Autonomic Computing—a Means of Achieving Dependability?

Roy Sterritt[1] Dave Bustard[2]
Centre for Software Process Technologies (CSPT)
[1]School of Computing and Mathematics
[2]School of Computing and Information Engineering
Faculty of Informatics
University of Ulster
Northern Ireland
{r.sterritt, dw.bustard}@ulster.ac.uk

Abstract

Autonomic Computing is emerging as a significant new approach to the design of computing systems. Its goal is the development of systems that are self-configuring, self-healing, self-protecting and self-optimizing. Dependability is a long-standing desirable property of all computer-based systems. The purpose of this paper is to consider how Autonomic Computing can provide a framework for dependability.

1. Introduction

Dependability is the system property that integrates such attributes as reliability, availability, safety, security, survivability and maintainability [2].

Autonomic Computing has as its goal the engineering of computer-based systems that are self-configuring, self-healing, self-protecting and self-optimizing.

This paper discusses the potential for Autonomic Computing to provide a framework for achieving dependability. It considers Randell's work on establishing a consensus on the meaning, concepts and definitions of dependability [1]. It then proceeds to look at Autonomic Computing in similar terms. A motivation for this approach is a perception that many areas of computing are addressing similar issues without being fully aware of related work in other fields and thus missing potential insights from that work. This is particularly important at this early stage of Autonomic Computing since contributions from a range of disciplines will be needed for its successful realisation.

2. Dependability

For the last 30 years Randell has made a substantial contribution to defining and indeed creating the field of software dependability. In 2000, in his Turing Memorial Lecture [1] "Facing up to Faults", he described software fault tolerance as still somewhat controversial as historically the main software engineering research challenge has been to find ways of developing error-free software, rather than managing faults.

There are strong and active communities in the area—IEEE-CS TC on Fault-Tolerant Computing, IFIP WG 10.4 Dependable Computing and Fault Tolerance as well as communities such as IEEE-CS TC on Engineering of Computer-Based Systems, all of whom consider dependability as a key property of software systems. With such systems becoming ever-more complex there is a growing need for developers to pay greater attention to dependability.

Dependability is defined as that property of a computer-based system that enables reliance to be placed on the service it delivers. That service is its behavior as perceived by other systems or its human users [1].

Figure 1 [1][2], (an update on earlier work [3][4]) depicts the concepts of dependability in terms of threats to, attributes of, and the means by which, dependability is attained.

The effectiveness of these four mechanisms has a substantial influence on the dependability of a computer-based system.

Proceedings of the 10 th IEEE International Conference and Workshop on the Engineering of Computer-Based Systems (ECBS'03)
0-7695-1917-2/03 $17.00 © 2003 IEEE

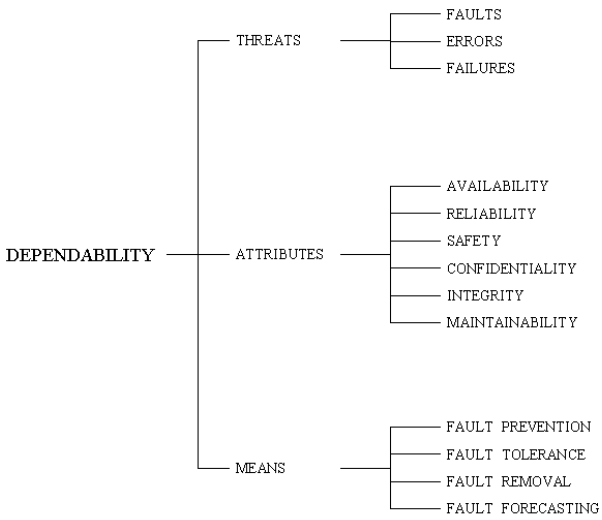

Figure 1 The Dependability Tree

Randell describes dependability in terms of *failures*, *faults* and *errors*, arguing that they follow a "fundamental chain" [1], thus:

… ➔failure ➔ fault ➔ error ➔ failure ➔ fault ➔…

More abstractly, this can be described by the sequence:

… ➔event ➔ cause ➔ state ➔ event ➔ cause ➔…

For example, the failure of a system (event) occurs when a fault is encountered during its operation (cause), because of an error in its implementation (state). This might be attributed to a failure in the test process (event) because the relevant code was not exercised (cause) meaning that the test suite was incomplete (state).

These chains may of course be broken by effective in-built fault tolerance that prevents failure.

Overall, the breadth of issues involved suggests the need for an holistic approach to designing fault tolerant systems.

3. Autonomic Computing

Autonomic Computing, launched by IBM in 2001 [5]-[8], is emerging as a significant new strategic and holistic approach to the design of computing systems. Two of IBM's main objectives are to reduce the total cost of ownership of systems and to find better ways of managing their increasing complexity.

As well as IBM, many major software and system vendors, such as HP, Sun, and Microsoft have established strategic initiatives to help create computer systems that manage themselves, concluding that this is the only viable long-term solution.

As the name implies, the influence for the new paradigm is the human body' s autonomic system, which regulates vital bodily functions such as the control of heart rate, the body's temperature and blood flow—all without conscious effort.

The desire for automation and effective robust systems is not new; in fact this may be considered an aspect of best practice software engineering. Similarly, the desires for systems self-awareness, awareness of the external environment, and the ability to adapt, are also not new, being major goals of artificial intelligence (AI) research for many years. What may be considered new in Autonomic Computing is its overall breadth of vision and scope.

Research in Autonomic Computing is likely to see a greater collaboration between the AI and software engineering fields. Such collaboration has been encouraged by increasing system complexity and a more demanding user community. For example, software engineers have used AI techniques to provide more sophisticated support for user interfaces and to help address soft issues in the development and operation of software. Likewise, the AI community has increasingly been looking to software engineering for disciplined methodologies to support the production of intelligent systems.

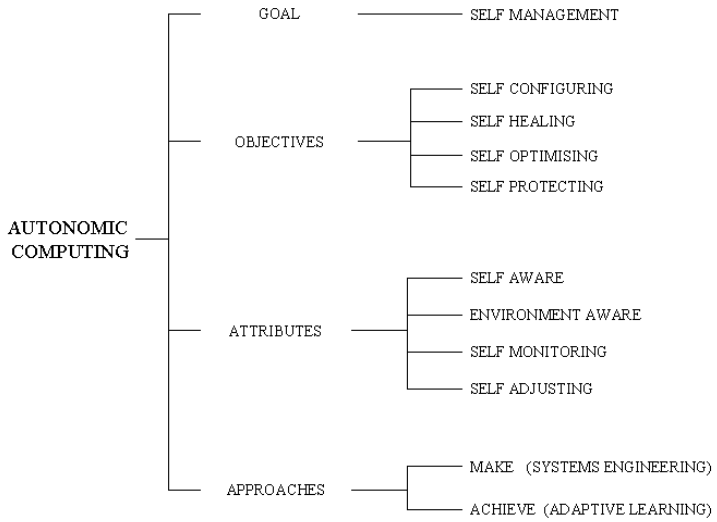

Figure 2 Autonomic Computing Tree

Consequently, Autonomic Computing is perhaps best considered a strategic refocus for the engineering of effective systems rather than a revolutionary new approach [9].

3.1. Attributes of Autonomic Computing

The overall goal of Autonomic Computing is the creation of self-managing systems; these are proactive, robust, adaptable and easy to use. Such objectives are achieved though *self-protecting*, *self-configuring*, *self-healing* and *self-optimizing* activities, as indicated in Figure 2.

To achieve these objectives a system must be both *self-aware* and *environment-aware*, meaning that it must have some concept of the current state of both itself and its operating environment. It must then *self-monitor* to recognize any change in that state that may require modification (*self-adjusting*) to meet its overall self-managing goal. In more detail, this means a system having knowledge of its available resources, its components, their desired performance characteristics, their current status, and the status of inter-connections with other systems.

The ability to operate in a heterogeneous environment requires the use of open standards to understand and communicate with other systems.

In effect, autonomic systems are proactive in their operation, hiding away much of the associated complexity from users.

Self-healing is concerned with ensuring effective recovery under fault conditions, without loss of data or noticeable delays in processing, while identifying the fault and where possible repairing it. Fault prediction techniques might also be used, leading to re-configuration to avoid the faults concerned or reduce the likelihood of their occurrence.

With self-optimization, the system seeks to optimize its operations in both proactive and reactive ways.

With self-protection, a system will defend itself from malicious attack and may also have to self-heal when problems are detected, or self-optimize to improve protection.

With self-configuring, the system may automatically install, configure and integrate new software components seamlessly to meet defined business strategies.

IBM discusses the characteristics or 'elements' of Autonomic Computing in more detail in its manifesto [5].

3.2 Creating Autonomic Computing

The creation of Autonomic Computing, or methods of enabling electronic systems to respond to problems, recover from outages and repair faults, all on their own without human intervention, is a major challenge [7].

While Autonomic Computing represents a new strategy for the IT industry it will be building on existing advanced levels of automation available today. Open standards that can support autonomic behavior include Java, Linux, XML, the Open Source Consortium, Apache, and UDDI.

There are two perceived strategies for introducing Autonomic Computing:

- Making individual systems autonomic
- Achieving autonomic behavior at the global system level.

The first approach can be taken in the near-to-medium term, with human experts generating rules for autonomic functions. Over time, this could be supplemented with self-learning algorithms and processes to achieve autonomic behavior.

The required self-adapting behavior has been classified into three levels by the Smart Adaptive Systems community. These are [10]:

1. Adaptation to a changing environment
2. Adaptation to a similar setting without explicitly being ported to it
3. Adaptation to a new/unknown application

Level 1 would appear to fit best with the 'making systems autonomic' approach, while level 2 would fit with 'achieving autonomic behavior'. An autonomic system has prior knowledge of itself (self-aware) so, for level 3 to match the autonomic model, the constraint of starting from zero knowledge has to be relaxed.

3.3. Autonomic Computing in relation to other Research Initiatives

Autonomic Computing is not only related to the Smart Adaptive Systems field. Several other research areas are also relevant, though often having a different emphasis.

Introspective Computing involves proactive and reactive approaches to improving overall system behavior by sharing and utilizing excess computing, memory, storage and other resources [12]. These are very similar aims to Autonomic Computing's self-configuring and self-optimizing objectives. Kubiatowicz, has commented that Autonomic Computing implies a system reacting to events, whereas introspective computing involves both reactive and proactive behavior [8]. This may appear to be the case from the first Autonomic Computing initiatives, such as IBM's eLiza project, which introduced autonomic functionality into some of IBM's products and services, including DB2 [11]. However, the aim for truly autonomic computing is to encompass proactive behavior as well, for instance through evolutionary learning.

Ubiquitous Computing emphasizes usability. It has compared the current state of computing with early scribes who had to know how to prepare and make a parchment and ink just to be able to write. Autonomic computing will go a long way in making computing systems more usable.

The Ambient Network view of the world is in effect a single system with billions of ' networked information devices' . Although currently the research emphasis for making this a reality is on usability, dependability will increasingly become an issue [1] with a critical role for Autonomic Computing.

4. Autonomic Computing and Dependability

Randell and colleagues [2]-[4] give two main reasons for their interest in and focus on the concepts and definitions of dependability, failures, errors, faults and tolerance. First, there is a need to clarify the subtleties involved. Secondly, and possibly more important, is a desire to avoid dependability concepts being reinvented in other research domains such as safety, survivability, trustworthiness, security, critical infrastructure protection, information survivability, and so on [1]. Often the associated research communities do not realize that they are dealing with different facets of the same concept, and are failing to build on existing research advances and insights [1].

This focus on concepts and definitions is also critical for Autonomic Computing. Research and development from many disciplines will be required and, as already mentioned, the successful integration of AI and software engineering, will be particularly important.

In the IBM manifesto for Autonomic Computing [5] its success is linked to the use of open standards, open source code, and open technologies in general. Yet there is also a need for common concepts and indeed common or open definitions for the researchers from the many disciplines that are going to make Autonomic Computing a reality.

On first consideration, dependability and fault tolerance would appear to be specifically aligned to the self-healing facet of Autonomic Computing. Yet any system that is incorrectly or ineffectively configured and/or inefficiently optimized is likely to lead to failures in the future. Similarly, any system that is not adequately protected is vulnerable to malicious faults, be they from hackers or viruses. Thus, essentially all facets of Autonomic Computing are concerned with dependability.

Referring again to Randell's fundamental chain:

… ➔failure ➔ fault ➔ error ➔…

and its abstract form:

… ➔event ➔ cause ➔ state ➔…

then each facet of Autonomic Computing (Figure 2) can be considered 'states of undependability' or 'states of dependability' according to how well they are addressed in a system.

States of Undependability
Faulty (unhealthy)
Ill-configured
Sub-optimal
Unprotected

That is, if any of these states exist within a system they are liable to lead to subsequent errors; in turn, that may lead to subsequent faults and on to failure. Autonomic Computing, through self-healing, self-configuring, self-optimization and self-protection, with therefore increase dependability.

5. Conclusion

Autonomic computing is an emerging holistic approach to computer system development that aims to bring a new level of automation to systems through self-healing, self-optimizing, self-configuring and self-protection functions.

For Autonomic Computing to reach its goal open standards and technologies are required. This paper has further highlighted the need for open concepts and definitions to facilitate understanding among disciplines. To illustrate that aim the paper specifically discussed autonomic concepts in relation to 'dependability'.

Acknowledgements

This work was undertaken through the Centre for Software Process Technologies, which is supported by the EU Programme for Peace and Reconciliation in Northern Ireland and the Border Region of Ireland (PEACE II).

References

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

[2] A. Avizienis, J.-C. Laprie, B. Randell, "Fundamental Concepts of Dependability", UCLA CSD Report #010028, 2000.

[3] J.C. Laprie, "Dependability: basic concepts and terminology-in English, French, German, Italian and Japanese", In Dependable Computing and Fault Tolerance, p.265, Springer-Verlag, Vienna, 1992.

[4] J.C. Laprie, "Dependable computing: concepts, limits, challenges". In Proceedings 25th IEEE International Symposium on Fault-Tolerant Computing –Special Issue, Pasadena, CA, pp42-54, 1995.

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

[6] E. Mainsah, "Autonomic computing: the next era of computing", IEE Electronics Communication Engineering Journal, Vol. 14, No. 1 (Feb), pp2-3, 2002

[7] A. Wolfe, "IBM sets its sights on ' Autonomic Computing' ", IEEE Spectrum, Jan., p18-19, 2002

[8] L.D. Paulson, "IBM Begins Autonomic Computing Project", IEEE Computer, Feb., p25, 2002.

[9] R. Sterritt, "Towards Autonomic Computing: Effective Event Management", Proceedings of the 27th Annual IEEE/NASA Software Engineering Workshop, Greenbelt, MD, Dec. 2002.

[10] D. Anguita, "Smart Adaptive Systems: State of the Art and Future Direction of Research", European Symposium on Intelligent Technologies, Hybrid Systems and their implementation on Smart Adaptive Systems (EUNITE 2001), Dec. 2001

[11] G.M. Lohman, S.S. Lighstone, "SMART: Making DB2 (More) Autonomic", Proc. 28th VLDB Conf., Hong Kong, China, 2002.

[12] H. Weatherspoon, T. Moscovitz, J. Kubiatowicz. "Introspective Failure Analysis: Avoiding Correlated Failures in Peer-to-Peer Systems", Proceedings of International Workshop on Reliable Peer-to-Peer Distributed Systems, Oct 2002.