Dependability Benchmarking: making choices in an n-dimensional problem space

Henrique Madeira
University of Coimbra, Portugal
henrique@dei.uc.pt

Philip Koopman
Carnegie Mellon University, USA
koopman@cmu.edu

Abstract

Dependability benchmarks should provide cost-effective ways to evaluate the behavior of components and computer systems in the presence of faults, allowing the quantification of dependability attributes or the characterization of the systems into well defined dependability classes. Beyond existing evaluation techniques, a dependability benchmark should represent an agreement accepted by the computer industry or/and by the user community, and specify the measures, the methods, and techniques required to perform measurements. This paper discusses the different dimensions involved in the dependability benchmarking problem and presents basic components required to specify dependability benchmarks. Although several obstacles still persist and are currently the subject of research, the definition of all the dimensions of the problem and the agreement of the community on a basic set of components that constitute a possible framework for dependability benchmarking seem to us a first step in the proposal of actual dependability benchmarks.

1. Introduction

Our society is increasingly dependent on the correct service of computers. Classical features such as raw performance and functionality have long driven the computer industry to improve their products. But now, dependability and maintainability are becoming seen as equally important. Unfortunately, while there are relatively straightforward ways to evaluate and compare performance and functionality of different systems or components, the evaluation of dependability features is much more difficult.

The ascendance of networked information in our economy and daily lives has increased awareness of the importance of dependability features. In many cases, such as in e-commerce systems, computer outages can result in a huge loss of money or in an unaffordable loss of prestige for companies. In more personal applications such as access to news and weather, dependability problems are not as catastrophic, but are nonetheless undesirable. In part because it is futile to attempt to make every computer on the Internet work perfectly all the time, system designers are increasing emphasis on tradeoffs involving both functionality and performance, and in particular designing systems to operate in degraded mode or with reduced performance in the presence of faults or other unavoidable upsets. Operating in degraded modes is usually desirable, but complicates the notion of measuring dependability.

Another clear trend is the use of Commercial Off-The-Shelf (COTS) components and COTS-based systems in application areas requiring high dependability. Using COTS components is seen as desirable to reduce costs and speed up time to market. However, successfully creating dependable systems with COTS components that may not have been designed to be particularly robust demands practical ways to evaluate the dependability and the behavior in the presence of faults within these components/systems.

Even the validation of fault handling mechanisms, which are just a piece of the dependability picture, is traditionally a challenging task. Given the huge complexity involved in the design of a computer system (hardware, operating system, application software, user interface, etc.), there is no single general approach for dependability evaluation and validation. Instead, several methods have been used, ranging from pure modeling and analytical techniques to experimental approaches based on fault injection and robustness testing [1, 2, 3, 4, 5]. Most of the techniques used for the evaluation of dependability and validation of fault handling mechanisms have been developed for mission-critical systems and applications or for the high-end business-critical area, and therefore might make assumptions about design or operating environment that affect their suitability for more mainstream computing components.

The goal of dependability benchmarking is to provide cost-effective ways to evaluate the behavior of components and computer systems in the presence of faults, allowing the quantification of dependability attributes or the characterization of the systems in well defined dependability classes. Beyond existing experimental techniques such as fault injection and robustness testing, we feel that dependability

benchmarking must provide a uniform, repeatable, comparable way of performing this evaluation.

Dependability benchmarking is subject of a growing interest today. Both the research community and the industry are involved in research projects [6, 7] and groups meant to advance the dependability benchmarking area. Particularly, the IFIP Working Group 10.4 has created a Special Interest Group (SIG) to promote the research, practice, adoption, and dissemination of benchmarks for computer-related system dependability (the authors of this paper are the chair and co-chair of this SIG). The activities of the SIG endeavor to attain a clear understanding and articulation of the fundamental reasons for undependability across multiple disciplines and have thus far been focused on the identification of all the fundamental issues that must be addressed to create dependability benchmarks. A major task for this group has been the establishment of a framework for dependability benchmarking. Although this discussion is still ongoing within the SIG, this paper presents the authors' vision on this framework and hopes to enrich the discussion by soliciting input from a larger diverse community.

In the next section we present a general view of the benchmarking problem, defining many of the dimensions of dependability benchmarking. Section 3 presents our choices for a framework for dependability benchmarking, followed by some concluding remarks in Section 4.

2. Dependability benchmarking dimensions

Benchmarking is an experimental approach to measure well-defined features of a system or component according to an agreed (i.e., accepted as standard) set of methods and procedures. No matter what a benchmark is intended to measure (most of the existing benchmarks for computer systems are intended for performance measurement) it has a set of properties that makes it different from just a measuring/evaluating technique.

First of all, a benchmark represents an agreement that must be accepted by the industry or/and by the user community. This agreement (i.e., the benchmark) not only specifies the features that are relevant to be measured but also describes the methods and techniques required to perform the measurements. Additionally, a benchmark must be scalable to address systems of different sizes, portable to be used across different platforms and operating systems, repeatable to provide confidence in the measurements, and easy to use. Most importantly, useful benchmarks strive to measure a quality that is important to users and vendors, and often must go beyond things that simply happen to be convenient to measure.

Benchmarks are normally targeted to well defined application areas or to specific types of systems. This is the only practical way to cope with the huge diversity of applications and systems in the computer industry. However, in spite of this inevitable diversity, we hope that all dependability benchmarks might share a common framework, at least at an abstract level. The framework that is proposed in this paper defines all the key components of a dependability benchmark and the general approach of benchmarking for dependability. In this context, an actual dependability benchmark is just an instantiation of this general framework to a specific application domain or to particular kind of computer system.

The definition of a framework for dependability benchmarking requires first of all the identification and clear understanding of all the dimensions of the problem. Then, defining the framework corresponds to making choices in the different dimensions of the problem. To simplify this process we divide the problem space in dimensions of the problem and components of a dependability benchmark. The distinction between these two groups is subtle (in general, all of them could be considered as dimensions of the problem, but we find making this distinction useful). Basically, we want to isolate in the first group the dimensions of the problem that affect the dependability benchmarks in terms of providing an application area, operating environment, or

Dimensions and Components		Comments
Dimensions	Products vs. Processes	Identify the target of the benchmarks: product or the manufacturing process.
	Life cycle phase	Identify the phase in the product life that will be addressed in the benchmark.
	Application area	Identify the application area with the adequate granularity (specificity).
	Operating environment	Typical environment for an application area and the way it affects the benchmark components.
	User Perspective	Different perspectives for the benchmark measures (*e.g.*, end user, developer).
Components	Target	Defines the target system expected system behavior in different fault situations.
	Measures	Defines the measures (results) of the benchmark.
	Workload	Defines a working profile that should be representative of an application area.
	Upsetload	Defines the set of upsets, stressful conditions and faults that could affect the system.
	Procedures and rules	Defines the procedures and the rules to perform the benchmarking.

Table 1. Dimensions of the problem and possible components of a dependability benchmark.

constraint set, but cannot be considered as a direct component of a dependability benchmark. For example, "Products vs. Processes" is a dimension of the problem that can deeply affect the form of the benchmark (benchmarks could be quite different if we decided to benchmark the product or the manufacturing process to produce that product) but cannot be considered a component of a benchmark (it is something that we have to decide beforehand and include in the process of defining a dependability benchmark). Table 1 presents a breakdown of the major dimensions and components.

Dependability benchmarking is an n-dimensional problem space. The definition of a possible framework for dependability benchmarking corresponds to making choices for the different dimensions and/or defining the way these choices affect the components of the benchmark framework. In this sense different frameworks for the different flavors of dependability benchmarking can co-exist.

3. A dependability benchmarking framework

In this section we propose and discuss a possible framework for dependability benchmarking, which corresponds to making choices for the different dimensions and/or defining the way these choices affect the components of a dependability benchmark. There are of course many details that fit within these general categories, but we feel that the categories selected serve to focus attention on the different general types of factors that are relevant for most cases.

Product vs. process - Although benchmarking the process of creating a computer system or component could be conceivable (in fact, the ISO 9000 series of standards are based on the idea of certifying processes), the traditional notion of benchmarking is focused on the products. Products can be benchmarked either directly by experimentation or indirectly through the inspection of the product features, as stated in the product documentation. Our preference concerning this dimension is to benchmark actual products via direct observation and experimentation.

Life cycle phase - The main phases of a typical life cycle of a computer system or component are the requirements, design, implementation/manufacturing, deployment, and operational phases. A dependability benchmark could be specifically targeted to any of these phases (e.g., design: evaluate design correctness or design robustness to component failure; implementation/manufacturing: evaluate the impact on dependability of manufacturing defects or coding errors, etc.).

However, a product-based benchmarking means that we are mainly interested in benchmarking dependability features at the operational phase of the target system or component to the maximum degree possible (which also encompasses all the other phases). The operational phase includes normal system operation and maintenance in a realistic environment.

Application area - The application area is clearly a key dimension. The division of the application spectrum into well-defined application areas is necessary to cope with the diversity of systems and applications and to make it possible to decide on most of the other dimensions. In fact, most of the problem dimensions and dependability benchmark components are very dependent on the application area. For example, the benchmark measures, the working profile, the operational environment, or the typical upsets and faults that may affect the systems can only be defined if we chose first a specific application area. The main difficulty with this dimension is clearly the establishment of the right granularity to divide the application spectrum. The application areas must be general as possible but, at the same time, specific enough to allow the definition of all the aspect directly dependent on the application area. Obviously, different application areas will tend to need different dependability benchmarks.

Operating environment - The operating environment is traditionally one of the things that affect system dependability. In fact, many computers faults are induced by external sources, and these are intimately related to the operating environment. The main problem is to identify a way to take into account operating environment features in the benchmarking process. This could be particularly difficult if we include the human aspects such as the actions of operators and users. One possible solution could result from the observation that the operating environment is very dependent on the application area, as mentioned above. Thus, it should be possible to define a typical operating environment for a given application area which includes the set of upsets and external faults that are typical for that application area. In other words, the operating environment is dependent on the application area and is one of the things to take into account in the definition of the upsetload.

User perspective - A dependability benchmark could have different kinds of users (e.g., end user, system integrator, developer). These different user perspectives affect mainly the type of measures expected from the benchmark and the type of target system. Typically, end users are interested in general system dependability attributes (e.g., unconditional system availability) while a system integrator or a developer could be more interested in specific measures related to the behavior of the system or a component in the presence of faults (e.g., error detection coverage or recovery efficiency). Thus the different user perspectives potentially create demands for different types of measurements from a benchmark suite.

So far, we have proposed to focus our framework on the benchmarking of products in a direct way as they are

used in its operational phase. The framework is applied to a specific application area, which works as a key dimension in the definition of other dimensions and benchmark components. We assume that a typical operating environment can be characterized for each application (and depends only on the application area) and that different user perspectives must be considered in the same benchmark. In this context, Figure 1 represents the main components of a dependability benchmark.

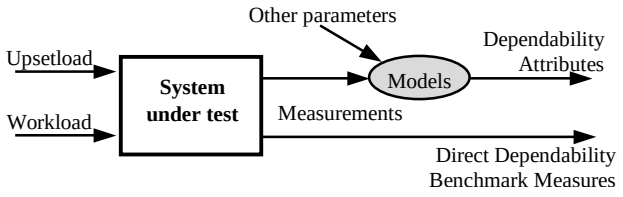

Figure 1. Main dependability benchmark components.

Measures - Dependability benchmarks must include direct measures related to the behavior of a computer in the presence of upsets (faults and stressful conditions). Another form of direct measures is the performance measures in both normal conditions (baseline) and in the presence of upsets. These measures express the impact of upsets in the system performance and functionality and allow the measurement of graceful degradation conditions, which includes reduced performance modes and/or operation in reduced functionality.

It is worth noting that the need for establishing baseline performance measures must not be confused with performance benchmarking, as in our case we just need a baseline performance (i.e., non optimized beyond normal tuning) to evaluate the relative effect of faults and upsets on the of performance degradation and on the quality of results.

The direct measures are dependent on the system under test and on the application area. This last case is particularly true for the measures related to impact of upsets in the system performance and functionality. Another important thing is that the direct measures represent the only system observation points in the proposed benchmark. The techniques and instrumentation required to collect these measures are still being researched.

Dependability attributes (e.g., reliability, availability, safety) cannot be obtained directly from running the experiments specified in a benchmark. If these attributes are required (and end users are normally very interested in attributes such as availability and reliability) we need modeling and some input from field experience, including fault distributions and fault probabilities which are very hard to estimate (the other parameter in the models ellipse in Figure 1). These are still direct measures, but novel ones in the context of benchmarking, because performance benchmarks only rely on direct measures taken at the time of a specific benchmarking event.

Although not explored in the framework proposed in this paper, another form of benchmark measures is the definition of dependability classes. In this approach, systems are divided in classes according to dependability criteria. The benchmarking process in this case consists of identifying an appropriate dependability class for the system under test. These measures have been proposed by Don Wilson [8] with additional contributions from other SIG members, and are making good progress toward a formal proposal.

Workload - The workload represents a typical operational profile for the considered application area. Widely accept performance benchmarks can provide workloads for many application areas and clearly show that it is possible to agree on a workload that reasonably represents a given application area. In the case of dependability benchmarking the concept of workload must be expanded to take into account preventive maintenance actions that are conducted as part of routine operations for a given application area. For example, in a database server events such as system backups or log files management are included in the typical profile and should be considered in the workload.

System under test - The system under test can be defined in a simplistic way as a system able to run the workload. That is, no particular assumptions on the system architecture or on the existence of specific techniques in the system should be made. All it ought to be required is that the system be able to run the workload. It should be noted that this reflects the actual situation in the field, as systems using different architectures and techniques are used to run similar applications. Of course some documentation of the actual system configuration employed would be appropriate to enable informed inter-system comparisons.

The benchmarking of a specific component in a system must be carried out in the same conditions of system benchmarking (the component integrated in the system, the system running the workload, etc.). The primary difference between system and component benchmarking is the set of measures produced. However it might be possible to benchmark components using a synthetic workload or other workload that is a subset of a complete system workload depending on the particular component interface.

Upsetload - The upsetload consist of a set of faults and stressful conditions that are intended to emulate all the real exceptional situations the system would experience in the field. This is clearly dependent on the operating environment for the external upsets, which dependents in turn on application area. Internal faults (e.g., software faults and some hardware faults) are mainly determined by the target system implementation. Because few assumptions are made on the actual target system structure (physical and logical), it is likely that the upsetload must

be equivalent on a statistical basis for different target systems (used in the same application area) rather than being literally identical.

The definition of representative upsetloads is probably the most obscure and difficult part of defining a dependability benchmark. Even for well-defined application areas, the definition of the relative percentages of the different classes of upsets and faults is essentially a best guess process, much less accounting for potential fine-grain interactions among workload and upsetload. Additionally, the mechanisms and instrumentation required introduce this upsetload in the target system are clearly open research issues.

Procedures and rules - It is well known that any benchmark can be "gamed" to produce optimistic results. A dependability benchmark would have to include standards for conducting measurement and to ensure uniform conditions for measurement. In addition to the obvious items such as system configuration disclosures for performance metrics, dependability metrics might also include requirements or disclosures involving all factors that affect dependability. Another important aspect is the need of scaling rules to adapt the same benchmark to systems of very different sizes (but used in the same application area). These scaling rules would define the way the system load can be changed.

The proposed framework consists of basic components required to specify a dependability benchmark and the way these components are related to the dimensions of the benchmarking problem. Although several obstacles still persist and are currently subject of research (e.g., representativeness of upsetloads, measures, instrumentations techniques, etc), the definition of all the dimensions of the problem and the agreement of the community on the basic set of components that can be fault in all dependability benchmarks seem to us as the first step to the proposal of actual dependability benchmarks.

4. Conclusion

Dependability benchmarking is an n-dimensional problem space that is currently a subject of intense discussion in the dependability community, and particularly within the IFIP WG 10.4 Special Interest Group on Dependability Benchmarking. Although there is a general agreement in the community on the identification of the key components of possible dependability benchmarks (measures, workload, upsetload,…), improved understanding of all the dimensions involved in this problem is essential to identify all the components of future dependability benchmarks and to devise a general approach. Of course there are many factors that determine the success or failure of a benchmarking effort beyond the technical content of the benchmark. Some of the factors that must be accounted for include the cost of conducting a benchmark trial, the accessibility of benchmarking materials, keeping benchmark implementations fresh in the face of evolving technology, and of course political issues as well as the ramifications of benchmarking results in different marketplaces. While the importance of these factors must not be underestimated, we believe that it is also important to advance one or more concrete benchmarking technical approaches as a step in addressing a wider range of concerns.

5. References

[1] K. Kanoun, M. Kaâniche and J.-C. Laprie, "Qualitative and Quantitative Reliability Assessment", *IEEE Software*, 14 (2), pp.77-86, mars 1997.

[2] N. Suri and P. Sinha, "On the Use of Formal Techniques for Validation", in *Proc. 28th Int. Symp. on Fault-Tolerant Computing* (FTCS-28), (Munich, Germany,1998.), pp.390-399, IEEE CS Press, June 1998.

[3] P. Folkesson, S. Svensson and J. Karlsson, "A Comparison of Simulation Based and Scan Chain Implemented Fault Injection", *in Proc. 28th Int. Symp. on Fault-Tolerant Computing* (FTCS-28), (Munich, Germany), pp.284-293, IEEE Computer Society Press, June 1998.

[4] D. Costa, T. Rilho, H. Madeira, "Joint Evaluation of Performance and Robustness of a COTS DBMS through Fault-Injection", *Proc. of DSN 2000 – Int. Conference on Dependable Systems and Networks*, New York, USA, June 25-28, 2000.

[5] P. Koopman and J. DeVale, "Comparing the robustness of POSIX Operating Systems", *Proceedings of 29th International Symposium on Fault-Tolerant Computing*, June 15-18, Madison, Wisconsin, 1999, pp. 22-29

[6] "Dbench – Dependability Benchmarking", Technical Annex of the project Dbench, funded by the European Commission. This project started on January 2001 and involves the following research centers, universities and enterprises: LAAS, France, Univ. of Erlangen, Germany, Univ. of Chalmers, Sweden, Univ. of Valência, Spain, Critical Software Ltd, Portugal, and Microsoft as a sponsor.

[7] Aaron Brown and David Patterson, "Towards availability benchmark: a case study of software RAID systems", *Proceedings of 2000 USENIX Annual Technical Conference*, San Diego, California, USA, June 18-23, 2000, pp 263-276.

[8] Don Wilson, "Benchmark availability classes", *working document of the IFIP WG 10.4 Special Interest Group on Dependability Benchmarking*, 2001.

Acknowledgements

Funding for this paper was provided, in part, by DARPA under contract DABT 63-96-C-0064, and by the Portuguese Government/European Union through R&D Unit 326/94 (CISUC).