Improving the Data Quality of Web Services Composition

Xitong Li
Stuart Madnick
Hongwei Zhu

Working Paper CISL# 2011-07

November 2011

Composite Information Systems Laboratory (CISL)
Sloan School of Management, Room E62-422
Massachusetts Institute of Technology
Cambridge, MA 02142

Improving the Data Quality of Web Services Composition

Xitong Li
MIT Sloan School of Management
100 Main St., E62-427
Cambridge, MA 02142
(617) 253-6629, USA
xitongli@mit.edu

Stuart E. Madnick
MIT Sloan School of Management
100 Main St., E62-422
Cambridge, MA 02142
(617) 253-6671, USA
smadnick@mit.edu

Hongwei Zhu
Old Dominion University
2079 Constant Hall
Norfolk, VA 23529
(757) 683-5175, USA
hzhu@odu.edu

Keywords:
Web service, data quality, data interpretation, semantics, context

ABSTRACT

The Internet era has evolved from a *Web of documents* to a *Web of services*. Web services are intended to be application components that can be reused and integrated to create more advanced, innovative Web applications without needing to develop them from scratch. Unfortunately, Web services distributed on the Internet are usually independently developed by different organizations and/or individuals and have diverse assumptions about the interpretation of the exchanged data, such as inconsistent data representation and conceptualization. Such data misinterpretation can result in Data Quality (DQ) problems and hamper the potential of Web services. We identify important DQ challenges in Web services composition and present a classification of the resulting DQ problems. We suggest a novel reconciliation framework for addressing these problems and evaluate the framework in terms of scalability, adaptability, and extensibility. Finally, we identify important future directions in data quality and Web services.

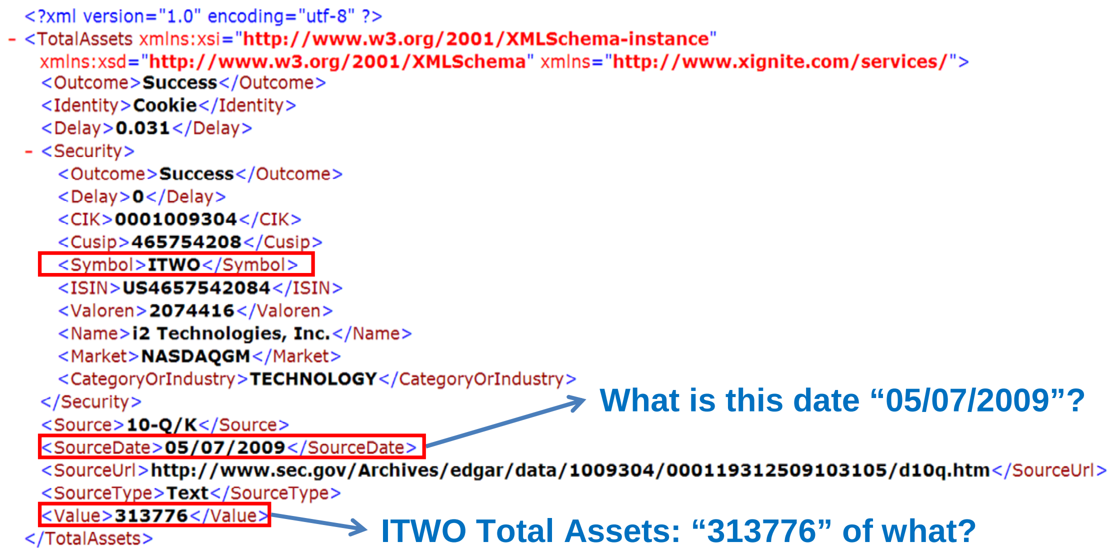
1. INTRODUCTION

With the increasing popularity of Service-Oriented Architecture (SOA) and Web 2.0[12], the Internet has evolved from a *Web of documents* to a *Web of services*[13] (e.g, SOAP-based and RESTful services). Web services are intended to be used for the development of more advanced, innovative Web applications, so that the development from scratch is avoided. The achievement of the full potential of Web services largely relies on how their integration (e.g, composition, mash-up[2]) can be accomplished in a convenient or even automatic way. Unfortunately, Web services distributed on the Internet are usually independently developed by different organizations and/or individuals and have diverse assumptions about the interpretation of the data exchanged between them. The differences in data interpretation can result in Data Quality (DQ) problems[8] and prevent easy and correct use and integrating of Web services.

As seen in various online discussions, users of existing Web services are often confused by data interpretations. For example, users of services provided by Amazon (1) and Paypal (2) have made the following comments[14]: (1) "*I have met a problem when I was trying to get the sales rank of digital cameras using the web service. It showed me pretty different sales ranks from what I saw through Amazon Website*"; and (2) "*Can someone explain the precise distinction between 'Completed' and 'Processed' [as the output of a payment service]?*" These users have in fact experienced DQ problems of Web services, because data quality is usually defined from the users' point of view in terms of *fitness for use*[8]. Certain DQ issues in distributed data sources are carried over to, and even amplified by, Web services that provide easy access to the data. The use of SOA and Web 2.0 both poses challenges and provides opportunities for developing effective solutions to address DQ problems in Web services.

2. MOTIVATING EXAMPLES

Xignite, Inc. is an established provider of on-demand, global financial market data services and application components. Among its published Web services, *XigniteEdgar* consumes the ticker symbol of any specific corporate entity and returns its total assets. When requested using the ticker symbol "ITWO" for i2 Technology, *XigniteEdgar* returns the data as shown in Figure 1. It can be seen that the returned total assets of i2 Technology is associated with the date "05/07/2009". But should the service users interpret the date as May 7th, 2009 or July 5th, 2009? Also, how should the figure of total assets "313776" be interpreted? Another request to *XigniteEdgar* returns "68853" as the total assets of Microsoft (ticker symbol: "MSFT"). Is it possible that i2 Technology has more than four times the total assets of Microsoft? Manual investigation finds that the numeric data for i2 Technology is in thousands and that for Microsoft is in millions, both using "$". But does the symbol "$" mean US dollar, Canadian dollar, or Hong Kong dollar? If these plausible differences were not explicitly clarified, the service *XigniteEdgar* could be incorrectly used by different users, and even cause surprising financial losses.



```xml
<?xml version="1.0" encoding="utf-8" ?>
- <TotalAssets xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns="http://www.xignite.com/services/">
    <Outcome>Success</Outcome>
    <Identity>Cookie</Identity>
    <Delay>0.031</Delay>
  - <Security>
      <Outcome>Success</Outcome>
      <Delay>0</Delay>
      <CIK>0001009304</CIK>
      <Cusip>465754208</Cusip>
      <Symbol>ITWO</Symbol>
      <ISIN>US4657542084</ISIN>
      <Valoren>2074416</Valoren>
      <Name>i2 Technologies, Inc.</Name>
      <Market>NASDAQGM</Market>
      <CategoryOrIndustry>TECHNOLOGY</CategoryOrIndustry>
    </Security>
    <Source>10-Q/K</Source>
    <SourceDate>05/07/2009</SourceDate>
    <SourceUrl>http://www.sec.gov/Archives/edgar/data/1009304/000119312509103105/d10q.htm</SourceUrl>
    <SourceType>Text</SourceType>
    <Value>313776</Value>
  </TotalAssets>
```

What is this date “05/07/2009”?
ITWO Total Assets: “313776” of what?

Figure 1. Total assets returned from a real Web service showing how different interpretations of the source date and total assets undermine its data quality.

Meaningfully understanding and use of only one simple service is not easy, but integrating multiple services is even more challenging. Let’s consider a very simple composition scenario consisting of only two services in which a Chinese developer wants to create a composite service *ConferenceBooking* which consumes a certain international conference code and returns the registration expense of the conference and hotel expense in the city where the conference is held during the conference period. *ConferenceBooking* is supposed to be used by Chinese clients. After searching a public service registry, the developer decides to implement the service by composing two existing services: *ConfRegistration* and *HotwireDeals*[a]. Given a conference code, the operation *queryConfInfo* of *ConfRegistration* provides basic information of the conference, including the start/end dates, registration fee and the city. The operation *queryRoomCharge* of *HotwireDeals* returns the room charge of the deals based on the city name and start/end dates. It appears that *ConferenceBooking* can be developed by feeding the output of *ConfRegistration*’s operation *queryConfInfo* as the input to *HotwireDeals*’s operation *queryRoomCharge*. Figure 2 illustrates the composition process[b].

However, since these services are developed by independent providers, they may have different assumptions of data interpretation. For example, *ConferenceBooking* being composed is intended to return the monetary expenses in Chinese yuan (“CHY”) and the hotel expense should include the value-added taxes. *ConfRegistration* provides the dates in the format “dd-mm-yyyy” and the registration fee in euros. *HotwireDeals* assumes dates are in the format “mm/dd/yyyy” and returns the room charge in US dollars (“USD”) which doesn’t include the value-added taxes. Apparently, *HotwireDeals* cannot successfully interact with *ConfRegistration* because of inconsistent data formats, rendering the entire composition invalid. Also, *ConferenceBooking* would misinterpret the monetary figures due to different currencies used by the component services. Without properly resolving the data misinterpretation problems, conflicts would happen in the composition process, as noted in Figure 2 by little “explosions”.

The assumptions of data interpretation are usually not explicitly specified in service descriptions (e.g., WSDL) and severely undermine data quality and usability of independently developed Web services. The data misinterpretation problems primarily affect the DQ dimensions in the intrinsic, contextual and representational DQ categories[8].

[a] HotwireDeals originates from Hotwire.com, available at http://developer.hotwire.com/docs/Hotel_Deals_API
[b] Note that unlike traditional data bases, which typically use query languages, web services composition is done by specifying the information flow amongst the services using languages like Business Process Execution Language (BPEL).

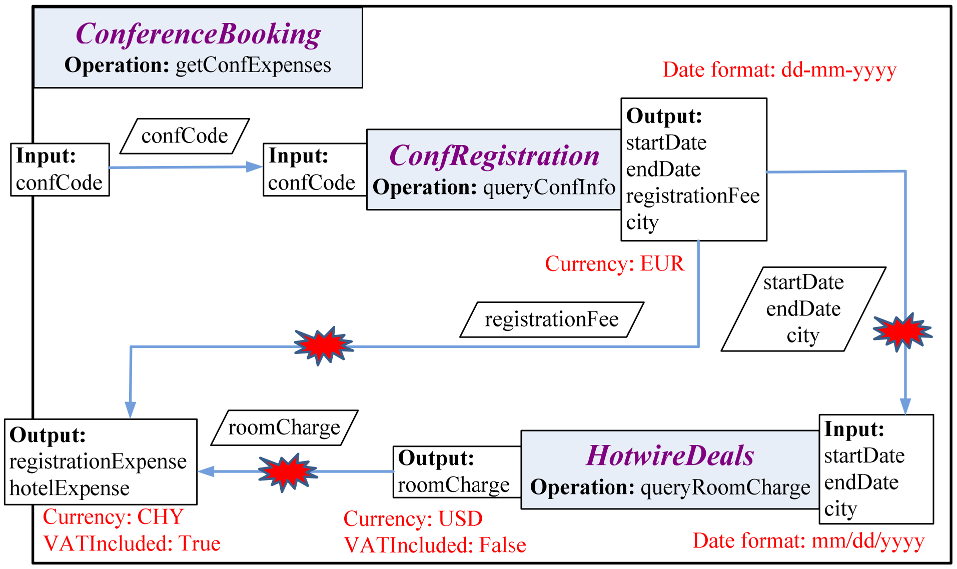


Figure 2. Composition process of the composite service consisting of two services. Data misinterpretation causes semantic conflicts in the composition.

3. CLASSIFICATION OF DATA MISINTERPRETATION PROBLEMS

We categorized data misinterpretation problems into representational and conceptual levels, both of which can be further classified into subcategories. These problems are not unique to Web services, yet they become more prevalent and important now, because Web services are more open, dynamic and autonomous, and the solution must be addressed in a way that works with Web service standards.

3.1 Representational

Different organizations may use different representations for a certain data concept, which can result in representational misinterpretation problems. Five subcategories are further identified at this level: format, encoding, unit of measure, scale factor, and precision. Format differences occur because there often exist various format standards, such as for representing date, time, geographic coordinates, and even numbers (e.g., “1,234.56” in USA would be represented as “1.234,56” in Europe). Encoding differences may be the most frequent cause of representational misinterpretation, because there are often multiple coding standards. For example, the frequently used coding standards for countries include the FIPS 2-character alpha codes, the ISO3166 2-character alpha codes, 3-character alpha codes, and 3-digit numeric codes. Also, IATA and ICAO are two major, different standards for airport codes. Data misinterpretation problem can occur in the presence of different encoding standards (e.g., country code “BG” can stand for Bulgaria or Bangladesh, depending on whether the standard is ISO or FIPS). Besides the format and encoding differences, numeric figures are usually represented using different units of measure, scale factors, and precisions. For example, financial services use different currencies to report the data to corresponding consumers. Scientific services may use different units of measure (e.g., meter or feet) or scale factors (10^3 or 10^6) to record the data.

3.2 Conceptual

The same term and representation are often used to refer to similar but slightly different data concepts. This category of data misinterpretation usually occurs when the concept can have different assumptions of the interpretation, that is, whether a specific entity is included by the concept or not. For example, a retail price reported by European services usually includes the value-added taxes, while retail prices reported by US services usually do not include the value-added taxes[c]. An even more challenging problem in this category is “*Corporate Householding*”[8] which refers to misinterpretation of corporate household data. For example, to query the sales of a certain corporate (e.g., “What were the total sales of IBM?”), one (of many) questions that needs to be clarified is whether majority owned subsidiaries of the corporate should be included or not. The answers may be very different due to different reporting rules adopted in different countries or for different purposes. Besides the entity aggregation issue, the conceptual extension of the inter-entity relationship may also have different interpretations. For instance, in order to answer the question “How much did MIT purchase from IBM in the last fiscal year?”, we need to clarify whether the purchasing relationship between MIT and IBM should be interpreted as direct

[c] Usually called “sales taxes” in the USA.

purchasing (i.e., purchased directly from IBM) or to include indirect purchasing through other channels (e.g., third-party brokers, distributors, retailers). In some cases, only the direct purchasing from IBM to MIT are considered, whereas in other cases indirect purchasing through other channels also needs to be included.

3.3 Temporal

It is worth noting that most of the above-mentioned possibilities of data interpretation can change over time.[8] For example, a Turkish auction service may have listed prices in millions of Turkish liras (TRL)[d], but after the Turkish New Lira (TRY) was introduced, it may start to list prices in unit of Turkish New Lira. Also, an accounting service may or may not aggregate the earnings of Merrill Lynch into that of Bank of America which acquired the former in 2008. Considering the high dynamics of Web services distributed on the Internet, these data misinterpretation problems resulting from the temporal evolvement can become very challenging. That is, a component service might change its data interpretations after the composition has been created, tested and deployed.

4. RECONCILIATION FRAMEWORK

Data misinterpretation can cause poor data quality when composing multiple Web services. It is thus necessary to resolve data interpretation conflicts among them, to avoid data quality problems. In simple cases, reconciliation can be manually done by service developers (though still vulnerable to later temporal changes.) In most non-trivial cases, however, a manual reconciliation will be difficult and error-prone, as a large number of services and data elements are involved. For example, a current SOA implementation of a health care accounting system consisted of over a hundred Web services. Figure 3 illustrates a suggested reconciliation framework to resolve the various data misinterpretation problems that may occur in Web services composition and improve DQ of Web services. A preliminary composition of multiple Web services is first produced using existing composition tools that do not consider data interpretation issues. The preliminary composition can be described using BPEL or other languages, depending on whether the participant services are WSDL- or RESTful-based[9]. Given the composition specification with possible interpretation conflicts, the framework consisting of several tools (indicated as shaded boxes) can automatically produce a mediated composition specification in which all interpretation conflicts are resolved.

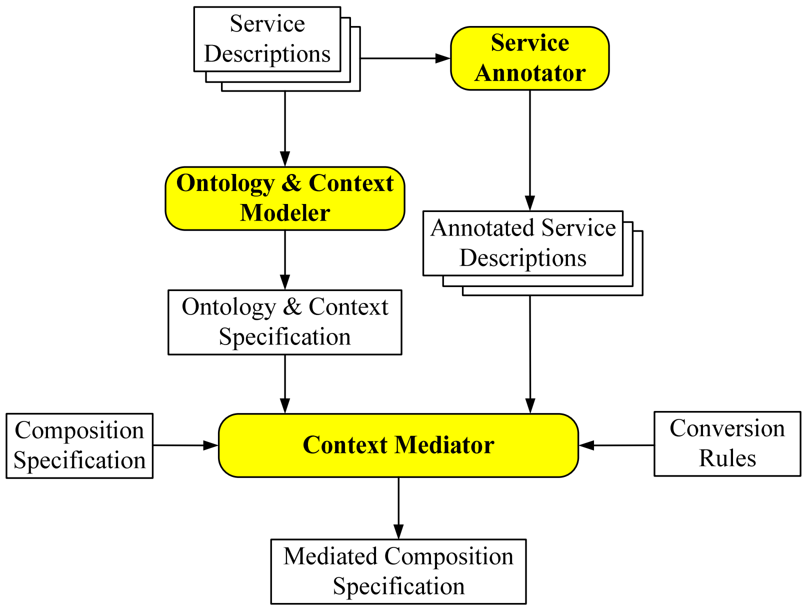

Service Descriptions
Service Annotator
Ontology & Context Modeler
Annotated Service Descriptions
Ontology & Context Specification
Composition Specification
Context Mediator
Conversion Rules
Mediated Composition Specification

Figure 3. Reconciliation framework that improves data quality of Web services and the composition.

4.1 Ontology & Context Modeler

The *Ontology & Context Modeler* is used to define a common ontology that specifies the concepts and their relationships. As an extension to traditional light-weight ontological models, a special attribute *modifier*[6] can be used to capture multiple variations of the generic concepts, that is, different data interpretations. A generic concept can have multiple modifiers, each of which indicates a dimension of the interpretation variations. Different value assignments to these modifiers

[d] About one million TRL equaled one US dollar.

are referred to as different *contexts,* and in a certain context each modifier is assigned by a specific modifier value. Each service involved in the composition may be associated with a context corresponding to its assumptions of data interpretation.

Figure 4 shows the ontology for the composition example in Figure 2. The ontology has a generic concept *roomCharge* having two modifiers: *currency*[e] and *vatIncluded*. In the context of *HotwireDeals*, *currency*'s value is "USD" and *vatIncluded*'s value is "False", which means that *HotwireDeals*'s output data of *roomCharge* should be interpreted by currency "USD" and not including value-added taxes. The modifier values of *currency* and *vatIncluded* in the context of *ConferenceBooking* are "CHY" and "True", which means that *ConferenceBooking* has a different data interpretation from *HotwireDeals*. Using such an ontology enriched with contexts, different data interpretations among the services can be treated as context differences to be resolved.[6]

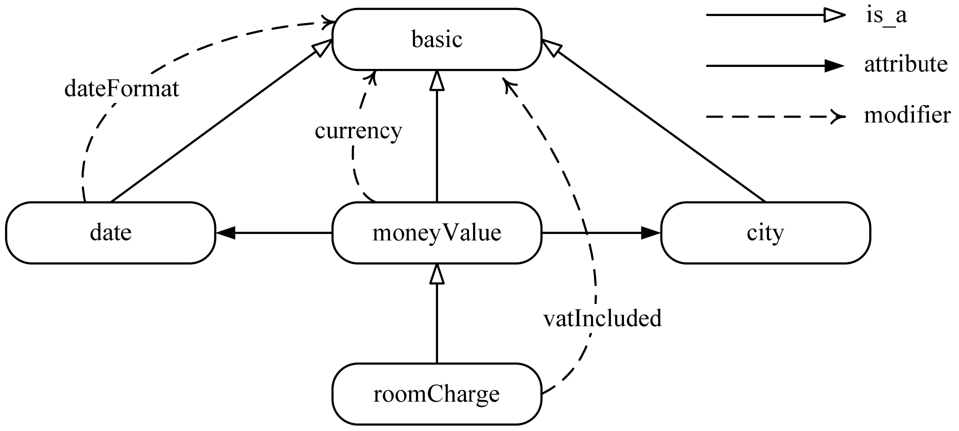


Figure 4. Light-weight ontology extended with modifiers that supports the reconciliation of data misinterpretation for the composition example.

4.2 Service Annotator

In practice, SOAP-based services are described using WSDL and descriptions of RESTful services are often embedded in Web pages written in XHTML or microformat. Both are described at a syntactic level, rather than a semantic level. In order to facilitate the integration (e.g., composition, mashup) of Web services, developers need to use *Service Annotator* to annotate syntactic service descriptions, so that these services can interact with each other at a semantic level. W3C has published the recommendation standard: Semantic Annotations for WSDL and XML Schema (SAWSDL).[5] Recent efforts are also trying to adapt SAWSDL to annotate semantics for RESTful services.[10] Compliant with SAWSDL, we propose two annotation methods (i.e., global and local)[6] to establish correspondences between WSDL-based service descriptions and the context-enriched ontology model. When semantic correspondences are established, context differences among the services can be detected by reasoning algorithms.[6]

Context differences, once detected, can be reconciled using conversions for converting the exchanged data from the source value *vs* to the target value *vt*. An atomic conversion is defined for each modifier between two different modifier values. The general representation of the conversions is: *cvt*(*C*, *m*, *ctxt_s*, *ctxt_t*, *mvs*, *mvt*, *vs*, *vt*), where *C* is the generic concept having a modifier *m*, *mvs* and *mvt* are two different values of *m* in source context *ctxt_s* and target context *ctxt_t*, respectively. To resolve the differences of currency and value-added taxes[f] that exist in the composition example in Figure 2, two atomic conversions cvt_{currency} and $cvt_{\text{vatIncluded}}$ need to be defined, both of which are implemented as external Web services. XPath functions can also be used to implement some relatively simple conversions, such as the date format conversion $cvt_{\text{dateFormat}}$. These conversion rules, registered in the conversion library, can be automatically incorporated in the service composition by *Context Mediator* as introduced below.

4.3 Context Mediator

Context Mediator is the core component of this proposed reconciliation framework. It takes the composition specification described by the developer using Web Service Business Process Execution Language (WS-BEPL) or Web mashup languages,[9] and produces a mediated composition specification in which data misinterpretation problems among the services are resolved. Specifically, the *Context Mediator* first identifies all explicit and implicit data transfers in the

[e] The modifier *currency* of *roomCharge* is actually inherited from its super concept *moneyValue*.

[f] The two dimensions of context differences are identified by the two modifiers *currency* and *vatIncluded*.

composition process. Then, it examines the contexts of the source and target of each data transfer. By tracking the semantic annotation and reasoning on the context-enriched ontology, potential interpretation conflicts (i.e., context differences) are determined. Lastly, *Context Mediator* incorporates the appropriate conversion rules from the conversion library into the composition process, so that the interpretation conflicts are resolved. All these steps can be performed by the *Context Mediator* using the automatic algorithms.[6]

5. EVALUATION

We have implemented a prototype of the suggested reconciliation framework and evaluated its feasibility in resolving various data misinterpretation problems discussed above.[6] After the reconciliation, the mediated composition process can be successfully deployed and executed without human intervention.

A quantitative evaluation of the reconciliation framework is conducted with the focus on measuring human efforts involved in developing the conversions, i.e., the number of conversions to be manually specified and maintained over time. The key evaluation metrics are 1) *Scalability*: number of conversions needed for the reconciliation among the involved services; 2) *Adaptability*: number of conversions to be updated when data semantics of the involved services change; 3) *Extensibility*: number of conversions to be added (or removed) when a service is added (or removed). In general, the reconciliation framework requires much less conversions than brute-force approaches and is more flexible to be adapted for changes. This evaluation reveals that the reconciliation framework holds the desired properties of scalability, adaptability and extensibility.[3,6]

6. CONCLUSION

The number of Web services published on the Internet continues to rise dramatically. Since the spring of 2008, the European Commission has initiated a collaborative research effort for the next generation of the Internet titled “Future Internet” (http://www.serviceweb30.eu/). Semantic technologies are being suggested to support the integration of Web services, indicating the direction of the evolving Internet. There needs to be a move beyond simply annotating data on Web pages to annotating exposed functionality in the form of Semantic Web services.[4] However, data misinterpretation problems, as summarized in this paper, could undermine the DQ of Web services and hamper their integration (e.g., composition, mashup) as well as the execution of scientific workflows involving data-intensive services.[16] Therefore, this paper aims to establish the connection between DQ and Web services composition and draw attention to the DQ challenges in composing Web services.

Ontologies and semantics are required to address these DQ challenges. As suggested[15], we expect authors of Web services will specify certain metadata (e.g., ontologies) definition and semantic annotation. Besides that, context is an important aspect of data semantics for interpreting the meaning of the data provided by Web services.[1,7] Researchers have begun to develop solutions[2,11] albeit with limited scope, to address some of these problems. Thus, we call for further research to develop full solutions which will include intelligent techniques that can be used to facilitate ontology/context construction and semantic annotation for Web services. We expect over time such ontologies will become increasingly available and service developers will provide the appropriate ontologies/context and service annotations. With the context-enriched ontologies in place, we can develop the approaches (such as the one presented in this paper[6]) to address various data misinterpretation problems of Web services.

We intend for this paper to draw the challenges and opportunities of Web services composition to the attention of both practitioners and researchers and open up the new research directions towards improving DQ of Web services. We believe previous data approaches can be adapted and new techniques can be developed. Some initial approaches are presented in this paper.

7. REFERENCES

1. Agrawal, R., Ailamaki, A., Bernstein, P.A., Brewer, E.A., Carey, M.J., Chaudhuri, S., Doan, A., Florescu, D., Franklin, M.J. and Garcia-Molina, H. The claremont report on database research. *Communications of the ACM, 52* (6), 2009. 56-65.
2. Di Lorenzo, G., Hacid, H., Paik, H. and Benatallah, B. Data integration in mashups. *ACM SIGMOD Record, 38* (1), 2009. 59-66.
3. Gannon, T., Madnick, S., Moulton, A., Siegel, M., Sabbouh, M. and Zhu, H., Framework for the Analysis of the Adaptability, Extensibility, and Scalability of Semantic Information Integration and the Context Mediation Approach. in *System Sciences, 2009. HICSS '09. 42nd Hawaii International Conference on,* (Hawaii, 2009), 1-11.
4. Hepp, M. Semantic Web and semantic Web services: father and son or indivisible twins? *Internet Computing, IEEE, 10* (2), 2006. 85-88.

5. Kopecký, J., Vitvar, T., Bournez, C. and Farrell, J. SAWSDL: Semantic Annotations for WSDL and XML Schema. *IEEE INTERNET COMPUTING*, *11* (6), 2007. 60-67.
6. Li, X., Madnick, S., Zhu, H. and Fan, Y.S., Reconciling semantic heterogeneity in Web services composition. in *Proceedings of the 30th International Conference on Information Systems (ICIS 2009)*, (Phoenix, AZ, USA, 2009).
7. Maamar, Z., Benslimane, D. and Narendra, N.C. What can context do for web services? *Communications of the ACM*, *49* (12), 2006. 98 - 103.
8. Madnick, S., Wang, R., Lee, Y. and Zhu, H. Overview and Framework for Data and Information Quality Research. *ACM Journal of Data and Information Quality (JDIQ)*, *1* (1), 2009. 1-22.
9. Marwan, S., Jeff, H., Salim, S. and Danny, G. Web mashup scripting language *Proceedings of the 16th international conference on World Wide Web*, ACM, Banff, Alberta, Canada, 2007.
10. Meersman, R., Herrero, P., Dillon, T., Maleshkova, M., Kopecký, J. and Pedrinaci, C. Adapting SAWSDL for Semantic Annotations of RESTful Services. in *On the Move to Meaningful Internet Systems: OTM 2009 Workshops*, Springer Berlin / Heidelberg, 2009, 917-926.
11. Mrissa, M., Ghedira, C., Benslimane, D., Maamar, Z., Rosenberg, F. and Dustdar, S. A context-based mediation approach to compose semantic Web services. *ACM Transactions On Internet Technology*, *8* (1), 2007. 4.
12. Oreilly, T. What is Web 2.0: Design patterns and business models for the next generation of software. *Communications & Strategies, No. 1, p. 17, First Quarter 2007. Available at SSRN: http://ssrn.com/abstract=1008839*.
13. Raman, T.V. Toward 2 W, Beyond Web 2.0. *Communications of the ACM*, *52* (2), 2009. 52-59.
14. Saleh, I., Kulczycki, G. and Blake, M.B. Demystifying Data-Centric Web Services. *IEEE INTERNET COMPUTING*, *13* (5), 2009. 86-90.
15. Savas, P., Evelyne, V. and Tony, H. A "Smart" Cyberinfrastructure for Research. *Communications of the ACM*, *52* (12), 2009. 33-37.
16. Tsalgatidou, A., Athanasopoulos, G., Pantazoglou, M., Pautasso, C., Heinis, T., Gr, R., nmo, Hj, rdis, H., Arne, J., rgen, B., Glittum, M. and Topouzidou, S. Developing scientific workflows from heterogeneous services. *SIGMOD Rec.*, *35* (2), 2006. 22-28.