Managing Web Resource Compositions

Mahdi Bennara, Youssef Amghar
Université de Lyon, CNRS
INSA-Lyon, LIRIS UMR5205
F-69621, France
{*mahdi.bennara, youssef.amghar*}*@liris.cnrs.fr*

Michael Mrissa
Université de Lyon, CNRS
Université Lyon 1, LIRIS UMR5205
F-69622, France
michael.mrissa@liris.cnrs.fr

***Abstract*—Nowadays, the use of RESTful Web services promotes stateless service interaction and decentralized hypermedia-driven discovery and composition. However, there is a need for models and tools to drive user interaction as well as description, discovery and composition of RESTful services. In this paper, we provide a solution to help users manage, share and discover workflows of RESTful Web services. We annotate RESTful Web services with semantic information, and introduce the notion of *composition directory* as a Web resource that assists a user in sharing, managing and discovering workflows. Users' composition directories form a decentralized repository of service workflows connected by hypermedia links. We illustrate the benefits of our approach with a typical scenario and show through a set of experiments that the breadth-first search algorithm combined with the exploitation of semantic annotations efficiently answers users' goals by crawling through composition directories.**

***Keywords*-RESTful Web services, linked services, semantic Web, composition**

I. Introduction

The Web has moved from a Web of documents to a distributed application platform where applications are exposed as Web resources, as witnesses the growing number of available APIs[1]. Leading research topics are related to discovery, composition and invocation of Web resources via their API. In addition, the emergence of semantic Web technologies gives the opportunity to improve the use of APIs with semantic annotation over Web resources. Semantic annotation helps to drive the interaction with APIs by providing explicit description of domain-specific information about resources.

Another key concept that drives today's Web is distributed affordance. Affordance is the ability for a user to use a Web resource. The idea is to dynamically create affordance based on the information already present in a resource representation, with knowledge from distributed sources [12]. Distributed affordance combines the information on resources and the knowledge on service providers, as well as user profiles in order to generate possibilities for manipulating Web resources. It should allow client-side software to dynamically drive the interaction with Web resources, therefore service providers do not have to anticipate user interaction and avoid deploying static business processes that constrain users.

In order to enable distributed affordance, Web resources must be semantically described, and user agents needs to be able to exploit such descriptions. In this paper, we build on previous work to semantically annotate Web resources [1] and facilitate resource discovery and browsing. We introduce the concept of composition directory to help users manage and share compositions, and show that the breadth-first search algorithm can be used in this context to crawl and discover resources according to a composition workflow the user provides.

Our paper is organized as follows: Section II introduces the challenges we want to overcome while trying to solve the composition problem. Section III presents related work and highlights the advantages our solution offers. Section IV details the main aspects of our contribution and shows its innovation. Section V shows a scenario example and details how our prototype operates in the context of the scenario to demonstrate the applicability of our solution. Section VI discusses our results and gives some guidelines for future work.

II. Context and Challenges

In the context of our work, Web services are seen as Web resources that comply with the REST architectural style. The REST architectural style is based on the notion of resource as a conceptual entity that represents abstract or concrete things such as books, orders, payments, etc. Resources are identified by URIs, their state is passed to the client through representations using the adequate media type according to the principle of context negotiation. In this paper we consider a RESTful Web service as a set of resources that provide a coherent access to the state and functionality of the software it represents [9].

Another principle that drives the REST architectural style is the HATEOAS[2] principle. Using HATEOAS requires hyperlinks to be established between Web resources that form an open and very large graph. HATEOAS means that the discovery process is realized progressively, user agents

[1]http://www.programmableweb.com

[2]Hypermedia As The Engine Of Application State

should be able to discover other Web resources accessible from any given resource in the graph.

In this context, we identified several challenges to address, which can be summarized as follows :

- Web resource description and interlinking: resources need to be appropriate described with semantic annotations and also linked to each other with hyperlinks to enable client-side discovery (how to interact with the resource) and crawling (how to discover other resource from a given one according to the HATEOAS principle).
- Web Resource discovery: as a follow-up to the first challenge, user agents should be able to implement an efficient algorithm to crawl between resources and exploit their annotations to realize users' objectives.

In order to answer these challenges, we build on previous work to annotate resources. We introduce the notion of composition directory to manage and share composition workflows and we show that the breadth-first search algorithm can be used to crawl through and and efficiently discover Web resources.

III. Related Work

In this section, we overview existing work about Web resource management and discovery algorithms.

A. Resource Discovery

RESTdesc [13] is based on the Notation3 RDF syntax. It involves all the operational semantics of Notation3 which allows for a versatile discovery methods. We can take advantage of all the advances in the reasoning domain in the Notation3 syntax in order to determine whether a resource satisfies a set of conditions defined for the discovery. More advanced reasoning is performed in order to achieve service matching. Authors estimate that this is an important prerequisite for services in order for them to contribute to the future Web of clients, because new functionality can only be obtained by on-demand compositions tailored to a specific problem [13].

RESTdoc [3] is a format that combines multiple microformats in order to semantically describe RESTful resources. RESTdoc offers a discovery mechanism that distinguishes two different aspects of REST services discovery problem: (1)the discovery as a client, or discovery as you browse, concerns the client-side browsers. This discovery uses on HTML Link element on a Web site in order to point to other resource descriptions. and (2) the discovery as a service, also called automated discovery, is the ability for a service to access and link to other related resources in the same application domain. The solution provided by RESTdoc describes a fully peer to peer discovery mechanism. The main idea is to construct a graph by running through links and identifying resources. This graph can be subject to later extending in order to explore new related resources.

LinkedWS [6] is a Web service discovery model based on social networks. The idea behind LinkedWS is to construct a social network for every service on the Web in order to allow social-based discovery process. The social network is built the moment the service participates a first time in a composition. Generally speaking, a social network of a Web service consists of nodes and edges. The nodes represent an object or an entity (book, person, organization, etc.) and the edges represent the relationships between nodes (distance between two cities, relationship between two persons, etc.) Each edge has its own weight, that is used by the search or ranking algorithms that navigate through the network in order to find Web services that suit specific purposes. Every Web service is the entry point of his own social network. LinkedWS allow the discovery of additional Web services in a specific composition. The discovery itself triggers the re-evaluation of the weight of the edge that led to this discovery.

Our proposition aims to enhance the discovery process by embedding the related resources directly in the resource description. It allows also the discovery of resources using existing composition work-flows.

B. Resource Composition

Many researchers are interested in the problem of describing the semantics of the sequences in the executions flows, and have proposed many solutions.

One of the most important works in this domain is the BPMN [3] [14] specification. BPMN specifies a set of flow control sequences that allows us to describe the progressing of a process. The main interest of BPMN for us is that it can be used in order to construct and store dynamic service composition processes which can be reused afterward by another user that wants to do similar service composition. The use of BPMN relies on a Process-oriented approach rather than being Resource-oriented. This may conflict with the principles of the REST architectural style, nevertheless some concepts can be used naturally on resource oriented architectures.

Linked USDL [4] [10], is another work in this perspective. Unlike, BPMN, Linked USDL vocabulary has been designed especially for the service-oriented domain, making it easier to adapt for our solution. Some of the important concepts introduced by this vocabulary include: Service, ServiceOffering, InteractionPoint as well as services roles including: Producer, Provider, Intermediary, etc. which constitute the main semantic concepts of workflow control in service-oriented architectures. Linked USDL is being used in many projects, and it proved its efficient for the service community.

BPEL for REST [9] is a work that proposes to reuse the BPEL language principles and apply it on the REST architectural style. BPEL for REST either uses the WSDL

[3]Business Process Modelling Notation

[4]Unified Service Description Language

2.0 description language without changing the current BPEL or extends BPEL in order to support HTTP operations on the resource API. The main drawback of BPEL comes with its centralized approach that relies on a static composition engine, which does not fit with the HATEOAS principle.

Our approach aims to use work-flows in order to enable reuse of popular compositions. This is also a flexible way of composing resources as the work-flows can be duplicated and edited.

C. *Graph discovery algorithms*

Exploring very large graphs such as the Web requires efficient algorithms in order to have acceptable response times. As we are discovering resources on the Web, the efficiency of the exploration algorithm is one of the most important elements of our research work. Practically the classic algorithms are not used as such because they may result in important response times due to the Web size. Instead, variants of these algorithms are used with specific parameters (often limiters) in order to yield reasonable response times and acceptable results. The most known examples of algorithms are Breadth First Search and Depth First Search algorithms. Other algorithms include variants of these two with limiting parameters, for example limiting the depth of the search (number of consecutive edges counting from the root) also known as depth-limited search or limiting the total number of nodes accessed during the whole process [11] [2].

According to [8] the Breadth-First Search graph traversal algorithm yields high-quality pages early on in a crawl. In other words, the most relevant pages/resources to the search are discovered early on in the process. In our work, this means Breadth-First Search finds the most relevant resources to answer a user's request by finding multiple (or single) resources that can perform the tasks needed in order to answer the request. In addition to that, Breadth-First Search is a very natural search strategy in the context of Web. Also, compared to other efficient search algorithms, it has a relatively low computational cost for a large scale graph such as the Web.

IV. Contribution

Based on the related work presented above, we have built our solution that promotes the concept of composition directory and uses the breadth-first search algorithm to manage, share and crawl Web resources. Our solution must respect the following requirements in order to facilitate resource discovery and composition:

- Scalability: the increasing number of today's Web APIs makes the scalability of solutions important.
- Responsiveness: the increasing number of users generates an important load of requests on servers. We want server responses to be as fast as possible in order to handle all the requests in a reasonable time.
- Diversity: We want our resource descriptions to propose rich and diverse links to other resources in order to give them a chance for being used. In other words, the users which make a request should have different propositions rather than only popular services in a given field and thus giving the chance to less popular services to emerge if the users are interested in the services they offer.
- Dynamism: the results of the resource discovery process should not be static, in other words it should be different from one request to another, because on one hand the availability of resources involved in the request as well as the context of the request may have changed in the meantime, and on the other hand, the user context may also have changed, which implies that users might not get same results because they browsed new resources which may impact the response.
- Serendipity: the serendipity concept allows APIs to be used in a non-specific process. In other words we do not want the clients to use APIs in a deterministic way where every next API to use is already known in advance

Today, the main advances in Web resource composition are centered around the description of the resources. The focus of these advances is how to describe a resource in order to give as much information as possible to identify the nature of the resource, its activity and what type of data it exchanges. Too few efforts focus on how it links to other resources and how to follow these links, as well as how to manage and share composition workflows. The latter aspects are presented in the following in order to enable value-added resource discovery and composition.

A. *Describing and Discovering Resources*

In order to semantically describe resources, we rely on the notion of resource descriptor discussed in [1]. The resource descriptor has been slightly modified and the data model we use to implement the descriptor concept is the Hydra core vocabulary [4]. The main reason for this choice is that Hydra defines explicit semantics of its `Operation` and `Link` elements, the major two elements present in the descriptor.

The resource representation contains the business-level information about the given resource, while the descriptor contains semantically annotated information on how to use this resource (the available HTTP Operations on the current resource, not on other resources) plus information about the related resources (links to other related resources). Hence, resource descriptors separate resource representations from their descriptions, to promote separation of concerns between resource interaction and management (discovery and composition). We deem appropriate to make sure that every resource must have a descriptor to enable machine-to-machine interaction. As every resource must have a descriptor, and descriptors are also considered as resources, descriptors also must have their own descriptors. To address this issue, we define the Universal Descriptor[5], which describes all the resources including itself.

With the help of resource descriptors, a generic-client is able to interact with resources and to crawl from one to another in order to compose resources. From a resource URI, a client can get its descriptor by executing a GET/HEAD operation on the given URI, and checking the LINK header element in the HTTP response. Another GET operation on the retrieved link returns a HTTP response with the descriptor containing all the necessary meta-data that describes not only the interaction model with the current resource, but also annotated links to internal or external resources that can be composed together with the given resource. In order to document the semantics of sets of links as well as operations that are present in a given resource descriptor, we rely on ontology concepts. On the operation side, we want to know the exact concept that is accomplished by the operation being described. The concept is part of a larger global composition process that implies several other operations on different resources. Describing operations requires ontologies of operational semantics to describe how to realize a complex task with a combination of simpler, specific tasks.

B. Discovery Process

The discovery process we introduce in this work relies on semantic annotations of operations in the descriptors. When a user enters a request, it is processed by a reasoner in order to know what actions should be realized in order to prepare a response, as presented in previous work [7]. These actions are represented by an ordered list of ontology concepts. The discovery process will take as input this list of concepts as well as an entry point (the URL of a Web resource). Starting from this entry point, the process tries to find resources on the Web that provide the required operations to respond to the request, on the basis of the concept list taken as input. We consider the Web as a big oriented cyclic graph, where nodes are resources and links are Web hyperlinks. We adopt the Breadth-First Search (BFS) algorithm in order to traverse the Web in search of resources that provide the operations corresponding to the concepts of the request.

C. Managing and Sharing Compositions

In order to enable users to record, reuse, manage and share their composition workflows, we propose a specific resource called *composition directory*. The Composition Directory resource contains information about its owner, and a sub-resource called *repository* to store as sub-resources the composition workflows the user creates. The Composition Directory of a user links to other connected users Composition Directories. Note that this is completely compatible with the descriptor concept because the links to other Composition Directories, the Repository and the created compositions scenarios represent the external part of our descriptor.

[5]http://soc.univ-lyon1.fr/universal.md

We define the following API in order to qualify possible interactions with Composition Directories

1) GET on the base URI of a Composition Directory should send back the information about this Composition Directory and its owner.
2) GET on the Repository of the Composition Directory should send back the set of links to every composition on the Repository.
3) GET on a specific composition URI should send back the representation of the composition. This may require an authentication and may send a 401 code (Unauthorized) in case the authentication fails.
4) POST from the user on the Repository of his Composition Directory should create a new composition. Composition attributes and its accessibility should be indicated by the use beforehand in the representation.
5) POST from the user on his own composition directory in order to add a new Composition Directory URI of another user that exposes interesting compositions for him.

This offers many advantages, first it is a scalable and decentralized solution as every user stores a part of compositions on the web, it also respects the serendipity concept as a given client may find part of the solution to the user's problem in another user's composition set. Our model includes access control features relying on HTTP authentication. We define public compositions that everyone on the Web can access from its URI, and private compositions that are not disclosed and require authentication.

V. Tests and Evaluation

In this section, we illustrate our contribution with different scenarios, we detail our implementation setup, and discuss the results obtained.

A. Illustrative Scenarios and Experiment Setup

In order to illustrate our approach, we consider three scenarios. The first scenario involves three different Web resources and illustrates how resources and descriptors are disposed, and how resources link to each other using descriptors. It includes :

- A book selling service: users can select books, read abstract and place an order.
- A shipping service: its task is to deliver goods that users buy online.
- An online payment service: the task of this service is to debit money from users' bank accounts to the benefit of online stores for the goods they buy.

The second scenario involves two users to illustrate the fact that users do not use Web resources in the same way.

Each user orchestrates services in a different manner and shares it with the other user or with everyone on the Web using the Composition Directory mechanisms. The Web resources involved are similar to the first scenario with an additional computer accessories store, where users can select accessories, read description and place an order. The first user's objective is to buy a book online, pay it online and receive it by mail. The second user wants to buy some computer accessories, pay them online and receive them by mail. The third scenario involves 25 resources with their descriptors. These resources can perform multiple actions, which are annotated in the descriptors. The objective of this scenario is to illustrate the discovery process with the descriptor mechanism. The prototype illustrating our work is available online[6].

In our work, we rely on Java^{TM} language and the Jersey framework[7] in order to implement our services and JavaScript as a client-side scripting language in our Web pages. We use the Google Gson module[8] in order to manipulate Json objects in Java. We use Apache Tomcat[9] as an application server in order to accommodate our different Web resources. Our descriptors rely on the Hydra core vocabulary [5] to describe resources. The machine used for the experiments has an Intel^{TM} Core i5-3340M CPU and 8GB RAM. The Web browser used is Mozilla^{TM} Firefox. The tests were performed on the local university network.

B. Evaluation and Discussion

In this section, we discuss the choices of our implementation and the impact of these choices on the challenges and the properties we want to achieve. As we are using the REST architectural style to build our solution, the respect of Web constraints is ensured by design.

1) Resource Description: We evaluate and discuss our contribution for the resource description challenge on the basis of the first scenario. The use of descriptors allows to benefit from separation of concerns between business-level information and description-level information (meta-data contained in the descriptor). The descriptor also allows access to relations with other Web resources, which enables interactions according to HATEOAS. The first scenario illustrates well this separation.

In our example, the book selling resource only contains business layer information such as the name of the service as well as a brief description. Its descriptor, on the other hand, contains links to internal and external related resources (the shipping and payment services for instance) as well as the functioning of the resource itself with the help of semantic annotations. The order resource in the book selling service gives another example about resource description: a GET operation returns the representation of the repository with all the placed orders and their statuses (with per user authentication), a POST operation takes an order as request body and adds it to the repository as an unpaid order, and PUT operation changes the order state and a DELETE operation (only available when the order has been fully completed) deletes an order from the repository.

2) Resource Composition: We evaluate and discuss our contribution for the composition challenge with the second scenario. Composition directories allow users to create, store and share compositions of Web resources. Compositions can be thereafter entirely of partially reused by the user himself or other authorized users. This solution allows for dynamic creation of new compositions rather than follow inflexible server-side compositions. It allows also a large scale sharing of popular compositions that users find useful and offers flexible ways to reuse and adapt compositions to user's needs. We do not rely on a central repository to store the compositions.

The second scenario illustrates these statements: the first user can create a composition for the process of buying a book involving the shipping and online payment services. He can share this composition with the second user to buy things online with goods delivery and payment services. The second user can reuse the same composition if he wants to buy a book, or only a part of this composition if he wants to buy computer accessories. The creation of a new composition as well as its reuse depend on the discovery process discussed below.

3) Resource Discovery: We evaluate and discuss the application of the Breadth-First Search algorithm for discovery of resources in our work.

One important thing to note while applying the algorithm is that nodes are not simple in the context of descriptors, but are double. A node of our graph is the resource and its descriptor while the edges are simply the links in the descriptor of a given resource. In order to avoid unnecessary use of bandwidth, we use HEAD requests on resources instead of GET, in order to retrieve the descriptor link, and then we use a GET request in order to retrieve the descriptor contents.

We illustrate the application of the algorithm on our resources with the third scenario. The algorithm takes a set of concepts and an entry point as input data, it crawls through the graph formed by resources and their descriptors in order to give back the results. The timings of our experiment are shown in Table I and discussed below.

The number of nodes indicates the total number of nodes in the graph that have been traversed in order to give a full response to the request. The response time indicates the time in milliseconds taken in order to respond to the request. The response time per node is simply the response time divided by the number of nodes. The response times per node are under 20 millisecond if the research involves a small number

[6]https://liris.cnrs.fr/~mbennara/doku.php?id=wetice2015
[7]https://jersey.java.net/
[8]https://code.google.com/p/google-gson/
[9]http://tomcat.apache.org/

Table I
DISCOVERY ALGORITHM RESPONSE TIMES

Number of nodes	1	5	10	15	20	25
Response time (ms)	18	89	184	241	301	334
Response time per node (ms)	18	18	18	16	15	13

of nodes, but as the number of nodes grows, it becomes lower due to the caching mechanism of the Web browser and the factoring of descriptors for resources that share the same descriptor contents. In other words, additional GET requests on the same descriptor that is shared by multiple resources in the graph are processed faster, as the results are stored in the local cache.

This experiment demonstrates that response times are below a linear progression as the global response times indicate. Hence, our approach scales quite well as the composition directories as well as the descriptions are decentralized and each resource stores a part of the global graph. Our approach also ensures dynamic results, as the content of descriptors may be subject to change especially the links part. The diversity is also ensured as the algorithm may find multiple resources that implement a specific operation needed in the process, this number can be limited before the algorithm starts and is given as input with the concepts. The serendipity has also its share in our approach, as the order in which the links of a descriptor are crawled may influence the final result, giving chance to more or less popular resources to be used within the composition process.

VI. CONCLUSION

In this paper, we have introduced the notion of *composition directory* as a REST resource that allows users to record, manage and share composition workflows. We have proposed a solution to link users' composition directories over the Web to each other to form a distributed directory of composition workflows that can be crawled with graph traversal algorithms. We demonstrate the adequacy of our solution with a set of experiments that rely on the well-known breadth-first search algorithm to discover Web resources according to a set of concepts that represent a user's goal. The obtained results show the scalability of our proposal. As future work, we envision extending our solution with quality models combined with user-side reasoning to enhance the discovery algorithm efficiency. We aim to enable automatic reuse of compositions by reasoning about their semantic annotations in order to respond to a user's request.

REFERENCES

[1] M. Bennara, M. Mrissa, and Y. Amghar. An approach for composing restful linked services on the web. In C.-W. Chung, A. Z. Broder, K. Shim, and T. Suel, editors, *WWW (Companion Volume)*, pages 977–982. ACM, 2014.

[2] K.-T. Förster and R. Wattenhofer. Directed graph exploration. In *Principles of Distributed Systems*, pages 151–165. Springer, 2012.

[3] D. John and M. S. Rajasree. RESTDoc: Describe, Discover and Compose RESTful Semantic Web Services using Annotated Documentations. *International Journal of Web & Semantic Technology (IJWesT)*, 4(1), 2013.

[4] M. Lanthaler. Hydra Core Vocabulary. http://www.markus-lanthaler.com/hydra/spec/latest/core/.

[5] M. Lanthaler and C. Guetl. Hydra: A Vocabulary for Hypermedia-Driven Web APIs. In C. Bizer, T. Heath, T. Berners-Lee, M. Hausenblas, and S. Auer, editors, *LDOW*, volume 996 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2013.

[6] Z. Maamar, L. K. Wives, Y. Badr, S. Elnaffar, K. Boukadi, and N. Faci. Linkedws: A novel web services discovery model based on the metaphor of "social networks". *Simulation Modelling Practice and Theory*, 19(1):121–132, 2011.

[7] M. Mrissa, L. Médini, and J. Jamont. Semantic discovery and invocation of functionalities for the web of things. In S. Reddy, editor, *2014 IEEE 23rd International WETICE Conference, WETICE 2014, Parma, Italy, 23-25 June, 2014*, pages 281–286. IEEE, 2014.

[8] M. Najork and J. L. Wiener. Breadth-first crawling yields high-quality pages. In *Proceedings of the 10th international conference on World Wide Web*, pages 114–118. ACM, 2001.

[9] C. Pautasso. Restful web service composition with bpel for rest. *Data Knowl. Eng.*, 68(9):851–866, 2009.

[10] C. Pedrinaci, J. Cardoso, and T. Leidig. Linked usdl: A vocabulary for web-scale service trading. In V. Presutti, C. d'Amato, F. Gandon, M. d'Aquin, S. Staab, and A. Tordai, editors, *ESWC*, volume 8465 of *Lecture Notes in Computer Science*, pages 68–82. Springer, 2014.

[11] S. Russell, P. Norvig, and A. Intelligence. A modern approach. *Artificial Intelligence. Prentice-Hall, Egnlewood Cliffs*, 25, 1995.

[12] R. Verborgh, M. Hausenblas, T. Steiner, E. Mannens, and R. V. de Walle. Distributed affordance: an open-world assumption for hypermedia. In L. Carr, A. H. F. Laender, B. F. Lóscio, I. King, M. Fontoura, D. Vrandecic, L. Aroyo, J. P. M. de Oliveira, F. Lima, and E. Wilde, editors, *WWW (Companion Volume)*, pages 1399–1406. International World Wide Web Conferences Steering Committee / ACM, 2013.

[13] R. Verborgh, T. Steiner, D. V. Deursen, J. D. Roo, R. V. de Walle, and J. G. Vallés. Description and Interaction of RESTful Services for Automatic Discovery and Execution. In *Proceedings of the FTRA 2011 International Workshop on Advanced Future Multimedia Services*, Dec. 2011.

[14] P. Y. H. Wong and J. Gibbons. A process semantics for bpmn. In S. Liu, T. S. E. Maibaum, and K. Araki, editors, *ICFEM*, volume 5256 of *Lecture Notes in Computer Science*, pages 355–374. Springer, 2008.