ELSEVIER
Available online at www.sciencedirect.com
ScienceDirect
Procedia Computer Science 20 (2013) 235 – 241
Procedia Computer Science

Complex Adaptive Systems, Publication 3
Cihan H. Dagli, Editor in Chief
Conference Organized by Missouri University of Science and Technology
2013- Baltimore, MD

Autonomic Computing: A Framework to Identify Autonomy Requirements

Mona A. Yahya[a], Manal A. Yahya[a], Dr. Ajantha Dahanayake*

[a]*Prince Sultan University – College for Women, King Abdullah Road, Riyadh 11586 Saudi Arabia*

Abstract

Computing systems are ever growing in complexity. With that growth, the challenge of operating and maintaining them increased. In certain conditions, these systems may exist in harsh and distant environments making such operations even more difficult. To address the previous issues, the concept of autonomic computing originated. This concept, when applied fully will result in machines capable of evolving and managing themselves. This research aims to develop a framework for software engineers to apply autonomy in their Software Requirement Engineering phase by answering the question “What aspects affect the definition of autonomy requirements?”. The findings shall ease the understanding of the complex problem of capturing Adaptive requirements. This paper will present a proposed Requirements Engineering framework for Autonomic systems, in addition to some examples of systems applying autonomy.

Keywords: Autonomic Computing; Self-Management; Adaptive Systems Requirements; Requirement Engineering

1. Introduction

Autonomic Computing (AC) is a relatively new term coined by IBM in the year of 2001. It describes systems that are self-managing. The concept of autonomy was inspired by the autonomic nervous system that controls vital body functions without an individual’s knowledge. Autonomic computing systems incorporate four main features: Self-Configuration, Self-Healing, Self-Optimization, and Self-Protection [1], [2].

A Self-configured system configures itself according to the provided platform information to adopt the environmental change. The Self-healing feature allows the system to detect and diagnose abnormal behaviour and system failure then prepare an appropriate system repair accordingly. Moreover, self-optimization means that the system is able to optimize all available resources automatically without searching for extra resources. Finally, self-protection feature allows the system to protect itself from all unwanted behaviour and outside attacks [1], [2], [3].

* Corresponding author. Tel.: +966-11-494-8319.
E-mail address: ADahanayake@pscw.psu.edu.sa

Some agents or systems are used every day without the AC features in them being recognized. An example is plug and play feature which uses self configuration. Moreover, repair features of Microsoft Office are another AC application. The repair feature uses self-healing procedures to fix the system whenever a problem occurs [9].

In 2001, IBM built a self-managing computing system to overcome the rapidly growing software complexity problem. At 2005, IBM merged around 475 autonomic features into more than 75 products.

NASA is one of the leading organizations that build complex mission critical systems with autonomous behaviour. To them, Autonomy provides great benefit; it helps developing spacecraft systems that can explore regions of space where traditional crafts cannot explore. Some of the successful systems with autonomic features developed by NASA are Deep Space 1, Earth Observing 1 and Mars Exploration Rovers.

The main idea is to give computing systems the capability to manage themselves given high-level objectives from administrators [1]. These systems are most valuable in places that humans cannot reach due to distance or danger. The main question in this research is: "What aspects affect the definition of autonomy requirements?". From the complexity of such systems, we infer that it requires systematic collection of requirements on levels not considered in conventional requirements engineering practices. Our research is concerned with exploring the challenges associated with engineering the requirements of these systems; with a goal to provide software engineers with a roadmap to identify and manage these requirements.

The remaining of the paper is structured as follows: part two will provide a history and background of autonomic computing along with the related work done on adaptive systems. Part three describes our research methodology. Part four is about the Challenges of Requirements Engineering for Autonomic Systems. Part five is our proposed framework for Identifying Autonomy Requirements. Afterwards, we will discuss some future work and our findings in part six. Part seven will conclude the research paper.

2. Related Work

Capturing requirements of an adaptive system is difficult. Much research was conducted in this topic but there was no sufficient information on how to make a system autonomic. In [4], the authors suggested to use requirement's goal modelling as the basic step of developing autonomic systems and created an architecture for this system which helps to use this approach. The paper also highlighted three basic methods to make a system autonomic. V´ıtor E. Silva Souza in [5] looked at two different types of requirements – Awareness Requirements and Evolution Requirements – to help in defining a proper design to be implemented in adaptive systems. Then, the author proposed a goal-oriented modelling language to help analysts identify requirements of an adaptive system and a runtime framework to help developers in implementing those requirements.

In [6], the authors explored the uncertainty condition in Dynamically Adaptive Systems (DAS). They used the concept of threat modelling for the exploration of uncertainty environment that have great impact on DAS requirements. Moreover, they provided a process that handles the requirements of a DAS.

A general model for self adaptive systems was discussed in [7]. The paper was about producing a general model called SOTA "State Of The Affairs" to combine the goal oriented modelling with the context aware system modelling that identifies requirements for adaptive systems. Goal modelling helps in identifying the functional and non-functional requirements of an adaptive system.

Stakeholders usually tend to change their mind about system-to-be requirements. In [8], the authors argued about capturing stakeholder's system requirements, modelling and operationalizing them at runtime environment. They used goal modelling to identify system requirements.

More studies on requirement engineering for autonomic systems are described in table 1.

Table 1. Studies on Requirements Engineering for Autonomic /Adaptive Systems

Authors	Study Description	Tools, Languages, Approach	Findings
Qureshi and Perini [10]	Presented a scenario from the tourism domain to explain how to capture variability and flexibility in requirements	Use goal-models and ontologies to elicit requirements	Proposed a characterization of adaptive software. Variability and alternative software behavior is key to understanding adaptive requirements
Silva Souza, Lapouchnain, Robinson, Mylopoulos [11]	Attempted to formalize a new class of requirements to address adaptive requirements.	Goal Oriented RE (GORE). Event engineering & Analysis Toolkit (EEAT). Object Constraint Language (OCL)	Proposed a new type of requirements "Awareness Requirements". Described elicitation and formalization of such requirements.
Nauman A. Qureshi, Anna Perini, Neil A. Ernst, John Mylopoulos [12]	Authors presented a Travel Companion system scenario to explain their research as an approach to enable requirements-aware systems	Continuous adaptive requirements engineering approach (CARE) and RE language "Techne"	Proposed a goal-and user-oriented framework for building Self-Adaptive systems
N. Bencomo, J. Whittle, P.Sawyer, A. Finkelstein, E.Letier [13]	Authors studied the change of requirement during runtime	-	Defined a new concept "Requirements Reflection"
Alexei Lapouchnian, Sotirios Liaskos, John Mylopoulos, Yijun Yu [4]	Authors studied the requirements goal modeling and how to apply autonomy on those requirements	-	Proposed a way to use requirements goal modeling and gives a basic architecture that can be used to develop autonomic software
V´ıtor E. Silva Souza [5]	The author looked at two different types of requirements to help in defining a proper design to be implemented in adaptive systems	-	Proposed a design for adaptive system using goals requirements
Betty H.C. Cheng, Pete Sawyer, Nelly Bencomo, Jon Whittle [6]	The authors introduce a goal based modeling to develop requirements for dynamically adaptive systems	RELAX specification language	Uncertainty must be handled when developing for dynamically adaptive systems
Pete Sawyer, Nelly Bencomo, Jon Whittle, Emmanuel Letier, Anthony Finkelstein [14]	The authors studied the uncertainty in adaptive systems	-	Proposed to use analogous mechanisms to achieve requirements reflection
Paola Inverardi, Marco Mori [15]	The authors studied the foreseen and unforeseen requirement evolution during runtime	-	Proposed a framework to handle context-dependent requirements at runtime

3. Research Methodology

An exploratory research methodology is used to answer the question of this paper. The procedure undertaken involved as an initial phase: first, the collection of research papers discussing the idea of requirements engineering for adaptive and autonomic systems. Second, the study of certain applications with autonomic capabilities [2], and third, we reviewed some reports describing autonomic systems [16]. As a second phase a selection of papers in direct relation to our topic were chosen, and the provided information was analyzed to derive an understanding. The principles used in managing evolving requirements were identified. As a final phase we devised a framework that explains to software analysts the main aspects to consider for accommodating software autonomy in the requirements engineering phase.

4. Challenges of Requirements Engineering for Autonomic Systems

Meeting the vision of autonomic computing has placed many challenges on the software engineering practice. These challenges are manifested in all development stages: design, implementation, testing, and verification and validation. Autonomic computing design is a current research topic. According to [17], designing autonomic systems can be based on either hard computing or soft computing principals. A combination of these principles may provide the solution to the design problem of autonomic systems.

Our focus in this research is on the first stage of the software lifecycle "Requirements Engineering". Opposing conventional computing systems in which Software Requirements define specifically the actions captured from the users and the required processing and output, autonomic systems should be context aware allowing for monitoring the environment and adapting to the changes in it. That imposes a number of challenges on the Software engineer.

One of the main challenges is defining appropriate models to specify, understand, and implement autonomic behavior. Engineers must understand the role of high level goals and how they will be decided upon and achieved dynamically; they should also decide on the level of dynamism to incorporate into the software. High dynamism, in which systems evolve and change throughout their lifetime, may lead to unexpected behavior [18]. As presented in section 2, a number of research papers addressed the problem of autonomy requirements representation. Two main concepts were most intriguing, Awareness Requirements and Requirements Reflection; with the first defining a new type of requirements and the second proposing a way to analyze requirements at runtime. Autonomic systems require the dynamic selection of optimal solutions from a number of alternatives at runtime. This selection is guided by the occurring changes in the environment, an unattainable capability with static requirements [13]. Goal-oriented modelling languages such as KAOS and i* are thought to be promising in this field since they integrate certain aspects [13] allowing for automated reasoning about goals.

A second challenge recognized by the authors of [13] is the synchronization between requirements and architecture. Changing requirements at runtime might affect the architecture of the software. That effect must be managed carefully to ensure that the architectural components are changed smoothly so that no requirements or goals are broken during the process. Consequently, Autonomy specific architectures have been formed. IBM introduced what is now considered a prevalent architecture for autonomic systems [19], the MAPE architecture. This architecture is composed of four key components: Monitor, Analyze, Plan, and Execute. Although it is widespread, some researchers think that this architecture does not reflect autonomic systems completely [19]. Other architectures are proposed, however with no solid achievement, such as the Intelligent machine Design (IMD) Architecture [19].

A major challenge with autonomic computing is dealing with uncertainty. The constantly changing environment, and the efficiency of certain goals to attain the greater benefit impose a great challenge that must be well managed during all stages of life-cycle including runtime [13].

5. Framework for Identifying Autonomy Requirements

This section presents the proposed framework for identifying Autonomy Requirements. As discussed earlier, the focus of this paper is identifying Autonomy Requirements at the Requirements Engineering (RE) phase. The framework developed shall be applied during the RE phase; although separate from the elicitation, analysis, and validation of the base requirements.

6.1 Framework Elements

Following is a decomposition of the suggested framework. Inputs to the framework are the Requirements specification and System Goals as defined by the stakeholders.

- **System Environment:** It is essential to analyze the environment in which the system will live. This will assist in predicting the possible changes and required responses to be attained by the system.
- **System Capability:** By capabilities we mean what the system can do to interact with the environment in terms of physical connections and data interaction.
- **Level of Autonomy:** autonomic computing is considered progressive in nature. There are five levels of autonomy [17]: Basic, Managed, Predictive, Adaptive, and Autonomic. With the basic being the level in which most IT systems are in nowadays; and the Autonomic level being the highest level. Deciding the level of autonomy shall enable the engineers make correct choices on how the system in hand will be developed.
- **Choice of Technology:** a development technology might be dictated by the system requestors, or the domain in which the software will operate. However, developing for autonomy requires a different development environment that enables runtime adaptation. The XML-Based Autonomic Computing Expression Language (ACEL) and Software Component Ensemble Language (SCEL) operating on a Java Runtime Environment are two examples of technology supporting autonomic computing.

- **Runtime Requirement Assessment:** autonomic computing requires a mechanism to monitor and evaluate the success or failure of achieving the system goals at runtime. This mechanism could be based on the chosen technology or developed separately. Prior research has suggested a number of ways to this such as the definition of Awareness Requirements [11], and the concept of requirements reflection [13].
- **Decision-Making:** understanding the runtime behavior of requirements helps in deciding about the action to be taken by the system.
- **Goal Achievement Alternatives:** clarifying alternative ways to achieve goals is an essential part for developing autonomic requirements. The uncertainty associated with the environment in which autonomic systems exist, may hinder certain actions. However, that should not prevent the system from achieving the main goal.

Table 2. Autonomy Levels [17]

Level	Administrator Role	System Role
Basic	Set Up, Monitor, Enhance System	Execute Tasks
Managed	Analyze information, Make decisions, take action	Collect information in a consolidated view
Predictive	Make decisions, take action	Recognize patterns, provide advice on action
Adaptive	Observe, ensure following policies	Take the right action
Autonomic	Monitor business processes, alter objectives	Operation is governed by business policies

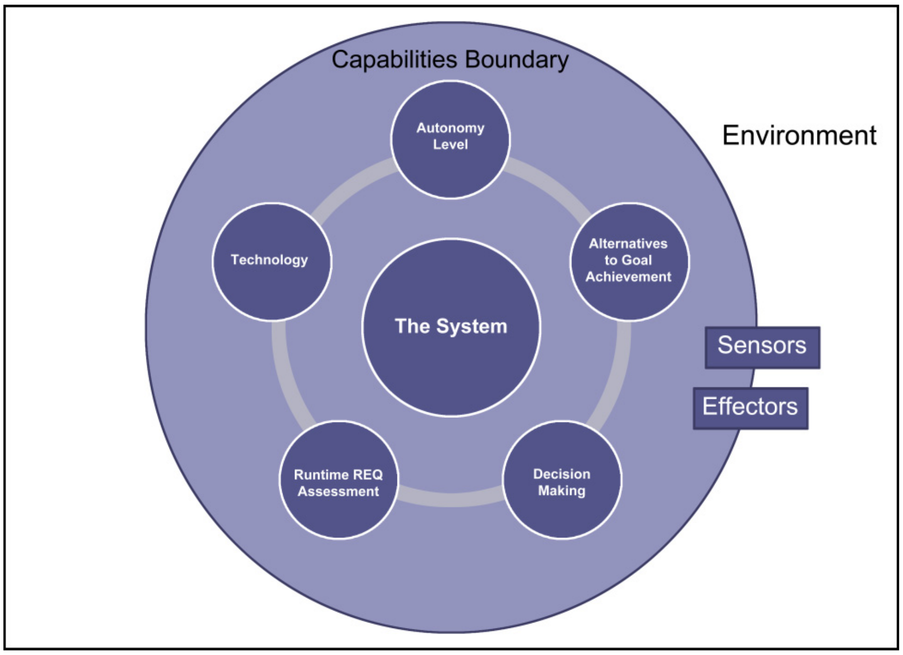

Fig 1. Depicts the requirements abstraction for autonomic systems.

Table 3. Framework to Identify Autonomic Requirements

Framework to Identify Autonomic Requirements			
Input	Requirements Specifications, System Goals	Output	Autonomy Requirements to include in the Requirements Document

Aspects to Consider	Description	Objective	Available Approaches
System Environment	Define the Environment in which the developed system will exist in.	Capture environment information	• Natural language representation of environment elements
System Capabilities	Define the systems connections and interaction with the environment.	Identify all connections, sensors and Effectors	• Analyze the base requirements of the system
Autonomy Level	Decide on the desired level of autonomy, to determine functionality.	Minimize cost, Increase efficiency and dependability	• Five levels of Autonomy [17]: Basic, Managed, Predictive, Adaptive, Autonomic • Eight levels of Autonomy Assessment Scale [20]
Technology Choice	Describes the development environment and programming language.	Familarity of technology or available training	• Autonomic Computing Expression Language (ACEL) • Software Component Ensemble Language (SCEL) • Agent-Oriented Programming Languages [21]
Runtime Requirement Assessment	Monitor and evaluate the success or failure of achieving the system goals at runtime.	Provide an acuurate measure of requirements success	• Awareness Requirements [11] • The concept of requirements reflection [13].
Decision Making	Describe a mechanism for task planning and decision making to achieve goals	Optimize decision making process	• Rule-Based approach • Control Theory Approach • Biology inspired processes
Alternatives to Goal Achievement	Describe possible alternatives to achieve goals	Provide more than one path to goals	• Further Analysis of the goals

6. Discussion

We started this research with an avid curiosity to understand the development process of autonomic systems. This process happened to be more complex than the development of conventional software. It required the achievement of high level goals taking into consideration the changing environment. Defining the requirements for autonomic systems requires identifying certain aspects surrounding and contributing to the system. It is not adequate to consider only the base requirements and goals. The identification of elements from our proposed framework shall support the system analyst further understand the problem and ease the definition of autonomy requirements. The Application of the framework is out of the scope of this paper; it will be considered in future work.

7. Conclusion

Autonomic computing is a promising approach to the development of computing systems that aim to minimize the effort done by IT personnel and reduce cost. At the present time, some computing systems are equipped with basic autonomy features. However, achieving full autonomy is still far from being accomplished. Many studies approached the problem of autonomic systems' RE from the representation point of view. However, none attempted to create a generic view of the elements surrounding such systems. In our work we explored the arena of autonomic computing in an attempt to create a framework for software engineers to understand the aspects affecting the RE process associated with developing such systems. The presented framework is based on prior research in the field; however with upcoming research other factors may emerge and require the expansion of the framework.

Future work will focus on the application of this framework on a real case of autonomic system development to evaluate its effectiveness. With practical examination, further enhancement to the framework may prove necessary.

Acknowledgement

This work was done as a required course for a Masters Degree in Software Engineering at Prince Sultan University. We would like to thank our families for their constant support and patience in our pursuit of knowledge.

References

1. J. O. Kephart and D. M. Chess, "The Vision of Autonomic Computing," IEEE Computer Society, January 2003.
2. S. S. Laster and A. O. Olatunji, "Autonomic Computing: Towards a Self-Healing System," in American Society for Engineering Education, Illinois, Indiana, 2007.
3. H. A. Müller, L. O'Brien, M. Klein and B. Wood, "Autonomic Computing," Software Architecture Technology, April 2006.
4. A. Lapouchnian, S. Liaskos, J. Mylopoulos and Y. Yu, "Towards Requirements-Driven Autonomic Systems Design," in DEAS 2005, St. Louis, Missouri, USA, May 21, 2005.
5. V. E. S. Souza, "A Requirements-Based Approach for the Design of Adaptive Systems," in ICSE, Zurich, Switzerland, 2012.
6. B. H. Cheng, P. Sawyer, N. Bencomo and J. Whittle, "A Goal-Based Modeling Approach to Develop Requirements of an Adaptive System with Environmental Uncertainty," in DEAS 2005, East Lansing, Michigan 48824, USA, 2005.
7. D. B. Abeywickrama, N. Bicocchi and F. Zambonelli, "SOTA: Towards a General Model for Self-Adaptive Systems," in Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE), 2012 IEEE 21st International Workshop on, Toulouse, 2012.
8. V. E. S. Souza, A. Lapouchnian and J. Mylopoulos, "(Requirement) Evolution Requirements for Adaptive Systems," in Software Engineering for Adaptive and Self-Managing Systems (SEAMS), 2012 ICSE Workshop, Zurich, 2012.
9. "Chapter 1," in Autonomic & Grid Computing: What, Why, and How?, Retrieved April, 2013, pp. 1 - 35.
10. A. P. Nauman A. Qureshi, "Engineering adaptive requirements," in SEAMS '09 Proceedings of the 2009 ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems, Vancouver, 2009.
11. A. L. W. N. R. Vítor E. Silva Souza, "Awareness Requirements for Adaptive Systems," in SEAMS '11 Proceedings of the 6th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, Waikiki, Honolulu, 2011.

12.4 N. A. Qureshi, A. Perini, N. A. Ernst and J. Mylopoulos, "Towards a Continuous Requirements Engineering Framework for Self-Adaptive Systems," in First International Workshop on IEEE, Sydney, NSW, Sept 2010.

13. J. W. P. S. A. F. E. L. Nelly Bencomo, "Requirements Reflection: Requirements as Runtime Entities," in ICSE '10 Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering, New Your, 2010.
14. P. Sawyer, N. Bencomo, J. Whittle, E. Letier and A. Finkelstein, "Requirements-Aware Systems A research agenda for RE for self-adaptive systems," in 2010 18th IEEE International Requirements Engineering Conference, 2010.
15. P. Inverardi and M. Mori, "Requirements Models at Run-time to Support Consistent System Evolutions," in 2nd International Workshop, Trento, Aug. 2011.
16. G. Rabideau, D. Tran, S. Chien, B. Cichy, R. Sherwood, D. Mandl, S. Frye, S. Shulman, J. Szwaczkowski, D. Boyer and J. V. Gaasbeck, "Mission Operations of Earth Observing-1 with Onboard Autonomy," in Space Mission Challenges for Information Technology, 2006. SMC-IT 2006. Second IEEE International Conference, Pasadena, CA, Retrieved Feb, 2013.
17. A. S. Sandeep Kumar Chauhan, "Autonomic Computing: A Long Term Vision In Computing," Journal of Global Research in Computer Science, vol. 3, no. 5, pp. 65-67, 2012.
18. G. J.C., v. d. Hoek and A. T. R.N., "Architectural Runtime Configuration Management in Support of Dependable Self-Adaptive Software" In: Workshop on Architecting Dependable Systems. St. Louis (MO), USA, 2005.
19. R. J. A. M. P. Haffiz Shuaib, "A Framework for Certifying Autonomic Computing Systems," in ICAS 2011: The Seventh International Conference on Autonomic and Autonomous Systems, Venice/Mestre, 2011.
20. R. W. Proud, J. J. H. and R. B. Mrozinski, "Methods for Determining the Level of Autonomy to Design into a Human Spaceflight Vehicle: A Function Specific Approach," NASA Johnson Space Center, Houston, Texas, 2010.
21. C. Bădică, Z. Budimac, H.-D. Burkhard and M. Ivanović, "Software Agents: languages, tools, platforms," Advances in Formal Languages, Modeling and Applications, vol. 8, no. 2, May 2011.