SPAACE :: Self- Properties for an Autonomous & Autonomic Computing Environment

SPAACE :: Self- Properties for an Autonomous & Autonomic Computing Environment

Roy Sterritt
School of Computing and Mathematics,
Faculty of Engineering
University of Ulster
Northern Ireland
r.sterritt@ulster.ac.uk

Mike Hinchey
NASA Goddard Space Flight Center
Software Engineering Laboratory
Greenbelt, MD 20771
USA
michael.g.hinchey@nasa.gov

Abstract

Self- properties, or "selfware", refers to current and emerging behaviours exhibited by systems that are considered to be "autonomic" or inspired by another view of self-management. We describe some emerging properties, which may range from self-adjusting to self-destruction. We describe the architecture required to create self-ware and discuss the relationship between autonomicity and autonomicity.*

Keywords: Self-Managing Systems, Selfware, Self-*, Autonomic Computing, Autonomic Systems, Autonomy.

1. Introduction

Various initiatives related to the development of self-managing systems have been proposed as means of addressing issues in the development of complex computer-based systems. These initiatives include: Autonomic Computing (IBM), Adaptive Infrastructure (HP), N1 (Sun), Dynamic Systems Initiative (Microsoft), Adaptive Network Care (Cisco), Proactive Computing (Intel), Organic Computing (Fujitsu). A major focus has been on biologically-inspired approaches, taking inspiration from the human body and from nature. At the heart of this vision of self-managing systems is *selfware*, incorporating such self-* properties as self-configuring, self-healing, self-protecting and self-optimizing. Achievement of this "selfware" is dependant on achieving system self-awareness and environmental awareness, implemented by means of a feedback control loop consisting of sensors and effectors within the computer system to provide the self-monitoring and self-adjusting properties [1].

We consider these properties, the architecture of self-managing systems, and the relationship between autonomicity (self-management) and autonomy (self-government).

2. Selfware: Self-managing Software / Hardware /Firmware & Communications

IBM, upon launching their initiative, identified the computing industry's main concerns as system complexity and total cost of ownership (TCO). Their solution, *viz.* Autonomic Computing, was described as compromising of eight elements [2];

- Possess system identity - detailed knowledge of components
- Self configure & re-configure - adaptive algorithms
- Optimize operations - adaptive algorithms
- Recover - no impact on data or delay on processing
- Self protection
- Aware of environment and adapt
- Function in a heterogeneous world
- Hide complexity

These eight elements can be expressed in terms of properties that a system should possess in order to constitute autonomicity [3]. These are described in Section 2.1 and elaborated upon in Section 2.2, which discusses the very constructs that constitute these properties.

2.1 Self-* and Autonomic Properties

The properties that a system should possess in order to constitute an autonomic system are depicted in Figure 1 [2],[3],[4].

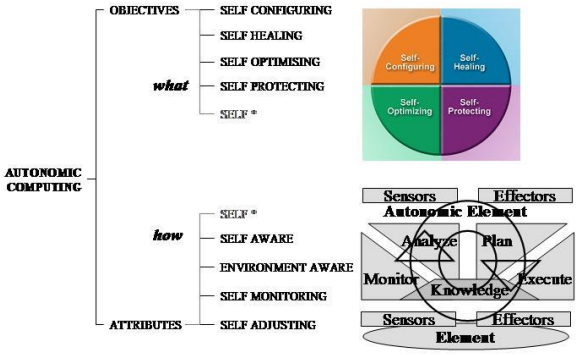

Figure 1 Autonomic Computing Properties

The general properties of an autonomic (self-managing) system can be summarised by four objectives: being self-configuring, self-healing, self-optimizing and self-protecting, and four attributes: self-awareness, environment-awareness, self-monitoring and self-adjusting (Figure 1). Essentially, the objectives represent broad system requirements, while the attributes identify basic implementation mechanisms. Since the 2001 launch of the Autonomic Computing initiative, the self-* list of properties has grown substantially [5],[6] yet this initial set still represents the general goal.

Self-configuring represents a system's ability to re-adjust itself automatically; this may simply be in support of changing circumstances, or to assist in self-healing, self-optimization or self-protection. Self-healing, in reactive mode, is a mechanism concerned with ensuring effective recovery when a fault occurs, identifying the fault, and then, where possible, repairing it. In proactive mode, it monitors vital signs in an attempt to predict and avoid "health" problems (reaching undesirable situations). Self-optimization means that a system is aware of its ideal performance, can measure its current performance against that ideal, and has defined policies for attempting improvements. It may also react to policy changes within the system as indicated by the users. A self-protecting system will defend itself from accidental or malicious external attack. This necessitates awareness of potential threats and a means of handling those threats (Figure 1) [3].

In achieving such self-managing objectives (Figure 1) a system must be aware of its internal state (self-aware) and current external operating conditions (environment-aware). Changing circumstances are detected through self-monitoring and adaptations are made accordingly (self-adjusting) [3]. As such, a system must have knowledge of its available resources, its components, their desired performance characteristics, their current status, and the status of inter-connections with other systems, along with rules and policies of how these may be adjusted. Such ability to operate in a heterogeneous environment will require the use of open standards to enable global understanding and communication with other systems [2].

These mechanisms are not independent entities. For instance, if an attack is successful, this will include self-healing actions, and a mix of self-configuration and self-optimisation, in the first instance to ensure dependability and continued operation of the system, and later to increase the self-protection against similar future attacks. Finally, these self-mechanisms should ensure there is minimal disruption to users, avoiding significant delays in processing.

Other self-* properties have emerged or have been revisited in the context of autonomicity. We highlight some of these briefly here.

self-*

Self-managing properties.

self-anticipating

The ability to predict likely outcomes or simulate self-* actions.

self-assembling

Assembly of models, algorithms, agents, robots, etc.; self-assembly is often influenced by nature, such as nest construction in social insects. Also referred to as self-reconfigurable systems.

self-awareness

"Know thy self"; awareness of internal state; knowledge of past states and operating abilities.

self-chop

The initial four (and generic) self-properties (Self-Configuration, Self-Healing, Self-Optimisation and Self-Protection).

self-configuring

The ability to configure and re-configure in order to meet policies/goals.

self-critical

The ability to consider if policies are being met or goals are being achieved (alternatively, self-reflect)

self-defining

In reference to autonomic event messages between Autonomic Managers: contains data and definition of that data–metadata (for instance using XML).

In reference to goals/policies: defining these (from self-reflection, etc.).

self-governing

As in autonomous: responsibility for achieving goals/tasks.

self-healing

Reactive (self-repair of faults) and Proactive (predicting and preventing faults).

self-installing

As in a specialized form of self-configuration – installing patches, new components, etc or re-installation of OS after major crash.

self-managing

Autonomous, along with responsibility for wider self-* management issues.

self-optimizing

Optimization of tasks and nodes.

self-organized

Organization of effort/nodes. Particularly used in networks/communications.

self-protecting

The ability of a system to protect itself.

self-reflecting

The ability to consider if routine and reflex operations of self-* operations are as expected. May involve self-simulation to test scenarios.

self-similar

Self-managing components created from similar components that adapt to a specific task, for instance a self-managing agent.

self-simulation

The ability to generate and test scenarios, without affecting the live system.

selfware

Self-managing software, firmware and hardware.

2.2 Autonomic Element

Figure 2 represents a view of an architecture for an autonomic element that consists of the component required to be managed, and the autonomic manager [6].

It is assumed that an autonomic manager (AM) is responsible for a managed component (MC) within a self-contained autonomic element (AE). This autonomic manager may be designed as part of the component or provided externally to the component, as an agent, for instance. Interaction will occur with remote autonomic managers (cf. the autonomic communications channel shown in Figure 2) through virtual, peer-to-peer, client-server or grid configurations.

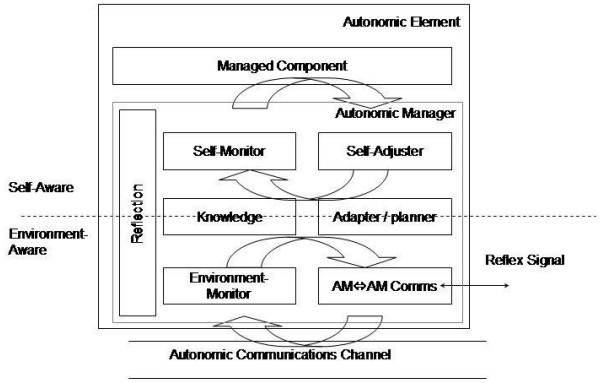

Figure 2 An Autonomic Element, including reflection and reflex layers.

At the heart of the architecture of any autonomic system are sensors and effectors. A control loop is created by monitoring behavior through sensors, comparing this with expectations (knowledge, as in historical and current data, rules and beliefs), planning what action is necessary (if any), and then executing that action through effectors. The closed loop of feedback control provides the basic backbone structure for each system component [7]. Figure 2 highlights that there are at least two control loops in an Autonomic Element – one for self-awareness and another for environmental awareness.

IBM represents this self-monitor/self-adjuster control loop as the monitor, analyze, plan and execute (MAPE) control loop (Figure 1). The monitor-and-analyze parts of the structure process information from the sensors to provide both self-awareness and an awareness of the external environment. The plan-and-execute parts decide on the necessary self-management behavior that will be executed through the effectors. The MAPE components use the correlations, rules, beliefs, expectations, histories, and other information known to the autonomic element, or available to it through the knowledge repository within the AM [7].

2.3 Reflex Signal – Lub-Dub Pulse Emission

The autonomic environment requires that autonomic elements and, in particular, autonomic managers communicate with one another concerning self-* activities, in order to ensure the robustness of the environment. Figure 2 depicts that the autonomic manager communications (AM⇔AM) also includes a reflex signal. This may be facilitated through the additional concept of a pulse monitor—PBM (an extension of the embedded system's heart-beat monitor, or HBM, which safeguards vital processes through the emission of a regular "I am alive" signal to another process) with the capability to encode health and urgency signals as a pulse [8]. Together with the standard event messages on the autonomic communications channel, this provides dynamics within autonomic responses and multiple loops of control, such as reflex reactions among the autonomic managers [9].

This reflex component may be used to safe-guard the autonomic element by communicating its health to another AE [10]. The component may also be utilized to communicate environmental health information [11]. For instance, in the situation where each PC in a LAN is equipped with an autonomic manager, rather than each of the individual PCs monitoring the same environment, a few PCs (likely the least busy machines) may take on this role and alert the others through a change in pulse to indicate changing circumstances.

An important aspect concerning the reflex reaction and the pulse monitor is the minimization of data sent – essentially only a "signal" is transmitted. Strictly speaking, this is not mandatory; more information may be sent, yet the additional information must not compromise the reflex reaction. For instance, in the absence of bandwidth concerns, information that can be acted upon quickly and not incur processing delays could be sent. The important aspect is that the information must be in a form that can be acted upon immediately and not involve processing delays (such as is the case of event correlation).

Just as the beat of the heart has a double beat (lub-dub) the autonomic element's (Figure 2) pulse monitor may have a double beat encoded – as described above, a *self* health/urgency measure and an *environment* health/urgency measure. These match directly with the two control loops within the AE, and the self-awareness and environment awareness properties.

2.4 Reflection

Reflection techniques allow the system to perform analysis computation on itself [12] (cf. the reflection component within the autonomic manager shown in Figure 2). In terms of an autonomic system, this is particularly relevant in order to allow the system to consider the self-managing policies, and to ensure that they are being performed as expected. This is acutely key since autonomicity involves self-adaptation to the changing circumstances in the environment.

2.5 Autonomy and Autonomicity at the System level

A high level perspective for an intelligent machine design is depicted in Figure 3 (adapted from [13], [8]). It describes three levels for the design of intelligent systems:

1. Reaction—lowest level, where no learning occurs but there is immediate response to state information coming from sensory systems.
2. Routine—middle level, where largely routine evaluation and planning behaviors take place. Input is received from sensors as well as from the reaction level and reflection level. This level of assessment results in three dimensions of affect and emotion values: positive affect, negative affect, and (energetic) arousal.
3. Reflection—top level, receives no sensory input or has no motor output; input is received from below. Reflection is a meta-process, whereby the mind deliberates about itself. Essentially, operations at this level look at the system's representations of its experiences, its current behavior, its current environment, etc.

Input from, and output to, the environment only takes place within the reflex and routine layers. One may consider that reaction level essentially sits within the "hard" engineering domain, monitoring the current state of both the machine and its environment, with rapid reaction to changing circumstances; and, that the reflection level may reside within the AI domain utilizing its techniques to consider the behavior of the

system and learn new strategies. The routine level may be a cooperative mixture of both (Figure 3).

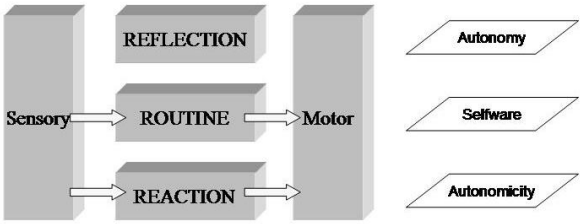

Figure 3 Comparing intelligent machine design and system level autonomy and autonomicity

This high-level intelligent machine design is appropriate for autonomic systems as depicted here since the case has been made for the dynamics of responses including reflex reactions and also for reflection of the self-managing behavior.

Some researchers hold the perception that autonomic computing resides solely within the domain of the reaction layer. This is understandable due to the metaphoric link with the autonomic nervous system, where no conscious or cognitive activity takes place. These researchers would point to other biologically-inspired computing (also referred to as nature-inspired computing, organic computing, etc.) as providing such higher level cognitive approaches for instance as in swarm intelligence. Within the autonomic computing research community, autonomicity is not normally considered to imply this narrower view. Essentially, the autonomic self-managing metaphor is considered to aim for a user/manager to be able to set high-level policies, while the system achieves the goals. Similar overarching views exist in other related initiatives and, increasingly, they are influencing each other.

In terms of autonomy and autonomicity, autonomy may be considered as being self-governing while autonomicity is considered being self-managing. At the element level, an element will have some autonomy and autonomic properties, since to self-manage implies some autonomy, while to provide a dependable autonomous element requires such autonomic properties as self-healing along with the element's self-directed task. From this perspective, it would appear that the separation of autonomy and autonomicity as characteristics will decrease in the future and eventually will become negligible. On the other hand, at the system level if one considers again the three tiers of the intelligent machine design (reaction, routine, and reflection) and accepts the narrower view of autonomicity, there is a potential correlation between the levels. That is, the reaction level correlates with autonomicity, and the reflection level with autonomy, as in self-governing of the self-managing policies within the system. In the end, different classifications or different perspectives on the matter will be academic unless they assist and inspire new means to achieve the self-managing vision.

3. Conclusion

We have discussed emerging self-* properties, self-managing elements incorporating reflex and reaction layers, along with the emerging standard components of self- and environmental control loops, and the necessary components to provide the self-monitoring and self-adjusting properties.

We have summarized how the reflex reaction component – the pulse monitor – may be used to encode and transmit health/urgency signals of the element (self) or the environment. We propose that like the heart with its double beat (lub-dub) that the self and environmental values may be transmitted together.

The relationship of Autonomous & Autonomic computing, framed in the context of an intelligent machine design architecture with reaction, routine, and reflection layers was also discussed

Self-managing systems, whether viewed from the autonomic computing perspective, or from the perspective of another initiative, offers a holistic vision for the development and evolution of computer-based systems that aims to bring new levels of automation and dependability to systems, while simultaneously hiding their complexity and reducing their total cost of ownership.

Acknowledgements

This research is partly supported at University of Ulster by the Computer Science Research Institute and the Centre for Software Process Technologies (CSPT) which is funded by Invest NI through the Centres of Excellence Programme, under the EU Peace II initiative.

Part of this work has been supported by the NASA Office of Systems and Mission Assurance (OSMA) through its Software Assurance Research Program (SARP) project, Formal Approaches to Swarm Technologies (FAST), and by NASA Goddard Space Flight Center, Software Engineering Laboratory (Code 581).

References

[1] Sterritt, R., Towards Autonomic Computing: Effective Event Management, *Proceedings of 27th Annual IEEE/NASA Software Engineering Workshop (SEW)*, Maryland, USA, December 3-5, IEEE Computer Society, pp 40-47.

[2] Horn, P., Autonomic computing: IBM perspective on the state of information technology, IBM T.J. Watson Labs, NY, 15 October 2001. Presented at AGENDA 2001, Scottsdale, AZ (available at http://www.research.ibm.com/autonomic/), 2001

[3] Sterritt, R. and Bustard, D.W., Autonomic Computing: a Means of Achieving Dependability? *Proceedings of 10th IEEE International Conference on the Engineering of Computer Based Systems (ECBS '03)*, Huntsville, Alabama, USA, April 7-11, IEEE CS Press, pp 247-251.

[4] IBM. An Architectural Blueprint for Autonomic Computing.

[5] Tianfield, H., Multi-Agent Based Autonomic Architecture for Network Management, *Proceedings of INDIN 2003, IEEE International Conference on Industrial Informatics*, 21-24 August 2003, pp 462-469.

[6] Sterritt, R., Autonomic Computing, *Innovations in Systems and Software Engineering: a NASA Journal*, Springer, 1(1), April 2005.

[7] Ganek, A.G., and Corbi, T.A., The Dawning of the Autonomic Computing Era, *IBM Systems Journal*, 42(1):5–18.

[8] Sterritt, R., Pulse Monitoring: Extending the Health-check for the Autonomic GRID. *Proceedings of IEEE Workshop on Autonomic Computing Principles and Architectures (AUCOPA 2003) at INDIN 2003*, Banff, Alberta, Canada, 22–23 August, pp 433–440.

[9] Sterritt, R., Gunning, D., Meban, A., and Henning, P., Exploring Autonomic Options in a Unified Fault Management Architecture through Reflex Reactions via Pulse Monitoring. *Proceedings of IEEE Workshop on the Engineering of Autonomic Systems (EASe 2004) at the 11th Annual IEEE International Conference and Workshops on the Engineering of Computer-Based Systems (ECBS 2004)*, Brno, Czech Republic, 24–27 May, pp 449–455.

[10] Sterritt, R. and Chung, S., Personal Autonomic Computing Self-Healing Tool, *Proceedings of IEEE Workshop on the Engineering of Autonomic Systems (EASe 2004) at 11th Annual IEEE International Conference and Workshop on the Engineering of Computer Based Systems (ECBS 2004)*, Brno, Czech Republic, 24-27 May, pp 513-520.

[11] Sterritt, R. and Bantz, D.F., PAC-MEN: Personal Autonomic Computing Monitoring Environments, *Proceedings of IEEE DEXA 2004 Workshops - 2nd International Workshop on Self-Adaptive and Autonomic Computing Systems (SAACS 04)*, Zaragoza, Spain, August 30th - September 3rd, pp 737-741.

[12] Maes, P., Concepts and Experiments in Computational Reflection, *Proceedings of the International Conference on Object-Oriented Programming Systems, Languages and Applications*, 1987, pp 147-155.

[13] Norman, D.A., Ortony, A. and Russell, D.M., Affect and Machine Design: Lessons for the Development of Autonomous Machines, *IBM Systems Journal*, 42(1):38–44.