An Ontology-based Representation of the Google+ API

Konstantinos Togias
School of Science and Technology
Hellenic Open University
Patras, Greece
ktogias@eap.gr

Achilles Kameas
School of Science and Technology
Hellenic Open University
Patras, Greece
kameas@eap.gr

Abstract—**Social Networking Services (SNS) provide users with functionalities for developing their on line social networks, connecting with other users, sharing and consuming content. While most of popular SNS provide open Web 2.0 APIs, they remain disconnected from each other thus fragmenting user's data, social network and content. Semantic social web technologies such as public vocabularies and ontologies can be used for bridging the semantic gap between different SNS. Ontology-based representations of SNS APIs can help developers share knowledge about SNS APIs and can be used for linking APIs with public Social Semantic Web ontologies and vocabularies and for enabling automatic ontology-based service composition. In this paper, we study the API of Google+ SNS and create an ontology based representation of its structural and functional properties. The proposed ontology describes valuable structural and functional details of the API, in a machine processable format useful for understanding the API and appropriate for integrating into ontology based Mashups.**

Keywords—Semantics; Social Networking System; Web Mashup; Social Semantic Web.

I. Introduction

Social Networking Services (SNS) are web applications that allow users create and maintain an online network of close friends or business associates [1]. Typical examples of SNS are Facebook, Myspace, Twitter and the most recent Google+. While SNS have much common functionality they do not usually interoperate and therefore require the user to re-enter her profile and redefine her connections when registering for each service [1]. Also content shared in one SNS is not available to users of other SNS.

Web 2.0 is a widely-used term characterizing the modern web made popular by Tim O' Reilly. Web 2.0 is the network as platform, spanning all connected devices [2]. Web 2.0 applications consume data and services from other applications and enable the reuse and remixing of their own data and services through public Application Programming Interfaces (APIs). Experienced users and programmers use those APIs for creating new integrated web applications, popular known as mashups [3] that combine different data sources and APIs into an integrated end user experience.

Most SNS participate to the Web 2.0 ecosystem by providing their own open APIs. Those APIs provide a first step towards bringing down the walls between SNS. Nevertheless, every SNS use its own terms for defining concepts and representing resources, while it interconnects the resources it provides in its own custom way. Thus common concepts, resources and functionalities are described and provided in different ways in each SNS API.

The Social Semantic Web is the vision of a Web where all of the different collaborative systems and SNS, are connected together through the addition of semantics, allowing people to traverse across these different types of systems, reusing and porting their data between systems as required [1]. Social Semantic Web uses Semantic Web technologies in order to describe in an interoperable way users' profiles, social connections and content creation, sharing and tagging accross different SNS and Sites in the Web.

Ontologies have become the means of choice for knowledge representation in recent years as they provide common format and understanding on domain concepts, while being machine processable [4]. Hendler [5] supports that the ontology languages of the Semantic Web can lead directly to more powerful agent-based approaches. Furthermore, ontologies are used for representing and sharing knowledge about structural and behavioral properties of software [6], for building context-aware and pervasive applications [7], and for achieving context-aware web service discovery and automatic service composition in Service Oriented Software (SOA) [8][9].

Web 2.0 APIs, SOA technologies and Social Semantic Web approaches provide the basic means for bridging the gap between today's SNS and for unifying users' data, social networks and interactions scattered across various SNS. However, today's SNS APIs lack semantic representations, while existing Semantic Web Ontologies and Vocabularies do not provide links with the API resources and methods used for actually accessing and manipulating users, social networks and content within SNS. Thus, Social Semantic Web approaches, SOA service discovery and service composition techniques cannot be directly applied on them. Moreover, combining multiple SNS APIs for building Mashups require for developers to search, read and combine information from miscellaneous documentation pages scattered across the web. Using Ontologies for describing those APIs can help addressing those shortcomings by providing common, machine processable representations suitable for both sharing knowledge between developers and achieving automatic service discovery and service composition in SNS Mashups.

In this work, we study the API provided by Google+, one of the most popular and most recent SNS and we propose an ontology based representation of its structural and functional characteristics. Our ontology is compatible with the technologies of the Semantic Web and aims to be useful for sharing knowledge about the Google+ API between developers of Web 2.0 Mashups and as part of future inter-operable ontology based social networking software.

The paper is organized as follows. Section 2 briefly reviews related work in the areas of Social Semantic Web, Web 2.0 Mashups, ontology representation of software properties, and Service Oriented Architectures (SOA). Section 3 presents the proposed ontology-based representation of Google+ API. Section 4 discusses the representation and visualization of the ontology, while Section 5 presents test queries run on the proposed ontology. Section 6 presents conclusions and suggestions for future work.

II. Related Work

Berslin and Decker [10] and Berslin et al. [1] propose the use of Semantic Web mechanisms in order to bridge the isolation and fragmentation of todays SNS. Public vocabularies and ontologies can be used to give meaning to Social Networks and interconnect social websites. The FOAF ontology [11] provides a formal, machine readable representation of user profiles and friendship networks. The SIOC Core Ontology provides the main concepts and properties required to describe information from online communities (e.g., message boards, wikis, weblogs, etc.) on the Semantic Web [12]. The SIOC and FOAF ontologies are used in combination with metadata vocabularies like Dublin Core [13] and SKOS [14] for describing user-generated content on the Social Web. Zhou and Wu in [15] propose an ontology representing SNSs based on FOAF in order to resolve the problem of social data inconsistency and to achieve interoperability among multiple social network services. Their ontology defines some of the basic attributes of a generic SNS API, such as operations, arguments and responses, combined with some user profile and contact attributes borrowed by FOAF ontology, but it does not provide any structural description of the resources that can be accessed through it.

While the above approaches describe generic concepts about people, content and SNS, they do not describe the functional and structural aspects of specific SNS APIs necessary for building ontology based Mashups. Specialized ontology-based representations of the APIs of existing SNS could be used in combination with the above ontologies and vocabularies in order to bridge abstract concepts with specific resources and actions provided by each API.

Hartmann et al. [16], Zang et al. [3], and Wong and Hong [17] investigate how users with programming skills and programmers build Mashups that make use of public APIs provided by popular web 2.0 services. Most of those users are self-taught and depend on the documentation of the API they want to use. Some of the most common problems encountered when creating Mashups is the complexity of communicating data from one server to another and the lack of proper tutorials and examples in the documentation [3].

Dietrich and Elgar [6] propose that knowledge about structural and behavioural properties of software can be shared across the software engineering community in the form of design patterns expressed in the web ontology language (OWL). The inherent advantage of their approach is that it yields descriptions that are machine processable, but also suitable for a community to share knowledge taking advantage of the decentralized infrastructure of the Internet [6]. Ontology-based representations of SNS APIs can bring the same advantages for the community of Mashup developers.

Kurkovsky, Strimple and Nuzzi in [18] discuss the possibility of convergence of Web 2.0 and SOA, while Xiao et al [8][9] propose the use of ontologies for context-aware web service discovery and automatic service composition. The availability of ontology-based representations of SNS APIs can also help to build software able to automatically compose services that integrate data and functionality from SNS.

Our work takes into consideration the above works by providing an ontology-based representation of Google+ API, compatible with Semantic Web mechanisms and ontology based service discovery and composition approaches that can be used for knowledge sharing and as part of ontology-based Mashups that integrate Google+ functionality and data.

III. An Ontology Based Representation of the Google+ API

Google+ is an SNS operated by Google Inc. The service was launched on June 28, 2011in an invite-only testing phase and went public on September 20, 2011. Google+ integrates longer existent Google social services such as Google Profiles and Google Buz, and introduces new features identified as Circles for organizing users' connections into custom groups and Hangouts for group video chat [19]. Google+ became popular form the very first days of its testing phase and in Octomber 2011 reached 40 million users [19].

On September 15, 2011 Google released its first open API for Google+ [20]. Google+ API follows a RESTful API design, meaning that applications use standard HTTP methods to retrieve and manipulate Google+ resources. The API is currently read only, thus it provides only methods for retrieving and searching resources through the HTTP GET method. The API can be used free of charge, with applications being limited to a courtesy usage quota. Developers can request a higher limit for their applications for a fee. Many API calls require that the user of the application is granted permission to access their data. Google uses the OAuth 2.0 [21] protocol to allow authorized applications to access user data. Resources in the Google+ API are represented using JSON [22] data formats. It also supports pagination and partial responses for sending only requested fields instead of the full representation of a resource. The API currently provides read only access to three main types of resources named "Persons", "Activities" and "Comments". Person resources represent Google+ API users, Activities resources stand for content shared by users and Comments resources are content posted as a replies to Activities. Google also provides free client libraries for various programming languages including Python, PHP, Ruby, Javascript and Java.

In order to describe the structural and be properties of Google+ API in a way that can be shared among Software Developers and automatically interpreted by software components, we have introduced an ontology based representation of its main characteristics, resources and actions. For designing our ontology we followed the steps described by Noy and MacGuinness in [23]:

A. Specification of the domain and the purpose of the ontology

The domain of the ontology is the Google+ API and more specifically its structural and functional properties. That is, the data interchange and auhentication methods it uses, the types of entities that can be accessed through it and their attributes, and the actions that can be performed through it on these entities. The purpose of the ontology is dual: On the one hand the ontology is playing the role of a shareable and browsable knowledge base for researchers and programmers that want to develop applications and Mashups that integrate Google+ data and functionality, while on the other hand, because of its machine interpretable format, it may be used for building inter-operable ontology based social networking software. Such software will be programmed in a higher level of abstraction and use automatic reasoning on ontologies for providing integration with Google+.

B. Enumeration of important terms in the ontology

For enumerating the important terms in the ontology we studied the Google+ API documentation available online [24]. Through the documentation pages we identified references to key terms such as "Authorization Protocol", "Value Type", and "Parameter". Other terms like "Action Type", "Field" and "Resource Type" where produced through generalization of the descriptions provided by the documentation.

C. Considering reusing existing ontologies

The FOAF ontology describes user profiles and friendship networks, while the SIOC Core Ontology provides the main concepts and properties required to describe information from online communities. Both describe concepts relative to SNS at a high level of abstraction. For describing Google+ API, we needed lower level concepts such us urls, resources and methods that are not provided by those ontologies. The ontology proposed by Zhou and Wu in [15] defines some of the basic attributes of a generic SNS API, such as operations, arguments and responses, without describing them further or defining relations between them and the resources accessed through them. Thus, there was no important gain in reusing concepts from these ontologies for building our ontology. However, we would like to connect our ontology with ontologies like those in the future.

D. Specification of the classes of the ontology and class hierarchy

The classes of an ontology describe the main concepts of its domain. Since the domain of our ontology is the Google+ API, its classes will represent the concepts that are necessary for describing its structural and functional properties. Based on the documentation of the API we defined the following classes: *API* (an API), *APIType* (an API type), *DataFormat* (a data interchange format), *AuthorizationProtocol* (an authorization protocol used to access the API), *ResourceType* (a resource type provided by the API), *Field* (a field of a resource; fields represent attributes of a resource), *Action* (an action that can be performed to Resource), *ActionType* (an action type), *Parameter* (a parameter of an action), *ValueType* (the type of the value contained in a field or a parameter) and *DataStructure* (the type of the data structure contained in a field or a parameter).

The domain of the ontology is found to be flat in terms of generalization. The concepts we used for describing the API are considered to belong all at the same level of generality. Thus the classes of the ontology are disjoint with each other and no subclasses where defined.

E. Specification of the properties of the classes and property value types

The properties of a class represent the characteristics of the corresponding concept. The API is described in terms of its type, the format in which it exchanges data, the authorization protocol it supports and the resource types it provides. It has a name property, a base url used to build http request urls, and a documentation url where developers can access the official documentation of the API. The API provides some types of Resources. A Resource type has a name and may have a specific documentation url. A Resource type consists of Fields and can provide Actions. A Field is characterized by the type of its value (e.g. String, Integer, or Resource) and the type of its data structure (a single value or a structure like a list). An Action can have required or optional parameters and be performed by an HTTP/1.1 GET, PUT, POST or DELETE method. The Action also has a url mask used to build the http request url, and may require authentication using a token that has been granted to the caller application. Finally, a Parameter has a name, and it (the parameter) may be required or not.

Figure 1 depicts the classes and object properties of the Google+ API ontology. While analyzing the Google+ API we found that in some cases the Field of a Resource Type provides a reference to another Resource Type. This type of connection between resource types through their fields is not clearly presented in the API documentation, and a developer has to study the detailed documentation of the responses of various actions in order to detect it. We describe this type of connection in our ontology with the *connectsWith* object property of *Field* Class. There are also some common optional parameters that can be applied to any action. We used the *hasCommonParameter* object property connecting *API* and *Parameter* classes to describe this relation.

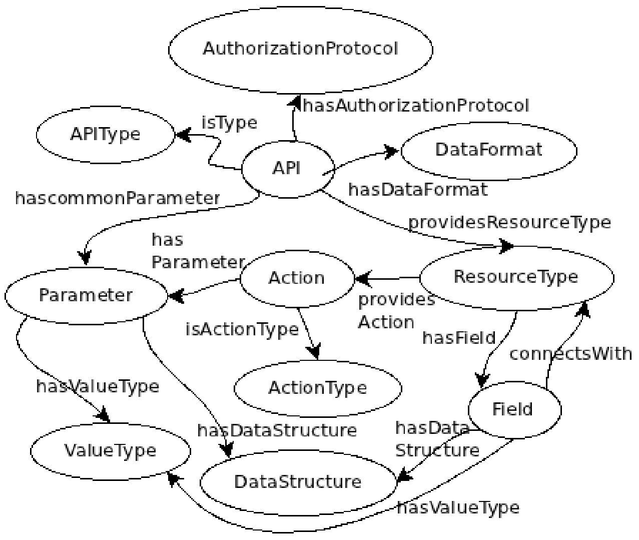
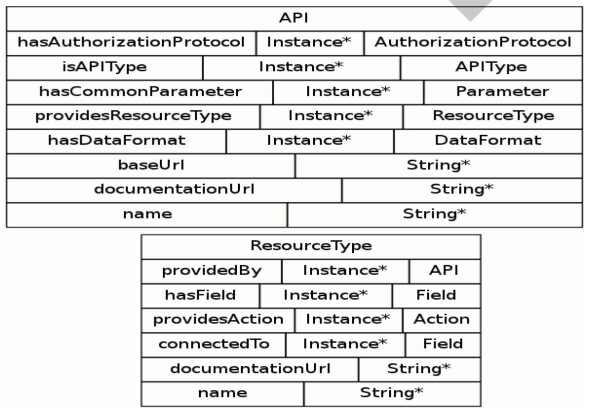


Figure 1. The classes and object properties of the ontology. An oval represents a class and an arrow stands for an object property.

F. *Specification of the value types and restrictions of the properties*

We defined the value types and restrictions of the properties of the ontology by analyzing the classes specified at the previous step. For example, the type property of the *API* class takes exactly one value that has to be instance of the *APIType* class, while the *connectsWith* property of the Field class can have at most one value of the type *ResourceType*. Figure 2 lists the properties and their value types for the *API* and *ResourceType* classes.



API		
hasAuthorizationProtocol	Instance*	AuthorizationProtocol
isAPIType	Instance*	APIType
hasCommonParameter	Instance*	Parameter
providesResourceType	Instance*	ResourceType
hasDataFormat	Instance*	DataFormat
baseUrl	String*	
documentationUrl	String*	
name	String*	

ResourceType		
providedBy	Instance*	API
hasField	Instance*	Field
providesAction	Instance*	Action
connectedTo	Instance*	Field
documentationUrl	String*	
name	String*	

Figure 2. Properties and value types of API and ResourceType classes.

G. *Creation of instances*

We defined the Instances of the Ontology based on the documentation of the API. We firstly created an instance of the *API* class representing the Google+ API. Since Google+ API is a Restful API, we created the *RestfullAPI* instance of the *APIType* class. The API uses the JSON data structure, so we created a *DataFormat* instance for it. The API also uses the OAuth authentication protocol for granting access to applications, so *OAuth* is an instance of the *AuthorizationProtocol* Class. Google+ API is currently read only, so all its actions are of *ActionType GET*, corresponding to the GET HTTP/1.1 method.

Based on our study of the parameters and return values of the Actions provided by the API, we identified 5 instances of the *ValueType* class: *String*, *UnsignedInteger*, *Boolean*, *DateTime* and *ResourceType*.

Two instances of the *DataStructure* class where also created: *SingleValue* and *List*.

The API explicitly specifies three main resource types (People, Activities and Communities), but with a more thorough study we identified a much larger number of resource types. The API does not currently provide actions for directly accessing all those resource types, but they can be indirectly accessed through the actions provided by the main three resource types. In our ontology we defined all the identified resource types as instances of *ResourceType* Class. Thus we created 25 instances of the *ResourceType* class: *Access* (identifies who has access to see an activity), *AccessItem* (an Access entry), *Activity* (a note that a user posts to her stream), *ActivityFeed* (list all of the activities in the specified collection for a particular user), *Actor* (the person who performes an activity), *Attachment* (the media objects attached to this activity), *CommentObject* (the object of a comment), *Circle* (a Google+ Circle), *Comment* (a comment is a reply to an activity), *CommentFeed* (list of all comments for an activity), *Email* (an email adderess for a person), *Embed* (if an attachment is a video, the embeddable link), *Name* (an object representation of the individual components of a person's name), *Object* (the object of an activity), *Organization* (an organization with which a person is associated), *PeopleFeed* (a list of all public profiles), *Person* (a person as represented in the Google+ API), *Place* (a place where a person has lived), *Plusoners* (people who +1'd an activity), *PreviewImage* (the preview image for photos or videos), *ProfileImage* (the representation of the person's profile photo), *Provider* (the service provider that initially published an activity), *Replies* (comments in reply to an activity), *Resharers* (people who reshared an activity) and *Url* (a URL for a person).

Finally, we created an instance of Field class for every property of every *ResourceType*, an instance of *Action* class for every action presented in the documentation of the API, and an instance of the *Parameter* class for every action parameter.

IV. Representation and Visualization of the Ontology

For the representation of the ontology we used the RDF/XML exchange syntax for the OWL ontology language. We used VIM text editor for editing the XML expressions of the classes and the properties and the specialized ontology editing software Protégé for checking the ontology, creating instances, and producing visualizations. Figure 3 is a visualization depicting the connections detected between the main resource types in the ontology. From this visualization we observe for example that a resource of type Person can be the Actor of an Activity or a Comment, or a member of a feed of people that Reshared or "PlusOned" (a term that is used by Google+ for evaluating other user's activities) the Object of an Activity.

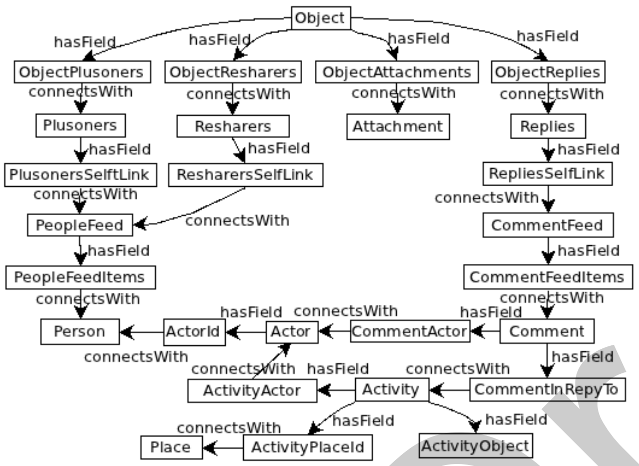


Figure 3. Connections between the main resource types in the ontology.

Figure 4 depicts all the fields of the Object resource type and their types.

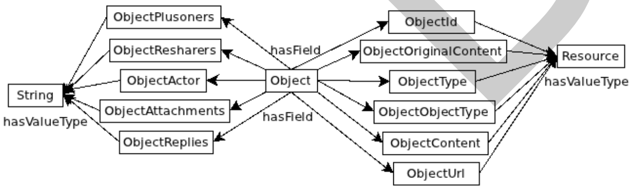
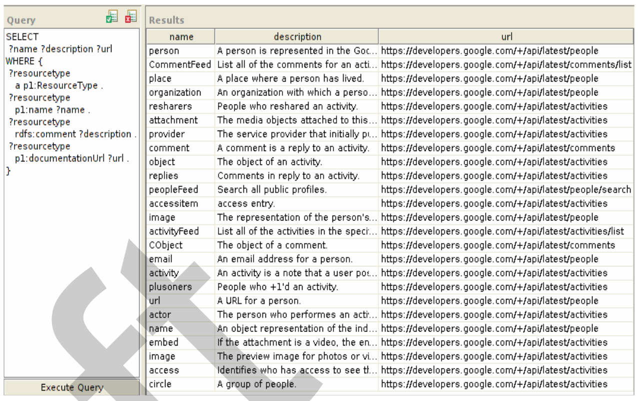


Figure 4. Fields of the Object resource type and their value types.

V. Test Queries

In order to test the proposed ontology we run test queries regarding the completeness and correctness of the resulting ontology and validated the results. We queried for all class instances and their properties and cross-checked the returned results with the API documentation pages. We also made sure that all the identified instances were returned. Figure 5 depicts the query for getting the name, description and documentation url for all instances of *ResourceType* class.

We also the run two sets of usage test queries and verified the returned results. For the first set of queries, we tried to extract information useful for developers that wish to use the API for building Mashups. Such queries are: (1) What authentication protocol is supported by Google+ API? (2) What is the API's documentation url? (3) What actions and what parameters can be used for directly accessing a Person resource? (4) What resources can be directly accessed through the API? (5) What are the resource types that provide a second rank reference to the Person resource type (i.e. Have a field that connects to a resource type that has a field that connects to Person)?



```
SELECT
 ?name ?description ?url
WHERE {
 ?resourcetype
   a p1:ResourceType .
 ?resourcetype
   p1:name ?name .
 ?resourcetype
   rdfs:comment ?description .
 ?resourcetype
   p1:documentationUrl ?url .
}
```

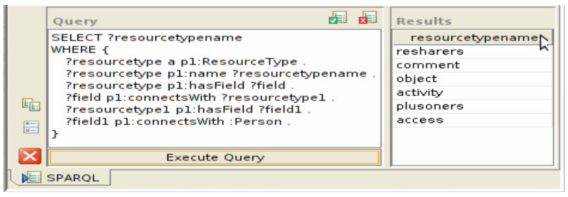
name	description	url
person	A person is represented in the Goc...	https://developers.google.com/+/api/latest/people
CommentFeed	List all of the comments for an acti...	https://developers.google.com/+/api/latest/comments/list
place	A place where a person has lived.	https://developers.google.com/+/api/latest/people
organization	An organization with which a perso...	https://developers.google.com/+/api/latest/people
resharers	People who reshared an activity.	https://developers.google.com/+/api/latest/activities
attachment	The media objects attached to this...	https://developers.google.com/+/api/latest/activities
provider	The service provider that initially pı...	https://developers.google.com/+/api/latest/activities
comment	A comment is a reply to an activity.	https://developers.google.com/+/api/latest/comments
object	The object of an activity.	https://developers.google.com/+/api/latest/activities
replies	Comments in reply to an activity.	https://developers.google.com/+/api/latest/activities
peopleFeed	Search all public profiles.	https://developers.google.com/+/api/latest/people/search
accessitem	access entry.	https://developers.google.com/+/api/latest/activities
image	The representation of the person's...	https://developers.google.com/+/api/latest/people
activityFeed	List all of the activities in the speci...	https://developers.google.com/+/api/latest/activities/list
CObject	The object of a comment.	https://developers.google.com/+/api/latest/comments
email	An email address for a person.	https://developers.google.com/+/api/latest/people
activity	An activity is a note that a user pos...	https://developers.google.com/+/api/latest/activities
plusoners	People who +1'd an activity.	https://developers.google.com/+/api/latest/activities
url	A URL for a person.	https://developers.google.com/+/api/latest/people
actor	The person who performes an acti...	https://developers.google.com/+/api/latest/activities
name	An object representation of the ind...	https://developers.google.com/+/api/latest/people
embed	If the attachment is a video, the en...	https://developers.google.com/+/api/latest/activities
image	The preview image for photos or vi...	https://developers.google.com/+/api/latest/activities
access	Identifies who has access to see tl...	https://developers.google.com/+/api/latest/activities
circle	A group of people.	https://developers.google.com/+/api/latest/activities

Figure 5. The SPARQL Query for getting the name, description and documentation URL for all ResourceType instances returns correct info for all the 25 identified resource types.

For the second set of queries we assumed that the ontology is used in ontology-based software for automatically invoking API's methods. Such software needs to extract low-level information about the actual method calls needed for performing an action and the structure of the data needed to be exchanged. Some example queries of this type are the following: (1) What is the APIs base url? (2) What is the APIs data format? (3) What is the urlMask of an Action? (4) What fields are contained in a Person resource type and what value type and data structure is each of them?

Moreover if such software is programmed in a higher level of abstraction, it may execute complex queries on the ontology in order to combine data form multiple API resources or to translate generic actions into sequences of API calls. For example: (1) What resource types that can be directly accessed through a GET Action provide a reference to an Email resource type? (2) What sequence of Actions can be called in order to get the image (PersonImage) of the Actor of an Activity?

We expressed the above queries in the SPARQL ontology querying language and executed using Protege. Figure 6 depicts a usage test query and the returned results.



```
SELECT ?resourcetypename
WHERE {
  ?resourcetype a p1:ResourceType .
  ?resourcetype p1:name ?resourcetypename .
  ?resourcetype p1:hasField ?field .
  ?field p1:connectsWith ?resourcetype1 .
  ?resourcetype1 p1:hasField ?field1 .
  ?field1 p1:connectsWith :Person .
}
```

resourcetypename
resharers
comment
object
activity
plusoners
access

Figure 6. SPARQL query for getting all the resource types that provide second rank access to Person resource type.

VI. CONCLUSIONS AND FUTURE WORK

Ontology-based representations of SNS APIs can help developers comprehend the structure and functionalities of SNS and their APIs and share this knowledge. Moreover they can be used to link those APIs with public Social Semantic Web ontologies and vocabularies and for enabling automatic ontology-based service composition.

We studied the API provided by Google for its popular Google+ SNS and created an ontology based representation of its structural and functional properties. For designing the ontology we followed the methodology proposed by Noy and MacGuinness in [23]: First we specified the domain and the purpose of the ontology, then specified the classes of the ontology, the hierarchy, the properties and finally we created the instances. We tested the ontology with SPARQL queries. The proposed ontology reveals the existence of important resources and connections between them that are not clearly presented in the official documentation. We identified a total of 25 resource types in Google+ API connecting with each other in various ways. We have made the ontology publicly accessible in OWL format at http://goo.gl/Oefl2.

In this work, we focused on representation of the basic structural and functional features of Google+ API such as the resources it provides, the way they connect with each other and the actions they provide. We would like to extend the ontology with descriptions of the authentication process, the manipulation of paging and partial queries and bindings of the actions to client libraries method calls, in order to support automatic invocation of the API calls from ontology driven applications. In the near future we would also like to connect the ontology with ontologies and vocabularies like FOAF and SIOC that describe more abstract concepts about users, social networks and content. Finally, we would like to create ontology based representations for the APIs provided by other popular SNS such as Facebook and Twitter and to use them for building ontology-based mashups that automatically combine data and functionalities from multiple SNS.

REFERENCES

[1] J.G. Breslin, A. Passant, and S. Decker , “The Social Semantic Web”, Springer-Verlang Berlin Heidelberg, 2009

[2] Tim O'Reilly, “What Is Web 2.0: Design Patterns and Business Models for the Next Generation of Software”, Published in: International Journal of Digital Economics No. 65, March 2007, pp. 17-37.

[3] N. Zang, M.B. Rosson, and V. Nasser, “Mashups: who? What? Why?”, In: CHI 2008: CHI 2008 extended abstracts on Human factors in computing systems, ACM, New York, 2008, pp. 3171-3176.

[4] T. R. Gruber, “Toward Principles for the Design of Ontologies Used for Knowledge Sharing”, In International Journal of Human-Computer Studies, Vol 43 Issue 5-6, Nov./Dec. 1995, pp. 907-928.

[5] J. Hendler, “Agents and the Semantic Web”, In IEEE Intelligent Systems, Vol. 16 No 2, 2001, pp. 30-37.

[6] J. Dietrich and C. Elgar, “Towards a web of patterns”, In: Web Semantics: Science, Services and Agents on the World Wide Web, vol. 5, num. 2, Elsevier, 2011.

[7] B. Guo, D. Zhang, and M. Imai, “Toward a cooperative programming framework for context-aware applications”, In Personal and Ubiquitous Computing, Vol 15, Issue 3, March 2011, pp. 221-233.

[8] H. Xiao et al, “An automatic approach for ontology-driven service composition”, Proc. IEEE International Conference on Service-Oriented Computing and Applications (SOCA) 2009, Taipei, Taiwan, 14-15 December 2009, pp 1-8.

[9] H. Xiao et al, “An Approach for Context-Aware Service Discovery and Recommendation”, Proc., IEEE International Conference on Web Services (ICWS), 5-10 July 2010, Miami, FL, 2010, pp. 163 – 170.

[10] J. Berslin and S. Decker, “The Future of Social Networks on the Internet: The Need for Semantics”, IEEE Internet Computing, vol. 11, November 2007, pp. 86-90.

[11] The Friend of a Friend (FOAF) project, online at http://goo.gl/Rdpja, retrieved December 2011.

[12] U. Bojārs and J.G. Breslin (editors), “SIOC Core Ontology Specification”, W3C Member Submission 12 June 2007, online at http://goo.gl/8OQV1, 2007, retrieved December 2011.

[13] Dublin Core Metadata Initiative, “Dublin Core Metadata Element Set”, Version 1.1, online at http://goo.gl/MHLlw, 2010, retrieved December 2011.

[14] A. Miles and S. Bechhofer (editors), “SKOS Simple Knowledge Organization System Reference”, W3C Recommendation 18 August 2009, online at http://goo.gl/ypDOU, 2009, retrieved December 2011.

[15] B. Zhou and C. Wu, “Social networking interoperability through extended FOAF vocabulary and service”, Proc. 3rd International Conference on Information Sciences and Interaction Sciences (ICIS), 23-25 June 2010, Chengdu, China, 2010, pp. 50 – 55.

[16] B. Hartman, S. Doorley, and S.R. Klemmer, “Hacking, Mashing, Gluing: Understanding Opportunistic Design”, in IEEE Pervasive Computing, vol. 7 issue 3, July 2008.

[17] J. Wong, J. and J. Hong, “What do we "mashup" when we make mashups?”, Proc. WEUSE '08: Proceedings of the 4th international workshop on End-user software engineering, 2008.

[18] S. Kurkovsky, D. Strimple, and E. Nuzzi, “Convergence of Web 2.0 and SOA: Taking Advantage of Web Services to Implement a Multimodal Social Networking System”, proc. 11th IEEE International Conference on Computational Science and Engineering - Workshops, 2008, pp. 227-232.

[19] Wikipedia, Google+, online at http://goo.gl/N5rou, retrieved December 2011.

[20] C. Chabot, “Getting Started on the Google+ API”, The Google+ Platform Blog, online at http://goo.gl/EPfiM, September 15, 2011, retrieved December 2011.

[21] E. Hammer-Lahav (editor), “The OAuth 1.0 Protocol, Internet Engineering Task Force (IETF)”, online at http://goo.gl/eN6VT, April 2010, retrieved December 2011.

[22] D. Crockford, “The Media Type for JavaScript Object Notation (JSON)”, online at http://goo.gl/7oDGo, July 2006, retrieved December 2011.

[23] N. F. Noy and D. L. McGuinness, “Ontology development 101: a guide to creating your first ontology”. Technical Report KSL-01-05 and Stanford Medical Informatics Technical Report SMI-2001-0880. Stanford Knowledge Systems Laboratory. Available at http://goo.gl/kr6n4, 2001, retrieved December 2011.

[24] Google, Inc., “Google+ API”, online at http://goo.gl/q2jai, retrieved December 2011.

ACKNOWLEDGEMENT

This research described in this paper was partly funded by the National Strategic Reference Framework programme 2007-2013, project MIS 296121“Hellenic Open University”.