

Issues in Defining, Analyzing, Refining, and Specifying System Dependability Requirements

Bonnie Melhart
Texas Christian University
email: b.melhart@tcu.edu

Stephanie White
Long Island University
email: Stephanie.White@liu.edu

Abstract

Requirements specification has long been acknowledged as an area for further research in the development of systems, particularly for those that are computer based. In addition, an IEEE White Paper by Tripp and Keane indicates that current practice regarding System Dependability is in large part based on “tricks of the trade”, and that there is a need to codify “best practices”. This paper attempts to codify the practice for specifying dependability requirements, by classifying and describing the process and product requirements needed for such specification.

1. Introduction

A dependable system is one that does not fail in an unexpected or catastrophic way. This simple, intuitive definition belies the difficulty with specifying exactly what properties such a system must have and how these can be ensured. Often the dependability of the ultimate system is determined by the quality of the analysis performed at the requirements stage. To improve requirements engineering, a taxonomy was developed for defining, analyzing, refining, and specifying requirements [1]. The taxonomy consists of a structure, definitions, templates, examples, and keywords for requirements retrieval. The purpose of the taxonomy is to increase the integrity of requirements: their consistency, completeness, and correctness. This paper extends the taxonomy, which heretofore addressed only functional requirements, to include dependability requirements¹.

Dependability is a system property. Acceptable thresholds for system dependability are defined in the requirements (for example, acceptable mean time to repair). So is the required functionality (for example,

recovery functions) that ensures the desired dependability in the final product. Process requirements are also specified, either in the requirements document, the statement of work, or a dependability plan. Such a plan addresses the use of dependability techniques throughout system development, such as (1) fault removal, (2) data collection, and (3) measuring the level of dependability. Developers must collect data for such measurements throughout the system life cycle. For example, early predictions of the number of faults that remain in the system are based on data about fault removal during development. Such predictions allow systems engineering tradeoffs concerning technologies for later development.

Required dependability levels are actually measured near the end of or after development. That is a primary goal of testing, which consumes a large part of the development and maintenance budget. When the system is not dependable, test, repair and retest become exorbitant. The direct costs of software unreliability, in particular, are substantial, let alone the indirect costs [2]. Information that defines what should be measured is defined in the requirements. Thus, if requirements specification is improved, costs should be significantly reduced.

2. Dependability Terms

Many terms are related to or a part of dependability discussions. Some definitions are given here.

Dependability measures the degree to which a system is operable at any random time during a specified mission profile, given that its services are available at the start of the mission [3]. Tripp and Keane have called this “fitness for use” [4]. Dependability is not one measure, but a collection of related measures that includes reliability and availability, and often safety and security. It may include maintainability, usability, and extendibility [5].

¹ Research was funded by Naval Surface Warfare Center and the Office of Naval Research.

Survivability from *information warfare*² is a recent addition to the list of dependability measures. Table 1 identifies facets commonly associated with the most commonly used dependability measures.

Table 1. Dependability measures and related facets

Reliability	Availability	Security	Safety	Survivability
MTBF	MTTR	Information integrity	Unsafe states	Vulnerability
Probability of failure	Probability of being operational	Confidentiality	Fail safe	Protection
Probability of Success	Down time	Authentication	Life critical	Intrusion detection
Zero defects	Maintenance	Access Control	Property critical	React to attack

Reliability, R, is the probability that the system performs its required functions under specified environmental conditions for a specified period of time, t [2]³. Reliability is concerned with departures of any kind from the specifically required service. It is assumed that the system is functioning at the start of the time interval, $R(0) = 1$, and that the system will fail sometime, $R(\infty) = 0$. The expected environmental conditions may be specified in a mission or *operational profile*.

Availability addresses the ability of the system to provide service at any given time. It is the probability of being operational, under given use conditions, at a given instant in time [5].

Safety is the ability of a system to deliver service under given conditions with no catastrophic affect. This absolute definition implies no “grace” period or margin for safety; the requirements specify how close the system must come to this ideal [7]. Safety attributes mean the system will include requirements to detect and avoid catastrophic failures. *Security* is the ability of a system to deliver its required service under given conditions for a given time without unauthorized disclosure or alteration of sensitive information, or denial of service to legitimate users by unauthorized persons [5]. The security attribute adds requirements to the system to detect and avoid intentional faults. *Survivability* is the capability of a system to accomplish its mission despite a man-made hostile environment; i.e., the power of the system to detect and withstand an attack. Survivability of Computer Based Systems (CBSs) is gaining attention because of recent

attacks on computers and communications channels and concern about Information Warfare. Though often related to dependability, these last three measures are not directly included in the dependability discussion here. A complete treatment of requirements for these three attributes is beyond the scope of this work.

Measures used for dependability incorporate the terms, fault, error, and failure. From a systems perspective, a *fault* is a condition that is thought to be the cause of an error. An *error* is that part of a system state which is liable to lead to a failure for the system. A *failure* is any deviation from the specified behavior [8].

The *Mean Time to Failure (MTTF)* measure is frequently used for system reliability. The MTTF along with the maintenance time, Mean Time to Repair (MTTR) can be used to calculate availability as

$$\frac{MTTF}{MTTF + MTTR}$$

Note that this simple model makes assumptions about shut down and constant failure and repair rates that may be inappropriate. In particular, software components usually have negligible *MTTR*, since repairs do not require the program to be idle unless the failure is critical to life or property. For software systems, the *MTTR* may be weighted by the failure intensity for a more realistic and usable measure of availability.

Failure intensity measures the frequency at which failures occur. It is the number of failures per some unit of measure [9,10]. The measurement unit is usually execution time, but may be something else like the number of transactions, number of runs, etc. This failure rate must address all system failures: those due to primary defects, manufacturing defects, operator errors, wear-out, along with dependent failure, maintenance induced failure, and equipment damage rates. The number of faults in the system and the evaluation environment affect failure intensity; i.e., the measurement must reflect proper use so it gives a fair indication of the actual reliability of the system. The failure intensity, i , at time, t , is related to the reliability: $R(t) = e^{-it}$.

Robustness and fault-tolerance are important properties that affect system dependability. *Robustness* implies that frequently occurring errors have little effect on the system. A *fault-tolerant* system is desirable when the system must recognize a potential failure and take action to avoid it. This implies at least some redundancy built into the system to check results [2].

3. Types of Dependability Requirements

Dependability requirements address measurement and accompanying performance thresholds for the *product*, and the *process* by which those measurements are accomplished. It is important to specify what process will be used to increase confidence in the product, including

² Panda and Giordano define *Information Warfare* to be any malicious attack against an organization’s information base through electronic means [6].

³ As pointed out by Mellor, this traditional definition excludes the class of “failures” in which the specification is wrong or does not exist [5].

the collection of data, and then to collect relevant data throughout development. The latter enhances the ability to assess dependability in the product.

There are four types of dependability requirements, which address the means for achieving dependability: fault tolerance, fault removal, fault prevention, and fault prediction [8]. Their combined utilization provides the desired coverage of dependability concerns for both the product and the development process. The objectives and means for these are often part of a reliability program plan. Table 2 identifies facets associated with these four categories of dependability requirements. We address fault tolerance in the section on Product Requirements, and we address fault prediction, fault prevention, and fault removal in the section on the Dependability Process.

Table 2. Four categories of dependability requirements, and related facets

Fault Prediction	Fault Prevention	Fault Tolerance	Fault Removal
FMEA/FMECA	Models	Redundancy	Fault detection
Fault tree analysis	Prototypes	Failure detection	Inspection
Hazard analysis	Simulations	Failure avoidance	Reviews
Models	Information hiding	Fault recovery	Walkthroughs
Measures	Low coupling	Graceful degradation	Failure reporting
Tolerable threshold	High cohesion	Reconfiguration	Proof of correctness
Fault/Error/Failure	Reuse	Rollback	Dynamic analyses
Data Collection	Traceability	N-version software	Test/Debug/Repair
Operational Profile	Formal verification	Hot shadow	Latent faults

4. Dependability Process

Process requirements describe how dependability must be achieved, and include the means for data collection and analysis to verify its achievement.

4.1 Fault Removal

Fault removal requirements are process requirements. Such requirements imply the need for fault detection. In addition to dynamic analyses of the system such as testing, the required process should include static techniques such as inspections, reviews, walkthroughs, proofs-of-correctness.

4.2 Fault Prevention

Fault prevention requirements are also process requirements. In a sense, every good development practice is a fault prevention technique. A constructive approach to developing a dependable system dictates that it be designed and built according to a more dependable development process. All the process requirements have fault prevention as a goal or sub-goal.

4.3 Fault Prediction

Fault prediction implies process requirements for data collection so that adequate information is available, and product requirements regarding level of achievement. The information to measure the dependability and thus predict numbers of failures in the future is based on failures and their history of occurrence. Requirements indicate what quantitative measurements will be taken (availability, reliability, etc.); the specific data needed is determined by the chosen model used for evaluation. To measure reliability the number of observed failures over a measured period of time, usually execution time (i.e., actual time that the system is operational) must be recorded.

Failure modes and effects analysis or criticality analysis (FMEA/FMECA), fault-tree analysis, and other techniques may be used to determine the relevant failure events. Analysis includes determining what faults may arise in dependent components due to failures of another component, and what faults within a component may lead to an eventual error state and failure of that component itself. For example, FMEA identifies the ways in which failures can occur, the effects of each failure on the system element in which it occurs, probable overall consequences (severity) of each failure mode on the success of the system and safety of using personnel, and the recommended corrective actions/procedures to reduce the occurrence of the potential failure mode.

Dependability objectives for the given time period of operation must be specified. The operational profile that will be used to make the measurement must be described. If the complete operational profile is not established at the time of the requirements specification, a process requirement to develop one must be included. Musa, in fact, suggests a series of profiles leading to the final operational profile: Customer profile, Use profile, System-mode profile, Functional profile, and Operational profile [11].

Early profiles may be developed as part of the requirements specification, with a requirement to complete the final profile during development. A Customer Profile specifies all the groups that will use the system, along with the proportion of use they represent. The Use Profile refines the previous profile into groups of users, where users in the same group employ the system in a similar

way. The ratio of a group's users to total users can be used as the probability, or a proportion of actual use can be obtained. Section 5 has more to say concerning operational profiles.

5. Product Dependability Requirements

A product requirement is a statement in some notation that describes one or more characteristics or properties a system must have. Product requirements are normally classified as capabilities or constraints. "It is common to sub-classify constraints as, for example, performance, security, reliability, maintainability, etc. requirements" [12]. To fully describe a system's dependability constraints, one must include dependability objectives for the system and its components, operational profiles for the above measurement, specification of failure modes, and requirements addressing fault tolerance, including redundant component specifications and recovery techniques to be employed.

5.1 Dependability Objectives

The product dependability objective is the threshold of acceptability and is determined from the desire for reliability, safety or other dependability attribute, the urgency with which the product is needed, and the price the customer is willing to pay. These requirements are initially defined as part of the conceptual design, which is usually an outcome of advanced development. Dependability/reliability objectives for the given time period of operation must be specified. The operational profile that will be used to make the measurement must be described.

5.2 Operational Profiles

Dependability of the product is measured during operation in a defined environment. So the dependability requirements must not only specify the measurement process and tolerable threshold, but also state the expected operational conditions. These conditions include environmental factors (e.g. temperature cycles, humidity, vibration, shock, sand, heat), geographical location, and expected interaction with external systems and humans. Operational profiles should address times the system is operating, and times it is stored or being transported. These latter conditions are sometimes more critical than the operating conditions [9].

The operational profile is a tool for specifying the use environment. It is a quantified list of the elements and factors related to expected product use, and how often they will occur. It should be developed through communication between the customer and the developers. The operational profile provides information about which

capabilities will be used the most, and which capabilities will be needed first if the system will be delivered in stages.

5.3 System Modes

System modes may be determined by partitioning the system input space into equivalence classes based on some condition(s), such as system traffic level, predominant function, user experience, etc. A system may have a fail-safe mode, for example, of reduced or basic functionality that has a very different profile from the normal mode of operation. Modes allow us to consider fewer possibilities than considering every function. Also, using modes is more appropriate for the requirements phase since we may not know all the functions. The System Mode Profile shows the modes and their proportional occurrence.

These profiles may be used to predict initial dependability levels for the system and to develop tests later on. They also assist in assuring the completeness of the reliability requirements as they increase communication opportunities for customers and developers.

5.4 Failure Modes

Functional requirements are a "positive" specification of what the system must do. A specification of what it must not do, a "negative" specification, is also important, and should include information concerning failure modes for the system. Failures are defined from the product user perspective (e.g. loss or degraded capability of a function or feature) and grouped into categories by cost/impact on the product user. Each of these categories affects the user to the same degree and is called a severity class. Some systems, fault tolerant systems for example, will need to detect and count run-time failures in distinct categories.

5.5 Fault Tolerance

Fault tolerance requirements imply the system must include redundancy for error detection and/or it must avoid the consequent failure state. That is, the fault at issue shall not become active. Systems that include such safety, security, and survivability requirements specify the states and conditions that the system must detect and avoid. It is usually necessary to include recovery techniques, though specific components will employ different means for this. Example recovery alternatives are reconfiguration with a new component, and reduced functionality to avoid exercising the fault in the current component (graceful degradation). Reconfiguration is generally supported with redundant computers. The system design may employ extra general-purpose computers, or each computer may have a "Hot Standby" which shadows the active computer and executes the same

software. The operating system must support system requirements for redundancy and hardware reconfiguration. Requirements for error recovery may include exception handling and recovery blocks in the application software.

An alternative to error recovery is fault masking (also called error compensation). Classically this is achieved by modular redundancy, in which several components perform each computation independently, and the final result is selected by majority voting [13].

Fault tolerance affects reliability as it reduces the likelihood that faults will be manifested as failures and therefore also affects the availability at a specific time. It may be used to increase safety and security attributes if their relevant classes of failures are addressed. System failures that may result in loss of human life or critical property are a special concern. With fault tolerance, the system may be able to withstand a sequence of failures and fail in a safe state (e.g. fail operational, on first failure, fail operational on second failure, fail in a safe state on the third failure).

5.6 Software and Hardware Differences Affect Measurement

Dependability attributes are emergent properties of the system. That is, they are a measure of the system as a whole and not just a sum of measures for the component parts. Hardware and software dependability must be managed as a system measure. However, there are recognized differences between these characteristic subsystems. The differences affect the overall evaluation of the integrated system. The American Institute of Aeronautics and Astronautics has summarized the important hardware to software contrasts.

First, changes to hardware require time consuming steps including fabrication and assembly, so requirements normally call for replacement or repair, rather than change. Repairs generally restore the hardware to its previous state. Changing software is frequently more feasible, though not always effective. Therefore, maintainability requirements address modularity and other good design techniques that support fixing “bugs” and improving the software. Changing the software always changes the software state. A high rate of change is likely to decrease the software reliability.

Software developments have traditionally made little use of existing components; hardware is often manufactured from standard parts, and requirements call for use of common parts, which supports maintenance. Manufacturing tolerances affect hardware, and requirements call for specific tolerances. Exact copies of software can be easily produced, so tolerance is not an issue. However computation accuracy is important.

Software does not experience physical wear-out. Rather, failures from software faults come without

advance warning, often with little indication of their occurrence. Hardware on the other hand, often provides a period of graceful degradation and is susceptible to wear-out. Both hardware and software reliability measures are in operational time. For hardware, this is expressed in wall-clock time. For software, execution time is used [14].

5.7 Example: Air Traffic Control System

The following example illustrates the concepts of dependability objective, operational profile, system mode, failure mode, and fault tolerance.

Assume that the customer for an Air Traffic Control (ATC) system requires that the system be “down” at most six hours per year. This is an availability requirement that requires very high reliability and very fast repair. The customer defines operational profiles that include aircraft management, and limited time on duty for controllers due to fatigue and stress. Controllers manage aircraft flying through their sector, taking off, and landing. When runways are not available for landing, controllers assign the aircraft a flight pattern and altitude, and may bring the aircraft to successively lower altitudes prior to landing. Eventually they assign aircraft a runway, manage aircraft as they approach the runway and land, and assign aircraft a gate. Such high level use profiles (or scenarios) are further detailed and are used with other profiles to design the system and verify that it meets the dependability objectives. The customer and users also define failure modes, such as (1) one (or more) radars out of operation, (2) understaffed operation, and (3) insufficient computer resources. System and subsystem designers use the requirements, operational profiles and mode definition, and working together, they identify potential designs, perform tradeoffs, and select the best design. They define system modes, which may include (1) test modes, and (2) operational modes. The latter may include (2a) sector control (controls aircraft flying through the sector); (2b) approach control (controls multiple aircraft landing on multiple runways; and (2c) aircraft emergency mode (e.g. aircraft must land immediately due to passenger illness or aircraft failure). System modes should be consistent with operational profiles, for user understanding and maintainability.

Based on the system architecture, contractors and subcontractors allocate requirements for reliability (MTTF) and repair (MTTR), or allocate requirements for failure intensity, as appropriate, to subsystems. They then define subsystem requirements, including those for humans, communication links, hardware, and infrastructure and application software. Subsystems must operate in unison to meet the ATC system objective of at most six hours down time, so a fault tolerant architecture must be part of the overall system architecture. Allocated fault tolerance requirements for the communication

subsystem may include redundant broadcast on multiple channels, data backup, and rebroadcast when data is corrupted. There are a number of basic concepts for fault tolerant computing that could be used in the ATC system. All may be used. The fault tolerance requirements for the hardware normally specify the need for redundant computers.

6. Summary

Dependability requirements are related to both process and product. The product part is concerned with specified quantitative measures related to functions, behavior, and data. These measures may differ by scenario, mode, or operational profile. The process part specifies required practices that the contractor shall perform during the system life cycle, to ensure that dependability requirements will be met and can be verified.

The requirements specification may require a dependability program plan as part of the system engineering management plan. A typical reliability plan should address these product requirements issues:

- Quantified dependability requirements for the system as related to system operational requirements and operating conditions, including the operational profile and the maintenance concept.
- Allocation or apportionment of dependability requirements to the subsystem components, hardware or software.
- Effects of maintenance.

It should address these process requirements issues:

- Failure Modes and Effects (Criticality) Analysis (FMEA/FMECA).
- Design techniques and practices.
- Formal reviews.
- Analysis of behavioral and mathematical models, worst case analysis, vulnerability analysis.
- Data collection, analysis and corrective actions.
- Dependability prediction and assessment.
- Dependability testing and other evaluations.

7. Future Work

The following open problems are recommended as research topics to improve specification of dependability requirements.

- Examination of requirements documents and plans for ultra-reliable systems may provide improved insight for ultra-reliable and normally reliable systems, as concepts used for ultra-reliable systems can be applied in a relaxed form to systems that are not safety or mission critical.
- Corrective changes result in dependability growth; all types of maintenance (change) can lead to

dependability decay. Can change/maintainability be specified at the requirements stage, and how does this impact the requirements for dependability?

- Testability is important and has an affect on dependability [15]. Is it possible to require that hardware and software be more controllable or observable? How are requirements for this stated?
- Different information is available and recorded for defects discovered at different phases of development. What are appropriate lists of this information for each phase?

8. References

- [1] White, S., *A Faceted Structure for Capturing and Analyzing System Requirements*, Report for Engineering of Complex Systems Technology Project, Naval Surface Warfare Center, Dahlgren Division. Research performed under contract N60921-94-C-0073, February 1997.
- [2] Kopetz, H., *Software Reliability*, Springer-Verlag New York Inc., 1976.
- [3] Department of Defense, Glossary, Defense Acquisition Acronyms and Terms, Fifth Edition, Defense Systems Management College, Fort Belvoir, Virginia, Sept. 1991.
- [4] Tripp, L. and S. Keene, *System Dependability White Paper*, unpublished paper, sponsored by IEEE Computer Society and IEEE Reliability Society, 1999.
- [5] Mellor, P., "Failures, Faults and Changes in Dependability Measurement," *Information and Software Technology*, **34** (10), October 1992, 640-654.
- [6] Panda, B. and J. Giordano, "Defensive Information Warfare, Communications of the ACM, July 1999, 31-32.
- [7] Leveson, N. G., *Safeware: System Safety and Computers*, Addison-Wesley, Reading, MA, 1995.
- [8] Laprie, J. (ed.), *Dependability: Basic Concepts and Terminology*, Springer-Verlag New York, 1992.
- [9] Blanchard, B.S. and W. Fabrycky, *Systems Engineering and Analysis*, Prentice Hall, Englewood Cliffs, N.J., 1990.
- [10] Musa, J., A. Iannino, and K. Okumoto, *Software Reliability: Measurement, Prediction, Application*, McGraw-Hill Book Company, New York, 1987.
- [11] Musa, J., "Operational Profiles in Software-Reliability Engineering," *IEEE Software*, **10**(2), March 1993, 14-32.
- [12] Sawyer, P. and G. Katonya, *SWEBOK: Software Requirements Engineering Knowledge Area Description*, Version 0.5, IEEE and ACM Project on Software Engineering Body of Knowledge, July 1999.

[13] Rushby, J., *Critical System Properties: Survey and Taxonomy*, Computer Science Laboratory Technical Report SRI-CSL-93-01 written for NASA Langley Research Center under Contract NAS1-18969, SRI International, May 1993.

[14] AIAA Space-based Observation Systems Committee on Standards, Software Reliability Working group, "Recommended Practice for Software Reliability" R-013-1992, Draft, April 1992.

[15] Kapur R., T. Williams, and E. Miller, "System Test and Reliability: Techniques for Avoiding Failure," *IEEE Computer*, November 1996, 28-30.