

REST 形式 Web サービスのテスト実行に基づく高精度な検索

重井康宏[†] 鷲崎弘宜^{†,††} 深澤良彰[†]

Web サービスを組み込んだ開発において、Web サービスの選定・実行にはコストが発生する。Web サービスには通信形式により REST 形式と SOAP 形式が存在する。サービス記述の仕様が厳密に定められている SOAP 形式に対し、REST 形式は厳密な仕様が定められていない。本稿では REST 形式 Web サービスを対象として、テスト実行に基づいた検索を行う事で Web サービス選定のコストを減らすと共に従来のメタ検索より高精度な結果を得る事を確認した。

Precision Search for REST Web Service

YasuhiroShigei[†] and HironoriWashizaki^{†,††} and
YoshiakiFukazawa[†]

To select and execute Web Service costs many hours and quantity of work. There are two communication formats, REST and SOAP. SOAP format is defined strict specification of service description, but REST format is not defined it. Then in this paper, we proposed the search based on test execution covering REST Web Service and achieve to reduce web services selection costs and raise the precision than existing meta-search.

1. はじめに

Web サービスの普及に伴い、Web サービスを組み込んだ開発が増加している。中でも Representational State Transfer (REST) 形式 Web サービス[1]に関しては仕組みが容易であり、マッシュアップ開発によく用いられる。しかし多様な REST 形式 Web サービスが存在する中で、それらを整理する手法は無く開発者は Web サービスのそれぞれの仕様書を読み、検証する必要がある。また Web サービス全体の問題としても可用性・運用性が Web サービス提供側に依存するという問題があり、Web サービスを組み込んだシステムは常に代替のサービスを用意しておく必要がある。そこで提案手法ではテストケースを用意したテスト実行を行う事で、REST 形式 Web サービスの検索を行い、開発者の Web サービス選定のコストを削減する手法を提案する。

2. Web サービス

2.1 Web サービスとは

Web サービスとはソフトウェアの機能をネットワークを通じて使用できるようにしたソフトウェアコンポーネントである。また Web サービスは複数の Web サービス同士を繋ぎ合わせる事で新たな機能・アプリケーションを構築する事が出来る。この構築はマッシュアップと呼ばれ昨今有益な Web サービスの普及によりマッシュアップが多くなされている。このマッシュアップ開発の例としてパーソナルユース向けの開発者にとっては blog・個人サイトにおけるウィジェットが挙げられる。またエンタープライズサービス開発の一環としても Web サービスは組み込まれ、例えば SNS サイトである mixi[2]では amazon[3]の Web サービスを利用した商品レビュー機能・Youtube[4]や niconico 動画[5]の Web サービスを利用して日記に動画が貼れるという機能を実装しており、mixi 利用者のコミュニケーションの促進に利用されている。

また Web サービスの特徴として、多くのサービスは複雑で有益な機能が無償で利用できる、リソースを呼び出し側が負担しないといった点が挙げられる。

2.2 Web サービスの分類

Web サービスはメッセージ通信方式により次のように分類ができる。

- REST 形式 Web サービス

REST 形式 Web サービスとはクライアントと Web サービスのメッセージ通信を REST で行う Web を指す。REST 形式の通信方法とは http メソッドに

[†] 早稲田大学
Waseda University
^{††} 国立情報学研究所 GRACE センター
NII GRACE Center

より Web サービスにリクエストを送ると結果の XML が返ってくるメッセージ通信 (図 1) である。また REST 形式の Web サービスはステートレスである。

`http://search.yahooapis.jp/web...&query=%e6%b2%96%e7%b8%84&results=2`

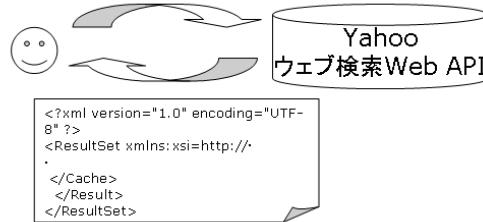


図 1 REST 形式 Web サービスの通信例

● SOAP 形式サービス

SOAP 形式 Web サービスとは、クライアントと Web サービスのメッセージ通信を SOAP で行う Web サービスを指す。SOAP 形式の通信方法とは WSDL によって定められた Web サービスの仕様に基づき XML 形式のリクエストを Web サービスに送る事でレスポンスが XML で得られるメッセージ通信方法である。

2.3 提案手法の対象

SOAP 形式・REST 形式の Web サービスにおいて REST 形式の Web サービスは通信方法が容易でありマッシュアップで盛んに使用されている。また REST 形式の Web サービスには有益な Web サービスが多いため提案手法では REST 形式の Web サービスを対象とする。

3. Web サービスの問題とその解決

3.1 Web サービス全般の問題

Web サービス全体の問題点として以下が挙げられる。

「P1: Web サービスを組み込んだシステムは、Web サービスによって賄われている機能の可用性・信頼性が Web サービス提供側に依存する。」そのため Web サービス提供者による Web サービスの停止、Web サービスの仕様の変更を行った場合、システム全体が停止してしまうという危険性がある。

3.2 REST 形式 Web サービス全般の問題

REST 形式 Web サービス固有の問題点として以下のものが挙げられる。

「P2: SOAP 形式 Web サービスが WSDL という形式が定められているのに対し REST 形式はパラメータ名・パラメータ数などが自由に定められこれらを表現する仕様が厳密に定まっていない。」これにより開発者は REST 形式の Web サービスを探す・試すといった場合において統一されていない仕様書を読むというコストが発生する。

3.3 既存手法の問題と解決

P1, P2 を解決する手法として「S0: REST 式 Web サービスの検索」が存在する。また既存手法として「S1: メタ情報によるテキスト検索」「S2: サービスの仕様をフォーマルな仕様書として用意しての検索」という二つの手法が挙げられる。

S2 については API Compare-and-Matching Service[6]というサービスが存在する。しかしこれらの手法には「P3: 検索者の要求に合致するかの保障は無い」という問題点がある。そこで提案手法ではこれらの問題を解決する手段として「S3: テスト実行を基底とした REST 形式 Web サービス検索」を提案する。

テスト実行に基づく検索において安藤ら[7]はソフトウェアコンポーネントに対するテスト実行に基づく検索を提案している。またテスト実行に基づく評価において W.T.Tsai ら[8]は SOAP 形式のサービスを対象としたテストケースを用いたテスト実行による評価手法を提案している。

3.4 提案手法

提案手法を以下に提示する。まず P1, P2 を解決するため解決法 S0 を採用するが既存手法の問題点である P3 が発生する。そこで P3 を解決するために S3, 「S4: サービス合成を検索対象に含めた検索」を提案する。

しかし S3・S4 を実行するに当たり検索システムが行うテスト実行数は莫大となるため「P4: 実行時間が莫大になる」という問題を持つ。そこで「S5: メタ情報での絞り込み」を行った対象を S3, S4 の処理にかけることにより P4 を解決する。

また S3 に関してはテストケースの準備をするコストが発生するが、一般プロダクトの開発においてはテストケースを用意するテスト駆動な開発が出現しつつあり、この開発手法はプロダクトの品質向上に繋がる。Web サービスを利用したマッシュアップに基づく開発はトライアンドエラー的な部分があり、今後 Web サービスの数が増大し開発者が検討すべき Web サービスが増大することを考えるとこのテスト駆動な開発が使用される事が予想される。また P1 においてすでにシステムに組み込んである Web サービスの非機能部分に関して要求に合致しなく代替の Web サービスと差し替えた場合に実行結果があればそれをテストケースの記述に使用できるというメリッ

トを持つ。

4. 提案手法における REST 形式 Web サービス検索システム

4.1 全体像

本システムの全体構成の概要を示す(図2)。まず検索を行うためのリポジトリに対しシステム管理者が Web サービスの登録・管理をする。またユーザーの検索要求としての入力メタデータ・テストケース・評価軸であり本システムはこの検索要求により Web サービスの検索を行う。まずリポジトリに登録してある Web サービスのメタデータとユーザーの入力したメタデータによりテスト実行の対象を絞り込む。次にテスト実行対象の Web サービスのスキーマとテストケースの入力実データからテスト実行を行う。この際パラメータへの値の与え方・パラメータの組み合わせによってテスト実行の結果は変化するためリクエスト URI は網羅的に作成をしてテスト実行を行う。ユーザーが検索要求としてサービス合成を望んでいた場合はテスト実行結果を入力としてサービス合成候補を抽出し再度テスト実行を行う。これらのテスト実行によって得られたレスポンス XML とユーザーの入力したテストケースの出力期待データ・評価軸からテスト実行を行った Web サービスに関しソートを行い検索結果としてユーザーに提示をする。この章の以降では本システムの各動作について詳しく述べてゆく。

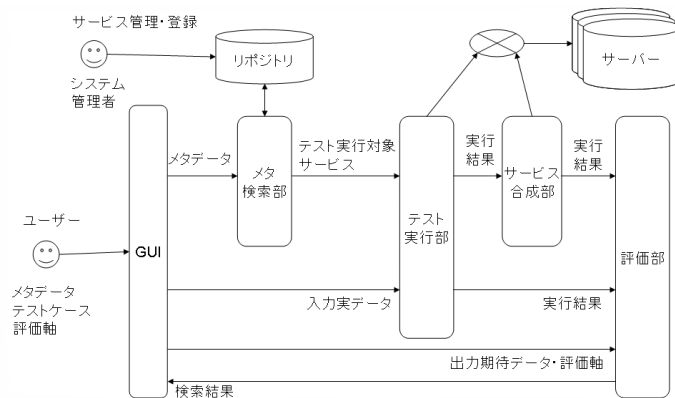


図2 提案手法の全体像

4.2 REST 形式 Web サービスのスキーマ定義

本システムにおいて REST 形式 Web サービスに対しテストケースを入力として実行するために、REST 形式 Web サービスの情報をスキーマとして定義する必要がある。REST 形式 Web サービスを実行するに当たり、「サービス名称」「サービスを表すメタ情報」「パラメータが付与されていない状態の URI」「パラメータ情報」が必要となる。

まずリクエスト URI に与えるパラメータの特性として。

- **mandatory**
必須であるパラメータ。
例：API 使用のためのキー
- **or_mandatory**
複数のパラメータ群でその群で最低1つが必須であるもの。
例：カテゴリ指定や検索ワード指定
- **optionally**
必須ではないパラメータ
例：検索結果を何件レスポンスするかの指定

と定義する。

またパラメータに指定する値の特性として

- **default**
パラメータに指定する値が決まっている
- **URI_encode**
値を URI エンコーディングする必要がある

と定義する。

これらの属性を用い、提案手法では REST 形式 Web サービスのスキーマを定義する。スキーマの例として、簡単なレストランサービスのスキーマを以下に提示する。(図3) サービスの仕様としては、サービスを利用するために API のキーを入れ、検索条件として駅名が入力された場合には駅周辺のレストラン名を出力、またはレストラン名を入力すると最寄りの駅を出力するという仕様である。

```
<Service>
  <ServiceName>Example Restaurants Service</ServiceName>
  <meta>レストラン 駅</meta>
  <requestURI>http://example.com</requestURI>
  <inputData>
    <mandatory>
      <parameter>
```

```

        <name>api_key</name>
        <default>xyz</name>
    </parameter>
</mandatory>
<or_mandatory>
    <parameter>
        <name>station_name</name>
    </parameter>
    <parameter>
        <name>restaurants_name</name>
    </parameter>
</or_mandatory>
<optional>
    <parameter>
        <name>result_number</name>
    </parameter>
</optional>
</inputData>
</Service>

```

図 3 Example Restaurants Service

4.3 リポジトリの作成

REST 形式 Web サービスのスキーマに従い REST 形式 Web サービスをリポジトリに登録する。一般に REST 形式の Web サービスの仕様はそのサービス提供側が仕様をまとめた Web ページを用意している。

その Web ページには

- 元となるリクエスト URI
- 入力パラータ名
- 入力パラメータが必須であるか
- 入力パラメータに指定する値
- 入力パラメータに値を指定する際の規定

といった仕様が明記されている。本システムでのリポジトリの登録においてはこれらの Web ページを閲覧しそこに明記してある仕様に基づき 4.2 で定義したスキーマを記述する。

4.4 検索クエリの入力

本システムがユーザーから入力される値として以下のものがある。

- サービスのメタデータ
- サービスに与えるテストケース
- サービス合成を検索結果に含めるかの有無
- 評価尺度

評価尺度としては

- テストケースにおける出力データが現れる XML のノードの深さ
- 応答ファイルサイズ
- Web サービスの応答時間

があり、これらの指定によりテスト実行結果をソートする。

また本研究では、Web サービスのパラメータに指定する入力実データと期待出力の組み合わせをテストケースとして定義する。例えばレストラン名を入力し、そのレストランの最寄り駅を出力するというテストケースは表 1 の通りである。

表 1 テストケースの例

入力実データ	期待出力
レストラン A	高田馬場
レストラン B	渋谷
レストラン C	新宿

4.5 テスト実行対象の絞り込み

ユーザーが入力した入出力の意味データを元にサービススキーマのメタタグ内に記述されているメタデータを参照しテスト実行対象の絞り込みを行う。例えばメタ意味データとして「レストラン」と入力された場合、リポジトリ内のサービススキーマのメタタグを参照し「レストラン」というメタデータを持つ Example Restaurants Service のようなサービスを抽出する。

4.6 リクエスト URI の網羅作成

4.5 でメタ的に絞り込んだテスト実行対象の各 Web サービスをテスト実行させるために Web サービスに対するリクエスト URI を作成する必要がある。しかしユーザーの入力したテストケースの入力値をどのパラメータに付与するかは判らないためリクエスト URI の網羅的作成が必要となる。またパラメータの対象として optionally 属性のパラメータに関しては Web サービスへのリクエストの補助、例えば結果の提示量などのため Web サービス実行における本質的なパラメータではないので本システムで

は mandatory 属性と or_mandatory 属性を対象としてリクエスト URI の網羅作成を行う。

mandatory 属性に関しては必ずリクエスト URI に組み込まなくてはサービスは動かないので default 属性が決められている場合はその値を、定められていない場合はテストケースの入力データを指定する。この際 1 つのテストケースに入力データが複数存在する場合は網羅のため全て指定できるようにリクエスト URI の数を増やす。また or_mandatory 属性でまとめられているパラメータ群に関してはそれらのパラメータ群の中で最低 1 つ指定が必要であるがパラメータ群が n 個あった場合は n 個から 1 ~ n 個選ぶという組み合わせに対して値を指定しリクエスト URI を作成する必要がある。

リクエスト URI 作成の手順は以下の通りである。

Step1) mandatory 属性かつ default 属性によりパラメータに付与する値が指定されているパラメータ集合 $Pmd = \{pmd_1, pmd_2, \dots, pmd_a\}$ に対し, $pmd_1 = \{\text{指定値}\}$ & $pmd_2 = \{\text{指定値}\} \dots$ & $pmd_a = \{\text{指定値}\}$ というパラメータ列を作成する。

Step2) mandatory 属性かつ default 属性が指定されていないパラメータ列 $Pm = \{pm_1, pm_2, \dots, pm_b\}$ に対し, テストケースの実入力データ集合

$D = \{d_1, d_2, \dots, d_n\}$ との組み合わせパラメータ列を一つ作成する。

ex.) $pmd_1 = d_1$ & $pmd_2 = d_2$ & $\dots$ & $pmd_b = d$

Step3) or_mandatory 属性のパラメータ集合 $Po = \{po_1, po_2, \dots, po_c\}$ に対し,

Step2) にて使用されていない ($m \cdot b$) 個の実入力データとの全ての組み合わせを作成する。

Step4) Pm と D の全ての組み合わせに対し Step2, Step3 の作業を繰り返す。

具体的な手順として, Example Restaurants Service に対し表 1 のテストケースである「レストラン A」という実入力データが与えられた場合を例に挙げる。

Step1) mandatory 属性であるパラメータ「api_key」が存在するため, URI として組み込む必要が有る。また default 属性として「xyz」という値を指定するという記述があるため, URI 候補は「http://example.com?api_key=xyz」となる。

Step2) or_mandatory 属性である「station_name」「restaurants_name」が存在し, 両者とも default 属性が指定されていない。そのためテストケースの実入力である「レストラン A」を指定する必要が有る。しかし「レストラン A」という値が駅名であるのかレストラン名とであるのかは判断できないため, 各々に指定する URI 候補を作成する。つまり「http://example.com?api_key=xyz&station_name=レストラン A」

「http://example.com?api_key=xyz&restaurants_name=レストラン A」という URI が作成される。

Step3) optionally 属性であるパラメータである「result_number」が記述されているが, 前述の通り検索結果のレスポンス数等は Web サービス実行における本質ではないため考慮しない。

Step4) これらの Step により「http://example.com?api_key=xyz&station_name=レストラン A」「http://example.com?api_key=xyz&restaurants_name=レストラン A」というリクエスト URI が網羅作成される。

また「 r : テストケースの入力実データ数」「 n : パラメータの総数」「 m : mandatory 属性のパラメータ数」「 l : or_mandatory 属性のパラメータ数」とした場合の URI 発行数に注目をする。これらの網羅作成を行う際, サービススキーマで定義した mandatory・or_mandatory・optionally といった性質を用いない場合, リクエスト

URI の数は $\sum_{k=1}^r C_k * P_k$ となる。しかしこれらの性質を用い, 制約をかける事で

$m P_r + \sum_{k=1}^{r-m} C_k * P_k$ の数に減少させ, 無駄なリクエスト URI の発行を削減している。

4.7 Web サービス単体のテスト実行

提案手法ではテストケースの入力値をパラメータに指定し Web サービスを実行した際のレスポンス XML からの機能性である期待出力が含まれるか・期待出力が出現するノードの深さの測定, Web サービスを実行した際の効率性であるレスポンス時間・応答ファイルサイズの測定を行う事をテスト実行と定義する。例えば図 4 のように, 入力として「レストラン名」・期待出力として「駅名」の実データが入力されているテストケース (表 1) から 4.6 で作成したリクエスト URI 「http://example.com?api_key=xyz&station_name=レストラン A」を Web サービスに対し実行し, 図 5 のようなレスポンス XML を取得する。またその際実行時の効率性の品質としてレスポンス時間, 応答ファイルサイズの測定を行う。リクエスト URI 数が多かった場合, Web サービス提供側のサーバーに異常な負荷をかけてしまうという問題があるため, 本システムでは 2 秒おきの実行を行うようにしている。

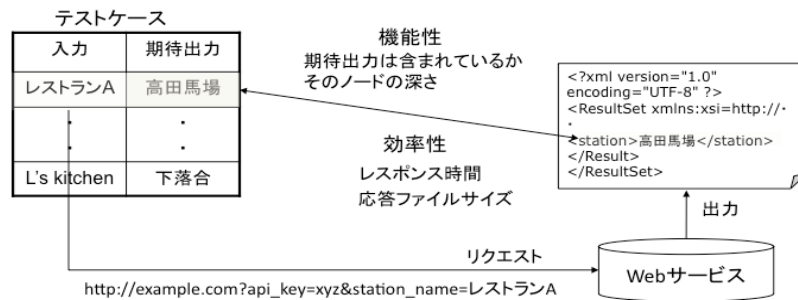


図4 テスト実行の例

```
<ResultSet>
  <Result>
    <name>レストラン A</name>
    <zip_code>xxx-yyy</zip_code>
    <address>東京都新宿区高田馬場 X 番地</address>
  </Result>
</ResultSet>
```

図5 Example Restaurants Service のレスポンス XML 例

4.8 Web サービス合成のテスト実行

ユーザーが Web サービスの合成を行った検索を指定した場合、メタ的に絞り込んだサービスの入力にテスト実行結果で得られた出力を指定するという手法によりサービス合成を行う。しかし一般にサービスのレスポンス XML に含まれるタグで囲まれたデータは量が多いためこれらのデータに対しサービス単体のテスト実行のように網羅的にリクエスト URI を作成することはリクエスト URI の数が膨大になりシステム全体の検索時間を考えた際に現実的ではない。

そこで Web サービス合成を行う際のリクエスト URI の作成方法としてテスト実行対象のサービススキーマとレスポンス XML のテスト実行対象のパラメータ名の比較によって入出力の候補を抽出しリクエスト URI を作成する。

この比較の方法としては注目しているパラメータ名とタグ名が

- 完全一致している
- どちらかの名がもう一方に含まれる

という 2 点を調べる。完全一致の場合はその組み合わせを入出力の対応とし、どちらかがもう一方に含まれる場合は含まれる文字列の長さに対する含む文字列の長さの割合を保持しておき一つのパラメータ名に対しレスポンス XML 全てのタグ名における含まれる文字列の長さに対する含む文字列の長さの割合でもっとも大きいものを出力の対応とする。

例えば Example Restaurants Service と Example Station Service (図 6) を合成する場合、Example Station Service のパラメータとして mandatory 属性の address が存在する。この address の入力値を指定するため Example Restaurants Service のレスポンス XML (図 5) のタグ名である、「name」「zip_code」「address」から比較を行う。この場合レスポンス XML の address というタグ名と Example Station Service の address というパラメータ名が一致するため、address パラメータに指定する値はレスポンス XML に含まれる「東京都新宿区高田馬場 X 番地」となる。そのため合成後のリクエスト URI として「http://station.com?address=東京と新宿区高田馬場 X 番地」が生成される。

また Web サービスの合成の数が上がるほどテスト実行の回数は増えそのオーダーは莫大なものになってしまうため本システムでは 2 つの Web サービスの合成を対象とする。

```
<Service>
  <ServiceName>Example Station Service</ServiceName>
  <meta>駅</meta>
  <requestURI>http://station.com</requestURI>
  <inputData>
    <mandatory>
      <parameter>
        <name>address</name>
      </parameter>
    </mandatory>
  </inputData>
</Service>
```

図6 Example Station Service

4.9 テスト実行結果の評価

テスト実行結果のレスポンス XML を解析しユーザーが指定した評価軸に従ってテスト実行結果を評価し、Web サービス検索結果の表示順を決定する。

優先順位としては

1)入力されたテストケースの出力期待データの総数に対するレスポンス XML に出力期待データが含まれた割合

2)入力された評価軸の平均値を正規化したものの平均値

である．入力されたテストケースの出力期待データの総数に対するレスポンス XML に出力期待データが含まれた割合はテストケースを用いたテスト実行において機能性において一番重要な評価尺度である．またユーザーが入力する評価軸である

- テストケースにおける出力データが現れる XML のノードの深さ
- 応答ファイルサイズ
- Web サービスのレスポンス時間

においては Web サービスの機能性・効率性を表す評価尺度であるが，ユーザーの Web サービスの利用用途により大幅に要求が変化する項目である．

例えば携帯電話向けアプリケーションを構築する際に Web サービスを利用する事を考えた時，携帯電話のリソースの少なさから応答ファイルサイズが小さい方が好ましい．また反対にリソースは充分なので多少応答ファイルサイズが大きくともレスポンス時間が短く実行速度が速いほうが良いという要求も存在すると考えられる．このように効率性に関してはトレードオフな関係がある．

まず 1)については入力された各テスト実行結果のレスポンス XML を解析し与えられたテストケースの出力期待データが含まれているかを判定する．またテストケースが複数あった場合，テストケースの出力期待データがレスポンス XML に含まれているテストケースの割合を基準とする．次に 2)については各テスト実行を行った際に測定したレスポンス時間・応答ファイルサイズとレスポンス XML にテストケースの出力期待データが出現するノードの位置の中からユーザーが評価尺度として指定したものに對し正規化を行う．

レスポンス時間・応答ファイルサイズ・レスポンス XML のテストケースの出力期待データが含まれるノードの位置らは全て小さい方が評価が高い．

正規化の方法としては 0 が最低評価，1 が最高評価として以下のように行う．

$$\text{正規化した値} = \frac{x - z}{x - y}$$

x=テスト実行した全体の結果の最高値

y=テスト実行した全体の結果の最低値

z=そのテスト実行結果の注目している評価の値

またサービス合成を用いているテスト実行結果に関しては以下のように評価を行う

レスポンス時間：1 つ目の Web サービスでのテスト実行にかかったレスポンス時間とその出力を用いて 2 つ目の Web サービスの入力としたテスト実行にかかったレスポ

ンス時間の和．

応答ファイルサイズ：1 つ目の Web サービスでのテスト実行にかかったレスポンス時間とその出力を用いて 2 つ目の Web サービスの入力としたテスト実行にかかった応答ファイルサイズの和．

レスポンス XML のテストケースの出力期待データが含まれるノードの位置：1 つ目の Web サービスでのテスト実行の出力を用いて 2 つ目の Web サービスの入力としたテスト実行で出力されるレスポンス XML での出現ノードの位置．としてソートを行う．

5. 本システムの評価

本システムの評価方法として再現率・精度による評価手法を用いる．本システムの評価環境としてリポジトリにはレストランに関する Web サービスが 4 件，駅情報に関する Web サービスが 3 件入っており検索要求として「入力したレストランの最寄り駅を出力する」に関するメタデータとテストケースを用意し同一リポジトリにおけるメタ検索の結果と本システムの検索結果を比較した (図 7)．尚メタ検索の再現率・精度を測る際，メタ検索において順位が出ないためメタ検索における最高の検索結果・最悪の検索結果・予想される標準の検索結果の 3 ケースを用意した．比較の結果，本システムはメタ検索における最高の検索結果に対しては精度の点で劣るがメタ検索において具体的な検索要求がある場合メタ情報として細かい情報を大量に付与する事は現実的ではなく，テストケースをテスト実行する事で初めて検索要求を満たすと考えられる．またメタ検索の最低の検索結果と予想されるメタ検索の標準検索結果とは精度の点で勝っている．

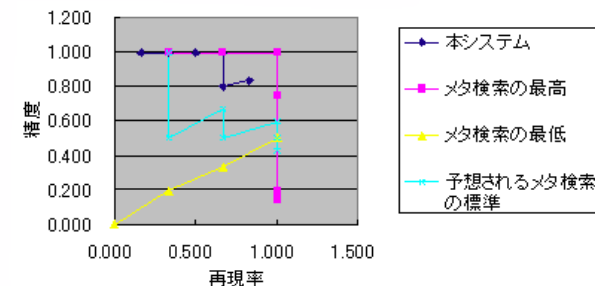


図 7 再現率・精度グラフ

6. 関連研究

関連研究としてテスト実行に関する研究・Web サービスの自動合成に関する研究を示す。

安藤らはソフトウェアコンポーネントに対しテスト実行を行うことでコンポーネントの振舞いを評価するコンポーネント検索手法を提案した。この手法ではソフトウェアコンポーネントに対しテスト実行のための入力としてテストプログラムが対応し、評価としてテスト結果とコンポーネントの接続コストを採用している。

提案手法では対象が Web サービスであり、入力として入出力のテストケースであるためテストプログラムを作成するよりもコストが低いと考えられる。

W.T.Tsai らは SOAP 形式のサービスに対しテストケースを用いたテスト実行を行うことでサービスの評価を行うテスト手法を提案した。この手法はテストケースを与えテストケースの総数に対するサービスのレスポンスが出力期待と一致する数の割合をもって品質を評価している。この手法に対し提案手法では対象が WSDL により形式的に表される SOAP 形式とは異なり REST 形式の Web サービスの品質評価による検索結果のソートを行っており、また品質評価においてユーザーに品質を指定する事で機能性・効率性のトレードオフを解消している点で異なっている。

Web サービス合成方法として西村ら[9]は階層プランニングを応用した Web サービス合成手法を提案している。この手法では Web サービスの仕様をサービス適用時・適用前後における制約条件等を交え形式的に記述した上で Web サービス合成のプランニングを行っている。提案手法では REST 形式の Web サービスの仕様をパラメータのみ記述しリポジトリ登録コストを抑えている。

7. おわりに

REST 形式 Web サービスに対し、メタ検索によって絞りこまれた対象に Web サービス合成を含むテスト実行に基づいた検索手法を提案した。本検索手法により REST 形式 Web サービスの検索精度の向上を確認した。

今後の課題としては以下が挙げられる。

1.) リポジトリ登録の際、属人性を排除するための入力支援ツールの提案

本論文で述べた通り REST 式 Web サービスのリポジトリへの登録の際において、パラメータが「必須である」「いずれかが必須である」「必須ではない」といった属性を指定するにはその Web サービスを提供している仕様書を読み登録する必要がある。この際に属人性により解釈違いを起こす可能性がある。そこで入力支援ツールにより登録の際にテスト実行を行い登録の間違いがないかをチェックすることが望ましい。

2.) Web サービス合成プログラミングを補助するコードの提示

Web サービスの出力を別の Web サービスの入力とする際、レスポンス XML を DOM によって解析し入力側の Web サービスに対しパラメータを http メソッドにより付与しリクエストを送信する事が多い。本システムでは Web サービス合成のためレスポンス XML のタグと入力側の Web サービスのパラメータの対応をとっている。これらの対応を応用し Web サービスの合成のプログラミングを補助するコードの提示が出来ることが望ましい。コードの提示により本システムによる Web サービスの選定のコスト軽減に加え、Web サービス合成における開発のコストの軽減が望めると考える。

3.) テストケースを用いたテスト駆動開発への応用

昨今テスト駆動型の開発が広まりつつある。本システムのテストケースを用いたテスト実行はテスト駆動型開発に近い指向があると考えられる。そこでテスト駆動型のマッシュアップ型開発においての応用を検討する。

参考文献

- 1) Leonard Richardson, Sam Ruby, "RESTful Web Services", O'Reilly Media, 2007
- 2) 株式会社ミクシィ, ソーシャル・ネットワークング サービス [mixi(ミクシィ)], <http://mixi.jp/>(2010.2.22 時点).
- 3) An amazon.com company, Amazon Web Services, <http://aws.amazon.com/>(2010.2.22 時点).
- 4) YouTube, YouTube - Broadcast Yourself, <http://jp.youtube.com/>(2010.2.22 時点).
- 5) 株式会社 ニワンゴ, ニコニコ動画(9), <http://www.nicovideo.jp/> (2010.2.22 時点).
- 6) メタデータ株式会社, WebAPI 比較・マッチングサービス <http://www.api-match.com/>(2010.2.22 時点).
- 7) 安藤恭平, 金子伸幸, 山本晋一郎, 阿草 清滋, "テスト実行に基づくコンポーネント検索", コンピュータ ソフトウェア, Vol. 25 2008.
- 8) W.T.Tsai, Xinyu Zhou, Yinong Chen, Xiaoying Bai, "On Testing and Evaluating Service-Oriented", vol41, No8 SoftwareIEEE Computer, pp40-46, 2008.
- 9) 西村一彦, 中川博之, 中山健, 田原康之, 大須賀昭彦, "階層プランニングによる Web サービスの自動合成", ソフトウェアエンジニアリングシンポジウム 2008, 2008.