A Technique for Measuring the Level of Autonomicity of Self-managing Systems

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

***Abstract*— Autonomic and self-managing systems are increasingly pervasive across an ever-widening spectrum of application domains. Autonomic technology is advancing at a high rate, yet there are no universal standards for the technology itself and the design methods used. There are also significant limitations to the way in which these systems are validated, with heavy reliance on traditional design-time techniques, despite the highly dynamic behaviour of these systems in dealing with run-time configuration changes and environmental and context changes. These limitations ultimately undermine the trustability of these systems and are barriers to eventual certification. This paper is concerned with setting the groundwork for the introduction of standards for Autonomic Computing (AC), in terms of technologies and the composition of functionality as well as validation methodologies. We propose that the first vital step in this chain is to introduce robust techniques by which the systems can be described in universal language, starting with a description of, and means to measure the extent of autonomicity exhibited by a particular system. We present a novel technique for measuring the Level of Autonomicity (LoA) along several dimensions of autonomic system self-CHOP (self-configuration, self-healing, self-optimisation and self-protection) functionalities.**

Keywords- autonomicity; level of autonomicity; autonomic system; trustworthiness; metrics

I. INTRODUCTION

AC seeks the development of self-managing (or autonomic) systems to address management complexities of systems. The high rate of advancement of autonomic technology and methodologies has seen these systems increasingly deployed across a broad range of application domains yet without universal standards. Also the widening acceptance of Autonomic Systems (AS) is leading to more trust being placed in them with little or no basis for this trust, especially in the face of significant limitations regarding the way in which these systems are validated. The traditional design-time validation techniques fail to address the run-time requirements of AS' environmental and contextual dynamism. These limitations undermine trustability and ultimately impinge on certification. The more this proliferation goes on without these challenges being addressed, the more difficult it gets to introduce standards and eventually achieve certifiable AS. It has therefore become pertinent and timely to address these issues. A vital first step in this course would be standards for the universal description of these systems and a standard technique for measuring LoA achieved by these systems. Standards for AC would be concerned with technologies, composition of functionalities and validation methodologies. By autonomicity we mean the ability of a system to pursue its goal with minimal external interference in the form of configuration or control. Then, the extent of this interference defines autonomicity levels. Now the questions facing the AC community are, for a given system, "How autonomic should a system be?" and "How autonomic is a system and how is this determined?" The two questions address both pre and post system design phases. The first question is of primary importance to the designers of systems where autonomic specification is a critical part of the whole system requirements definition. A good example would be spaceflight vehicles addressed in [1], where a *level of autonomy assessment tool* was developed to help determine the level of autonomy required for spaceflight vehicles. The second question is in two parts. On the one hand is the need to define systems according to a measure of autonomicity and another is the method and nature of the measure. Addressing this issue is the main thrust of this paper. Another significant aspect addressed here is the need for evaluation of systems in terms of individual functionalities. Not only do we measure autonomicity but also look at how systems can be evaluated and compared in terms of their autonomic compositions. We call this '*ranking*' (see Section IV B).

Thaddeus *et al* [2] identified that defining LoA is one of the critical stages along the path towards certifiable AS. Along this path also is the need for an appropriate testing methodology that seeks to validate the AS decision-making process. But to know what testing (validation) is appropriate requires knowledge of the system in terms of its extent of autonomicity. Another issue that underpins the need for measuring LoA is that a means of answering the identified questions is also a solution for AS evaluation and ranking and facilitates a proper understanding of such systems.

Currently, the vast majority of research effort in this direction has progressed in answering the first question ("How autonomic should a system be?") by providing us with scales that describe autonomy in systems. These scales, referenced by many researchers, provide fundamental understanding of system autonomy by categorising autonomy according to level of human-machine involvement in decision-making and execution. Some key works in this area include [1], [3], and [4]. For us, these scales only characterise autonomy levels qualitatively and offer no means of quantitatively measuring extent of autonomicity. We would simply say that they are more sufficient for the purposes of proposing an appropriate level of autonomy during the design of a new system.

ISO/IEC 9126-1 standard [13] decomposes overall software product quality into characteristics, sub characteristics (attributes) and associated measures. Adapting

 ISBN: 978-1-61208-187-8

this we define a framework for measuring LoA along several dimensions of AS self-CHOP functionalities. Systems are well-defined by their set of functional capabilities and a measure of these capabilities will form a better representation of the systems. In our proposal we look at the core functionalities of ASs, the self-CHOP functionalities (hereafter referred to as CHOP), and identify specific metrics for each of the functionalities. The cumulative measure of these metrics defines a LoA. Our method is based on the establishment of a generic technique that can be applied to any application domain. This work is novel as it offers a quantitative measure of LoA in terms of system's functionalities-based description. It also opens a new research focus for autonomicity measuring metrics. We believe this is timely because if not addressed we not only run the risk of classifying systems as trusted without basis but also risk losing track and control of these systems as a result of spiraling complexities in terms of technology and methodologies. [15] also raised the concern that if the proliferation of unmanned systems (and by extension ASs) is not checked by putting appropriate measures (or mechanisms) in place that ensure trustworthiness, the systems may ultimately lose acceptance and popularity.

The remainder of this paper is organised as follows: related work is presented in section II. In Section III, we introduce metrics for measuring autonomicity. Our proposed LoA measure and a case study is presented in Section IV. Section V concludes the work.

II. RELATED WORK

The study of AC is now a decade old. However, its rapid advancement has led to a wide range of views on meaning, architecture, and implementations. The criticality of understanding *extent* of autonomicity in defining AC systems has necessitated the need for evaluating these systems. The majority of research in this area has targeted specific application domains with Unmanned Systems Technology (UST) topping the list.

One major proposal for measuring LoA is the scale-based approach. This approach uses a scale of (1– *n*) to define a system's LoA where '1' is the lowest autonomic level usually describing a state of least machine involvement in decision-making and '*n'* the highest autonomic level describing a state of least human involvement. Clough [3] proposes a scale of (1–10) for determining Unmanned Aerial Vehicles' (UAV's) autonomy. Level 1 '*remotely piloted vehicle'* describes the traditional remotely piloted aircraft, while level 10 '*fully autonomous'* describes the ultimate goal of complete autonomy for UAVs. Clough populates the levels between by defining metrics for UAVs. Sheridan [7] also proposes a 10-level scale of autonomic degrees. Unlike Clough's scale, Sheridan's levels 2-4 centre on who makes the decisions (human or machine), while levels 5-9 centre on how to execute decisions. Ryan *et al* [1], in a study to determine the level of autonomy of a particular AS decision-making function, developed an 8-level autonomy assessment tool. The tool ranks each of the OODA (Observe, Orient, Decide and Act) loop functions across Sheridan's proposed scale of autonomy [7]. OODA is decision-making loop architecture for ASs. The scale's bounds (1 and 8) correspond to complete human and complete machine responsibilities respectively. They first identified the tasks encompassed by each of the functions and then tailored each level of the scale to fit appropriate tasks. The challenge here is ensuring relative consistency in magnitude of change between levels across the functions. The levels are broken into three sections. Levels 1-2 (human is primary, computer is secondary), levels 3-5 (computer and human have similar levels of responsibility), and levels 6-8 (computer is independent of human). To determine the LoA needed to design into a spaceflight vehicle Ryan *et al* needed a way to map particular functions onto the scale and determine how *autonomous* each function should be. They designed a questionnaire (sent to system designers, programmers and operators) that considered what they call '*factors for determining LoA*' –these are LoA *trust limit* and *cost/benefit ratio limit*. This implies that a particular LoA for a function is favoured when a balance is struck between trust and cost/benefit ratio limits. Ultimately the pertinent question is "How autonomous should future spaceflight vehicles be?" IBM's 5 levels of automation [4] describes the extent of automation of the IT and business processes. We consider these to be too narrowly defined and [5] observes that the differentiation between levels is too vague to describe the diversity of self-management. In general the autonomy scale approach is qualitative and does not discriminate between behaviour types. We posit that a more appropriate approach should comprise both qualitative and quantitative measures.

Barber and Martin [8] supposes that in a multi-agent system environment, agent autonomy is measured in terms of a system-wide goal. It proposes a collaborative decision-making algorithm for multi-agent systems. In the proposed algorithm, a plan for achieving the system's goal is decided by the agents. Every agent suggests a complete plan with justification for how to achieve the entire system's goal. Each agent evaluates each suggested plan and determines the value of its justification. Each plan receives an integer number of votes from the deciding agents. The plan with the highest votes becomes the plan for the entire system. The ratio of an agent's number of votes (received for suggested plan) to the total number of votes cast is a measure of that agent's autonomy and the extent of its capability to influence the system. This method, however, does not offer a measure for LoA but gives a valuable description of agents' individual influence in a multi-agent system environment.

Fernando *et al* [6] proposes measures for evaluating the autonomy of software agents. It believes that a measure of autonomy (or any other agent feature) can be determined as a function of well-defined characteristics. Firstly it identifies the agent autonomy attributes (self-control, functional independence, and evolution capability) and then defines a set of measures for each of the identified attributes. By normalising the results of the defined measures using a set of functions, the agent's LoA is defined. [6] considers autonomicity measure with reference to system's

 ISBN: 978-1-61208-187-8

characteristics and attributes. But in that work *'characteristics'* are a broad range of attributes that describe a system which also include features outside the system's core functionalities and so in a way are vague and limited in offering a proper and conclusive description of that system. We have adapted this approach in our proposal for ASs but with reference to self-CHOP functionalities.

III. AUTONOMICITY MEASURING METRICS

In this section, we introduce metrics for each of the functionalities that define autonomicity of AS. Though metrics are application domain dependent, the metrics presented here are generic and serve as examples. We present at least one metric for each of the functionalities. This is part of a wide (and separate) research focus. This section only focuses on how autonomic metrics can be generated. We also show how metrics can be normalised to yield autonomic values (see Section IV A). We will start with a definition of each CHOP. (For more on these definitions see [10] and [12]).

***Self-Configuring*:** A system is self-configuring when it is able to automate its own installation and setup according to high-level goals. When a new component is introduced into an AS it registers itself so that other components can easily interact with it. The extent of this interoperability (I) is a measure of self-configuration, measured as ratio of the actual number of components (${}_{actual}n_i$) successfully interacting with the new component (after configuration) to the number of components expected (${}_{expected}n_i$) to interact with the new component.

$$I = \sum_{1}^{i} \frac{{}_{actual}n_i}{{}_{expected}n_i} \tag{1}$$

Interoperability ratio I measures to what extent a system is distorted by an upgrade. A system is self-configuring to the extent of its ability to curb this distortion. This example can be related to the problem diagnosis system for AS upgrade discussed in [10]. Here an upgrade introduces 5 software modules. The installation regression testers found faulty output in 3 of the new modules. This implies that only 2 modules out of 5 successfully integrate with the system.

***Self-Optimising*:** A system is self-optimising when it is capable of adapting to meet current requirements and also of taking necessary actions to self-adjust to better its performance. Resource management (e.g., load balancing) is an aspect of self-optimisation. An AS is then required to be able to learn how to adapt its state to meet the new challenges. Also needed is consistent update of the system's knowledge of how to modify its state. State is defined by a set of variables such as current load distribution, CPU utilization, resource usage, etc. The values of these variables are influenced by certain event occurrences like new requirements (e.g., process fluctuations or disruptions). By changing the values of these variables, the event also changes the state of the system. The status of these variables is then updated by a set of executable statements (policies) to meet any new requirement. A typical example would be an autonomic job scheduling system. At first, the job scheduler could assign equal processing time quanta to all systems requiring processing time. The size of the time quantum becomes the current state and as events occur (e.g., fluctuations in processing time requirement, disruptions of any kind, etc.), the scheduler is able to adjust the processing time allocation according to priorities specified as policies. In this way the state of the system is updated. But this may lead to erratic tuning (as a result of over or under compensation) causing instability in the system. We define *Stability* as a measure of self-optimisation. If an event leads to erratic behaviour, incoherent results or system not been able to retrace its working state beyond a certain safe margin (a margin within which instability is tolerated) then the system is not effectively self-optimising.

***Self-Healing*:** A system is self-healing when it is able to detect errors or symptoms of potential errors by monitoring its own platform and automatically initiate remediation [11]. Fault tolerance is one aspect of self-healing. It allows the system to continue its operation possibly at a reduced level instead of stopping completely as a result of a part failure. One critical factor here is latency; the amount of time the system takes to detect a problem and then react to it. We define *reaction time* T as a metric for self-healing capability. This is crucial to the reliability of a system. If a change occurs at time t_a and the system is able to detect and work out a new configuration and ready to adapt at time t_b then (2) defines the reaction time T. (Average is taken instead where variations of T are possible).

$$T = t_b - t_a \tag{2}$$

A case scenario is a stock trading system where time is of paramount importance. The system needs to track changes (e.g., in trade volumes, price, rates etc.) in real time in order to make profitable trading decisions.

Self-Protecting: A system is self-protecting when it is able to detect and protect itself from attacks by automatically configuring and tuning itself to achieve security. It may also be capable of proactively preventing a security breach through its knowledge based on previous occurrences. While self-healing is reactive, self-protecting is proactive. A proactive system, for example, would maintain a kind of log of trends leading to security problems (threats and breaches) and a list of solutions to resolve them (a list of problems and corresponding solutions only applies to self-healing). One major metric here is the ability of the system to prevent security issues based on its experience of past occurrences. For example let's assume $p \in \{p_{ij}\}$ to be true if i^{th} trend leads to j^{th} problem where p_{ij} is a log of all identified trends and corresponding problems. p is a particular instance of trend-problem combination. A self-protecting manager will avoid a situation of same trend leading to the same problem again by blocking the problem, addressing it or preventatively shutting down part of the system. We define *ability to detect repeat events* R as a self-protecting metric. R is a Boolean value (True indicates the manager is able to stop a repeating

problem and False otherwise). If we choose two samples of $\{p_{ij}\}$ at different times (t_1 and t_2) then (3) defines R. (Different trends may lead to the same problem but a repeated trend-problem combination indicates a failure of the system to prevent a reoccurrence).

$$R = True \ \forall_{ij} \ if \ \{p_{ij}\}_{t1} \cap \{p_{ij}\}_{t2} = \varnothing \quad (3)$$

One typical implementation of this is an antivirus system. Some antivirus systems learn about trends or patterns (signatures) and are able to make decisions based on this to proactively protect a system from an attack. The antivirus is able to stop repeatable patterns. Detecting problem reoccurrence is an active research focus in Autonomic Computing [18].

IV. PROPOSED LOA MEASURE

An AS is defined based on its achievement of the CHOP capabilities [11]. In our approach, we define a level of AS in terms of its extent of achieving the identified functionalities. (We understand that these functionalities may overlap i.e., are not necessarily orthogonal, thus some algorithms may influence several functionalities, but to make progress in this area we assume orthogonality for this preliminary work). If a system fails to provide at least a certain level of one of the CHOP, the system is said to be non autonomic. On the other hand if the system provides a full level of all the four capabilities, it is said to have achieved full autonomicity (as defined by our proposed scheme). Each functionality is defined by a set of metrics. An autonomic value contribution is assigned to each functionality which is spread across the set of metrics for that functionality. It then follows that each metric contributes a certain definite level of the assigned value. The cumulative normalisation of the measure of all metrics (for all functionalities) defines a LoA. Let the maximum LoA value for an AS be M. In generic terms, this will mean an AS having a LoA value of N ($0 \leq N \leq M$) and each functionality contributing a value in the range ($0 \leq x \leq M/4$), while each metric of each functionality contributes ($M/4$)/m ($m \neq 0$) autonomic value, where m is the number of identified metrics defining a particular functionality, and the constant 4 represents four CHOP functionalities. (With an ongoing debate on the composition of AS functionalities and the list substantially growing [16, 17], we choose to limit it to the original, and generally accepted four).

Given that any AS is defined by the four autonomic functionalities, the expression (4) is the representation of the possible combinations of the functionalities.

$$\sum_{r=1}^{4} {}^{n}C_r \quad \Rightarrow 16 \text{ Combinations} \quad (4)$$

This will give 15 possible combinations (excluding zero value which is a special case and not considered further as it means the system provides no autonomic functionality) where (n = 4) is the number of functionalities (the CHOP) and r is a category of the possibilities (a specific implementation combination of the functionalities). The CHOP functionalities may not be of equal importance to an application domain so categories indicate what CHOP is important to an application domain. Category 2 means that only two functionalities are of importance to the system's domain –so for example {C, H, not O, not P} is a specific category representing a system indicated by line 4 in Figure 1.

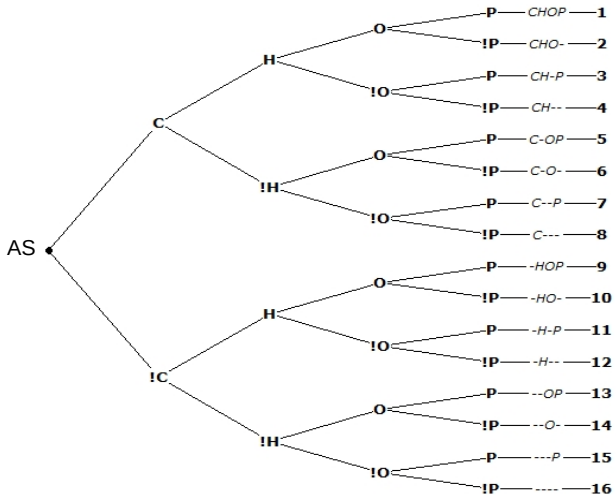

Figure 1: Combination of autonomic functionalities.

Figure 1 implies that, in terms of autonomic functionality composition, a system deemed autonomic (an AS) can be defined (or described) in one of fifteen ways. Each trace of line from start to finish represents an AS except line 16. If we define autonomic metrics for each of the functionalities, then the sum of the autonomicity in each of the constituent functionalities for a particular AS gives the system's LoA (5). For example, the LoA of a system represented by line 9 in Figure 1 will be the summation of the autonomic metrics defining the self-healing, self-optimising and self-protecting functionalities.

$$LoA = \sum_{i=1}^{m_c}[c_i] + \sum_{j=1}^{m_h}[h_j] + \sum_{k=1}^{m_o}[o_k] + \sum_{l=1}^{m_p}[p_l] \quad (5)$$

Subscripted m is the number of identified metrics for the respective functionalities. c_i, h_j, o_k and p_l are the autonomic metric contributions of the functionalities. These can be composed of functions of different measures but as explained in Section III they are normalised to yield autonomic values.

A. Normalising Autonomic Metrics

Depending on the application domain, metrics could be scalar (of different measures) or non scalar values (e.g., observing a capability). One challenge here is defining and normalizing appropriate autonomic metrics. The metrics' values (irrespective of units of measure) are normalized into real numbers that are summed to give LoA (N). We identify two simple methods for normalization: 1) By ranking values according to *high*, *medium*, and *low*. The meaning of this ranking is metric-dependent and is based on a defined margin. For example, if a maximum expected value is 6, a value of 0-2 will be ranked *low*, while 3-4 will be ranked *medium* and 5-

6 *high*. A medium value would contribute fifty percent of the metric's autonomic value contribution of (*M/4*)/*m*, while the two extremes would contribute zero and hundred percents. This can be used for scalar metrics like the *interoperability ratio* and *reaction time* metrics discussed in Section III. 2) By having a Boolean kind of contribution where two values can suggest two extremes –either affirming a capability or not. For example, if a '*True*' outcome affirms a capability then it contributes hundred percent of the autonomic value contribution, while a '*False*' outcome contributes zero. Another example in this category is where an instance of an event either does or doesn't confirm a capability (e.g., the *stability* metric for self-optimising).

B. Evaluating Autonomic Systems

Evaluating Autonomic Systems using (5) gives their separate LoA values. Systems are classified according to categories. This is in terms of what CHOP functionalities are required in their specific application domains. One thing remains to be clarified at this point –'how do we rank each functionality in the autonomic composition of a system?' This can be in terms of importance or extent of functionality provided. We focus on the later –the extent of functionality provided as against what is needed. Take for instance, if two systems are of the same category we may wish to know which of them provides a greater degree of say self-healing or self-protection in any application domain. To address this we adapt a function that measures agent's decision-making power in a multi-agent AS defined in [8]. The rank *R* of a functionality in the autonomic composition of a system is defined by the ratio of its autonomic contribution *x* to the total autonomic contribution of all metrics defining the composite functionalities of that system.

$$R = \frac{x}{LoA} \tag{6}$$

where *x* is the autonomic contribution of the considered functionality which could be the summation of c_i, h_j, o_k or p_l as in (5). With (6) any composite functionality can be ranked in terms of their autonomic contribution. (See case study).

C. Autonomic Systems Evaluation Case Study

Our case study is *Dynamic Qualitative Sensor Selection System* (DQSSS), based on work in [14]. The goal of DQSSS is to *dynamically select a sensor (amongst many) based on continuously variable qualitative characteristics* (e.g., signal quality and noise levels). This is typical of an application that accesses several sensors generating raw data from monitoring a particular context; these could be physical attributes of a system or perhaps information feeds from a service (e.g. financial data). In such applications, it is expected that a DQSSS would generate and differentiate signal characteristics and trends, choose the best signal and without compromising stability, be continuous, unsupervised, dynamic, and detect and react if a sensor goes down. Autonomic metrics are drawn from these characteristics. By definition self-configuration, optimization and healing are of importance to this system (*r=3*). The DQSSS presented in [14] is in three stages which we refer to as systems A, B and C. All three systems are able to differentiate sensors by their signal characteristics such as noise level and spikes. These are then combined in a *utility function* to determine the better quality sensor. Systems B and C are able to generate trends in signal quality using *trend analysis* logic. Only system C ensures stability (avoiding unhealthy oscillation in sensor selection) by implementing *dead zone* logic, while none of the systems has a way of detecting a failed sensor.

TABLE I: REPRESENTATION OF THE DQSSS [14]

Characteristics (metrics)	**Contributing CHOP**	**Sys A**	**Sys B**	**Sys C**
Continuous	C	√	√	√
Unsupervised	C	√	√	√
Trends examination	O	-	√	√
Stability	O	-	-	√
Dynamic (logic switching)	O	-	-	√
Signal characteristics	C	√	√	√
Signal differentiation	C	√	√	√
Failure sensitivity (sensors)	H	-	-	-
Robust (fault tolerance)	H	-	-	√

We adopt the 8-level autonomy assessment scale in [1] as a way of qualitatively interpreting our results. In keeping with this we adopt the arbitrary value 8 as the maximum LoA implying that each CHOP contributes an autonomic value in the range ($0 \le x \le 2$) spread across its metrics. Normalizing the identified metrics in Table I (the numbers of metrics in each category are: C=4, H=2, O=3) in the autonomic value range ($0 \le x \le 2$) gives the result in Table II.

Figure 2 is a radar chart analysis of systems A, B and C in terms of their separate autonomic functionality composition. Recall that only three functionalities (CHO-) are of importance here which means maximum LoA value of 6. Out of this maximum value systems A, B and C achieved the values 2 (i.e., 33%), 2.67 (i.e., 45%) and 5 (i.e., 83%) respectively. This means that in a dynamic sensor selection application domain (as defined), system C can be depended upon to carry out the task with a confidence level of 83% and 0.17 risk factor, B with 45% and 0.55 risk factor, while A with 33% and 0.67 risk factor. Furthermore this can also be interpreted using Ryan *et al*'s level of autonomy assessment scale [1]. The scale as explained in Related Work section is an 8-level autonomy assessment tool used for either identifying (qualitatively) the level of autonomy of an existing system or for proposing an appropriate level of autonomy during the design of a new system. System A falls within level 2 of the scale which points to a situation where '*computer shadows human*' in the self-management process. This indicates that system A only has a narrow envelope of environmental conditions in which it is both autonomic and returns satisfactory behaviour. System B tends toward level 3 on the scale which is '*human shadows computer*' which translates into a wider operational envelope, but once the limits of that envelope are reached human input is needed in the form of retuning, or manual override in the case of oscillation, which for example system C can deal with autonomicaly. System C falls within level 5, which points to

 ISBN: 978-1-61208-187-8

'*collaboration with reduced human intervention*'. This indicates that C is sufficiently sophisticated to operate autonomicaly and yield satisfactory results under almost all perceivable operating circumstances.

Employing (6) to rank the functionalities and taking just self-configuration for example, we find that in system A self-configuration contributes 100% of its autonomic achievement, while in systems B and C the contribution is 75% and 40% respectively. This shows that system A is entirely a self-configuring system, while system C is more of a self-optimising system than B.

TABLE II: ANALYSIS RESULT

	Sys A	Sys B	Sys C
C	2	2	2
H	0	0	1
O	0	0.67	2
LoA	**2**	**2.67**	**5**

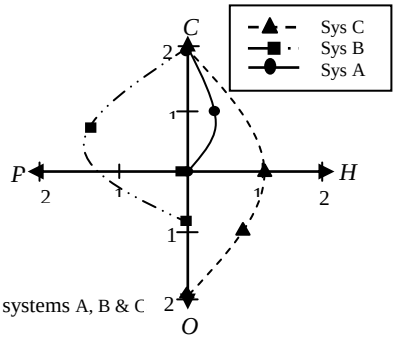

Figure 2: LoA representation of systems A, B & C in the four CHOP domains.

The benefit of analyzing Autonomic Systems in terms of their extent of autonomicity not only offers a path to Autonomic Systems' certification as stated earlier, it also offers a way of comparing these systems, and also facilitates a proper description of these systems to users.

V. CONCLUSION AND FUTURE WORK

A system is better defined by its capabilities and so measuring the LoA of Autonomic Systems without a reference to autonomic functionalities would be inaccurate. We have proposed a CHOP-based LoA measurement. In our proposal a typical AS is defined by the four CHOP functionalities (self-configuring, -healing, -optimising and -protecting) and LoA is measured with respect to these functionalities. Each functionality is defined by a set of metrics. The metrics values are normalised and aggregated to give the autonomic contribution of each functionality which are then combined to yield a LoA value for an AS. We have adopted the maximum autonomic value of 8 to correspond with the autonomy assessment scale defined in [1] to enable a qualitative understanding of the quantitative LoA measure proposed here. We have also shown how systems can further be evaluated by looking at the ratio of autonomic contributions of their separate functionalities. In this, we found that only systems of the same autonomic categorisation can be compared (e.g., a space exploration system cannot be directly compared with a resource allocation system as both are uniquely defined in terms of context and functionalities).

The standardization of a technique for the measurement of LoA will bring many quality-related benefits which include being able to compare alternative configurations of ASs, and even to be able to compare alternate systems themselves and approaches to building ASs, in terms of the LoA they offer. This in turn has the potential to improve the consistency of the entire lifecycle of Autonomic Systems and in particular links across the requirements analysis, design and acceptance testing stages.

As future work, we are looking at exploring areas where the CHOP are not orthogonal and also how to properly define and generate autonomic metrics to strengthen our framework. This is a key component towards our wider research which focuses on the challenge of validating AC systems to achieve trustworthiness in Autonomic Systems.

REFERENCES

[1] Ryan W. Proud, Jeremy J. Hart, and Richard B. Mrozinski. *Methods for Determining the Level of Autonomy to Design into a Human Spaceflight Vehicle: A Function Specific Approach*. http://handle.dtic.mil/100.2/ADA515467 accessed 28/03/11

[2] Thaddeus O. Eze, Richard J. Anthony, Chris Walshaw and Alan Soper. *The Challenge of Validation for Autonomic and Self-Managing Systems.* In proceedings of The 7th International Conference on Autonomic and Autonomous Systems (ICAS), May 22-27, 2011 – Venice/Mestre, Italy

[3] Clough B T. *Metrics, Schmetrics! How The Heck Do You Determine A UAV's Autonomy Anyway?* In Proceedings of PerMis Workshop, pp 1–7. NIST, Gaithersburg, MD, 2002.

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

[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] Fernando Alonso, José Fuertes, Löic Martínez, and Héctor Soza. *Towards a Set of Measures for Evaluating Software Agent Autonomy*. In proceedings of 8th Mexican Int'l Conference on Artificial Intelligence (MICAI), 2009

[7] Sheridan T. B.. *Telerobotics, Automation, and Human Supervisory Control*. The MIT Press. Cambridge, MA, USA 1992. ISBN:0-262-19316-7

[8] Barber, K. S. and Martin, C. E.. *Agent Autonomy: Specification, Measurement, and Dynamic Adjustment*. In Proceedings of the Autonomy Control Software Workshop at Autonomous Agents 1999 (Agents'99), 8-15. Seattle

[9] Hui-Min Huang, Kerry Pavek, James Albus, and Elena Messina. *Autonomy Levels for Unmanned Systems (ALFUS) Framework: An Update*. In proceedings of SPIE Defense and Security Symposium, Orlando, Florida. 2005

[10] Kephart J., Chess D. *The Vision of Autonomic Computing*. Computer, IEEE, Vol 36, Issue 1, 2003, pp 41-50

[11] Bantz D. F. Bisdikian, C. Challener, D. Karidis, J. P. Mastrianni, S. Mohindra, A. Shea, D. G. and Vanover, M.. *Autonomic Personal Pomputing*. IBM Systems Journal, Vol 42, No 1, 2003

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

[13] ISO/IEC 9126-1:2001(E), Software engineering — Product quality — Part 1: Quality model

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

[15] Gaea Honeycutt, *How Much Do we Trust Autonomous Systems?* Unmanned Systems -2008

[16] H. Tianfield. *Multi-agent Based Autonomic Architecture for Network Management*. In Proc. IEEE International Conference on Industrial Informatics, pp. 462–469, 2003

[17] W. Truszkowski, L. Hallock, C. Rouff, J. Karlin, J. Rash, M. G.Hinchey, and R. Sterritt, *Autonomous and Autonomic Systems*. Springer, 2009

[18] Mark B., Sheng M., Guy L., Laurent M., Mark W., Jon C. and Peter S., *Quickly Finding Known Software Problems via Automated Symptom Matching*, The 2nd International Conference on Autonomic Computing (ICAC), 2005, Seattle, USA