JOURNAL OF AEROSPACE COMPUTING, INFORMATION, AND COMMUNICATION
Vol. 8, April 2011

Next Generation Autonomic System Environment

Roy Sterritt*
University of Ulster, Newtonabbey, Northern Ireland BT37 0QB, United Kingdom

DOI: 10.2514/1.54467

Next generation computer-based environments are ever increasingly pervasive and ubiquitous. Such interconnected device-based environments necessitate little or no human involvement in the management loop. Self-* properties, or "selfware," refers to current and emerging behaviors exhibited by systems that are considered to be "autonomic" or inspired by another view of self-management. We describe some emerging properties, which range from self-adjusting to self-destruction. We describe the architecture required to create self-ware and discuss the relationship between autonomicity and autonomy, and document some additional novel autonomic constructs inspired from biology for the next generation self-managing environments.

I. 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), Autonomic Networking (Motorola), and 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-optimizing, and self-protecting (often referred in shorthand as self-CHOP). Achievement of this "selfware" is dependant on achieving system self-awareness and environmental awareness (self-situation), 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].

II. Selfware: Self-managing Software/Hardware/Firmware and 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 comprising of eight elements [2]:

- Possess system identity—detailed knowledge of components
- Self-configure and 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

Received 21 July 2009; accepted for publication 4 April 2011.
* School of Computing and Mathematics, Faculty of Computing and Engineering, University of Ulster, UK. r.sterritt@ulster.ac.uk.

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 Sec. II.A and elaborated upon in Sec. II.B, which discusses the very constructs that constitute these properties.

A. A Self-* and Autonomic Properties

The properties that a system should possess to constitute an autonomic system are depicted in Fig. 1 [2–5].

The general properties of an autonomic (self-managing) system can be summarized by four objectives: being self-configuring, self-healing, self-optimizing, and self-protecting; and four attributes: self-awareness, environment-awareness (self-situation), self-monitoring, and self-adjusting (Fig. 1). Essentially, the objectives represent broad system requirements, whereas the attributes identify basic implementation mechanisms. Since the 2001 launch of the Autonomic Computing initiative, the self-* list of properties has grown substantially [6–8], 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 a 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 (Fig. 1) [3].

In achieving such self-managing objectives (Fig. 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 mixture of self-configuration and self-optimization, in the first instance to ensure dependability and

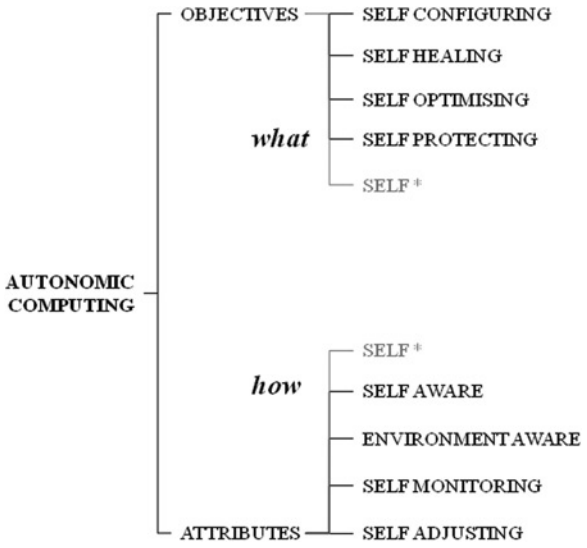

Fig. 1 Autonomic computing properties with associated AE.

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, in some domains understood to indicate emerging behavior.
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-optimization 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 (AMs): 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.

B. Autonomic Element

Figure 2 represents a view of an architecture for an autonomic element (AE) that consists of the component required to be managed, and the AM [7].

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

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 [9]. Figure 2 highlights that there are at two conceptual control loops in an AE—one for self-awareness and another for environmental awareness (self-situation).

IBM represents this self-monitor/self-adjuster control loop as the monitor, analyze, plan, and execute (MAPE) control loop (Fig. 3) or MAPE-K (inc Knowledge). 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

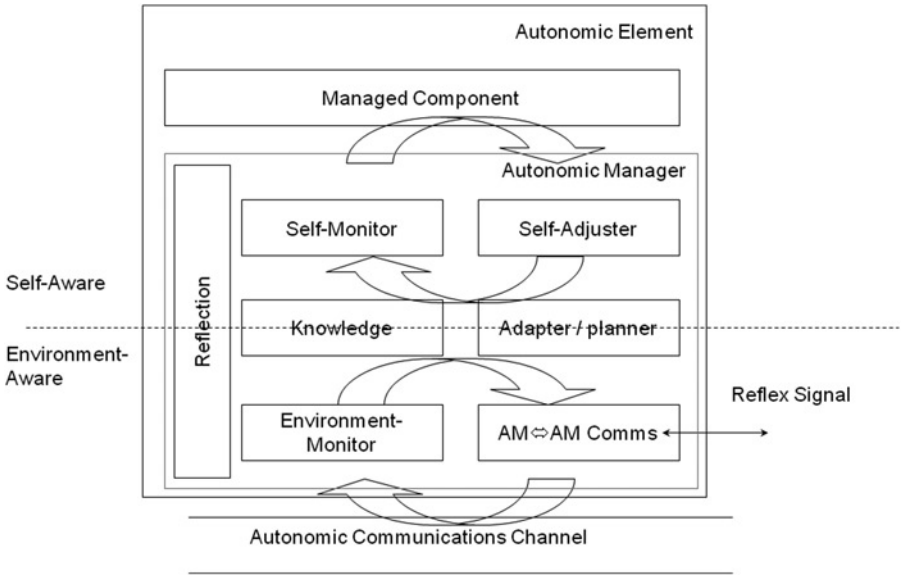

Fig. 2 An AE, including reflection and reflex layers.

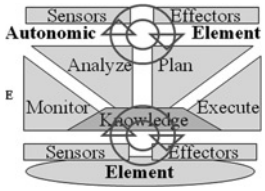

Fig. 3 IBM view of an AE—the "MAPE."

components use the correlations, rules, beliefs, expectations, histories, and other information known to the AE, or available to it through the knowledge repository within the AM [9].

Figure 2 represents an AE mainly in agreement with this MAPE view, with exception in terms of the external awareness control loop. This loop, in terms of sensing and effecting, can in general only effect through communication (reflex signals or standard communications) that it has become aware of a situation requiring management in the environment. This as in good software engineering practice is effectively designing in scoping rules. The AE can in general only directly effect change in its MC. Logically, the MAPE implies that it can change the environment. By implication, then because this is a distributed collaborative self-managing environment many individual MAPEs in real time will spot the issue and effect change creating undesirable states.

As such the scoping rule is critical. That said, an AE may send a mobile agent (or even RPC) instead of the reflex signal or event message to effect change, because this is a distributed environment and its scope may be wider than its local MC, for instance, to implement a policy change or update passwords throughout the environment. Yet logically this is still within its scope (distributed remit) and the receiving AE and environment still have final say as to if these are implemented, i.e., passing security and "still in context" credentials before execution. As such, these are in effect still self-managing communications but that also include the code "how to."

C. Reflex Signal

The autonomic environment requires that AEs and, in particular, AMs communicate with one another concerning self-* activities, in order to ensure the robustness of the environment. Figure 2 depicts that the AM communications (AM ⇔ AM) also includes a reflex signal. This indicates that certain situations may be too critical to relay on standard event messages (that compete with the many other self-managing event messages residing on the system

for correlation and action). It is also inspired by the biological autonomic nervous system (ANS), where the system is subdivided into the sympathetic (SyNS) and parasympathetic (PaNS) nervous systems and tends to have opposite effects: parasympathetic slows the heart rate whereas the sympathetic speeds it up. The sympathetic nervous activity increases in response to fear, i.e., the "fight or flight" response, whereas the parasympathetic nervous activity acts to calm, the "rest and digest" response. This biological sympathetic autonomicity may be used to inspire similar mechanisms in the system-to create a reflex "flight or fight" self-configuration response when necessary. One approach is discussed further in Sec. III.

D. Reflection

Although reflection is a slower response control loop, its inspiration to parasympathetic perhaps ends there. Reflection technique is to allow the system to perform analysis computation on itself [10] (cf. the reflection component within the AM shown in Fig. 2). In terms of an autonomic system, this is particularly relevant in order to allow the system to consider the self-managing policies over time, and to ensure that they are being performed as expected. This is acutely key because autonomicity involves self-adaptation to the changing circumstances in the environment as well as to ensure the self-configuration, self-healing, self-optimizing and self-protecting are not just flip-flopping over time between states.

E. Autonomy and Autonomicity at the System Level

These individual AEs collaborate together to create a self-managed environment, yet the collaboration approach is often debatable and application specific: client-server, peer-to-peer, agent-based, SWARM, and so forth. We believe the intent of the paradigm is at least peer-to-peer (localized handing of self-management issues instead of passing them on to an element controller in a client-server environment) although often client-server self-management remains as the application domain is such. Another debate is what constitutes autonomicity (self-management) and autonomy. The following depicts a three-tier architecture we have found useful with clients when designing such systems.

A high-level perspective for an intelligent machine design is depicted in Fig. 4 (adapted from [11,12]). 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 (Fig. 4).

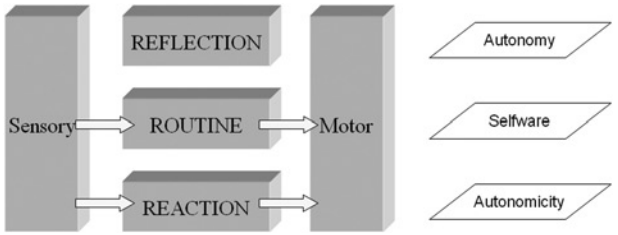

Fig. 4 Comparing intelligent machine design and system level autonomy and autonomicity.

This high-level intelligent machine design is appropriate for autonomic systems as depicted here because 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 ANS, where no conscious or cognitive activity takes place. These researchers would point to other biologically inspired computing (also referred to as nature-inspired 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, whereas 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 whereas 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 except where it is useful to design and develop them as separate abstractions—the task to be done (autonomy) and the capability to make sure it will happen (autonomicity).

At the system level if one considers again the three tiers of the intelligent machine design (reaction, routine, and reflection) and accepts a 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. The routine level will be a mixture of self-direction (actual application goals) and self-managing (dependability and survivability goals). 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.

III. The Autonomic Environment and Extending the Self-* Properties

As has been highlighted, Autonomic Computing is a metaphor based on the biological ANS, taking the ANS as inspiration to achieve self-managing computer-based systems without "conscious effort" from the user. Biology can also inspire other constructs and properties for self-management to create a functional rich next generation management system, some of these novel constructs are examined in this section. Section II highlighted the properties and elements within an AE. For this self-managing environment to work, these AEs must communicate and cooperate to ensure survivability and dependability as individual components will fail. Figure 5 depicts such an environment. Although there is no requirement to insist an agent-based approach is utilized the need for cooperation and autonomy makes a strong argument. With this view the AMs may be considered stationary agents and in addition to self-managing messaging between the AMs (S* in Fig. 5) on the autonomic communications channel there may be self-managing/autonomic mobile agents (☺ in Fig. 5) traveling between the AEs to facilitate the tasks of self-CHOP.

A. Reflex Signal: Pulse Monitoring and Lub-Dub Pulse Emission

Section II.C highlighted that the autonomic environment requires that AMs communicate with one another concerning self-* activities (Fig. 2) depicting that AM ⇔ AM comms also includes a reflex signal. This reflex signal may be implemented through an extension of heart-beat monitoring.

Heat-beat monitoring originated from embedded systems, where instead of sending a reactive message once a fault has occurred (which is typical in computer-based systems space) a message is sent when everything is okay, i.e., "I am alive" signal from a chip (aka the heart-beat). The absence of this signal allows more proactive actions to be taken. As the AMs and MCs are the key elements within the environment the actual manager's (AM) internal critical processes and MC itself may be protected with heart-beat monitors to detect immediate failure and initiate failover or restart (Fig. 6).

By extension then the complete AM should be protected by an HBM emitting to a neighboring AM (peer-to-peer) or north-bound (if in a hierarchy), so that if the AM fails self-managing responses can be induced including failover to another AM.

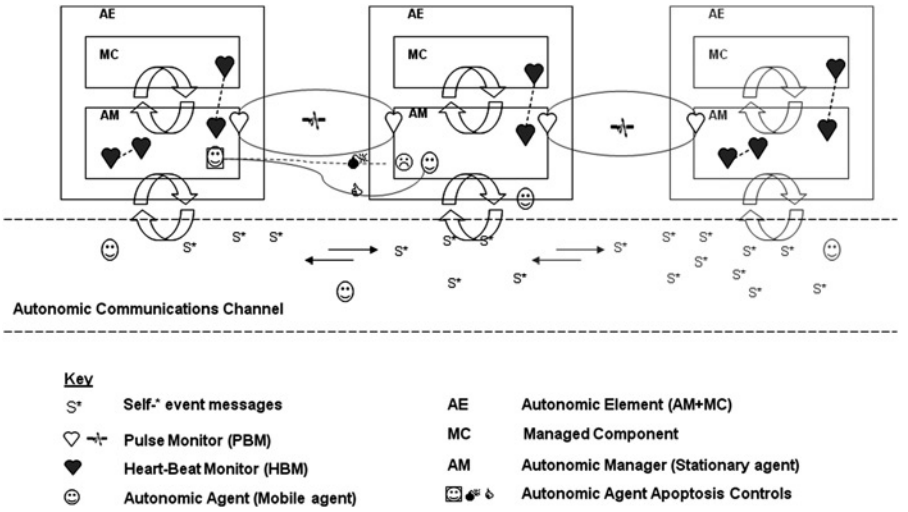

Fig. 5 A next generation autonomic environment.

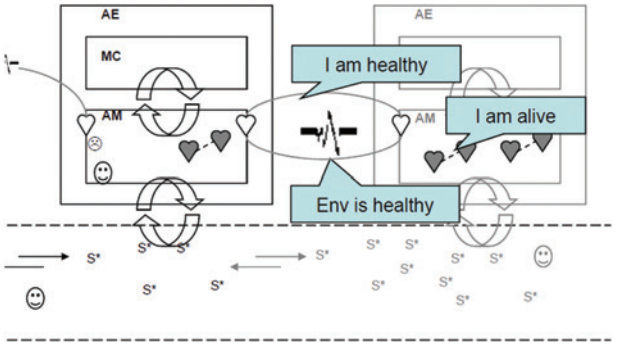

Fig. 6 An Autonomic Environment, including HBM and PBM Lub-Dub.

Returning to our requirement for a reflex signal—because part of the AMs activity is to monitor their MC and the environment they are able to assess the health of both and evaluate if an urgent situation is arising. As such, the reflex reaction may be facilitated through modifying the heart-beat monitor to the novel pulse monitor (PBM) which has the capability to encode health and urgency signals as a pulse [11] (i.e., from the regular emission of an "I am alive" to "I am healthy" signal). 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 AMs [13] (Fig. 6).

This reflex component may be used to safe-guard the AE by communicating its health to another AE [14]. The component may also be utilized to communicate environment health information [15]. For instance, in the situation where each PC in an LAN is equipped with an AM, 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 in the environment.

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 biological heart has a double beat (lub-dub) the AE's (Fig. 6) 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 conceptual control loops within the AE, and the self-awareness and environment awareness properties [16].

B. Autonomic Apoptosis (Self-Destruct)

It is believed that a biological cell knows when to commit suicide because cells are programmed to do so—self-destruct (sD) is an intrinsic property. This self-destruction is delayed due to the continuous receipt of biochemical reprieves. This process is referred to as *apoptosis* [17], meaning "drop out", the process has also been nicknamed "death by default" [18], where cells are prevented from putting an end to themselves due to constant receipt of biochemical "stay alive" signals.

Further investigations into the apoptosis process [19] have discovered more details about the sD predisposition. Whenever a cell divides, it simultaneously receives orders to kill itself. Without a reprieve signal, the cell does indeed sD. It is believed that the reason for this is self-protection, as the most dangerous time for the body is when a cell divides, as if just one of the billions of cells locks into division the result is a tumor, while simultaneously a cell *must* divide in order to build and maintain a body.

The suicide and reprieve controls have been compared to the dual-key on a nuclear missile [20]. The key (chemical signal) turns on cell growth but at the same time switches on a sequence that leads to self-destruction. The second key overrides the sD [20].

Inspired by this biological mechanism, Autonomic Apoptosis (*Stay alive*) is proposed as an additional construct used to safeguard the system and component/agent both in terms of security (including self-protection) in the environment and from *ones self* (in case of incorrect adaptive behavior); the component (for instance an agent) has a preprogrammed death by default and receives a reprieve (stay-alive) signal indicates that the agent is still operating within a safe, correct context and behavior mode, and should not sD (Fig. 7) [21,22]. This approach is emerging as an area referred to as Apoptotic Computing [23].

Fig. 7 also indicates (as a dotted line) a sD signal may be sent. This we refer to as weak Apoptotic Computing because the preprogrammed death is triggered upon receiving an explicit sD signal as opposed to implicit default death unless a stay alive signal is received (strong Apoptotic Computing) [23].

C. ALice Signal (Autonomic Licence)

An aspect to this research is that Anonymous Autonomous/Autonomic Agents need to work within the Autonomic System to facilitate self-management; as such the agents and their hosts need to be able to identify each other's credentials through such means as an ALice (Autonomic License) signal [24]. This would allow a set of communications to ensure the visiting mobile agent has valid and justified reasons for being there as well as providing security to the visiting agent in interaction with other agents and host (Fig. 8). An unsatisfactory ALice exchange may lead to self-destruction for self-protection.

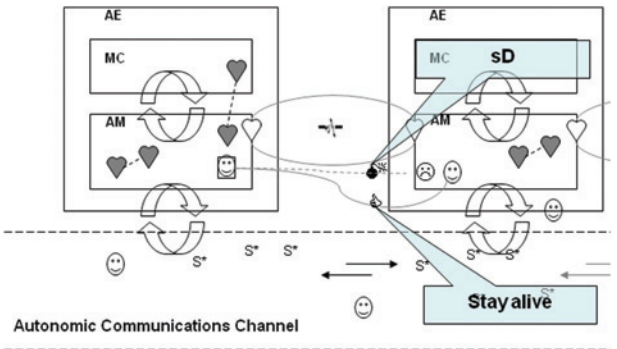

Fig. 7 An autonomic environment, including autonomic apoptosis; stay alive and self-destruct.

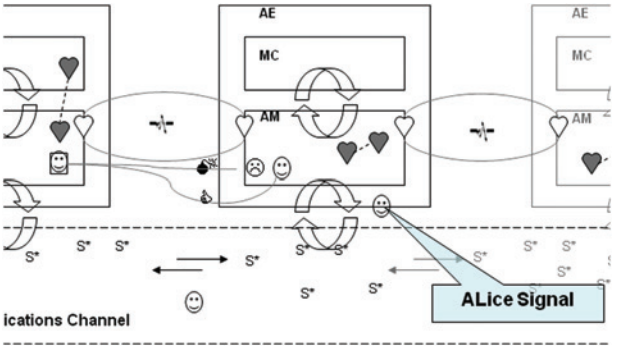

Fig. 8 An autonomic environment, including ALice.

D. Autonomic Quiescence (Self-Sleep)

The biological cell cycle is often described as a circle of cell life and division. A cell divides into two "daughter cells" and both of these cells live, "eat," grow, copy their genetic material, and divide again producing two more daughter cells. Because each daughter cell has a copy of the same genes in its nucleus, daughter cells are "clones" of each other. This "twinning" goes on and on with each cell cycle. This is a natural process.

But there is a kind of "parking spot" in the cell cycle, called "quiescence". A quiescent cell has left the cell cycle, it has stopped dividing. Quiescent cells may re-enter the cell cycle at some later time, or they may not; it depends on the type of cell. Most nerve cells stay quiescent forever. On the other hand, some quiescent cells may later re-enter the cell cycle in order to create more cells (for example, during pubescent development) [25].

The agent self-destruction proposed earlier (Autonomic Apoptosis) to facilitate security measures may be considered an extreme or ultimate self-protection measure—for cases when the agent's security has been breached or the agent is endangering the system (for instance demonstrating undesirable emergent behavior) [21,22]. Yet, not all cases may require this extreme reaction. Self-sleep (Quiescent state) [26] instead of sD (Apoptosis) may be all that is required for certain circumstances. As the situation emerges and is clarified, the agent may resume its activity or be put into an apoptotic state (Fig. 9). Strong/weak quiescence is akin to the strong/weak apoptotic process with reference to stay awake versus self-sleep in Fig. 9.

IV. Related Work

Initially we applied these radical concepts such as apoptosis to future space exploration systems and specifically to NASA's ANTS (Autonomous NanoTechnology Swarm) [27] where potentially a swarm of thousands of adaptive nano space craft would require such radical measures to ensure the success of an autonomous mission.

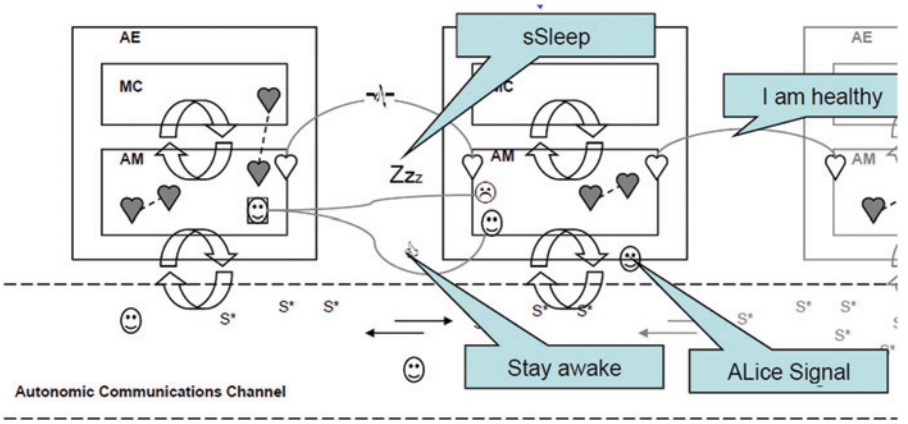

Fig. 9 An autonomic environment, including autonomic quiescence and stay awake.

As it happens some of these constructs are also applicable to today's systems and have been applied in the much wider field of Autonomic Systems for instance to telecommunications, agent-based systems, biometric security systems, grid, personal and pervasive computing [13,15].

Specifically, in the Space Exploration Systems field work is being channeled through AA-SES series (Autonomous and Autonomic Space Exploration Systems) of workshops with contributions to the field from NASA, Academia and industry [28–31].

V. Conclusion

We have discussed emerging self-* properties, self-managing elements incorporating reflex and reflection 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.

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

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.

We extended the properties for an autonomic environment by seeking further inspiration from biology, namely Apoptosis (sD) and Quiescence (Self-Sleep).

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 next generation computer-based systems that aims to bring new levels of automation and dependability to systems, while simultaneously hiding their complexity and reducing their TCO.

Acknowledgments

The author is supported by the University of Ulster's Computer Science Research Institute and School of Computing and Mathematics. Some of the research described in this article is patented and patent-pending by Roy Sterritt (University of Ulster) and Mike Hinchey (Lero—the Irish Software Engineering Research Centre) with NASA and assigned to the US government. This paper is a combined and extended view of work first presented at Autonomous & Autonomic Space Exploration Systems workshop series: AA-SES-I [16] – AA-SES-IV [32].

References

[1] Sterritt, R., "Towards Autonomic Computing: Effective Event Management," *Proceedings of the 27th Annual IEEE/NASA Software Engineering Workshop (SEW)*, MD, USA, 3–5 December 2002, 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, 2001. http://www.research.ibm.com/autonomic/

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

[4] IBM. "An Architectural Blueprint for Autonomic Computing," 2003.

[5] Kephart, J. O., and Chess, D. M., "The Vision of Autonomic Computing," *Computer*, Vol. 36, No. 1, 2003, pp. 41–52. doi: 10.1109/MC.2003.1160055

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

[7] Sterritt, R., "Autonomic Computing," *Innovations in Systems and Software Engineering: a NASA Journal*, Vol. 1, No. 1, 2005, pp. 79–88.

[8] Sterritt, R., Parashar, M., Tianfield, H., and Unland, R., "A Concise Introduction to Autonomic Computing," *Advanced Engineering Informatics*, Vol. 19, No. 3, 2005, pp. 181–187. doi: 10.1016/j.aei.2005.05.012

[9] Ganek, A. G., and Corbi, T. A., "The Dawning of the Autonomic Computing Era," *IBM Systems Journal*, Vol. 42, No. 1, 2003, pp. 5–18. doi: 10.1147/sj.421.0005

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

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

[12] Norman, D. A., Ortony, A., and Russell, D. M., "Affect and Machine Design: Lessons for the Development of Autonomous Machines," *IBM Systems Journal*, Vol. 42, No. 1, 2003, pp. 38–44.
doi: 10.1147/sj.421.0038

[13] 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 the 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.

[14] Sterritt, R., and Chung, S., "Personal Autonomic Computing Self-Healing Tool," *Proceedings of the 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.

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

[16] Sterritt, R., and Hinchey, M. G., "SPAACE: Self- Properties for an Autonomous & Autonomic Computing Environment," *Proceedings of the Autonomic & Autonomous Space Exploration Systems (A&A-SES-1) at 2005 International Conference on Software Engineering Research and Practice (SERP'05)*, Las Vegas, 27–30 June 2005, CSREA Press, pp. 3–8.

[17] Lockshin, R., and Zakeri, Z., "Programmed Cell Death and Apoptosis: Origins of the Theory," *Nature Reviews Molecular Cell Biology*, Vol. 2, 2001, pp. 542–550.

[18] Ishizaki,Y., Cheng, L., Mudge,A.W., and Raff, M. C., "Programmed Cell Death by Default in Embryonic Cells, Fibroblasts, and Cancer Cells," *Molecular Biology of the Cell*, Vol. 6, No. 11, 1995, pp. 1443–1458.

[19] Klefstrom, J, Verschuren, E. W., and Evan, G. I., "c-Myc Augments the Apoptotic Activity of Cytosolic Death Receptor Signaling Proteins by Engaging the Mitochondrial Apoptotic Pathway," *The Journal of Biological Chemistry*, Vol. 277, 2002, pp. 43224–43232.
doi: 10.1074/jbc.M206967200

[20] Newell, J., "Dying to Live:Why our Cells Self-Destruct," *Focus*, December 1994.

[21] Sterritt, R., and Hinchey, M. G., "Apoptosis and Self-Destruct: A Contribution to Autonomic Agents?," *Proceedings of the Third NASA-Goddard/IEEE Workshop on Formal Approaches to Agent-Based Systems (FAABS III)*, Washington, DC, 26–27 April 2004, in "LNAI 3228", Springer, December 2004, pp. 262–270.

[22] Sterritt, R., and Hinchey, M. G., "Engineering Ultimate Self-Protection in Autonomic Agents for Space Exploration Missions," *Proceedings of the IEEE Workshop on the Engineering of Autonomic Systems (EASe 2005) at 12th Annual IEEE International Conference andWorkshop on the Engineering of Computer Based Systems (ECBS 2005)*, Greenbelt, MD, 3–8 April 2005, pp. 506–511.

[23] Sterritt, R., "Apoptotic Computing: Programmed Death by Default for Computer-Based Systems," *Computer*, Vol. 44, No. 1, 2011, pp. 59–65.
doi: 10.1109/MC.2011.5

[24] Sterritt, R., and Hinchey, M. G., "Radical Concepts for Self-Managing Ubiquitous and Pervasive Computing Environments," *Proceedings of theWRAC-II, 2nd NASA/IEEEWorkshop on Radical Agent Concepts*, September 2005, in LNAI 3825, 2007.
doi: 10.1007/11964995_33

[25] Coller, H.A., Sang, L. and Roberts, J.M.,"A new description of cellular quiescence," *PLoS Biol.* 4(3) e83, 2006.
doi: 10.1371/journal.pbio.0040083

[26] Sterritt, R., and Hinchey, M. G., "Biologically-Inspired Concepts for Autonomic Self-Protection in Multiagent Systems," *Proceedings of the 3rd InternationalWorkshop on Safety and Security in Multi-Agent Systems (SASEMAS 2006) at AAMAS 2006*, Hakodate, Japan, May 2006.

[27] Clark, P. E., Rilee, M. L., Curtis, S. A., Cheung, C. Y., Truszkowski, W., Marr, G., and Rudisill, M, "PAM: Biologically Inspired Engineering and Exploration Mission Concept, Components, and Requirements for Asteroid Population Survey," *Proceedings of the 55th International Astronautical Congress (IAC 04), International Astronautical Federation*, 2004. http://ants.gsfc.nasa.gov/documents/Clark308IAC2004.pdf.

[28] Sterritt, R., et al. (eds.), *Proceedings of AA-SES-I: InternationalWorkshop on Autonomic &Autonomous Space Exploration Systems*, at the *2005 International Conference on Software Engineering Research and Practice (SERP'05)*, 27–30 June 2005, Las Vegas, NV.

[29] Sterritt, R., et al. (eds.), *Proceedings of AA-SES-II: 2nd IEEE InternationalWorkshop on Autonomic & Autonomous Space Exploration Systems II*, at the *2006 IEEE International Conference on Space Mission Challenges for IT (SMC-IT)*, 17–21 July 2006, Pasadena, CA.

[30] Sterritt, R., et al. (eds.), *Proceedings of AA-SES-III: 3rd IEEE InternationalWorkshop on Autonomic & Autonomous Space Exploration Systems III (at EASe)*, in *Proceedings of the Forth IEEE InternationalWorkshop on Engineering of Autonomic and Autonomous Systems (EASe 2007)*, 6–8 March 2007, Baltimore, MD.

[31] Sterritt, R., et al. (eds.), *Proceedings of AA-SES-IV: 4th IEEE International Workshop on Autonomic and Autonomous Space Exploration Systems (at SMC-IT), Pasadena, CA, USA, June 2009*, in *Proceedings of the Seventh IEEE International Conference and Workshops on Engineering of Autonomic and Autonomous Systems (EASe 2010)*, IEEE CS Press.

[32] Sterritt, R., and Hinchey, M. G., "SPAACE IV: Self-Properties for an Autonomous &Autonomic Computing Environment–Part IVA Newish Hope," *Proceedings of the AA-SES-IV: 4th IEEE InternationalWorkshop on Autonomic and AutonomousSpace Exploration Systems (at SMC-IT)*, Pasadena, CA, June 2009; *Proceedings of the Seventh IEEE International Conference and Workshops on Engineering of Autonomic and Autonomous Systems (EASe 2010)*, IEEE CS Press, pp. 119–125.

Michael G. Hinchey
Editor-in-Chief