# Progress on Defining Standardized Classes for Comparing the Dependability of Computer Systems

Don Wilson
NonStop Enterprise Lab.
Hewlett-Packard Company
Aptos, CA, USA

Brendan Murphy
Microsoft Corporation
Cambridge
UK

Lisa Spainhower
International Business Machines
Poughkeepsie, NY
USA

## Abstract

A number of the industrial partners of the IFIP WG 10.4 Dependability Benchmarking SIG (SIGDeB) have identified a set of standardized classes for characterizing the dependability of computer systems. The proposed classification system seeks to enable comparison of different computer systems in the dimensions of availability, data integrity, disaster recovery, and security. Different sets of criteria are proposed for computer systems that are used for different application types, e.g. transaction processing, process control, etc. This paper describes the classification system, and gives a progress report on the work to fill in the details of the classification criteria

## 1. Introduction

As computer systems become more and more continuously integrated into the daily activities of business, engineering, and scientific users, there is increased interest in being able to evaluate and compare the dependability of these systems. Considerable research has been done in an effort to establish benchmark tests for this purpose (e.g., see [1-5]), usually based on some form of fault-injection testing focused on single computers [6,11].

In spite of these efforts, and even considering that fault injection techniques are commonly used by developers to assess and tune their designs (e.g., see [7-10]), nothing has emerged which has gained even modest adoption in the industry for making comparisons among systems. Researchers acknowledge that:

- emulated faults will not represent the variety and scope of actual field faults [6, 14],
- fault injection cannot predict actual availability or MTBF [13],
- comparison of dissimilar architectures is extremely problematic [7, 13, 14].

The authors, working as members of the IFIP WG 10.4 Dependability Benchmarking SIG (SIGDeB) [15], are proposing a different method for making dependability comparisons. This method is to create a standardized classification system that could rate systems in each of the dimensions that affect dependability.

## 2. Dimensions of Dependability

Unlike performance benchmarks, which need to compare only the rate at which specific, pre-defined work gets done, dependability comparisons must consider many different aspects [12]. Is the system accessible when needed? Are the results correct? Is the data protected from physical hazards and unauthorized access?

To simplify the creation of comparison classes, it is useful to separate the various threats to dependability and treat them as different dimensions of the problem space. The authors have chosen Availability, Data Integrity, Disaster Recovery, and Security as the dimensions to be considered. When a system is rated according to the proposed classification scheme, it would receive an independent rating for each of these dimensions.

These dimensions are not truly independent; for example corrupted data could easily make an application unavailable. Thus, the problem space could certainly be divided up differently. However, these dimensions were chosen because they are readily understandable and are important concerns to users. It may also be argued that there are still other dimensions to the problem, such as physical safety. The structure of this proposal makes it simple to add further dimensions if they are deemed useful enough to the user community.

Unlike performance benchmarks, it was felt that dependability assessments should not restrict the system under test to be a single computer. Very few customers, requiring a highly dependable solution, would implement it on a single computer with a single data store. Any solution would likely be designed with a minimum of two interconnected computers, with some form of highly dependable storage configuration. As the system configuration is unrestricted, then any comparison of different systems should include the cost of each solution.

| | |
|---|---|
| **Transaction Processing** | Typical applications are order processing, automated billing, credit authorization, automated retail, securities trading, reservations. |
| **Message Handling** | Typical applications are telephony, email, packet switching and routing, protocol conversion. |
| **Process Control** | Typical applications are manufacturing control, embedded device controllers, servo systems, network and system management. |
| **Search and Retrieval** | Typical applications are web-page serving, decision support, classical business reporting, broadcasting. |
| **Analytical Calculation** | Typical applications are simulation, modeling, and scientific data reduction. |

**Table 1: Application Types**

## 3. Application Space

Users of different types of applications tend to have differing priorities when it comes to dependability.

A telephony application prizes availability and responsiveness very highly, but can afford minor data errors.

A stock trading application prizes data integrity above all else, requires high availability, must often adhere to strict securities regulations, but may have frequent off-hours where maintenance can be done.

A factory control system finds availability and accuracy essential, but is not concerned about operating during a power failure.

The authors have accordingly divided the application space into types that appear to have similar dependability requirements, as listed in Table 1.

The proposed classification scheme will define a different set of classes for each application type, on each dependability dimension. This structure is shown in Table 2.

The boundaries between classes are intended to be natural breakpoints in the spectrum of user-perceived availability requirements. The highest class is always intended to be essentially perfect behavior, whether or not it is achievable with current technology.

| Application Type | Availability Classes | Data Integrity Classes | Disaster Recovery Classes | Security Classes |
|---|---|---|---|---|
| **Transaction Processing** | 1. Perfection<br>2a. Retryable Workload<br>2b. Retryable / Planned Outages<br>3a. Delayable Workloads<br>3b. Delayable / Planned Outages<br>4. Enhanced<br>5. Unprotected | 1. Perfection<br>2. Complete Detection<br>3. Enhanced<br>4. Unprotected | 1. Perfection<br>2. Resume with Delay<br>3. DB Preservation<br>4. Unprotected | 1. Perfection<br>2. Less<br>3. Lesser<br>4. Unprotected |
| **Message Handling** | Classes 1 to n | Classes 1 to n | Classes 1 to n | Classes 1 to n |
| **Process Control** | Classes 1 to n | Classes 1 to n | Classes 1 to n | Classes 1 to n |
| **Search and Retrieval** | Classes 1 to n | Classes 1 to n | Classes 1 to n | Classes 1 to n |
| **Analytical Calculation** | Classes 1 to n | Classes 1 to n | Classes 1 to n | Classes 1 to n |

**Table 2: Classification Structure**

| Class | Basic Requirements |
|---|---|
| **1. Perfection** | • The system must be able to correctly process every transaction submitted to it, within normal response times, all of the time.<br>• All single failures, corrective actions, maintenance, and other potentially disruptive events are handled by the system without any noticeable effect on the users. |
| **2a. Retryable Workload** | • The system must be able to correctly process every transaction submitted to it, either within normal response times, or after a brief delay to recover from a disruptive event.<br>• Disruptive events may not cause users to have to re-establish connection to the system nor to suspend submitting transactions.<br>• No transactions may be lost nor may the database be left with inconsistent data due to incomplete transactions.<br>• The system may request that incomplete transactions be re-submitted after recovering from an event, but the number of such transaction may not exceed 1 second worth of Normal Transaction Processing Capacity.<br>• The Recovery Period for disruptive events may not exceed 10 minutes. Average capacity during the period may not be below 80% of Normal Transaction Processing Capacity. |
| **2b. Retryable / Planned Outages** | • Same as 2a, but the system may rely on taking an outage to perform certain planned operations, maintenance, repair, and upgrade tasks. |
| **3a. Delayable Workloads** | • The system must be able to respond to disruptive events and be ready to process transactions within 5 minutes.<br>• The recovery period for disruptive events may not exceed 25 minutes. Average capacity during the recovery period may not be below 60% of Normal TP Capacity.<br>• No transactions may be lost, nor may the database be left with inconsistent data due to incomplete transactions.<br>• The users may be required to re-establish connection to the system and to identify and re-submit transactions that did not complete before the event. The number of such transactions may not exceed 1 second worth of Normal Transaction Processing Capacity. |
| **3b. Delayable / Planned Outages** | • Same as 3a, but the system may rely on taking an outage to perform certain planned operations, maintenance, repair, and upgrade tasks. |
| **4. Enhanced** | • The user can evaluate, for cost and effectiveness, the individual features that are intended to improve availability.<br>• Disruptive events may result in a total outage of a system (requiring user intervention to perform reboot and recovery).<br>• No transactions may be lost, nor may the database be left with inconsistent data due to incomplete transactions. |
| **5. Unprotected** | • None. |

**Table 3: Basic Requirements for Transaction Processing Availability Classes**

## 4. Example: Availability for Transaction Processing

The authors have been developing the details of the proposal for Transaction Processing applications.

A discussion of the classes defined for Availability will illustrate how the classification system will function and suggest the work that still needs to be done for other application types and dimensions.

Systems are assigned to one of five *Classes* by meeting *Basic Requirements* over a set of *Factors* that affect availability. The Classes and Basic Requirements are shown in Table 3. The Factors are shown in Table 4.

To qualify for a specific rating, the system is evaluated against a list of criteria for each availability factor. An example of the evaluation criteria, for the Hardware Failure Factor, is shown in Table 5.

The criteria vary widely in scope, so each will need to be evaluated by its own appropriate method, e.g. a standard benchmark, a design audit, or analysis of field data. Since there are very many criteria to be satisfied, it is proposed that systems may still be evaluated for a given class, even if their designs do not meet all of the criteria, as long as all exceptions are disclosed.

One of the drawbacks of standardized performance benchmarks is the high cost of conducting and certifying a test run. Since a comprehensive dependability benchmark would be more complex, the implementation cost would be even more discouraging and the authors believe that vendors would not bear it.

| Factor | Description |
|---|---|
| **HW Failure** | Intermittent or permanent HW fault, including design errors that manifest as component failure. |
| **SW Failure** | Improperly designed, built, or installed software that results in a detected failure that takes resources out of use. |
| **Environmental Failure** | A failure to provide power, cooling or other environmental requirement of the system. |
| **HW Repair or Upgrade** | Repair failed components, re-integrate repaired components, or install newer version (same form, fit, function). |
| **SW Repair or Upgrade** | Action to replace a faulty component, or install newer versions (same external interfaces); or to revert to previous versions. |
| **Operating Configuration Change** | Action to adjust system parameters for performance tuning, policy administration, access control, etc. |
| **System Maintenance** | Action required to maintain the integrity of the application, e.g. data backups, log dumps, resource monitoring. |
| **Capacity Expansion (or Reduction)** | Scaling the system for changes in volume of usage, e.g. additional HW, new SW, database reorganization. |
| **System Management Skills** | The level of skills, training, process control, and other human factors required to obtain the desired availability. |
| **Denial of Service Attack** | Attempt by un-authorized users to render the system inaccessible or unusable |

**Table 4: Factors that Affect Availability**

The intention of this classification system is that the evaluation would be self-certified. Vendors or others wishing to classify a particular system would do the evaluation and publish the results, answering a set of standardized questions or performing tests that validate the evaluation criteria. This approach rests on the assumption that vendors would not risk their reputations, nor product liability claims, by making false statements against very specific standards.

## 5. Progress on Specifying the Classification System

To date, initial drafts have been written for Availability, Data Integrity, and Disaster Recovery for Transaction Processing applications. These drafts include classes, factors, minimum standards and evaluation criteria. Evaluation methods have not yet been proposed. A sub-committee of SIGDeB is currently doing a trial evaluation of a specific system to see if a standardized set of questions or evaluation tests can be developed.

| Classes | Perfection | Retryable Workloads | Delayable Workloads | Reduced Impact of Failure |
|---|---|---|---|---|
| **Minimum Standard** | • No single HW failure may cause a properly configured system to violate the Basic Requirement. | | | |
| **Required Disclosures** | • What, if anything, is required in the application code or configuration to meet the standards.<br>• Any cases where the system might lose its ability to recover from a HW failure, but not report this condition to the user (e.g., a backup resource fails, but the system does not detect the failure until it attempts to use the resource in a recovery action).<br>• History of user-reported HW defects that violated minimum standards.<br>• Any exceptions (to the minimum standards) in the system design, with quantified impacts. | | | • Cost / benefit of features to be evaluated |
| **Comparative Measurements** | • How long is the system susceptible to a second (unprotected) failure, i.e., while the first failure is detected and repaired?<br>• Frequency of failures that cause a recovery process or remove a resource from the system.<br>• Duration and impact of the recovery process. | | | • Frequency of failures that benefit from each feature |

**Table 5: Evaluation Criteria for Hardware Failure**

## References


[1] D. P. Siewiorek, J. J. Hudak, B.-H. Suh and Z. Segall, "Development of a Benchmark to Measure System Robustness", in *Proc. 23rd Int. Symp. on Fault-Tolerant Computing (FTCS-23)*, Toulouse, France, 1993, pp. 88-97 (IEEE CS Press).

[2] T. K. Tsai, R. K. Iyer and D. Jewitt, "An Approach Towards Benchmarking of Fault-Tolerant Commercial Systems", in *Proc. 26th Int. Symp. on Fault-Tolerant Computing (FTCS-26)*, Sendai, Japan, 1996, pp. 314-323 (IEEE CS Press).

[3] P. Koopman and J. DeVale, "Comparing the Robustness of POSIX Operating Systems", in *Proc. 29th Int. Symp. on Fault-Tolerant Computing (FTCS-29)*, Madison, WI, USA, 1999, pp. 30-37 (IEEE CS Press).

[4] A. Brown and D. A. Patterson, "Towards Availability Benchmarks: A Cases Study of Software RAID Systems", in *Proc. 2000 USENIX Annual Technical Conference*, San Diego, CA, USA, 2000 (USENIX Association).

[5] J. Arlat, J.-C. Fabre, M. Rodríguez and F. Salles, "Dependability of COTS Microkernel-Based Systems", *IEEE Trans. on Computers*, vol. 51, no. 2, pp. 138-163, February 2002.

[6] J. V. Carreira, D. Costa and J. G. Silva, "Fault Injection Spot-checks Computer System Dependability", *IEEE Spectrum*, vol. 36, pp. 50-55, August 1999.

[7] R. Chillarege and N. S. Bowen, "Understanding Large System Failures — A Fault Injection Experiment", in *Proc. 19th Int. Symp. on Fault-Tolerant Computing (FTCS-19)*, Chicago, IL, USA, 1989, pp. 356-363 (IEEE CS Press).

[8] A. M. Amendola, L. Impagliazzo, P. Marmo and F. Poli, "Experimental Evaluation of Computer-Based Railway Control Systems", in *Proc. 27th Int. Conf. on Fault-Tolerant Computing Systems (FTCS-27)*, Seattle, WA, USA, 1997, pp. 380-384 (IEEE CS Press).

[9] C. Constantinescu, "Validation of the Fault/Error Handling Mechanisms of the Teraflops Supercomputer", in *Proc. 28th Int. Symp. on Fault-Tolerant Computing (FTCS-28)*, Munich, Germany, 1998, pp. 382-389 (IEEE CS Press).

[10] H. Madeira, R. Some, F. Moreira, D. Costa and D. Rennels, "Experimental Evaluation of a COTS System for Space Applications", in *Proc. Int. Conference on Dependable Systems and Networks (DSN-2002)*, Washington, DC, USA, 2002 (IEEE CS Press).

[11] J. Arlat, A. Costes, Y. Crouzet, J.-C. Laprie and D. Powell, "Fault Injection and Dependability Evaluation of Fault-Tolerant Systems", *IEEE Transactions on Computers*, vol. 42, no. 8., pp. 919-923, August 1993.

[12] J.-C. Laprie, Ed., "Dependability: Basic Concepts and Terminology", (Dependable Computing and Fault Tolerance, vol. 5, A. Avizienis, H. Kopetz and J.-C. Laprie, Eds.), *Vienna: SpringerVerlag*, 1992.

[13] M. Hsueh, T. Tsai and R. K. Iyer "Fault Injection Techniques and Tools", *Computer*, vol. 30, no. 4, pp. 75-82, April 1997.

[14] J. Clark and D. Pradhan, "Fault Injection: A Method for Validating Computer-System Dependability", *Computer*, vol. 28, no. 6, pp. 47-56, June 1995.