On Identifying and Reducing Irrelevant Information in Service Composition and Execution*

Hong-Linh Truong[1], Marco Comerio[2], Andrea Maurino[2], Schahram Dustdar[1], Flavio De Paoli[2] and Luca Panziera[2]

[1] Distributed Systems Group, Vienna University of Technology, Austria
{truong,dustdar}@infosys.tuwien.ac.at
[2] Department of Informatics, Systems and Communication
University of Milano - Bicocca, Italy
{comerio,maurino,depaoli,panziera}@disco.unimib.it

Abstract. The increasing availability of massive information on the Web causes the need for information aggregation by filtering and ranking according to user's goals. In the last years both industrial and academic researchers have investigated the way in which quality of services can be described, matched, composed and monitored for service selection and composition. However, very few of them have considered the problem of evaluating and certifying the quality of the provided service information to reduce irrelevant information for service consumers, which is crucial to improve the efficiency and correctness of service composition and execution. This paper discusses several problems due to the lack of appropriate way to manage quality and context in service composition and execution, and proposes a research roadmap for reducing irrelevant service information based on context and quality aspects. We present a novel solution for dealing with irrelevant information about Web services by developing information quality metrics and by discussing experimental evaluations.

1 Introduction

Identifying and reducing irrelevant information have been always one of the main goals of Web information systems. Among other possibilities, information aggregation, filtering and ranking according to user's goals can be realized by means of (composite) Web services (either based on SOAP or REST). Due to the advantage of service-oriented computing models, data sources have been widely published using Web service technologies, and several service composition and execution engines and tools have been developed to foster the composition and execution of service-based information systems. The large number of available services and the easy-to-use composition tools leads to our questions of facing irrelevant information problems (i) during the service composition phase, and

* This work is partially supported by the Vienna Science and Technology Fund (WWTF), project ICT08-032 and by the SAS Institute srl (Grant Carlo Grandi)

(ii) due to poor response of composite services. The former question is mostly related to developers and novice users who build composite services and the latter is mostly for service consumers who suffer from unwanted results delivered by composite service execution. These questions become more relevant when we consider that there are many types of data provided by data-intensive services, and, in the Internet and cloud environments, such data and services have different context and quality constraints which, if not handled properly, might cause severe decrease of perceived service quality. In this paper, we contribute with a detailed analysis of factors affecting information relevance in service composition and execution, a road-map for future research for overcoming irrelevant information using quality and context information associated with data and services, and a particular solution for filtering irrelevant service information using quality of data metrics developed for Web services information.

The rest of this paper is organized as follows: Section 2 presents motivations of this paper. Section 3 analyzes irrelevant service information problems. Section 4 presents a roadmap to solutions for these problems. Section 5 presents a novel approach for reducing irrelevant service information using information quality metrics. Section 6 presents experimental results. Related works are described in Section 7. We conclude the paper and outline our future work in Section 8.

2 Motivating Examples

Motivating Example 1 – Irrelevant Service Information in Service Composition: Let us consider the service discovery scenario in the logistic domain presented in the Semantic Web Service Challenge 2009[3]. Several logistic operators offer shipping services each one characterized by offered non-functional properties (NFPs) specified in *service contracts* (i.e., conditional joint offers of NFPs). Relevant NFPs in this domain are *payment method*, *payment deadline*, *insurance*, *base price*, and *hours to delivery*. Let us consider a developer looking for a shipping service to be included into a service composition. Assume 100 functional-equivalent services, each one associated with an average of 5 different service contracts. Therefore, a developer might have to select the best among a set of 500 service contracts without the quality of contract information. For example, a developer is unaware of timeliness (i.e., how current the service information are) and completeness (i.e., the number of available information respect to an expected minimum set) of NFPs in the service contracts. Given this lack of quality information, the developer has to perform a time-consuming task to filter information to detect irrelevant contracts, such as contracts outdated, incomplete or not applicable to user context. In our scenario, the developer must perform a manual process to evaluate the 500 service contracts and discover the incomplete ones. Moreover, the lack of automatic support to evaluate applicability conditions and the presence of outdated contracts might lead to potentially wrong selection (e.g., contracts not compliant with user context or expired contracts). We believe that quality of service information should be evaluated in

[3] `http://sws-challenge.org/wiki/index.php/Scenario:_Logistics_Management`

advance in order to reduce the presence of irrelevant information that causes waste of time and energy in the service discovery and selection.

Motivating Example 2 – Irrelevant Information between Service Composition and Execution: Let us consider a developer who wants to define a composite service out of services associated with contracts. The service composition process needs to produce a composite service contract by composing the contracts associated with the services involved into the composition [1]. At execution time the service users must follow the clauses specified in the composite service contract. Typically, a temporal distance exists between the composition (i.e., the time in which the composite service contract is created) and the execution (i.e., the time in which the contract must be enforced). A problem is that, in this time span, some information in the composite service contract might become outdated and, therefore, irrelevant when the composition needs to be executed. Therefore, it is important to detect such unsuitable information to rate the contract w.r.t. irrelevancy. Such a rate could be used to define when a service composition needs to be adapted or even discarded.

Motivating Example 3 – Irrelevant Information in Service Usage: Let us consider a novice user who wants to create a data-intensive composite Web service to manage contents from different RESTful services and to present top news headlines on current events associated with images and videos. The diversity and complexity of context constraints (e.g., indicating free for non commercial purpose, particular user country, and suitable user devices) and of quality of data and services (e.g., indicating accuracy and timeliness of images and response time of service requests) under which services and data can be used and displayed, together with the growth in development and deployment of services, have led to irrelevant information augmenting the complexity of service composition. For example, at the time of writing, `Flickr` proposes Creative Commons[4] to define licenses related to content usage that can be retrieved only after accessing the service, making this information useless for service selection. `YouTube` adopts copyright policy to cover all the published contents which is unstructured and therefore it cannot be used for automatic service selection. Quality, context and legal aspect information must be considered during service selection to fulfill user expectations. Currently, we lack the evaluation of quality and context information associated with the service and legal aspects related to service usage (e.g., data ownership), preventing us to reduce irrelevant information faced by the developers and novice users in selecting the right services.

3 Irrelevant Information Problems in Service Composition and Execution

In order to examine possible irrelevant information problems in the context of service composition and execution, we consider typical information flows inherent in service composition and execution. Figure 1 depicts actors, components and

[4] `http://www.flickr.com/creativecommons/`

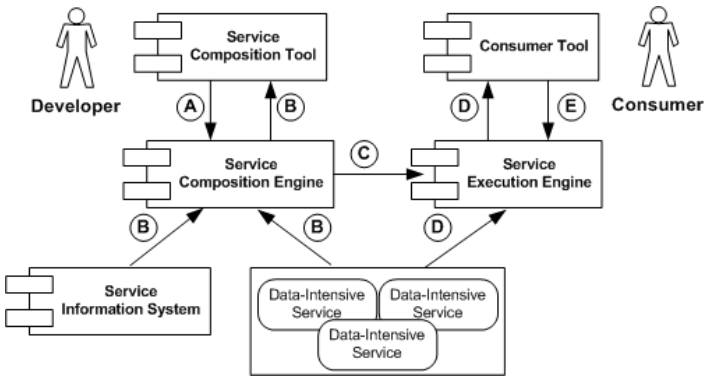

Fig. 1. Basic information flows in service composition and execution

interactions involved in service composition and execution in service-oriented computing environments, with a focus on data-intensive services, such as services providing company financial sheets and credits, images, and biodiversity data. In such environments, services offer well-defined interfaces (mostly implemented using SOAP and REST technologies) to consumers to access and store their data. In a typical service composition and execution lifecycle, the exchanged data are of:

- *Type A* - requirements about service and data schemas (functional information), NFPs, documentation, service contracts, and provenance information for the service composition.
- *Type B* - service and data schemas, NFPs, documentation, service contracts, and provenance information.
- *Type C* - information about the composite service and data provided by the composite service.
- *Type D* - data delivered by data-intensive services.
- *Type E* - information about data requested by the consumer.

These types of data are disseminated and manipulated through several abstract information flows in the service composition and execution lifecycle, as shown in Figure 1. In our analysis, these information flows are:

- *Flow 1 - developer* $\rightarrow$ *composition tool* $\rightarrow$ *composition engine*: the developer provides *Type A* information to the service composition tool and engine which support the composition process.
- *Flow 2 - services/service information system* $\rightarrow$ *composition engine* $\rightarrow$ *composition tool* $\rightarrow$ *developer*: the service and service information system, respectively, directly or indirectly, provide *Type B* information to the service composition engine and tool which filter, process, and propagate the information to the developer.
- *Flow 3 - composition engine* $\rightarrow$ *execution engine*: the composition engine passes *Type C* information to the execution engine.

– *Flow 4 - consumer → consumer tool → execution engine*: the consumer specifies *Type E* information through the consumer tool which invokes the service execution engine.
– *Flow 5 - service → execution engine → consumer tool → consumer*: the service provides *Type D* information to the execution engine which filters and processes them before returning the data to the consumer tool which presents the data to the consumer.

In all the above mentioned flows irrelevant information can exist. Irrelevant information during service composition is faced mostly by the composition developer, who relies on vast sources of information in order to construct composite services. A developer perceives this problem when information about services and data returned by the service composition engine/tool is incomparable, unsure, incomplete, or overwhelming. During the service execution, the service consumer faces irrelevant information problems when the data returned by data-intensive services are not comparable, inadequate or overwhelming. Table 1 gives some examples of irrelevant information. When we consider Internet-scale and cloud-based service composition and execution scenarios, in which services are provided by different providers, irrelevant information associated with these flows increases in many aspects due to the diversity and complexity of services and their descriptions.

Problems	**Examples**
Relevant to context and quality information models	
Unstructured description of context, quality of service, and quality of data	Several data intensive services do not provide structured description of context, quality of service and of data information [2]. Mostly, they publish such information in HTML.
Different specifications and terminologies	Several specifications with similar terminologies are used. The semantics are not the same and the specifications are not interoperable [1].
Mismatching semantics of information about services and data	Similar services are classified in different classes. Similar metric names have different meanings and different data quality metrics represent the same thing [3].
Relevant to context and quality information access APIs	
No/Limited description of data and service usage	Information about service and data licensing is not associated with service description. Unclear/no service contract clauses (e.g., data ownership) exist.
No/Limited quality of data	Quality of data (e.g., the completeness and timeliness of information about a service) not available. Services and data are registered but they are no longer available.
No API for retrieving quality and context information	Impossibility to query context and quality information directly from services [4].
No quality and context information associated with requested data	Data returned by services have not been linked with information about their usage context and quality.
Relevant to context and quality evaluation techniques	
Missing evaluation of compatibility of context and quality of multiple services	Composition tools and engines cannot deal with multiple types of context and quality information specified in different languages [1].
Large/Irrelevant data quantity	Several services returned due to the impossibility to match the context and qualities of the requested information.

Table 1. Examples of irrelevant information in service composition and execution

Causes	Effects	Flow 1	Flow 2	Flow 3	Flow 4	Flow 5
Heterogeneous specs and terminologies	Not comparable service information	X	X	X	X	X
Untrusted or low-quality data	Unsure, noisy service information		X	X		
No/Limited context and quality specifications	Incomplete service information		X	X		
Large quantity of data	Overwhelming service information		X			X

Table 2. Causes and effects of irrelevant information and information flows

Table 2 provides a mapping between causes and effects of irrelevant information problems and information flows described in Figure 1. The main causes and effects of irrelevant information problems that we have identified are:

- *Heterogeneous specifications and terminologies*: the management of information specified using different languages and terminologies causes semantic mismatching and the impossibility to apply automatic information processing in service composition and execution.
- *Untrusted or low-quality data*: the management of untrusted or low-quality information about service and data prevents from correct filtering activities, i.e., the selection of the best information according to consumer requests.
- *No/Limited context and quality specifications*: the lack of context and quality specifications about service and data prevents the implementation of information filtering because of incompleteness.
- *Large quantity of data*: the impossibility to filter information determines a large amount of data that overwhelms composition and execution tools.

4 Enhancing Context and Quality Support for Information about Services and Their Data

Existing irrelevant information problems in service composition and execution can be dealt by using several different techniques, for example, semantic matching, data mining, and similarity analysis [5–7]. In our work, we focus on how to evaluate and exchange the quality and the context of service information and utilize the quality and context for dealing with missing, ambiguous and inadequate service information used in the service composition and execution. In this paper, context information specifies situation under which services are constructed and used, the situation of the consumer, the situation under which the requested data can be used, whereas quality specifies the quality of service and quality of data provided by the service. We believe that if we are able to combine and utilize context and quality information in a unified way, several techniques could be developed to deal with irrelevant information problems. To this end, we propose the following research agenda to deal with information about services:

Topic 1 – Developing a meta-model and domain-dependent semantic representations for quality and context information specifications: by using such

representations, the problem of unstructured descriptions of quality and context information and ambiguous semantics when using multiple specifications can be reduced. Researchers have shown the benefit of linked data models for Web information and we believe that such semantic representations can help to link context and quality information for data-intensive services. Currently, context and quality information for data-intensive services are not modeled and linked in an integrated manner. The use of a common meta-model and domain-dependent ontologies should provide a partial solution to the lack of standard models and terminologies for service information specification. Furthermore, we should develop techniques to map quality and context descriptions defined using different formalisms and terminologies [1]. With such techniques the problem to compare heterogeneous service information specifications can be partially solved.

Topic 2 – Developing context and quality information that can be accessed via open APIs: these APIs should be implemented by services and service information systems. In particular, current data-intensive services do not provide APIs for obtaining quality and context information associated with their data. We foresee several benefits if services provide such APIs [8]. The composition engine could utilize such information to perform service selection and compatibility checking. The service execution engine could use context and quality to filter and select the most relevant requested information. The composition engine could improve the selection of relevant resources and services by utilizing service context and quality description together with consumer's context and quality description. Similarly, the execution engine could utilize this information to filter information according to user requests.

Topic 3 – Developing techniques for context and quality evaluation: these techniques are for context and quality compatibility evaluation and composition. They should be implemented in composition and execution engines. In particular, composition engines can utilize these techniques (i) for matching context and quality information requested by service developers and offered by service providers, (ii) for checking the compatibility among context and quality information associated with services involved in the composition and (iii) for defining the context and quality information to be associated with the composite service. Viceversa, execution engines can utilize these techniques for matching context and quality information for data requested by service consumers and offered by services. While several techniques have been developed for quality of service or context matching in service composition and execution [9–11], techniques to deal with context and quality of data in an integrated way are missing. The use of such techniques could increase the relevance of information about services and of data provided by services.

5 Reducing Services and Resources by Qualifying Non-functional Information

In order to deal with untrusted/low-quality data and large quantity of data, in this section, we present a particular solution to reduce irrelevant information

about service and data (*Type B* in Section 3). As stated before (see Figure 1), such information are exchanged between services/service information systems, composition engines, composition tools and developers.

The proposed solution consists in removing irrelevant information by evaluating the quality of information specified into service descriptions. Our solution is a concrete step in Topic 3 of the research agenda described in Section 4 since it could increase the relevance of information about services to be evaluated for service composition.

5.1 Quality of Data Metrics for Information about Service

Stimulated by information quality research, several metrics for evaluating the quality of information about services can be proposed. In this section, we propose only examples of these metrics that will be used to demonstrate the feasibility and the efficiency of our solution. An extended list of metrics is in [12].

Definition 1 (Interpretability). *Interpretability specifies the availability of documentation and metadata to correctly interpret the (functional and non-functional) properties of a service.*

This metric was originally described in [12] for data sources without concrete evaluation methods. In this paper, we extend it to service information using weighted factors and service document classification shown in Table 3.

Category	Service Information	Examples
schema	service and data schemas	WSDL, SAWSDL, pre/post conditions, data models
documentation	documents	APIs explanation, best practices
NFP	non-functional properties	categorization, location, QoS information
contract	service contracts and contract templates	service level agreements, policies, licenses
provenance	provenance information	versioning of schemas, NFPs, contracts

Table 3. Types of information used for evaluating the Interpretability metric

The *Interpretability* metric can be evaluated as follows:

$$Interpretability = \frac{\sum score(category_i) \times w_i}{\sum w_i} \quad (1)$$

where $\forall category_i \in \{schema, documentation, NFP, contract, provenance\}$, w_i and $score(category_i) \in [0..1]$ are a weighted factor (i.e., its relevance for interpretability evaluation) and the degree of available information of $category_i$, respectively. In our assumption, $score(category_i)$ can be obtained automatically, e.g., from the utilization of (document) analysis tools or from service information systems which collect, manage and rank such information.

For what concerns the evaluation of information about NFPs of a service, we propose the following QoD metrics:

Definition 2 (Completeness). *Completeness specifies the ratio of missing values of provided NFP information (NFP_p) respect to the expected minimum set (NFP_{min}).*

NFP_{min} includes all the NFPs that are considered relevant for service selection by the service developer (e.g., $NFP_{min} = \{availability, reliability, responsetime\}$).

$$Completeness = 1 - \frac{\|NFP_p \cap NFP_{min}\|}{\|NFP_{min}\|} \quad (2)$$

Definition 3 (Timeliness). *Timeliness specifies how current a NFP description is.*

Timeliness is evaluated based on the age of the NFP description and expected validation. Let $ExpectedLifetime$ be the expected lifetime of a NFP description whose age is Age. The following formula can be used:

$$Timeliness = 1 - min(\frac{Age}{ExpectedLifetime}, 1) \quad (3)$$

5.2 Filtering Service Information Using Quality of Data Metrics

Currently, most service composition tools do not support service information filtering based on QoD metrics and therefore service selection algorithms assume to have complete and clean service information. Actually, information about services is incomplete and noisy, as like many other types of information on the Web. Based on the above-mentioned QoD metrics, we illustrate two service information filters (i) based on all service documents or (ii) based on NFP descriptions. To implement the former by using the $Interpretability$ metric, we have to set weighted factors and determine scores of different categories of service documents. While scores can be evaluated based on service information provided by service information systems, weighted factors are request-specific. To implement the latter, the following steps are proposed:

- Step 1: Extract NFP_{min} and $ExpectedLifetime$ from requests;
- Step 2: Evaluate QoD metrics (e.g., $Completeness$ and $Timeliness$);
- Step 3: Establish filtering thresholds based on QoD metrics;
- Step 4: Eliminate services whose information does not meet the thresholds.

The above-mentioned information filters should be used before service selection and composition, either conducted by the developer or automatic tools, in order to support functional and non-functional matching only on relevant and high-quality service information. Note that the way to use these filters and how to combine them with other service selection and composition features are tool- and goal-specific.

6 Experiments

6.1 Reducing Services by Using the Interpretability Metric

We illustrate how the developer can detect irrelevant services by using $Interpretability$ metric. Using `seekda!` Web Services portal a service developer can discover more

than 150 Weather services[5]. We assume that `seekda!` is a service information system in our scenario. But due to the lack of well-structured documents in `seekda!`, for our experiments, we manually prepared information about the first 50 services returned by `seekda!` including service interface (e.g., WSDL file), information about documentation, availability, user rating, etc. For every service, we considered $score(schema) = 1$ as their schemas are basically a WSDL file. `seekda!` classifies service documentation to $\{none, partially, good\}$ which are equivalent to $score(documentation) = \{0, 0.5, 1\}$. We assumed $NFP_{min} = \{availability, reliability, responsetime\}$ whereas `seekda!` provides only `availability` and `response time`. Provenance information and service contract are missing.

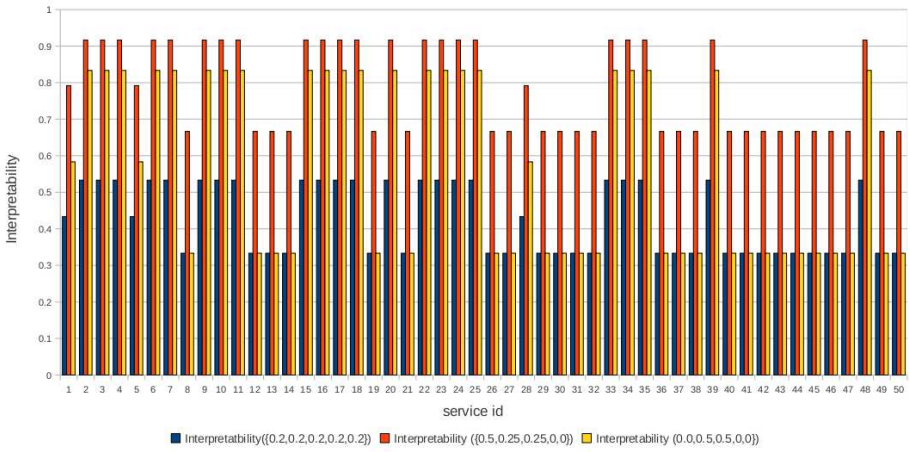

Fig. 2. Experimental values of the Interpretability metric for 50 weather services

Figure 2 describes different values of *Interpretability* metric based on different weighted factor sets: (i) all categories have the same weight (*Interpretability* {0.2,0.2,0.2,0.2,0.2}), (ii) only *schema*, *documentation* and NFP are considered and category *schema* is more important (*Interpretability* {0.5,0.25,0.25,0,0}), and (iii) only *documentation* and NFP are considered (*Interpretability* {0,0.5,0.5,0,0}). As shown in Figure 2 the values of *Interpretability* vary based on weighted factors reflecting different requirements, but in all three cases, less than a half of services have high *Interpretability* values (e.g., > 0.5). Consequently, the other half of services can be removed.

6.2 Reducing Irrelevant Services by Qualifying NFPs

In this experiment, we show how to improve the service contract selection described in Section 2. We start from the experiment where 500 WSML service

[5] search `http://webservices.seekda.com` with the keyword `weather` on 20 June 2010

contracts are ranked to find the best service according to a user request using the PoliMaR framework[6].

The filtering conditions are established on the basis of the user request. In this experiment, the user is looking for a shipping service able to satisfy specified conditions on *payment method*, *payment deadline*, *insurance*, *base price* and *hours to delivery*. Moreover, we suppose that the user submits the request on 19 June 2010 and that she is interested in service information not older than 1 year. According to the user request, the *Completeness* and *Timeliness* metrics (see Section 5.1) are applied with the following parameters: (i) NFP_{min}={*payment method, payment deadline, insurance, base price, hours to delivery*} and (ii) $ExpectedLifetime = 1year$.

In order to perform the experiments, we implemented two filters: one selects/discards WSML contracts according to specified NFP_{min} and completeness threshold; the other selects/discards WSML contracts according to specified $ExpectedLifetime$ and timeliness threshold. We performed two different experiments[7] using an Intel(R) Core(TM)2 CPU T5500 1.66GHz with 2GB RAM and Linux kernel 2.6.33 64 bits. The first experiment analyzed the time required for ranking of 500 WSML contracts in the following cases: (i) without filters; (ii) applying a filtering phase on *Completeness*; (iii) applying a filtering phase on $Timeliness$ and (iv) applying a filtering phase on *Completeness* and $Timeliness$. As an example, Table 4 reports the results of the experimentation for the following thresholds: $Completeness \geq 0.6$ and $Timeliness > 0.2$. Applying both the filters, we are able to halve the number of service contracts to be evaluated discarding irrelevant information with a reduction of the processing time equal to 52.8%.

	Filter 1	**Filter 2**	**Filtered Contracts**	**Filtering Time**	**Ranking Time**	**Total Time**
Exp. 1	no	no	500	0 sec	37.5 sec	37.5 sec
Exp. 2	yes	no	309	2.7 sec	19.9 sec	22.6 sec
Exp. 3	no	yes	395	2.2 sec	25.9 sec	28.1 sec
Exp. 4	yes	yes	246	3.5 sec	14.2 sec	17.7 sec

Table 4. Results of applying Completeness (Filter 1) and Timeliness (Filter 2) filters.

The second experiment filtered and ranked 500 WSML contracts according to different *Completeness* and $Timeliness$ thresholds. We consider the combination of the threshold values {0, 0.2, 0.4, 0.6, 0.8, 1} which are equivalent to {*not required*, *optional*, *preferred*, *strong preferred*, *required*, *strict required*}. Figure 3 shows the time required for filtering and ranking the 500 contracts. The following considerations emerge: (i) using equal threshold values for $Timeliness$ and *Completeness*, the filter on timeliness results to be more selective; (ii) applying a filter on completeness with threshold equals to 0.2, no irrelevant contracts are discharged. This means that each contract in the dataset (i.e., 500 WSML con-

[6] http://polimar.sourceforge.net/

[7] The dataset and the experimental results are available at http://siti-server01.siti.disco.unimib.it/itislab/research/experiments-for-wise10/

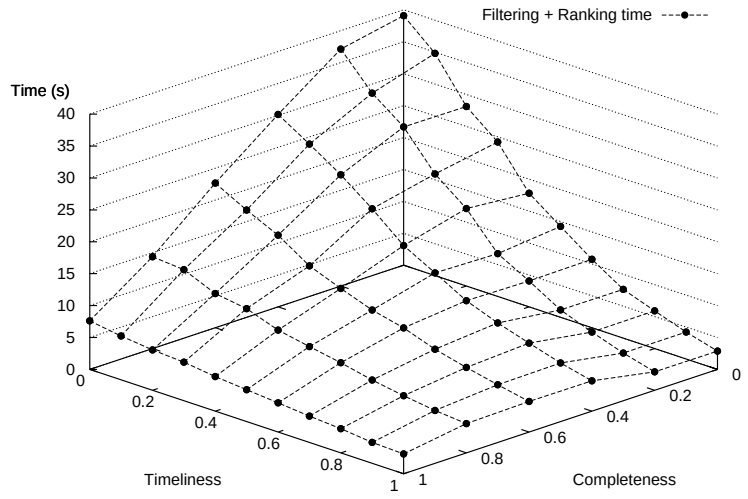

Fig. 3. Filtering and ranking time with different $Completeness$ and $Timeliness$ thresholds

tracts) contains at least one of the elements in NFP_{min}; (iii) applying a filter on $Timeliness$ with threshold equals to 1, all the contracts are discarded.

7 Related Work

Information overloading problems can be considered as a particular case of irrelevant information problem that are mainly due to the sheer volume of information that one receives and processes. Ho and Tang [13] have studied information overloading problems in five industry cases and concluded that the three major causes are information quantity, information structure, and information quality. While certain solutions to deal with information overload are generic, such as filtering or eliminating sources of information given in [14], existing solutions are targeted to information overload in personal and business organizational environments. Other proposals (e.g., [15]) to reduce irrelevant information are based on the use of tag clouds. A tag cloud is a visual representation of user-generated tags, or simply the word content of a resource description. Tags are usually hyperlinks that are listed alphabetically and by popularity. In this case, the relevance of the information is not ensured since a tag cloud interpretation is based on user intuition. Moreover, it is not clear how tag clouds can be used to describing quality of service information.

The most common way to deal with irrelevant information problems in service-oriented computing is to apply data mining or semantic computing techniques. Chen and Cohen have discussed irrelevant service information [6] faced

by developers and users in service composition by means of data mining techniques to classify and rank services. This approach, however, does not deal with dynamic and complex data sources, such as context and quality information. Semantic computing techniques, such as semantic annotation, similarity analysis, and concept analysis can be applied to improve service and data matching and selection processes [5, 7, 16]. Two limitations emerge in these techniques: (i) they concentrate on service interface and semantic meta-data about services, disregarding quality of data; (ii) they assume the availability of complete and clean service information so they cannot be used with unspecified/missing or incomplete information. This is very constrictive in particular for the approach in [16] which can require a wide set of information for the evaluation of applicability conditions associated with service contracts.

User context and quality of service (QoS) have been long considered as valuable source of information for supporting Web service design, discovery and composition [10, 11, 9, 17]. Most related works using QoS and context to deal with irrelevant information in SOC can be divided into two classes: (i) techniques to increase the relevance of service discovery, such as QoS-based Web service discovery [9], and (ii) techniques to improve the relevance of information offered by the services based on users and their interaction with a service [10], user experiences [17], and personalized Web services [11]. However, in both classes existing works do not deal with QoD and they do not combine QoD, QoS and context together in order to improve the relevance of the service discovery results.

With respect to reducing irrelevant service information, one can argue that by using existing research efforts for service ranking [9] irrelevant services can be discarded since they are associated with a lower rank. However, most of existing efforts assume the existence of high-quality service information. Our work is different since we remove this assumption and we propose techniques to identify irrelevant information before service ranking in order to reduce the service information to be processed.

8 Conclusion and Future Work

The quality of information about services is crucial to identify and reduce irrelevant information and to support efficient service selection algorithms and service execution. To overcome the lack of appropriate way to manage quality and context in Web services, we proposed a research roadmap, centered around the development of context and quality models, APIs and evaluation techniques for information about services, for reducing irrelevant information during the composition and execution of (data-intensive) Web services. We have proposed a concrete solution based on some basic quality of information metrics and demonstrated the usefulness of these metrics in reducing irrelevant service information.

Currently, we focus on the extension of metrics and the development of techniques for context and quality compatibility evaluation and on the integration of our solutions into other service selection mechanisms and service composition and design tools.

References

1. Comerio, M., Truong, H.L., De Paoli, F., Dustdar, S.: Evaluating contract compatibility for service composition in the seco2 framework. In: Proc. of ICSOC/ServiceWave '09, Stockholm, Sweden (2009) 221–236
2. Truong, H.L., Dustdar, S.: On Analyzing and Specifying Concerns for Data as a Service. In: Proceedings of The 4nd IEEE Asia-Pacific Services Computing Conference, APSCC 2009, December 7-11, 2009, Singapore, IEEE (2009)
3. Devillers, R., Jeansoulin, R.: Fundamentals of Spatial Data Quality (Geographical Information Systems series). ISTE (2006)
4. Al-Masri, E., Mahmoud, Q.H.: Discovering web services in search engines. IEEE Internet Computing **12** (2008) 74–77
5. Paolucci, M., Kawamura, T., Payne, T.R., Sycara, K.P.: Semantic matching of web services capabilities. In: ISWC '02: Proceedings of the First International Semantic Web Conference on The Semantic Web, London, UK, Springer-Verlag (2002) 333–347
6. Chen, Y., Cohen, B.: Data mining and service rating in service-oriented architectures to improve information sharing. In: Aerospace Conference, 2005 IEEE. (2005) 1 –11
7. Lamparter, S., Ankolekar, A., Studer, R., Grimm, S.: Preference-based selection of highly configurable web services. In: Proc. of the 16th International Conference on World Wide Web (WWW). (2007)
8. Truong, H.L., Dustdar, S., Maurino, A., Comerio, M.: Context, quality and relevance: Dependencies and impact on restful web services design. In: Proc. of the Second International Workshop on Lightweight Integration on the Web (ComposableWeb'10). (2010)
9. Ran, S.: A model for web services discovery with qos. ACM SIGecom Exchanges **4**(1) (2003)
10. Pernici, B., ed.: Mobile Information Systems: Infrastructure and Design for Adaptivity and Flexibility. Springer-Verlag New York, Inc., Secaucus, NJ, USA (2006)
11. Maamar, Z., Mostefaoui, S.K., Mahmoud, Q.H.: Context for personalized web services. In: Proc. of the 38th Annual Hawaii International Conference on System Sciences (HICSS), Washington, DC, USA, IEEE Computer Society (2005)
12. Batini, C., Scannapieco, M.: Data Quality: Concepts, Methodologies and Techniques (Data-Centric Systems and Applications). Springer-Verlag New York, Inc., Secaucus, NJ, USA (2006)
13. Ho, J., Tang, R.: Towards an optimal resolution to information overload: an infomediary approach. In: Proc. of International ACM SIGGROUP Conference on Supporting Group Work (GROUP), New York, NY, USA, ACM (2001) 91–96
14. Farhoomand, A.F., Drury, D.H.: Managerial information overload. Commun. ACM **45**(10) (2002) 127–131
15. Hassan-Montero, Y., Herrero-Solana, V.: Improving tag-clouds as visual information retrieval interfaces. In: InScit2006: International Conference on Multidisciplinary Information Sciences and Technologies. (2006)
16. Garcia, J.M., Toma, I., Ruiz, D., Ruiz-Cortes, A.: A service ranker based on logic rules evaluation and constraint programming. In: Proc. of 2nd Non Functional Properties and Service Level Agreements in SOC Workshop (NFPSLASOC), Dublin, Ireland (2008)
17. Kokash, N., Birukou, A., D'Andrea, V.: Web service discovery based on past user experience. In: Proc. of 10th International Conference on Business Information System (BIS). (2007) 95–107