Experiences in Benchmarking of Autonomic Systems*

Xavier Etchevers, Thierry Coupaye, and Guy Vachet

Orange Labs, France Télécom Group
28 chemin du Vieux Chêne, F-38240 Meylan, France
{xavier.etchevers,thierry.coupaye,guy.vachet}@orange-ftgroup.com

Abstract. Autonomic computing promises improvements of systems quality of service in terms of availability, reliability, performance, security, etc. However, little research and experimental results have so far demonstrated this assertion, nor provided proof of the return on investment stemming from the efforts that introducing autonomic features requires. Existing works in the area of benchmarking of autonomic systems can be characterized by their qualitative and fragmented approaches. Still a crucial need is to provide generic (i.e. independent from business, technology, architecture and implementation choices) autonomic computing benchmarking tools for evaluating and/or comparing autonomic systems from a technical and, ultimately, an economical point of view. This article introduces a methodology and a process for defining and evaluating factors, criteria and metrics in order to qualitatively and quantitatively assess autonomic features in computing systems. It also discusses associated experimental results on three different autonomic systems.

Keywords: Autonomic computing, benchmark, metrics, criteria, evaluation, comparison, return on investment, ROI.

1 Introduction

From an industrial perspective, the overall motivation underlying the emergence of autonomic computing is based on the observation that the costs related to the IT infrastructures are quickly and massively migrating from new investments (development and licensing costs) to maintenance expenses (deployment and exploitation costs). [10] This evolution illustrates the transformation of computing systems in terms of size, distribution, sophistication, dynamism, heterogeneity and interoperability that results in even more complex –and thus expensive– management tasks. In this context, autonomic computing aims basically insofaras possible at automating the deployment and management (administration) of computing systems in order to reduce human interventions and all associated costs.

* This work is partially funded by European IST FWP6 Selfman project [5].

A.V. Vasilakos et al. (Eds.): AUTONOMICS 2009, LNICST 23, pp. 48–63, 2010.

Bit by bit research in autonomic computing is starting to generate industrial solutions. This is the case, for example, of workload management in J2EE web applications servers (e.g. JOnAS, JBoss, WebSphere, WebLogic). However, even in such a well-known area, the autonomic features embedded in different products do not necessarily address the same problems and/or can have different maturity levels. Moreover, even if some experimental results like [2], [14] or [17] tend to demonstrate that autonomic behaviors[1] (ABs) improve systems efficiency (by comparing it when enabling and disabling the autonomic features), there are still no models and no tools for formally measuring and comparing the technical and economical benefits these autonomic capabilities are supposed to offer.

Nowadays industrial companies continuously strive to improve and simplify their process for increased customer satisfaction, operational efficiency and cost reduction. This implies improved efficiency of computing systems in terms of productivity, performances, quality of service (QoS), trust, etc. Autonomic computing appears to be an appealing solution. However this quite recent and disruptive approach still raises a number of issues regarding:

(technical) efficiency: can the impact of autonomic computing on the quality of service (QoS)[2] be exhibited or even "proved" - especially when ABs are built into the system? Is this impact measurable generically with respect to various business fields, architecture or implementation of different systems?

profitability (i.e. economical efficiency): when comparing all the costs linked to an autonomic system versus a non-autonomic equivalent one, does autonomic computing come out as a cheaper or a more expensive technology? How long will it take to become profitable?

applicability: considering the previous questions, are there some more or less pertinent areas to which autonomic computing should or should not be applied?

Such questions raise the need for some autonomic computing benchmark(s) that would allow for evaluation and comparison of autonomic systems, both from a technical and an economical perspective. This article hypothesizes that, even if developing benchmarks are business-specific (i.e. specific to web applications servers or to P2P systems for the benchmarks considered in this article), some generic benchmarking elements could be shared among all specific domains and therefore could allow for inter-area comparison. The contribution of this paper is twofold. First, it introduces a methodology and a process for defining and evaluating factors, criteria and metrics in order to qualitatively and quantitatively assess autonomic features in computing systems. Second, it analyzes associated experimental results on three different autonomic systems. The article is organized as follows. Section 2 discusses existing works on autonomics benchmarking. Section 3 defines a methodology and a process, based on criteria and metrics, in order to benchmark autonomics both qualitatively and quantitatively. Section 4 describes and analyzes experimental results. Section 5 concludes.

[1] *Autonomic behavior* (AB) designates the implementation of a particular control loop or MAPE-loop.

[2] In this article, the term 'QoS' is used in its most general meaning.

2 Background

2.1 Models and Metrics

Since autonomic computing aims essentially at improving the QoS of systems, [15] and [19] try to define an autonomic computing evaluation model based on the ISO/IEC 9126 standard[3] [13]. They describe the qualitative binding between autonomic characteristics[4] and the ISO/IEC 9126 factors (see figure 1). At the same time, [11] provides a qualitative hierarchy between the eight autonomic characteristics (the four main ones and the four secondary ones) (see figure 2). Finally some works focus on the definition of metrics in order to evaluate autonomic capabilities. For example, [12], [3] and [4] propose a non-exhaustive set of criteria and metrics that are not related to any evaluation standard such as ISO/IEC 9126.

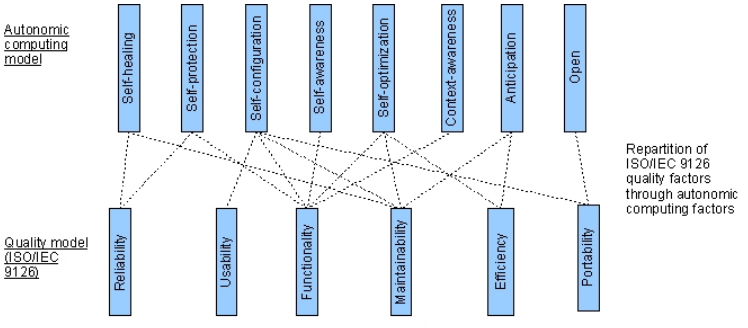

Fig. 1. Organization of autonomic computing characteristics based on ISO/IEC 9126 standard quality factors [15] [19]

These works constitute indispensable steps toward providing a complete autonomics benchmark, but they can be characterized by a qualitative and fragmented approach. [15] and [19] do not provide a quantitative composition of the factors for evaluating experimentally the autonomic characteristics, nor an adaptation of ISO/IEC 9126 model to the autonomic computing field. [11] is not based on any Factors-Criteria-Metrics model and still has to be quantitatively validated. Finally, [12], [3] and [4] list some criteria and metrics. However, for most of criteria (i.e. granularity, flexibility, robustness, adaptability), no metrics

[3] ISO/IEC 9126 is an adaptation of a generic Factors-Criteria-Metrics model [6] applied to the software quality field. However it defines only factors and criteria because software quality metrics are specific to the business area of the considered system.

[4] [7] and [9] define autonomic computing thanks to four main characteristics (self-configuration, self-healing, self-protection and self-optimization) and four secondary characteristics (self-awareness, context-awareness, openness and anticipation).

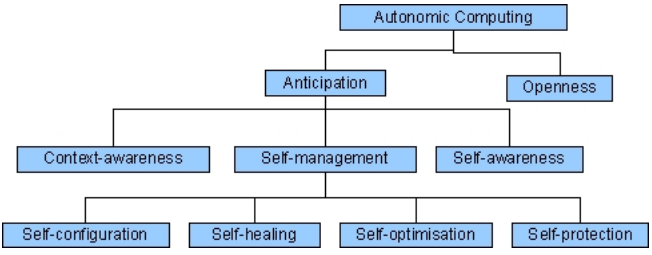

Fig. 2. Hierarchy between autonomic computing characteristics [11]

have been defined. Moreover the proposed metrics are either qualitative (i.e. degree of autonomy) and therefore their evaluation is subjective, or quantitative (i.e. adaptation time, reaction time, stabilization duration, latency) but they remain difficult to measure because of their abstraction level.

2.2 Benchmarking Methodologies and Tools

[1] lists the specificities associated to the benchmarking of autonomic features. A "classical" performance benchmark and an autonomic computing benchmark differ along three main axes: 1) environment stability that is questioned for autonomic computing by the injection of disturbances, 2) management interactions that must not occur in a classical performance benchmark, but that constitute the AB as well, 3) the antagonism between test realism (regarding the representativeness of the workload or the disturbances load the system under test will have to face) and the benchmark requirements, especially in terms of reproducibility, cost and legality. Among the works dealing with the experimental evaluation of autonomics efficiency, [2] seems to be one of the most advanced. It describes one of the first autonomic computing benchmark dealing with self-healing evaluation. It details its experimental protocol for validating the tool. This consists in measuring the impact of thirty different classes of disturbances on two metrics. The disturbances are sequentially injected.

The three main specificities of an autonomic computing benchmark (highlighted by [1]) illustrate the addition of a second dimension into an autonomic computing benchmark compared to a classical QoS benchmark. Each benchmark can indeed be associated with a function that get injection profiles as inputs and that returns a vector of evaluated metrics. Thus a classical QoS benchmark is a function with a single input (i.e. the injected workload profile) whereas an autonomic computing benchmark consists in a function getting two inputs (i.e. the injected workload profile and the injected disturbance profile). Concerning this last type of benchmark, constraints relative to costs, reproducibility and legality, which are antagonistic to test realism, can be declined into constraints on the injection profiles (synchronization of workload and disturbances injections, stability of workload injection, etc.). In other words an autonomic computing

benchmark consists of two coupled evaluation tools. The first one is dedicated to the QoS measurement: it is business specific and is essentially made of quantitative metrics. The second one focuses on the evaluation of self-management behaviors. It should include only generic (i.e. domain independent) aspects. Concerning the experimental assessment, works like [2], [14], [17], although they tend to demonstrate that autonomic features improve systems efficiency, do not achieve to define a scale offering a synthesized and absolute[5] view. Thus [2] and [3] define a value ranging between 0 (non autonomic) and 1 (fully autonomic) for assessing self-* features. However this indicator is obtained by restraining the number of different.disturbances.

3 Assessment Methodology and Process

This section introduces the first contribution of this article, namely the proposed generic benchmarking methodology and process.

3.1 Methodology

The first step of the proposed approach is to define the constituents, i.e. *factors*, *criteria* and *metrics*, of an hybrid (refined) ISO/IEC 9126 model for the autonomic computing area, similarly to the refinement of ISO/IEC 9126 model proposed in [18] and addressing the concrete case of test specification. Then high-level indicators are defined for qualifying autonomic features. A bottom-up approach integrating, supplementing and adapting some existing works (see section 2) is adopted.

Metrics. An exhaustive set of metrics participating in the empirical evaluation of the ISO/IEC 9126 model has been identified (by using and enriching the list defined by [12], [3] and [4]). These metrics have to be:

Generic, i.e. independent from the field of application, in order to be applied to evaluating various business areas or to inter-domain comparisons;

Measurable, i.e. these metrics can be assessed independently from architecture, design, implementation or technologies choices;

Quantifiable for metrics processed as inputs of composition functions (see below) in order to calculate quantitative higher-level indicators. Quantitative metrics are intrinsically quantifiable whereas qualitative ones are characterized by the subjectivity of their evaluation. However some of qualitative metrics can be composed of discrete but ordered values (e.g. the level of maturity of an AB) whereas others are made of non ordered values (e.g. the list of standards or technologies with which a component complies). Only the first ones are quantifiable.

[5] *Absolute* is relative to a scale that could classify the results from a system without any autonomic features to an idealistic system that could autonomously anticipate or deal with any disturbances (without any delay and any impact on the quality of service).

Composition Functions. After having qualitatively defined a hybrid ISO/IEC 9126 model for autonomic computing (see figure 3)[6] and tied it to the proposal of [15] and [19] concerning the coupling between autonomic characteristics and software quality factors, the next step consists in defining composition functions for enabling the computation of higher-level quantitative indicators. Among these indicators some have to be compared –and be thus quantifiable– whereas others only relate interesting properties. As for an autonomic computing benchmark, the four main self-* characteristics, i.e. *self-configuration, self-optimization, self-healing, self-protection*, have to be quantifiable so as to be compared, whereas *self-awareness, context-awareness, openness* do not directly impact ABs efficiency [7] . In order to get the property of *absoluteness* of the comparison scale (i.e. to avoid a restriction of the events set that can occur, see section 2.2), the value I of each quantifiable high-level indicators is ranged between 0 (non autonomic) and $+\infty$ (idealistic fully autonomic). This value I will be obtained by adding the result of the corresponding composition function f_I applied on each event e handled by the system and weighted with the occurrence probability p_e of this event:

$$I = \sum_e p_e f_I(e) \tag{1}$$

The indicator value highlights the wideness of the events spectrum the system is able to compute and the utility of dealing with each event.

3.2 Qualitative Assessment

Table 1 summarizes criteria and metrics defined for qualitative assessment of ABs.

3.3 Quantitative Assessment

Table 2 summarizes the criteria and metrics defined for quantitative assessment of ABs.

Four separate time measurements –monitorability, analyzability, planning capability and changeability– that are mapped on each phase of the widely accepted concept of MAPE-loop (see figure 4) are defined. This decomposition offers two main advantages. On the one hand, these quantitative (i.e. per se quantifiable) metrics are generic and measurable because they are based on the control loop that is a fundamental concept of ABs implementation. On the other hand, they can be separately measured, according to the system maturity, because of the use of a common concept (i.e. the stages of the MAPE-loop) for defining these metrics and the maturity levels.

[6] Efficiency factor and associated criteria and metrics are not, strictly speaking, constituents of this model but they appear on this figure in order to highlight the relationship between this hybrid model for autonomic computing and business specific metrics.

[7] Whether *anticipation* is or is not a quantifiable high-level indicator is still to be determined.

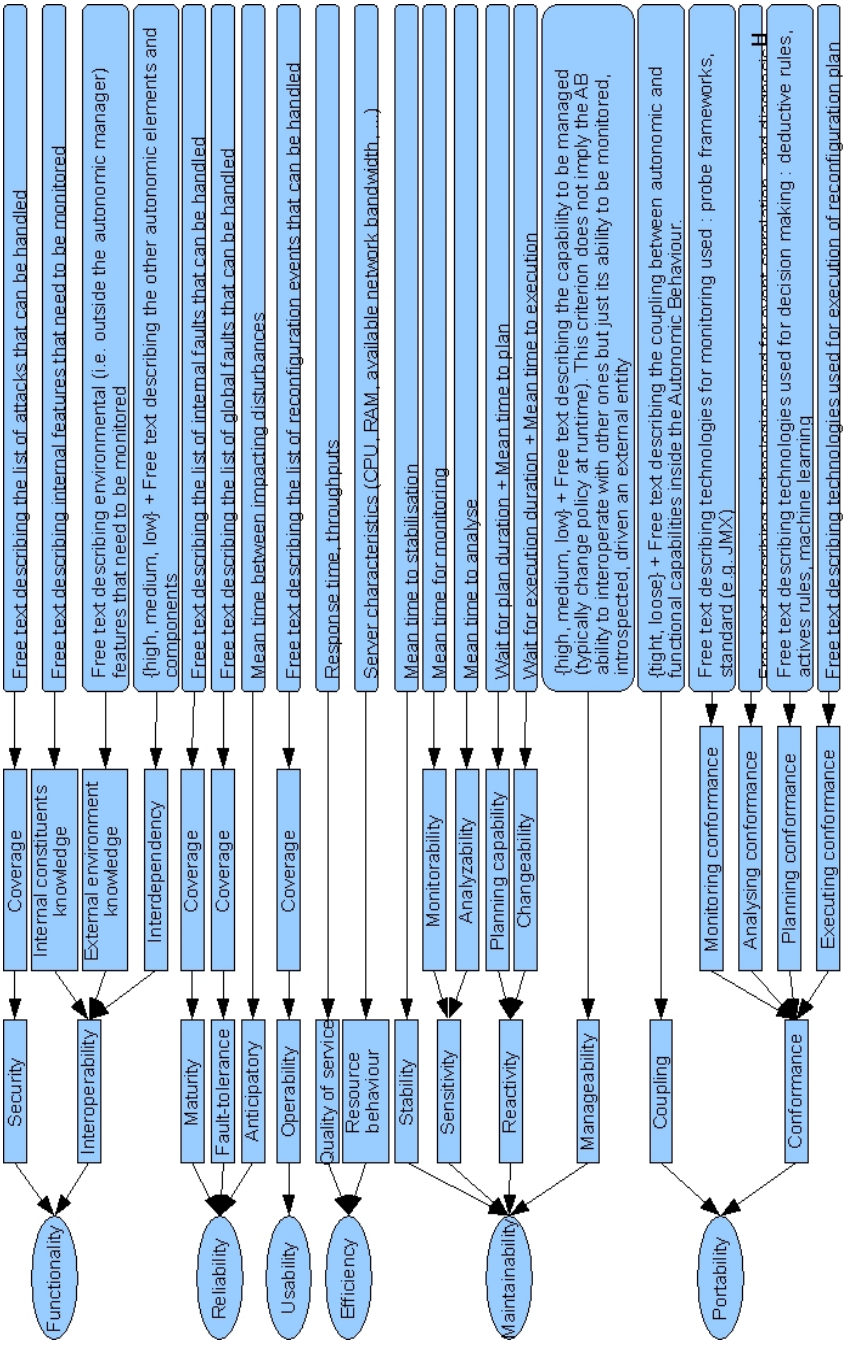

Fig. 3. Quality model for autonomic computing assessment

Table 1. Qualitative criteria for ABs assessment

Criterion	**Criterion type**
Description of AB	Free text. Short description of the AB.
Related to self-* characteristic	{self-configuration, self-healing, self-optimization, self-protection}
Coverage	Free text describing the list of disturbances the AB is able to deal with
Interdependency	high, medium, low + Free text. Description of the other ABs and components the current AB is depending on.
Internal constituents knowledge	Internal features that need to be monitored for this AB
External environment knowledge	Environmental (i.e. outside the AB) features that need to be monitored for this AB
Level of automation	{-, M, MA, MAP, MAPE}
Monitoring compliance	Free text. Technologies used for monitoring: probe frameworks, standard (e.g. JMX)
Analyzing compliance	Free text. Technologies used for event correlation and diagnosis
Planning compliance	Free text. Technologies used for decision making: deductive rules, actives rules, machine learning
Executing compliance	Free text. Technologies used for execution of reconfiguration plan
Coupling	tight, loose + Free text. Description of coupling between autonomic and functional capabilities inside the AB.
Manageability	high, medium, low + Free text. Capability to be managed (typically change policy at runtime). This criterion does not imply the AB ability to interoperate with other ones but just its ability to be monitored, introspected, driven an external entity.

Table 2. Metrics for quantitative assessment of ABs

Criterion	**Sub-criterion**	**Metric**
Sensitivity	Monitorability	Mean time for monitoring
	Analyzability	Mean time to analyze
Reactivity	Planning capability	Wait time for plan duration + Mean time to plan
	Changeability	Wait time for execution duration + Mean time to execution
Anticipation		Mean time between impacting disturbances
Stability		Mean time to stabilization

Anticipation measures the system ability to deal with events while maintaining its QoS at a satisfying level: the human administrator defines an interval of satisfaction for QoS values. Thus, among the overall disturbances the system is

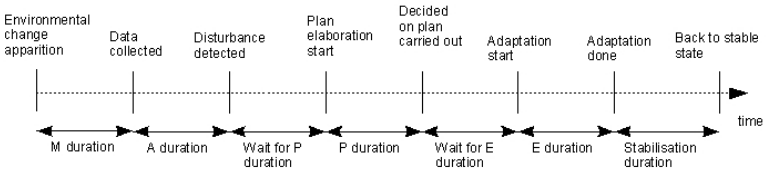

Fig. 4. Mapping between MAPE-loop stages and their duration

able to compute, the impacting ones are those which modify the system efficiency (QoS) outside this interval.

Stability measures the time the system needs to return to a stable state, i.e. a state for which QoS values are stable.

Test Process. [2] proposes a three-step test process for evaluating ABs.

1. First a workload, that will be maintained constant during all the three steps, is injected in the system under test (SUT). This first step lasts until the system reaches a stable state regarding its QoS metrics.
2. Then a single disturbance is injected in order to trigger an autonomic reaction of the SUT.
3. The last step consists in observing metrics (related to QoS and autonomic computing) until the SUT returns to a stable state (possibly different from the initial one) regarding the QoS metrics.

This test process measures autonomic features efficiency. It exhibits some kind of reproducibility property (see section 2.2) coming from the stability of the workload injection during all the test duration. However due to this stability and to the uniqueness of the injected disturbance, this test process does not fit the realism property (see the third difference between a "classical" benchmark and an autonomic computing benchmark in section 2.2). Notwithstanding, this process is adopted for quantitative experimentations. The injected disturbance is considered as a default for evaluating self-healing, an attack for evaluating self-protection, an extra constant workload for evaluating self-optimization and an event triggering a reconfiguration for evaluating self-configuration (e.g. *churn* in P2P systems).

3.4 Economical Assessment (Return on Investment)

As mentioned in section 1, the ultimate goal of autonomic benchmarking, from an industrial point of view, is to define means for evaluating the return on investment (ROI) of developing and deploying autonomic computing technologies. Once the technical evaluation stage is over, available data relative to the economical cost management (i.e. cost of licenses, disturbance rate and frequency, mean time to resolve a disturbance, human administrator salary, energy consumption, etc.) can be collected. Then these economical metrics will be placed in the autonomic computing model in order to obtain an estimation of the economical

return on investment. The economical evaluation could be carried out by using utility functions as proposed in [16]. Utility functions can be seen as composition functions whose result is a financial value expressed in a given currency. Utility functions might allow for comparison of the economical efficiency of different autonomic features in the system under consideration or in different autonomic systems (e.g. "is there more ROI to be expected by introducing autonomic features in web applications servers or machine-to-machine platforms?").

4 Experimental Results

This section introduces and discusses the experimental results that have been obtained by applying the methodology described to eleven ABs in three autonomic systems. The three systems considered cover the architectural classification of autonomic systems proposed by [12]:

- wide distributed systems composed of a large number of independent collaborative intelligent nodes. In our case, a peer-to-peer (P2P) transactional storage system, namely Scalaris from ZIB (Zuse Institute Berlin) designated as SUT1, and a P2P system on mobile phones, namely the gPhone application from UCL (Université Catholique de Louvain) designated as SUT2. Both are based on a common structured overlay network (SON). SUT1 and SUT2 implement eight different ABs (some are common, some are specific). SUT1 and SUT2 are research-oriented and have been developed in the EU funded IST Selfman project [5].
- more centralized systems composed of a hierarchy of components. In our case, an industrial workload management application, designated as SUT3[8] based on a centralized manager and a cluster of JOnAS [8] J2EE application servers. SUT3 implements three ABs.

4.1 Qualitative Assessment

The qualitative assessment consisted in filling out each line (i.e. computing each qualitative metric) of the table 1 in the context of each AB. Altogether, eleven tables have been obtained (eight for SUT1 and SUT2 and three for SUT3) that make up dozens of pages in Selfman project deliverables concerning SUT1 and SUT2. Due to space limitation in this article, results have been summarized in table 3 that reports the major tendencies highlighted by these individual assessments.

Among these major tendencies, some are common to all evaluated ABs. The lack of self-protection ABs is explained by the systems editors as follows: some of them assume that their solution runs in a safe environment whereas others estimate that security has to be delegated to another system contributor. However a further explanation is that self-protection ABs have to deal with more complex events and thus cannot be elementary behaviors, exclusively focusing

[8] For motives of confidentiality, the name of this solution has not been mentioned.

Table 3. Synthesis of qualitative assessment on eleven ABs coming from SUT1, SUT2 and SUT3

Criterion	**Tendency**
Related to self-* characteristic	None of the assessed ABs is related to self-protection. Moreover, the distinction between self-optimization and self-configuration is not self-evident.
Coverage	All of these ABs can be considered elementary: they only deal with one or two low-level events ('peer joining' or 'excessive response time to a request' for instance)
Interdependency	Only one of the ABs interoperates with another one but this interaction is limited to a single query/reply and does not consist in a structured dialog
Internal constituents knowledge	All these ABs run according to the monitoring data they get from the resources they manage
External environment knowledge	All these ABs have very little (or even no) knowledge of their external environment
Level of automation	However all are fully autonomous regarding their level of maturation: none of them claims to be autonomic without fully implementing the four stages of the MAPE-loop
Monitoring compliance	Concerning their compliance with a standard and/or a wide spread / opened technology, all ABs propose highly proprietary implementation for this MAPE-loop stage
Analyzing compliance	Concerning their compliance with a standard and/or a wide spread / opened technology, all ABs propose highly proprietary implementation for this MAPE-loop stage
Planning compliance	Concerning their compliance with a standard and/or a wide spread / opened technology, all ABs propose highly proprietary implementation for this MAPE-loop stage
Executing compliance	Concerning their compliance with a standard and/or a wide spread / opened technology, all ABs propose highly proprietary implementation for this MAPE-loop stage
Coupling	All the ABs of SUT1 and SUT2 are tightly coupled with the business functionalities in term of implementation whereas this coupling is loose concerning the three ABs of SUT3
Manageability	Three of the eleven ABs include manageability capabilities. However the management policy elements they can get are quite rudimentary (like a timer value or a combination of simple conditions). This illustrates the difficulty in converting high-level management policies into simpler rules understandable by the autonomic managers

on their internal resources. They might result from the composition (implying sophisticated interoperability) between different lower-level ABs. Thus the composition of ABs becomes an important research domain. However the lack of compliance with open standards, which characterizes elementary ABs, prevents any progress on their interoperability.

The main difference resulting from these qualitative assessments concerns the coupling between autonomic features and application specific (business) functionalities. This is due to:

systems architecture: SUT1 and SUT2 lie on a wide distributed multi-peers architecture whereas SUT3 is based on a centralized and hierarchical architecture;
development history: SUT1 and SUT2 have been developed from scratch whereas SUT3 consists in a clustered workload management layer added to an existing of J2EE application server.

4.2 Quantitative Assessment

Again, due to space limitation, this section only focuses on a single AB that illustrates the kind of results obtained by applying the quantitative metrics defined in table 2 on all implemented ABs. This AB concerns self-optimization in SUT3. It aims at increasing or reducing the number of physical machines (nodes of a cluster of JOnAS application servers) according to a level of workload addressed to a given web application. The workload manager is centralized and, in this example, is deployed on a dedicated machine. All requests addressed to the given web application are routed through the workload manager. The experimentation consists in injecting an extra workload causing a violation of the management policy concerning the response time limitation.

An evaluation methodology close to the one described in [2] (see section 3.3) has been used for this benchmark. Preliminarily the workload threshold at which the manager decides to release the AB, has been determined. This depends mainly on the material configuration of the test platform. A value of 50 requests per second (req/s) has been obtained. Then, the same three-step approach has been applied to each different experimentation.

1. The first stage consists in submitting SUT3 to the maximum workload a single J2EE application server can handle without breaching the response time limitation (set to 100 ms).
2. Then, after having achieved the stabilization of response time, an extra workload is injected.
3. The last step focuses on the observation of response time (QoS metric) and of the different autonomic durations until the system returns to a stable state (with a satisfying response time).

For any SUT, it is possible to draw a figure showing the impact of the injected disturbance on the QoS. According to the autonomic characteristic –and thus to the type of disturbance– different injection profiles can be defined. In the field of self-optimization for instance, the injection profile of extra workload consists in setting its rate and its emerging speed. Figure 5 illustrates the three-step test protocol and shows the impact of an extra workload on SUT3 response time. In this example, the injection profile is characterized by a rate of 20% (+10 requests per second) and an arising speed of 1 second (i.e. a step profile).

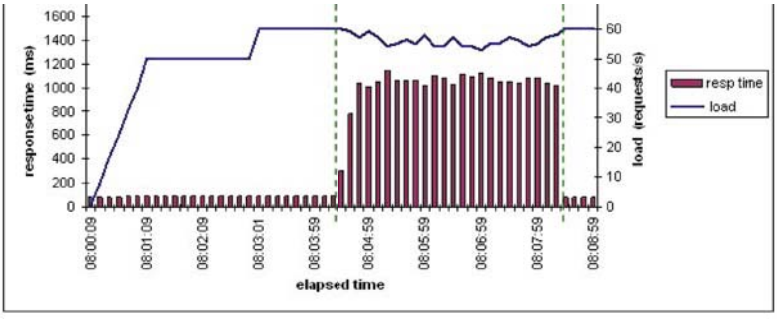

Fig. 5. Impact of an extra workload on SUT3 response time

Table 4. Quantitative evaluation of resource optimization of SUT3

	Workload profile 1	Workload profile 2	Workload profile 3
Extra workload rate	20% - 10 req/s	20% - 10 req/s	40% - 20 req/s
Extra workload arising speed	1 s - 10 req/s^2	60 s - 0.17 req/s^2	60 s - 0.33 req/s^2
M + A durations	61 s.	64 s.	66 s.
Wait for P duration	17 ms.	24 ms.	27 ms.
P duration	485 ms.	475 ms.	475 ms.
Wait for E duration	2 ms.	4 ms.	15 ms.
E duration	136 s.	136 s.	131 s.
Stabilization duration	46 s.	53 s.	39 s.

Table 4 synthesizes the results obtained by applying three different extra workload profiles. Autonomic computing durations were not obtained graphically but through the analysis of workload manager log files.

These results lead to three major observations.

1. Monitoring and analysis durations[9] are amazingly high whereas the time lag for planning is extremely short. A first explanation might conclude that monitoring and analysis are really inefficient. However this is not the case. It results from two causes. On the one hand SUT3 provides a mechanism for avoiding hyper-sensitivity (i.e. reacting too frequently): the violation of response time limitation has to last at least one minute before triggering a reaction plan. SUT3 is effectively an industrial solution whose objectives are an hourly optimization. It does not find any interest in a finer grained precision for resource optimization. On the other hand, SUT3 considers that the disturbance consists of the extra workload injection and its continuation during a configured duration (at least one minute) whereas in the approach proposed in this article, the disturbance consists only in the extra workload

[9] Data collected from the SUT3 benchmark did not make possible the distinction between monitoring and analysis durations.

(the configured duration defining the minimal length of the wait for planning duration).
2. Another observation is that the self-optimization behavior is independent from the injected disturbances profiles. SUT3 does not seem to include anticipation features because it remains insensitive to changes of arising speed or rate of extra workload (see table 4).
3. Stabilization duration is quite long. It confirms that this measure remains essential for evaluating the risk the system tends to remain a long time in an unstable state.

4.3 Discussion

This section discusses the proposed methodology for evaluating autonomic computing whereas section 4.1 and section 4.2 focus on presenting and analyzing the results coming from the technical benchmark.

On the one hand, the methodology used for qualitative benchmark is relatively generic: it has been applied successfully, especially as far as qualitative assessment is concerned, to eleven ABs coming from three applications that differ in business areas, architectures and implementations.

On the other hand, our experiments question the genericity of the approach, and in particular of the test process (see section 3.3), with respect to system architecture. Indeed, although the process is adapted for evaluating systems for which QoS and autonomic metrics are evaluated at the same abstraction level (this is the case of autonomic systems with a hierarchical architecture of components, like SUT3), it is not so adequate for systems where QoS and autonomic metrics are measured at different levels of abstraction, like in a P2P systems (such as SUT1 and SUT2) where ABs are local and QoS is evaluated at a macroscopic level. In the latter situation, an injected disturbance will trigger an autonomic adaptation that could have no measurable effect on the QoS (due to the important number of peers contributing to the QoS). A solution could be to define a methodology for composing the results obtained at the peer (local) level in order to extrapolate the impact of ABs at a global level. This is one of the subjects that needs further investigations.

Finally the definition of anticipation and stabilization durations independently from the domain of application and the architecture is also a challenge for a quantitative autonomics benchmark. It has been experimented indeed that for now, in some cases, the stabilization duration can be difficult or even impossible to measure (for example in an AB running periodically without any “real” triggering event).

5 Conclusion

This article has introduced a methodology and a process for qualitative and quantitative assessment of autonomic features in computing systems together with associated experimental results on eleven autonomic behaviors in three

different autonomic systems (two P2P systems and one n-tiers system). Our experiments provide an uneven feedback that altogether questions the reachable genericity of autonomic benchmarking and the possibility of evaluating autonomic systems in which autonomic features are deeply built-in (e.g. in the P2P systems considered in this article).

Further investigations related to this work are needed so as to validate or to invalidate these first feedbacks and:

- to improve the ways of measuring some criteria, such as anticipatory or stabilization;
- to work out a methodology for composing elementary ABs in order to obtain a higher level behavior and allow the comparison of QoS metrics and autonomic computing indicators at the same level of abstraction;
- to define ways of coupling cost models with a technical benchmark. This would enable to evaluate the return on investment of autonomic computing and to determine the pertinence of applying such a disruptive technology to different fields of application.

Since benchmarking tools have already managed to provide important improvements on the efficiency of microprocessors or middleware as mentioned in [1], autonomics benchmarking is expected to constitute a decision-making tool for IT managers to assess and hopefully foster the adoption of autonomics in industry - but this is not an easy road.

References

1. Brown, A.B., Hellerstein, J.L., Hogstrom, M., Lau, T., Lightstone, S., Shum, P., Yost, M.P.: Benchmarking Autonomic Capabilities: Promises and Pitfalls. In: 1st International Conference on Autonomic Computing, pp. 266–267. IEEE Computer Society, New York (2004)
2. Brown, A.B., Redlin, C.: Measuring the Effectiveness of Self-Healing Autonomic Systems. In: 2nd International Conference on Autonomic Computing, pp. 328–329. IEEE Computer Society, New York (2005)
3. Chen, H., Hariri, S.: An Evaluation Scheme of Adaptive Configuration Techniques. In: 22nd IEEE/ACM International Conference on Automated Software Engineering, pp. 493–496. ACM Press, New York (2007)
4. De Wolf, T., Holvoet, T.: Evaluation and Comparison of Decentralized Autonomic Computing Systems. Technical report. Department of Computer Science, K.U. Leuven, Leuven, Belgium (2006)
5. European IST 6th FWP Selman project (self management for large-scale distributed systems based on structured overlay networks and components), `http://www.ist-selfman.org/wiki/index.php/Selfman_project`
6. Forse, T.: Qualimétrie des Systèmes Complexes - Mesure de la Qualité du Logiciel (Qualimetry of Complex Systems - Measurement of Software Quality). Editions d'organization (1989)
7. Horn, P.: Autonomic Computing: IBM's Perspective on the State of Information Technology,
`http://www.research.ibm.com/autonomic/manifesto/autonomic_computing.pdf`

8. JOnAS OpenSource Java EE Application Server, http://jonas.ow2.org/
9. Kephart, J.O., Chess, D.M.: The Vision of Autonomic Computing. IEEE Computer 1, 41–50 (2003)
10. Kluth, A.: Make It Simple. The Economist (2004-10-28)
11. Lin, P., MacArthur, A., Leaney, J.: Defining Autonomic Computing: A Software Engineering Perspective. In: 16th Australian Software Engineering Conference, pp. 88–97. IEEE Computer Society, New York (2005)
12. McCann, J.A., Huebscher, M.C.: Evaluation Issues in Autonomic Computing. In: Grid and Cooperative Computing - GCC 2004 Workshops: GCC 2004 International Workshops, IGKG, SGT, GISS, AAC-GEVO, and VVS, pp. 597–608. Springer, Heidelberg (2004)
13. Milicic, D.: Software Quality Models and Philosophies. In: Lundberg, L., Mattsson, M., Wohlin, C. (eds.) Software Quality Attributes and Trade-offs, p. 100. Blekinge Institute of Technology (2005)
14. Oyenan, W.H., DeLoach, S.A.: Design and Evaluation of a Multiagent Autonomic Information System. In: 2007 IEEE/WIC/ACM International Conference on Intelligent Agent Technology, pp. 182–188. IEEE Computer Society, New York (2007)
15. Salehie, M., Tahvildari, L.: Autonomic Computing: Emerging Trends and Open Problems. ACM SIGSOFT Software Engineering Notes 4, 1–4 (2005)
16. Walsh, W.E., Tesauro, G., Kephart, J.O., Das, R.: Utility Functions in Autonomic Systems. In: 1st International Conference on Autonomic Computing, pp. 70–77. IEEE Computer Society, New York (2004)
17. Wildstrom, J., Stone, P., Witchel, E.: Autonomous Return on Investment analyzis of Additional Processing Resources. In: 4th International Conference on Autonomic Computing, p. 15. IEEE Computer Society, New York (2007)
18. Zeiss, B., Vega, D., Schieferdecker, I., Neukirchen, H., Grabowski, J.: Applying the ISO 9126 Quality Model to Test Specifications - Exemplified for TTCN-3 Test Specifications. In: Software Engineering 2007, Fachtagung des GI-Fachbereichs Softwaretechnik, pp. 231–244. GI (2007)
19. Zhang, H., Whang, H., Zheng, R.: An Autonomic Evaluation Model of Complex Software. In: International Conference on Internet Computing in Science and Engineering, pp. 343–348 (2008)