Exploring Cohesion, Flexibility, Communication Overhead and Distribution for Web Services Interfaces in Computational Science

Miguel-Angel Sicilia and Daniel Rodríguez

Computer Science Department, University of Alcalá
Ctra. Barcelona km. 33.6. 28871 Alcalá de Henares (Madrid), SPAIN
{msicilia, daniel.rodriguezg}@uah.es

Abstract. Computational science studies often rely on the availability of large datasets through the Web. Web services (WS) provide a convenient way for making those datasets available, since they rely on a standard and widely available technology. However, there are many ways to devise a Web service interface for a given dataset, and the resulting interfaces vary in their properties related to cohesion, distribution, flexibility and communication overhead, among other parameters. This paper explores these attributes and provides some directions on how they can be measured. Concretely, a well-known cohesion metric is explored as a way to characterize the possible kinds of Web service interface designs. This is discussed for a concrete distributed context of service-oriented architectures.

Keywords: Computational science, Web services, cohesion, performance, distribution, flexibility.

1 Introduction

Computational science requires the use of large amounts of data for the construction of models and simulation analyses in order to solve scientific problems. In recent years, vast amounts of data have been made available through the Web as Web services (WS) provide a way to obtain pieces of these kinds of information through standard Web technology, and many on-line databases currently use WS interfaces [8]. Most of these interfaces provide access to schemas containing the scientific data through a collection of services that clients must call to obtain the data.

It is known that current the current way of implementing Web services based on SOAP[1] messaging over the HTTP[2] protocol incorporates some additional overheads [7]. Depending on the design of the Web services interfaces, client applications would require to issue a larger or smaller number of Web service invocations, thus incurring in more or less overhead respectively. An extreme case occurs when a single Web

[1] http://www.w3.org/TR/soap12-part0/
[2] http://www.w3.org/Protocols/

service retrieves all the data in a particular dataset, which reduces to the minimum the number of remote method calls but clearly compromises flexibility to build applications on top of it. In the opposite extreme, a design in which the data in each entity in the data model is provided by a separate Web service provides maximum flexibility. In addition, some Web service interface designs delegate complex query processing to the server while others only retrieve data, and processing needs to be done in the client side. Depending on the computing power provided by the server, one or other option could be desirable. In consequence, Web service interface design critically impacts the flexibility, performance and distribution properties of a system. These trade-offs can be analysed with metrics defined for structured or Object Oriented (OO) systems. For example, Perepletchikov et al [6] have analysed the design of 3 different designs for a Service Oriented Architecture (SOA) with traditional metrics such as Lines of Code and Cyclomatic Complexity [2, 4] and the set of OO metrics defined by Chidamber and Kemerer (C&H) [1]. As authors state, most OO metrics need to be adapted for Web services, mainly assuming that a class is a service or set of services (business process).

Designers facing the analysis or design of Web service interfaces could find beneficial the availability of metrics, indicators or guidelines for measuring designs flexibility, degree of distribution and communication overheads. Computational science applications are typically characterized by relatively stable data schemas but with large volumes of data, which implies that communication overhead is an issue for this kind of applications. Also, since new research directions would impose additional requirements, designers need to be able to adapt systems to unpredicted new uses, and thus, *flexibility* is also a requirement.

This paper attempts to delineate some possible indicators that could be used to measure how a concrete Web service design affects these variables. It explores basic design issues and the potential use of cohesion as an indicator of some properties of the quality of the design.

The rest of this paper is structured as follows. Section 2 discusses the some metrics and the reformulation of a metric for Web services. Next, we present a small case study in Section 3. Finally, Section 4 concludes the paper and points to future research directions.

2 Communications overhead, Flexibility and Web Services Design

The Web service architecture defined by the W3C enables applications to communicate over the Internet, in other words, Web services allow applications to access software components through standard Web technology. The use of Web services introduces, however, additional overheads in communications as a trade-off for the increase in flexibility they provide. In particular, due to the usage of XML, not only requests and replies are larger when compared to traditional Web interactions [5] but also the need for parsing the XML code in the requests adds additional server overhead.

Here we are concerned with computational science applications, in which (i) large amounts of data need to be transferred through the net, and (ii) the interfaces need to

be flexible to serve the needs of different applications and research needs. Web service interfaces that wrap datasets can be devised with different styles, and this affects performance due to communication overheads. In general, finer grain service calls produce additional latency due to the increase in the number of verbose SOAP messages and established connections needed. In contrast, coarse-grained interfaces will reduce network latency but are less flexible for applications that require only some specific pieces of whole information. In addition, some Web services perform complex queries against the schema that should be performed at the client side if they were not available.

These tradeoffs could be subjected to measure in an attempt to develop a metric that combines flexibility, performance and distribution when designing Web services interfaces. Since the amount of services offered for the same schema will ultimately depend on the parts of the data model that are read, *cohesion* can be used as a candidate property to develop the measures sought. In the area of OO systems, one of the metrics defined by Chidamber and Kemerer (C&K) [1] is the *Lack of Cohesion in Methods* (*LCOM*). It is as a quality metric of the cohesiveness of a class by measuring the number of method pairs that do not have common instance attributes. More formally, LCOM measures the extent to which methods reference the class instance data. Let us consider a class C_1 with n methods $M_1, M_2, \ldots, M_n$. and let $\{I_j\}$ = set of instance variables used by method M_i. There are n such sets $\{I_1\},\ldots,\{I_n\}$. Let $P = \{(I_i, I_j) \mid I_i \cap I_j = \varnothing\}$, and $Q = \{ (I_i, I_j) \mid I_i \cap I_j \neq \varnothing\}$. If all n sets $\{I_1\},\ldots, \{I_n\}$ are $\varnothing$ then let $P = \varnothing$.

$$LCOM = \begin{cases} |P| - |Q| & if\ |P| > |Q| \\ 0 & otherwise \end{cases}$$

For example, consider a class C with 3 methods (M_1, M_2, M_3) and Let $\{I_1\}$ = {a, b, c, d, e}, $\{I_2\}$ = {a, b, e}, and $\{I_3\}$ = {x, y, z}. $\{I_1\} \cap \{I_2\}$ is nonempty but $\{I_1\} \cap \{I_3\}$ and $\{I_2\} \cap \{I_3\}$ are null sets. LCOM is the number of number of null intersections – number of nonempty intersections), which in this case is 1 A high value of *LCOM* implies that there is a lack of cohesion, i.e., low similarity between the methods of a class and as a result, the class can be composed of unrelated objects. High cohesiveness of methods within a class is desirable, since classes cannot be divided and promotes the *encapsulation*. Low cohesiveness increases complexity, thereby increasing the likelihood of errors during the development process.

There are other reformulations of cohesion that can be found in the literature since the original *LCOM* C&K metric has been criticized. For example, Henderson-Sellers [3] comments that two classes can have a *LCOM=0* while one has more common attributes than the other. Also, there is no maximum value so it is difficult to interpret the values. As a result Henderson-Sellers [3] defined a modification, *LCOM-HS*, as follows. Let us consider a set of methods $\{M_I\}$ (I=1,…,m) assessing a set of attributes $\{A_j\}$ (j=1,…,a) and the number of methods that access each attribute $\mu(A_j)$, then define *LCOM-HS* is defined as:

$$LCOM - HS = \frac{\left(\frac{1}{a}\sum_{j=1}^{a}\mu(A_j)\right) - m}{1 - m}$$

The LCOM-HS range is between 0 and 2. If all methods access all attributes, then $\Sigma\ \mu(A_j) = ma$, so that *LCOM-HS = 0*, which indicates perfect cohesion. If each method access only 1 attribute then $\Sigma\ \mu(A_j) = a$ and *LCOM-HS = 1*, which indicates already a high level of lack of cohesion. Therefore, for OO systems values near 0 are preferred for this metric, where most methods refer to most instance variables.

The same measure can be extrapolated to the case of Web services, if we consider methods to be *Web Services*, and attributes to be *Entities* in the data model. Therefore, the reformulated the Henderson-Sellers *LCOM* metric for Web services, let us call this metric *LCOM-WS*, can be defined as:

$$LCOM-WS = \begin{cases} \dfrac{\left(\dfrac{1}{|E|}\sum_{j=1}^{|E|} a(e_j)\right) - |S|}{1-|S|} & if\ |S| > 1 \\ 0 & if\ |S| = 0 \end{cases}$$

When all information is retrieved in a single service, the denominator is 0, so a separate definition is provided for it. Obviously, a single service retrieving all the data is fully cohesive according to the idea behind LCOM, but it lacks flexibility; the whole set of information, which many applications possibly do not need (at least at a give instant of time), is retrieved for all calls all the time.

Same as with the original metric, a case of minimum cohesion occurs when every entity has just one associated Web service to retrieve the data from it, including intermediate tables. In this case, the result of the formula is equals to 1, yielding a high lack of cohesion value. Regarding processing in that case, the server acts as a simple processor of queries on single entities, and no join or other kind of expensive computations are carried out.

The key here is to explore why low cohesion leads to more efficient interfaces in terms of the ratio of data transferred to total communication overload. They, however, tend to be less flexible when obtaining concrete data elements. When cohesion is increased, so is the network overload as more data than needed is transferred (interfaces are of coarser granularity).

3 Case Study

For the sake of contrasting designs, we will consider a data schema ξ that for practical purposes will be considered a relational data schema. The schema is considered to be formed by a group of entities, $E=\{e_1,\ldots, e_n\}$, some of which are related by referential integrity constraints $e_j \rightarrow e_k$. Then, a maximum cohesiveness (i.e., low Lack of Cohesion) in the Web services interface will be achieved when the entire data set is retrieved in a single Web service, or horizontal portions of the dataset (subsets of the tuples in a table, for example) which take one piece of information for each of the entities. Then, here communication overheads is minimal (but maximal latency, i.e., time to transfer the whole dataset or most of it), except for

the case that horizontal slices are taken, e.g when the Web services are instructed to retrieve tuples in slice of say, 30 rows per call. Obviously, the case of minimal communication overhead approximately corresponds to download or bulk export functionality. In this case, C&K LCOM is 0, which means high cohesion.

As a way to analyze the impact of Web services design in flexibility, performance and distribution in computational science applications, we will study a concrete, relatively simple application, using the Multilocus Sequence Typing (MLST) database[3] containing genetic information of different types of organisms. Figure 1 provides a summary of one of its databases in the form of a relational schema[4]. There are some attributes that occur several times depending on the kind of organism or information represented, e.g. `allelic_profiles`. Also, the table locus is actually a set of tables, but these issues do not affect our current analysis, since we can consider the tables to be conceptual entities accessed by the WS interface.

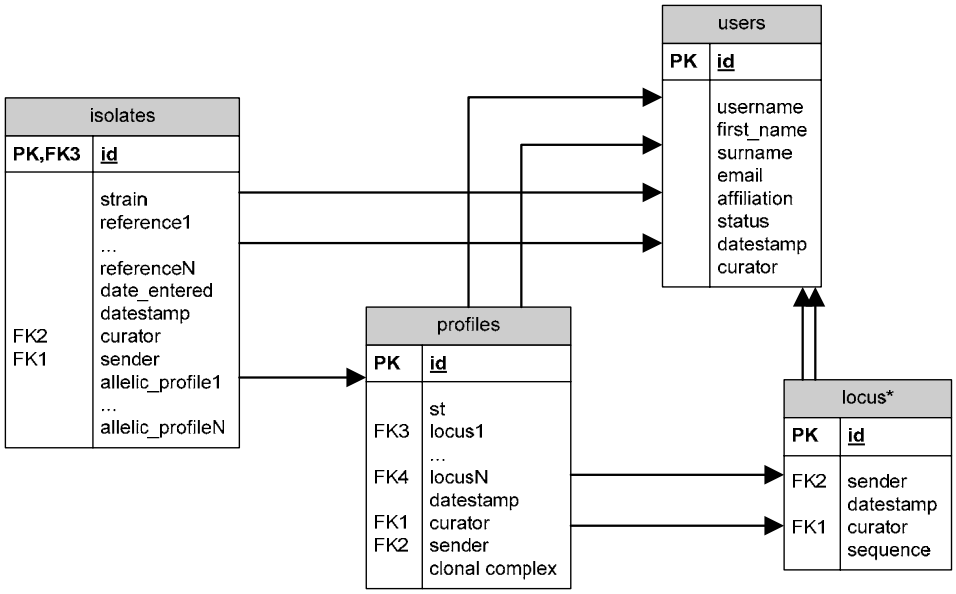

Figure 1 Database Schema for the MLST database

Table 1 provides the counts and attributes of the Web services interface provided by MLST at the time of this writing. Services marked with (*) have been excluded from the analysis since they are actually providers of URI-based ways of retrieving data, which is a mechanism not homogeneous with the rest of them. The column "attributes retrieved" identifies which attributes are the outputs of Web services. It should be noted that some of the services do not actually retrieve actual information, but information on the database schema or on the number of records (rows) contained

[3] `http://pubmlst.org/`

[4] The data model is an abstraction on the documentation of the MLST database for the purpose of analysis, we do not claim it is actually the underlying data schema of the service.

in a table, these are marked as such in the "support iteration" column, since their intent is that of providing information for calling other services. Also, the column "processing" indicates which services are considered to retrieve information from several entities, thus involving a join or other kind of processing.

Table 1 Web Services interfaces provided by MLTS

Web Service	*Entities read*	*Attributes retrieved*	*Multiplicity*	*Support iteration*	*Processing*
getIsolate	Isolates	all	1		
getIsolateCount	Isolates	none	1	yes	
getIsolateFields	Isolates	none	N	yes	
getRelatedIsolateIdsByProfile	isolates, profiles	id	N		join
getRelatedIsolateIdsByST	isolates	id	N		logical
isolateQuery	isolates	id	N		logical
Blast	locus	id	N		complex
getAlleleCount	locus	none	1	yes	
getAlleleSequence	locus	sequence	1		
getAlleleSequences	locus	sequence	No		
getAlleleSequencesURL (*)					
getForwardConsensus	locus	sequence	1		complex
getLocusLength	locus	none	1	yes	
getReverseConsensus	locus	sequence	1		complex
locusBlast	locus	id	N		complex
locusQuery	locus	id	N		complex
getClonalComplex	profiles	clonal complex	1		
getClonalComplexes	profiles	clonal complexes	N		
getProfile	profiles	Several	1		
getProfileCount	profiles	None	1	Yes	
getProfileListURL (*)					
getRelatedProfilesByST	profiles	All	N		
getRelatedProfilesByProfile	profiles	All	N		
getRelatedSTsByProfile	profiles	St	N		
getRelatedSTsByST	profiles	St	N		
getSTs	profiles	St	N		

If we apply the LCOM-WS and LCOM-CK metrics described above, the information can be contrasted with the design of the WS interface as showed in Table 2. Case (iii) in the Table is the counting for the interfaces analyzed in Table 1, and cases (i) and (ii) are the two "extreme" cases commented before. The contrast of cases (i) and (ii) shows that the former is measured as highly cohesive, but it provides the minimum flexibility since all the data is transferred in a single invocation. In contrast, the flexibility in case (ii) is maximum, in the sense that applications can obtain the concrete information blocks required down to the granularity of single entities (this can be seen considering the `getIsolate`, `getAlleleSequence` and `getProfile` services). For case (ii), cohesion is low, but cases of non-flexible interface might also yield intermediate cohesion values. For example, two Web services that obtain two unrelated parts of a data model would yield a *LCOM-WS* of

0.33. The first consequence then is that the studied *cohesion metrics cannot be used to compare any arbitrary WS interface design when flexibility is a requirement*. Flexibility is a property of design that can be inspected from the specification of WS.

Table 2 Results of the Analysis

Property	*Case (i): A single WS*	*Case (ii): A WS per entity*	*Case (iii): Actual interface*	*Case (iii): Actual interface plus one more query WS*	*Case (iii): Actual interface plus five more query WS*
LCOM-WS	0	1	0.77	0.74	0.63
LCOM-CK	0	6	274	270	254
Flexibility	Minimum	maximum	maximum[5]	=	=
Communications overhead	Minimum	maximum	intermediate	Eventually slightly increased	Eventually slightly increased
Distribution	all client side	all client side	6 out of 24	7 out of 25	11 out of 29

Table 2 shows also how the two extreme cases coincide in that they rely all the processing of data to the client, but they are opposites in communication overhead and flexibility. It should be noted that some of the services in Table 1 provide server-side processing of considerable complexity, e.g. the `blast` service provides sequence matching through heuristics with no linear computational complexity in the worst case. Then, for interfaces with maximum flexibility, the increase in cohesion comes from the addition of services that require joining information from different entities or performing complex queries, which entails delegating workload to the server side. This idea could be used to explore metrics of distribution in Web service interfaces, which are relevant for the design of solutions to complex applications. It is important to consider that data servers on the Web could be facades for parallel computing or Grid systems, so that delegating to the server is the preferred way of developing the application. In that direction, a possible additional metric for distribution could be derived from the WMC (*Weighted Methods per Class*) metric proposed by C&K [1]. Figure 2 depicts this abstract relationship.

[5] We do not consider the fact that some of the attributes in the users entity can not be retrieved with the list of services presented, since flexibility here deals with not getting undesired data only.

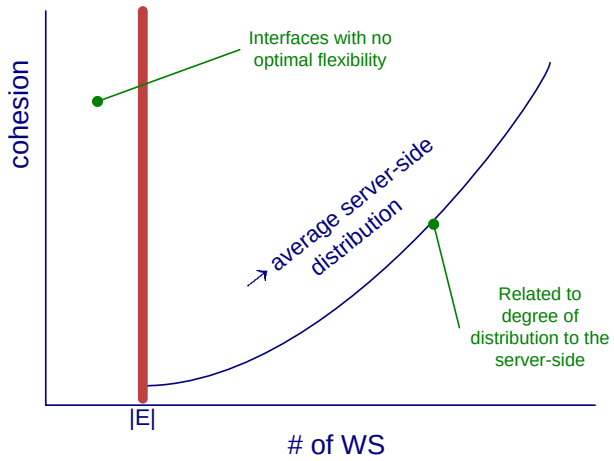

Figure 2. Relationship between number of WS, cohesion and distribution for interfaces with maximum flexibility.

4 Conclusions and Future Work

The comparison of potential designs for WS interfaces to computational science datasets reveals the importance of considering at least cohesion, flexibility, communication overload and distribution. Using the data entities as units, maximum flexibility is achieved in these interfaces by methods retrieving information for each of the entities, and measures of lack of cohesion go smaller if additional query services are provided. These services are often used in WS interfaces to distribute part of the processing to the server side, since they could be executed at the client side after getting the data. The data gathered for the case study points out that flexible interfaces will yield high values of the LCOM metric, and the provision of additional query services make these figures lower. However, less flexible interfaces would yield better cohesion figures but loosing entity-based flexibility.

Future work will deal with measuring the actual impact of additional communication overload in these interfaces as well as defining new metrics that combine trade-offs between the different properties studied in this paper.

Acknowledgements
The research was supported by grant number CCG07-UAH-TIC-1588, jointly supported by the University of Alcalá and the autonomic community of Madrid (*Comunidad Autónoma de Madrid*).

References

1. Chidamber, S.R, and C.F. Kemerer. A metrics suite for object-oriented design, *IEEE Transactions on Software Engineering*, Vol. 20, No. 6, June 1994, pp. 476-493.
2. Fenton, N.E, S.L. Pfleeger, Software Metrics: a Rigorous & Practical Approach, International Thompson Press, 1997.
3. Henderson-Sellers, Brian. *Object-Oriented Metrics Measures of Complexity*. Upper Saddle River, NJ: Prentice Hall, 1996.
4. McCabe, T.J. and A.H. Watson, Software Complexity. Journal of Defense Software Engineering, 1994. 7(12): p. 5-9.
5. Oh, S. and Fox, G.C. (2007) Optimizing Web Service messaging performance in mobile computing. *Future Generation Computer Systems* 23(4), pp. 623-632.
6. Perepletchikov, M., C. Ryan, and K Frampton, Comparing the Impact of Service-Oriented and Object-Oriented Paradigms on the Structural Properties of Software, Proc. of 2nd International Workshop on Modeling Inter-Organizational Systems (MIOS), in conjunction with the OTM 2005, Lecture Notes in Computer Science Volume 3762, October 2005, pp. 431-441
7. Tian, M., Voigt, T., Naumowicz, T., Ritter, H. and Schiller, J. (2004) Performance considerations for mobile web services, *Computer Communications* 27(11), pp. 1097-1105.
8. Tiwari, A. and Sekhar, A.K.T. (2007) Workflow based framework for life science informatics. *Computational Biology and Chemistry* 31(5-6), pp. 305-319.