IJCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Vol.-6, Issue-10, Oct 2018 E-ISSN: 2347-2693

Multi-Configuration Styles of Structured Data Mashup using SDXMapping

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

[1,2]Dept. of Computer Engineering and Applications, NITTTR, Bhopal, India

[*]*Corresponding Author: prakashnarayan007@gmail.com, Tel.: +91-9039296414*

Available online at: www.ijcseonline.org

Accepted: 16/Oct/2018, Published: 31/Oct/2018

***Abstract*—** The structured data mashup is the special kind of data mashup which deals with well structured data throughout the mashup development process. This paper highlights mashup development life cycle and the roles of IT developers as well as end users in various steps of mashup development. The pre-mashup configuration is the essential process of the any data mashup development that creates mashup module consisting of data module, service module, mapping module and UI module. In this work, we have explored mapping module called SDXMapping to design multi-configuration styles of structured data mashup, which will not only help the mashup developers but also the end users to develop the data mashup as per the situational need.

***Keywords*—**Data Mashup, Structured Data Mashup, Mashup Development Life Cycle, SDXMapping, Multi-Configuration Styles

I. INTRODUCTION

There are huge amount of personal & public information which are stored in different internet sources and accessed via webs, apps and other interfaces. The personal information is generally treated as private information and requires credentials to access it whereas public information is open and available to all without any identity required. But, in order to access this information, users have to dependent on IT developers who provide information system using webs/apps. The users are required to explore webs/apps to understand interfaces and the way to reach to the desired information available inside it. The data mashup [1][2] is becoming popular due to its nature of involving the user to develop its own web/app to view its required data at one place after fetching it from multiple internet sources.

There are basically three types of data i.e. structured data, unstructured data and semi-structured data. Among all three kinds of data, structured data is easier to understand and process because it is well defined by attributes and their values. The system with semi-structured data is less efficient as well as less flexible[3]. Data mashup is the general kind of mashup which deals with all kinds of data. This work explores special kind of data mashup called structured data mashup which fetches only well structured data from multiple data sources and integrate all of them to populate into Structured Data Mashup Box (SDMB)[4]. The mashed up data are stored in SDMB (kind of logical entity) and are shown to the user through user's defined integrated view. This work exploits mashup configuration styles using Structured Data eXchange Mapping (SDXMapping) which is the output of pre-configuration mashup process taken from our previous work [4] and is used for data mapping. The pre-mashup configuration is the essential process of the whole mashup development life cycle. The major contribution of this paper is to explore multi-configuration styles of structured data mashup using SDXMapping which will help the mashup service provider to design different mashup frameworks/applications to fulfill the different situational need of the user.

The rest of the paper is organized as follows. Section II describes the related work which covers types of data mashup, mashup configurations styles, and data mapping etc. Section III explores the mashup development life cycle to understand steps of mashup development based on two models called One Time Configuration (OTC) model and Any Time Access (ATA) model. Section IV explains structured data exchange mapping in brief. Section V describes multi-configuration mashup styles designed using SDXMapping and at last the conclusion.

II. RELATED WORK

Before understanding mashup configuration styles, let us understand the process of mashup development first. The development of mashed up webs/apps are different from that of traditional software development. Figure 1 shows the roles

© 2018, IJCSE All Rights Reserved

of IT developers and end users to develop normal/mashed up webs/apps. It can be seen from the figure 1 that IT developers follow the Software Development Life Cycle (SDLC) and use the software development framework/specification like .net, java based servlet, struts & spring, php, android etc to develop normal webs/apps. The end users are bounded to use normal webs/apps which are developed by IT developers in current traditional approach of developing webs/apps. But, mashed up webs/apps are developed by user itself.

The development of mashup applications is gaining the popularity but the approach of developing such applications is completely different from the traditional application development. The mashup development framework produced by IT developers would be used by end user to develop their own mashed up webs/apps using mashup development life cycle. Thus, the role of IT developers is limited to develop mashup develop framework/application. The mashup development framework should be designed and developed in such a way so that even an ordinary user can use it for developing information system for its own use. Due to usability of the data mashup, frameworks are being developed which not only support publicly available data like RSS, Atom etc but also facilitate interface to access database and legacy system [5].

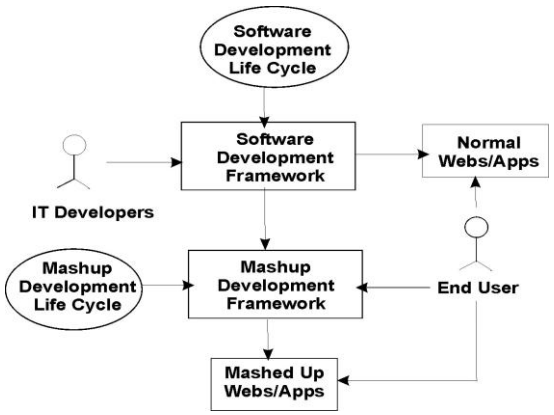

Figure 1. SDLC Vs MDLC

There are many mashup tools, techniques and approaches developed previously. Most of the tools, technique and approaches[5][6] involved end user developer[7][8]/ programmer to use mashup development framework and thus do not follow the MDLC to be performed by ordinary user. Reference [9] explained various mashup frameworks based on approaches like programming paradigm, scripting, spreadsheet, wiring, programming by demonstration and automatic mashup creation etc. The end user uses mashup development framework [10] and follows the complete life cycle of developing mashup but most of the works published do not cover the complete life cycle and are limited to focus few aspects only.

There are various types of data mashup like Photo mashup[11], Video mashup[12], Music Mashup[13], News mashup[14][15][16], Mobile Mashup[17], Linked Data Mashup[18][19][20], Map Mashup[21] [22] [23], Enterprise Mashup[24][25], Semantic Mashup[26],Service Mashup[27], IoTMaaS[28], DaaS[29] Web Mashup[5], Shallow Web & Deep Web Mashup[30][31]. There are two popular mashup styles known as client side mashup and server side mashup [29][32] which specifies the location where the mashup would be performed before providing it to the user. Many times, mashup applications are created by applying the mix approach mashup (may also be called hybrid approach). The pre-mashup configuration is one of the essential steps of MDLC which configures the mashup framework to fetch the various needed data from data sources. This step is used to clearly define the data requirement of the user before performing mashup. This step should be simple enough so that even ordinary user can explore it without any programming or scripting.

Generally, browsers, desktop apps or mobile apps are used to perform pre-mashup configuration during mashup development. Not only the process of pre-mashup configuration but also its output is important because it is used to perform the filter, transformation and population while performing the actual data mashup. For example, some of the outputs of pre-mashup configuration are graph signature & MashQL by [33], XML document by IBM Damia [34], pipes by Yahoo Pipes [14] and MSQL by [35] etc. In this paper, we have used mapping module called SDXMapping as major component of pre-mashup configuration to design multi-configuration styles of structured data mashup so that execution of filter, transformation and population of mashup can be done in different ways to fulfill the situational need of the end user.

There are many works previously published on unstructured or semi-structured data mashup but very few of them cover the structured data mashup. Reference [36] investigated various schema mapping techniques based on column names and their values. Building mashups by demonstration [37] is the best example of structured data mashup which primary focuses on the structured data and understands the user's requirement through demonstration. In this case, pre-mashup configuration is performed through the demonstration and output is generated accordingly.

III. MASHUP DEVELOPMENT LIFE CYCLE

The data mashup is neither similar to traditional web application nor like other standalone app i.e. desktop/mobile app. In traditional software development life cycle, the IT developers are involved throughout the life cycle of the development but the development of the data mashup involves the ordinary user throughout its development. We

© 2018, IJCSE All Rights Reserved

have divided the whole mashup development process called Mashup Development Life Cycle (MDLC) of the data mashup into two parts i.e. One Time Configuration (OTC) and Any Time Access (ATA) to understand whole mashup development in a simplified way (See figure 2). The OTC and ATA models are well explained in our previous work [4].

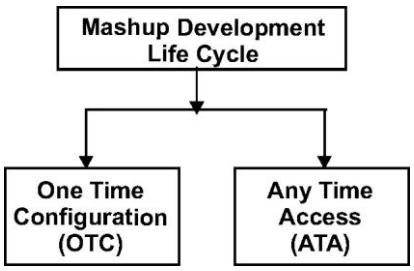

Figure 2. Mashup Development Life Cycle

We have elaborated one time configuration process using the OTC flow diagram as shown in figure 3(a) and any time access process using the ATA flow diagram as shown in figure 3(b). According to figure 3(a), user first chooses digital place to view his mashed up data. This digital place could be his personal page of social media account, email home or other mobile/desktop app. After determining the digital place for data mashup, user needs to define data mashup requirement in user friendly manner. Now, user selects some of the data sources which provide data for performing data mashup at its end.

Currently, not all the data sources are providing data for mashup purpose because of lack of popularity and security reasons hence choice of such data sources would be limited. In this paper, we have used the term Structured Data Server (SDServer) for mashup data sources which will provide structured data for performing structured data mashup and the term Structured Data Client (SDClient) for client interface (i.e. browser, apps etc) which would be used by end user to view the mashed up data. According to our previous work [4], user will fetch Mashup Configuration Attributes (MCA) from mashup data sources which would be required to perform pre-mashup configuration. Performing pre-mashup configuration is one time process which produces mashup configuration module which can further be divided into data module, service module, mapping module and UI module to understand the whole development process in a simplified manner.

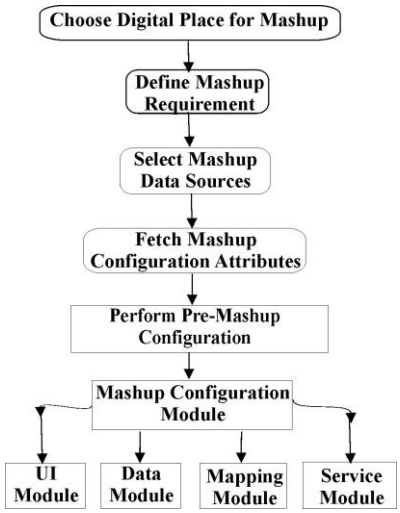

Figure 3(a). One Time Configuration Flow Diagram

The service module contains various components to integrate data services, protocols for mashup communication etc. The mapping module contains necessary configuration required for data filtering and transformation. The data module has necessary logic and other details to store & fetch mashed up data and UI module contains necessary codes and pre-defined interfaces/widgets to populate the mashed up data and used for showing it to its user in proper format. After completion of the first phase of OTC, user would be involved in second phase i.e. ATA. Figure 3(b) explores any time access flow diagram, which is required by user to perform necessary data mashup to fulfill its own data need.

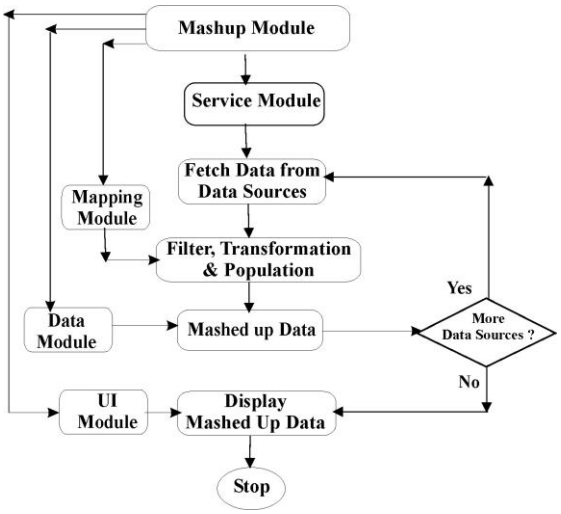

Figure 3(b). Any Time Access Flow Diagram

In order to perform data mashup, the user selects mashup module which contains data module, service module, mapping module and UI module which would be used for different purposes. Mashup module explores service module

© 2018, IJCSE All Rights Reserved

by calling various mashup services which were configure during OTC process. The service module further calls various data services and fetches required data from them and passes it to filter and transformation components. This component needs mapping module to complete its operation of data transformation as per rules defined by user. This component generates mashed up data which is stored in Structured Data Mashup Box (SDMB) with the help of data module and thereafter checks whether there are more data sources in the queue or not. The process of fetching, filtering, transformation and populating of the data continues till there are more data sources. At last, the mashed up data is shown to the user using UI module which was configured at the time of OTC. In this current work, we have focused mapping module called Structured Data eXchange Mapping (SDXMapping) which is briefly explained in the next section. More about SDXMapping can be found in our previous work [4].

IV. STRUCTURED DATA EXCHANGE MAPPING

The Structured Data eXchange Mapping (also called SDXMapping) is the core process of mashup used for structured data mashup. Structured Data (SD) is well defined at both ends which may be called Structured Data Client (SDClient) and Structured Data Source/Server (SDServer). The attributes and their schemas of the structured data are two important factors which are needed to define user's data requirement. The schema mapping and data mapping are two important aspects of structured data [38][39]. The mapping module in this work creates SDXMapping which can be further divided into two basic types based on two important factors called attributes and their schemas as mentioned above (See figure 4).

i. Structured Data Attribute eXchange Mapping i.e. SD(A)XMapping
ii. Structured Data Schema eXchange Mapping i.e. SD(S)XMapping

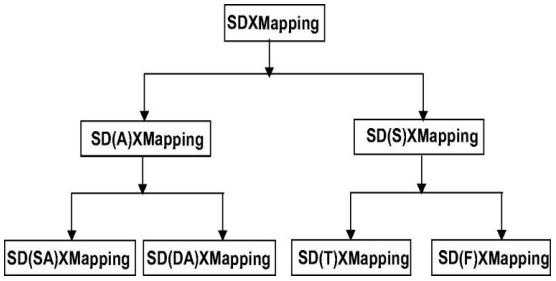

Figure 4. Types of SDXMapping

As shown in figure 4, SD(A)XMapping can be sub divided into Structured Data Simple Attribute eXchange Mapping i.e. SD(SA)XMapping and Structured Data Dependent Attribute eXchange Mapping i.e. SD(DA)XMapping. Similarly, SD(S)XMapping can be subdivided into Structured Data Type eXchange Mapping i.e. SD(T)XMapping and Structured Data Format eXchange Mapping i.e. SD(F)XMapping because schema of the data describes its type and format necessary to store and process it. In this paper, we are concerned with SD(A)XMapping only and rests of the mappings are left for the future work. Let us understand these types of SD(A)XMappings one by one.

A. SD(SA)XMapping

In this section, we will understand the basic differences between Simple Attribute (SA) and Dependent Attribute (DA) by taking an example. The simple attributes are those attributes whose values can be independently assigned by the user but dependent attributes are those attributes whose value can be chosen from set of some prefixed values. For example, a student can be defined by attributes like name, address, mobileNo, city and country. Here, name, address and mobileNo would be simple attributes which can be assigned values independently but attributes like city and country would be dependent attribute because their values can be chosen from set of prefix values provided by the IT developers. Let us understand two terms Mashup Configuration Attributes (MCA) and Data Mashup Definition (DMD) before understanding simple attributes mapping shown in figure 5. According to our previous work [4], MCA is the set of attributes published by SDServer so that SDClient can use it for performing pre-mashup configuration whereas DMD is the set of attributes defined by user at SDClient during defining its data requirement.

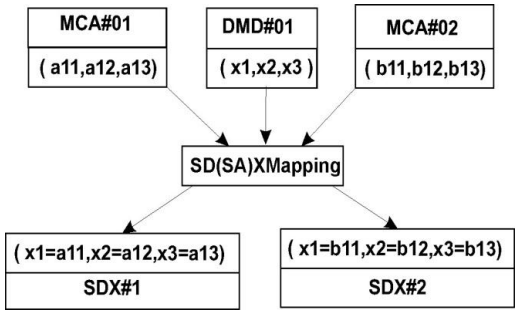

Figure 5. SD(SA)XMapping

As shown in figure 5, MCA#01 and MCA#02 are mashup configuration attributes published by two different SDServers. DMD#01 is the way of defining user's data requirement at his own digital place before performing the actual data mashup. SD(SA)XMapping algorithm as shown in Algorithm 1 is used to perform pre-mashup configuration to produce Structured Data eXchange Mapping for simple attributes. SDX#1 and SDX#2 are two different mappings created for two SDServers i.e. SDServer#01 and SDServer#02 respectively. This kind of mapping is simple and based on similarity of the semantical meaning of the attributes defined at both ends.

© 2018, IJCSE All Rights Reserved

The algorithm for performing one time configuration for simple attributes as well as dependent attributes is given in algorithm 1. This algorithm requires SDMB p and SDServer q to access attributes of DMD and MCA.

Algorithm-1:

OTC (SDMB p, SDServer q)

```
1: begin[1]
2: r = p.getDMDAttributes()
3: s = q.getMCAttributes()
4: For each MCAttribute t ∈ s
5:         begin[2]
6:         u = selectDMDAttribute(r,t)
7:         if(u is SA)
8:         p.SDXMapping.add(u=t)
9:         if(u is DA and t=DA)
10:        p.SDXMapping.add(u*=t*)
11:        Call OTC-DA(p,u*,t*)
12:        end[2]
13: end[1]
```

Line 2 & 3 read all the DMDAttributes & MCAttributes respectively. Each MCAttribute is corresponding to one DMDAttribute which is symantically equal to each other. Line 4 to 12 executes the loop for peforming SDXMapping for all MCAttributes received from SDServer. Line 6 selects DMDAttribute which is symantically equal to MCAttribute. Line 7 to line 11 calls SDXMapping based on type of the DMDAttribute i.e. simple attribute or dependent attribute. If DMDAttribute and MCAttributes are dependent attibutes then another algorithm 2 is called which has been developed for performing one time configuration between depdendent attributes.

B. SD(DA)XMapping

This section describes SDXMapping between dependent attributes of SDClient and SDServer. It can be seen from figure 6 that MCA#01 holds dependent attribute a14* whereas MCA#02 holds dependent attribute b14*. Here * has been used with an attribute to show that it is a dependent attribute not a simple attribute. It should noted here that data mashup definition must also contain dependent attibute before performing mapping between them. Here, DMD#01 defines x14* as a dependent attribute. Another important point can be noted that there is no direct mapping between two dependent attributes a14* and b14* rather mapping between their dependent values are used to create SDXMapping. The values of a14* can be assigned from one of the value from set (u1,u2,u3) and value of b14* can be assigned from set (v1,v2,v3). The dependent attribute x14* contains one of the value from set (z1,z2,z3).

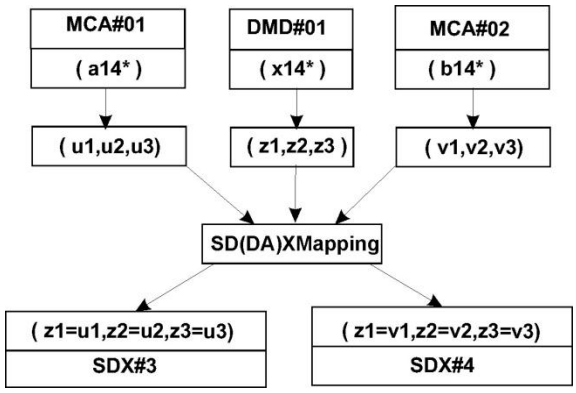

Figure 6. SD(DA)XMapping

Thus, SD(DA)XMapping generates mapping between all possible values of dependent attributes. SDX#3 shows mapping between DMD#01 and MCA#01 whereas SDX#4 shows mapping between DMD#01 and MCA#02. The rule for mapping used in this mapping is same as that of simple attributes.

The algorithm for performing one time configuration for dependent attribute is given in algorithm 2. This algorithm requires SDMB p and dependent DMDAttribute u* and MCAttribute *t.

Algorithm-2:

OTC-DA (SDMB p, DMDAttribute u*, MCAttribute t*)

```
1: begin[1]
2: a = u*.getDMDValues()
3: b = t*.getMCAValues()
4: For each value c ∈ a
5:         begin[2]
6:         d = selectMCAValue(b, c)
7:         p.SD(DA)XMapping.add(c=d)
8:         end[2]
9: end[1]
```

Line 2 & 3 read values of all the DMDAttributes & MCAttributes respectively. Line 4 to 8 executes the loop for peforming SD(DA)XMapping between value of MCAttribute and that of DMDAttribute. Line 6 selects MCAValue which is symantically equal to one of the value of the DMDAttribute.

C. SD(A)XMapping at a Glance

This section explains whole SD(A)XMapping process at a glance. It is clear that from figure 7 that simple attributes are

© 2018, IJCSE All Rights Reserved

passed to SD(SA)XMapping and dependent attributes are sent to SD(DA)XMapping.

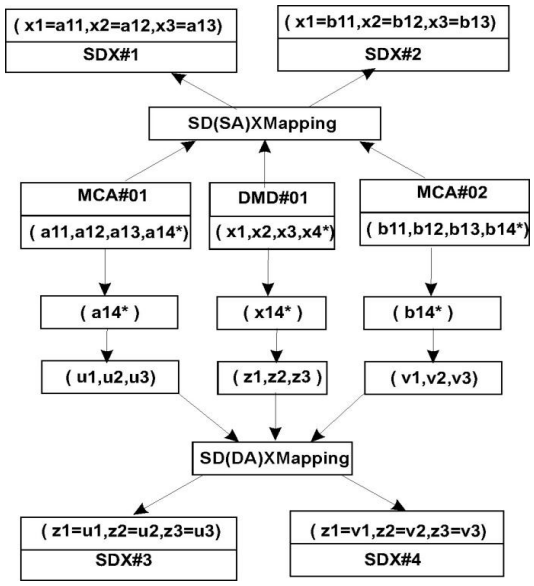

Figure 7. SD(SA)XMapping & SD(DA)XMapping

SDX#1 and SDX#2 have been created using simple attribute mapping algorithm between MCA#01 & DMD#01 and MCA#02 & DMD#01 respectively. Similarly, SDX#3 and SDX#4 have been created using dependent attribute mapping algorithm between MCA#01 & DMD#01 and MCA#02 & DMD#01 respectively.

V. MASHUP CONFIGURATION STYLES

The SDXMapping as explained above needs to be configured at either side or both side i.e. client side or server side. Reference [40] described various interface required for performing mashup using two popular mashup styles namely client side mashup and server side mashup. Reference [28] presented a novel mashup approach based on configuration theory and visual tool. Based on the location of performing the SDXMapping, mashup configuration can be of following styles-

A. Server Side (1:n) Mashup Configuration

B. Client Side (1:n) Mashup Configuration

C. Server Side (m:1) Mashup Configuration

D. Client Side (m:1) Mashup Configuration

E. Hybrid (m:n) Mashup Configuration

Here the term (m:n) indicates that there are m number of SDClients at client side and n numbers of SDServers at server side taking participation in data mashup. Let us understand these configuration styles one by one in the following sections.

A. Server Side (1:n) Mashup Configuration

In this type of configuration style, there exist one SDClient i.e. SDClient#1 at client side and n SDServers i.e. SDServer#1,SDServer#2,..,..,SDServer#n etc and SDXMapping is performed at server side. Hence, it is called Server Side (1:n) Mashup Configuration which can be seen in the figure 8.

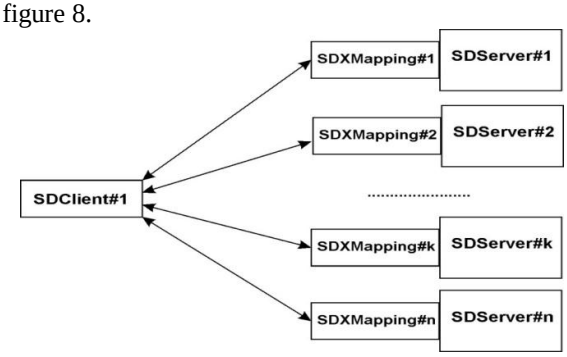

Figure 8. Server Side (1:n) Mashup Configuration

This style takes less time to perform data population at client side but takes more time for mashup communication because SDXMapping is performed at server side. Here, SDXMapping is performed at server side due to which SDClient gets mashed up data in its own format which can be directly populated into SDMB without performing filter and transformation.

B. Client Side (1:n) Mashup Configuration

In this type of configuration style, there exist one SDClient i.e. SDClient#1 at client side and n SDServers i.e. SDServer#1,SDServer#2,..,..,SDServer#n etc and SDXMapping is performed at client side. Hence, it is called Client Side (1:n) Mashup Configuration which can be seen in the figure 9.

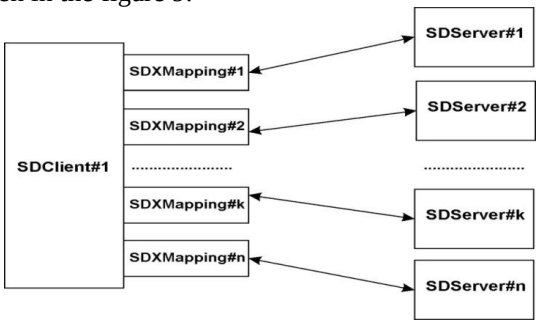

Figure. 9. Client Side (1:n) Mashup Configuration

This style takes more time to perform data population at SDClient but takes less time for mashup communication because SDXMapping is performed at client side. Here, SDXMapping is performed at client side for each mashup communication and hence data received from SDServer cannot be directly populated into SDMB and is passed to filter and transformation components for further process.

© 2018, IJCSE All Rights Reserved

C. Server Side (m:1) Mashup Configuration

In this type of configure style, there exist m SDClients i.e. SDClient#1,SDClient#2,..,..,SDClient#m at client side and one SDServer. SDXMapping is performed at server side and hence, it is called Server Side (m:1) Mashup Configuration which can be seen in the figure 10.

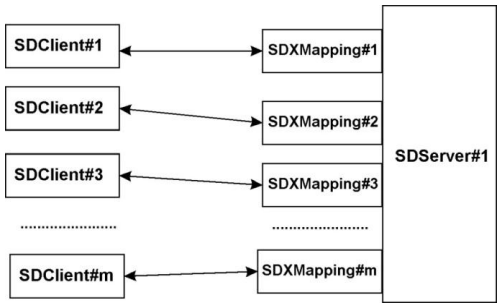

Figure 10. Server Side (m:1) Mashup Configuration

SDServer in this style takes more time to perform SDXMapping because SDXMapping is performed at SDServer for each SDClient connected to it and time taken for mashup communication for each SDClient depends on simultaneous execution of SDXMapping of SDClients at a time in the SDServer. Again, because SDXMapping is performed at server side hence data sent by SDServer can be directly populated into SDMB.

D. Client Side (m:1) Mashup Configuration

In this type of configuration style, there exist m SDClients i.e. SDClient#1,SDClient#2,..,..,SDClient#m at client side and one SDServer i.e. SDServer#1 and SDXMapping is performed on each SDClient. Hence, it is called Client Side (m:1) Mashup Configuration which can be seen in the figure 11.

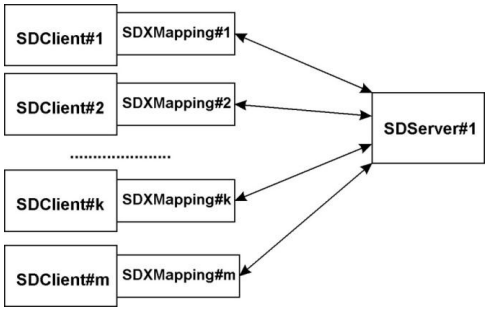

Figure 11. Client Side (m:1) Mashup Configuration

Time taken by SDServer to send data to its SDClients depends on the simultaneous request executed on it and each SDClient consumes some time for filter, transformation through SDXMapping before populating data into its SDMB.

E. Hybrid(m:n) Mashup Configuration

Hybrid (m:n) mashup configuration shows the hybrid combination of mashup configuration where some SDXMappings are performed at SDClients and some are performed at server side as can be seen from figure 12.

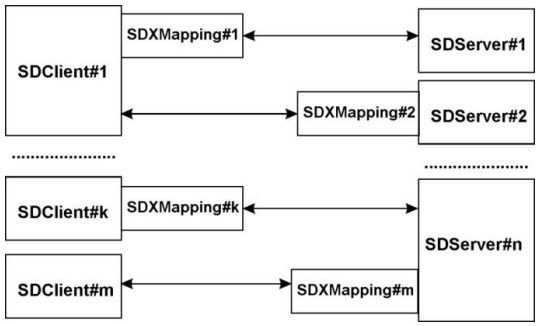

Figure 12. Hybrid Side (m:n) Mashup Configuration

Location of performing SDXMapping depends on the mutual understanding between SDClient and SDServer in the real world. The SDXMapping at client side is more secured for end users because it does not disclose DMDAttributes to anyone but user needs to update SDXMapping if any change is occurred in MCAttributes of SDServer. The SDXMapping at server side may not be secured because it discloses DMDAttributes of the end user but user does not need to perform any update on SDXMapping if any change is occurred in MCAttributes of the SDServer. The SDXMapping needs to be updated if there is change in either DMDAttributes or MCAttributes or both. Updates on SDXMapping can be performed by either of the side on mutual understanding but SDXMapping should be updated well before performing data mashup at SDClient. Synchronization of SDXMapping is another topic for researchers and can be explored as future work.

VI. CONCLUSION

Data mashup is the general kind of mashup which deals with all kinds of data whereas structured data mashup is the special kind of data mashup, which fetches only well structured data from multiple data sources and integrate all of them to populate into mashup box. In this paper, we divided the whole mashup development life cycle into two models called one time configuration and any time access model. The pre-mashup configuration process of one time configuration model creates mashup module which consists of data module, service module, mapping module and UI module to perform different task. All these mashup modules are used by any time access model for performing data mashup at various stages. The SDXMapping, which is the output of pre-mashup configuration process, has also been explained using the concept of simple attribute's mapping and dependent attribute's mapping through algorithms. We have designed

© 2018, IJCSE All Rights Reserved

multi-mashup configuration styles using SDXMapping to fulfil different situational need of data mashup, which will be helpful for both the stakeholders called mashup developer and end user for developing structured data mashup.

REFERENCES

[1] Fung, Benjamin CM, Thomas Trojer, Patrick CK Hung, Li Xiong, Khalil Al-Hussaeni, and Rachida Dssouli, "*Service-oriented architecture for high-dimensional private data mashup",* IEEE Transactions on Services Computing 5, no. 3, pp. 373-386, 2012.

[2] Lee, Yong-Ju, "*Semantic-Based Web API Composition for Data Mashups",* J. Inf. Sci. Eng. 31, no. 4, pp. 1233-1248, 2015.

[3] Yu, Daniel, "*Efficient Processing of Almost-Homogeneous Semi-Structured Data",* Master's thesis, 2018.

[4] Hardaha, Prakash, Shailendra Singh, "*Structured Data REST Protocol for End to End Data Mashup",* Future Internet10, no. 10, 2018.

[5] Beemer, Brandon, and Dawn Gregg, "*Mashups: a literature review and classification framework*.", Future Internet 1.1: 59-87,2009.

[6] Taivalsaari, Antero, "*Mashware: The future of web applications"* 2009.

[7] Desolda, Giuseppe, Carmelo Ardito, Maria Francesca Costabile, and Maristella Matera, "*End-user composition of interactive applications through actionable UI components*", Journal of Visual Languages & Computing 42 : 46-59,2017.

[8] Paternò, Fabio, "*End user development: Survey of an emerging field for empowering people*", ISRN Software Engineering 2013.

[9] Fischer, Thomas, Fedor Bakalov, and Andreas Nauerz, "*An Overview of Current Approaches to Mashup Generation*", In Wissensmanagement, pp. 254-259. 2009.

[10] Vancea, A.; Grossniklaus, M.; Norrie, M, "*Database-Driven Web Mashups*", International Conference On Web Engineering, Yorktown Heights, New York, pp. 162-174, 2008.

[11] Easton, Annette, and George Easton, "*Demystifying Mashups*", In Proceedings of Informing Science & IT Education Conference (InSITE), pp. 479-486, 2010.

[12] Gao, Lianli, Peng Wang, Jingkuan Song, Zi Huang, Jie Shao, and Heng Tao Shen, "*Event Video Mashup: From Hundreds of Videos to Minutes of Skeleton*", In Thirty-First AAAI Conference on Artificial Intelligence, 2017.

[13] Meerwaldt, Rick, Albert Meroño-Peñuela, and Stefan Schlobach, "*Mixing Music as Linked Data: SPARQL-based MIDI Mashups*", In Workshop on Humanities in the Semantic Web-WHiSe II (ISWC 2017), 2017.

[14] Di Lorenzo, Giusy, Hakim Hacid, Hye-young Paik, and Boualem Benatallah, "*Data integration in mashups*", ACM Sigmod Record 38, no. 1,pp. 59-66, 2009.

[15] Zaka, Bilal, Christian Safran, and Frank Kappe, "*Personalized Interactive Newscast (PINC): Towards a multimodal interface for personalized news*" In Semantic Media Adaptation and Personalization, IEEE Second International Workshop on, pp. 56-61, 2007.

[16] WALES, NEW SOUTH, "*Mashups for Data Integration: An Analysis*", 2008.

[17] Boulakbech, Marwa, Nizar Messai, Yacine Sam, and Thomas Devogele, "*Visual Configuration for RESTful Mobile Web Mashups*", In Web Services (ICWS), 2017 IEEE International Conference on, pp. 870-873, 2017.

[18] Heath, Tom, and Christian Bizer, "*Linked data: Evolving the web into a global data space*", Synthesis lectures on the semantic web: theory and technology 1, no. 1,pp. 1-136, 2011.

[19] Ruback, Livia, Marco Antonio Casanova, Alessandra Raffaetà, Chiara Renso, and Vania Vidal, "*Enriching Mobility Data with Linked Open Data*", In Proceedings of the 20th ACM International Database Engineering & Applications Symposium, pp. 173-182, 2016.

[20] Tran, Tuan Nhat, Duy Khanh Truong, Hanh Huu Hoang, and Thanh Manh Le, "*Linked data mashups: a review on technologies, applications and challenges*", In Asian Conference on Intelligent Information and Database Systems, pp. 253-262. Springer International Publishing, 2014.

[21] Zook, Matthew, and Jessica Breen, "*Mapping Mashups*", The International Encyclopedia of Geography, 2017.

[22] Ennals, Rob, Eric Brewer, Minos Garofalakis, Michael Shadle, and Prashant Gandhi, "*Intel Mash Maker: join the web*", ACM SIGMOD Record 36, no. 4, pp. 27-33, 2007.

[23] Hira, Shrabanti, and S. M. Labib, "*Conceptual study of Web-based PPGIS for Designing Built Environment: Identifying Housing Location Preferences in Littleborough*", 2017.

[24] Hoyer, Volker, and Marco Fischer, "*Market overview of enterprise mashup tools*", Service-Oriented Computing–ICSOC 2008, pp. 708-721,2008.

[25] Patel, Ahmed, Ibrahim AlShourbaji, and Samaher Al-Janabi, "*Enhance business promotion for enterprises with mashup technology*", Middle-East Journal of Scientific Research 22, no. 2 pp. 291-299, 2014.

[26] Ngu, Anne HH, Michael P. Carlson, Quan Z. Sheng, and Hye-young Paik, "*Semantic-based mashup of composite applications*" IEEE Transactions on Services Computing 3, no. 1,pp. 2-15, 2010.

[27] Ma, Shang-Pin, Chun-Ying Huang, Yong-Yi Fanjiang, and Jong-Yih Kuo, "*Configurable RESTful Service Mashup: A Process-Data-Widget Approach*", Appl. Math 9, no. 2L, pp. 637-644, 2015.

[28] Im, Janggwan, Seonghoon Kim, and Daeyoung Kim, "*Iot mashup as a service: Cloud-based mashup service for the internet of things*", In Services Computing (SCC), 2013 IEEE International Conference on, pp. 462-469, 2013.

[29] Barhamgi, Mahmoud, Chirine Ghedira, Djamal Benslimane, S-E. Tbahriti, and Michael Mrissa, "*Optimizing daas web service based data mashups*", In Services Computing (SCC), 2011 IEEE International Conference on, pp. 464-471, 2011.

[30] Madhavan, Jayant, Loredana Afanasiev, Lyublena Antova, and Alon Halevy, "*Harnessing the deep web: Present and future*", arXiv preprint arXiv:0909.1785, 2009.

[31] Hornung, Thomas, Kai Simon, and Georg Lausen, "*Mashups over the deep web*", In International Conference on Web Information Systems and Technologies, Springer, Berlin, Heidelberg, pp. 228-241, 2008.

[32] Auinger, Andreas, Martin Ebner, Dietmar Nedbal, and Andreas Holzinger, "*Mixing content and endless collaboration–MashUps: Towards future personal learning environments*", In International Conference on Universal Access in Human-Computer Interaction, Springer, Berlin, Heidelberg, pp. 14-23, 2009.

[33] Jarrar, Mustafa, and Marios D. Dikaiakos, "*A query formulation language for the data web*", IEEE Transactions on Knowledge and Data Engineering 24, no. 5, 783-798, 2012.

[34] Simmen, David E., Mehmet Altinel, Volker Markl, Sriram Padmanabhan, and Ashutosh Singh, "*Damia: data mashups for intranet applications*", In Proceedings of the 2008 ACM SIGMOD international conference on Management of data, pp. 1171-1182, 2008.

[35] Bouguettaya, Athman, Surya Nepal, Wanita Sherchan, Xuan Zhou, Jemma Wu, Shiping Chen, Dongxi Liu, Lily Li, Hongbing Wang, and Xumin Liu, "*End-to-end service support for mashups*", IEEE Transactions on Services Computing 3, no. 3, 250-263.2010.

[36] Kang, Jaewoo, and Jeffrey F. Naughton, "*On schema matching with opaque column names and data values*", In Proceedings of

© 2018, IJCSE All Rights Reserved

the 2003 ACM SIGMOD international conference on Management of data, pp. 205-216, 2003.

[37] Tuchinda, Rattapoom, Craig A. Knoblock, and Pedro Szekely, "*Building mashups by demonstration*", ACM Transactions on the Web (TWEB) 5, no. 3,16, 2011.

[38] Guo, Mingchuan, and Yong Yu, "*Mutual enhancement of schema mapping and data mapping*", In International Workshop on Mining for and from the Semantic Web, p. 88, 2004.

[39] Mecca, Giansalvatore, Paolo Papotti, and Donatello Santoro, "*Schema Mappings: from Data Translation to Data Cleaning*", In A Comprehensive Guide Through the Italian Database Research Over the Last 25 Years, Springer, Cham, pp. 203-217, 2018.

[40] Salminen, Arto, and Tommi Mikkonen, "*Mashups-Software Ecosystems for the Web Era*", In IWSECO@ ICSOB, pp. 18-32, 2012.

Authors Profile

Mr. Prakash Narayan Hardaha is pursing Ph.D. (Computer Science & Engineering) from Barkatullah Univeristy, Bhopal and currently working as System Analyst in the Department of Computer Engineering and Applications, National Institute of Technical Teachers Training and Research, Bhopal. His main research work focuses on code generation, data mashup, structured data engineering, web technolgy and integrated UI development. He has 12 years of teaching experience and 6 years of industry experience.

Dr. Shailendra Singh received his Ph.D. degree in Computer Science and Engineering from Rajiv Gandhi Proudyogiki Vishwavidyalaya, Bhopal, India in 2010. He is currently working as Professor in the Department of Computer Engineering and Applications, National Institute of Technical Teachers' Training and Research, Bhopal-462002, India. His research interests include internet technologies, intrusion detection systems using machine learning techniques & Support Vector Machine. He has published more than 75 research papers in the international journal and conferences of repute and presented many papers on international/national level conference/seminars. He has more than 21 years of teaching experience and 2 years of industry experience. He is also a senior member of IEEE.

© 2018, IJCSE All Rights Reserved