マッシュアップを利用したWeb上におけるシステム構築について

清水, 宏泰 / SHIMIZU, Hiroyasu

(発行年 / Year)
2014-03-24
(学位授与年月日 / Date of Granted)
2014-03-24
(学位名 / Degree Name)
修士(工学)
(学位授与機関 / Degree Grantor)
法政大学 (Hosei University)

2013 年度　修士学位論文

論文題名　マッシュアップを利用した Web 上におけるシステム構築について

指導教員　藤井章博

大学院　工学研究科

情報電子工学専攻修士課程

12R4125

シミズ　ヒロヤス

清水　宏泰

目次

1．序論

現在、クラウドサービスが広く普及してきた。このクラウドサービスでは、様々なリソースが提供されている。これらのサービスの多くは、インターフェースが用意され、ユーザがブラウザ上から直接リソースを操作・管理できるようにされている。また、API を設置しプログラムからもアクセス可能にしているものもある。こうしたサービスは、Web を介して複数のサービスを連携させ新たなサービスを構築することができる。こうした手法をマッシュアップと呼び、一般にもマッシュアップによって作成された Web サービスも存在している[1][2]。このような Web サービス・Web アプリケーションをマッシュアップアプリケーションと呼ぶ。マッシュアップアプリケーションでは、既存の API やサービスを活用するため、一から開発を行うより簡単に高度な機能を実現しやすい。

こうしたマッシュアップアプリケーションの開発を行う際、一定の規格に従って API を公開しておくことで、その開発をスムーズに行うことができる。この規格として近年の Web サービス・Web API に用いられるのが REST(Representational State Transfer)である[3][4]。この REST はソフトウェアアーキテクチャの 1 つで、アドレス可能性・接続性・ステートレス性・統一インターフェースというような原則が存在する。これらに従って開発されたものを RESTful Web サービスや RESTful Web API と呼ぶ。また、近年公開されている Web API は、統一インターフェースとして HTTP を用いているものが多い。このため、Web 上でマッシュアップを行う際、新たなプロトコルや操作用のメソッドを覚える必要がなく、開発者は簡単にマッシュアップアプリケーションを作成できる。

複数の Web API をマッシュアップする際、製作者の異なる API をマッシュアップする場合もある。このとき問題となるのは、フォーマットやデータラベルといったデータの表現である。こうしたデータの差異は、マッシュアップを行う開発者が処理する必要がある。これは、マッシュアップする API の数が増えれば当然負担が大きくなり、大規模なマッシュアップアプリケーションの開発は実質的に困難である。

一方で、Web 上のデータが膨大になり、情報収集の際にデータの精査が困難になっている。そこで、データに意味を付加することで、検索の質を上げ、情報精査をある程度機械に任せる等、Web の利便性を高める Semantic Web という技術がある[5][6]。これは、RDF(Resource Description Language)や、OWL(Web Ontology Language)といった、標準的なツール群を用いてデータに意味を付加するものである。メタデータとして意味を記述することにより、そのデータを機械が解釈しやすくなり、プログラムによるデータの運用が容易になる。また、記述されるのはデータそのものの意味だけでなく、データ同士のリンクにも意味が記述できる。これにより、リンクの意味を辿っていくような高度な推論も期待される。

本論文の目的は、マッシュアップを利用して、Web 上に高度な機能を持ったシステムを構築することである。Web 上に複雑なシステムを構築することで、クライアントの端末は HTTP を利用できるソフトがあればシステムを利用できるようになる。また、データに関

しても Web 上のシステムに保管することで、複数の端末でデータを管理しながら作業する必要がなくなる。こうしたシステムを構築する際、マッシュアップを利用することで既存の Web API を利用できる。これにより、システムの開発コストを軽減することができる。前述したように、多数の Web サービスをマッシュアップすることで、複雑なシステムを簡単に構築することが可能になる。このときに発生する、データフォーマットのずれを吸収するツールとして、Semantic Web の技術が有用であると考えられる。RDF や OWL といった標準的なツール群は、メタデータを記述することでフォーマットに影響されることなくデータを処理できる。また、RDF ではデータ同士がリンクでつながっている状態なので、データのラベルが違うものでも同じものであると明示してあればそのまま利用できるという利点もある。

本論文では、2 章で Web サービスの概念とマッシュアップについて説明する。ここでは、本論文において Web サービスとは何かを述べる。また、Web サービスとマッシュアップの関係性についても説明する。3 章では、Semantic REST の説明と本論文における位置づけを行う。本論文では、Semantic REST をデータ表現の誤差をプログラムで一致させるためのツールとして利用する。2，3 章で示した内容に基づいて、4 章で提案システムに利用するセマンティックパーサについて紹介する。このセマンティックパーサは、RDF のリソースと独自の書式を用いたテンプレートをマッシュアップして新たなリソースを出力する。5 章では、4 章で紹介したパーサを利用した提案システムの構成について述べる。ここでは、システムの具体的な構成や機能の追加等の運用や、提案システムの評価について説明する。最後に、6 章でこのシステムについて考察を行う。

2．Web サービスとマッシュアップ

Web では、様々なリソースが公開されている。そうしたリソースを、Web を介して操作可能にするために Web サービスや Web API が公開されていることも多い。ここでは、そうした Web サービスや Web API についての説明を行う。また、既存の Web サービスや Web API を利用して新たなサービスを制作するマッシュアップという手法や、マッシュアップを用いてシステムを構築する方法について述べる。また、REST アーキテクチャに基づいて設計された Web API や Web サービスはマッシュアップを行うのに都合が良い。REST とマッシュアップについてもこの章で解説を行う。

2.1.REST とマッシュアップ

マッシュアップとは、既存の複数の Web API や Web サービスを利用して新しいサービスを提供する手法である。既存のものを利用するため、Web サービス開発の際に細かい機能の実装やリソースの準備を自分で行う必要がなくなる。これにより、開発の労力や時間といったコストの削減が期待できる。

マッシュアップを行う上で重要な点として、公開されているリソースに URL が振られていることと、HTTP を利用していることが挙げられる。リソースごとにユニークな識別子として URL が振られていることで、Web 上であっても特定のリソースを呼び出すことができる。また、HTTP は Web の基本プロトコルである。これを利用することにより、リソースの利用方法について、新たなプロトコルやメソッドを学習する必要がなくなる。このため、Web サービスや Web API を作成する際には、こうした性質を持つ REST アーキテクチャに沿って設計された RESTful Web サービスや、RESTful Web API として作成するのが望ましい。

REST とは、ソフトウェアアーキテクチャの一種で、以下に挙げる性質を持つ。

- アドレス可能性
- ステートレス性
- 接続性
- 統一インターフェース

マッシュアップによるシステム構築において重要なものは、先に挙げたアドレス可能性と統一インターフェースである。アドレス可能性とは、提供するリソース全てに、一意な識別子としての URI(Uniform Resource Identifier)が振られていることである。統一インターフェースとは、全てのリソースに適用できるようなよくまとまったインターフェースであり、Web サービスや Web API では HTTP が用いられる。また、1 つのリクエストで処理要求が完結するステートレス性も重要である。この性質があることにより、リソースの URL に処理要求を送信するだけでマッシュアップが行えるようになる。

マッシュアップを利用して Web 上にシステムを構築する場合、以下の図 1 のような構成になる。

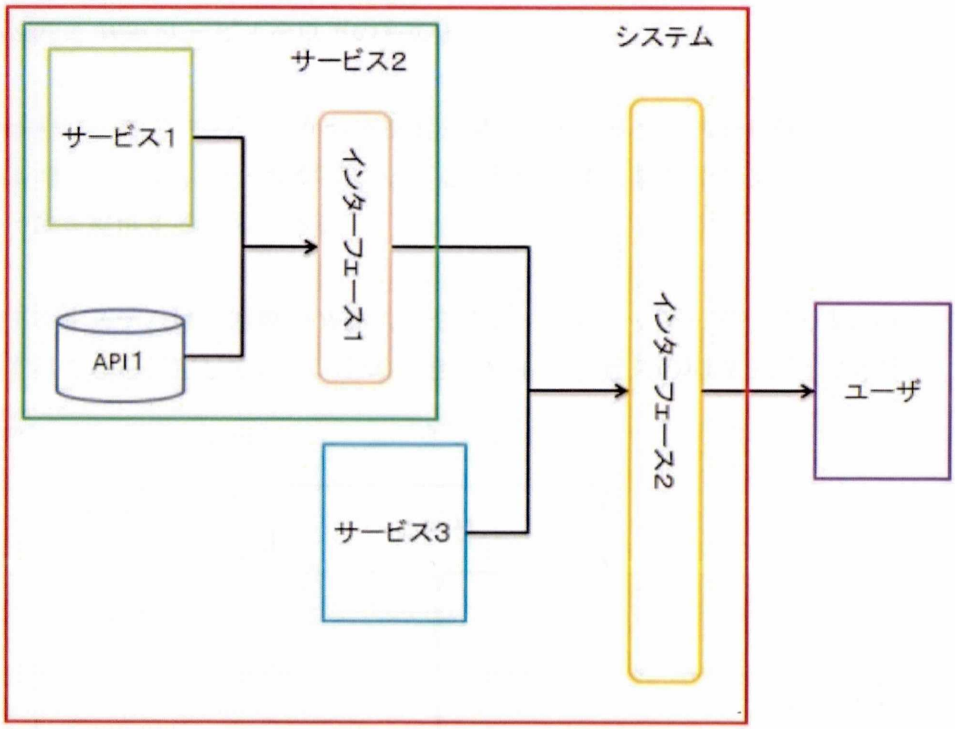

図 1　提案システムの構成

この構成では、単純なリソースを提供するサービス 1 や API1 が存在する。それらを、インターフェース 1 でマッシュアップすることでサービス 2 として公開される。これを更に、他のサービスとマッシュアップを行うことで、1 つの高度な Web サービスとしてユーザに提供することが可能になる。これは、十分な種類の単純な Web API が多数公開されていれば、それらを組み合わせて高度なサービスを構築できるということである。これは、システムに利用するデータや機能を URL として記述することが可能になる。これは、URL を追加・修正・削除することで、自由にシステムの機能を変更できるということである。また、必要なリソースや機能を持ったものが公開されていれば、開発者はインターフェースのデザインや機能に集中できる。

2.2.Web サービスと WebAPI

本論文では、Web API をマッシュアップしてシステムを構築する。このため、利用する Web API と Web サービスは REST に準拠したものを想定している。ここで、本論文における Web API と Web サービスを以下のものとする。

- Web API とは、プログラムからアクセス可能な形でリソースを提供しているものである。
- Web サービスは、ユーザがリソースに対して操作可能なインターフェースを付けたリソースを提供するものである。

構築するシステムは、複数の Web API やリソースをマッシュアップし Web サービスの形で提供することを想定している。このときの Web サービスの構成を図 1 に示す。

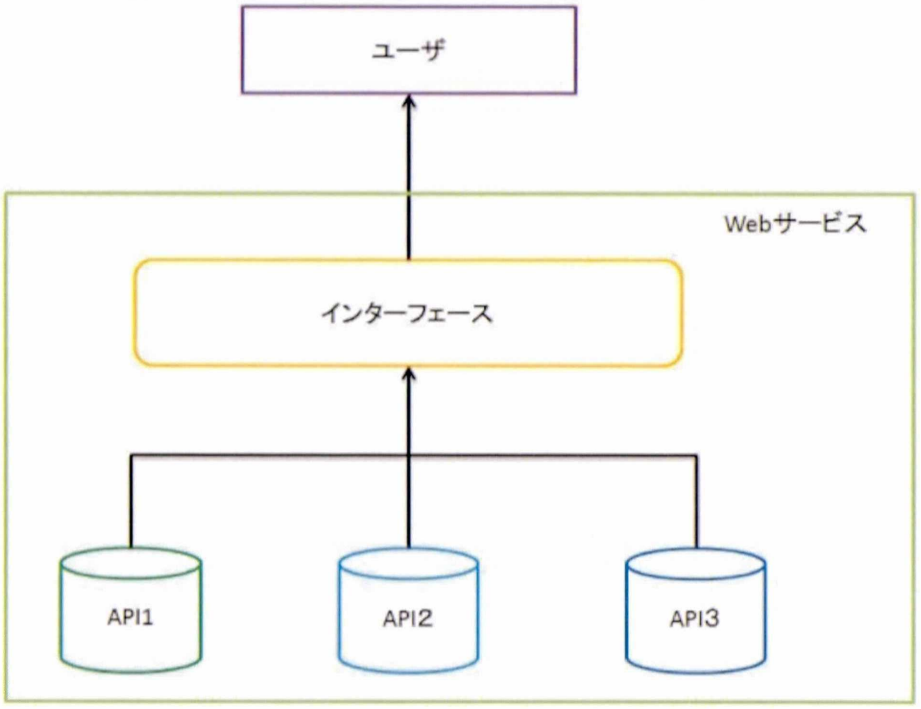

図 2　Web サービスの構成

マッシュアップは、開発コストの削減や柔軟性の高いサービス開発といったメリットが予想される。しかし、利用が想定される既存の API やサービスは製作者が異なることが想定され、プロトコルやリソース等に対する操作メソッドが違うことも考えられる。これに対し、本論文では RESTful Web サービスや RESTful Web API を利用することを想定している。このため、利用するプロトコルは HTTP であり、リソースは HTTP メソッドで操作可能なものとして考える。

2.3.マッシュアップを利用したWeb上でのシステム構成

マッシュアップでシステムを構築することで、システムのリソースや機能といったシステム構成はURLで表現できるようになる。これにより、システムに高い拡張性を与えることができる。しかし、利用する複数のWeb APIの開発者が異なる場合、提供するリソースのフォーマットが異なっていることが考えられる。また、同じデータを指していてもデータのラベルが異なることも考えられる。本論文では、これをデータ表現の差異と呼ぶ。データ表現の差異は、利用するAPIの数が1つや2つといった小規模ならば、開発者がそうした差異に対応すればよい。しかし、本論文のように高度なサービスや複雑なシステムを構築する場合、10や20のAPIやサービスを利用する大規模なマッシュアップが予想される。こうした大規模なマッシュアップを行う際、利用する全てのAPIやサービスのデータ表現を把握し、調整することは困難であると考えられる。これを解決するためのツールとして、本論文ではSemantic Webを用いる。次章では、Semantic Webの技術によるリソースを、RESTアーキテクチャで提供するSemantic RESTについて説明していく。

3．Semantic REST

Semantic REST とは、Semantic Web の技術で生成されるリソースを REST アーキテクチャで公開するものである。本論文では、データ表現の差異を解消するツールとして Semantic REST を利用する。これにより、データ表現の差異を気にせずマッシュアップが容易な Web API や Web サービスを作成することが可能になる。本章では、Semantic REST について述べる。また、Semantic Web で利用するツールである RDF やそのフォーマットについて解説する。その後、本論文における Semantic REST の役割について説明する。

3.1.Semantic Web と REST

Semantic Web とは、データの意味情報を記述することでプログラムによる処理を容易にする技術である。また、意味情報を付与したデータを Web 上で公開するものでもある。意味情報は、主語・述語・目的語からなるトリプルという構文で記述される。このとき、目的語を主語として新たなトリプルを繋げることができる。こうして、多数のトリプルを繋げることにより、図 3 のように巨大な意味ネットワークを形成することができる。

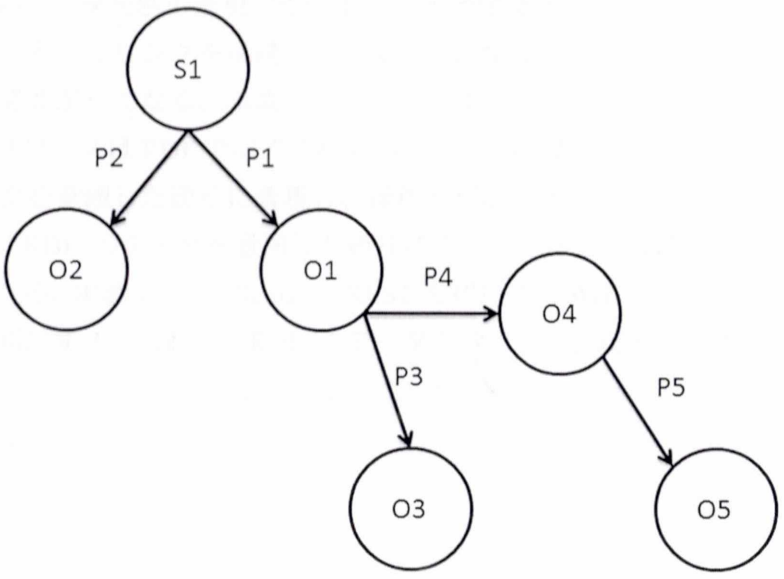

図 3　トリプルを連続させたグラフ構造

Semantic Web では、意味ネットワークをグラフとして扱うことで、テキストベースの検

索よりも精度の高い検索や高度な推論が可能である。意味情報の記述には、RDF や OWL といった標準のツールを用いる。

記述された意味情報もリソースの 1 つであり、マッシュアップに利用できる。このため、意味情報も REST アーキテクチャによって提供されることが望ましい。REST アーキテクチャで意味情報を提供することで、データ表現の差異をプログラムによって調整し、マッシュアップを行うことができる。

3.2.RDF フォーマット

意味情報を記述する標準ツールとして RDF がある。RDF はフレームワークであり、様々なフォーマットが用意されている。RDF 特有のフォーマットである N-Triples や N3、XML で記述される RDF/XML 等がある。また、HTML に記述するものとして、RDFa(Resource Description Framework in attribute)や microformat といったものもある。

3.3.ツールとしての Semantic REST

RDF の体系では、意味情報を主語・述語・目的語のトリプルで管理する。これは、表現したい物に URI を割り振り、2 つの物の関係を述語によりリンク付けするものである。これにより、同じ意味のデータを別の表現で提供する API が複数存在しても、利用するデータ間に「同義である」というリンクを接続してしまえば、マッシュアップの際にデータ表現の差異を考慮する必要がなくなる。本論文では、この性質を利用してシステムを構築するため、利用できるリソースは RDF で記述されていることが前提となる。また、RDF を解釈し、解釈したデータを意図した通りに表現し、操作を可能にするインターフェースが必要となる。あるいは、RDF のデータを意図した通りにインターフェースに組み込むアプリケーションが必要になる。本論文では、Semantic REST を利用する Web サービスとしてセマンティックパーサを利用する。これは、RDF のデータを意図した表現としてインターフェースに組み込む機能がある。これを利用することで、大規模なマッシュアップにより複雑なシステムを構築することができる。

4．セマンティックパーサ

RDF を解釈し、複数のリソースをマッシュアップして新たなリソースを出力するツールとして、セマンティックパーサがある。本論文では、これを利用して様々なリソースとインターフェースをマッシュアップして、複雑なシステムを 1 つの Web サービスとして公開する。以下では、セマンティックパーサが本論文で提案するシステムにおいてどのような役割を果たすかを述べる。次に、セマンティックパーサの機能について説明する。その後、セマンティックパーサを利用して、インターフェースのロジックへリソースに埋め込む手法について説明する。

4.1.提案システムにおける役割

システム構築において、インターフェースかそれを出力するツールが必要になる。本論文では、インターフェースを出力するツールとして、セマンティックパーサを利用する。セマンティックパーサは、Semantic REST による RDF データの活用で紹介されている Web サービスである[7]。これは、HTML を RDFa に変換するためのサービスである。また、RDF/XML を任意の表現に変換するための機能もある。

図 1 の Web サービスの構成にパーサを組み込んだものについて図 4 に示す。

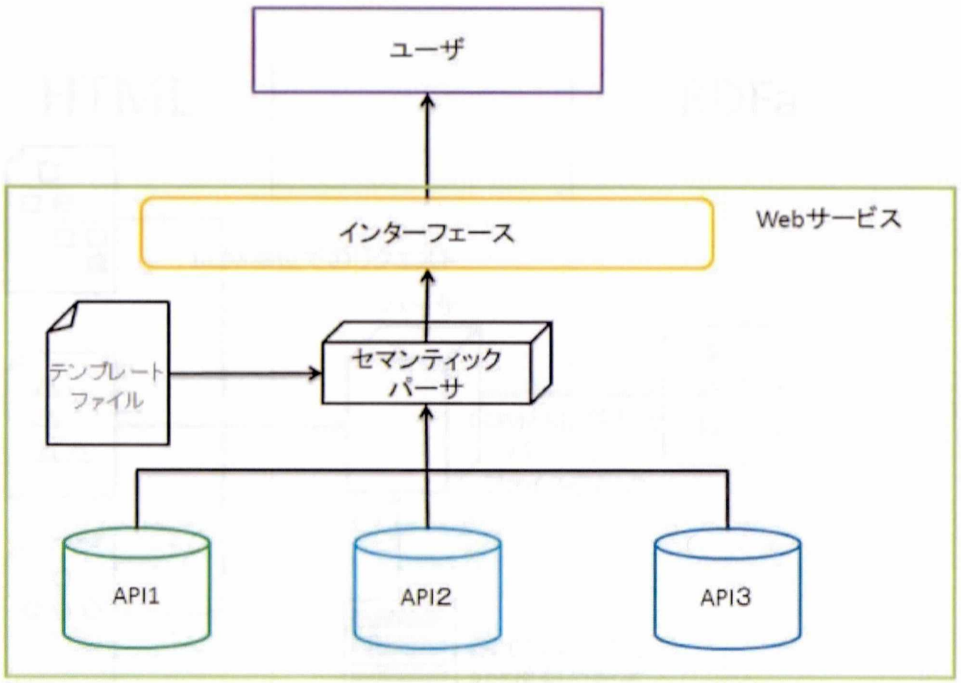

図 4　パーサを利用した Web サービスの構成

セマンティックパーサは、複数の RDF リソースと、テンプレートファイルをマッシュアップすることで新たなリソースを出力する。このとき、テンプレートファイルにリソースを操作するような、インターフェースとしてのロジックが記述されていれば、図 4 に示すように複数の RDF リソースを提供する API とテンプレートファイルから、Web サービスとしてのインターフェースを出力することが可能になる。

本論文において、Web サービスはリソースに対して操作可能なインターフェースを付与した形で提供するものとしている。これは、Web サービスがリソースとインターフェースからできているということである。これに対し、セマンティックパーサは複数のリソースとテンプレートをマッシュアップして新たなリソースを生成し提供する。このとき、テンプレートがリソースを操作するためのインターフェースであれば、新たなリソースは Web サービスであると言える。つまり、セマンティックパーサは、既存のリソースやテンプレートを用いて Web サービスを生成するサービスであると言える。

4.2.利用する機能

まず、セマンティックパーサの機能を紹介する。セマンティックパーサには大きく分けて 2 つの機能がある。1 つ目の機能は、図 5 に示すように HTML と JSON をパーサに入力することで RDFa を出力する機能である。

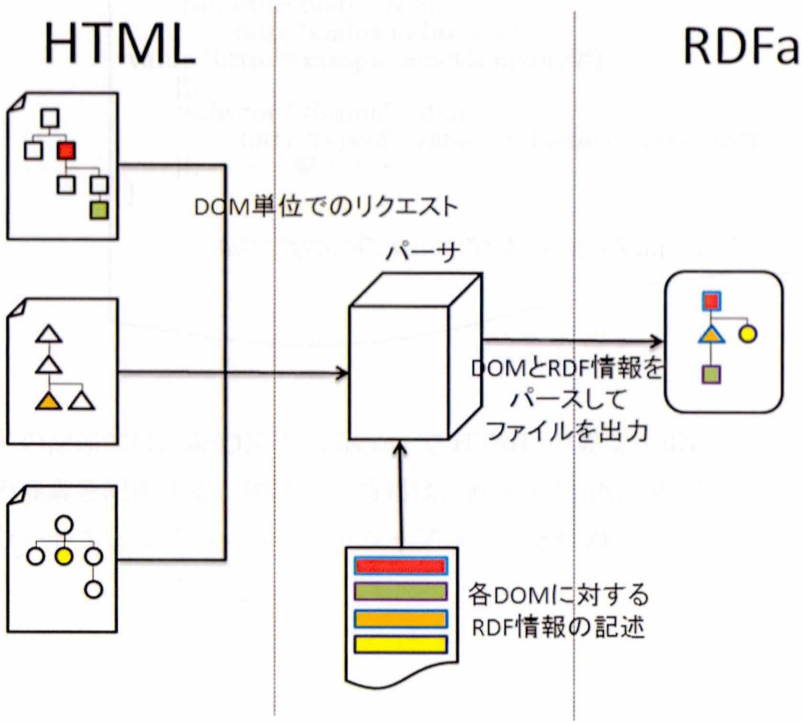

図 5　セマンティックパーサの機能 1

出力される RDFa は、入力した HTML に JSON で記述されたセマンティック情報を埋め込んだものである。JSON には、HTML のタグ名やタグで利用している ID、クラスを指定し、対応するタグの属性として意味情報を埋め込む。具体的な例として、[7]で紹介されている会社情報の RDFa 化を説明する。まず、HTML で記述された会社情報のコードをソースコード 1 に示す。また、ソースコード 1 にセマンティック情報を埋め込むための、JSON で記述された RDF 情報をソースコード 2 に示す。

ソースコード 1　元の HTML

```
<div class="thumb">
        <span class="cname"> KT 社</span>
        <div class="address">
          <span class="postcode">○○○-○○○○</span>
          <span class="site">東京都墨田区向島××</span>
        </div>
        <div class="introduction">  ・・・略・・・</div>
</div>
```

ソースコード 2　HTML へ RDF 情報を埋め込むための JSON

```
items:[
      {selector:"body", rdfas:[
            {attr:"xmlns:industory",
value:"http://example.com/Company#"}
      ]},
      {selector:".thumb", rdfas:[
            {attr:"typeof", value:"industory:Company"}
      ]},  ・・・略・・・
]

          {attr:"typeof", value:"industory:Company"}
```

コードの記述法は、JSON の selector で HTML の要素を指定し、rdfas の値に埋め込む RDFa の要素を記述する。埋め込む情報は、attr = “value”の形で埋め込まれる。例示したソースコード 1 と 2 をパーサでマッシュアップしたものをソースコード 3 に示す。

ソースコード 3　パーサが出力する RDFa

```
<div class="thumb" typeof="industory:Company">
      <span class="cname" property="industory:name"> KT 社</span>
      <div class="address" rel="industory:exist">
       <span class="postcode" typeof="industory:Postcode" property="industory:name">
〇〇〇-〇〇〇〇</span>
      <span class="site" typeof="industory:Site" property="industory:name">東京都墨
田区向島××</span>
      </div>
      ・・・略・・・
</div>
```

今回の例では、ソースコード 2 で thumb クラスのタグに、attr:"typeof" value:"industry:Company"を指定している。これにより、ソースコード 3 のように、HTML の thumb クラスの div タグに、属性として typeof="industry:Company"が埋め込まれた RDFa が出力される。

2 つ目の機能は、RDF/XML とテンプレートを入力することで、入力した RDF/XML のデータを任意表現に成型したリソースが出力される機能である。この機能の説明として、人物紹介のリソースを生成する例を図 6 に示す。

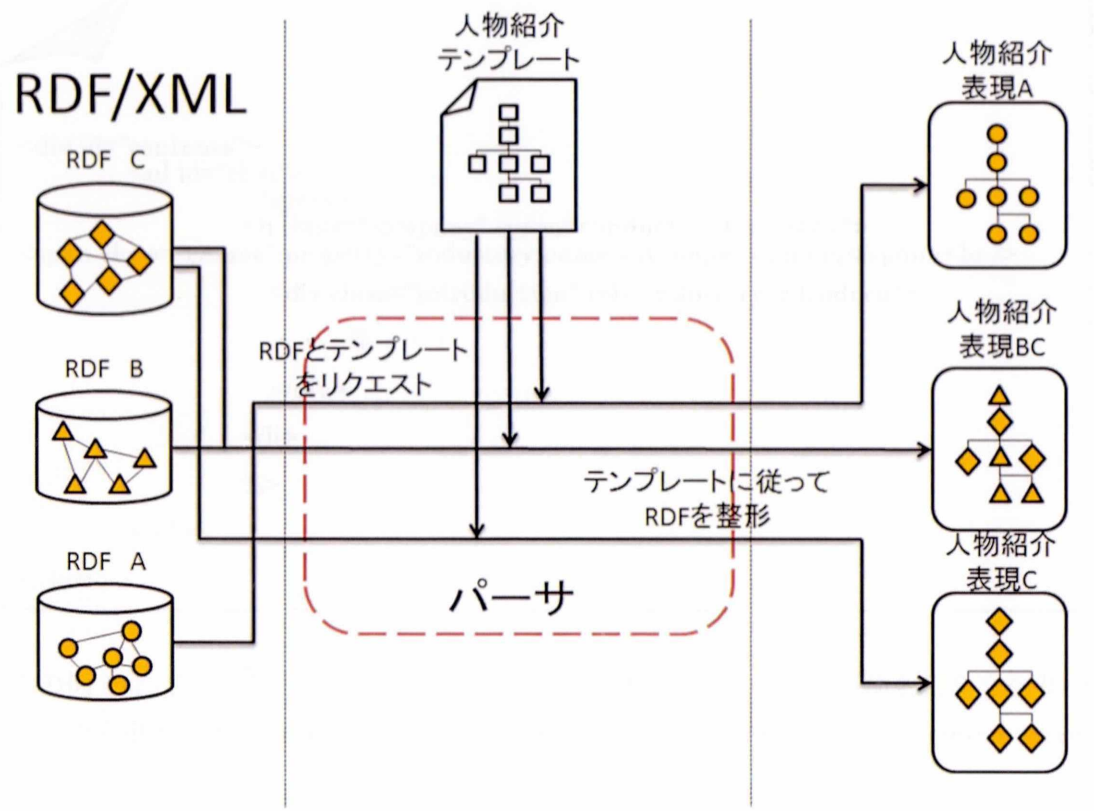

図 6　セマンティックパーサの機能 2

テンプレートは、入力する RDF/XML をどう表現するかを記述する。このとき、{クラス名.プロパティ名}の記述によって、データを埋め込む位置を指定する。今回の例では、人物紹介のテンプレートを利用することで、パーサに入力する URL を変更することで、内容を RDF A、B、C と切り替えることができる。また、RDF B と C 両方の URL を入力することで、2 つのデータを人物紹介の形式で出力することもできる。パーサは、テンプレートの表現次第で HTML や XML 等任意のフォーマットでデータを出力することができる。テンプレートの記述例として、[7]に示されている企業情報を出力するコードについて解説する。

ソースコード 4　企業情報の RDF/XML

```
<rdf:Description rdf:about="http://example.com/Company#〇〇〇-〇〇〇〇">
        <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#NamedIndividual"/>
        <rdf:type rdf:resource="http://example.com/Company#Postcode"/>
        <name>〇〇〇-〇〇〇〇</name>
      ・・・略・・・
</rdf:Description>
<rdf:Description rdf:about="http://example.com/Company#Company_MG">
        <rdf:type rdf:resource="http://example.com/Company#Company"/>
        <name>MG 社</name>
        ・・・略・・・
</rdf:Description>
```

ソースコード 5　名前と紹介文の一覧を表現するテンプレート

```
<div id="contents">
        <ul id="clist">
                <%parse
                <li class="company" typeof="industory:Company">
<span class="cname" property="industory:name">{Company.name}</span><br />
                  <div class="introduction" rel="industory:introduce">
                   ・・・略・・・
                  </div>
                </li>
                %>
        </ul>
</div>
```

この例では、名前や住所、紹介文等が含まれる企業情報から、企業の名前と紹介文を RDFa の形式で表現する例である。ソースコード 5 のように、テンプレート内に{Company.name}の部分に、ソースコード 4 の
<rdf:type rdf:resource=http://example.com/Company#Company />が記述されている rdf:Description の子要素である name の内容が埋め込まれる。ソースコード 4 と 5 をパー

サに入力することで得られる RDFa をソースコード 6 に示す。

ソースコード 6　名前と紹介文が記載された RDFa

```
<div id="contents">
    <ul id="clist">
     <li class="company" typeof="industory:Company">
<span class="cname" property="industory:name">&amp;quot; A 社&amp;quot;</span><br />
      <div class="introduction" rel="industory:introduce">
          ・・・略・・・
        </div>
    </li>
     ・・・略・・・
    </ul>
</div>
```

提案システムに利用するのは 2 つ目の機能である。この機能を利用することで、インターフェースのロジックやデザインはテンプレートに集約されることになる。このため、システム内でリソースとインターフェースを分離して管理・運用することが可能になる。

4.3.ロジックへのリソースの埋め込み

パーサの機能を利用してシステム構築を行うには、インターフェースとしてのロジックを持ったテンプレートが必要になる。また、そのテンプレートのロジックが必要なリソースを操作可能にする必要がある。パーサを利用してこれを実現するには、大きく分けて 2 通りの方法が考えられる。1 つ目の方法は、ロジックが読み込み可能なリソースを生成して、それを呼び出す方法である。これを、以下の図 7 に示す。

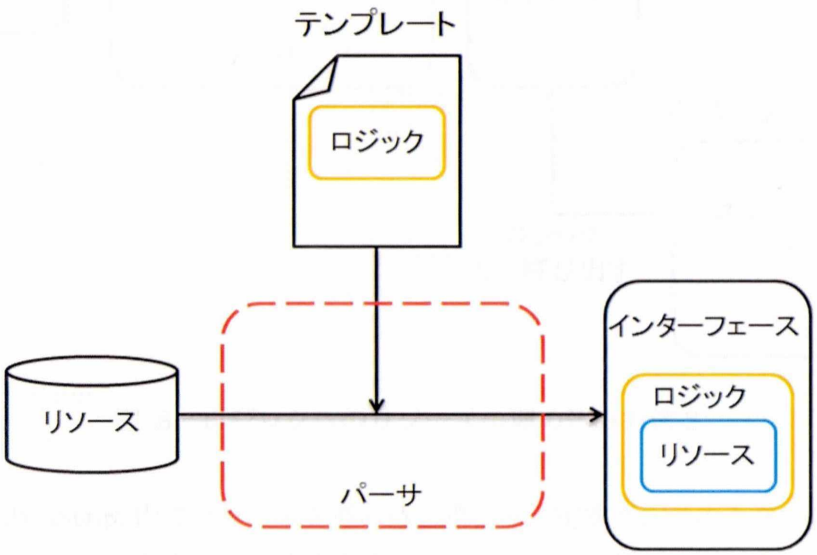

図 7　ロジックへのリソースの埋め込み方法 1

例としてテンプレート内に JavaScript でロジックを記述し、そのロジック内でリソースを呼び出す場合を想定する。これは、パーサを利用して、リソースを JSON のような JavaScript が読み込める形で生成する。あとは、JavaScript から生成されたリソースを保管している URL を呼び出すか、そのリソースを生成する URL を呼び出すことでシステムのインターフェースを生成することが可能になる。

2 つ目の方法は、テンプレートのロジック内にリソースを埋め込む記述をする方法を想定する。このときのインターフェースとリソースの関係を図 8 に示す。

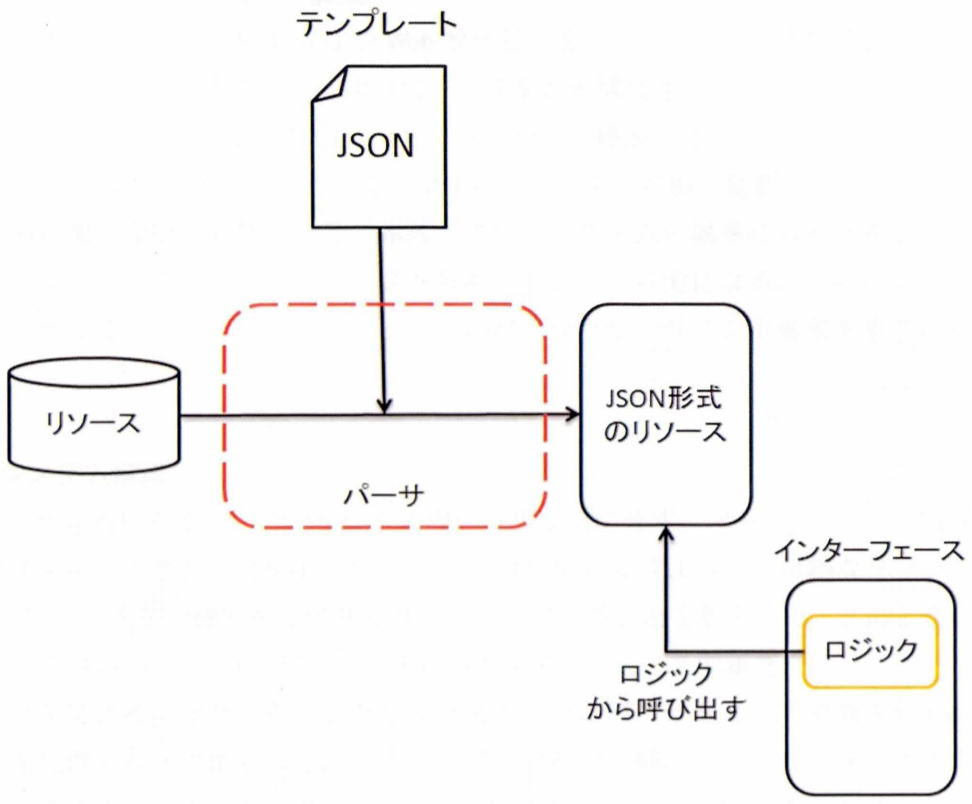

図 8　ロジックへのリソースの埋め込み方法 2

例として、JavaScript 内にリソースを埋め込む場合を想定する。このとき、ロジック内に array = {example.name} のように記述することで、パーサを介してロジックにリソースを埋め込むことができる。本論文では、後者のロジック内にリソースを埋め込む方法でシステムを構築することを想定する。パーサを利用することで、テンプレートとリソースをマッシュアップして、リソースを操作可能なインターフェースを持ったリソース出力できる。これは、パーサがマッシュアップを行い、Web サービスを出力していると言える。つまり、セマンティックパーサによって、データ表現の差異を意識することなく、マッシュアップを利用して Web サービスを生成できるということである。

5．クラウドシステムの構築

本論文では、単純な Web API や Web サービスをマッシュアップで組み合わせることで複雑なシステムを構築する。このように、システムを構成するリソースや機能が Web 上に分散しているものを、本論文ではクラウドシステムと呼ぶ。このシステムは、構成が URL で表現でき、URL を変更することで利用するリソースの追加・変更や、組み合わせによって機能の追加・変更が可能になる。本論文では、システムの構築にパーサを用いる。これにより、リソースとインターフェースを分離することが可能になる。このため、システムに影響を与えることなくインターフェースのロジックやデザインを編集することが可能になる。

5.1.システム構成

パーサを利用することで、データ表現の差異を気にせず、マッシュアップにより Web サービスを作成できることを示した。ここで、図 9 に示すように、単純なリソースを提供する API1 や、簡単な機能を提供するサービス 1 があることを想定する。このとき、簡単なサービスや API をパーサによって、任意のインターフェースで新たなサービスとして提供することができる。また、そうしたサービスを、更にパーサを介して連携させて高度なサービスを提供するが可能となる。このように、複数の単純なリソースやサービスを、マッシュアップすることで、より複雑で高度なシステムを、1 つの Web サービスとして提供することができる。

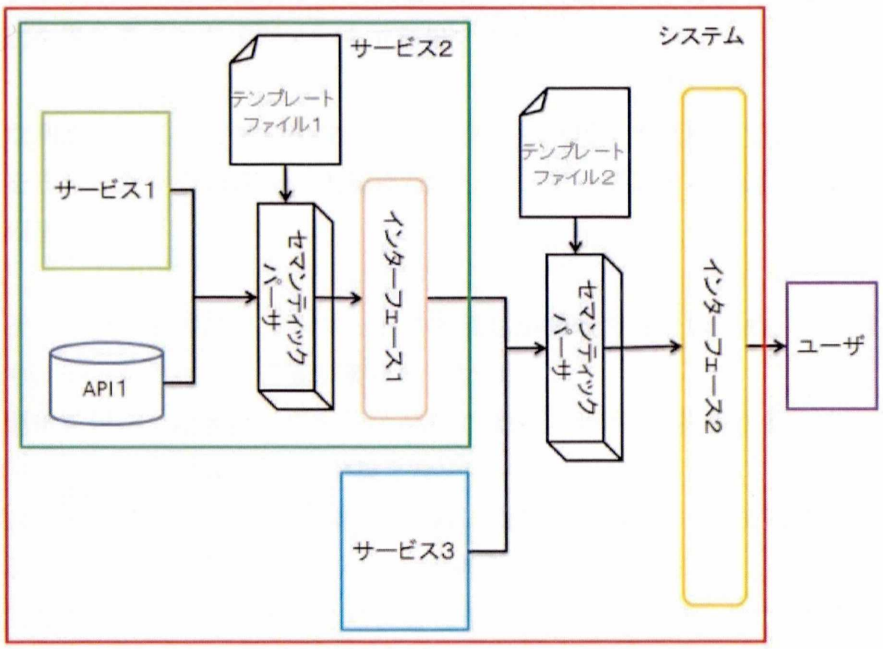

図 9　複数サービスによるシステムの構築

このシステムは、利用する各サービスを機能として捉えている。このとき、システム構成はパーサが読み込む URL として表現できる。この URL を変更することでシステムの構成も変更が可能である。このため、パーサを利用して構築されたシステムは高い拡張性を確保できる。また、システムは Web 上に存在するため、HTTP を利用できるアプリケーションが動作すれば、どんな端末でも同じ機能を利用できると考えられる。更に、パーサを介することにより、利用するテンプレートファイルを変更することで、複数の表現でインターフェースを出力できる。これにより、リッチクライアントに対してはリッチなインターフェースを提供できる。逆に、シンクライアントに対してはシンプルなインターフェースを提供が可能になる。このように、ユーザは端末に依存せず、同じサービスを最適な形で利用できるようになる。本方式によりシステムを構築することで、高い拡張性と相互運用性を備えたシステムを提供することが可能になる。

5.2. システムの評価

提案システムは、セマンティックパーサを利用して、マッシュアップによりシステムを構築する。このときの利点を以下に列挙する。

- URL の追加・変更により、利用するリソースの追加・変更が行える。
- URL の編集や組み合わせにより、システム構成の変更や機能の変更が行える。
- Web 上で構築することにより、Web ブラウザを介して様々な端末で同一のシステムを利用できる。
- セマンティックパーサによりリソースとインターフェースが分離されることで、同一システムを異なるインターフェースで利用できる。

このような利点がある一方で、セマンティックパーサだけでは解決できていない点や、実際の運用を想定した場合の問題、大規模なマッシュアップによる問題も存在する。この点を以下に列挙する。

- セマンティックパーサはフォーマットの差異しか吸収できないため、データ表現の差異のうち、データラベルの差異が吸収できない。
- 実際の運用を想定した場合、提案システムでは、システムを構築するための Web API や Web サービスが既知のものしか扱えない。
- 大規模なマッシュアップを行う場合、頻繁に通信が行われることが予想できるため、通常のシステムよりも遅延が大きくなると考えられる。
- 多数の既存 Web API や Web サービスを利用するため、各 API やサービスの稼働率や公開期間といった信頼性が保証されない。

6．結論

セマンティックパーサを利用してシステムを構築することで、データ表現の差異を意識することなく Web 上に高度なシステムを構築できる。このシステムは、パーサに入力する URL によりリソースや機能の拡張ができる。また、Web ブラウザが動作する端末であれば同じシステムを利用できる。本方式でシステムを構築することで拡張性や相互運用性が確保される。また、パーサを介することでインターフェースも URL で指定できるようになっている。これにより、タブレット端末やデスクトップ PC のような異なるプラットフォームで利用する際に、端末ごとに適切なインターフェースでシステムを提供できる。また、ユーザがインターフェースを用意し、それを利用するよう URL を変更すれば、ユーザがインターフェースをカスタマイズしてシステムを利用することも可能である。

今後の課題として、本研究で利用するセマンティックパーサではデータ表現の差異を吸収しきれていない点が挙げられる。これは、セマンティックパーサがフォーマットの差異を吸収する機能しか実装されていないためである。このため、本研究で述べているシステム構築に際して、データラベルの差異を吸収するようなサービスや API を作成し、セマンティックパーサとマッシュアップさせる必要がある。また、実用のサービスを想定する場合、API やテンプレートが存在する URL を発見・管理するサービスが存在するとより便利になる。こうしたサービスが存在すれば、システム構築の際、そこからサービスを検索しマッシュアップすることが可能になる。

本研究では触れていないものの、大規模なマッシュアップを行う上で通信速度の問題や、公開されているサービスや API の稼働率や、いつまで公開されているかといった信頼性の問題も存在する。こうした問題は、ローカルネットワーク上で利用するといった運用の工夫によりある程度解決できると考えられる。グローバルネットワークでの活用は、試験的に公開しデータを取得したり、技術革新や議論を重ねて解決策を模索していく必要があると考えられる。

謝辞

本研究を行うにあたり、法政大学理工学部応用情報工学科の藤井章博准教授には、研究に対する提案や意見を頂き、大変お世話になりました。厚く御礼申し上げます。

また、本研究に対するご助言を頂き、苦楽を共にした法政大学藤井研究室のゼミ生の皆様に感謝いたします。

本研究に携わっていただいた皆様、本当にありがとうございました。

参考文献

1) kehi.biz, doodle Beta, http://doodle.st/, 2014/3/4 閲覧
2) convivial-web, 子育て MAP, http://convivial-web.com/kosodate/, 2014/3/4 閲覧
3) Roy Thomas Fielding, Architectural Styles and the Design of Network-based Software Architectures,博士論文, カリフォルニア大学アーバイン校, 2000
4) 山本陽平, Web を支える技術 —HTTP, URI, HTML, そして REST, 技術評論社, 2010-4
5) W3C, Semantic Web – W3C, http://www.w3.org/standards/semanticweb/, 2014/3/4 閲覧
6) Toby Segaran, Colin Evans, Jamie Taylor, セマンティック Web プログラミング, 大向一輝, 加藤文彦, 他訳, オライリー・ジャパン, 2010-6
7) 清水宏泰, 藤井章博, Semantic REST における RDF データの活用, 情報処理学会論文誌, Vol.55, No.2, pp.865-873(2014-2)