情報処理学会論文誌　Vol. 51　No. 2　503–516 (Feb. 2010)

SOAP-REST サービス連携のための実行制御機能の研究開発

山登庸次[†1]　宮城安敏[†1]　中野雄介[†1]
中村義和[†1]　大西浩行[†1]

SOAP コンポーネントと REST コンポーネントを連携したマッシュアップサービスを，容易に開発するための，サービス実行制御機能を提案し，その有効性を示す．近年，Web 2.0 サービスの 1 つとして，マッシュアップサービスの開発がさかんである．しかし，SOAP と REST の異なるアクセス手法のコンポーネント連携には，異なるミドルウェア設定が必要で，また，認証方式も異なるため，マッシュアップサービス開発稼動が大きい．本稿では，SOAP-REST マッシュアップを促進するための，手動設定ファイル記述不要の SOAP-REST 自動変換方式，SAML を用いた SOAP-REST シングルサインオン方式を提案する．提案方式を実装し，アプリケーション試作を通じて，開発が容易になることを示す．さらに，性能評価を行い，パスワード認証の場合は，SOAP-REST 変換の性能劣化がわずか数%程度で方式として有効であることを示す．

Study of Service Control Function for SOAP-REST Mash-up Service

Yoji Yamato,[†1] Yasutoshi Miyagi,[†1]
Yusuke Nakano,[†1] Yoshikazu Nakamura[†1]
and Hiroyuki Ohnishi[†1]

We propose the Service Control Function for SOAP-REST mash-up service, and we show our proposal effectivness. Today, Web 2.0 becomes a famous keyword, and many mash-up services are developed by many users. However, it is difficult to develope mash-up services of SOAP components and REST components because different middlewares and different authentication methods are needs to be set. In this paper, we propose a method of automatic SOAP-REST transformation which does not need setting files and a SOAP-REST single sign on method using SAML technology. We implemented our method and showed the easiness of developing mash-up services through a sample application development. We also measured performance of our system and showed the method effectiveness.

1. はじめに

近年，Web 2.0 [1] と呼ばれる新たなインターネットの利用法が注目されている．Web 2.0 は，ユーザ参加や集合知といった側面があり，その代表例として，ユーザが作成して公開するマッシュアップサービスがあげられる．マッシュアップサービスとは，複数の Web コンポーネントの API（Application Program Interface）を組み合わせ，1 つのサービスのように見せるサービスのことである．Web コンポーネントにアクセスする方法として，REST（REpresentation State Transfer）[2] や SOAP（Simple Object Access Protocol），JSON（JavaScript Object Notation）等がある．

しかし，コンポーネントを提供している企業は，これらすべてのアクセス方法を公開していることは少なく，業界によって公開するアクセス方法は異なることが多い．たとえば，テレコム機能の Web 向け API 仕様である Parlay-X [3] は SOAP が用いられており，アマゾン，ホットペッパー等の検索系サービスは REST が主流である．そのため，SOAP と REST コンポーネントのマッシュアップを行うユーザは，各コンポーネントに合わせてミドルウェア設定やプログラム作成をする必要があり，開発稼動（設計，環境構築，製造，試験）が増大する課題がある．また，SOAP と REST では，認証情報の渡し方が異なるため，SOAP と REST コンポーネントをシングルサインオンで利用することが困難であるという課題がある．一方，コンポーネント提供側も，SOAP と REST のアクセス手法に応じて，認証認可課金等の共通的処理を提供する必要があるため，提供コストが増大する課題がある．

そこで，本稿では，これらの課題を解決するための，実行制御機能を提案する．実行制御機能は，主に以下の 3 つの機能を提供する．A）SOAP から REST，REST から SOAP の自動変換を行う．B）SOAP と REST コンポーネントの認証認可を SAML（Security Assertion Markup Language）[4] の Assertion と Artifact を用いて行うことで，シングルサインオンでの連携を可能にする．C）SOAP や REST コンポーネントの SLA（Service Level Agreement）監視や擾乱防止等の非機能要件処理を一括して行う．これらにより，異

†1 日本電信電話株式会社 NTT ネットワークサービスシステム研究所
NTT Network Service Systems Laboratories, NTT Corporation

なるドメインの SOAP と REST コンポーネントの連携を促進する．

さらに，提案した実行制御機能を実装し，アプリケーション開発および性能評価を通じて，有効性を確認する．また，提案の実行制御機能を，関連技術と比較を行う．

なお，SOAP はメッセージ交換プロトコルで，REST はアクセススタイルである．本稿では，特に誤解がない限り，SOAP プロトコルで受信し REST アクセススタイルへ変換して転送することを，SOAP から REST の変換と記述し，SOAP ⇒ REST で表し，REST アクセススタイルで受信し SOAP プロトコルへ変換して転送することを，REST ⇒ SOAP で表すこととする．

本稿の構成は以下のとおりである．2 章で，SOAP と REST コンポーネント連携時の課題をあげる．3 章で，それらの課題を解決するための，実行制御機能の提案を行う．4 章で，実装した実行制御機能を用いたアプリケーション例を説明する．5 章で，実装した実行制御機能の性能評価を行う．6 章で関連技術について言及する．7 章でまとめを行う．

2. SOAP-REST 連携時の課題

現在，多くの公開コンポーネントがあるが，そのアクセス方法は SOAP と REST が主流である．SOAP は，Web Service シリーズと呼ばれる数多くの標準仕様があり拡張性が高い．しかし，SOAP を利用する際は，SOAP を解釈処理するミドルウェア（Axis，JAX-WS 等）が必要で，それらの設定が必要である．REST は，HTTP の 4 つのメソッドである GET，PUT，POST，DELETE を用いて，コンポーネントにアクセスするため，ブラウザで容易に検証できる．しかし，認証については，HTTP で通常用いられる認証方式（Basic 認証，Digest 認証等）が推奨されているが[5]，マッシュアップサービスにおいて，マッシュアップサーバとエンドユーザが異なる際に，どのように認証するかは統一された見解はない．

このように，SOAP，REST とも一長一短があるが，管理稼動が増大する問題から，両方を公開している企業は少ない．そのため，以下の課題がある．

a）クライアントが SOAP，REST の異なるアクセス手法のコンポーネントを連携する場合，別のミドルウェア，プログラム言語が必要で開発稼動がかかる．

b）SOAP，REST で認証方法が異なるため，シングルサインオンでの SOAP-REST の連携が困難．

c）コンポーネント提供者は外部公開のために，コンポーネントの特長機能以外に，SOAP，REST のアクセス手法に応じて，認証認可，擾乱防止，SLA 監視等の非機能要件処理が必要．

SOAP と REST を変換する技術として，ESB（Enterprise Service Bus）がある．ESB は，サービス連携のためのバスで，SOAP，REST に限らず，CORBA（Common Object Request Broker Architecture）や DB 等の各種アダプタを提供し，変換を行う．しかし，ESB では，SOAP と REST の変換は，変換のマッピング情報を設定ファイルに手動で記述する必要があり，設定稼動が大きい．

シングルサインオン技術に関しては，OpenID[6]，SAML[4]，Liberty Alliance[7]，Windows Cardspace があるが，主ターゲットは，Web ページの認証認可であり，SOAP と REST の異なるアクセス手法のコンポーネントのシングルサインオンについては，どの方式で行うべきかの議論は少ない．

そのため，本稿では上記の a），b），c）の課題を解決し，SOAP-REST の連携を促進するため，下記の A），B），C）を備えた，実行制御機能を検討する．

A）クライアントが REST，SOAP でアクセスした際に，手動記述の設定ファイル不要で，それぞれ SOAP，REST にオンラインで変換を行い，SOAP，REST コンポーネントを利用可能にする変換機能．

B）クライアントの認証処理は 1 度のみで，複数の SOAP，REST コンポーネントの認証認可をシングルサインオンで行う，シングルサインオン機能．

C）提供コンポーネントの非機能要件（擾乱防止，シェイピング，SLA 制御，認証認可，ID 変換等）を一括して処理する，非機能要件処理機能．

A）により，マッシュアップを作成するユーザは，SOAP か REST のどちらか一方だけ用いて開発できるため，得手の技術だけ用いればよい．また，コンポーネント提供者が設定ファイルを作成する必要がない．B）により，マッシュアップを作成するユーザは，認証方式の違いを意識せずに開発できるため，開発稼動が低減できる．C）により，コンポーネントを提供する企業は，コンポーネントの特長機能だけ提供すればよく，擾乱防止等の共通的処理を開発する必要がなくなるため，提供障壁が低減できる．

なお，C）の非機能要件の一括処理については，著者らの以前の研究である文献 8)，9) において，SOAP コンポーネントの非機能要件一括処理について考察がされている．そのため，本稿では，文献 8)，9) をベースに，これらの非機能要件処理を，SOAP だけでなく REST でも利用可能とする機能配備検討を行う．

3. 実行制御機能の提案，実装

本章では，まず，C）について，文献 8)，9) の SOAP に対する非機能要件処理を，SOAP，REST の両方で利用できるための機能配備について検討を行い，次に，SOAP-REST 変換

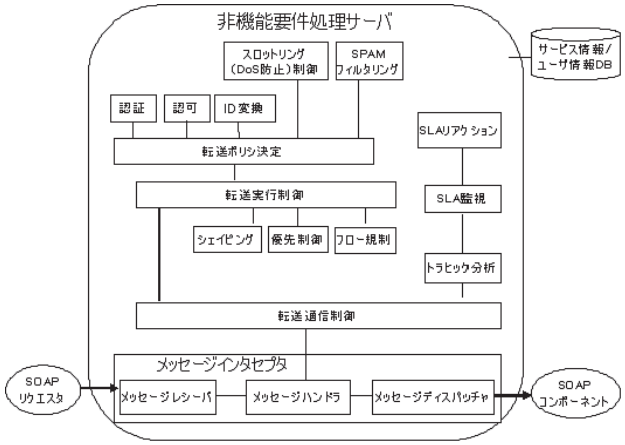


図 1　文献 8) の非機能要件処理機能ブロック図
Fig. 1　Function blocks of non-functional requirement processing part.

方式とシングルサインオン方式について検討する．

3.1　非機能要件処理の機能配備検討

図 1 は文献 8)，9) の機能ブロックの概略図である．SOAP コンポーネントと SOAP リクエスタの間に，非機能要件処理サーバが位置し，SOAP リクエストを受けると，メッセージインタセプタで SOAP メッセージがメモリに展開され，転送ポリシ決定部で，認証認可や ID 変換，DoS 攻撃や SPAM を防ぐ擾乱防止処理等が行われ，転送してよいリクエストの場合は，転送実行制御部で，シェイピングや優先制御等がされて，メッセージインタセプタを介して，SOAP コンポーネントにリクエストが転送される．SOAP コンポーネントからのレスポンスは，転送通信制御部にて，応答時間がロギングされ，集計された情報は，SLA 監視部に監視され，必要に応じてフロー規制やアラート等の SLA リアクションが行われる．

この中で，擾乱防止，優先制御等の非機能要件処理は，利用するアクセス手法に非依存に利用できる．一方，メッセージの受信，転送を行うメッセージインタセプタ部はアクセス手法に依存する．しかし，JSON や XML-RPC 等の少数のコンポーネントも，SOAP や REST で利用したい要望が出た際に，非機能要件処理サーバを変更するのは，運用面で問題がある．そこで，**図 2** のように，メッセージインタセプタ部は SOAP 透過と REST 透過のみ備え，アクセス手法を変換するトランスレータを非機能要件処理サーバの外部に配備する構成を提案する．このように機能分離することで，登録コンポーネントの種類が増えた際も，非機能要件処理サーバのファイル差換えをする必要がない．さらに，SOAP，REST，XML-RPC の変換は，XML 処理であるため，トランスレータは XML を処理する専用アプライアンス（例：IBM 社 WebSphere Datapower）を用いる等，必要処理能力に応じた柔軟な構成をとることができる．

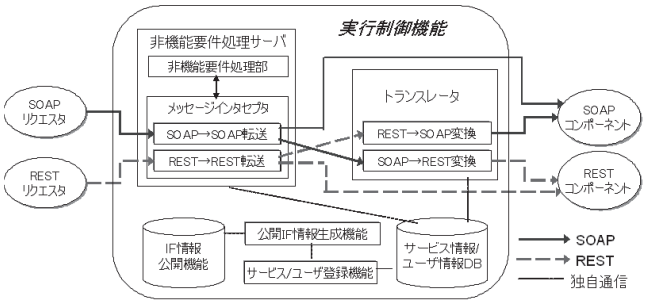


図 2　変換機能部の機能配備
Fig. 2　Functional disposition of a transform functional part.

図 2 の機能を補足説明する．サービス/ユーザ登録機能は，コンポーネント提供者がコンポーネントを登録するための GUI である．公開 IF（Interface）情報生成機能は，公開用のインタフェース文書を作成する機能で，ここで作成されたインタフェース文書は，IF 情報公開機能で公開される．サービス情報/ユーザ情報 DB は，登録されたコンポーネントの URL，サービス名，SOAP か REST の種別，その他非機能要件処理に必要な設定情報，ユーザ認証情報等を保持する．コンポーネント登録情報は非機能要件処理サーバ，トランスレータのメモリにもキャッシュされる．

3.2　SOAP-REST 変換方式の検討

ESB では，個々のコンポーネント登録時に，SOAP ⇔ REST のマッピングを設定ファイルとして作成する必要があるため，多くのコンポーネントを登録する際は，大きな稼動が必要である．また，実行制御機能は，SOAP ⇔ REST 変換のたびに処理を行うため，高い処理性能が必要である．そこで，本節では，SOAP-REST 変換方式として，① 手動設定ファイルが不要，② 高い処理性能，を実現する方式を検討する．

まず，手動設定ファイルを不要とするため，個々のコンポーネントに対してマッピングを記述するのではなく，全コンポーネントに対するマッピングルールを定義し，そのルールに

応じて，リクエスト受信時にオンラインでメッセージ構造を解析し，変換を行う方式とする．さらに，高い処理性能を実現するため，マッピングルールを可能な限り少なくすることで，効率化を図る．

なお，登録をすべて機械処理とするため，REST コンポーネントのインタフェース定義には，WADL（Web Application Description Language）[10] を用いる．WADL は，SOAP Web サービスにおける WSDL（Web Service Description Language）にあたるもので，REST インタフェースを記述する方法として最有力視されている．

まず，REST ⇒ SOAP について説明する．コンポーネント提供者は WSDL をサービス/ユーザ登録機能に登録すると，公開 IF 情報生成機能は，以下のルールでマッピングして公開用の WADL を生成し，IF 情報公開機能で公開する．

（R ⇒ S 1）HTTP のメソッドは GET とする．

（R ⇒ S 2）SOAP のオペレーション名をクエリパラメータ「operation」に設定する．

（R ⇒ S 3）SOAP パラメータフィールドを，以下のルールで，REST クエリパラメータに直列化する．

（R ⇒ S 3-1）配列：element 名@配列インデクス番号

（R ⇒ S 3-2）構造体入れ子：element 名を@で連結

（R ⇒ S 3-3）その他：element 名

（R ⇒ S 4）SOAP のレスポンス定義を，そのまま REST レスポンスの XML 文書とする．

REST リクエスト受信時は，トランスレータにおいて，上記の逆の処理が行われ（例：REST パラメータが operation = buy であれば，SOAP オペレーションは buy に設定），指定サービス名の SOAP コンポーネント URL にリクエストが送信される．**図 3** に，登録する SOAP コンポーネント（Parlay-X 3rdPartyCall サービス）へのリクエスト例，レスポンス例と，実行制御機能で公開される REST へのリクエスト例，レスポンス例を示す．図 3 の，矢印の番号は，上記のルール番号に対応している．

ここで，ポイントは，1，2 番目のルールである．1 番目は，SOAP オペレーションの意味の判別は機械ではできないため，自動変換を可能とするため，すべて HTTP の GET にマッピングしている．そのため，REST の DELETE，PUT，POST は生成できない．しかし，POST 等は，リクエストに続くエンティティにデータを入れるため操作稼動が増えるのに対し，GET は，大部分の試験はブラウザの URL に様々なクエリ文を入れてアクセスすることで可能なため，ブラウザを用いてマッシュアップするユーザにとっては，GET のみで大きな問題はないと考える．2 番目は，すべて GET にマッピングするため，SOAP

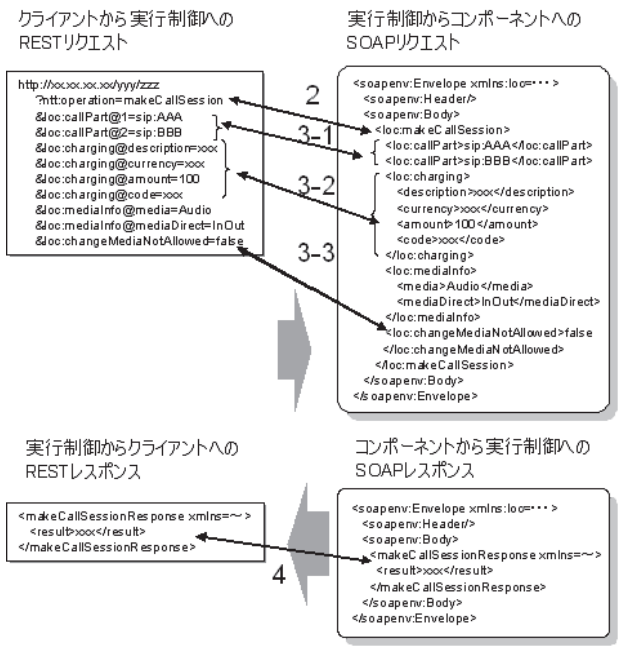


図 3　REST から SOAP への変換例
Fig. 3　Example of transformation from a REST request to a SOAP request.

のオペレーション名を区別するため，新たなパラメータとして，operation というパラメータを付与して区別を行う．

なお，SOAP の相互接続性を向上させるための仕様である WS-Interoperability（WS-I）において，literal 形式の配列，構造体等の入れ方が規定されており，WS-I に準拠した SOAP Web サービスであれば，3 番目のルールでパラメータをマッピング可能である．

次に，SOAP ⇒ REST について説明する．コンポーネント提供者は，サービス/ユーザ登録機能に WADL を登録すると，公開 IF 情報生成機能は，以下のルールでマッピングして公開用の WADL を生成し，IF 情報公開機能で公開する．

（S ⇒ R 1）SOAP オペレーション名は「HTTP メソッド名＋連番」とする．

（S ⇒ R 2）SOAP エンドポイントはリソースパス（resources base 属性＋ resource path 属性）とする．

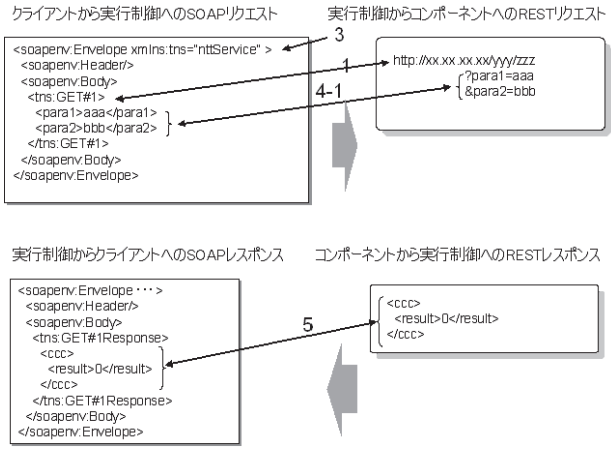


図 4　SOAP から REST への変換例
Fig. 4　Example of transformation from a SOAP request to a REST request.

（S ⇒ R 3）SOAP 名前空間は，コンポーネント登録時のサービス名，コンポーネント提供者 ID，実行制御機能のドメイン名から自動生成し，一意性を保証する．

（S ⇒ R 4）REST パラメータを，以下のルールで，SOAP リクエストパラメータへマッピングする．

（S ⇒ R 4-1）メディアタイプが “application/x-www-form-urlencoded” の場合は，各クエリパラメータを SOAP パラメータと 1 対 1 に対応付ける．

（S ⇒ R 4-2）メディアタイプが “application/xml” または “text/xml” の場合は，当該 XML 文書を SOAP の一パラメータとして扱う．

（S ⇒ R 5）SOAP レスポンスパラメータは，REST レスポンスで定義された XML 文書とする．

SOAP リクエスト受信時は，トランスレータにおいて，上記の逆の処理が行われ（例：SOAP オペレーション名が PUT2 であったら，REST コンポーネントに PUT でアクセス），指定サービス名の REST コンポーネント URL にリクエストが送信される．**図 4** に，登録する REST コンポーネントのリクエスト例，レスポンス例と，実行制御機能で公開された SOAP へのリクエスト例，レスポンス例を記述する．図 4 の，矢印の番号は，上記のルールの番号に対応している．

ここで，ポイントは，1，2 番目のルールである．1 番目は，REST では 1 リソースに同一メソッド名を複数定義できるためで，パラメータの違いを SOAP では別オペレーションとして区別するため，連番で区別している．2 番目は，1 つの WADL に複数のリソースパスを定義できるため，1 つの WADL から複数の SOAP サービスが生成されることがありうることを示す．

なお，3 番目のルールにおいて，コンポーネント提供者 ID は，実行制御機能運用者が管理し，他のコンポーネント提供者と重複した際は別の ID にするよう指示することで，一意性を保証する．図 4 では，tns:nttService の ntt がコンポーネント提供者 ID，Service が登録サービス名に該当する．この SOAP 名前空間を含む，登録コンポーネント情報（URL，サービス名，オペレーション名等）は，サービス情報/ユーザ情報 DB に保持され，非機能要件処理サーバやトランスレータが再起動した際は，DB から情報を取得する．

マッピングルールは，個々のコンポーネントに依存しないため，設定ファイル不要で，オンラインでの SOAP，REST 変換を可能とする．ただし，機械処理であるため制限がある．たとえば，PUT を用いる REST コンポーネントを実行制御機能で変換した後，再度実行制御機能で変換すると，元々の REST とは異なり GET だけになるため，非可逆となる．

3.3　SOAP-REST シングルサインオン機能の検討

SOAP-REST シングルサインオンは，ユーザは 1 回の認証で，複数のコンポーネントを連携できることが必要である．そのためには，① 多くのコンポーネントに対して共通的に使える技術を用いて認証を行い，② 未対応のコンポーネントに対して実行制御機能が代理処理することが必要となる．

SOAP ではセキュリティ仕様として，WS-Security [11] が整備され，username token profile，SAML token profile 等の，認証情報に応じたセキュリティ方式が標準化されている．一方，REST では，HTTP の通常の認証方式が推奨されているため，Web ページと同様の方式で問題ない．そこで，認証情報の流通方式として，SOAP，REST ともに親和性の高い SAML を用いた方式を検討する．

SAML の Assertion は資格証明であり，これを流通させることで，Assertion を受け取ったコンポーネントは，Assertion 発行者に問合せを行うまたは信頼する等をして，認証するため，ユーザは 1 度の認証でシングルサインオンでサービス利用ができる．

SOAP では，WS-Security の SAML token profile により Assertion を SOAP ヘッダに格納できる．一方，REST では特に規程されておらず，HTTP 拡張ヘッダや REST の GET や POST パラメータとして資格証明を入れることができる．ブラウザで利用するユー

ザにとっては，GET パラメータとしての利用が主流と考えられるが，GET パラメータに Assertion を入れると URL が長くなるため，Web プロキシの実装によっては，途中で切られてしまう可能性がある．そこで，REST では，Assertion のワンタイムポインタとして使える Artifact を用いて，REST リクエストのたびに Assertion を解決または発行することで，資格情報を流通させる．

コンポーネント側が，SAML に対応していない場合は，SAML を用いて実行制御が代理認証した後，コンポーネント側の認証方式（例：WS-Security username token や，REST パラメータでの ID/パスワード渡し等）に応じて，ID 関連情報の変換を行い，コンポーネントにリクエストを転送する．ID 関連情報の変換については，文献 8) で実装済みで，3.1 節の方式で非機能要件処理部を利用することで解決できる．

図 5 (a) に，SOAP ⇒ REST の場合の非機能要件処理サーバ，トランスレータの処理フローを示す．SOAP リクエスタは事前に非機能要件処理サーバで認証し，Assertion を付与してリクエストを送信する．REST コンポーネントが SAML に対応している場合は，トランスレータは Assertion から Artifact の発行を非機能要件処理サーバの認証機能に依頼し，Artifact を REST コンポーネントに転送する．REST コンポーネントは，必要に応じて Assertion を取得し，認証を行う．また，REST コンポーネントが SAML 非対応の場合は，非機能要件処理サーバの ID 変換機能が，REST コンポーネント用の認証 ID，パスワード等の情報を代理入力して転送を行う．

図 5 (b) に，REST ⇒ SOAP の場合の非機能要件処理サーバ，トランスレータの処理フローを示す．REST リクエスタは事前に非機能要件処理サーバで認証し，Artifact を付与してリクエストを送信する．SOAP コンポーネントが SAML に対応している場合は，非機能要件処理サーバは，Artifact から Assertion を取得して認証処理をした後，再度 Artifact を発行してトランスレータに転送し，トランスレータは Artifact から Assertion を取得して SOAP ヘッダに付与して，SOAP コンポーネントに転送する．

なお，図 5 では，SAML 対応と個別の ID 関連情報を必要とするコンポーネントのみ記述しているが，認証不要なコンポーネントや username token を必要とするコンポーネントもある．その場合も，ID 変換部が，認証方式に合わせて，ID 関連情報の削除や加工を行う．

このように，ユーザは 1 度認証を行えば，ID 体系が異なる SOAP や REST のコンポーネントも同様に利用できるため，たとえば，BPEL（Business Process Execution Language）[12] エンジンを用いて，各コンポーネントを呼び出す際に，Assertion を引き継ぐように設定すれば，複数の SOAP，REST コンポーネントをシングルサインオンで利用できる．ここで，

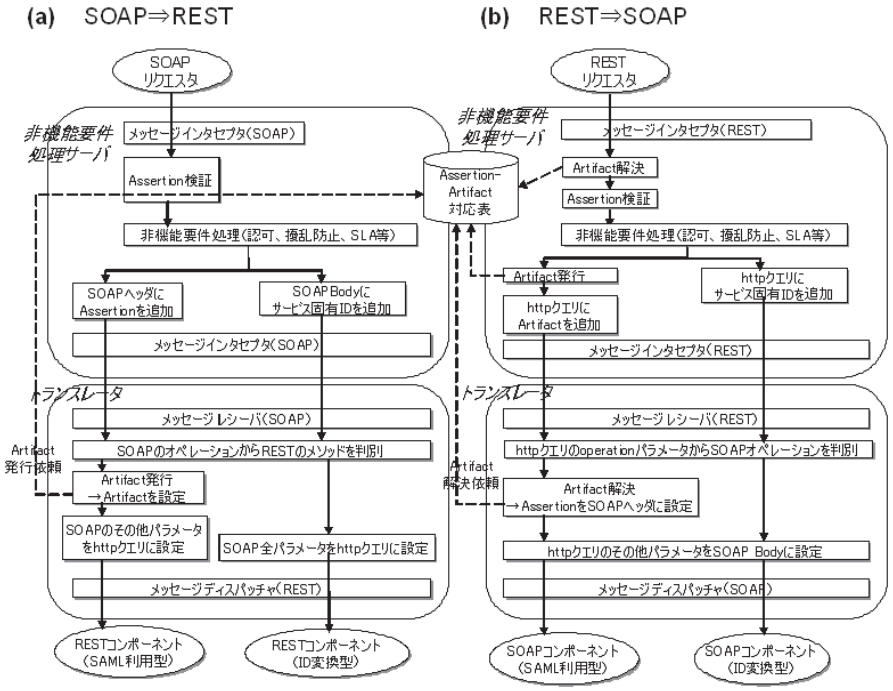


図 5　SOAP と REST 変換時の認証方式．(a) SOAP から REST 変換時，(b) REST から SOAP 変換時
Fig. 5　Authentication method of SOAP and REST transformation. (a) Transformation from SOAP to REST, (b) Transformation from REST to SOAP.

BPEL とは，Web サービスを連携するための標準仕様である．

これらの機能を備えた実行制御機能を，非機能要件処理サーバとトランスレータの 2 つの筐体に実装した．非機能要件処理サーバおよびトランスレータは，文献 8)，9) のソフトウェアも利用して，OS が RedHat Enterprise Linux ES4.0，サーブレットコンテナとして Tomcat 6.0.18 を用いて，JDK1.6.0 を用いて実装した．

4. アプリケーション例と設定工数比較

4.1 駅情報取得サービス

SOAP-REST 連携実行制御機能を用いたアプリケーションとして，駅情報取得サービス

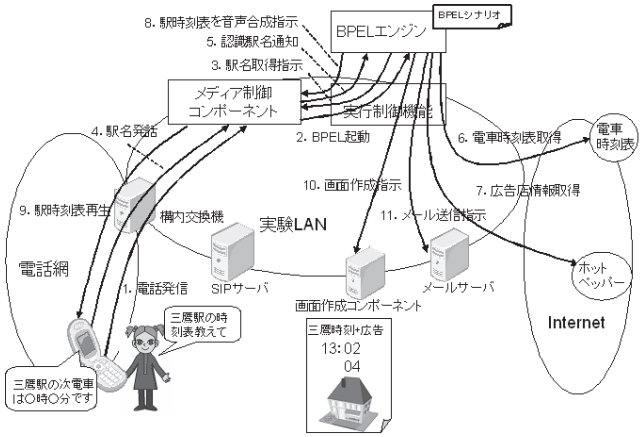


図 6　アプリケーション例：駅情報取得アプリケーション．なお，図の矢印は SIP 信号のやりとりは省略している
Fig. 6　Sample application: station information application.

を実装した（**図 6**）．本アプリケーションは，NTT R and D Forum 2009[13] に展示された．

アプリケーションの狙いを述べる．移動中に電車時刻を知りたい際に，携帯電話のブラウザで出発駅を指定して検索を行うが，文字入力や画面遷移が多く，操作が煩雑になる．本アプリケーションは，音声により駅時刻表取得を指示し，音声やメールで時刻表と駅周辺の店舗情報を提供することで，煩雑な操作を必要とせず欲する情報を提供する．

本アプリケーションでは，音声認識や音声合成等のテレコム機能制御を行うメディア制御コンポーネントと，店舗情報を提供するホットペッパー，ポータルサイトの電車情報コンポーネントを，BPEL エンジンを用いて連携する．メディア制御コンポーネントは SOAP で WS-Security の username token で認証し，ホットペッパーは REST でアクセスし，パラメータの 1 つの API キーで認証する．なお，メディア制御コンポーネントは，Parlay-X を一部拡張し，音声認識や音声合成を SOAP で利用可能とする，著者らが開発したコンポーネントである[14]．

アプリケーション動作を図 6 を用いて説明する．携帯電話を持つユーザは，指定番号に電話すると，メディア制御コンポーネントを介して，メディアサーバと電話がつながる．それとともに，メディア制御コンポーネントは，BPEL エンジンに，ある電話番号のユーザから着信があったことを通知し，駅情報取得 BPEL サービスが起動する．起動の際に，実行制御機能において，携帯電話ユーザに対応した Assertion が発行される．BPEL エンジンは，メディア制御コンポーネントに駅名を取得するよう指示し，メディア制御コンポーネントはユーザの音声を認識した駅名を BPEL エンジンに通知する．ここで，BPEL エンジンは，資格証明として Assertion を付与してリクエストを実行制御機能に送信し，実行制御機能が，認証後，username token に変換して，メディア制御コンポーネントに転送している．

次に，BPEL エンジンは通知された駅名をキーに，時刻表，店舗情報検索を行うため，それぞれ，実行制御機能に，SOAP で Assertion をヘッダに付与してリクエストする．実行制御機能は，時刻表検索は認証情報が不要であるため，トランスレータで SOAP から REST 変換を行いポータルサイトの時刻表検索サービスにリクエストを行う．一方，店舗情報検索の場合は，Assertion のユーザ ID から，ホットペッパーの API キーに変換し，API キーを付与した後，トランスレータで SOAP から REST 変換を行い，リクエストされる．これらの取得情報を用いて，BPEL エンジンは，メディア制御コンポーネントに，合成音声の指示を行い，メディアサーバからユーザに対して，「〜駅の次の電車は〜時〜分です」と音声が流れる．また，BPEL エンジンは，駅時刻表と店舗情報が載ったページを作成しそのページ情報をユーザのメールアドレスに送信する指示を出す．

このように，SOAP-REST 連携実行制御機能を用いることで，アプリケーション開発者は，BPEL ファイルだけ記述すればよく，また，認証情報も各コンポーネントを呼び出す際に Assertion を渡す設定にすれば，実行制御機能が，コンポーネントに合わせた REST 変換および認証情報変換を行ってくれるため，開発が容易である．

なお，駅情報取得サービスでは，BPEL エンジンや SOAP-REST 変換処理は，100 msec 弱と短いのに対し，広告や駅情報取得は 1 秒程度，メディア制御コンポーネントに SOAP で指示を出してから携帯電話に発音されるまで数秒程度と，全体処理時間の内，コンポーネント処理の割合が高い．そのため，マッシュアップサービスで要求される処理時間に応じて，利用コンポーネントは考慮する必要がある．

4.2　変換設定工数比較

SOAP-REST 変換の設定工数を，本アプリケーションのホットペッパーを題材に，ESB と比較する．

実行制御機能は，登録する WADL があれば，登録 GUI に沿って登録するだけであるため，10 分程度で変換設定ができる．今回は，ホットペッパーの API 説明文書から WADL を手動作成したため，30 分程度 WADL 作成時間がかかった．しかし，REST コンポーネント提供者が自ら登録する場合は，プログラムから WADL を自動生成する技術（たとえば，ServiceLayer[15]）があるので，この作業は不要である．

また，ホットペッパーは認証のための API キーが必要なので，登録時は，API キーが認証情報であることを，登録 GUI で設定する必要がある．この設定により，SAML のユーザ ID とホットペッパーの API キーのマッピングを，利用ユーザが実行制御機能のユーザ情報 DB に登録可能になる．ホットペッパーをマッシュアップサービスで利用したいユーザは，SAML のユーザ ID とのマッピングを登録する．

ESB では，公開インタフェース記述を作成し，登録コンポーネントのインタフェースとのマッピング記述を作成する．WSDL と WADL であれば，XSLT（eXtensible Stylesheet Language Transformations）相当を作成する．ESB によって作成手段は異なるが，たとえば IBM 社の WebSphere ESB では，GUI ツールを用いて，ホットペッパーの場合で，作成に 2 時間，デプロイ，動作確認に 1 時間程度かかる．

ESB で，ID 変換を行う際は，Assertion の認証処理後，外部 DB にアクセスし，利用サイトのユーザ ID とパスワードを取得し，そのパラメータに変換する処理を，カスタムメディエーションで記述する方法がある．WebSphere ESB では，Java で記述するため，開発，検証に 2 日程度はかかる．

このように，提案の SOAP-REST 自動変換機能により，ESB に比べて変換設定工数を大きく下げることができている．

5. 性 能 評 価

5.1 性能測定目的

実行制御機能の性能測定を通じて，評価を行う．性能測定では，以下の 3 つの測定を行い評価する．

① SOAP ⇔ REST の変換処理にともなう性能変化

SOAP 透過と SOAP ⇒ REST，REST 透過と REST ⇒ SOAP の比較により，アクセス手法の変換処理によりどの程度性能が劣化するかを検証する．この比較では，変換に関する影響だけ見るため，SAML 認証は用いず，SOAP では username token 認証，REST ではユーザ ID とパスワードによる認証を行う．この検証により，オンラインでの変換処理が問題ないかを確認する．

② REST 利用時の Artifact 処理にともなう性能変化

SAML 認証を用いて，SOAP 透過と SOAP ⇒ REST，REST ⇒ SOAP を比較することで，REST の Artifact 解決，発行にともない，どの程度性能が劣化するかを検証する．

③ 非機能要件処理の市販製品と比べた性能検証

SOAP 透過の非機能要件処理が，市販の Policy Enforcer 製品である Oracle Communications Services Gatekeeper 4.0（OCSG）と比較して，問題ない性能であるかを検証する．この検証により，アクセス手法に応じたメッセージインタセプタと共通の非機能要件処理部とした構成が問題ないかを確認する．ここで，Policy Enforcer とは，OMA（Open Mobile Alliance）[16] で定義されている，認証認可，SLA 制御等のポリシ制御を行う機能のことである．

5.2 性能測定条件

性能測定では，実行制御機能に一定の負荷をかけた際の，CPU 使用率，応答時間，処理時間内訳を測定した．測定は 3 回行われ，グラフは 3 回の平均値である．なお，OCSG の性能は今回の条件での性能であり，条件により大きく変わる可能性があることは注意されたい．性能測定条件を以下に示す．

測定パターン：

・スループット測定（1 時間あたりのリクエスト数を 30,000〜5,000,000 と変化）

・複数同時実行（同時処理スレッド数を 1〜300 と変化）

測定項目：

・CPU 使用率（Linux の top コマンドで取得）

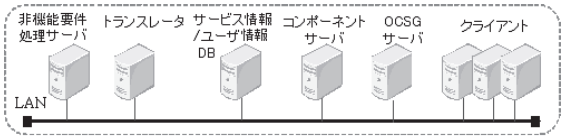
・応答時間（リクエスタのログ出力で取得）

・処理時間内訳（受信，転送ポリシ決定，トラヒック制御，転送等の区間ごとの処理時間．非機能要件処理サーバやトランスレータでは，各処理ごとに機能ブロックが分かれており，機能ブロックの処理開始と処理終了のタイミングでタイムスタンプがログ出力され，その差が各処理の処理時間となる．）

測定設定：

・非機能要件処理サーバおよび OCSG の行う処理は，認証，認可，スロットリング，シェイピング，優先制御，SLA 監視である．① ③ では，SOAP は username token，REST はユーザ ID とパスワードによる認証を行う．② では，SAML token を用いた認証を行う．

・SOAP コンポーネントは，string 型パラメータ 1 つを引数で受け取り，100 msec スリープし，string 型パラメータ 1 つを戻り値で返却する document/literal 型の Web サービス．REST コンポーネントも string 型パラメータ 1 つを引数で受け取り，100 msec スリープし，string 型パラメータ 1 つを XML 文書として返却する HTTP GET を用いた Web サービス．100 msec スリープは，実運用を想定して，Web サービス側での処理時間をとる形としている．



	ハードウェア	OS	ソフトウェア
非機能要件処理サーバ	Dell PowerEdge1950 CPU: Xeon5160 3.0GHz× 2 RAM: 4.0GB NIC: Broadcom Gigabit Ethernet	RedHat Enterprise Linux ES4	JDK 1.6.0_07 Tomcat 6.0.18 非機能要件処理機能
トランスレータ	Dell PowerEdge1950 CPU: Xeon5160 3.0GHz× 2 RAM: 4.0GB NIC: Broadcom Gigabit Ethernet	RedHat Enterprise Linux ES4	JDK 1.6.0_07 Tomcat 6.0.18 トランスレータ機能
サービス情報/ユーザ情報DB	Dell PowerEdge1950 CPU: Xeon5420 2.50GHz× 2 RAM: 4.0GB NIC: Broadcom Gigabit Ethernet	RedHat Enterprise Linux ES4	PostgreSQL 8.3.3
コンポーネントサーバ	Dell PowerEdge1950 CPU: Xeon5420 2.50GHz× 2 RAM: 4.0GB NIC: Broadcom Gigabit Ethernet	RedHat Enterprise Linux ES4	JDK 1.6.0_07 Tomcat 6.0.18 SOAPコンポーネント RESTコンポーネント
OCSG サーバ	Dell PowerEdge1950 CPU: Xeon5160 3.0GHz× 2 RAM: 4.0GB NIC: Broadcom Gigabit Ethernet	RedHat Enterprise Linux AS4	Oracle Communications Services Gatekeeper 4.0
クライアント	Dell Precision T3400 CPU: Core2 Extreme X9650 3.0 GHz RAM: 3.25GB NIC: Broadcom NetXtreme Gigabit Ethernet	Windows XP SP2	JDK 1.6.0_07 soapUI 2.5

図 7　性能測定環境
Fig. 7　Performance measurement environment.

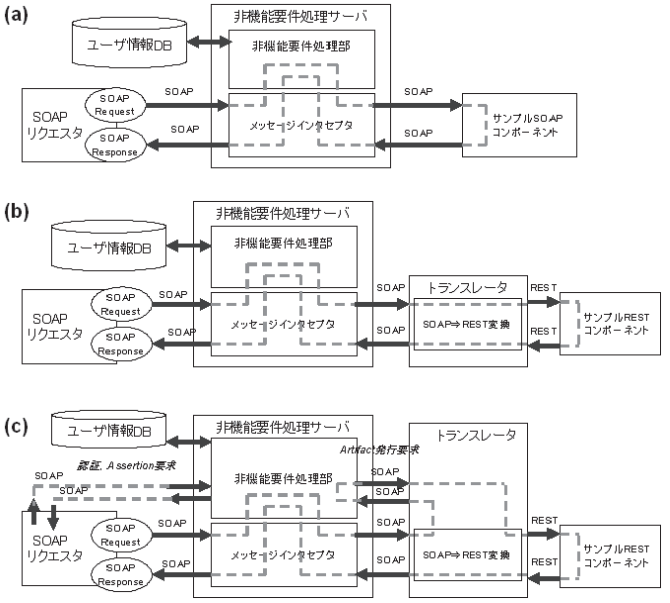


図 8　実行制御機能の処理フロー．(a) SOAP 透過時，(b) SOAP ⇒ REST 変換時，(c) SAML 認証を用いた SOAP ⇒ REST 変換時
Fig. 8　Processing flow of service control function. (a) SOAP, (b) Transformation from SOAP to REST using username token authentication, (c) Transformation from SOAP to REST using SAML authentication.

測定環境：

測定環境を**図 7** に示す．測定環境は，非機能要件処理サーバ，トランスレータ，サービス情報/ユーザ情報 DB，リクエスタ端末 3 台，コンポーネント提供端末，OCSG サーバを，100Base-TX のハブ CentreCOM FH716XL を用いて同一 LAN に接続した．使用したハードウェア，ソフトウェアは図 7 記載のとおりである．

処理フロー：

測定時の処理フローを**図 8** に示す．図 8 (a) は測定 ① ③ で SOAP 透過の場合である．図には載せないが，測定 ① の REST 透過は (a) の SOAP を REST に置き換えたもの，測定 ③ の市販 Policy Enforcer の SOAP 透過は (a) の非機能要件処理サーバを市販 Policy Enforcer に置き換えたものである．username 認証により，リクエスタはユーザ ID とパスワードを付与してリクエストするため，非機能要件処理サーバにおいて，ユーザ情報 DB を用いて認証が行われた後，リクエストが転送される．

図 8 (b) は測定 ① で SOAP ⇒ REST 変換の場合である．測定 ① の REST ⇒ SOAP 変換は，(b) の SOAP を REST に置き換えたものである．それぞれ，非機能要件処理サーバで処理が行われた後，トランスレータで変換が行われ，リクエストが転送される．

図 8 (c) は測定 ② で SOAP ⇒ REST 変換の場合である．測定 ② の SOAP 透過は，(c) のトランスレータがない場合である．測定 ② の REST ⇒ SOAP 変換は，(c) の SOAP を REST に，トランスレータからの Artifact 発行要求を Artifact 解決要求に置き換えたものである．② の SOAP ⇒ REST 変換の場合は，リクエスタは非機能要件処理サーバから Assertion を発行してもらい，それを付与して，非機能要件処理サーバにリクエストを送信する．非機能要件処理後，リクエストはトランスレータに転送される．トランスレータは，

Assertion を元に Artifact を非機能要件処理サーバに発行してもらい，REST パラメータにセットした後，REST コンポーネントにリクエストを転送する．

5.3 測 定 結 果

5.3.1 ① SOAP ⇔ REST の変換処理にともなう性能変化

図 9 (a) は，横軸非機能要件処理サーバの CPU 使用率，縦軸スループット（リクエスト処理数/h），(b) は，横軸同時処理スレッド数，縦軸平均応答時間，(c) は横軸トランスレータの CPU 使用率，縦軸スループットをとったグラフである．

図 9 (a) より，同等 CPU 使用率での，REST 透過のスループットは SOAP 透過の約 2 倍を示しているが，最大スループットはほとんど差が生じていないことが分かる．これは，非機能要件処理サーバ内部の認証，認可処理の部分でシングルスレッドで動作している箇所があり，そこで待ち行列ができるため，CPU には余裕があってもスループットが頭打ちになることが原因と考えられる．SOAP の方が同等 CPU 使用率でのスループットが低いのは，SOAP リクエスト受信時の XML パースの処理で CPU をより多く使うためである．

また，SOAP 透過と SOAP ⇒ REST のスループット比較，および REST 透過と REST ⇒ SOAP のスループット比較を見ると，プロトコル変換処理にともなう性能劣化は，数%程度であり，大きな劣化は見られない．図 9 (b) の，同時処理スレッド数を変化させた際の，応答時間に関しても同様で，スレッド数が同じ際の平均応答時間の性能劣化は少ない．

図 9 (c) より，トランスレータのスループットは，SOAP ⇒ REST に比べ，REST ⇒ SOAP の方が大きいことが分かる．REST ⇒ SOAP では，HTTP のクエリ文字列から必要な情報を取得するだけなのに対して，SOAP ⇒ REST では SOAP メッセージのボディを解析して情報を抽出する必要があり，そのパラメータ解析処理の差が性能差に現れたと考えられる．

5.3.2 ② REST 利用時の Artifact 処理にともなう性能変化

図 10 は，横軸非機能要件処理サーバの CPU 使用率，縦軸スループットをとったグラフである．図 10 より，SOAP の username 認証に比べて，SAML 認証の場合は，同等 CPU 使用率でのスループットが 1/2 程度で，最大スループットも 1/3 程度である．これは，username 認証が単純な文字列照合処理であるのに対し，SAML 認証は Assertion に対する署名検証や有効期限チェック等が必要であるためである．さらに，図 10 より，SAML 認証で，SOAP ⇒ REST の場合は，SOAP 透過の場合と比べて最大スループットが 1/3 程度，REST ⇒ SOAP の場合は，SOAP 透過の場合と比べて最大スループットが 1/2 程度と劣化していることが分かる．

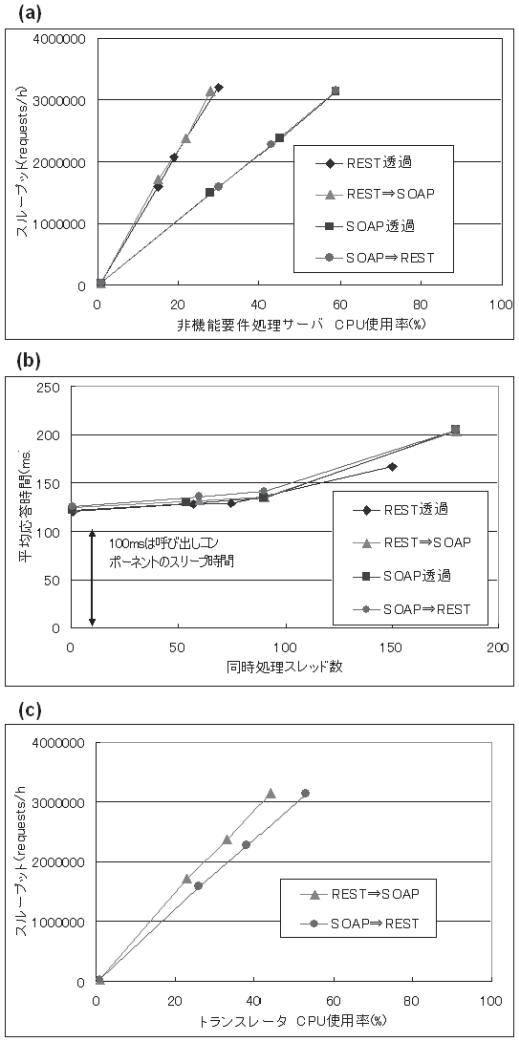


図 9 (a) スループットの非機能要件処理サーバ CPU 使用率変化（username token 認証時），(b) 平均応答時間の同時処理スレッド数変化，(c) スループットのトランスレータ CPU 使用率変化

Fig. 9 (a) Result of throughput measurement of non-functional requirement processing server, (b) Result of average response time, (c) Result of throughput measurement of translator.

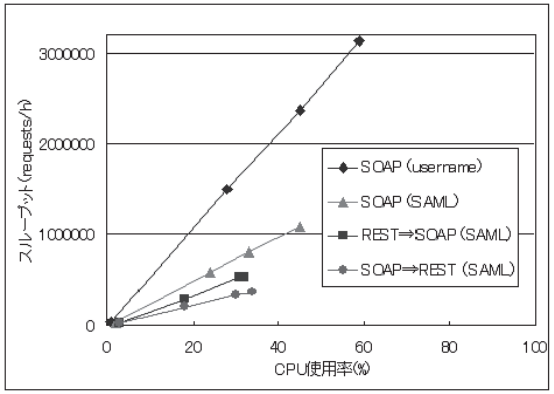


図 10　スループットのサーバ CPU 使用率変化（SAML 認証時）
Fig. 10　Result of throughput measurement.

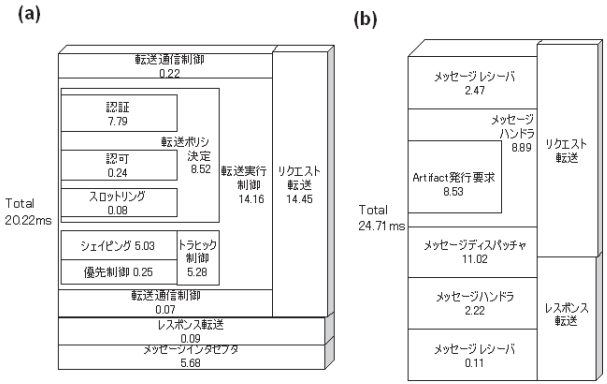


図 11　SAML 認証を用いた SOAP から REST 変換時の処理時間内訳．(a) 非機能要件処理サーバの処理時間内訳，(b) トランスレータの処理時間内訳
Fig. 11　Processing time of each section when transformation from SOAP to REST using SAML authentication. (a) Processing time of each section of non-functional requirement processing server, (b) Processing time of each section of transelator.

図 11 は SOAP ⇒ REST の 30 スレッド同時処理時の，(a) は非機能要件処理サーバ，(b) はトランスレータの処理時間内訳（ms）の平均である．図 11 から，非機能要件処理サーバ内での Assertion の検証にかかる認証処理時間が大きく，また，トランスレータ内での Artifact 発行要求にかかる処理時間が大きいことが分かる．この Artifact 処理時間が性能を大きく下げているといえる．SOAP ⇒ REST が REST ⇒ SOAP よりスループットが低いのは，① と同様，SOAP ボディのメッセージ解析処理に時間がかかっているためである．

5.3.3　③ 非機能要件処理の市販製品と比べた性能検証

図 12 は，(a) 横軸同時処理スレッド数，縦軸平均応答時間，(b) 横軸非機能要件処理サーバまたは OCSG の CPU 使用率，縦軸スループットをとったグラフである．(a) より，低負荷時の平均応答時間は，非機能要件処理サーバの方が大きいが，高負荷時では市販 Policy Enforcer 製品が急激に大きくなっていることが分かる．(b) より，同等 CPU 使用率では非機能要件処理サーバが OCSG の 1.4 倍で，最大スループットも 1.6 倍程度を示していることが分かる．

これは，非機能要件処理部が，文献 8) で提案した認可情報キャッシュを用いているため，高負荷時も効率良く処理できているためである．

5.4　考　　察

① の測定結果から，SOAP ⇒ REST，REST ⇒ SOAP とも，SOAP 透過，REST 透過に比べて大きく性能劣化していないことが分かる．これは，マッピングルールを少なく汎用的にしたことで，オンラインでの変換も問題なく処理できることを示している．スループット絶対値を見ても，300 万 requests/h 程度処理できている．たとえば，処理時間が 100 msec 程度の 4 個のコンポーネントを連携するマッシュアップサービスでは，60〜70 万サービス/h 程度処理でき，電話網加入者交換機の 30 万コール/h の性能と同等のオーダであり，十分な性能といえる．現在の実装では，認証認可の転送ポリシ決定や Assertion 処理がシングルスレッドであるため，CPU を使いきれない課題があるため，そこをマルチスレッドにすることで，より高いスループットが実現できると考える．

② の測定結果から，SOAP 透過の SAML 認証時は username 認証時に比べて，1/2〜1/3 程度のスループットとなり，SOAP ⇒ REST 変換や REST ⇒ SOAP 変換時は，さらに 1/2 程度のスループットに下がっていることが分かる．この理由には，SAML の Assertion は，数多くの情報が含まれているため，その検証に多くの CPU リソースが使われること，さらに，Artifact 解決や発行のため，非機能要件処理サーバとの通信が行われることがあげられる．

これを高速化するためには，Artifact 発行依頼のキャッシュ利用が考えられる．マッシュアップサービスでは，複数個のコンポーネントが同時に利用される．そのため，最初の Artifact 発行依頼は一から SOAP 文書を作成して依頼するが，2 回目以降は同じ依頼内容となるた

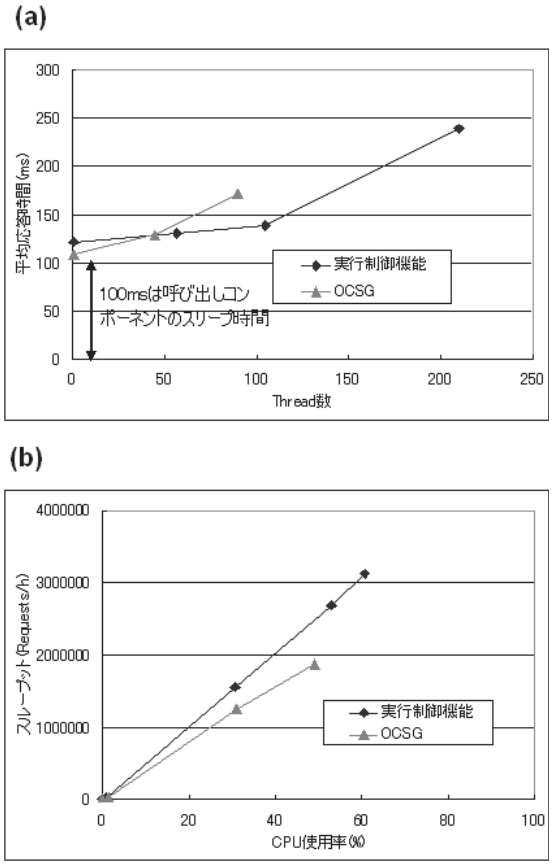


図 12　(a) 平均応答時間の同時処理スレッド変化（username token 認証），(b) スループットのサーバ CPU 使用率変化（username token 認証）

Fig. 12　(a) Result of response time measurement using username token authentication, (b) Result of throughput measurement using username token authentication.

め，1 回目に利用した SOAP 文書をキャッシュで用いることができる．このようなキャッシュ機能の実現により，システムの規模は増大するが，性能向上が見込める．

そのほか，XML 処理高速化手法の適用（たとえば文献 17)），前述の Assertion 処理のマルチスレッド化も高速化に効果的と考える．

③ の測定結果から，非機能要件処理は，市販製品 Oracle Communications Services Gatekeeper の 1.6 倍程度の高い性能を示していることが分かる．これは，アクセス手法に応じたメッセージインタセプタが共通の非機能要件処理部を利用する構成で，十分に性能を発揮できることを示している．

6. 関 連 技 術

Web サービス連携の仲介ハブとして，ESB がある．ESB 製品としては，IBM 社の WebSphere ESB や Oracle 社の Aqualogic Service Bus 等がある．ESB は JBI（Java Business Integration）[18] で一部機能が標準化されているが，ベンダにより実装機能は異なることが多い．ただし，コンポーネントのプロトコル変換機能は，多くの ESB 製品が実装している機能である．しかし，ESB は SOAP と REST の変換を行う際は，マッピングの設定ファイルの手動記述が必要である．

SCA（Service Component Architecture）[19] は，SOA（Service Oriented Architecture）のコンポーネント実装モデルであり，JMS（Java Message Service），EJB（Enterprise Java Beans），SOAP 等のバインディング非依存性を，SCA 実行環境が実現する．しかし，SCA は実装モデルであり，Java コード作成時に，外部呼び出しコンポーネントは@Remotable 等の記述が必要で，すでに動作しているコンポーネントの場合は変更が必要である．

本稿の SOAP-REST 変換機能は，コンポーネントの変更や，事前の設定ファイル手動作成を必要とせず，SOAP-REST 変換をオンライン処理で行う．

WSDL2.0 は，SOAP binding および HTTP binding が可能である．コンポーネント提供者は，WSDL2.0 対応のアプリケーションサーバを用いて，SOAP，REST の両方を公開できる．しかし，現状，WS-Interoperability の推奨が WSDL1.1 であるため，WSDL2.0 対応サービスはきわめて少なく，また，コンポーネント提供者は，大規模事業者以外は，運用負担から SOAP，REST の片方のみ公開していることが多い．そのため，本稿の SOAP-REST 変換機能は重要と考える．

シングルサインオン技術に関しては，OpenID，SAML，Liberty Alliance 等が検討されているが，主ターゲットは，Web ページの認証認可のシングルサインオンであり，SOAP と REST のシングルサインオンについては，どの方式で行うべきかの議論は少ない．本稿では，実行制御機能がコンポーネントの代理で認証認可を行い，SOAP 利用時は SAML の Assertion を WS-Security を用いて認証情報を伝播させ，REST 利用時は SAML の Artifact を用いてパラメータとして認証情報を伝播させることで，シングルサインオンを実現して

いる．

テレコムコンポーネントのポリシ制御を行う機能として，OMA では，Policy Enforcer を定義しており，その実現方法の 1 つとして PEEM（Policy Evaluation，Enforcement and Management）が規定されている．Policy Enforcer 製品として，Oracle Communications Services Gatekeeper が著名である．しかし，OMA の Policy Enforcer は，どのようなポリシを適用するかはドメイン管理者依存で，また実装方法についても規定はない．本稿の SOAP-REST 変換を備えた実行制御機能は，4 章のアプリケーション例のように，Web-テレコムを連携する際の Policy Enforcer の 1 つということができる．

実装した実行制御機能は，SOAP-REST 連携によるマッシュアップサービスを主ターゲットにしており，マッシュアップに必要な機能に絞り一体化しており，処理，管理効率向上を図っている．5 章の性能評価により，市販製品を上回るスループットを実現しており，処理効率が良いといえる．

7. ま と め

本稿では，SOAP と REST コンポーネントを容易に連携させるための，SOAP-REST 変換機能を備えた実行制御機能の検討を行った．SOAP-REST 連携において，変換設定ファイル作成稼動が大きい，シングルサインオン認証認可が困難という課題を解決するため，マッピングルールに従ったオンラインの SOAP-REST 自動変換方式，SAML の Assertion と Artifact を用いたシングルサインオン方式を提案した．検討した実行制御機能を実装し，アプリケーション試作および性能評価を行った．

アプリケーション試作を通じて，アプリケーション開発者は，アクセス手法や認証を意識せずに作成できるため開発が容易であることを確認した．パスワード認証の場合は，SOAP 透過と SOAP から REST 変換および，REST 透過と REST から SOAP 変換の性能比較から，変換にともなう性能劣化は数%程度で，オンラインでの変換処理が性能上問題ないことを確認した．しかし，コンポーネントの SAML 認証の場合は，REST が利用される際は，Artifact の発行および解決により，性能が大きく劣化することが分かった．

今後は，SAML 認証時の実行制御機能の高速処理に向けた方式検討を行う．また，SOAP-REST 連携実行制御機能を用いた開発と，従来の手動でのマッシュアップサービス開発の開発効率を詳細に比較する．さらに，今回，実装した実行制御機能を著者らが検討する SDP [20] に適用して，別機関と新たな連携サービス創出トライアルを行い，SOAP-REST 変換のルールや性能に問題がないかの検証を進める予定である．

参 考 文 献

1) What is Web 2.0 web site. http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html
2) Fielding, R.: Architectural Styles and the Design of Network-based Software Architectures, Doctoral dissertation, University of California, Irvine (2000).
3) Parlay-X web site. http://www.parlay.org/en/specifications/pxws.asp
4) OASIS Security Assertion Markup Language web site. http://saml.xml.org/saml-specifications
5) Prescod, P.: Common REST mistakes. http://www,prescod.net/rest/mistakes
6) OpenID web site. http://openid.net/
7) Liberty Alliance web site. http://www.projectliberty.org/
8) 山登庸次，大西浩行，須永　宏：Web-テレコム連携のためのサービス実行制御技術の研究開発，電子情報通信学会論文誌，Vol.J91-B, No.11, pp.1417–1427 (2008).
9) 山登庸次，大西浩行，中野雄介：Web-Teleco 連携サービスの擾乱防止機能の検討，電子情報通信学会ソサイエティ大会，B-19-1 (Sep. 2008).
10) Web Application Description Language web site. https://wadl.dev.java.net/wadl20061109.pdf
11) OASIS WS-Security web site. http://www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf
12) Business Process Execution Language web site. http://www.oasis-open.org/committees/download.php/18714/wsbpel-specification-draft-May17.htm
13) NTT R and D forum 2009 web site. http://event.ecl.ntt.co.jp/forum2009/info/index.html
14) 櫻田玲子，入江道生，金子雅志，森谷高明，大西浩行，東　勲，平野美貴：対話連動型 Telecom 連携アプリケーションの実現方法に関する一検討，電子情報通信学会総合大会，B-19-25 (Mar. 2009).
15) ServiceLayer web site. http://agileitinc.com/product.html/
16) Open Mobile Alliance web site. http://www.openmobilealliance.org/
17) Takase, T. and Tajima, K.: Lazy XML Parsing/Serialization Based on Literal and DOM Hybrid Representation, *International Conference on Web Services* (*ICWS2008*), pp.295–303 (Sep. 2008).
18) JSR (Java Specification Request) 312: Java Business Integration 2.0 web site. http://jcp.org/en/jsr/detail?id=312
19) OASIS Service Component Architecture web site. http://www.oasis-opencsa.org/sca
20) Ohnishi, H., Yamato, Y., Kaneko, M., Moriya, T., Hirano, M. and Sunaga, H.: Service Delivery Platform for Telecom-Enterprise-Internet Combined Services,

IEEE Global Telecommunications Conference 2007 (*GLOBECOM2007*), pp.108–112 (Nov. 2007).

(平成 21 年 5 月 18 日受付)
(平成 21 年 10 月 2 日採録)

山登　庸次

2000 年東京大学理学部卒業．2002 年同大学大学院理学系研究科修士課程修了．2009 年同大学院総合文化研究科にて博士（学術）取得．2002 年日本電信電話株式会社入社．同社にて，Peer-to-Peer ネットワーク，ユビキタスコンピューティング，Service Delivery Platform 研究開発に従事．現在，NTT ネットワークサービスシステム研究所および米 Verio 社勤務．2007 年電子情報通信学会通信ソサイエティ功労賞，2009 年電子情報通信学会学術奨励賞受賞．電子情報通信学会会員．

宮城　安敏（正会員）

2005 年立命館大学理工学部卒業．2007 年奈良先端科学技術大学院大学情報科学研究科修士課程修了．2007 年日本電信電話株式会社入社．同社にて，Service Delivery Platform，サービス開発手法の研究開発に従事．現在，NTT ネットワークサービスシステム研究所社員．

中野　雄介（正会員）

2005 年和歌山大学大学院システム工学研究科修了．同年日本電信電話株式会社入社．同社，NTT ネットワークサービスシステム研究所にて Web マイニング，ユビキタスコンピューティング，グループウェア研究に従事．2004 年情報処理学会第 66 回全国大会学生奨励賞受賞．平成 19 年度電子情報通信学会学術奨励賞受賞．電子情報通信学会会員．

中村　義和

2000 年九州大学工学部卒業．2003 年同大学大学院工学府材料物性工学専攻修了．2003 年日本電信電話株式会社入社．同社にて，移動体向けセキュリティ技術，センサネットワーク構築技術，Service Delivery Platform の研究開発に従事．電子情報通信学会会員．

大西　浩行

現在，東日本電信電話株式会社担当課長．1996 年早稲田大学にて機械工学学士，1998 年早稲田大学大学院にて機械工学修士取得．1998 年に日本電信電話株式会社入社．同社にて次世代モバイル IP ネットワークアーキテクチャ，サービスアプリケーション開発プラットフォームの研究開発に従事．1999 年に電子情報通信学会交換システム研究賞，2005 年に電子情報通信学会研究奨励賞，2007 年に電子情報通信学会通信ソサイエティ功労賞受賞．