Discovering Data Quality Issues in Service-oriented Architectures
A Framework and Case Study

Plamen Petkov, Markus Helfert and Thoa Pham
School of Computing, Dublin City University,
Glasnevin, Dublin 9, Ireland
{ppetkov, markus.helfert, thoa.pham}@computing.dcu.ie

Keywords: Service Oriented Architectures, Service Composition, Data Quality, Data Monitor, Business Rules, Business Process, Quality of Service.

Abstract: In this paper we examine web services from a data quality perspective. Based on a data quality management approach, we propose a framework for analysing data produced by the composite service execution. Apart from other monitoring tools available which targeting the technical aspect of service composition – service time response, service throughput and etc., we propose data consistency and accuracy monitoring, focusing on the business value of the data produces by the services. We developed framework that will store business rules into a rules repository. By analysing service data against the rules we will be able to identify problems in service composition and execution. Moreover taking into account the Quality of Service (QoS) we are able to provide an approximate location of the error.

1 INTRODUCTION

As a result of the global networks, information systems progress, and the need for flexibility, traditional closed, static and centralized architectures have evolved to dynamic and heterogeneous. The tasks of developing completely new applications, making certain adaptors for legacy systems, or rewriting present applications are now outdated. Principally boosted by Web services connectivity, service-oriented architectures (SOA) are now considered the preferred way to designing an information system. SOA endeavours to provide existing functions of an information system as "services" that can be accessed in a loosely coupled way (Papazoglou, 2007), independently from the technical platform. The architecture is seen as an orchestration of requests for those services. Generally, In SOA, workflow or orchestration processes are fundamental.

However in more complex architectures orchestrating the services can be difficult to handle. There is no efficient way to managing such architectures without having the awareness of the data, processes and events running within the enterprise environment. To support the process of orchestrating, as well as development and evolving progress, a monitor tool(s) must be integrated. These tools, of course, must comply with business requirements, in order to achieve adequate surveillance result.

In other words, management tools and techniques are inadequate without using an appropriate monitoring. Thus why, crucial assistant for proficient and effective deployment and operation of an SOA-based net-centric system is a comprehensive monitoring capability. Nevertheless, present monitoring solutions fall short with respect to such systems because they do not hold the capabilities to implicitly aggregate metrics, effectively detect inconsistent or inaccurate data, and so to provide comprehensive shared situational perception.

In this paper we propose data quality monitoring approach by developing framework that will be able to identify data quality problems.

The remainder of the paper will be structured as follows: Section 2 describes the importance of Quality of service (QoS) for service selection. In this section we also present some data quality issues related with service composition. In Section 3 we propose framework that will identify and localize the data related problems. In section 4 deals with simple case study and discussion of our framework. Finally, conclusions are presented along with suggestions for future work.

Helfert M., Petkov P. and Pham T.
Discovering Data Quality Issues in Service-oriented ArchitecturesA Framework and Case Study.
DOI: 10.5220/0004461300720080
In *Proceedings of the Second International Symposium on Business Modeling and Software Design (BMSD 2012)*, pages 72-80
ISBN: 978-989-8565-26-6

2 TOWARD TO MONITORING UTILIY IN SOA

2.1 Importance of Service Selection

Web services are the key technology in SOA, in which services are considered as “autonomous, platform-independent entities that can be described, published, discovered, and loosely couple in novel ways” (Papazoglou, 2007). A service oriented application includes a service provider and a service requester. A service discovery agency (e.g. Universal Description Discovery and Integration UDDI)) may act as intermediate between provider and requester and provides functionality to promote available services. The service provider defines a service description and publishes it (to the agency). After retrieving a suitable service, the service requester is able to invoke that service (The SoA Open Group). In this regard, service composition encompasses the process of searching and discovering relevant services, selecting suitable web services of best quality and finally composing these services to achieve an overall goal that usually in a business context aims to support an underlying business process.

The Publish – Discover – Binding schema is depicted below.

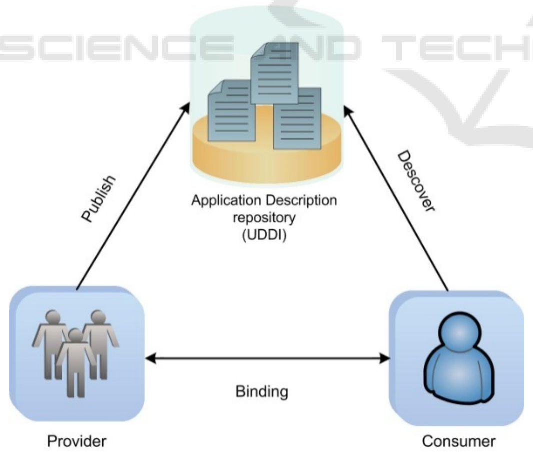


Figure 1: Publish – Discover – Binding.

On reviewing prominent approaches for service discovery, it appears they mainly involve functional attributes of service advertised in the service description. These include service type, operation name, input/output data format and semantics (Zeng, 2003). In order to select suitable services, quality of service (QoS) evaluation is usually used as approach for service selection among many services of similar functionality. In literature, many approaches have been proposed to measure QoS with non-functional quality criteria. QoS dimensions often refer to non-functional criteria that include execution price, execution duration, reputation, reliability and availability (Jeong, 2009).

Meanwhile the functional quality of a set of composed services receives little attention. Often, it is assumed that the service functions execute according to the stated and published service description. However, as with any execution and operation of applications, this may not be the case. Indeed, discussions with practitioners show, web services often do not fulfil the functional quality and thus the expected output is not achieved. In contrast to other research, we consider this problem and provide a framework that can help to detect some of the problems during the execution of the services.

Service selection is crucial stage in service composition and QoS act as blueprint of SoA reliability. QoS could also play significant role into service composition monitoring as we will show later in this paper.

2.2 Data Quality Issues in Web Service Composition

Data can be considered as output/input product of service orientated composition and execution. The product quality is key factor for entrepreneurs and customers. Therefore the quality of data component is the most important in a SoA project. Moreover, data can be manipulated by different parties like portals, devices or even orchestration engine. This means that data quality is mirror reflection of the quality of overall SoA implementation. Many researchers have been done investigations about defining Data Dimensions (Wand 1996), Data Quality Requirements Analysis and Modelling (Wang, 1998). However most of these studies apply to monolithic information systems. Since the introduction of the composite architecture few researcher take into account data quality issues (Thoa and Helfert, 2011).

Incorporating quality issues into web service composition and execution (WSCE), is a major problem regarding the application integrity. (Fishman, 2009) represented the problem with bad data quality in SoA by comparing the different services with animals and human – they are different by nature but share the same diseases. In other words, integrating an application into SoA, which does not comply with the business rules and thus provides inaccurate or inconsistent data, can affect all other applications and cause cumulative effect of errors typical for system development lifecycle. Figure 2 illustrates the service composition and service incompliance.

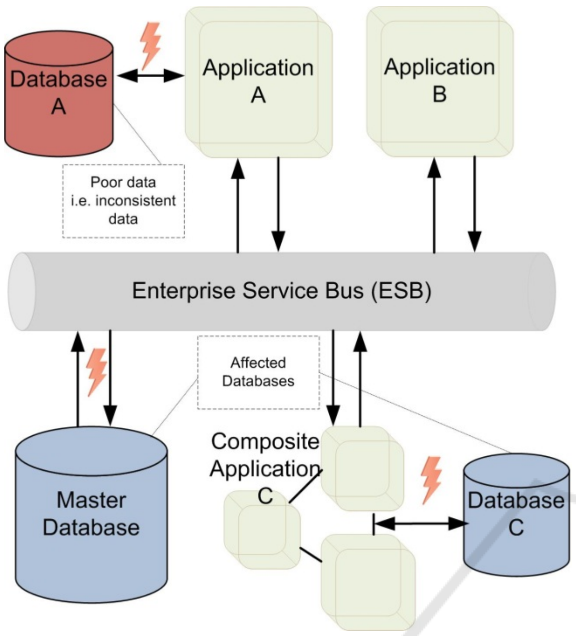


Figure 2: Data incorporation in SoA framework.

As it can be seen from the diagram, incompliance at service level (Application A) can ‘spread’ all irrelevant data through the Service Enterprise Bus to all other application and its databases (Master enterprise data, Composite Application C and its database D). Note that Application B has not its own database but it supplies the whole infrastructure with functionalities and thus it has been involved indirectly in the process of distributing poor data – e.g. affecting master enterprise database.

The data quality literature explored the most four dimensions of data quality which are accuracy, completeness, consistency, and timeliness (Wand, 1996).Corresponding to the functional quality of WSCE, in this paper we will explore these dimensions of data quality (Table 1). Analysing those dimensions help to clarify the causes of poor data quality in WSCE.

Taking into account quality issues in WSCE, many approaches for static or dynamic web service composition and execution have been developed (Agarwal, Narendra, Silva, Zhang). However, quality issues of mapping functional requirements to service composition and execution are current.

Table 1: Quality Dimensions and problem description.

Quality Dimension	Problem description
Accuracy	The data value does not correctly reflect the real-world condition.
Consistency	The data values persist from a particular data element of the data source to another data element in a second data source. Consistency can also reflect the regular use of standardized values, particularly in descriptive elements
Completeness	Data element is always required to be populated and not defaulted Data element required based on the condition of another data element
Timeliness	When two source data systems are synchronized outside of a shared transaction boundary, there exists a timing window during which a service accessing both systems may encounter mismatched data. The entity represents the most current information resulting from the output of a business event.

3 FRAMEWORK FOR ANALYSING DATA QUALITY

The data quality analysing process will be separated in two stages – problem discovery stage (1) and problem localization stage (2).

In order to execute the process above, our proposed framework is spitted into two modules. The first module will be able to detect a data quality issues. It follows a business rules-based approach to data quality. Based on detected problem by the first module and the QoS properties of the services, the second module will be able to recommend a particular service where the problem stems from. A simple block schema along with the analysing process is given on the Figure 3.

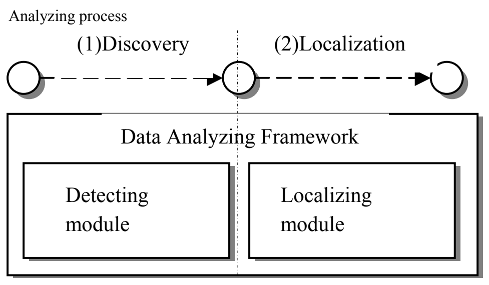


Figure 3: Data Analysing Framework.

3.1 Data Quality Issue Detecting Framework

The proposed framework follows a business rules-based approach to data quality. Business rules are "statements that defines or constrains some aspects of a business" A business rule is normally described by the natural language, which can be reconstructed in form of Event-Condition-Action (Bubenko and Herbst), in form of If-Then expression, or in form of <Subject> Must <constraints> (Morgan, 2002). Based on our earlier work (Thoa and Helfert, 2011) and the approach above, we developed a framework for service composition that is presented in Figure 4.

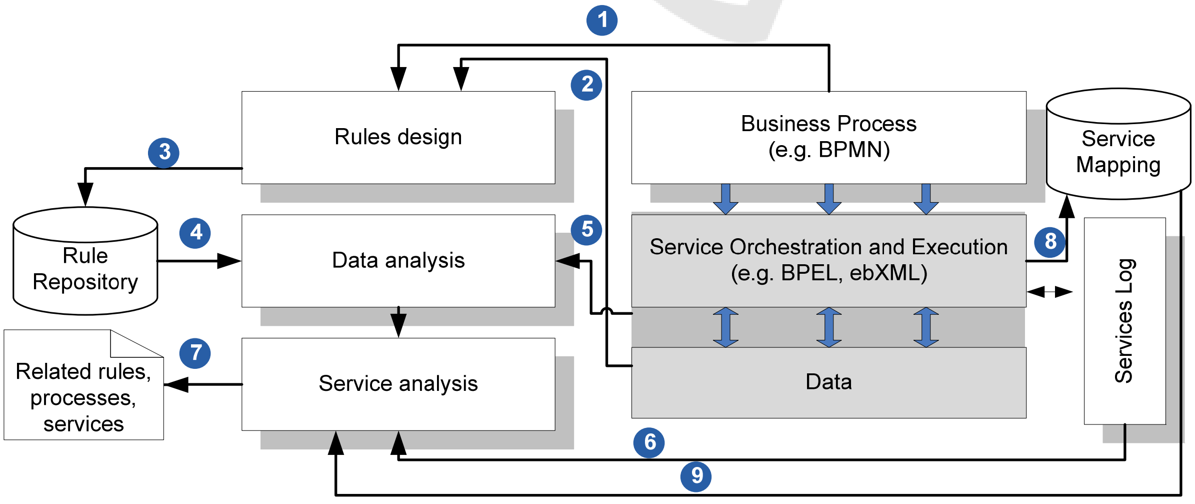
In the next few paragraphs we will give more detailed overview of our detecting module.

Business Process:
This block represents the conceptual business process (BP) model that a service composition must comply with. A BP model can be described with BPMN, UML or EPC model. In our framework, we are interested in the input and output of data of activities/tasks in the BP. A part of the meta-data of the BP model is stored in the Service Mapping and Rule Repository (Figure 3).

Business Rules Specification:
This component concerns specifying business rules. A business rule is related to one or many activities/tasks in a business process and/or data objects (1 and 2). The specified rules then are stored in a Rule repository which could be relational databases or XML files (3). The rule is usually in form of If <Boolean logic expression>* Then <Boolean logic expression>*, or an assertion of aforementioned. <Boolean logic expression>* can be composed of one or many Boolean logic expressions combined together with logical operators i.e. <Boolean logic expression> = <left expression> <comparison operator> <Right expression>. The left and right expression can be mathematical formula, including data values, data attributes, mathematical operations or aggregation functions. Moreover logic expression consisting of more logic expressions can be presented as binary tree.

Service Mapping Repository:
This repository captures the mapping between services to be composed to tasks/activities defined in the underlying business process. A task corresponds usually to a service, and a service can correspond to one or many tasks. However in the case that a task corresponds to many services, the service composition is significantly more complex and is currently not subject of this study. The information is usually stored within the service composition phase (8).

Service Log:
The service log captures specific events occurred during the service execution. In our framework we are particularly interested in events related to data updates and changes. It is necessary to detect what service instance in what composite service instance writes what data to the database. Since in the most service compositions the access to the service



1
2
3
4
5
6
7
8
9
Rules design
Business Process (e.g. BPMN)
Service Mapping
Rule Repository
Data analysis
Service Orchestration and Execution (e.g. BPEL, ebXML)
Services Log
Related rules, processes, services
Service analysis
Data

Figure 4: Data problem discovery framework.

database is indirect, we cannot mine the data directly Therefore our framework will work with the already 'digested; data from service interface described with WSDL. All these information should be stored in the service log (9).

Although there are approaches to specify log formats (Gaaloul, 2008), current approaches such as the Common Log Format and Combined Log Format of W3C are not sufficient for our approach as they are not directly able to represent the required information. Therefore, we propose a practical oriented log file. The log file entries contain following information:

```
<Entry>
<srv_location> …</Srv_location>
<Service_ID>… </Service_ID>
<instance_ID>… </ instance_ID>
<endPointName>… </endPointName>
<operationName>… </operationName>
<dataEntityName>… </dataEntityName>
<Data Value>...</Data Value>
</Entry>
```

The XML template can handle all data needed for recording a problem. `<endPointName>` tag will store the interface name of the service, while `<operationName>` tag will store the function delivering the error.

Data and Service Analysing
This component analyses the data produces by the endpoint of the service against the rule repository. Information about the service and data content produced is reported using Service log template.

3.2 Problem Localization Framework

Problem localization process (figure 3 (2)) is a part of overall data quality analysing process. This process aims to provide approximate problem location, since the problem can involve two or more services. In fact, in service orientation environments it is very likely that the problem is either in one or another service.

In order to localize the potential error giving service, we will propose localization framework will provides us with estimate solution. To do so, we will use the service log delivered by the problem detection framework during the data issues discoverer stage (figure 3 (1)). Moreover we will use the Quality of Service (QoS) properties of the 'infected' services to decide which one is more likely to be the source of the problem.

Base on approach above, we developed a framework that is represented in Figure 5.

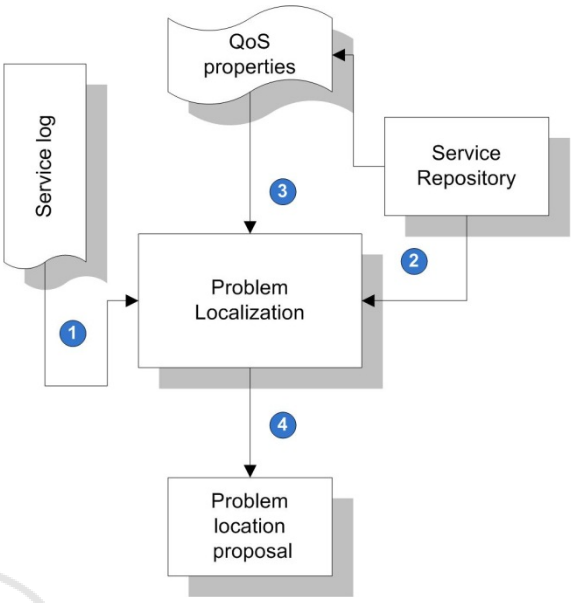


Figure 5: Framework for data problem localization.

The following paragraphs describe the main elements of our framework.

Service Log
In section 3.1 we proposed a framework for detecting quality issues and notating the services that produce data into a XML log file. We also proposed exhaustive log format that provides our localization module with needed information.

The service log is been generated during the first stage (discovery stage) of analysing process and acts as a starting point for this framework. Logged data then is passed to the 'Problem localization' element (1) to be processed.

Service Repository and QoS Properties
Service Repository in a place where services descriptions are store. Usually this repository is called Universal Description Discovery and Integration (UDDI) and supplies the 'Problem localization' with service information (2).

Quality of Service block is vital in this framework. This element contains all list with all QoS properties concerning given Service(s). QoS dimensions often refer different criteria but most of them include execution price, execution duration, availability and reliability. Reliability criteria are maybe one of the most important one, in order to localize objectively the problem. By reliability we mean, least error productivity and least time that service is out-of-order. For the sake of brevity, we will not discuss QoS criteria further.

The QoS criteria are granted (3) to the localization element for further treatment.

Problem Localization

Problem localization is the core element in our structure. It provides the framework with the following functionalities:

- collects the necessary data from the *Service log and Service repository* along with appropriate *Quality of Service properties* i.e. out-of-order delivery
- performing a comparison (based on QoS criteria) of the problematic services.
- prioritise the services using '*the least reliable, the more likely to be problematic*' method.
- ability to tracing a problem
- working up a detailed report and proposing approximate problem location(4).

As you can perceive, because of the duality nature of the problems, our methodology can provide only rough solution of the problems. Therefore we strongly recommend manually examination on the generated by localisation framework report and further investigation of problems.

In the next chapter we will apply our data analysing framework to given simple 'package booking' study case.

4 CASE STUDY AND DISCUSSION

The following case study illustrates the developed framework with a common service that provides the booking of travelling packages. The case study is motivated by a real case study. Initially, customers search information about available travel packages, and then subsequently may book a flight and a hotel. Once the booking is completed, the user pays the total amount and confirms the booking. Alternatively the user may cancel the booking. We developed a service oriented application for Travel package booking and analyzing data quality of the application along the two phases: preparation and analyzing.

Initially, a conceptual business process model for the booking travelling package is modelled. Next, the service composition process is realized; the mapping information between the tasks of the BP and individual services are stored. We suppose this is a design time service composition. The service composition can be described with BPEL based on the orchestration depicted in Figure 6.

The composite service *Booking package* is composed of a set of available services: *Book Flight* service, *Book Hotel* service, and *Payment* service. The flows of data/message between services are also described in the orchestration.

Mapping services and tasks in BP are as following table:

Table 2: Service mapping.

4.1 Analysing: Problem Discovery Stage

Once the study case framework is deployed and all service contacts are made we are ready to move to analysing stage and problem discovery stage. Mind that business rules sored into rule repository are design according to the business model,

For our case scenario we propose the following rules:

R1: If the booking is confirmed then the payment must be fully paid. This rule relates to Payment task and Confirm booking task (see Figure 6) and is described with pseudo logic predicate language as follows:

```
If PackageBooking.Status =
'confirmed' then
Payment.Status='Full_Paid'
```

R2 and **R3** relate to the Booking task and Payment task. These rules state that if the Flight or Hotels is booked, it must be paid:

R2: `If Flight.Status = 'booked' then Payment.Status='Paid'`

R3: `If Hotel.Status = 'booked' then Payment.Status='Paid'`

Once the rules are set up the needed data is ready to be retrieved through services endpoints. Then this data is analysed against the rules we composed.

If there is any data that violates a rule, then the related operation will be identified based on information stored in the rule repository. For example, we will focus on **R1** and suppose there are incorrect data produced of the services it scope.

Table 3: Service Log Report.

Service location	Service ID	Service instance ID	Service Endpoint	Service Operation	Data value	Record
url://	BookingPackage	3	BookingEndpoint	getBookingStatus()	'confirmed'	1055
url://	Payment	1	PaymenBookngEndpoint	getPaymentStatus()	'unpaid'	300014

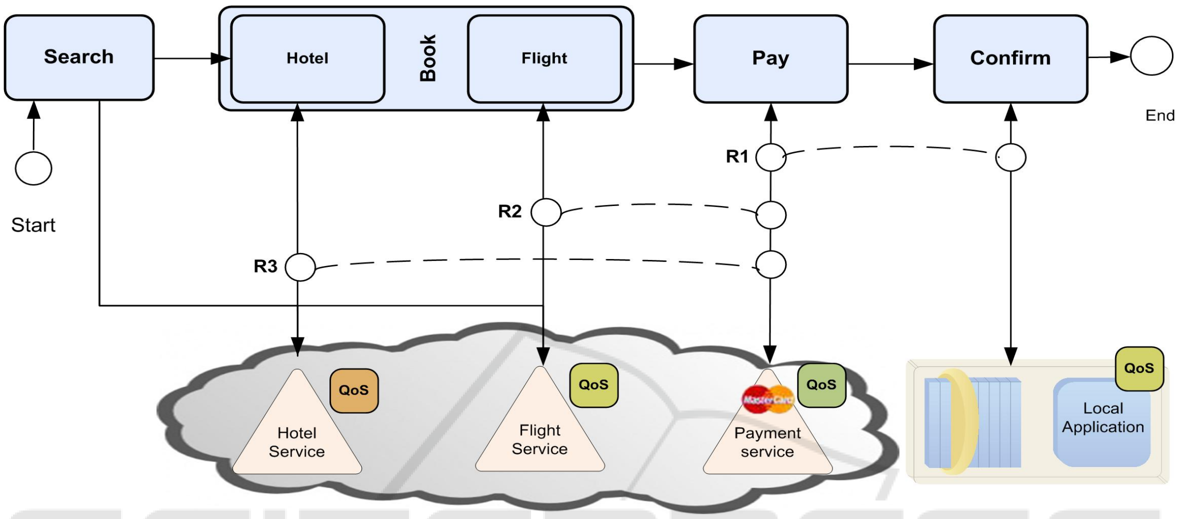


Figure 6: Booking package service composition.

Assume there is a confirmed Booking which is not fully paid. This relates to two service operation; one related to the Booking (the local one) service and the other to the Payment service. The rule R1 is related to the tasks Payment and Confirm booking. A log extraction of the occurred problem discovery is displayed in Table 3.

The service log file records that during the discovery stage the service *BookingPackage* returns through its endpoint booking status of *'confirmed'* while Payment service returns value of '*unpaid'*.

From the above information, we can identify that the cause of the incorrect data is related to the *Payment* service and the *BookingPackage* composite service.

Although this is not enough to identify which of the listed services is the problematic one. That is why we will put into further investigation our generated log file in next stage – 'Problem localization'.

4.2 Analysing: Problem Localisation Stage

In this stage we will examine produced XML log in the discovery stage, while taking into account the QoS criteria of involved services. Based on detected problem by the first module and the QoS properties of the services, the second module will be able to recommend a particular service where the problem stems from.

It is very important that Quality of Service criteria apply to all services which are part of the problem. It is also essential that QoS criteria are chosen in the context of the services and the business process. Failing to fulfil the latter requirements will result in invalid estimation by the error localization module.

In our framework we will adhere to service reliability QoS criteria. Reliability property of a service can be number of errors generated for certain period of time or time that service is out-of-order.

In this study case scenario we will take into account *number of erroneous transactions per 10000 units.*

We want to remark that this is only example criteria and there are many other criteria that services can be compared. This is very important in cases where two services have similar indexes. Bearing in mind the aforementioned QoS property and the affected services we apply our problem localization framework. Problem localization then performs a comparison based on the selected criteria and prioritise the services using '*the least reliable, the more likely to be problematic'* method. Applying to 'booking package' scenario, it will generate a report which is given in Table 4.

Table 4: QoS Report Chart.

Service name	QoS*	Priority
Booking Package	107/10000	1
Payment	12/10000	2

* incorrect transactions per 10000 units

As the QoS dimensions may differs, for every QoS criteria a new chat will be generated. The table above shows that QoS measured for booking package service is 107 incorrect transactions per 10000 committed while payment service gives only 12 per 10000. This comparison makes payment service more trustworthy than the booking package service. Therefore the source of the problem is more likely to be the Booking package Service.

Despite of our efforts to deliver approximate location of the problem, we do not disregard the chance that the error may occur in the more trustworthy service. That is why we encourage for manual investigation by the system analytic.

5 CONCLUSIONS

Inspired from research in the area of data quality and service oriented architectures, in this paper we have presented a framework for monitoring web service composition and execution. More specifically, we have separated the monitoring process into two sub processes, namely 'problem discovery' and 'problem localization'.

We have proposed a framework and illustrated relevantly every sub-process. The framework was demonstrated using a case study.

Our problem discovery framework follows a data quality management approach and incorporates business rules concept. The core of latter is based on the comparison of the business rules and the data output of the services. Problem localization framework, on the other hand, uses the output of the first framework and the functional Quality of Service criteria to provide system analytic with approximately location of the problem.

Our approach differs from others well known approaches by inspecting data delivered by the services and Quality of Service properties.

In future we aim to improve the service log technique in the discovery module as well as expand some quality of service criteria used in localization stage. We also aim to apply the concept in further case studies.

ACKNOWLEDGMENTS

This work was supported by the Irish Research Council for Science, Engineering and Technology (IRCSET) under the Postgraduate Scholarship Scheme.

REFERENCES

Agarwal V, Chafle G, Mittal S, Sribastava B (2008) *Understanding Approaches for Web Service Composition and Execution*, COMPUTE '08 Proceedings of the 1st Bangalore Annual Compute Conference, ACM

Bubenko J, Jr & Wangler B (1993) *Objectives driven capture of business rules and of information systems requirements*. Proceedings of the International Conference on Systems, Man and Cybernetics, 670

Fishman, Neal A. *Viral Data in SOA: An Enterprise Pandemic*. IBM Press, 2009.

Gaaloul W, Baïna K, Godart C (2008) *Log-based mining technique applied to web service composition reengineering, Service Oriented Computing and Applications 2*, pp.93-110.

Herbst H (1995) *A meta-model for specifying business rules in system analysis*. Proceedings of CaiSE'95, 186–199.

Jeong B, Cho H, Lee C (2009) *On the functionality quality of service (FQoS) to discover and compose interoperable web services, Expert Systems with Applications* 36, pp.5411-5418.

Morgan T (2002) *Business Rules and Information Systems: Aligning IT with Business Goals*, Addison-Wesley, Boston, MA.

Narendra NC, Orriens B (2007) *Modelling Web Service Composition and Execution via a Requirement-Driven Approach*, ACM SAC'07, Korea.

Papazoglou M.P, Traverso P, Dustdar S, Leymann F (2007) *Service-Oriented Computing: State of the Art and Research Challenges*. IEEE Computer, November, 64-71.

Silva E, Pires LF, Sinderen MV (2009) *On the support of Dynamic Service Composition at Runtime*, Springer-Verlag, ICSOC'09.

The Business Rules Group: www.thebusinessrules group.org

The Open Group: Service Oriented Architecture, http:// www.opengroup.org/projects/soa/

Thoa P., Helfert M. Monitoring Information Quality within Web Service Composition and Execution. Dublin City University, ISD 2011.

Zhang D (2004) *Web Service Composition for Process Management in E-Business*. Journal of Computer Information Systems pp.16-18.

Zeng L, Benatallah B, Dumas M, Kalagnanam J, Sheng QZ (2003) *Quality Driven Web Service Composition*, ACM WWW.

Wand Y, Wang R. (1996) *Anchoring Data Quality Dimensions in Ontological Foundations,* Communications of the ACM, November 1996. pp. 86–95

Wang R (1998*) A product perspective on total data quality management*. Communication of ACM 41