ProHeal
(State of the art on Self*-principles)
ProHeal - Automated Protection and Healing Software Solutions

v1, 2015-02-17

Contents

List of Figures

1. State of the art on Self*-principles

The need for high-performance and QoS of smart environment applications demands novel solutions for autonomic adaptation and evolution of systems at run-time to support self-* principles software solutions. In the last few years, many papers discussed the research challenges in developing self-* solutions. In the following, we provide an overview of the state-of-the-art self-* principles practices such as self-healing, self-protection, and self-configuration, in various domains and related works (1) (2).

1.1. Self-healing

Self-healing addresses attributes like reliability, availability, and maintainability of dependability (3) (4). It is also a first attempt to attain the various attributes of dependability in smart environment applications with the self-healing property. The adoption of SEs is hindered by the fact that there is constant need for human (or even expert) intervention and the cost of maintenance of such systems is very high. Thus, dependable systems are required, evolving at runtime to maximize the reliability and availability of their applications. Two levels of dependability mechanisms are identified, i.e., proactive mechanisms in the absence of faults and reactive mechanisms in the presence of faults. Systems that have the ability to predict faulty behaviors and make the necessary alterations to prevent faults before they occur, or at least delay them without human intervention (5) are said to be self-healing. There are two goals of this:

1. to maintain application functionality as long as possible,
2. graceful degradation of application performance.

Self-healing accordingly comprise monitoring of the healed system, analysis of the monitored data to identify problems, decision up on the healing action to be performed, and performance of this action. The results of the action will feed into the monitoring component, thus closing the control loop, allowing feedback and improvement of the healing (6). Examples of such loops can be found in multiple specific self-healing solution as well as a few generic architectures such as the Panacea architecture in (7). The main challenges of self-healing systems are presented in (6), (8). Further, they describe the self-healing process notions, and basic characteristics of self-healing systems. Self-healing systems are emerging as a useful approach in addressing the rising complexity requirements of systems management. A survey of self-healing frameworks and methodologies in multi-tier architectures is presented by Chris et al. (9).

The complexity of large-scale smart environment applications technologies impose significant maintenance challenges, primarily due to sudden failures and environmental changes. Low capacity sensor and actuator nodes play an important role in smart environment on the technology side as they provide the bridge between the digital world and the physical world. Therefore, a smart environment application relies first and foremost on sensory data acquired from multiple sensors in various locations (10). These sensor nodes are typically small, wireless, and battery-powered nodes, prone to faults due to internal and external influences, such as low battery and memory, hardware/software faults, environmental interferences and sensor aging (11). Figure 6 visualizes how a fault can evolve into a failure in smart environment applications (3). Fault is a deviation of at least one characteristic property or parameter of the system from normal operation (3). Error is an inconsistent or erroneous state because of a deviation from the required operation of system or subsystem (3). The presence of an error might cause a whole system to deviate from its required operation. A failure is defined as an event that occurs when the delivered service deviates from correct service (3). Accordingly, an application failure occurs when the system fails to perform its required function because provided service not according to specification. Therefore, it is necessary to detect and recover faults at run-time to minimize manual interventions.

The goal of self-healing property is to provide applications that are reliable, highly available and dependable. Self-healing attempts to monitor and analyze sensory data to autonomously predict and detect, and prevent and heal faults respectively. Avizienis et al. (3) present basic concepts of dependability attributes such as availability, reliability, safety, confidentiality, integrity, maintainability, in systems. A taxonomy for describing the problem space for self-healing systems is presented by Koopman (12). Harald and Dustdar (13) provide an insight into self-healing approaches in different challenging research areas such as embedded and operating, multi agent-based, and Web services systems. Further, Ghosh et al. (5) present an in-depth review of the various approaches to self-healing systems in different domains. Mostly, research efforts aim at developing of self-healing systems focuses on fault-recovery on detection of faults. For example, a fault management mechanism is proposed by Asim et al. (14) to detect and recover faults in sensor network applications using a reactive approach. Further, a self-healing application for autonomic pervasive computing is presented by Ahmed et al (15), which detect faults, notify other devices of these faults, and try to recover from these faults. Further, a fault-tolerance middleware that stores all crucial information including log status of the faulty device is presented in (15). Park et al. (16) present a self-healing system that monitors, diagnoses and heals its own internal problems using self-awareness as contextual information.

Moreover, a failure detection, assessment and adaptation approach is proposed to recover from failures using application semantics for smart home applications in (17). A healing manager that recovers from detected faults is introduced (17). In (18), a framework for satisfying reliability requirements in embedded and mobile software systems is presented. The software is continuously analyzed and dynamically adapted based on detected events. To the best of our knowledge, the fault tolerance approaches in the literature are based on reactive mechanisms in the presence of faults to prevent failures, and minimizing the recovery time of a detected fault (19). The proposed fault-prevention framework distinguishes itself from existing literature by its proactive approach in the absence of faults. The benefits of using a proactive approach are well recognized in the area of real-time embedded systems (20).

A reference architectural framework is developed in (21) to structure the requirements for the design of computing system with self-management properties, such as self-aware, self-expressive. Chris et al. (9), present an approach to identify autonomously the source of a fault within a system by using machine learning algorithms. This is envisioned as a baseline for future evaluations of the relative performance of self-healing approaches. A middleware, namely, MARKS addressed the knowledge usability, resource discovery and self-healing properties (fault-detection, resource recovery) of pervasive computing (22). However, the proposed middleware in (22) followed the reactive dependability approach. A generic framework for modeling self-healing software systems is presented in (23). Self-healing is achieved by transforming the self-healing models into application specific implementations that provides failure resolutions to mitigate the effect of software faults (23). Li et al. (24) present a prediction based dependency constraint directed self-healing scheme for wireless sensor networks that monitors for system failures and maintain system performance under faults and failure conditions.

1.2. Self-protection

Today, human activity is getting ever more dependent on computing systems. They are used extensively to process and store confidential information. Hackers and intruders make successful attempts to attack company networks, web services, and even their database system directly on a daily basis. Hence, security is now major concern for and IT infrastructure. Smart environments business something like Remote Health Care System, Smart Transportation, or any services related IOTs must have Data Storage systems to provide better services, to keep profitable operations both for customer or company itself (25). Self-Protection is the ability of an autonomic system to secure itself against attacks, i.e. to detect illegal activities and to trigger counter measures in order to stop them. In the project of the first year, WareValley will focus on developing enhanced Database Security features. So, we aimed two goals of this:

1. Supports cloud environment such as Amazon Web Service.
2. Develop enhanced security feature which is in Distributed Database System that allows the various sites to implement different security policies.

Recently, Interest variety of IT services and computing resources are increasing. As a result, the interest in the security of cloud environment is also increasing. Cloud environment is stored that to provide services to a large amount of IT resources on the Cloud. Therefore, Cloud is integrity of the stored data and resources that such as data leakage, forgery, etc. security incidents that the ability to quickly process is required (26). However, the existing developed various solutions or studies without considering their cloud environment for development and research to graft in a cloud environment because it has been difficult. Therefore, we proposed wire-wireless integrated Security management Model in cloud environment.

According to Gartner Research, 78% of IT managers cite security and compliance as a barrier to cloud adaption, which is why we architected the Chakra Max database security solution to address and mitigate these issues (27). WareValley will provide customers with comprehensive real-time interception of malicious and inadvertent threats to Cloud Environment such as Amazon Web Services databases, while also allowing developers and contractors to perform their duties without ever gaining access to sensitive or protected information. Chakra Max first shall discover where sensitive data is located and then protects customers from SQL injection attacks. It enables organizations to secure sensitive information (e.g., SSNs, credit card number, medical history, etc.) from unauthorized database access, enforce separation-of-duties, mask sensitive data, monitor and audit data access and meet regulatory compliance requirements, such as PCI and HIPPA. It will also install as a front-end to the cloud database (28).

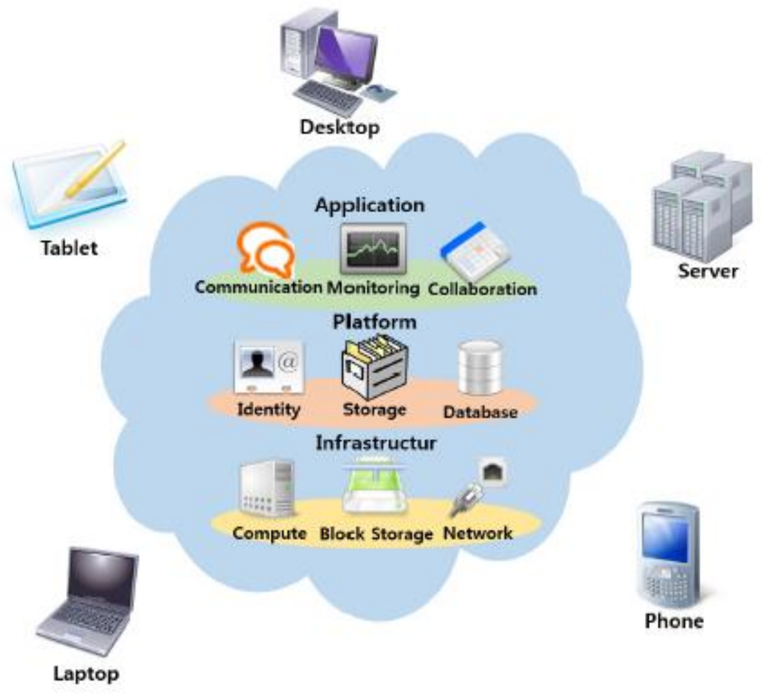

Figure 1: General Cloud Environment

Another major features in the concept of Self-Protection is secured federated database. It is a distributed database that allows the various sites to implement different security policies. The policy of a site is then enforced by all sites for data owned by this site. WareValley will describe a proof of concept prototype of such a multi-policy secure federated database. The prototype implements a multilevel federal security policy- that is a policy that applies to all members of the federation. The individual sites can then extend this policy for data owned by them with discretionary and/or multilevel site security policies.

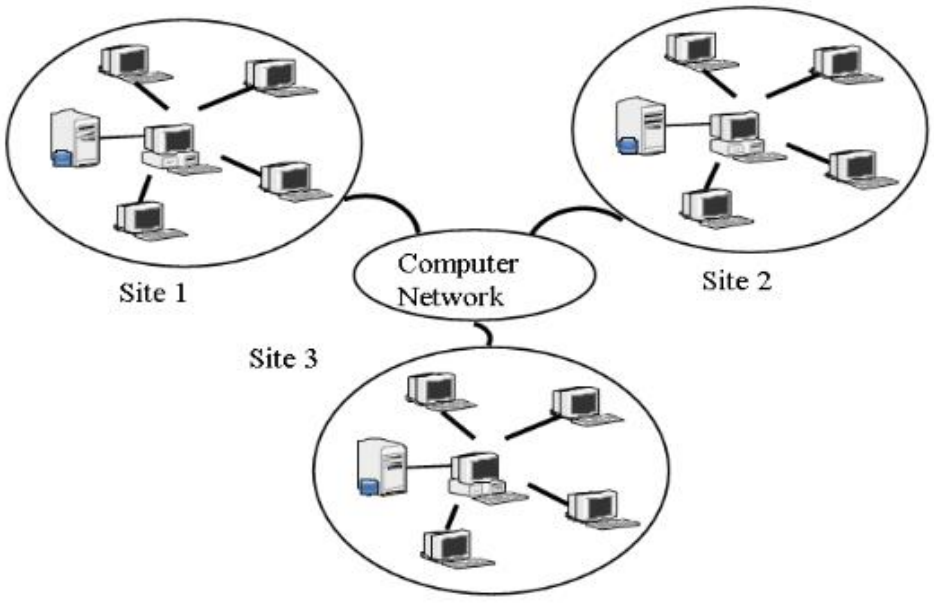

Figure 2: Secure Distributed Database

Distributed database system functions include distributed query management, distributed transaction processing, distributed metadata management, enforcing security and integrity across multiple nodes.

Database security is the system, processes and procedures that protect a database from unintended activity. Unintended activity can be categorized as authenticated misuse, malicious attacks or inadvertent mistakes made by authorized individuals or processes. All organizations, ranging from commercial organizations to social organizations, in a variety of domains such as healthcare and homeland protection, may suffer heavy losses from both financial and human points of view as a consequence of unauthorized data observation. Some of the most important security requirements for database management systems are: multi-level access control, confidentiality, reliability, integrity, and recovery. Thus, a complete solution to data security must meet the following three requirements: 1) secrecy or confidentiality refers to the protection of data against unauthorized disclosure 2) integrity refers to the prevention of unauthorized and improper data modification and 3) availability refers to the prevention and recovery from hardware and software errors and from malicious data access denials. Often such a database is owned by more than one organization, each with its own view about security. Such different views on security complicates interaction. A DBMS fulfilling these mandatory requirements becomes capable of providing security at different level. But in order to provide concurrent access of replicated data to multiple users at different locations, multilevel security mainly on distributed environment becomes a major issues.

1.3. Self-configuration

For deployment of scalable self-organizing networks there is an inherent functionality that is called self-configuration. Such self-configuration is implemented through a so-called *joining* mechanism. A smart device not currently part of a wireless network (http://en.wikipedia.org/wiki/MyriaNed) when exposed to a fellow smart device will eventually form a network of two. When another smart device is in the range of the network-of-two it will also join the network. The network becomes larger and larger as new members are added. This leads to a self-organizing network without human intervention. Meta information is passed in the network messages and is used by each and everyone to determine the correct parameters for it to join the group.

The joining mechanism is always active so the self-organizing network will always add new devices or otherwise reconnect (lost) devices again. The messaging through the network is maintained and is resilient against breaking up as there is no a-priori routing of messages.

Smart devices are communicating among each other using time-slotted communication. The allocated slots (horizontal) are displayed in time (vertical), which results in a waterfall diagram in Figure 5 .

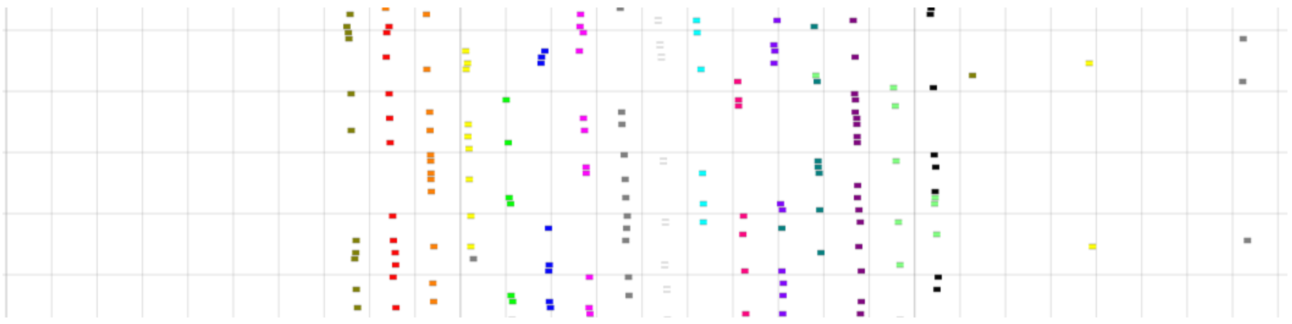

Figure 3: Time-slotted communication

In Figure 6 all possible time slots are displayed. The spreading of the random joining slots is visible along with a cluster of smart devices that are already formed an ad-hoc network.

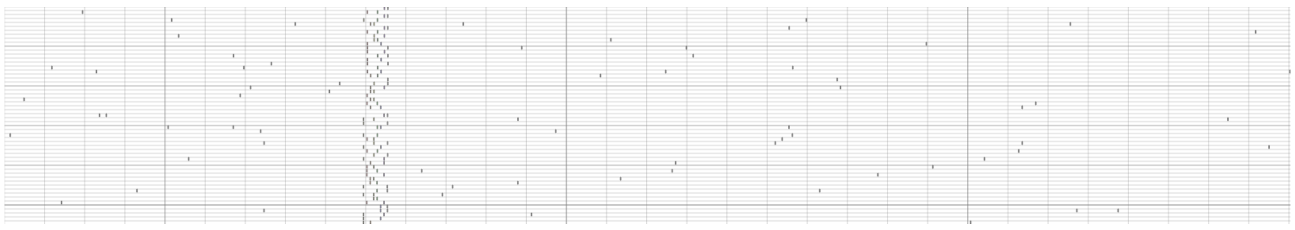

Figure 4: Random joining slots

Bibliography

1. *Towards an autonomic computing environment.* **R. Sterritt, D. Bustard.** 2003. 14th International Workshop on Database and Expert Systems Applications. pp. 694-698.

2. *Self Management for Large-Scale Distributed Systems: An Overview of the SELFMAN Project.* **P. Roy, S. Haridi, A. Reinefeld, J. Stefani, R. Yap, Coupaye, T.** 153-178, s.l. : Formal Methods for Components and Objects, 2008, Vol. 5382.

3. *Basic concepts and taxonomy of dependable and Secure computing.* **A. Avizienis, J.-C. Laprie, B. Randell, C. Landwehr.** 1, 2004, IEEE Trans. on Dependable and Secure Computing, Vol. 1, pp. 11-33.

4. *A survey of self-healing systems frameworks.* **Schneider, Chris and Barker, Adam and Dobson, Simon.** s.l. : Software: Practice and Experience, 2014.

5. *Self-healing systems - survey and synthesis.* **D. Ghosh, S. Raj, H.R. Rao, U. Shambhu.** 4, 2007 , Decision Support System, Vol. 42, pp. 2164-2185.

6. *Self-Healing and Recovery Methods and their Classification.* **O. Shehory, J. Martinez, A. Andrzejak, C. Cappiello, W. Funika, D. Kondo, L. Mariani, B. Satzger, M. Schmid, A. Andrzejak et al.,.** 2009.

7. *PANACEA Towards a Self-healing Development Framework.* **Breitgand, D. and Goldstein, M. and Henis, E. and Shehory, O. and Weinsberg, Y.** 2007. 10th IFIP/IEEE International Symposium on Integrated Network Management . pp. 169-178.

8. *Self-Healing Systems: Foundations and Challenges.* **Gabi Dreo Rodosek, kurt Geihs, Hartmut Schmeck, and Burkhard Stiller.** s.l. : Dagstuhl Seminar Proceedings, Combinatorial Sceintific Computing, 2009.

9. *Autonomous Fault Detection in Self-Healing Systems: Comparing Hidden Markov Models and Artificial Neural Networks.* **Chris Schneider, Adam Barker, and Simon Dobson.** 2014. In Proceedings of International Workshop on Adaptive Self-tuning Computing Systems (ADAPT). pp. 8-24.

10. **Lewis, F. L.** *Wireless Sensor Networks, in Smart Environments: Technologies, Protocols, and Applications.* s.l. : John Wiley & Sons, Inc., Hoboken, NJ, USA, 2005.

11. *Anomaly-based fault detection System in Distributed System.* **K. Byoung, S. Hariri.** 2007. 5th ACIS Interernational Conference on Software Engineering Research, Management and Applications . pp. 782-789.

12. *Elements of the Self-Healing System Problem Space.* **Koopman, P.** Portland, Oregon : s.n., 2003. Workshop on Software Architectures for Dependable Systems, ICSE'03 International Conference on Software Engineering. pp. 3-11.

13. *A survey on self-healing systems: approaches and systems.* **P. Harald, S. Dustdar,.** 1, 2011, Computing, Vol. 91, pp. 43-73.

14. *A self-managing fault management mechanism for wireless sensor networks.* **A. Muhammad, M. Hala, M. Madjid.** 4, 2010, Journal of Wireless & Mobile Networks, Vol. 2, pp. 1-14.

15. *Self-healing for Autonomic Pervasive Computing.* **S. Ahmed, A. I. Sheikh, M. Sharmin, C. Hasan.** 2009, Autonomic Communication, pp. 285-307.

16. *Proactive Self-healing System for Application Maintenance in Ubiquitous Computing Environment.* **J. Park, G. Yoo, C. Jeong, E. Lee.** 2006, Computer Science and Its Applications, Vol. 3981, pp. 430-440.

17. *Being SMART About Failures: Assessing Repairs in SMART Homes.* **K. Kapitanova, E. Hoque, J. Stankovic, K. Whitehouse, S.H. Son.** 2012, ACM Conference on Ubiquitous Computing, pp. 51-60.

18. *Proactive Self-Adaptation for Improving the Reliability of Mission-Critical, Embedded, and Mobile Software.* **Cooray, D. and Kouroshfar, E. and Malek, S. and Roshandel, R.** 12, 2013, IEEE Transactions on Software Engineering, Vol. 39, pp. 1714-1735.

19. *A Survey of Online Failure Prediction Methods.* **Salfner, Felix and Lenk, Maren and Malek, Miroslaw.** s.l. : 42, 2010, ACM Comput. Surv., Vol. 3, pp. 10:1--10:42.

20. *System-scenario-based Design of Dynamic Embedded Systems.* **Gheorghita, Stefan Valentin and Palkovic, Martin and Hamers, Juan and Vandecappelle, Arnout and Mamagkakis, Stelios and Basten, Twan and Eeckhout, Lieven and Corporaal, Henk and Catthoor, Francky and Vandeputte, Frederik and Bosschere, Koen De.** 1, 2009, ACM Trans. Design Automation of Electronic Systems, Vol. 14, pp. 3:1--3:45.

21. *application independent modeling and simulation environment for systems with self-aware and self-expressive capabilities.* **Tatiana Djaba Nya, and Stephan C. Stilkerich.** 2014. ADAPTIVE. pp. 48-55.

22. *MARKS (Middleware Adaptability for Resource Discovery, Knowledge Usability and Self-healing) for Mobile Devices of Pervasive Computing Environments.* **Sharmin, M. and Ahmed, S. and Ahamed, S.I.** 2006. Third International Conference on Information Technology: New Generations, ITNG . pp. 306-313.

23. *A Fault Model Centered Modeling Framework for Self-healing Computing Systems.* **Lu, Wei and Zhu, Yian and Ma, Chunyan and Zhang, Longmei.** 2011. Artificial Intelligence and Computational Intelligence. pp. 61-68.

24. *Predictive dependency constraint directed self-healing for wireless sensor networks.* **Jingyuan Li and Yafeng Wu and Stankovic, J.A. and Son, S.H. and Ziguo Zhong and Tian He and Bong Wan Kim and Seong-Soon Joo.** 2010 . Seventh International Conference on Networked Sensing Systems (INSS), . pp. 22-29.

25. *Self-Stabilizing Leader Election in Optimal Space.* **Ajoy K. Datta, and Maria Gradinariu.** 2006. Internation symosium Stabilization, Safety, and Security of Distributed Systems.

26. *A Study on Integration Security Management Model in Cloud Environment.* **Kwak, Yun Sang Byun and Jin.** 12, 12 2013, Jouranl of Digital Convergence, Vol. 11.

27. **SQL, Green.** *Security and compliance solutions for AWS Databases.* s.l. : Database Security Solution Provider.

28. **paper, McAfee White.** *Database Security in Virtualization and Cloud Computing Environments.* March 2013.