Sustainable and Autonomic Space Exploration Missions

Roy Sterritt[1], Mike Hinchey[2], Christopher Rouff[3]
James Rash[2] and Walt Truszkowski[2]

[1] University of Ulster, [2] NASA GSFC, [3] SAIC ACBU,
r.sterritt@ulster.ac.uk, michael.g.hinchey@nasa.gov, rouffc@saic.com, james.l.rash@nasa.gov, walter.f.truszkowski@nasa.gov

Abstract

Visions for future space exploration have long term science missions in sight, resulting in the need for sustainable missions. Survivability is a critical property of sustainable systems and may be addressed through autonomicity, an emerging paradigm for self-management of future computer-based systems based on inspiration from the human autonomic nervous system. This paper examines some of the ongoing research efforts to realize these survivable systems visions, with specific emphasis on developments in Autonomic Policies.

1. Introduction

The vision for future Space Exploration Missions (SEMs) is "not [merely] looking at planting flags," and then "not being able to go back for 100 years" [1]. Systems that would take humans (possibly using as stepping stones the International Space Station (ISS) and the Moon), to Mars, or take unmanned missions to the asteroid belt would be reusable systems, with mission durations lasting upwards of 10 years. At this stage we are only beginning to "build the rail roads" [2] and lay the foundations for such missions.

This vision for future exploration missions requires *sustainable space capabilities* [1], in particular since it will involve the establishment of bases on the Moon for the eventual trip to Mars [2].

Sustainable SEMs will have many dependant properties, not least of which is survivability. Survivable Systems are systems that are able to complete their mission in a timely manner, even if significant portions are compromised by attack or accident [3],[4].

The case has been well presented in the literature for the need not only to create self-managing systems, due to the complexity problem that causes ever increasing total costs of ownership, but also to provide the way forward in enabling future pervasive and ubiquitous computation and communications [5]-[8]. Another requirement for self-management is in the facilitation of survivable systems [9],[10]. To enable self-management (autonomicity) a system requires many self properties (self-* or selfware), such as self-awareness.

This paper looks at some ongoing research in the autonomic and autonomous systems area that will contribute to the creation of future exploration missions that are even more survivable and sustainable than missions of today. Specifically it focuses on the aspect of policies (in these SEM scenarios, the high level sustainable scientific and mission goals) as the means to guide the self-managing survivable systems.

2. Requirements for Sustainable Systems

Computer-based systems are expected to be effective. This means that they serve a useful purpose when they are first introduced and continue to be useful as conditions change. From this perspective they should also be survivable. Decisions and directions taken by the system automatically without real-time human intervention are autonomous decisions. Responses taken automatically by a system without real-time human intervention are autonomic responses [11]. The NASA view of autonomous systems is slightly different from this general systems view. NASA views "autonomy" as indicating operating without assistance from ground control, and as such this could mean the inclusion of an astronaut in the

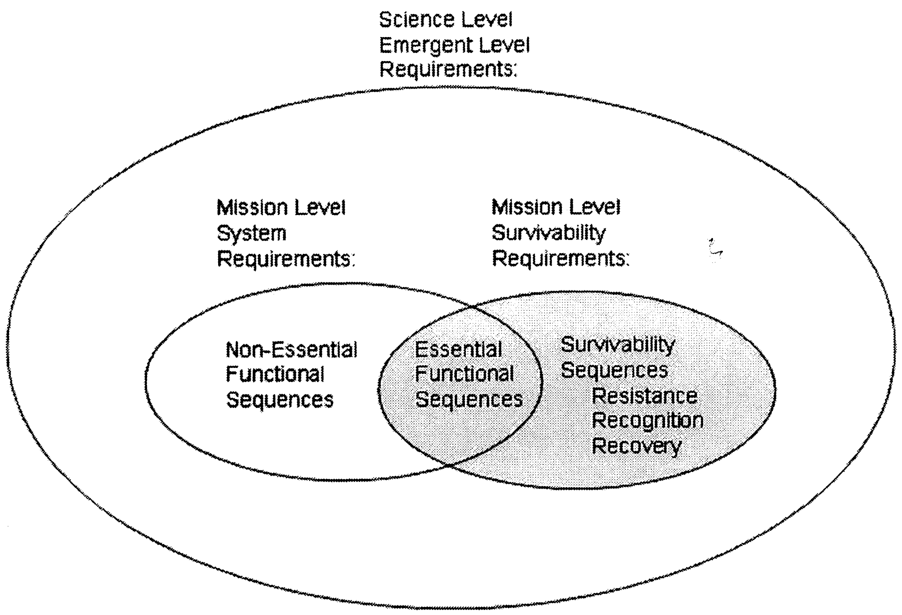

Figure 1. Integrating Survivability Requirements with System Requirements (adapted from [4]).

loop. In this paper we consider the general systems view.

Many branches of computer science research and development will contribute to progress in this area. Research on dependable systems should be especially influential, as dependability covers many relevant system properties such as reliability, availability, safety, security, survivability and maintainability [13],[14].

Figure 1 highlights the fact that when the mission requirements are being established there will be intrinsic survivability requirements underpinning the mission.

3. Survivability through Autonomic Systems

The autonomic concept is inspired by the human body's autonomic nervous system. Humans have good mechanisms for adapting to changing environments and repairing minor physical damage. The autonomic nervous system monitors heartbeat, checks blood sugar levels and keeps the body temperature normal without any conscious effort from the human. This biological autonomicity is influencing a new paradigm for computing to create self-management within computer-based systems (Autonomic Computing, Autonomic Communications and Autonomic Systems). There is an important distinction between autonomic activity in the human body and autonomic responses in computer-based systems. Many of the decisions made by autonomic elements in the body are involuntary, whereas autonomic elements in computer-based systems make decisions via special tasks that implement autonomic technology [12].

In the late 1990s DARPA/ISO's Autonomic Information Assurance (AIA) program studied defense mechanisms for protecting information systems against malicious adversaries. The AIA program resulted in two hypotheses: (1) fast responses are necessary to counter advanced cyber-adversaries and (2) coordinated responses are more effective than local reactive responses [11]. These hypotheses may provide general guidance for creating autonomic survivable systems.

"Autonomic Computing" became mainstream within the Computing milieu in 2001 when IBM launched their perspective on the state of information technology [5]. IBM defined four key self properties: self-configuring, self-healing, self-optimizing and self-protecting [12]. In the few years since, the self-x list has grown as research expands, bringing about the general term selfware or self-*; yet these four initial self-managing properties along with the four enabling properties: self-aware (of internal capabilities and state of the managed component), self-situated (environment and context awareness), self-monitor and self-adjust (through sensors, effectors and control loops), cover the general goal of self management [14].

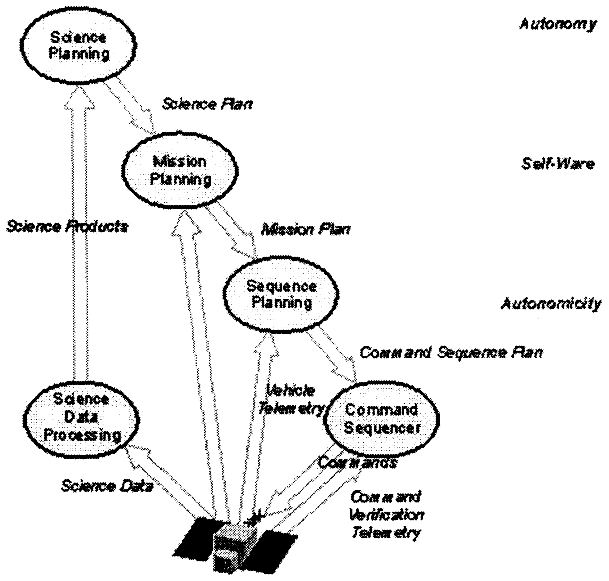

Figure 2. Progressive Autonomy & Autonomicity [15].

The premise is that the sustainable need for survivable properties such as resistance, recognition and recovery (Figure 1) can be provided through autonomic techniques.

Autonomic Systems work through creating a cooperative environment where elements, nodes and components are each assigned an autonomic manager (Figure 3).

These autonomic managers provide the self-awareness (self-monitoring and self-adjusting of the managed component) and environment-awareness (monitoring and reacting to the dynamic conditions of the environment). The autonomic manager to autonomic communications (AM⇔AM in Figure 3) includes several dynamic loops of control—for instance a fast loop provide reflex reactions and a slower loop providing coordinated event telemetry. These loops will not only trigger autonomic/self-management activity but also feed up into higher planes.

Figure 2 depicts the layers in a mission. The top layer contains the goal of the mission—the science. This has been classified as the autonomous layer due to the fact that it contains the self-governance, high level goals, and policies that the mission must meet, including the emergent constraints for discovering and planning new opportunistic science.

The middle (self-ware) layer (Figure 2) depicts the day-to-day autonomous and autonomic activity necessary to meet the mission plans.

The bottom (autonomic) layer depicts the instant/reflex reaction activity that will need to occur to address arising situations, and to ensure that correct and survivable activity takes place.

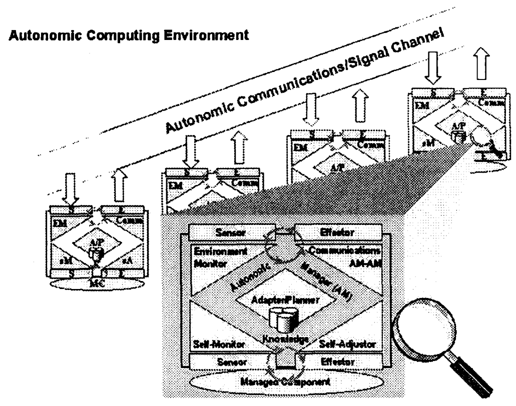

Figure 3. Autonomic Elements (Autonomic Manager + Managed Component).

4. Developing Sustainable Systems

The required complexity in these systems is evident. By their very nature the systems are critical systems due to the remoteness of the missions, the risk to and necessary safety of human life, and the significant costs involved. Components in the system may fail but the system must be sufficiently flexible and dynamic in nature to self-configure and self-heal to avoid total system failure and provide an effective work-around. This requirement to adapt encourages the facilitating of self-adaptation and emergence in the system, while at the same time raises concerned regarding non-desirable emergent behavior that may endanger the mission. Ongoing efforts to address this are briefly discussed in Section 4.3, in terms of verifying the correctness of the policies.

Another challenge is the orchestration between the different planes (autonomic ⇔ selfware ⇔ autonomous planes). As in communications and more recently in IT (through Autonomic Computing) the research topic of policy-based management has been identified as a potential means of specifying top-level policies that are then implemented and self-managed by the system. Ongoing research efforts in this area are briefly discussed in Section 4.2, in terms of using an AOSE approach and social hierarchy to orchestrate the policies between the planes.

4.1. Policies for Autonomic Systems

Policies have been described as a set of considerations designed to guide decisions of courses of action [16], and policy-based management (PBM) may be viewed as an administrative approach to systems management that establishes rules in advance to deal with situations that are likely to occur. From this perspective, policy-based management works by controlling access to, and setting priorities for, the use of computing and communication resources [17], so that, for instance, a (human) manager may simply specify the business objectives and the system will make it so in terms of the needed computing resources [18]. For example: (1) "The customer database must be backed up nightly between 1 a.m. and 4 a.m.," (2) "Platinum customers are to receive no worse than 1-second average response time on all purchase transactions," (3) "Only management and the HR senior staff can access personnel records," and (4) "The number of connections requested by the Web application server cannot exceed the number of connections supported by the associated database." [19]. These examples highlight the wide range and multiple levels of policies available, the first concerned with system protection through backup, the second with system optimization to achieve and maintain a level of quality of service for key customers; while the third and fourth are concerned with system configuration and protection.

Policy-Based Management (PBM) has been the subject of extensive research in its own right. The IETF has investigated Policy-Based Networking as a means for managing IP-based multi-service networks with quality of service guarantees. More recently, PBM has become extremely popular within the telecommunications industry, for next generation networking, with many vendors announcing plans and introducing products. This is driven by the fact that policy has been recognized as a solution for managing complexity, and to guide the behavior of a network or distributed system through high-level user-oriented abstractions [20].

A policy-based management tool may also reduce the complexity of product and system management by providing uniform cross-product policy definition and management infrastructure [21].

One perspective of Autonomic Computing is Policy-Based Self-Management.

The long term strategic vision of Autonomic Computing highlighted an overarching self-managing vision where the system would have such a level of "self" capability that a senior (human) manager in an organization could specify business policies—such as profit margin on a specific product range, or system quality of service for a band of customers— and the computing systems would do the rest [22].

It has been argued, and counter-argued, that for this vision to become a reality would require several computer science grand challenges to be solved beforehand—AI completeness, Software Engineering completeness and so on [23]. What is clear in this vision is the importance of policies to empower the system to self-manage at all levels.

With one definition of Autonomic Computing being *Self-Management based on high level guidance from humans* [24] and considering IBM's high-level set of self-properties (self-CHOP: configuration, healing, optimisation and protection) against the types of typical policies mentioned previously (optimization, configuration and protection), the importance and relevance of polices for achieving autonomicity become clear [25].

4.2. AOSE Approach

The field of Agent-Oriented Software Engineering (AOSE) has arisen to address methodological aspects and other issues related to the development of complex multi-agent systems. AOSE is a new software engineering paradigm that augurs much promise in enabling the successful development of more complex systems than is achievable with current Object-Oriented approaches which use agents and organizations of agents as their main abstractions [26].

The organizational metaphor has been proven to be one of the most appropriate tools for engineering Multi-Agent Systems (MAS). The metaphor is used by many researchers to guide the analysis and design of MASs, e.g., [27]-[29]. A MAS organization can be observed from two different points of view [29]:

Acquaintance point of view: shows the organization as the set of interaction relationships between the roles played by agents.

Structural point of view: shows agents as artifacts that belong to sub-organizations, groups, teams. In this view agents are also structured into hierarchical structures showing the social structure of the system.

Both views are intimately related, but they show the organization from radically different viewpoints. Since any structural organization must include interactions between their agents in order to function, it is safe to say that the acquaintance organization is always contained in the structural organization. Therefore, if we first determine the acquaintance organization, and we define the constraints required for the structural organization, a natural map is formed between the acquaintance organization and the corresponding structural organization. This is the process of assigning roles to agents [29]. Thus, we can conclude that any acquaintance organization can be modeled orthogonally to its structural organization [30]. We use this separation to specify policies at the acquaintance organization level, and deploy them over the structural organization of the running system. The scope of policies usually implies features of several acquaintance sub-organizations. In such cases, we must first compose the acquaintance sub-organizations, this process being guided by the policy specification, to deploy it later.

For more information on this work please refer to [31].

4.3. Development and Verification of Autonomic Policy-based Systems

As autonomic systems are essentially concerned with bringing self-management to highly complex systems, all the existing issues with developing and maintaining complex systems are still present from developing effective systems and software in the first place, and then managing their evolution through reverse- and re-engineering of the systems.

In this research we are working on formal requirements-based programming and extending it as a means to provide provably correct code generated from policies for autonomic systems. Specifically, we are developing NASA's R2D2C technologies for mechanically transforming policies (expressed in restricted natural language, or appropriate graphical notations) into a provably equivalent formal model that can be used as the basis for code generation and other transformations, including reverse engineering the process and working towards the self-generation of provable autonomic policies.

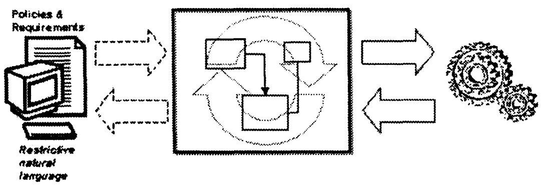

Figure 4. The R2D2C approach, generating a formal model from requirements and producing code.

Our experience at NASA Goddard Space Flight Center (GSFC) has been that while engineers are happy to write descriptions as natural language scenarios, or even using semi-formal notations such as UML use cases, they are loath to undertake formal specification. Absent a formal specification of the system under consideration, there is, in general, no possibility of determining any level of confidence in the correctness of an implementation. More importantly, we must ensure that a formal specification fully, completely, and consistently captures the requirements set forth at the outset. Clearly, we cannot expect requirements to be perfect, complete, and consistent from the outset, which is why it is even more important to have a formal specification, which can highlight errors, omissions, and conflicts. The formal specification must also reflect changes and updates from system maintenance as well as changes and compromises in requirements, so that it remains an accurate representation of the system and its requirements.

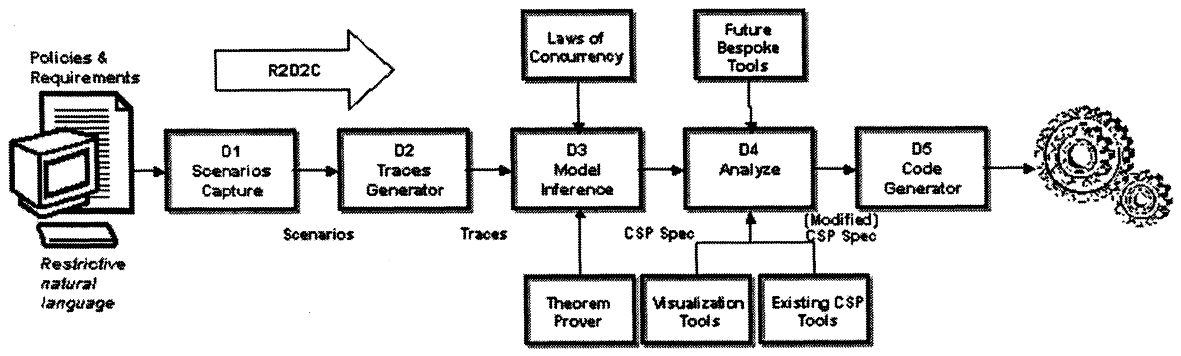

Figure 5. The entire forward process with D1 thru D5 illustrating the development approach.

R2D2C, or Requirements-to-Design-to-Code [32][33], is a NASA patent-pending approach to Requirements-Based Programming that provides a mathematically tractable round-trip engineering approach to system development. In R2D2C, engineers (or others) may write specifications as scenarios in constrained (domain-specific) natural language, or in a range of other notations (including UML use cases). These will be used to derive a formal model (Figure 1) that is guaranteed to be equivalent to the requirements stated at the outset, and which will subsequently be used as a basis for code generation. Policies expressed in natural language will, in effect, constitute the requirements for a system and through the R2D2C method will be transformed into an equivalent formal model. The formal model can be expressed using a variety of formal methods. Currently we are using CSP, Hoare's language of Communicating Sequential Processes [34][35], which is suitable for various types of analysis and investigation, and as the basis for fully formal implementations as well as for use in automated test case generation, etc.

R2D2C is unique in that it allows for full formal development from the outset, and maintains mathematical soundness through all phases of the development process, from policies and requirements through to automatic code generation. The approach may also be used for reverse engineering, that is, in retrieving policies, models and formal specifications from existing code, as shown in Figure 4. The approach can also be used to "paraphrase" (in natural language, etc.) formal descriptions of existing systems.

This approach is not limited to generating code from policies. It may also be used to generate business processes and procedures, and we have been experimenting with using it to generate instructions for robotic devices that were to be used on the Hubble Robotic Servicing Mission (HRSM), which, at the time of writing, has not received a final go-ahead. We are also experimenting with using it as a basis for an expert system verification tool, and as a means of capturing domain knowledge for expert systems.

For more information on this work please refer to [25][32][33].

5. Conclusion

Sustainable and Survivable Systems are essential to realize future space exploration missions. Autonomic Systems—self-managing computer-based systems inspired by the self-managing activity of the biological autonomic nervous system—may contribute to achieving sustainable systems.

One vision of Autonomic Computing is Self-Management based on high level guidance from humans. Policies and policy-based management are a key enabling technology for achieving autonomicity. Their importance to space exploration missions lies in realizing and orchestrating high-level science goals (or policies) with the low-level dynamic day-to-day survivability requirements.

This paper has briefly described (and indicated some further reading on) some of our research in this area including a method that can produce fully (mathematically) tractable development of policies for autonomic systems from requirements through to code generation and an approach to modeling, specifying and deploying the policies throughout the sustainable systems planes. Fitting with the SEM vision, at this stage we are only beginning to establish how to "build the rail roads".

Acknowledgements

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 Software Engineering Laboratory, Goddard Space Flight Center (Code 581).

This research is partly supported at University of Ulster by the Computer Science Research Institute (CSRI) 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.

References

[1] G. Martin, "NASA Exploration Team (NExT) program", World Space Congress, Houston, Texas, Oct. 10-19, 2002.

[2] D.J. Atkinson, "Constellation Program Return to the Moon: Software Systems Challenges", Keynote presentation, 3rd IEEE International Workshop on Engineering of Autonomic and Autonomous Systems (EASe 2006), Columbia, MD, USA, April 2006.

[3] R.C. Linger, N.R. Mead, H.F. Lipson, "Requirements Definition for Survivable Systems," In Proc. *3rd IEEE Int. Conf. Requirements Engineering*, Colorado Springs, CO, 6-10 April, 1998, pp. 14-23.

[4] N. Mead, "Requirements Engineering for Survivable Systems", Technical Report CMU/SEI-2003-TN-013, Software Engineering Institute, Pittsburg, PA, 2003.

[5] P. Horn, "Autonomic computing: IBM perspective on the state of information technology," IBM T.J. Watson Labs, NY, 15th October 2001.

[6] R. Sterritt, "Towards Autonomic Computing: Effective Event Management", In *Proc. IEEE/NASA Software Engineering Workshop*, Greenbelt, MD, Dec. 2002, IEEE Computer Society Press.

[7] J.O. Kephart, D.M. Chess. "The Vision of Autonomic Computing", *Computer*, 36(1):41–52, 2003.

[8] R. Sterritt, "Autonomic Computing," *Innovations in Systems and Software Engineering: a NASA Journal*, 1(1):79-88, Springer, 2005.

[9] R. Sterritt, G. Garrity, E. Hanna, P. O'Hagan, "Survivable Security Systems through Autonomicity," In *Proc. 2nd NASA/IEEE Workshop on Radical Agent Concepts (WRAC 2005)*, Greenbelt, MD, USA, 20-22 September 2005, Springer LNAI 3825.

[10] R. Sterritt, G. Garrity, E. Hanna, P. O'Hagan, "Autonomic Agents for Survivable Security Systems," In *Proc. 1st IFIP Workshop on Trusted and Autonomic Ubiquitous and Embedded Systems (TAUES 2005)* at EUC'05, Nagasaki, Japan, 6-9 December 2005, Springer-Verlag LNCS 3823, pp 1235-1244.

[11] S.M. Lewandowski, D.J. Van Hook, G.C. O'Leary, J.W. Haines, L.M. Rossey, "SARA: Survivable Autonomic Response Architecture," In *Proc. DARPA Information Survivability Conference and Exposition II*, Vol. 1, pp. 77-88, June 2001.

[12] IBM, "An architectural blueprint for autonomic computing", 2003.

[13] B. Randell, "Turing Memorial Lecture – Facing Up to Faults", *Comp. J.* 43(2):95-106, 2000.

[14] R. Sterritt and D.W. Bustard, "Autonomic Computing: a Means of Achieving Dependability?" In *Proc. IEEE International Conference on the Engineering of Computer-Based Systems (ECBS'03*, Huntsville, AL, 7-11 April 2003, pp 247-251, IEEE Computer Society Press.

[15] W. Truszkowski, C.A. Rouff, H.L. Hallock, J. Karlin, J.L. Rash, M.G. Hinchey and R. Sterritt, *Autonomous and Autonomic Systems: With Applications to NASA Intelligent Spacecraft Operations and Exploration Systems*, NASA Monographs in Systems and Software Engineering, Springer Verlag, London, 2006.

[16] M.J. Masullo and S.B. Calo, "Policy Management: An Architecture and Approach", In *Proc. 1st IEEE International Workshop on Systems Management*, Los Angeles, CA, 14-16 April 1993.

[17] Whatis?com, Online computer and internet dictionary and encyclopaedia, 2005.

[18] L. Lymberopoulos, E. Lupu and M. Sloman, "An adaptive policy-based framework for network services management," *J. of Network Systems Management* 11(3), 2003.

[19] D. Kaminsky, "An Introduction to Policy for Autonomic Computing", IBM white paper, March 2005.

[20] A. Meissner, S. B. Musunoori and L. Wolf, "MGMS/GML - Towards a new Policy Specification Framework for Multicast Group Integrity", In Proc. 2004 International Symposium on Applications and the Internet (SAINT2004), Tokyo, Japan, 2004.

[21] A.G. Ganek, "Autonomic computing: implementing the vision." Keynote talk at the autonomic computing workshop (AMS 2003), Seattle, 25th June 2003.

[22] P. Horn, "Autonomic computing: IBM perspective on the state of information technology," Presented at AGENDA 2001, Scottsdale, AR, 2001.

[23] O. Babaoglu, A. Couch, G. Ganger, P. Stone, M. Yousif and J. Kephart, "Panel: Grand Challenges of Autonomic Computing," International Conference on Autonomic Computing (ICAC'05), Seattle, WA, June 2005.

[24] J. Kephart and W.E. Walsh "An Artificial Intelligence Perspective on Autonomic Computing Policies," Policy, Fifth, pp 3-12, 2004.

[25] R. Sterritt, M.G. Hinchey, J.L. Rash, W.F. Truszkowski, C.A. Rouff and D. Gracanin, "Towards Formal Specification and Generation of Autonomic Policies," *In Proc. of 1st IFIP Workshop on Trusted and Autonomic Ubiquitous and Embedded Systems (TAUES*

2005) at EUC'05, Nagasaki, Japan, 6-9 December 2005, pp 1245-1254, Springer LNCS 3823.

[26] N. Jennings. "An agent-based approach for building complex software systems." *Comm. of the ACM*, 44(4):35–41, 2001.

[27] J. Odell, H. Parunak, and M. Fleischer. "The role of roles in designing effective agent organizations", in LNCS 2603, pages 27–28, Berlin, 2003. Springer–Verlag.

[28] H. V. D. Parunak and J. Odell. "Representing social structures in UML," In *Proc. of the Fifth International Conference on Autonomous Agents*, pp. 100–101, Montreal, Canada, 2001, ACM Press.

[29] F. Zambonelli, N. Jennings, and M. Wooldridge. "Developing multiagent systems: the GAIA methodology," *ACM Transactions on Software Engineering and Methodology*, 12(3):317–370, July 2003.

[30] E. A. Kendall. "Role modeling for agent system analysis, design, and implementation," *IEEE Concurrency*, 8(2):34–41, Apr./June 2000.

[31] J. Pena, M.G. Hinchey, R. Sterritt, "Towards Modelling, Specifying and Deploying Policies in Autonomous and Autonomic Systems Using an AOSE Methodology," In *Proc. Third IEEE International Workshop on the Engineering of Autonomic and Autonomous Systems (EASe 2006)*, March 2006.

[32] M. G. Hinchey, J. L. Rash, and C. A. Rouff. *Requirements to design to code: Towards a fully formal approach to automatic code generation*. Tech. Rep. TM-2005-212774, NASA Goddard Space Flight Center, Greenbelt, MD, 2004.

[33] J. L. Rash, M. G. Hinchey, C. A. Rouff, and D. Gračanin, "Formal requirements-based programming for complex systems," In Proc. *IEEE International Conference on Engineering of Complex Computer Systems (IECCCS 2005)*, Shanghai, China, 16–20 June 2005, IEEE Computer Society Press.

[34] C.A.R. Hoare, "Communicating sequential processes", *Comm. of the ACM*, 21(8):666–677, 1978.

[35] C.A.R. Hoare, *Communicating Sequential Processes*, Prentice Hall International Series in Computer Science. Prentice Hall Int., Englewood Cliffs, NJ, and Hemel Hempstead, UK, 1985.