ISSN:2320-0790

An International Journal of Advanced Computer Technology

Mash up Candidate Prediction: A Survey on Mashup Techniques, Tools and Framework

P.Suganya[1], S. Lavanya[2], E.Vijayavani[3] , E.Elakiya[4]

[1,2,3,4]Assistant Professor, Department of Information Technology, EGS Pillay Engineering College, Nagapattinam, Tamilnadu, India

Abstract-The evolution of web 2.0 introduces the complementary features of service composition which focuses on community and usability of web services. The increasing number of applications on the web and a growing need to combine them in order to meet user requirements.. Mashup is the process of assimilating web services for generating new services; it extracts data from various resources like PDFs, databases, legacy systems, and web applications. Before performing Mashup, the possible candidates for aggregation should be generated. In dynamically changing internet scenario, predicting service Mashup candidates are tedious one. This paper uses Syntactic technique to predict candidates and used for determine the equivalences among the services with reasonable precision, and it also analyzes the naming tendency of web service developers. The result makes the service search process to identify candidate services faster. This paper deals with the design of client side Mashup architecture with viable candidates for aggregating services. The system has a pallet of services that are clustered by their input and output. It would involve service connection, composition and data visualization. This framework allows users to play with services.

Keywords: Service Oriented Computing, Service Mashup, Distributed Services

I. INTRODUCTION

The emerging phenomenon of web 2.0 describes the new characteristics of web. It demonstrates that the end users have keen interest in developing services through different static services available on web such as wikis and social networking sites. Customizable web feeds are also gaining popularity to create personalized web pages containing information feeds and gadgets.

The customizable web portals are easy to use but they do not support the creation of advanced application, because the software services and data repositories cannot be combined with each other. From O'Reilly's Dale Daugherty description about the web 2.0, the web experiences that fundamentally engages users who have no significant computing knowledge and experience. by: (a) Allowing them to participate in sharing information and enriching data freely; (b) readily offering their core functionality as an open services to be composed or "mashed up" into new services and sites; (c) placing the web at the center of software experience both in terms of data location and where the software is[I tech viewpoint].

To achieve the goal, web 2.0 introduces new design pattern and architectural styles to ensure the user community in development of web. They are Service Oriented Architecture (SOA), MigrAtion to Service Harmonization compUting Platform(MASHUP).

SOA visualizes web of service made up of integrating resources and it empowers the end users to ubiquitously exploit these resources by collaboratively remixing them. Services in SOA are loosely coupled and changing software development approach from traditional "product centric" manufacturing to "consumer centric" service composition. There are three issues in SOA. They are; (a) SOA requires experts in tools and environment; (b) not allow on the fly composition and (c) not support legacy and existing system in service composition.

Mashup combines distributed resources of services and contents on the presentation layer of network model into new composite web application. Mashup compose application from reusable parts and it integrates services with different functionality but similar operation contents can be executed together. It makes consumer, free to compose services as they wish as well

as simplifies the composition tasks. It is simpler, more cost effective, self served approach for service composition. When compared to SOA Mashup provides user centric application than the programmer centric.

This survey paper tries to explore the research area of Mashup, classes of Mashup, Mashup architecture and tools supported, auto completion debugging, Mashup accountability and Mashup metrics.

II. CLASSES OF MASHUP

Nowadays information technology moves towards web 2.0 architecture. The Mashup supports ad-hoc web application integration. Mashup is classified into different varieties with different characteristics based on number of Users, Pages and Workflow.

Before actually talking about the types of Mashup, the characteristics of multiuser, multipage and workflow are being investigated. The business process contain the workflow; an executable part of a process that consist of several activities and defines a series of tasks that need to be managed by different source.[tow]. The multiuser refers, multiple users being allowed to simultaneously access the instance of Mashup. Multi page describes that the implemented Mashup provide multiple page navigation in hierarchical structure.

III. TYPES OF MASHUP

a. SIMPLE MASHUP**:**

This Mashup type exclusively addresses single user and the Mashup is implemented in single page with no workflow. This type of Mashup is supported by mashArt platform which comes with models, language, composition paradigm and users are allowed to abstract from low level implementation details and compose within the same development environment. Tools support this Mashups are yahoo pipes and Intel mash maker.

b. MULTIPAGE MASHUP

This Mashup type allows single user and multipage navigation with no workflow. The Mashup composition direct to multiple page view on mashed data. EzWeb platform support this type of composition by wiring gadgets. Gadgets consist of multiple screens. The connection between screens is not explicitly modeled but automatically generated based on mapping of their input, output and semantics. The tool supporting this multipage Mashup is FAST(Fast and Advanced Storyboard Tool).

c. GUIDED MASHUP:

This absolutely implements the single user and single page navigation Mashup. It provides control flow for the Mashup architecture, and requires user guidance to aggregate services. As per the survey any knowledge about the tool is not obtained.

d. PAGE FLOW MASHUP:

This Mashup type addresses single user, multipage routing with control flow. *ServFace Builder* is created based on this platform. It supports non IT people in the design and creation of service based on interactive application in a *WYSIWYG*. Applications are created as a set of pages that can be connected to create a navigation flow.

e. Shared Page Mashup:

This kind of Mashup deals with multiple users and single page with no control flow. Upto this survey still there is no tool to support this type of integration.

f. Shared space Mashup:

This Mashup concentrates on multi user and multipage navigation with no workflow control. IBM Mashup Center is a collection of tools that supports Eclipse and allows user to create enterprise Mashup.

g. Co-operative Mashup:

This kind of Mashup focuses on multi user and multipage routing with workflow. Gravity is a lightweight collaborative and client targeting platform which focuses on non IBM experts to create immediate application based on business process modeling.

h. PROCESS MASHUP

This category of Mashup concerns multiuser and multipage navigation with workflow. *MarCoFlow* platform supports application development approach that allows one to bring together UIs, Web Services and People in a single Orchestration logic, language and tool.

IV. MASHUP SERVICE COMPOSITION

This will explain the service composition through Mashup. Two different approaches are used in end user Mashup. They are passive and proactive.
Passive approach designs widgets and suggest potential sources for Mashup, it encourages creation of new service by end user without the need of new programs and permits local data in aggregation.
Proactive approach is a complicated one. Mashup environment should first provide some examples of Mashup which the end user likes, and then exposes the

end result. Proactive approachs are used by end users who have no programming knowledge.

Mashup uses widgets for service composition. Widgets are small client side application for offering atomic functionalities of an enterprise application packaged in a way to allow a single downloading and installation on a client machine, mobile phone or mobile Internet devices. The drag and drop mechanism combined with the widget concept enables enterprise applications to collaborate easily, even if they are developed independently from each others. This mechanism belongs to the semiautomatic service composition category, which is performed by the end user actions [12]. Mashup supports the following characteristics for service composition [5]

1 Leveraging web as the design-time and runtime tool for service composition, so as to significantly reduce overhead to composite service consumers
2 On-the-fly customization and deployment to make the service composition to be more responsive for consumer's requirement changes
3 Easy reuse and remix of existing applications and data which can be accessed through the web.

Below Figure 1 represents the generic Mashup, pulling sources from web application like e-mail, excel files, PDF's etc. which are fed into user browser for creating new service and visualized to user.

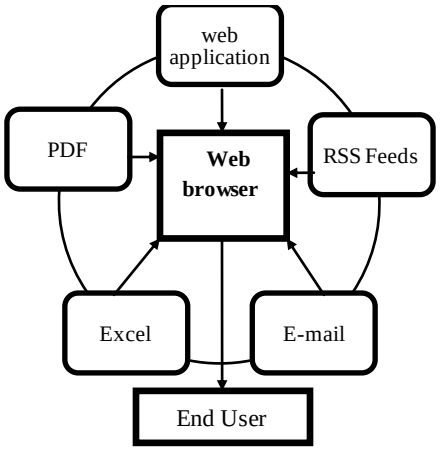

Figure 1. Service Oriented composition

V. MASHUP FRAMEWORK

[2]Analyzed the Mashup framework comprised of three different participants: API/content providers, the Mashup hosting site and the consumer's web browser. Figure 2 describes the Mashup architecture.

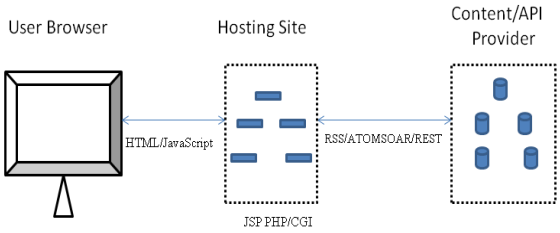

Figure 2. Mashup Architecture [2]

Mashup extracts content from web sites like Wikipedia, PDFs, TV guides, Excel, E-mail etc which are called as API/content provider. Widgets are used to aggregate services on user browser. The screen scraping technique is used in content extraction process [2].

The Mashup hosting site [2] refers to the area where mashed contents are hosted. In general, hosting site contains the Mashup logic. The client side logic is often the combination of code directly embedded in the Mashup web page as well as scripting API libraries or applets referenced by the web pages.

The consumer's web browser [2] is where application is rendered graphically and where the user interaction takes place.

A. GENERIC REQUIREMENT FOR COMPOSITION SYSTEMS:

Some aspects are needed to be defined in order to describe software composition system. The component based software architecture should contain Component model, Composition Technique and Composition Language.

VI. MASHUP COMPONENT MODEL

From the Mashup view, [2] web is no longer represented as a markup document, but a data driven application. Therefore, there must be a well-defined component model that can encapsulate the data from multiple sources and manipulate the existing web resources through the standard services (REST, ATOM/RSS, and so on).

[2] Classified Mashup component model into three models as shown in Figure 3 The components are UI Component, Service Component, and Action Component.

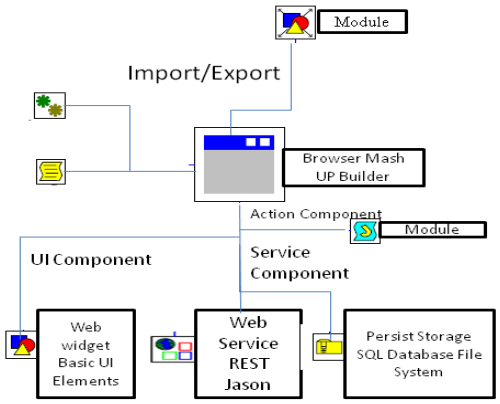

Figure 3. Mashup Component model [2]

UI Component: UI components are represented as a set of widgets in the browser (a window, a button, a drop-down list, etc). Enhanced by AJAX, UI components and its binding service component can be connected and updated dynamically. UI component masks the service components details to the consumer so as the composition is done at UI level. In other words, to consumers, UI component is the unique entity that survives in the Mashup applications.

Service Component: Service component represents data manipulation interface which will contain the data content, for example, a web service interface, which can be accessed by SOAP and REST; or it can be a DB interface, which can retrieve and store data in local or remote database. Data standardization is achieved in simple script by web container or service container. In our current implementation, the service component is mainly the web services or services with open APIs (such as Google Map).

Action Component: Action component acts like the connector between UI components and service components. For example, it defines an action driven by events (e.g., onClick or onMouseOver). It can be an action which changes the display value of a UI component, or one that invokes a service component interface.

FEATURES OF COMPOSITION MODEL FOR USER CENTERED SERVICE BASED APPLICATION:

a. SERVICE DISCOVERY:

It needs component discovery. The three approaches followed in component discovery are based on metadata, global catalog and register.

b. SERVICE INVOCATION MECHANISM:

This mechanism should have the ability to combine service from diverse sources and the inputs have to be put in respective services and translated back as the result.

Service orchestration and choreography: It relay on loosely coupled service, which do not call each other. The process built on top of the service provide coordination.

c. USER INTERFACE:

The application interacts with the user at all times through a set of interface elements.

d. PRESENTATION LOGIC:

Presentation logic is all the user interface-related logic that exploits context information for adaptation and customization purposes.

e. CHARACTERISTICS

Different elements of composition model require modularity, parameterizability and standard interface.

Composition Technique: It determines the available mechanism to compose the middle elements.

Features of composition technique: Connection: Component should connect to other component and it is necessary to adapt the components, parameter, protocol and assertions.

Extensibility: Automatically extending existing functionality and non functionality.

Aspect Separation: It covers functional and non functional features.

Scalability and Modeling: Scalable in binding time and technique.

f. COMPOSITION LANGUAGE

The language should be powerful and expressive enough to support any composition based software design process. It express variants and version. Language itself based on composition process.

VII. AUTO COMPLETION FOR MASHUP

In general Mashup contain several smaller components namely Mashlet. Mashlet is a module that may implements a specific functionality, data source, operator and support interface of variable and methods visible from other Mashlets. The state of the Mashlet is maintained and represented by a set of relation. The logic of the Mashlet is represented by a set of data log like rules. The main problem discussed in this paper is gluing which is non-trivial. The name of the Mashlet's input and output variables are not always meaningful or uniform. They include state variable are not always aware of inconsistency.

In today's scenario browsing through all to identify common and suitable wiring or gluing is too time consuming. Mashlet instantly retrieves those Glue Pattern(GP) that are potentially most relevant to the user's current needs. This paper identify two challenges They are: Identification of potentially relevant GP and Ranking of candidate GP.

A. IDENTIFICATION OF POTENTIALLY RELEVANT GP:

A good GP would glue all the Mashlets selected by the user without introducing additional Mashlets in the Mashup. It may relax the requirements of user. The solution is, the GP does not link to exact Mashlets, but instead links Mashlets that are similar to them.

B. RANKING OF CANDIDATE GP:

The rank depends on the "tightness". It penalizes the quality of candidates.

- Tightness of the GP with respect to inheritance relationship is important.
- Frequent GP tools rank higher even if they are a little less tight.
- This architecture takes collective wisdom of user community.

Mashlet leverage the programmer for understanding semantically. This takes advantages of the recent new phenomenon, massive volumes of developer's sharing experience.

VIII. DEBUGGING

Debugging is the process of executing Mashup, checking output, refining Mashup definition and executing again. Software engineering technology is not well supported because of the lack of support for interactive debugging. The solution for this problem is: Mashup definition is transformed into graph representation comprising of individual process steps and their dependencies. Based on the Mashup graph, developers may define the breakpoints to pause the state and the intermediate result.

To implement this graph, the developer should specify what data can occur in the Mashup result, the platform should suggest which point the Mashup definition is likely flawed and should specify the graphical debugging environment that executes the Mashup and indicates source of error.

The graph contains both control flow and data flow. This framework supports "undo" features which allows to pause the execution and to inspect the state of a running process. The execution path is used to resume the Mashup process starting from the most recently processed block.

IX. ACCOUNTABILITY

[3] Analyzing the accountability of Mashup architecture. However, in Mashup several sources require identification and these may need to be trusted sources in an accountability sense. Accountability in services refers to the obligation that several persons, groups or organizations assume for the execution and fulfillment of a service. This obligation includes [3]:

Answering which provides an explanation or justification for the execution of that authority and/or fulfillment of that responsibility. Full disclosure on the result of that execution and/or fulfillment Undeniable liability for those result(non-repudiation); and Obtain trusted agreement of accountability from all entities involved in the services that in turn are bound to the obligations set out above.

In the Mashup service scenario, client send request to a Mashup service environment, who in turn forwards the request to the service owner, before aggregating. This identifies two issues, the clients who are not known to the owner of the service and the owner of the service not aware how the content is aggregated. This leads to the disclosure of roles and responsibilities in Mashup service environment.

Disclosure of roles and responsibility, to a large extent, can be enabled by rich service metadata adding semantics to allow machine interpretation and reasoning [3]. and facilitated by functions provided.

Service provider is a special type of role in Mashup environment which plays both requester and provider at the same time. When sourcing content from a broker or service source, the provider acts as the requestor. The service source publishes a single or discrete set of content sources that may be accessed directly by the service requester, or can be built upon and merged with other content source by a Mashup service provider[3].

Figure 4 show a model defines the roles and responsibilities in service metadata. This model is useful for information systems developers, helping them to identify roles and responsibilities in accountable Mashup service solutions [3].

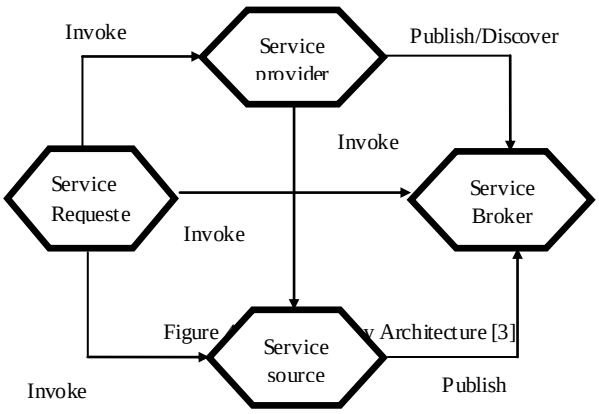

Figure [illegible] Architecture [3]

X. MASHUP METRICS

This paper discusses [6] the metrics that should be followed for Mashup. ISO 8402-86 standards define Quality as the totality of features and characteristics of a software product that relate to its ability to satisfy stated or implied needs. Software quality metrics are not suitable for Mashup. Quality metrics for Mashup depends on what content is mashed up, location Mashup, Mashup process.

There are three things that can be Mashup: presentation, data and functionality. Presentation Mashup focus on information and layout which in the form of widgets are dragged and dropped into a common interface. Metrics for this type of Mashup are size, style and color. All widget should be consistent [6].

Data Mashup integrate various source data into one target location. The metrics for this type of data are efficient connectivity of desperate mashed data and efficient modularization of Mashup [6].
Functionality Mashup create new services by integrating data and functionality of different services. The metrics for this type of services are Smooth access to the functionality of disparate mashed data, Efficient accessing of the Mashup [6].

The extraction Mashup can be considered as a data wrapper collecting and analyzing resources from different sources and merging the resources to one content page. For this type of Mashup the same metrics as for the presentation Mashup can be used [6].

In a flow Mashup the user customizes the resource flow of the Web page combining resources from different sources. For this type of Mashups, for instance, the metrics connectivity, availability of components and errors rates should be considered [6].

XI. SERVICE MASHUP CANDIDATE PREDICTION

It is a process of discovering service candidates for Mashup. KDS follows systematic approach to identify viable candidate other than semantic approach Because open services randomly available over the internet are not described in terms of semantics. Even in case if they use semantics they do not adhere to a common ontology which would unify semantics across disparate domains.

Although syntactical technique lacks the confidence of semantic approaches their flexibilities are an advantage in the open environment. It analyses the characteristics of the individual service and capture the naming tendency of developer.

KDS follows three steps to identify the candidates: Equivalence processing which identify services which are equivalent using direct and indirect information from service specification. There are three trends are followed to discover the naming tendency of developer are Subsumption Relation, Common subsets and Abbrevation.

Clustering integrates services capable for Mashup. Based on the following principles the services are clustered together

1 A group of web services have 1 or more related output parts.
2 A group of web services have any combination of equivalent parts whether those parts are associated with in input or output messages.
3 A group of services share a potential Mashup candidate with another web service whereas complementary data are effectively chained together.

Categorization and Filtering identifies value added services for Mashup from the clustering phase [1]. Categorization phase use Categorization On Pairing(COP) algorithm; it cluster services into categories based on the similarity of the specification. Filtering sort service based on add value to end user. The services within its own category have greater viability for candidate.

XII. CONCLUSION

The main focus of this survey is identifying research area in Mashup. This survey provides a study about component model, techniques and languages are already proposed. The concepts and fundamental principles of UI centric design are described. Application area of metrics for Mashups and current solution spaces are discussed

REFERENCES

[1] M. Brian Blake and /Michael F. Nowlan. "Knowledge Discovery in Services (KDS): Aggregating software Services to Discover Enterprise Mashups" in IEEE Transaction On Knowledge And Data Engineering, Vol. 23, No. 6, June 2011.

[2] Liu, Y. Hui, W. Sun, and H. Liang, "Towards Service Composition Based on Mashup," Proc. IEEE Congress on Services, pp, 332-339, July 2007.

[3] J. Zou and C.J. Pavlovski, "Towards Accountable Enterprise Mashup Services," IEEE International Conference on E-Business Engineering, pp 205-212, Oct. 2007.

[4] S. Cetin, N.I. Altintas, H. Oguztuzun, A. Dogru, O. Tufekci, and S.Suloglu, "A Mashup-Based Strategy for Migration to Service- Oriented Computing," Proc. IEEE Int'l Conf. Pervasive Services, 2007.

[5] M.B. Blake and M.F. Nowlan, "Taming Web Services from the Wild," IEEE Internet Computing, vol. 12, no. 5, pp. 62-69, Sept./Oct. 2008.

[6] Agnes Koschmider, Volker Hoyer, Andrea Giessmann, "Quality Metrics for Mashups," Proc. ACM SAICSIT.

[7] Jiawei Han and Michline Kamber, "Data Mining Concepts and Techniques," Second Edition, Morgan Kaufman Publishers, 2006.

[8] M.B. Blake, "Knowledge Discovery in Services," IEEE Internet Computing, vol. 13, no. 2, pp. 88-91, Mar. 2009.

[9] [9] Nassim laga, Emmanual Bertin, Noel Crespi, " A web Based Framework for Rapid Integration of Enterprise Applications," Proc., ACM ICPS'09, July.

[10] Hinchecliffe D.,"i-Technology viewpoint: is Web 2.0 the Global SOA?", SOA World Magazine, 2006.

[11] Michael Cooper, "Accessibility of Emerging Rich Web Technologies: web 2.0 and Semantic Web," ACM W4A2007 – Keynote, May, 2007.

[12] Arto Salminen, Tommi Mikkonen,Feetu Nyrhinen, Antero Taivalsaari, " Developing Client side Mashups: Experiences, Guidelines and the Road Ahead" Proc., ACM MindTrek, October 2010.

[13] http://www.startvbdotnet.com/sdlc/sdlc.aspx

[14] [14]http://codebetter.com/raymondlewallen/2005/07/13/software-development-life-cycle-models/

[15] Florian Daniel, Agnes Koschmider, Tobias Nestler, Marcus Roy, Abdallah Namoun," Toward Process Mashups:Key Ingredients and Open Research Challenges", ACM Mashups 2010,December 2010.

[16] Ohad Greenshpan, Tova Milo, Neoklis Polyzotis," Autocompletion for Mashups_", ACM VLDM Endownment August 2009.

[17] Waldemar Hummer, Philipp Leitner, and Schahram Dustdar," A Step-By-Step Debugging Technique To Facilitate Mashup

[18] Development and Maintenance", ACM Mashups 2010,December 2010.

[19] Rafael Fernández, David Lizcano, Sebastián Ortega, Javier Soriano," Towards a user-centered composition system for service-based composite applications", ACM iiWAS2009 , December 2009.