International Journal on Future Revolution in Computer Science & Communication Engineering
Volume: 4 Issue: 1
ISSN: 2454-4248
237 - 240

Framework for Automatic Checkpoint Generation

A Vani Vathsala

Dept of CSE, CVR College of Engineering, Hyderabad, India

e-mail: atlurivv@yahoo.com

Tel.: +919866586106

Abstract Web services provide services to their consumers in accordance with terms and conditions laid down in a document called as Service Level Agreement (SLA). Web services have to abide by these terms and conditions failing which, SLA faults result. Fault handling of web services is a key mechanism using which SLA faults can be avoided. We propose fault handling of choreographed web services using **checkpointing and recovery.** We propose checkpointing in three stages: design, deployment and **dynamic** checkpointing. In this paper we propose a framework for generation of checkpoint locations automatically in a given choreography document by applying design time checkpointing rules. We have also developed a tool to demonstrate that the proposed framework is indeed implementable.

Keywords- web service, checkpointing, framework, automatic

*****

I. INTRODUCTION

A web service is a piece of software that provides a service and is accessible over Internet. A web service that implements a business process, without invoking another web service, is called as atomic web service. Otherwise it is called as a composite web service.

A set of services working in tandem with each other to achieve a business goal are called as choreographed web services. They interact with each other according to a predefined sequence specified in a design document called as choreography document.

As composite web services operate over Internet, providing fault handling to composite web services is of primary importance. We have proposed a design time checkpointing policy[1] that introduces checkpoints in a choreography. In the event of transient failures this checkpoint arrangement avoids re-invocation of web services that perform non repeatable actions.

In this paper we propose a framework to generate checkpoint locations in a given a choreography document (specified in xmi format), by applying design time checkpoint rules discussed in [1]. We would like to extend the framework to incorporate deployment time and dynamic checkpoint rules also for generating checkpoint locations, in our future work.

This paper is organized as follows: Section II presents the proposed framework for automatic checkpoint generation, section III presents design time checkpoint policy in brief. Section IV describes the developed tool for automatic checkpoint generation. Section V presents conclusion and future work.

II. FRAMEWORK FOR AUTOMATIC CHECKPOINT GENERATION

In this section a framework is presented that supports automatic generation of checkpoints at design time in a given choreography.

It mainly consists of two modules: 1. **Design time Checkpointing Module (DSM) 2. Recovery module**.

A. Design time Checkpointing Module (DSM)

DSM in turn consists of two sub modules: Pattern identification module and Checkpointing module. Figure 1 presents the proposed framework.

Pattern identification module:

It takes as input, a choreography depicted in the form of a UML activity diagram and identifies atomic patterns present in the given choreography, using the interaction patterns proposed in [1] . These patterns are stored in the database. It then models the choreography as a composition of the identified patterns and expresses the given choreography as a pattern string [1]. It outputs this pattern string *PS* and the set of atomic patterns[1] *SP* which constitute the pattern string.

Checkpointing module

It takes the pattern string and the set of patterns generated by pattern identification module, as input and then applies the checkpointing and logging rules to insert checkpointing locations in the choreography. It implements design time checkpointing algorithm presented in[1].

Output of the checkpointing Module is the UML activity diagram which is given as input but has checkpointing locations appended to it. This output diagram helps the participating web services to insert checkpoints at appropriate locations in their code.

 ISSN: 2454-4248
237 - 240

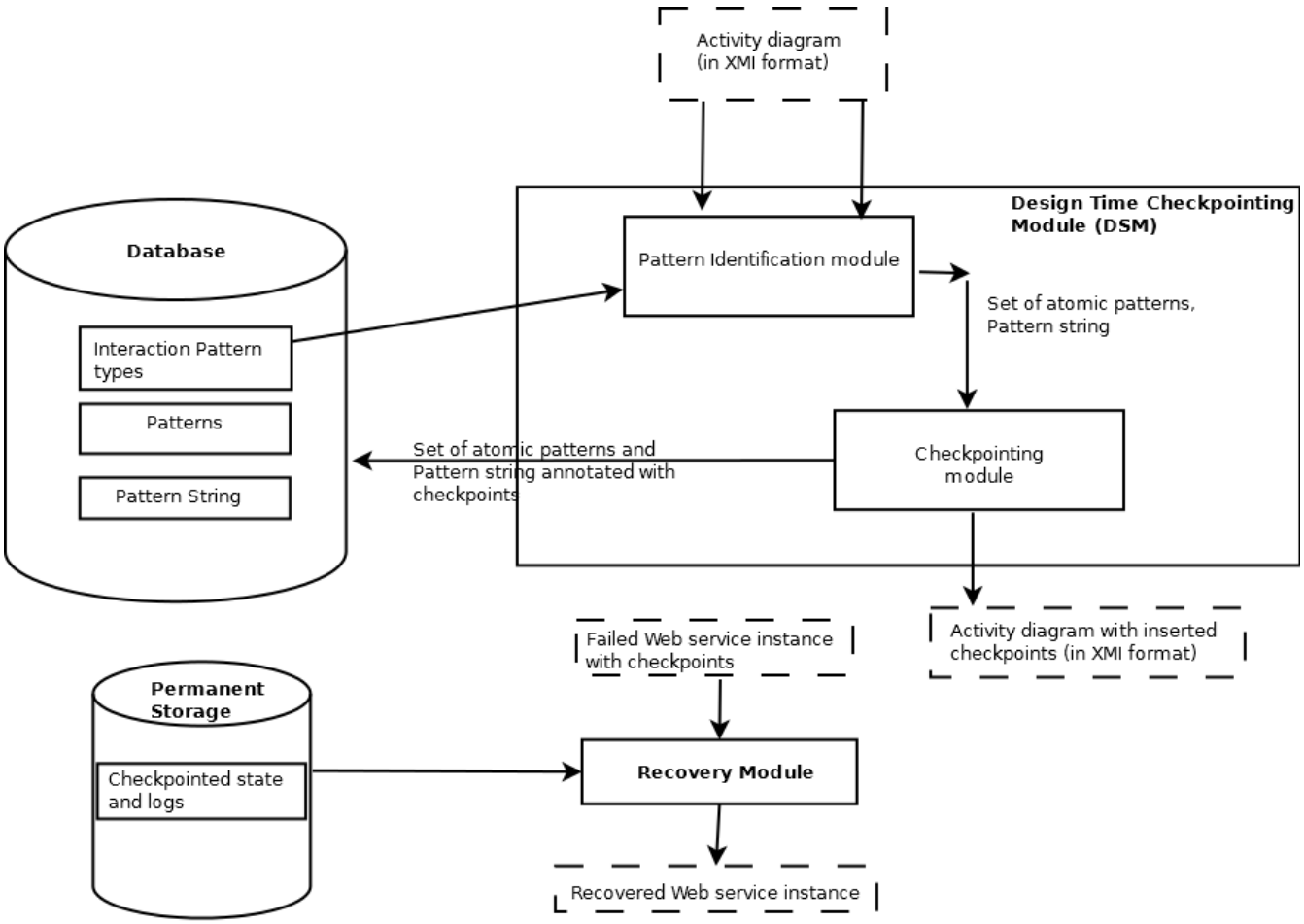

Figure 1: Framework for automatic checkpoint generation

The module also outputs the pattern string appended with design time checkpoints. This pattern string is used as input to checkpointing algorithms executed at later stages, i.e. at deployment stage and execution stage.

Figure 2 depicts the working of DSM.

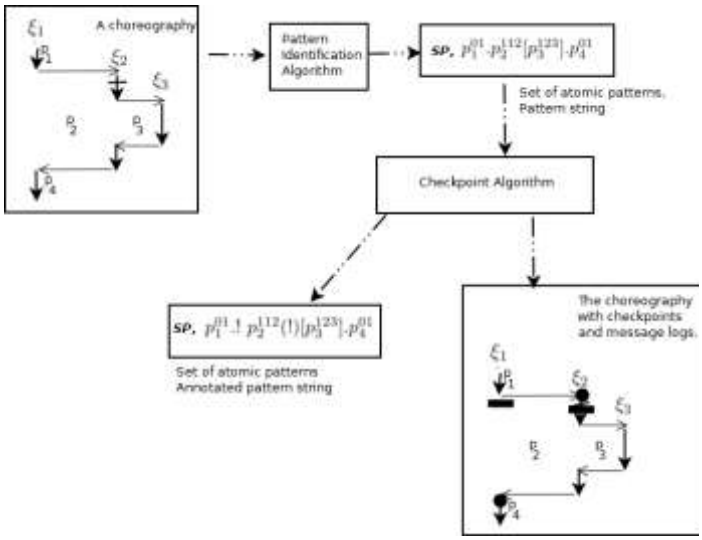

Figure 2: Working of DSM

B. Recovery module

Recovery module takes as input failed web service instance id. It restores the failed web service instance to the latest checkpointed state and replays logged messages as indicated by recovery rules[1]. It may be noted here that the proposed recovery rules do not require recovery of other web services, other than the failed instance.

III. CHECKPOINTING POLICY

Design time checkpointing policy identifies checkpointing locations in patterns.
Checkpointing policy is based on the following two principles:

1. A participant executing a non repeatable action must be checkpointed immediately after performing the non repeatable action.
2. If the receiver in a pattern executes a non repeatable action, the initiator of the pattern should not initiate the interaction again in case of its failure.

IV. AUTOMATIC CHECKPOINT GENERATION TOOL

In this subsection an implementation of the proposed framework is presented. We have developed a tool called ACGT (Automatic Checkpoint Generation Tool) that inserts checkpoints into a given choreography.

 ISSN: 2454-4248
237 - 240

ACGT is implemented in Java under Eclipse IDE. The following four step procedure is followed:

As a first step of implementation the choreography of web services is generated using UML Activity diagrams. Input to ACGT is a UML diagram that is generated using Visual Paradigm for UML 10.2 Enterprise Edition. Any other UML tool such as Rational Rose, Umbrella etc may also be used.

In the second step, the diagram is exported in XMI (XML Metadata Interchange) format. XMI is an OMG standard for exchanging metadata information which uses XML tags.

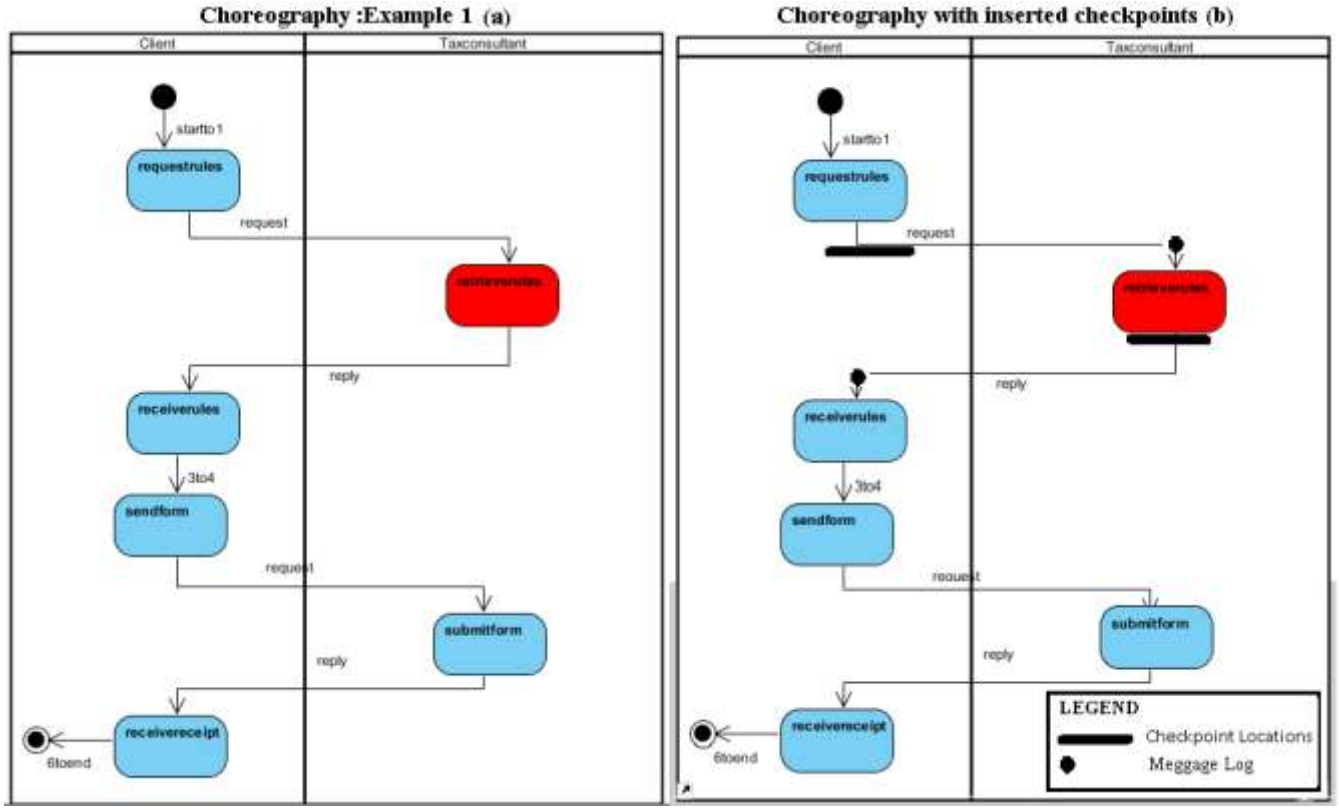

Figure 3: Example Choreography

In the third step the XMI file generated in step 2 is given as input to ACGT. In order to extract patterns from XMI file, DOM (Document Object Model) and SAX(Simple API for XML) APIs are used. ACGT generates graphical elements representing checkpointing locations in the given UML activity diagram. The output again is in XMI format.

As a fourth step the XMI file generated in step three is imported in Visual Paradigm to view the diagram along with appended checkpoints.

Figure 3 (a) depicts an example choreography which is given as input to ACGT and Figure 3 (b) depicts the same choreography with checkpointing locations inserted by ACGT. In this example a choreography that consists of non repeatable actions (red coloured actions) is depicted. Here, ACGT generates checkpoints based on checkpointing rules[1] *CR1* and *CR2* and message logging rules *LR1* and *LR2* resulting in checkpointing pattern CP2.

V. CONCLUSION AND FUTURE SCOPE

In this paper we have presented a framework for generating checkpoint locations in a choreography document given as input. To start with, we have considered only design time checkpointing rules for automatic checkpoint generation. We have even developed a tool that uses the proposed framework for inserting checkpoint locations into a given choreography document.

As part of our future work we would like to extend the framework to incorporate deployment time and dynamic checkpointing rules in taking checkpointing decisions.

VI. REFERENCES

[1] Vani Vathsala A, HrushikeshaMohanty, "*Interaction patterns based checkpointing of choreographed web services".* In the Proceedings of the 6th International Workshop on Principles of Engineering Service Oriented and Cloud Systems (PESOS 2014),Hyderabad, India, pp 28–37,**2014.**

[2] Vani Vathsala A, HrushikeshaMohanty, *"Time and cost aware checkpointing of choreographed web services",* In the Proceedings of the 11th International Conference on Distributed Computing and Information Technology (ICDCIT 2015), India, pp 207–219,**2015.**

[3] Chtepen M, Dhoedt B, De Turck F, Demeester P, Claeys F, Adaptive checkpointing in dynamic grids for uncertain job durations. In the Proceedings of the 31st International Conference on Information Technology Interfaces, (ICITI) pp 585–590, **2009.**

[4] Nazir, Qureshi, and Manuel. Nazir B, Qureshi K, Manuel P, " *Adaptive checkpointing strategy to tolerate faults in economy based grid*". The Journal of Supercomputing, Vol.50, Issue 1, pp 1–18, **2009**

ISSN: 2454-4248
237 - 240

[5] HE Mansour, TDillon, "*Dependability and rollback recovery for composite web services*". IEEE Transactions on Services Computing Vol 4, Issue 4,pp 328–339,**2011.**

[6] Chandy KM, Lamport L, "*Distributed snapshots: determining global states of distributed systems*". ACM Transactions on Computer Systems Vol 3, Issue 1, pp 63–75, 1985.

[7] Vani Vathsala A, HrushikeshaMohanty, "*A Survey on checkpointing web services".* In the Proceedings of the 6th International Workshop on Principles of Engineering Service Oriented and Cloud Systems (PESOS 2014),Hyderabad, India, pp 11–17,**2014.**