SaaS/クラウドサービス間のメッセージ連携方法の提案と評価

2006MI008　BHANDARI Swati　　2006MI210　吉川　千絵

指導教員　青山　幹雄

1. はじめに

ソフトウェアの機能をネットワークを介して提供するSaaS/クラウドサービスに注目が集まっている．　本稿では，異なるSaaS/クラウド間を連携する方法を提案し，その妥当性を評価する．

2. SaaS/クラウド間の連携の現状と課題

2.1. クラウド環境上でのサービス開発環境

クラウド環境で実行できるサービスの種類や規模は，現在急速に拡大している[4]．以下に，主な開発環境を示す．

(1)　Amazon : Amazon S3，Amazon EC2
(2)　Salesforce.com : Force.com
(3)　Google : Google App Engine
(4)　Microsoft : Windows Azure，Azure Services Platform

本研究では，簡単に開発に携わることができ，実際にアプリケーションの開発，実行ができる Salesforce.com が提供する Force.com を開発基盤として用いることにした．

2.2. 研究課題

Force.com の CRM(Customer Relationship Management)を使用し，フランチャイズ店舗管理システムを作成した[2]．作成した際に発見した問題点から以下の研究課題を挙げる．

(1)　アプリケーションの連携が困難
(2)　データベースの値に簡単にアクセス不可

(1), (2)に重点を置き，マッシュアップ技術を用いてアプリケーションの連携を実現する[3]．しかし，異なる SaaS/クラウド間には独自の制約があるため，相互運用性が大きな課題となっていると考える．　相互運用性を図るために，異なる SaaS/クラウド間において，プロトコルやメッセージの標準化を行うことを課題とし，柔軟な連携を目指す．このような連携を SaaS 連携と定義する．

3. SaaS 連携のアプローチ

3.1. SaaS 連携

SaaS 連携は，より高度な利便性の高いサービスを実現するためには必要不可欠である．しかし，各クラウド環境は連携を意識して作られてはいない．SaaS 連携を行うことにより，機能の拡張性や高度化が期待される．よって，ビジネスにおける業務効率の向上やコスト削減につながる．

3.2. 前提条件

異なる SaaS/クラウド間をメッセージを基盤とし，リアルタイムに連携を実現する．

(1)　リアルタイム性
(2)　標準メッセージによるメッセージ連携

3.3. 適用する SaaS 連携

Salesforce.com はデータ連携方式としてバッチ処理とアウトバウンドメッセージの 2 種類を提供している．

アウトバウンドメッセージは，あるシステムやネットワークから外部にメッセージを送信することである．

バッチ処理は，データをファイル形式で一括に送信するのに対し，アウトバウンドメッセージはデータをトランザクション形式でイベントごとに送信する．

本研究では，前提条件に従いアプリケーションをリアルタイムに連携するため，SaaS 連携方式として，アウトバウンドメッセージを用いる．

3.3.1. アウトバウンドメッセージ

トランザクション単位にデータを外部にメッセージとして送信する．Force.com 上のデータ変更を速やかに他システムに送信したい場合に使用する．　このデータを利用者が指定した Web サービスプロバイダに向けて SOAP メッセージとして送付する (図 1)．

図 1　アウトバウンドメッセージの仕組み

4. SaaS 連携アーキテクチャの提案

Web 上で，アプリケーション同士の連携をするために，Web サービスが用いられる．Web サービスの基本アーキテクチャとしてサービス指向アーキテクチャ(SOA: Service-Oriented Architecture)がある[1].

4.1. 提案するアーキテクチャ

提案するアーキテクチャは，SOA を元に，サービスコンシューマとサービスプロバイダ間にアウトバウンドメッセージを使用するアーキテクチャを提案する(図 2).

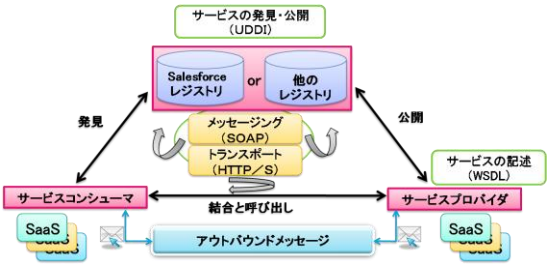

図 2　提案するアーキテクチャ

4.2. SOA に基づいたアウトバウンドメッセージによる SaaS 連携アーキテクチャ

提案アーキテクチャに基づいた，プロトタイプのモデルを図 3 に示す．

サービスプロバイダ A(Salesforce.com)から送信されたアウトバウンドメッセージは Web サーバ(Tomcat)で受信する．次に MySQL にデータを保存し，サービスプロバイダ B(Google)の Google Maps とマッシュアップしサービスコンシューマに表示する．

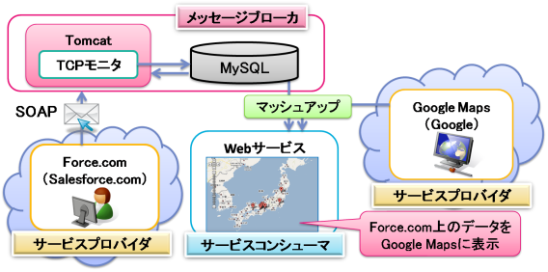

図 3　アウトバウンドメッセージによる SaaS 連携アーキテクチャ

4.2.1. メッセージブローカ

SaaS 連携の際，プロトコルとメッセージを Web サービスに変換するために，メッセージブローカを提案する．

以下にメッセージブローカの役割を示す．

(1) メッセージの変換

(a) Apache Tomcat 上で，Salesforce.com から送信された SOAP/HTTP であるアウトバウンドメッセージから要素を String 型に変換する．

(b) MySQL のデータを XML 形式に変換する．

(c) XML 形式を JSON/HTTP に変換する．

(2) タイミングの調整

Salesforce.com はトランザクション単位でアウトバウンドメッセージを送信する．しかし，データの反映はコンシューマから要求があった際に行われる．よって，タイミングの調整を行うために，MySQL にデータを格納する．

図 4 にアーキテクチャと，各処理を説明する．

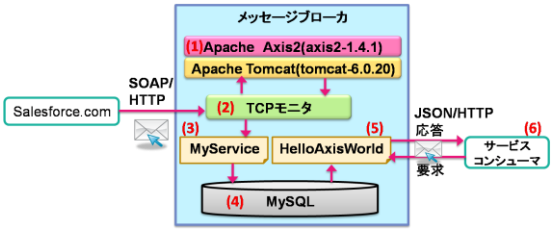

図 4　メッセージブローカのアーキテクチャ

(1) Apache Axis2/Tomcat

Apache のプラグインであり，Axis2 は Web サービスのフレームワークであり，Tomcat は Web サービスを構築するための，Web サーバである．

(2) TCP モニタ

Apache Axis2 のプラグインであり，Web サービスコンシューマと Web サーバ間に割り込み，SOAP メッセージの送受信を確認する．

(3) MyService

Tomcat 上で実行する Web サービスである．ここでは，TCP モニタで受信したアウトバウンドメッセージの SOAP メッセージを解釈し，要素を抽出し，データベースに格納する．

(4) MySQL

SOAP メッセージから抽出したデータを JDBC を介して格納する．

(5) HelloAxisWorld

Tomcat 上で実行する Web サービスである．データベースのデータを XML に変換する．サービスコンシューマからの要求に応じて XML 形式のデータを応答として送る．

(6) サービスコンシューマ

Salesforce.com の取引先データを，Google Maps を利用し，マッシュアップを実現する Web サービスである．

この処理をまとめてシーケンス図で示す(図 5)．

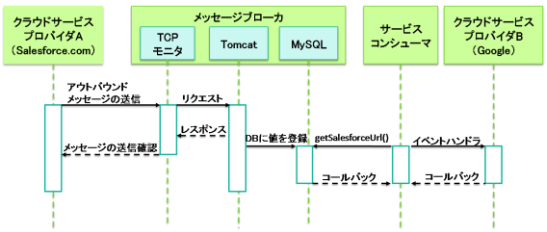

図 5　アウトバウンドメッセージの流れのシーケンス図

5. プロトタイプ

5.1. プロトタイプの仕様

クラウドサービスプロバイダ A(Salesforce.com)の CRM において，アウトバウンドメッセージの送信条件として，取引先情報が新規登録された場合に，アウトバウンドメッセージが送信されるというワークフロールールを設定した．

クラウドサービスプロバイダ B(Google)で Google Maps を利用する．

クライアント上で 2 つの Web サービスをマッシュアップする．

5.2. プロトタイプの実装

プロトタイプの実行環境を以下に示す．

(1) OS：Windows Server 2003

(2) クラウドサービスプロバイダ A：Salesforce.com (Salesforce CRM Winter'10)の CRM

(3) クラウドサービスプロバイダ B：Google の Google Maps

(4) 開発言語：Java（jdk1.6.0_17），JavaScript，HTML，XML

図 6 にプロトタイプの全体像を示し，各処理を説明する．

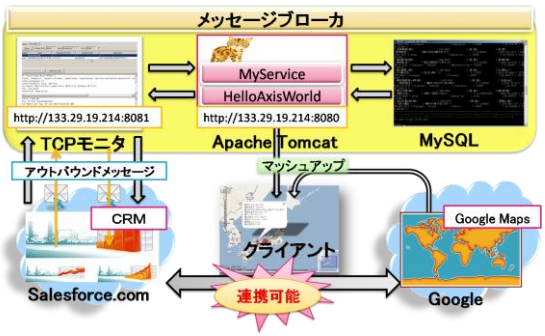

図 6 プロトタイプの全体像

5.2.1. SaaS サービス(Salesforce.com)

Force.com でフランチャイズ店舗管理システムを作成する[5]．アプリケーションビルダーツールを用いてメタデータを設定し，実現した．4 種類の店舗プランを設定した．それぞれに 1 人ずつ店主を設け，店舗を 1 つ配備した．

5.2.2. メッセージブローカ

Force.com で取引先が新規登録された際に送られてくるアウトバウンドメッセージを処理する．アウトバウンドメッセージを送信するエンドポイントを指定する．

メッセージブローカによって，両プロバイダ間のプロトコルとメッセージを標準化した．サービスコンシューマによって，クラウドサービスプロバイダ B(Google)の Google Maps にマッシュアップされ，Salesforce.com と Google の SaaS 連携が確認できる．

(1) Force.com からメッセージの抽出(TCP モニタ)

ポート 8080 で，Tomcat が起動している．TCP モニタは，ポート 8081 を使用し，Web サービスコンシューマと Web サーバ間に割り込み，要求と応答の SOAP メッセージを確認する．

(2) メッセージ変換(Apache Tomcat)

Tomcat 上で MyService と HelloAxisWorld の 2 つの Web サービスを作成した(表 1)．

表 1 プログラムの詳細

サービス名	言語	クラス数	行数
MyService	Java	1	106
HelloAxisWorld	Java	1	115

(3) メッセージの保存(MySQL)

アウトバウンドメッセージから送られた 14 個の項目のデータを 1 つのテーブルに格納した．

5.2.3. サービスマッシュアップ(Google)

取引先マップを作成するために，Google Maps API を取得した．

5.2.4. クライアント

Salesforce.com からのフランチャイズ店舗管理システムの情報と Google Maps を JSON 形式でマッシュアップする．

表 2 に実装の詳細を示す．

表 2 マッシュアップ時のファイルの詳細

サービス名	言語	モジュール数	行数
マッシュアップ	JavaScript	1	106
表示形式	CSS	1	38
表示	HTML	1	19

Google Maps の画面上にクリックするというイベントが発生すると，コールバック関数を利用し，Salesforce.com と Google Maps にコールバックし，現在データベースに登録されているデータの全てを Google Maps 上に反映する．

図 7 と図 8 にプロトタイプのユースケース図とシーケンス図をそれぞれ示す．

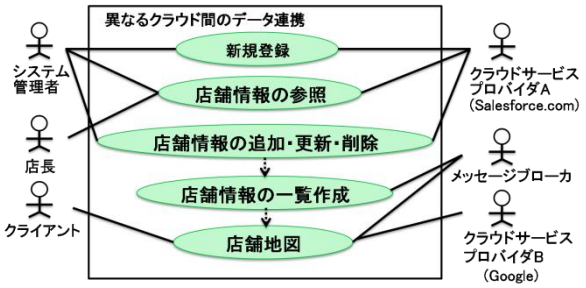

図 7 プロトタイプ全体のユースケース図

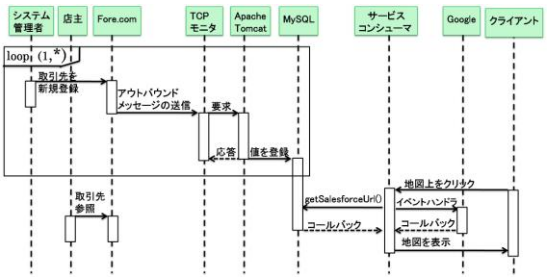

図 8 プロトタイプ全体のシーケンス図

6. 評価と考察

6.1. 提案するアーキテクチャの妥当性

異なる SaaS/クラウド間には，独自の制約が混在し，データの連携を行うことが困難である．

本研究では，SOA に基づき，標準メッセージを使用し，メッセージブローカを介し，SaaS 連携を実現させるアーキテクチャを提案した．メッセージブローカによってプロトコル変換と，メッセージ変換を行い，SaaS 連携を可能とした．

Google Maps 上に Salesforce.com の取引先情報が表示された結果を図 9 に示す．Salesforce.com の取引先情報のデータが，Google Maps 上の情報ウィンドウに表示された．さらに，サービスコンシューマ側でメッセージ(SOAP，JSON)によるデータの相互運用性を確認した．

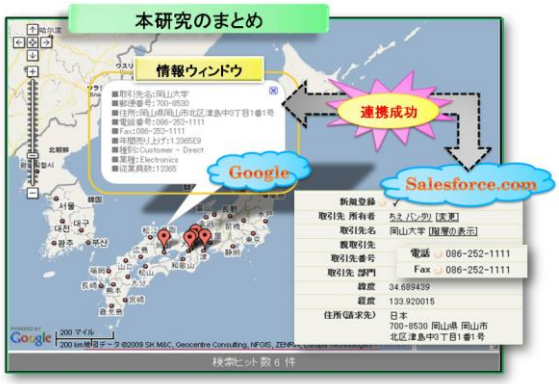

図 9 プロトタイプの実行

(1) メッセージの相互運用性

(a) 標準メッセージ

標準化されている SOAP メッセージを適用することにより，Google 以外のいろいろな Web サービスも，使用可能となる．

(b) 柔軟性

メッセージを使用することによって連携が柔軟にできる．エンドポイント URL を変更するだけで宛先を自由に変更可能となる．

(c) 相互運用性

メッセージブローカを使用することにより，処理が一連的に標準化できる．

(2) タイミングの整合性

アウトバウンドメッセージを送信すると，瞬時にデータベースに値が格納される．また，クライアントが取引先地図において，画面をクリックし，検索を開始すると，ほぼ同時に取引先のデータが地図上に表示される．よって，前提条件である，リアルタイム性が保証されていることが確認できる．

7. 今後の課題

本稿で提案したアーキテクチャは，クラウド環境上のデータを Web サーバやデータベースに保管している．これは，Web を経由してソフトウェアの利用やデータの保管が出来るクラウド環境の利点を活用していない．異なるクラウド環境上の SaaS 連携を容易に可能にするために，SaaS 連携に必要なサーバ，データベース，アプリケーションをサービスとしてクラウド上で提供すべきである．

また，連携の際に，プログラムを利用して双方に合う規約に変更すると，連携のスピードに遅延が発生し，システム全体の複雑さが高度になるため，異なるクラウド間で，基本的な通信規約を統一すべきである．

8. まとめ

本研究では，異なるクラウド環境間における SaaS 連携のために，SOA にアウトバウンドメッセージを適用し，標準メッセージを用いた SaaS 連携アーキテクチャを提案した．

メッセージブローカを介して，プロトコルの変換や，メッセージの変換を行い，標準メッセージを用いることで，連携の柔軟性や相互運用性を示し，提案したアーキテクチャの有効性を示した．

参考文献

[1] 青山 幹雄，サービス指向アーキテクチャの誕生と進化，ソフトウェアエンジニアリング最前線 2008（情報処理学会ソフトウェアエンジニアリングシンポジウム 2008 論文集），近代科学社，Sep. 2008，pp. 9-16.

[2] C. Weissman , Development with the Force.com Platform, Salesforce.com, 2009.

[3] 本田 正純，マッシュアップかんたん AtoZ-マッシュアップで作る Web 秘密基地，シーアンドアール研究所，2007.

[4] 中田 敦，ほか，クラウド大全，日経 BP 社，2009.

[5] Salesforce.com, http://www.salesforce.com/jp