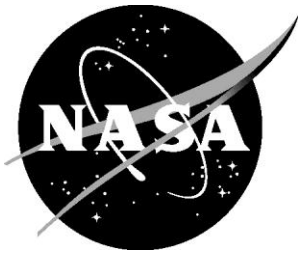
NASA/TM–2013-217798



An Approach for the Assessment of System Upset Resilience

Wilfredo Torres-Pomales
Langley Research Center, Hampton, Virginia

January 2013

NASA STI Program . . . in Profile

Since its founding, NASA has been dedicated to the advancement of aeronautics and space science. The NASA scientific and technical information (STI) program plays a key part in helping NASA maintain this important role.

The NASA STI program operates under the auspices of the Agency Chief Information Officer. It collects, organizes, provides for archiving, and disseminates NASA's STI. The NASA STI program provides access to the NASA Aeronautics and Space Database and its public interface, the NASA Technical Report Server, thus providing one of the largest collections of aeronautical and space science STI in the world. Results are published in both non-NASA channels and by NASA in the NASA STI Report Series, which includes the following report types:

- TECHNICAL PUBLICATION. Reports of completed research or a major significant phase of research that present the results of NASA Programs and include extensive data or theoretical analysis. Includes compilations of significant scientific and technical data and information deemed to be of continuing reference value. NASA counterpart of peer-reviewed formal professional papers, but having less stringent limitations on manuscript length and extent of graphic presentations.

- TECHNICAL MEMORANDUM. Scientific and technical findings that are preliminary or of specialized interest, e.g., quick release reports, working papers, and bibliographies that contain minimal annotation. Does not contain extensive analysis.

- CONTRACTOR REPORT. Scientific and technical findings by NASA-sponsored contractors and grantees.

- CONFERENCE PUBLICATION. Collected papers from scientific and technical conferences, symposia, seminars, or other meetings sponsored or co-sponsored by NASA.

- SPECIAL PUBLICATION. Scientific, technical, or historical information from NASA programs, projects, and missions, often concerned with subjects having substantial public interest.

- TECHNICAL TRANSLATION. English-language translations of foreign scientific and technical material pertinent to NASA's mission.

Specialized services also include organizing and publishing research results, distributing specialized research announcements and feeds, providing information desk and personal search support, and enabling data exchange services.

For more information about the NASA STI program, see the following:

- Access the NASA STI program home page at *http://www.sti.nasa.gov*

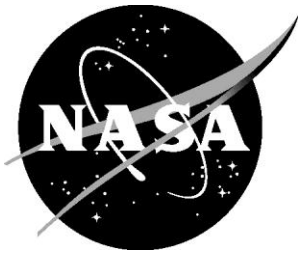
- E-mail your question to help@sti.nasa.gov

- Fax your question to the NASA STI Information Desk at 443-757-5803

- Phone the NASA STI Information Desk at 443-757-5802

- Write to:
  STI Information Desk
  NASA Center for AeroSpace Information
  7115 Standard Drive
  Hanover, MD 21076-1320

NASA/TM–2013-217798



An Approach for the Assessment of System Upset Resilience

Wilfredo Torres-Pomales
Langley Research Center, Hampton, Virginia

National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23681-2199

January 2013

Acknowledgment

The author is grateful to Paul S. Miner, Anthony Narkawicz, Daniel M. Koppen and Kevin M. Somervill for their comments and suggestions about the ideas presented in this report.

Available from:

NASA Center for AeroSpace Information
7115 Standard Drive
Hanover, MD 21076-1320
443-757-5802

Abstract

This report describes an approach for the assessement of upset resilience that is applicable to systems in general, including safety-critical, real-time systems. For this work, resilience is defined as the ability to preserve and restore service availability and integrity under stated conditions of configuration, functional inputs and environmental conditions. To enable a quantitative approach, we define novel system service degradation metrics and propose a new mathematical definition of resilience. These behavioral-level metrics are based on the fundamental service classification criteria of correctness, detectability, symmetry and persistence. This approach consists of a Monte-Carlo-based stimulus injection experiment, on a physical implementation or an error-propagation model of a system, to generate a system response set that can be characterized in terms of dimensional error metrics and integrated to form an overall measure of resilience. We expect this approach to be helpful in gaining insight into the error containment and repair capabilities of systems for a wide range of conditions.

Table of Contents

1. Introduction

A research effort is underway to develop practical validation and verification (V&V) methods that can enable rigorous safety assurance for the next generation of aviation systems. These systems are characterized by highly complex, large-scale, network-based distributed architectures with software-implemented functionality and advanced computation and communication capabilities. To meet the safety goals, these systems must be demonstrably robust with respect to system design and implementation errors, component degradations and failures, and partial system failures. The V&V challenge is compounded by strong coupling of system components in the software and the hardware, as well as the need to consider unexpected and possibly malicious component behaviors [26].

To support this research effort, verification approaches are being developed for robust distributed algorithms that support system redundancy management in a fault space with a wide range of severity. A system architecture for safety-critical real-time applications must have the ability to mitigate the effects of internal component faults of varying severity [18]. A safety-critical system must have sufficient design fault tolerance to accommodate the more frequent uncorrelated random faults without malfunctions at the system services. A robust system must also mitigate infrequent but more severe correlated faults that can exceed the system design assumptions, disrupt internal coordinated operation among the system components and propagate effects outward to the external service interfaces. Analysis techniques will be developed for system designs intended to ensure continued safe operation in the presence of component misbehavior while simultaneously minimizing their adverse effects. These techniques should enable designs with strongly assured safety properties under the weakest possible (i.e., least restrictive) assumptions in terms of the number and types of faults a system can handle.

In this research context, a physical fault injection experiment was conducted in which a prototype implementation of an onboard data network for distributed safety-critical, real-time Integrated Modular Architectures (IMA) was exposed to a High Intensity Radiated Field (HIRF) environment in a mode-stirred electromagnetic reverberation chamber [54, 87, 88, 90, 91]. The purpose of the experiment was to gain insight into the response of the system to a wide range of internal faults, including conditions that exceed the design safety margins. There is special interest in examining the response to functional **upsets**, which are error modes that involve no permanent component damage, can simultaneously occur in multiple channels of a redundant distributed system and can cause unrecoverable distributed state error conditions [9, 29, 38].

The fault injection experiment was divided in two parts. The HIRF Susceptibility Threshold Characterization (**HSTC**) experiment was intended to identify and examine factors that determine the measured minimum HIRF field strength level at which a particular electronic System Under Test (SUT) begins to experience HIRF-induced interference to its internal operation (i.e., faults). The results and lessons learned in the execution of the HSTC experiment are described in report [88]. The HIRF Effects Characterization (**HEC**) experiment was intended to assess the system response to functional system upsets. Different system configurations were tested with variations on the communication data rate, the degree of redundancy, and the number of simultaneously irradiated components. The objective was to characterize the effect of a HIRF environment on the behavior of the system and its components. The characterization will consider the effects at the external system interfaces and at the interfaces of internal components. Of special interest is determining the severity of component faults and assessing the robustness of the system to multiple simultaneous faults. We would like to identify weaknesses in the design of the system and desirable features for more robust communication systems. The test results are expected to contribute to the development of redundancy management mechanisms and policies for robust processing architectures.

This report describes the approach to assess the HEC-experiment fault effects at the interfaces of the system and its components. We also expect that the approach will help us gain insight into the relation between the severity of internal faults and the propagated effects. However, a thorough understanding of that relation is outside the scope of this assessment approach and will be the subject of future work using error-propagation system models to perform simulated fault injection experiments.

The characterization of fault effects is based on the concept of **resilience**. In [55], Laprie defines resilience as "the persistence of service delivery that can justifiably be trusted, when facing changes." In [59], Leveson states that resilience is often defined as "the ability to continue operations or recover stable state after a major mishap or event". We are interested in an objective and quantitative characterization of fault effects. For this, we define a set of metrics to measure various dimensions of error manifestations. We also define composite metrics that integrate the error dimensions. These metrics are described in detail in subsequent sections.

This report is organized as follows. The next section reviews background concepts that will be used later in this report. This is followed by a description of the approach for the assessment of resilience. Severity metrics for faults and their effects are presented after that. This report concludes with a summary of accomplishments and an overview of the plan to analyze the data collected in the HEC experiment. The appendix has sketches of proofs showing that the newly defined error metrics satisfy the required mathematical properties.

2. Background Concepts

This section is a review of concepts relevant to the presentation in later sections. It covers the definition of a system, threats to achieving dependable service, the desired attributes for dependable systems, a brief description of the design process, and an overview of the concept of a metric, including the required mathematical properties.

2.1. System

For our purpose, a **system** is an entity that consists of an arrangement of components and interacts with its environment (i.e., other entities and the natural physical world) at external interfaces to perform a specific function [4, 42]. The environment defines the boundaries of the system. A system can be specified at various levels of abstraction (i.e., with varying amount of detail) in three domains: **behavioral** (i.e., in terms of the input-output response, without reference to implementation), **structural** (i.e., in terms of an interconnection of more primitive functional components, without reference to the external system-level function), and **physical** (i.e., in terms of physical components and physical characteristics, without reference to functionality). A system viewed as a **black box** is described in the behavioral domain in terms of inputs and outputs and the relation between them. In a **white-box** view of a system, the internal functional structure is visible and the system can be described in terms of the interaction among the components. The structural description of a system is recursive in the sense that each component is itself a system with its own function and structure. The recursion stops when a level is reached at which it is not possible, or of interest, to further decompose a component, and thus the component can be thought of as **atomic** or **primitive** (i.e., as a black box).

A **distributed computation system** consists of a set of processing elements (or **nodes**) interconnected by a communication network [50]. The nodes communicate with each other by sending messages over

the network to exchange data and coordinate their actions in order to achieve a common functional goal. A system is said to be **synchronous** if it performs its function within finite and known time bounds; otherwise, the system is said to be **asynchronous** [42]. This concept applies both to computation and communication systems.

The actions of a system are triggered by the **events** (i.e., changes in state [50]) of internal and external signals. A **time-triggered** system is triggered (or driven) by the progression of a **clock** that generates a sequence of events equally spaced in time (i.e., the events are **periodic** and constitute a measure of elapsed physical time). The most stringent time-related requirements for a computer system originate in applications involving the control of physical systems such as engines, airplanes, or industrial plants. These applications require sampling the state of the controlled process at regular intervals, followed by the computation and application of suitable control commands with strict timing constraints. A **distributed time-triggered system** requires a clock synchronization mechanism to establish a common time base at the processing elements, which can then be used as the foundation for coordinated action to deliver the required system-level function [50, 86, 93].

The **behavior** of a system is its sequence of outputs in time [50]. The **service** delivered by a system is the behavior as perceived by its user [4]. A **user** is a system that receives the service. In [72, 73], Powell defines a service as a sequence of **service items**, each characterized by a value (or content) and a time of observation. A service item is **correct** if the existence of the item was actually specified and its value and time are within the specified set of allowed values and time interval for the service item. In general, the specification of a service item depends on the history of inputs to the system. The correctness of a service can be accurately judged by an **omniscient observer** that has complete knowledge of the sequence of service items that should be delivered according to the specification. A **real observer** may have to rely on incomplete knowledge to derive expected (or acceptable) value and time sets for the service items.

A system may deliver a single service to multiple users. In this case, the service is defined as a sequence of replicated service items [72, 73]. This type of **multi-user** (or **broadcast) service** consists of a sequence of **broadcast service items** delivered to a set of users, with each broadcast service item consisting of a set of **single-user** (or **simplex**) service items, with one simplex item per user. A broadcast service item is **correct** if all the simplex service items are correct and there is **consistency** (i.e., agreement or symmetry) between every pair of simplex items. In the value domain, two types of agreement are possible: exact or approximate (i.e., inexact). Two values are in **exact agreement** if they are exactly equal. **Approximate agreement** is defined with respect to a specified error bound, such that two values are in approximate agreement if the difference between them is smaller than or equal to the error bound. In the continuous physical time domain, only the concept of approximate agreement is defined. Two simplex service items are in agreement if their value and time elements are in agreement. The selection of either exact or approximate value agreement in the specification of a broadcast item is dependent on the nature of the service being delivered.

2.2. Defects and Failures

Next we review service failure terminology, categories and models.

2.2.1. Fault, Error, Failure

The service delivered by a system can be either correct or incorrect. A service **failure** event is a transition from correct to incorrect service [4]. A service **restoration** event is the transition from incorrect to correct service. A service **outage** is an interruption in correct service delivery lasting from

the time of the failure to the time of service restoration. For the purpose of this report, it is useful to model a service outage as an **error burst**, which is an incorrect subsequence of service items*. The service is correct and said to be **available** before and after an error burst.

The terms fault, error, and failure are used to describe a cause-and-effect relationship between undesired circumstances in the context of the hierarchical composition of a system. Failure is assessed at the external interface of a system and is determined by deviations from the behavior expected according to the specification. An **error** is a deviation from the intended value and/or timing of data somewhere in a system. A **fault** is a defect in a system component that is the cause of errors. A fault in a system corresponds to a failure of a component. The fault, error, and failure terms facilitate the structured analysis of the failure characteristics of a system and the determination of failure causality chains from low-level components to higher-level components. In a simple chain, the failure of a system is due to the presence of errors in it, which are caused by one or more faulty components that failed to deliver the intended service. At this point, a faulty component can be seen as a failed system and the failure causality chain can be expanded by further exploring the hierarchical structure. The chain ends when a component is reached beyond which no internal structure can be discerned or is of interest [4].

2.2.2. Fault Classification

Faults can be classified according to a multitude of criteria. Avizienis et al. [4] proposed a fault taxonomy based on the following classification criteria: phase of creation (either development or operations), system boundaries (internal or external defect), phenomenological cause (natural or human-made), dimension (hardware or software defect), objective (malicious or non-malicious), intent (deliberate or non-deliberate), capability (accidental or incompetence), and persistence (permanent or transient). Suri et al. [85] proposed the following fault classification criteria: activity (either latent or active, i.e., generating errors), duration (permanent or transient), perception (symmetric or asymmetric, as manifested at the service users), cause (random or generic, i.e., systemic), intent (benign or malicious, i.e., detectable or not by the users), count (single or multiple), time of multiple faults (coincident or distinct), and cause of multiple faults (independent or common mode, i.e., same or different causes).

For characterizing the resilience of a system, we prefer fault classification criteria more suitable to the analysis of system effects. In general, a system is a recursive composition of (sub-)systems, and the main purpose of a system (as well as a sub-system) is to deliver a service to it users, which are other systems. The output service of a system is the input service of another. Thus, we focus on fault classification criteria that characterize the service delivered by a component. Our preferred service classification criteria are the following.

- **Correctness**: Whether the service delivered is correct or incorrect.
- **Inline detectability**: Whether a user can independently detect incorrect service. If a user can detect input service errors using inline acceptance checks (e.g., coding, timing and reasonableness checks [42, 94]), then it may be able to take appropriate actions to prevent the propagation of the errors to its own computation.

---

* More precisely, an *error burst* is a sequence of service items in which the first and last are incorrect and there is not a subsequence of *g* or more correct service items within the burst [32]. An outage is preceded and followed by sequences of at least *g* correct service items. The parameter *g* is the *guard band* of the burst, and its value may vary depending on the purpose and specifics of the service failure analysis performed.

- **Symmetry**: Qualitatively, a service failure can be symmetric (i.e., all users receive the same service) or asymmetric (i.e., not all users receive the same service). Quantitatively, we can count, for example, how many pairs of users observe the same failure manifestations and how many pairs disagree on their observations.

- **Persistence**: Qualitatively, a service failure can be permanent or transient. Quantitatively, the failure persistence has a specific duration.

2.2.3. Hybrid Fault Models

For proper coordinated action, the processing elements of a distributed system must have a consistent view of the system state and the results of distributed computations. The processing elements use distributed protocols to achieve and preserve valid agreement on the state and data [42, 61]. The Omissive-Transmissive Hybrid (OTH) fault model [5, 6, 89, 90, 93] defines a fault classification suitable for the analysis of distributed agreement protocols. The OTH fault categories are defined as follows for a broadcast service item.

- **Correct Symmetric (CS)**: All users accept the same correct simplex service item.

- **Omissive Symmetric (OS)**: All users reject the service item.

- **Transmissive Symmetric (TS)**: All users accept the same incorrect simplex service item.

- **Strictly Omissive Asymmetric (SOA)**: Some users accept the same correct simplex service item and others reject the item.

- **Single-Data Omissive Asymmetric (SDOA)**: Some users accept the same incorrect simplex service item and others reject the item.

- **Transmissive Asymmetric (TA)**: The users have other patterns of disagreeing simplex service items.

A user **rejects** a service items if the item is not received at all or the received item is detectably incorrect based on input error detection checks; otherwise, the user **accepts** a received service item. In the OTH model, an incorrect item is **omissive** if it is detectable by the input acceptance checks at the user; otherwise, the item is **transmissive**. Alternative equivalent terms for omissive and transmissive incorrect items are **detectable** and **undetectable** by input acceptance checks, respectively.

Notice that a set of omissive items is symmetric irrespective of the actual content or timing of the items because symmetry in this case is assessed based on input unacceptability (i.e., error detectability). Also, notice that the SOA category divides the users into two subgroups, one CS and the other OS, so each is symmetric on its own. Likewise, the SDOA category divides the users into TS and OS subgroups. Additionally, notice that a broadcast item is TA only if there is at least one pair of users that accepts disagreeing input items, which can be either one correct and one transmissive, or two transmissive items.

The OTH fault model is complete in the sense that it covers all possible error patterns for a broadcast service item. Furthermore, the fault categories are mutually exclusive and form a partition of the set of possible error patterns (i.e., the manifestations of any given service item fall under exactly one of the OTH categories). Note that other fault-space partitions not based on the OTH model might be better

suited for a particular analysis being performed.

A service outage consisting of multiple broadcast service items may have error manifestations under one or more OTH categories. Often, a service outage is classified based on the worst-case error manifestation. With respect to error severity, an incorrect item is considered to be worse than a correct item, an undetectable incorrect item is worse than a detectable one, and an asymmetric item is worse than a symmetric one. Miner et al. [61] used the following classification for a hybrid fault model.

- **Good**: The service is correct.
- **Benign**: The service items are either correct or omissive symmetric.
- **Symmetric**: The service items may be arbitrary (i.e., correct, omissive or transmissive) but all users receive the same service.
- **Asymmetric**: The service items may be arbitrary and asymmetric.

This classification has a failure semantics (i.e., failure mode) [16] order of increasingly severe behavior from Good to Asymmetric, and forms a constrained-behavior hierarchy such that an Asymmetric service includes Symmetric service, which in turn includes Benign, which includes Good service.

2.3. Attributes

Next, we consider a series of system qualities that are related to and provide a context for the concept of resilience in a safety-critical real-time system. Here the qualities of a system are defined in terms of the service it delivers.

2.3.1. Real-Time

In a **real-time** (i.e., **time-critical**) system service, the correctness of the service is determined not only by the value of the service items, but also the time of delivery [50]. A **hard real-time** service must always deliver service items within the specified time interval, as there may be severe consequences on the users if this constraint is violated. A **soft real-time** service may fail to deliver service items within the specified time constraint, but the utility of the item decreases when the constraint is violated [85]. Some systems have **firm real-time** service requirements in which infrequent timing constraint violations are tolerable but may degrade the quality of the service. Some systems may be firm real-time with respect to the quality of the service, but hard real-time with respect to safety. For these systems, the quality of the service degrades as the update delay increases beyond the firm timing constraint until the hard real-time constraint is reached, at which point safety is compromised. This hard real-time delay threshold corresponds to the **time-to-criticality**[*] of a system, which is the time interval between the occurrence of a failure and the user or environment reaching an unsafe state. For example, Paulitsch et al. [69] and Pimentel [71] reference a design requirement of 50 ms maximum service outage duration for an automobile steer-by-wire system.

[*] Based on definition at http://www.faa.gov/library/manuals/aviation/risk_management/ss_handbook/media/app_j_1200.pdf. Accessed September 2, 2012.

2.3.2. Reliability

Reliability refers to the uninterrupted delivery of correct service [4]. Reliability is measured as the probability that correct service will continue for a time interval of specified duration and under stated conditions [50, 59]. The conditions can be physical environmental conditions (e.g., temperature, vibration, etc) in the case of a physical system, and include the configuration of the system, the functional input patterns and possibly the types and number of faults experienced by the system.

2.3.3. Recoverability

Recoverability is the ability to restore correct service delivery after experiencing a failure. Here we use the term recoverability to refer to the ability of a system to restore service on the fly. This falls under the larger context of **maintainability**, which includes physical replacement and repair of system components. For our purpose, recoverability is the complement of reliability and is measured as the probability that the service is restored within a specified time interval after the occurrence of a failure. Suri et al. [85, p. 5] ranked fault-handling strategies by their best achievable recovery-delay performance. In order of decreasing minimum recovery delay, these strategies are: diagnosis and reconfiguration policies, active and passive replication policies, and fault masking policies.

2.3.4. Availability

In this report, we use the term **availability** to refer to the fraction of time that the delivered service is correct. The availability of a service is a function of the reliability and recoverability. Availability is highest when both reliability and recoverability are high, as in this case the service remains correct for long time intervals and is quickly restored after a failure occurs.

2.3.5. Integrity

In a general sense, integrity is related to the concept of truthfulness. Avizienis et al. [4] defined **integrity** as the absence of improper system state alterations. A service item satisfies this condition when it is correct, or incorrect but detectable by the user. Integrity is violated when the user accepts an incorrect service item. In [69], Paulitsch et al. defined integrity as the probability of an undetected failure. Service integrity is an important quality as it is related to the likelihood that the effects of a fault in a system will propagate and corrupt other systems.

For distributed systems, proper coordinated action depends on consistency of state. Integrity in a distributed system is violated when users expect to receive the same service but are actually delivered different services.

2.3.6. Safety

Avizienis et al. [4] defines **safety** as the absence of catastrophic consequences on the user(s) and the environment. In the IEC 61508 standard [41], safety is defined as freedom from an unacceptable combination of the probability of the occurrence of injury or damage to people, equipment or the environment, and the severity of the occurrence. Kopetz [50] defines safety as the probability that a system will survive for a given time interval without a critical failure mode (i.e., a failure mode that can lead to catastrophic consequences), and thus, in this sense, safety is reliability regarding critical failure modes.

For a safety-critical real-time system, safety requirements can be stated in terms of system functional quality attributes related to two basic service failure modes: loss of function (i.e., passive failure) and malfunction (i.e., active failure) [99 - 102]. The system safety requirements can be expressed in terms of the availability and integrity of the service delivered to the users. Unavailability of the service with preserved integrity is a passive failure that may not be catastrophic if service can be restored before the time-to-criticality. An integrity violation corresponds to an active failure and is considered (or assumed to be) immediately unsafe (i.e., catastrophic). The ability to suppress incorrect service items at the source and to detect incorrect service items at the users are critical determining factors of safety.

2.3.7. Robustness

Siewiorek et al. [80] defines robustness as the ability of a system to identify and handle errors (at the functional inputs or internal to the system) of varying severity in a consistent and predictable manner. Kopetz in [50] states that a system is **robust** if the severity of the consequences of a fault is inversely proportional to the probability of fault occurrence (i.e., frequent faults have less severe effects on service quality than infrequent faults). Bishop et al. [11] considers a system to be robust if it resists a wide range of attacks (i.e., faults) and operational conditions without significant service degradation, but may not have the ability to restore lost functionality (i.e., service quality).

The main aspect of interest to us relative to system robustness is the assessment of service quality over a wide range of operational conditions (including number and severity of internal faults). Under these conditions, a service is robust if it satisfies Kopetz's criterion for robustness (i.e., severity of effects is inversely proportional to the frequency of the fault). In general, robustness does not imply ability to recover the service, but recovery may be essential for safety-critical real-time systems in order to satisfy Kopetz's criterion.

2.3.8. Survivability

A survivable system has the ability to continue to operate, possibly with highly degraded service, even under severe conditions. In essence, a survivable system may be easily degraded but nearly impossible to disable completely [11]. A survivable system may have the ability to effect some degree of recovery, but this is not essential.

2.3.9. Resilience

A resilient system may experience degraded operation due to faults, but will eventually recover. In [55], Laprie defines resilience as "the persistence of service delivery that can justifiably be trusted, when facing changes." Trivedi et al. [96] states that "resilience deals with conditions that are outside the design envelope" and, in general, refers to the ability of a system to resist and recover from shock or strain. In [59], Leveson states that resilience is often defined as "the ability to continue operations or recover stable state after a major mishap or event". Bishop et al. [11] states that "a resilient system is effectively a survivable system that is capable of restoring not only its performance level back to desirable levels, *but also the capacity of the system itself to recover*, maintaining its ability to sustain future attacks or failures."

From our perspective, resilience describes the ability of a system to mitigate the effects of component-level service degradations. A system is resilient to faults if its service quality is hard to degrade and quality is restored after the fault condition has subsided. For safety-critical real-time systems, availability (in terms of reliability and recoverability) and integrity are the most significant measures of service

quality. Therefore, for safety-critical real-time systems, we define **resilience** as the ability to preserve and restore service availability and integrity under stated conditions.

2.4. Fault Hypothesis

Depending on the application (e.g., commercial transport aircraft, manned military aircraft, autonomous vehicles, engine controls), the probability requirement for critical failure modes of safety-critical systems may be in the range of 10^{-6} to 10^{-10} per hour [50, 52, 53]. Given that electronic components (e.g., chips, circuit boards, computer modules) have failure rates on the order of 10^{-4} to 10^{-6} per hour [50, 52] with failure modes that can be difficult or impossible to characterize [17], physical redundancy and suitable redundancy management mechanisms are necessary to meet system safety requirements regarding failure modes and rates.

The first major step in the design of a fault-tolerant system is the specification of the **fault hypothesis** (or **fault assumptions**) [50]. The fault hypothesis specifies a partition of the system into **fault containment regions** (FCR), which are components assumed to experience defects with a high degree of independence (in a probabilistic sense) [49, 52, 53, 65 - 67]. This implies that whatever causes a defect in an FCR is unlikely to coincidentally also cause a defect in another FCR, and that a defect in an FCR is unlikely to cause a defect in another FCR (i.e., a fault cascade). Thus, the FCRs are the basic units of failure in a system [50]. The fault hypothesis states the expected FCR failure modes and rates, as well as the maximum number of simultaneously failed FCRs that the system may experience in operation [50, 72, 73, 85]. (Note that the fault hypothesis is a way of specifying part of the "stated conditions" in the definition of system reliability, with other parts being the functional inputs and the configuration of the system.)

The system is designed to meet the functional and service quality requirements while handling the fault space defined by the fault hypothesis. The operational fault-handling effectiveness of a system depends on two basic factors: the **fault-assumption coverage** (i.e., the probability that actually occurring faults are within the assumed fault space) and the **fault-handling coverage** (i.e., the probability that assumed faults are properly handled by the system) [4, 72, 73]. Thus, the design development is an iterative optimization process involving refinements to the fault hypothesis and the fault handling strategy and mechanisms. Usually, the fault modes and rates of non-redundant, primitive system components are fixed as determined by the implementation technology. The component fault rate is a determining factor in the amount of redundancy needed to satisfy the system service availability requirement, and the fault modes (i.e., failure semantics) influence the amount and organization of redundancy to satisfy the integrity requirement [5, 6, 52, 72, 73]. In general, to satisfy particular availability and integrity requirements, higher fault rates necessitate increased redundancy, and less constrained fault modes demand increased redundancy and more complex organization. However, using redundancy in hardware, software, time and/or information domains [43], it is possible to define higher-level structural components with less severe (i.e., safer) failure-mode rate profiles [16]. This approach increases the complexity of these higher-level components in exchange for easier-to-handle failure modes, which enables a simpler high-level system design. Examples of this include Honeywell's SAFEbus with self-checking-pair components [17], Airbus' Command-Monitor (COM-MON) computers [13, 94, 95], and the Boeing 777 primary flight computers with triple internal redundancy in a command-monitor-standby configuration [99 - 102]. The design of the B777 flight computers shows that it is possible to constrain the failure-mode rates of high-level components while preserving their availability.

The fault hypothesis divides the fault space into *normal* and *rare* fault regions (or subsets) based on assumed FCR failure rates [50]. (These regions are also labeled, respectively, *expected* and *unexpected*,

or *credible* and *non-credible*.) A primary system design goal is to achieve fault-handling coverage as close as possible to 100% for normal faults. To mitigate the risk of a fault assumption violation, the system fault-handling design should also cover a significant percentage of the most likely fault scenarios in the rare-fault region. Ideally, a robust system should have fault-handling coverage that is proportional to the probability of occurrence of fault scenarios. An example of a fault-handling robustness approach is the *never-give-up* operational strategy of the Boeing 777 primary flight control computer (PFC), which required a high probability that the PFC would continue operating as long as there were known good resources and that it would recover from temporary failures [85, p. 12]. According to Kopetz [49], in a properly designed system, a likely scenario for a fault-hypothesis violation is a transient correlated failure of multiple FCRs. For such scenarios, a robust safety-critical real-time system should recover to an operational state with high probability. The ability of a system to recover from any arbitrary state is called **self-stabilization** [1, 3, 33, 34].

2.5. Measurements and Metrics

Measurement is the process of determining the amount (i.e., degree, size or extent) of some property present in an entity. There are four basic scales of measurement [83]: nominal, ordinal, interval, and ratio. A **nominal** scale defines categories (or classes) of the property of interest in terms of exemplars and/or descriptions of membership in a category. A measurement on a nominal scale simply assigns a category by determining equality with members of the category. An **ordinal** scale adds a ranking relationship between nominal categories such that it is now possible to compare the amount of the property of interest in any two categories and determine which is greater. An **interval** scale defines the difference (or "distance") between any two entities in their amounts of the property of interest. This requires the definition of a *unit of measurement*, which is an accepted or standard amount such that any quantity of the property of interest can be expressed as a multiple of it. In an interval scale, the definition of the *zero* amount point is arbitrary or by convention, and therefore, the ratio between numbers on an interval scale is meaningless. A **ratio** scale introduces the concept of an absolute zero. A ratio scale is the kind commonly used in physics and engineering, and requires the definition of the four relations: equality, rank, difference, and ratio.

A **metric** is a mathematical function that defines, for every pair of elements in a set, how far apart the elements are from each other (i.e., the distance between the elements) [36]. Thus, a metric is defined on an interval scale with a suitable unit of measurement. Let x, y, and z denote elements in a set S, and let d denote a function defined on the set S. Function d is a metric if it satisfies the following properties.

- Non-negativity: $d(x, y) \geq 0$
- Symmetry: $d(x, y) = d(y, x)$
- Identity of Indiscernibles: $d(x, y) = 0$ if and only if $x = y$
- Triangle Inequality: $d(x, z) \leq d(x, y) + d(y, z)$

3. Resilience Assessment Approach

We have defined resilience in safety-critical real-time systems as a measure of the ability to preserve and restore service availability and integrity under stated conditions. The statement of conditions

specifies the system configuration and the functional inputs, as well as the threats to the delivery of proper system service. These *threats* may be described as fault conditions occurring internal to the system, or as external environmental conditions (e.g., HIRF, lightning, high-energy particle radiation, power system transients, etc) that may cause faults in the system. In what follows, we treat the configuration and functional inputs as given, and we focus on the relation between the threat conditions and the quality of delivered services.

The system can then be viewed from a stimulus-response (i.e., cause and effect) perspective (see Figures 1 and 2). The threat conditions specify the *stimulus space*, which is a subset of all possible system threat patterns. A **disturbance** is an external system stimulus that may cause a **perturbation**, defined here as an internal fault condition in the form outages on the services provided by the components. Alternatively, we can skip the specification of the disturbance and specify the stimulus as a perturbation. The effects of a perturbation (i.e., errors) may propagate throughout the system and reach the external functional interface, thus causing an outage on the external system service, which we refer to as a **disruption**. The *response space* is the set of system disruptions resulting from the application of the stimulus space.

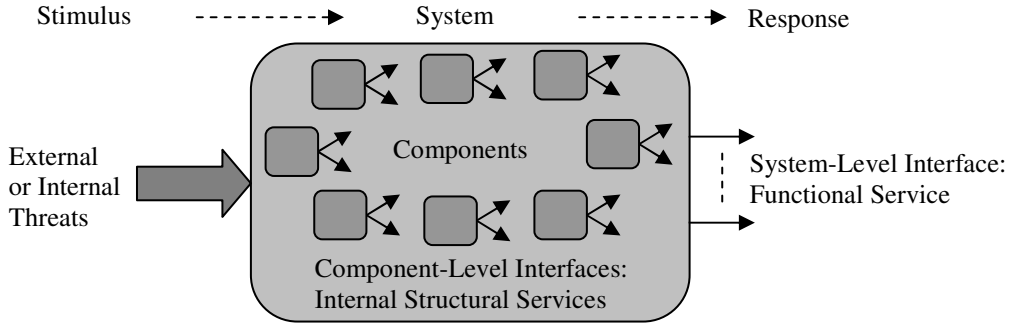


Figure 1: Stimulus-Response System Model

As Figure 2 suggests, for a given perturbation, the severity of the disruptions is determined by the error propagation characteristics of the system. Thus, from this perspective, **resilience** can be defined as the ability to contain the propagation of internal errors and to repair propagated errors. Note that these two aspects of resilience (i.e., containment and repair) correspond to the two key system attributes of integrity and availability.

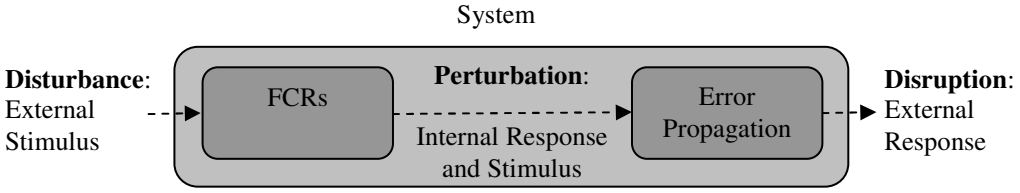


Figure 2: Stimulus-Response Chain

For a quantitative system resilience analysis, the disturbance, perturbation and disruption spaces may be described in terms of probability distributions (PDs) of random variables whose values correspond to the severity of occurrences (i.e., items, events or instances) in these spaces. Figure 3 illustrates the use of PDs to describe the spaces in the stimulus-response chain. To enable the use of such distributions in analyses, we need to define severity metrics for the occurrences in each space. Report [87] offers a

simple occurrence severity model for a HIRF disturbance. For perturbations and disruptions, we use the concept of **corruption**, which we define as the amount of error in a service outage. Sections 4 and 5 in this report present our proposed corruption metrics for disruptions and perturbations. Given a stimulus probability distribution, we can perform a stimulus injection experiment (e.g., a Monte Carlo experiment) to generate a response set, for which we can then compute the probability distribution.

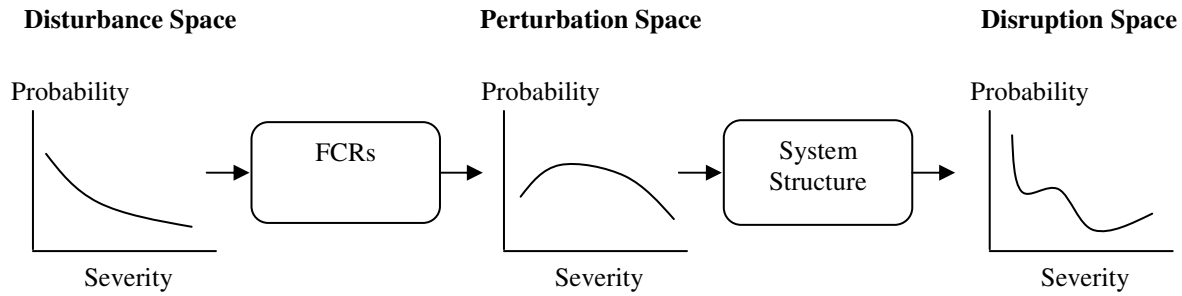


Figure 3: Description of spaces in the stimulus-response chain using probability distributions

If we have a PD-based description of a space, we can compute basic statistical measures like percentiles (e.g., minimum, median, maximum and quartiles) and averages (e.g., mean, standard deviation SD, and root-mean-square RMS) to characterize the distribution. Note that a PD-based description of the spaces in the system stimulus-response chain is, in effect, an abstract system description that is suitable for quantitative comparison of system quality attributes for different systems or the same system under different conditions.

Our preferred statistical measure for these error or error-inducing spaces is the RMS value of the distribution. The RMS value of a space (i.e., a set) described by the probability distribution of the severity of occurrences is defined as follows. We consider the case of a discrete-valued severity range, but the description is similar for a continuous range, using an integral instead of a summation. Let λ be the index of the severity levels in the space, with $0 \le \lambda \le \lambda_{max}$, and let s_λ and p_λ denote the magnitude and probability of the λ-th severity level. The RMS severity S_{rms} is given by:

$$S_{rms} = \sqrt{\sum_{\lambda=0}^{\lambda max} p_\lambda \cdot s_\lambda^2}$$

The RMS value has the interesting property that it can be expressed in terms of the mean S_{mean} and standard deviation S_{SD} as follows:

$$S_{rms}^2 = S_{mean}^2 + S_{SD}^2$$

The reason we prefer the RMS value to the mean is that if we compare two distributions with the same mean, the one with the largest dispersion as measured by the standard deviation has a larger proportion of occurrences at higher severity levels. Intuitively, we consider that distribution to have a higher aggregate degradation severity, and we want our statistical measure to reflect that. The RMS value is a more conservative measure of the occurrence severity distribution than the mean. Of course, a single measure is usually not an adequate characterization of a distribution, and we should always consider the other statistical measures, including averages and percentiles.

Let $s_{\lambda max}$ (= s_{max}) denote the highest disruption severity. We define the RMS severity of the disruption space as the **RMS corruptibility** Q_{rms} of a system for a given stimulus space. Q is an aggregate measure of the deterioration in system service quality due to the disturbance or perturbation stimulus. That is the opposite of resilience. Thus, we define the **RMS resilience** R_{rms} as the complement of corruptibility:

$$R_{rms} + Q_{rms} = s_{\lambda max} \quad (1)$$

We normalize the severity scale to simplify the interpretation of these measures. The **normalized RMS corruptibility** and **resilience**, denoted q_{rms} and r_{rms} , are given by:

$$q_{rms} = Q_{rms}/s_{\lambda max} \quad (2)$$

$$r_{rms} = R_{rms}/s_{\lambda max} = (s_{\lambda max} - Q_{rms})/s_{\lambda max} \quad (3)$$

Equation (1) then becomes:

$$r_{rms} + q_{rms} = 1 \quad (4)$$

The following sections describe our proposed disruption and perturbation severity metrics.

4. Service Disruption Metrics

We seek meaningful measures for the amount of error in a service disruption, which is an error burst consisting of a sequence of one or more service items, some of which are in error. We would like the error measures to be useful in error propagation analyses, especially for synchronous distributed systems, where the validity and agreement of the data and state are paramount [61, 93]. Based on our system analysis experience, the OTH fault model is defined at a suitable level of abstraction for service items [61, 89, 93]. Thus, in accounting for service errors, we choose the service item as the lowest level of granularity and we abstract out the service item dimensions of value and time. Furthermore, for a single-user service, the OTH model describes a service item based on two criteria: *correctness* and *detectability*. For a multi-user service, the OTH model for a broadcast service item adds the dimension of *symmetry* for the classification of error patterns. To describe an error burst, we also need to consider the dimension of *persistence* (i.e., duration) of the burst. The selected strategy to measure the disruption error is to define a separate *dimensional-error* metric for each of these criteria and then properly combine the dimensional metrics to form a composite *total-error* metric. Note that depending on the purpose of a particular resilience analysis being performed, the disruption error may be measured in terms of one of the basic dimensional errors or some function of these.

We assume a synchronous user model by which the timeline is partitioned into a complete set of mutually exclusive time intervals such that there is exactly one item expected for each interval. The range of each time interval coincides or extends beyond the correct time range of the corresponding service item. In the case of a multi-user service, all the users are assumed to have perfect mutually synchronized time intervals. This model accounts for errors in which the number of items a user receives in a time interval is fewer or greater than expected. In the assumed user model, such errors are allocated to the expected service item.

4.1. Single-User Service Item

Figure 4 illustrates the model for a single-user service. With respect to *correctness*, a delivered service item at the user can be either correct or incorrect. With respect to *detectability* by the input acceptance check, the item can be either detectable or undetectable. As shown in Figure 4, a service item x can be expressed in terms of correctness x_C and detectability x_D "coordinates", both of which are Boolean variables. To quantify the values in each of these dimensions, we use the same basic idea as in the *Hamming distance* [98], where the unit of measurement is a disagreement (≠) between values. This way the distance between correct and incorrect is assigned the numeric value of 1, and the distance between detectable and undetectable is also 1. Let $d_C'(x, y)$ and $d_D'(x, y)$ denote the distance between items x and y in the dimensions of correctness and detectability, respectively, *assuming independence between the dimensions*. Table 1 shows the values of d_C' and d_D' for all possible combinations of correctness and detectability.

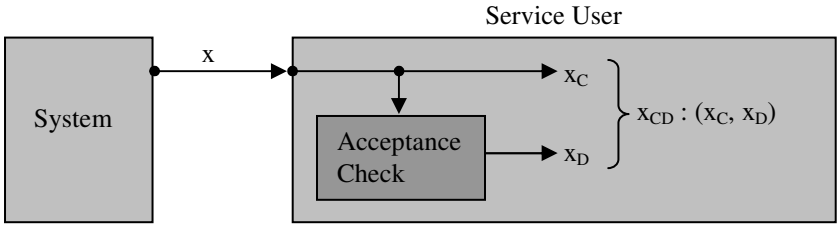


Figure 4: Single-user service model with correctness and detectability dimensions

Table 1: Correctness and detectability distance functions for single-user service items assuming dimensional independence

x_C	y_C	$d_C'(x, y)$
T	T	0
T	F	1
F	T	1
F	F	0

x_D	y_D	$d_D'(x, y)$
T	T	0
T	F	1
F	T	1
F	F	0

In theory, the correctness and detectability dimensions of the model can be independent, but in practice, a correct item should not be declared invalid by the input acceptance check as that effectively reduces the availability of the system. Here we assume that the acceptance check never invalidates a correct service item (i.e., no false-positives). However, it is not always possible to achieve perfect inline detectability, which means that some incorrect items may not be detectable (i.e., possible false-negatives). This is related to the trade-off between *correctness* and *completeness* in system diagnosis, where a correctness-biased diagnosis policy will not declare bad any good component but may have to allow some bad components to be declared good, whereas a completeness-biased diagnosis policy identifies all bad components but may also declare bad some components which are actually good [93, p. 27]. Therefore, in effect, our service model assumes a correctness policy for input acceptance checks.

To measure the total service-item error for correctness and detectability, we need to consider the interaction between these dimensions taking into consideration the above discussion. In particular, a correct item must not be said to have an error in the detectability dimension. An incorrect item may be

either detectable or undetectable. Thus, there are only three outcomes for a service item: correct (denoted C), detectable incorrect (D), and undetectable incorrect (U). Let $d_C(x, y)$ and $d_D(x, y)$ denote the distance between items x and y in the dimensions of correctness and detectability, respectively, *with dependence between the dimensions*. $d_{CD}(x, y)$ denotes the correctness-and-detectability distance between items x and y. The value of d_{CD} is the sum of the distances in the correctness and detectability dimensions (i.e., the *Manhattan Distance*).

$$d_{CD}(x, y) = d_C(x, y) + d_D(x, y) \quad (5)$$

Table 2 shows the values of d_C and d_D for all combinations of x and y. Note that in order to accommodate the dependence between the dimensions, we set $d_D(C, D) = 0$ despite the fact that C is not detectable while D represents a detectable item. This is justified by the fact that there is no misdetection for either C or D.

Table 2: Correctness and detectability distance functions for single-user service items with dimensional dependence

x	y	$d_C(x, y)$	$d_D(x, y)$	$d_{CD}(x, y)$
C	C	0	0	0
C	D	1	0	1
C	U	1	1	2
D	D	0	0	0
D	U	0	1	1
U	U	0	0	0

To measure the total amount of correctness and detectability error, we define C as the zero-error reference item. Let e_C and e_D denote the correctness and detectability dimensional errors, respectively, which are defined as follows.

$$e_C(x) = d_C(x, C) \quad (6)$$

$$e_D(x) = d_D(x, C) \quad (7)$$

Then the total correctness-and-detectability error, denoted e_{CD}, is given by the following expression.

$$e_{CD}(x) = e_C(x) + e_D(x) = d_{CD}(x, C) \quad (8)$$

Table 3 breaks down of correctness and detectability error for a single-user service item. Notice that the case of correct and detectable is assumed not to occur. Also, notice the correspondence between the combination of correctness and detectability and the attributes of availability and integrity. Specifically, an item is available if it is correct, and it has integrity if it is either correct or detectable incorrect. An undetectable incorrect item has a total error count of 2 because it is both incorrect and undetectable.

Table 3: Correctness and detectability errors for a single-user service item

x_{CD}	x_C	x_D	Availability	Integrity	e_C	e_D	$e_{CD} = e_C + e_D$
--	T	T	F	T	0	1	(1)
C	T	F	T	T	0	0	0
D	F	T	F	T	1	0	1
U	F	F	F	F	1	1	2

From Table 3, it is easy to see that the following relations are valid. The proofs are given in Appendix A.

$$d_C(x,y) = |e_C(x) - e_C(y)| \quad (9)$$

$$d_D(x,y) = |e_D(x) - e_D(y)| \quad (10)$$

$$d_{CD}(x,y) = |e_C(x) - e_C(y)| + |e_D(x) - e_D(y)| \quad (11)$$

$$d_{CD}(x,y) = |e_{CD}(x) - e_{CD}(y)| \quad (12)$$

4.2. Multiple-User Service Item

For multi-user service, we must account for the error in the dimension of symmetry among the simplex items. The total error, then, includes the dimensions of correctness, detectability and symmetry.

We begin by defining the error for the dimensions of correctness and detectability using the concepts in the definition of error for a single-user service item. We then define the symmetry error for approximate and exact agreement. Finally, we define the total error as a combination of the dimensional errors.

4.2.1. Correctness and Detectability

We use the letter n to denote the number of service users. A multi-user service item X is a vector of simplex service items denoted by x_i for $1 \le i \le n$. Thus:

$$X = (x_1, x_2, \ldots, x_n)$$

In the correctness and detectability dimensions, the distance between two multi-user items X and Y is given by the sum of the distances between respective elements.

$$d_C(X, Y) = d_C(x_1, y_1) + \ldots + d_C(x_n, y_n) \quad (13)$$

$$d_D(X, Y) = d_D(x_1, y_1) + \ldots + d_D(x_n, y_n) \quad (14)$$

The total correctness-and-detectability (CD) distance is also given by the sum of the distances between the elements.

$$d_{CD}(X, Y) = d_{CD}(x_1, y_1) + \ldots + d_{CD}(x_n, y_n) \quad (15)$$

The correctness and detectability dimensional errors of service item X are given by the sum of the errors of its elements:

$$e_C(X) = e_C(x_1) + \ldots + e_C(x_n) \quad (16)$$

$$e_D(X) = e_D(x_1) + \ldots + e_D(x_n) \quad (17)$$

The total CD error is given by:

$$e_{CD}(X) = e_{CD}(x_1) + \ldots + e_{CD}(x_n) \quad (18)$$

A multi-user service item X in the correctness and detectability model can be expressed as vector $X_{CD} = (x_{1,CD}, x_{2,CD}, \ldots, x_{n,CD})$. Because the correctness-and-detectability error is commutative, the order of the elements in X_{CD} is not significant in the computation of e_{CD}. Based on this, X_{CD} can expressed using the following compact notation, with b + p + w = n.

$$X_{CD} = C^b D^p U^w \quad (19)$$

Therefore, $e_{CD}(X)$ can also be expressed as follows.

$$e_{CD}(X) = [b \cdot e_{CD}(C)] + [p \cdot e_{CD}(D)] + [w \cdot e_{CD}(U)] = p + 2w \quad (20)$$

4.2.2. Symmetry

Approximate agreement and exact agreement have fundamentally different relational structures. In defining the symmetry error for a multi-user service item, we must consider these structures in defining the "amount of error". Thus, we define separate error functions for each type of agreement.

For both approximate and exact agreement we assume that all correct simplex items are in mutual agreement (i.e., any C is in agreement with every other C and C^n is symmetric), all detectable incorrect items mutually agree (i.e., any D is in agreement with every other D and D^n is symmetric), and detectable incorrect items are in disagreement with all other kinds of items (i.e., Ds do not agree with Cs nor Us).

4.2.2.1. Approximate Agreement

Figure 5 illustrates the pair-wise approximate agreement relations for n = 4. In general, there are a total of n(n + 1)/2 pair-wise (i.e., binary) relations that define the agreement pattern for an n item set. In Figure 5, a link represents agreement between the pair of connected items. The graph is fully connected for a *symmetric* set. An *asymmetric* set has one or more missing links.

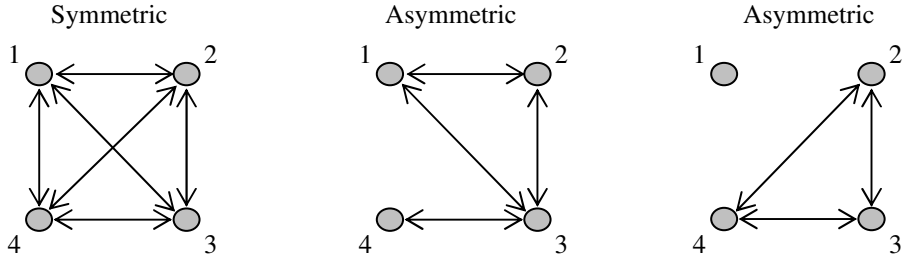


Figure 5: Examples of symmetric and asymmetric approximate-agreement patterns for n = 4

However, the links in Figure 5 may not be independent. To see this, consider the concept of approximate agreement for a set of n items in which each item is mapped to a point on a number line, which can be either integer or real valued. The items can thus be sorted by increasing magnitude. A pair of items is in agreement if their distance on the line is smaller than or equal to a specified bound, denoted

ε. Figure 6 illustrates the pair-wise agreement relations when the individual items are placed in order on a number line. A link between nodes i and j is denoted $l_{i,j}$. The set of all possible links, denoted L, is given by:

$$L = \{ l_{i,j} , 1 \le i \le n, 1 \le j \le n, i \ne j \}$$

This set is composed of all links between first (i.e., immediate) neighbors, all links between second neighbors, and so on, with the last link being between the nodes at the extreme ends. On a number line, a link $l_{i,j}$ is said to **cover** link $l_{k,m}$ if the segment from k to m is included in the segment from i to j. Notice that if a link is missing (e.g., $l_{1,2}$), then all longer links that cover the missing link are also missing (i.e., $l_{1,3}$ and $l_{1,4}$). This corresponds to the property that, if items 1 and 2 disagree, then all items at or to the left of 1 disagree with all items at or to the right of 2. This property does not apply in the opposite direction from longer links to covered shorter links. For example, if $l_{1,3}$ is missing, $l_{1,2}$ and $l_{2,3}$ may still be present.

Considering these agreement graphs, we can think of two different approaches to measure the symmetry distance between two multi-user service items. One approach is to focus on the individual links as independent units of symmetry and define the symmetry distance between two multi-user items as the number of agreement links in which they differ. For example, patterns (b) and (c) in Figure 6 differ in links $l_{1,2}$, $l_{1,3}$, $l_{3,4}$, and $l_{2,4}$, so the distance between the graphs is 4. The second approach is to focus on the *clusters* of fully connected subsets of individual items and compare different multi-user items based on the relative size of their clusters. In Figure 6, graph (a) has a single 4-item cluster, and both graphs (b) and (c) have clusters of size 3 and 1. Comparing cluster sizes we can say that graph (a) has a greater degree of symmetry and graphs (b) and (c) have equal symmetry, but it is not obvious how to define the distance between the patterns. We address these two symmetry-distance approaches separately.

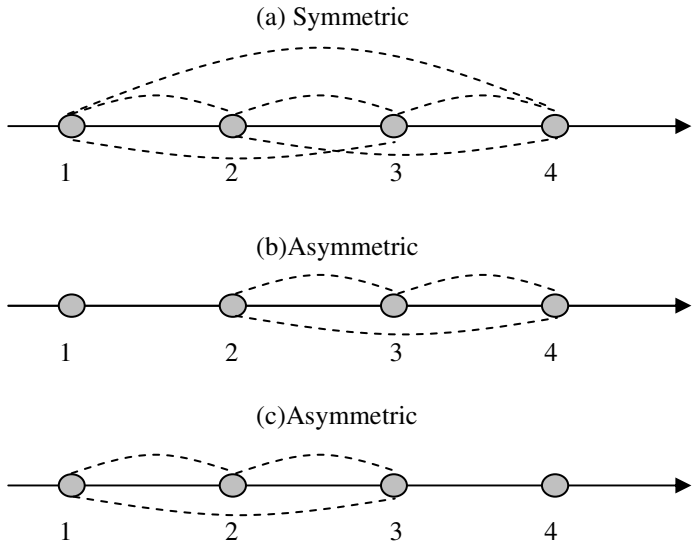


Figure 6: Examples of approximate-agreement patterns on a number line for n = 4

4.2.2.1.1. Link-based Distance

We define $l_{i,j}$ to be a Boolean variable that is TRUE (T) when node i is in approximate agreement with node j, and FALSE (F) if otherwise. L can then be represented as a Boolean vector L = (… , $l_{i,j}$, …). To define the link-based symmetry distance between two multi-user service items X and Y, we again borrow

the basic idea used in the definition of the Hamming distance. We specify the unit of link-based symmetry distance as a disagreement (≠) between corresponding links in X and Y. Link $l_{i,j}$ in X and Y is denoted $l_{X,i,j}$ and $l_{Y,i,j}$, respectively. $d_S(l_{X,i,j}, l_{Y,i,j})$ denotes the symmetry distance between items X and Y with respect to link $l_{i,j}$, such that $d_S(l_{X,i,j}, l_{Y,i,j}) = 1$ if $l_{X,i,j} \neq l_{Y,i,j}$ and $d_S(l_{X,i,j}, l_{Y,i,j}) = 0$ if $l_{X,i,j} = l_{Y,i,j}$. The symmetry distance between X and Y is given by:

$$d_S(X, Y) = \sum_{i,j} d_S(l_{X,i,j}, l_{Y,i,j}) \quad (21)$$

with $1 \leq i \leq n$, $1 \leq j \leq n$, $i \neq j$.

The total symmetry error e_S of a multi-user service item X is given by the distance from a symmetric item, represented here by C^n for convenience.

$$e_S(X) = d_S(X, C^n) \quad (22)$$

4.2.2.1.2. Cluster-based Distance

The approximate agreement graph for a multi-user item can have up to n clusters. We use the symbol α to denote the size of a cluster. Thus, for cluster-based symmetry, a multi-user service item X can be represented by a vector of n cluster sizes:

$$X_S = (\alpha_1, \ldots, \alpha_n),$$

where the α_i elements are sorted by decreasing value such that $\alpha_1 \geq \alpha_2 \geq \ldots \geq \alpha_n$. Because some of the n single-user items of the multi-user service item X may be in multiple clusters, the cluster sizes may vary such that $\alpha_i \geq 0$ and $\alpha_1 + \ldots + \alpha_n \geq n$.

We have been unable to gain sufficient insight into this cluster-based representation to identify an objective unit of measure to define a symmetry distance function. Therefore, we resort to define an ordinal scale based on ranking rules for vectors of cluster sizes. To do this, we introduce the concept of **dominance**. Let $X_S = (\alpha_{X,1}, \ldots, \alpha_{X,n})$ and $Y_S = (\alpha_{Y,1}, \ldots, \alpha_{Y,n})$ be the cluster-size vector representations of multi-user items X and Y. We say that X_S *dominates* Y_S if there is a vector-element index i, $1 \leq i \leq n$, such that $\alpha_{X,i} > \alpha_{Y,i}$ and $\alpha_{X,j} = \alpha_{Y,j}$ for $1 \leq j < i$. If X_S dominates Y_S, we say that X has more symmetry than Y, or equivalently, Y has larger symmetry error than X.

To define the cluster-based symmetry scale, we have to list all possible cluster vectors sorted by dominance, and then form a one-to-one mapping from the dominance scale to the natural numbers beginning with a value of 0 for the symmetric case, i.e., $X_S = (n, 0, \ldots, 0)$. Table 4 shows the cluster vectors with n = 4 ranked by dominance and their assigned symmetry error values. The symmetry distance between items X and Y is given by their distance on the error scale.

$$d_S(X, Y) = |e_S(X) - e_S(Y)| \quad (23)$$

Table 4: Cluster-based symmetry error scale for approximate agreement (n = 4)

Cluster Vector	Symmetry Error (e_S)
(4, 0, 0, 0)	0
(3, 3, 0, 0)	1
(3, 2, 0, 0)	2
(3, 1, 0, 0)	3
(2, 2, 2, 0)	4
(2, 2, 1, 0)	5
(2, 1, 1, 0)	6
(1, 1, 1, 1)	7

4.2.2.2. Exact Agreement

We have not been able to identify an objective unit of measure on which to base a symmetry distance function for exact agreement. Therefore, we proceed with a similar approach as for cluster-based symmetry for approximate agreement.

Again, a multi-user service item X can be represented by a vector of n cluster sizes:

$X_S = (\alpha_1, \ldots, \alpha_n)$

where the α_i elements are sorted by decreasing value such that $\alpha_1 \geq \alpha_2 \geq \ldots \geq \alpha_n$. For exact agreement, the simplex items of a multi-user service item form a partition (i.e., a set of mutually exclusive clusters incuding all the individual items) as now an item can belong to at most one cluster. The cluster sizes under exact agreement may vary such that $\alpha_i \geq 0$ and $\alpha_1 + \ldots + \alpha_n = n$. Thus, in effect, the possible cluster vectors for an n-user service item are given by the set of *integer partitions* of n [36]. Table 5 lists the integer partitions for n = 4 ranked by dominance and their corresponding symmetry error values. Equation 20 also applies here for the exact-agreement symmetry distance d_S between multi-user service items.

Table 5: Cluster-based symmetry error scale for exact agreement (n = 4)

Cluster Vector	Symmetry Error (e_S)
(4, 0, 0, 0)	0
(3, 1, 0, 0)	1
(2, 2, 0, 0)	2
(2, 1, 1, 0)	3
(1, 1, 1, 1)	4

4.2.3. Correctness, Detectability and Symmetry

The total distance between two multi-user service items over the combined dimensions of correctness, detectability and symmetry must take into account any possible dependence between the dimensions. Our definition of the correctness-and-detectability (CD) distance already accounts for the dependence between correctness and detectability. In defining the contribution of the symmetry dimension, we would like our distance function to ignore differences already accounted for in the definition of the CD distance. In particular, note that a D item always disagrees with C and U items. In addition, for exact agreement, a C item always disagrees with a U item. Thus, these "built-in" disagreements should not contribute to the

symmetry component of the total correctness-detectability-and-symmetry (CDS) distance. We consider approximate and exact agreement separately.

For approximate agreement, it is possible for a C to agree with a U. This relation must be accounted for in the CDS distance. We use X_A to denote the acceptable single-user items (i.e., items that pass the input acceptance check at the users) of a multi-user service item X. X_A can include C and U items. Using our compact CD notation for a vector, X can be expressed as follows.

$$X_{CD} = C^b D^p U^w = D^p X_A \tag{24}$$

where:

$$X_A = C^b U^w \tag{25}$$

We use the cluster-size notation to represent the agreement pattern in X_A, which we denote $X_{A,S}$.

$$X_{A,S} = (\alpha_{A,1}, \ldots, \alpha_{A,b+w}) \tag{26}$$

The total CDS distance between multi-user service items X and Y is given by:

$$d_{CDS}(X, Y) = d_C(X, Y) + d_D(X, Y) + d_S(X, Y) \tag{27}$$

with:

$$d_S(X, Y) = d_S(X_{A,S}, Y_{A,S}) = |e_S(X_{A,S}) - e_S(Y_{A,S})| \tag{28}$$

where $e_S(X_{A,S})$ and $e_S(Y_{A,S})$ are computed on their respective symmetry error scales.

The CDS error in item X is given by:

$$e_{CDS}(X) = d_{CDS}(X, C^n) = d_C(X, C^n) + d_D(X, C^n) + d_S(X, C^n)$$

$$e_{CDS}(X) = e_C(X) + e_D(X) + e_S(X_{A,S}) = e_{CD}(X) + e_S(X_{A,S}) \tag{29}$$

For exact agreement, C and U items never agree. Therefore, the symmetry distance contribution is only with respect to the U items in the multi-user service item. Let X_U denote the undetectable incorrect items in X. For exact agreement, X can be expressed as:

$$X_{CD} = C^b D^p U^w = C^b D^p X_U \tag{30}$$

with:

$$X_U = U^w \tag{31}$$

Using cluster-size notation, the exact agreement pattern in X_U, denoted $X_{U,S}$, is expressed as:

$$X_{U,S} = (\alpha_{U,1}, \ldots, \alpha_{U,w}) \tag{32}$$

Equations (27) to (29) also apply to exact agreement with $X_{A,S}$ replaced by $X_{U,S}$. Table 6 shows the

total CDS error for patterns of OTH faults assuming exact agreement and n = 4.

Table 6: Total CDS error with exact agreement for patterns of OTH faults (n = 4)

OTH Category	X_{CD}	$X_{U,S}$	e_C	e_D	$e_S(X_{U,S})$	e_{CDS}
Correct Symmetric (CS)	C^4	(0, 0, 0, 0)	0	0	0	0
Strictly Omissive Asymmetric (SOA)	C^3D	(0, 0, 0, 0)	1	0	0	1
	C^2D^2	(0, 0, 0, 0)	2	0	0	2
	CD^3	(0, 0, 0, 0)	3	0	0	3
Omissive Symmetric (OS)	D^4	(0, 0, 0, 0)	4	0	0	4
Single-Data Omissive Asymmetric (SDOA)	D^3U	(1, 0, 0, 0)	4	1	0	5
	D^2U^2	(2, 0, 0, 0)	4	2	0	6
	DU^3	(3, 0, 0, 0)	4	3	0	7
Transmissive Symmetric (TS)	U^4	(4, 0, 0, 0)	4	4	0	8
Transmissive Asymmetric (TA)	U^4	(3, 1, 0, 0)	4	4	1	9
	U^4	(2, 2, 0, 0)	4	4	2	10
	U^4	(2, 1, 1, 0)	4	4	3	11
	U^4	(1, 1, 1, 1)	4	4	4	12
	C^3U	(1, 0, 0, 0)	1	1	0	2
	C^2U^2	(2, 0, 0, 0)	2	2	0	4
	C^2U^2	(1, 1, 0, 0)	2	2	1	5
	CD^2U	(1, 0, 0, 0)	3	1	0	4
	CDU^2	(1, 1, 0, 0)	3	2	1	6

4.3. Service Outage

As stated in Section 2.2.1, a service outage can be modeled as an error burst. We use k to denote the number of service items in an outage. Note that k, in effect, measures of the *persistence* (i.e., duration) of the outage. k_{good} and k_{bad} denote the number of correct and incorrect items in the error burst, respectively, such that:

$$k = k_{good} + k_{bad} \tag{33}$$

The error in a service item can be measured with respect to correctness, detectability or symmetry, or combinations of these. The choice of error metric depends on the purpose and specifics of the analysis being performed. Here we use e_i to denote the error in the i-th item of the error burst, with $1 \le i \le k$. The severity of corruption in an outage, denoted s, is the amount of error in the outage, which we define as the sum of the error in the service items.

$$s = \sum_i e_i \tag{34}$$

Let δ be an index for the error levels (i.e., magnitude of the error) in the chosen item error scale, such that $0 \le \delta \le \delta_{max}$, where δ_{max} denotes the largest value on the item error scale. k_δ denotes the number of items in the error burst that have an error of δ. Then, the corruption severity can be expressed as follows.

$$s = \sum_\delta \delta \cdot k_\delta \tag{35}$$

The number of service items in the outage can be expressed as:

$$k = k_0 + \ldots + k_{\delta max} \tag{36}$$

Also:

$$k_{good} = k_0 \tag{37}$$

$$k_{bad} = k_1 + \ldots + k_{\delta max} \tag{38}$$

Alternatively, the corruption severity can be expressed in terms of the occurrence rates of the error levels. To do this, let ϕ_δ denote the rate of item error level δ, such that:

$$\phi_\delta = k_\delta / k \tag{39}$$

Then:

$$s = k \cdot (\sum_\delta \delta \cdot \phi_\delta) \tag{40}$$

The summation in (40) is the mean value of service item error in the outage, denoted e_{mean}:

$$e_{mean} = \sum_\delta \delta \cdot \phi_\delta \tag{41}$$

So:

$$s = k \cdot e_{mean} \tag{42}$$

The operation of the system may be periodic (or cyclic) such that the pattern of service repeats every certain number of items. In that case, we use m to denote the number of expected service items per cycle. The duration of an error burst can be expressed in terms of the number of cycles, denoted Δ, such that:

$$k = m \cdot \Delta \tag{43}$$

By normalizing the corruption severity with respect to the number of items per cycle, we can directly compare outage severity for periodic systems performing the same function, with the same user interface and with the same real-time cycle duration, but operating at different data rates such that they deliver different number of items per cycle. Let s^* denote the data-rate-normalized corruption severity.

$$s^* = s/m = \Delta \cdot (\sum_\delta \delta \cdot \phi_\delta) = \Delta \cdot e_{mean} \tag{44}$$

Finally, to compute the normalized corruptibility and resilience, we need to define a maximum value for the corruption severity, denoted s_{max}. Let k_{max} denote the maximum possible (or expected) number of items in an outage. Then:

$$s_{max} = k_{max} \cdot e_{max} \tag{45}$$

For a cyclic system, the maximum data-rate-normalized corruption severity s^*_{max} is given by:

$$s^*_{max} = \Delta_{max} \cdot e_{max} \tag{46}$$

where Δ_{max} denotes the maximum number of cycles in an outage.

5. System Perturbation Metrics

A perturbation is an outage of the services provided by one or more internal system components due to (transient or permanent) defects in them. These defects mean that, in effect, the defective components are not performing their specified functions and thus act as sources of errors in the system. During a perturbation, it is possible for other components to fail to deliver proper services because of effects propagated out from the defective components. The severity of a perturbation is determined by the service corruption only at the defective components, and non-defective components are treated as if they were performing correctly for the purpose of computing the corruption in a perturbation. Note that it is perfectly legitimate for all system components to become defective during a perturbation. However, depending on the purpose of the analysis being performed, there may be an upper bound constraint on the number of simultaneously defective components.

All the concepts developed to measure disruption corruption directly apply to an outage at an internal component. Let s_j denote the severity of corruption at the j-th component, with $1 \leq j \leq \gamma$, where γ denotes the number of components in the system. s_j is determined as described previously for a disruption. Note that $s_j = 0$ for a unperturbed component. Then, the total perturbation severity s is given by:

$$s = \sum_j s_j \tag{47}$$

To determine the maximum perturbation severity, we simply need to maximize the values of the component corruption severities.

$$s_{max} = \sum_j s_{j,max} \tag{48}$$

6. Final Remarks

In this report, we have proposed a new approach for the assessment of system upset resilience. This approach is based on an analysis of desirable, top-level system attributes interpreted in terms of a generic system service model. Special emphasis is given to the attributes for safety-critical, real-time systems, including distributed systems. Combining the service model with a structural system description model, we develop a stimulus-response concept linking the events internal to the system to the behavior observable at the external system interface. We propose the use of fault injection experiments to stimulate the system and generate a set of corresponding responses. We have proposed a quantitative definition of resilience based on the statistical characterization of this response space. To enable this, we defined a set of service error metrics derived from insight into the classification criteria implicit in the OTH fault model. Throughout, we have tried to develop a general approach that leverages existing system concepts and applies novel and mathematically sound error metrics that are also meaningful in the analysis and design of systems.

This approach will be applied in the analysis of observed HIRF effects in the HEC experiment. To characterize the relation between field disturbances and internal perturbations, we will use the relation described in [87] between the strength of the radiated field and the severity of error bursts in a system, as well as analyses of state data collected during the experiment to identify low-level physical components directly affected by the disturbances. Quantitative analyses of the response space using the metrics

described in this report will enable us to gain insight into the effectiveness of the fault handling mechanisms of the system. In particular, we intend to identify error containment and recovery weaknesses in the network and to propose ways to strengthen the design.

We will also apply the resilience assessment approach in a simulated fault-injection experiment using error-propagation system models. The stimulus will be perturbations with a severity distribution based on the disturbance space characterized in [87]. The models will be validated through review and comparison with the results from the HEC experiment. The purpose of this experiment will be to achieve a thorough understanding of the relation between internal upsets and the propagated effects observable at the external system interface.

We expect that successful application of the proposed upset resilience assessment approach to the analysis of the HEC physical-fault injection experiment and to the simulated-fault injection experiment will affirm the practicality of the approach.

With the insight gained from the analysis of the fault injection experiments, we will then tackle the problem of analysis of self-stabilization-based resilience in systems with required safety and real-time attributes. The first goal in this direction is the design of an advanced version of the ROBUS-2 system [93] with *provable* self-stabilization properties while retaining its foundation on the unified fault-tolerance theory described Miner et al. [61], including the dynamic fault assumptions. We expect that attainment of this goal, especially proving self-stabilization properties, will demand a fundamental breakthrough in the analysis of distributed clock synchronization and membership protocols in the presence of U-type faults.

Appendix A. Proofs for Metrics of Service Item Error

In this appendix it is shown that the service-item error metrics defined in Section 4 satisfy the required mathematical properties for a metric. The properties, introduced in Section 2.5, are restated here for convenience.

- Non-negativity: $d(x, y) \geq 0$
- Symmetry: $d(x, y) = d(y, x)$
- Identity of Indiscernibles: $d(x, y) = 0$ if and only if $x = y$
- Triangle Inequality: $d(x, z) \leq d(x, y) + d(y, z)$

A.1. Error metrics for a single-user service item

We consider separately the distance metrics for the dimensions of correctness and detectability, followed by the case of correctness-and-detectability distance.

A.1.1. Correctness

Recall from Section 4.1 that with respect to correctness, a simplex item x can be either correct or incorrect, and it can be represented by the Boolean variable x_C, which is TRUE for a correct item. Table A.1 shows that the following relation holds.

$$d_C(x,y) = |e_C(x) - e_C(y)| \qquad (A.1)$$

Table A.1: Correctness error and distance for a simplex item

x_C	y_C	$e_C(x)$	$e_C(y)$	$\lvert e_C(x) - e_C(y)\rvert$	$d_C(x, y)$
T	T	0	0	0	0
T	F	0	1	1	1
F	T	1	0	1	1
F	F	1	1	0	0

We use equation (A.1) and Table A.1 to show that $d_C(x,y)$ satisfies the properties of a metric.

- Non-negativity: This is a property of the absolute-value function and applies to $d_C(x,y)$ from equality (A.1). This is also shown by inspection of the last column of Table A.1.
- Symmetry: This is shown by inspection of the first, second and last columns in Table A.1. Also, using (A.1):

  $d_C(x,y) = |e_C(x) - e_C(y)| = |e_C(y) - e_C(x)| = d_C(y,x)$

- Identify of Indiscernibles: This is shown by comparing the last column in Table A.1 with the first and

second columns combined.

- Triangle Inequality: It is well known that the Triangle Inequality applies to the absolute-value function, such that $|a + b| \leq |a| + |b|$ for arbitrary real-valued a and b. If we substitute $e_C(x) - e_C(y)$ for a and $e_C(y) - e_C(z)$ for b, we get:

$$|[e_C(x) - e_C(y)] + [e_C(y) - e_C(z)]| \leq |e_C(x) - e_C(y)| + |e_C(x) - e_C(z)|$$

Using equation (A.1):

$$d_C(x,z) \leq d_C(x,y) + d_C(y,z)$$

A.1.2. Detectability

The proofs that equation (A.2) is valid and that $d_D(x,y)$ satisfies the properties of a metric are identical to the proofs for correctness.

$$d_D(x,y) = |e_D(x) - e_D(y)| \quad (A.2)$$

A.1.3. Correctness and Detectability

In Section 4.1, $d_{CD}(x,y)$ was defined as:

$$d_{CD}(x, y) = d_C(x, y) + d_D(x, y)$$

Using (A.1) and (A.2), we get:

$$d_{CD}(x,y) = |e_C(x) - e_C(y)| + |e_D(x) - e_D(y)| \quad (A.3)$$

Table A.2 is based on the content of Tables 2 and 3 in Section 4.1. Table A.2 shows that the following relation is valid.

$$d_{CD}(x,y) = |e_{CD}(x) - e_{CD}(y)| \quad (A.4)$$

Table A.2: Correctness-and-detectability error and distance for a simplex item

x	y	$e_{CD}(x)$	$e_{CD}(y)$	$\lvert e_{CD}(x) - e_{CD}(y)\rvert$	$d_{CD}(x, y)$
C	C	0	0	0	0
C	D	0	1	1	1
C	U	0	2	2	2
D	D	1	1	0	0
D	U	1	2	1	1
U	U	2	2	0	0

From basic algebra, we know that for natural numbers a and b, the distance between them is given by $d(a, b) = |a - b|$. From Table 3 in Section 4.1, note that the CD error scale is a simple linear scale over the natural numbers from 0 to 2 where the values correspond to the CD errors of C, D, and U. (A.4) is valid simply by definition of the distance between points on that scale.

Equation (A.4) and Table A.2 can be used to show that $d_{CD}(x,y)$ satisfies the properties of a metric. The reasoning is similar to the one used in Section A.1 for the correctness distance of a simplex item.

Note that, by equations (A.3) and (A.4), the following relation is valid.

$$|e_C(x) - e_C(y)| + |e_D(x) - e_D(y)| = |[e_C(x) + e_D(x)] - [e_C(y) + e_D(y)]| \quad (A.5)$$

A.2. Error metrics for a multiple-user service item

We now show that the metrics for multi-user service items satisfy the properties of a mathematical metric. Recall that a multi-user service item X is a vector of simplex service items denoted by x_i for $1 \leq i \leq n$.

$X = (x_1, x_2, \ldots, x_n)$

A.2.1. Correctness

From Section 4.2.1:

$$d_C(X, Y) = d_C(x_1, y_1) + \ldots + d_C(x_n, y_n) \quad (A.6)$$

Using equation (A.1), equation (A.6) can be expressed as:

$$d_C(X, Y) = |e_C(x_1) - e_C(y_1)| + \ldots + |e_C(x_n) - e_C(y_n)| \quad (A.7)$$

We leverage equations (A.6) and (A.7) to show that $d_C(X, Y)$ satisfies the properties of a metric.

- Non-negativity: This is a property of the absolute-value function. It applies to all $d_C(x_i, y_i)$ and it is preserved by addition in (A.6).

- Symmetry: Using (A.7):

  $$d_C(X, Y) = |e_C(x_1) - e_C(y_1)| + \ldots + |e_C(x_n) - e_C(y_n)| = |e_C(y_1) - e_C(x_1)| + \ldots + |e_C(y_n) - e_C(x_n)| = d_C(Y, X)$$

- Identify of Indiscernibles: From Section A.1.1, we know that Identify of Indiscernibles applies to each of the summands in equation (A.6). Because $d_C(x_i, y_i) \geq 0$, $d_C(X,Y) = 0$ requires $d_C(x_i, y_i) = 0$ for all i. Therefore, with respect to correctness, $x_i = y_i$, which means that X = Y with respect to correctness. Reasoning in the opposite direction completes the proof.

- Triangle Inequality: From (A.6) :

  $d_C(X, Z) = d_C(x_1, z_1) + \ldots + d_C(x_n, z_n)$

  Using the proof in Section A.1.1:

  $d_C(x_i, z_i) \leq d_C(x_i, y_i) + d_C(y_i, z_i)$

Thus:

$d_C(X, Z) \leq [d_C(x_1, y_1) + \ldots + d_C(x_n, y_n)] + [d_C(y_1, z_1) + \ldots + d_C(y_n, z_n)]$

$d_C(X, Z) \leq d_C(X, Y) + d_C(Y, Z)$

A.2.2. Detectability

The proof that $d_D(X, Y)$ is a metric is essentially the same as in Section A.2.1 with $d_C(X, Y)$ replaced by $d_D(X, Y)$.

A.2.3. Correctness and Detectability

From equation (15) in Section 4.2.1:

$$d_{CD}(X, Y) = d_{CD}(x_1, y_1) + \ldots + d_{CD}(x_n, y_n) \quad (A.8)$$

Using equation (5) from Section 4.1:

$d_{CD}(X, Y) = [d_C(x_1, y_1) + d_D(x_1, y_1)] + \ldots + [d_C(x_n, y_n) + d_D(x_n, y_n)]$

$d_{CD}(X, Y) = [d_C(x_1, y_1) + \ldots + d_C(x_n, y_n)] + [d_D(x_1, y_1) + \ldots + d_D(x_n, y_n)]$

Using equations (13) and (14) from Section 4.2.1:

$$d_{CD}(X, Y) = d_C(X, Y) + d_D(X, Y) \quad (A.9)$$

Equation (A.9) can be leveraged to show that $d_{CD}(X,Y)$ satisfies the properties of a metric.

- Non-negativity: From Sections A.2.1 and A.2.2, $d_C(X, Y)$ and $d_D(X, Y)$ individually satisfy this property. Their sum in (A.9) preserves this property.

- Symmetry: From Sections A.2.1 and A.2.2, we know that $d_C(X, Y)$ and $d_D(X, Y)$ individually satisfy this property. Then:

  $d_{CD}(X, Y) = d_C(X, Y) + d_D(X, Y) = d_C(Y, X) + d_D(Y, X) = d_{CD}(Y, X)$

- Identify of Indiscernibles: From Sections A.2.1 and A.2.2, we know that $d_C(X, Y)$ and $d_D(X, Y)$ individually satisfy this property. $d_{CD}(X, Y) = 0$ if $d_C(X, Y) = 0$ and $d_D(X, Y) = 0$, which require X = Y with respect to correctness and detectability. Reasoning in the opposite direction, it can be shown that X = Y with respect to correctness and detectability implies $d_{CD}(X, Y) = 0$.

- Triangle Inequality: From Sections A.2.1 and A.2.2, we know that $d_C(X, Y)$ and $d_D(X, Y)$ individually satisfy this property. Thus:

  $d_{CD}(X, Z) = d_C(X, Z) + d_D(X, Z) \leq [d_C(X, Y) + d_C(Y, Z)] + [d_D(X, Y) + d_D(Y, Z)]$

  $d_{CD}(X, Z) \leq d_{CD}(X, Y) + d_{CD}(Y, Z)$

A.2.4. Symmetry

Recall from Section 4.2.2 that for cluster-based symmetry, a multi-user service item X can be represented by a vector X_S of n cluster sizes:

$X_S = (\alpha_1, \ldots, \alpha_n)$,

where α_i denotes the size of the i-th cluster and the α_i elements are sorted by decreasing value such that $\alpha_1 \geq \alpha_2 \geq \ldots \geq \alpha_n$. From Tables 4 and 5 in Section 4.2.2, note that the cluster-based symmetry scale is a simple linear scale over the natural numbers from 0 up to some maximum value, where the values correspond to the symmetry errors of the cluster patterns. (A.10) is valid simply by definition of the distance between points on that scale.

$$d_S(X, Y) = |e_S(X) - e_S(Y)| \quad \text{(A.10)}$$

Equation (A.10) can be used to show that $d_S(X,Y)$ satisfies the properties of a metric.

- Non-negativity: This is a property of the absolute-value function and applies to $d_S(X,Y)$ from equality (A.10).

- Symmetry: Using (A.10):

  $d_S(X,Y) = |e_S(X) - e_S(Y)| = |e_S(Y) - e_S(X)| = d_C(Y,X)$

- Identify of Indiscernibles: From (A.10), $d_S(X, Y) = 0$ implies that $e_S(X) = e_S(Y)$. On our simple linear symmetry error scale, this means that $X_S = Y_S$.

- Triangle Inequality: It is well known that the Triangle Inequality applies to the absolute-value function, such that $|a + b| \leq |a| + |b|$ for arbitrary real-valued a and b. Thus, if we substitute $e_S(X) – e_S(Y)$ for a and $e_S(Y) – e_S(Z)$ for b, we get:

  $|[e_S(X) – e_S(Y)] + [e_S(Y) – e_S(Z)]| \leq |e_S(X) – e_S(Y)| + |e_S(X) – e_S(Z)|$

  Using equation (A.10):

  $d_S(X,Z) \leq d_S(X,Y) + d_S(Y,Z)$

A.2.5. Correctness, Detectability and Symmetry

The total CDS distance between multi-user service items X and Y is given by:

$$d_{CDS}(X, Y) = d_C(X, Y) + d_D(X, Y) + d_S(X, Y) \quad \text{(A.11)}$$

with:

$$d_S(X, Y) = d_S(X_{A,S}, Y_{A,S}) = |e_S(X_{A,S}) - e_S(Y_{A,S})| \quad \text{(A.12)}$$

for approximate agreement, or for exact agreement:

$$d_S(X, Y) = d_S(X_{U,S}, Y_{U,S}) = |e_S(X_{U,S}) - e_S(Y_{U,S})| \quad (A.13)$$

where $e_S(X_{A,S})$, $e_S(Y_{A,S})$, $e_S(X_{U,S})$ and $e_S(Y_{U,S})$ are computed on their respective symmetry error scales. Using equation (A.9), equation (A.11) can be expressed as:

$$d_{CDS}(X, Y) = d_{CD}(X, Y) + d_S(X, Y) \quad (A.14)$$

We know that $d_{CD}(X, Y)$ and $d_S(X, Y)$ satisfy the mathematical properties of a metric. We use this fact to show that $d_{CDS}(X, Y)$ also satisfies the properties of a metric.

- Non-negativity: $d_{CD}(X, Y) \geq 0$ and $d_S(X, Y) \geq 0$ imply that $d_{CDS}(X, Y) \geq 0$.

- Symmetry: $d_{CDS}(X, Y) = d_{CD}(X, Y) + d_S(X, Y) = d_{CD}(Y, X) + d_S(Y, X) = d_{CDS}(Y, X)$

- Identify of Indiscernibles: $d_{CDS}(X, Y) = 0$ implies that $d_{CD}(X, Y) = 0$ and $d_S(X, Y) = 0$. $d_{CD}(X, Y) = 0$ implies that X = Y with respect to correctness and detectability. Using the notation in Section 4.2.1, $X_{CD} = Y_{CD}$. For approximate agreement, this implies that X_A and Y_A have the same number of elements: b + w (see equation (25) in Section 4.2.3). For exact agreement, X_U and Y_U have the same number of elements: w (see equation (31) in Section 4.2.3). This, combined with $d_S(X, Y) = 0$, implies that $X_{A,S} = Y_{A,S}$ for approximate agreement and $X_{U,S} = Y_{U,S}$ for exact agreement. Thus, X = Y with respect to correctness, detectability and symmetry. In the other direction, X = Y with respect to correctness, detectability and symmetry implies $d_{CD}(X, Y) = 0$ and $d_S(X, Y) = 0$, which implies $d_{CDS}(X, Y) = 0$.

- Triangle Inequality: We know that:

  $d_{CD}(X, Z) \leq d_{CD}(X, Y) + d_{CD}(Y, Z)$

  and

  $d_S(X, Z) \leq d_S(X, Y) + d_S(Y, Z)$

  Adding these two inequalities:

  $d_{CD}(X, Z) + d_S(X, Z) \leq [d_{CD}(X, Y) + d_S(X, Y)] + [d_{CD}(Y, Z) + d_S(Y, Z)]$

  Therefore:

  $d_{CDS}(X, Z) \leq d_{CDS}(X, Y) + d_{CDS}(Y, Z)$

References and Bibliography

[1] Antonoiu, Gheorghe; and Srimani, Pradip K.: *Self-Stabilization: A New Paradigm for Fault Tolerance in Distributed Algorithm Design*. Colorado State University, Department of Computer Science, Technical Report CS-97-120, November 1997.

[2] Arlat, Jean; et al.: *Fault Injection and Dependability Evaluation of Fault-Tolerant Systems*. LAAS-CNRS, Research Report 91260, January 1992.

[3] Arora, Anish; and Gouda, Mohamed: *Closure and Convergence: A Foundation of Fault-Tolerant Computing*. IEEE Transactions on Software Engineering, Vol. 19, Issue 11, November 1993, pp. 1015 – 1027.

[4] Avizienis, Algirdas; et al.: *Basic Concepts and Taxonomy of Dependable and Secure Computing*. IEEE Transactions of Dependable and Secure Computing, Vol. 1, No. 1, January – March 2004, pp. 11 – 33.

[5] Azadmanesh, M.H.; and Kieckhafer, R.M.: *New Hybrid Fault Models for Asynchronous Approximate Agreement*. IEEE Transactions on Computers, Vol. 45, No. 4, April 1996, pp. 439 – 449.

[6] Azadmanesh, M.H.; and Kieckhafer, R.M.: *Exploiting Omissive Faults in Synchronous Approximate Agreement*. IEEE Transactions on Computers, Vol. 49, No. 10, October 2000, pp. 1031 – 1042.

[7] Barborak, Michael; and Dahbura, Anton: *The Consensus Problem in Fault-Tolerant Computing*. ACM Computing Surveys, Vol. 25, No. 2, June 1993, pp. 171 – 220.

[8] Barlett, Wendy; and Spainhower, Lisa: *Commercial Fault Tolerance: A Tale of Two Systems*. IEEE Transaction on Dependable and Secure Computing, Vol. 1, No. 1, January-March 2004, pp. 87 – 96.

[9] Belcastro, Celeste M.: *Laboratory Test Methodology for Evaluating the Effects of Electromagnetic Disturbances on Fault-Tolerant Control Systems*. NASA Technical Memorandum 101665, November 1989.

[10] Bell, Ron: *Introduction to IEC 61508*. Proceedings of the 10th Australian Workshop on Safety Critical Systems and Software (SCS '05), Vol. 55, pp. 3 – 12.

[11] Bishop, Matt; et al.: *Resilience is More than Availability*. Proceedings of the 2001 Workshop on Network Security Paradigms (NSPW '11), 2011, pp. 95 – 104.

[12] Bondavalli, A.; and Simoncini, L.: *Failure Classification with Respect to Detection*. Proceedings of the Second IEEE Workshop on Distributed Computing Systems, October 1990, pp. 47 - 53.

[13] Briere, Dominique; Favre, Christian; and Traverse, Pascal: *Electrical Flight Controls, From Airbus A320/330/340 to Future Military Transport Aircraft: A Family of Fault-Tolerant Systems*. The Avionics Handbook, Cary R. Spitzer (Editor), CRC Press LLC, 2001, Chapter 12.

[14] Brown, Simon: *Overview of IEC 61508 – Design of electrical/electronic/programmable electronic safety-related systems*. Computer & Control Engineering Journal, Vol. 11, No. 11, February 2000.

[15] Cholda, Piotr; et al.: *Quality of Resilience as a Network Reliability Characterization Tool*. IEEE Network, March/April 2009, pp. 11 – 19.

[16] Cristian, Flavin: *Understanding Fault-Tolerant Distributed Systems*. Communications of the ACM, Vol. 34, No. 2, February 1991, pp. 57 – 78.

[17] Driscoll, Kevin, et al.: *Byzantine Fault Tolerance, from Theory to Reality*. In Proceedings of the 22nd International Conference on Computer Safety, Reliability and Security (SAFECOMP 2003), 2003, pp. 235 – 248.

[18] Driscoll, Kevin, et al.: *Data Network Criteria Evaluation Report*. Federal Aviation Administration, DOT/FAA/AR-09/27, July 2009.

[19] Dugan, Joanne Bechta; and Trivedi, Kishor S.: *Coverage Modeling for Dependability Analysis of Fault-Tolerant Systems*. IEEE Transactions on Computers, Vol. 38, No. 6, June 1989, pp. 775 – 787.

[20] Dunn, William R.: *Designing Safety-Critical Computer Systems*. Computer, Vol. 36, Issue 11, November 2003, pp. 40 – 46.

[21] Edwards, Nigel; and Rees, Owen: *A Model for Failures in Dependable Systems*. ANSA Project Phase III, March 1994. http://www.ansa.co.uk/ANSATech/94/Primary/114301.pdf Accessed August 21, 2012.

[22] Elks, Carl R.; et al.: *Application of a Fault Injection Based Dependability Assessment Process to a Commercial Safety Critical Nuclear Reactor Protection System*. 2010 IEEE/IFIP International Conference on Dependable Systems & Networks (DSN), July 2010, pp. 425 – 430.

[23] Favre, C.: *Fly-by-wire commercial aircraft: the Airbus experience*. International Journal of Control, Vol. 59, No. 1, 1994, pp. 139 – 157.

[24] Federal Aviation Administration (FAA): *Advisory Circular: Aviation Databus Assurance*. AC No. 20.156, August 4, 2006.

[25] Federal Aviation Administration (FAA): *Advisory Circular: System Design and Analysis.* AC No. 25.1309-1A, June 21, 1988.

[26] Federal Aviation Administration. http://www.faa.gov/nextgen/ Accessed August 21, 2012.

[27] Fetzer, Christof; and Cristian, Flaviu: *Fail-Awareness: An Approach to Construct Fail-Safe Systems*. Real-Time Systems, Vol. 24, 2003, pp. 203 – 238.

[28] Fox, Armando; and Patterson, David: *When does Fast Recovery Trump High Reliability?* Proceedings of the 2nd Workshop on Evaluating and Architecting System Dependability*, 2002.*

[29] Fuller, Gerald L.: *Understanding HIRF – High Intensity Radiated Fields*. Aviation Communications, Inc., Leesburg, VA, 1995, p. 7-2.

[30] *Functional Safety and the IEC 61508: A Basic Guide*. International Electrotechnical Commission, May 2004.

[31] Gartner, Felix C.: *Fundamentals of Fault-Tolerant Distributed Computing in Asynchronous Environments*. ACM Computing Surveys, Vol. 31, No. 1, March 1999.

[32] General Services Administration: *Federal Standard 1037C: Telecommunications: Glossary of Telecommunication Terms*. August 7, 1996. Available at http://www.its.bldrdoc.gov/fs-1037/fs-1037c.htm. Accessed November 17, 2012.

[33] Ghosh, Sukumar; et al.: *Fault-Containing Self-Stabilizing Algorithms*. Proceedings of the Fifteenth Annual ACM Symposium on Principles of Distributed Computing (PODC '96), 1996, pp. 45 – 54.

[34] Ghosh, Sukumar; et al.: *Fault-Containing Self-Stabilizing Distributed Protocols*. Distributed Computing, Vol. 20, No. 1, 2007, pp. 53 – 73.

[35] Gray, Jim; and Siewiorek, Daniel P.: *High-Availability Computer Systems*. IEEE Computer, vol. 39, No. 8, September 1991, pp. 39 – 48.

[36] Grimaldi, Ralph P.: *Discrete and Combinatorial Mathematics: An Applied Introduction*. 3rd Addison-Wesley Publishing Company, 1994.

[37] Halpern, Y. Joseph; and Moses, Yoram: *Knowledge and Common Knowledge in a Distributed Environment*. Journal of the Association for Computing Machinery, Vol. 37, No. 3, July 1990, pp. 549 – 587.

[38] Hess, Richard: *Computing Platform Architectures for Robust Operation in the Presence of Lightning and Other Electromagnetic Threats.* 16th Digital Avionics Systems Conference (DASC), 1997.

[39] Hitt, Ellis F.; and Mulcare, Dennis: *Fault-Tolerant Avionics*. The Avionics Handbook, Cary R. Spitzer (Editor), CRC Press LLC, 2001, Chapter 28.

[40] Hopkins, Albert L., Jr.: *Fault-Tolerant System Design: Broad Brush and Fine Print*. Computer, Vol. 13, No. 3, March 1980, pp. 39 – 45.

[41] *IEC 61508: Functional safety of electrical/electronic/programmable electronic safety-related systems.* Edition 2.0:2010-4, International Electrotechnical Commission.

[42] Jalote, Pankaj: *Fault Tolerance in Distributed Systems*. Prentice Hall, 1994.

[43] Johnson, Barry W.: *An Introduction to the Design and Analysis of Fault-Tolerant Systems*. in Fault-Tolerant Computer System Design, Dhiraj K. Pradhan, Prentice Hall, 1996, pp. 1 – 87.

[44] Johnson, David M.: *A Review of Fault Management Techniques Used in Safety-Critical Avionic Systems*. Progress in Aerospace Sciences, Volume 32, Issue 5, October 1996, pp. 415 – 431.

[45] Kaufman, M. Lori; Bhide, Sanyukta; and Johnson, Barry W.: *Modeling of Common-Mode Failures in Digital Embedded Systems*. Proceedings of the Annual Reliability and Maintainability Symposium, 2000, pp. 350 – 357.

[46] Knight, John C.: *Safety Critical Systems: Challenges and Directions*. Proceedings of the 24th International Conference on Software Engineering (ICSE 2002), May 2002, pp. 547 - 550.

[47] Kopetz, H.: *Fault Containment and Error Detection in TTP/C and FlexRay*. Technical University of Vienna, Research Report 23/2002, August 2002.

[48] Kopetz, H.: *From a Federated to an Integrated Architecture for Dependable Embedded Systems.* September 2004. http://www.dtic.mil/dtic/tr/fulltext/u2/a433397.pdf Accessed August 21, 2012.

[49] Kopetz, H.: *On the Fault Hypothesis for a Safety-Critical Real-Time System.* Proceedings of the Automotive Workshop, 2004.

[50] Kopetz, Hermann: *Real-Time Systems – Design Principles for Distributed Embedded Applications*. Second Edition, Springer, 2011.

[51] Kopetz, Hermann; et al.: *Distributed Fault-Tolerant Real-Time Systems: The MARS Approach*. IEEE Micro, Vol. 9, Issue 1, February 1989, pp. 25 – 40.

[52] Lala, Jaynarayan H.; and Harper, Richard E.: *Architectural Principles for Safety-Critical Real-Time Applications*. Proceedings of the IEEE, Vol. 82, No. 1, January 1994, pp. 25 – 40.

[53] Lala, Jaynarayan H.; Harper, Richard E.; and Alger, Linda S.: *A Design Approach for Ultra-Reliable Real-Time Systems*. Computer – Special Issue On Real-Time Systems, Volume 24, Issue 5, May 1991, pp. 12 – 22.

[54] Ladbury, J.; Koepke, G.; Camell, D.: "Evaluation of the NASA Langley Research Center Mode-Stirred Chamber Facility", NIST Technical Note 1508, January 1999.

[55] Laprie, J.C.: *From Dependability to Resilience*, 38th IEEE/IFIP International Conference On Dependable Systems and Networks, Anchorage, Alaska, June 2008, Sup. Vol., pp. G8-G9

[56] Laprie, Jean-Claude: *Dependability of Computer Systems: from Concepts to Limits*. http://ww2.cs.mu.oz.au/641/papers/laprie-safety.pdf Accessed August 21, 2012.

[57] Laprie, Jean-Claude: *Dependable Computing and Fault Tolerance: Concepts and Terminology*. Twenty-Fifth International Symposium on Fault-Tolerant Computing, Highlights from Twenty-Five Years, June 1995.

[58] Leveson, Nancy G.: *Safeware: System Safety and Computers*. Addison-Wesley, 1995.

[59] Leveson, Nancy; et al.: *Engineering Resilience into Safety-Critical Systems*. Book Chapter. http:// sunnyday.mit.edu/papers/resilience-chapter.pdf Accessed August 21, 2012.

[60] Lundteigen, Mary Ann; and Rausand, Marvin: *Assessment of Hardware Safety Integrity Requirements*. Proceedings of the 30th European Safety, Reliability and Data Association (ESReDA) Seminar, June 2006.

[61] Miner, Paul; et al.: *A Unified Fault-Tolerance Protocol*. Formal Techniques, Modeling and Analysis of Timed and Fault-Tolerant Systems (FORMATS-FTRTFT), Yassine Lakhnech and Sergio Yovine, Editors, Lecture Notes in Computer Science, vol. 3253, Springer, 2004, pp. 167 – 182.

[62] Moore, Jim: *Advanced Distributed Architectures*. The Avionics Handbook, Cary R. Spitzer (Editor), CRC Press LLC, 2001, Chapter 33.

[63] Najiar, Walid; and Gaudiot, Jean-Luc: *Network Resilience: A Measure of Network Fault Tolerance*. IEEE Transactions on Computer, Vol. 39, No. 2, February 1990, pp. 174 – 181.

[64] Nelson, Victor P.: *Fault-Tolerant Computing: Fundamental Concepts*. Computer, Vol. 23, No. 7, July 1990, pp. 19- 25.

[65] Obermaisser, R.: *Fault and Error Containment of Gateways in Distributed Real-Time Systems*. Journal of Software, Volume 4, No. 7, 2009, pp. 686 – 695.

[66] Obermaisser, R.; and Peti, P.: *A Fault Hypothesis for Integrated Architectures*. 2006 International Workshop on Intelligent Solutions in Embedded Systems, June 2006, pp. 1 – 18.

[67] Obermaisser, R.; and Peti, P.: *The Fault Assumptions in Distributed Integrated Architectures*. SAE 07ATC-203, 2007.

[68] Papadopoulos, Gregory M.: *Redundancy Management of Synchronous and Asynchronous Systems*. Fault Tolerant Hardware/Software Architecture for Flight Critical Functions, AGARD-LS-143, 1985.

[69] Paulitsch, Michael; et al.: *Coverage and the Use of Cyclic Redundancy Codes in Ultra-Dependable Systems*. Proceedings of the International Conference on Dependable Systems and Networks (DSN 2005), June-July 2005, pp. 346 – 355.

[70] Pimentel, Juan R.: *Designing safety-critical systems: A convergence of technologies*. 2003 Symposium on Reliable Distributed Systems (SRDS 2003), September 2003. http://paws.kettering.edu/~jpimente/ flexcan/pimentel-ieee Accessed August 21, 2012.

[71] Pimentel, Juan R.: *An Architecture for a Safety-Critical Steer-by-Wire System*. SAE Technical Paper 2004-01-0714, 2004.

[72] Powell, David: *Failure Mode Assumptions and Assumption Coverage*. LAAS-CNRS, Research Report 91462, March 30, 1995.

[73] Powell, David: *Failure Mode Assumptions and Assumption Coverage*. Twenty-Second International Symposium on Fault-Tolerant Computing (FTCS-22), Digest of Papers, July 1992, pp. 386 – 395.

[74] Prew, Roger: *Why the Architecture of Safety Systems Doesn't Matter*. ABB Safety Systems Corporation, 2008. http://www.abb.us/product/us/9AAC115764.aspx Accessed August 21, 2012.

[75] Rierson, Leanna; and Lewis, John: *Criteria for Certifying Databuses on Civil Aircraft*. The 22nd Digital Avionics Systems Conference (DASC '03), Vol. 1, 2003, pp. 1.A.2-1 – 1.A.2-9.

[76] Rousand, Marvin: *Reliability of Safety Systems*. http://www.ntnu.no/ross/slides/chapt10.pdf Accessed August 21, 2012.

[77] Rushby, John: *Bus Architectures for Safety-Critical Embedded Systems*. Presented at EMSOFT 2001: First Workshop on Embedded Software, October 2001, Springer-Verlag Lecture Notes in Computer Science, Vol. 2211, pp. 306 – 323.

[78] Rushby, John: *Critical System Properties: Survey and Taxonomy*. SRI International, Technical Report CSL-93-01, February 1994.

[79] Saridakis, Titos: *Design Patterns for Fault Containment*. Eighth European Conference on Pattern Languages of Programs (EuroPLoP 2003), June 2003.

[80] Siewiorek, Daniel P.; et al.: *Development of a Benchmark to Measure System Robustness*. The Twenty-Third International Symposium on Fault-Tolerant Computing (FTCS-23), Digest of Papers, June 1993, pp. 88 – 97.

[81] Steiner, Wilfried; Paulitsch, Michael; and Kopetz, Hermann: *The TTA's Approach To Resilience After Transient Upsets*. Real-Time Systems, Vol. 32, No. 3, 2009, pp. 213 – 233.

[82] Sterbenz, James P.G.; et al.: *Resilience and survivability in communication networks: Strategies, principles, and survey of disciplines*. Computer Networks, Vol. 54, No. 8, June 2010, pp. 1245 – 1265.

[83] Stevens, Stanley Smith: *On the Theory of Scales of Measurements*. Science, Vol. 103, No. 2684, June 1946, pp. 677 – 680.

[84] Storey, Neil: *Design for Safety*. Towards System Safety: Proceedings of the 7th Safety-Critical Systems Symposium, February 1999.

[85] Suri, N.; Walter, C.J.; and Hugue, M.M.: *Advances in Ultra-Dependable Distributed Systems*. IEEE Computer Society Press, 1995.

[86] Suri, Neeraj; Hugue, Michelle M.; and Walter, Chris J.: *Synchronization Issues in Real-Time Systems*. Proceedings of the IEEE, Vol. 82, Issue 1, January 1994, pp. 41 - 54.

[87] Torres-Pomales, W.: *Analysis of the Radiated Field in an Electromagnetic Reverberation Chamber as an Upset-Inducing Stimulus for Digital Systems*. NASA/TM, 2012, under review.

[88] Torres-Pomales, W.: *Characterization of HIRF Susceptibility Threshold for a Prototype Implementation of an Onboard Data Network*. NASA/TM-2012-217754, August 2012.

[89] Torres-Pomales, W.: *Fault Injection and Monitoring Capability for a Fault-Tolerant Distributed Computation System*. NASA/TM-2010-216834, August 2010.

[90] Torres-Pomales, W.; et al.: *Design of Test Articles and Monitoring System for the Characterization of HIRF Effects on a Fault-Tolerant Computer Communication Systems.* NASA-TM-2008-215322, July 2008.

[91] Torres-Pomales, W.; et al.: *Plan for the Characterization of HIRF Effects on a Fault-Tolerant Computer Communication System*. NASA-TM-2008-215306, May 2008.

[92] Torres-Pomales, Wilfredo; et al.: *Design of the Protocol Processor for the ROBUS-2 Communication System*. NASA/TM-2005-213934, November 2005.

[93] Torres-Pomales, Wilfredo; et al.: *ROBUS-2: A Fault-Tolerant Broadcast Communication System*. NASA/TM-2005-213540, March 2005.

[94] Torres-Pomales, Wilfredo: *Software Fault Tolerance: A Tutorial*. NASA/TM-2000-210616, October 2000.

[95] Traverse, Pascal; Lacaze, Isabelle; and Souyris, Jean: *Airbus Fly-By-Wire: A Total Approach to Dependability*. Building the Information Society, IFIP 18th World Computer Congress, Topical Sessions, Rene Jacquart (Editor), August 2004, pp. 191 – 212.

[96] Triverdi, Kishor S.; Kim, Dong Seong; and Ghosh, Rahul: *Resilience in Computer Systems and Networks*. Proceedings of the 2009 International Conference on Computer-Aided Design (ICCAD '09), 2009, pp. 74 – 77.

[97] Watney, Garth: *A Model-Based Architecture for a Small Flexible Fault Protection System*. AIAA-2009-2028, April 2009.

[98] Wicker, Stephen B.: *Error Control Systems for Digital Communication and Storage*. Prentice Hall, 1995.

[99] Yeh, Y.C.: *Design Considerations in Boeing 777 Fly-By-Wire Computers*. Proceedings of the Third IEEE International High-Assurance Systems Engineering Symposium, November 1998, pp. 64 – 72.

[100] Yeh, Y.C.: *Triple-Triple Redundant 777 Primary Flight Computer*. Proceedings of the IEEE Aerospace Applications Conference, Vol. 1, 1996, pp. 293 – 307.

[101] Yeh, Ying C.: *Safety Critical Avionics for the 777 Primary Flight Controls System*. 20th Digital Avionics Systems Conference (DASC), 2001, Vol. 1, pp. 1C2/1 – 1C2/11.

[102] Yeh, Ying C.: *Unique Dependability Issues for Commercial Airplane Fly by Wire Systems*. Building the Information Society, IFIP 18th World Congress, Topical Sessions, Vol. 156/2004, 2004, pp. 213 – 220.

[103] Youssef, A.; et al.: *Communication Integrity in Networks for Critical Control Systems*. Sixth European Dependable Computing Conference (EDCC '06), October 2006, pp. 23 – 34.

Abbreviations

AC	Alternating Current
ASCII	American Standard Code for Information Interchange
ASIC	Application Specific Integrated Circuit
BIU	Bus Interface Unit
CCP	Controller Coordination Protocol
COTS	Commercial Off-The-Shelf
CPU	Central Processing Unit
CS	Correct Symmetric
CW	Continuous Wave
DC	Direct Current
DSI	Derivation Systems, Inc.
ECR	Error Containment Region
EM	Electromagnetic
EMI	Electromagnetic Interference
FCR	Fault Containment Region
FHSTC	Fine HIRF Susceptibility Threshold Characterization
FPGA	Field Programmable Gate Array
FSM	Finite State Machine
HC	Hardware Configuration
HEC	HIRF Effects Characterization
HFA	Hub Fault Analyzer
HIRF	High Intensity Radiated Field
HSTC	HIRF Susceptibility Threshold Characterization
IEC	International Electrotechnical Commission
IMA	Integrated Modular Architecture
IVHM	Integrated Vehicle Health Management
LUF	Lowest Usable Frequency
Mbps	Mega-bits per second
NFA	Node Fault Analyzer
NIST	National Institute of Standards and Technology
OS	Omissive Symmetric
OTH	Omissive-Transmissive Hybrid (fault model)
PD	Probability Distribution
PE	Processing Element
RC	Reverberation Chamber
RF	Radio Frequency
RMS	Root Mean Square
RMU	Redundancy Management Unit
ROBUS	Robust Bus; also Reliable Optical Bus
RPP	ROBUS Protocol Processor
RSPP	Reconfigurable SPIDER Prototyping Platform
RTCA	Radio Technical Commission for Aeronautics
SBIR	Small Business Innovation Research
SD	Standard Deviation
SDOA	Single Data Omissive Asymmetric
SHM	System Health Monitor
SIM	Stirrer Induced Modulation
SOA	Strictly Omissive Asymmetric
SPIDER	Scalable Processor-Independent Design for Extended Reliability
SUT	System Under Test
TA	Transmissive Asymmetric
TCS	Test Control System

TDMA	Time Division Multiple Access
TS	Transmissive Symmetric
V&V	Validation and Verification
VHDL	VHSIC Hardware Description Language
VHSIC	Very High Speed Integrated Circuit

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
01-01-2013	Technical Memorandum	

4. TITLE AND SUBTITLE

An Approach for the Assessment of System Upset Resilience

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Torres-Pomales, Wilfredo

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

534723.02.02.07.30

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

NASA Langley Research Center
Hampton, VA 23681-2199

8. PERFORMING ORGANIZATION REPORT NUMBER

L-20209

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

National Aeronautics and Space Administration
Washington, DC 20546-0001

10. SPONSOR/MONITOR'S ACRONYM(S)

NASA

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

NASA/TM-2013-217798

12. DISTRIBUTION/AVAILABILITY STATEMENT

Unclassified - Unlimited
Subject Category 33
Availability: NASA CASI (443) 757-5802

13. SUPPLEMENTARY NOTES

14. ABSTRACT

This report describes an approach for the assessement of upset resilience that is applicable to systems in general, including safety-critical, real-time systems. For this work, resilience is defined as the ability to preserve and restore service availability and integrity under stated conditions of configuration, functional inputs and environmental conditions. To enable a quantitative approach, we define novel system service degradation metrics and propose a new mathematical definition of resilience. These behavioral-level metrics are based on the fundamental service classification criteria of correctness, detectability, symmetry and persistence. This approach consists of a Monte-Carlo-based stimulus injection experiment, on a physical implementation or an errorpropagation model of a system, to generate a system response set that can be characterized in terms of dimensional error metrics and integrated to form an overall measure of resilience. We expect this approach to be helpful in gaining insight into the error containment and repair capabilities of systems for a wide range of conditions.

15. SUBJECT TERMS

Avionics; Distributed systems; Fault injection; Hybrid fault model; Resilience; Safety

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			STI Help Desk (email: help@sti.nasa.gov)
					19b. TELEPHONE NUMBER *(Include area code)*
U	U	U	UU	47	(443) 757-5802