J Internet Serv Appl (2012) 3:127–131
DOI 10.1007/s13174-012-0059-x

SI: FOME - THE FUTURE OF MIDDLEWARE

Data interoperability in the future of middleware

Massimo Paolucci · Berrand Souville

Received: 21 December 2011 / Accepted: 8 February 2012 / Published online: 24 March 2012

Abstract Data interoperability is one of the main problems of system interoperability. Indeed, it is estimated that the cost of data interoperability ranges in the billions of dollars every year. The traditional approach to data interoperability is to define mappings between different data structures and different data formats. Whereas this is surely a very important part of the problem, it is not its only aspect. Often systems need to aggregate data coming from different systems, and to reason and derive conclusions from these data. In this paper, we will review the efforts performed in the semantic web to unlock this problem and highlight trends and pitfalls.

Keywords Data interoperability · Semantics · Semantic Web

1 Introduction

Computer systems have been evolving from single monolithic systems built with a precise intention to solve a precise problem, to highly distributed systems of systems often assembled in ad-hoc ways. In many cases, systems of systems connect widely available systems deployed across clouds with extremely localized ubiquitous systems such as sensor networks or networks of actuators. Indeed, such systems of systems are already so common that we barely notice them. The complete cycle of airline tickets requires the interaction of systems such as flight scheduling systems, payment systems that are distributed across the Internet, and localized systems such as location sensors to help the user to specify the starting location, and boarding systems at the gate that verify that the purchased ticket is valid for the departing flight.

To enable such heterogeneity among systems of systems, different levels of interoperability are required. In this paper, we will concentrate on the data interoperability problem, which deals with the agreements on the format and the meaning of the data that allow different systems to work together. Continuing with the flight ticketing example above, the booking and payment systems will have to agree on messages describing charges to the user. Such agreements will have to specify whether any message describes a charge to the user, a payment of the user, or a reimbursement to the user. Without any of such agreements no money transfer would be possible, and with it no ticket payment In a nutshell, the data interoperability problem can be reduced to the specification of a set of agreements that are required so that data are processed correctly by the different systems.

Data interoperability is by no means a minor problem, rather its dimension and economic impact is enormous. Indeed the World Wide annual integration and data quality costs have been estimated in $1 Trillion/year,[1] and the problem is becoming worse as systems of systems are becoming the norm rather than the exception and the amount of data exchanged increases. To lower those costs, it is essential then to lower data interoperability barriers.

In this paper, we will look at interoperability problems that we encountered in many years of working with distributed systems, and look at different ways in which we addressed them and the problems that emerged. In Sect. 2

M. Paolucci (✉) · B. Souville
DOCOMO Euro-Labs, 812451 Munich, Germany
e-mail: paolucci@docomolab-euro.com

B. Souville
e-mail: Souville@docomolab-euro.com

[1] Michael Brodie; invited talk at ISWC 2003.

BBC:

```
<item>
  <title>
    Sunday: light rain, ...
  </title>
  <link> ... </link>
  <description>
    Max Temp:...
    Min Temp:...
    Wind Direction:...
  </description>
  <guid isPermaLink="false">
  ...</guid>
  <pubDate>...</pubDate>
  <geo:lat> ... </geo:lat>
  <geo:long> ... </geo:long>
</item>
```

Weather Channel:

```
<item>
  <guid isPermaLink="false"> ...
  </guid>
  <pubDate> ... </pubDate>
  <title>
  ...Weather Conditions In
Manchester...
  </title>
  <link> ... </link>
  <description>
    ...Mostly Cloudy,
    and 70 &deg; F....
  </description>
</item>
```

Fig. 1 Item of two weather RSS/Item Feeds

we will characterize the problem of data interoperability; in Sect. 3 we will look at different approaches and we will argue that we need to look deeper to data semantics to deal with data interoperability; in Sect. 4 we will look at the promise and problems of exploiting data semantics; finally, in Sect. 5 we will conclude looking at open challenges

2 Characterization of the problem

The data interoperability problem emerges whenever two systems exchange data. The problem becomes evident when one of the systems, say S_1, is replaced with S'_1, which is functionally equivalent to S_1. In this situation it is normal to expect that S'_1 will use different messages than S_1, therefore the system or systems needs to be modified to accommodate the changes and guarantee that the system of systems works correctly. In this context, data interoperability stops being a problem only when such modifications can be done automatically without requiring any programming effort.

In its most common incarnation, the data interoperation problem emerges when two systems cannot interoperate because the information that they share is encoded in two different data structures. As an example consider RSS/Feed services that report weather information shown in Fig. 1. The left side of the figure shows the RSS/Feeds published by BBC weather, while the right side shows the RSS/Feed published by Weather.com. Both feeds were collected at the same time for the same location: Manchester, UK. The example shows that although both services report the same information, namely the weather in Manchester UK, the organization of the item feeds is completely different.

Because of the differences in the format of the data exchanged, the two weather services are not automatically substitutable. Indeed, any system that has been programmed to work with BBC Weather would fail to interoperate with Weather Channel service just because the information will be misplaced. Ultimately, interoperation will require a programming effort to modify the interfaces between the weather system and its clients. Such an effort contributes to the $1T costs that our society consumes on the problem.

Although the format mismatch problem is widely studied (see [6] for a recent literature review), it is the easiest form of data interoperability since it assumes that the semantics of the messages that the systems exchange is equivalent. There are many conditions in which this assumption does no hold. A simple case is when the information expected by one system is different from the information provided by the other one. For example, one weather system may provide only condition and temperature; while the other system may provide a richer set of weather details. In more complex cases, systems may distribute information across different messages so that it is difficult to produce a data match at the message level.

These cases show that the problem of data interoperability is deeper than format mismatch problem. Rather, it deals with how the two systems cover the same information and how the information is abstracted and modified to satisfy the requirements of the two systems. Many times the incompatibility of the encoding has deep roots in the processing of the data. Often the problem is that two systems look at the same problem from two very different points of view.

An example of the latter problem is provided by VANETs (Vehicular Ad Hoc Network) systems.[2] VANETs are a class of protocols to be used in cars and other vehicles to improve driving safety and reduce traffic congestion. In the literature, two types of VANETs have been proposed [4]: Broadcast-

[2]We are in debt with Gordon S. Blair, Paul Grace and Vatsala Nundloll that provided us this example.

Based VANETs (also called Geography-based) which exploit ad-hoc networks formed by vehicles during their travel. On the other type of VANETs are the Location-based (also called Topology-based) that exploit Wide Area Networks such as the cellular network to exchange information across vehicles and uses absolute location information such as GPS to inform about the location of problem on the road.

The different routing strategies strike different trade-offs along many different parameters ranging from infrastructure requirements, power consumption, and availability and so; therefore, neither strategy will perform better under all conditions. It is therefore reasonable that systems based on these different strategies will be deployed, and that they will have to interoperate at some level. The problem is that, although the information transmitted by the two types of system is the same, for example both systems may indicate that there is a problem at a given location on the road, the different strategies impose different messages. For example, the location information may be encoded as absolute in the case of location-based VANETs and relative to a car in the case of Broadcast-based VANETs.

The data interoperation problem in this case is much more severe than in the case of format mismatch because it relates to how the data are processed by the systems. Ultimately, in these systems will not be enough to look at how information from one system maps into the information expected by the other system; rather an automatic translation will have to analyze how the information is processed by the other system.

3 Approaches

The skeptical reader may wonder whether we are spending so many words to invent a problem that does not really exist. At the end, it would be enough to standardize the message structure and the problem would immediately disappear. Anyway, there is XML that addresses the problem of data interoperability.

Standardization has been widely used of address the problem of data interoperability with efforts such as EDI (Electronic Data Interchange) [9]. The problem is that standardization alone is too weak, slow and expensive to address the problem. The deeper problem of standardization is that generating a standard takes years, a time frame that is too long for any integration work. Furthermore, the amount of data produced and transmitted World Wide is beyond what can be achieved standardization. XML facilitates the problem of data interoperability because it provides a common grammar for representing data, but it does not specify what to do with the data. Figure 1 proves the point: any system implementing an XML parser will ultimately be able to parse the data sent by both BBC Weather and Weather Channel. But parsing the data does not address the problem of interpreting the data received. The final result will be that the systems receiving the data will be able to parse it correctly but then they will be left in the awkward position of not knowing what to do with it, and in the specific example, of not knowing the weather.

The problem of data interoperability has been widely analysed by the Data-Base community that has the problem of integrating data from different databases very quickly and efficiently [5]. The lesson from databases is that systems of systems need to be able to recognize automatically the information that is encoded in the different types of data and then reason about the different encoding to derive a mapping between data structures. Essentially, what is needed are *ontologies* that model the meaning of the messages that systems exchange, so that data mappings can be derived automatically [14] at the level of the meaning of the data, rather than at the level of the data format.

4 Exploiting semantics

Within Computer Science, the universally accepted definition of was proposed by Tom Gruber, who formulated it as follows: "*An Ontology is a specification of a conceptualization*".[3] There are two problems with Gruber's definition. The first one is that any data structure provides a conceptualization of information. The second problem of Gruber's definition is that ontologies as conceptualizations can model the information that systems of systems exchange; but there is no description of the mechanisms to derive mappings between conceptualizations.

A more precise definition of ontologies relates the conceptualization proposed by Gruber with the underlying computational mechanism. From this view point, an ontology is defined by a tuple $\langle A, L, P\rangle$ where A is a set of axioms, which implements the conceptualization highlighted in Gruber's definition, L is a language in which to express the axioms, and P is a proof theory that supports the automatic derivation of consequences from the axioms.

Defining ontology in terms of conceptualization and of the supporting logics, uncovers the computational mechanisms underlying the ontology. Furthermore, since the logic imposes very strong constraints on the conceptualization, it shows how the ontology affects the interoperability of the whole system. Specifically, the *expressivity* of the language L limits the concepts that can be expressed within the ontology, and in turns it affects the *adequacy* of the ontology, i.e. the ability of the ontology to describe the domain that it wants to represent; the proof theory P limits the derivations that can be done within the ontology limiting the ability to

[3] http://www-ksl.stanford.edu/kst/what-is-an-ontology.html.

construct a mapping between the different data structures. Finally, L and P raise new interoperability problems between the logics that are used by the different systems pushing the data interoperability in a new level.

To ground our definition of ontology in existing technologies, the Semantic Web effort is currently the field of Computer Science most involved with ontologies. It defined RDF as ground language for describing the relations between data, OWL as language to express concepts and derive conclusions exploiting logics in the family of Description Logics, and finally it defines RIF as an extension of OWL and RDF to implement a rules on top of the existing relation and concept representations.

Once ontologies are available, at least as conceptual tools, the problem is to make use of them in systems of systems to address the problem of data interoperability. OWL-S [8] has been the first effort, and arguably the most influential, attempt to use semantics and ontologies to enrich descriptions of Web service descriptions with semantic descriptions of data. Essentially OWL-S provides a description of the semantics of the messages exchanged by the service (process model), and a description of how this semantics maps onto the concrete messages exchanges (grounding). OWL-S provides a starting point and works like the composition engine proposed in [7] that tackled deep interoperability issues including different types of data interoperability, but it is still very unclear how OWL-S could address architectural issues that address the VANETs problem above.

4.1 Limitations of semantics

The assumption so far is that ontologies provide a rich enough semantic model of data would address the problem of data interoperability or considerably reduce it. While this assumption is still to be proven, three important questions need to be addressed. The first one is whether there are ontologies to leverage on. The second one is whether these ontologies are authoritative enough to prevent the case of multiple conflicting ontologies; otherwise, the hope of semantics would be void. The third one is whether the underlying logic frameworks are computationally feasible.

As for the first point, currently there are a few authoritative ontologies, such as SUMO[4] (Suggested Upper Merged Ontology), standardized at IEEE, but these ontologies are not enough to address the interoperability problem. On the other side, the Link Open Data[5] (LOD) initiative is producing billions of statements of semantically marked up data, but the resulting ontologies are both logically very inexpressive, and often inconsistent reproducing at the semantics level the interoperability problems that are present at the data level.

To address the second problem, in the semantic web there is a very active subfield that goes under the label of "Ontology Matching" [3] which develops algorithms and heuristics to infer automatically the relation between concepts in different ontologies. Essentially, the goal of Ontology Matching is to construct a mapping between ontologies automatically. The result of the match is a relation and a degree *confidence* that the relation holds providing an estimate of the correctness of the matches. But the interleaving of ontology matching and systems descriptions is still by and large an open problem. An initial attempt to provide a unifying vision has been provided by the WSMO initiative [13]. WSMO stipulates the existence of different types of mediator which implement the matches found. But it did not provide any handling of the confidence value or a representation of how such confidence values affects the dependability of systems of systems [1].

The third issue is quite problematic since many of the logics that are at the bases of the semantic web, such as the family of OWL logics, have prohibitive computational complexity. Often the worst case complexity is well above exponential. Nevertheless, efficient greedy algorithms exist, and more are emerging, and OWL can be used for practical reasoning in many cases. Still, many of the algorithms gain efficiency by giving up the completeness of the results [12]. As a result, although the results returned are correct, there is no guarantee that all possible results are found.

4.2 Looking beyond semantics

The flaws of semantics described in Sect. 4.1 are considerable and researchers in the semantic web field are actively working on them. Despite them, in principle, ontologies can be used as "meta-models" of the information contained in the messages exchanged by systems of systems. As such, they can provide a way map across different data formats with essentially the same semantics. Essentially ontologies can help to map between the different types of data shown in Fig. 1.

Still the problems highlighted in Sect. 2 show that this type of matching is only the simplest form of data interoperability. Deeper problems emerge when the information contained in the messages is at different levels of abstraction, or when the information is distributed across different messages, or, as in the case of VANETS, the information exchanged depends on the approach to the problem that the different systems take. These problems go beyond the use of ontologies as "meta-models" of the information contained in the messages, and rather require a new use of ontologies as tools for reasoning about data, protocols and system architecture.

For example, matching data at different levels of abstraction often requires reasoning that *explains* the difference between two semantic representations. The inference required

[4] http://suo.ieee.org/SUO/SUMO/index.html.

[5] http://linkeddata.org/.

in this case is *abduction* [2] to find the missing axioms so that from the semantic representation of one message it is possible to derive the semantic representation of another. The problem is that abduction is in general not a valid inference because it could invent arbitrary explanations of the mismatch. As a result to address data mismatches due to abstraction is at this time a very open challenge.

In addition Sect. 2 shows that two systems may distribute their data in the protocol in very different ways. To some extent this problem can be seen as a form of service composition where the goal is to synchronize the different systems so that together they provide all information that is requested by a given protocol. Works such as [7] define a composition engine that provides an initial approach, but the deeper problem is to interleave reasoning about actions and protocols with reasoning about data. Such forms of reasoning are often difficult to interleave.

The "VANET" problem, dealing with different system assumptions is more complex because it assumes a considerable mismatch in the way the systems address the problem that they have to solve. Without addressing the root of the problem, and therefore modelling the different assumptions of the systems it is difficult to imagine how to tackle the problem. An initial semantic model of the middleware is provided in [11] and independently an initial approach to this problem is provided in [10].

5 Conclusions

The problems highlighted above fundamentally challenge the middleware community because they seem to break the neat assumptions that separate data models, behavioural models, and architectural models allowing us to reason about them separately. Specifically, as shown above we cannot address some of the data interoperability problems without reasoning about the architecture of the different systems and their protocols since architectural and protocol discrepancies are at the root of the data mismatches. More fundamentally, the question is to what extent a unifying conceptual framework exists that can support such broad range of reasoning problems combining reasoning across very different models of systems.

The second problem is that data matching introduces new forms of fuzziness in the systems that seem to challenge the traditional idea of data passing as deterministic. Above we highlighted a number of sources of fuzziness that may affect automatic data interoperability. One of them is ontology matching that matches terms in ontologies up to a given confidence value. Another is the complexity of reasoning that inevitably calls for approximate solutions. Furthermore, as discussed above, some of the inferences required by data interoperability may not be valid in general and may require restrictions to work appropriately.

From this prospective addressing data interoperability has potentially effect to force us to think of systems in ways that put data at the centre of the description rather than abstracting them out. The real challenge for the Middleware community, and Computer Science, in general is to find a way to address this problem effectively reducing the $ 1T that we spend on the data problem, avoiding the major pitfalls that are highlighted above.

Acknowledgement This work is partially supported by the FP7 ICT FET IP Project CONNECT.

References

1. Bertolino A, Di Giandomenico F (2011) Dependability assessment of dynamic connected systems. In: Bernardo M, Issarny V (eds) Formal methods for eternal networked software systems. Springer, Berlin
2. Du J, Qi G, Shen Y, Pan J (2011) Towards practical ABox abduction in large OWL DL ontologies. In: Proc of AAAI2011
3. Euzenat J, Shvaiko P (2007) Ontology matching. Springer, Berlin
4. Gerla M et al (2009) Survey of routing protocols in vehicular ad hoc networks. Gerla M et al (eds) Advances in vehicular ad-hoc networks: developments and challenges, s.l: IGI Global
5. Haas M, Lin ET, Roth MA (2002) Data integration through database federation. IBM Syst J 41:578–596
6. Haslofer B, Klaus W (2010) A survey of techniques for achieving metadata interoperability. ACM Comput Surv (CSUR) 42(2):1–37
7. Kuter U et al (2005) Information gathering during planning for web service composition. In: Web semantics: science, services and agents on the World Wide Web
8. Martin D et al (2007) Bringing semantics to web services with OWL-S. J World Wide Web 10(3):243–277
9. Narayanan S, Marucheck AS, Handfield RB (2009) Electronic data interchange: research review and future directions. Decis Sci 40(13):121–163
10. Nundloll V et al (2011) The role of ontology in enabling dynamic interoperability. In: Proceedings of the 11th IFIP international conference on distributed applications and interoperable systems
11. Oberle D (2006) Semantic management of middleware. Springer, Berlin
12. Ren Y, Pan JZ, Zhao Y (2010) Soundness preserving approximation for TBox reasoning. In: The proc of the 25th AAAI conference.
13. Roman D et al (2005) Web service modeling ontology. Appl Ontol 1(1):77–106
14. Vetere G, Lenzerini M (2005) Models for semantic interoperability in service-oriented architectures. IBM Syst J 44(4):887–903