future internet

MDPI

Article

Structured Data REST Protocol for End to End Data Mashup

Prakash Narayan Hardaha [1,*] and Shailendra Singh [2]

1 Barkatullah University Institute of Technology, Barkatullah University, Bhopal 462026, MP, India
2 Department of Computer Engineering & Applications, National Institute of Technical Teacher's Training and Research, Bhopal 462002, MP, India; ssingh@nitttrbpl.ac.in
* Correspondence: prakashnarayan007@gmail.com

Received: 1 August 2018; Accepted: 12 September 2018; Published: 4 October 2018

Abstract: Due to the exponential growth of the data and its services, visiting multiple webs/apps by a user raises three issues—(1) consumption of extra bytes; (2) time killing process of surfing inside the webs/apps; (3) tedious task of remembering address of webs/apps with their credentials. The data mashup is a set of techniques and user-friendly approaches which not only resolves above issues but also allows ordinary user to fetch required data from multiple disparate data sources and to create the integrated view in his defined digital place. In this paper, we have proposed an extension of existing REST protocol called Structured Data REST (SDRest) protocol and user-friendly novel approach which allows even ordinary users to develop end to end data mashup, using the innovative concept of Structured Data Mashup Box (SDMB) and One Time Configuration (OTC)-Any Time Access (ATA) models. Our implementation shows that pre-mashup configuration can easily be performed by an ordinary user and an integrated user interface view of end user data mashup can be created without any technical knowledge or programming. We have also evaluated the proposed work by comparing it with some of the related works and found that the proposed work has developed user friendly configurable approach using the current state of the art techniques to involve not only the ordinary user but also the mashup service provider and the data service provider to develop public, private and hybrid data mashup.

Keywords: structured data; REST protocol; structured data mashup box; one-time configuration model; any time access model; end to end data mashup; ordinary user; mashup service provider; data service provider

1. Introduction

In the current state of technology, not only the data and its services but also their users are increasing day by day. A user is required to visit multiple webs/apps using some credentials to access the personal/private information. In order to access public information, a user searches the appropriate webs/apps and explores them to reach to the information as per their need. It is obvious that instead of searching and visiting the multiple webs/apps, the users would like to view all their needed information at one digital place, which could be his personal web page, email account, social media page, desktop application or their mobile screen, etc. The data mashup is a set of techniques and user-friendly approaches to allow ordinary users to fetch the required data from multiple data sources and facilitates them to mashup the data and view them together in a single digital place. The industries and researchers are continuously working on the development of user-friendly approach for performing the data mashup after fetching the hybrid kind of data from the disparate data sources even by ordinary users. Mashup users can be divided into two types of groups namely End User Developer (EUD)/End User Programmer (EUP) [1,2] and Ordinary User

(OU) [3]/naive End User (EU) who are supposed to use mashup applications according to their skills. The end user developer possesses some technical skills and can use mashup tools and techniques to develop mashup application for himself but the ordinary user does not have any technical skill and can use only those mashup tools and techniques which expect simple skills like internet browsing, mouse click, key press or filling the web forms, etc.

Most of the mashup tools expect some technical skills from users and are suitable for end user developers only. Initially, the mashup development was started to fulfill the need of ordinary users but later on, the development of mashup was diverted to make the mashup tools targeted for end user developer. The proposed work targets to ordinary user and we will use the term 'End User (EU)' or 'user' for ordinary user who does not have any technical skill. Other than end user, there are two other major stakeholders called Mashup Service Provider (MSP) and Data Service Provider (DSP) involved in mashup development [4]. The details about MSP and DSP and their roles have been explained later in this paper. Generally, data mashup performed by end user includes seven basic steps—(1) determining its own digital place; (2) defining requirement of the data to be mashed up in a data mashup box; (3) searching appropriate MSPs/DSPs for mashup services and/or data services; (4) configuring each mashup box for mashing up data from relevant services; (5) fetching required data from multiple MSPs/DSPs; (6) mashing up the data, received from MSPs/DSPs into mashup box and injecting them into pre-determined/customized view; (7) showing the integrated view of mashed up data to the end user. Unlike other research works on data mashup, this paper covers all the steps required in the data mashup development as mentioned above.

The proposed work is based on the standard REpresentational State Transfer (REST) protocol [5] to perform end to end data mashup which includes all stakeholders (EU, MSP, DSP) for performing data mashup using mashup communication between any two stakeholders (EU and MSP, EU and DSP, MSP and MSP or MSP and DSP). Data mashup techniques and approaches have not become as popular as email till now because of the less utilization of the standard communication protocols for mashup communication and the lack of generalized user-friendly approaches. Any mashup tool and technique cannot become popular among ordinary users until it provides user-friendly approach, which does not expect any technical skills from users and uses the general but secured communication protocol for performing the data mashup.

This paper focuses on SDRest protocol to develop user-friendly mashup approach, which allows ordinary users to define requirement of the data to be mashed up, fetch and integrate the needed Structured Data (SD) from multiple data sources at their own defined digital place without any programming and facilitate them to view the integrated UI in a single screen.

The rest of the paper is organized as follows. The Section 2 covers the related literature review and the major challenges. The Section 3 clearly mentions the motivation behind the proposed work and sets the goal. The Section 4 describes our contributions in brief. Section 5 explains the model and architecture of the proposed Structured Data Rest Protocol (SDRest) and various algorithms required in different stages of the data mashup development. The Section 6 explains the algorithms used by end user and mashup service provider for performing data mashup. The Section 7 shows the implementation of the proposed work by developing the data mashup service network for three services which explores the user-friendly approach for performing the pre-mashup configuration and end user data mashup by an ordinary user. The Section 8 evaluates the proposed work by comparing it with some other related works based on some important parameters needed to evaluate each step of the mashup development and at last the conclusion and the future work.

2. Related Work and Challenges

The data mashup is useful in terms of the direct involvement of ordinary users for developing the information system. It will not only increase the user's satisfaction but also will reduce the network traffic and overcome so many challenges of IT system developers. Many researchers and market players developed many mashup tools and techniques based on various approaches and also got

success up to certain level but the actual aim for developing mashup by ordinary user without any programming/scripting in user friendly manner is still ahead. According to [6,7] widget approaches generally become popular among ordinary users because these include drag and drop like feature to mashup the desired data. In this approach, EU selects a widget, drops it onto a canvas, customizes the widget, and species how to connect a widget to other widgets by creating a connected graph [8]. Reference [9] proposed a mashup model that enables the integration at the presentation layer of "actionable UI components" which are equipped with both data visualization templates and a proper logic consisting of functions to manipulate the visualized data. References [1,8,10] discussed mashup technologies of the market players which include Google Mashup Editor, IBM Mashup Center, Intel MashMaker, Microsoft Popfly, Yahoo Pipes etc. Google Mashup editor [8] was the simple graphical editor to create AJAX based application with no plug-ins required and used to pull data from RSS feeds and Atoms. IBM Mashup Center [8] allows users to create widgets and to wire them to assemble pages to create mashup. Intel® Mash Maker [11] uses browser plug-ins which allows the user to see information from other websites in a single page. It learns new mashups by simply copy and paste method which can further be used to suggest other users. Yahoo! Pipes [10,12] allows to mix popular data feeds using a visual editor to create data mashup. Yahoo pipe consists of various data sources like RSS, Atom, XML etc. and defines the set of operations to perform some task. It was providing visual tools to connect one widget to other widget in such a way that output of one widget was input to other and so on. Three problems as identified in widget approaches include locating the appropriate widget, customization of some widgets expects knowledge of programming and most of the mashup tools are not complete in all respect to perform independent mashup. FlexMash [13] is an approach and tool implemented for flexible data mashups based on pattern-based model transformation and subdivided the data mashup into modeling, transformation, execution and presentation level.

Another approach is the End User Programming (EUP) [14,15] approach which allows end users to write and/or edit various script codes. Some tools provide the visual editor to copy and paste the script to integrate the data from various data resources but simply copy and paste of the script is inadequate because it requires re-scripting, modification and debugging as per situational need. Looking to the need of the ordinary user, programming by example/demonstration approach [16] was designed which allows them to just perform simple drag and drop operations with instant output as an example so as to teach the mashup framework to do the remaining task automatically. According to this, five steps are required to perform mashup i.e., data retrieving (data extraction from web sites), source modeling (mapping between source data and destination data model), data cleaning (dealing with data formats, spelling etc.), data integration (combining the data from multiple sources) and data visualization (integrated UI). They used the "drag and drop" approach from a web page to extract web data based on Document Object Model (DOM) by allowing ordinary user to try with an example. It was really simple approach for normal user but it has limitations like extracting data from website, time consuming process of data cleaning and its limitation on data sources (suitable for a well-structured web page only).

Web data extraction [17] has been the focus of researchers, but still remains challenging because a web page may contain the data in structured, semi-structured or unstructured format. Reference [18] explained extracting Web API Specifications from HTML documentation but due to uncertainty on structure of the web page, the approach of data extraction from web page can never be reliable. Another problem with web data extraction is to extract data of deep web pages because a deep page is generated when some inputs are given through some web form. Reference [19] present discovery algorithms which can generate optimal plans by applying strategies and can play a critical role in conducting further API composition which follows a graph-based approach. Reference [20] presented the Linked Web APIs dataset which supports API consumers and API Provider in the process of discovery, selection and use of Web APIs. Spread sheet approach [21,22] is another UI approach which aims to provide a spreadsheet-like view to mash up the data and is becoming popular because most of the ordinary users are familiar with spread sheet. The RESTful architectural model is helpful for widget

development for mashup purpose because of its nature of portability, scalability and multi-platform support [5]. Reference [23] presented process data widget approach to assist designers by configuring RESTful services but this is also not suitable for ordinary user because ordinary user can not define his requirement in terms of web services. Service oriented approaches [24,25] involve selection and composition of various services made available by MSP/DSP to the end user. Because of increasing popularity and demands of cloud computing, cloud mashup service approaches [24] are also being experimented to promote Mashup as a Service (MaaS)/Data as a Service (DaaS). In database driven approach, reference [26] proposed the service data model for adaptation of heterogeneous web services, the service relation model for representation and refinement of data interaction between services, and the service process graph for describing business logics of mashup applications. Another approach is the linked data mashup approach [27,28] which uses semantic web technology based on linked data for combining, aggregating, and transforming data from heterogeneous data resources to build linked data mashups.

The followings are the major challenges found during study on the data mashup.

2.1. Involvement of Stakeholders

Ordinary users are the actual targeted users of the mashup hence their involvement cannot be avoided. In mashup applications, users are given freedom to fetch data from multiple data sources and to perform the data mashup at one digital place, most preferably their personal page/desktop or mobile application. In order to provide the data to its users, data sources should have service interfaces and user-friendly access mechanism so that even an ordinary user can use it. Generally, MSP does not own the user's data but provides mashup services or tools to its users to mashup the data at their end. Based on various skills, reference [21] divided mashup users into three categories namely developer, power user and casual user. The developer should be familiar with web technologies and programming, power user has no programming skills but has functional knowledge of specific tools but casual user or ordinary user should have the skills to use the web browser only. Our work is completely concerned with casual user or ordinary user. Reference [29] discussed the mashup ecosystems consisting of mashup authors, service providers and mashup users. Thus, mashup has three major stakeholders called Mashup Service Provider (MSP), Data Service Provider (DSP) and Ordinary User (OU). While performing the mashup, the ordinary users generally face problems like high learning curve of the tools, expectation of some technical skills from end user, dependency on mashup developers for processing of mashup and approaches, the reliability of services and lack of user-friendly mechanism of searching the correct data sources, etc. Involvement of MSPs as intermediate between actual end users and DSPs always raise issues such as disclosure of private data and their security, which have been mentioned in Section 2.3. Involvement of end users and DSPs are compulsory in data mashup but the involvement of MSP is optional.

2.2. Searching the Right MSP/DSP

Data services as well as mashup services, are increasing day by day and their users too. Due to an exponential growth of the services, searching the right MSP/DSP is becoming the challenging task for users. Each MSP/DSP publishes one or more services and thus searching the appropriate service of that MSP/DSP again is tedious and time consuming. Reference [3] designed and implemented lightweight services mashup platform for service creations by ordinary user without expecting specific computing knowledge. Reference [30] recommended the framework for data service mashup, based on several mashup patterns and the corresponding recommendation method. Reference [31] proposed a novel framework for service discovery which exploits social media information and different methods to measure social factors with weight learning algorithm to learn an optimal combination of the measured social factors. Reference [32] proposed a novel category-aware service clustering and distributed recommending method for automatic mashup creation and developed a category-aware distributed service recommendation model, which is based on a distributed machine learning framework.

Reference [33] explored API recommendation for mashups in the reusable composition context with the goal of helping developers to identify the most appropriate APIs for their composition tasks and proposed a probabilistic matrix factorization approach to solve the recommendation problem and enhance the recommendation diversity. Reference [24] suggested the MapReduce in skyline query processing for optimizing composite web services in large scale cloud mashup applications and proposed a block-elimination-based partitioning approach to shorten the process. Reference [34] proposed a service-oriented approach to generate and manage mashups and also developed the mashup services system to support users to create, use, and manage mashups with little or no programming effort. All these approaches were developed for service discovery for creating the mashup but cover only one aspect of the whole life cycle of the mashup development and suitable for MSP/EUD but not for ordinary users because of some learning expectation and the complexity. Searching the mashup or data services can be automatic, manual or semi-automatic. The method for searching of the MSP/DSP and then selecting appropriate service is assumed to be manual in the current work. There is no need of searching the DSP when a user would like to mashup his own private data because its source is already known to him but it still requires selection of the right data service among all the services available at DSP. In case of public data mashup, users are required to search the right MSP/DSP and select the appropriate service thereafter.

2.3. Data Privacy and Security

The data mashup involves ordinary user as well as mashup service providers hence data privacy and security has become the major concern of the researchers and the industries. Fung et al. [35] proposed service-oriented architecture to resolve privacy problem in real life mashup applications. According to it, there is always chance of revealing sensitive data of a user when third party MSP is involved in data mashup. According to [36], REST is an abstract model for designing large-scale distributed systems and can be adopted with suitable technologies of any kind, such as HTTP, CoAP, or RACS, to build highly scalable service systems such as the web, IoT, SOA, or cloud applications.

Reference [37] developed an ID-based authentication algorithm to achieve a secure RESTful web service using Boneh–Franklin ID-based encryption and REST URI which enables server to handle client's request by acknowledging client's URI rather than storing client's entire status for stateless REST. Reference [38] proposed an extended UsernameToken-based approach for REST-style web service by adding UsernameToken and secondary password into the HTTP header which makes current security aspect of REST-style web services more secured. According to [39], instead of choosing SOAP, the service providers nowadays are shifting to REST-based services but the same time it is vulnerable to security. In our work, we have used the concept of transactions of mashup keys and uniquely identified private Mashup Configuration Service Identifier (MCSI) and private Mashup Data Service Identifier (MDSI) for private data mashup to make the mashup communication secured (See Section 5.10).

2.4. Data Refreshing

Data mashup not only fetches the data from multiple data sources but also facilitate the end user to integrate and view mashed up data in a single screen. Data to be mashed up are fetched instantly by end users from MSP and DSP to get refreshed data, every time they visit their own mashed up page/screen. According to [40], the refresh rate of the data made available to the user, depends on pull or push strategies used for data mashup. Based on the pull/push strategies of request-response pattern [41,42], the data mashup can be divided into two types, i.e., Pull Data Mashup and Push Data Mashup. In pull data mashup, the data are fetched by end user on each and every request sent to MSP/DSP hence refresh rate of mashed up data is high and it may also be called live data mashup. However, in push data mashup, data are sent by MSP/DSP to end user, whenever there are some updates on it. The refresh rate of push data mashup is not as high as pull data mashup but it consumes fewer bytes of the network traffic as compared to pull data mashup. The data mashup approach used

in the proposed work is based on pull data mashup which is highly recommended for public data mashup but it can also be implemented for push data mashup. The public data mashup and private data mashup are two different types of data mashup which can be discriminated by their nature of data accessibility.

2.5. Data Mapping

Data mapping is the process needed to identify the correspondences between the elements of the source data model and the internal data model [43]. Reference [44] discussed general data mapping problem, which addresses two related data exchange scenarios. Like other steps of mashup development, the data mapping can also be manual, semi-automatic or automatic. Reference [45] presented the schema, data and query mapping algorithms for storing xml into relational database. Reference [46] proposed method to solve schema mapping and data mapping both using mutual enhancement mechanism and also described how prior knowledge of the schema mapping reduce the complexity for the comparison between two data attributes. In data mashup, every stakeholder (OU/MSP/DSP) should be independent to develop its own internal data schema and hence data mapping has become the most important task while developing the data mashup. There is a need of strategies to specify the correspondences between their internal data model and the desired data sources [40]. Each stakeholder defines not only the different attributes but also different data types and their formats independently which makes it more challenging while data mapping. In our proposed work, SDXMapping algorithm (See Section 5.9) has been successfully developed for performing data mapping between the data models of any two stakeholders of data mashup. SDXMapping works as a bridge between them and allows to create data mapping based on the semantical meaning of the attributes.

2.6. Standard Communication Protocol

Mashup communication is an important process to complete the life cycle of the data mashup development. Data mashup techniques cannot be generalized and secured until the standard communication protocol is commonly used. The proposed work uses the standard REST protocol for communication purpose but other standard protocols like SMTP, HTTP etc. can also be used for data mashup. References [47–49] discussed the use of REST/web services for data fetching, integration and composition of data mashup and services. According to [47], REST has features of light weight, scalability, multiple data format support, superior performance and popularity which have made it a better approach for web service composition and communication. Reference [48] focused on the need of service composition using REST and discussed six composition issues like coordination, transaction, context, conversation modeling, execution monitoring and infrastructure. Reference [49] presented overview of the life cycle of web services composition and also discussed standards, research prototypes, and platforms with several research opportunities and challenges for web services composition.

2.7. Data Accessibility

Data accessibility has been the major concern for all its stakeholders. MSPs/DSPs publish their services for accessibility of public data and private data separately. Data resides in multiple and heterogeneous data sources are accessed using web APIs, REST, SOAP, HTTP and XML RPC techniques, RSS Feeds, Atom Feeds, SOAP Services, Restful Web Services, JSON, CSV, Web Page and Annotated Web Pages etc. [1,8]. In our proposed work, the Mashup Configuration Service Identifier (MCSI) and Mashup Data Service Identifier (MDSI) (See Sections 5.1, 5.2 and 5.10) have to be accessed for pre-mashup configuration before performing the actual data mashup. In private data mashup, MCSI and MDSI are accessible to only authorized users, which make it more secured.

3. Motivational Scenario

When users are given facility of data mashup at their defined digital place, they will definitely avoid visits of multiple webs/apps and thus network traffic would be reduced up to the certain level. Data mashup fetches the required data from multiple data sources, does the mashup and presents the integrated view to its users. For example, in a public data mashup, a student would like to see all the admission notices of his interested universities/institutes in his personal mashed up page. In case of private data mashup, he would like to see the transaction details of all his bank accounts, review status of his research papers submitted to various journals and results of his all interviews of various recruiters together in a single screen using Single Sign-on (SSO) policy. In such cases, instead of information like various codes/scripts, advertisement, navigation links, menus, images etc., only useful data would be flown in the network.

Above discussion for performing the public and private data mashup together and presenting their views in a single screen, related to particular user is the main motivation behind this work. The goals of this work are summarized as follows:

- To provide the public and private both kind of information related to particular user which are scattered throughout the internet, to his/her defined digital location (web/desktop or mobile)
- To implement the single sign-on (SSO) scheme for private data mashup
- To enhance the network performance by allowing the flow of needed data only
- To develop the data mashup technique having user friendly approach for end to end data mashup using standard REST protocol for mashup communication.

4. Our Contributions

As far as the authors know, this work is the first effort which has introduced the innovative concept of Structured Data REST (SDRest) protocol which uses existing REpresentational State Transfer (REST) protocol for end to end data mashup using the concept of Structured Data Mashup Box (SDMB) (See Section 5.4) and SDX Mapping (See Section 5.9). REST protocol has now become the general-purpose protocol for communication between the web client and the server and has been effectively explored in this work to develop the user-friendly approach to perform data mashup by an ordinary user. REST is universally accepted and hence its implementation for public and private data communication for data mashup would not only be effective but may also be generalized for data mashup. Our contributions in this paper are summarized as follows.

4.1. Contribution#1

Involvement of all the stakeholders and the development of hybrid data mashup are two most important achievements of the proposed work. The data mashup can be classified as direct data mashup, intermediate data mashup and hybrid data mashup, based on mashup communication among stakeholders for performing data mashup. In the direct data mashup, end user directly communicates with DSPs, which are the actual data sources and highly recommended for private data mashup. In intermediate data mashup, MSPs are placed in between end user and the DSPs and is used for public data mashup but may cause of security breach if it is used for private data mashup. We have developed user friendly approach to develop hybrid data mashup in such a way that an ordinary user can perform mashup to get the private data directly from the original data source without any intervention of MSP and can perform public data mashup through MSP/DSP. The proposed work recommends the standard REST protocol for communication between various stakeholders for both the public and private data mashup. Hybrid data mashup shown in Figure 1 is most suitable for EU because it not only involves MSPs for public data mashup but also allows mashing up the private data directly from DSPs. In Figure 1, EU#01 is connected with MSP#01 and DSP#03 directly. MSP#01 is communicating with DSP#01 and DSP#02 for providing mashed up data to EU#01.

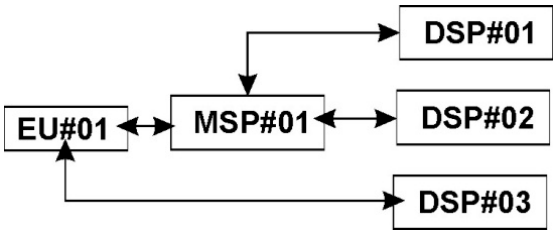


Figure 1. Hybrid Data Mashup.

4.2. Contribution#2

Data mashup includes the complete life cycle of application development but most of the research works cover only one or two aspects of mashup development, not the whole life cycle steps. The proposed work has been designed in such way, which includes the complete cycle of mashup development from the requirement definition to the generation of integrated user interface view by ordinary user, mashup service provider and data service provider. (See Steps 1 to 8 in the Section 5.2).

4.3. Contribution#3

We have introduced the novel protocol called Structured Data REST (SDRest) protocol with an innovative concept of Structured Data Mashup Box (SDMB) which would be used by mashup stakeholders to store and/or forward the mashed-up data after applying filter and transformation algorithm. SDMB is defined by the schema called Data Mashup Definition (DMD) (See Section 5.3) and used to store the relevant data as per requirement specified by Data Service Consumer (DSC) (See Section 5.1). The main contribution of proposed model is that it promotes the data mashup as a structured data communication which can easily be used by an ordinary user to fetch data from multiple data sources without any technical knowledge.

4.4. Contribution#4

We have proposed SDRest system architecture based on the model called OTC-ATA i.e., One Time Configuration-Any Time Access model (See Section 5.2). The one-time configuration is performed by data service consumer once only but ATA model can be used by data service consumer at any time to access required data from MSP/DSP. OTC model uses the Mashup Configuration Service Identifier (MCSI) for getting the attributes for configuration of structured data mashup box and ATA model uses the Mashup Data Service Identifier (MDSI) to access the data as per mashup request. OTC is the pre-mashup configuration process and ATA is post mashup operation performed by stakeholders. We have implemented the REST protocol for all types of communication (public, private and hybrid) between the stakeholders of the data mashup using both mashup strategies i.e., pull and push. Thus, the proposed work covers all ranges of data mashup i.e., public, private, hybrid, and push and pull data mashup as compared to previous works.

4.5. Contribution#5

Data mapping is the most important contribution of this paper. We developed the concept of the configuration of data service consumer using MCAttributes (See Section 5.8) received through mashup configuration service identifier to generate Structured Data eXchange Mapping (SDXMapping) which is used further for ATA communication with MSP/DSP for performing data mashup.

4.6. Contribution#6

Mashup Data Service (MDS) which may also be called Structured Data as a Service (SDaaS) is identified by mashup data service identifier (MDSI) and provides data to data service consumer on request. Public MDSI and private MDSI both can be implemented using REST protocol for public

data mashup and private data mashup respectively. It is also possible to discriminate the MDSI for filtering the data service consumer's request for providing the different data as per the need of the data consumer. Structured Data Module (SDM) (See Section 5.13) has been implemented using JSON format which makes the whole data mashup process easy and generalized for data communication using REST.

The details of the proposed work have been described in the following sections. The sample data like URLs, emails etc. used in this paper are intended for example purpose only and these have nothing to do with the real-world entities.

5. Structured Data Rest Protocol

This section describes model and architecture of the proposed protocol along with algorithms required to develop end to end data mashup.

5.1. The Model of Proposed SDRest Protocol

The model of the proposed protocol called Structured Data REST Protocol (SDRest) has been shown in Figure 2 which is the extension of existing REpresentational State Transfer [5] (REST) protocol for end to end data mashup. Here, the term "end to end data mashup" implies performing the data mashup directly between any two stakeholders (EU, MSP and DSP).

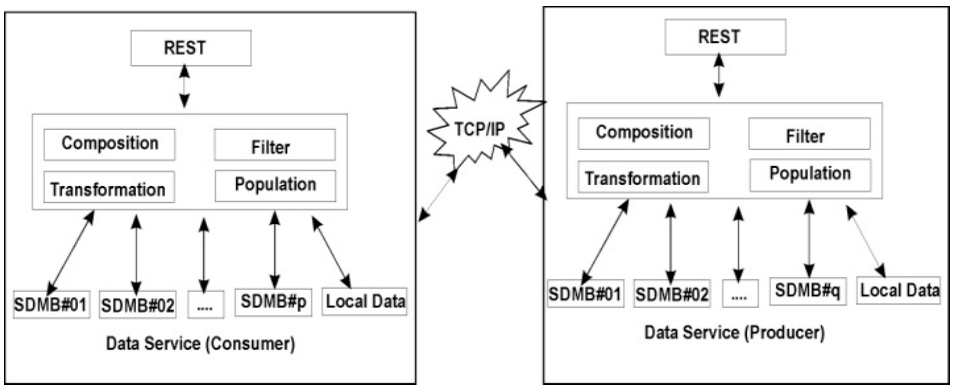


Figure 2. Model of Structured Data REpresentational State Transfer (REST) (SDRest) Protocol.

The SDRest protocol has two entities called Data Service Producer/Provider (DSP) and Data Service Consumer (DSC). DSP will produce structured data which would be consumed by DSC using the proposed SDRest protocol. In our proposed work, the entity which takes services of MSP or DSP would be called DSC and entity which provides data services to DSC would be called DSP. Thus, end user or MSP may play the role of DSC and MSP/DSP may play the role of DSP. In Figure 1, EU#01 is DSC for MSP#01 and DSP#03 both and MSP#01 is further DSC for DSP#01 and DSP#02. The DSP is required to publish the data services through Mashup Data Service Identifier (MDSI) so that DSC can access the data in the structured format using standard REST protocol. Mashup Data Service Identifier (MDSI) is the unique REST URI used by DSC to fetch the required data from DSP. The major contribution in this model is introducing the concept of one or more Structured Data Mashup Box (SDMB) which would be used by Data Service Consumer (DSC) to view, store or forward required mashed up data.

The data service consumer as end user will use Structured Data Mashup Box (SDMB) to store and view the required mashed up data whereas data service consumer as MSP will perform mashup on the data received from various DSPs/MSPs and forward it to respective data service consumer. Further, the group of SDMBs called End User Data Mashup (EUDM) would be explored by end user to see the integrated UI view of mashed up data (fetched from multiple MSPs/DSPs) in a single screen which is the main objective of the proposed work. According to our proposed model, DSC (the REST

client) will read all the Structured Data Records (SDRs) made available by DSP through Mashup Data Service Identifier (MDSI) and filter component will parse each SDR to check whether it is a valid SDR or not. Transformation component will transform each valid SDR into proper format so that it can be populated into Structured Data Mashup Box. The structured data records, which are successfully populated into SDMB would be called Mashed up Data Records (MDRs).

Valid SDR would be transformed and populated into appropriate SDMB at DSC's end using proposed algorithm (See Section 5.15) whereas invalid SDRs would be ignored. Cloud computing paradigm called Structured Data as Service (SDaaS) or Mashup Data Service (MDS) at MSP's/DSP's end will generate (compose) one or more structured data records and publish them so that they can be accessed by DSCs for data mashup. Each publication of MSP/DSP would be identified by mashup data service identifier. It can be seen from Figure 2 that there are p SDMBs at DSC's end and q SDMBs at DSP's end. Here, p and q may or may not be equal to each other and SDMB at DSP's end is optional but SDMB at DSC's end is compulsory to implement this protocol. Following section describes the system architecture and the working of this model in detail.

5.2. SDRest System Architecture

The system architecture of the proposed SDRest protocol is based on One Time Configuration-Any Time Access (OTC-ATA) model. The SDRest system architectures based on OTC and ATA models are shown in Figures 3 and 4 respectively.

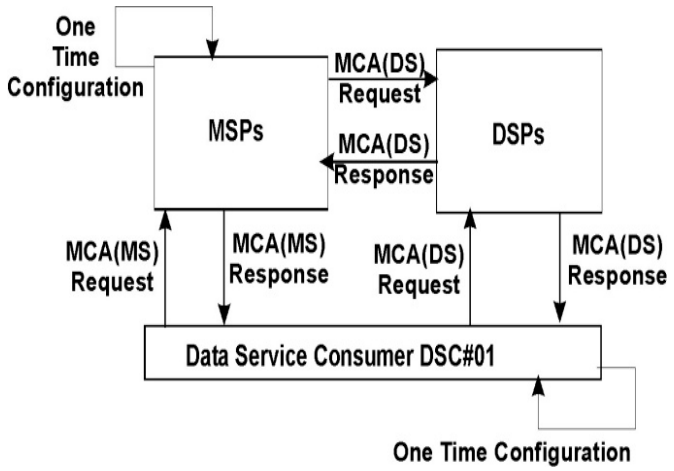


Figure 3. SDRest System Architecture for One Time Configuration (OTC).

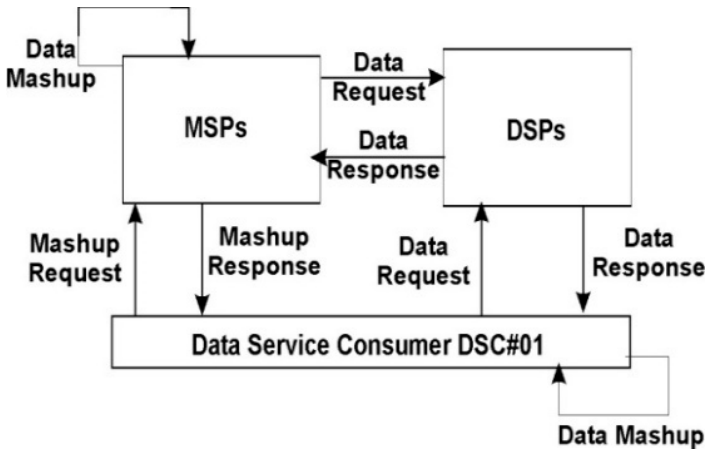


Figure 4. SDRest System Architecture for Any Time Access (ATA).

In OTC model, every user who needs mashed up data at his defined digital address (web page/desktop client etc.) searches for appropriate MSP/DSP to collect Mashup Configuration Attributes (MCA) made available through Mashup Configuration Service Identifier (MCSI). MCSI is the unique identification of REST URI available at MSP/DSP which provides mashup configuration attributes on DSC's request. Mashup Configuration Attributes (MCA) is the set of attributes required to perform one-time configuration of DSC at its end so that later on, data can be fetched from MSP/DSP for the purpose of data mashup. DSC collects mashup configuration attributes of Mashup Services (MS) from MSP and collects mashup configuration attributes of Data Services (DS) from DSP. Accessing the mashup configuration service identifier by DSC can be manual or automatic. In manual approach, the user needs to visit each and every mashup service configuration identifier as per his requirement and use the MCA (Mashup Configuration Attributes) once only to create Structured Data eXchange (SDX) mapping (See Sections 5.7–5.9).

In automatic approach, DSC calls mashup configuration service identifier i.e., REST URI, which returns MCA as a response. This mashup configuration attributes should be in well-structured data format i.e., JSON, XML, CSV etc. The SDXMapping algorithm developed for DSC helps the user to create Structured Data eXchange (SDX). The processing of the mashup configuration attributes to create Structured Data eXchange (SDX) is performed by DSC once only and later on, used to access MSP's/DSP's data for performing data mashup. It is clear from Figure 3 that DSC#01 requests for MCA(MS) from MSP and MCA(DS) from DSP which return mashup configuration attributes as a response for performing one time configuration which resulted into creation of SDX Mapping so that mashup can be performed any time by DSC#01. All MSPs are also required to perform one time configuration for each and every DSP as per their service needs before providing mashup services to its users. After performing OTC for each and every MSP/DSP, data service consumer is ready to perform data mashup by sending data request any time to MSPs/DSPs through mashup data service identifier.

It can be seen from Figure 4 that DSC#01 sends mashup request to MSPs and data request to DSPs and performs data mashup after getting mashup response and data response from them. All MSPs also send the data request to DSPs and perform data mashup on receiving data response from them. When a DSC sends mashup request to MSP then MSP further sends the data request to respective DSPs and resultant data response received by MSP are mashed up and are sent back to DSC as a mashup response. Let us understand how major stakeholders (EU, MSP and DSP) of data mashup, communicate with each other.

The diagram as shown in Figure 5 has three types of stakeholders i.e., end users, MSPs and DSPs. Each user will define his requirement of data mashup by creating one or more SDMBs as per his data need. Similarly, each MSP will also create one or more SDMBs as per the service need. DSPs need not have SDMBs but should have Mashup Data Service Identifier (MDSI) to provide structured data to its clients. It is clear from Figure 5 that the user can access mashed up data from MSPs and MSPs can further communicate with other MSPs/DSPs to access required data. The user can also directly communicate with DSPs to fetch the required data and then perform mashup at his end.

Table 1 depicts the role of DSC, MSP and DSP for various mashup communications as shown in Figure 5. SDMB#1 of EU#1 is taking mashup services of MSP#1(SDMB#2). MSP#1(SDMB#2) is further taking mashup services of MSP#2(SDMB#3) and also taking the data services of DSP#1, DSP#2 and DSP#3. MSP#2(SDMB#3) is not connected with any mashup service but taking data services of DSP#k and DSP#r. SDMB#2 of EU#1 is connected with MSP#2(SDMB#3) and directly to DSP#4. All the stakeholders are free to connect to any MSP or DSP but there could be the problem of multi-path data mashup resulting into redundant data mashup if services are not carefully chosen. The minimum requirement of this architecture is that each user should have SDMB, OTC and REST enabled machine to communicate with the REST services of DSPs/MSPs.

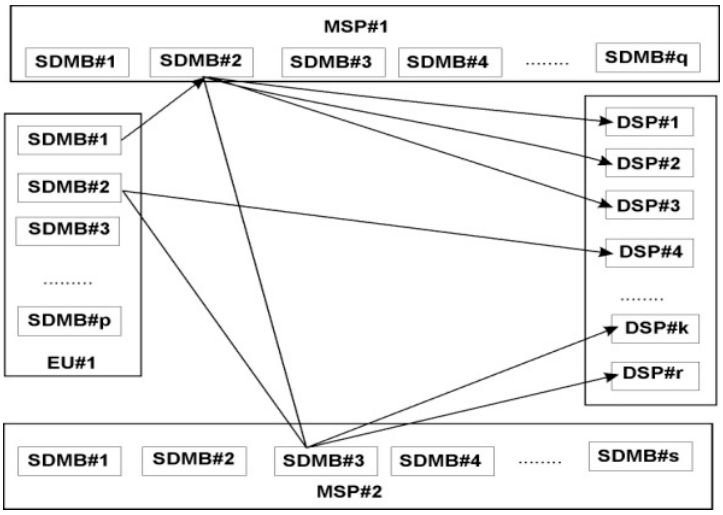


Figure 5. Stakeholder's Communication Model.

Table 1. Roles of Stakeholders.

DSC	MSP	DSP
EU#1(SDMB#1)	MSP#1(SDMB#2)	-
MSP#1(SDMB#2)	MSP#2(SDMB#3)	DSP#1 DSP#2 DSP#3
MSP#2(SDMB#3)	-	DSP#k DSP#r
EU#1(SDMB#2)	MSP#2(SDMB#3)	DSP#4

One Time Configuration-Any Time Access Model

OTC-ATA model as shown in Figure 6 follows OTC-ATA communication between Data Service Consumer (DSC) and MSP/DSP. In order to perform complete life cycle of data mashup, DSC first sends One Time Configuration (OTC) request to Mashup Configuration Service (MCS) available at MSP/DSP to get mashup configuration attributes before fetching required data for the mashup. After the completion of OTC, DSC can access mashed up data at any time by calling Mashup Data Service (MDS) available at respective MSP/DSP.

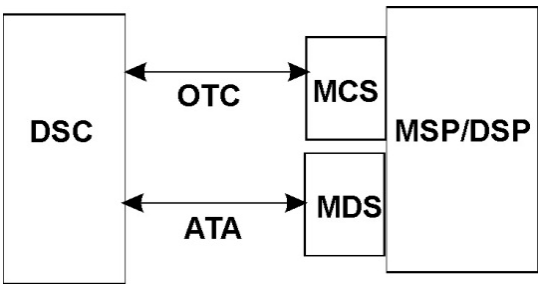


Figure 6. One Time Access Any Time Access (OTC-ATA) Model.

Let us understand how OTC-ATA model works in the proposed system architecture. The steps required to perform data mashup in OTC-ATA model are described below. Each step is either manual or automatic which needs to be performed either once or at any time.

Step-1(Manual-Once): DSC(EU/MSP) defines its data requirement using the schema called Data Mashup Definition (DMD) for mashing up structured data (See Section 5.3).

Step-2(Manual-Once): DSC creates a logical entity called Structured Data Mashup Box (SDMB) to store Mashed up Data Records (MDRs) based on the schema called data mashup definition as defined in step-1 (See Sections 5.4 and 5.5).

Step-3(Manual-Once): DSC defines the association between structured data mashup box and data mashup definition (See Section 5.6).

Creating DMD, SDMB and association between them as mentioned in Step-1, 2 and 3 include simple manual GUI operations like filling data in the table of a web form, word processor, worksheet or database client tools etc.

Step-4(Manual-Once): DSC searches for right MSP/DSP as per its need and thereafter selects the appropriate mashup configuration service identifier available at MSP/DSP. This step is manual and needs to be performed once only.

Step-5(Manual-Once): DSC performs one time configuration after fetching Mashup Configuration Attributes (MCA) from MSP/DSP through mashup configuration service identifier. Upon receiving MCA, EU/MSP configures the mapping between SDMB attributes and MCAttributes, which results in the creation of Structured Data eXchange (SDX) and the association between structured data mashup box and mashup data service (See Sections 5.7–5.11).

Step-6(Manual-Any Time): After completion of one time configuration by DSC for various mashup configuration service identifiers provided by MSPs/DSPs, data mashup can be performed any time by accessing data through mashup data service identifier. When a user clicks on any of the SDMB to view mashed up data then DSC automatically calls mashup data service identifier of MSPs/DSPs which were configured at the time of OTC (See Section 5.12).

Step-7(Automatic-Any Time): MSPs return mashed up data as a response and DSPs return normal structured data (without mashup) as a response whenever their services (MDSIs) are called by DSC. The data received from MSP/DSP would be validated and transformed into the proper format before populating into SDMBs at DSC's end (See Sections 5.13–5.15).

One time configuration mentioned in Step-5 is manual but is as simple as solving the "Match Column Problems". Step 6 is manual but is as simple as clicking the mouse.

Step-8(Manual-Any Time): A user can view mashed up data of each SDMB by just clicking on it and can also make the group of two or more SDMBs to create End User Data Mashup (EUDM). EUDM can be explored by a user by clicking on it to open single mashed up screen which contains the integrated UI views of multiple SMDBs. This step is the final output of the proposed work (See Sections 6.1 and 6.2).

Let us understand details of above Steps 1 to 8 in the following sections.

5.3. Data Mashup Definition (DMD)

Defining the requirement of the data by an ordinary user is really challenging because it cannot be fixed in advance. The requirement of users has been defined in various ways such as service composition [40], document specification [23], scripts [8], plans [13] etc. in the past. In this proposed work, we are introducing simple approach of defining data requirement which are called Data Mashup Definition (DMD) and Structured Data Mashup Box (SDMB). DSCs are required to define the configuration of each SDMB before mashing up the desired structured data in it. This definition is called Data Mashup Definition (DMD) and is the way of defining requirement of data by DSC. Thus, DMD is defined as the collection of metadata and the relation of attributes defined by DSC for filtering, transforming and populating the Structured Data Module (SDM) received from MSP/DSP into various Structured Data Mashup Boxes (SDMBs).

Following Table 2 describes attributes of the Data Mashup Definition (DMD) created by DSC. Here, x_{ij} represents jth attribute of ith DMD. DSC can define any number of DMDs as per its need. Thus, Data Mashup Definition (DMD) can be defined as

$$\text{DMD} = (\text{Dmd\#Id, DMDAttributes})$$

For example,

$$\text{DMDAttributes} = (x_{i1}, x_{i2}, x_{i3}, \ldots\ x_{ij})$$

Table 2. Data Mashup Definition (DMD).

DMD#Id	DMDAttributes			
Dmd#1	x_{11}	x_{12}	...	x_{1k}
Dmd#2	x_{21}	x_{22}	...	x_{2l}
...	...	...	...	...
Dmd#i	x_{i1}	x_{i2}	...	x_{ij}
...	...	...	...	...
Dmd#n	x_{n1}	x_{n2}	...	x_{ns}

Here, Table 2 shows n DMDs, each with the different number of attributes. It should be noted that k, l, j, s etc. show the total number of attributes defined in the DMDs #1, #2, #i ... #n respectively. The value of k, l, j, s etc. may or may not be equal to each other. The scope of this paper is limited to the number and name of attributes of the DMD. The data schema (data types and formats) of these attributes has not been covered in this paper. By default, all the data items are considered as text/string for implementation purpose. After defining DMD, DSC will create SDMB as explained in the next section.

5.4. Structured Data Mashup Box (SDMB)

Structured Data Mashup Box (SDMB) is the logical entity which contains the mashed-up data, defined by DSCs at their end under the schema called Data Mashup Definition (DMD) and has been introduced to store and/or to forward the mashed up data. According to Figure 2, DSC contains one or more SDMBs along with local data. DSC has the filter, transformation and population components so that structured data module received from MSP/DSP can be filtered, transformed into the proper format and populated into appropriate SDMB. Generally, the user will use SDMB for storing and viewing mashed up data but MSP will use the SDMB to perform data mashup and forward them to respective DSCs to support the live (instant) data mashup. There could be any number of SDMBs as per requirement of DSC. Let us understand the composition of SDMB in the next section.

5.5. Composition of SDMB and MDRs

5.5.1. SDMB Composition

Each DSC receives Structured Data Module (SDM) through mashup data service identifier from MSP/DSP and uses SDMB to store and/or forward mashed up data. Structured Data Module (SDM) is the collection of Structured Data Records (SDRs) in the standard format like XML, JSON or CSV etc. Each SDR is filtered and transformed into DSC's defined Mashup Data Record (MDR) and is stored in SDMB using the proposed algorithms (See Section 5.15). Thus, SDMB is the collection of unique Sdmb#Id, Sdm#Id, Msp#Id/Dsp#Id, Mds#Id and one or more MDRs.

$$\text{SDMB} = (\text{Sdmb\#Id, Sdm\#Id, Msp\#Id/Dsp\#Id, Mds\#Id, MDRs})$$

Sdmb#Id is generated when DSC creates new SDMB. Mds#Id is contained in MCAttributes received via mashup configuration service identifier. Msp#Id and Dsp#Id are nothing but URLs of

MSP and DSP respectively. Mds#Id is the unique id of mashup data service identifier through which structured data module would be made available at MSP's/DSP's end.

5.5.2. Mashup Data Records (MDRs)

MDRs are the collection of one or more mashup data record stored in SDMB. Structured data (unmashed) is provided by DSP whereas structured data (mashed up) is provided by MSP. The record provided by MSP/DSP is called Structured Data Record (SDR) but when it is populated into SDMB then it is called Mashup Data Record (MDR). SDR of each structured data module is transformed into DSC's MDR by applying the SDXMapping algorithm. Each MDR consists of unique Mdr#Id and MDR values. Thus, it can be defined as follows:

$$\text{MDR} = (\text{Mdr\#Id}, \text{MDRValues})$$

5.5.3. MDRAttributes and MDRValues

MDRAttributes are the collection of attributes of mashup data record. MDRValues are the collection of one or more Name-Value (N-V) pairs. Thus,

$$\text{MDRAttributes} = (x_1, x_2, x_3, \ldots, x_n)$$

$$\text{MDRValues} = (x_1 = ?, x_2 = ?, x_3 = ?, \ldots, x_n = ?)$$

where x_1, x_2, x_3 ... x_n are the name of attributes defined by DSC using Dmd#Id which is further associated with Sdmb#Id. The symbol "?" shows the value of the attribute which has to be extracted from structured data module by applying filter and transformation algorithm at DSC's end. The next step is to establish the association between structured data mashup box and data mashup definition.

5.6. SDMB-DMD Association

After defining Data Mashup Definition (DMD) and creation of SDMB, an association should be established between them. Each SDMB is associated with one DMD but the reverse is not true (i.e., one DMD may be associated with more than one SDMB).

Thus, association between SDMB and DMD would be defined as:

$$\text{SDMB-DMD} = (\text{Sdmb\#Id}, \text{Dmd\#Id})$$

For example, association

$$\text{SDMB-DMD\#01} = (\text{Sdmb\#01}, \text{Dmd\#i})$$

$$\text{SDMB-DMD\#02} = (\text{Sdmb\#02}, \text{Dmd\#k})$$

show that attributes of Sdmb#01 are defined by Dmd#i and attributes of Sdmb#02 are defined by Dmd#k. It can be observed here that DSC defines its requirement in terms of DMD, SDMB and association between them. After defining data requirement for mashup, DSC is required to fetch mashup configuration attributes from MSP/DSP for one time configuration which has been explained in the next section.

5.7. One Time Configuration (OTC) Algorithm

After determining the right MSP/DSP, DSC is required to get Mashup Configuration Attributes (MCA) by calling mashup configuration service identifier of those MSPs/DSPs. The MCA is the set of attributes required to configure DSC so as to perform the data mashup later on, as and when needed. This is the one time process and can be performed manually or through the pre-defined algorithm. MSP/DSP provides structured data through cloud computing feature like Structured Data as a Service

(SDaaS). Thus, MCA is nothing but the container of data and service attributes and can be identified by Mashup Configuration Service Id i.e., MCS#Id. In order to get mashup configuration attributes, DSC should know MCS#Id of the service provided by MSP/DSP. The mashup configuration service of MSP/DSP will return MCA as a response to DSC. It should also be noted that SDMB should be created and associated with DMD before configuring DSC client. Table 3 shows mashup configuration service and their attributes published by various MSPs/DSPs.

Table 3. MSP's/DSP's Mashup Configuration Service Publication.

Msp#Id/Dsp#Id	Mcs#Id	MCAttributes
www.p.com	Mcs#1	$(p_1, p_2, p_3, p_4 \ldots p_a)$
www.q.com	Mcs#2	$(q_1, q_2, q_3, q_4 \ldots q_b)$
www.r.com	Mcs#3	$(r_1, r_2, r_3, r_4 \ldots r_c)$

Thus, Mashup Configuration Service (MCS) can be defined as:

$$\text{MCS} = (\text{Msp\#Id/Dsp\#Id, Mcs\#Id, MCAttributes})$$

For example, MCS = (www.p.com, Mcs#1, $(p_1, p_2, p_3, p_4, \ldots, p_a)$) where Mcs#1 = www.p.com/mcs1.

The Algorithm 1 is based on one time mashup configuration which would be performed by DSC to send the request to MSP/DSP to get mashup configuration attributes to configure its SDMB so that data can be fetched, transformed and populated into it.

It is clear from the Algorithm 1 that data service consumer selects one or more MSPs/DSPs as per its need and configures SDMB using Mashup Configuration Attributes (MCAttributes) after receiving it from Mashup Configuration Service Identifier (MCSI) calls. The method selectMCAttibute (z, w) will return one of MCAttribute h which will be semantically matched with DMDAttribute z from the list of MCAttributes w. Structured Data eXchange (SDX) would be created at DSC's end as a result of the mashup configuration and identified by SDX#Id. Thus, SDXMapping is the pair of DMDAttribute and MCAttribute which is semantically equal to each other. Table 4 describes DSC's mashup configuration done after receiving MCAttributes and is the one time process performed by data service consumer at its end. In order to perform one time configuration for private data mashup, mashup key is also provided through mashup configuration attributes by private mashup configuration service identifier. It can be noticed here that private mashup configuration service identifier is accessible to authorized data service consumer after proper authentication. In Table 4, Mds#1 and Mds#2 are providing public data mashup and Mds#3 is providing private data mashup through MKey#3 mashup key. The data service consumer can access private mashup data service identifier only after sending this mashup key to MSP/DSP while requesting through the Any Time Access (ATA) model.

Algorithm 1. One Time Mashup Configuration Algorithm

```
Let say p be the DSC which wants to configure its own
SDMBs for data mashup

    q = getSDMBs(p)
    For each SDMB r ∈ q
    begin1
    x = createSDX()
      x.Sdx#Id = getUniqueID()
      x.Dmd#Id = SDMB-DMD.getDmdId(r)
      s = selectDSP_MSP(r)
      For each DSP_MSP t ∈ s
      begin2
        u = selectMCSI(t)
        w = getMCAttributes(u)
        y = DMD.getDMDAttributes(x.Dmd#Id)
        For each DMDAttribute z ∈ y
        begin3
          h = selectMCAttibute(z, w)
          x.SDXMapping.add(z = h)
        end3
      end2
    end1
```

Table 4. DSC's Mashup Configuration.

Msp#Id/Dsp#Id	Mcs#Id	SDX#Id	Mds#Id	Mashup#Key
www.p.com	Mcs#1	Sdx#1	Mds#1	-
www.q.com	Mcs#2	Sdx#2	Mds#2	-
www.r.com	Mcs#3	Sdx#3	Mds#3	MKey#3

Let us understand the attributes, which are required to configure DSC in the following section.

5.8. Mashup Configuration Attributes (MCA)

MCAttributes is the collection of name of the attributes of the data to be mashed up and is associated with unique mashup data service identifier of MSP/DSP. Hence,

For public data mashup,

$$\text{MCAttributes} = (\text{Mds\#1}, (p_1, p_2, p_3, p_4 \ldots, p_m))$$

For private data mashup,

$$\text{MCAttributes} = (\text{Mds\#3}, (r_1, r_2, r_3, r_4 \ldots\ r_n), \text{MKey\#3})$$

Mds#Id is the unique REST URI of mashup data services available at MSP/DSP and would be used by DSC, later on, to send the request to get data from MSP/DSP. Similar to DMDAttributes, MCAttributes mentioned in this paper is limited to name of the attributes only and does not cover its data type and format etc. By default, all the attributes of mashup configuration are of type text/string for implementation purpose. Let us explore the Algorithm 1 to understand the processing of MCAttributes to create Structured Data eXchange (SDX) as per requirement of the DSC.

5.9. SDX Mapping

The data exchange is the process where the data structured under one schema (the source schema) must be restructured and translated into an instance of a different schema (the target schema) [50]. According to them [50], a data exchange setting consists of a source schema S, a target schema T, a set of source-to-target dependencies, and a set t of target dependencies. They also explained that the input to a data exchange problem is a source instance only; the data exchange setting itself (source schema, target schema, and dependencies) is considered fixed. In our proposed work, we have developed SDXMapping in such a way that source to target and target to source mapping is fixed and unique. This process is performed at DSC's end upon receiving MCAttributes which creates Structured Data eXchange (SDX). The SDX is the logical entity which contains the mapping between attributes of DMD and that of MCA. It is used to validate and transform the structured data records received from MSP/DSP and populate them into appropriate SDMB.

Thus, SDX can be defined as follows:

$$\text{SDX} = (\text{Sdx\#Id}, \text{Dmd\#Id}, \text{SDXMapping})$$

where Sdx#Id is the unique id of SDX, Dmd#Id is the id of DMDAttributes and SDXMapping is the collection of a pair of DMDAttribute and MCAttribute semantically equal to each other. For example,

$$\text{SDXMapping} = (x_1 = p_3, x_2 = p_n, x_3 = p_2, \ldots, x_n = p_1)$$

where $x_1, x_2, \ldots x_n$ are the name of DMD attributes defined by data service consumer and $p_1, p_2, \ldots p_n$ are the name of attributes of MCAttributes received from MSP/DSP for one time configuration. Here, pair $x_1 = p_3$ shows that DMDAttribute x_1 is semantically equal to MCAttribute p_3. There should be one to one mapping between these attributes. The Algorithm 1 explains how MCAttribute is processed to create SDX.

Let us understand the output of Algorithm 1 (i.e. SDXMapping) using sample data in tabular format. Table 5 shows the mapping between DMD attributes and MCAttributes which looks like 'Column Matching Problem'. It is clear that the attribute "x_1" defined in DMD is mapped with the attribute "p_3" in MCA.

Table 5. Structured Data eXchange (SDX) Mapping between DMD and MCAttributes.

DMDAttributes	MCAttributes
x_1	p_1
x_2	p_2
x_3	p_3
…	…
x_n	p_n

SDX table shown on Table 6 contains unique Sdx#Id, Dmd#Id and SDXMapping.

Table 6. SDX Mapping between DMD and MCAttributes.

Sdx#Id	Dmd#Id	SDXMapping
Sdx#01	Dmd#02	$(x_1 = p_3, x_2 = p_n, \ldots, x_n = p_1)$
Sdx#02	Dmd#04	$(y_1 = q_4, y_2 = q_m, \ldots, y_m = q_1)$
…	…	…
Sdx#n	Dmd#01	$(z_2 = r_6, z_3 = r_k, \ldots, z_k = r_1)$

After the creation of SDX by data service consumer, the association between structured data mashup box and mashup data service should be updated. Let us understand Mashup Data Service (MDS) first and then its association with SDMB in the next section.

5.10. Mashup Data Service (MDS)

Mashup Data Service (MDS) is the essential feature of MSP/DSP which is nothing but cloud computing paradigm called Structured Data a Service (SDaaS) to provide structured data in the standard format like XML, JSON, CSV etc. on DSC's request. Data as a Service (DaaS) enables multiple users to access data simultaneously on demand but reliability to store and manage data securely is always required to gain customer trust to use data service [51]. The mashup data service of any MSP/DSP is identified by REST URI called Mashup Data Service Identifier (MDSI). Every MSP/DSP is required to publish its mashup services/data services to make the MDSI available to its users. The information about MDSI is made available to DSC through mashup configuration service identifier while performing one time configuration. The MSP/DSP may have one or more mashup data service identifiers as per data available with them. Based on accessibility features of services, mashup data service could be of two types namely private mashup data service and public mashup data service. Private mashup data service provides the data to DSC after proper authentication and authorization of mashup keys whereas public mashup data service provides the data without any verification. Thus, private mashup data service results in private data mashup and public mashup data service results in public data mashup.

5.11. SDMB-MDS Association

Each Structured Data Mashup Box (SDMB) of DSC should be associated with Mashup Data Services (MDS) of MSP/DSP and structured data exchange so that structured data records of structured data module received at DSC's end would be validated and transformed into proper mashup data record format before populating into structured data mashup box. Association between structured data mashup box and mashup data services would be as follows

SDMB-MDS for public data mashup = (Sdmb#Id, Sdx#Id, Mds#Id, Msp#Id/Dsp#Id)

SDMB-MDS for private data mashup = (Sdmb#Id, Sdx#Id, Mds#Id, Msp#Id/Dsp#Id, Mashup#Key)

Table 7 shows one of the snapshots of sample SDMB-MDS association.

Table 7. Structured Data Mashup Box Mashup Data Services (SDMB-MDS) Association.

Sdmb#Id	Sdx#Id	Mds#Id	Msp#Id/Dsp#Id	Mashup#Key
Sdmb#2	Sdx#1	Mds#1	www.p.com	-
Sdmb#2	Sdx#4	Mds#2	www.q.com	-
Sdmb#4	Sdx#6	Mds#2	www.r.com	Key#3

It is clear from Table 7 that Sdmb#2 is associated with www.p.com for Mds#1 using Sdx#1 and it is also associated with www.q.com using Sdx#4 but for Mds#2. Once SDMB-MDS association is determined by DSC, One Time Configuration (OTC) is over for an SDMB. The second phase of the proposed work is based on the ATA model, which includes the Any Time Access (ATA) algorithm, which has been described below.

5.12. Any Time Access (ATA) Algorithm

After defining the association between SDMB and mashup data service, DSC is ready to access data any time by sending service/data request to respective MSP/DSP. Following Algorithm 2 is used

by any DSC at any time by just calling Mashup Data Service Identifier (MDSI) to access Structured Data Module (SDM) which is generated by mashup data service.

Here, DSC will select one SDMB and will receive SDM by calling each Mashup Data Service Identifier (MDSI) associated with SDMB. The SDM t received from MDSI would be validated thereafter transformed and populated into SDMB q using the algorithm (See Section 5.15). Let us understand Structured Data Module and its components in the next section.

Algorithm 2. Any Time Access (ATA) Algorithm

Let p be the DSC which sends request to MSPs/DSPs through one SDMB q for data mashup

```
q = selectSDMB(p)
  r = getMDSIs(q)
  For each MDSI s ∈ r
  begin1
t = getSDM(s)
    if (validate(t))
    transform&populate(t, q)
  end1
```

5.13. Structured Data Module (SDM)

The data made available to data service consumer through mashup data service identifier would be called Structured Data Module (SDM). This paper proposes novel structured data processing system in which structured data module consisting of one or more structured data records would be further filtered, transformed and populated by algorithms mentioned in the Section 5.15 and resultant mashed up data would be placed in Structured Data Mashup Box (SDMB). Let us understand a few terms used in this section.

5.13.1. SDM

The SDM is the well-structured content provided by MSP/DSP through mashup data service identifier calls. It is a module because it may contain multiple structures of data as per the specification mentioned in mashup configuration service identifier. It provides not only structured data but may also contain other information like mashup keys, SDMB details and many other information. Currently, SDM used in this paper is limited to provide structured data record in the standard format like JSON. Every MSP/DSP has one or more mashup data services which generate Structured Data Records (SDRs) to be made available to its users. Thus, each SDM contains Sdm#Id, Mds#Id and one or more SDRs generated by mashup data service. Thus,

$$\text{SDM} = (\text{Sdm\#Id}, \text{Mds\#Id}, \text{SDRs})$$

5.13.2. Structured Data Records (SDRs)

The SDRs can be defined as the collection of one or more structured data record generated by mashup data service and each SDR contains Sdr#Id and one or more SDR values. Thus,

$$\text{SDR} = (\text{Sdr\#Id}, \text{SDRValues})$$

5.13.3. SDRAttributes and SDRValues

SDRAttributes can be defined as the collection of attributes of structured data record.
For example,

$$\text{SDRAttributes} = (p_1, p_2, p_3, \ldots, p_n)$$

SDRValues can be defined as the collection of one or more Name-Value (N-V) pairs.

For example,

$$\text{SDRValues} = (p_1 = v_1, p_2 = v_2, \ldots, p_n = v_n)$$

where $p_1, p_2, \ldots\ p_n$ are name of attributes of SDR and $v_1, v_2, \ldots\ v_n$ are values of those attributes. Next section explains how Structured Data Module would be composed using JSON.

5.14. JSON Composition of Structured Data Module

In our proposed system, various mashup data services of MSP/DSP generate many SDRs and put them together in an SDM. The mashup data service generates SDM on each data request made by DSC. Thus, the refresh rate of SDM provided to DSC is high. The mashup data service composes JSON-SDM as given in Algorithm 3.

Algorithm 3. JSON Composition of SDM

```
  p = newSDM()
  q = getSDRs(p)
  r = newJSONArray()
For each SDR s ∈ q
  begin1
    t = getJSONObject(s)
    r.add(t)
  end1
```

Thus, each SDR generated by mashup data service would be converted into JSONObject and JSONObject would be added to JSONArray to make it available to DSC through mashup data service identifier.

For example, SDR = (Sdr#01, ($p_1 = v_1, p_2 = v_2, \ldots, p_n = v_n$)) would be converted into JSON Object as follows

$$\{\text{“sdrId”:“Sdr\#01”}, \text{“}p_1\text{”:“}v_1\text{”}, \text{“}p_2\text{”:“}v_2\text{”}, \ldots, \text{“}p_n\text{”:“}v_n\text{”}\}$$

Next section explains how SDR would be validated, transformed and populating into SDMB.

5.15. SDR Validation, Transformation and Population Algorithm

Data Mashup Definition (DMD) is defined by DSC at its end whereas SDR is generated at MSP's/DSP's end and provided to DSC through mashup data service identifier. Numbers of attributes in DMD and that of mashup data service may differ or may be equal. The SDR, which is supposed to be populated, may be ignored, if it is not matched with the specification of SDX defined for the SDMB. SDR would be valid if all of its attributes are matched with DMDAttributes of targeted SDMB. Valid SDR is transformed into mashup data record and populated into relevant SDMB. The Algorithm 4 depicts how each SDR is validated and transformed into Mashup Data Record (MDR) before populating into proper SDMB.

The transformation of structured data record into mashup data record requires fetching DMDAttribute of the corresponding SDRAttribute from SDXMapping and value of SDRAttribute is assigned to DMDAttribute. Created MDRValues (name-value pair) is added to MDRs of SDMB as mentioned in line SDMB.MDRValues.add(h=z.value).

Algorithm 4. Validation, Transformation and Population of SDR

Let p be the DSC receives SDM t through MDSI s from DSP r in SDMB q

```
t = getSDM(s)
v = getSDRs(t)
For each SDR w ∈ v
begin1
  y = getSDRAttributes(w)
  if (valid(y))
  begin2
    [fetch Sdx#Id from Table 7]
    Sdx#Id = getSDXId(r, s, q)
    [fetch SDX Mapping from Table 6]
    x = getSDXMapping(Sdx#Id)
    For each SDRAttribute z ∈ y
    begin3
      [fetch DMDAttribute from SDXMapping]
      h = getDMDAttribute(z, x)
      [populate SDMB q]
      SDMB.Dsp#Id = r
      SDMB.Mds#Id = s
      SDMB.Sdmb#Id = q
      SDMB.MDRValues.add(h=z.value)
    end3
  end2
end1
```

6. DSC's Data Mashup

DSC may be either a user or an MSP. The user does mashup for viewing the required mashed up data in a single screen whereas MSP fetches data from DSPs or from other MSPs and forward them to its user after performing the data mashup. The following section explains how data mashup is performed by MSP and EU.

6.1. MSP's Data Mashup

MSPs generally do not require creating views of mashed up data but need to integrate mashed up data and forward the resultant structure data module to its users. Algorithm 5 processes JSON based structure data module to integrate and forward the resultant mashed up JSON-SDM to its users.

The line v.replaceJSONAttribute(x, z) shows that each JSON attribute of JSONObject of the structured data module is replaced by corresponding DMDAttribute while composing the new structured data module. Thus, the value of an attribute of JSON object is assigned to DMDAttribute which is semantically equal as per definition of SDXMapping.

Algorithm 5. MSP's Data Mashup Algorithm

```
p = getSDMB()
q = getMDSIs(p)
h = new JSON_SDM()
For each MDSI s ∈ q
begin1
  r = getDSP(p, s)
  t = getJSON_SDM(s)
  u = getJSONObjects(t)
  For each JSONObject v ∈ u
  begin2
    w = getJSONAttributes(v)
    For each JSONAttribute x ∈ w
    begin3
      [fetch Sdx#Id from Table 7]
      Sdx#Id = getSDXId(r, s, p)
      [fetch SDX Mapping from Table 6]
      y = getSDXMapping(Sdx#Id)
      z = getDMDAttribute(x, y)
      v.replaceJSONAttribute(x, z)
    end3
    h.add(v)
  end2
end1
```

6.2. EU's Data Mashup

The Algorithm 6 explains how end user can explore the integrated UI view of data mashup by grouping various SDMBs in a single screen.

Algorithm 6. Integrated UI View of End User Data Mashup

```
p = getEUDM()
q = getSDMBs(p)
r = getScreen()
For each SDMB s ∈ q
begin1
  t = getView(s)
  r.addView(t)
end1
showScreen(r)
```

When a user selects End User Data Mashup (EUDM) by clicking on it then the view of each SDMB is added to the screen and shown to the user after traversing all the SDMBs of respective end user data mashup.

7. Implementation

7.1. Experimental Setup

In order to implement the proposed work, the experimental setup of the data mashup service network was developed, which has been shown in Figure 7. It includes three services namely admission services, job services and lodging services and the end users.

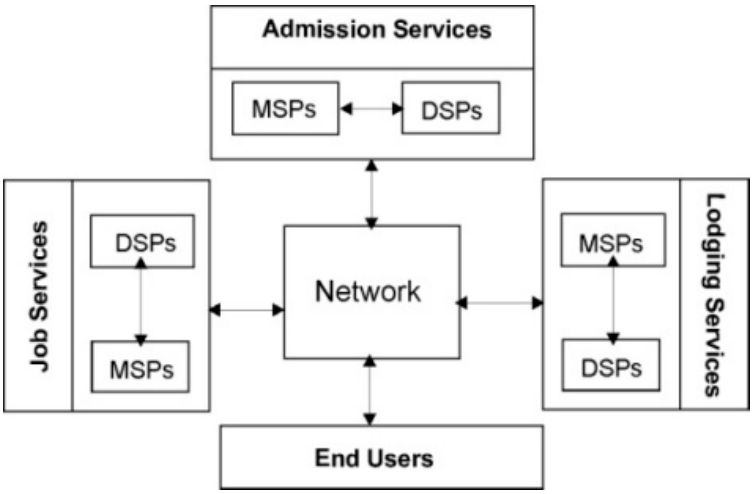


Figure 7. Data Mashup Service Network.

Each service has some MSPs and DSPs connected to each other. For experimental purpose, we developed one MSP and two DSPs of job services, one MSP and three DSPs of admission services and one MSP and three DSPs of lodging services. In our implementation, admission services provide admission notices to students, job services provide job alerts to job seekers and lodging services provide rooms available in various hotels to their travelers. Here the student, the job seeker and the traveler may be the same person who would like to view all the information (admission notices, job alerts and availability of rooms) together in a single screen which may be called mashed up screen. All the stakeholders in our data mashup network used REST protocol and JSON data format for mashup communication.

7.2. Pre-Mashup Configuration (OTC Model)

This section describes how the ordinary user can define mashup requirement of the data in user-friendly manner and what kinds of outputs are generated after pre-mashup configuration. Here, not only the output of pre-mashup configuration is important but also the user-friendly approach used by ordinary user needs to be observed. The success of any mashup tool depends on output of the pre-mashup configuration and the way this output is generated by ordinary user. In our proposed work, a very simple approach (i.e., filling the web forms/selecting the radio buttons) is used by an ordinary user to create the SDXMapping which is the output of pre-mashup configuration.

In the proposed work, defining mashup requirement of the data is as simple as making the list of the name of attributes. Structured data mashup box can also be associated with data mashup definition soon after the declaration of the requirement of the data mashup.

We took the following sample data set to execute proposed work for experimental purpose only and it has nothing to do with real world's data.

Suppose, Data Mashup Definition (DMD) defined by EU#01 are as follows

DMD for SDMB#01 (Admission Services) =
(notice_id, notice_title, name_of_university,
contact_no, link_to_apply, name_of_course, last_date)
DMD for SDMB#02 (Job Services) =
(job_id, job_title, job_description, salary, contact_no,
link_to_apply, years_of_exp, last_date)
DMD for SDMB#03 (Lodging Services) =
(hotel_name, hotel_address, total_rooms,
room_charges, contact_no, booking_link)

Let us understand the SDXMapping of lodging services between various stakeholders of hybrid data mashup as shown in Figure 1 where EU#01 works as data service consumer for MSP#01 and DSP#03.MSP#01 further works as data service consumer for DSP#01 and DSP#02.

Suppose in Figure 1,

EU#01 = www.user1.com
MSP#01 = www.msp1.com
DSP#01 = www.hotel1.com
DSP#02 = www.hotel2.com
DSP#03 = www.hotel3.com

According to above assumptions, Figure 1 can be redrawn as given in the Figure 8.

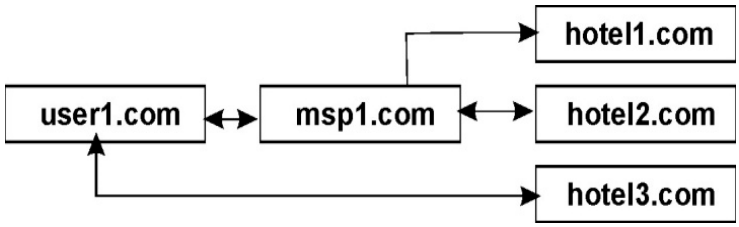


Figure 8. Hybrid Data Mashup for Lodging Services.

The graphical interface, which we developed in the proposed work for pre-mashup configuration has been shown in Figure 9. This interface easily creates the SDXMapping between EU #01 and MSP#01 and looks like 'Column Match Problem'.

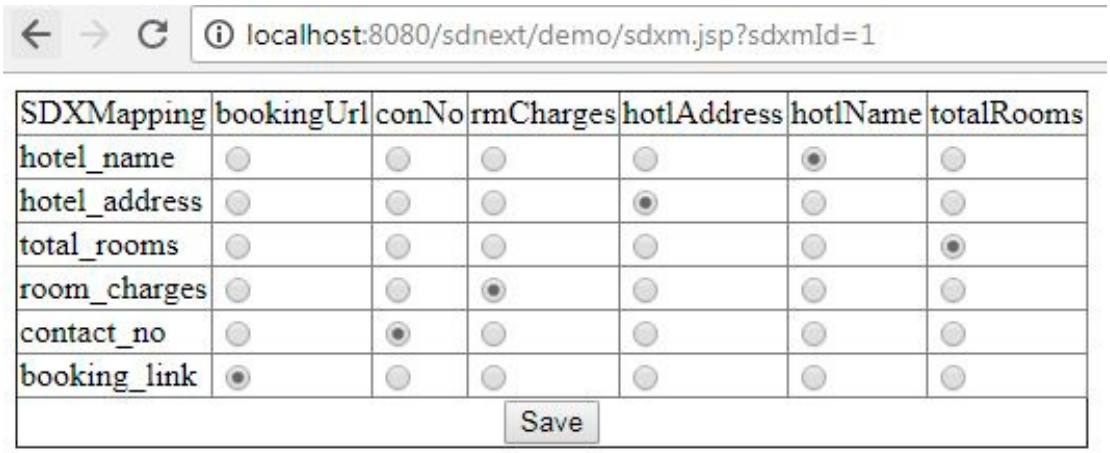


Figure 9. SDXMapping between EU#01 and MSP#01.

It can be noticed in Figure 9 that attribute "hotel_name" of EU#01's data mashup definition is mapped with attribute "hotlName" of MSP#01's mashup service. Similarly, the other attribute's mapping can also be observed. The SDXMapping is created as a result of pre-mashup configuration between data service consumer and MSP/DSP but once only. Similar kinds of user-friendly graphical interfaces were also developed for pre-mashup configuration between other stakeholders in this example. It can be observed here that this kind of graphical approach for pre-mashup configuration is easier for any kind of ordinary user because it simply includes clicking on the radio buttons. Other kinds of user-friendly graphical interfaces can also be designed for an ordinary user for pre-mashup configuration. Attributes of data service consumer are shown in leftmost vertical column and attributes of MSP/DSP are shown in horizontal top rows in Figure 9.

Let us consider the data set taken for end to end SDXMapping between EU#01 and other stakeholders (i.e., MSP#01 and DSP#03) in Table 8. It is called end to end SDXMapping because there is a direct mapping between DMDAttributes of EU#01 and MCAttributes of MSP#01 as well as a direct mapping between DMDAttributes of EU#01 and MCAttributes of DSP#03.

Table 8. End to End SDXMapping between EU#01 and (MSP#01 and DSP#03).

EU#01 (www.user1.com)	MSP#01 (www.msp1.com)	DSP#03 (www.hotel3.com)
hotel_name	hotlName	nameOfHotel
hotel_address	hotlAddress	addressHotel
total_rooms	totalRooms	roomsAvailable
room_charges	rmCharges	chargeOfRooms
contact_no	conNo	moNumber
booking_link	bookingUrl	clickForBook

Let us also consider the data set taken for end to end SDXMapping between MSP#01 and other stakeholders (i.e., DSP#01 and DSP#02) in Table 9. It is clear from Figure 9 that MSP#01 directly interacts with DSP#01 and DSP#02 hence there is need of end to end SDXMapping between them.

Table 9. End to End SDXMapping between MSP#01 and (DSP#01 and DSP#02).

MSP#01 (www.msp1.com)	DSP#01 (www.hotel1.com)	DSP#02 (www.hotel2.com)
hotlName	hotelName	htlName
hotlAddress	hotelAddress	hotelAddr
totalRooms	totalRooms	nosRooms
rmCharges	roomCharges	perDayCharge
conNo	contactNo	mobileNo
bookingUrl	bookingLink	linkForBook

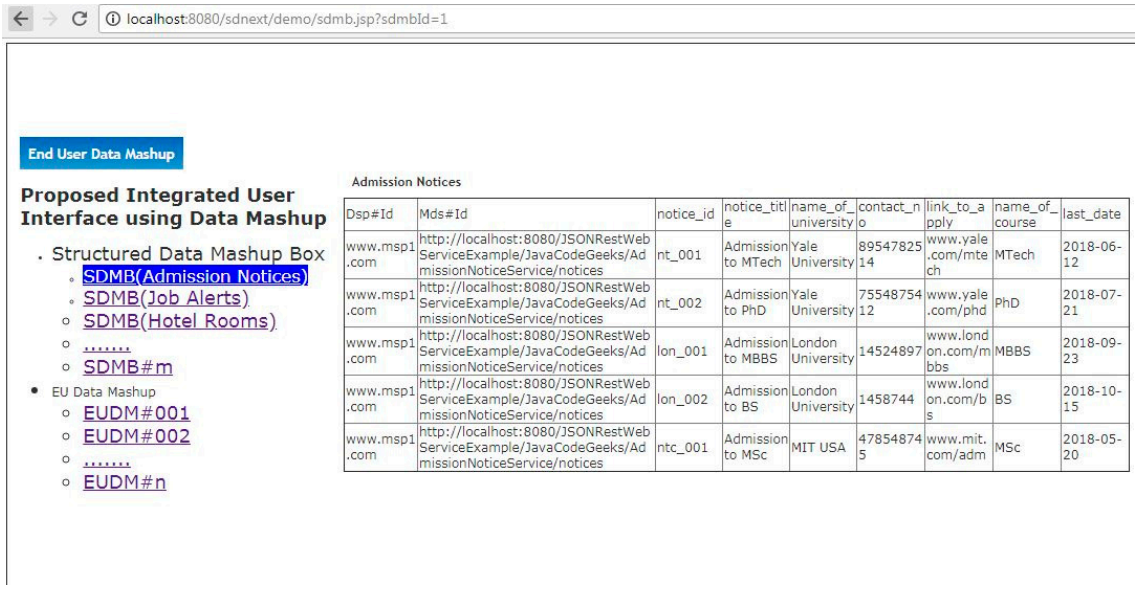
The Tables 8 and 9 show the output of pre-mashup configuration in form of SDXMapping for lodging services which would be used by any time access model by an ordinary user to fetch the required data from multiple data sources and populate them into structured data mashup box. Similar kind of end to end SDXMapping can also be generated for other services (admission services and job services) as mentioned above. The output of post data mashup operation has been explained in the next section.

7.3. SDMB and End User Data Mashup (Any Time Access Model)

This section describes how end user will see the result of the mashed-up data. The web-based UI was developed to implement the proposed work. The mashed-up views of various SDMBs (admission services, job services and lodging services) have been shown in Figures 10–12. Each SDMB is also showing Dsp#Id which is the source of mashed up data and its data service identifier which was used to fetch the structured data module.

Figure 13 shows End User Data Mashup (EUDM) created by end user after grouping various SDMBs at its end. It is the integrated screen of three SDMBs i.e., Admission Notices, Job Alerts and Lodging Services as shown in Figures 10–12 respectively. It can be observed here that SDMB contains similar kind of data, which have been mashed up for the same purpose. For example, SDMB of admission notices contains admission notices sent by various universities or institutes.

It is clear from Figure 13 that the integrated user screen of hybrid kind of mashed up data (fetched from multiple data sources) has been successfully created by end user without applying any technical skills. It can easily be observed that EUs need not to know underlying technologies for data mashup in overall process used in the implementation of the proposed work.

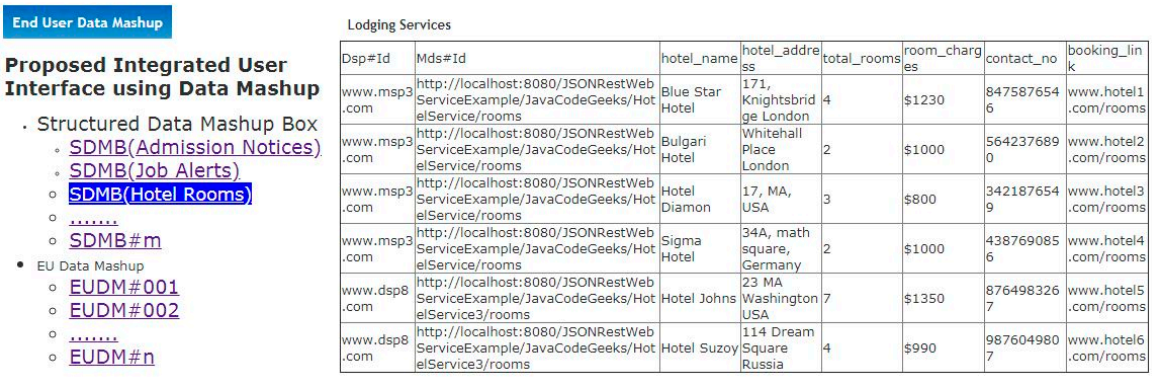


Dsp#Id	Mds#Id	notice_id	notice_title	name_of_university	contact_no	link_to_apply	name_of_course	last_date
www.msp1.com	http://localhost:8080/JSONRestWebServiceExample/JavaCodeGeeks/AdmissionNoticeService/notices	nt_001	Admission to MTech	Yale University	8954782514	www.yale.com/mtech	MTech	2018-06-12
www.msp1.com	http://localhost:8080/JSONRestWebServiceExample/JavaCodeGeeks/AdmissionNoticeService/notices	nt_002	Admission to PhD	Yale University	7554875412	www.yale.com/phd	PhD	2018-07-21
www.msp1.com	http://localhost:8080/JSONRestWebServiceExample/JavaCodeGeeks/AdmissionNoticeService/notices	lon_001	Admission to MBBS	London University	14524897	www.london.com/mbbs	MBBS	2018-09-23
www.msp1.com	http://localhost:8080/JSONRestWebServiceExample/JavaCodeGeeks/AdmissionNoticeService/notices	lon_002	Admission to BS	London University	1458744	www.london.com/bs	BS	2018-10-15
www.msp1.com	http://localhost:8080/JSONRestWebServiceExample/JavaCodeGeeks/AdmissionNoticeService/notices	ntc_001	Admission to MSc	MIT USA	478548745	www.mit.com/adm	MSc	2018-05-20

Figure 10. SDMB#01 (Admission Notices).

Dsp#Id	Mds#Id	job_id	job_title	job_description	salary	contact_no	link_to_apply	years_of_exp	last_date
www.msp2.com	http://localhost:8080/JSONRestWebServiceExample/JavaCodeGeeks/JobService/jobs	jb_001	System Engineer	Programming	1500$	4587445	www.hp.com	10	2018-05-26
www.msp2.com	http://localhost:8080/JSONRestWebServiceExample/JavaCodeGeeks/JobService/jobs	jb_002	Project Manager	Management	2000$	4785452	www.microsoft.com	20	2018-06-12
www.msp2.com	http://localhost:8080/JSONRestWebServiceExample/JavaCodeGeeks/JobService/jobs	j_001	Professor	Teaching	3000$	1458744	www.mit.org	20	2018-10-10

Figure 11. SDMB#02 (Job Alerts).



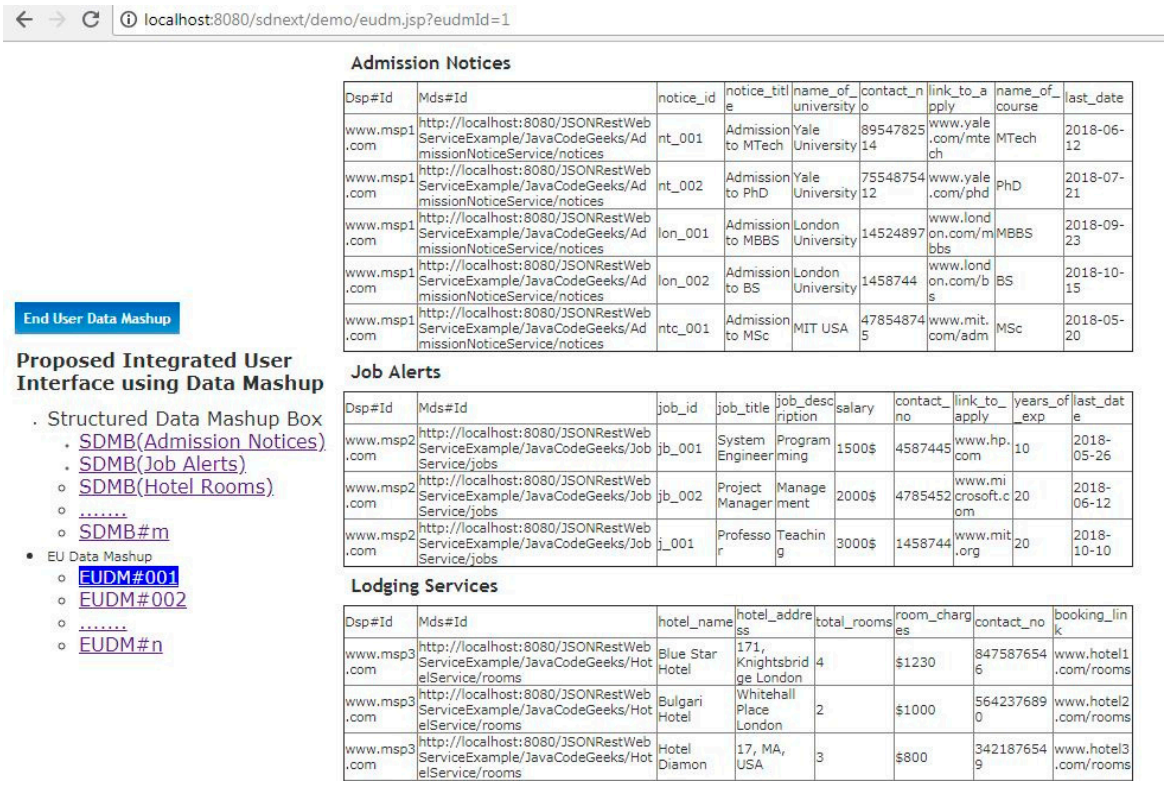
Dsp#Id	Mds#Id	hotel_name	hotel_address	total_rooms	room_charges	contact_no	booking_link
www.msp3.com	http://localhost:8080/JSONRestWebServiceExample/JavaCodeGeeks/HotelService/rooms	Blue Star Hotel	171, Knightsbridge London	4	$1230	8475876546	www.hotel1.com/rooms
www.msp3.com	http://localhost:8080/JSONRestWebServiceExample/JavaCodeGeeks/HotelService/rooms	Bulgari Hotel	Whitehall Place London	2	$1000	5642376890	www.hotel2.com/rooms
www.msp3.com	http://localhost:8080/JSONRestWebServiceExample/JavaCodeGeeks/HotelService/rooms	Hotel Diamon	17, MA, USA	3	$800	3421876549	www.hotel3.com/rooms
www.msp3.com	http://localhost:8080/JSONRestWebServiceExample/JavaCodeGeeks/HotelService/rooms	Sigma Hotel	34A, math square, Germany	2	$1000	4387690856	www.hotel4.com/rooms
www.dsp8.com	http://localhost:8080/JSONRestWebServiceExample/JavaCodeGeeks/HotelService3/rooms	Hotel Johns	23 MA Washington USA	7	$1350	8764983267	www.hotel5.com/rooms
www.dsp8.com	http://localhost:8080/JSONRestWebServiceExample/JavaCodeGeeks/HotelService3/rooms	Hotel Suzoy	114 Dream Square Russia	4	$990	9876049807	www.hotel6.com/rooms

Figure 12. SDMB#03 (Lodging Services).

Following points evaluate the result of the implementation of the proposed work by analyzing the approach and output of experiments with the goals which were set in the beginning (See Section 3) of the proposed work.

- Developed the integrated UI after performing the hybrid data mashup from multiple disparate data sources
- Explored the Single Sign-on (SSO) scheme for private data mashup using mashup keys.

- Only required data were fetched from the MSP/DSP which eliminates the unwanted flow of data between the DSC and MSP/DSP and thus enhance the network performance.
- Simple user-friendly approach used in all steps to perform data mashup
- The REST protocol implemented for all mashup communication



Admission Notices

Dsp#Id	Mds#Id	notice_id	notice_title	name_of_university	contact_no	link_to_apply	name_of_course	last_date
www.msp1.com	http://localhost:8080/JSONRestWebServiceExample/JavaCodeGeeks/AdmissionNoticeService/notices	nt_001	Admission to MTech	Yale University	8954782514	www.yale.com/mtech	MTech	2018-06-12
www.msp1.com	http://localhost:8080/JSONRestWebServiceExample/JavaCodeGeeks/AdmissionNoticeService/notices	nt_002	Admission to PhD	Yale University	7554875412	www.yale.com/phd	PhD	2018-07-21
www.msp1.com	http://localhost:8080/JSONRestWebServiceExample/JavaCodeGeeks/AdmissionNoticeService/notices	lon_001	Admission to MBBS	London University	14524897	www.london.com/mbbs	MBBS	2018-09-23
www.msp1.com	http://localhost:8080/JSONRestWebServiceExample/JavaCodeGeeks/AdmissionNoticeService/notices	lon_002	Admission to BS	London University	1458744	www.london.com/bs	BS	2018-10-15
www.msp1.com	http://localhost:8080/JSONRestWebServiceExample/JavaCodeGeeks/AdmissionNoticeService/notices	ntc_001	Admission to MSc	MIT USA	478548745	www.mit.com/adm	MSc	2018-05-20

Job Alerts

Dsp#Id	Mds#Id	job_id	job_title	job_description	salary	contact_no	link_to_apply	years_of_exp	last_date
www.msp2.com	http://localhost:8080/JSONRestWebServiceExample/JavaCodeGeeks/JobService/jobs	jb_001	System Engineer	Programming	1500$	4587445	www.hp.com	10	2018-05-26
www.msp2.com	http://localhost:8080/JSONRestWebServiceExample/JavaCodeGeeks/JobService/jobs	jb_002	Project Manager	Management	2000$	4785452	www.microsoft.com	20	2018-06-12
www.msp2.com	http://localhost:8080/JSONRestWebServiceExample/JavaCodeGeeks/JobService/jobs	j_001	Professor	Teaching	3000$	1458744	www.mit.org	20	2018-10-10

Lodging Services

Dsp#Id	Mds#Id	hotel_name	hotel_address	total_rooms	room_charges	contact_no	booking_link
www.msp3.com	http://localhost:8080/JSONRestWebServiceExample/JavaCodeGeeks/HotelService/rooms	Blue Star Hotel	171, Knightsbridge London	4	$1230	8475876546	www.hotel1.com/rooms
www.msp3.com	http://localhost:8080/JSONRestWebServiceExample/JavaCodeGeeks/HotelService/rooms	Bulgari Hotel	Whitehall Place London	2	$1000	5642376890	www.hotel2.com/rooms
www.msp3.com	http://localhost:8080/JSONRestWebServiceExample/JavaCodeGeeks/HotelService/rooms	Hotel Diamon	17, MA, USA	3	$800	3421876549	www.hotel3.com/rooms

Figure 13. Integrated UI View using EU Data Mashup.

8. Work Evaluation

Now, the turn has come to evaluate the proposed work, which includes the complete life cycle of the data mashup development by ordinary user using SDRest protocol for end to end data mashup. As already mentioned, the data mashup is the set of techniques and approaches developed in such a way that an ordinary user can use it to fetch their required data from multiple data sources and mash them up in single place without any programming or technical skills. The data mashup always intends to involve the ordinary user throughout the mashup development process and provides a user-friendly approach so that whole process can be adopted by everyone without any technical assistance. The main motivation behind the proposed work is to develop user-friendly approach using the existing technologies and the communication protocol for developing the data mashup between any two stakeholders (Ordinary User/Mashup Service Provider/Data Service Provider) and hence called end to end data mashup.

In order to evaluate the proposed work, there is need of selecting some of the related tools, techniques and approaches developed previously and the parameters to be fixed to compare the proposed work with them. We selected some tools, which were developed by few IT giants and other related works to compare with our proposed work. The comparison of the output of the data mashup only, is not enough to evaluate the work because output covers only one aspect of the whole mashup development process. The techniques and approaches used in each step of the mashup development needs to be evaluated using various parameters. Let us discuss the parameters, which

should be selected for the evaluation of the work then the observations based on these parameters would be summarized.

The ordinary user has been the focus of the industries and the researchers for the mashup development since last two decades but a killer mashup application like email is still waiting for structured data communication for data mashup. Other than ordinary users, there are end user developers, mashup service providers and data service providers who use the data mashup. We have chosen "the targeted users" as a first primary parameter for evolution of the proposed work because it is really a challenging task to make the ordinary user as a major stakeholder of the IT system development. If any of the tools or technique targets the ordinary user then the development of user-friendly UI approach becomes a compulsory feature.

After the popularity of service-oriented architecture, many organizations started providing the data services and mashup services to their users but due to the availability of large numbers of services, it became cumbersome for the user to select the appropriate services from reliable service provider. Hence, selection of mashup service provider or data service provider is another important parameter for evaluating the data mashup approach and technique. The selection of data/service providers could be manual, semi-automatic or automatic. The mashup data or services made available by MSP/DSP could be private, public or hybrid, which have already been discussed previously, is the third parameter for comparison. The method of pulling or pushing the data in request-response pattern is also an important parameter because the bandwidth consumption by data mashup is an important factor to study. Data source, which is another important parameter of this comparative study, plays important role because data source and its interface decides the success of the mashup tools and approaches.

The users are given some GUI based tools like visual editors, web browser etc. and drag and drop like user-friendly approach to allow them to define their own data mashup before fetching the actual data from various disparate data sources. Hence, the mashup development tool and approach has been included as a major parameter to evaluate the proposed work. Pre-mashup configuration is compulsory steps of any mashup development tools or techniques because its output defines the requirement of the user which would be used by mashup algorithm/tools to complete the remaining task of data mashup in automatic mode. Although, the output of mashup can be made available in any desirable form but it is important to know whether this output can be shared, forwarded and reused by same/other user or not. The End User Data Mashup (EUDM) which we discussed in Section 7.3 is also included as one of the parameters which tells whether output of various mashups itself can be combined further to create integrated UI view or not.

Based on above discussions, we have chosen the following nine parameters to evaluate the proposed work

(A) Targeted End Users
(B) End User Data Mashup (EUDM)
(C) Selection of MSPs/DSPs
(D) Types of Mashups
(E) Mashup Strategies
(F) Data Sources
(G) Mashup Development Tools/Approaches
(H) Output of the Pre-Mashup Configuration
(I) Output of the Mashup

We have chosen few reference works and tools developed previously and compared their features with the proposed work using the above parameters as shown in Table 10.

Table 10. The Comparisons of mashup tools, techniques and approaches.

SN	References	(A) Targeted EUs (B) EUDM	(C) Sel. of MSPs/DSPs (D) Types of Mashup (E) Mashup Strategy	(F) Data Source (G) Mashup Development Tools and Approaches	(H) Output of Pre-Mashup Config. (I) Output of Mashup
1.	Query Form. Language [52]	EUD(Type-2)/No	Manual/Public/Pull	RDF/MashQL Ed./Drag and Drop	Graph Sign. and MashQL/SPARQL
2.	IBM:Damia [53]	EUD(Type-1)/No	Manual/Public/Pull	RSS,ATOM,XML/Damia GUI/Drag and Drop	XML document/Damia Feed
3.	Yahoo!Pipes [40]	EUD(Type-1)/No	Manual/Public/Pull	HTML,JSON,XML,CSV,RSS, ATOM,RDF/Visual YPipe Ed./Drag and Drp.	Wired Services (Pipes)/Web Widgets
4.	Intel Mash Maker [11,40]	EUD(Type-2)/No	Manual/Public/Pull	XML,JSON,RSS,RDF,Annotated Web/Browser Extn./Drag and Drop	Tagged Widgets/Web Widgets
5.	IBM Mashup Center [8]	EUD(Type-1)/No	Manual/Public/Pull	REST API/Vis. Mashup Builder/Drag and Drop	Pre-fabricated widgets, feeds/Data Feeds
6.	Microsoft Popfly [40]	EUD(Type-2)/No	Manual/Public/Pull	Web Page/Popfly Editor/Drag and Drop	Connected Blocks/Visual. Data Widget
7.	Google Mash. Editor [8]	EUD(Type-1)/No	Manual/Public/Pull	APIs,RSS,Atom/GME/Programming	Source Code/Feeds, Scripts
8.	Building by Demo [16]	OU/No	Manual/Public/Pull	Web Page/Browser/Drag and Drop	Karma Framework/Worksheet
9.	Configurable RESTful [23]	MSP,EUD(Type-1)/No	Manual/Public/Pull	XML,JSON/PAW mashup engine	SDD, MD/Web Widgets
10.	SOA for High Dim. Prv. Data Mashup [35]	MSP/No	Manual/Private/Pull	NA/Privacy Preserved Algorithm	Service Integration Privacy/Preserved Mashed up Data
11.	Dataflow Patt. Frame. [30]	MSP/No	Semi-Auto./Public/Pull	Nested relational model/Pattern b. frame.	Data Mashup Plan/Mixed Graph
12.	FlexMash [13]	EUD(Type-1)/No	Manual/Public/Pull	Business Objects/Mashup Plan Ed./Drag Drop	Mashup Plan/Executable Representations
13.	EFESTO [54]	EUD(Type-1)/No	Manual/Public/Pull	Web Page/Visual Editor/Drag Drop	UI Templates/Visualized Data
14.	Marmite [55]	EUD(Type-1)/No	Manual/Public/Pull	Web Page/Vis. Web Ed. Drag and Drop	Data Extraction Tool/Spreadsheet View
15.	D-Mash [56]	MSP/No	Manual/Public/Pull	DaaS/Cloud-based computing platform	SOAP API/Mashed up Tables
16.	Mashroom+ [57]	EUD(Type-2)/No	Manual/Public/Pull	HTML,XML,JSON,rel. db/Interactive Data Int. Env./Drag Drop	Matching Algorithm and GUI Tool/Nested Table
17.	End to End Service [34]	MSP,EUD(Type-2)/No	Semi-Auto./Public/Pull	Web Services/MSQL Editor	MSQL/Mashed up Service
18.	Proposed Work	OU,MSP,DSP,EUD/Yes	Manual/Public, Private, Hybrid/Pull,Push	JSON through REST/Browser,Web Form Entry or Selecting Radio Buttons	SDMB and SDXMapping/Auto Populated SDMB

The following points may be summarized after analyzing the above table data.

(A) We can analyze by observing this parameter that the most of the previous works could target to end user developers (type-1 and type-2) and but only few of them got success in developing the data mashup platform for ordinary user. Here, the EUD(type-1) is the technical person who knows some programming skills with script editing and capable to use the mashup tools. The EUD(type-2) is not fully technical person but he/she is more computer savvy than the ordinary user and have some knowledge of browser configuration, plug-ins etc. The ordinary user is completely a not technical person and knows browsing, web form entry, selecting the radio buttons and clicking the submit button only and even one cannot expect operations like drag and drop of some widgets on canvas and connecting them together. The proposed work supports end to end data mashup which facilitates the mashup communication between any two stakeholders, thus it can be used by any of the stakeholder i.e., ordinary user, mashup service provider, data service provider and end user developer. It is obvious that anything, which is developed for ordinary user, can also be used by other technical persons. There are five values taken for the parameter called 'targeted user' i.e., EUD(type-1), EUD(type-2), ordinary user (OU), MSP or DSP. We have explained the reasons about the values assigned to this parameter in the point 'F' i.e., Mashup Development Tool/Approach.

(B) The integrated UI is an important parameter, which integrates the various mashup outputs together to create a single screen of hybrid data mashup. It indicates that output of different data mashups can also be integrated so that user can view them together in single place. This feature has been easily implemented in the proposed work as compared to previous works.

(C) The method of selection of the MSPs/DSPs is the next parameter to evaluate the mashup development works. There are three ways to select MSP/DSP for data mashup i.e., manual, semi-manual and automatic. It can be seen from value of this parameter that the most of the works for selecting the MSP/DSP are manual. Refs. [30,34] could achieve the semi-automatic mode of selection of mashup/data services but are limited to be used by MSP only. The proposed work is also limited to the manual selection of MSP/DSP. It is one of the most challenging steps of mashup development because it involves methods and techniques to transform user's requirement in terms of services or in some other forms. In case of private data mashup, the user already knows the source of the data and hence searching the MSP/DSP is not needed, but it still requires selection of the right services available within that MSP/DSP. In manual approach of selecting MSPs/DSPs in the proposed work, the user will choose one of the MSP/DSP just as he/she selects one email service among all the email service providers.

(D) The fourth parameter is the type of mashup, which has three possible values i.e., private, public, and hybrid. Actually, this parameter describes the accessing modes of the data available with MSP/DSP. It can be seen that most of the previous works focused their work on public data mashup. But, the mashup is generally more towards personal data mashup hence private and hybrid data mashup need to be developed to make this technology more popular among end users. Our proposed work and [35] focused to perform mashup for private data. Study [35] is used to preserve the privacy aspect of user's data before providing it to third party and targets to MSP but our work can be used to perform the data mashup directly by ordinary user without any involvement of third party mashup service provider hence there is no question of preserving privacy of the user's data.

(E) The mashup strategies based on pull and push pattern are important to study because these decide the consumption of the bandwidth of the network and should be carefully chosen. It can be observed here that almost all mashup techniques and approaches focused on pull data mashup but our proposed work recommends both patterns (pull/push) while performing the data mashup as per the nature of the data.

(F) The source of data that is chosen by mashup service provider or the end user is an important aspect of the whole data mashup process. The web pages were the target data source for most of the mashup developers at the initial stage of mashup development and thereafter developers targeted to data in well-structured format. The proposed work supports data in well-structured format provided

by standard REST protocol only. It can easily be observed that the data sources mentioned here, can be categorized into four categories. The first category is the well-structured data source, which contains data in ATOM, JSON, XML, CSV, RSS, RDF etc. format. The second category is the customized data source like nested relational model, business objects etc. Some works directly used HTML, Web Page, Annotated Web, Web Page APIs to fetch the required data and can be placed in third type category and in fourth category, cloud computing paradigm and web service as DaaS, SOAP Web Services, REST calls, REST API etc. were used as data source. The proposed work has been currently implemented to support JSON data format but it can also be easily implemented to support data in well-structured format which would be provided by REST protocol.

(G) The mashup tool, technique and approach is the core thought of the whole mashup development process because it decides not only the UI approach adopted for development of data mashup by ordinary user but also decides the values of all other parameters as mentioned above. The Google Mashup Editor [8] is an ajax based framework and runs in Mozilla Firefox and Microsoft Internet Explorer without any plug-ins and the source code is generated as a result of pre-mashup configuration in XML, Javascript API etc. The script generated by this editor can be used by EUD(type-1) only. Yahoo Pipes [40] is a web-based tool, which allows the end user to drag, and drop pipes (some kind of UI components) and connect them with other pipes to create sequence of pipes as pre-mashup configuration which would be accessed by unique URL to get RSS or JSON etc. Yahoo pipes is suitable for EUD(type-2) user. Intel mash maker [11,40] is also web-based tool, which works directly with web page and suggests the end user whether there is some mashups/widgets in the visited page or not. The Microsoft popfly [40] is very much similar to Yahoo Pipes which uses the term 'block' instead of pipes and used to integrated different services as a pre-mashup configuration by making chain of blocks. Reference [52] developed a graphical Interface to use MashQL with some assumption to generate SPARQL. The approach and technologies used in MashQL editor is limited to be used by EUD(type-1) only.

Damia [53] provides a browser-based client interface with powerful collection of set-oriented feed manipulation operators for importing, filtering, merging, grouping, and otherwise manipulating feeds to accommodate various feed formats such as RSS and ATOM, as well as other XML formats. Again, this tool is limited to be used by EUD(type-1) because the involvement of various operators cannot be easily understood by ordinary user. As far as ordinary user is concerned, the building mashup by demonstration [16] is really a simple approach, which can be used by ordinary user, but major challenge in this work is the extraction of required data from web page whose DOM is not in the proper shape. Reference [23] uses the PAW mashup engine which generates Service Description Document (SDD), Mashup Document (SD) which can be used by the mashup service provider and EUD(type-1) level users only because specification of documents generated here cannot be easily understood by either the ordinary user or the EUD(type-2). Reference [30] targets to mashup service provider which allows to analyze the relationship among data services for discovering mashup patterns based on the history records of data mashup plans. This approach recommends data mashup patterns by analyzing the input and output parameters of the data services if pattern is not found but it is also suitable for MSP.

Reference [13] proposed a flexible execution of data processing and integration scenarios to solve the Extract-Transform-Load processes related issues for end users. They presented an approach for modeling and pattern-based execution of data mashups based on mashup plans, a domain-specific mashup model which can be suitable for EUD(type-1) because the mapping from non-executable model onto different executable ones depending on the use case scenario cannot be performed by ordinary user. Reference [54] uses the graphical interface for visual mapping between data attributes and UI templates which is not easily be used by ordinary user and suitable for EUD(type-1) because it also includes implicit control flow.

Marmite [55] is an end-user programming tool for creating web-based mashups. It uses a set of operators to extract and process data from web pages and uses data flow to create spreadsheet view

that shows the current values of the data. Reference [56] used the cloud computing paradigm Data as a Service (DaaS) to develop mashup framework using SOAP API and limited to be used by MSP. Mashroom+ [57] was developed to support users to easily process and combine data with visualized tables. This work focuses on an interactive matching algorithm which is designed to synthesize the automatic matching results from multiple matchers as well as user feedbacks. It can be studied in detail that it can be found suitable for EUD(type-2) because of complexity of defining the user's requirement, data mapping and the output supported.

There is always need of pre-mashup configuration before performing actual data mashup. Pre-mashup configuration is used to define data requirement of the user. Most of the previous works tried to develop user-friendly graphical interface because of the involvement of ordinary users. The proposed work recommends to use the existing web browser without any extension or plug-ins installation or configuration. Alternatively, mobile app screen or desktop application can also be used for pre-mashup configuration. It can be seen from Figure 9 that it simply requires skill of web form entry, selection of radio buttons etc. to complete the pre-mashup requirement, which is easier than any other tools/approaches developed previously. It can be observed here that every mashup developer has generated pre-mashup configuration before performing the actual data mashup but approaches used by them were not much suitable for ordinary user.

(H) Output of pre-mashup configuration is the one of evaluation parameter which would be used to perform data extraction, filtering, data mapping and population etc. The success of the data mashup depends on the algorithm and output of pre-mashup configuration. The outputs of pre-mashup configuration of the proposed work are DMD, SDMB, SDXMapping and data validation, transformation and population algorithm which would automate the task of fetching required data from multiple data sources and populate them into SDMB which can be viewed by ordinary user in any desired format.

(I) Output of the mashup can be a web widget, purely structured data or in some other formats which depends upon the tools and technologies used to perform the data mashup. Till now, there is no standard format of output of data mashup but it would be better if one can provide data in well-structured format so that it can be consumed and viewed in user's preferable format.

In brief, the following points may be considered as improvements in the proposed work over the previous works.

i. The proposed work supports all stakeholders of mashup development i.e., Ordinary User, End User Developer, Mashup Service Provider, Data Service Providers by providing user-friendly end user configurable approach at every stage of the mashup development.
ii. The integration of structured data mashup box helps the ordinary user to produce integrated view of his mashed-up data in single screen.
iii. The proposed work includes all kinds of data mashups i.e., public, private and hybrid data mashup with both mashup strategies i.e., push and pull.
iv. It supports mashup of data in any structured format which would be made available by REST protocol.
v. It uses simple web browser without any extensions or plug-ins and involves the user interface events like selecting the radio buttons, fill the web form and submit the button which is most suitable for an ordinary user.
vi. Outputs of pre-mashup configuration (i.e., structured data mashup box and structured data exchange mapping) are used to populate the data in auto mode.

Currently, followings are few limitations of the proposed work which need to be investigated further.

i. The manual intervention on structured data exchange mapping
ii. The manual searching of mashup and data services

iii. The schema mapping is not covered in the current work while performing structured data exchange mapping

9. Conclusions and Future Scope

The largest group of internet users belongs to ordinary users who would like to view all their required information at one digital place instead of visiting multiple webs/apps because of the exponential growth of data and its services. The data mashup is the set of techniques and approaches which have always been on the focus of researchers and industries for last 2 decades because it facilitates its users to develop their own application for viewing required information from multiple data sources at one digital place defined by them. The data mashup targets to ordinary users and their involvement cannot be avoided but most of the works previously published expect some technical skills from them hence data mashup could not become as popular as email communication. This work eliminates the need of technical skills from users and provides user-friendly approach to develop end to end data mashup, even by ordinary users.

In this paper, we have introduced the concept of Structured Data REST (SDRest) protocol which includes the innovative concept of Structured Data Mashup Box to store, view and integrate structured data from multiple data sources using standard REST communication between Data Service Consumer and Data Service Provider. SDRest protocol has been explored using two models namely One Time Configuration (OTC) Model and Any Time Access Model (ATA).

The one time configuration model includes the pre-mashup configuration which is the essential process of any data mashup. The success of the data mashup depends on the algorithm and output of pre-mashup configuration, which would be used by data service consumer to fetch and mashup the required data from multiple data sources. Our implementation shows that the outputs of pre-mashup configuration of the proposed work are data mashup definition, structured data mashup box, SDXMapping and other supporting algorithms and a very simple approach (i.e., filling the web forms/selecting the radio buttons) has been developed to involve ordinary user to generate the desired output. The Any Time Access (ATA) model shows that the integrated UI view of hybrid kind of mashed up data (fetched from multiple data sources) has been successfully created by an ordinary user without applying any technical skills.

The proposed work has been evaluated by comparing it with other related works and it concludes that the proposed work supports all stakeholders of mashup development i.e., Ordinary User, End User Developer, Mashup Service Provider, Data Service Providers by providing user-friendly end user configurable approach at every stage of the mashup development. It not only includes public, private and hybrid data mashup but also supports push and pull mashup strategies. Thus, the main objective to develop user-friendly approach using the existing technologies and the communication protocol for developing the data mashup between any two stakeholders which may also be called end to end data mashup, has been successfully achieved in the proposed work. In future, this protocol can be generalized to develop end to end data mashup with unified approach between any two entities of the internet. Data schema mapping has not been covered in this paper and can be explored as future research work. Further, there is much scope to generalize this work for Structured Data Communication among IoT devices in future.

Author Contributions: This paper is the collaborative efforts of both the authors. Both have read and approved the final manuscript.

Acknowledgments: Authors are thankful to Barkatullah University, Bhopal, India for providing platform for research work, research centre (NITTTR, Bhopal, India) for providing research lab & other resources for completing the research work and reviewers for their valuable comments.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Beemer, B.; Dawn, G. Mashups: A Literature Review and Classification Framework. *Future Internet* **2009**, *1*, 59–87. [CrossRef]
2. Imran, M. An Effective End-User Development Approach through Domain-Specific Mashups for Research Impact Evaluation. *arXiv*, 2013; arXiv:1312.7520.
3. Cheng, B.; Zhai, Z.; Zhao, S.; Chen, J. LSMP: A Lightweight Service Mashup Platform for Ordinary Users. *IEEE Commun. Mag.* **2017**, *55*, 116–123. [CrossRef]
4. Khokhar, R.H.; Benjamin, C.M.F.; Farkhund, I.; Dima, A.; Jamal, B. Privacy-Preserving Data Mashup Model for Trading Person-Specific Information. *Electron. Commer. Res. Appl.* **2016**, *17*, 19–37. [CrossRef]
5. Fielding, R.T.; Richard, N.T. Architectural Styles and the Design of Network-Based Software Architectures. Ph.D. Dissertation, University of California, Irvine, CA, USA, 2000.
6. Díaz, O.; Aldalur, I.; Arellano, C.; Medina, H.; Firmenich, S. Web mashups with WebMakeup. In *Rapid Mashup Development Tools*; Springer: Cham, Switzerland, 2016; pp. 82–97.
7. Aghaee, S.; Pautasso, C.; De Angeli, A. Natural end-user development of web mashups. In Proceedings of the 2013 IEEE Symposium on Visual Languages and Human Centric Computing, San Jose, CA, USA, 15–19 September 2013; pp. 111–118.
8. Taivalsaari, A. *Mashware: The Future of Web Applications*; Sun Microsystems, Inc.: Mountain View, CA, USA, 2009.
9. Desolda, G.; Carmelo, A.; Maria, F.C.; Maristella, M. End-User Composition of Interactive Applications through Actionable UI Components. *J. Vis. Lang. Comput.* **2017**, *42*, 46–59. [CrossRef]
10. Yu, J.; Benatallah, B.; Casati, F.; Daniel, F. Understanding Mashup Development. *IEEE Internet Comput.* **2008**, *5*, 44–52. [CrossRef]
11. Ennals, R.; Brewer, E.; Garofalakis, M.; Shadle, M.; Gandhi, P. Intel Mash Maker: Join the web. *ACM SIGMOD Rec.* **2007**, *36*, 27–33. [CrossRef]
12. Stolee, K.T.; Elbaum, S. Identification, impact, and refactoring of smells in pipe-like web mashups. *IEEE Trans. Softw. Eng.* **2013**, *39*, 1654–1679. [CrossRef]
13. Hirmer, P.; Mitschang, B. FlexMash–flexible data mashups based on pattern-based model transformation. In *Rapid Mashup Development Tools*; Springer: Cham, Switzerland, 2016; pp. 12–30.
14. Ghiani, G.; Fabio, P.; Lucio, D.S.; Pintori, G. An Environment for End-User Development of Web Mashups. *Int. J. Hum.-Comput. Stud.* **2016**, *87*, 38–64. [CrossRef]
15. Paternň, F. End user development: Survey of an Emerging Field for Empowering People. *ISRN Softw. Eng.* **2013**, *2013*, 532659. [CrossRef]
16. Tuchinda, R.; Knoblock, C.A.; Szekely, P. Building mashups by demonstration. *ACM Trans. Web (TWEB)* **2011**, *5*, 16. [CrossRef]
17. Ferrara, E.; De Meo, P.; Fiumara, G.; Baumgartner, R. Web data extraction, applications and techniques: A survey. *Knowl.-Based Syst.* **2014**, *70*, 301–323. [CrossRef]
18. Yang, J.; Wittern, E.; Ying, A.T.; Dolby, J.; Tan, L. Automatically Extracting Web API Specifications from HTML Documentation. *arXiv*, 2018; arXiv:1801.08928.
19. Lee, Y.J. Semantic-Based Web API Composition for Data Mashups. *J. Inf. Sci. Eng.* **2015**, *31*, 1233–1248.
20. Dojchinovski, M.; Vitvar, T. Linked web APIs dataset. *Semant. Web.* **2018**, *9*, 381–391. [CrossRef]
21. Fischer, T.; Fedor, B.; Andreas, N. An Overview of Current Approaches to Mashup Generation. Available online: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.567.349 (accessed on 12 September 2018).
22. Hoang, D.D.; Paik, H.Y.; Benatallah, B. An analysis of spreadsheet-based services mashup. In Proceedings of the Twenty-First Australasian Conference on Database Technologies, Brisbane, Australia, 18–22 January 2010; Volume 104, pp. 141–150.
23. Ma, S.P.; Huang, C.Y.; Fanjiang, Y.Y.; Kuo, J.Y. Configurable RESTful service mashup: A process-data-widget approach. *Appl. Math. Inf. Sci. (AMIS)* **2015**, *9*, 637–644.
24. Zhang, F.; Hwang, K.; Khan, S.; Malluhi, Q. Skyline Discovery and Composition of Inter-Cloud Mashup Services. *IEEE Trans. Serv. Comput.* **2016**, *1*, 72–83. [CrossRef]
25. Liu, X.; Ma, Y.; Huang, G.; Zhao, J.; Mei, H.; Liu, Y. Data-driven composition for service-oriented situational web applications. *IEEE Trans. Serv. Comput.* **2015**, *8*, 2–16. [CrossRef]

26. Zhai, Z.; Cheng, B.; Tian, Y.; Chen, J.; Zhao, L.; Niu, M. A Data-Driven Service Creation Approach for End-Users. *IEEE Access* **2016**, *4*, 9923–9940. [CrossRef]
27. Nečaský, M.; Helmich, J.; Klímek, J. Platform for automated previews of linked data. In Proceedings of the 19th ACM International Conference on Information Integration and Web-Based Applications & Services, Salzburg, Austria, 4–6 December 2017; pp. 395–404.
28. DiFranzo, D.; Graves, A.; Erickson, J.S.; Ding, L.; Michaelis, J.; Lebo, T.; Patton, E.; Williams, G.T.; Li, X.; Zheng, J.G.; et al. The web is my back-end: Creating mashups with linked open government data. In *Linking Government Data*; Springer: New York, NY, USA, 2011; pp. 205–219.
29. Salminen, A.; Tommi, M. Mashups-Software Ecosystems for the Web Era. In Proceedings of the Fourth International Workshop on Software Ecosystems 2012, Boston, MA, USA, 18 June 2012; pp. 18–32.
30. Wang, G.; Han, Y.; Zhang, Z.; Zhang, S. A Dataflow-Pattern-Based Recommendation Framework for Data Service Mashup. *IEEE Trans. Serv. Comput.* **2012**, *8*, 889–902. [CrossRef]
31. Liang, T.; Chen, L.; Wu, J.; Xu, G.; Wu, Z. SMS: A Framework for Service Discovery by Incorporating Social Media Information. *IEEE Trans. Serv. Comput.* **2016**. [CrossRef]
32. Xia, B.; Fan, Y.; Tan, W.; Huang, K.; Zhang, J.; Wu, C. Category-Aware API Clustering and Distributed Recommendation for Automatic Mashup Creation. *IEEE Trans. Serv. Comput.* **2015**, *8*, 674–687. [CrossRef]
33. Yao, L.; Wang, X.; Sheng, Q.Z.; Benatallah, B.; Huang, C. Mashup Recommendation by Regularizing Matrix Factorization with API Co-Invocations. *IEEE Trans. Serv. Comput.* **2018**. [CrossRef]
34. Bouguettaya, A.; Nepal, S.; Sherchan, W.; Zhou, X.; Wu, J.; Chen, S.; Liu, D.; Li, L.; Wang, H.; Liu, X. End-to-End Service Support or Mashups. *IEEE Trans. Serv. Comput.* **2010**, *3*, 250–263. [CrossRef]
35. Fung, B.C.; Trojer, T.; Hung, P.C.; Xiong, L.; Al-Hussaeni, K.; Dssouli, R. Service-Oriented Architecture for High-Dimensional Private Data Mashup. *IEEE Trans. Serv. Comput.* **2012**, *5*, 373–386. [CrossRef]
36. Nguyen, H.V.; Iacono, L.L. RESTful IoT Authentication Protocols. In *Mobile Security and Privacy*; Syngress: Rockland, MA, USA, 2016; pp. 217–234.
37. Lee, S.; Jo, J.Y.; Kim, Y. Method for Secure RESTful Web Service. In Proceedings of the 2015 IEEE/ACIS 14th International Conference on Computer and Information Science (ICIS), Las Vegas, NV, USA, 28 June–1 July 2015; pp. 77–81.
38. Peng, D.; Li, C.; Huo, H. An Extended Usernametoken-based Approach for REST-Style Web Service Security Authentication. In Proceedings of the 2nd IEEE International Conference on Computer Science and Information Technology, Beijing, China, 8–11 August 2009; pp. 582–586.
39. Serme, G.; de Oliveira, A.S.; Massiera, J.; Roudier, Y. Enabling Message Security for Restful Services. In Proceedings of the IEEE 19th International Conference on Web Services, Honolulu, HI, USA, 24–29 June 2012; pp. 114–121.
40. Di, L.G.; Hacid, H.; Paik, H.Y.; Benatallah, B. Data Integration in Mashups. *ACM SIGMOD Rec.* **2009**, *38*, 59–66.
41. Bhide, M.; Deolasee, P.; Katkar, A.; Panchbudhe, A.; Ramamritham, K.; Shenoy, P. Adaptive Push-Pull: Disseminating Dynamic Web Data. *IEEE Trans. Comput.* **2002**, *51*, 652–668. [CrossRef]
42. Wang, G.; Zhang, S. Freshness-Aware Data Service Mashups. In Proceedings of the Asia-Pacific Services Computing Conference, Zhangjiajie, China, 16–18 November 2016; pp. 435–449.
43. Rahm, E.; Bernstein, P.A. A survey of approaches to automatic schema matching. *VLDB J.* **2001**, *10*, 334–350. [CrossRef]
44. Del Fabro, M.D.; Bézivin, J.; Jouault, F.; Valduriez, P. Applying Generic Model Management to Data Mapping. Available online: https://www.semanticscholar.org/paper/Applying-Generic-Model-Management-to-Data-Mapping-1-Fabro-B%C3%A9zivin/503d665eb6c99f9f7200712365f47edb362d7ecc (accessed on 12 September 2018).
45. Atay, M.; Chebotko, A.; Liu, D.; Lu, S.; Fotouhi, F. Efficient schema-based XML-to-Relational data mapping. *Inf. Syst.* **2007**, *32*, 458–476. [CrossRef]
46. Guo, M.; Yu, Y. Mutual Enhancement of Schema Mapping and Data Mapping. Available online: https://www.semanticscholar.org/paper/Mutual-Enhancement-of-Schema-Mapping-and-Data-Guo-Yu/491cc554f774337c0bfae8f7ff9e8d546b42f3e7 (accessed on 12 September 2018).
47. Zhao, H.; Doshi, P. Towards Automated Restful Web Service Composition. In Proceedings of the IEEE International Conference on Web Services, Los Angeles, CA, USA, 6–10 July 2009; pp. 189–196.

48. Dustdar, S.; Schreiner, W. A Survey on Web Services Composition. *Int. J. Web Grid Serv.* **2005**, *1*, 1–30. [CrossRef]
49. Sheng, Q.Z.; Qiao, X.; Vasilakos, A.V.; Szabo, C.; Bourne, S.; Xu, X. Web Services Composition: A Decade's Overview. *Inf. Sci.* **2014**, *280*, 218–238. [CrossRef]
50. Mecca, G.; Papotti, P.; Santoro, D. Schema Mappings: From Data Translation to Data Cleaning. In *A Comprehensive Guide through the Italian Database Research over the Last 25 Years*; Springer: Berlin, Germany, 2018; pp. 203–217.
51. Aftab, S.; Afzal, H.; Khalid, A. An approach for secure semantic data integration at data as a service (DaaS) layer. *Int. J. Inf. Educ. Technol.* **2015**, *2*, 124. [CrossRef]
52. Jarrar, M.; Dikaiakos, M.D. A Query Formulation Language for the Data Web. *IEEE Trans. Knowl. Data Eng.* **2012**, *24*, 783–798. [CrossRef]
53. Simmen, D.E.; Altinel, M.; Markl, V.; Padmanabhan, S.; Singh, A. Damia: Data mashups for intranet applications. In Proceedings of the ACM SIGMOD International Conference on Management of Data, Vancouver, BC, Canada, 9–12 June 2008; pp. 1171–1182.
54. Desolda, G.; Ardito, C.; Matera, M. EFESTO: A platform for the end-user development of interactive workspaces for data exploration. In *Rapid Mashup Development Tools*; Springer: Cham, Switzerland, 2016; pp. 63–81.
55. Wong, J. Marmite: Towards end-user programming for the web. In Proceedings of the IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC 2007), Coeur d'Alene, ID, USA, 23–27 September 2007; pp. 270–271.
56. Arafati, M.; Dagher, G.G.; Fung, B.C.; Hung, P.C. D-mash: A framework for privacy-preserving data-as-a-service mashups. In Proceedings of the IEEE 7th International Conference on Cloud Computing, Anchorage, AK, USA, 27 June–2 July 2014; pp. 498–505.
57. Liu, C.; Wang, J.; Han, Y. Mashroom+: An interactive data mashup approach with uncertainty handling. *J. Grid Comput.* **2014**, *12*, 221–244. [CrossRef]

© 2018 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0/).