Research Inventy: International Journal Of Engineering And Science
Vol.3, Issue 3 (June 2013), PP 50-54
Issn(e): 2278-4721, Issn(p):2319-6483, Www.Researchinventy.Com

Run-Time Verification of Behavioral Conformance for Conversational Web services

[1,]M.Radha & [2,]R.Saranya
Raja College of Engineering and Technology, Madurai, Tamilnadu, India

***ABSTRACT**: Transactional patterns are used to specify flexible and reliable composite Web services. A transactional pattern is a convergence concept between workflow patterns and advanced transactional models. These are of coordination patterns and as a structured transaction. This project proposes an event-driven approach to validate the transactional behavior of service compositions. The transactional behavior verification is done either at design time to validate recovery mechanisms consistency, or after runtime to report execution deviations and repair design errors, and therefore, formally ensure service execution reliability. By using the trust based rule ontology to specify and check the transactional behavior consistency of service composition, our approach provides a logical foundation to ensure service execution reliability.*

I. INTRODUCTION

A key challenge of (Web) service compositions is how to ensure reliable execution.Service-Oriented Architecture (SOA) and Business Process Management (BPM) will continue to evolve dependently. SOA is gaining prominence as a key architecture to support BPM and integrate applications in diverse and heterogeneous distributed environments. Without loss of generality, we consider services offering only one operation. Indeed, a composite service, defined using BPEL, for instance, is, in fact, a flow of operations and not a flow of services.In order to avoid such confusion, we supposed for simplicity purpose that a service has one operation so that its consumption coincides with its operation invocation.It is widely recognized that one of the barriers preventing widespread adoption of this technology is a lack of products that support nonfunctional features of applications, such as execution reliability. Different from business process components, services are generally provided by different organizations independently from any execution context. Since each organization has its own rules, services treated like strictly autonomous entities. Due to the inherent autonomy and heterogeneity of Web services, the guarantee of correct CS executions remains a fundamental problem issue. An execution is correct if it reaches its objectives or fails according to the designers requirements or users needs.

II. TRANSACTIONAL COMPOSITE SERVICE

The term composite service (CS) is usually used to denote composition of operations offered by different services.Without loss of generality, consider services offering only one operation. Acomposite service is a conglomeration of existing Web services working in tandem to offer a new value-added service. It orchestrates a set of services as a composite service to achieve a common goal. A transactional composite (Web) service (TCS) is a composite service composed of transactional services. Such a service takes advantage of the transactional properties of component services to specify failure handling and recovery mechanisms. Concretely, a TCS implies several transactional services and describes the order of their invocation, and the conditions under which these services are invoked. A TCS defines a set of predicates on each component service's external transition in order to define the orchestration. These predicates specify for each component service when it will be activated, canceled, or compensated. More formally, we define a TCS as the set of its composing services and the set of predicates defined on their external transition.

III. EVENT CALCULUS

A formalism based on the Event Calculus (EC) for specifying composite service failure handling policies. This formal specification is used to formally validate the consistency of the transactional behavior of the composite service model at design time, and to analyze and check composite service execution reliability at runtime according to users' needs. The transitional behavior of the service refers to two recovery action once it fails. First, in contrast to pure state transition representations, the EC ontology includes an explicit time structure.This helps managing event-based systems where a number of input events may occur simultaneously.Second, the trust based rule ontology is close enough to popular standards like WSBPEL and provides an automatic support into the logical representation.

III. RUN-TIME VERIFICATION

Runtime Checking Our work attempts to apply Web service log-based analysis and process model checking techniques to provide knowledge about discrepancies between process models and related instances using Runtime or a posteriori verification. More precisely, given an event log, have to verify TCS's transactional properties after runtime, to provide knowledge about the context of and the reasons of discrepancies between process models and related instances.

This kind of verification is necessary since some interactions between Web services that constitute a process may be dynamically specified at runtime, causing unpredictable interactions with other services, and making the a priori verification method insufficient as it only takes into account static aspects. For this step, CS execution should be auditable by providing functionalities to collect execution logs. Runtime checking techniques is based on a delta analysis between the initially designed TCS and TCS log.

IV. METHODOLOGY

a) Monitoring

Monitoring the "effective" transactional behavior allows us to detect design gaps and improve the application reliability. Some deviations from the expected behavior may be highly desirable to detect process evolution and execution anomalies showing initially hardly foreseeable process parameters, constraints, and needs.

i) Collecting CSlog

There already exist many investigations and proposals on Web server log and associated analysis techniques. However, the arising paradigm of Web services requires richer information in order to fully capture business interactions and customer electronic behavior in this new Web environment. Since the Web server log derives from pages requests accessed by a user, it is not tailored to capture service utilization and composition. In order to track customer electronic behavior, it is necessary to consider both traditional Web server log data and data originated from Web services usage in an integrated way. Web server log data provide information about user navigation and his interactions with Web sites, while Web services log data provide information about the use of Web services, supporting information concerning business processing as well as quality of services.

b)Trust Based Rule

Trust is central to all transactions, where own actions are dependent on the actions of others. Thus, excluding instances where trust in someone has no in influence on decisions. Trust can be strong or weak depending on the environment.

Morton Deutsch defines trust as follows.

• If an individual is confronted with an ambiguous path, a path that can lead to an event perceived to be beneficial (Va+) or to an event perceived to be harmful (Va-);

• He perceives that the occurrence of Va+ or Va- is contingent on the behavior of another person; and

• He perceives that strength of Va- to be greater than that strength of Va+. In this definition, Deutsch describes an event with a beneficial outcome as Va+, while a harmful result is entitled as Va-. His view of a trust relationship is such that any event is linked with other events by either conducive or detrimental paths. To reach another event, one of the available paths has to be taken. This definition is based on psychology but outlines one of the most important requirements to establish trust. If the perceived benefit were greater than the perceived harmfulness then the significance of trust in the choice would not be that big. In other words, a trust relation requires a harmful path with more significance than the beneficial one. An employee for example, would not choose to hack a database and switch identity with another employee if the others salary was less than his own.

A more concrete and mathematical definition is given by Diego Gambetta Trust or symmetrically, distrust is a particular level of the subjective probability with which an agent assesses that another agent or group of agents will perform a particular action, both before he can monitor such action or independently or his capacity ever be able to monitor it and in a context in which it acts his own action. When we say we trust someone or that someone is trustworthy, we implicitly mean that the probability that he will perform an action that is beneficial or at least not detrimental to us is high enough fur us to consider engaging in some form of cooperation with him. Both definitions combined will give the requirements that have to be met when building an environment based on trust. Gambetta measures trust with a range from 0 to 1, where 1 represents complete or blind trust and 0 is complete distrust.

i)TRANSACTIONAL SERVICE PATTERNS

An AND-split pattern defines a point in the process where a single thread of control splits into multiple threads of control which can be executed in parallel, thus allowing services to be executed simultaneously or in any order.

An AND-join pattern defines a point in the process where multiple parallel subprocesses/services converge into one single thread of control, thus synchronizing multiple threads.An XOR-split pattern defines a point in the process where, based on a decision or control data, one of several branches is chosen .

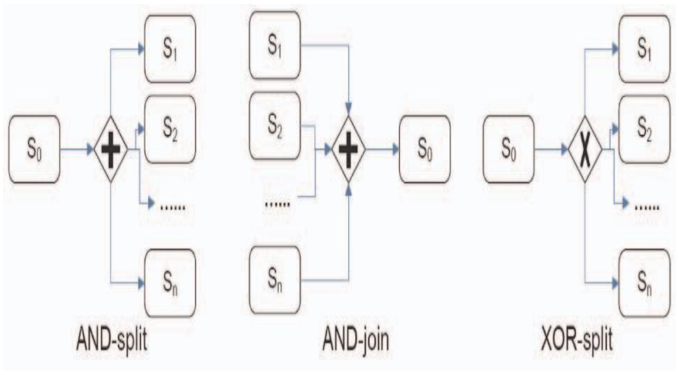

FIG 1.1 WORK FLOW PATTERNS

Thus, the transactional flow should respect some consistency rules given a pattern. The transactional consistency rules ensure transactional consistency according to the context of the used pattern. In the following, formally specify these patterns and related transactional consistency rules using EC. These rules are inspired from which specifies and proves the potential transactional dependencies of workflow patterns.

VI. MODULES

This project consists of following modules**.**

- Accessing the User Interface
- Web Event – Monitoring
- Web Trust based rule
- Trust based rules
- Expected Result

a) Accessing the User Interface

Accessing the web to enter the services in the web server and get the user requirement.Choose the web service application and give the full detail for the authenticated user.

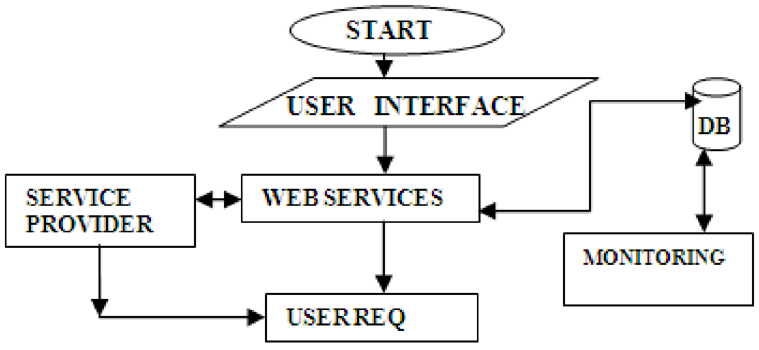

Fig 1.2 Accessing User Interface

These details are stored in the web server database. Confirm the chosen application for further process.

b) Web Event – Monitoring

To monitor the user requested event and store the user requirement into the web service database and get the requested application from the service provider and send to the authorized requester.The above service got by the requester will be monitored run time and the relevant service details are stored in the web service database.

c) Web Trust based rule

Heterogeneity- verifying how the event is running. This event is called "AND SPLIT" again The methodology for the transaction of web services between the requester and provider are called "AND JOIN". In this session the co-ordinator monitor the service provider are performed in the requested application

d) Trust based rules

Rules are the one which specifies which participant has to commit, cancel or are getting compensation.

A .Event is completed without compensation
B .Event is processed getting compensation and then completed
C .Event is processed getting compensation and failed.

Requested Process:

1) Choosing the event
2) Select the application
3) Run the application

Trust based rule is also monitored.

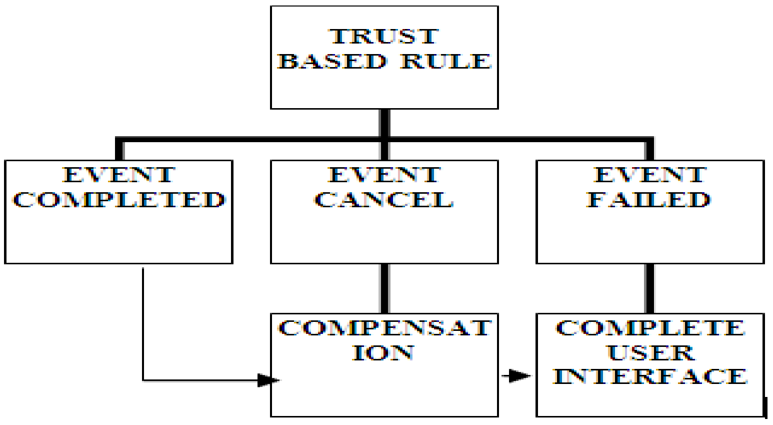

Fig1.3 TRUST BASED RULE

e) Expected Result

The user will get the expected output from the web service provider and thus the final result has achieved.

To adapt/combine workflow mining techniques to the Web-services-related fields. we applied mining techniques to discover and improve composite Web service transactional behavior.

VII. CONCLUSION

Generally, formal previous approaches develop, using their modeling formalisms, a set of techniques to analyze the composition model and check related properties. Bultan et al. propose a formal framework for modeling, specifying, and analyzing the global behavior of Web services compositions.

This approach models Web services by mealy machines (finite-state machines with input and output). Based on this formal framework, the authors illustrate then expected nature of the interplay between local and global composite Web services. the authors propose a Petrinet- based algebra for composing Web services. This formal model enables the verification of properties and the detection of inconsistencies both between and within services.

REFERENCES

[1] S. Bhiri, O. Perrin, and C. Godart, “Extending Workflow Patterns with Transactional Dependencies to Define Reliable Composite Web Services,” Proc. Advanced Int’l Conf. Telecomm. and Int’l Conf. Internet and Web Applications and Services (AICT/ICIW), p.145, 2006.

[2] J.Eder and W.Liebhart, “The work flow activity model Wamo,”Proc.Int’l Conf.Cooperative Information Systems,pp.87-98,1995.

[3] M.Kamath and K.Ramamritham, “Failure Handling and Coordinated Execution of Concurrent Workflows,” Proc. Int’l Conf. Data Eng. (ICDE ’98), pp. 334-341, 1998.

[4] R. Kowalski and M.J. Sergot, “A Logic-Based Calculus of Events,”New Generation Computing, vol. 4, no. 1, pp. 67-95, 1986.

[5] B. Medjahed,B. Benatallah, A. Bouguettaya, A.H.H. Ngu, and A.K. Elmagarmid, “Business-to-Business Interactions: Issues and Enabling Technologies,” The VLDB J. vol. 12, no. 1, pp. 59-85, 2003.

[6] S.Mehrotra, R. Rastogi, H.F. Korth, and A. Silberschatz, “A Transaction Model f o r Multidatabase Systems,” Proc. Int’l Conf. Distributed Computing Systems (ICDCS ’92), pp. 56-63, 1992.