Why Computer-Based Systems *Should* be Autonomic

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

The objective of this paper is to discuss why computer-based systems should be autonomic, where autonomicity implies self-managing, often conceptualized in terms of being self-configuring, self-healing, self-optimising, self-protecting and self-aware. We look at motivations for autonomicity, examine how more and more systems are exhibiting autonomic behavior, and finally look at future directions.

Keywords: Autonomic Computing, Autonomic Systems, Self-Managing Systems, Complexity, Total Cost of Ownership, Future Computing Paradigms.

1. Introduction

Autonomic Computing and other self-managing initiatives have emerged as a significant vision for the design of computing based systems. Their goals are the development of systems that are self-configuring, self-healing, self-protecting and self-optimizing, among other self-* properties. The ability to achieve this selfware is dependant on self-awareness and environment awareness implemented through a feedback control loop consisting of sensors and effectors within the CBS (providing the self-monitoring and self-adjusting properties) [1]. Dependability is a long-standing desirable property of all computer-based systems while complexity has become a blocking force to achieving this. The autonomic initiatives offer a means to achieve dependability while coping with complexity [2].

The ECBS workshop on Engineering of Autonomic Systems (EASe) [3][4] aims to establish autonomicity as an integral part of a CBS and explore techniques, tools, methodologies, and so on, to make this happen. The spectrum and implications are so wide that we need to reach out to other research communities to make this a reality.

The purpose of this paper is to consider why systems should be autonomic, how many more than we might first think *are* already on the evolutionary autonomic path, and to look to the future and see what more we need to consider.

2. Facing Life As It Is…

Upon launching the Autonomic Computing initiative, IBM called on the industry to face up to the ever-increasing complexity and total cost of ownership.

2.1. Business Realities

- It is estimated that companies now spend between 33% and 50% of their total cost of ownership recovering from or preparing against failures [5].
- Many of these outages, with some estimates at as high as 40%, are caused by operators themselves [6].
- 80% of expenditure on IT is spent on operations, maintenance and minor enhancements [7].
- IBM has been estimated to add about 15,000 people per year to its service organization in order to assist customers in dealing with complex platforms [8]. One can assume that other large organizations are being required to make correspondingly large expansions to keep ahead (or even to keep up).

These realities together with the complexity and total cost of ownership (TCO) problem all highlight the need for a change.

2.2. Complexity

The world is becoming an ever-increasingly complex place. In terms of computer systems, this complexity has been confounded by the drive towards cheaper,

Proceedings of the 12th IEEE International Conference and Workshops on the Engineering of Computer-Based Systems (ECBS'05)
0-7695-2308-0/05 $20.00 © 2005 **IEEE**

faster and smaller hardware, and functionally rich software. The infiltration of the computer into every day life has made the reliance on it critical. As such, there is an increasing need throughout design, development and operation of computer systems to cope with this complexity and the inherent uncertainty within. There is an increasing need to change the way we view computing; there is a need to realign towards facing up to computing in a complex world.

The IT industry is a marked success; within a 50 year period it has grown to become a trillion dollar per year industry obliterating barriers and setting records with astonishing regularity [9][10]. Throughout this time the industry has had a single focus, namely to improve performance [11] which has resulted in some breathtaking statistics [12]:

- Performance/price ratio doubles around every 18 months,
- resulting in 100 fold per decade;
- Progress in the next 18 months will equal ALL previous progress;
- New storage = sum of all old storage, ever;
- New processing = sum of all old processing;
- Aggregate bandwidth doubles in 8 months.

This performance focus has resulted in the emergence of a small number of critical inherent behaviors in the way the industry operates when designing, developing and deploying hardware, software, and systems [11]:

- That humans can achieve perfection; that they avoid making mistakes during installation, upgrade, maintenance or repair.
- Software will eventually be bug free; the focus of companies has been to hire better programmers, and universities to train better software engineers, in development life-cycle models.
- Hardware mean-time between failure (MTBF) is already very large - approximately 100 years - and will continue to increase.
- Maintenance costs are a function of the purchase price of hardware; and as such with decreasing hardware costs (in terms of price/performance) results in decrease in maintenance costs.

When made explicit in this way, it is obvious that these implicit behaviors are flawed and result in contributing factors to the complexity problem.

Within the last decade, problems have started to become more apparent. For an industry that is used to metrics always rising we saw some key decreases. Figure 1 [9] highlights that key modern day systems – cell phones and the internet – have seen a decline in availability, changing the established trend of their counterparts.

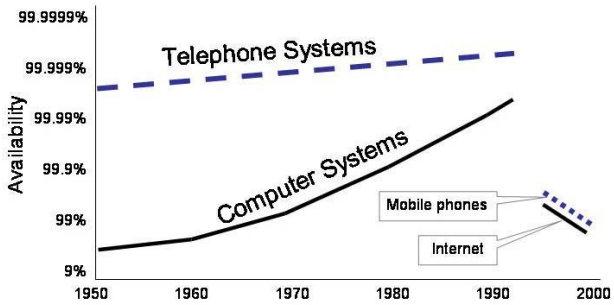

Figure 1 Systems Availability over the Decades [9]

We increasingly require our complex systems to be dependable. This is obvious, when one considers how dependant we have become on our systems and how much it costs for a single hour of downtime. For instance, in 2000: $6.5m brokerage operations, $2.5m credit card authorization and $¼m for eBay [11][13][14].

2.3. Total Cost of Ownership (TCO)

As hardware gets cheaper, the actual total cost of ownership – managing the complexity – increases. The following depicted statistics indicate where some of these costs lie.

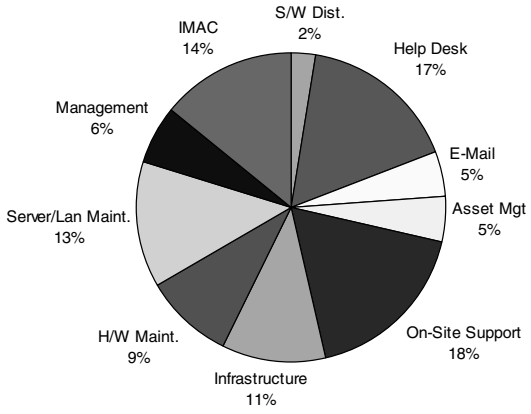

Figure 2 Client Computing TCO (%) [15]

In terms of direct accountable IT costs (Figure 2) the average costs per desktop is $100-180 per month for organizations with greater than 5000 employees. Figure 2 indicates online support (18%) and helpdesk (17%) are the biggest costs; activities like IMAC and email support

and require 14% and 5% of costs. A lot of these activities would require less attention if they contained of more self-management capability.

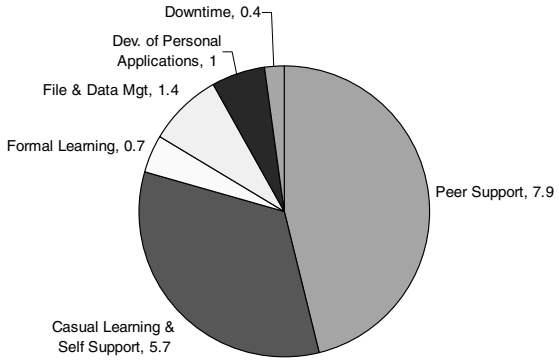

Figure 3 Client computing TCO – indirect costs (hrs/mth) [15]

Figure 3 expresses indirect costs of client computing in terms of hours per month – where those that are not employed directly in an IS role and are users of computing, have to resort to some of their own self help, peer help and their own administration. It is estimated that it requires on average 205.2 hr/yr; if you cost this at $50 an hour these indirect costs amount to $10,260 per user! As with direct costs, these activities would be reduced if our systems had more autonomicity.

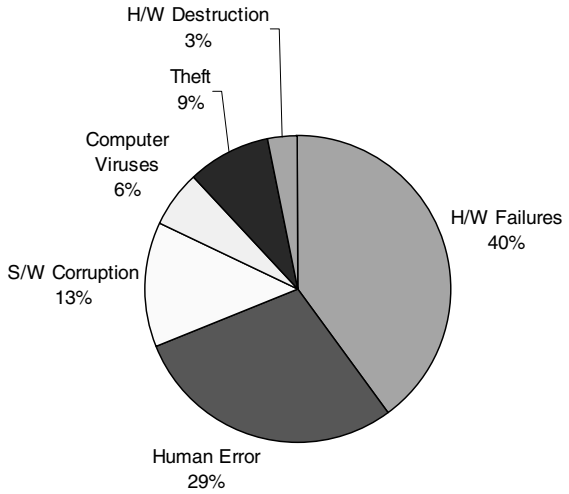

Figure 4 The sources of data loss (Incident rate 6% of devices per year) [15]

Figure 4 depicts the sources of data loss where incidents occur on 6% of devices. The value of data/information/knowledge is commonly acknowledged within organizations as a valuable asset. Yet, with client and personal computing, apart from the expensive direct costs of recovery from major failure, there is a substantial risk of indirect costs associated with work lost through ineffective back-up procedures. Selfware that facilitates smart back-ups, that reduces situations that cause human error in the first place, and is more proactive in monitoring the health of the device, provides the potential to reduce these costs and incidents of data loss in the first place.

3. Autonomicity is (Already) All Around

Autonomicity is already being built into many systems in an effort, primarily, to increase dependability and reduce downtime, but also as a means of achieving greater usability and user friendliness.

An obvious example, that many of us encounter every day, is in the operating systems of our home or work computers, or of our cellphones and PDAs. These are increasingly being enhanced with mechanisms for reducing the number of errors we can make, and coping with updates and additions.

3.1. A Simple Example -Windows XP

Consider the Windows XP operating system as an example. XP already meets a lot of the requirements now recognized as necessary to meet the criteria of being an autonomic system. As always, the four criteria are interrelated and what contributes to being "self-healing" may also contribute to "self-optimizing", etc.

Self-configuring

XP, like many other operating systems, is designed to be installed automatically. The user is guided through a new installation, even given the option of accepting a standard installation, or selecting a customized installation which XP will complete for the user.

XP is also self-reconfiguring. The operating system automatically detects newly installed programs and newly installed (or connected) hardware devices. Long gone are the days of needing to install drivers and download new drivers from manufacturer's websites. XP automatically detects new hardware connected, and installs it appropriately. In most cases, it can even identify the device and name it correctly. It's not surprising that it can do so with newer devices, but it can successfully identify devices that are many years old and configure (or re-configure) itself to deal with those.

Self-healing

XP is able to recover from a plethora of errors, all automatically.

It can detect the unsafe removal of devices and will attempt to overcome this. It is able to automatically download patches to deal with new viruses, security breaches, or merely errors that were undetected before that version of the operating system was released. Where an error cannot be recovered from, the operating system prepares an appropriate report and returns it to Microsoft.

Self-optimizing

XP is able to download updates and enhancements. Some of these are for the purposes of self-healing and self-protecting (we've all had experiences of XP's vulnerability to hackers and worms) and correcting previously undetected errors. Others are to enhance the operation of the system and to improve its performance.

Self-protecting

Just like many other operating systems, XP is able to protect itself from various errors, such as unsafe removal of devices, sudden loss of resources, etc. Additionally it provides a form of checkpointing so that documents, etc., can be recovered following a crash.

In many cases, this protection takes the simple form of returning to a state of limited operation, or "safe state" where current settings and data will be protected to a certain extent. Subsequently, data and open files can be restored to that last checkpoint. In other cases, patches can be downloaded as an automatic update.

Self-aware

To perform many of these functions, XP must have a certain degree of self-awareness. It must be aware of its current status (so that it can recover from crashes, etc.) as well of its peripherals, etc. More importantly, it must be aware of the current version of the operating system that it itself is comprised of. Only in this way will it know which updates, patches, etc., to download and install.

The Dynamic Systems Initiative (DSI) is Microsoft's initiative to facilitate this self-awareness through knowledge (creation, modification, transfer, and operation) about the system for the lifecycle of that system [16]. These are seen as core principles in addressing the complexity and manageability challenges.

3.2. A More Complex Example – NASA Missions

At the other end of the scale, we have systems which are less common and less likely to be encountered on a regular basis, but which attract significant publicity, particularly when they fail. These are exemplified by NASA missions, amongst others.

NASA missions require the use of complex hardware and software systems, and embedded systems, often with hard real-time requirements. Most missions involve significant degrees of autonomous behavior, often over significant periods of time. There are missions which are intended only to survive for a short period, and others which will continue for decades, with periodic updates to both hardware and software. Some of these updates are pre-planned; others, such as with the Hubble Space Telescope, were not planned but now will be undertaken (with updates performed either by astronauts or via a robotic arm).

While missions typically have human monitors, many missions involve very little human intervention, and then often only in extreme circumstances. It has been argued that NASA systems *should* be autonomic [1][17], and that all autonomous systems should be autonomic by necessity. Indeed, the trend is in that direction in forthcoming NASA missions.

We take as our example, a NASA concept mission, ANTS, which has been identified [18] as a prime example of an autonomic system.

3.2.1 ANTS

ANTS is a concept mission that involves the use of intelligent swarms of spacecraft. From a suitable point in space (called a Lagrangian), 1000 small spacecraft will be launched towards the asteroid belt.

As many as 60% to 70% of these will be destroyed immediately on reaching the asteroid belt. Those that survive will coordinate into groups, under the control of a leader, which will make decisions for future investigations of particular asteroids based on the results returned to it by individual craft which are equipped with various types of instruments.

Self-configuring

ANTS will continue to prospect thousands of asteroids per year with large but limited resources. It is estimated that there will be approximately one month of optimal science operations at each asteroid prospected. A full suite of scientific instruments will be deployed at each asteroid. ANTS resources will be configured and

re-configured to support concurrent operations at hundreds of asteroids over a period of time.

The overall ANTS mission architecture calls for specialized spacecraft that support division of labor (rulers, messengers) and optimal operations by specialists (workers). A major feature of the architecture is support for cooperation among the spacecraft to achieve mission goals. The architecture supports swarm-level mission-directed behaviors, sub-swarm levels for regional coverage and resource-sharing, team/worker groups for coordinated science operations and individual autonomous behaviors. These organizational levels are not static but evolve and self-configure as the need arises. As asteroids of interest are identified, appropriate teams of spacecraft are configured to realize optimal science operations at the asteroids. When the science operations are completed, the team disperses for possible reconfiguration at another asteroid site. This process of configuring and reconfiguring continues throughout the life of the ANTS mission.

Reconfiguring may also be required as the result of a failure, such as the loss of, or damage to, a worker due to collision with an asteroid (in which case the role may be assumed by another worker, which will be allocated the task and resources of the original).

Self-healing

ANTS is self-healing not only in that it can recover from mistakes, but self-healing in that it can recover from failure, including damage from outside forces. In the case of ANTS, these are non-malicious sources: collision with an asteroid, or another spacecraft, etc.

ANTS mission self-healing scenarios span the range from negligible to severe. A negligible example would be where an instrument is damaged due to a collision or is malfunctioning. In such a scenario, the self-healing behavior would be the simple action of deleting the instrument from the list of functioning instruments. A severe example would arise when the team loses so many workers it can no longer conduct science operations. In this case, the self-healing behavior would include advising the mission control center and requesting the launch of replacement spacecraft, which would be incorporated into the team, which in turn would initiate necessary self-configuration and self-optimization.

Individual ANTS spacecraft will have self-healing capabilities also. For example, an individual may have the capability of detecting corrupted code (software), causing it to request a copy of the affected software from another individual in the team, enabling the corrupted spacecraft to restore itself to a known operational state.

Self-optimizing

Optimization of ANTS is performed at the individual level as well as at the system level.

Optimization at the ruler level is primarily through learning. Over time, rulers will collect data on different types of asteroids and will be able to determine which asteroids are of interest, and which are too difficult to orbit or collect data from. This provides optimization in that the system will not waste time on asteroids that are not of interest, or endanger spacecraft examining asteroids that are too dangerous to orbit.

Optimization for messengers is achieved through positioning, in that messengers may constantly adjust their positioning in order to provide reliable communications between rulers and workers, as well as with mission control back on Earth.

Optimization at the worker level is again achieved through learning, as workers may automatically skip over asteroids that it can determine will not be of interest.

Self-protecting

The significant causes of failure in ANTS will be collisions (with both asteroids and other spacecraft), and solar storms.

Collision avoidance through maneuvering is a major challenge for the ANTS mission, and is still under development. Clearly there will be opportunity for individual ANTS to coordinate with other spacecraft to adjust their orbits and trajectories as appropriate. Avoiding asteroids is a more significant problem due to the highly dynamic trajectories of the objects in the asteroid belt. Significant planning will be required to avoid putting spacecraft in the path of asteroids and other spacecraft.

In addition, charged particles from solar storms could subject spacecraft to degradation of sensors and electronic components. The increased solar wind from solar storms could also affect the orbits and trajectories of the ANTS individuals and thereby could jeopardize the mission. One possible self-protection mechanism would involve a capability of the ruler to receive a warning message from the mission control center on Earth. An alternative mechanism would be to provide the ruler with a solar storm sensing capability through on-board, direct observation of the solar disk. When the ruler recognizes that a solar storm threat exists, the ruler would invoke its goal to protect the mission from harm from the effects of the solar storm, and issue instructions for each spacecraft to "fold" the solar sail (panel) is uses to charge its power sources.

Self-aware

Clearly, the above properties require the ANTS mission to be both aware of its environment and self-aware.

The system must be aware of the positions and trajectories of other spacecraft in the mission, of positions of asteroids and their trajectories, as well as of the status of instruments and solar sails.

4. Aiming For What You Would Like It To Be…

4.1. Future Computer-Based System Paradigms

Over the years we have seen the developments and gradual move from M:1 computing (1 computer for many people; the mainframe era), to 1:1 computing (personal computing era) towards the future era of 1:M computing where as individuals we utilize many computing devices, both in our working and personal daily lives. Bud Lawson in his 'Rebirth of the Computer Industry' ACM commentary [8] highlights that the complexity issues started long ago when the first general purpose mainframe was created and has increasingly gotten worse since resulting in a fundamental need to change to overcome the direction in which the industry is heading.

The driving force behind the future paradigms of computing is the increasing convergence between technologies [19]:

- proliferation of devices
- wireless networking
- mobile software

as well as industries converging; e.g., the Computer and Telecommunications industries. Also the increasing fuzziness of boundaries between devices used at work for business or in the home for entertainment (who ever had a mainframe at home?!) plays it part.

As Weiser first described what has become know as ubiquitous computing [20]; *"For thirty years most interface design, and most computer design, has been headed down the path of the "dramatic" machine. Its highest ideal is to make a computer so exciting, so wonderful, so interesting, that we never want to be without it. A less-traveled path I call the "invisible"; its highest ideal is to make a computer so embedded, so fitting, so natural, that we use it without even thinking about it"*.

These ideas of bringing computers into our world, rather than asking us to enter into the computer's world, has become widespread among researchers — albeit often under the alternative names of "pervasive computing", "ambient computing", or the term to emerge from the communications research community, "ambient networks", often referred to as "ambient intelligence", expressing the need for more intelligent computer networks. Other (more explicit) research names for future computing paradigms include "invisible computing" and "world computing" to express the concept of, in effect, a single system with (potentially) billions of "networked information devices".

Behind these different terms and research areas, emphasis is made on three properties [19]:

1. nomadic,
2. embedded and
3. invisible.

This reality of an increasingly networked world has also established the notion that computation need no longer be confined to computers. Instead, computation can be proliferated as a collection of processes, moving among desktops, mobiles, PDAs, servers, and any number of other devices, accumulating and re–accumulating themselves on the fly to meet the task at hand [19]. This computing vision goes by many names: distributed computing, Web services, Person-to-Person, Peer-to-Peer, organic IT, utility computing, and grid computing.

A grid infrastructure promises seamless access to computational and storage resources, and offers the possibility of cheap, ubiquitous distributed computing. Grid technology will have a fundamental impact on the economy by creating new areas, such as e-Government and e-Health, new business opportunities, such as computational and data storage services, and changing business models, such as greater organizational and service devolution [21][22]. The Grid is a very active area of research and development; with the number of academic grids jumping six fold in 2002 [23]. Its aim to fulfill the vision of Corbato's Multics [24] – like a utility company, a massive resource to which a customer gives his or her computational or storage needs [25].

With the ever increasing complexity and TCO from the M:1, 1:1 eras moving to 1:M era with all these billions of devices, is the future one of chaos?

For 1:M computing to become a successful reality will require a self-managing approach such as autonomic computing (along with other things such as new business models).

These future computer paradigms such as grid computing, utility computing, pervasive computing, ubiquitous computing, invisible computing, world computing, ambient intelligence, ambient networks, and so on, all will reside within the ECBS domain – a fusion of systems and software engineering - due to the fact that these systems will be highly dependent on devices and embedded systems. All these next generation infrastructures in one form or another will require an autonomic – self-managing – infrastructure.

The TC-ECBS with its focus on computer-based systems which incorporates the area of embedded systems is in a prime position to meet the future with these new paradigms.

5. Conclusion

This paper has recapped some problems facing the computer industry, and described its envisaged future paradigms, highlighting how the emerging autonomic and self-managing initiatives are necessary for current and future needs.

In order for Autonomic Computing to meet these needs, open standards and technologies will be necessary.

These précis of past, current, and future, can only lead to the conclusion that computer-based systems *should* be autonomic.

Acknowledgements

The development of this paper was supported at University of Ulster by the Computer Science Research Institute and the Centre for Software Process Technologies (CSPT), 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] 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

[3] Workshop on the Engineering of Autonomic Systems (EASe), *Proceedings of 11th IEEE International Conference and Workshop on the Engineering of Computer-Based Systems (ECBS'04)*, May 24 - 27, 2004 Brno, Czech Republic, p441.

[4] Sterritt, R. and Bapty, T., (eds.) Engineering Autonomic Systems, special issue of *IEEE Transactions on Systems, Man and Cybernetics*, forthcoming (see http://www.ulster.ac.uk/ease)

[5] Patterson, D.A., Brown, A., Broadwell, P., Candea, G. , Chen, M., Cutler, J., Enriquez, P., Fox, A., Kiciman, E., Merzbacher, M., Oppenhiemer, D., Sastry, N., Tetzlaff, W., Traupman, J., Treuhaft, N., *Recovery-Oriented Computing (ROC): Motivation, Definition, Techniques, and Case Studies*, U.C. Berkeley Computer Science Technical Report, UCB//CSD-02-1175, University of California, Berkeley, March 15, 2002..

[6] Patterson, D., 2002, Availability and Maintainability Performance: New Focus for a New Century, *USENIX Conference on File and Storage Technologies (FAST '02)*, Keynote Address, Monterey, CA, 29 January, 2002.

[7] Ganek, A.G., Hilkner, C.P., Sweitzer, J.W., Miller, B. and Hellerstein, J.L. The response to IT complexity: autonomic computing, *Proceedings 3rd IEEE International Symposium on Network Computing and Applications, (NCA 2004),* Aug. 30 - Sept. 1, 2004, pp 151 – 157

[8] Lawson, H.W., Rebirth of the Computer Industry, *Communications of the ACM,* **45** (6), June 2002.

[9] Gray, J., "Dependability in the Internet Era", http://research.microsoft.com/~gray/talks/InternetAvailability.ppt

[10] Horn, P., Invited Talk to the National Academy of Engineering at Harvard University, March 8, 2001

[11] Patterson, D., "*Recovery-Oriented Computing"*, Keynote, at High Performance Transaction Systems Workshop (HPTS), October 2001

[12] Gray, J., "What Next? A dozen remaining IT problems", Turing Award Lecture, FCRC, May 1999

[13] Internetweek, "*Per Hour Downtime Costs*", 4/3/2000

[14] Kembel, R., Fibre Channel: A Comprehensive Introduction, 2000.

[15] Bantz D.F., "*Autonomic Personal Computing*", presentation at Pace University, White Plains, University of Southern Maine, 25th January 2003.

[16] Microsoft Corporation, "Dynamic Systems Initiative Overview", White Paper, March 31, 2004, revised November 15, 2004.

[17] Truszkowski, W.F., Hinchey, M.G., Rash, J.L. and Rouff, C.A. Autonomous and Autonomic Systems: A Paradigm for Future Space Exploration Missions. *IEEE Trans. on Systems, Man and Cybernetics, Part C*, 2006, to appear.

[18] Truszkowski, W., Rash, J., Rouff, C. and Hinchey, M. Asteroid Exploration with Autonomic Systems. *Proceedings 11th IEEE International Conference on Engineering Computer-Based Systems (ECBS), Workshop on Engineering Autonomic Systems (EASe),* Brno, Czech Republic, 24-27 May 2004, pp 484-489, IEEE Computer Society Press.

[19] The Future of Computing Project; http://www.thefurtureofcomputing.org, 2004

[20] Weiser, M., Creating the Invisible Interface. *Symposium on User Interface Software and Technology*, New York, NY, ACM Press, 1994.

[21] De Roure, D., Jennings, N. and Shadbolt, N. *A Future e-Science Infastructure aka Research Agenda for the Semantic Grid*, EPSRC/DTI Core e-Science Programme, December 2001.

[22] Foster, I., Kesselman, C., Nick, J.M. and Tuecke, S., *The Physiology of the Grid – An Open Grid Service Architecture for Distributed Systems Integration*, June 2002. http://www.globus.org/research/papers/ogsa.pdf

[23] Malik, O., Ian Foster = Grid Computing, *Grid Today*, October 2002.

[24] Corbato, F.J. and Vyssotsky, V.A., Introduction and Overview of Multics System, *Proceedings of AFIPS FJCC*, 1965.

[25] Ledlie, J., Shneidman, J., Seltzer, M. and Huth, J., Scooped, Again. *Proceedings of IPTPS 2003,* Berkeley, CA, February 2003.