Autonomic Computing in Total Achievement of Quality

Joel Bennett, Roy Sterritt
School of Computing
Ulster University
Belfast, Northern Ireland
email: bennett-j8@ulster.ac.uk |
r.sterritt@ulster.ac.uk

Abstract—**This paper presents a Systemization of Knowledge (SoK) on Autonomic Computing (AC) for Total Quality Management (TQM), i.e. a review of the domain of Quality GxP manufacturing environments considered through the paradigm of Autonomic Computing. The development of autonomic computing concepts and how they are applied currently are discussed. The paper then examines quality systems for each of; manufacturing and automation; product testing validation; data integrity; and supporting IT infrastructure; as pertaining to GxP manufacturing environments, being subject to high levels of regulatory compliance, before concluding with considerations about the need for this self-managing computing paradigm for quality manufacturing, and some avenues of progress identified in the current and future state.**

Keywords-Autonomic Computing; quality system; total quality; achievement; self-x; TQM; MES.

I. Introduction

The objective of this review will be considering the area of Quality manufacturing environments, particularly how those environments are supported by computing systems and how they benefit, or could benefit from the Autonomic Computing (AC) paradigm. Quality, in the context of manufacturing, encompasses a wide range of frameworks, standards and procedures, which include implementation of Good Manufacturing Practice (GMP) [1], Continuous improvement (CI) [2] and for computerised systems supporting the processes, adherence to the Good Automated Manufacturing Practice (GAMP) framework "which aims to safeguard patient safety, product quality and data integrity" [3]. These requirements are commonly found in any setting which requires a high level of regulatory compliance and accountability, such as that found in food, pharmaceutical, or health care.

The goal of these kinds of standards is not always easy to define, but generally is captured under the term Total Quality Achievement (TQA). Standards are updated and there is always an expectation that cases for CI will exist in any organisation, which are found through a combination of internal review and external audits. This is important in ensuring that customer safety standards are met, maintained and kept front and centre.

Underneath the quality activities, there are a range of computer systems and software, from machine automation to product testing, to scheduling and batch release activities, with an overarching Quality Management System (QMS). These environments generate a lot of records and documents, data, data sets and require well defined data retention policies and most often this requires a high level of human effort.

The autonomic question is about how much humans needs to be involved in directly managing systems and how they can be designed beyond this. We can consider first a brief overview of what AC is and then some distinct areas within Quality manufacturing , so as to make some application of it.

This paper presents a Systemization of Knowledge (SoK) on Autonomic Computing for TQM, as such, the first section summarizes AC, then examines quality systems for each of; manufacturing and automation; product testing; validation; data integrity; and supporting IT infrastructure; before concluding.

II. AC & TQM: Systemization of Knowledge (SoK)

This section presents a Systemization of Knowledge (SoK) on Autonomic Computing for TQM.

A. Autonomic Computing Paradigm

Having a brief introduction to AC, what it is and sets out to achieve will be useful to understanding where it might fit into the area of Quality systems. The term autonomic is borrowed from the bodies nervous system which governs unconscious functions such as regulating heart rate and temperature, without burdening the conscious area of the brain [4]. As computer and computer supported systems with their software, have become larger, more sophisticated and more interconnected, with growing intranetworks and internetworks, the complexity eventually reaches a level, where the best of systems experts cannot account for all configurations, points of failure and providing timely response to errors in the whole system. The initial recognition of this has its beginnings in IBM, whose Paul Horn, introduced the idea to the National Academy of Engineers at Harvard University in a March 2001 keynote address [5]. IBM envisioned computer systems of systems, with their smallest edge endpoints, up to the largest datacentres, with all of those interconnections as somewhat analogous to the human bodies smallest molecular machines and the bodies signalling equipment, being zoomed out to view entire societies with all their interactions. AC as a paradigm has drawn inspiration from these initial ideas and as IBM and others had predicted, it has become recognised in the computing industry as a necessity to start trying to achieve this goal of systems that are self-managing guided by an autonomic principle.

Following the talk by Paul Horn, IBM released a printed work in October 2001 titled *"Autonomic Computing: IBM's Perspective on the State of Information Technology"* [5]. In this, IBM outlined the problems that AC was seeking to address, the necessity for it and how it might be achieved. The authors present a case, that human progress has always been rooted in the support provided by technology and automation, which frees up human work effort, in order to enable achieving bigger things. However, while computers and the IT industry have supported business and innovation to a certain point, the rising complexity of these systems eventually presents a risk of even reversing these benefits [5, p. 4]. The human effort required to support these very same IT systems as they expand, rises exponentially. The case is compelling and presents proposals for the capabilities an autonomic system should have, which is provided in 8 main points which are briefly, that an autonomic system should:

1) "Know itself" which is perhaps best summarised by this statement *"a system can't monitor what it doesn't know exists"*. [5, p. 21] This self term has become definitional to AC component descriptions throughout the field.

2) Be able to configure and re-configure itself in response to changing and unpredicted conditions.

3) Not be settled on the current state and always seek to optimise.

4) Be able to perform functions analogous to healing.

5) Be security aware and self-protecting

6) Know its supporting environment and activities, so that it can respond appropriately to them

7) and therefore must not be sealed off and isolated, but must be able to function interdependently facilitated by open standards

8) Constantly anticipate what resources are needed. *[5, pp 21-31]*

This is expanded upon in a further article published by IBM Systems Journal 2003 "The dawning of the autonomic computing era", in which the need for AC is re-stated and then how the industry can begin to adapt by shifting its design objectives from price/performance to instead prioritising "robustness and manageability" and the cost of ownership. [6 p. 7] Here autonomic self-x properties are also elucidated, namely the fundamental self-Configuring, self-Healing, self-Optimising, self-Protecting, which is otherwise captured by the term self-CHOP [6, pp. 8-9].

1) Self-Configuration – is awareness of components and the environment, so as to be able to dynamically re-configure, to automatically integrate new components and adapt. A simple example might be the Plug and Play or hot-swap features of modern hardware and operating systems, that are completely unsupervised after the component has been introduced. This is a distinct development upon the original features of automatic detection, followed by configuration wizards, manual bus assignments, or disk drive rebuilds.

2) Self-Healing – detect and diagnose errors in components, isolate and repair them, or disconnect as necessary. Prevent failures from occurring if possible through component management with a view to maintaining constant availability.

3) Self-Optimising – continual automatic tuning across the systems available resources. This may include some element of built-up knowledge, in order to predict and schedule, as well as the ability to dynamically allocate resources in response to demands.

4) Self-Protecting – Securing system resources via users identification, intrusion detection, secure backup and restore services.

A framework for achievement of this goal, was described in a so-called intelligent control loop as outlined in IBM's work on an autonomic blueprint [7] known as MAPE-K (in Figure 1). Envisioned as an abstraction from the underlying managed resources, an Autonomic Manager (AM), of which there may be several for different specialties, is based upon the MAPE-K loop, though not all of those capabilities need necessarily be used in every situation. AM's could be in a dedicated role, e.g., Self-Configuring, Self-Protecting, while other AM's can occupy a higher level with overall system supervision, described as orchestrating AM's.

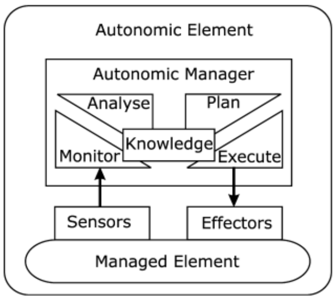

Figure 1. MAPE-K intelligent control loop [8, p. 6]

The make-up of the AM, includes Sensor and Effector interfaces which make one AM available to other AM's and the system components. From sensors, the Monitoring collects and filters data from a resource, which is then Analysed, providing correlations, modelling and predictions so the AM can learn from its environment. In Planning, the AM formulates strategy utilising its formed policies and finally these plans are Executed, whilst remaining open to being updated by new information. The results of these cycles update Knowledge for the purpose of improving outcomes.

With AC implementation as a background, we will now consider a few areas of Quality systems, whether they benefit from autonomic computing currently and where applicable, if there is any future development we might expect.

B. Quality Systems - Manufacturing and Automation

Manufacturing relies upon scheduling and execution controls, usually built upon a Manufacturing Execution System (MES). Typically, an MES system will provide some kind of information about a manufacturing floor and a level of control, for example if certain limits are exceeded, they will be reported to the appropriate receiver [9 p.3]. Computerised automation has greatly enhanced the ability of manufacturers to scale production and improve product quality, but has also increased system complexity. Tasks such as transfer of raw materials whether obtained, retained or disposed, must go through and from approved suppliers. The customers, manufacturer and suppliers often need to audit one another. Every record pertaining to the manufacturing process, including, but not limited to documentation of batch records and product release must be retained and retrievable. Such data integrity quality requirements will be considered later. Product manufacture may, for the most part, be described as automated, but is still heavily supported at almost every level by human activity and decision making. It seems in MES and automation, we can see some parallel, with the problems IBM drew attention to in its early autonomic works on computing.

Autonomic Smart Manufacturing [9] has been proposed for improving upon MES, modelling itself upon the MAPE-K framework, as in Figure 2, using a monitoring phase to collect metrics relevant to the manufacturing process, which are analysed to infer unknown relationships within the environment using machine learning (ML) to make what-if predictions, thus anticipating situations and performing the necessary calibrations. A plan phase would further reflect upon the findings, with a holistic treatment of individual behaviours allowing the system to propose and implement optimisations. In this model, there is still human supervision of the returns by an engineer, which is important to a quality process in terms of accountability, but an autonomic controller maintains optimal parameters as conditions change using a toolkit of built-up models in a knowledgebase to minimise human resource use.

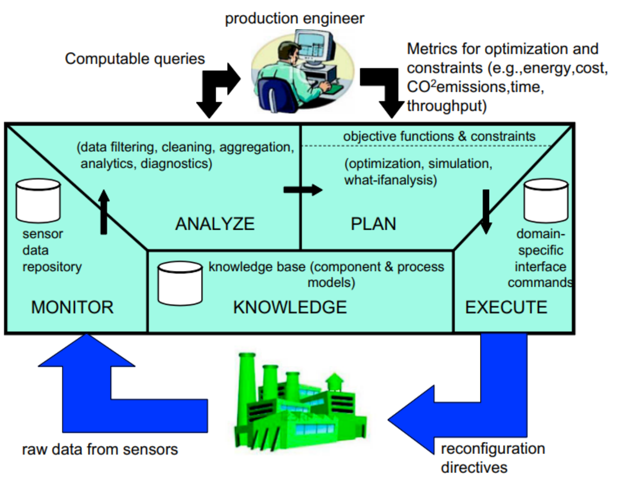
production engineer
Computable queries
Metrics for optimization and constraints (e.g.,energy,cost, CO²emissions,time, throughput)
(data filtering, cleaning, aggregation, analytics, diagnostics)
objective functions & constraints
(optimization, simulation, what-ifanalysis)
ANALYZE
PLAN
sensor data repository
knowledge base (component & process models)
domain-specific interface commands
MONITOR
KNOWLEDGE
EXECUTE
raw data from sensors
reconfiguration directives

Figure 2. MAPE-K applied to smart manufacturing [9, p. 5]

A further work on "Generative simulation modeling of an Autonomic Manufacturing Execution System (@MES)" [10] noted that traditional MES systems rely on rigid schedules which are shown to be inefficient and ineffective. It proposes a multiagent simulation model where an Order Acceptance autonomic manager has end-to-end knowledge of the shop floor, customer orders and can delegate tasks, while a number of Resource Agents have supervision of specific areas of the MES. The proposal is to implement a loop which incorporates the scheduling and control functions for specific resources, allowing the system to respond dynamically to shop floor requirements (environment) and reconfigure accordingly, whilst also informing the other agents, which will likewise recalibrate. The agents can simulate scenarios (i.e., is the next order feasible?) in conjunction with one another and are able to autonomically optimise routes, responding with suitable planning and execution. A later 2012 paper proposing a selfish multi-agent MES system atop @MES, states one issue remains in that enterprise networking of MES systems remains an open problem [11].

In terms of Quality systems, the concerns around such proposals, may be the validation of the autonomic agents controlling manufacturing processes at a high level and how to ensure accountability and oversight. However, if the autonomic agent has a validated means of justifying its mitigations, then perhaps this concern could be overcome. Validation will also be considered later.

C. *Quality Systems - Product Testing*

Product testing is necessary in any manufacturing process, but the burden of regulatory concerns in quality manufacturing is in many ways higher and must meet the requirements of an internationally recognised standard QMS such as ISO9001. Product testing supports the development of product, as well as forming part of the batch release and CI processes. Chiefly we will consider the typical laboratory setting, where product samples are taken from a batch, whether for production, or from a development cycle. The laboratory comprises a number of systems and instruments forming a testing suite and will of course vary, depending upon the kinds of products being analysed. The instruments utilised in testing must comply with national standards for regular calibration. Failed samples are reported against a batch, so that a determination can be made by quality assurance, investigating whether the sample test failure affects the whole batch, if further testing is required and what is the root cause of the product failures. The results of testing must be retained along with the records of any investigations.

The types of computing systems supporting these operations are typically the Laboratory Information Management Systems (LIMS) supporting clerical activity around recording test results [12], instrumentation control software on an integrated, or external computer and statistical analysis tools. Usually, each test or the days testing is preceded by a sample run to ensure the instrument is operating within defined parameters. Some tests are quite short in duration and others can run for many days. The instrument machinery is largely automated once configured and validated, but experiments are selected manually, including the passing of results and aforementioned calibration.

Laboratory performance is measured by the number of tests performed against erroneous tests performed. LIMS were created to improve the automation of product test data flows and ensure integrity, thereby reducing error and this is the main motivation for adoption of LIMS systems by laboratory management [12 p.2]. While LIMS systems exist that are automatically collecting and approving results, there are also many that involve manual entry of results. Many do incorporate, as with the MES and AM, a part of their control loop which automatically notifies and involves the relevant party when this is necessary. However, very often this is, per machine automation control, based purely on preset tolerance value thresholds.

There does appear to be little work done in the area of autonomic laboratory systems, but we can perhaps identify some autonomic-in-principal elements. There are many safeguards included with laboratory instrumentation, both with a view to safe-guarding results and the instrument itself. Many instruments have awareness of and do not allow operation outside the pre-defined calibration windows. This avoids producing costly invalidated results. Instruments also have internal sensors and diagnostics which prevent operation if a fault is encountered, or if a part has passed its expiration date. Very often, an instrument may have modules for different kinds of experiments and if not needed, the module may be bypassed.

Certainly, an implementation such as the proposed @MES will also need RA's associated with testing in order to simulate requirements. Furthering of autonomic principles in this area, may lead to systems utilising acquired knowledge during analysis to self-adapt. For example, if the current state involves manual investigation and root cause analysis due to test results, autonomic self and environment awareness, perhaps facilitated by other autonomic agents may allow for automatic determination of root cause. An external factor, such as the temperature of the laboratory may be a simple example. Even more desirable, might be autonomic agency preventing invalid tests from being conducted in the first instance and informing the other instruments of current issues. Also, in an autonomic laboratory system of systems, the whole test suite could likewise message other agents that a particular test has successfully completed, allowing subsequent tests on hold for another instrument to proceed, thus closely relating the data generated by those tests.

D. *Quality Systems - Validation*

The purpose of validation is per GAMP to provide documentary evidence to support a high level of confidence that all parts of a system will work correctly when used [13 p.1]. It comprises various levels of qualification, which describe and contain activities that test the functions a system is supposed to be able to perform. The qualifications include a Design Qualification (DQ) – documenting that all quality aspects of the system have been considered during the design of system, Install Qualification (IQ) – ensures that a system has been installed per its specifications, Operational Qualification (OQ) – as implied, that the listed system functions operates as expected according to the tasks which have been identified, Performance Qualification (PQ) – is evidence that the system works on an ongoing basis in its final setting. Validation applies in a quality setting whether a hardware, or software implementation. In some cases, it may include auditing the vendor of the product to be validated and one reason for this, is that they are supplying some of the documentation that supports the validation – for example, the IQ and OQ.

Security is also an important area of system validation, including that logins work correctly [13 p.3], since it has a direct bearing on the accuracy of records and traceability as per FDA Title 21 CFR Part 11. Data entry validation is also an important aspect, ensuring that data is formatted and saved appropriately, as well as being retrievable [13 p.4]. Evidence for the tests having been carried out may also be required where this is part of the specification.

The creation of design and test validation documents, collection of evidence and need for reviewers and approvers can be a lengthy and time-consuming process. Automation of systems has increased rather than decreased the level of effort required during validation, due to remaining distrust of automated systems, even when automated software testing is considered. Concerns about transparency arise, particularly when increasing automation may come to rely upon black box solutions where the underlying reasoning behind an activity cannot be directly observed. Automated testing and modelling can assist with allaying fears, such as injecting deliberate faults to see how the system handles different scenarios [14] and automated software testing is sometimes employed in validation, but automation can surely only get us so far.

If instead these systems were built autonomically from the ground up, with the autonomic agents forming almost a digital twin of business level hierarchy, which is trusted to provide the same level of assurance for each respective area, as human information gathering and review authority, then this would have the effect of increasing confidence in all of the systems which implement this by virtue of being recognised as self-correcting, self-securing, self-optimising, self-healing. We propose, that it is conceivable, that systems could become self-validating.

E. *Quality Systems - Data Integrity*

Integral to modern quality systems is Data Integrity (DI), which is data that meets standards of completeness, accuracy and consistency, i.e., the data must be ALCOA, i.e. attributable to a person or persons, legible, contemporaneous to what is being recorded, the original record and accurately recorded [15]. DI also requires consideration for how long records should be retained. On a computer, data can be in raw form, or processed form, but even in the current state of a heavily computerised environment with relatively high levels of automation, the original primary data records are often still paper based. However, as automation moves towards industry 4.0 smart manufacturing, much of the discussion inevitably turns to digitised data and data security.

A large concern in DI, is not just the maintenance of the original record [23 p.3], but the assurance of a validated backup and restore functionality, along with the data retention [23 p.49-50]. Legacy backup solutions were often manual, or even if automatic/scripted are triggered by simple rules within a time window. If a backup is missed, it likely does not run at all, although it may be followed by a notification. Many modern backup solutions do incorporate some autonomic elements, such as those described in "Lifeboat" for IBM, as far back as 2004 [16]. The solution proposes a decentralised peer-to-peer network backup model, with a distributed file system and awareness of disk quotas on each participating system. This eliminates single points of failure and relies on autonomic agency to determine the most appropriate use of available resources. It also discusses the scenarios of a server addition to the system, since availability of clients cannot always be guaranteed, as well as local backups. Many of the concerns with peer availability, have been solved in subsequent technology solutions which implement a Grid computing approach. Sharding algorithms can distribute redundant copies of partial data efficiently across as many peers as available and reconstruct it from x number of peers, way storage systems can reconstruct from parity data across x numbers of disks. A prime example of this in practice with regards to storage are Microsoft's own DFS and BranchCache technologies. The other main autonomic element of the IBM Lifeboat system seemed to be self-configuration in having awareness of new clients by referring to an asset collection database and adding detected clients to the backup automatically. Current backup solutions are relatively self-aware of available bandwidth and the size of data on individual clients, so as to be able to automatically allocate the order and groups of clients to backup queues ensuring that this happens within the provided timeframes. They can also perform self-signing or verification of the backup's integrity using hash checks. Very often the backups are self-optimising and regularly consolidate sets to remove duplication and clean-up to free storage. It seems reasonable to call these elements autonomic developments. However, most backup solutions to the present are still centralised, even if with redundancy. The incremental improvements are welcome, but achieving truly autonomic backup and restore would seem to require something further. Almost all backups still rely on schedules, but in a Quality environment any loss of data, even if rare is not acceptable. Even if the time gap is only a few hours between backups, data lost due to disk failure can affect product release, or manufacturing traceability and will be questioned by an auditor. Simply increasing the schedule frequency may be effective in reducing risk and simple, but is not an autonomic approach and increases system utilisation. The backup solution could instead have an AM, which senses relevant disk transactions and efficiently commits those to the backup storage incrementally. Similarly, being able to restore a failed disk from a backup is not an autonomic solution. An autonomic backup should try to anticipate the fault and ensure the system data is safeguarded and then offline the faulty component. When the fault is addressed, the AM should be able to recover from the fault without user intervention. Perhaps it could also be aware of similar/like systems and offer another suitable systems resource to carry on performing the same function, by making the faulty systems data and functionality available non-destructively. A similar solution was proposed in 2000, albeit using limited computing nodes, that was able to autonomically transfer running applications complete with memory and CPU register content, from faulty nodes to any available working node, which ensured continuity [17]. There has been much progress in the area of thin applications, which can make this level of availability more common. The main issue for a quality environment, is to establish trust of the underlying AM decisions and how to validate data integrity.

Most quality systems rely on a database to store, maintain and retrieve their generated data. Traditional relational databases may have some degree of autonomic design. Many include a self-repair and compact functionality which also serves as a self-optimiser and this is also recognised in an analysis published by IEEE in 2003 [18]. However, the limitations of several popular contemporary DBMS products were also discussed in terms of how they failed to be autonomic and how these products may get there. Under the heading "what is missing?", the article describes the high level of human input needed, the inclusion of data advisers and wizards, rather than self-configuration and the lack of self-optimisation in the form of ensuring the most efficient memory usage with optimal indexing. Likewise, databases tend to include recovery tools and not necessarily the autonomic self-healing property.

Another key component of quality systems data integrity since the late 90's is the Audit Trail. An audit trail provides evidence of actions performed by the system, or in the system. A weakness of the current audit trail implementations, is that, it often doesn't actually influence outcomes, but merely records activities. A security audit trail records that a user/system login took place, or that a particular record was saved by a given logged in user at a given time, but it is often a flat text log file, which doesn't have any actual connection to the potentially affected data records.

A type of database paradigm which seems to address the limitations of traditional database and logging systems in this respect is blockchain technology. The underlying principles of a blockchain are essentially autonomic. A blockchain database consists of a series of blocks, to which any kind of data can be written, and that data is then encrypted by a hash. The entries are both time-stamped and signed by a unique identifier which indicates ownership of the block, even if that identification is anonymous, the transactions are completely transparent on a ledger. Every subsequent entry relies on the hash of the previous block to decrypt itself, forming a chain. It is impossible to update the historical blocks without possessing every single cryptographic hash key, so it is self-protecting. The earliest blockchains were designed around the concept of functioning as currency, with unique identifiers being wallets, but more sophisticated blockchains can integrate so-called smart contracts which are automatically executed on the chain once agreed between parties, as well as hosting digital applications. Underneath the data layer of a blockchain is the concept of the nodes whether full, or partial, which host identical copies of the blockchains data and increase the reliability of the network and its bandwidth, whilst also providing a framework for consensus that block transaction being committed to the database are valid. Blockchains are largely self-configuring, in that they require no user intervention as to their data structure once started and automatically eliminate invalid blocks. They are self-healing because any potential corruption is automatically eliminated by making comparison with other nodes. They can also include apoptotic self-destruction of transactions once these have expired, which automatically releases the storage utilised by the transaction block.

Therefore, given these properties, it isn't surprising that a framework for utilising blockchain in an ISO compliant QMS to achieve Total Quality Management (TQM) has been suggested [19], but it would seem particularly suitable for ensuring contemporaneous association of data records with a digital identification, as per the requirements of DI, without the need for a separate audit trail.

F. *Quality Systems - Supporting IT Infrastructure*

Finally, although some of the areas above have touched upon the associated IT technologies which support the various quality processes, a brief consideration of how IT infrastructure as a whole supports these systems and how IT has benefited and may yet benefit from autonomicity to support quality seems appropriate. IT Infrastructure refers to all of the components necessary to deliver IT services within an organisation, e.g. equipment, network, software and services, including Internet based services and datacenters [24, p32]. A subset of IT Infrastructure examples are briefly considered below, which have been selected due to suitable existing and potential AC properties.

1) Storage – the dominant centralised server storage paradigm in enterprises of all sizes remains some version of Redundant Array of Independent Disks (RAID), usually level 5 or above. RAID storage systems have been available since at least the early 1990's and above 0 have a self-healing functionality [20, p.8] whether via duplication, or parity, as well as self-configuration in allowing hotswap disk replacement. It seems that due to historicity, these autonomic mechanisms being invisible to the quality process are trusted to perform their data management tasks. This may form a basis for trust in further autonomic developments.

2) Network – most business networks use a tiered star topology utilising centralised switches. Setting aside the increased expenditure of additional cabling and network switches, this is an improvement on the old bus-based networks, since the failure of one cable does not bring down the others segments. However, network switches and ports still often represent single points of failure. There may be a case for Survivable Network Architectures (SNA) as employed in telecommunications [21 p.2] for certain quality operations, particularly automated machine networks in which the network supports communication between machine components and when down causes the whole system to cease operating. Many modern managed switches do offer some autonomic features. Spanning Tree Protocol (STP) prevents physical network loops, by blocking them at Layer-2. Quality of Service (QoS), manages traffic and automatically adjusts network bandwidth allocated to prioritise specific services.

3) Servers – a server historically was usually hardware dedicated to a specific service, or set of services with the needed storage, memory and CPU specification. Typically, in the last 10 years at least, localised servers have become divided into storage arrays, shared via dedicated Storage Area Network (SAN) and separate server hosts, which run the services in virtualisation containers and share their memory and CPU across the respective virtual servers. Virtualisation has incorporated many autonomic elements, including features such as VMWare Fault Tolerance (FT) [22, p.4] which can automatically migrate a running server from one host to another is a fault condition is detected. This kind of self-awareness and corrective action is essentially autonomic. VMWare FT also uses heartbeat monitoring to detect server crashes, borrowed directly from AC designs. Many physical servers employ heartbeat health monitoring (HBM) to notify IT of any hardware/software issues which are detected. Again, because it is invisible to the end user, these technologies are generally trusted, even if data migration is involved.

4) DataCentres/Cloud – cloud services, hybrid cloud and Software as a Service (SaaS) being hosted in large international datacentres have rapidly gained traction in recent years and many quality supporting applications have begun to transition to, or incorporate cloud elements, as well as an increasing number of standard business applications. Datacentres are essentially by todays standards autonomic powerhouses, with redundancy and the capability to self-configure an entire server loss with zero-touch. However, although telecoms services are generally reliable, the rarity of redundant Internet access required to use these services remains a concern.

III. CONCLUSIONS

In reviewing the state of Quality systems and AC, there seemed to be remarkable parallels between the steps and goals used in TQM and the descriptions of AC processes, such that AC can be envisioned in the various frameworks performing the same functions which are currently requiring a high level of human input. The need for more autonomicity in the various components of the quality system is apparent, but so is the difficulty of overcoming concerns around trust and accountability. There are some ethical concerns around impacts upon job satisfaction, when autonomous systems add autonomic, as it has been noted that the introduction of LIMS “led to an explosion of paperwork” and that automation “usurped” any control/autonomy a worker had and handed it directly to management [12, pp. 1-3].

It was noteworthy, that a lot of literature tends to focus upon the potential for autonomous aspects of systems, perhaps enabled by machine learning and not necessarily the autonomicity of the systems which will allow them to function independently in a trusted way.

Among emerging technologies, Blockchain is an exciting autonomic development which may gain traction in quality environments future state, where ultimate performance is not the highest concern.

Open autonomic standards [6, p.4] will continue to be a way forward and crucial to allowing Quality environments to trust and utilise AC. In the authors’ view, the achievement of AC and TQA will mature together.

Acknowledgements

This paper was produced as part of COM760 Autonomic Computing & Robotics for Ulster University’s MSc. in Artificial Intelligence.

References

[1] World Health Organization (WHO), “Good Manufacturing Practices”, Available:https://www.who.int/teams/health-product-policy-and-standards/standards-and-specifications/gmp, [Accessed 02.2024].

[2] J. Singh, and H. Singh, "Continuous improvement philosophy – literature review and directions", Benchmarking: An International Journal, Vol. 22 No. 1, pp. 75-119. Available: https://doi.org/10.1108/BIJ-06-2012-0038

[3] ISPE, "GAMP 5 A Risk Based Approach to A Risk-Based Approach to Compliant GxP Compliant GxP Computerized Systems.", 2nd ed., 2022.

[4] J. O. Kephart, and D. M. Chess, "The vision of autonomic computing," in Computer, vol. 36, no. 1, pp. 41-50, Jan. 2003, doi: 10.1109/MC.2003.1160055.

[5] P. Horn, “Autonomic Computing: IBM's Perspective on the State of Information Technology”, IBM Presentation, 2001.

[6] A. G. Ganek and T. A. Corbi, "The dawning of the autonomic computing era", in IBM Systems Journal, vol. 42, no. 1, pp. 5-18, 2003, doi: 10.1147/sj.421.0005.

[7] IBM, "An architectural blueprint for autonomic computing." IBM White Paper, 2006.

[8] M. C. Huebscher, and J. McCann, “A survey of autonomic computing—degrees, models, and applications”, ACM Comput. Surv., 40, 3, August 2008, DOI:10.1145/1380584.1380585.

[9] D. A. Menascé, M. Krishnamoorthy, and A. Brodsky, “Autonomic smart manufacturing”, Journal of Decision Systems, 24:2, 206-224, 2015, DOI:10.1080/12460125.2015.1046714

[10] M. Rolón, and E. Martínez, “Agent-based modeling and simulation of an autonomic manufacturing execution system, Computers in Industry”, Volume 63, Issue 1, 2012, pp 53-78, doi: 10.1016/j.compind.2011.10.005.

[11] M. Rolón, and E. Martínez, “Agent learning in autonomic manufacturing execution systems for enterprise networking”, Computers & Industrial Engineering, Volume 63, Issue 4, 2012, pp 901-925, DOI:10.1016/j.cie.2012.06.004.

[12] J. E. H. Stafford, "LIMS: an automating or informating technology?", Laboratory Automation & Information Management,Volume 33, Issue 3, 1998, pp 163-168, doi:10.1016/S1381-141X(98)80002-6.

[13] D. Friedli, W. Kappeier, and S. Zimmermann, “Validation of computer systems: Practical testing of a standard LIMS”, Pharmaceutica Acta Helvetiae, Volume 72, Issue 6, 1998, pp 343-348, DOI:10.1016/S0031-6865(97)00032-0.

[14] C. Ebert, and M. Weyrich, “Validation of Autonomous Systems”, InfoQ, 2021, Available: https://www.infoq.com/articles/validation-autonomous-systems/, [Accessed 02.2024].

[15] H. Alosert, et. al., “Data integrity within the biopharmaceutical sector in the era of Industry 4.0”, Biotechnology Journal, 17, 2022, DOI:10.1002/biot.202100609.

[16] T. Bonkenburg, et al. "LifeBoat: An Autonomic Backup and Restore Solution." LISA '04: Proceedings of the 18th USENIX conference on System administration, November 2004, pp 159–170.

[17] R. Mukai, S. Yamada, S. Tanaka, A. Tanaka, M. Kubota, and I. Kogiku, “A networkwide backup system with inter-memory autonomic copy mechanism”, Syst. Comp. J., 34: 2003, pp 89-99. DOI:10.1002/scj.1211

[18] S. Elnaffar, W. Powley, D. Benoit, and P. Martin, "Today's DBMSs: how autonomic are they," Proceedings of the 14th International Workshop on Database and Expert Systems Applications,. Prague, Czech Republic, 2003, pp. 651-655, doi: 10.1109/DEXA.2003.1232095.

[19] R. Muruganandham, K. Venkatesh, S. R. Devadasan, and V. Harish “TQM through the integration of blockchain with ISO 9001:2015 standard based quality management system”, Total Quality Management & Business Excellence, 34:3-4, pp 291-311, 2023, doi: 10.1080/14783363.2022.2054318

[20] R. J. T. Morris and B. J. Truskowski, "The evolution of storage systems," in IBM Systems Journal, vol. 42, no. 2, pp. 205-217, 2003, doi: 10.1147/sj.422.0205.

[21] R. Sterritt, “Autonomic networks: engineering the self-healing property”, Engineering Applications of Artificial Intelligence, Volume 17, Issue 7, 2004, pp 727-739, DOI:10.1016/j.engappai.2004.08.028.

[22] D. J. Scales, M. Nelson, and G. Venkitachalam, “The design of a practical system for fault-tolerant virtual machines”, SIGOPS Oper. Syst. Rev. 44, 4, December 2010, pp 30–39. DOI:10.1145/1899928.1899.

[23] Ronolo, S.C., "Assuring Data Integrity towards Regulatory Compliance: A Study on Process Improvement in Data Integrity Compliance of Computerized Systems.", May 2023,

[24] Laan, Sjaak. “IT infrastructure architecture-infrastructure building blocks and concepts second edition”. Sjaak Laan, 2012.