量メッセージ交換の構成要素と属性を構造と挙動の視点から定義したモデルと属性を満たす構成要素を組み合わせたパターン(ABAP：Attribute-Based Architectural Pattern)を定義し，モデルとパターンに基づく軽量 SOA の体系的な設計方法を提案する.

2. 軽量サービス指向アーキテクチャ

2.1. 軽量メッセージ交換と軽量サービス指向アーキテクチャ

本稿では，軽量メッセージ交換に基づくサービス指向アーキテクチャを軽量サービス指向アーキテクチャ(軽量 SOA)と定義する．軽量メッセージ交換とは，通信プロトコルに HTTP を用いたメッセージ記述量の少ないメッセージ交換と定義する.

一般に，Web サービスのメッセージ交換には図 1 に示す SOAP が利用される．しかし，送信情報が少量のメッセージ交換を行う Web サービスの場合，メッセージの複雑さと処理量の増大が問題となる.

この解決方法として，REST の技術要素を適用した軽量メッセージ交換が提案されている．その一例として，図 1 に示す POX がある．この軽量メッセージ交換は SOAP のように標準化されていないため，構成要素が多様であり，その組み合わせにより，多様な特性を取りうる.

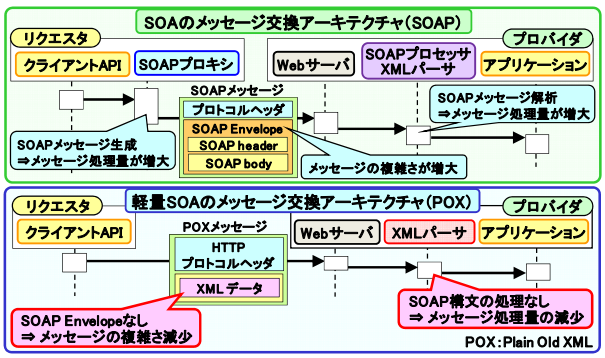

図 1：SOA と軽量 SOA

2.2. 軽量 SOA の設計問題

軽量 SOA の設計には，以下の問題がある.

(1) 軽量メッセージ交換の構成要素の多様化

メッセージ形式やメッセージ交換形態など軽量メッセージ交換の構成要素は多岐に渡るため，それらの組み合わせによる軽量 SOA の特性も多様化する．そのため，軽量 SOA の特性が不明確となり，特定の特性を満たす軽量 SOA 設計が困難となる.

(2) 軽量メッセージ交換の構成要素の相互運用性

軽量 SOA は，軽量メッセージ交換の構成要素の組み合わせによって実現する．構成要素の組み合わせによっては SOA 間で相互運用性が保証されない可能性がある.

3. 軽量 SOA のモデルと開発プロセス

3.1. 問題に対するアプローチ

アーキテクチャの構成要素が持つべき特性を「属性」と定義する．属性は非機能要求と対応付けられるため，非機能要求に対応する属性を満たすように軽量 SOA を設計することで，非機能要求を満たす軽量 SOA が実現可能となる．上記の問題は，属性を考えることで統一的に扱うことが可能である．軽量メッセージ交換の構成要素の多様性を扱うために，軽量メッセージ交換の構造と挙動における構成要素の持つべき特性を属性として捉える．この属性を満たす構成要素を組み合わせて軽量 SOA を設計することで，軽量 SOA 全体で属性を満たすことが可能となる.

本稿では，属性を満たす構成要素の組み合わせを容易とするため，モデルとパターンに基づく設計方法を採る.

3.2. 軽量 SOA のモデルとパターン

軽量 SOA 設計のモデルとパターンを図 2 に定義する[3].

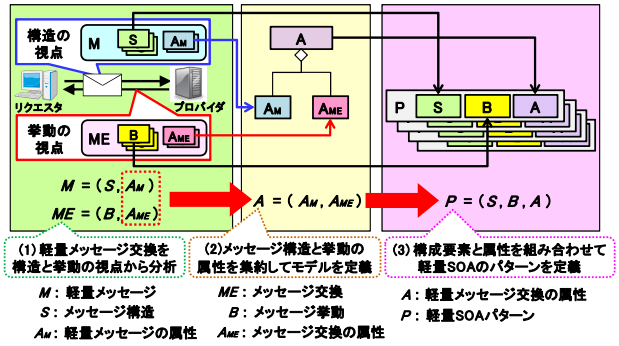

図 2：軽量 SOA のモデルとパターン

(1) 軽量メッセージ交換を構造と挙動の視点から軽量メッセージとメッセージ交換に分類し，構成要素と属性を分析.

(2) 軽量メッセージの属性とメッセージ交換の属性を集

約し，軽量メッセージ交換の属性モデルを定義.

(3) 軽量メッセージ交換の属性に関連する構造と挙動の構成要素を組み合わせて軽量 SOA パターンを定義.

3.3. モデルとパターンに基づく軽量 SOA 開発プロセス

図 3 に軽量 SOA の開発プロセスを示し，軽量 SOA のモデル化とパターン化のプロセスを以下に述べる. モデルとパターンに基づく軽量 SOA 設計のプロセスは後述する.

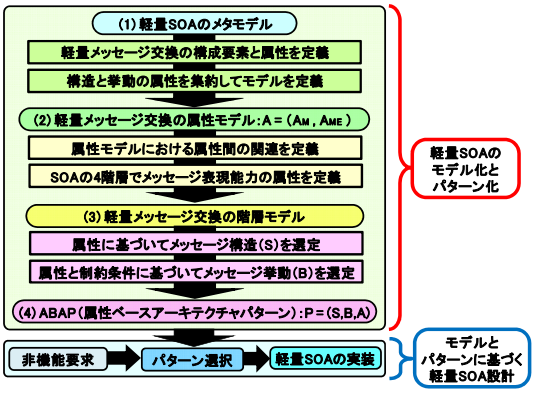

図 3：軽量 SOA の開発プロセス

(1) SOA と軽量メッセージ交換に基づく軽量 SOA の関係を表した軽量 SOA のメタモデルを定義する.

(2) 軽量メッセージ交換の構成要素と属性を構造と挙動の視点から定義し，構造と挙動の属性を集約した属性モデルとその属性間の関連を定義する.

(3) SOA はメッセージ送受信に関与する 4 階層に分割可能であり，メッセージ表現能力の属性を各階層で定義する.

(4) 属性を満たす構造と挙動の構成要素を選定し，それらを組み合わせたパターン(ABAP)を定義する.

4. 軽量 SOA のモデル化とパターン化

4.1. 軽量 SOA のメタモデル

SOA はコンポーネントとコネクタによって構成されるアーキテクチャの拡張であり，サービスとメッセージから構成される．サービスはメッセージ交換により利用される．SOA のメタモデルを拡張し，軽量メッセージ交換を用いた軽量 SOA を定義する(図 4)．また，軽量メッセージ交換の構成要素と属性を組み合わせたパターン(ABAP)を定義する.

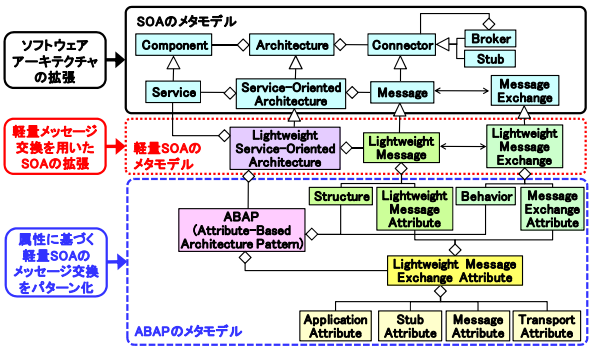

図 4：軽量 SOA と ABAP のメタモデル

4.2. 軽量メッセージ交換の構成要素と属性

本稿では，軽量メッセージ交換の構成要素と属性を構造の視点における軽量メッセージ(M)と挙動の視点におけるメッセージ交換(ME)に基づいて定義する. 軽量メッセージは，メッセージ構造(S)と軽量メッセージ属性(A_M)の対として定義する．メッセージ交換は，メッセージ挙動(B)とメッセージ交換属性(A_{ME})の対として定義する．構成要素であるメッセージ構造とメッセージ挙動の各パラメータを選択することで，軽量メッセージ交換が実現される．また，軽量メッセージ交換の属性(A)には，軽量性に関する属性と同期性などの軽量性に関連しないアーキテクチャ属性がある.

4.3. 軽量メッセージ交換の属性モデル

構造と挙動の視点から抽出した属性を集約し，各属性を構成する属性や要素の関係をモデルとして定義する(図 5). 属性モデルの属性や要素には関連する構成要素が存在し，そのパラメータ選択により属性に差異が生じる. そのため，属性に関連する構成要素のパラメータ選択が重要となる.

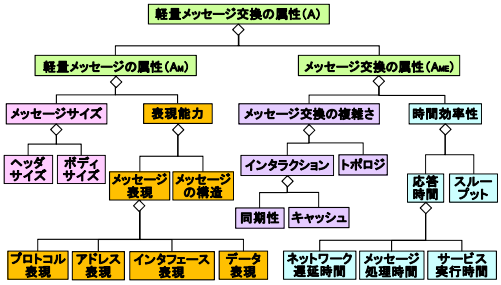

図 5：軽量メッセージ交換の属性モデル

4.4. 属性モデルにおける属性間の関連

図 5 に示す属性モデルの各属性は，他の属性に関与しているが，属性間の依存関係が不明確である．そこで，図 6 に示すように属性モデルの各属性に関連する構成要素から属性間の依存関係を分析し，属性間の関連を定義する.

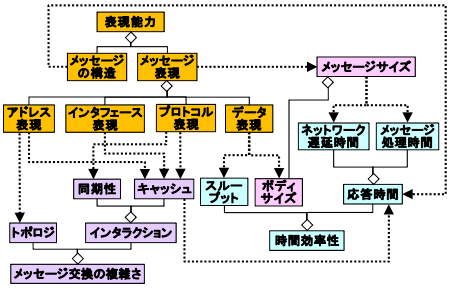

図 6：属性間の関連

4.5. 階層モデルによる軽量メッセージパターンの選定

図 6 より，表現能力の属性は多くの属性に影響を与えるため，表現能力の属性に基づいて軽量メッセージを構成する. SOA のメッセージ交換は 4 階層に分割可能であるため，SOA の階層モデルを用いて各階層の性質[5]に基づいてメッセージ表現能力の属性を定義する(表 1)．また，各階層のメッセージ表現能力の属性に関連する構成要素のパラメータ選択により，表 1 に示す四つの軽量メッセージパ

ターンをメッセージ構造(S)として抽出する[7].

表 1：階層モデルによる四つの軽量メッセージパターン

階層（レベル）	表現能力の属性	構成要素	軽量メッセージパターン			
			SOAP	POX	REST	Ajax
アプリケーション	データ表現	データ型	*	*	単純データ型	*
		メッセージ形式	SOAP +XML	XML	*	*
スタブ	インタフェース表現	インタフェース記述	WSDL	WSDL 2.0 WADL	WSDL 2.0	WSDL
		メソッド	*	*	HTTP メソッド	HTTP メソッド
	メッセージ表現	メッセージの構成	ヘッダ + エンベロープ	ヘッダ + ボディ	ヘッダ + ボディ	ヘッダ + ボディ
メッセージ	アドレス表現	エンドポイント	処理プロセッサ	サービス	サービス	処理プロセッサ
トランスポート	プロトコル表現	プロトコル	*	HTTP	HTTP	HTTP
	インタフェース表現	HTTP メソッド	POST/ (GET)	GET/ POST	*	*

4.6. 制約条件によるメッセージ交換モデルの選定

メッセージ交換の各構成要素のパラメータ選択により，三つのメッセージ交換モデルをメッセージ挙動(B)として抽出する(表 2)．メッセージ交換モデルに対して，表 1 の軽量メッセージパターンを用いる場合，メッセージ交換モデルのパラメータ選択に制約がある．制約条件の例を表 2 に示す．

表 2：メッセージ交換モデルの制約条件

メッセージ交換モデル				軽量メッセージパターン			
トポロジ	インタラクション						
接続形態	同期性	交換方式	交換形態	SOAP	POX	REST	Ajax
直接	同期	リクエスト/レスポンス (R/R) 方式		○	○	○	○
	非同期		Pull 型	非同期 API の利用	非同期通信モデルが必要		プロバイダ側も非同期が必要
			Push 型				○

5. 属性ベースアーキテクチャパターン

属性を満たす軽量 SOA を体系的に設計するために，属性を満たすように構成要素を組み合わせたパターンを属性ベースアーキテクチャパターン(ABAP)と定義する[4].

表 3 に ABAP のスキーマと ABAP の具体例を示す．ABAP は，属性(A)に対する条件と構成要素の条件を満たすように各構成要素のパラメータを選択した軽量メッセージパターン(S)とメッセージ交換モデル(B)の組み合わせによって構成される．

表 3：ABAP スキーマと ABAP の具体例

ABAP スキーマ		ABAP の具体例
項目	内容	
軽量性の属性(A)	軽量性に直接関連する属性	メッセージサイズ
アーキテクチャ属性(A)	軽量性に直接関連しない属性	二者間通信
属性の条件	属性に対する前提条件	メッセージサイズが n バイト以下
構成要素の条件	構成要素に対する制約条件	ブローカの利用不可
メッセージ構造(S)	軽量メッセージパターン	REST パターン
メッセージ挙動(B)	メッセージ交換モデル	直接 / 同期 / R/R 方式 / -
効果	パターンの効果や副作用	HTTP に依存

6. 軽量 SOA 設計方法

上記のモデルとパターンに基づいて軽量 SOA を設計する．軽量 SOA の設計プロセスを図 7 に示す．

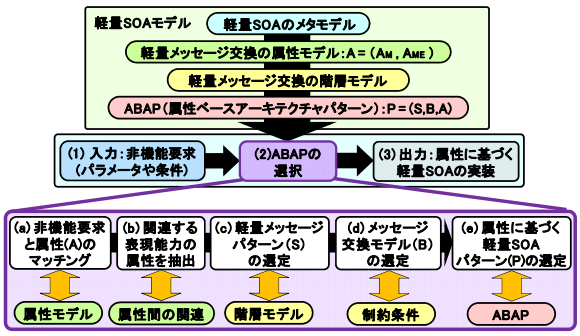

図 7：モデルとパターンに基づく軽量 SOA 設計プロセス

(1)入力として軽量性に関する非機能要求とアーキテクチャ属性に関する非機能要求のパラメータと条件を受け取る．

(2)非機能要求に基づいて ABAP を選択する．ABAP 選択により，軽量 SOA パターンを選定する．この軽量 SOA パターンの選定プロセスをアクティビティ図で示す(図 8).

(a) 非機能要求を満たす軽量 SOA を設計するために，属性モデルを用いて軽量性に関する非機能要求とアーキテクチャ属性に関する非機能要求のマッチングを行い，非機能要求に対応する属性を抽出する(図 5).

(b) 階層モデルにより，抽出した軽量性の属性を満たす軽量メッセージを構成するために，抽出した軽量性の属性に影響するメッセージ表現能力の属性を抽出する(図 6).

(c) 抽出したメッセージ表現能力の属性と非機能要求の条件から軽量メッセージパターンを選定する．非機能要求の条件から，あらかじめ設定した順序関係に基づいて各階層の表現能力に関連する構成要素のパラメータを選択し，軽量メッセージパターンの候補を選定する(表 1).

(d) 選定した軽量メッセージパターンの候補に対して，アーキテクチャ属性と制約条件の両方を満たすメッセージ交換モデルの候補を選定する(表 2).

(e) 選定した軽量メッセージパターンとメッセージ交換モデルを組み合わせることで，非機能要求に対応する属性に基づく軽量 SOA パターンを選定する．

(3)ABAP 選択により，選定した軽量 SOA パターンに基づいて，Apache Axis2 などの実装技術を用いて軽量 SOA を実装する．

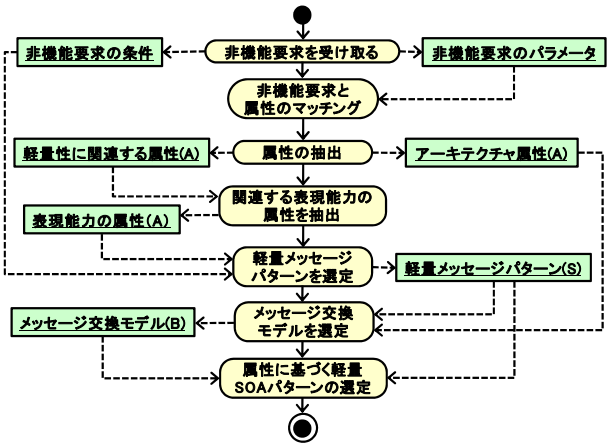

図 8：軽量 SOA パターンの選定プロセス

7. 例題による軽量 SOA 設計方法の評価と考察

7.1. 例題による軽量 SOA 設計

提案した設計方法を評価するために，図 9 に示す例題を用いて軽量 SOA を設計する．非機能要求としてメッセージサイズの削減と非同期メッセージ交換を想定し，本稿で定義したモデルとパターンに基づいて非機能要求を満たす Ajax 非同期 Push 型軽量 SOA パターンを選定した．

また，Apache Axis2 を用いてプロトタイプを試作し，選定した軽量 SOA パターンと他のパターンを比較して，メッセージサイズの削減と送信時間の減少を確認した．

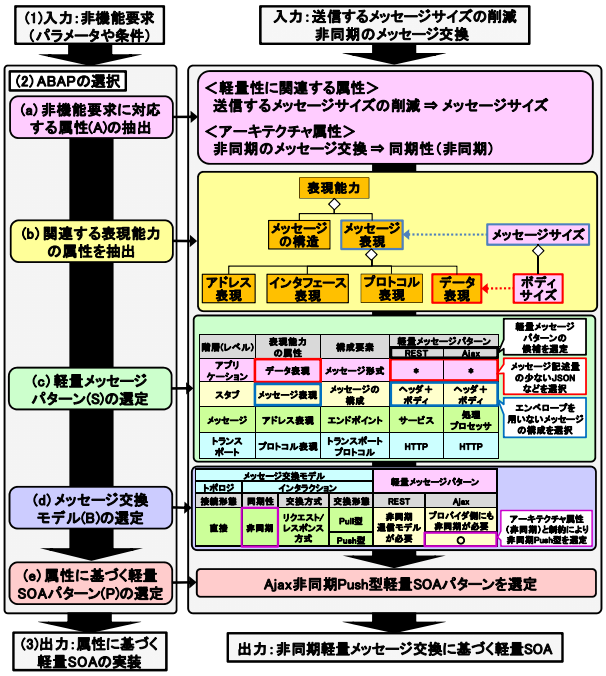

図 9：非同期軽量メッセージ交換に基づく軽量 SOA

7.2. 軽量 SOA 設計方法の評価と考察

本稿では，属性に基づいて軽量 SOA モデルを定義することで，多様な軽量メッセージ交換の構成要素と属性を包括的に扱うことが可能となる．

また，提案した設計方法では，属性を満たす構成要素の組み合わせをパターン(ABAP)として定義し，非機能要求とマッチングした属性のパターンに基づいて軽量 SOA を設計することで，非機能要求を満たす軽量 SOA を体系的に設計することが可能となる．

さらに，試作したプロトタイプにおいて Apache Axis2 の Handler と軽量 SOA パターンの構成要素を対応付けることで，非機能要求を満たす軽量 SOA を体系的に実現することが可能となる．

8. 関連研究との比較と考察

従来の軽量 SOA は，REST と SOAP のアーキテクチャを決定する要素を比較した研究[6]などに基づいて実現しているが，軽量 SOA の設計方法は提示されていないため，軽量 SOA の体系的な設計が困難である．本稿では，属性に基づく軽量 SOA のモデルとパターンを定義し，それに基づく軽量 SOA の体系的な設計方法を提案した．

また，非機能要求に基づくアーキテクチャパターンの選択によるアーキテクチャ設計に関する研究(ABAS)[4]では，パターン選択により，非機能要求(属性)を満たすアーキテクチャ設計が可能となる．しかし，属性の一貫性が保証されていないと考える．本稿では，階層モデルを用いて属性を階層構造と関連付けることで，階層間で属性の一貫性が保証された軽量 SOA の設計が可能となる．

9. 今後の課題

(1) 非機能要求と属性モデルの定量的マッチング方法

メッセージ交換の効率化など抽象度の高い非機能要求に対応する属性を抽出する際に，マッチング精度を上げるための定量的なマッチング方法が必要である．

(2) 各構成要素のパラメータに対する優先度設定方法

属性に基づいて各構成要素のパラメータを選択する際に，パラメータの優先順位を決定するために基準となる定量的な値を設定する方法が必要である．

10. まとめ

本稿では，軽量メッセージ交換の構成要素と属性の多様化による軽量 SOA の設計問題に対して，属性に基づいて定義したモデルとパターンによる軽量 SOA の設計方法を提案し，例題を用いて妥当性を示した．提案した設計方法により，非機能要求に対応する属性を満たすように軽量メッセージ交換の構成要素を組み合わせることで，非機能要求を満たす軽量 SOA の体系的な設計が可能となる．

参考文献

[1] R. T. Fielding , Architectural Styles and the Design of Network-based Software Architectures, PhD Dissertation, University of California at Irvine, 2000.

[2] 池﨑　崇 ほか, REST を用いた軽量 Web サービスアーキテクチャの提案と評価, 情報処理学会第 69 回全国大会論文集(1), No. 3T-6. Mar. 2007, pp. 571-572.

[3] 池﨑　崇 ほか, 軽量メッセージ交換のモデルとパターンに基づく軽量サービス指向アーキテクチャ設計方法の提案, 情報処理学会第 163 回ソフトウェア工学研究会, Mar. 2009 [発表予定].

[4] M. Klein, et al, Attribute-Based Architectural Styles, Technical Report, CMU/SEI-99-TR-022, Oct. 1999.

[5] 森 晃 ほか，非同期メッセージ交換のモデルとパターンに基づく非同期サービス指向アーキテクチャ設計方法，情報処理学会論文誌，Vol. 48, No. 8, Aug. 2007, pp. 2566-2577.

[6] C. Pautasso, et al, RESTful Web Services vs. "Big" Web Services, Proc. WWW '08, Apr. 2008, pp. 805-814.

[7] L. Richardson, et al, RESTful Web Services, O'Reilly, 2007.