The Challenge of Validation for Autonomic and Self-Managing Systems

Thaddeus O. Eze, Richard J. Anthony, Chris Walshaw and Alan Soper
Autonomic Computing Research Group
School of Computing & Mathematical Sciences (CMS)
University of Greenwich
Park Row, SE10 9LS, London, United Kingdom
{T.O.Eze, R.J.Anthony, C.Walshaw and A.J.Soper}@gre.ac.uk

Abstract **-** The research community has achieved great success in building autonomic systems (AS) in line with the vision of autonomic computing (AC) released by IBM in 2001. The success is gaining ground in addressing the perceived concerns of complexity and total cost of ownership of information technology (IT) systems. But we are now faced with a challenge springing from this very success. This challenge is *trustworthiness* and there are limited research results published in this direction. This, if not addressed will definitely undermine the success of AC. How do we validate a system to show that it is capable of achieving a desired result under expected range of contexts and environmental conditions and beyond? This paper identifies the challenges and significance of AS validation and proposes a roadmap towards achieving trustworthiness in autonomic systems.

Keywords-trustworthiness; autonomicity; validation; certification

I. INTRODUCTION

IBM in 2001 observed in a manifesto [1] that the main obstacle to the future growth of IT is a looming software complexity crisis. The innovations in software development have increased exponentially, the sophistication and complexity of system design thereby stretching human capabilities to the limits to install, configure, optimise, maintain and manage these systems. The software is so complex that almost no single person knows everything about the software anymore. Another issue is how upgrades are handled; it is not clear whether or not an upgrade in one part of a system will result in loss of functionalities in other parts that integrate with it.

AC is chosen as a way forward [2]; the idea of building self-managing computing systems in the same fashion as the biological autonomic nervous system using high-level policy objectives set by human administrators. Though complete AS do not yet exist, some products from leading AC enterprises now claim to have self-managing features [3]. We have also seen great level of research interest in AC. A large number of surveys e.g., [4][3][5] have considered the work along a number of criteria and dimensions. In [4] the paper looked at AC to highlight which characteristics are necessary to evaluate and compare AS and derive definitive metrics for the evaluation. The survey in [3] categorizes complexity in IT systems and identifies which AC self-* properties address which complexity. [5] looks at IBM's MAPE-K (monitor, analyze, plan, execute and knowledge) control loop and identifies works that have been done in each of its components and also proposes an alternative to IBM's method of measuring system autonomicity [6]. There is also progress in injecting autonomic capabilities into legacy systems [7]. Other efforts such as those reported in [8][9] focus on using policy-based autonomic techniques to build generic AC frameworks arguing that the integration of techniques gives rather greater flexibility and more powerful adaptation than the individual techniques. With this huge effort devoted to the design and development of ASs, emphasis is lacking on the certification of these systems. We suggest that ASs must reach trustworthy status and be 'certifiable' to achieve the full vision of AC. Appropriate measures for validating AS decision-making processes should be defined. We identify this as the core challenge facing the success of AC. This is our main research focus. Another major problem facing the AC research field is the lack of standards. We have seen proliferation of approaches and the misuse of AC terms –different terms mean different things to different researchers. This shortcoming can only be addressed by standards.

Certification of ASs is a specific work area that needs attention and we believe this can be achieved through defining proper AS validation mechanisms. AC systems are designed and deployed across many application domains to address the challenge of human management complexities. We may come to a point where these systems take over full control of operations in those domains (e.g., businesses, military, health etc.) and any failure can be extremely costly –in terms of down time, danger to life, loss of control etc. This underpins the criticality of AS validation. Robust self-management in AC systems resulting in dynamic changes and reconfigurations requires that ASs should be able to continuously perform self-validation of their own behaviour and configuration, against their high-level behavioural goals and be able to reflect on the quality of their own adaptation behaviour. Such systems are considered trustworthy and then certifiable. It is then necessary to have a testing approach that combines design/run-time elements and is also an integral part of the self-management architecture. We have a longer term vision to develop certifiable systems. By trustworthiness, we mean a state where we can be confident that an AS will remain correct in the face of any possible contexts and environmental inputs and sequences of these; this is achieved through robust validation.

 ISBN: 978-1-61208-134-2

Our motivation is driven by the identified challenge of lack of certifiable ASs. In this paper we present a roadmap towards achieving certifiable AC systems by defining a layered autonomic solution architecture that incorporates validation as an integral part of the self-management structure. In our proposal, we identify the features and define a proper validation approach as one that is generic, cuts across design/run-time, and is an integral part of the whole self-management structure. Currently, most AS are tested in the same way other software is tested: *unit testing*. This includes simulations for performance analysis. In [10] it is suggested that a complete testing plan will require developmental stage-by-stage testing. This approach is limited because it is only design-time based and cannot guarantee trustworthiness. Though autonomic solutions methods establish system policies at design-time, AS must be able to deal with unforeseen conditions (unpredicted at design-time) that might arise during run-time and with this comes the possibility of ASs to deviate from intended behaviour and/or yield inconsistent results. As a result, what is needed is a system of validation that will not only test AS behaviour at design-time but also tests the system's behaviour under environmental circumstances or contexts not predictable at design time. It is not the component's behavior that is of key focus in the sense of being unknown – it is the circumstances in which it executes which is fundamentally unknown – and this may in turn cause unknown behavior in the component we are testing – or indeed in the whole system as a result. Only few works e.g., [11][12] provide means of run-time testing of AS adapted behaviours by introducing self-testing activities to ASs. The main goal of this paper is to outline challenges in current AS validation methods and propose a strategy leading to the achievement of certification of autonomic systems.

The remainder of this paper is organised as follows: Background, significance and challenges of AS validation are discussed in Section II. Analysis of identified validation techniques is presented in Section III. We propose a roadmap towards AS trustworthiness in Section IV and conclude the work in Section V.

II. BACKGROUND OF AS VALIDATION

In this section we define the problem of trustworthiness, identifying its significance and the extent of its challenge. We believe that the ultimate goal of AC should be the certification of AC systems. Yet to achieve certification requires a process and the meeting of some conditions (explained with Figure 1). For unknown reasons and in a bid to get things working faster, the AC research community has concentrated efforts on designs and architecture with little or no emphasis on system validation. Only very few researchers have identified trustworthiness as a major AC challenge and yet fewer [11][12][21] have actually suggested or proposed techniques. The problem in clear terms is the ignoring of AS trustworthiness and the general lack of validation efforts that specifically target the dynamic aspects of these systems.

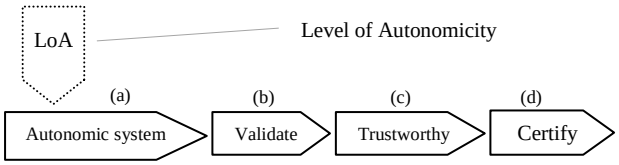

Figure 1: Proposed Certification process and requirements

Figure 1 represents a section in the journey towards full AC. At point (a) we assume that a system is developed and is considered autonomous at some level. This level is determined by a LoA measurement methodology which needs some form of standardization. The definition of LoA at this point is prerequisite to the next step. At point (b) is the system's self-validation distributed across design-time and run-time. When it is ascertained that a system is validated then it is trustworthy and trustworthiness is a vital foundational step on the road towards certification. It then follows that for a system to be certified, it must be trusted and only validated systems can be trusted. We draw a conclusion here that *for an AS to be certifiable there must be a standard for measuring the level and extent of its autonomicity* as it makes no meaning to certify a system whose extent of autonomic capability cannot be measured.

A. Significance of The Problem

The consequence of a lack of validation comes in two dimensions. On the one hand is the risk of losing control and loss of confidence that the autonomic system will not fail. This is obvious owing to the nature of ASs which includes dynamic changes caused by the self-* features in unpredictable environments and conditions. On the other hand is the issue of standardization of AS design processes. It is very unlikely to secure standards for invalidated systems as the general standardisation of a system will largely depend on the level of confirmation of its 'process correctness'. Process correctness is the demonstration of the correctness of a system's behaviour under a range of environmental conditions. According to [13] the requirement for determining process correctness arises from the human fear of selfish and uncontrolled behaviours that potentially might emerge inside self-managed systems. This is yet another factor that underpins the urgent need for standardisation in autonomics. We believe that once there is validation in place to ensure process correctness, standardization will be achieveable. Despite the huge effort in AC there are no known standards in the field [9]. The lack of certification and standards leads to proliferation of designs.

B. Extent of The Challenge of Trustworthiness

Trustworthiness is a broad area that needs extensive investigation and requires a carefully thought-out approach. In this section we look at the challenges of achieving AS trustworthiness and what form a trust solution must take.

 ISBN: 978-1-61208-134-2

- **Level of Autonomicy (LoA):** We have identified in Figure 1 the role of defining LoA in AS certification. LoA is a way of categorising ASs according to degrees; levels of dependences on humans for decision-making. Autonomic systems come in different degrees. Take for instance, one UAV (Unmanned Aerial Vehicle) that has a ground human pilot and one that hasn't are both autonomic systems but of different capacities. Now certifying both systems will demand different requirements as the later system will obviously need to meet higher requirements being more autonomic than the first. Classifying systems will give us an idea of their full capabilities and also identifies what conditions they must meet for certification. It also explains what level or extent of validation that is needed. This point is supported in [14] where it is argued that for a UAV to be certified, the level of its autonomicity should first be established to point out the direction and level of its certification. Currently, there is no agreed method of classifying AS and the only solution to this is standardization of approaches.

- **Design-time and run-time consideration:** As we have identified earlier, validation approaches will need to take care of both design-time and run-time owing to the nature of AC. Validating design-time decision making process does not suffice as systems' decisions that handle evolving conditions also need to be validated. The challenge here is extending validation algorithms to deal with run-time changes that may or may not be anticipated at design. For ordinary software applications where outcomes are as expected or predicted, design-time validation can suffice.

- **Reusability:** Validation approaches should be generic in nature –i.e. approaches should not be specifically defined for given self-adaptation processes. They should be adaptable to different processes. But with LoA in mind, approaches are expected to be generic within levels. What that means is that the reusability of approaches will be restricted according to LoA.

- **Robustness:** Validation solutions must show consistency with the dynamism of the AC environment. In other words solutions should also be autonomic in their approach. Validation approaches are also expected to be integral parts of the AC process.

III. CURRENT APPROACHES TO AS VALIDATION

In this section we survey a cross section of validation and testing approaches. These have been predominantly design-time or laboratory based although the nature of system in question determines, to a large extent, the type of validation or testing needed. Other forms of testing include simulations for performance analysis and self-testing approaches. We group the approaches as follows:

Unit Testing: Unit testing deals with the testing of known testable parts of a system. Usually the tested part, at the point of test, has definite (or well known) functions and outcomes. At this level of testing are laboratory and simulation based testing. [15] has proposed AML (Agent Modelling Language), a visual simulation software that models systems operations and concepts that are multi-agent based. Since AS are multi-agent based, this simulator makes good case for modelling the operations of individual agents (Autonomic Elements –AEs in this case). The paper shows how AML can be used to '*comprehensively*' and '*efficiently*' model the NASA's Prospecting Asteroids Mission (PAM) system. However, this simulation based validation does not suffice for AS trustworthiness as it depends on (or is limited to) the designer's knowledge of the system's environment and operations. Simon Dobson in [16] attempted adaptive network calculus which allows for both design and verification of adaptive systems. In this approach, the description of the adaptive behaviour and its verification are done mathematically; network calculus. This method however is specifically for network functionality. For example, in its expression it is assumed by definition that $R^*(t) \geq R(t)$ for all t. Where $R(t)$ and $R^*(t)$ define, respectively, the sum of bytes received and that of output by a network element at time t. For us this expression can only hold under ideal circumstances (for systems of well known and predictable service curves) but cannot hold in a dramatically different set of circumstances beyond the design-time expected conditions.

Real Life: Testing and validation in a live system is arguably the most accurate way of verifying a system to ensure its compliance with the set system's goal. We understand that not all systems can be exposed to real life validation nonetheless researchers have identified it as one of the main approaches. In [17] the VisLab at the University of Parma, Italy, in seeking for new ways to test (validate) their developed autonomic vehicle for the 2010 World Expo in China, decided to drive their autonomic vehicles (through real-world traffic) 13000Km from Parma to China. Alberto Broggi (VisLab's director) says "*When you do things in the lab, it all really works. But when you go out in the real road, with real traffic, real weather, it's another story.*"

Pervasive Supervision: Pervasive supervision is a monitoring approach proposed in [18] to ensure process correctness of ASs. The supervision system is designed to continuously monitor the (known) configurations of each AE, interpret the monitored data according to certain operations requirements, like functional correctness, performance, consistency etc., and enforce corrective measures in case of requirement(s) violation. An example of pervasive self-supervision is found in [22] in which the policy mechanism monitors its own rate of decision change.

 ISBN: 978-1-61208-134-2

It was found that if a policy is faced with a situation where the utility of two outcomes are very close it is possible for the policy to rapidly switch between these options - effectively adding noise into the system for no additional benefit. By monitoring its own rate of decision change, the policy mechanism is able to detect this instability and temporarily shuts down its 'execute' function whilst continuing to run the 'monitor', 'plan' and 'analyse' components. A similar approach, *model checking*, is found in [19]. It provides automatic analysis of models for adherence to specified properties. A kind of logic is used to specify model properties that should hold during adaptation processes and these properties are automatically checked for adherence so as to provide assurance.

Self-testing: Self-testing is an approach to allow the managed system to carry out self validation of its decision making process. In [11] a framework to validate change requests in ASs is proposed. The approach is based on extending the current MAPE-based autonomic structure to include self-testing as an integral and implicit part of the AS. The same model or structure for AS management using autonomic managers (AMs) is replicated for the self-testing. In the self-test structure, test managers TMs (which extend the concept of AMs to testing activities) implement closed control loops on AMs (such as AMs implement on managed resources) to validate change requests generated by AMs. The work in [11] is extended in [20] to include auxiliary test services components that facilitate manual test management and a detailed description of interactions between the TMs and these new components. [21] proposes a reusable object-oriented design for developing self-testable autonomic software by providing a detailed reusable design for AMs, TMs, touchpoints and also extending the proposed self-testing framework in [11] to include knowledge sources to testing activities in autonomic software. This design (proposed for autonomic software systems) also applies the concepts of AMs. Arguing that these approaches are *not* generic, [12] has proposed a '*generic self-test approach*'.

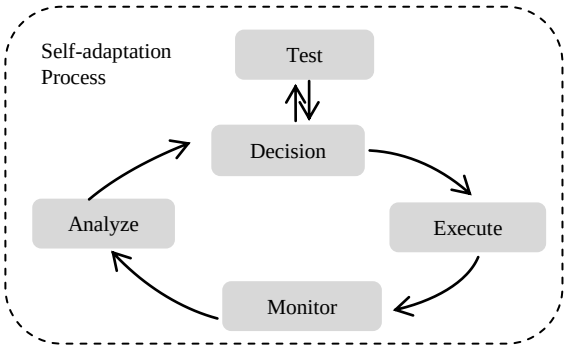

Figure 2: New control loop with test activity [12]

In the structure presented in Figure 2 the authors of [12] extended IBM's MAPE control loop to include a new function called *Test*. By this they define a new control loop comprising Monitor, Analyze, Decision, Test and Execute – MADTE activities. The MADTE loop works like the MAPE loop only that the *Decision* activity calls the *Test* activity to validate a chosen action should it determine to adapt a suggested behaviour. The *Test* activity carries out a test on the action and returns its result to the *Decision* activity which then decides whether to implement, skip or choose another action. (An adaptation is favoured if *Test* indicates that it will lead to component's better performance in terms of characteristics such as optimization, robustness or security.) The process is repeated if the latter is the case. When an action is decided on, the decision activity passes it to the *Execute* activity for implementation. A general method for testing context aware applications, which in a way simplifies the understanding of self-testing in AS is presented in [23]. The paper simplifies the concept of system management using context information while also testing the whole process including the interactions within.

A. Taxonomy of Validation Approaches

Our research has shown that different approaches can be used complementarily depending on what system is being tested and validated. In some cases, for example, using the software environment (simulation) to build a testing setup makes it relatively easier (and complementary) to build models for the real-world testing.

Table 1: Taxonomy of validation approaches

Validation Approach		Generic	Design-time	Run-time	Integrated
Unit Testing	Sim. [15]	√	√	–	–
	Lab. [8] [16]	√	√	–	–
		√	√	–	–
Real World	[17]	√	√	√	NA
Pervasive Supervision / Model Checking	[18]	√	–	√	√
	[19]	–	–	√	–
Self-testing	[11]	–	–	√	√
	[12]	√	–	√	√
	[21]	√	–	√	√

We present the taxonomy of approaches under some selected properties as shown in Table 1. (Note that Approaches and referenced works are selected examples directly related to AS validation and so not exhaustive.). By Generic, we mean approaches that can be adapted to different adaptation processes. *Design-time* and *Run-time* indicate approaches that are design-time based and run-time based respectively. By *Integrated*, we mean approaches that are not separated from the autonomic management architecture. In some senses real world testing is more of testing than validation. It actually shows whether or not a particular AS is able to make appropriate decision(s) in the face of any change but says nothing about scrutinizing the decision(s) before implementing them. In the end, humans are needed to methodologically justify the decision(s) made. In our view, pervasive supervision is more of a component

 ISBN: 978-1-61208-134-2

control paradigm than a validation method. In simple terms, pervasive supervision makes sure that a component takes a defined action (according to specified configurations) when a change occurs (e.g., contextual) but doesn't bother with validating the action taken. The works in [12][21] meet all the properties with the exception of design-time. The difference between [12] and [21] is that [21] defines a separate test loop (consistent with the self-management control loop) and integrates both loops while [12] integrates testing to the self-management control loop. Again the architecture in [21] is more complex than that in [12].

What we have discovered so far is that AS validation is much in its earliest stages with only self-testing as a promising approach amongst all identified approaches. What is then needed is a more robust and less complex self-testing validation methodology for AC systems. Research has also shown that AS validation is still much underdeveloped in some areas e.g., with respect to reusability of validation techniques. Though researchers claim their approaches are generic, it is not yet clear to what extent this is true considering the level of tweaking that needs to be done. In [12] for example, necessary actions to make the proposed framework generic are listed. This can be argued in terms of its robustness and the expected range of application domains (or self-adaptation processes) to be covered. It then follows that techniques for reusable validation is another research area needing attention.

IV. OUR PROPOSED ROADMAP TOWARDS AS TRUSTWORTHINESS

From the ongoing investigation discussed above, we now draw up a roadmap towards AS trustworthiness. This entails our view on how to achieve certifiable AC systems. This is a long road but it is vital that we take steps along this road. This forms the basis of our research strategy. First, we identify characteristics or features, if you like, a proper validation approach in our opinion should possess. Then we look at the inter-related steps towards certifiable Ass.

A. Features of AS Validation Approach

It is our opinion that a proper validation approach should have the characteristics shown in Table 1. ***Generic:*** Reusability reduces complexity and cost (in terms of time and effort) in developing validation processes for AS. A good validation approach should be flexible to be adapted to different adaptation processes and the procedure or process for this adaptation clearly detailed. ***Design/Run-time:*** The dynamic changes and reconfigurations in AS could result in drawbacks such as the possibilities of policy conflicts and incorrect goal specifications. Again it is clear that some AS frameworks facilitate decision-making both at design-time and run-time. It is then necessary to consider testing both at design-time and run-time. ***Integrated:*** In our view testing should be an integral part of the whole self-management architecture. Testing being integrated to the management structure achieves real time validation which is necessary to mitigate adaptation conflicts and promote consistency. ***Automatic***: We emphasize the importance of self-validation. Validation activity should be human independent (i.e. should be triggered by a change in application context, environmental volatility or a locally-detected failure requiring reconfiguration) following a defined validation process. But proving that a validation mechanism actually meets its set requirements is another issue of concern.

B. Towards Certifiable AC Systems

We define a proper validation approach as one that is generic, cuts across design/run-time, and is an integral part of the whole self-management structure. These features can be defined and implemented in a '*class'* architecture, i.e. each feature being seen as a class thereby defining four classes (a – d in Figure 3). This makes the design process flexible as it allows designers to tackle features within the boundaries of their separate classes. Take the *integrated* feature for example; at this class the designer will be concerned with such issues as defining algorithms for components interactions, spontaneous test activity call, etc. One of the ways to achieve robust validation may be through heterogeneity of approaches but the challenge still remains the lack of approaches in this direction. But following from Figure 1 we identify that validation is a process towards certifiable AC systems.

Standardization required

1	**Autonomic System (*defining autonomous*)**	Autonomic characteristics, decision-making algorithms and policies are defined. (Architecture + self-* properties)
2	**Classified AS – according to LoA (*defining autonomicity*)**	Autonomicity measuring metrics are specified.
3	**Tested AS *(Appropriate validation for identified LoA)***	Validation is defined according to system's LoA. Validation is also implemented in a layered structure
4	**Trustworthy AS**	Trustworthy AS is dependable AS. It is not reasonable to consider other properties such as evolvability without first achieving trustworthiness. Validation is prerequisite for trustworthiness
5	**Certifiable AS**	Certifiable AS is at the height of AC goal. It is shown at this point, beyond every reasonable doubt, that a system can be trusted

a	**Generic (*within LoA*)**	Validation approach should be reusable across LoA. Procedure/process for approach to adaptation is clearly specified
b	**Design-time**	Policies that handle design-time validation are defined
c	**Run-time**	Run-time validation policies and algorithms are defined
d	**Integrated**	Algorithms for components interactions and spontaneous (automatic) test activity call are defined

Figure 3: Layered autonomic solution

If we consider (1 - 5) in Figure 3 as layers, each layer is characterized by different attributes. To put it in more context we define three cardinal processes in the path towards certifiable AS; (1) *defining autonomous* – acceptable characteristics that define an autonomic system. This includes issues such as AC architecture (including decision-making algorithms) and self-* properties. The defined characteristics will influence the design and structure of the LoA in layer 2; (2) *defining autonomicity* (LoA) –this is concerned with the whole study of classifying AC systems according to the level of machine or human dependency. This entails measuring or calibrating AS using such metrics as LoA [24]. The designed AS is measured in (2) to determine its autonomic capacity. This gives an idea of the required validation for the system; (3) *validation of systems* –following the capacity of the system an appropriate validation mechanism is mapped out in layer 3 following the validation sub-layer (defined as classes *a* - *d*). These three processes are separate research areas and standards are required at each level. The roadmap identifies that a conceived AS is first designed following a predefined architecture, evaluated according to a set of autonomic characteristics to determine LoA and then validated against its goal. A trustworthy AS is achieved when the design of an AS follows layers 1 to 3. A trustworthy AS is dependable thereby making it certifiable.

V. CONCLUSION AND FUTURE WORK

Designing an autonomic and self-managing system is one thing while validating its management processes is another. Notwithstanding the emergence of products now claiming to have self-managing features, there is very little research effort towards the validation of AS and hence there are no known trustworthy AC systems. We have evaluated some proposed validation approaches against some features which are *generic*, *design-time*, *run-time*, *integrated*, and *automatic*. We have identified the importance of validation and measuring AS level of autonomicity to achieving certifiable AS, outlined challenges in current validation methods and have proposed a roadmap towards certifiable AC systems. As a future work we will be addressing the three cardinal processes identified in Figure 3 which includes autonomic solution architecture, methodology for measuring LoA and developing validation approaches.

VI. REFERENCES

[1] Horn Paul, *Autonomic computing: IBM perspective on the state of information technology*, IBM T.J. Watson Labs, NY, 15th October 2001. Presented at AGENDA 2001, Scottsdale.

[2] J. O. Kephart and D. M. Chess, *The vision of autonomic computing*, In IEEE Computer, volume 36, pp 41–50, January 2003

[3] Mazeiar Salehie and Ladan Tahvildari, *Autonomic Computing: Emerging Trends and Open Problems*, Workshop on the Design and Evolution of Autonomic Application Software (DEAS 2005), St. Louis, Missouri, USA, 2005

[4] J. A. McCann and M. C. Huebscher, *Evaluation issues in Autonomic Computing*, Grid and Corporative Computing (GCC) Workshop, LNCS 3252, pp. 597-608, Springer-V erlag, Birlin Heidelber, 2004

[5] Huebscher M. C. and McCann J. A., *A survey of autonomic computing—degrees, models, and applications*, ACM Computer Survey, 40, 3, Article 7 (August 2008)

[6] IBM Autonomic Computing White Paper, *An architectural blueprint for autonomic computing*. 3rd edition, June 2005, pp 25

[7] Kaiser G, Parekh J, Gross P, and Valetto G., *Kinesthetics extreme: an external infrastructure for monitoring distributed legacy systems*. In: Proceedings of the AC workshop, 5th international workshop on active middleware services (AMS), Seattle. pp 22–30, 2003

[8] Radu Calinescu, *General-Purpose Autonomic Computing*, In: Autonomic Computing and Networking, SpringerLink, 2009

[9] Richard John Anthony, *Policy-centric Integration and Dynamic Composition of Autonomic Computing Techniques*, Int'l Conference on Autonomic Computing (ICAC), June 2007, Jacksonville, FL

[10] Walt Truszkowski, Lou Hallock, Christopher Rouff, Jay Karlin, James Rash, Michael G. Hinchey and Roy Sterritt, Autonomous and Autonomic Syste*ms: with Applications to NASA Intelligent Spacecraft Operations and Exploration Systems*, Springer-Verlag London Ltd, London, 2009

[11] Tariq King, Djuradj Babich, Jonatan Alava, Peter Clarke and Ronald Stevens, *Towards Self-Testing in Autonomic Computing Systems*, Proceedings of the Eighth International Symposium on Autonomous Decentralized Systems (ISADS'07), Arizona, USA, 2007

[12] Andrew Diniz, Viviane Torres and Carlos José, *A Self-adaptive Process that Incorporates a Self-test Activity*, Monografias em Ciência da Computação, No. 32/09, Rio – Brasil, Nov. 2009.

[13] Mikhail Smirnov, Jens Tiemann, Ranganai Chaparadza, Yacine Rebahi and Symeon Papavassiliou, *Demystifying self-awareness of autonomic systems*, Conference Proceedings ICT-MobileSummit 2009, Santander, Spain. Dublin: IIMC, 2009, pp 9.

[14] R. D. Alexander, M. Hall-May and T. P. Kelly, *Certification of Autonomous Systems under UK Military Safety Standards*, ScientificCommons, 2007

[15] Radovan Cervenka, Dominic Greenwood, and Ivan Trencansky, The AML Approach to Modeling Autonomic Systems, International Conference on Autonomic and Autonomous Systems (ICAS'06), Silicon Valley, USA, July 19-21, 2006.

[16] Simon Dobson, *An adaptive systems perspective on network calculus, with applications to autonomic control*, Int. J. Autonomous and Adaptive Communications Systems, Vol. 1, No. 3, pp.332–341. 2008

[17] Erico Guizzo, *Autonomous Vehicle Driving From Italy to China*, IEEE Spectrum tech alert, 23rd Sept. 2010

[18] Luciano B., Matthias B., Maurice M., Chris N., Kevin C. and Peter H., *Towards Pervasive Supervision for Autonomic Systems*, Proceedings of the IEEE Workshop on Distributed Intelligent Systems: Collective Intelligence & Its Applications, 2006

[19] J. Zhang, H. J. Goldsby, and Betty H.C. Cheng. *Modular verification of dynamically adaptive systems*. In Proceedings of the 8th Int'l Conference on Aspect-Oriented Software Development, 2009.

[20] Tariq M. King, Alain E. Ramirez, Rodolfo Cruz, and Peter J. Clarke, *An Integrated Self-Testing Framework for Autonomic Computing Systems*, Journal of computers, vol. 2, no. 9, november 2007

[21] Tariq M. King, Alain Ramirez, Peter J. Clarke, Barbara QuinonesMorales, *A Reusable ObjectOriented Design to Support SelfTestable Autonomic Software*, Proceedings of the 2008 ACM symposium on Applied computing, Fortaleza, Ceara, Brazil, 2008

[22] R. J. Anthony, *Policy-based autonomic computing with integral support for self-stabilisation*, International Journal of Autonomic Computing, Inderscience, Vol. 1, No. 1, pp.1-33. 2009.

[23] Stefan Taranu and Jens Tiemann, *General Method for Testing Context Aware Applications*, Proceedings of the 6th international workshop on Managing ubiquitous communications and services (MUCS), Barcelona, Spain, June 15, 2009

[24] Haffiz Shuaib, Richard J. Anthony and Mariusz Pelc, *Towards Certifiable Autonomic Computing Systems*, Technical report 1, Autonomic Research Group, CMS, University of Greenwich, 2010