Enhancing the Diagnosis Module in a Self-healing Architecture Supporting Web Service 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-Tablaje 13615, Mérida, Yucatán, Mexico
{mmena,cquintal,gortiz,luis.basto}@uady.mx, jose.koh@fmat.uady.mx

Abstract. A Self-healing infrastructure allows to observe the behavior of a system, determine its health status, and apply 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). A previously developed architecture adopted a centralized approach regarding to the topology and control components. As a diagnosis technique used a statistical method based on box-plot diagrams. In this paper we present the modifications made to the diagnosis module of that Self-healing architecture. The proposed change gets an architecture with distributed control components. The diagnosis also adds the use of ontologies and inference rules that contribute to improving awareness about system health. The results obtained by applying new Self-healing architecture to a distributed digital library application show a trend towards more precise diagnosis.

1 Introduction

The development of technology in recent decades has had a major impact on human lifestyle, creating a dependence on systems that handle storage, processing and management of meaningful information in a diversity of areas, like academic, industrial and services.

To meet the current needs, systems are more complex and their own maintenance and repair mechanisms are very hard to implement. It is a requirement their permanent availability, reliability and safety in heterogeneous contexts, which are susceptible to failures or malicious attacks.

The fault tolerance strategies and problem solving by managers are no longer sufficient in highly dynamic environments. New strategies are needed to have more available and efficient services [1].

Claiming to offer a solution to that kind of problems, IBM presented, in 2001, the Autonomic Computing initiative [2], inspired by the autonomous function of central nervous system of animals, adopting four perspectives:

Advances in Computer Science and Engineering.
Research Computing Science 45, 2010, pp. 337-348

Received 30/04/10
Final version 19/05/10
Accepted 14/05/10

1. *Self-configuring.* Systems adapting in response to their environment changes, based on high-level policies.
2. *Self-healing.* Ability to detect, diagnose and recover from errors.
3. *Self-optimizing.* Improves system performance by optimizing its operation and resource usage.
4. *Self-protecting.* Defense strategies are proposed to solve problems caused by attacks or failures that cannot be repaired by Self-healing.

Implementing these four perspectives results in systems that automate the maintenance, repair, optimization and protection tasks, requiring minimal human or other systems intervention.

An autonomous system is composed of autonomous elements that contain resources and offer services. The elements handle system's internal behavior and relationships with other elements according to established policies.

The Self-healing and Self-protecting approaches are interesting because they are focused in maintaining the permanent availability, accessibility and integrity of the system, by preventing and repairing faults.

However, from another point of view, Self-healing perspective are more important because a system may have the ability to auto-configure, self-optimize and self-protection, but if it cannot recover from failures is likely to stop working.

Some Self-healing approaches propose a process that helps to detect and recover system's faults. Their stages are [3], [4]:

1. *Monitoring.* Consists in the monitoring and recording of information about the system health status.
2. *Diagnosis.* Analyzes and evaluates the information gathered in the monitoring stage and determines whether the performance level is correct or not.
3. *Recovery.* Strategies are implemented to help correct problems found and recover the proper system performance.

This paper adopts an approach focused on the architecture of the application, applying interceptor techniques for monitoring, Quality of Service (QoS) analysis for the diagnosis and redundancy of components for recovery.

The aim of our project is to obtain an architecture based on WS, which allows the creation of a distributed system where the interaction between WS has Self-healing capabilities, ensuring the quality of communication among them.

We use a distributed digital library application in order to test our proposal. In this application each Web Service hosts information corresponding to an area of knowledge library [5], pretending create a distribution of content in different WS, ensuring each has access to the contents of the other. The reason for the distribution of content is to avoid a central point of failure, and improve their performance using the Self-healing scheme.

This work is based on the architecture and Self-healing techniques presented in [5], [6] and [7] in order to make a new proposal by adding a distribution perspective, and a better diagnosis process.

The rest of the paper is organized as follows: in Section 2 an overview of related work regarding Self-healing systems is presented. Section 3 introduces

the Self- healing architecture previously developed. Section 4 describes in general terms the new Self-healing architecture with emphasis on changes carried out to the diagnosis module. Section 5 shows experimental results. And Section 6 describes conclusion and ongoing work.

2 Related Work

In recent years, there have been different perspectives on Autonomic Computing concepts, leading to propose new initiatives like Self-protecting, Self-knowledge, Self-diagnosis, Self-destruction and Self-adjustement, according to emerging needs, fitting the concept of autonomy [3].

For the implementation of Self-healing systems the main problems encountered are designing the mechanisms for fault detection and diagnosis, and recovery strategies. Solutions have been proposed for specific cases that can be applied to other approaches. Hardware-based autonomous designs find their counterpart in software systems. Some projects related to Autonomic Computing that propose Self-healing approaches are:

2.1 Bio-Net

Led by the National Science Foundation, Bio-Networking Architecture [8] is a paradigm and a middleware for the design and implementation of scalable, adaptable and available network applications. It is based on principles and mechanisms used by biological systems to adapt to changing environmental conditions, like colonies of ants and bees. Abstracts the biological model of cooperation and autonomy in a system composed of autonomous agents, cyber-entities (mobile autonomous agents) and Bio-Networking Architecture platforms (execution environments and support services for the cyber-entities). The autonomous interaction and cooperation between the components results in a stable and adaptable survival system.

2.2 HYDRA

Hydra [9] is a middleware belonging to Hydra EU project, which allows developers to incorporate heterogeneous devices offering WS interfaces for administration.

Hydra incorporates mechanisms for Service Discovery, Semantic Model Driven Architecture, P2P communication and Diagnosis.

Hydra adopts the "awareness context" concept and presents an OWL ontology and SWRL rules based on the Self-management approach, particularly in Self-diagnosis. In Hydra, each component has an awareness of their own state, knowing and defining the optimal values for parameters benchmarked to run smoothly.

SWLR rules and ontologies provide a Knowledge Base where information about the current state of the middleware is represented. If a fault occurs the

possible solutions are within the same ontology OWL. The inference about the information stored during the monitoring helps the diagnosis process to make intelligent decisions regarding the recovery Self-healing tasks.

2.3 CODA

CODA (Complex Organic Distributed Architecture) [10] represents a new generation of decision-making system that includes a means monitoring and controlling objectives to allow the enterprise to evolve with certain degree of autonomy.

The architecture includes concepts and principles of Self-organization, Self-regulation toward an intelligent architecture. The main challenge is to achieve a distributed object-oriented reference architecture with support for reconfigurable mobile networks.

2.4 Discussion

Our work has focused on creating platforms and application models using the Self-healing perspective.

Previous works have presented concepts such as adaptation of the application's components to new environments, the"context awareness" for diagnosis (using ontological models), distribution of components into an intelligent architecture. Other works have developed concepts that focus on developing applications based on WS, applying Self-healing principles, for example, WS-DIAMOND [11].

Our goal is to develop a new perspective for the creation of WS-based applications, in which the cooperative interaction is guaranteed by applying Self-healing principles.

We have developed an ontological-statistical model for performance analysis, approaching to the "context awareness" based on QoS, which considers the reconfiguration of the techniques used for fault detection in dynamic environments.

Our case study consists in to implement a distributed WS-based digital library application in which each component is responsible for storage and information management of an area of knowledge.

3 Previous Self-healing Architecture

Nowadays, systems implementation with Self-healing properties has reached a considerable importance. Among the diverse approaches covered, the main problem worked is the components complexity. It is no longer enough to have fault tolerance mechanisms, additional strategies are needed to ensure an effective recovery. This paper presents improvements made to an architecture defined in [5].

The previous architecture is centralized and consists in a Web Service Consumer, a Web Service Provider and a Self-healing Core, where databases and Monitoring, Diagnosing and Recovering modules are located.

The architecture was implemented with the Java Web Service technology. Apache Axis2 is the Web Service engine, running the consumer and the provider. The Self-healing Core is a component whose interface with WS is through Java RMI.

3.1 Monitoring Module

Every transaction between WS consumer and WS provider is monitored using Apache Axis2 Handlers, making the role of interceptors, which are responsible to collect and save timestamps defined as follows:

- T1: Service Request's start time
- T2: Service Request's end time.
- T3: Service Response's start time.
- T4: Service Response's end time.

The lack of any of timestamps indicates a corrupted performance in the provider.

When the transaction is complete, the recorded timestamps are used to calculate QoS parameters of time, defined as follows:

- QoS1: T3 - T2 : Computation time
- QoS2: T2 - T1 : Requesting time.
- QoS3: T4 - T3 : Responding time.
- QoS4: (T4 - T1) - QoS1: Communication time.

The parameters are retrieved from the PaRe database table after a certain number of transactions, to perform diagnosis operations.

3.2 Diagnosis Module

The diagnosis stage implements the QoS analysis strategy, using a statistical model to compare the set of values obtained from monitoring with a reference model defined by analyzing the performance of the architecture. The result let to determine whether or not any recovery action is required.

The statistical model, exposed in [6], is based in the box-plot method. The Interquartile Range (IQR) and Right Outer Fence (Q3+3.0*IQR) were defined using samples of parameters from a first test . The portion of outlier values is observed, according to the measures established by each WS provider stored in the Service Level Parameter (SLP) table. The SLP table contains the measure known as "Left Outer Fence" for the QoS parameters of each WS with which it has interacted. During the first diagnosis process, SLP contains values that were calculated in ideal conditions and would represent good performance.

The model determines the percentage of outliers values (greater than Right Outer Fence) that are in a set of data collected. If the percentage is between 0% and 5% means that the health status is correct, between 5% and 10% indicates that the WS provider is degrading, and a second WS provider is requested to divide the work (Duplication). If the percentage is greater than 10% then the WS provider is degraded and other WS provider is necessary (Substitution).

3.3 Recovery Module

Recovery strategies are based on the system's component redundancy. The duplication and substitution of the WS provider allow that requests made by the WS consumer are met, avoiding results loss. Recovery depends on the availability of redundant WS.

Inside of the Self-healing Core is the WSTA database (Web Service Table Access) that stores information about the WS providers available. The records shall indicate the endpoint, status (online, offline, active, inactive), a description and operations offered by the WS. Records in the database are defined by an external agent, such as an administrator. The architecture does not add or delete information.

For the execution of recovery actions, the records in the WSTA are modified. Duplicating a WS is reflected as the activation of more than one provider (change of the *active* attribute). The WS substitution consists in disable the current provider (*active=false, offline=true*) and activate another available WS (*active=true*).

3.4 Overview Operation

The operation of the architecture is simple, keeping the Self-healing principles. The operation begins with a WS consumer request. To send it, a WSTA database query is performed requesting the most appropriate WS provider available (the database query includes the action and description required). When the WS consumer receives the endpoint to the WS provider, sends the request, initiating a monitored transaction; in case of failure, a recovery action is performed. Each transaction between the provider and the consumer follows the same process, whether they are equal.

After a certain number of transactions, the diagnosis module executes to detect and correct possible faults.

3.5 Critical Analysis of the Previous Architecture

The previous architecture has limitations with regard to its structure and not allowing the dynamism of WS.

The main limitation of the architecture is the centralized and rigid approach, only allowing a WS consumer interacts with one or more WS providers. Besides setting out a critical point, the Self-healing Core, where a fault may represent the complete loss of functionality, since the absence of recovery mechanisms on this component.

But the key point lies in the diagnosis module, where the QoS analysis is limited to observing time parameters of transactions. The dispersion measures applied do not consider variations in the interactions time intervals. In the diagnosis module, the WSTA table hasn't the ability to upgrade itself according to actual active providers, it depends on an update by the administrator.

It is necessary to accomplish the architecture distribution, improve the diagnosis process and establish adaptive mechanisms for the statistical model.

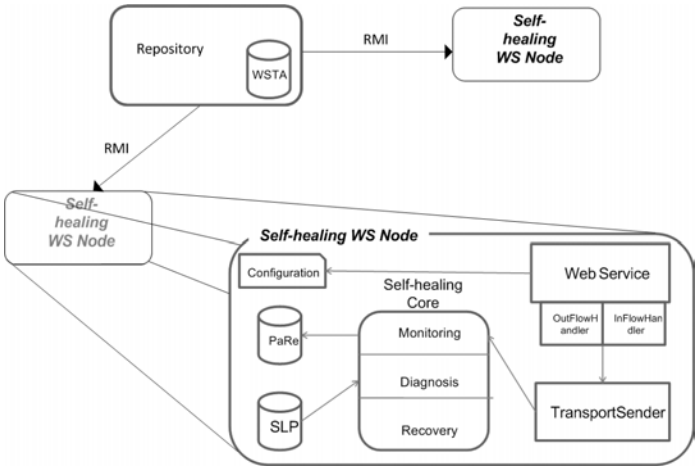

Fig. 1. Distributed Self-healing architecture.

4 Enhanced Distributed Self-healing Architecture

In our work we have made changes in the former architecture, improving Self-healing features, which were restricting the WS performance. This section presents the changes in the architecture's components. According to the paper's title, the major changes are in the diagnosis module.

The components distribution adapts the architecture to the dynamic interaction of the WS. WS can interact with each other according to their needs at any given time.

According to the project's objective, a digital library with distributed contents is obtained when each WS plays either provider or consumer roles, even both simultaneously.

Then each WS monitors the performance of its providers, assesses and determines the need to implement a recovery action, either changing provider or sending a request to more of them. The Self-healing Core, which was an isolated component, now is part of the Web Service. Each WS contains its own Self-healing components (Fig. 1).

Distributing components avoids critical points of failure that could have major consequences. The failure of a WS or a stack of Self-healing components has no effect beyond that WS. A serious flaw could require the full restoration of a WS, at most.

4.1 Repository Module

Actually, WSTA database is an independent component with a similar structure to the previous Self-healing Core. Its function is to register the WS availability, besides being the architecture's time server for consistency of timestamps.

Registering WS corresponds to each WS that integrates itself into the architecture. The changes applied to the records, as part of recovery actions, are visible to all other WS in the architecture.

4.2 Monitoring Module

The monitoring process has no major changes. The records and parameters are stored in local databases in every WS and are divided by each interacting WS provider.

The WS playing the consumer role in any interaction performs the monitoring.

4.3 Diagnosis Module

This module, in the enhanced architecture, will continue applying the defined statistical model, but is now complemented by an ontological model responsible for making inferences about the performance of the WS provider and setting the conditions to require a repairing action.

Adaptation of the reference measurements. The analysis of the observed parameters returns the percentage of outliers in the set according to the Right Outer Fence defined in the model.

A portion less than 5% indicates that the provider's performance is good, although there may be signs of increased time intervals. Values greater than 2.5% could indicate that the parameters are changing, perhaps due to changes in the underlying networking system.

Ignoring possible variations, the following diagnosis operations may invoke recovery actions, even if everything is working properly. Variations in parameters must be considered since the Right Outer Fence defined for a first test does not apply to sets of parameters in all contexts. Adapting the model's descriptive measures is required.

In the new diagnosis implementation, if the percentage of outliers parameters is between 3% and 5%, a necessary measure adaptation action is established. The changes are stored in the SLP table for further use. The box-plot method is executed using the set of values for each QoS parameter, determining a new Right Outer Fence's value giving a new diagnosis' perception.

If a set of values of a QoS parameter requires a measures adaptation, the same procedure for the three remaining parameters is performed.

The variation in one parameter may involve variations in the other, so if a parameter requires their descriptive measure redefinition, it is likely that someone else also makes that request. Executing adaptation strategies for all parameters prevents possible future recurrences.

The first execution of the diagnosis process uses the descriptive measurements defined in the previous architecture and according to the results may or may not request a re-calculation (adaptation). The adjustment is incremental in the

$s \in status \wedge HasParameter(s, qos) \wedge GreaterThan(qos, qosdup) \wedge$
$LessThan(qos, qossub)$
$\Rightarrow HasDegradation(s, moderate)$

$s \in status \wedge HasDegradation(s, moderate)$
$\Rightarrow NeedRecovery(s, duplication)$

Fig. 2. Rules in diagnosis module.

Left Outer Fence value; no change in this measure indicates system's operation stability.

When the implementation of recovery actions is determined, any kind of adaptation does not take place, since degradation might be ignored.

Finally, the results of the statistical model are sent to the ontological model to proceed with the stage of diagnosis.

Diagnosis based on ontologies. Diagnosis based on the statistical method may be considered incomplete. WS performance is reflected in several factors.

The QoS parameters such as availability, accessibility, integrity, performance, reliability, control and safety [12] may also be appropriate indicators of the WS provider performance.

To get an insight into these parameters, the WS must be aware of their own state and act to prevent or recover from failures or degradations in its performance.

Ontologies are an alternative to model the environment and the state of the WS, showing capacity to make inferences, using logical rules, to determine a good or bad performance.

In a previous work [7] we defined an ontological model that includes QoS parameters and rules to infer the current state of architecture. In the present work, the Self-healing architecture's diagnosis module is attached to a rules engine that is responsible for analyzing the data collected by monitoring, using the ontological model. So far, the parameters analyzed were the time intervals already defined.

The change resides in the fact that results are no longer mere assertions, now consist of logical conclusions by applying first-order logic sentences as shown in Fig. 2.

For implementation, we used Jess rule engine [13], which use OWL-DL ontologies via the Protégé and its complement JessTab. With those tools we build a package that provides methods to send and receive data from the ontological model. This package is accessible from the methods implemented in architecture components.

When the ontological model receives the necessary parameters from the statistical model, those are represented in the ontology and the logical rules compare the current performance with a reference model for diagnosis

4.4 Recovery Module

The Recovery Module implements strategies to recover an appropriate of performance when the diagnosis module may determine necessary. Currently, each WS applies recovery strategies from its recovery module.

The strategies are techniques of component redundancy and consist in changes in WSTA information indicating the activation or inactivation of WS providers in the architecture.

WS modifications over the WSTA database do affect the operation of everyone. If a provider is set to offline, will not receive requests from customers, however, it can continue with the role of consumer.

5 Results

The main result to present is the adaptation of the statistical model's measures.

In the test showed in Fig. 3, we obtained a set of values of QoS1 parameter during the interaction between WS "botany" and WS "math". If this set is analyzed using the default Right Outer Fence, defined in an earlier trial (365.5), the percentage of outlier values is 3.16% of the total (40) value obtained from the previous architecture (See Fig. 4).

Implementing the adaptation strategy, in the enhanced diagnosis module, gets a new IQR = 142.5 and redefines the Right Outer Fence in 585.5. In this case, the percentage of outliers values is 0%, preventing any unnecessary recovery action.

The value 0% is transferred to the ontology, inferring a good performance.

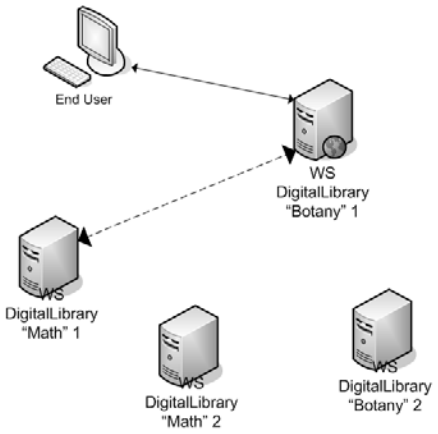

Fig. 3. Test scenario.

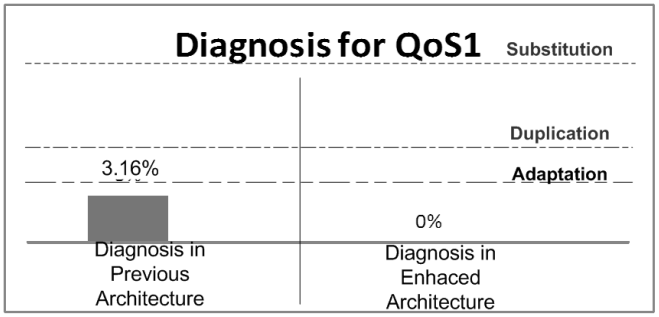

Fig. 4. Comparing diagnosis modules.

6 Conclusion

All changes applied to previous Self-healing architecture were not complicated to apply and contributed to obtain better results.

Distribution added to previous architecture allowed the possibility to distribute WS applications modules.

Quality of service reference measures updated constantly, allowed to architecture precision about the application healing, by providing the capacity to identify normal changes from real failures.

Current Self-healing architecture allows modifying constantly QoS measures, adapting to new conditions. However, adaptation is incremental as a future work that could be improving finding the best measures to be applied on each transaction.

An ontology module integrated into diagnosis, allowed other information about WS provider's performance giving better results. By now, only time parameters are considered, the addition of more parameters will be a future work.

7 Acknowledgments

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

References

1. Shaw, S.: "Self-Healing": Softening Precision to Avoid Brittleness. Proceedings of the first workshop on Self-healing systems, pp. 111-114. ACM, New York, USA (2002)
2. Kephart, J., Chess, D.: The vision of autonomic computing. Computer, 1(36):41-50 (2003).
3. Halima, R., Drira, K., Jmaiel M.: A comparative study of self-healing architectures in distributed systems. Rapport LAAS No 06568 (2006).
4. Ghosh, D., Sharman, R., Raghav, H., Upadhyaya S.: Self-healing systems - survey and synthesis. Decision Support Systems 42, pp. 2164-2185 (2007).

5. Moo-Mena, F., Garcilazo-Ortiz, J., Basto-Díaz, L., Curi-Quintal, F., Alonzo-Canul, F.: Defining a SelfHealing QoS based Infrastructure for Web Services Applications. In: IEEE 11th International Conference on Computational Science and Engineering, pp. 215-220. IEEE Press (2008).
6. Moo-Mena, F., Garcilazo-Ortiz, J., Basto-Díaz, L., Curi-Quintal, F., Medina-Peralta, S., Alonzo-Canul, F.: 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. IEEE Press (2009).
7. Moo-Mena, F., Garcilazo-Ortiz, J., Basto-Díaz, L., Curi-Quintal, F., y Canul-Centeno, F., Una ontología para el diagnóstico de la QoS en aplicaciones basadas en servicios Web (WS). In: XXII Congreso Nacional y Vlll Congreso Internacional de Informática y Computación ANIEI (2009).
8. Wang, M., Suda, T.: The bio-networking architecture: A biologically inspired approach to the design of scalable, adaptive, and survivable/available network applications. Tech. Rep. 00-03, Department of Information and Computer Science, Unversity of California, Irvine, California (2000).
9. Zhang, W., Hansen, K.: Towards Self-managed Pervasive Middleware using OWL/SWRL ontologies. In: Fifth International Workshop on Modeling and Reasoning in Context, pp. 1-12. (2008)
10. Ribeiro-Justo, G., Karran, T.: An Object-Oriented Organic Architecture for Next Generation Intelligent Reconfigurable Mobile Networks. In: Blair, Gordon, (ed.) DOA'01: 3rd International Symposium on Distributed Objects and Applications, pp. 31-40. IEEE Conference Proceedings . IEEE Computer Society, Las Alamitos, USA (2001).
11. Console, L.,Fugini, M.: WS-Diamond: An Approach to Web Services, Diagnosability, Monitoring, and Diagnosis. tech. report 2007.57, Dept. Electronics and Information, Politecnico di Milano. (2007).
12. IBM: Understanding quality of service for Web services. http://www.ibm.com/developerworks/library/ws-quality.html
13. Ernest Friedman-Hill: Jess The Rule Engine for the JavaTM Platform Version 7.1p2. Sandia National Laboratories. (2008).