A Generic Approach towards Measuring Level of Autonomicity in Adaptive Systems

Thaddeus Eze, Richard 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*— This paper is concerned with setting the groundwork for the introduction of standards for Autonomic Computing, in terms of technologies and the composition of functionalities as well as validation methodologies. This is in line with addressing the lack of universal standards for autonomic (self-managing) systems and design methods used for them despite the increasingly pervasiveness of the technology. There are also significant limitations to the way in which these systems are assessed and 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. 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. Existing techniques have mainly qualitatively classified autonomic systems according to some defined levels with no reference to the building blocks (core functionalities) of the systems. In this paper we present a novel and generic technique for measuring the Level of Autonomicity along several dimensions of autonomic system self-* (e.g., self-configuration, self-healing, self-optimisation and self-protection) functionalities. To demonstrate the feasibility and practicability of our approach, a case example of two different scenarios is examined. One example focuses on a specific case approach for LoA measure within a Dynamic Qualitative Sensor Selection scenario. The second example is a deployment of a generic case approach to an envisioned Autonomic Marketing System that has many dimensions of freedom and which is sensitive to a number of contextual volatility.**

Keywords - autonomicity; level of autonomicity; autonomic system; trustworthiness; metrics; autonomic marketing, sensor selection

I. INTRODUCTION

Autonomic Computing (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 Level of Autonomicity (LoA) achieved by these systems –and we have made progress in this area [1]. 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 the spaceflight vehicles addressed in [2], 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 and here we improve on our initial work [1] in this area. Another significant aspect addressed here is the need for a standard way for assessing, comparing and evaluating different systems (with flexibility across many domains) and also in terms of their 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.

Eze *et al* [3] 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 assessing AS 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 and analyse 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. A naturally upcoming concern with this approach is that high human involvement does not always necessarily translate to low autonomicity and vice versa. Also, most (if not all) of such approaches do not assess ASs based on demonstrated functionalities but on perceived or observed outcomes (performance). Some key works in this area include [2], [4], and [5]. For us, these scales only characterise autonomy levels qualitatively and offer no generic or robust 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 [6] decomposes overall software product quality into characteristics, sub characteristics (attributes) and associated measures. Adapting this, we define a framework for measuring LoA along several dimensions of AS self-* 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. These functional capabilities may be extended to mean, in other systems, characteristics (or attributes) and sub-characteristics (or sub-attributes). While in our initial work [1] we restrict the functionalities to the core functionalities of ASs, the self-CHOP (self-configuration, self-healing, self-optimisation and self-protection) functionalities, in this paper we extend the reach (scope) to cover all possible essential functionalities and identify specific metrics for each of the functionalities. (This allows the approach to be entirely more generic.) 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 and can be flexibly applied across different application instances. 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. [7] 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 two case studies are presented in Sections IV and V respectively. Section VI 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 datacentre applications topping the list [8]. Now, to the extent of our research review [8], there is no known (or published) quantitative approach for assessing autonomic systems. There are nonetheless, efforts towards classifying ASs according to *extent of autonomicity* but these efforts have not successfully met the need for assessing autonomic systems. In this section we review some of the proposed (existing) approaches.

One major proposal for classifying ASs according to *extent of autonomicity* (or measuring LoA) is the *scale-based* approach. This approach, based on level of human-machine involvement in decision-making and execution, uses a scale of (1 – *max*) to define a system's LoA where '1', the lower bound, is the lowest autonomic level usually describing a state of least machine involvement in decision-making and '*max'*, the upper bound, is the highest autonomic level describing a state of least human involvement. Prominent in this category of approach are efforts in [2], [4], [9, 27], and [20]. Clough [4] 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 [9] 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* [2], 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 [9]. OODA is a 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 level of autonomy needed to design into a spaceflight vehicle, Ryan *et al* [2] needed a way to map particular functions onto the scale and determine how *autonomous* each function should be. They designed a questionnaire and sent it to system designers, programmers and operators. The questionnaire considered what they call '*factors for determining level of autonomy*', which include level of autonomy *trust limit* and *cost/benefit ratio limit*. This implies that a particular level of autonomy 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?" This is a brilliant technique towards answering the first identified question ("How autonomic should a system be?") IBM's 5 levels of automation [5] describes the extent of automation of the IT and business processes. We consider these to be too narrowly defined and [10] observes that the differentiation between levels is too vague to describe the diversity of self-management, making it difficult to align ASs with those levels [28]. One major concern with the *scale-based* approach is that a system is not necessarily less autonomic when human interferes with its operations and vice versa. Another is the complexity of applying the approach across different application instances (systems) –this is in terms of populating the levels in-between the scales: the differentiation between levels is complex (and can vary significantly depending on who is using the approach) to determine appropriate magnitude for each level. 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 (as a way of assigning magnitude or value to the description and classification of systems) measures. These concerns are considered and addressed in our approach.

Hui-Min *et al.* [20] is a government's front for addressing the challenge of classifying the pervasive unmanned systems (UMS) according to their levels of autonomy. [20] alludes that UMS' autonomy cannot be rightly evaluated quantitatively without thorough technical basis and that the development of autonomy levels for unmanned systems must take into account factors like task complexity, human interaction, and environmental difficulty. The product in [20] is Autonomy Levels for Unmanned Systems (ALFUS) Framework which, more specifically, provides the terminology for prescribing and evaluating the level of autonomy that an unmanned system can achieve. The framework, in which the levels of autonomy can be described, addresses the technical aspects of UMS and includes terms and definitions (set of standard terms and definitions that support the autonomy level metrics), detailed model for autonomy levels, summary model for autonomy levels, and guidelines, processes, and use cases. While we accept that autonomicity cannot be correctly evaluated without thorough technical basis, our approach further takes into account key functionalities of ASs rather than individual breakdown of technical operations and operational conditions –a major difference with our work. The work in [20], which is updated in [21], focuses more on standardised categorisation of UMS.

Barber and Martin [11] 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 which is useful to our approach: In further evaluating a system, we adapt this formula to determine the rate of individual functionality contribution in our proposed LoA measure (see Section IV B).

Fernando *et al* [12] 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 (as *self-control*, *functional independence*, and *evolution capability*) and then defines a set of measures for each of the identified attributes. The agent's LoA is defined by normalising the results of the defined measures using a set of functions. [12] considers autonomicity measure with reference to system's 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. Not going into the argument of right/wrong constitution of system attributes (or functionalities), the important aspect to note is the idea of defining a system with respect to its attributes and characteristics. We have adapted this approach in our proposal for autonomic systems but with reference to [core] autonomic self-* functionalities.

III. AUTONOMICITY MEASURING METRICS

In this section, we introduce example metrics for each of the core four functionalities that define autonomicity of AS. Though metrics are application domain dependent, the metrics presented here are generic and serve as examples only. We understand that autonomic functionalities are emergent and these vary (or are defined) according to application instances. The point is that, for any system (whether or not autonomic), there are required functionalities (determined by designers and/or users) which can be measurable by some identified metrics. We present at least one metric for each of the functionalities (using the self-CHOP for example). This is part of a wider (and separate) research focus. This section only focuses on how autonomic metrics can be generated. We also show how metrics can be normalised (see Section IV). We will start with a definition of each CHOP. (For more on these definitions see [13] and [14]).

***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 the ratio of actual number of components ($n_{i_{actual}}$) to expected number of components ($n_{i_{expected}}$) successfully interacting with the new component after configuration.

$$I = \sum_{1}^{i} \frac{n_{i_{actual}}}{n_{i_{expected}}} \quad (1)$$

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 [13]. 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 integrated 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 autonomic system is required to be able to learn how to adapt its state to meet 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 sizes 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 is not 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 performance and automatically initiate remediation [15]. 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).

$$T = t_b - t_a \quad (2)$$

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* E as a self-protecting metric. E 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 E. (Different trends may lead to the same problem but a repeated trend-problem combination indicates a failure of the system to prevent a reoccurrence).

$$E = True\ \forall_{ij} \text{ if } \{p_{ij}\}t_1 \cap \{p_{ij}\}t_2 = \emptyset \quad (3)$$

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 these 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 [16].

IV. PROPOSED LOA MEASURE

An AS is defined based on its achievement of the self-* capabilities [15]. In our approach, we define a level of AS in terms of its extent of achieving the identified functionalities. If a system fails to demonstrate at least a certain level of one of the self-* (required for the system in question), the system is said to be non-autonomic. On the other hand, if the system demonstrates a full level of all identified (or required) capabilities, it is said to have achieved full autonomicity (as defined by our proposed scheme). In this section, we present our updated approach towards measuring autonomic systems LoA. In the most part, a mathematical algorithm is used for the proposed approach.

Each functionality is defined by a set of metrics. Each functionality contributes a level of autonomic value which is spread across the set of metrics for that functionality. It then follows that each metric contributes a certain quota of the autonomic value for that functionality. Metrics and functionalities are weighted according to relevance or importance. The cumulative normalisation of the measure of all metrics (for all functionalities) defines a LoA. The need for normalisation of values enables comparison of systems across different implementations. With an ongoing debate on the composition of AS functionalities and the list substantially growing [17, 18], our approach is generic to accommodate evolving functionalities as may be defined by the user. Figure 1 is a pictorial illustration of our approach.

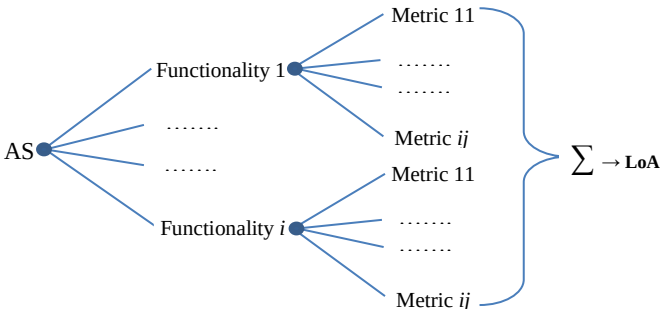

Figure 1: Pictorial illustration of how LoA is achieved by summing the metric autonomic value contributions of all metrics defining all functionalities of a particular AS.

Given that any AS is defined by a number of self-* autonomic functionalities, say n, the mathematical *Combination* expression (4) is the representation of the possible combinations of the functionalities:

$$\sum_{r=1}^{n} {}^{n}C_{r} \rightarrow \#\text{ of possible combinations} \qquad (4)$$

The number of possible combinations indicates the possible functionality compositions of a system where n is the number of functionalities (the self-*) and r is an enumerator of the possible implementation combinations (see the rightmost enumerated values in Figure 2). The functionalities may not be of equal importance to an application domain so combinations indicate what functionality is important to an application domain. And depending on choice of usage, this may be defined as *required* functionalities (in which case r may be equal to n) or demonstrated functionalities (in which case $r \leq n$).

Autonomic functionalities may overlap i.e., are not necessarily orthogonal. For example, a function that primarily achieves self-healing may change internal configuration and thus may also be described as self-configuring. To represent this, we allocate weights to indicate the extent to which a particular algorithm achieves the different functionalities. Further, self-managing actions are not necessarily linear in their operation; i.e., the relationship between a self-tuning parameter change internally and the externally seen effect of the change may be non-linear. In addition, for a given system, one autonomic behaviour e.g., self-healing may have a much more significant effect on system behavior than perhaps self-optimisation which may be more subtle. Such non-linearity in the contribution to LoA is catered for by a combination of weighting and normalisation (see Section IV part C). Weights are applied to reflect the extent of impact one of a particular functionality. Our current technique caters for orthogonality and non-linearity although to some extent these are open challenges that need further addressing.

Table I is a description of notation keys used. To measure the LoA of a system, we require the following:

- Number of functionalities: this is a value indicating the number of functionalities present or required in a particular system – a specific implementation combination of the functionalities.
- Number of metrics: this is the number of identified metrics for the respective functionalities.
- Weighting: weights are assigned to functionalities and metrics according to priority or importance.

TABLE I: NOTATION KEYS

Key	Description
a_{ij}	autonomic value contribution for individual metric j of functionality i
k_i	autonomic value contribution for individual functionality i
LoA	total level of autonomicity measure for all f_i & m_{ij}
M_i	number of metrics for functionality i
M_c, M_h, M_o, & M_p	number of metrics for each of the self-* functionalities respectively
m_{ij}	individual metric j for functionality i
n	number of functionalities
n_i	individual functionalities
r	possible combinations of functionalities
R_i	rank of a functionality i in the autonomic composition of a system
v_i	weighting for functionality i
w_{ij}	weighting for metric j of functionality i
c_i, h_i, o_i and p_i	autonomic metric contributions of the functionalities for a CHOP-based system
All indices (i and j) begin at 1	

A. Preliminary Work: A Specific Case Approach

To make progress in this approach, a preliminary effort is set out in [1]. This initial effort works perfectly well in cases where functionalities are orthogonal and for specific systems of limited (known) number of functionalities. Now, following on from equation (4) and taking a specific system in isolation, for example, (say a system with only four functionalities, e.g., the CHOP), this will give 16 possible combinations as shown in Figure 2, although the 16th combination is a special case which implies the system demonstrates no autonomic functionality and thus it is not considered further. The CHOP

functionalities may not all be of equal importance to a particular application domain hence we enumerate the possible combinations of functionalities, for reference. Combination 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 combination representing a system with enumeration 4 in Figure 2.

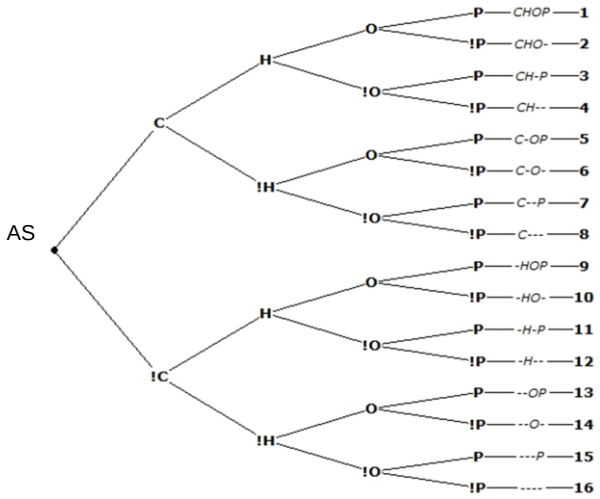

Figure 2: Combination of autonomic functionalities (for n = 4).

Figure 2 implies that, in terms of autonomic functionality composition, a system deemed autonomic (within the self-CHOP boundary) can be defined (or described) in one of n^2-1 ways. The remaining combination (enumerated 16 in Figure 2) represents a non-autonomic system, as it exhibits none of the autonomic functionalities. 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 2 will be the summation of the autonomic metrics defining the self-healing, self-optimising and self-protecting functionalities.

$$LoA = \sum_{i=1}^{Mc}[c_i] + \sum_{i=1}^{Mh}[h_i] + \sum_{i=1}^{Mo}[o_i] + \sum_{i=1}^{Mp}[p_i] \quad (5)$$

Subscripted *M* is the number of identified metrics for the respective functionalities. c_i, h_i, o_i and p_i are the autonomic metric contributions of the functionalities. These can be composed of functions of different measures but as explained in Section IV(C), they are normalised to yield autonomic values. For more details regarding the preliminary work and the specific case approach of the proposed measure, see [1].

B. Measuring LoA: A Generic Case Approach

Having looked at specific (of known number of functionalities) case instance of the proposed approach, we seek, in this subsection, to establish a generic case in which this approach is suited for application across different scenario instances. Now, extending the approach and making it more generic, weighting is introduced. Functionalities are not necessarily orthogonal –i.e. a single behaviour could enhance the contribution of more than one metric and this could be across more than one functionality. This is important because the measurement approach has to work in situations where the functionalities are and are not orthogonal. In cases of non-orthogonality, the weighting is applied to tune sensitivity of contributing behaviours.

For flexibility of applying the technique across different application instances, *LoA* is normalised to a value in the range 0 to 1. It also follows that all autonomic value contributions and weighting are normalised within the same interval range:

$$0.0 \le LoA \le 1.0, \;\; 0.0 \le a_{ij} \le 1.0$$
$$0.0 \le v_i \le 1.0, \;\; 0.0 \le w_{ij} \le 1.0 \quad (6)$$

Normalisation of the individual components of the formulae is important to enable comparison of different systems with different implementations, and also to address non-linearity aspects. The way we measure the system should not on its own change the outcome –for example, higher number of metrics should not result in higher *LoA* value and as well does not translate to being 'more autonomic'. So in all cases, and for normalisation purposes, the following rules must apply:

$$\sum_{j=1}^{M_i} w_{ij} = 1.0, \qquad \sum_{i=1}^{n} v_i = 1.0\,, \qquad \sum_{j=1}^{M_i} a_{ij} = 1.0 \quad (7)$$

The metric weighting (w_{ij}) and metric autonomic value contribution (a_{ij}) are both with reference to individual functionalities and so are bound to the number of metrics for those functionalities (M_i). However, the functionality weighting (v_i) is with reference to the system itself and so is bound to the total number of functionalities (n) .This explains why the total individual autonomic value contribution ($\sum k_i$) can go up to n –see equation (9). If we ignore, for now, all indices and have a top level view of the proposed *LoA* calculation, for a single functionality, then:

$$k = (a \times w) \le 1.0 \quad (8)$$

$$\sum k \le n \sum \quad (9)$$

$$LoA = \sum(k \times v)$$
$$\rightarrow \sum[(a \times w) \times v] \;\; \forall a, w, v \le 1.0 \quad (10)$$

Decomposing (9) and (10) above, and for total autonomic value contribution of all functionalities n_i:

$$k_i = \sum_{j=1}^{M_i}\left(a_{ij} \times w_{ij}\right) \quad \forall n_i and\, m_{ij} \quad (11)$$

And applying the functionality weighting to the individual autonomic value contribution (k_i), we have:

$$k_i = v_i \times \left(\sum_{j=1}^{M_i} (a_{ij} \times w_{ij}) \right) \quad \forall n_i and\ m_{ij} \qquad (12)$$

LoA is then given by summing equation (12) for all values of n_i and m_{ij}:

$$LoA = \sum_{i=1}^{n} \left(v_i \times \left(\sum_{j=1}^{M_i} (a_{ij} \times w_{ij}) \right) \right) \qquad (13)$$

In the case of orthogonality or where weighting is not required, level of autonomicity is given by the basic expression:

$$LoA = \sum_{i=1}^{n} \sum_{j=1}^{M_i} (a_{ij}) \qquad (14)$$

This is equivalent to equation (5). Procedure 1 is a basic algorithm of the implementation of the proposed measure of autonomicity.

Procedure 1: Algorithm for implementing LoA

```
Input (main) variables: n and Mi
i = 1, 2, ..., n and j = 1, 2, ..., Mi
    if at n1, M1 = 3, then j = 1, 2, 3
  k1 = (w11 × a11) + (w12 × a12) + (w13 × a13)
k(1) = 0 //initialising k array
for i = 1 to n
 for j = 1 to M(i)
  sum(j) = w(ij) × a(ij)
  k(i) = k(i) + sum(j)
 next j
next i
 LoA = (k1 × v1) + ... + (kn × vn)
```

Note that the proposed approach is a 2-dimensional definition. That is, it supports only two levels of description, e.g., a system on one hand and its functionalities or characteristics on the other hand. A bit of tweaking and adaptation is required to support higher dimensional definitions e.g., a system, its functionalities or characteristics, sub-functionalities or sub-characteristics, etc.

C. Normalisation and Scaling of Autonomic Metrics Dimensions

There is still a point though that needs to be addressed. When computing for *LoA*, we are normalising values that are products of aggregated metric values of different units and dimensions. Depending on the application domain, metrics can be scalar (of different measures) or non-scalar values (e.g., observing a capability, Boolean based decisions, etc.). So, despite what measure or form these metrics take, there needs to be a way of scaling the metric values (of all contributing metrics) to a centric unit of *autonomic metric contribution* within a certain normalised range. But, because the range of values and metrics can vary significantly, each choice of how these are scaled can influence very differently the final *LoA*. A possible solution is to define scaling factors for all contributing metrics within a normalised range (of [0, 1] in our case). In this way, the metrics' values (irrespective of units of measure) are normalised into real numbers that are summed to give *LoA*. One challenge here, though, is defining the scaling factors. We identify two simple methods for normalisation: 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 ranked *high*. A medium value would contribute fifty percent of the metric's autonomic value contribution in the range of [0, 1] (recall that $0.0 \leq a_{ij} \leq 1.0$ from equation (6)), while the two extremes would contribute zero and hundred percents –these may differ depending on choice of usage. 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 does not confirm a capability (e.g., the *stability* metric for self-optimising). Other specific methods, like the *Mahalanobis Distance* [22] discussed and used in [23], have been proposed. In scaling the different dimensions of distances between points (measured in different distance measurement units), the authours of [23] use a simplified form of the *Mahalanobis Distance*, where for each dimension, they compute the standard deviation over all available values and then express the components of the distances between points as multiples of the standard deviation for each component.

In the end, anyone can choose any form of scaling and normalisation as long as it is uniformly used across board for all systems to be evaluated and all values are within the range [0, 1] as explained in equations (6) and (7).

D. Measuring LoA: Comparison of Approaches

Assessing autonomic systems and being able to analyse and compare diverse systems of different degrees is an open research challenge that needs significant attention. There have been several attempts to develop a way of measuring autonomicity but unfortunately a universal solution has not been found. [8] shows that up to this point, there is one main approach to measuring the extent of autonomicity of autonomic systems (the *scale-based* approach which is explained in Section II), and a number of variations of this

have been explored. The fundamental purpose of this approach is to reflect the level of involvement in decision-making between the system and the human user. The major variations of the *scale-based* approach are Clough [4], Sheridan [9], and Ryan *et al.* [2]. Clough's 10-level scale is a result of developing national intelligent autonomous UAV metrics for the Department of Defence (DoD). Though it is tied to UAVs, its use of metrics to measure the level of autonomy of UAVs makes it stand out. The levels in-between the scale are populated by defining metrics for UAVs. This is good because using metrics that define functionalities gives a clearer understanding of the systems. Yet there is no normalised single point of reference that can be used in comparing two systems using this approach. Sheridan's 10-level scale measures two aspects; decision making (levels 2-4) and decision execution (levels 5-9). Ultimately, Sheridan focuses on human-machine relations (and human supervisory control) and not necessarily on the level of autonomicity of systems. Ryan *et al* extended Sheridan's concept and developed an 8-level scale that determines the level of autonomy *needed* in designing autonomic systems; although their work cannot actually be said to offer a way of *measuring* autonomic systems' level of autonomicity.

None of these is sufficiently sophisticated in measuring *LoA*. The technique we propose here is more sophisticated in a number of ways: it is the only technique that ties down *LoA* to a numeric value; it takes into account individual weights; it is flexible in the sense that it can take any number of degrees (functionalities), and the fact that the numeric value is scaled always to a normalised value (to cater for comparisons between systems with different numbers of dimensions of autonomicity and different numbers of metrics for measuring the extent of functionality achieved in each dimension). Normalisation gives you the power to compare two different systems no matter the number of individual metrics.

E. Evaluating Autonomic Systems

Evaluating Autonomic Systems using equation (5) or equation (13) gives their separate *LoA* values –which are aggregated values. This, however, does not give a fine-grained picture of the systems' performances in terms of individual functionalities. Systems are classified according to their implementation combinations (*r*). This is in terms of what self-* functionalities are required or demonstrated 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 combination 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 autonomic system defined in [11]. The rank of a functionality R_i in the autonomic composition of a system is defined by the ratio of its autonomic contribution (k_i or a_{ij}) to the total autonomic contribution of all metrics defining the composite functionalities of that system:

$$R_i = \frac{k_i}{LoA} \tag{15}$$

This applies where weighting is considered. If weighting is not considered, R_i is given by equation (16):

$$R_i = \frac{\sum_{j=1}^{M_i} a_{ij}}{LoA} \tag{16}$$

where (k_i or a_{ij}) is the autonomic contribution of the considered functionality which could be the summation of c_i, h_i, o_i or p_i in equation (5) or the calculation of k_i in equation (11) or the summation of a_{ij} (e.g., the case in equation (14)). With equations (15) and (16), any composite functionality can be ranked in terms of their autonomic contribution.

V. AUTONOMIC SYSTEMS EVALUATION CASE STUDY

In this section, two example cases that cover the specific and generic case approaches (explained in Section IV) are examined. The first is based on *Dynamic Qualitative Sensor Selection System* (DQSSS) application scenario (see [19] for full details of the DQSS system). This is used to demonstrate a case where functionalities are assumed to be orthogonal and for specific systems of fixed number of autonomic functionalities. This is consistent with the preliminary work in [1] and suites the proponents of the view that autonomic systems are only defined by the generally accepted and core functionalities of the self-CHOP.

The second case example deploys one of the current technology innovations –Autonomic Marketing. This is used to demonstrate a generic case instance where functionalities are not necessarily orthogonal and where systems are defined by *n* number of autonomic functionalities. For more details on the autonomic marketing system scenario see [24] and [25].

For each case example, three systems (or autonomic managers) are examined. When comparing these systems, it is important to look closely at the performances of individual autonomic functionality to give clearer understanding of the calculated *LoA*.

A. Dynamic Qualitative Sensor Selection Case Example

In this example, autonomic functionalities are limited to the original, and generally accepted four self-CHOP functionalities, supposing that any autonomic system is defined by them. This is representative of many real-world systems of known (fixed) functionalities or characteristics. The DQSSS case study is based on work in [19]. 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, self-optimization and self-healing are of importance to this system (i.e., *r=3* and also *n* is fixed at 4). The DQSSS in [19] is presented in three progressive stages which we refer here 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 II: REPRESENTATION OF THE DQSSS [19]

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	-	-	√

In keeping with the normalisation of values as contained in (6) and (7), the maximum achievable *LoA* becomes '1' implying that each CHOP contributes an autonomic value in the range ($0 \leq a_{ij} \leq \frac{1}{M_i}$) spread across its metrics. Normalising the identified metrics in Table II (the numbers of metrics in each combination are: $M_1 = 4$ (for self-configuring C), $M_2 = 2$ (for self-healing H), and $M_3 = 3$ (for self-optimising O)) in the autonomic value range ($0 \leq a_{ij} \leq \frac{1}{M_i}$) and applying equation (5) or (14) gives the result in Table III. Equation (17) is an expression of how each instance of the metrics contribution is calculated.

$$a_{ij} = \left(\frac{1}{n}\right) \times \left(\frac{1}{M_i}\right) \tag{17}$$

Figure 3 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 explains why self-protection P has no value. Based on the *LoA* achievements of the three systems A, B and C as shown in Table III (0.25, 0.33 and 0.63 respectively), it means that in a dynamic sensor selection application domain (as defined), system C can be depended upon to carry out the task with a higher confidence level and lower risk factor compare to systems B and A.

One powerful aspect of our proposal, particularly the specific case approach with fixed number of functionalities, is that it offers the flexibility of qualitatively interpreting *LoA* results using any *scale-based* approach. This is done by applying the upper bound of the chosen scale to equation (17) as in equation (18) and then interpreting the results within the levels of the scale.

$$a_{ij} = \left(\frac{max}{n}\right) \times \left(\frac{1}{M_i}\right) \tag{18}$$

Where *max* is upper bound of the scale used.

Applying Ryan *et al* level of autonomy assessment scale [2] which, 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), *max* becomes 8. So, in computing (18) with *max* = 8, 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 autonomically. System C falls within level 5, which points to '*collaboration with reduced human intervention*'. This indicates that C is sufficiently sophisticated to operate autonomically and yield satisfactory results under almost all perceivable operating circumstances.

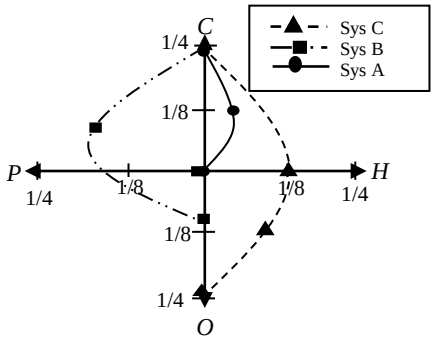

Figure 3: LoA representation of systems A, B & C in the CHOP domains.

TABLE III: ANALYSIS RESULT

	Sys A	Sys B	Sys C
C	0.25	0.25	0.25
H	0.00	0.00	0.13
O	0.00	0.08	0.25
P	0.00	0.00	0.00
LoA	**0.25**	**0.33**	**0.63**

Employing (16) 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 provides another analytic spectrum that gives a clearer understanding of the composition of the calculated *LoA*.

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.

B. Autonomic Marketing Case Example

In this example, we consider a specific aspect of autonomic marketing system based on the work and experiment presented in [25]. As there are yet no standardised (or defined) lists of autonomic metrics, at least for the case example system, we draw autonomic metrics and functionalities for the purposes of this case example from the specified goal of the system as detailed in [25]. So, metrics are drawn based on (or limited to) what is explained in the experiment and not on what autonomic marketing systems, generally, should have as metrics. In the end, the interest here is to show how the proposed *LoA* measurement approach can be applied to systems.

The particular case example autonomic marketing system studied here is that of targeted television advertising during a live sports competition airing. A company is interested in running an adaptable marketing campaign on television with different adverts (of different products appealing to audiences of different demographics) to be aired at different times during a live match between two teams. There are three adverts (Ad1, Ad2 and Ad3) to be run and the choice of an ad will be influenced by, amongst other things, viewer demographics, time of ad (local time, time in game, e.g., half time, TV peak/off-peak time, etc.), length of ad (time constraint), cost of ad, who is winning in the game, etc. This is a typical example of a system with many dimensions of freedom and very wide behaviour space. The behaviour space is divided into four zones along two dimensions of freedom (*Mood* and *CostImplication*) as shown in Figure 4.

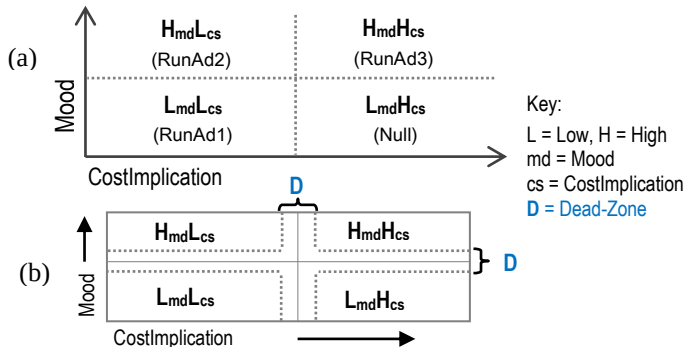

Figure 4: System behaviour space in 2 dimensions of freedom [25]

The two dimensions of freedom, which are influenced by several contextual variables, represent a collation of all possible decision influencers. Each action (ad) is thus activated only in its allocated zone following specified policy in order to achieve the system's goal defined by a set of rules (Figure 5). Following the set policy, the autonomic manager, at every *decision* instance (of a sample collection) decides on which ad to run. The optimisation of the system is in terms of achieving balance between efficient just-in-time target-marketing decision and cost effectiveness (savings maximisation) while maintaining improved trustability, stability and dependability in the process.

1. Extract external variables (decision parameters) at defined time interval and decide on action
2. Send trap msg and change action if (*condition omitted*) otherwise retain previous action
3. If current action is same as previous action, do not send trap and do not change action

==========Measure of Success==========

4. Cost of action change (total ad run) must fall within budget
5. Rate of change should be considerably reasonable
6. Maximum of one ad change within the first five sample collections and subsequently maximum of two in any three sample instances
7. Turnover should justify cost

Figure 5: Excerpt of rules defining system goal [25].

Three autonomic managers, based on three different levels of autonomic architectures, are designed to implement this system. In this example, we evaluate these three managers (full detail of experiment is available in [25]). Basically, the first manager, AC (AutonomicControlling at its core), is concerned with making decisions within the boundaries of the rules while the second, VC (ValidationCheck at its core) goes beyond decision making to validate decisions for conformity with the rules. The third manager, DC (DependabilityCheck at its core) verifies that the measure of success is achieved. DC also improves reliability by instilling stability in the system. This is done by introducing dead-zone boundaries (Figure 4b) within which, no action is taken (avoiding erratic and unnecessary changes) and implementing a TRC (Tolerance-Range-Check) to address rules in particular.

Based on the goal of the system, specified in the rules of Figure 5, we have drawn three functionalities (self-configuration, self-optimisation and self-stability) and six metrics. Table IV shows the metrics and contributing rules. The simulation was run for a total duration of 50 sample collection instances. This means that for the duration of the simulation, external variables (that influence decisions for ads) were fed to the autonomic managers fifty times.

TABLE IV: AUTONOMIC METRICS FOR SYSTEMS AC, VC & DC

Metrics	Description	Contributing rule
# of ad change (x)	Number of times ads changed	Rule 2
# of ad run (y)	Number of ads that were run	Rules 3 and 4
# of decision (d)	Number of decision instances	d is a constant
Rate of ad change (T_x = x/50)	Rate at which ads were changing	Rules 2 and 5
Rate of ad run (T_y = y/50)	Rate at which ads were run	Rules 4 and 7
Decision ad change ratio (d_x = x/d)	Number of decision ad change relation	Rule 3
Stability (s)	Number of times TRC bounds were breached	Rules 5 and 6

Recall that functionalities are not always orthogonal as in this example some metrics contribute to more than one functionality (see Table V). The rate of the influence of the metrics on the functionalities is tuned by applying weighting. Table V shows the autonomic value contributions of all metrics and the corresponding functionalities. Weights are discretely allocated to reflect relevance and importance of functionalities (based on the goal of the system). Metric values are averages of 10 different simulation runs of the 50 sample collection instances (see Appendix A for more details). In this example, we assume all metrics to be of equal weight within their respective functionalities.

TABLE V: DISTRIBUTION OF METRIC VALUES

Functionality (n_i)	Weight (v_i)	Metric (M_i)	Metric weight (w_{ij})			Metric contribution (a_{ij})		
			AC	VC	DC	AC	VC	DC
Self-configuration	0.20	x	0.50	0.50	0.50	0.000	2.000	4.800
		T_x	0.50	0.50	0.50	0.768	0.808	0.864
Self-optimisation	0.40	y	0.20	0.20	0.20	2.600	3.900	5.800
		T_y	0.30	0.30	0.30	11.42	11.45	11.49
		s	0.30	0.30	0.30	9.600	10.20	11.60
		d_x	0.20	0.20	0.20	0.000	0.171	0.414
Self-stability	0.40	T_x	0.25	0.25	0.25	0.768	0.808	0.864
		y	0.05	0.05	0.05	2.600	3.900	5.800
		T_y	0.20	0.20	0.20	11.42	11.45	11.49
		s	0.50	0.50	0.50	9.600	10.20	11.60

From the values of Table V, we can now calculate the *LoA* of all three systems (AC, VC, and DC). Because of space we will only show the calculations for that of system AC and then use the LoA Calculator (see Section V C) to calculate the rest.

$n = 3$

For n_1: $M_1 = 2$, $v_1 = 0.20$, $w_{11} = 0.50$, $w_{12} = 0.50$, $a_{11} = 0.00$, and $a_{12} = 0.768$

For n_2: $M_2 = 4$, $v_2 = 0.40$, $w_{21} = 0.25$, $w_{22} = 0.25$, $w_{23} = 0.25$, $w_{24} = 0.25$, $a_{21} = 2.600$, $a_{22} = 11.42$, $a_{23} = 9.600$, and $a_{24} = 0.000$

For n_3: $M_3 = 4$, $v_3 = 0.40$, $w_{31} = 0.25$, $w_{32} = 0.25$, $w_{33} = 0.25$, $w_{34} = 0.25$, $a_{31} = 0.768$, $a_{32} = 2.600$, $a_{33} = 11.42$, and $a_{34} = 9.600$

$$k_1 = (a_{11} \times w_{11}) + (a_{12} \times w_{12}) = (0.00 \times 0.50) + (0.768 \times 0.50)$$
$$= (0.00) + (0.384)$$
$$= 0.384$$

$$k_2 = (a_{21} \times w_{21}) + (a_{22} \times w_{22}) + (a_{23} \times w_{23}) + (a_{24} \times w_{24})$$
$$= (2.600 \times 0.25) + (11.42 \times 0.25) + (9.60 \times 0.25) + (0.0 \times 0.25)$$
$$= (0.65) + (2.80) + (2.40) + (0.00)$$
$$= 5.850$$

$$k_3 = (a_{31} \times w_{31}) + (a_{32} \times w_{32}) + (a_{33} \times w_{33}) + (a_{34} \times w_{34}) =$$
$$(0.768 \times 0.25) + (2.600 \times 0.25) + (11.42 \times 0.25) + (9.60 \times 0.25)$$
$$= (0.192) + (0.650) + (2.855) + (2.400)$$
$$= 6.097$$

Applying equation (13):

$$\text{LoA} = (k_1 \times v_1) + (k_2 \times v_2) + (k_3 \times v_3)$$
$$= (0.384 \times 0.20) + (5.850 \times 0.40) + (6.097 \times 0.40)$$
$$= 0.0768 + 2.340 + 2.439$$
$$= 4.8558$$

Figure 6 is a snapshot of the *LoA* Calculator's result console showing the *LoA* results of systems VC and DC. Recall that the choice of scaling and normalisation used can influence very differently the final *LoA*. In making a choice, the nature of system and metrics need to be considered. In this example, we can choose to normalise the individual sub-columns of the a_{ij} column within the range ($0 \leq a_{ij} \geq 1$), but this will negatively affect the values across all three systems and make it difficult for *LoA* calculations. Also we cannot normalise within the column across rows as that will overshoot the normalisation range within the sub-columns. So, we calculate with raw values and then normalise the final *LoA* values. How the metric values in Table V are scaled is shown in Appendix A.

From Figure 6:

LoA for system VC = 5.4887
LoA for system DC = 6.4722

The calculated *LoA* for system AC = 4.8558

Then, normalising all three values within the normalisation range of ($0 \leq LoA \geq 1$) using expression (19):

$$AC = \frac{AC}{(AC + VC + DC)}, VC = \frac{VC}{(AC + VC + DC)}, DC = \frac{DC}{(AC + VC + DC)} \quad (19)$$

For system AC

$$LoA = \frac{4.8558}{16.8167} = \mathbf{0.2887}$$

For system VC

$$LoA = \frac{5.4887}{16.8167} = \mathbf{0.3263}$$

For system DC

$$LoA = \frac{6.4722}{16.8167} = \mathbf{0.3849}$$

The results clearly indicate the superiority of the systems from DC down to AC in terms of level of autonomicity (based on the criteria set as system goal). What this means is that, in terms of the criteria specified as the goal of the system, the autonomic manager of system DC is more autonomic than the others followed by system VC. The margins reflect, almost with the same magnitude, the performances of the systems as results show in [25].

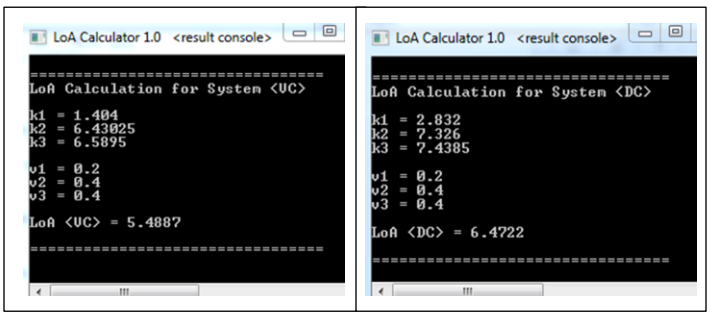

Figure 6: LoA calculate console result for systems VC and DC

Recall that the optimisation of the system in question is in terms of achieving balance between efficient just-in-time target-marketing decision and cost effectiveness (savings maximisation) while maintaining improved trustability, stability and dependability in the process. From experimented results, system DC shows significant gain in stability and cost savings. It also smoothened the high fluctuation rate (high adaptability frequency) experienced by other systems and in general, reduces the average ad change ratio of about one change in three sample collections (1:3) to one change in ten sample collections (1:10), representing an overall gain of about 31.25% in terms of stability and cost efficiency.

C. LoA Calculator

The LoA Calculator is an application that helps in calculating system's level of autonomicity. The application is developed using C# and can be used in calculating the LoA of any system at any level of complexity. The application can be used for both the generic and specific case approaches. For the specific case approach (that may require no weights), the user enters the value '1' (one) in the place of all weights and that will automatically cancel the weighting effects. Basically, all variables and values used are user-defined and are fed into the application for LoA computation. There are two formats for the application. The basic format is for simple systems of few variables and provides a dialogue interface (a form) for a user to enter system variables. This is suitable when there are only a few data to be fed into the application and can be done through the keyboard. Figure 7 is a snapshot of the dialogue interface. The other format is more complex and is used for complex systems (of multiple data). The complex format feeds data into the application using comma separated text file (csv file). The user specifies raw data in a CSV file template and provides the file path to the application during run-time. Both formats work on the same principle (the core is based on equation (13)) and can be used interchangeably but it is more tedious using the basic format for complex systems.

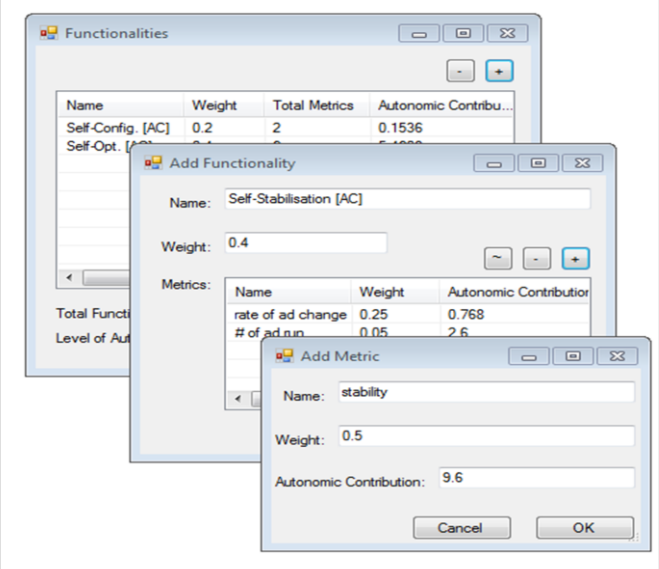

Figure 7: LoA Calculator basic format dialogue interface

In the current (first) version of the application, only variables and values for one system can be fed into the system at any one time. Subsequent versions will be able to take values for more than one system at one instance and evaluate the systems in terms of their separate LoAs. The application is available for download at [26].

VI. 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 functionality-based LoA measurement. In our proposal, a typical AS is defined by some core autonomic functionalities 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. Our proposed approach is in two forms; the specific case approach and the generic case approach. The specific case approach works perfectly well in cases where functionalities are orthogonal and for specific systems of limited (fixed) number of functionalities. We have shown how this approach can adapt any scale-based approach to enable a qualitative understanding of the quantitative LoA measure proposed here. The generic case approach is used to demonstrate a generic case instance where functionalities are not necessarily orthogonal and where systems are defined by *n* number of autonomic functionalities. We have also shown how systems can further be evaluated to give a fine-grained picture of the systems' performances in terms of individual functionalities looking at the ratio of autonomic contributions of their separate functionalities. In this, we found that only systems within the same implementation combination can be compared. We have carried out two case study examples (for the specific and generic case approaches) to demonstrate the usage and applicability of our proposed LoA measure. There are several other research works trying to develop a way of measuring autonomicity but have not succeeded. Some approaches have been proposed but none of these is sufficiently sophisticated in measuring LoA. Our technique here is more sophisticated in a number of ways: the fact that it is the only one that ties down LoA to a numeric value, the fact that it takes into account individual weights, it is flexible in the sense that it can take any number of degrees (properties), and the fact that numeric values are scaled always to a normalised value – (which otherwise gives the wrong impression that more metrics mean more autonomicity. Normalisation gives you the power to compare two different systems no matter the number of individual metrics).

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 autonomic systems, and even to be able to compare alternate systems themselves and approaches to building autonomic systems, 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.

One research challenge is the study and standardisation of autonomic metrics for different autonomic systems. The metrics definitions can be grouped or modularised (e.g., the standardised categorisation of UMS in [20]). So, as future work, we are looking at standardised ways of properly defining and generating autonomic metrics to strengthen our framework. This is a key component towards our wider research which focuses on the challenge of validating autonomic computing systems to achieve trustworthiness. We will be developing more sophisticated versions of the LoA Calculator.

REFERENCES

[1] Eze T., Anthony R., Walshaw C. and Soper A. *A Technique for Measuring the Level of Autonomicity of Self-managing Systems.* In proceedings of The 8th International Conference on Autonomic and Autonomous Systems (ICAS 2012), St. Maarten, The Netherlands Antilles, March 2012

[2] Proud R., Hart J., and Mrozinski R. *Methods for Determining the Level of Autonomy to Design into a Human Spaceflight Vehicle: A Function Specific Approach*. Report date September 2003 http://handle.dtic.mil/100.2/ADA515467 accessed 04/09/2012

[3] Eze T., Anthony R., Walshaw C. and Soper A. *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

[4] Clough B. *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.

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

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

[7] Honeycutt G. *How Much Do we Trust Autonomous Systems?* Unmanned Systems -2008

[8] Eze T., Anthony R., Walshaw C. and Soper A. *Autonomic Computing in the First Decade: Trends and Direction*. In proceedings of The 8th International Conference on Autonomic and Autonomous Systems (ICAS), St. Maarten, The Netherlands Antilles, March 2012

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

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

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

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

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

[14] McCann J. and Huebscher M. *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

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

[16] 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

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

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

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

[20] Huang H., Albus J., Messina E., Wade R., and English W. *Specifying Autonomy Levels for Unmanned Systems: Interim Report*, SPIE Defense and Security Symposium 2004, Conference 5422, Orlando, Florida, April 2004.

[21] Huang H., Pavek K., Albus J., and Messina E. *Autonomy Levels for Unmanned Systems (ALFUS) Framework: An Update*. In proceedings of SPIE Defense and Security Symposium, Orlando, Florida. 2005

[22] Online article. *Mahalanobis Distance*, available via http://classifion.sicyon.com/References/M_distance.pdf viewed 10/09/2012

[23] Huebscher M. and McCann J. *An adaptive middleware framework for context-aware applications*, Springer Volume 10, Issue 1, pp 12-20, February 2006

[24] Adams C., Anthony R., Powley W., Bell D., White C., and Wu C. *Towards Autonomic Marketing*, The 8th International Conference on Autonomic and Autonomous Systems (ICAS), pp. 28-31, St. Maarten 2012.

[25] Eze T., Anthony R., Walshaw C. and Soper A. *A New Architecture for Trustworthy Autonomic Systems*. In proceedings of The 4th International Conference on Emerging Network Intelligence (EMERGING), Barcelona, Spain, 2012

[26] LoA Calculator Downloads via http://thaddeus-eze.com. Accessed 19/12/2012

[27] Parasuraman R., Sheridan T., and Wickens C. "A model for types and levels of human interaction with automation," IEEE Transactions on Systems, MAN, And Cyberneticsparta: Systems and Humans, vol. 30, pp. 286–297, MAY 2000

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

APPENDIX A

	# of ad change (x)			# of ad run (y)			stability			rate of ad change (x/50)			rate of ad run (y/50)			(d) # of	decision ad change ratio (x/d)		
Runs	AC	VC	DC	AC	VC	DC	AC	VC	DC	AC	VC	DC	AC	VC	DC	decisions	AC	VC	DC
1	12	12	7	7	7	5	2	2	0	0.24	0.24	0.14	0.14	0.14	0.1	12	1	1	0.5833333
2	8	6	3	7	5	3	3	2	0	0.16	0.12	0.06	0.14	0.1	0.06	8	1	0.75	0.375
3	15	11	8	12	8	6	4	2	0	0.3	0.22	0.16	0.24	0.16	0.12	15	1	0.73333	0.5333333
4	10	7	6	7	7	5	1	0	0	0.2	0.14	0.12	0.14	0.14	0.1	10	1	0.7	0.6
5	15	12	9	10	9	7	2	2	0	0.3	0.24	0.18	0.2	0.18	0.14	15	1	0.8	0.6
6	10	9	8	9	9	8	2	0	0	0.2	0.18	0.16	0.18	0.18	0.16	10	1	0.9	0.8
7	13	10	6	12	9	6	3	3	0	0.26	0.2	0.12	0.24	0.18	0.12	13	1	0.76923	0.4615385
8	11	11	7	7	7	5	3	3	0	0.22	0.22	0.14	0.14	0.14	0.1	11	1	1	0.6363636
9	11	9	7	10	9	7	0	0	0	0.22	0.18	0.14	0.2	0.18	0.14	11	1	0.81818	0.6363636
10	11	9	7	9	7	6	0	0	0	0.22	0.18	0.14	0.18	0.14	0.12	11	1	0.81818	0.6363636
Avg	**11.6**	**9.6**	**6.8**	**9**	**7.7**	**5.8**	**2**	**1.4**	**0**	**0.232**	**0.192**	**0.136**	**0.18**	**0.154**	**0.116**	**11.6**	1	**0.82889**	**0.5862296**

Table A1: Raw metric values as collected from experimental results

Table A1 shows the raw values of 10 different runs of the same simulation. It is not scientifically reliable to work with values of a single simulation.

Note that *Stability* measures the rate at which the tolerance range check (TRC) is breached. The column for DC is all zero because, in all 10 simulation runs DC did not breach the TRC bound.

Metric	AC	VC	DC		AC	VC	DC
# of ad change (x)	11.6	9.6	6.8	11.6-all i.e, max minus values	0	2	4.8
# of ad run (y)	9	7.7	5.8	11.6-all i.e, max minus values	2.6	3.9	5.8
stability	2	1.4	0	11.6-all i.e, max minus values	9.6	10.2	11.6
rate of ad change (x/50)	0.232	0.192	0.136	1-all i.e, max minus values	0.768	0.808	0.864
rate of ad run (y/50)	0.18	0.154	0.116	11.6-all i.e, max minus values	11.42	11.45	11.49
# of decisions (d)	11.6	11.6	11.6				
decision ad change ratio (x/d)	1	0.829	0.586	1-all i.e, max minus values	0	0.171	0.414

Table A2: Scaled metric values

Table A2 (left side) is a collation of the averages of the values of Table A1. On the right side of the table are the working values used. These are generated with reference to the possible maximum values of the individual metrics. For example, ads are changed or retained at every decision instance and since on the average, there are 11.6 decision instances, there are as much maximum possible number of ad change. So, the actual number of ad change is the difference between the number of possible ad changes and observed number of ad changes

The AC number of ad change metric is 11.6 (same as the number of decisions) indicating that AC changes ad at every decision instance. This tends to instability. Compare this to the value for DC and observe the difference.