Semantic Service Design Framework for Mobile Business Applications

Girum Bizuayehu
Tulane International, Addis Ababa, Ethiopia
girum.bizuayehu@gmail.com

Fekade Getahun
Department of Computer Science, Addis Ababa University, Ethiopia
fekadegetahun@gmail.com

Abstract

Semantic web services conceal the promises of service automation. By service automation, we mean to mechanize those service tasks such as *service discovery*, *invocation*, and *composition*. As services increase in number and type, manual searching and binding become practically infeasible if not impossible. Therefore, the automation of service tasks remains as a first level requirement for currently growing service based systems. Ontological injecting of semantics to the mostly syntactic only service descriptions is the best kept secret of enabling automation of web services. The requirement becomes even more acute when one considers consuming services from a mobile device, as dynamism and context changes in favor of user mobility are realties of mobile clients.

Based on a recent effort in the Semantic Web Service community, i.e., WSMO-Lite ontological description of services, this work proposes a light weight conceptual architecture for semantically augmented business services to be consumed from a modern smart phone. User *goal capturing* and *discovering available services* in accordance with user's intentions are the major focuses of the work. For this realization, a light weight HTML 5 based user goal, preference, and context capturing scheme are proposed. Moreover, for the service discovery pivotal task, a variant of multilevel service discovery algorithm is implemented that looks for matching of ontological service description at three relevant levels.

Keywords: Semantic Web Service; Service Automation; Ontology; HTML 5

1. Introduction

In the dawn of the most anticipated "Internet of Things", service based software systems deployment remains at the forefront. In such cases, Service Oriented Architecture (SOA) is one of the preferable architectural styles for the development of business applications [1] as it puts a service at its core center. Service is a piece of programmable logic offered by a provider for another entity, a consumer, without revealing its internal mechanics [2]. The service has an address or end point and accepts what it needs as an input; it will be easier to interact with it without peeking to the inside. According to [3], the benefits of SOA include *Increased Interoperability, Increased Business and Technology Domain Alignment, Increased ROI, Increased Organizational Agility, and Reduced IT Burden*.

In order to realize these promises a service has to be implemented in two parts: (1) Service contract - description & configurations of the services formatted in XML flavor as WSDL [4] and SOAP [5]; (2) Service implementation - piece of programming logic implemented in any language and platform one desires.

Currently there are a number of XML based standard protocols known as ws.* protocol stack used to describe the service contract. The common protocols are XML-RPC, WS-Addressing, SOAP, WSDL, UDDI, and BPEL. However, since most of these protocols are structural XML descriptions and protocols like SOAP introduce yet another layer of specification on top of the web, they face two main problems:

1. Rely only on syntax for searching a service; in essence unable to scale well.
2. Protocols (e.g., SOAP) in the stack (WS.*) are heavy weight as the protocol follows a separate implementing specification.

When the services increase in number and type, the problem associated with manual and syntax only search for a service is most visible. In addition, early or design time service binding to a client may cause

"service unavailability" due to the removal of a service or malfunctioning of a service at run time.

These problems are very visible especially in the realm of computing in mobile environment where contexts are dynamically altered and need for late binding between service and the client is of paramount importance.

Considering a mobile platform, there need to be a lightweight and more web friendly framework for delivering the promise of the semantic web to the pillars of Service Oriented Architecture (SOA). Existing frameworks like SESA [6] become too heavy to be usable in the context of mobile clients. The concern of *goal transformation*, a means of communication between the machine and the user is a crucial step to increase semantic understanding. This goal transformation has a direct link with the service discovery scheme used.

This paper proposes a conceptual framework that blends semantic web initiatives to SOA in the context of mobile platforms. For the goal transformation and context identification, a lightweight approach that relies on a custom Microformat; hGOAL, hPREFERNCES, and hCONTEXT, is realized. Moreover, a service discovery algorithm that considers a layered ontological description of the service is proposed. To gain the interoperability advantage, the framework relies on the ontological descriptions of WSMO-Lite [7].

The remainder of this paper is organized as follows. In Section 2, motivating scenario is presented. Section 3 discusses related works. Section 4 covers our conceptual framework with its components. Section 5 presents prototype and experimental results. Finally, Section 6 concludes the paper and draws some future research directions.

2. Motivating Scenario

In order to show key problems associated to this paper, let us consider a hypothetical shopping mall scenario. Alice, carrying her smartphone, walks into a local shopping mall to buy a surprise present/gift for her latest acquaintance. She has registered her preferences, and maximum allocated budget. She wants to know the list of items her friend prefers and likes. In addition, she wants to get a feed specifying the availability of a service that supports her goal when she approaches the mall. If she acknowledges the notification, she wants to see services offered in the mall and an interaction scheme that elevates her shopping experience is displayed in her screen in user amenable manner. Right from the screen she wants to compare prices, check customer rating and recommendation, perform order directly from the device, pay and notify a shipping agent about its delivery, and share her experience with friends.

The scenario demonstrates the SOA based interaction between the various stakeholders in the shopping mall example, mainly initiated and dictated by Alice's goal and preference. Figure 1 shows a high-level system interaction and a bidirectional arrow is a representation of communication between the service ready members. In order for services to be discovered automatically, the first side of the story focuses on "How to process what she (Alice) wants to do?" From the other spectrum, the middleware in the server side is tasked in the overall coordination of the services. However, the service contract could be different both in name, description, and format and yet might be designed to achieve the specified goal. Thus the main concern is the "semantic identification of services that suite to the need of the user". In addition, based on the level of complexity of the business addressed, there may also be a need to dictate the creation of composite services out of existing services, which is known as composition of services. The above scenario will not be handled with the existing enterprise SOA framework as the approach is heavy weight as explained in Section 1.

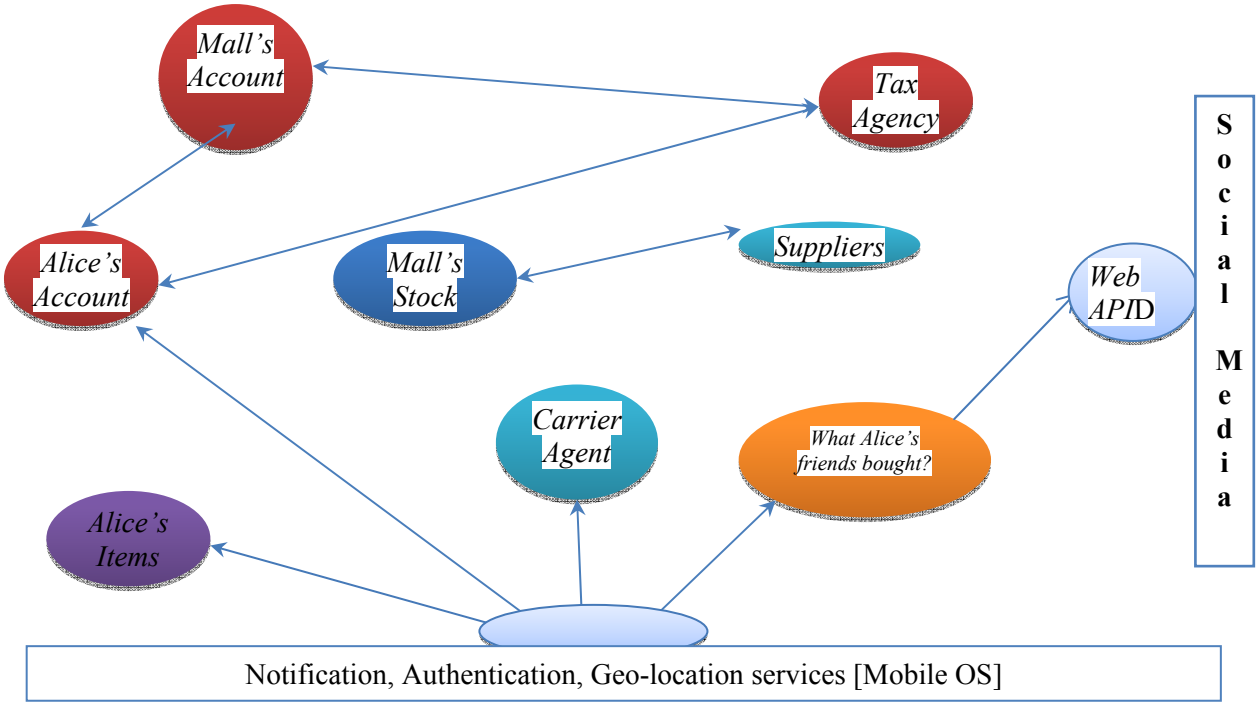


Figure 1: Motivating Scenario

3. Related Work

Various research endeavors are ongoing both by the academia and the industry 6, 8, 9] to create a semantically enhanced service framework. In this section, the technical basis this framework relies on, i.e., the WSMO-Lite service annotation is overviewed. Moreover, the most relevant related works are reviewed.

Automation of the basic routines in service based software systems is one way of avoiding the previous problems. Automating service discovery, composability, invocation, and mediation take SOA to its full potential and avoid the ever increasing burden on the user or software agent that has to search for a service and invoke it. The semantic web has already achieved a noticeable success in terms of linking data semantically for the web platform [10]. Languages, tools, and conceptual frameworks in semantic web are also maturing. From the perspectives of services, there are various successful initiatives that attempt to enrich the descriptions of services with semantic such as WSMO [11], OWL-S [12], and WSMO-Lite [7]. These initiatives demonstrated a considerable success in injecting semantics to a service description to enhance its basic routines. The first two approaches, WSMO and OWL-S, are heavyweight that put "semantics first approach". That means both approaches need an ontological description of services before realizing the framework. WSMO-Lite is lightweight semantic description for services on the Web, which adopts specific elements of WSMO ontologies and it follows a "bottom- up" approach to service annotation in that it doesn't demand a full scale ontology description up front. It can use a lowering (converting to syntax only WSDL's description) and lifting (linking to a service ontology through a reference).

WSMO-Lite performs semantic annotation of web service description on the following three ontologies:

1. *Functional*: captures the capability, condition, and effect of the web service and depicts the functionality of a web service.
2. *Informational*: captures the information aspect of the service, the input, output, and fault messages.
3. *Non Functional*: captures the non-functional aspects like QoS attributes.

The Semantically Enabled Service-oriented Architectures/SESA [6] proposes a semantic framework that realizes major aspects of SOA. The global view of the architecture shows five related layers: (1) stakeholders layer: forming several groups

of users of the architecture, (2) problem-solving layer: responsible to build environment for stakeholder access to the architecture, (3) service requesters: represents client systems of the architecture, (4) middleware layer: responsible to provide intelligence for integration and interoperation of business services, and (5) service providers: responsible for exposing the functionality of back-end systems as business services. SESA addresses various issues of SOA automation using these components. However, in the context of a mobile platform that demands high level of dynamism through context changes, the SESA framework is unable to address the following concerns.

(1) Service interoperability and integration: the execution environment SESA is based on is WSMX [13], one of the popular approaches towards the SWS initiative. Nonetheless the ontology building scheme adheres to one specific approach, i.e., the WSMO [11], ignoring other enterprise level SWS annotations like OWL-S [11], WSMO-Lite [7], and SA-WSDL. Thus SESA faces interoperability among the other popular ontologies.

(2) SESA targets heavyweight enterprise level endpoints and it is infeasible for mobile application platforms where the devices demand to be part of the SOA cycle and their context change is considered by the middleware.

Moran in [9] showed reference architecture for Semantic SOA (SSOA) [14] with a deeper and more comprehensive architectural framework. The author tried to identify essential set of elements, properties, and constraints required to define a reference Semantic Service Oriented Architecture and explored the capability of existing frameworks in supporting the dynamic and automation requirements of SOA routines. However, the reference architecture is too broad, as it attempts to address various stakeholder's (such as developers, domain experts, goal owners, and system admins) demand and different steps of SOA like complex composition, process mediations, and orchestration have been incorporated that contribute to the heaviness of the framework to realize it for a mobile device. Moreover, goal capturing and processing are not addressed to a reasonable level of detail.

Bose and Ennai [15] showed a service oriented framework for realizing SOA for an enterprise. A service oriented framework that allows mobile applications to interface with enterprise backends has been presented. An enterprise backend controls the dynamic provision of services on mobile devices. The approach used in this work follows an asynchronous message pushing and polling scenario in the mailbox user-machine interface layer.

This layer doesn't explicitly indicate how a request of the user is handled to continue in the service discovery process. User request needs to be formatted for creating an effective level of understanding between the user and the machine. The paper continues explaining the service discovery process it uses. It avoids the use of existing protocols for mobile platforms like UDDI by arguing the heavy nature of UDDI but how to build the service repository and maintain "the updated list of services" is not explained. As crucial as the process of service discovery in SOA contexts, an explicit service discovery scheme is not shown.

4. Semantic Service Design Framework

Figure 2 shows the various *elements*, *connectors*, and *containers* of the semantic service design framework as an architectural blueprint. The architecture follows a layered pattern, to realize the separation of concern requirement, with bidirectional communication between the layers. The architecture is composed of 6 main components: *Semantic Layer, UI Layer, Business Service Composer, SOA Layer, Middleware,* and *Phone API*.

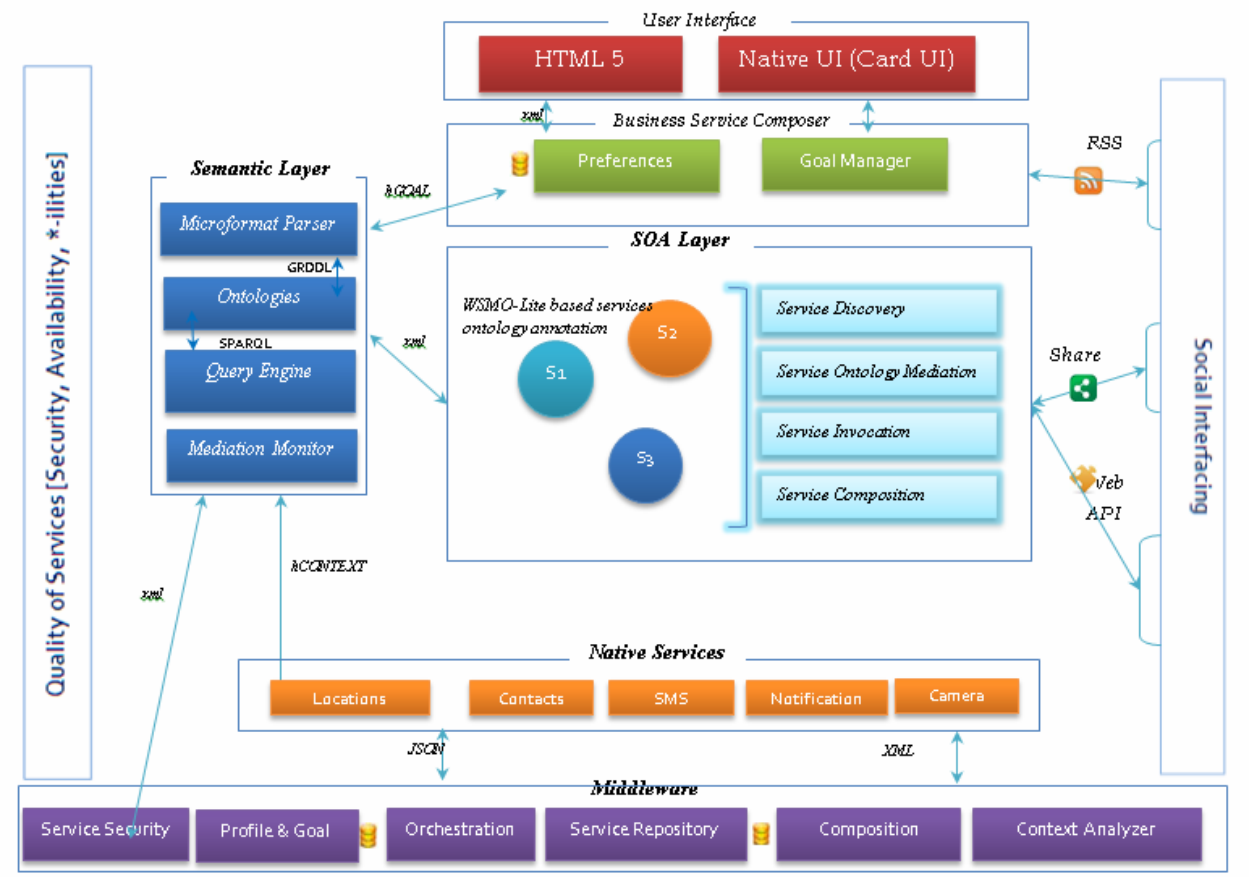


Figure 2: Conceptual architecture of Semantic service design framework

SOA Layer: is responsible to provide the semantic base, publish, find, and bind architectural pattern of SOA. The semantic annotation of service description is based on WSMO-Lite.

Semantic Layer: is responsible to perform semantic computation such as parsing the different Microformat; maintain the semantic knowledge representing background information about Goal, preferences, process; semantic query engine. It also employs a mediation scheme through the mediation manger that utilizes the underlying WSMO-Lite for interoperability with other semantic service ontologies.

Middleware: is responsible to coordinate computationally expensive service routines like service discovery & composition of services to create complex services. In this process there is a need to check the security, analyze profile and Goal of user, orchestrate services, access distributed service repository, perform service composition, and analyze context.

User Interface Layer: is a lightweight presentation layer defined using HTML 5 or native card UI for capturing user goal & preferences.

Business service composer: is a logical layer that monitors the rendering of UI for capturing the goal. It has a feature of persisting goal & preference in the local phone using HTML 5 stores.

Phone API: Various useful pieces of context information ranging from GPS location to sophisticated sensors can help understand the user's current perspective. This layer uses the phone API to build the context.

4.1 User Goal & Preference Capturing Using Microformats

Capturing of user's goal & preference in a consistent and repeatable manner is the prerequisite for successful service discovery. In order to take the lightweight advantages of pure HTML based user goal capturing, Microformats have been chosen to capture user's side goal. Microformat reuses existing HTML/XHTML tags to convey metadata and other attributes in web pages and other contexts that

support (X)HTML [16]. Custom Microformats, hGOAL, hPREFERNCES, and hCONTEXT are used for capturing the user's goal, preferences, and contexts, respectively. These Microformats will be transformed into RDF for ontologically capturing user's needs. The specification of hGOAL, hPREFERNCES and hCONTEXT are given below.

a hGOAL Microformat

- Goal represents what the user wants to do using the application such as "Order an Item", "Recommend purchased item", etc. Goal of the user during a business interaction shall be captured using hGOAL Microformat. hGOAL have mandatory attribute, action (with class tag gaction) and optional attributes that link the specific goal to a description of user's preference represented using gpreferences.
- textual description (with class tag *gdescription*) - a narration about what the Microformat represents.

Action: represents what the users explicitly do that needs a user aware confirmation, identified by gaction predefined attribute. Actions are usually verbs like Order, which will be mapped to an operation of a service entity.

Example: Figure 3 shows a visual goal of a user for ordering Samsung Galaxy Note I and the Microformat that shows details of the basic attributes and the attributes of these Microformats is presented in Figure 4.

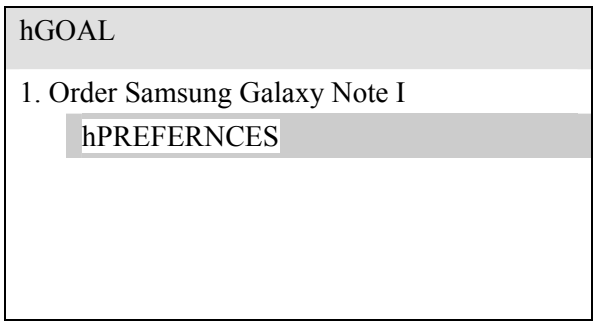


Figure 3: Example hGOAL, hPREFERNCES, Microformat

```
<div id="ggoal-buy-galaxy-note-userx" class="gaction">Order Samsung Galaxy Note I

    <div class="gprefences" >
       <span> Preferences </span>
       <a href="http://www.fast-buy-sell.com/preferences/ggoal-buy-galaxy-note-
          userx"></a>
       <ol id="gpreferences-ggoal-buy-galaxy-note-userx" class="gprefences">
          <li id="gprefence-payment-scheme-buy-galaxy-note-userx" class="gprefences"
             >Payment-scheme is credit card
          </li>
          <li id="gprefence-acc_no-buy-galaxy-note-userx" class="gprefences"> Account
             No is 99-99-999-9999
            <li id="gprefence-bank-buy-galaxy-note-userx" class="gprefences">Bank X
            </li>
          </li>
          </li>
          <li id="gprefence-price-range-galaxy-note-userx" class="gprefences">
             Price Between $399 and $599
          </li>
          <li id="gprefence-delivery-date-galaxy-note-userx" class="gprefences">
             Delivery date before Christmas
          </li>
       </ol>
    </div>
</div>
```

Figure 4: hGOAL & hPREFERNCES attributes

b. hPREFERNCES Microformat

Preferences are the non-action attributes, also called non-functional aspect, of user interaction that qualify the goal of the user with additional information. For instance, fixing the maximum price of an item or choosing particular kind of payment scheme to fulfill the user's goal are preferences.

Like goal, user preferences are captured in Microformat hPREFERNCES. Since the scenario of tracking preferences of the user is usually in business cases, optional attributes like prices range, expected date of delivery, payment scheme, etc. can be part of the preferences of a user.

For the preferences captured, rule based connectors are used, such as is, between, from, to, before, etc. These connectors will be determined and known by the server to generate the preference that will make it usable for parsing.

c. hCONTEXT Microformat

Contexts represent temporal situations of the user's environment like, geo location, time of the day, etc. Contexts are also represented in Microformat, hCONTEXT, but the representation of context doesn't require user or developers intervention. The middleware component tracks user contexts automatically. This representation contains optional list of attributes that can describe the user's context.

```
<div id="gcontext-userx" class="gcontext">CONTEXT
       <ul class="gcontext">
<li id="gcontext-attrib1" class="context-location">
<span>Location </span>
<ol class="gcontext">
<li class="context-latitude">Latitude=8.999764599999999</li>
<li class="context-longitude">Longitude=38.7795026</li>
</ol>
</li>
<li id="gcontext-attrib2" class="context-locale">
<span>Locale </span>
                                          <ol>
<li class="context-lang">Language=Amh</li>
<li class="context-country">Country=Eth</li>
                                          </ol>
</li>
<li id="gcontext-attrib3" class="context-time-of-day">
<span>Date and Time </span>
                     <ol>
<li class="context-date">November32012</li>
<li class="context-time">03:00 PM</li>
                     </ol>
</li>
</ul>
</div>
```

Figure 5: hCONTEXT Microformat attributes

d. Microformat parser

Merging of the three Microformats, goal, preference and context and transform the known class tags to an RDF creates the ontological representation of what the user wants to accomplish in the business application.

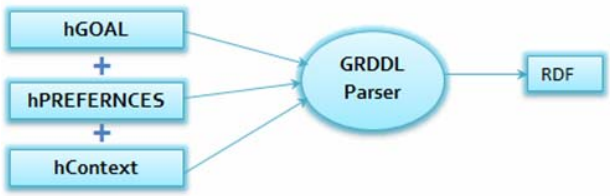


Figure 6: GRDDL Transformation of hGOAL, hPREFERNCES and hCONTEXT

e. Mapping RDF to WSMO-Lite

Mapping of the ontologies of WSMO-Lite elements to the framework follows the scheme presented in Table 1. The SSM Framework enables on the fly dynamic conversion of user goal, preferences, and context which show the user's intention and the matching to a WSMO-Lite equivalent ontology. Then matching algorithm can easily search for a possible candidate service from the repository.

Table 1: Mapping of SSM Framework to WSMO-Lite

* *SSM Framework*	*WSMO-Lite*
Goal	
⊙ Action	Functional [Capability Description], Informational
⊙ Preferences	Non Functional, Informational [Input and Output]
Context	
⊙ Location	Non Functional, Informational[Input Only]
⊙ Locale	Non Functional, Informational[Input Only]
⊙ Time of day	Non Functional, Informational[Input Only]

** SSM Framework – Semantic Service design framework for Mobile business applications*

Then the next major step is to run the service discovery algorithm from a middleware. The middleware takes the merged user business goal to use it as input to execute the matching algorithm.

4.2 Discovery Algorithm

A local or universal repository is considered for building the ontological service descriptions from the server side. How the service repository is built is out of the scope of this work, but the structure of the ontological descriptions of the services is assumed to be compatible and interoperable with the reference WSMO-Lite based service descriptions.

A Service Discovery algorithm that matches the ontologies of the representations of user goal, preference, and context with that of the available service descriptions in the repository is needed to be implemented as part of the middleware. Whenever the scheme of goal representation takes a different approach, the discovery algorithm has to be somehow changed to reflect the changes. Therefore, a multilevel service ontology matching algorithm that run ontological mapping at three levels, is proposed. This section shows the algorithm and the next section attempts to show the analysis of the algorithm's complexity.

Considering the WSMO-Lite based service descriptions available in the repository, the discovery algorithm is executed at three ontological levels. The algorithm is a variant of the multilevel service discovery algorithm, hence matching of user intentions with service ontologies happens at three levels: functional level, informational level, and QoS level. At each level, a degree of match (DoM) is computed and returned, and later it will be aggregated to a total DoM value. Based on the aggregated DoM value the services will be ranked.

Given a set of services, S, each described with WSMO-Lite annotation, in the repository, a set of ontological description of services' functionality, F, in the repository; a set of ontological description of services' information (i.e., Input Output data model), I, in the repository, and a set of ontological description of quality of services attributes, Q, in the repository.

When a service requester presents its request, a semantic representation of the request is categorized as for functional ontology, f; informational ontology, I, and aggregated QoS metrics, Q.

Algorithm 1: Main algorithm

```
// Main Algorithm
Matched-Services = ǿ
DoMfunc = 0, DoMInfo = 0, DoMQoS = 0, DoMAgrregated =0
For each-f in F
 DoMfunc += matchFunctional(each-f, f, F, ref Matched-
Services)
If(DoMfunc == 0) //No functional ➔ match not found!!
     return;
```

```
For each-I in I
     DoMInfo += matchInformational(each-I, i, I, ref
Matched-Services))
 For each-QoS in Q
     DoMQoS = matchQoS(each-QoS, q, Q, ref Matched-Services
DoMAgrregated = DoMfunc + DoMInfo + DoMQoS;
Return DoMAgrregated
```

Algorithm 2: Functional matching algorithm

```
// Functional Matching returns DoM for functional
matchFunctional(each-f, f, F, ref Matched-Services)
     if(DoMFuc = isMatch(each-f, f)){
         Matched-Services.add(each-f)
           return DoMFuc
           }
     return 0
```

Algorithm 3: Informational ontology matching algorithm

```
// Informational Matching returns DoM for Informational onto
matchInformational (each-i, i, I, ref Matched-Services)
     if(DoMInfo = isMatch(each-i, i)){
          Matched-Services.add(each-i)
          return DoMInfo;
     }
     return 0
```

Algorithm 4: QoS Matching algorithm

```
// QoS Matching returns DoM for QoS attributes
matchQoS (each-q, q, Q, ref Matched-Services)
     if(DoMQoS = isMatch(each-q, q)){
          Matched-Services.add(each-i)
          return DoMQoS;
     }
     return 0
```

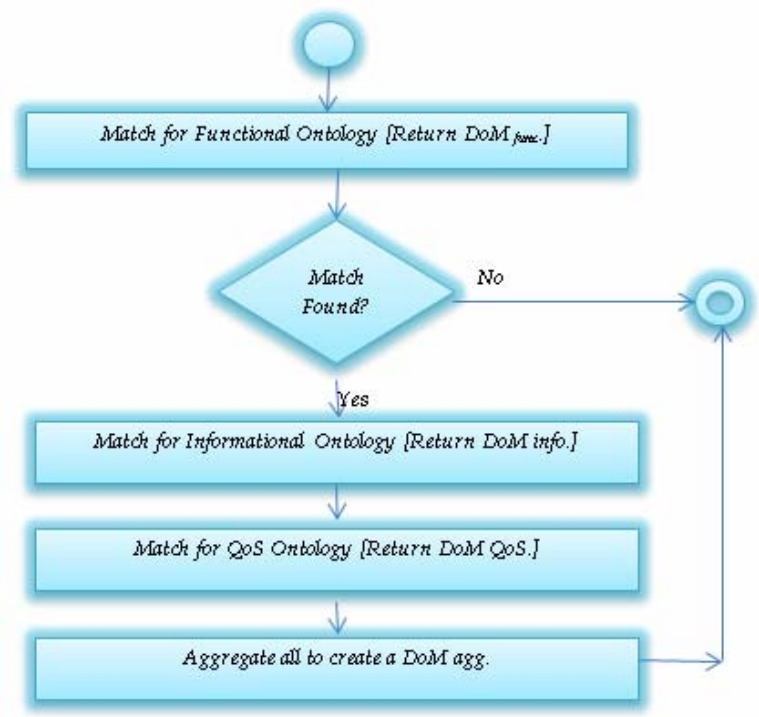


Figure 7: Flow chart of Multilevel Service Discovery Algorithm

The output of the multilevel service description algorithm, i.e., DoM agg is used to rank the matched services and select for execution.

The function *isMatch* returns a match between two ontology concepts in a typical "is a type of" or "has property" type of relationship.

5. Evaluation

This section presents the evaluation of the implemented discovery algorithm. For completeness, the framework needs to go through an architectural evaluation using one of the common architectural evaluation means like ATAM (Architectural Thread off Analysis Method) [17]. The full architectural evaluation is not covered and is future work.

The algorithm is evaluated by theoretically analyzing its running cost by using a time complexity algorithm analysis. We consider the following assumptions for showing the analysis of the discovery algorithm;

- The service repository exposes an RDF store that contains annotation for service descriptions in different categories in a manner compatible with the WSMO-Lite service annotations.
- The RDF format of User goal, context, and preferences show an ontology from the user side shall be reasonability small in number or remain to be constant since the user's goal ontological representation will not exceed a description of the main services the user interacts as an entry level

To evaluate the complexity of the algorithm, consider a functional ontology description of F_{user}, and $F_{repository}$ to represent the functional ontology descriptions of services in the repository, I_{user} as representing the informational domain ontology from the user side, while $I_{repository}$ representing informational domain ontology in the repository and finally let Q_{user} represent the user's QoS preferences and $Q_{repository}$ represents the QoS ontologies of the services in the repository. For each of these categories, we try to check the complexity in three schemes,

a. Keeping repository side constant, we vary the value of user side
b. Keeping user side constant, we vary the values of repository side
c. Vary both the user side and the repository

Since the type of matching is similar for each of these categories, we can generalize from the result of the functional.

a. Keeping repository side constant, we vary the value of user side

Since the matching is a loop that attempts to look for a match between the ontologies, the complexity is always a product of what the user presents (i.e., F_{user}) with that of what is available in the repository ($F_{repository}$) which is generalized with m*$F_{repository,}$ where m is a varying value of F_{user}. Since $F_{repository}$ is constant, this expression is a linear value.

F_{user}	0	5	10	15
$F_{repository}$	20	20	20	20
X	0	100	200	300

b. Keeping user side constant, we vary the values of repository side

Almost the same kind of trend is expected by keeping user side constant and varying the repository side that results in a linear complexity.

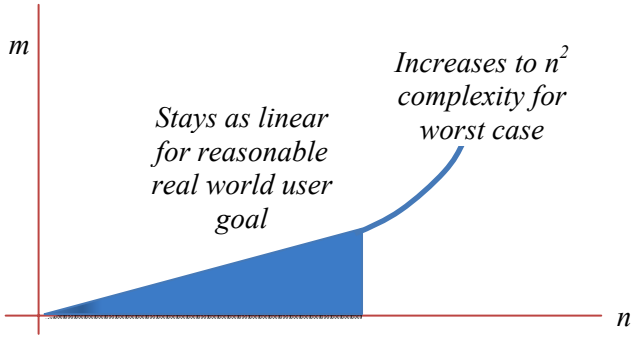
F_{user}	5	5	5	5	5
$F_{repository}$	0	20	40	60	80
X	0	100	200	300	400

c. Vary both the user side and the repository

In a typical case where user goal, preference ontology varies and the repository also varies, the complexity computed by the product of the two, since for each ontology of user side the repository also needs to be computed in the loop.

F_{user}	0	5	10	15
$F_{repository}$	0	20	40	60
X	0	100	400	900

We can naturally generalize the previous pattern to the following graph.



It results in a complexity of linear, i.e., *m*n* in the best and average cases. For worst case where various RDF graphs emerge as a representation of the user goal, a complexity of n^2 may result.

6. Conclusion and Future Work

Adding layers of semantics to the basic service routines of SOA enables some degree of automation. The concern for design time searching and binding of service can be avoided through a dynamic service discovery scheme. To this end, in this paper, a blended approach that relies on a custom Microformats used to capture user goals, preferences and contexts in a mobility realm has been proposed as part of the conceptual architecture. From the server side, a variant of a multilevel service discovery algorithm that is based on a reference semantic web service ontological description i.e., WSMO-Lite, has been proposed as part of the middleware, which enables the middleware to automatically match service's ontological descriptions to select and execute the most suited service.

Automation of service composition aspect of SOA was not addressed. Moreover, the scheme of goal capturing used in this paper is one dimensional in that it attempts to collect end user input but the business analyst or developer user may have a need of encoding a business driven goal that contains schematic representation of the business domain much like the BPM's workflow. In addition, for the business cases handled in the framework, the goal description may be limiting for real world diversified and complex services; so extending the goal with additional attributes and processing it through a GRDDL transformation is suggested in the building of the ontology. But the process follows the same path.

Building of a service repository can use techniques like indexing and daemon based crawling or other means. How this happens is not addressed in this paper. But the structure of the service repository adheres with the inspirational interoperating works of WSMO-Lite.

References

[1] Thomas Erl, “SOA Governance, Governing Shared Services on Premise and in the Cloud”, 1st ed., Boston, USA, Prentice Hall, 2011.

[2] Thomas Erl, “SOA Design Patterns”, 1st ed. Boston, USA, Pearson Education, Inc, 2009.

[3] Thomas Erl, “SOA, Principles of Service Design”, Boston, USA, Prentice Hall, 2008.

[4] Francisco Curbera, Greg Meredith, Sanjiva Weerawarana, and Erik Christensen. A w3c website, http://www.w3.org/TR/wsdl

[5] Yves Lafon nd Nilo Mitra, W3c website, http://www.w3.org/TR/2007/REC-soap12-part0-20070427/

[6] Dieter Fensel, Mick Kerrigan, and Michal Zaremba, “Implementing Semantic Web Services”, The SESA Framework, Innsbruck, Austria, Springer, 2008.

[7] Jacek Kopeck, Jana Viskova, and Dieter Fensel Tomas Vitvar, "WSMO-Lite Annotations forWeb Services", Digital Enterprise Research Institute (DERI), Ireland, Galway, 2008.

[8] Siddhartha Bose and Anuraj Ennai, "MobileSOA: A Service Oriented Web 2.0 Framework for Context-Aware, Lightweight and Flexible Mobile Applications", Motorola India Research Labs, Bangalore, India, 2008.

[9] Matthew John Moran, "Semantic Service Oriented Architecture Component Model, Reference Architecture and Evaluated Prototype", Digital Enterprise Research Institute (DERI), Ireland, Galway, Doctoral Dissertaion, 2011.

[10] D. B. Pedia website, http://wiki.dbpedia.org/About

[11] D., Lausen, H., Polleres, A., De Bruijn, and J., Fensel, "Enabling Semantic Web Services: The Web Service Modeling Ontology", Berlin, 2007.

[12] Mark Burstein, Jerry Hobbs, Ora Lassila, Drew McDermott, Sheila McIlraith, Srini Narayanan, and David Martin, w3 web site. http://www.w3.org/Submission/OWL-S/

[13] WSMX Website, http://www.wsmx.org/

[14] Dieter Fensel, James A. Hendler, and John Domingue, "Handbook of Semantic Web Technologies", Springer-Verlag, Berlin, 2011.

[15] Siddhartha Bose and Anuraj Ennai, "MobileSOA: A Service Oriented Web 2.0 Framework for Context-Aware, Lightweight and Flexible Mobile Applications," Motorola India Research Labs, Bangalore, India, 2008.

[16] Microformat Initiative, Microformat Website, http://microformats.org/wiki/what-are-microformats

[17] Mario R. Barbacci, Paul Clements, Rick Kazman, Mark Klein, Liam O’Brien, James E. Tomayko, and Robert L. Nord, "Integrating the Architecture Tradeoff Analysis Method (ATAM) with the Cost Benefit Analysis Method (CBAM)", Carnegie Mellon University, Technical Note 2003.