A View Based Approach for Enhancing Web Services Availability

Hela Limam and Jalel Akaichi

Abstract With the advance of Web services technologies and the emergence of Web services into the information space, tremendous opportunities for empowering users and organizations appear in various application domains including electronic commerce, travel, intelligence information gathering and analysis, health care, digital government. Hence, Web services appear to be a solution for integrating distributed, autonomous and heterogeneous information sources. However, as Web services evolve in a dynamic environment which is the Internet many changes can occur, affect them and make them become unavailable. This presents us with the problem of substituting them, while maintaining the whole Web service functionality. In this paper, we propose an approach for solving this problem. We present several algorithms for solving several variants of this problem. Our work is illustrated with a healthcare case study.

Keywords Web services · Substitution · Schema changes · Healthcare

1 Introduction

Synchronization within highly dynamic environments such as Internet is far away from being trivial. In fact it is the trickiest environment one could image due to its unpredictable behavior. Unfortunately, there is no satisfactory solution which guaran-tees Web service availability in such a situation. It makes sense to work on a solution for the usual scenario, which, anyway, has to deal with the unavailable Web services situation due to unemployment of individual Web services caused by changes which can alter their contents. Motivation for substitution includes Web

H. Limam (✉) · J. Akaichi
BestMod Laboratory, Institut Supérieur de Gestion, Tunis, Tunisia
e-mail: Hela1.limam@laposte.net

J. Akaichi
e-mail: Jalel.akaichi@isg.rnu.tn

Á. Herrero et al. (eds.), *10th International Conference on Soft Computing Models in Industrial and Environmental Applications*, Advances in Intelligent Systems and Computing 368, DOI 10.1007/978-3-319-19719-7_2

services non-responsiveness to client requests and better arrangement with another, competitor Web service. To perform Web services substitution with less impact on the ongoing, and sometimes critical, business processes, we propose in this paper an approach based on different components and algorithms.

The solution is a mediator able to integrate information sources into Web services while addressing the substitution issue for affected Web services based on EVE framework [1].

Since EVE system proposes a prototype solution to automate view definitions re-writing [2, 3]. We proposed new algorithms for solving the problem of changes within the Web services leading to unavailability of systems and resources. Problem can be emphasized together with the growing dynamics of online services and changes of schema. Solution is based on three components including meta knowledge base, view knowledge base and web services synchronization algorithm. Approach uses service equivalents and evaluation of substitute service.

This paper is organized as follows: Sect. 2 describes the related works. Section 3 introduces the Web service model for gathering information sources. In Sect. 4, we present the Web services synchronization solution by presenting the middleware main components and our illustration by the healthcare application. In Sect. 5, we intro-duce our Web services synchronization algorithms AS2 W. Section 6 gives details of the implementation and Sect. 7 concludes our work and presents some insights for future works.

2 Related Works

In the information space, data providers are autonomous. However, they usually have control over the schemas of their information sources which raises the question of the influence of schema changes, that can render affected view definition undefined [4–6]. Different approaches for addressing this problem have been presented in the literature. Service synchronization or substitution based on the functional properties of components has been addressed by many authors [7–11].

What sets us apart from the proposed approaches is that we aim at addressing the service synchronization problem taking into account the detection of changes which can occur on information sources from which Web services are constructed. In this context, EVE project [12, 13], offers a generic framework within which a view adaptation is solved when underlying information sources change their capabilities. It neither relies on a globally fixed domain nor on ontology of permitted classes of data, both strong assumptions that are often not realistic. Instead, views are built in the traditional way over a number of base schemas and those views are adapted to base schema changes by rewriting them using information space redundancy and relaxable view queries [14]. The benefit of this approach is that no pre-defined domain is necessary, and a view can adapt to changes in the underlying data by automatically rewriting user queries, thanks to synchronization algorithms. This framework has opened up a new direction of research by identifying view

synchronization as unexplored problem of current view technology in the WWW. Our approach distinguishes itself from EVE [12] by the fact that we rely on specific advanced applications on the WWW which are Web services. Another novelty of our approach is to apply our work is the health care domain.

3 Web Services Model

In today's collaborative environment, Web services appear to be a privileged mean to interconnect applications across organizations. Web services are software systems designed to support interoperable machine-to-machine interaction over a network [15]. They are modular applications with interface descriptions that can be published, located, and invoked across the Web [16].

Different formalisms are proposed in the literature for modelling Web services. In [17–19], state machine formalism is used for the description of Web services. This choice is justified by the fact that the state machine is simple especially to describe Web services conversation. The states represent phases through it passes the service while interacting. In [20, 21], Web services are modelled using state chart diagram which is a graphic representation of a state machine. The service chart diagram is based on the UML state chart diagram to specify Web services components. None of the studied formalisms can be suitable for modelling the changes that can occur on Web services.

In this section we introduce a novel approach for modelling and specifying Web services. This approach sheds the light on two types of behaviours: presentation and dynamic parts where:

- The dynamic Web service includes information sources access using views
- The static Web service part contains the presentation components

Web service presentation and dynamic parts are executed iteratively as given in Algorithm 1.

```
                              Algorithm 1
While (true)
  {       Presentation part;
          Dynamic part;
          Vi // views call
    }
```

Web services are constructed from views which are built from distributed, heterogeneous and autonomous information sources. Each information source has its own schema composed of relations and attributes. In several cases, Web service is undefined so it should be substituted by another Web service as modelled in Fig. 1.

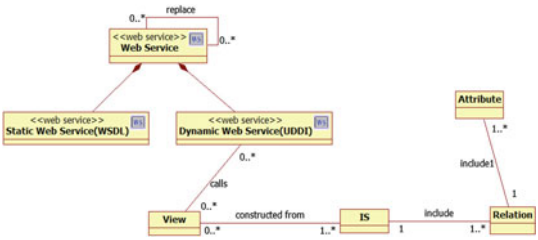

Fig. 1 Web service components relation

Table 1 Relations between web services components

Let WS be a Web service, $WS = \{V_1,\ldots,V_n\}$ With V_i: views called by Web service WS, $\lvert V_i\rvert \geq 1$, n : total number of views called by WS
Let V be a view, $V = \{IS_1,\ldots, IS_n\}$ With IS_i : information sources from which the view V is constructed. $\lvert IS_i\rvert \geq 1$, n : total number of information sources from which the view V is constructed
Let SI be an information source, $IS = \{R_1,\ldots,R_n\}$ With R_i : relations which belong to the information source IS, $\lvert R_i\rvert \geq 1$, n : total number of relations which belong to the information source IS
Let R be a relation, $R = \{A_1,\ldots,A_n\}$ With A_i : the attributes which belong to the relation R, $\lvert A_i\rvert \geq 1$, n : total number of attributes which belong to the relation R
Let WS be a Web service, $WS = \{WS_1,\ldots,WS_n\}$ with WS_i the Web service replacement list

The relations between the different components are formalized in Table 1

In several cases, Web services are unavailable so we need to substitute them. In our case Web services are undefined due to schema changes which may render views (dynamic part) undefined. So Web service substitution reach on substituting Web Service dynamic part by rewriting affected views.

Let WS be a Web service and Vi the set of views defined accessed by Web service dynamic part. We suppose that the view V is undefined after schema changes. The Web service WS is synchronized to the Web service WS' with V rewritten on V'∈ Vi according to Algorithm 2.

Algorithm 2

```
While (true)                          While (true)
{                                     {
Presentation part;                     Presentation part;
Dynamic part;                          Dynamic part';
Vi// views call                        Vi'// views call
  }                                     }
```

The substitute Web service can be equivalent (≡), superset (⊇), subset (⊆) or indifferent (≈) to the initial Web service.

- The substitute Web service is equivalent (≡) to the initial Web service, if all dynamic part views of the substitute Web service are equivalent to all dynamic part views of the initial Web service.
- The substitute Web service is a superset (⊇) of the initial Web service, if at least one of the dynamic part views of the substitute Web service is a superset of one of the dynamic part views of the initial Web service.
- The substitute Web service is a subset (⊆) of the initial Web service, if at least one of the dynamic part views of the substitute Web service is a subset of one of the dynamic part views of the initial Web service.
- The substitute Web service is indifferent (≈) of the initial Web service, if all dynamic part views of the substitute Web service are indifferent to all dynamic part views of the initial Web service.

4 Web Services Synchronization Framework

Web services are built from distributed, heterogeneous, autonomous information sources which change continuously not only contents but also their schema which may render Web services undefined. We propose therefore a synchronization process which consists of detecting schema changes and substituting affected Web services. Only the two operations attribute deletion and relation deletion affect Web services. The Web service synchronization algorithm searches possible substitution of the affected component (attribute or relation) using WSMKB constraints and WSVKB constraints. Our solution takes the form of a middleware connecting Web services to information sources as shown in Fig. 2.

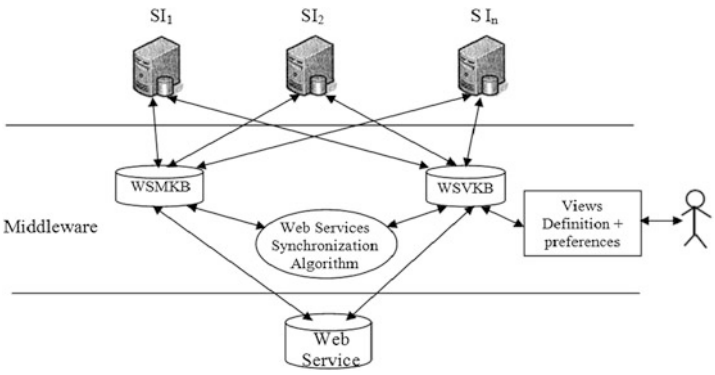

Fig. 2 The system architecture

4.1 Web Services Meta Knowledge Base (WSMKB)

Information sources joining the system must provide its structures and its contents to be stored in the WSMKB (Fig. 3). Relationships between information sources have to be added to WSMKB as substitution rules as given in Figs. 4, 5 and 6. The WSMKB constraints are represented respecting a model called MISD [3]. WSMKB can be organized as follow:

- WS (WSid, WSISidList): Web services with information sources from which they are built as given in Fig. 7.
- IS (ISid, ISRidList): information sources and their included relations. Relations (Rid, AttList): relations and their included attributes.
- The relationships between information sources or substitution constraints such as type integrity constraints, join constraints and partial/complete constraints.
- Replacement (WSid, WSreplacementList): Web services and their substitution Web services list as given in Fig. 8.

In the following, we give an example of healthcare application. Each information sources have their own schemas and contents.

S1	Patient (IdP, Name, Age, Tel, IdH) Doctor (IdD, Name, Speciality) Hospital (IdH, Name, Localization) Doctor_Hospital (IdD, IdH) Diagnostic (IdP, IdD, DateT, Result) Operation (IdP, IdD, DateO, Result)
S2	Patient (IdP, Name, Age, Tel, IdH, Med_Resp) Doctor (IdD, Name, Speciality, IdS) Hospital (IdH, Name, Localization) Doctor_Hospital (IdD, IdH) Diagnostic (IdP, IdD, DateT, Result) Operation (IdP, IdD, DateO, Result) Service (IdS, Speciality)

Fig. 3 Information sources schemas

A type integrity constraint of a relation $R(A_1,\ldots,A_n)$ states that an attribute A_i is of domain type $Type_i$. It allows verifying substitution possibility of an attribute by another while synchronizing Web services. A type integrity constraint is defined as follow: $\mathbf{TC_{R(A1,\ldots,An)} = R(A_1,\ldots,A_n) \subseteq A_1(Type_1) \times \ldots \times A_n(Type_n)}$

The type integrity constraints are expressed in the following

TC	Type integrity constraints
TC1	TCS1.Patient(IdP, Name, Age, Tel, IdH) = Patient (IdP, Name, Age, Tel, IdH)⊆ IdP(Number)×Name(String)×Age(Number)×Tel(Number)×IdH(Number)
TC2	TCS1.Doctor(IdD, Name, Speciality) = Doctor (IdD, Name, Speciality)⊆ IdD(Number) ×Name(String) ×Speciality(String)

Fig. 4 Type integrity constraints

Join constraint between two relations R1 and R2 states that attributes in R1 and R2 can be joined. It allows verifying substitution possibility of a relation by another while synchronizing Web services. Join constraint between two relations R1 and R2 is defined as follow: $\mathbf{JC_{R1,R2} = (C_1 \text{ AND } \ldots \text{AND } C_n)}$

In the following we state the list of the join constraints related to our example

JC	Join constraints
JC1	S1.Patient.Name = S2.Patient.Name
JC2	S1.Patient.Name = S3.Patient.Name

Fig. 5 Join constraints

Partial/complete constraint between two relations R1 and R2 states that the relation R1 (or a fragment of the relation R1) is a subset, a superset or equivalent to the relation R2 (or a fragment of the relation R2). Partial/complete constraint is defined as follow: $\mathbf{PC_{R1,R2} = (\pi_{Ai1,\ldots,Aik}(\sigma_{C(Aj1,\ldots,Ajp)}R1)\ \theta\ \pi_{An1,\ldots,Ank}(\sigma_{C(Am1,\ldots,Aml)}R2))}$

PC	Partial/ complete constraints
PC1	$PC_{S1.Patient,S2.Patient} = (\pi_{IdP,\ Name,\ Age,\ Tel}(S1.Patient) \subseteq \pi_{IdP,\ Name,\ Age,\ Tel}(S2.Patient))$
PC2	$PC_{S1.Doctor,S2.Doctor} = (\pi_{IdD,\ Name,\ Speciality}(S1.Doctor) \subseteq \pi_{IdD,\ Name,\ Speciality}(S2.Doctor))$
PC3	$PC_{S1.Hospital,S2.Hospital} = (\pi_{IdH,\ Name,\ Localization}(S1.Hospital) \supseteq \pi_{IdH,\ Name,\ Localization}(S2.Hospital))$

Fig. 6 Partial/complete constraints

(WS1, {S1, S2}): The Web service WS1 is construct from information sources S1 and S2
(WS2, {S1, S2}): The Web service WS2 is construct from information sources S1 and S2
(WS3, {S3}) : The Web service WS3 is construct from information sources S3

Fig. 7 Relation between Web services and information sources

(WS1, {WS2, WS3}): The Web service WS1 can be replaced by the Web service WS2 or WS3
(WS2, {WS3}): The Web service WS2 can be replaced by the Web service WS3

Fig. 8 Web services substitution constraints

4.2 Web Services View Knowledge Base WSVKB

Views definition using E-SQL and relations between Web services and its accessed views are given in Fig. 11. E-SQL [3] language allows user preferences inclusion in views definition to indicate how views can evolve after schema changes.

In an E-SQL query, each attribute, relation or condition has two evolution parameters. The dispensable parameter indicates if view components can be conserved (XD=False) or dropped (XD=True) from the substitute view. The replaceable parameter indicates if view components can be substituted (XR=True) or not (XR=False). Here X can be an attribute, a relation or a condition and the default value is False.

View extension parameters VE proposed by E-SQL states that the substitute view can be equivalent (≡), superset (⊇), subset (⊆) or indifferent (≈) to the initial view.

WSVKB contents can be organized as follow:

- VIEW (VDid, VDText): View definition using E-SQL.
- WS (WSid, VDidList): Web services and their views definition list.

The synchronization process consists on detecting schema changes (relations or attributes deletion), detecting affected Web services and applying synchronization algorithm to determine possible substitution of the affected Web services.

Suppose that Name attribute from the relation S1.Doctor is deleted, then it is substituted by Name attribute from the relation S2.Doctor since [TCS1.Doctor(IdD, Name, Speciality)=Doctor(IdD, Name, Speciality) $\subseteq$ IdD(Number) $\times$ Name (String) $\times$ Speciality(String)] and [TCS2.Doctor(IdD, Name, Speciality, IdS) = Doctor(IdD, Name, Speciality, IdS) $\subseteq$ IdD(Number) $\times$ Name(String) $\times$ Speciality (String) $\times$ IdS(Number)] and [PCS1.Doctor,S2.Doctor=(πIdD, Name, Speciality (S1.Doctor) $\subseteq$ πIdD, Name, Speciality(S2.Doctor))] and [S1.Doctor.Name = S2. Doctor.Name]. The view definition of V1 becomes:

```
CREATE VIEW V1′ VE=′⊇′ AS
SELECT D.IdD, D2.Name (AD=false, AR=true)
FROM S1.Doctor D (RD=false, RR=true, S2.Doctor D2
(RD=false, RR=true)
 WHERE(D.Speciality= "Cardiologist") (CD=false, CR=true)
AND (D.IdD = D2.IdD);
```

5 Web Services Synchronization Algorithms

Web services are composed by presentation and dynamic parts including information sources access using views call. As previously said dynamic part includes services gathered from information sources, the latter change continuously which may render views undefined then may render Web services undefined and inaccessible. So these Web services must be substituted by other ones.

Web services synchronization consists on substituting affected Web services referring to constraints embodied into the WSMKB and into the WSVKB. So synchronization process consists of detecting change and according to this change Delete_Attribute procedure or Delete_Relation procedure will be executed as given in substitution Algorithm. Only the two operators delete attribute and delete relation are treated by our algorithm.

```
Substitution Algorithm
Step 1: Identify the type of change (relationship removal
or attribute deletion)
Step 2: Check the characteristics of the attribute or
relationship (essential or replaceable) in the view base.
Step 3: Search in the Knowledge Base the Web service
related to the attribute or relationship.
Step 4: Go to view base to verify the characteristics of
Web services (essential or replaceable).
Step 5: Find substituent of the attribute or relationship
in question.
Step 6: Do substitution
```

6 Performance Evaluation

A prototype of the proposed system has been implemented. We used AXIS 1.1 which is Java platform for creating and deploying web services applications for creating Web services. The simulation of the case study serves the purpose of determining a good trade-off between the need of speed change detection and a need of a quick replacement of affected Web services. To achieve this, we tested the substitution system with different monitoring intervals (2 s–6 s) in order to compare the curves of average response time for both detection of changes and replacement of affected Web services. The response time was almost solely impacted by the monitoring interval and therefore was growing proportionally with it (Fig. 9).

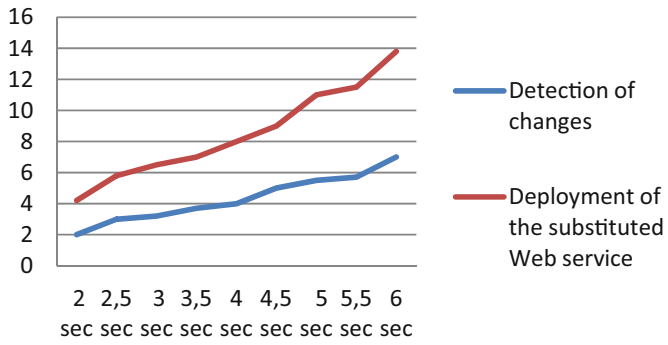

Fig. 9 Average response time

7 Conclusion

In this paper we proposed a solution to the problem of Web services synchronization caused by changes which can occur to information sources from which Web services are built and which may render Web services partially or totally inaccessible.We have presented as solution a middleware connecting Web services to information sources. The middleware is composed by a Web service Meta Knowledge Base WSMKB, a Web Service View Knowledge WSVKB and Web services synchronization algorithms. Our model proved the feasibility of marrying Web services concepts, and the EVE approach [12] which offers a solid foundation for addressing the general problem of how to maintain views in dynamic environments. Future work focus on a total synchronization of Web Services and will not be limited to the two operations attribute deletion and relation deletion which affect Web service. We also intend to develop algorithms for view maintenance of the view extent under both schema and data changes of the information sources.

References

1. Lee AJ, Nica A, Rundensteiner A (2002) The EVE approach: view synchronization in dynamic distributed environments. IEEE Trans Knowl Data Eng 14(5):931–954
2. Zhang X, Rundensteiner EA, Ding L (2001) PVM: Parallel view maintenance under concurrent data updates of distributed sources. In: Proceedings in Data Warehousing and Knowledge Discovery, pp 230–239
3. Lee AJ, Nica A, Rundensteiner EA (1997) The EVE framework: view evolution in an evolving environment, technical report WPI-CS-TR-97-4. Department of Computer Science, Worcester Polytechnic Institute
4. Blakeley JA, Larson P-E, Tompa FW (1986) Efficiently updating materialized views. In: Proceedings of SIGMOD, pp 61–71
5. Zhuge Y, Garcìa-Molina H, Wiener JL (1996) The strobe algorithms for Multi-SourceWarehouse consistency. In: International Conference on Parallel and Distributed Information Systems, pp 146–157
6. Agrawal D, El Abbadi A, Singh A, Yurek T (1997) Efficient view maintenance at data warehouses In: Proceedings of SIGMOD, pp 417–427
7. Benatallah B, Casati F, Toumani F (2006) Representing, analysing and managing web service protocols. Data Knowl Eng 58:327–357
8. Bordeaux L, Salaun G, Berardi D, Mecella M (2005) When are two web services compatible. Lect Notes Comput Sci 3324:15–28
9. Liu F, Zhang L, Shi Y, Lin L, Shi B (2005) Formal analysis of compatibility of web services via ccs. In: Proceedings of the International Conference on Next Generation Web Services Practices, IEEE Computer Society, pp 143
10. Martens A, Moser S, Gerhardt A, Funk K (2006) Analyzing compatibility of bpel processes. In: International Conference on Internet and Web Applications and Services, p 147
11. Pathak J, Basu S, Honavar V (2007) On context-specific substitutability of web services. In Proceedings of the International Conference on Web Services, IEEE Computer Society, pp 192–199
12. Rundensteiner EA, Lee AJ, Nica A (1997) On preservingviews in evolving environments. In: Proceedings of 4th International, Athens, Greece, pp 13.1–13.11

13. Koeller A, Rundensteiner EA (2000) History-Driven View Synchronization. Springer, Greenwich, pp 168–177
14. Nica A (1999) View evolution support for information integration systems over dynamic distributed information spaces. Ph.D. thesis, University of Michigan, Ann Arbor (in progress)
15. W3C (2004). Web services architecture. http://www.w3.org/TR/ws-arch/
16. Dollimore J, Kindberg T, Coulouris G (2005) Distributed systems concepts and design, 4th edn. Addison Wesley, Pearson Education
17. Benatallah B, Casati F, Toumani F (2004) Web service conversation modeling. IEEE Computer Society
18. Benatallah B, Casati F, Toumani F, Hamadi R (2003) Conceptual Modeling of Web Service Conversations. In: CAiSE. Springer
19. Ponge J (2005) Modeling and Analyzing Web Services Protocols
20. Benslimane D, Maamar Z, Ghedira C (2005) A view-based Approach for tracking composite web services. In: Proceedings of the Third European Conference on Web Services (ECOWS 2005)
21. Maamar Z, Benatallah B, Mansoor W (2003) Service chart diagrams - description and application. In Proceedings of The Alternate Tracks of The 12th International World Wide Web Conference (WWW'2003)

http://www.springer.com/978-3-319-19718-0

10th International Conference on Soft Computing Models in Industrial and Environmental Applications

Herrero, A.; Sedano, J.; Baruque, B.; Quintián, H.; Corchado, E. (Eds.)

2015, XIX, 486 p. 165 illus., Softcover
ISBN: 978-3-319-19718-0