International Journal of Electrical and Computer Engineering (IJECE)
Vol. 3, No. 6, December 2013, pp. 779~783
ISSN: 2088-8708

Architectural Approaches for Self-Healing Systems Based on Multi Agent Technologies

Hamid Bagheri[1], Mohammad Ali Torkamani[2], Zhaleh Ghaffari[1]
[1]University of Kurdistan, Iran
[2]Iranian Telecommunication Manufacturing Company, Iran

Article Info

Article history:

Received Aug 2, 2013
Revised Oct 3, 2013
Accepted Oct 25, 2013

Keyword:

ATM
Biometric sensor
Multi Agent
Self healing
Software Architecture

ABSTRACT

Self-healing systems are able to adapt themselves at runtime time in response to changing environmental or operational circumstances, shifting user requirements, and unanticipated faults without human intervention. Conceptually, a self-managing system is composed of four key capabilities; Monitoring, performing Analysis, Planning and Executing the plan. The preferred way to enable repair in a self-healing system is to use externalized repair/adaptation architecture. Adaptability, dynamicity, awareness, observability, autonomy, robustness, distributability, mobility and traceability are requirements that an architecture style for self-healing system should satisfy. In this paper we discuss Multi agent based self-healing system has a characteristics that can satisfy mentioned requirement. We define associations between architecture style requirements for self-healing system and MAS characteristics. As a case study in a real project we have designed Automated Teller Machine (ATM) combination with biometric sensors based on multi-agent architecture.

Corresponding Author:

Hamid Bagheri,
Information Technology,
Kurdistan University,
Iran, Kurdistan, Sanandaj, Pasdaran street
Email: bagheri.hamid@gmail.com

1. INTRODUCTION

Software change at the level of its architecture-that is, in terms of its components and connectors, is the approach that offers the most flexibility in the types of repairs that can be performed in a system. Component boundaries are, ideally, the most loosely coupled connection points in a software system, making them the most flexible points of reconfiguration [1-5].

Software architectures provide high-level abstractions for representing the structure, behavior, and key properties of a software system [3]. These abstractions involve descriptions of the elements from which systems are built, interactions among those elements, patterns that guide their composition, and constraints on those patterns [4]. Self-healing systems are an emerging class of software systems that exhibit the ability to adapt themselves at run time to handle situations such as resource variability, changing user needs, and system faults [6-7]. In [1] requirements that an architectural style for self-healing systems should satisfy is discussed: adaptability, dynamicity, awareness, observability, autonomy, robustness, distributability, mobility, and traceability.

The paper is organized as follows. Section II covers self-healing systems requirements. Section III describes the Multi Agents systems characteristics. In section IV we correlate requirements on Self-healing Systems with the properties of Multi Agents Systems. Next, in section V we present our solution in a real project as a case study. In section VI we present our conclusions.

2. SELF-HEALING SYSTEMS REQUIREMENTS

Self-healing systems have the ability to modify their own behavior in response to changes in their environment, such as resource variability, changing user needs, mobility, and system faults. The traditional approach to performing repairs on a system is to stop the system, make the necessary updates and restart the modified system. However, based on the conceptual architecture for a self-managing system introduced in [8] we expect a self-healing system to be able to perform a repair of its components as part of a proactive [2], preventative or reactive response to its operating environment while it executes.

The lifecycle of self-healing systems as consisting of four major activities:

1. Monitoring the system at runtime
2. Planning the changes
3. Deploying the change descriptions, and
4. Enacting the changes.

Requirements that an architectural style for self-healing systems should satisfy is [1]:

- Adaptability: The style should enable modification of a system's static (i.e., structural and topological) and dynamic (i.e., behavioral and interaction) aspects.
- Dynamicity: Encapsulates system adaptability concerns during run-time (e.g., communication integrity and internal state consistency).
- Awareness: The style should support reflection i.e., monitoring of a system's own performance (state, behavior, correctness, reliability, and so forth) and recognition of anomalies in that performance. The style should also support observability i.e., monitoring of the system's execution environment.
- Observability: The style should also support observability i.e., monitoring of the system's execution environment.
- Autonomy: The style should provide the ability to address the anomalies in the performance of a resulting system and/or its execution environment. Autonomy is achieved by planning, deploying, and enacting the necessary changes.
- Robustness: The style should provide the ability for a resulting system to effectively respond to unforeseen operating conditions. Such conditions may be imposed by the system's external environment (e.g., malicious attacks, unpredictable behavior of the system's run-time substrate, unintended system usage), as well as errors, faults, and failures within the system itself. Note that this definition of robustness subsumes fault tolerance.
- Distributability: The style should support effective performance of a resulting system in the face of different distribution/deployment profiles.
- Mobility: The style should provide the ability to dynamically change the (physical or logical) locations of a system's constituent elements.

Traceability: The style should clearly relate a system's architectural elements to the system's execution-level modules in order to enable change enactment in support of the above requirements.

3. MAPPING SELF-HEALING REQUIREMENT TO MULTI AGENT CHARACTERISTICS

Agents are software or hardware elements that operate within an environment, act and sense, and communicate and collaborate with other elements. Agents are defined by attributes specific to their functional domain. For example, intelligent agents include attributes that emulate human mental processes. Properties of multi agent systems include:

- Distribution: multi agent systems are distributed which are spread across network. Socket communication or middleware are used to develop multi agent systems [6].
- Decentralization: there is no central node to collect results or issue commands. Agents act independently and attain their personal goal.
- Local views: no agent has a full global view of the system, or the system is too complex for an agent to make practical use of such knowledge.
- Self-organization and self-steering Multi-agent systems can manifest self-organization as well as self-steering and other control paradigms and related complex behaviors even when the individual strategies of all their agents are simple.
- Social: in multi-agent system every agent knows other agents and their capabilities and may ask them for help.
- Interaction: agents interact and exchange information to cooperate.
- Goal-oriented: every agent follows goals which are specified by system or user.

4. MAPPING SELF-HEALING REQUIREMENT TO MULTI AGENT CHARACTERISTICS

Characteristics of MAS provide a natural framework for developing a self-healing methodology. Each requirement for self-healing systems is decomposed into MAS characteristics:

1. Adaptability: Agents are Intelligent
2. Dynamicity: Agents have dynamic behavior.
3. Awareness: Monitoring Agent can monitor resources, size of the log files generated by components.
4. Observability: Monitoring Agent can monitor system's environment.
5. Autonomy: Agents are autonomous
6. Robustness: Agents have collaborative behavior and autonomous
7. Distributability: multi agent systems are distributed which are spread across network. Socket communication or middleware are used to develop multi agent systems.
8. Mobility: Agents are mobile.
9. Traceability: Agents can operate within an environment, act and sense, and communicate and collaborate with other agents.

In next section we present real project which has designed in Iranian Telecommunication Manufacturing Company (ITMC).

5. CASE STUDY

Iranian Telecommunication Manufacturing Company (ITMC) is a company operating in Electrical engineering and ICT areas. ITMC besides some products in electrical and communication area participates in software areas. One of these projects which are designed is Automated Teller Machine (ATM) combination with biometric sensors.

Traditionally, access to secure areas or sensitive information has been controlled by possession of a particular key card or password. Nowadays, people have PINs and passwords for a large number of devices, from the different social networks and sites, to their bank information. Techniques which leverage biometric sensors may ease so many problems related to authentication and authorization issues. Biometric sensors can confirm that a person is actually present without requiring the user to remember anything and there is no security threat.

We have designed the projects based on multi agent architecture. From self-healing point of view, functional requirements for an ATM for instance observability, traceability and awareness are satisfied with multi-agent architecture. This project leverages agents for reaching awareness, traceability and observability, more details are shown in table 1. This project was developed based on SQL Server and Visual C#.

Table 1. ATM requirements

Architectural requirement for ATM		Solutions of the system with multi agent
Data integrity & consistency in ATM	–	Leverage dynamic agents
Monitoring ATM bandwidth and anomaly detection	–	Monitoring agent
	–	Alarm to the police
	–	Send log to the Bank manager
Monitoring ATM's execution	–	Monitoring agents
Address the anomalies in the performance of the system	–	Autonomy of Agents
	–	Planning skills of Agents
High security in ATM and monitoring attacks to the ATM	–	collaborative behavior with monitoring and security agents
Effective performance in responding to Bank customers	–	Decentralization in MAS across the network
	–	Replication fingerprints database in all regions in country
Monitoring all component of the system	–	In cooperation with monitoring agent
	–	DDOS attacks monitoring

By leverage MAS approach availability and security level of the ATM system have been increased result in stockholder satisfactory. As we showed in previous work [9], based on results of this paper theoretically we can reach the availability about 99%.We developed an agents which responsibility was Heartbeat. Besides that, system was tested in three month and we had no security threat and unavailability.

6. CONCLUSION

Self-healing systems are an emerging class of software systems that exhibit the ability to adapt themselves at run time to handle situations such as resource variability, changing user needs, and system faults. Requirements that an architectural style for self-healing systems should satisfy is: adaptability, dynamicity, awareness, observability, autonomy, robustness, distributability, mobility, and traceability. Multi agent based self-healing system has characteristics that can support mentioned requirement. In this paper we defined associations between architecture style requirements for self-healing system and MAS characteristics (Table 2). As a case study in a real project we have designed Automated Teller Machine (ATM) combination with biometric sensors based on multi-agent architecture. This project leverages agents for satisfying self-healing systems requirements such as observability, traceability and awareness.

Table 2. Support Architectural Requirement by Mas Characteristics

No	Architectural requirement for self-healing systems	MAS characteristic
1	Adaptability	Agents Are Independent and Intelligent
2	Dynamicity	Dynamic agents
3	Awareness	Monitoring Agents
4	Observability	Monitoring agents
5	Autonomy	Autonomy of Agents Planning skills of Agents
6	Robustness	Autonomy of Agents collaborative behavior
7	Distributability	Distributed across network Decentralization in MAS
8	Mobility	Mobility in Agents
9	Traceability	Reactivity in Agents

FUTURE WORK

For the future work we would like there to be more discussion of this mapping and the issues that arise. One of the most important characteristics in ATM systems is availability; based on MAS characteristics we can reach the availability about 99%. We will discuss more details about how we achieve availability based on MAS.

REFERENCES

[1] Nikunj Mehta, 2002. Architectural Style Requirements for Self-Healing Systems WOSS '02, Nov 18-19, 2002, Charleston, SC, USA.Copyright 2002 ACM 1-58113-609-9/02/0011 ...$5.00,(2002).
[2] Jeongmin Park, Giljong Yoo, and Eunseok Lee. "*Proactive Self-Healing System based on Multi-Agent Technologies*". School of Information and Communication Engineering Sungkyunkwan University. 2005.
[3] M Shaw and D Garlan. Software Architecture: Perspectives on an Emerging Discipline. Prentice-Hall. 1996.
[4] David S. Wile and Alexander Egyed. "An Externalized Infrastructure for Self-Healing Systems". *Teknowledge Corporation*. 2004.
[5] Michael E Shin and Jung Hoon An. "Self-Reconfiguration in Self-Healing Systems". Department of Computer Science,Texas Tech University. 2006.
[6] Hong Mei, Gang Huang, Wei-Tek Tsai. *Towards Self-Healing Systems via Dependable Architecture and Reflective Middleware*. Proceedings of the 10th IEEE International Workshop on Object-Oriented Real-Time Dependable Systems (WORDS'05)0-7695-2347-1/05 $20.00 © 2005 IEEE. 2005.
[7] ACM SIGSOFT Workshop on Self-Healing Systems (WOSS'02) Call for Papers. http://www-2.cs.cmu.edu/~garlan/woss02/
[8] JO Kephart and DM Chess. The vision of autonomic computing. *Computer Magazine*. 2003.
[9] Mohammad Ali Torkamani, Hamid Bagheri and Salam Abdollah Shaltooki, Mohammad Reza Tabandehd, Amin Eghlidi Negad. *Architectural Solution for reaching high availability*. 3rdWorld Conference on Innovation and Computer Sciences, Turkey. 2013.

BIOGRAPHIES OF AUTHORS

Hamid Bagheri was born in sanandaj, Iran, in 1979. He received the B.S. and M.S. degrees in Software engineering from the Shahid Beheshti University, Tehran, in 2011.
Since 2009, he has been working in Information Technology in Kurdistan University. His research interests include Service Oriented Arhcitectureand Ultra large Scale Systems.

Mohammad Ali Torkamani born in Iran, Shiraz City, in 1975. He received the M.S. degree in software engineering from the ShahidBeheshti University, in 2011. He is the author of 15 books (in Persian), more than 35 articles. His research interests include software architecture, Ultra Large Scale systems, cryptography and Network security holds one patent.
He is working in R&D Department of Iranian Telecommunication Manufacturing Company now. He is currently teaching at the ITMC University of Applied Science and Technology in Shiraz.

Zhaleh Ghaffari was born in sanandaj, Iran, in 1981. She received the B.S. in Software engineering from the AzadUniversity, Hamedan, in 2006.
Since 2009, she has been working in Engineering Facuty in Kurdistan University. Her research interests include Web Services and Database.