Abstract: Service mashups are the act of incorporating the causing data of two complementary software services into a common picture. Such an approach is promising with respect to the discovery of new types of knowledge. However, before service mashup routines can be executed, it is necessary to predict which services (of an open repository) are viable candidates. Similar to Knowledge Discovery in Databases (KDD), we introduce the Knowledge Discovery in Services (KDS) process that identifies mashup candidates. In this work, the KDS process is specialized to address a repository of open services that do not contain semantic annotations. In these situations, specialized techniques are required to determine equivalences among open services with reasonable precision.

Keywords - service mashup, web-based services, filtering, clustering, and categorization.

I. INTRODUCTION

SERVICE-ORIENTED COMPUTING (SOC) supports the notion of thousands (even millions) of modular software capabilities discoverable and usable by disparate organizations. Underlying SOC are specifications, such as the Web Service Description Language (WSDL), and the invocation methods supported by SOAP or REST that provide machine-interpretable interfaces and interactions. Such open approaches provide an environment where atomic modules (i.e., web services) can be discovered and composed on demand. This type of interaction, in a business setting, is optimal because it allows the underlying business units or divisions to act autonomously while, at the same time, facilitating active collaboration at a layer of granularity that each stakeholder can dictate and control. As such, a business unit can discover relevant capabilities then consume and compose them in terms of their existing capabilities to create entirely new offerings.

A service mashup is the simultaneous execution of two or more services to create an integrated data provision with a more complete description about some object or task. For example, web services from a web search company such as Google Corporation that provides mapping capability can be integrated with capabilities from a shipping business such as the United Parcel Service (UPS).

II. Allied Slants

Although the area of data integration has had a long-standing background, the use of these techniques to accomplish mashup has only just recently started to be addressed. A majority of the work in this area addresses the tools and environments that support the visualization and presentation of mashup.
III. KDS and KDD
The four phases of KDD, cleaning and integration, selection and transformation, data mining, and evaluation and presentation can be compared to the KDS phases of discovery, equivalence processing, clustering, categorization, filtering, and presentation. The KDS process is shown together with the KDD process. The Cleaning and Integration step in the KDD process assumes that noise must be removed from initial data in parallel with the integration of multiple data sets.

Fig – 2 ARCHITECTURE

In contrast, the integration of multiple repositories in KDS’s Discovery phase is aided by common standards for storing services (e.g., Universal, Description, Discovery, and Integration (UDDI)). Discovery in KDS also relates to the extraction of parts and other descriptive service-based information (such as service name, operation name, type name, descriptive fields, etc.) from the service specification. Even before considering equivalent services, this step attempts to identify equivalent parts. Similar to KDD’s Selection phase, the KDS’s Categorization phase attempts to group services that use similar terms into like categories. For example, services that use a large number of financial terms may be grouped into a set of services characteristic of banking operations. In our approach, a category is less defined by a title or operation but more defined by a bag of characteristic strings taken from its underlying service specifications.

Once equivalences are made between specific parts, services can be linked together by their output and input messages during the Equivalence processing and clustering phases. In this step, a list of potential mash up can be determined solely based on the fact that certain services share equivalent parts. The Filtering phase attempts to identify the mash up that are indeed useful to a particular stake-holder.

IV. Determining Equivalences in Software Services (Equivalence Processing)
Approaches to service mash up are similar to the fundamental techniques for the discovery and composition of web services. Two common approaches to composition, consumption and perhaps mash up are semantic and syntactic techniques. Semantic approaches generally support the integration of web services by exploiting the semantic description of their functionality using ontological approaches. Here, in the experimentation with service mash up, we operate with non-semantically annotated services. A recent approach attempts to create semantics from extensible Markup Language (XML)-based specifications. Similarly, syntactical approaches are the focus of this work.

V. Determining Service Provisions That Are Complementary (Clustering)
The Clustering phase attempts to define groups of services that may be complementary and eligible for mash up. This approach also attempts to understand when two complementary services add value to a user. This determination tends to be relegated to semiautomatic approaches. This approach attempts to fully automate this determination. The major questions that we seek to answer are and “Can it be assumed that a service that integrates with a large number of services (i.e., congenial services) are likely to add value for new mash up. Since service mash up is such a new area, there were no closely related projects investigating these questions in the service-oriented computing domain.

VI. Equivalence Processing
An effective mash up blends two services that are not similar in function, but rather similar in the messages that they provide. In this way, identifying a similar message part, either input or output, between two services is the first step in our approach to predicting a service mash up. This approach combines syntactical methods with human naming tendencies to increase the probability of the messages having equivalent meanings. After finding similarity between web service messages, we place services with similar messages into clusters. Services can be associated to more than one cluster, but later they will have been attached to only one category. In the final phase, we attempt to predict the clusters that will add value to the end users. DS promotes the bottom-up evaluation of a repository of web services to discover new information via mash up. As such, these experiments are important in determining how the suite of KDS tools should be customized. In many cases, the choice of assessment level (i.e., service level, operation level, or part level) is dictated by the nature of the repository that is under assessment. Although categorization is most effective at the service level, initial clustering requires the use of input/output parts. Future experiments will determine hybrid approaches that blend levels based on the specific KDS objectives.

VII. Categorization
The idea of categorizing a repository or set of information is not a new one. It is useful to be able to target a certain type of information within a larger collection. Here, as
with the search for instrumental services, we work from a predefined set of services. Upon creating our repository of 545 services, we classified each service into one of nine categories. These categories followed closely the categories defined in related work.

VIII. CONCLUSIONS
The Filtering phase within the KDS process attempts to predict which mash up predictions will add value for a particular user. Although we intend to use human-generated data to filter our predictions in the future. To assess the variance in performance across the prediction methods, we recorded the average time for each method over five different executions. Existing web services on the Internet to gather insight about how services are developed and how service-based messages are named. (i.e., TSM-L, TSM-P, and TSM-LP) that mirror the nature of the open web services. In this work, we explored how these approaches could be applied to the domain of predicting service mash up. Results show that the TSM-L method provides the largest percentage of valid predictions. In addition, the recommendation performance is favorable with regard to making real-time recommendations. The TSM-L method is most effective on message names that utilize abbreviations. This suggests that the open repository contains many abbreviations as a part of service messages. In future: we plan to assess the ability to predict service mash up using a combination of inputs and outputs. Future application is the creation of a distributed web services development environment that leverages the knowledge of existing services.

References

AUTHOR
C.BALA SARAVANAN received the M.Tech (I.T) from Sathyabama University in 2011. During 2009-2010, I stayed in orbit technologies as software engineer to develop health care automation tool. And I am doing (Ph.D) in VELTECH University and I am now working in VelTech MultiTech Dr.R & Dr.SR Engineering College as Assistant Professor and IBM TGMC Project Coordinator.

A.SINGSTON CHANDAR is currently a student of Vel Tech Multi Tech Dr.Rangarajan Dr.Sakunthala Engineering college, Chennai and he is doing his Masters in “Network Engineering”. He received his Bachelor’s degree in “Information Technology” from Rajiv Gandhi College of Engineering and technology affiliated to Pondichery University 2010. His research interests include wireless and ad hoc networks.

G.SHANMUGASUNDARAM is currently a student of Vel Tech Multi Tech Dr.Rangarajan Dr.Sakunthala Engineering college, Chennai and he is doing his Masters in “Network Engineering”. He received his Bachelor’s degree in “Electronics and Communication” from Aalim Muhammed Salegh College of Engineering and technology Chennai, 2009. His research interests include wireless and ad hoc networks.

S.SURYA is currently a student of Muthayammal Engineering College, Rasipuram and she is doing her Masters in “Computer Science and Engineering”. She received her Bachelor’s degree in “Computer Science and Engineering” from Thangavelu College of Engineering and technology affiliated to Anna University, 2012. His research interests include Cloud Computing.