

非同期サービス指向アーキテクチャの サービスプログレス可視化アーキテクチャの提案と評価

長澤 伸治^{†1} 上野 佳宏^{†2} 服部 正敏^{†2} 中道 上^{†3} 青山 幹雄^{†3}

南山大学 大学院 数理情報研究科^{†1} 南山大学 数理情報学部 情報通信学科^{†2} 情報理工学部 ソフトウェア工学科^{†3}

1.はじめに

サービス指向アーキテクチャ(SOA: Service-Oriented Architecture)[1]は主としてステートレスで同期型の Web サービスを基盤とする。本稿では Web サービスのプログレス確認可能なアーキテクチャを提案し、プロトタイプにより評価を行い、妥当性を示す。

2. 非同期 SOA の問題点

非同期メッセージモデル[3]の要求/応答型メッセージ交換モデルではブロッキングがないため処理効率が向上する。しかし、コンシューマはリクエストの処理状態を確認できない。これにより、コンシューマのリクエストメッセージがプロバイダに受信されたか確認できない。またプロバイダ内の処理中にエラーを引き起こした場合、コンシューマはエラーを確認できない。

3. 関連研究

Web サービスで、既存のサービスプロバイダを変更せずにポーリングを用いた非同期メッセージ交換の実現方法が提案されている[4]。コンシューマとプロバイダ間にプロキシサーバを実装することにより、コンシューマはプロキシサーバに対してポーリングし、プロバイダからのレスポンスの有無を取得可能になる。

4. サービスプログレス可視化アーキテクチャ

4.1. アーキテクチャの適用対象

要求/応答型のメッセージ交換で、リクエスト送信後のサービスの処理状態をサービスプログレスと定義する。従来の非同期の要求/応答型メッセージ交換では、コンシューマはプロバイダのサービスプログレスを確認できない。本稿では、要求/応答型メッセージモデルを対象とし、サービスプログレスを確認できる可視化アーキテクチャを提案する。

4.2. アーキテクチャの狙い

提案するアーキテクチャは可視化するメッセージ交換パターンを定義することによって、サ

Design Method of Service-Progress Visualization Architecture for Asynchronous Service-Oriented Architecture.
†1Shinji Nagasawa, Graduate School of Mathematical Sciences and Information Engineering, Nanzan University.
†2Yoshihiro Ueno, Masatoshi Hattori, Faculty of Mathematical Sciences and Information Engineering, Nanzan University.
†3Noboru Nakamichi, Mikio Aoyama, Department of Software Engineering, Nanzan University.

ービスプログレスの可視化を実現する。コンシューマはプロバイダのサービスプログレスを確認することによって、LRT (Long-Running Transaction)などの振舞いが予測可能になる。

4.3. ステートフル非同期メッセージアーキテクチャ

提案アーキテクチャの設計プロセスを図 1 に示す。非同期 SOA はブローカを介したメッセージ交換を定義する非同期 SOA 参照モデルに基づき 3 段階のメッセージ交換パターンを構成する。このパターンをサービスの要求、設計条件に基づき選択し、サービスプログレスの可視化アーキテクチャを実現する。このアーキテクチャをステートフル非同期メッセージアーキテクチャ(SAMA: State-full Asynchronous Message Architecture)として実装する。

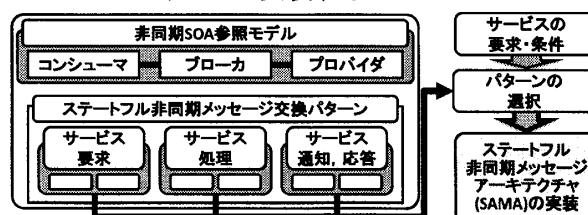


図 1: 提案アーキテクチャの設計プロセス

4.4. 可視化を実現するための仕組み

非同期 SOA 参照モデルは可視化を実現するためにブローカを用いる。ブローカはサービスプログレスを格納するレジストリを持つ。これにより、プロバイダから取得したサービスプログレスを保持し、コンシューマからの要求に応じてサービスプログレスを通知可能となる。

4.5. ステートフル非同期メッセージ交換パターン

ステートフル非同期メッセージ交換はブローカを介してコンシューマとプロバイダ間のメッセージ交換を行う。可視化のためのメッセージ交換方法をアーキテクチャの設計条件の中から、次の 3 段階のパターンの組み合わせで実現する(図 2)。

- (1) サービス要求: リクエストメッセージ到達の確認メッセージの有無
 - 1)In-Only: コンシューマに対する確認不要
 - 2)In-Out: コンシューマに対する確認が必要
- (2) サービス処理: サービス実行時のサービスプログレス通知方法
 - 1)Push: プロバイダが一定周期毎にサービスプログレスを通知

- 2) Pull : プロバイダがサービスプログレスの問合せを受けたときに通知
- (3) サービス通知, 応答レベル : サービスプログレス, レスポンスメッセージの取得方法
- 1) Call Back : コンシューマのコールバックスレッドがレスポンスメッセージを受信
- 2) Polling : コンシューマが問合せを行い, 応答としてレスポンスメッセージを取得

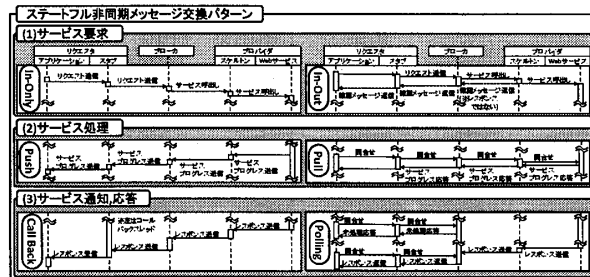


図 2 : 3 段階のメッセージ交換パターン

5. プロトタイプの実装

3 段階のパターンから In-Out, Push, Polling を選択したステートフルポーリングアーキテクチャのプロトタイプを図 3 に示す。要求, 問合せの SOAP over HTTP は Apache Axis2 を用いて実装した。状態, 結果送信は POX over SMTP を用いて行う。POX メッセージを受信しデータベースへのサービスプログレス登録は Maillet 機能を持つメールサーバである Apache James を用いて実装した。サービスプログレスを格納するためのデータベースに MySQL を実装した。

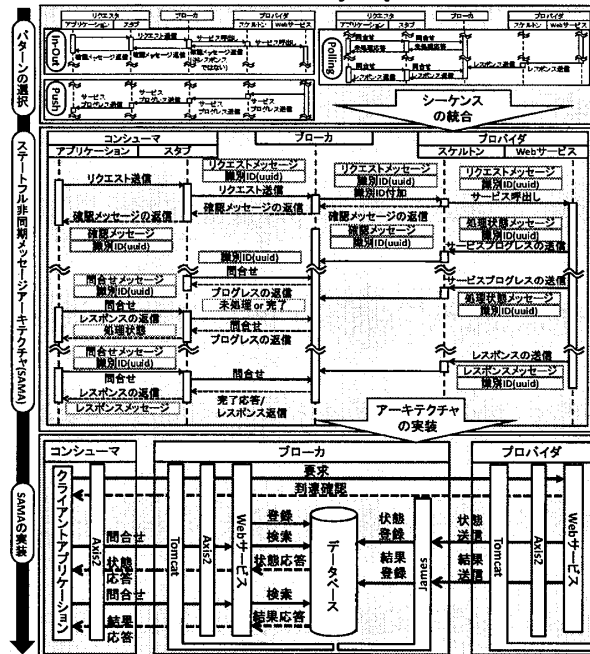


図 3 : ステートフルポーリングアーキテクチャ

6. プロトタイプの検証と評価

プロトタイプの妥当性を以下 2 つの観点から評価した。

6.1. コンシューマのメッセージ交換の複雑さ

プロトタイプはコンシューマ, プロバイダ間で行う非同期メッセージ交換の複雑な処理にブローカを適用した。これにより, コンシューマに複雑な処理を付加せずにサービスプログレスの取得が可能となった。

6.2. 処理のオーバーヘッド

コンシューマ, ブローカ, プロバイダ間のサービスプログレス通知に要するオーバーヘッドを処理シーケンスに沿った時間要素に分けて測定した。各時間要素の定義を図 4 に示す。

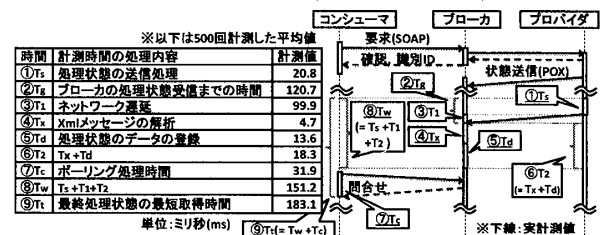


図 4 : サービスプログレスの通知時間

図 4 より, プロバイダのサービスプログレスをコンシューマに通知する時間(T₉)は平均 183 ミリ秒であった。よって, サービスプログレス取得のためのオーバーヘッドは十分小さく, 提案アーキテクチャは, LRT などのプロバイダで長時間保留されるサービスのプログレス可視化に有用である。

7. まとめ

非同期 Web サービスの処理状態をサービスプログレスとして確認できるアーキテクチャを提案した。プロトタイプにより可視化のオーバーヘッドが実用上問題ないことを確認し, 提案アーキテクチャの妥当性を確認した。これは, 非同期 SOA の状態を可視化する拡張ともなっている。

参考文献

- [1] T. Erl, Service-Oriented Architecture, Prentice Hall, 2005.
- [2] 服部 正敏, 上野 佳宏, 長澤 伸治, 非同期 Web サービスの信頼性向上に関する研究, 南山大学 2008 年度卒業論文, 2009.
- [3] 森 晃, 青山 幹雄, 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. 2008, pp. 2567-2576.
- [4] K. Qian, et al., Asynchronous Callback in Web Service, Proc. IEEE SNPD '06, Jun. 2006, pp. 375-380.