Asian Journal of Information Technology 12 (2): 60-69, 2013
ISSN: 1682-3915

A Fault Ontology for Managing Run-Time Faults in Web Services

K. Jayashree, Sheila Anand and R. Chithambaramani
Rajalakshmi Engineering College, Chennai, India

Abstract: Detecting faults at run-time is one of the crucial elements in fault management of web services. In this study, researchers present a generic ontology for describing run-time faults in web services. Ontology descriptions have been developed to describe the semantics of generic Service Oriented Architecture (SOA) faults. The fault ontology includes various run-time faults that occur during publishing, discovery, binding, composition and execution of web services. The ontology can be used to provide semantic error information to the user to aid them in proper identification and correction of errors that occur at run-time. Researchers have tested the ontology using a sample web service application and present the results.

Key words: Web service faults, fault ontology, publishing fault, binding fault, discovery fault, composition fault, execution fault

INTRODUCTION

Web Services (WS) are self describing, self contained and loosely coupled applications that are used across the Internet using standards such as Simple Object Access Protocol (SOAP), Web Services Description Language (WSDL) and Universal Description Discovery and Integration (UDDI) (W3C, 2004). A composite web service invokes one or more web services and combines their functionality to offer service to the users (Charfi *et al*., 2009). Consumers of WS want uninterrupted and reliable service from the service providers. To provide services to the level expected by the users, it is essential to detect all faults and failures and take appropriate action to enable continuity of the services offered (Alam, 2009).

Ontologies have been widely used in many areas such as knowledge management, content management, intelligent databases, electronic commerce and the semantic web (Fensel *et al*., 2001). Ontology plays a key role in the semantic web by providing machine readable vocabularies and the relationships among them (McIlraith and Martin, 2003). Applications can use these ontologies to search, merge and intelligently interpret information. Ontology plays a major part in solving the problem of interoperability between applications across different organizations by providing a shared understanding of common domains (Taye, 2010). Ontologies enrich web services with expressive and computer interpretable languages (Yu *et al*., 2008). Several ontologies have been developed for web services and SOA architecture. In this study, researchers present an ontology that has been developed specifically to describe faults that occur during run-time of web services and provide semantic descriptions to enable appropriate action.

LITERATURE REVIEW

Service Oriented Architecture consists of three key components, namely, Service Consumer (SC), Service Registry (SR) and Service Provider (SP) as shown in Fig. 1 (Erl, 2005). Service providers register their services for public use in service registries using the publish operation. UDDI provides a platform-independent standard framework for describing and publishing web services for discovery over the internet (Curbera *et al*., 2002). Web services are expressed in WSDL, an XML-based language for describing web services and how to access them. WSDL is a document written in XML which describes a web service and specifies the location

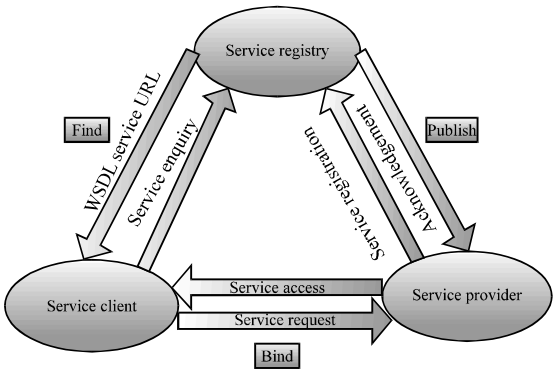

Fig. 1: SOA architecture

Corresponding Author: K. Jayashree, Rajalakshmi Engineering College, Chennai, India

of the service and the operations (or methods) the service exposes. A consumer uses find operation to look up the registry to find an appropriate web service from the service registry. The service registry returns the Uniform Resource Locator (URL) for the requested service. The user service then binds itself to the service provider for service execution. The service input parameters are sent to the service provider who executes the service and returns the results to the consumer. These interactions happen through the use of SOAP messages. SOAP is a simple XML-based protocol used for accessing a web service and for message exchanges (Yu *et al*., 2008). SOA architecture enables services to be dynamically selected and integrated at runtime thus enabling system flexibility and adaptability.

Researchers now discuss research related to web service fault taxonomy and ontology for web service. While there has been considerable research relating to web service testing, debugging and fault tolerance, there has been less research in the area of fault management in web services and developing ontology for managing faults in web services.

Delgado *et al*. (2004) have presented taxonomy for runtime software fault-monitoring for traditional software. Monitoring of web services require additional issues because it is loosely coupled and dynamic in nature. These additional issues and semantics can be addressed only by developing ontology specifically for web services.

Bruning *et al*. (2007) have proposed fault taxonomy for possible faults in Service Oriented Architecture (SOA). Faults discussed include hardware fault, software fault, network fault and operator fault. The interactions for SOA architecture have been analyzed and the different possible faults have been defined and presented in the fault taxonomy. They have demonstrated the application of the taxonomy with an example of travel agency. The taxonomy represents only the class/subclass relationship of faults and does not represent the relationship between the different kinds of faults.

Chan *et al*. (2009) proposed a novel taxonomy that captures the possible failures that occur in web service composition and classify the faults that caused the failures. The taxonomy covers physical, development and interaction faults. An important use of the taxonomy is in identifying the faults that can be excluded when a failure occurs. Mahdian *et al*. (2009) present an approach to detect faults in the architecture level of service oriented systems which extends the formal SOA core meta model to support fault tolerance. Monitor components present in the architecture are used to detect three types of faults namely timeout, no service found and binding denied.

Ardagna *et al*. (2006) have classified the faults as infrastructure and middleware level faults, web service level faults and web application level faults. Infrastructure and middleware level fault types are node faults, network faults and generic faults. Web service level fault types are web service execution faults and web service coordination faults. Web application fault types are internal data faults, application coordination faults, actor faults and Quality of Service (QoS) violation faults. They have classified the faults and provided a set of strategy. Semantic approaches have not used by Chan *et al*. (2009), Kim *et al*. (2007) and Ardagna *et al*. (2006).

Zhou *et al*. (2004) proposed ontology for description of quality of service metrics for web services based on DAML. The ontology contains three definition layers: the QoS profile layer, the QoS property definition layer and QoS metrics layer. The ontology is designed to help users to compare web services offerings on basis of QoS metrics. Kritikos and Plexousakis (2006) and Jayashree and Anand (2012) developed QoS ontology to describe the QoS non-functional aspect of web services. QoS is used to distinguish web services based on their performance and for refining web service advertisements.

Qiu and Xiong (2007) have proposed an ontology for semantic web services to enrich web services description. The proposed construction of service ontology is used to identify and maintain the relationships among services. It is also helpful for requesters to find their suitable services according to their own preferences.

Hai and Li (2012) proposed a QoS ontology and BP neutral network module to choose the qualified web service from candidates. All these developed ontologies are based on QoS of web services. Kim *et al*. (2007) have presented the framework for Mid-level Ontologies for Quality (MOQ) representing general quality concepts that can be used for web services, among other applications. Mohammad *et al*. (2011) have presented an Ontology Based Access Control (OBAC) to support semantic web service. In this study, security ontologies have been developed to specify concepts and terms involved. Faycal *et al*. (2011) have proposed a development-oriented process for building web service ontology using the Ontology Web Language for Services (OWL-S) language. The process was divided into four phases such as specification of requirements, conceptualization, implementation and mapping, evaluation and maintenance. With the earlier proposed process, they have effectively build a web services ontology in different domain and this ontology allows software agents to discover, compose and invoke automatically a web services without the intervention of human beings.

In this study, researchers present a generic ontology for describing run-time faults in web services. Ontology provides a flexible and easy approach for expressing the common vocabulary and semantic description for faults that occur at run time.

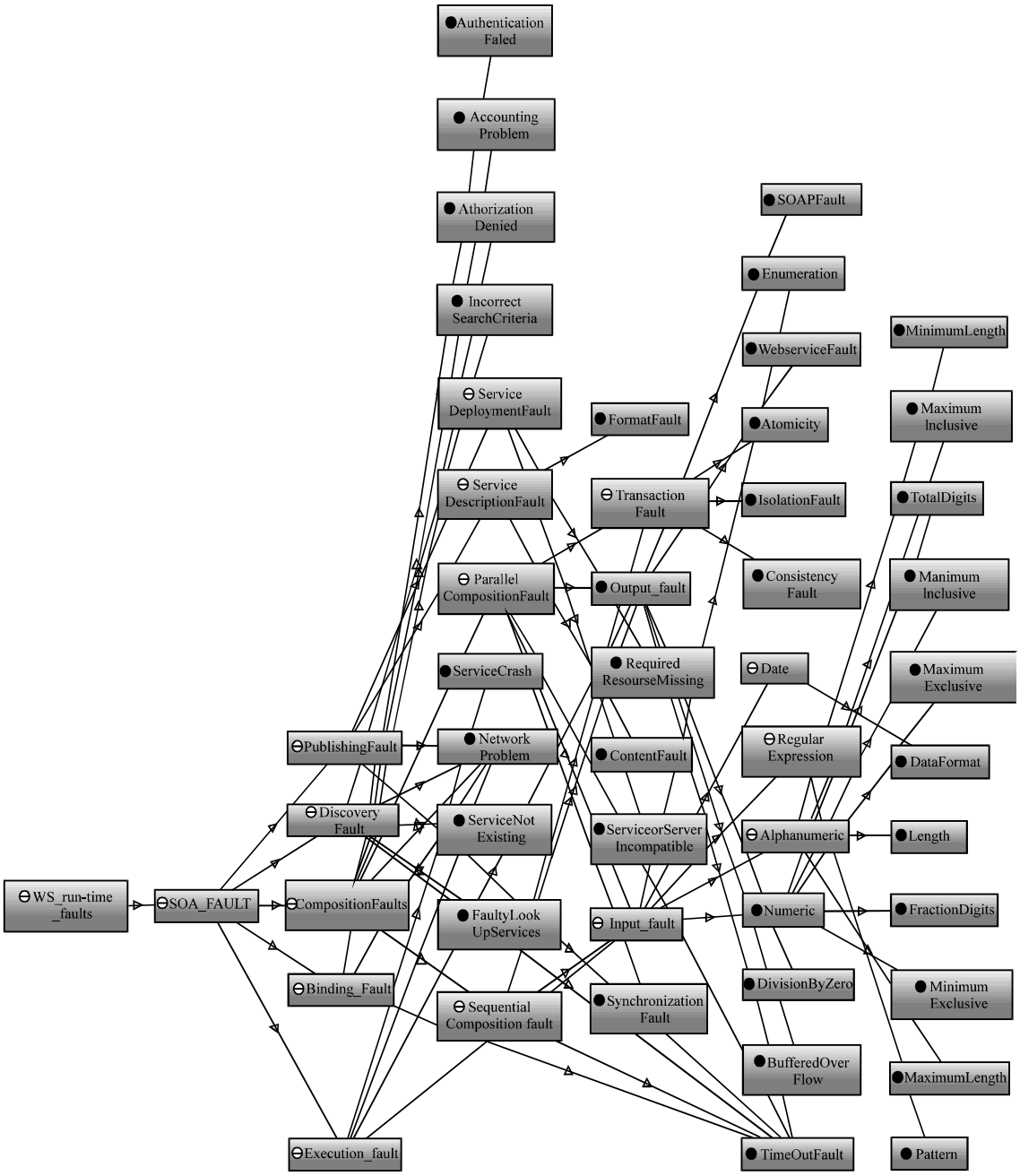

Fig. 2: Fault ontology for web services

PROPOSED FAULT ONTOLOGY FOR WEB SERVICES

Failures occur during web service execution due to various causes such as network faults, server crashes or application-related errors. Faults include the unavailability of a requested web service or errors in the orchestration of choreography of web services. Specific web service faults may be due to missing data or parameters in an execution flow or low Quality of Service (QoS) (Boumhamdi and Jarir, 2010). These faults need to be detected and handled appropriately to avoid system failure and provide the users with meaningful error messages.

Runtime monitoring of web services enables run-time faults to be detected. In the earlier research, researchers had proposed the use of policies and runtime monitoring to detect faults during execution of web services

(Jayashree and Anand, 2012). In this study, researchers present a fault ontology that can be used to provide semantic error information to the user. This would enable the user to correct the error and use the web service in its intended manner.

Ontology is the formal and explicit specifications of a knowledge domain and includes critical concepts, entities and objects of the domain and their inter-relationships. Ontology can be used to show the different types of relationship between terms. In this study, researchers present ontology developed using the OWL, a semantic markup language that is used to represent ontologies (Taye, 2011). OWL has been developed as a vocabulary extension of RDF (the Resource Description Framework) to represent the meaning of terms and the relationship between the terms (Brickley and Guha, 2000). Using OWL, information is given explicit meaning so that it makes it easy to develop applications to automatically process and integrate the information on the web. The proposed fault ontology is given in Fig. 2. The fault ontology includes various run-time faults that occur during publishing, discovery, binding, composition and execution of web services.

The fault ontology comprises of four main components: concepts, instances, relations and axioms. A concept is also known as class and it is collection of objects. It is the base element of the fault domain. A class is called as super class when it represents the parent class and when it represents the child class of the parent class it is called as a sub class. In the ontology SOA fault is represented as super class. The subclasses for the SOA fault are publishing faults, discovery faults, composition faults, binding faults and execution faults. The inheritance relationship that exists between the classes is shown in the ontology.

Instance is the ground level component of an ontology which are the things represented by a concept. It is also known as an individual. For example service description fault is an instance of publishing or SOA fault. If a class publishing fault is a subclass of SOA fault then every instances of publishing fault is also an instance of the SOA fault.

A relation is used to express relationships between two concepts in given domain. More specifically, it describes the relationship between the first concept, represented in the domain and the second, represented in the range. For example, date format could be represented as a relationship between the domain date and with the range YYYYMMDD.

An axiom is used to impose constraints on the values of classes or instances. For example min length class represents constraints for the input parameter password of a web services. For example it should have a minimum length of 8 and maximum length of 20. This axiom is represented as policy as:

```
<xs:element name ="password">
  <xs:simpleType>
    <xs:restriction base ="xs:string">
      <xs:minLength value ="8"/>
      <xs:maxLength value ="20"/>
    /xs:restriction>
  </xs:simpleType>
</xs:element>
```

A CASE STUDY

In this study, researchers illustrate the use of the fault ontology with a sample case study. Runtime monitoring enables faults to be detected and fault ontology can be used to trace and report the source of the error to the users. The proposed ontology has been created using the limited free version of altova semantic works which supports OWL DL (Description Logic). A sample web service application from the railway services domain has been taken as an example. Simple and composite sample web services, namely, train enquiry, ticket availability, ticket reservation, booking details, ticket payment, reservation status enquiry and ticket cancellation have been created. The web services and their descriptions are given in Table 1.

Publishing error: Researchers take an example from the publishing faults scenario to depict publishing error in the train enquiry service. The publishing faults depicted in the ontology include service description, service deployment fault, network problem and timed out errors. service description faults can occur due to format fault and content fault.

Web service registry has been created using jUDDI registry. To publish a service in the service registry, the provider has to give details like publisher name, service name, business name, WSDL description, etc. The fault scenario depicted in Fig. 3 shows a fault that has occurred while trying to publish the service with the name @ PNR service. Fault has occurred due to the presence of @ in the service name as special characters are not allowed in the service name. Invalid service name causes content fault, a sub class of service description fault which is a sub class of publishing fault which in turn is a sub class of SOA fault. This information is obtained as shown in Fig. 3 by backtracking through the fault ontology.

If the user knows the semantic reason for the fault, then the user would be able to make the necessary corrections and get the service successfully published.

Table 1: List of web services for railway ticketing application

Web services	Description
Train enquiry	Allows the users to determine the list of trains to the intended destination
Ticket availability	Allows users to find out whether seats or berths are available in the train of their choice. They can also obtain information about the number of seats available in the required class
Ticket reservation	Reservation service is a composite web service, comprising of the ticket payment and booking details services. The user is returned a PNR number that is used to uniquely identify the reserved tickets
Booking details	The user first provides details for booking the tickets which includes date of travel, train number, train name, departing station, destination station, class, name, age, payment details, etc
Ticket payment	Payment service is used to process the payment amount. The user is required to give the credit card details for processing the payment through an authorized payment gateway
Reservation status enquiry	Requires the passenger to key in the PNR number and the reservation status is displayed
Ticket cancellation	Allows the users to cancel their seats of their choice by giving the PNR number

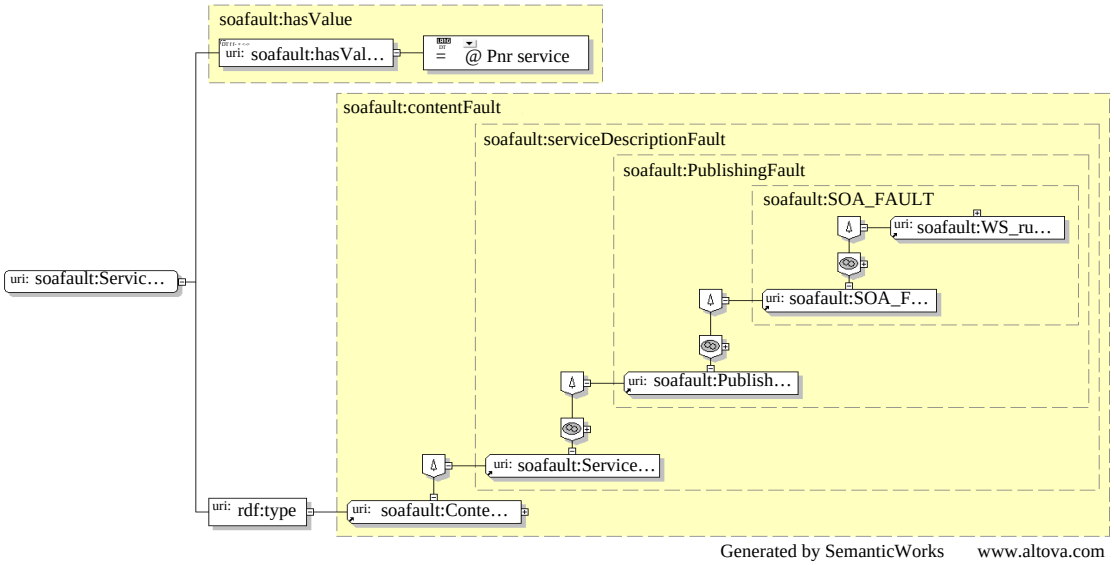

Fig. 3: Fault occurrence in publishing of a web service

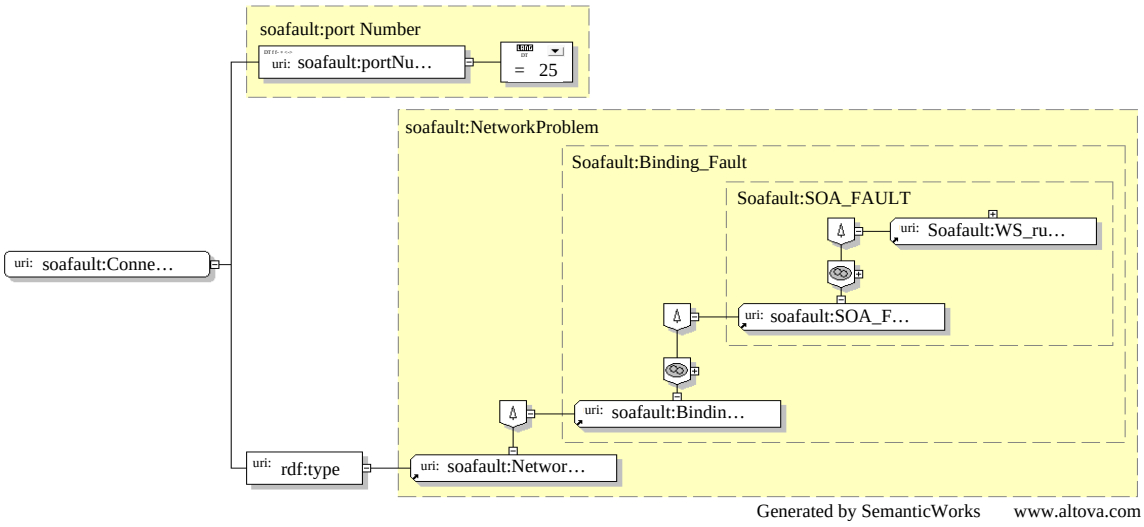

Fig. 4: Fault occurrence in binding of train enquiry service

Using the ontology, it is possible to report to the user that the fault in publishing has occurred due to content fault in the service name that is invalid service name.

Binding error: For invoking the service from the service provider the service client has to bind with the service provider. Binding faults occur when the service client is unable to access the specific service of the service

provider. The binding faults depicted in the ontology are authorization denied, authentication failed, accounting problem, network problem and timed out fault.

To bind to a particular service, the user has to specify the port number, target namespace, discovery URL, operation name, port type and service name. The fault occurrence while trying to bind the train enquiry service is given in Fig. 4. As seen from the Fig. 4, fault occurs in the port number which has been specified as 25. Port 25 is the default port number of Simple Mail Transfer Protocol (SMTP) service and cannot be used by any other service. The port number given is invalid for the train enquiry service and hence the binding is not successful. Invalid port number causes port number fault, a sub class of network problem which is a sub class of binding fault which in turn is a sub class of SOA fault.

Discovery error: For discovering a particular service from the registry, the user searches the web service directory with appropriate search criteria to locate the required service. The service URL address is returned to the user. Discovery faults include faults in the given search criteria and also in returning the found service. The ontology includes the various faults like incorrect search criteria, network problem, service not existing, faulty lookup service and timed out fault.

To discover the ticket reservation service, the service client has to provide the details like business name and service name, etc. The fault occurrence when the service name given is incorrect is shown in Fig. 5. The service name has been given as train reservation instead of ticket reservation. Using the fault ontology, it is possible to identify that the error in the given search criteria has occurred due to invalid service name.

Execution error: There could be many types of execution errors. The errors could be in the input parameters specified to invoke the service, execution errors or in the results (output) returned by the service.

For example, let us consider that data given as input parameter to the web service has to be numeric. Further, constraints can be specified in numeric data which includes total digit, minimum inclusive, minimum exclusive, maximum inclusive, maximum exclusive and/or fraction digits. Researchers take an example of PNR number given as an input parameter to invoke the reservation status enquiry service to explain execution error. In the sample case study, PNR number is a numeric with a length of 10 digits, minimum inclusive value of 4000000001 and maximum inclusive value of 4999999999. The constraints in the PNR number given as XML tags are:

```
<input>
  <element name ="pnrno">
     <simpleType>
       <restriction base ="long">
           <TotalDigits value ="10"/>
           <MinimumInclusive value ="4000000001"/>
           <MaximumInclusive value ="4999999999"/>
           <NonNegative>
     <Pattern value ="[4][0-9][0-9][0-9][0-9] [0-9][0-9] [0-9][0-9] [0-9] "/>
           </NonNegative>
       </restriction>
     </simpleType>
  </element>
</input>
```

Errors in specifying the PNR number could be in the total digits or in not giving the data as per the constraints specified. Some of the common possible errors are listed in Table 2. The fault occurrence when PNR number is

Table 2: Some errors in PNR number

PNR number	Error
411100111	The length is not 10 digits
3000003555	The number is not valid because it is not in the minimum inclusive value range
6012345678	The number is not valid because it is not in the maximum inclusive value range
A123456789	The number is not numeric as it contains the alphabet A
10/12/2012	The number is not numeric
-568459345	The number is not valid because it is negative number

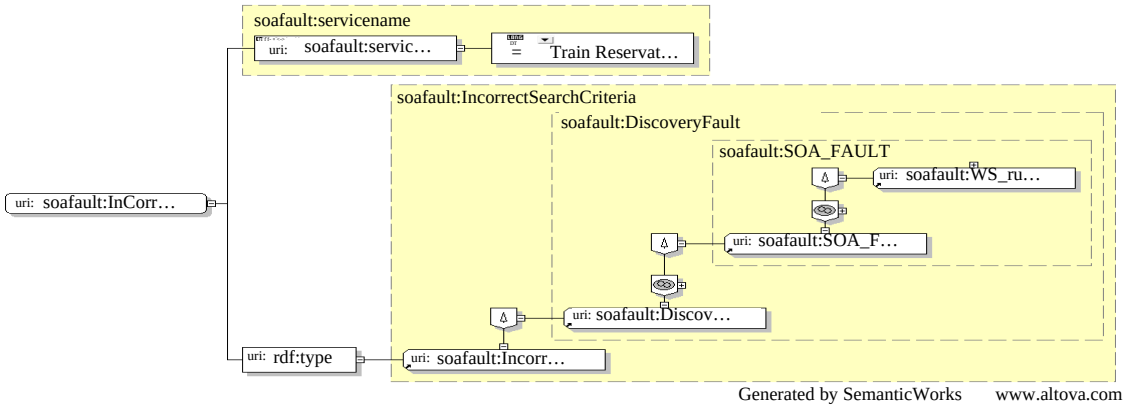

Fig. 5: Fault occurrence in discovery of ticket reservation service

given as 411100111 is given in Fig. 6. The error occurs because PNR number specified is incorrect as it is not 10 digits in length.

Likewise, errors could occur in the result returned by the service. For example, the PNR number given may satisfy the input parameter criteria as specified. During execution, the PNR number is matched with the PNR entries in the database to obtain the status. If the given PNR number is not available in the database then the error would be returned as database fault as shown in Fig. 7.

Composition service error: Composition faults occur when there is an error in the execution of any of the services that comprise the composite service. Researchers take an example of the ticket reservation service which is a composite service comprising of the two services ticket booking and ticket payment. Ticket booking service is executed first to obtain the ticket and train details and block the tickets for the specified train. The ticket payment is automatically invoked to process the payment for the tickets. Once the payment process is successfully completed, ticket booking details are updated in the database and tickets are printed. To the user however it would appear as one service only. Error can occur in execution of either of the two services. Two sample scenarios are considered.

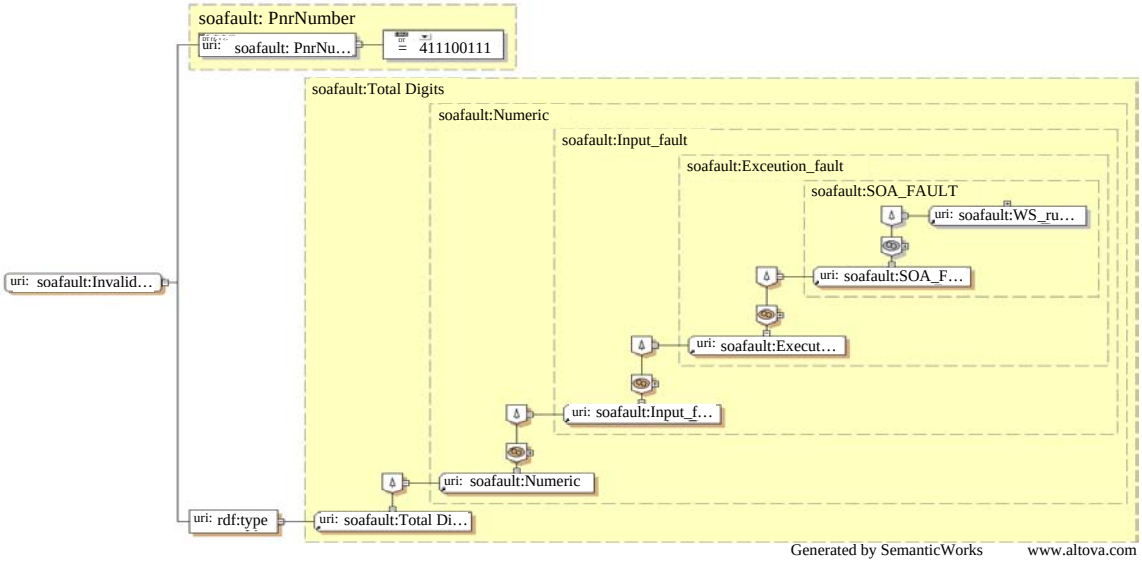

Fig. 6: Fault occurrence in reservation status enquiry for invalid PNR number

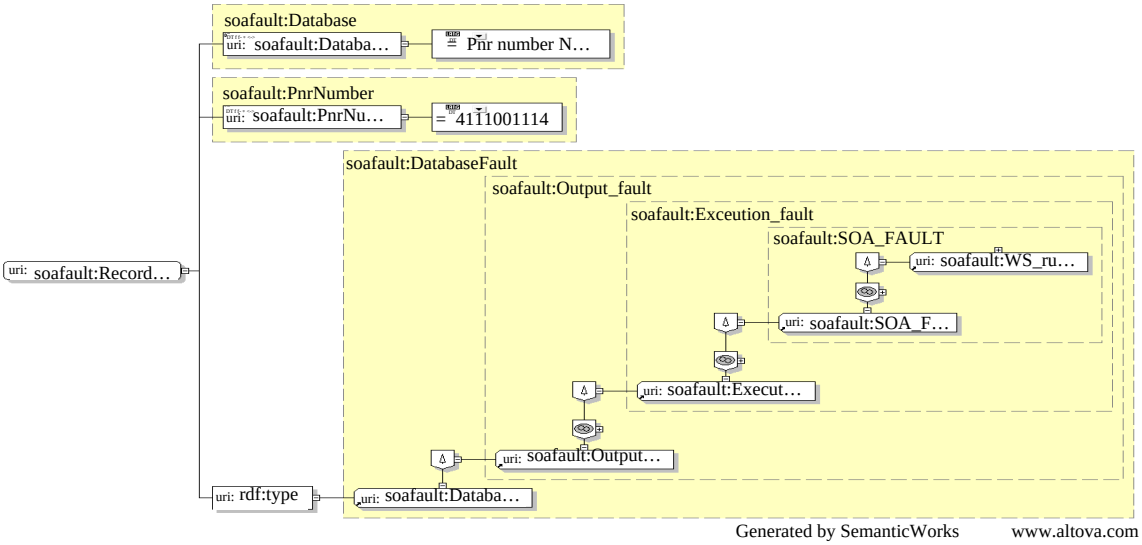

Fig. 7: Fault occurrence in reservation status enquiry for PNR number not listed in the database

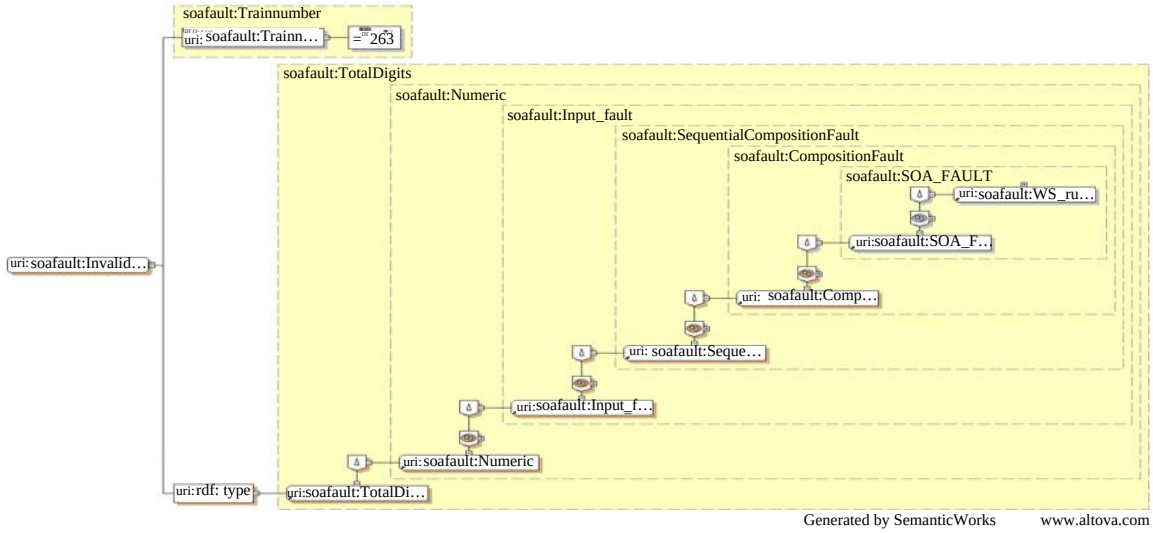

Fig. 8: Fault occurrence in input of booking details service

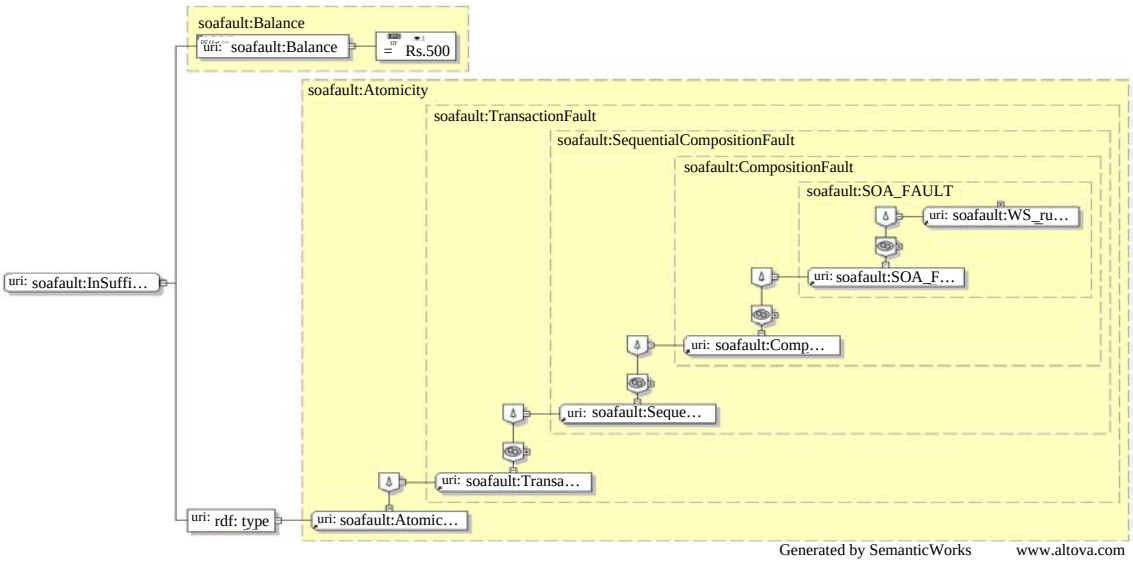

Fig. 9: Fault occurrence in output of ticket payment service

Case 1 (The booking details service is not successful): The booking details service would not be successful if the user gives incorrect travel/ticket details and/or if there is an execution error in the service. The details provided by the user includes date of travel, train number, train name, departing station, destination station, class, name, age and payment details. For example, if the user gives a train number which is invalid then the booking service will fail as shown in Fig. 8. In the given example, train number given as input to the service does not satisfy the input data length constraint. As the first service in the composite service is not successful, the second service payment service will not be called.

Case 2 (The ticket payment service is not successful): The booking details service which is the first service in the composite service is successful. So, the payment service is invoked with payment details from the booking service. The payment service could fail if the connection to the payment gateway is not successful. The payment service could fail if the credit card given is invalid or does not have sufficient balance. Using an instance of the ontology, it would be possible to convey the appropriate error message to the user.

Transactions should satisfy the ACID (Atomicity, Consistency, Isolation and Durability) properties. In an atomic transaction, a series of database

update operations should either be all successful or nothing should occur. This prevents partial updates to the database. In the given example, payment service failure has been represented as a fault in transaction atomicity as shown in Fig. 9. For the ticket reservation service the database is updated with:

- Train seat booking
- Payment details
- Ticket details

The payment of Rs.500/- is not successful as there is insufficient balance in the user's account. Hence, the payment update does not go through and it is represented as a transaction fault. Train seat booking and ticket details accepted in the booking details service would not be also updated in the database, thereby ensuring transaction atomicity.

CONCLUSION

Fault detection is the first step in managing faults in web services. With the commercialization of web service usage, it becomes necessary to detect faults and convey to the user why and where the fault has occurred. In this study, researchers have presented fault ontology for run-time faults in web services. Fault ontology can be used to give meaningful error messages to the user for fault occurrence during web service publishing, discovery or execution. Fault ontology defines a common vocabulary for information sharing and plays an important role in semantic web services that allow such knowledge to be transmitted and shared using application systems. The ontology can also be extended to represent semantic descriptions of other run-time faults and faults that may be specific to particular web services.

REFERENCES

Alam, S., 2009. Fault management of web services. M.Sc. Thesis, University of Saskatchewan, Canada.

Ardagna, D., C. Cappiello, M. Fugini, E. Mussi, B. Pernici and P. Plebani, 2006. Faults and recovery actions for self healing web services. Proceedings of the World Wide Web Conference, May 22-26, 2006, Edinburgh, UK.

Boumhamdi, K. and Z. Jarir, 2010. A flexible approach to compose web services in dynamic environment. Int. J. Digital Society, Vol. 1.

Brickley, D. and R.V. Guha, 2000. Resource Description Framework (RDF) schema specification 1.0. W3C Candidate Recommendation 27. http://www.w3.org/TR/2000/CR-rdf-schema-20000327/.

Bruning, S., S. Weibleder and M. Malek, 2007. A fault taxonomy for service-oriented architecture. Proceedings of the 10th IEEE High Assurance Systems Engineering Symposium, November 14-16, 2007, IEEE Computer Society, Washington, DC., USA., pp: 367-368.

Chan, K.S., J. Bishop, J. Steyn, L. Baresi and S. Guinea, 2009. Fault taxonomy for web service composition. Lecture Notes Comput. Sci., 4907: 363-375.

Charfi, A., T. Dinkelakery and M. Meziniy, 2009. A plug-in architecture for self- adaptive web service compositions. Proceedings of the IEEE International Conference on Web Services, July 6-10, 2009, IEEE Computer Society, Washington, DC., USA., pp: 35-42.

Curbera, F., M. Duftler, R. Khalaf, W. Nagy, N. Mukhi and S. Weerawarana, 2002. Unraveling the web services web: An introduction to SOAP, WSDL and UDDI. IEEE Internet Comput., 6: 86-93.

Delgado, N., A.Q. Gates and S. Roach, 2004. A taxonomy and catalog of runtime software fault monitoring tools. IEEE Trans. Soft. Eng., 30: 859-872.

Erl, T., 2005. Service-Oriented Architecture: Concepts, Technology and Design. Prentice Hall, Upper Saddle River.

Faycal, Z., K. Lazhar and Z. Mohamed, 2011. A development-oriented process for building web services ontology using OWL-S Language: Application in medical web services. Int. J. Comput. Appli., Vol. 34.

Fensel, D., F. van Harmelen, I. Horrocks, D.L. McGuinness and P.F. Patel-Schenider, 2001. OIL: An ontology infrastructure for the semantic web. IEEE Intell. Syst., 16: 38-45.

Hai, Y. and X. Li, 2012. A web service QoS predicting algorithm based on QoS ontology and BP neutral networks. Int. J. Adv. Comput. Technol., 4: 199-199.

Jayashree, K. and S. Anand, 2012. Policy based distributed run time fault diagnoser model for web services. Lecture Notes Institute Comput. Sci. Social Informat. Telecommun. Eng., 86: 9-16.

Kim, H.M., A. Sengupta and J. Evermann, 2007. MOQ: Web services ontologies for QoS and general quality evaluations. Int. J. Metadata Semantics Ontol., 2: 195-200.

Kritikos, K. and D. Plexousakis, 2006. Semantic QoS metric matching. Proceedings of the 4th European Conference on Web Services, December 4-6, 2006, Zurich, Switzerland, pp: 265-274.

Mahdian, F., V. Rafe, R. Rafeh and M.R. Zand, 2009. Considering faults in service -oriented architecture: A graph transformation based approach. Proceedings of the ICCTD'09 International Conference on Computer Technology and Development, November 13-15, 2009, Kota Kinabalu, Malaysia.

McIlraith, S. and D. Martin, 2003. Bringing semantics to web services. IEEE Intell. Syst., 18: 90-93.

Mohammad, A., G. Kanaan, T. Khdour and S. Bani-Ahmad, 2011. Ontology-based access control model for semantic web services. J. Inf. Comput. Sci., 6: 177-194.

Qiu, Q. and Q. Xiong, 2007. An ontology for semantic web services. High Perform. Comput. Commun., 4782: 776-784.

Taye, M.M., 2010. Understanding semantic web and ontologies: Theory and applications. J. Comput., 2: 2151-9617.

Taye, M.M., 2011. Web-based ontology languages and its based description logics. Res. Bull. Jordan ACM, 2: 2078-7952.

W3C, 2004. Web services architecture W3C working group note. http://www.w3.org/TR/ws-arch/#stakeholder_using.

Yu, Q., X. Liu, A. Bouguettaya and B. Medjahed, 2008. Deploying and managing Web services: Issues, solutions and directions. VLDB J., 17: 537-572.

Zhou, C., L.T. Chia and B.S. Lee, 2004. DAML-QoS ontology for web services. Proceeding of the IEEE International Conference on Web Services, July 6-9, San Diego, California, USA., pp: 472-479.