Multi-Criteria Analysis of Measures in Benchmarking: Dependability Benchmarking as a Case Study

Jesús Friginal[a], Miquel Martínez[b], David de Andrés[b], Juan-Carlos Ruiz[b]

jesus.friginal@laas.fr, {mimarra2, ddandres, jcruizg}@disca.upv.es

[a]*LAAS-CNRS, 7 Avenue du Colonel Roche. 31077 Toulouse Cedex, France*
[b]*STF-ITACA Universitat Politècnica de València, Campus de Vera s/n, 46022, Spain*

Abstract

Benchmarks enable the comparison of computer-based systems attending to a variable set of criteria, such as dependability, security, performance, cost and/or power consumption. Despite its difficulty, the multi-criteria analysis of results remains today a subjective process rarely addressed in an explicit way in existing benchmarks. It is thus not surprising that industrial benchmarks only rely on the use of a reduced set of easy-to-understand measures, specially when considering complex systems.This is a way to keep the process of result interpretation straightforward and unambiguous. However, it limits at the same time the richness and depth of the analysis process. This is why the academia prefers to characterize complex systems with a wider set of measures. Marrying the requirements of industry and academia in a single proposal remains a challenge today. This paper addresses this question by reducing the uncertainty of the analysis process using quality (score-based) models. At measure definition time, these models make explicit (i) which are the requirements imposed to each type of measure, that may vary from one context of use to another, and (ii) which is the type, and intensity, of the relation between considered measures. At measure analysis time, they provide a consistent, straightforward and unambiguous method to interpret resulting measures. The methodology and its practical use are illustrated through three different case studies from the dependability benchmarking domain, which usually consider several different criteria including both performance and dependability ones. Although the proposed approach is limited to dependability benchmarks in this document, its usefulness for any type of benchmark seems quite evident attending to the general formulation of the provided solution.

Keywords: Multiple-Criteria Decision Making (MCDM), Logic Score of Preferences, Dependability benchmarking, Quality models

1. Introduction

Benchmarks are well-known tools to compare and select distributed systems mainly attending to their performance, cost and power consumption. Standardization bodies, such as the Transaction Processing Performance Council [1], currently propose a set of representative (since widely accepted by the community) benchmarks for distributed systems. In the last decade, some initiatives have addressed the challenging goal of including the evaluation of dependability and security properties in conventional benchmarks. Resulting benchmarks are typically called dependability benchmarks.

Like in conventional benchmarks, controllability and repeatability of experiments and interpretation of results are essential in dependability benchmarks [2], [3], [4]. To date, most of the efforts done in the community around this topic have been oriented towards providing controllability and repeatability of experiments. These efforts can be understood given the need to obtain the same (or at least statistically similar or comparable) experimental measures when the same experimental setup is considered.

However, and without taking importance away from this point, controllability and repeatability also affects other stages of the benchmarking process, such as the analysis of results. The reader should understand that dependability benchmarks introduce the need of performing a more complex analysis of target systems, considering their behavior in the presence of faults and attacks, and characterizing such behavior with a larger set of measures, including dependability and security specific ones. This evidence becomes a challenge when considering the evaluation of complex systems formed by a large and heterogeneous set of sub-systems and components. This is a challenge not only for the amount of measures to consider, but also for their variety of origin and typology.

To date, the analysis of results from dependability benchmarks has been an aspect strongly relying on the human factor. Evaluators subjectively interpret measures following considerations that are usually omitted in the finally generated reports. In consequence, repeating the same analysis of measures and obtaining the same conclusions, even when results are the same, becomes sometimes a complex task.

The underlying problem is that most proposals limit their purpose to the delivery of benchmark measures. In deed, the consideration of a representative set of measures has been traditionally enough to justify their selection for benchmarking purposes [5]. Then, the analysis of such measures, and consequently the related comparison of alternatives, is typically considered outside the purpose of the specification of most benchmarks, including dependability benchmarks. This can be something acceptable in the context of conventional benchmarks but it is unaffordable in the case of dependability benchmarks, since any aspect leading to a wrong alternative selection may have a negative impact on the safety or security of the system, with the subsequent losses, in the case of critical systems, of reputation, money or lives.

On the one hand, benchmark measures must be contextualized during the analysis process. Without contextualizing their meaning throughout factors such as the environment, the type of system targeted, or the evaluation performer, same results may have different interpretations depending on the evaluation consumer's subjectivity. On the other hand, it must be clearly specified in the analysis process which are the relations considered among measures, and the intensity of such relations. Otherwise, it may be very difficult to guess which have been all the assumptions adopted by someone analyzing a set of benchmark measures. In other words, it may be difficult to verify the conclusions issued from the analysis of a set of benchmark measures.

It is worth mentioning that even if all this effort is done, the analysis and interpretation of results remains an error-prone process requiring a very deep dependability expertise, in the case of dependability benchmarks. This situation increases the *uncertainty* of evaluation analyses and thus negatively affects the credibility of the conclusions obtained. This ambiguous interpretation of concepts is commonly known as *semantic heterogeneity* [6].

This challenge could be addressed through a process of *semantic reconciliation* [6]. Such process involves covering the existing gap between the explicit result of the evaluation, that is, the conclusions distilled from the analysis of measures, and the implicit real intention of evaluators, which concerns the interpretation procedure to obtain such conclusions. This fact increases the sensitivity of analyses, potentially revealing surprising insights about the system under evaluation. This approach is specially useful when there is no obvious optimal (or unanimous) solution due to the large number of criteria that need to be taken into account, or when decisions often require the fulfillment of conflicting objectives (e.g., design or choice of systems maximizing their dependability or performance). It has also the potential for improving the work of system evaluators by leading them to unequivocal and more objective conclusions. Unfortunately, to date, *semantic reconciliation* remains a non-addressed issue in the domain of distributed systems dependability benchmarking.

The main novelty of this paper relies on a double fact. First, providing a multi-criteria analysis methodology to ease the multiple interpretations that the measures issued from dependability benchmarks may have depending on the criteria followed by evaluators. The goal of this methodology is to make explicit the subjective interpretation rules that evaluators typically apply implicitly when determining to what extent measures satisfy evaluation requirements. Doing this in a systematic and repeatable way is essential when different evaluators need to make a fair comparison of their results, so the methodology relies on a mathematical formalism. Second, defining our methodology in such a way it may satisfy the conflicting positions between (i) those evaluation consumers that prefer having all the possible measures as field data for enabling deep result analysis and promote data sharing among community members [7] (e.g., people from academia), and (ii) those adopting a more pragmatical viewpoint that ask for an small set of meaningful and representative scores to characterize, rank and compare evaluated systems [8] (e.g., people from industry). To cope with this goal we rely on the notion of quality model, adopted from ISO/IEC 25000 standards [9], to formulate not only rigorous but also usable and flexible interpretation rules.

Before closing this introduction, it is important to say that the integration of a multi-criteria analysis methodology in very simple benchmarks may be useless, specially where few, or only one, measure or measure type is under consideration. The use of the methodology proposed in this paper makes sense in benchmarking contexts where the analysis process asks for the simultaneous consideration (aggregation and/or comparison) of different measures of different type. The higher the number of measures or the heterogeneity of such measures the higher the usefulness of the proposal. Since this is what happens in dependability benchmarks, the present proposal limits its purpose to this type of benchmarks, and this despite its obvious potential for any other type of benchmarks.

The rest of the paper is structured as follows. Section 2 introduces a brief background about dependability benchmarking and multi-criteria analysis. Section 3 presents our multi-criteria analysis methodology. Section 4 shows the feasibility of our approach through three different case studies and finally. Section 5 concludes the paper.

2. Background

Computer benchmarks are standard tools that enable the evaluation and comparison of different systems, components, and tools according to specific characteristics [10]. Benchmarks have been widely used to compare the performance of systems (e.g. transactional systems [1] or embedded systems [11]. From a high-level viewpoint, the specification of a conventional benchmark encompasses with the definition of the following components:

- The *system under benchmarking* and the *benchmark target*, which specify the context of use of the system under evaluation and the model of the considered target;
- The *measures* that will be employed to characterize and compare existing alternatives;
- The *execution profile* required to parameterize and exercise both the system under benchmarking and the benchmark target during experimentation. This is typically a *workload*;
- The *experimental procedure* specifying how to run the selected execution profile and how to trace the resulting activity;
- The process to follow in order to transform traces (experimental measurements) into expected benchmark measures.

The main benefit of conventional benchmarks is that, once the set of proposed measures are widely accepted by a community, systems produced by such community can be compared in a quite straightforward and unambiguous way. The key issue here is the that most of the considered measures are homogeneous. In deed, they simply characterize evaluated systems in terms of either their performance or their cost. As a result, comparisons among systems are carried out in a more *representative* way, since based on the use of a set of measures widely accepted by a given community.

Things become however quite different when conventional benchmarks evolved to dependability benchmarks. The seminal work on dependability benchmarking dates from 15 years ago and was produced in the context of the European project *DBench* [2]. Dependability benchmarks characterize the ability of evaluated systems to cope with their purpose not only in the absence of faults and attacks, as conventional benchmarks do, but also in their presence. The feasibility of the approach and its applicability to different application domains and systems have been shown in [12]. Roughly speaking, dependability benchmarks are specified as conventional benchmarks, but revisiting the concepts of performance profile and experimental procedure as follows:

1. The notion of execution profile is enriched with the specification of a set of accidental faults and attacks, those to which the system must be exposed during experimentation. This set is called the *perturbation-load*.
2. The experimental procedure is reformulated in order to specify not only how considered systems or components must be exercised using the workload, but also how to apply the specified perturbation-load.

Recently, the concept of dependability benchmarking has been also applied in the context of autonomous system, resulting in a new type of benchmark called *resilience benchmark*. In the context of these new benchmarks, benchmarks targets are evaluated, not only in the absence and presence of perturbation-loads, but also in the presence of changes affecting the behavior and/or structure of such targets.

Contrary to conventional benchmarks, the number and heterogeneity of the considered measures is a constant in the various existing dependability benchmarking proposals [12]. Indeed, researchers have proposed, from the very beginning, the use of on-line analytical processing and data warehousing approaches for the analysis and sharing of results from dependability benchmarking experiments [13]. Some other have proposed also the definition of a common repository for sharing the experimental data produced by dependability benchmarks, like the one conducted by the European project AMBER [14]. However, the problem of combining measures in a meaningful and repeatable way was not address by any of these initiatives, although it is of major importance when considering a large number of heterogeneous measures, as in the case of dependability benchmarks.

2.1. Comparison of alternatives through aggregation

Measures aggregation is a common approach trying to enable meaningful comparisons among systems that eases the analysis of benchmarked systems or components. However, although these techniques are usually applied in the community of dependability benchmarking, it is surprising that so far there is still a lack of unified criteria when addressing the aggregation of measures and their subsequent analysis. Common methods applied by users for aggregation range from simple mathematical operations (e.g., addition or mean average) to more serious and systematic distribution fitting [15] and custom formulae [16] approaches.

Kiviat or radar diagrams [17] are graphical tools which represent the results of the benchmark in an easy-to-interpret footprint. Kiviat diagrams can show different measures using only one diagram and, although some training is required, the comparison of different diagrams is fairly simple. The scalability of Kiviat diagrams enables the representation of up to tens of measures. However, managing such a huge amount of information may make difficult the interpretation and analysis of results. The problem previously stated is solved in [17] throughout the use of an analytical technique named the *figure of merit* which, imposing certain restrictions to the graph axes, synthesizes all the measures into a unique numerical value associated to the footprint shape. However, the problem of this solution, as it happens with most techniques using the mean or the median, is that valuable information could be hidden behind a unique number, and consequently, the comparison between systems could result quite vague [18].

Other approaches, like the presented in [15], characterize the level of goodness of the measures according to their ability to fit with a particular statistical distribution. Nevertheless, this approach presents three main drawbacks. First, it assumes that a measure follows the same distribution for all the systems, which may be false depending on the context of use. Second, to understand this type of characterization, it is necessary to understand the assumed statistical model, which is not straightforward. Third, the subjectivity of the probability distributions will strongly affect the sensitivity analysis. Finally, it is necessary to handle those situations when there is not enough information to build probability distributions for evaluation data.

Finally, Correia et al. [19] apply the notion of thresholds to map measures into a particular scale for software systems certification. Yet, they assume all the measures have the same importance when it is not always the case.

In sum, previous methodologies lack the ability of aggregating measures into a meaningful way. Generally, these techniques focus on aggregation of results and do not provide any insights on how to cope with the interpretation of the resulting aggregated scores. Accordingly, open questions requiring further research in the domain of dependability benchmarking are (i) how to systematically aggregate such measures to capture in a single or small set of scores the information required to characterize the overall system quality, and (ii) how to ensure the consistency of interpretations issued from the use of such scores with respect to the conclusions obtained from the direct analysis of benchmark measures. Next section is focused on describing how these open questions are coped in this work.

2.2. *A potential step forward using multi-criteria analysis*

The problem of comparing a set of targets according to an heterogeneous set of measures has many similarities with the *multi-criteria decision* problems typically considered in the *operational research* field. So, the use of *multi-criteria decision making* (MCDM) methods to support the analysis of dependability benchmarking measures seems quite promising.

There exist multiple MCDM methods that can be used to address this problem, some of them are widely used in many application domains like business industry, social science, engineering, etc. Among the large number of MCDM methods, some have gained more popularity than others, the *Analytic Hierarchy Process* (AHP) [20] for example, and its use can be found in many works ([21] and [22], for example). Our previous work ([23], [24] and [25]) already presented the feasibility of using MCDM methods to perform the analysis of measures in dependability benchmarking.

The methodology presented in this work will adapt the concepts that apply to MCDM methods with the aim of not only providing mechanisms to better compare different alternatives from benchmarking results, but also to cover the lacks in the analysis that can make dependability benchmarks in particular improve the confidence people from the industry have on them. To that end, next section deeply describes the methodology developed in this work, and its integration in the benchmarking process.

3. A multi-criteria analysis methodology to interpret evaluation results

The proposed multi-criteria analysis methodology does not intend to automate the task of benchmarking performers when selecting a proper system; it rather tries to support and guide the comparison of the systems or components fulfilling the system requirements for a particular application, and the selection of the most suitable one.

What makes it interesting for dependability benchmarking with respect to the rest of approaches presented in Section 2.1, is its capability to systematize the way to compute the global score of a component not only considering the measures themselves, but also formalizing their interpretation attending to aspects such as the relationship among the measures, and their relative importance within a particular context of use. Accordingly, it is easy to obtain a hierarchical quality model, inspired in the software quality model proposed by the ISO/IEC 25000 (SQuaRE) standard [9], which assists the navigation from the fine-grained measures to the coarse-grained scores without losing the numerical perspective of results. In such a way, one can keep the consistency in the interpretation and analysis of results independently from the viewpoint (fine or coarse) acquired by the benchmark user.

Figure 1 illustrates how the quality model (QM) should be integrated into the dependability benchmarking process, and when it should be applied to provide conclusions from the resultant measures. The definition of the benchmark characteristics in the *experimental set up* lets the evaluator determine the quality model that will later be used to analyze the final measures. The early definition of the analysis process, even before benchmarks are performed, reduces the subjectivity that can be introduced in the analysis process when partial results are being obtained or conclusions are anticipated, which may bias this analysis. This will also ease the cross-comparison among works done from third-party evaluators, as results will be comparable under exactly the same procedure, which may also contribute to the acceptance of dependability benchmarks by the industry.

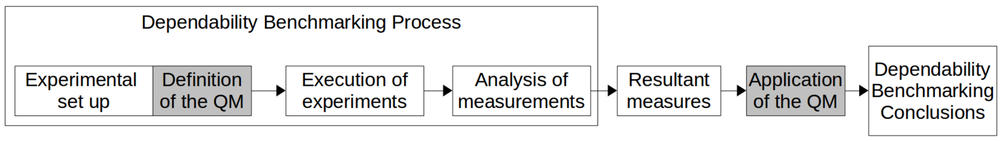

Figure 1: Integration of the quality model in the dependability benchmarking process

Defining the quality model according to the requirements of the evaluator (or evaluators) demands the definition of a set of features for the analysis. Upcoming subsections describe these features in detail, identifying their role in the methodology and mapping them to their respective characteristic in the evaluators requirements. The application of the quality model in the analysis process will be later illustrated in Sectionc̃aseslabel through different case studies.

3.1. Benchmark user and target system

The first step is to identify the benchmark targets (in case of more than one alternative), the application context where they operate in and their goal, that obviously depend on the evaluation performer. These aspects are crucial to (i) determine the requirements of the system; and (ii) fix their level of accomplishment.

System requirements can be expressed through the notion of quality model, previously introduced in standards such as [9]. A quality model is a framework to ensure that all the information required by the stakeholder to perform the proper decision-making is taken into account to carry out the analysis of benchmark measures. With respect to this point, the rest of this methodology will introduce the instruments (thresholds, relationships, weights) required to enrich the meaning of measures within the benchmarking process.

3.2. Criteria under evaluation

During the *experimental set up*, benchmark performers determine a set of measurable attributes (noted m_1 to m_n) that are representative of the system quality or simply of interest for the evaluation performer. These measures constitute the output of the benchmark, and they are used to compare different benchmark targets and perform the election of the most suitable choice.

In the proposed methodology, the measures defined by the benchmark performer in the first step of the benchmarking process conform the base level criteria of the quality model. These criteria must be understood as the inputs for the quality model that will be used in the analysis process to determine the relative quality of the system according to the defined model. Obviously, the quality and precision of the measures selected in the *experimental set up*, which correspond to the criteria defined in the quality model, will have a high influence on the quality of the conclusions extracted from applying that model in the analysis process. Different works have focused on the selection of attributes in benchmarks to provide good quality measures. Authors in [26] dealt with this problem from a metrology point of view, pointing out the attributes that selected measures must fulfill, so good quality conclusions can be extracted from them. When benchmark performers lack of criteria to determine which measures should be selected, it would be convenient that measures were non-redundant, independent and thoroughly selected attending to their capability to represent quantitative elemental aspects of the system, such as delay, throughput or data availability in a network. This involves that no measure should be derived from other. According to this remark, if we are already taking into account the system's throughput in presence of faults as a measure, considering any other throughput-based measure, such as a ratio between the throughput in absence and presence of faults, would be unfairly providing more importance to throughput than the rest of measures. Despite its importance, and as it has already been considered in other works, the selection of measures is out of the scope of the proposed methodology, that aims at providing mechanisms to improve the comparison of benchmark targets based on the (high quality) resultant measures.

3.3. Scales of measures

Given the heterogeneity of the measures considered in dependability benchmarking, it is easy to find different measures using distinct scales and dimensions, e.g., seconds or milliseconds if measuring time, joules if measuring energy, and so on. Obviously, this hinders the analysis and comparison of measures for non-skilled users.

To compare various alternatives, the measures should be brought to the same scale, and normalization methods can be applied to do so. Although normalization methods scale the values in different ways, they share some common properties. Normalizing by the sum of all the values keeps the proportion between values in the normalized ones. This means, that if a result r_i is the double of r_k, the normalized result v_i will still be the double of v_k. When normalizing by an extreme value (either Max or Min), proportion is also kept, but in both methods, normalized values tend to be grouped together. The use of thresholds, on the other hand, does not tend to group the normalized values but they are distributed along the given range according to their original value.

With the aim of coping with this normalization problem, this methodology propose the use of thresholds within the definition of quality criterion functions $c_i(m_i)$ which specify how to quantitatively evaluate each measure, i.e., they establish an equivalence between the measured value and the system quality requirements within a 0-to-100 quality scale. The result of each criterion function, known as elementary score (or elementary preference), corresponds to s_i. Formally, such elementary preferences s_i can be interpreted as the degree of satisfaction of a measure m_i with respect to the quality requirements specified by the benchmark performer for such measure in the form of a minimum and a maximum threshold (T_{min_i} and T_{max_i} respectively). Since all the measures are scored according to the same normalized scale, resulting elementary preferences are directly comparable. Such equivalence can be mapped to discrete or continuous functions. Equations (1) and (2) show an example of lineal increasing and decreasing functions when measures are the higher the better and the lower the better, respectively. However, these criterion functions can be adapted to satisfy the evaluator's requirements for the normalization of the measures. Examples of how these functions can be adjusted are shown in Section 4 through the case studies presented.

$$s_i = c_i(m_i) = \begin{cases} 0, & m_i \leq T_{min_i} \\ 100\frac{m_i - T_{min_i}}{T_{max_i} - T_{min_i}}, & T_{min_i} < m_i < T_{max_i} \\ 100, & m_i \geq T_{max_i} \end{cases} \tag{1}$$

$$s_i = c_i(m_i) = \begin{cases} 100, & m_i \leq T_{min_i} \\ 100\frac{T_{max_i} - m_i}{T_{max_i} - T_{min_i}}, & T_{min_i} < m_i < T_{max_i} \\ 0, & m_i \geq T_{max_i} \end{cases} \tag{2}$$

The use of minimum and maximum thresholds within criterion functions is necessary to position and compare the value of measures with respect to reference values of the applicative domain, thus easing their interpretation. For example, the interpretation of the measured throughput in a communication system (let us assume 8 Kbps) will be better if the measure is obtained from a Wireless Sensor Network in charge of monitoring temperature (where the optimum value may round 10 Kbps) rather than if it is obtained from a Wireless Mesh Network to provide Internet access (where even the minimum value allowed for a quality communication, let us assume 500 Kbps, is greater than the value obtained). For each applicative domain, thresholds can be obtained through previous experimentation, the opinion of experts in the domain, or certification and widely-used references. Evaluators or experts in the field should agree on their definition for each measure in a given applicative domain. In this way, comparing the results obtained for different systems is easier, as normalized results are distributed along the range defined by thresholds, instead of being grouped together as happens with other normalization methods. Indeed, the definition of thresholds gives meaning to the values obtained for each measure. Consequently providing the minimum and maximum values that can receive each measure will be very important to determine their preference.

Once measures have been scored, evaluation performers have a founded intuition about the system behavior. In fact, they are able to determine if the individual goal for each particular measure has been accomplished or not. For example, obtaining a score of 75% in one measure could be interpreted as a positive feedback. However, their global preferences about the system requirements are not mapped yet in the result of the evaluation. The idea of the following stage is to aggregate the characteristics of the system according to the evaluation performer's requirements and preferences.

3.4. Preferences aggregation

To address the aggregation of scores, this stage of our methodology structures a quality model through a hierarchy of high-level objectives, sub-objectives, etc., where previously computed scores are located at the leaves of the hierarchy. The construction of such hierarchy is relative. First, it is necessary to classify each single score regarding the system characteristic it better fits in. For example, let us assume a transactional system where four measures such as *throughput*, *delay*, *availability* and *reliability* have been considered. In this case, the first level of aggregation could group *throughput* and *delay* within the characteristic of *performance*, and *availability* and *reliability* within the characteristic of *dependability*. This classification of measures can continue grouping similar sub-characteristics into characteristics. Thus, a second level of aggregation would group both *performance* and *dependability* to determine the global quality of the system.

Despite modeling the hierarchical structure of the system, not all the system requirements may have the same importance depending on factors such as the benchmark performer's preferences and the application domain. To cope with this problem, the proposed methodology enables the refinement of the quality model using *weights* to determine the relative importance among requirements for the analysis.

The benchmark performer's requirements that define the quality model should be able to reflect the purpose of the benchmarked system in a given application domain. In some application domains, some measures might be considered of greater importance than others when benchmarking the same system, and thus the quantification of that importance should be implicit in the performer's requirements. Then, the importance that each particular measure has for the analysis is quantified with a weight w_i, where w_i is the weight of the i^{th} particular measure (criterion or resulting sub-characteristic) in a hierarchical level. This measures are weighted according to their relative importance or influence to their direct upper level measure, in such a way that for k measures in a level, $\sum_{i=1}^{k} w_i = 1$. Weights enable to tune the way in which system characteristics contribute to the global quality of the system. Then, consensus between benchmark performers on how measures must be weighted for a given application domain is necessary to contribute to the acceptance of dependability benchmarks in the industry. As an example of weighting in different application domains, let us take into consideration a distributed system within a non-critical solution such as comfort electronic control in cars, probably a rapid response in terms of performance aspects will have more weight than dependability ones (e.g., weighting them 75% and 25% respectively). Conversely, if for example we refer to the Antiblock Brake System (ABS) of the vehicle, evaluation performers may weight dependability above performance assigning weights of 75% and 25% respectively. Fig. 2 illustrates this last example. The number above the tree branches indicates the weight assigned in each case.

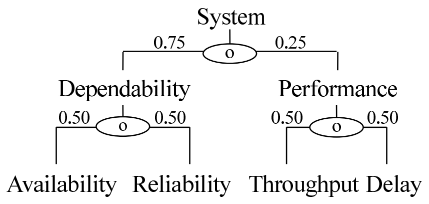

Figure 2: Example of weights assignment.

Once weights are assigned, it is essential to determine the relation between the elements of the model. For this, different types of operators o may be used to define the conditions under which characteristics are aggregated in Fig. 2. The power or generalized mean [27], defined in (3), is a generic expression to compute an infinity of aggregation types, considering the notions of scores and weights previously stated. When exponent $r = 1$, this expression is equivalent to traditional arithmetic mean, widely used for aggregation. However, strikingly, the use of different aggregation operators has been rarely considered despite their power to represent, for instance, a punishment in the aggregation result when requirements are not being accomplished or a reward for those requirements that satisfy evaluation criteria. Thanks to (3), it is possible to define as many aggregation types as values may take exponent r. Indeed, authors such as Dujmovic propose up to 20 different ones [28]. However, the selection of the proper aggregation operator is a task whose complexity increases as far as more alternatives are considered. Thus, our goal is to define a reduced set of equivalence classes that intuitively represent the different possible levels of aggregation through distinct values of r.

$$S = (\sum_{i=1}^{k} w_i s_i^r)^{\frac{1}{r}} \tag{3}$$

To address this challenge, first, it is necessary to introduce the notion of *andness* [29], and how it relates to exponent r. The *andness* of an aggregation operator o, defined in (4), is a 1-to-0 coefficient where $andness = 1$ represents that all the system requirements must be satisfied at the same time, and $andness = 0$ involves that just accomplishing any system requirement (regardless which one) is enough.

$$andness(o) = \frac{max(x) - o(x)}{max(x) - min(x)} \tag{4}$$

According to [28], $andness = 1$ is associated to $r = -\infty$ whereas $andness = 0$ equates to $r = \infty$. Mathematically, it is quite easy to prove how min is the operator $o(x)$ that makes $andness = 1$, and max is that making $andness = 0$. For the sake of homogeneity, let us denote min with $S+$ to intuitively illustrate the idea that all the system requirements keep a relationship of *strong simultaneity*. Following the analogous reasoning, let max be represented with $R+$ to show the notion that any accomplished system requirement *strongly replaces* the rest (despite they are not satisfied). In the middle, $andness = 0.5$ matches to *arithmetic mean*, which, as previously introduced, is represented with $r = 1$. Let us denote this operator with N to associate its use with the meaning of *neutrality*. Between $andness = 1$ and $andness = 0.5$ there is a gradation of aggregation operators that can be explained as filters that progressively boost the influence of simultaneity against replaceability in system requirements, as far as $andness$ tends to 1. Mathematically, this implies minimising the influence of higher scores while maximizing that of lower ones in the aggregation result. For the sake of simplicity, we have selected $andness = 0.75$ as a representative value of this range. Let us denote this operator of *weak simultaneity* as S. Conversely, the range of operators among $andness = 0.5$ and $andness = 0$ boosts the influence of replaceability with respect to simultaneity as far as $andness$ tends to 0. Similarly, this implies minimizing the influence of lower scores while maximizing that of higher ones. We have selected the aggregation operator with $andness = 0.25$ to represent this equivalence class. Let us denote the *weak replaceability* of this aggregation operator with R. The different values exponent r takes depending on the number of inputs of the aggregation can be found in Table 1. For instance, considering the aggregation of 5 different scores with normalized values of 90, 70, 70, 50 and 20, with evenly distributed weights, the final score obtained for operators $R+$, R, N, S, and $S+$ are 90 (max), 72, 60 (arithmetic mean), 48, and 20 (min), respectively.

Table 1: Value of exponent r for the operators considered.

Aggregation operators	**2 inputs**	**3 inputs**	**4 inputs**	**5 inputs**
S+ (strong simultaneity)	$+\infty$	$+\infty$	$+\infty$	$+\infty$
S (weak simultaneity)	3.93	4.45	4.83	5.11
N (neutrality)	1	1	1	1
R (weak replaceability)	-0.72	-0.73	-0.72	-0.71
R+ (strong replaceability)	$-\infty$	$-\infty$	$-\infty$	$-\infty$

Previous simple aggregations between scores can be nested to denote those requirements having a special meaning or priority, i.e., a certain degree of mandatoriness or sufficiency for a particular system requirement within the same hierarchical level. For example, Fig. 3a illustrates a case where *characteristic A* feedbacks its own simultaneity aggregation (e.g., S), which basically means that satisfying that characteristic is a mandatory condition for the system. Logically, this can be seen as $A \wedge (A \vee B)$, with different degrees of *andness* depending on the selected operators. Thus, not satisfying the requirements of that characteristic would severely penalize the system. Conversely, applying a replaceability operator (e.g., R), would involve defining that characteristic as a sufficient requirement. Likewise, this could be logically expressed as $A \vee (A \wedge B)$, with the selected degrees of *andness*. Fig. 3b depicts exactly the same model as Fig. 3a but using a simplified notation to ease the use of mandatory and sufficient requirements. Thick branch represents priority requirements in such a way they become mandatory if using S or $S+$ operators, and sufficient if using R and $R+$. To complete this simplification, neutrality operator N and equitable weighs are assumed for the branches omitted. In the rest of the paper the simplified notation will be used.

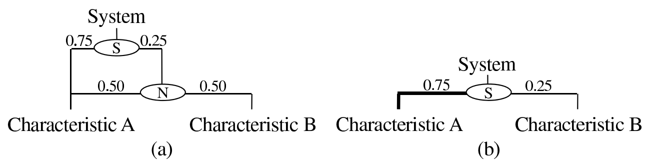

Figure 3: Model representing the priority of Characteristic A versus Characteristic B: (3a) full model showing how Characteristic A feedbacks its own simultaneity operator (Characteristic A is mandatory), and (3b) compact version of that model representing exactly the same hierarchy.

3.5. Sensitivity of the quality model

The sensitivity of the quality model is determined by how the sources of uncertainty present in the inputs of the model are translated into uncertainty in the conclusions provided from the application of this quality model.

The aforementioned inputs of the quality model might suffer from a certain degree of uncertainty. For example, errors in the process of measurements (inaccurate measures), a poor understanding of the relevance that each criterion has for the application domain (leading to erroneous weights), or a lack of comprehension of the common behavior of the targeted systems (wrong definition of thresholds). This uncertainty present in the inputs of the model will certainly impact the confidence that benchmark users can place in the conclusions provided as output of the quality model.

Accordingly, the quality model must be analyzed to determine the sensitivity that its output has to the uncertainty in its inputs. This sensitivity analysis can be performed through different methodologies, like those that can be found in [30].

As it was explained in earlier sections, works like [26] have studied the uncertainty from the measurements point of view, setting guidelines to obtain good quality measurements in the system to generate measures with a low uncertainty. Even though studying the uncertainty of the base measures is of prime importance, analyzing the sensitivity of the whole quality model requires a great effort. An extensive analysis on how the combined uncertainty of the inputs of the quality model affect the output conclusions has already been studied in [31] and [32] from the perspective of *multi-criteria decision making* methods.

As the main goal of the paper focused on the definition of a methodology to deal with the analysis of results and comparison of targets in benchmarking, no sensitivity analysis will be done in this work. Nevertheless, this analysis could be very important towards the acceptance of proposed quality models by the industry in different application domains.

Next section presents a set of three different scenarios in the domain of dependability benchmarking that will be used as case studies to illustrate the application of the proposed methodology.

4. Case studies

This section shows the feasibility of our multi-criteria analysis methodology along three case studies in the domain of distributed systems, such as web servers, on-line transactional databases and wireless ad hoc networks. As it is possible to apply our methodology at any stage of the analysis (even if measures are already selected, or normalized into scores), as well as to increase the confidence of our study, we apply our methodology from the results delivered by accepted papers in the community. Thus, the information extracted from the papers will be used to elaborate adequate quality models matching author's requirements. The goal is to objectively model the system characteristics to compare the results we are able to obtain through our methodology with those originally delivered by authors. The case studies have been selected in such a way they show the power of our methodology when benchmarking users need to (i) exploit the meaning of measures to properly analyze the system; (ii) rank systems attending to different potentially countered criteria; and (iii) determine the influence that a particular characteristic of the system may have in its behavior. In this way it will be shown the usefulness of the methodology to carry out the analysis of systems following a structured, simple and repeatable way under well-defined evaluation criteria.

4.1. Intermediate and global scores to benchmark web servers

In [33], authors perform the comparison of two well-known web servers (Apache and Abyss), running on top of three different operating systems (Windows XP, Windows 2000 and Windows 2003) through the SPECWeb99 benchmark [34]. Thus, authors aim at selecting the best combination of the pair {web server, operation system}. Despite target systems are subjected to 12 different faults encompassing both software and hardware faults, authors finally present only two types of results: those regarding the execution of the system in absence of faults (baseline) and execution in presence of faults.

4.1.1. Criteria under evaluation

The results of the benchmark are analyzed using 6 measures (3 from performance and 3 from dependability). The set of performance measures is composed of the number of simultaneous connections (con) correctly established (SPECf); the number of operations (op) per second (THRf); and the average time in milliseconds (ms) that the operations requested by the client take to complete (RTMf). With respect to dependability, authors consider autonomy, as a percentage of administrative interventions with respect to the number of faults injected (AUT); accuracy, as a percentage of requests with error with respect to the total amount of requests (ACR); and the percentage of time the system is available to execute the workload from the total (AVL). Table 2 collects the results for these measures.

Table 2: Measures characterizing the behavior of the pair {web server, operating system} in presence of faults [33].

System	**AUT** *(%)*	**AVL** *(%)*	**SPECf** *(# con)*	**THRf** *(# op/s)*	**RTMf** *(ms)*	**ACR** *(%)*
Apache-2000	93.98	95.28	13.82	79.24	382.2	97.21
Apache-XP	95.48	97.94	18.07	71.63	359.7	97.60
Apache-2003	96.77	97.62	11.27	79.21	373.1	97.29
Abyss-2000	94.36	96.35	10.32	75.96	363.7	94.78
Abyss-XP	95.97	97.31	13.71	68.22	362.0	94.50
Abyss-2003	96.25	97.53	12.91	66.18	358.7	95.55

4.1.2. Scales of measures

As previously mentioned in Section 3.3, thresholds can be determined in different ways. In this case, given the need of authors for ranking systems in the presence of faults, and the lack of field references to determine proper thresholds, an adequate way to get them is using the maximum and minimum values of each measure obtained during the experimentation in presence of perturbations. This enables a relative comparison between targeted systems in such a way that the maximum value will obtain a score of 100 and the minimum a score of 0. This assignation of scores is suitable when authors are not so interested in the sensibility or meaning of the quantitative measure, since baseline results are not considered, but just in establishing a clear ranking of systems in presence of faults. Thus, we have defined two linear criterion functions $c_i(m_i)$, one increasing for the-higher-the-better measures such as SPECf, THRf, AUT, ACR and AVL; and another decreasing, for RTMf, which is the-lower-the-better, similar to those shown in (1) and (2) respectively. Maximum and minimum thresholds are shown in Table 3.

Table 3: Minimum and maximum thresholds for the measures of web servers.

		Thresholds	
Measure	**Function**	**Min**	**Max**
AUT	Increasing	93.98	96.77
AVL	Increasing	95.28	97.94
SPECf	Increasing	10.32	18.07
THRf	Increasing	66.18	79.21
RTMf	Decreasing	362.0	382.2
ACR	Increasing	94.5	97.60

4.1.3. *Preferences aggregation*

According to authors [33]: *"In this case study we assumed a general-purpose web-server scenario and assigned equal relevance to all six benchmark measures"*. To satisfy such considerations, the quality model has been established following a trade-off solution. In particular, measures have been equally weighted within their category, and neutral operator (N) has been used for the aggregation. The representation of the complete quality model is depicted in Fig. 4.

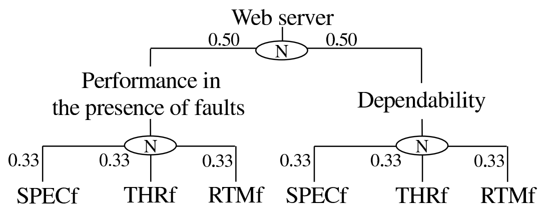

Figure 4: Quality model defined for web servers.

4.1.4. *Analysis of results*

It is worth noting that the results obtained when computing the quality model, shown in Table 4, match those obtained by the authors in the paper. When comparing the operating systems for each web-server, *"Windows XP seems to provide the best platform for Apache and Windows 2003 the best for Abyss"*. The comparison of the 6 systems brings up the same conclusions as those given by the authors: *"the combination Apache/XP seems to be the one where the service degradation caused by faults is less noticeable"*. A global score of 81 points quantifies this fact.

Table 4: 0-to-100 normalized results (scores) after applying the quality model shown in Fig. 4.

System		**AUT**	**AVL**	**SPECf**	**THRf**	**RTMf**	**ACR**	**Perfor-mance**	**Depen-dability**	**Global score**
Apache	2000	0	0	45	**100**	0	87	48	29	38
	XP	54	**100**	**100**	42	96	**100**	**78**	84	**81**
	2003	**100**	88	12	**100**	39	90	50	**92**	71
Abyss	2000	14	40	0	75	79	9	51	21	36
	XP	71	76	44	16	86	0	48	49	48
	2003	81	85	33	0	**100**	34	44	66	55

Apart from that, it is remarkable that scores at leaves are consistent with those delivered at intermediate ones (performance and dependability scores) and the root (global score). As seen, it is possible to navigate from fine-grained to coarse-grained scores through intermediate ones. Indeed, it is possible to discover sensitive information that is not provided in the original paper. Attending to intermediate criteria, it is possible to observe that the pairs {Apache, XP}, with 78 points, and {Apache, 2003}, with 92 points, are the best candidates from a performance and dependability viewpoint respectively. As observed, the use of quality models can be useful to improve the exploitability of measures in the analysis of results.

4.2. Managing multiple criteria for comparing OLTP systems

In [5] the authors propose a dependability benchmark for On-Line Transaction Processing (OLTP) systems. Thus, ten targets (A to J) are defined based on the combinations of (i) two different versions (DB 1, DB 2) of a leading commercial Data Base

Management System (DBMS), (ii) two DBMS configurations (Conf A, Conf B), (iii) three operating systems (Windows 2000, Windows XP, SuSE Linux 7.3) and (iv) two different hardware platforms (HW 1, HW 2).

4.2.1. Criteria under evaluation

From the benchmarking process, the authors obtain measures based on three different criteria: baseline performance, performance in presence of faults, and dependability. Such measures, typically used in the TPC-C [1] benchmark, are the *number of transactions (trans) per minute (m)* and *price ($) per transaction*. When these measures are obtained in absence of faults (baseline performance), they are labeled as tpmC and $/tpmC respectively, but when obtained in presence of faults (performance in presence of faults) they are labeled as Tf and $/Tf. Dependability measures make reference to the percentage of time the server is available (AvtS), and the percentage of time the client is available (AvtC). Table 5 shows the original values of the measures provided in the paper.

Table 5: Original measures extracted from [5] characterizing the 4-tuple {operating system, DBMS, configuration, hardware platform}.

	tpmC	**$/tpmC**	**Tf**	**$/Tf**	**AvtS**	**AvtC**
System	*(#trans/m)*	*($/#trans)*	*(#trans/m)*	*($/#trans)*	*(%)*	*(%)*
A: {Win 2000, DB 1, Conf A, HW 1}	2244	12	1525	17.7	86.1	75.4
B: {Win 2000, DB 2, Conf A, HW 1}	2493	11.6	1818	16	87.2	79.5
C: {Win XP, DB 1, Conf A, HW 1}	2270	11.9	1667	16.2	88	79.4
D: {Win XP, DB 2, Conf A, HW 1}	2502	11.6	1764	16.4	88.6	79.5
E: {Win 2000, DB 1, Conf B, HW 1}	1411	19.1	896	30.1	74.2	68.7
F: {Win 2000, DB 2, Conf B, HW 1}	1529	19	969	29.9	76.6	69.7
G: {SuSE 7.3, DB 1, Conf A, HW 1}	1961	12.7	1406	17.8	86.3	77
H: {SuSE 7.3, DB 2, Conf A, HW 1}	1958	13.8	1400	19.3	93.5	83.9
I: {Win 2000, DB 1, Conf A, HW 2}	3655	7.7	2784	10.1	89.4	79.5
J: {Win 2000, DB 2, Conf A, HW 2}	4394	6.8	3043	9.9	88	80.9

4.2.2. Scales of measures

Given the absence of clear or explicit arguments of authors to carry out the comparison of systems in this case study, let us perform the selection of thresholds positioning the results of their evaluation with respect to referenced values obtained in the community [35] in the last years. This choice pursues a double goal. First, not only to compare target systems among one another in a local way, but also to provide a useful feedback about their behavior when adopting a wider perspective and comparing them with other systems using TPC-C benchmarks, even when they are not subjected to faults. Second, showing the capability of our methodology to incorporate multiple ways to select scales of measurement. Hence, for the definition of thresholds, we have taken into account the results delivered in [35] for the year 2000, when the hardware platforms considered in this case study appeared. Table 6 shows the upper (maximum threshold) and lower (minimum threshold) values of the trend for TPC-C in the intersection with that year. It must be noted that tpmC and Tf, on the one hand, and \$/tpmC and \$/Tf, on the other, represent the same measures but in absence and presence of faults, respectively. This is why the same thresholds are defined for both measures.

Table 6: Thresholds determined for the different measures of OLTP systems.

		Thresholds	
Measure	**Function**	**Min**	**Max**
tpmC	Increasing	1400	4800
\$/tpmC	Decreasing	1	20
Tf	Increasing	1400	4800
\$/Tf	Decreasing	1	20
AvtS	Increasing	74	100
AVtC	Increasing	70	100

4.2.3. Preferences aggregation

The authors classify the ten systems attending, each time, to a different criterion (baseline performance, performance in presence of faults and dependability). Despite this situation may require the generation of three different quality models, one per criterion considered, it is also possible to generate just one quality model that can be parameterized in such a way that the different cases are represented at the same time. Let us take into account this last alternative to show the expressiveness power of our approach.

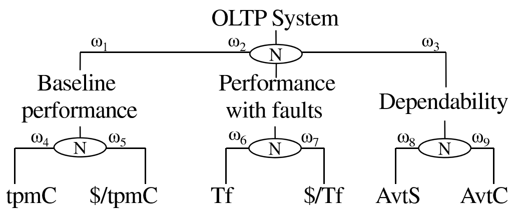

Figure 5: Parameterized quality model gathering all the single criterion stated by authors and the proposed trade-off between all measures.

Table 7: Weights for the parameterized quality model shown in Fig. 5.

Characteristics	w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8	w_9
Baseline performance	1	0	0	1	0	0	0	0	0
Performance with faults	0	1	0	0	0	1	0	0	0
Dependability	0	0	1	0	0	0	0	0.70	0.30
Trade-off	0.33	0.33	0.33	0.5	0.5	0.5	0.5	0.70	0.30

Each branch of the quality model defined in Fig. 5 has been assigned a given weight, whose value can be modified as shown in Table 7 to model the three different criteria defined by authors. Weights for tpmC, Tf, $/tpmC and $/Tf scores have been properly parameterized, as the last two are not considered by authors in the definition of the classifications. Likewise, being the availability of the server more critical than the exhibited by clients, as explicitly commented by authors, weights have been accordingly adapted. Finally, the authors also propose the generation of a trade-off ranking to reach a consensus between the three criteria previously tackled. Unfortunately, despite they let the reader know that it is based on the previous rankings, they do not structure a clear reasoning on how this classification is achieved. Given the role of our methodology to cover potential ambiguities and lacks of thoroughness, it would be possible to define alternative weights to adequately address the trade-off ranking concerned.

4.2.4. Analysis of results

Table 8 shows the intermediate and global scores for each system after computing the trade-off quality model previously proposed. Table 9 collects the different rankings to ease the comparison between systems. From the intermediate scores that belong to the different criteria, it can be appreciated that the single criterion rankings match those defined by the authors. Nevertheless, the ranking established according to the trade-off criterion presents a similar, but not equal order. While in the paper the trade-off ranking is "*I, J, D, B, C, H, G, A, F and E*", with the methodology proposed systems "*I, J*" and "*G, A*" swap their positions. The problem, in consequence, is not so the analysis done by the authors, probably correct, but the difficulty to exactly reproduce it again with the tools they provide. This result shows the need for establishing clear and explicit rules when addressing the analysis of benchmarked systems. As observed, the use of quality models can be useful not only to easily rank different systems despite applying different criteria, but also to unequivocally repeat this ranking when needed.

Table 8: 0-to-100 normalized results (scores) after applying the trade-off weights from Table 7 to the quality model shown in Fig. 5.

System	Baseline performance	Performance with faults	Dependability	Trade-off
A	33	7	37	26
B	38	16	45	33
C	34	13	47	32
D	39	14	49	34
E	2	0	1	1
F	4	0	7	3
G	27	5	39	24
H	24	1	67	31
I	65	46	51	53
J	78	50	49	58

Table 9: Original rankings carried out in [5] against those obtained from applying quality models.

		Ranking of systems									
		1st	2nd	3rd	4th	5th	6th	7th	8th	9th	10th
Baseline performance	Original	J	I	D	B	C	A	G	H	F	E
	Quality model	J	I	D	B	C	A	G	H	F	E
Performance in presence of faults	Original	J	I	B	D	C	A	G	H	F	E
	Quality model	J	I	B	D	C	A	G	H	F	E
Dependability	Original	H	I	D	J	C	B	G	A	F	E
	Quality model	H	I	D	J	C	B	G	A	F	E
Trade-off	Original	**I**	**J**	D	B	C	H	**G**	**A**	F	E
	Quality model	**J**	**I**	D	B	C	H	**A**	**G**	F	E

4.3. Evaluating perturbations on ad hoc networks

This case study aims to show the feasibility of this methodology to determine the impact that each single perturbation has over a system when considering its injection separately from the rest of perturbations compounding the faultload. In [36], the authors perform the evaluation of two different and representative types of ad hoc networks, a static Wireless Sensor Network (WSN) where 6 real nodes execute AODV routing protocol (Network A) and a Mobile Ad Hoc Network (MANET) where 6 real mobile nodes run OLSR routing protocol (Network B), when subjected to perturbations. Such set of perturbations is formed by accidental faults like *signal attenuation* and *ambient noise*; and attacks such as *flooding attack*, *replay attack* and *tampering attack*.

The networks studied on this paper are mapped into a specific context of use, representing each one different situations of the real world. The specifications of each network are represented in Table 10.

4.3.1. Criteria under evaluation

In the paper, the authors evaluate the impact of each perturbation in the network considering two performance measures: the applicative throughput (or Goodput), and

Table 10: Experimental configuration of Network A and Network B presented in [36].

Network	RP	Speed	Area	Range	Workload
A	AODV	6 nodes: 0 m/s	30 x 50 m	20 m	Text data (500 bps)
B	OLSR	6 nodes: [0-3] m/s	300 x 150 m	125 m	VoIP traffic (100 Kbps)

RP: Routing Protocol

the increment of delay (or Jitter); and two measures of dependability: the percentage of packets correctly delivered (or Integrity), and the percentage of time the network is ready to be used (or Availability). Table 11 illustrates the values measured by the authors for each considered perturbation in Network A and Network B.

Table 11: Measures obtained from the case study of ad hoc networks.

			Perturbations				
	Measure	**Golden run**	**Signal attenuation**	**Ambient noise**	**Replay attack**	**Flooding attack**	**Tampering attack**
Netw. A	*Availability (%)*	92.94	73.98	88.74	93.89	51.22	90.12
	Integrity (%)	99.03	97.53	92.12	98.54	97.56	8.01
	Goodput (Kbps)	0.19	0.17	0.18	0.19	0.10	0.19
	Jitter (ms)	319.89	353.45	332.66	300.78	721.66	312.44
Netw. B	*Availability (%)*	95.14	73.9	87.00	75.20	65.00	90.33
	Integrity (%)	98.34	98.73	92.26	99.44	98.23	62.90
	Goodput (Kbps)	96.45	85.19	90.56	70.90	80.18	96.45
	Jitter (ms)	199.98	210.23	211.11	220.88	230.55	195.00

4.3.2. Scales of measure

This case study has an interesting detail that can not be found in the previous case studies. Unlike the others, the authors establish a discrete three level criteria (Low, Medium or High) to evaluate the impact of perturbations on the measures: *"In this way, the impact is considered low, medium or high if the measure is degraded underneath 5%, over 5% or over 10% respectively, according to the golden run results"*. Accordingly, (5) and (6) define a discrete three-level criterion function for the-higher-the-better measures (availability, integrity and goodput), and the-lower-the-better measure (jitter), respectively. In these equations, $B(m_i)$ refers to the baseline computed value for measure m_i.

$$s_i = c_i(m_i) = \begin{cases} 0, & m_i \leq 0.90 \cdot B(m_i) \\ 50, & 0.90 \cdot B(m_i) < m_i < 0.95 \cdot B(m_i) \\ 100, & m_i \geq 0.95 \cdot B(m_i) \end{cases} \tag{5}$$

$$s_i = c_i(m_i) = \begin{cases} 100, & m_i \leq 1.05 \cdot B(m_i) \\ 50, & 1.05 \cdot B(m_i) < m_i < 1.10 \cdot B(m_i) \\ 0, & m_i \geq 1.10 \cdot B(m_i) \end{cases} \tag{6}$$

4.3.3. Preferences aggregation

After identifying the three different levels quantifying the impact of perturbation on the obtained measures, authors do not detail how to determine the impact of the perturbation on the whole system. Instead, they perform a qualitative analysis (also based on three discrete levels) with no clear rules about how it was perform. Accordingly, as no special requirements for the scores aggregation are defined, equitable weights and neutral aggregations have been considered for all the branches of the proposed quality model shown in Fig. 6.

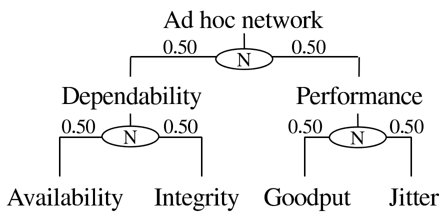

Figure 6: Quality model to determine the impact of each perturbation on the considered ad hoc network.

Table 12: Characterization of the impact level according to the scores for Network A and Network B.

	Perturbation	**Score**	**Quality Model Impact level**	**Original Impact level**
Network A	Signal Attenuation	25.0	High	High
	Flooding attack	25.0	High	High
	Ambient noise	75.0	Low	Low
	Replay attack	100.0	Low	Low
	Tampering attack	75.0	Low	Low
Network B	Signal Attenuation	37.5	High	High
	Flooding attack	25.0	High	High
	Ambient noise	50.0	Medium	Medium
	Replay attack	25.0	High	High
	Tampering attack	62.5	**Medium**	**Low**

4.3.4. Analysis of results

The global scores obtained for each of the networks are listed in Table 12. As previously stated, authors make a qualitative analysis of the impact of each perturbation on each measure to determine the actual impact of the perturbation on the whole system (Low, Medium, High). Since there is no explicit information about how this analysis is performed, we propose to determine the impact level according to the global score obtained for each perturbation. As measures are normalized according to their deviation with respect to the baseline, final scores between 100 and 70 indicate that the perturbation is barely affecting the system (low impact level), scores between 69 and 40 show a medium impact level, and scores between 39 and 0 reflect a high impact.

The resulting classification for perturbations affecting both networks matches that obtained in the original paper, but for the *tampering attack* on Network B, which is now classified as having a Medium instead of Low impact. This divergence obviously derives from the vague description of the characterization performed on the original paper. As in Section 4.2.4, this shows the necessity of precisely defining the criteria and procedure followed during the results analysis. Otherwise, the same results could be interpreted in a completely different way, preventing this process from being repeatable.

In addition to the analysis performed in the original work, and to show the potential of the proposed approach, it could be possible to define a new quality model to help evaluators when deploying a new routing protocol in the network, tuning routing protocol parameters, or introducing new fault tolerance mechanisms, for instance. This model could take into account the information extracted from this case study, so those perturbations presenting a high impact on the system could be aggregated with equal weight under *critical* perturbations category, and those with a lower impact could be grouped under the *non-critical* perturbations category. The severity of critical perturbations could be remarked by punishing those critical scores with a low value. So, a mandatoriness relationship with the simultaneity operator S, could be used to illustrate this purpose. Medium and low impact perturbations could present different weights, like 0.75 and 0.25 respectively, to reflect their different importance. Fig. 7 and 8 show the resulting quality models for Network A and Network B respectively.

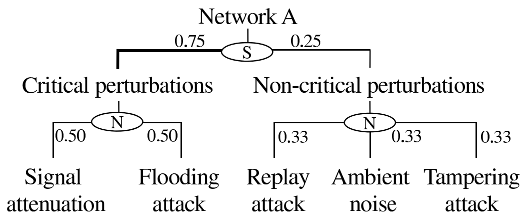

Figure 7: Aggregation of perturbations for Network A (WSN).

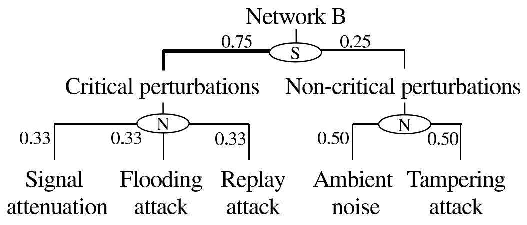

Figure 8: Aggregation of perturbations for Network B (MANET).

5. Conclusion

In this paper, we have presented a methodology to make straightforward, consistent and objective the analysis of dependability benchmarking measures, a big challenge in todays distributed systems. Our methodology addresses how to adequately select and gather the types of measures to represent the system quality. Since there are distinct ways to do it, our methodology enables the generation of multiple representations (or quality-scores-based models) from the same system when different criteria are applied by evaluators. Among their benefits, the scores obtained from our methodology are repeatable simply following the explicit criteria defined in each quality model, which eases the comprehension of evaluation assumptions, thus assisting the benchmark user to minimize errors during the results interpretation. Indeed, the model provided becomes not only a way to express which measures are under consideration, but also a mean to drive their analysis in a more objective and systematic way. Objectiveness is important to minimize the provision of biased conclusions, while the systematization of the approach enables the provision of tools to assist users in the consideration of a big number of targets, faults and measures during experimentation.

Furthermore, our methodology results a very useful approach to overcome the problem of measures scalability and gets a more quantitative vision of the system despite the multiple aggregation of scores. Nevertheless, regarding previous results, the application of this technique requires the adequate definition of the quality thresholds (X_{min} and X_{max}) for each criterion functions, the weight (w_i) assigned to each score within the same hierarchical level, and the operator type (o_i) in charge of the scores aggregation. All these aspects highly depend on the applicative context the system is conceived to be deployed in. Despite the selection of these parameters may result subjective, our methodology forces the benchmark performer to make them explicit, which eases the transparency and comparison between systems. This is an advantage with respect to traditional benchmarking, where the criteria considered usually remain subjective and hidden to the benchmark report consumer.

The application of our methodology in the case studies presented in the paper begin from a stage of the evaluation where measures are already available, which is very often when authors compare their results. However, conversely to other measures-aggregation techniques, our methodology could play an active role during the benchmark definition, being applied from the very beginning, i.e., before benchmark experiments are carried out. Considering this point is a first step towards improving the characterization of the wide amount of applicative domains in distributed systems. We argue that this type of approaches can be useful not only to quantify the impact of faults with respect to the actual application context (where components and systems are planned to be deployed), but for the comparison and selection of those targets which best fit the system requirements.

In the future work, we ambition to provide evaluators different templates with precomputed parameters that they could customize for their particular deployments to semi-automate the application of this methodology for the quantitative benchmarking of different types of distributed systems.

Acknowledgments

This work is partially supported by the Spanish project ARENES (TIN2012-38308-C02-01), ANR French project AMORES (ANR-11-INSE-010), the Intel Doctoral Student Honour Programme 2012, and the "Programa de Ayudas de Investigación y Desarrollo" (PAID) from the Universitat Politècnica de València.

References

[1] TPC. Transaction Processing Performance Council. [Online]. Available: http://www.tcp.org/, 2013.

[2] DBench. Dependability Benchmarking. IST Programme, European Commission, IST 2000-25425, [Online]. Available: http://www.laas.fr/DBench, 2003.

[3] Raquel Almeida, Naaliel Mendes, and Henrique Madeira. Sharing experimental and field data: the amber raw data repository experience. In *Distributed Computing Systems Workshops (ICDCSW), 2010 IEEE 30th International Conference on*, pages 313–320. IEEE, 2010.

[4] A. Ceccarelli. *Analysis of critical systems through rigorous, reproducible and comparable experimental assessment*. PhD thesis, Università Degli Studi di Firenzi, 2012.

[5] Marco Vieira and Henrique Madeira. A dependability benchmark for oltp application environments. In *Proceedings of the 29th international conference on Very large data bases - Volume 29*, VLDB '03, pages 742–753. VLDB Endowment, 2003.

[6] Ateret Anaby-Tavor, Avigdor Gal, and Alberto Trombetta. Evaluating matching algorithms: the monotonicity principle. In *IIWeb*, pages 47–52, 2003.

[7] K. Kanoun, Y. Crouzet, A. Kalakech, A.-E. Rugina, and P. Rumeau. Benchmarking the dependability of windows and linux using postmark/spl trade/ workloads. In *16th IEEE International Symposium on Software Reliability Engineering (ISSRE)*, pages 10–20, nov. 2005.

[8] European New Car Assessment Programme (EuroNCAP). EuroNCAP. [Online]. Available: http://www.euroncap.com/, 2013.

[9] International Organization for Standardization (ISO) and the International Electrotechnical Commission (IEC). ISO/IEC 25000. Software Engineering - Software product Quality Requirements and Evaluation (SQuaRE) - Guide to SQuaRE. Geneve ISO, 2010.

[10] Jim Gray. *Benchmark Handbook: For Database and Transaction Processing Systems*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1992.

[11] EEMBC's Benchmarks. Embedded Microprocessor Benchmark Consortium, [Online]. Available: http://www.eembc.org/benchmark/products.php, 2014.

[12] Karama Kanoun and Lisa Spainhower, editors. *Dependability Benchmarking for Computer Systems*. Wiley and IEEE Computer Society Press, 2008.

[13] Henrique Madeira, Marco Vieira, et al. The olap and data warehousing approaches for analysis and sharing of results from dependability evaluation experiments. In *IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pages 86–86, 2003.

[14] Amber project. Amber (assessing, measuring, and benchmarking resilience). FP7 Coordination Action, European Commission, 2008.

[15] Giulio Concas, Michele Marchesi, Sandro Pinna, and Nicola Serra. Power-laws in a large object-oriented software system. *IEEE Transactions on Software Engineering*, 33:687–708, October 2007.

[16] Yazeed A. Al-Sbou, Reza Saatchi, Samir Al-Khayatt, Rebecca Strachan, Moussa Ayyash, and Mohammad Saraireh. A novel quality of service assessment of multimedia traffic over wireless ad hoc networks. In *Proceedings of the 2008 The Second International Conference on Next Generation Mobile Applications, Services, and Technologies*, pages 479–484, 2008.

[17] Michael F. Morris. Kiviat graphs: conventions and figures of merit. *ACM/Sigmetrics Performance Evaluation Review*, 3(3):2–8, 1974.

[18] David de Andres, Juan Carlos Ruiz, and Pedro Gil. Using dependability, performance, area and energy

consumption experimental measures to benchmark ip cores. In *Forth Latin American Symposium on Dependable Computing (LADC)*, pages 49–56, 2009.

[19] J. P Correia and J Visser. Certification of technical quality of software products. In *Proceedings of the International Workshop on Foundations and Techniques for Open Source Software Certification*, pages 35–51, 2008.

[20] ThomasL. Saaty. What is the Analytic Hierarchy Process? In Gautam Mitra, HarveyJ. Greenberg, FreerkA. Lootsma, MarcelJ. Rijkaert, and HansJ. Zimmermann, editors, *Mathematical Models for Decision Support*, volume 48 of *NATO ASI Series*, pages 109–121. Springer Berlin Heidelberg, 1988.

[21] Nian Liu, Jianhua Zhang, Hao Zhang, and Wenxia Liu. Security assessment for communication networks of power control systems using attack graph and mcdm. *Power Delivery, IEEE Transactions on*, 25(3):1492–1500, 2010.

[22] Christopher W Karvetski, James H Lambert, and Igor Linkov. Scenario and multiple criteria decision analysis for energy and environmental security of military and industrial installations. *Integrated Environmental Assessment and Management*, 7(2):228–236, 2011.

[23] M. Martínez, D. de Andrés, J.-C. Ruiz, and J. Friginal. From Measures to Conclusions Using Analytic Hierarchy Process in Dependability Benchmarking. *Instrumentation and Measurement, IEEE Transactions on*, 63(11):2548–2556, 2014.

[24] M. Martínez, D. de Andrés, and J.-C. Ruiz. Gaining Confidence on Dependability Benchmarks' Conclusions through "Back-to-Back" Testing. In *2014 Tenth European Dependable Computing Conference*, pages 130–137, 2014.

[25] M. Martínez, D. de Andrés, J.-C Ruiz, and J. Friginal. Analysis of results in dependability benchmarking: Can we do better? *International Workshop on Measurements and Networking*, pages 127–131, 2013.

[26] A. Bondavalli, A. Ceccarelli, L. Falai, and M. Vadursi. Foundations of measurement theory applied to the evaluation of dependability attributes. In *Dependable Systems and Networks, 2007. DSN '07. 37th Annual IEEE/IFIP International Conference on*, pages 522 –533, june 2007.

[27] P.S. Bullen. *Handbook of Means and Their Inequalities*. Mathematics and Its Applications. Springer, 2003.

[28] J.J. Dujmovic and R. Elnicki. *A DMS Cost/Benefit Decision Model: Mathematical Models for Data Management System Evaluation, Comparison, and Selection*. National Bureau of Standards, Washington D.C., No. GCR 82-374. NTIS No. PB 82-170150, 1982.

[29] R.R Yager. A note on weighted queries in information retrieval systems. *Journal of The American Society for Information Science*, 38(1):23–24, 1987.

[30] Andrea Saltelli, Marco Ratto, Terry Andres, Francesca Campolongo, Jessica Cariboni, Debora Gatelli, Michaela Saisana, and Stefano Tarantola. *Global sensitivity analysis: the primer*. John Wiley & Sons, 2008.

[31] Evangelos Triantaphyllou and Alfonso Sánchez. A sensitivity analysis approach for some deterministic multi-criteria decision-making methods*. *Decision Sciences*, 28(1):151–194, 1997.

[32] Y. Chen, J. Yu, and S. Khan. Spatial sensitivity analysis of multi-criteria weights in gis-based land suitability evaluation. *Environmental Modelling & Software*, 25(12):1582 – 1591, 2010.

[33] Marco Dur?, Jo?and Vieira and Henrique Madeira. Dependability benchmarking of web-servers. In Maritta Heisel, Peter Liggesmeyer, and Stefan Wittmann, editors, *Computer Safety, Reliability, and Security*, volume 3219 of *Lecture Notes in Computer Science*, pages 297–310. Springer Berlin Heidelberg, 2004.

[34] SPEC's Benchmarks. Standard Performance Evaluation Corporation, [Online]. Available: https://www.spec.org/benchmarks.html, 2014.

[35] Jim Gray. A measure of transaction processing 20 years later. *CoRR*, abs/cs/0701162, 2007.

[36] J. Friginal, D. de Andres, J.-C. Ruiz, and P. Gil. On selecting representative faultloads to guide the evaluation of ad hoc networks. In *Dependable Computing (LADC), 2011 5th Latin-American Symposium on*, pages 94 –99, april 2011.