Computers, Materials & Continua CMC, vol.61, no.3, pp.929-950, 2019

XML-Based Information Fusion Architecture Based on Cloud Computing Ecosystem

I-Ching Hsu[1, *]

Abstract: Considering cloud computing from an organizational and end user computing point of view, it is a new paradigm for deploying, managing and offering services through a shared infrastructure. Current development of cloud computing applications, however, are the lack of a uniformly approach to cope with the heterogeneous information fusion. This leads cloud computing to inefficient development and a low potential reuse. This study addresses these issues to propose a novel Web 2.0 Mashups as a Service, called WMaaS, which is a fundamental cloud service model. The WMaaS is developed based on a XML-based Mashups Architecture (XMA) that is composed of Web 2.0 Mashups technologies, including Web Data, Web API, Web Interaction, and Web Presentation to associate with existing service models. To demonstrate the feasibility of this approach, this study implemented a Ubiquitous Location-based Service System (ULSS) that is a cloud computing application developed based on WMaaS to provide continuous and location-based schedule information for organization monitoring and end user needs.

Keywords: Web 2.0 mashups, open data, XML, cloud computing.

1 Introduction

Authors are encouraged to use the template for Microsoft Word, to prepare the final version of their manuscripts and facilitate typesetting. Authors may elect to submit two versions of their manuscript, one for the printed version of the journal, and the other for the on-line version of the journal. Illustrations in color are allowed only in the on-line version of the journal. Cloud computing has become one of the most promising information solutions and business trends in recent years. The first to introduce the term cloud computing was Google's CEO Eric Schmidt. The term refers to the important and long-term trend in computing over the Internet. Many institutions and companies provide definitions and solutions for cloud computing [Dillon, Wu and Chang (2010)]. However, there is still no widely accepted definition of cloud computing. The NIST [NIST (2019)] defines three types of cloud computing service models: Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS). SaaS provides services to cloud clients, while IaaS and PaaS provide services to cloud application developers.

[1] Department of Computer Science and Information Engineering, National Formosa University, Huwei Township, Yunlin County 632, Taiwan.

[*] Corresponding Author: I-Ching Hsu. Email: hsuic@nfu.edu.tw.

Cloud computing applications generally incorporate combinations in the three different service models.

Cloud computing is service-oriented architecture providing a set of distributed heterogeneous cloud resources delivered via the Internet to cloud computing developers for facilitating the rapid building cloud computing applications. Considering cloud computing from a software engineering point of view, it is a new paradigm for deploying, managing and offering services through a shared infrastructure. Web 2.0 Mashups offer entirely new opportunities for information system developers by integrating cloud resources to facilitate the rapid building new generation of information systems over the Internet. The development of the existing cloud computing environment lack a uniform approach to cope with the heterogeneous information fusion, including various cloud computing resources, cloud client constraints, and cloud computing middleware. This leads cloud computing to inefficient development and a low potential reuse. The Web 2.0 Mashups provide a catalytic solution to this problem.

Within the last few years, the Internet has greatly changed our way of sharing resources and information. As well known, Web 2.0 is recognized as the next generation of web applications proposed by O'Reilly [O'Reilly (2005)]. The main feature of Web 2.0 applications is that they provide a medium for the sharing and exchange of Web resources [von Alberti-Alhtaybat and Al-Htaybat (2016); Simeonova (2018)], such as knowledge, data, Web Multimedia, Web Data and Web API. These resources can be considered regarded as cloud resources. They allow cloud computing developers to take advantage of these resources to reduce the cost or produce new integrated solutions by associating with resources, which they could not have provided on their own. Additionally, many studies adopt Web 2.0 technologies to build web-based applications in various domains [Iannone (2019); Wu, Huang, Li et al. (2019)].

To address heterogeneous issues of cloud computing, this study argues that Web 2.0 Mashups can be adopted as a common scheme to uniformly integrate cloud resources via a fundamental cloud service model. The fundamental cloud service model is Web 2.0 Mashups as a Service, called WMaaS, which developed based on an XML-based Mashups Architecture (XMA). The XMA is composed of Web 2.0 Mashup technologies, including Web Data, Web API, Web Interaction, and Web Presentation, to remove the heterogeneous issues of cloud computing ecosystem. Additionally, WMaaS can also be combined with existing service models, SaaS, PaaS, and IaaS to facilitate cloud computing applications development. This implies at least two requirements for the development of the XMA. The first is heterogeneity. It can present a metamodel to integrate Web 2.0 Mashups technologies into cloud computing applications independently from the SaaS, PaaS, and IaaS. The second is performance. It can facilitate to develop a generalized cloud computing architecture to promote adequate Web-based information transcoding for various clients.

This study makes three main contributions. First, an XML-based Mashups Architecture (XMA) based on XML technologies is presented to cope with the heterogeneous issues of cloud computing. Second, the XMA is adopted to develop a novel service model, Web 2.0 Mashups as a Service (WMaaS), which enables cloud developers to combine cloud resources that are distributed over the Internet to create new cloud applications that can

satisfy customers' needs. Third, a cloud computing application, Ubiquitous Location-based Service System (ULSS), is implemented based on WMaaS to provide continuous and location-based schedule information for organization monitoring and end user needs. The ULSS carries out to integrate four emerging research areas: Web 2.0, Open Data, Cloud Computing, and Ubiquitous Context-awareness [Hsu (2013a)].

The remainder of paper is organized as follows. The next section presents some related works. Section 3 presents an XML-based Mashups Architecture (XMA) based on XML technologies. Section 4 describes Web 2.0 Mashups as a Service (WMaaS). In Section 5, the study implemented a Ubiquitous Location-based Service System (ULSS) to demonstrate the feasibility of WMaaS. Finally, summary and concluding remarks are included.

2 Related work

In recent years, more and more cloud service providers have published APIs that enable Web application and APP developers to easily integrate open data and web services, instead of developing them by themselves. Mashups can be considered to have an important medium in the evolution of Web 2.0 era. In the past years, there are many studies to discuss the mashups application in various domains [Boulakbech, Messai, Sam et al. (2016); Ghiani, Paternò, Spano et al. (2016); Zhang, Fu, Sun et al. (2016); Zhong, Fan, Tan et al. (2018); Wang, Wu and Hsu (2019)]. In Ghiani et al. [Ghiani, Paternò, Spano et al. (2016)], system developers can create new mashups by existing interaction components via a graphical environment. A mobile-application prototype [Boulakbech, Messai, Sam et al. (2016)] is implemented adopted mashups Web services to provide customized touristic plans. In Lee [Lee (2015)], authors focus on web services mashup to develop novel algorithms for the automatic discovery and composition of Web APIs. It should be noted that above-mentioned studies did not provide a complete, flexible, and versatile mashup architecture to facilitate software development. In Chavarriaga et al. [Chavarriaga, Jurado and Rubio (2017)], authors propose an XML-based domain specific language approach for client-side Web applications. This study proposes the XML-based Mashups Architecture (XMA) based on XML technologies is revealed to cope with the heterogeneous, flexible, and versatile issues of mashup applications.

Cloud computing is an emerging paradigm where computing resources are offered over the Internet. There are two problems of using cloud computing resources. Cloud resource providers cannot easily publish their services for cloud users, and cloud users cannot easily find useful cloud resources. These problems result from the heterogeneous cloud resources. The Mobile Ubiquitous Brokerage as a Service (MUBaaS) [Yan, Sun, Liu et al. (2016)] permits n-devices of a user to access diverse cloud services. The Workflow platform as a service (WaaS) [Fan, Hussain and Hussain (2015)] supports users to define, and integrate workflow based applications to facilitate the rapid development of cloud computing. In Herbold et al. [Herbold and Hoffmann (2017)], authors propose the model-based testing as a service to deal with the complexity of the Web service based on using cloud infrastructures. Additionally, many studies of integrating cloud services architecture with various application domains have been reported recently by researchers [Mohamadi Bahram Abadi, Rahmani and Alizadeh (2018); Borangiu, Trentesaux, Thomas et al. (2019); Deng, Ren, Liu et al. (2019); Feng, Wu, Zhang et al. (2019)].

The service models of cloud computing are usually classified as Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS) [Chakraborty, Ramireddy, Raghu et al. (2010)]. IaaS provides the hardware and administrative services that are required to store cloud applications and a physical platform for running cloud applications. These infrastructure resources, including computer server, storage system, network equipment, and data center, offer basic storage and computing capabilities as standardized services over the Internet. The cloud developers would typically deploy their applications on the infrastructure. Typical examples are Amazon EC2 (Elastic Cloud Computing) Service and S3 (Simple Storage Service) where compute and storage infrastructures are available to public access as a utility.

PaaS provides an integrated environment or middleware using which cloud developers can implement and deploy cloud applications without the cost or need to purchase hardware or software. In Kryukov et al. [Kryukov, Demichev and Polyakov (2016)], authors suggest that the traditional global grid systems will be transferred to creating convenient and efficient means of access to the distributed cloud resources. Usually, a set of development tools and services are run on servers in the cloud. Practical examples include Google App Engine, Microsoft Windows Azure platform, Salesforce Force.com, and Aneka. Google App Engine provides a python or java runtime environment and APIs using which developers can build Web applications on the same scalable systems that power Google applications. SaaS provides on-demand access to web-based applications that are maintained centrally by a provider.

Hadoop is a cloud computing platform that is an open source project under the Apache Software Foundation. Map Reduce and HDFS (Hadoop Distributed File System) are the two cores of Hadoop; the former enables distributed computing, and the latter achieves a distributed file system with the advantages of high fault tolerance. The YARN is a kind of cluster manager that is proposed in Hadoop 2 version. Apache Spark is also an open-source platform that adopts in-memory technique, and finally stores the computing results in the hard disk, thus the I/O time can be reduced. The Apache Mesos is a new distributed system core of Hadoop. Mesos regards the multi-node resources as a single high-performance computer, including the CPU, memory, hard disk and other computing resources [Saha, Beltre and Govindaraju (2018)]

3 XML-based mashups based on cloud ecosystem

Mashup is now a major Web 2.0 culture. To develop WMaaS, this study first needs to concrete the concept of the Web 2.0 Mashups. An XML-based Mashups Architecture (XMA) is proposed to include Web Data layer, Web API layer, Web Interaction layer, and Web Presentation layer. This research also describes that XML should form the backbone of XMA in support of technologies relevant to the function of each layer.

3.1 XML-based mashups architecture

XMA is developed based on XML technologies, including XML, XML Schema, XSLT, XPath, RDF, OWL, and Namespace. This architecture is depicted in Fig. 1, which can be represented in four layers: Web Data layer, Web API layer, Web Interaction layer, and Web Presentation layer. The XML-based technologies are adopted across the four layers.

XMA provides a flexible infrastructure that developer can dynamically add, replace, and remove components in each layer. Each layer contains multiple technologies, all of them providing a service suitable to the function of that layer. WMaaS can cope with heterogeneous information fusion, which is mainly due to use of XML technologies.

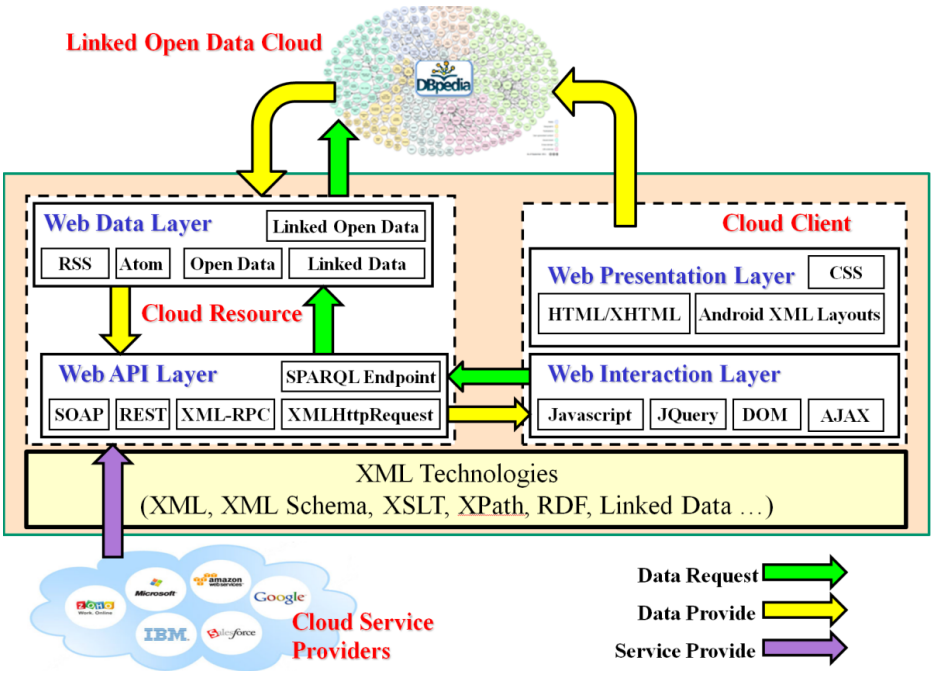

Figure 1: XMA developed based on XML-based technologies for cloud ecosystem

Because all layers are involved when a request is sent from a cloud client to a cloud computing application, upper layers must rely on lower layers to process cloud resources over the Internet. Web Data layer is widely used to distribute users regarding changes of contents at some SaaS website. Web API layer is used to facilitate data exchange between cloud computing applications and allow the creation of new applications. The Web Interaction layer supports interactive web technologies. The Web Presentation layer provides independence from differing data representations by translating the format for a specific cloud client from an application format to a valid markup language.

Fig. 2 shows the semantic structure between XML and Web 2.0 Mashups as a UML class diagram. The UML class diagram has as goal to give a graphical overview of the domain concepts and the relations among them. The components of Web 2.0 Mashups include Cloud Resource, Web Interaction, and Web Presentation. There are two primary Cloud resources, namely data resource and service resource. Web Data is a typical cloud data resource, while Web API is a typical cloud service resource. There is dependency relationship, annotated with a “core technology” stereotype, from Web 2.0 Mashups to XML. Its semantic indicates that XML is a core technology of Web 2.0 Mashsups and implies that a change to XML may cause a change in Web 2.0 Mashups.

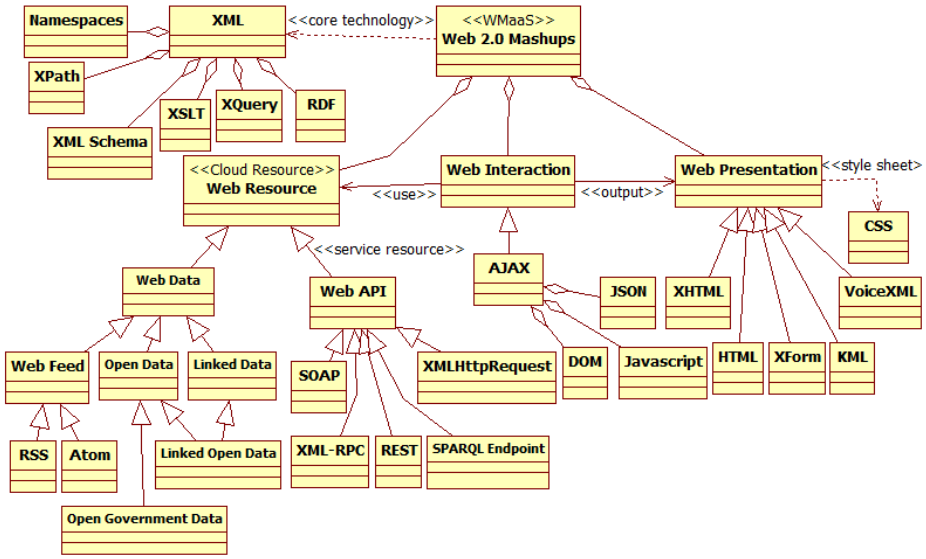

Figure 2: UML class diagram for XML-based mashups architecture

3.2 Web presentation

Web presentation technologies are mainly to provide a valid markup language for a specific cloud client. Markup languages include HTML, XHTML, XForm, KML, WML, and VoiceXML. These presentation markup languages are all based on XML standard to facilitate visually precise in Web 2.0 Mashups except for HTML. There are a number of different approaches in which data can be extracted from web pages to facilitate the reuse of data [Varlamov and Turdakov (2016)]. Tab. 1 lists some of the most popular presentation markup languages.

Table 1: Markup languages for Web representation

	XML-based language	Description	Cloud Client
HTML	No	Publishing language for Web 1.0	Web browser
XHTML	Yes	Publishing language for Web 2.0	Various cloud client devices
XForms	Yes	An XForms-based web form processes XML data using the Model-View-Controller approach that separates presentation, purpose and content.	Various cloud client devices
KML	Yes	Expressing geographic annotation and visualization	Google earth, Google map
WML	Yes	Publishing language for WAP phone	WAP mobile device
VoiceXML	Yes	Publishing language for voice phone	Voice phone

3.3 Web interaction

Cloud computing applications require more interactive web technologies than traditional Web systems. AJAX (Asynchronous JavaScript and XML) is not a new technology or language, but a new framework that combines various existing technologies, including XML, XHTML, XMLHttpRequest, and JavaScript. With the promotion of the XML and JavaScript, the AJAX framework becomes a basic ingredient in web 2.0 applications. It supports a more interactive navigation of Web 2.0 application has enhanced online collaboration and sharing information among users. Tab. 2 lists related technologies adopted by AJAX.

Table 2: Related technologies in AJAX

Technology	Description
Javascript	JavaScript is used in client-side development to enable interactivity to HTML/XHTML pages.
XML	XML allows users to define their own elements and attributes. The primary purpose of XML is to facilitate the sharing or exchange of structured data across different information systems or platforms.
XMLHttpRequest	XMLHttpRequest object can be used to create a GET or POST request asynchronously to communicate directly with the server.
XHTML	XHTML a reformulation of HTML 4.0 based on XML standard, which is intended to replace HTML to produce web documents.
CSS	CSS (Cascading Style Sheets) is a widely used mechanism for adding style (e.g. fonts, colors, spacing) to Web-based documents, including HTML-based, XHTML-based, and XML-based documents.
DOM	DOM is used to parse XML-based document that fetches from server, such as Web Feeds.
JSON	JSON a lightweight computer data interchange format that a text-based and human-readable format for representing simple data structures.

3.4 Web data

Web Data contains three types of data: Wed Feed, Open Data, and Linked Data. A Web Feed contains a structured information source, which is written in XML to provide machine-readable content on the Web [Hsu (2013b)]. This means that Web Feeds can be used to automatically transfer information from one website to another, without any human intervention. One major feature of Web 2.0 is to adopt Web Feeds to build a more maintainable and cooperative Web. Web Feeds allow both websites to publish frequently updated content such as Weblog, news headlines, real time information or Podcasts. RSS and Atom are currently the two main formats of Web Feed.

The intent of the Open Data motivation is similar to the other “open” motivations, for example Open Source, Open content, and Open Access. The Open Data is available on the

Internet with an open license [Berners-Lee (2009)]. Therefore, Open Data can be regarded as a kind of cloud resource. Anyone can use open source to facilitate the implementation of their information systems to reduce development costs and time. At present, the main provider of open data is from government agencies. Open Data provided by the government is filtered, and its availability and reliability are high. Such Open Data is called Open Government Data (OGD) [Máchová, Hub and Lnenicka (2018)]. In recent years, many countries are actively providing OGD to their citizens to promote the reuse of OGD [Vracic, Varga and Curko (2016); Nascimento, Da Rocha and Garcia (2018); Saxena (2019); Taiwan (2019); Talukder, Shen, Hossain Talukder et al. (2019)].

Linked Data is simply about using the Web to create typed links between data from different cloud resources [Bizer, Universität, Heath et al. (2009)]. The term Linked Data was coined by Tim Berners-Lee in the following principles [Berners-Lee (2009)]:

(1) Use URIs as names for things.

(2) Use HTTP URIs so that people can look up those names.

(3) When someone looks up a URI, provide useful information, using the standards (RDF, SPARQL).

(4) Include links to other URIs so that they can discover more things.

The Resource Description Framework (RDF) [W3C (2014)] is a standard model for data description and interchange on the Semantic Web applications. The SPARQL [W3C (2013)] is a query language for RDF. SPARQL can be used to express queries across diverse RDF-based data sources. According to the third principle, a Linked Data is a RDF-based document that can be queried by SPARQL to extract useful information. The RDF is developed based on XML. A RDF-based document has to meet the requirements of Well-formed XML. That is to say a Linked Data document also has to meet the Well-formed XML standard. In recent years, many studies [Nogales, Sicilia, Sánchez-Alonso et al. (2016); Kobayashi, Kume, Lenz et al. (2018)] adopted Linked Data to describe the relationship between different cloud resources to facilitate development of Web applications.

Linked Open Data (LOD) is a kind of Linked Data that meet the features of Open Data. As a result, LOD can benefit from both Linked Data and Open Data. LOD Cloud has published by major contributor, the Linking Open Data community project [W3C (2016)], has emerged as a collection of interlinked LOD datasets on the web. The new version of COD Cloud has been published in 2014, which contains 570 LOD datasets [Mital, Pani, Damodaran et al. (2015)]. In the LOD Cloud, there are many well-known LOD datasets, such as DBpedia, FOAF, W3C, GeoNames, etc. DBpedia is the largest LOD dataset containing extracted data from Wikipedia. It contains about 3.4 million concepts described by 1 billion RDF-based relationships. Many novel studies and applications are developed using DBpedia to get more useful information to enhance their usability and innovation [Zhu, Ren, Liu et al. (2016)].

3.5 Web API

Web 2.0 Mashups use Web API technologies to facilitate data exchange between applications and allow the creation of new applications. Most of the Web APIs are constructed based on the Web Services architecture [Sun, Rossing, Sinnema et al. (2010)].

They hide the detailed Web Services protocols from the developers and make it easier for the developers to use. Web service composition is a new paradigm to develop Web-based systems [Jeong, Rana, Hsu et al. (2016)]. The following are five main types of Web APIs. The detailed comparisons of the five Web APIs are shown in Tab. 3.

3.5.1 XMLHttpRequest

XMLHttpRequest is not a web service technology but a Web API that provides scripting languages to transfer XML or other text data between a client and a server. It is used to communicate asynchronously with a server-side component and dynamically update the source of an HTML page based on the response data. The data returned from XMLHttpRequest calls are often provided by third-party database servers. Besides XML-based format, XMLHttpRequest can be used to process data in other formats such as HTML, JSON, or plain text. For example, Google Map API uses Javascript XMLHttpRequest objects to send HTTP requests and receive responses from server. Google Maps API is one of the most widely known Web API among current Web 2.0 Mashup applications.

3.5.2 XML-RPC

XML-RPC is a remote procedure calling employing HTTP as the transportation protocol. It is a protocol for exchanging XML-based messages in a distributed environment. XML-RPC provides a standard for heterogeneous programs to communicate with each other regardless of their implementation language and system platform. An XML-RPC message is an HTTP-POST request. A request executes on the server and the body of the request is coded in XML-based format. The response value of the request is also formatted in XML.

3.5.3 Simple object access protocol

Simple Object Access Protocol (SOAP) is another communications protocol for Web services that emerged immediately after the XML-RPC. It is developed to address some of the limitations of XML-RPC, including only for RPC over HTTP, not easily extendable, and no support WSDL. SOAP adopts XML Base [Marsh (2001)] to determine a base URI for relative URI references used as values in message items. A SOAP binding describes how an underlying protocol is used to transport SOAP messages. Most of the current Web Services adopt SOAP over HTTP.

3.5.4 Representational state transfer

Representational State Transfer (REST) is a style of software architecture used to describe how Web resources, such as web service, web page, text, database, or website, are defined and addressed. REST is often used in a looser sense to describe service interfaces. Many of current Web services are developed on REST style, called REST-based Web services. The main advantages of REST-based Web services are lightweight, human readable, and easy to build [Barbaglia, Murzilli and Cudini (2017)].

3.5.5 SPARQL endpoint

SPARQL Endpoint [W3C (2017)] is a conforming SPARQL protocol service, which enables users to query a RDF-based dataset with the SPARQL language. The SPARQL Annotations in WSDL (SPDL) provides a specification for allowing SPARQL query indicates a specific URL associated with parameters to invoke web services and bind the returned information to SPARQL results. Additionally, some existing projects [Sbodio, Martin and Moulin (2010)] provide SPARQL Web Service or APIs for different programming languages to invoke SPARQL query.

Table 3: The comparison of various Web APIs

	XMLHttpRequest	XML-RPC	SOAP	REST	SPARQL Endpoint
Web API	Yes	Yes	Yes	Yes	Yes
Web Services	No	Yes	Yes	Yes	Yes
Specification	Yes	Yes	Yes	No	Yes
Architecture	No	No	No	Yes	No
XML binding message	No	Yes	Yes	No	Yes
Data format	XML, text, JSON	XML	XML	XML, text, JSON	XML, RDF

Many well-known companies, such as Google, Facebook, Microsoft, Yahoo, Amazon, and eBay, have published APIs based on web standards that allow users to access their cloud services and data. Tab. 4 lists some of the most popular Web APIs.

Table 4: The most popular websites for Web API

Web API	Cloud service model	Category	Web API Protocol	Data format
Google Apps	SaaS, PaaS	various services	XMLHttpRequest, SOAP, REST,	XML, RSS, JSON, KML, Atom
datahub.io	PaaS, PaaS	Linked Open Data Cloud	SOAP, REST, XML-RPC	XML, JSON,CSV, RDF
DBpedia	SaaS, PaaS	Wikipedia, Linked Open Data	REST, SPARQL endpoint	RDF
Facebook	SaaS, PaaS	Social network	SOAP, REST	XML, JSON
Microsoft Virtual Earth	SaaS, PaaS	Mapping services	XMLHttpRequest	KML, GeoRSS
Flickr	SaaS, PaaS	Photo sharing service	SOAP, REST, XML-RPC	XML, JSON, RSS
Amazon	SaaS, PaaS, IaaS	E-commerce	SOAP, REST,	XML
eBay	SaaS, PaaS	Online auction marketplace	SOAP, REST	XML
Del.icio.us	SaaS, PaaS	Social bookmark	REST	XML, RSS
Yahoo Geocoding	SaaS, PaaS	Yahoo Maps Geocoding service	REST	XML

4 Web 2.0 mashups as a service

The components of WMaaS, shown in Fig. 1, are described in the previous section. This section describes how WMaaS can be as a service model, and associated with existing service models of cloud computing. This study presents a stack framework, shown in Fig. 3, to locate and represent relevant service models of cloud computing. The layers IaaS, PaaS, and SaaS represent current service models of cloud computing. The top layer is cloud devices that are increasingly connected to the cloud SaaS applications. Therefore, the same web content needs to be rendered differently on various cloud devices. Heterogeneous issues span all the upper three layers. There are four characteristics of cloud computing lead to heterogeneous issues. The WMaaS are used to cope with these heterogeneous issues summary in Tab. 5.

Table 5: WMaaS supports cloud computing character

Cloud computing character	Heterogeneous issue	WMaaS
service-oriented computing paradigm	various kinds of remote service protocols	Web API
on-demand virtualization	various kinds of resources are organized in a virtual way	Web Data
more interactive web technologies	various kinds of interaction	Web Interaction
board Internet access	various kinds of cloud clients	Web Presentation

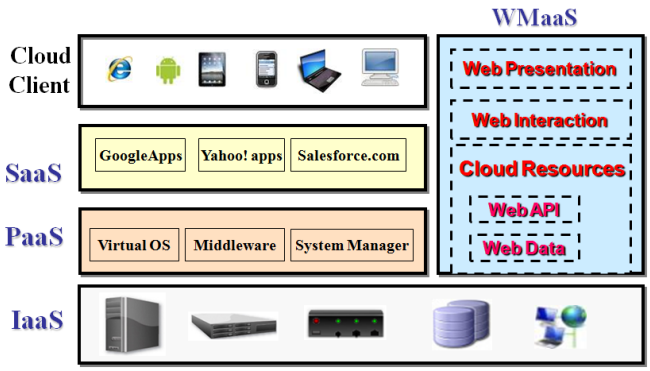

Figure 3: Cloud computing stack

The WMaaS is an extension of the Web 2.0 Mashups in which cloud resources are given well defined meaning, better enabling SaaS, PaaS, IaaS, and various cloud participants to work in cooperation. Additionally, the WMaaS can combine with existing cloud service models, SaaS, PaaS, and IaaS to facilitate the development of cloud computing applications. Fig. 4 shows the service-oriented architecture that is associated with various cloud participators and different cloud computing service models.

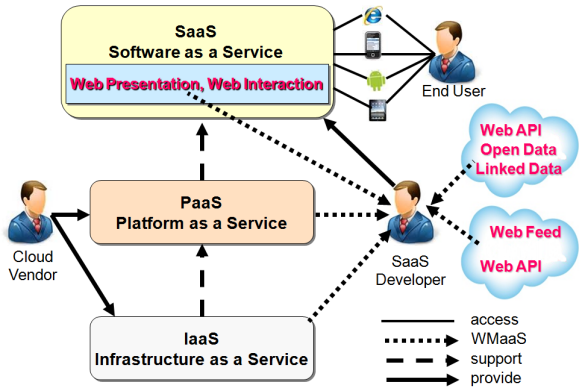

Figure 4: WMaaS associated with various cloud participators

5 Implementation and evaluation

Section 5.1 developed a Ubiquitous Location-based Service System (ULSS) based on the proposed Web 2.0 Mashups as a Service (WMaaS). The example is explained in detail, and help the reader better understand the WMaaS and how it can be adopted. Section 5.2 evaluated the ULSS based on the development of WMaaS.

5.1 Ubiquitous location-based service system based on WMaaS

Web 2.0 Mashups use Web API technologies to facilitate data exchange between applications and allow the creation of new applications. Most of the Web APIs are constructed based on the Web Services architecture [Sun, Rossing, Sinnema et al. (2010)]. They hide the detailed Web Services protocols from the developers and make it easier for the developers to use. Web service composition is a new paradigm to develop Web-based systems [Jeong, Rana, Hsu et al. (2016)]. The following are five main types of Web APIs. The detailed comparisons of the five Web APIs are shown in Tab. 3.

This section demonstrates the feasibility of WMaaS, we implemented a Ubiquitous Location-based Service System (ULSS) that is a cloud computing application to provide a continuous and location-based schedule information for personal needs. The main components of ULSS include: Location-Based Service Platform (LBS Platform), GPS Network, and Cloud Client. The Location-based Service Platform consists of the ULSS Portal Website that is deployed in Hadoop cloud computing environment to community inquiry for validated members. In the ULSS, GPS information are collected through the mobile phone and transferred to the Location-based Service Platform for storage. The dataflow-oriented architecture of ULSS is depicted in Fig. 5. The related technologies of WMaaS are used in ULSS summary in Tab. 6.

Table 6: The related Web 2.0 technologies employed in ULSS

Web 2.0 Technology	Service Type (WMaaS)	Description
XML	Web Feed (cloud resource)	The schedule and GPS location information are described in XML format for data sharing and transformation.
Atom	Web Feed (cloud resource)	The Google Calendar information is described in Atom format to support to Google Calendar API.
XML	Open Data (cloud resource)	The PM2.5 Open Data contains real-time PM 2.5 information presented in XML, which is provided by the Open Government Data of Taiwan.
RDF	Linked Data (cloud resource)	The introduction information of Taipei city presented in RDF-based Linked Data, which is offered by DBpedia.
AJAX	Web Interaction	A sample client was built using XHTML and AJAX to query the Google Calendar, and display the location information in real-time.
XHTML	Web Presentation	The location-based service information is rendered in XHTML for desktop PC.
Google Calendar	Web Presentation/ Web API	The schedule information, such as date, time, subject and location, can be displayed in Google calendar by the Google Calendar API.
Google Map	Web Presentation/ Web API	The location information can be displayed in Google Map by the Google map API.
Open Government Data(Taiwan)	Web API (cloud resource)	The Open Government Data of Taiwan [Taiwan (2019)] provides about twenty-one thousand open datasets. User can access these open data through Web API.
DBpedia	SPARQL (cloud resource)	DBpedia is the largest LOD dataset in the LOD cloud project. User can use SPARQL endpoint to access the RDF-based Linked Data.

Hadoop Cloud Computing is a PaaS for developing web applications that provides Web APIs to retrieve the various services, including data store, Google Accounts, Google Map, Google Calendar and Google email. Hadoop cloud computing environment also supports a web-based administration console for the SaaS developers to easily build, maintain, and extend their web applications.

Location-Based Service Platform (LBS Platform) is a website that provides the dynamic schedule information transcoding and inquiry. It uses XML-based documents and web services technologies to facilitate reusability of schedule information. The LBS Platform builds in a Hadoop cloud computing environment, which is composed of Transcoding Agent, user information and schedule information. The Transcoding Agent listens to the cloud client request to acquire schedule information from the data store. It then converts the schedule information into an XML-based document that is accepted by a cloud client. Additionally, The Transcoding Agent can call the Web API and SPARQL endpoint to access the Open Data and Linked Data.

GPS Network is composed of GPS satellite, GPS receiver, and Wi-Fi/GPRS base station. Mobile phone serves as a GPS receiver to receive location information form GPS stations.

The mobile phone is connected to the ULSS and the location information is sent through the Wi-Fi/GPRS network.

Cloud Client interacts with the Location-Based Service Platform through internet connections to retrieve the schedule information. Various client devices, including desktop PC, personal digital assistants (PDA), mobile phone, and notebook, are increasingly connected to the Internet. The same schedule information needs to be rendered differently on various client devices.

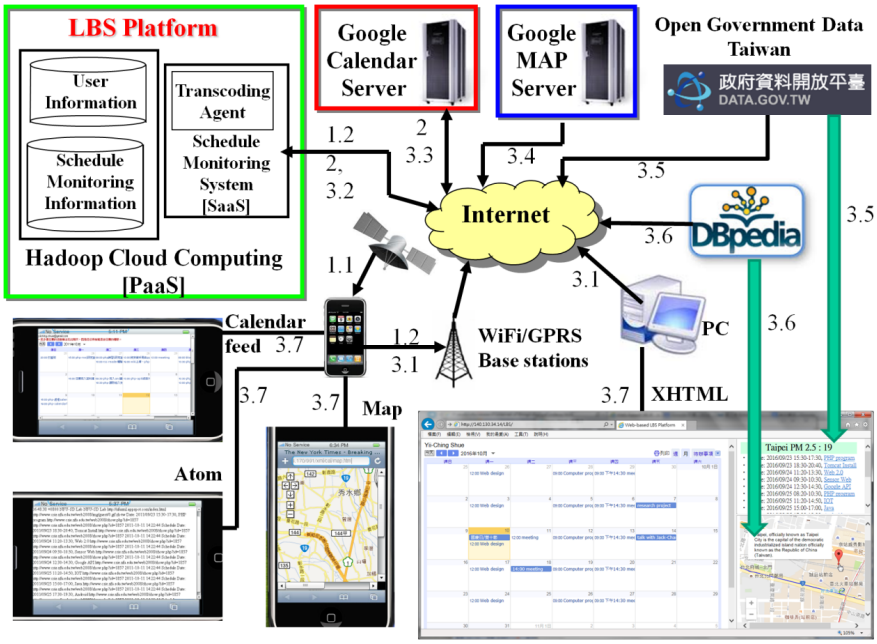

Figure 5: The dataflow-oriented ULSS architecture

The following steps explain the message flow illustrated in Fig. 5:

1. Each user has a personal mobile device associated with a unique user ID as the personal identification.

1.1 Mobile phone receives the location information form GPS satellites.

1.2 The location information is encoded to an XML-based document, as shown in Fig. 6, and then the XML-based document is imported into LBS Platform through the WiFi/GPRS base stations.

2. The LBS Platform filters the personal location information, and then calls Google Calendar API to save them into personal Google calendar.

3. The step is a pull-based interaction scheme that accomplishes the following tasks:

3.1 The various cloud devices, such as mobile phone or desktop PC, can send a request to LBS Platform with the user ID to browse the personal schedule information.

3.2 The LBS Platform invokes Transcoding Agent to acquire the schedule information form data store, and then converts this information into an XML-based schedule document.

3.3 The Transcoding Agent calls Google Calendar API to acquire the personal Google calendar.

3.4 The Transcoding Agent calls Google Map API to get Google Map information.
3.5 The Transcoding Agent accesses the PM2.5 Open Data through calls the Web API of Taiwan's Open Government Data. And then, it parses the PM2.5 Open Data to extract local PM2.5 information based on the current location of user.
3.6 The Transcoding Agent accesses the Taipei city Linked Data through calls the SPARQL endpoint of DBpedia. And then, it parses the RDF-based Linked Data to get the introduction of Taipei city.

The above information, including personal schedule, local PM2.5 information, Taipei city introduction, and Google map, will be converted to various XML-based documents, such as Atom (shown as Fig. 7) or XHTML document, to display in mobile phone and desktop PC, respectively.

```
<?xml version="1.0" encoding="utf-8" ?>
<schedule id="mu78919011>
<user id="wp321892">
<name> Yii-Ching Shue</name>
<title>GPS satellite positioning</title>
<item>
<subject> Day Trip</subject>
<time>Fri, 06 April 2018 16:12:16 GMT <time>
<device> iPhone (3289-329-7810A)</device>
<longitude>120.429481</longitude>
<latitude>23.702141</latitude>
…..
</item>
</employee>
</schedule>
```

Figure 6: Partial code of XML-based schedule information

```
<?xml version='1.0' encoding='UTF-8'?>
<feed xmlns='http://www.w3.org/2005/Atom'
    xmlns:openSearch='http://a9.com/-/spec/opensearchrss/1.0/'
    xmlns:gCal='http://schemas.google.com/gCal/2005'>
<id>http://www.google.com/calendar/feeds/yiiching.shue%40gmail.com/public/basic</id>
<updated>2018-04-06T14:50:18.000Z</updated>
<category scheme='http://schemas.google.com/g/2005#kind'
                term='http://schemas.google.com/g/2005#event'/>
<title type='text'>yiiching.shue@gmail.com</title>
…….
<entry>
<id> ….</id>
<published>2018-04-06T11:04:47.000Z</published>
<updated>2018-04-06T11:04:47.000Z</updated>
<category scheme='http://schemas.google.com/g/2005#kind'
            term='http://schemas.google.com/g/2005#event'/>
<title type='html'> GPS satellite positioning </title>
<suXMAry type='html'> Fri, 06 April 2018 16:12:16 GMT </suXMAry>
<content type='html'>
longitude :120.429481, latitude :23.702141
</content>
<link rel='self' type='application/atom+xml'  ….. />
<author>
<name>Yii-Ching Shue</name>
<email>yiiching.shue@gmail.com</email>
</author>
</entry>
</feed>
```

Figure 7: Partial code of Atom-based calendar feed

5.2 Evaluation

This section evaluates the Ubiquitous Location-based Service System (ULSS) for Web 2.0 Mashups as a Service (WMaaS) against our requirements. The requirements include heterogeneity and performance, which have been mentioned in Section 1.

5.2.1 Heterogeneous evaluation

The heterogeneous assessment contains Cloud Resources independence, Web Presentation independence, and Web Interaction independence. The study adopts WMaaS as a generalized architecture of cloud computing applications. Tab. 6 shows significant comparisons between the Web 2.0 technologies and WMaaS based on the heterogeneity. The independence of the platform and the hardware allows for a lightweight and simplified evolution of more complex web-based applications in the cloud computing ecosystem.

Furthermore, the constructs in the proposed WMaaS and ULSS are not specifically designed to match one particular cloud computing application. Therefore, they can support the heterogeneity to develop various Web 2.0-based cloud computing applications.

5.2.1 Representational state transfer

This section presents a preliminary experiment for evaluating the performance of the ULSS based on Hadoop cloud computing environment. This investigation employs an Hadoop Distributed File System (HDFS) as the file system, which can be set up to generate duplicates automatically, thus minimizing the risk of data loss. As shown in Fig. 8, a Hadoop/Spark cluster is composed of a master node and six data nodes. Each node consists of Intel core i7-8700 CPU, 32 GB memory, and 2 TB hard disk. HDFS is used for the cluster file system.

The Cluster Manager is the resource managers responsible for deploying the resources. Standalone resource manager is built in Spark and the developer can also choose other resource managers according to requirements. The study performs implementation and testing in three different cloud cluster computing environment, including Spark Standalone, Spark on YARN and Spark on Mesos.

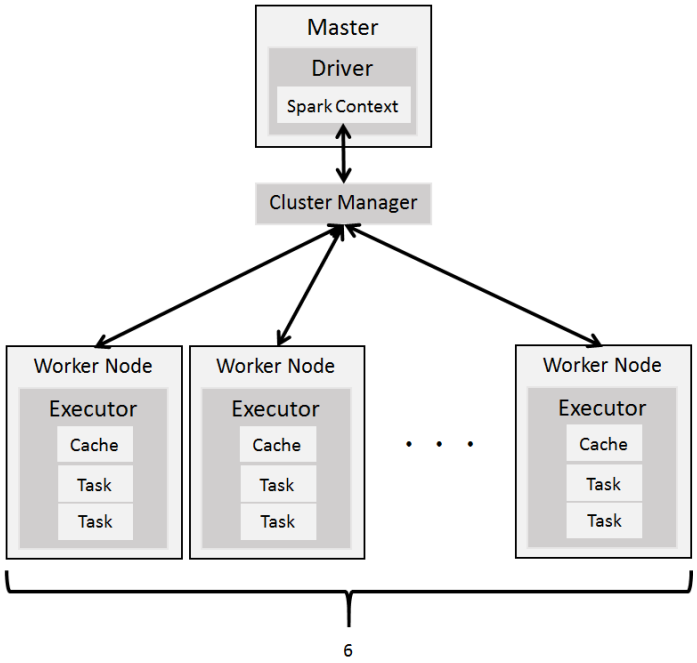

Figure 8: Hadoop/Spark cluster architecture

Performance evaluation of ULSS contains the Web-based information transcoding time, Google Calendar API execution time, Google Map API execution time, PM2.5 open data transcoding time, SPARQL endpoint execution time, and Internet transmission time. The same experiment will be executed on the Spark Standalone, Spark on YARN and Spark on Mesos, respectively. This experiment evaluated the ULSS as a personal schedule broker that processed data size from 1 GB to 7 GB. Fig. 9 shows the integrated test

results obtained from Spark on Standalone, YARN, and Mesos, which indicate that the ULSS under YARN mode works more efficiently than either Standalone or Mesos. This is mainly because that YARN can achieve better performance than Spark Standalone in resource scheduling, which provides different schedulers for selection, such as Capacity Scheduler and Fair Scheduler. Moreover, YARN is suitable for running with large numbers of nodes and highly complex data. Mesos is mainly responsible for providing proper resources for the assigned task, which will be used by the original application to run executor. Therefore, Mesos can be adopted to run multiple computing services. It can allocate proper sources dynamically through fine-grained mode, thus avoiding idle allocated resources. Notably, the threshold for Spark was about 5GB. When dataset size was lower than the threshold, the computing time is significant linear trend in the size of dataset. Conversely, when the dataset size increases exceeded this threshold, computing time increased very rapidly.

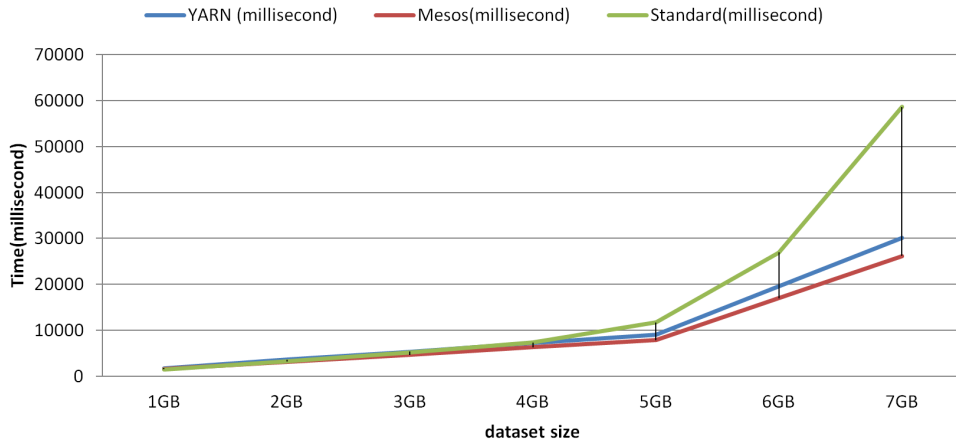

Figure 9: Performance comparison for Spark on YARN, Mesos, and Standard

6 Conclusion and future work

This study proposed a novel Web 2.0 Mashups as a Service, called WMaaS. The WMaaS is a fundamental cloud service model that is developed based a XML-based Mashups Architecture (XMA) to remove the heterogeneous issues of cloud computing. Additionally, WMaaS can also be associated with existing service models, SaaS, PaaS, and IaaS to facilitate organization monitoring and end user needs development. The main purpose of this study was to investigate how Web 2.0 Mashups technologies can be used to develop a novel service model of cloud computing ecosystem.

This study argues that Web 2.0 Mashups can be adopted as a common scheme to integrate cloud resources uniformly using a fundamental cloud service model. This study also demonstrates XML is a core technology of Web 2.0 Mashups. While, the main limitation of XML is that it emphasizes syntax and format rather than semantics and knowledge. XML provides an application-independent and syntactic structure for describing data and resource. Even though XML has the advantage of surface syntax for structured Web Data and Web APIs, it lacks the computer-interpretability to support

knowledge representation for organizational and end user computing applications development. One future work is to investigate how to integrate Semantic Web technologies [De Vocht, Softic, Verborgh et al. (2017); Selvan, Vairavasundaram and Ravi (2019)], such as RDF Schema, OWL and Ontology, into WMaaS to facilitate the development of intelligent cloud computing.

References

Barbaglia, G.; Murzilli, S.; Cudini, S. (2017): Definition of REST web services with JSON schema. *Software - Practice and Experience*, vol. 47, no. 6, pp. 907-920.

Berners-Lee, T. (2009): Linked data. http://www.w3.org/DesignIssues/LinkedData.html.

Bizer, C.; Universität, F.; Heath, T.; Berners-Lee, T. (2009): Linked data-the story so far. *International Journal on Semantic Web and Information Systems*, vol. 5, no. 3, pp. 1-22.

Borangiu, T.; Trentesaux, D.; Thomas, A.; Leitão, P.; Barata, J. (2019): Digital transformation of manufacturing through cloud services and resource virtualization. *Computers in Industry*, vol. 108, pp. 150-162.

Boulakbech, M.; Messai, N.; Sam, Y.; Devogele, T.; Etienne, L. (2016): SmartLoire: a web mashup based tool for personalized touristic plans construction. *25th IEEE International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises*.

Chakraborty, R.; Ramireddy, S.; Raghu, T. S.; Rao, H. R. (2010): The information assurance practices of cloud computing vendors. *IEEE IT Professional*, vol. 12, no. 4, pp. 9-37.

Chavarriaga, E.; Jurado, F.; Rubio, F. D. (2017): An approach to build XML-based domain specific languages solutions for client-side web applications. *Computer Languages, Systems and Structures*, vol. 49, pp. 133-151.

De Vocht, L.; Softic, S.; Verborgh, R.; Mannens, E.; Ebner, M. (2017): Social semantic search: a case study on web 2.0 for science. *International Journal on Semantic Web and Information Systems*, vol. 13, no. 4, pp. 155-180.

Deng, Z.; Ren, Y.; Liu, Y.; Yin, X.; Shen, Z. et al. (2019): Blockchain-based trusted electronic records preservation in cloud storage. *Computers, Materials and Continua*, vol. 58, no. 1, pp. 135-151.

Dillon, T.; Wu, C.; Chang, E. (2010): Cloud computing: issues and challenges. *24th IEEE International Conference on Advanced Information Networking and Applications*.

Fan, H.; Hussain, F. K.; Hussain, O. K. (2015): Semantic client-side approach for web personalization of SaaS-based cloud services. *Concurrency and Computation: Practice and Experience*, vol. 27, no. 8, pp. 2144-2169.

Feng, D.; Wu, Z.; Zhang, Z.; Fu, J. (2019): On the conceptualization of elastic service evaluation in cloud computing. *Journal of Information Technology Research*, vol. 12, no. 1, pp. 36-48.

Ghiani, G.; Paternò, F.; Spano, L. D.; Pintori, G. (2016): An environment for end-user development of web mashups. *International Journal of Human Computer Studies*, vol. 87, pp. 38-64

Herbold, S.; Hoffmann, A. (2017): Model-based testing as a service. *International Journal on Software Tools for Technology Transfer*, vol. 19, no. 3, pp. 271-279.

Hsu, I. C. (2013a): Multilayer context cloud framework for mobile Web 2.0: a proposed infrastructure. *International Journal of Communication Systems*, vol. 26, no. 5, pp. 610-625.

Hsu, I. C. (2013b): Personalized web feeds based on ontology technologies. *Information Systems Frontiers*, vol. 15, no. 3, pp. 465-479.

Iannone, A. E. (2019): Ballet education for the web 2.0 generation: a case for using youtube to teach elementary-school-aged ballet students. *International Journal of Technoethics*, vol. 10, no. 1, pp. 37-48.

Jeong, H. Y.; Rana, O. F.; Hsu, C. H.; Jeong, Y.-S. (2016): Cloud computing for mobile environments. *Concurrency and Computation: Practice and Experience*, vol. 28, no. 10, pp. 2753-2755.

Kobayashi, N.; Kume, S.; Lenz, K.; Masuya, H. (2018): RIKEN MetaDatabase: a database platform for health care and life sciences as a microcosm of linked open data cloud. *International Journal on Semantic Web and Information Systems*, vol. 14, no. 1, pp. 140-164.

Kryukov, A. P.; Demichev, A. P.; Polyakov, S. P. (2016): Web platforms for scientific research. *Programming and Computer Software*, vol. 42, no. 3, pp. 129-141.

Lee, Y.-J. (2015): Semantic-based web API composition for data mashups. *Journal of Information Science and Engineering*, vol. 31, no. 4, pp. 1233-1248.

Máchová, R.; Hub, M.; Lnenicka, M. (2018): Usability evaluation of open data portals: evaluating data discoverability, accessibility, and reusability from a stakeholders' perspective. *Aslib Journal of Information Management*, vol. 70, no. 3, pp. 252-268.

Marsh, J. (2001): XML base. http://www.w3.org/TR/xmlbase/.

Mital, M.; Pani, A. K.; Damodaran, S.; Ramesh, R. (2015): Cloud based management and control system for smart communities: a practical case study. *Computers in Industry*, vol. 74, pp. 162-172.

Mohamadi Bahram Abadi, R.; Rahmani, A. M.; Alizadeh, S. H. (2018): Server consolidation techniques in virtualized data centers of cloud environments: a systematic literature review. *Software - Practice and Experience*, vol. 48, no. 9, pp. 1688-1726.

Nascimento, F. R. A.; Da Rocha, J. C.; Garcia, A. C. B. (2018): Automated evaluation of open government data portals: a case study. *International Journal of Electronic Government Research*, vol. 14, no. 3, pp. 57-72.

NIST (2019): Cloud computing. https://www.nist.gov/itl/cloud-computing.

Nogales, A.; Sicilia, M.-A.; Sánchez-Alonso, S.; Garcia-Barriocanal, E. (2016): Linking from schema.org microdata to the web of linked data: an empirical assessment. *Computer Standards and Interfaces*, vol. 45, pp. 90-99.

O'Reilly, T. (2005): What is web 2.0.
http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html.

Saha, P.; Beltre, A.; Govindaraju, M. (2018): Exploring the fairness and resource distribution in an Apache Mesos environment. *11th IEEE International Conference on Cloud Computing*.

Saxena, S. (2019): Open Government Data (OGD) in Iran, Lebanon and Jordan: a comparative approach. *VINE Journal of Information and Knowledge Management Systems*, vol. 48, no. 1, pp. 47-61.

Sbodio, M. L.; Martin, D.; Moulin, C. (2010): Discovering semantic web services using SPARQL and intelligent agents. *Journal of Web Semantics*, vol. 8, no. 4, pp. 310-328.

Selvan, N. S.; Vairavasundaram, S.; Ravi, L. (2019): Fuzzy ontology-based personalized recommendation for internet of medical things with linked open data. *Journal of Intelligent and Fuzzy Sys*, vol. 36, no. 5, pp. 4065-4075.

Simeonova, B. (2018): Transactive memory systems and web 2.0 in knowledge sharing: a conceptual model based on activity theory and critical realism. *Information Systems Journal*, vol. 28, no. 4, pp. 592-611.

Sun, C. A.; Rossing, R.; Sinnema, M.; Bulanov, P.; Aiello, M. (2010): Modeling and managing the variability of web service-based systems. *Journal of Systems & Software*, vol. 83, no. 3, pp. 502-516.

Taiwan (2019): Open government data. http://data.gov.tw/.

Talukder, M. S.; Shen, L.; Hossain Talukder, M. F.; Bao, Y. (2019): Determinants of user acceptance and use of open government data (OGD): an empirical investigation in Bangladesh. *Technology in Society*, vol. 56, pp. 147-156.

Varlamov, M. I.; Turdakov, D. Y. (2016): A survey of methods for the extraction of information from web resources. *Programming and Computer Software*, vol. 42, no. 5, pp. 279-291.

von Alberti-Alhtaybat, L.; Al-Htaybat, K. (2016): Investor relations via web 2.0 social media channels: a qualitative study of middle eastern corporations and investors. *Aslib Journal of Information Management*, vol. 68, no. 1, pp. 33-56.

Vracic, T.; Varga, M.; Curko, K. (2016): Effects and evaluation of open government data initiative in Croatia. *39th International Convention on Information and Communication Technology, Electronics and Microelectronics*.

W3C (2017): Sparql endpoint description.
https://www.w3.org/wiki/SparqlEndpointDescription.

W3C (2013): SPARQL 1.1 query language. http://www.w3.org/TR/sparql11-query/.

W3C (2014): Resource Description Framework (RDF). https://www.w3.org/RDF/.

W3C (2016): SweoIG/TaskForces/CommunityProjects/LinkingOpenData.
https://www.w3.org/wiki/SweoIG/TaskForces/CommunityProjects/LinkingOpenData.

Wang, X.; Wu, H.; Hsu, C. H. (2019): Mashup-oriented API recommendation via random walk on knowledge graph. *IEEE Access*, vol. 7, pp. 7651-7662.

Wu, S.; Huang, C.; Li, L.; Crestani, F. (2019): Fusion-based methods for result diversification in web search. *Information Fusion*, vol. 45, pp. 16-26.

Yan, M.; Sun, H.; Liu, X.; Deng, T.; Wang, X. (2016): Delivering web service load testing as a service with a global cloud. *Concurrency and Computation: Practice and Experience*, vol. 27, no. 3, pp. 526-545.

Zhang, C.; Fu, W.; Sun, T.; Ji, Y. (2016): Resolving web services mismatch in mashup. *Wireless Personal Communications*, vol. 86, no. 4, pp. 1781-1796.

Zhong, Y.; Fan, Y.; Tan, W.; Zhang, J. (2018): Web service recommendation with reconstructed profile from mashup descriptions. *IEEE Transactions on Automation Science and Engineering*, vol. 15, no. 2, pp. 468-478.

Zhu, C.; Ren, K.; Liu, X.; Wang, H.; Tian, Y. et al. (2016): A graph traversal based approach to answer non-aggregation questions over DBpedia. *Lecture Notes in Computer Science*, vol. 9544, pp. 219-234.