A DISTRIBUTED SELF-HEALING ARCHITECTURE SUPPORTING WS-BASED APPLICATIONS

Francisco Moo-Mena, Fernando Curi-Quintal, Juan Garcilazo-Ortiz
Luis Basto-Díaz and Roberto Koh-Dzul
Universidad Autónoma de Yucatán - Facultad de Matemáticas, Periférico Norte km 33.3, Merida, Mexico

Keywords: Self-healing, Web service, Digital library.

Abstract: A Self-healing infrastructure allows to observe the behavior of a system, determining its health status, and applying measures to restore the correct state of the application. In recent years our work has focused on the design and implementation of Self-healing architectures, which support applications based on Web Services (WS). This paper presents an improved Self-healing architecture, which proposes a distributed control in its components. The results obtained by applying this new Self-healing architecture to a distributed Digital Library application show a trend towards a better availability of resources.

1 INTRODUCTION

The Web Service (WS) technology has made possible to develop complex distributed systems.

Distributed systems require mechanisms to prevent loss of component functionality and QoS degradation. Fault-tolerance techniques provide a mechanism to recover from specific failure situations, where a fault can make the system unavailable suddenly for indeterminate periods, minimizing the outages. However, systems in a high demand environment also need to predict problems and take actions to prevent the failure and its impact on the application or, to recover without any apparent overall disruption.

In 2001, IBM presented an initiative called "Autonomic Computing", adopting principles of biological systems (Kephart and M.Chess, 2003). The initiative presents four perspectives: Self-configuring, Self-healing, Self-optimizing and Self-protecting (Halima et al., 2006).

This work is focused on improving a distributed WS-based Self-healing architecture to support a Digital Library application, with contents organized and distributed by knowledge area, by applying Self-healing techniques and a better distribution of components.

Interceptors are used for Monitoring, QoS analysis for Diagnosis, and redundancy of components for Recovery (Ghosh et al., 2007). This work is based on previous work which defined the Self-healing architecture, the statistical model and the ontological model applied for the diagnosis (Moo-Mena et al., 2008) (Moo-Mena et al., 2009).

The rest of the paper is organized as follows: section 2 describes our distributed Self-healing architecture; section 3 shows experimental results; and section 4 discusses conclusions and future work.

2 SELF-HEALING ARCHITECTURE

In a previous work, a first Self-healing architecture was developed. One limitation of this architecture is its centralized approach that only considers a simple interaction scenario. It considers one WS component making requests to other WS component, the parameters collected during the interaction are analyzed by a "Self-healing Core", which is a critical central point because their failure could compromise the operation of the system.

Restrictions in the previous architecture are analyzed and a better schema is proposed to get a distributed Self-healing architecture based on cooperative WS.

2.1 Self-healing Components Distribution

The WS distributed operation is structured as nodes that create interaction channels to transfer requests

Moo-Mena F., Curi-Quintal F., Garcilazo-Ortiz J., Basto-Díaz L. and Koh-Dzul R..
A DISTRIBUTED SELF-HEALING ARCHITECTURE SUPPORTING WS-BASED APPLICATIONS.
DOI: 10.5220/0003498015701160
In *Proceedings of the 7th International Conference on Web Information Systems and Technologies* (WEBIST-2011), pages 157-160
ISBN: 978-989-8425-51-5

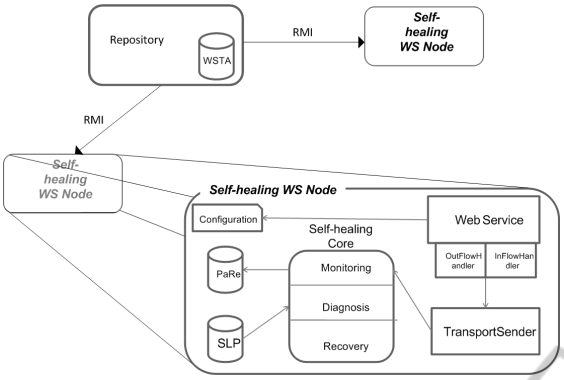

Figure 1: Distributed Self-healing Architecture.

and replies. Two roles are defined for WS, the consumer (WS requester) and the provider (WS provider).

The most important goal is to have an efficient communication between elements. The Self-healing principles implemented in this architecture ensure successful operations.

Self-healing components has been defined in the architecture, where the Self-healing functionality was implemented. These components are bonded to each WS and start its operation when the WS consumer invokes a function of WS provider.

The WS consumer is responsible for monitoring and evaluating the performance of their providers, and if it detects degradations it must choose another provider.

The architecture components are (See Figure 1):

- **Repository Module**, this component registers all WS in the WSTA (Web Service Table Access), which represents the composition of the architecture at any given time.
- **WS component**, this is a WS with defined functionality. Within the Digital Library application, each WS component is responsible for managing content of a knowledge area.

In the improved architecture, each WS component has a Self-healing Core composed by: Monitoring Module, Diagnosis Module and Recovery Module.

These modules are described below in the order they appear during a transaction between a consumer and its provider.

2.2 Repository Module

This component is responsible for maintaining the information of all active WS in the architecture in a table called WSTA (Web Service Table Access).

When a WS joins to the architecture, the first action that it performs is to register his information in

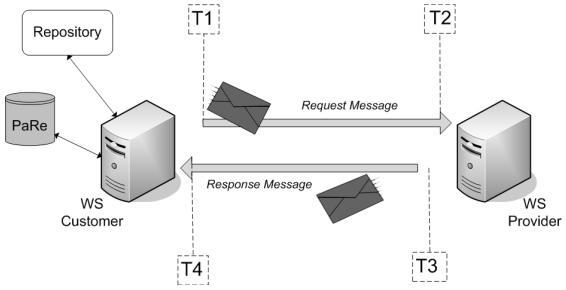

Figure 2: Web Services Interaction Scenario.

the WSTA and calculate a time delay between it and the Repository Module. To calculate the difference of time, we apply an algorithm based on Cristian's work (Cristian, 1989)

When a WS needs to work with another, it starts a transaction, asks to the repository for a provider which has a particular keyword; the Repository search in the WSTA and returns the address of an active WS provider to interact.

If there are only inactive providers, the Repository selects and actives one. Now the WS consumer can contact the provider to make the requests that are required.

2.3 Monitoring Process

The monitoring stage involves interception of messages between the consumer and the provider. The messages are intercepted from the WS consumer using Handlers, which is a tool provided by Apache Axis2 WS engine.

Each time that a message is sent or received in a WS, the Monitoring Module records timestamps and calculates the QoS time parameters, which are used for the diagnosis phase. The set of timestamps are four:

- T1: When a WS consumer sends a Request Message.
- T2: When the WS provider receives the Request Message.
- T3: When the WS provider sends a Response Message.
- T4: When the WS consumer receives the Response Message.

When the transaction commits, the timestamps are used to calculate the QoS time parameters:

- QoS1: T3 - T2: Computation time
- QoS2: T2 - T1: Requesting time.
- QoS3: T4 - T3: Responding time.

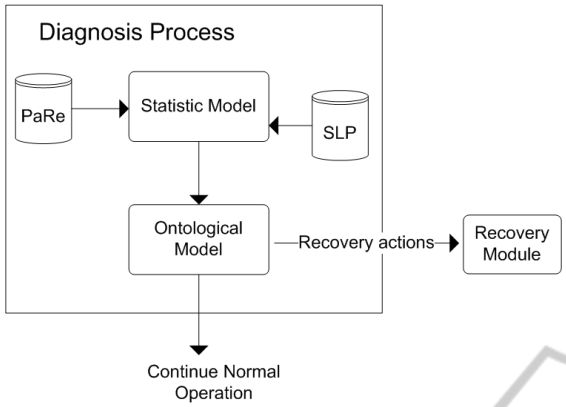

Figure 3: Diagnosis Process.

- QoS4: (T4 - T1) - QoS1: Communication time.

The value of the parameters are stored in the PaRe (Parameter Repository) database for further analysis. An interaction scenario is showed in Figure 2.

If the provider does not receive a response message after sending a request, the Monitoring Module determines an unsuccessful transaction and indicates that its provider is not available, therefore a replacement for the provider is required and the former provider is set offline as recovery strategy.

After a certain number of transactions, the Diagnosis Module is executed.

2.4 Diagnosis Process

The diagnosis is based on the analysis of QoS parameters using a statistical (Moo-Mena et al., 2009) and ontological model. The statistical model is based on the box-plot method, which is a technique used to analyze a data set (Mendenhall et al., 1998). The results of the statistical module are the inputs for the ontological method. The ontological model establishes a knowledge base that represents the current QoS level of communication with a WS provider.

For the QoS parameters analysis, the most recent set of values for each QoS parameter is taken from PaRe. The statistical model calculates how many recorded values are outliers, according to the Right Outer Fence (ROF) measures established by each WS provider in the Service Level Parameter (SLP) table.

When the diagnosis process is executing for the first time, the SLP table contains reference measures (ROF) for all providers.

The results of the statistical model are passed to the ontological model at runtime to determine if any recovery action is required (See Figure 3).

The ontological model defines an ontology where relations are established between the main QoS properties (W3C, 2009) and the health of a component. In this work, Integrity and Availability characteristics are defined and related to the State of performance entity to determine the Degradation level.

The reference measures were defined, applying the box-plot method, with a sample set of parameters obtained during the execution of the architecture at an early stage. After testing, *Normal Performance (OK)* was less than 5% of outliers, with a *Moderated Degradation* the results are between 5% and 10%, and a higher percentage indicates a *Severe Degradation*.

The application of rules over the knowledge base allows to infer the performance level of a provider, comparing the current QoS measures against reference measures.

For example, the rule that diagnoses a Severe Degradation checks that the percentage of outliers values for QoS parameters are greater than the limit values defined to activate a substitution action. In this case, the ontological model determines a Severe Degradation.

If the percent of outlier values is between 3% and 5%, means that the time intervals of interaction are increasing. Then, the ROF will be redefined using a recent data set for a each QoS parameter. The new values are stored in the SLP table for further use. After the adaptation, the diagnosis process is executed again to improve the diagosis result.

If the result is *Moderate degradation*, the Diagnosis Module requests to the Recovery Module a duplication of its WS provider to improve the performance level.

If the result is *Severe degradation*, the Diagnosis Process requests to the Recovery Module a substitution of its WS provider to interact with another WS provider.

2.5 Recovery Strategies

The diagnosis techniques applied in the Self-healing architecture are based on redundancy. Each WS provider should have a duplicate which offers the same functions. The WS are stateless.

Recovery actions are applied at the diagnosis module's request and include duplicating or replacing the current WS provider.

Duplicating a WS provider has the consequence that the consumer requests are divided between more than one provider, randomly selecting a provider for each transaction.

The replacement, however, is oriented to the requests that are sent to a new provider, which has not been diagnosed as corrupted.

The changes applied by the recovery module are

in the WSTA, changing the information corresponding to the current WS provider. Then, the changes made in WSTA table are visible to all WS in the architecture.

3 RESULTS

The implementation of a testing application that applies the principles of Self-healing perspective focuses on creating a distributed Digital Library.

In the Digital Library application each component has WS Self-healing components to evaluate the interaction with other WS components. The Repository records the components of the Digital Library under the name of the area to which they belong as "keyword" to the WSTA table.

The result of integrating the architecture for this type of application was helpful. The fact of having a mechanism to ensure suppliers always they are required for each knowledge areas, guarantees that the requests have a greater probability to be attended. For demonstration purposes a test was performed with a high number of requests.

Figure 4 shows a graph of QoS values, in which the number of outliers values is reduced, as a result of both, the diagnosis process and the recovery action of duplicating a WS provider.

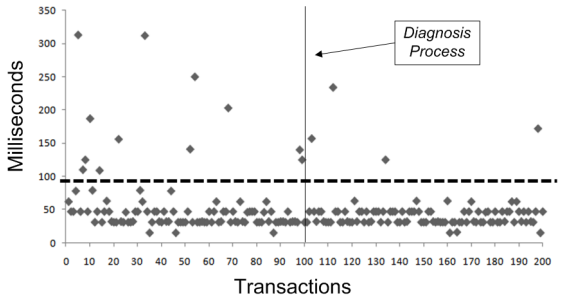

Figure 4: Graph of QoS values.

4 CONCLUSIONS

This distributed architecture opens the possibility to create new structures of interaction between WS components, where the main goal is to ensure the availability of the services.

The components distribution reduces the critical points of failure and takes advantage of the dynamic operation of each component.

Diagnosis module's adaptation strategies allow the architecture to work under variable conditions, determining a good performance.

The ontological model along with the rules engine added to the diagnosis module have the ability to make inferences of current performance and determine the recovery actions considering the current state representation in a knowledge base.

ACKNOWLEDGEMENTS

This project was developed under financial support from PROMEP and UADY.

REFERENCES

Cristian, F. (1989). A probabilistic approach to distributed clock synchronization. In *9th International Conference on Distributed Computing Systems*, pages 146–158.

Ghosh, D., Sharman, R., Rao, H., and Upadhyaya, S. (2007). Self-healing systems - survey and synthesis. *Decision Support Systems*, pages 2164–2185.

Halima, R., K.Drira, and M.Jmaiel (2006). A comparative study of self-healing architectures in distributed systems. Technical Report 06568, LAAS.

Kephart, J. O. and M.Chess, D. (2003). The vision of autonomic computing. *Computer Magazine, IEEE*, pages 41–50.

Mendenhall, W., Beaver, R. J., and Beaver, R. M. (1998). *Introduction to probability and statistics*. Brooks/Cole, USA.

Moo-Mena, F., Garcilazo-Ortiz, J., Basto-Diaz, L., Curi-Quintal, F., and Alonzo-Canul, F. (2008). Defining a selfhealing qos based infrastructure for web services applications. In *IEEE 11th International Conference on Computational Science and Engineering (CSE 2008)*, Sao Paulo, Brazil.

Moo-Mena, F., Garcilazo-Ortiz, J., Basto-Diaz, L., Curi-Quintal, F., and Alonzo-Canul, F. (2009). A diagnosis module based on statistic and qos techniques for self-healing architectures supporting ws based applications. In *IEEE International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC 2009)*, Zhangjiajie, China.

W3C (2009). Qos for web services: Requirements and possible approaches. http://www.w3c.or.kr/kr-office/TR/2003/ws-qos/.