An Approach to Classify Existing Constraints as Inputs for Web Service Composition

Amine Akhavan Sarraf
Faculty of Electrical and Computer Engineering
Shahid Beheshti University G.C.
Tehran, Iran

Hassan Haghighi
Faculty of Electrical and Computer Engineering
Shahid Beheshti University G.C.
Tehran, Iran

ABSTRACT

The selection of an appropriate Web service for a particular task has become a difficult challenge due to the increasing number of Web services offering similar functionalities. Therefore, when one wants to compose web services to catch a goal, she is faced with some preferences and constraints affecting the final configuration; for simplicity, the single terminology "constraint" in place of both "constraint" and "preference" is used throughout the paper. Most of these constraints in real applications are either functional or qualitative. These constraints have been scattered and unstructured until now, and therefore, when combining services, some of them are considered while many of them are forgotten. In addition to the possibility that some constraints are unthinkable to the user, some of them are overlapping and some may even be contradictory. In this paper a well-formed classification of all known composition constraints is presented. The classification structure is a tree whose parent-child relationships shape the proposed categorization. Leaves of the tree can contain metrics to satisfy the constraints, which are their parents. The tree structure of the classification helps one to deliver constraints as an input in the XML format to the composition process. Using a simple case study, the applicability of the presented classification structure is shown. Having this structure in place, the user can determine her constraints and their priorities more easily. Moreover, one can apply this structure to evaluate various composite services from user's point of view.

Keywords

Constraint, Service Oriented Architecture, Service Composition, Service Selection, Taxonomy.

1 INTRODUCTION

A set of individual web services which are combined using a specified and coordinated pattern and offers an improved and more ideal service forms a composite web service. Composition has emerged in the web as a chosen technology for creating inter-organizational applications. Standards currently being used for creating a process which use combined web services include BPML, BPEL4WS and OWL-S [1].

With the increase of web service providers, many web services with identical functionalities have been published in the web. In this case service applicants are required to select a service among several services with the same functionality. The problem of web service selection becomes more difficult when there is the need to execute a composite web service. With the existence of several web services for each composite web service component, the number of possible designs for combination is increased dramatically.

For applying distinction among similar web services, some constraints and preferences can be considered beforehand. Some of the constraints in real applications are functional whereas some of them are qualitative. There are also some cases which do not belong to neither of the above categories. These constraints have been scattered and unstructured until now, and therefore, when combining services, some of them are considered while many of them are forgotten. In this paper, a well-formed classification of all composition constraints is presented. This work can be regarded as one-step before service selection and composition. It is demonstrated that having this structure in place, the user can determine her constraints and their priorities more easily. Moreover, one can apply this structure to evaluate various composite services from user's point of view.

The paper is organized as follows. Section 2 introduces the problem layout. Section 3 presents our classification as a tree structure. Section 4 shows the applicability of the proposed classification structure. Finally, section 5 concludes the paper.

2 PROBLEM LAYOUT

When service composers compose several services, their goal is to create a composite service with the best possible composition according to the user's point of view. However, for creating the best possible composition, there are some constraints. What is of great importance in this process is to know that identifying the existing constraints in advance can be quite beneficial.

For example, suppose that somebody intends to do a composing operation. This person is asked what parameters she has in mind as a desired feature for her composite web service. With some thought, the user or composer can name a few features, for example the quality of her services; or service cost does not reach a specific point; or her composite web service should be compatible with specific structures and standards.

What draws our attention is that the user has no specific and predefined reference to identify the parameters constraining the composing process and to choose her desired parameters accordingly. There is the possibility that some parameters are unthinkable to the user, or some of the parameters are overlapping, or some may even be contradictory. On the other hand, assume that a composer wants to reach a secure composite web service. In this case, how her required security can be measured? Or which aspects of security does she have in mind? How important are these parameters to her? These are questions that occur to every composer's mind before the composing operation. But so far, no appropriate method and related structure for reaching the answer of these questions has been presented.

The classification method given in this paper is one step before the composing operation and presents its outcome to the selection process which is the first phase of the composing operation. In presenting the targeted tree structure, our effort has been to break down every feature to leaves which could be quantifiable by providing appropriate metrics. Of course, giving metrics for leaves is out of the scope of this paper; some appropriate metrics have been already introduced in the SOA literature.

3 TAXONOMY TREE

3.1 Composition constraints

In general, composition constraints can be classified as functional constraints and non-functional constraints. Also the non-functional category can be divided to qualitative constraints and environmental constraints. The presented taxonomy has a tree structure, and its effort is to break down each constraint into smaller parameters in its branches, in a way that quantifiable attributes are reached in the leaves. The following constraints are observed in the first two levels of the tree (figure 1). The paper proceeds by presenting further levels of the tree in the next sections, but before doing that, the next subsection shows how one can evaluate composite services based on the proposed structure.

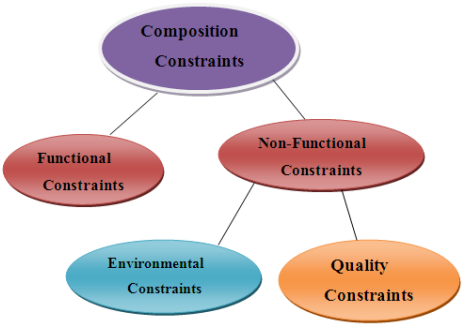


Fig 1: Composition constraints

3.2 Using Tree Structure in Composite Service Evaluation

Suppose that there are K constraints in the tree leaves. The following summation computes the value of a composite service.

$$\sum_{n=1}^{k} (W_n * S_n) \quad (1)$$

W_n shows the *priority* or *weight* of the n^{th} constraint from the user's point of view which is determined by her only once and independent from candidate composite services. Five levels of priority are considered for this parameter:

Unimportant (0) – Having Little Importance (1) – Having Average Importance (2) – Important (3) – Very Important (4)

The unimportant level with zero weight shows attributes which the user doesn't want to be considered in the calculation or, in other words, their existence is negligible for her in the composition.

Unlike W_n which is fixed for all compositions, S_n is the *score* assigned to each composite service independently. For every composition, each of the constraints presented in the tree leaves may be assigned a numerical score from 0 to 10 regarding the functionality or quality of that composition. The score will be assigned based on the classification and normalization of quantified values of the related constraint.

In simpler words, the achieved values for a constraint will be mapped to numbers between 0 and 10. For example, supposing maximum value 100% for availability, if availability of a composite service is 80%, it will be assigned a score of 8. As another example, suppose that the minimum response time of a special function could be 3 seconds. On the other hand, suppose that the maximum but yet acceptable response time is 7 seconds. Now, score 10 and 1 can be assigned for cases "response time=3" and "response time=7", respectively; response times upper than 7 seconds will be assigned score zero, and response times upper than 3 seconds and lower than 7 seconds could be assigned scores greater than 1 and less than 10 proportionally. Similar classification and normalization of quantified values could be done for every constraint.

Besides the notions of weight and score of leaves, a new concept is needed which is called *Crucial requirements* throughout the paper. Crucial requirements are those requirements vital to the composer in a manner that if they are not satisfied in the composition, the composition operation has no value to her and must be disbanded. These requirements are definable for every constraint in the tree. For example, a composer may want availability over 80%, and if availability is dropped below 80%, the composition has no value to her. Thus, in addition to giving this specific constraint a high level of weight, the composer declares a related crucial requirement. In this way, composite services with availability lower than 80% should be discarded regardless of their total value calculated by formula 1. In fact, if even one of the crucial requirements isn't fulfilled, there will be no need to calculate the summation, because the composite service would be worthless.

3.3 Functional constraints

Functional constraints are the first constraints encountered when choosing a service. In other words, the most important factor for a service composer is the functionality of the web service, which will allow it to reach its goal in the chain of existing web services. Usually, for these kinds of constraints, crucial requirements are defined. The taxonomy of these constraints is shown in the sub-tree of figure 2. In the following sections, quality and environmental constraints will be presented. The explanation of several constraints has been obtained from [2].

- **Input**

By input, the input of the final composite web service is meant. Generally, one of the existing constraints for combining services is the coordination of the output of one web service and the input of the subsequent web service. But here the input is considered as the input of the first service which is placed in the composition chain. This will be the input of the composite web service.

- **Output**

By output, the output of the final service of the composition chain is meant. This means the final output of the composite web service.

- **Pre-Conditions**

They are conditions which need to be satisfied before the operation of a web service is initiated.

- **Post-Conditions**

They are conditions which need to be satisfied after the operation of a web service.

- **Exceptions**

In the chain of web services, exceptions as part of a work flow of a process are known as constraints. Just as an operation can be known as a constraint for selecting a web service, considering cases which are not paid attention to and should be separated as an exception are also a part of the web services functionality.

  - Exception management mechanism

As mentioned, exceptions are part of the system functions which are not considered in normal situations. Therefore, there should be a mechanism for managing these conditions in the web service. On the other hand, combining several web services can introduce new exceptions; hence, creating an exception management mechanism for the final composite web service is crucial. For example, consider a case in which some data is sent from service 1 to service 2, and the data does not have a specific

definition in the functionality of the second service. In this case, this operation can be a problem for service 2. Thus, measures should be taken for the sent data so that it would not negatively affect the second service.

- **Composition Algorithm**

The composition algorithm and its definition can directly affect the selection, especially, when the composition is done in a dynamic manner. Each constraint which exists in the composition algorithm can naturally lead to new constraints of the web service selection.

- **Process Rules**

Generally the process rules which a combiner uses to create a combination of services, affects his/her selection. This part directly refers to the functionality of the web service and is considered an inseparable part of the of the combination process.

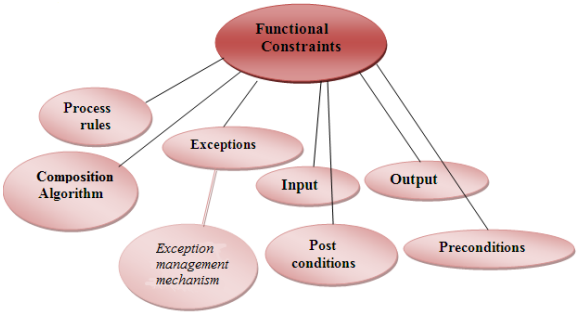


Fig 2: Functional Constraints

3.4 Quality constraints

The sub-tree for quality constraints consists of 11 quality attributes; see Figure 3. Obviously, a mentioned quality attribute is considered for the final composite web service and not for the services contributing to the composition operation. Due to space limitation, all attributes will not be explained in this paper. Each of these constraints has a specific and full explanation which is presented in reference [2]. Sub-trees of most of these attributes are given in turn via figures 4 to 12.

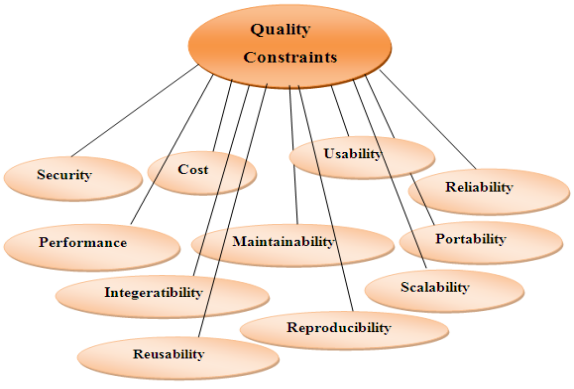


Fig 3: Quality Constraints

From the quality constraints sub-trees, the sections of cost and security are elaborated here. The details of other sections can be found in [2].

3.4.1 Cost.

The cost of web service composition can be classified into 6 main categories (figure 4):

- Service Cost (price of service) [3]
- Cost of Transaction Lost
- Execution Cost [4]
- Web Services Replacement Cost
- Network Cost
- Cost of Composition Application

❖ Service Cost

It is obvious that in order to create a composite web service, services must be bought or granted from providers individually. Therefore, the first cost that can be considered for a composition is the purchase cost of services given by providers.

Web services can be produced in return for a specific amount of money. Normally the price of a web service is determined by the provider. The price of a service can be permanent for each purchase or be measured for each service potentially and according to the required service and be finalized at the time of web service usage. Service providers can also ask for more money for services provided on faster and higher quality hardware [5].

❖ Cost of Composition Application

The creation of a composition algorithm can be done in a dynamic or static manner. Either way, an algorithm defined for this purpose must be used. Therefore gaining the algorithm and implementing it may have costs for the composers. These costs are usually called composition algorithm costs.

❖ Network Cost

Some compositions and the application of some services may required special network facilities. For example, to use a vital or real-time service, having a high speed network connection is essential. Therefore, before the composition, these constraints and costs to satisfy them should be accounted for.

❖ Cost of Transaction Lost

When a transaction is being done inside a composition of services, every service must do its job properly for the transaction to have the desired results. If any of the components of the composition are not able to do its job correctly or is unavailable when required, the transaction must restart. Other than the time consumed for the restart, the system might also suffer costs. Thus it is required to take into consideration these costs at the time of composition and, the composition process must be done in a way that in case of transaction loss, the suffering costs are minimal.

❖ Web Services Replacement Cost

As mentioned above, when an operation is being done by a composition of services, it is probable that some services may not be able to operate correctly. Therefore, it is required that one service is replaced, or a part of the web service composition is altered. This means that a new composition of several web services is substituted. This operation can have extra costs for the composer.

❖ Execution Price

For every function of a service, the execution price is the price which the user must pay to summon the service [6]. If the cost of executing a service each time is too high, the composer may not be interested in using that service in her composition. It's important to know that these costs are different from the initial costs which are paid for using the service in the first place. In some cases, the users must pay an amount of money for each time they use a service. If using a service in a composition may inflict high costs for the users, it is possible that the composers avoid using that service in their composition.

3.4.2 Security.

- Availability
  - Availability Rate
- Confidentiality
  - Authentication
  - Data Encryption
  - Firewall
  - Authorization
- Integrity
  - None-Repudiation

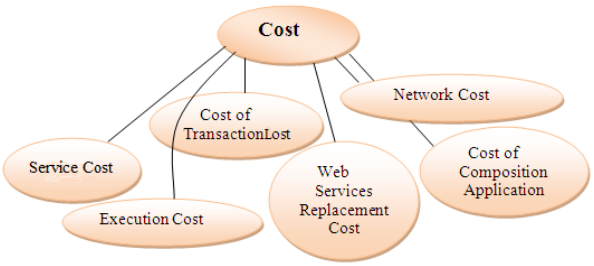


Fig 4: Cost Sub-tree

Security is one of the main concerns of web service and service oriented architecture [4]. With disregard to some expressions of security, this quality attribute is generally about providing the required conditions to prevent deliberate and accidental unauthorized access to services. Bass [3] has mentioned that "Security is the ability of a system to resist against unauthorized efforts of service utilization and the denial of service to unauthorized users, while providing service for rightful users."

Although according to software systems, security can relate to many things, in general it is classified into 3 concepts:

1. Confidentiality: the privilege of access to information/service is only offered to operators with permission.
2. Integrity: Sets of information (integrated and unified) are not prone to unauthorized changes or corruption (intentional or accidental)
3. Availability: information/services are available.

Any of the above concepts can be classified into more detailed components. Under the integrity sub-tree, the undeniability attribute has been pointed out. This attribute means that the sender of a message cannot deny sending one. Under the confidentiality sub-tree, 4 capabilities of authenticity verification, authorization, fire wall and data encryption have been mentioned. These capabilities are explained as follows:

1. Authenticity verification: confirmation of identity using one of below methods:

- "Having something": for example keys, tickets, membership card
- "Knowing something": for example a personal PIN, password
- "Being somebody": for example facial features, DNA test, fingerprint
- "Being somewhere": for example telephone number recognition, system verification for IP

2. Authorization or access control: a process for deciding which entities have permission for what activities.
3. Firewall: what walls are raised between networks to permit access according to given or prohibited access rights.
4. Data encryption: according to Bieberstein [3], encryption is the conversion of data from one structure to another using mathematical transition. Therefore, without the specific knowledge it would be unreadable.

As the final sub-tree of security, availability is seen, the degree of which a system or component is available and ready to function when required [4]. For measuring the availability of a service, the availability rate matrix can be used.

The availability of service s can be calculated using relation 2:

$$q_{av}(s) = T_a / \Theta \quad (2)$$

In which T_a is the total time (in seconds) where a service is available in the last Θ seconds. Θ is an adjustable constant. The quantity of Θ can vary based on the used application [7].

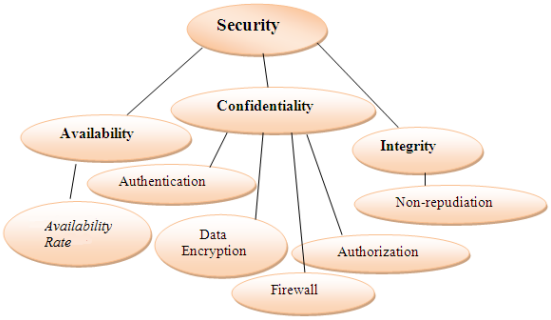


Fig 5: Security Sub-tree

Other than security and costs, other quality attributes have also been classified. In this part, only the sub-trees will be figured, and detailed explanation for each sub-tree will be avoided. For complete information on the attributes mentioned in this section refer to [2].

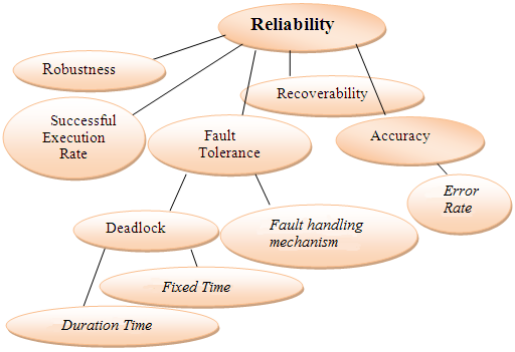


Fig 6: Reliability Sub-tree

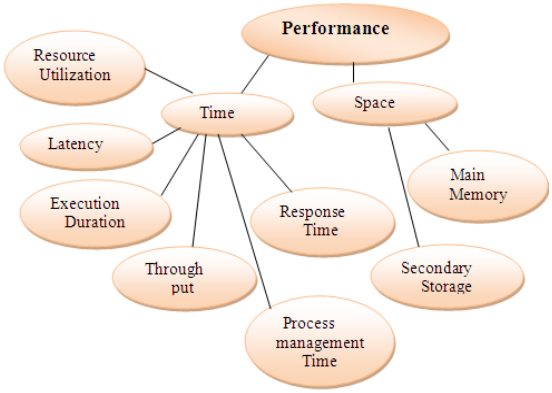


Fig 7: Performance Sub-tree

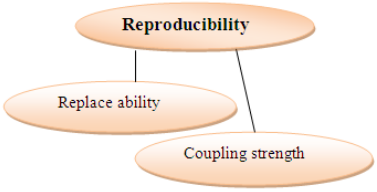


Fig 8: Reproducibility Sub-tree

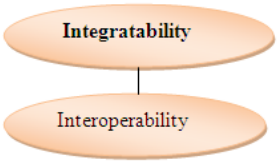


Fig 9: Integratability Sub-tree

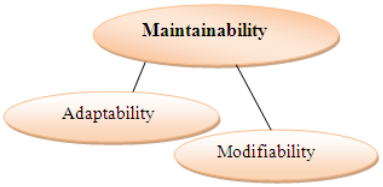


Fig 10: Maintainability Sub-tree

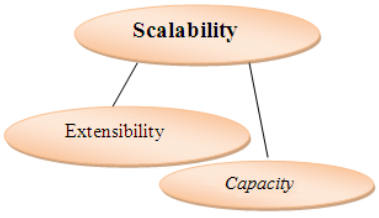


Fig 11: Scalability Sub-tree

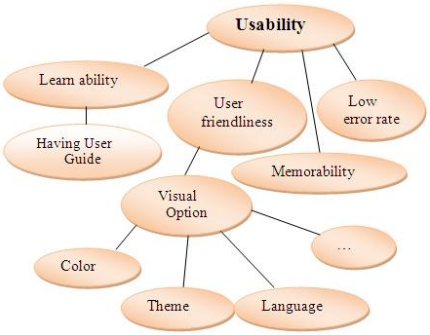


Fig 12: Usability Sub-tree

3.5 Environmental constraints

This subsection mainly addresses miscellaneous constraints, which are not classified as functional or qualitative attributes. This category of constraints consists of non-functional and non-qualitative items that are encountered in a composition, but are mainly never considered or in the best case addressed in a sporadic manner.

A main group of these constraints includes user preferences whose sub-tree consists of the users previous experiences. It's obvious that the ability for the user to implement her preferences into the composition can lead to a much better and desirable composition. The users or composers previous experiences can be a good guide to specify their required services fast and efficiently. For example, if a user is interested in a specific provider, it can be guessed that she will probably select a service from that provider in her new composition. This can be done using personal user accounts and profiles used to select services. For compositions which are done dynamically, this attribute is not efficient.

Besides user preferences, there are other environmental constraints which constitute environmental category include 6 branches (Figure 13). Web service attribute is itself the root of another sub-tree shown in Figure 14. Some nodes of Figure 13 is described as follows. The explanation of other nodes of figures 13 and 14 can be found in [6].

- Time (Date/Hour)

  The time of the service can also be very important. The existence of a time constraint for an applicant requires the service to be offered and delivered at an acceptable time. A composer might want to have a service for 24 hours for several years but the provider might not be able to offer the service in that manner. It is also possible that some services are available on special occasions like Christmas.

- Service Creation time

  The composer can consider services which have been produced after a specific date.

- Service Update time

  The composer can select services which have been updated after a specific date and are compatible with newer technology.

- Service Status

  A few of pre composition constraints can be the ability of the service to be free and active.

- Being active

  A service can be only used when active. It is possible that some service providers do not offer a service after a period of time or the service may become deactivated for any other reason [8].

- Being free

A web service can only be used when it has been removed from a previous composition (not be used in any other composition). When a new composition starts, the web services must be locked [8].

- Geographical Location

The location or geographical situation where the service is being offered can be very important to the composer. The closer the provider is to the applicants or even if the service is local, it will be much more advantageous.

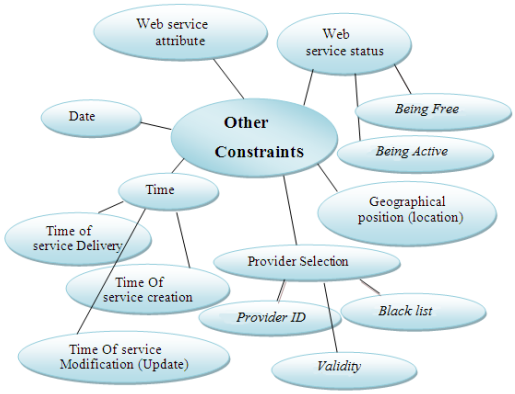


Fig 13: Other Constraints Sub-tree

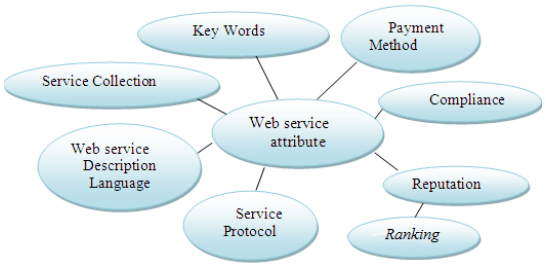


Fig 14: Web service attribute Constraints Sub-tree

4 EVALUATION

4.1 Based on a case study

To perform a case study, firstly the required scenario and related constraints for composition are determined. Then, several schemes for the selection operation are reviewed. And finally, by calculating the summation given in formula 1, the most proper scheme is selected.

4.1.1 Scenarios

Suppose that a user intends to purchase an airplane ticket from Frankfurt to New York. He intends to travel on August 20th 2012 and spend no more than 700 Euros. There are other constraints in this operation that are important for the service composer. Because of the high magnitude of the complete table of constraints, only some constraints have been presented as Table 1. Also, crucial requirements have been determined in the corresponding column by the required score of the related constraint. It is possible that the composer doesn't use normalized values to determine required scores. In this case, her desired values should be converted to normalized values (between 0 and 10) according to the method given in 3.2.

Table 1. Weights and crucial requirement of the composer

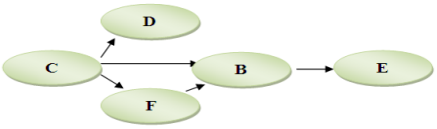
Constraint	Sub-constraints	Crucial Requirement	Weight
Functionality	Input	-	4
	Output	Score = 9	4
	Exception	-	4
	Post condition	-	3
Cost	Cost of Transaction Lost	-	3
	Web Services Replacement Cost	-	4
	Cost of composition application	-	0
	Network Cost	-	0
	Service Cost	Score < 5	2
	Execution Cost	-	3

4.1.2 Composition Schemas

Now each of the schemas is reviewed and the numerical value of each schema according to the desired preferences by the composer is calculated. Accordingly, three composition schemas based on real services will be reviewed. The source for selecting the services is the website given in [9].

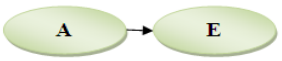
- Schema 1

In this schema, service C presents a list of existing flights. This list is presented to service D and the cost of the flights is determined. Similarly, the existing flight list is presented to service F and more flight information, such as flight class or flight line codes, are determined. Service B controls the flight information and service E register the flight.



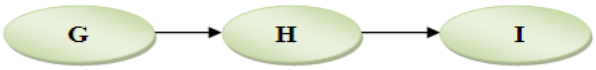
- Schema 2

In this schema flights information and their costs are received and then the flight is registered in service E.



- Schema 3

In this schema, firstly Airline information is received in service G, and the flight list of that flight line is presented. Service H receives the flight list and announces the flight numbers. Then service I receives the information and registers the flight.



In Table 2 quantitative values derived from each of the constraints for each schema are presented. These values can be

taken from mathematical calculations or from service providers and service level agreements (SLA). However, it is assumed that there are these numbers in hand. Presenting quantitative values for many of the constraints is an effortful task. According to calculations on the complete table, schema 3 showed the most success in fulfilling the user requirements. It should be noted that all three schemas fulfill the crucial requirements given in Table 1. In case one of the schemas doesn't fulfill all of the crucial requirements, it must be omitted from the comparison list.

Table 2. Scores of schemas

Constraints	Sub-constraints	Weight	Schema1 Score	Schema 2 Score	Schema3 Score
Functionality:	Input	4	9	8	8
	Output	4	9	9	9
	Exception	4	6	9	7
	Post condition	3	5	6	9
	Cost of Transaction Lost	3	6	9	8
Cost:	Web Services Replacement Cost	4	9	4	7
	Cost of composition application	0	5	5	5
	Network Cost	0	4	4	4
	Service Cost	2	2	2	2
	Execution Cost	3	4	3	3

4.2 Taxonomy evaluation in comparison with similar works

Prior to this study, much research has been done on web service composition, each defining and fulfilling functional or qualitative constraints based on their personal preferences. But none of them paid specific attention to the constraints. For example, some papers only noted time constraints. This section will compare instances of existing research in the literature and the current paper to indicate the integrity of this research. This comparison has been made in Table 3

In [10], composition is considered as a Constraint Satisfaction Problem. Generally it has an architectural outlook with special attention to cost. In [11], the key attribute of a composition is the ability of the user to have high preferences based on abstract operations. In this paper, web technology is used to show the requirements of each service in an operation. This is based on previous automated discoveries of semantic services which are based on user requirements. This paper has an architectural viewpoint and has not mentioned other constraints with detail, but has paid attention to service quality attributes.

[12] also has an architectural viewpoint and has considered some parameters, such as cost and run time, but because of its viewpoint, it has limited service quality attributes. [8] has classified constraints to pre composition, during composition and post composition. In pre composition it has paid attention to being free, being active and input-output accordance. In the during composition section, it has discussed deadlock and in the post composition section, it talks about the input and output of the final composite web service. In [13] response time, operational power, reliability, availability, and cost have been considered.

[14] has also had an architectural look into composition and has quantified service quality for services into a matrix, but has not paid attention to any details. The topics mentioned in [15] are in much more relevance to this paper but are a bit more limited. The topics discussed in that paper are execution, response time, availability, acceptance and the ability to succeed. Also cost, organizational sequence, scanning, payment methods, popularity and location have been mentioned. [16] has paid attention to constraints such as price, run time, availability and reliability. [17] has mostly emphasized on efficiency. In general, response time, reliability, cost and security have also been considered. [18] has studied an automated and dynamic composition along with flexibility. It has also discussed dynamic error management and has spoken of profession regulations to offer more flexibility for service oriented architecture. It has also studied the efficiency of processes combined with BPEL. Topics mentioned in that paper are generally technical run time failure, flexibility, and reliability. [19] which is based on ontology has mostly considered run time, summoned service cost, input-output coordination, pre conditions and effects and summoning mechanisms of unavailable services. Also in a case study, efficiency and scalability have been considered.

[20] is another work with the architectural viewpoint and uses a feature model. It has mostly concentrated on interdependence and mutual exclusion. This work describes an instance of valid configuration under feature hierarchies which presents different constraints. Components considered in this paper are message priority, synchrony, end time, delivery guarantee, message encryption, access control, message integration, regulated transmission, logging, message validation, message routing, multicast, queue and etc.

[21] mentions the user's viewpoint of the web service and considers the viewers previous experiences. In fact, this paper uses ontology to measure the accordance of a user's description with the description presented for the service. This work concentrates on cost, response time, reliability, input and output. [22] has an architectural point of view and has concentrated on topics such as response time, functional power, service rate, location, capacity and cost. [23] also has an architectural viewpoint and has focused mostly on the constraints of the establishment of service oriented architecture and its configurations, for example, the dependence of functional and nonfunctional source features and policies.

Most of the works mentioned have mainly concentrated on some limited constraints, and none of them have covered as much constraints as the current study. In fact each paper has discussed the required constraints and their satisfaction according to its needs. Table 3 shows a brief comparison of the mentioned studies. Due to limited space, all of the attributes have not been mentioned.

5 CONCLUSION

As seen in the evaluation section, the taxonomy of constraints related to the composition operation and the selection phase are items that have not been thoroughly investigated yet: these constraints have only been investigated in some cases, in a brief manner and in some specific fields. This kind of taxonomy was presented in this paper for the first time. Of course the presented taxonomy isn't claiming to be perfect, but in comparison to previous works, it covers much broader ground.

Completing this taxonomy is the first recommendation for future works. Another item worth mentioning for future study is presenting more metrics for items that were not presented so far. Also, the normalization of values derived from metrics and items that are not in a metric form can further improve the utilization of this structure.

6 REFERENCES

[1] Jafarpour, Nasrin, Khayambashi, Mohammadreza, 2009. Composite Web Service Creation Based On Users Quality Requirement, 2nd conference of Electronic city, Jahad Daneshgahi Institute of Information and Communication Technology, Tehran Municipality, Tehran, Iran.

[2] Akhavan Sarraf, Amine, 2012. An Approach to Classify Existing Constraints as Inputs for Developing Web Services Composition, MSC Thesis, Shahid Beheshti University, Iran.

[3] Annika Pettersson, October 2006. Service-Oriented Architecture (SOA) quality attributes – A research Model, MSC Thesis, University of Lund, Switzerland.

[4] Liam O'Brien, Paulo Merso and Len Bass, 2007. Quality Attributes for Service-Oriented Architectures, International Workshop on Systems Development in SOA Environments, Minneapolis, Minnesota.

[5] Moreno Marzolla, Raffaela Mirandola, 2010. QoS Analysis for Web Service Applications: a Survey of Performance-oriented Approaches from an Architectural Viewpoint, Technical Report UBLCS-2010-05, Department of Computer Science University of Bologn, Mura Anteo Zamboni 7, Bologna (Italy).

[6] Zhiqiang Fan, Li Zhang, Jufang Shen and Shouxin Wang. 2010. A User's Preference based Method for Web Service Selection, 2nd Inter. Conf. on Comp, Research and Dev. Kuala Lumpur, Malaysia.

[7] Rostampour, Ali, 2011. Metric-based evaluation of software services in service-oriented modeling phase, MSC Thesis, Shahid Beheshti University, Iran.

[8] Hu Yan and Wang Hui, 2008. Constraints in Web Services Composition, IEEE 4th International Conference on Wireless Communications, Networking and Mobile Computing WiCOM '08, Dalian, China.

[9] http://fusion.cs.uniena.de/OPOSSum/index.php?action=searchservices&showserviceid=-1.

[10] Nizamuddin Channa, Shanping Li, Abdul Wasim Shaikh and Xiangjun Fu, 2005. Constraint Satisfaction in Dynamic Web Service Composition, Proc. the 16th International Workshop on Database and Expert Systems Applications.

[11] Rohit Aggarwal, Kunal Verma, John Miller and William Milnor, 2004. Constraint Driven Web Service Composition in METEOR-S, Proceedings of the 2004 IEEE International Conference on Services Computing, Pages 23-30, IEEE Computer Society Washington, DC, USA.

[12] Ying Guan, Aditya K. Ghose and Zheng Lu, 2006. HCLP Based Service Composition, Proc. 2006 IEEE/WIC/ACM Inter. Conf. on Web Intel. and Intelligent Agent Technology, Pages 138-141, Hong Kong, China.

[13] Philip Bianco, Grace A. Lewis, Paulo Merson, 2008. Service Level Agreements in Service-Oriented Architecture Environments, Technical Report CMU/SEI-2008-TN-021, Software Endineering Institute, Carnegie Mellon University.

[14] Jiuxin Cao, Jingyu Huang, Guojin Wang and Jun Gu, 2009. QoS and Preference based Web Service Evaluation Approach, 8th Inter. Conf. on Grid and Cooper. Computing, Jiangsu Provincial Key Lab. of Network & Inf. Security, Southeast Univ, Nanjing, China.

[15] Youakim Badr, Ajith Abraham, Frédérique Biennier and Crina Grosan, 2008. Enhancing Web Service Selection by User Preferences of Non-Functional Features, 4th International Conference on Next Generation Web Services Practices, Nat. Inst. of Appl. Sci. of Lyon, Villeurbanne.

[16] Zhiyong Chen, Haiyang Wang, Peng Pan, 2009. An Approach to Optimal Web Service Composition Based on QoS and User Preferences, International Joint Conference on Artificial Intelligence, Hainan Island, China.

[17] Moreno Marzolla and Raffaela Mirandola, 2010. QoS Analysis for Web Service Applications: a Survey of Performance-oriented Approaches from an Architectural Viewpoint, Technical Report UBLCS-2010-05, University of Bologna (Italy). Department of Computer Science .

[18] MingXue Wang, Kosala Yapa Bandara, Claus Pahl, 2009. Constraint Integration and Violation Handling for BPEL Processes, 4th Inter. Conf. on Internet and Web Applications, Venice, Italy.

[19] Anna Hristoskova, Bruno Volckaert, Filip De Turck, 2009. Dynamic Composition of Semantically Annotated Web Services through QoS-aware HTN Planning Algorithms, 4th Inter. Conf. on Internet and Web Applications, Venice, Italy

[20] Hiroshi Wada and Junichi Suzuki, Katsuya Oba, 2007. A Feature Modeling Support for Non-Functional Constraints in Service Oriented Architecture, IEEE International Conference on Service Computing, Salt Lake City, Utah.

[21] Rohallah Benaboud, Ramdane Maamri and Zaidi Sahnou, 2010. User's preferences and experiences based web service discovery using ontologies, Fourth International Conference on Research Challenges in Information Science (RCIS 2010), Nice, France.

[22] Assel Akzhalova, Iman Poernomo, 2010. Model driven approach for dynamic service composition based on QoS constraints, IEEE 6th World Congress on Services, Miami, Florida, USA.

[23] Jing Luo, Ying Li, Jie Qiu, Ying Chen, 2008. Declarative Constraint Framework for SOA Deployment and Configuration, IEEE International Conf. on Web Services, Beijing, China.

Table 3. Comparing this taxonomy with similar works

Other option	Security	Input Output	Fault Tolerance	Recipient	Response Time	Capacity	Location	Throughput	Reliability	Availability	Execution Time	Cost	References
												*	[8]
											*	*	[10]
*		*											[6]
					*			*	*	*		*	[11]
*				*	*		*			*		*	[13]
									*	*	*	*	[14]
	*				*				*			*	[15]
*			*						*				[16]
*		*									*	*	[17]
		*			*				*			*	[18]
					*	*	*	*				*	[19]
*	*	*	*	*	*	*	*	*	*	*	*	*	This Work