NASA/TM-2014-218242

Selecting an Architecture for a Safety-Critical Distributed Computer System with Power, Weight and Cost Considerations

Wilfredo Torres-Pomales
Langley Research Center, Hampton, Virginia

April 2014

The NASA STI Program Office . . . 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 Office plays a key part in helping NASA maintain this important role.

The NASA STI Program Office is operated by Langley Research Center, the lead center for NASA's scientific and technical information. The NASA STI Program Office provides access to the NASA STI Database, the largest collection of aeronautical and space science STI in the world. The Program Office is also NASA's institutional mechanism for disseminating the results of its research and development activities. These results are published 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 that complement the STI Program Office's diverse offerings include creating custom thesauri, building customized databases, organizing and publishing research results ... even providing videos.

For more information about the NASA STI Program Office, see the following:

- Access the NASA STI Program Home Page at ***http://www.sti.nasa.gov***

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

- Fax your question to the NASA STI Help Desk at (301) 621-0134

- Phone the NASA STI Help Desk at (301) 621-0390

- Write to:
  NASA STI Help Desk
  NASA Center for AeroSpace Information
  7121 Standard Drive
  Hanover, MD 21076-1320

NASA/TM-2014-218242

Selecting an Architecture for a Safety-Critical Distributed Computer System with Power, Weight and Cost Considerations

Wilfredo Torres-Pomales
Langley Research Center, Hampton, Virginia

National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23681-2199

April 2014

Acknowledgment

I am grateful for the review comments and suggestions by Dr. Kenneth W. Eure and Cuong Chi (Patrick) Quach from NASA Langley Research Center. I would also like to thank Duane H. Petit and Yuan Chen from NASA Langley Research Center for allowing me access to the reliability analysis tool used in this study. Dr. Patrick Hester of Old Dominion University posed the original challenge that led to the work presented here.

Available from:

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

Abstract

This report presents an example of the application of multi-criteria decision analysis to the selection of an architecture for a safety-critical distributed computer system. The design problem includes constraints on minimum system availability and integrity, and the decision is based on the optimal balance of power, weight and cost. The analysis process includes the generation of alternative architectures, evaluation of individual decision criteria, and the selection of an alternative based on overall value. In this example presented here, iterative application of the quantitative evaluation process made it possible to deliberately generate an alternative architecture that is superior to all others regardless of the relative importance of cost.

Table of Contents

1. Introduction

A computer system is safety critical if its failure may cause or be a contributing factor in injury or death of people, damage or loss of property, or damage to the environment (NASA, 2011). During the development of a system, functional hazard assessments identify the potential system failure conditions and the severity of the consequences. The guiding principle of system safety is to produce a design with an inverse relationship between the severity and probability of occurrence of functional failure conditions (FAA, 1988). In general, it is not possible to satisfy the safety requirements without the use of hardware and/or software redundancy to mitigate the risks of system component failures (Miner, Malekpour, & Torres, 2002) (FAA, 1988). For applications such as air and space vehicles, this need for redundancy to satisfy safety requirements conflicts with other system design objectives such total weight, power consumption and development cost. The preferred system design is one that maximizes safety while minimizing weight, power and cost.

In this paper, we examine this cost-benefit relation to gain insight into the implications and trade-offs of architectural design features for safety-critical computer systems. The approach taken here is to define a representative system design problem and follow the basic steps of a structured decision making process (i.e., determine alternative solutions, identify evaluation criteria, and evaluate the alternatives) (Anderson, Sweeney, Williams, Camm, & Martin, 2011) to generate information that would help a decision maker in choosing the best option from a set of alternative system architectures. A brief overview of function and system safety is given in the next section. This is followed by the problem statement, a review of the relevant literature, and the definition and evaluation of alternative architectures.

2. Function and System Safety

A high-level functional safety assessment with a basic failure model defines three possible functional states: operational (i.e., not failed), failed passive and failed active. A *passive failure* state is a loss-of-function condition in which the function is not being performed. An *active failure* corresponds to a malfunction in which the function is performed incorrectly. Passive and active failures are also known as *omission* and *commission* failures, respectively.

The function performed by a computer system can be modeled as a *service* consisting of a sequence (or flow) of service items (i.e., output updates), each characterized by a value and a time of occurrence. For a worst-case functional safety assessment, we want to determine the highest severity effect of a system failure. For this, we can define the state of the system under an increasingly permissive behavioral classification hierarchy in which a service can be in one of three possible states: *operational*, if only proper service items are being delivered; *failed passive*, if it includes not-failed and failed-passive items; and *failed active*, if there are not-failed, failed-passive and failed-active items. An operational service is a subset of a failed-passive service, which is a subset of a failed-active service. Notice that in this model a failed-active service corresponds to an *arbitrary* failure with no constraints on the behavior exhibited by the system. Intuitively, we want a safety-critical system to either operate properly or not operate at all (i.e., stop), rather than operate in an arbitrary manner. The determination of the system state is based on the level of behavioral constraint that can be guaranteed at a particular point in time.

For some functions, the severity of a failure depends on the duration of the condition. The *time to criticality* after a failure is the minimum time to reach a particular severity level. From a system design perspective, the time to criticality is how much time the system has to restore proper service and prevent a particular failure severity level from being reached. For highly dynamic functions, the time to criticality can be very short (about tens or hundreds of milliseconds) and failure recovery within that time may be unfeasible or require an automated capability. For less dynamic functions, manual recovery by an operator may be adequate.

The outcome of a functional safety assessment is in the form of system integrity and availability requirements (Hasson & Crotty, 1997). *Integrity* is measured as the probability that the system will not be in a failed-active state for a specified mission duration and operational conditions. *Availability* is the probability that the system will be in the operational state at any point in time during a mission of specified duration and operational conditions. Availability during a mission has two components: reliability and recoverability (or maintainability). *Reliability* is the probability that the system will continuously deliver proper service for a specified time duration under specified operational conditions. *Recoverability* is the probability that the system will restore proper service after a failure within a specified time duration under specified operational conditions.

For some functions (e.g., aircraft flight control), the only safe condition is for the system to deliver proper service, as both loss of function and malfunction can be equally catastrophic. In this case, the required probabilities for both availability and integrity will be extremely high. For other functions, the relation between availability and integrity depends on the relative severity of passive and active failure conditions.

Because of the uncertainties in the characterization of probabilities for physical hardware faults and for logical faults (i.e., design errors) in hardware or software, the safety requirements are often stated in terms of the response of the system to a certain number of internal component failures. A *fail-operational* (FO) requirement means that the system shall continue proper service delivery after the failure of an internal component. For a *fail-passive* (FP) requirement, the system service shall transition to a passive state after an internal component failure. For example, a system may be required to remain operational after the first two internal failures and to fail passive after the third failure (i.e., FO/FO/FP). Normally, a safety-critical computer system is required to contain at least one internal component failure, which means that it will be at a minimum fail-passive. Fail-operational and fail-passive conditions are deterministic statements of availability and integrity requirements. For a computer system, a fail-passive requirement means that the failure must be either omissive or commissive but detectable by the service users or monitors, who would then take action to "passivate" the received service.

3. Problem Statement

The design problem consists of selecting a processing and communication architecture for a system with preselected input and output modules. Figure 1 illustrates the top-level system block diagram. Only point-to-point communication links with fail-passive failure modes will be used. As shown in Table 1, there are three different kinds of Input Modules and five different kinds of Output Modules. The Input Modules and Output Modules of every kind are replicated, have the indicated failure modes, and require either one- or two-way communication with the processors. These modules are common to all the system design solutions. For system reliability calculations, a constant failure rate of 10^{-6} failures per hour is assumed for each of the modules.

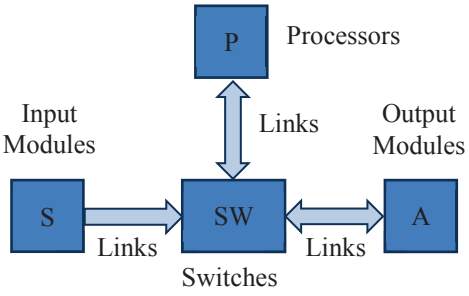

Figure 1: Top-Level Block Diagram of the System

Table 1: Characteristics of Input and Output Modules

Input Modules	**Quantity**	**Failure Mode**	**Communication with the Processors**
S1	3	Active	One-way
S2	2	Passive	One-way
S3	2	Passive	One-way
Output Modules	**Quantity**	**Failure Mode**	**Communication with the Processors**
A1	3	Active	Bidirectional
A2	3	Active	Bidirectional
A3	2	Passive	Bidirectional
A4	2	Passive	Bidirectional
A5	2	Passive	Bidirectional

From a functional perspective, it is assumed that the processors must be able to receive valid data from every type of Input and Output Module in order to compute updates to send to the Output Modules (i.e., each functional output depends on every functional input). It is assumed that all the functions are on demand (i.e., proper operation is desired) for the full duration of the mission. The performance of individual processors and communication switches and links is assumed to be adequate to meet the workload demand without the need for performance-specific architectural features such as parallel processing.

It is assumed that physical faults are statistically independent and that the probability of two or more simultaneously arriving faults, or any single fault affecting multiple components, is negligible. The logical design of the processors is classified as complex and the likelihood of a design error (i.e., a logical fault) in hardware or software is not insignificant. The switches and links are assumed to be logically simple devices and correct by design.

The system attributes to be considered in the evaluation of alternative system architectures include availability, integrity, weight, power, and cost. Availability and integrity are beneficial safety-related qualities; weight, power and cost are detrimental attributes. The minimum probabilistic safety requirements are unavailability $\leq 10^{-9}$ and integrity violation $\leq 10^{-9}$ for a 10-hour mission. The minimum deterministic safety requirements are to preserve availability for one internal physical fault and to preserve integrity for two internal physical faults (i.e., Fail-Operational, Fail-Passive). The system must

preserve availability and integrity for a minimum of one logical fault in either the hardware or software of the processors (i.e., Fail-Operational). We are interested in achieving an optimal balance between weight, power, or cost, and there are no specific constraints or goals for these.

4. Literature Review

To solve the design and decision problem, we need to generate alternative architecture designs and identify an evaluation function that combines the specified selection criteria.

4.1. System Architectures

The availability and integrity requirements can be satisfied with the application of fault-tolerance concepts and techniques. Nelson (1990) and Johnson (1989) offer insightful introductions to fault-tolerant systems. *Error containment* is the prevention of error propagation across defined boundary interfaces. *Error recovery* is the process of preserving or restoring an operational system state. Error containment techniques are the means to achieve fail-passive behavior, and error recovery techniques are used to realize fail-operational responses. Hammett (2002) describes the fault tolerance characteristics of several system architectures. Interestingly, a module with no error detection or recovery features can be conservatively assumed to always fail active as there is no way to guarantee passive failures. Notice that, from a safety standpoint, a fail-operational response satisfies a fail-passive requirement. Also, note that error recovery implies error containment, and this means that, in general, error recovery requires a higher degree of redundancy and redundancy management activity than error containment.

Black (2005) offers an interesting summary of the amount of redundancy needed for fail-operational response as a function of the failure modes of the processors and the communication system. Table 2 (next page) shows that processor redundancy for fail-operational response is directly dependent on the failure modes of the processors and the communication system. As would be expected, the architecture level design is simpler when the components exhibit benign fail-passive behavior. Achieving passive component failure modes requires local component-level redundancy. Thus, there is a trade-off between component-level and architecture-level redundancy and complexity. Powell (1992) showed that although conservative assumptions at the architecture level about component failure modes might seem intuitively advantageous from a safety (and integrity) perspective, the resultant increase in architecture-level complexity may actually lower the reliability (and availability) of the system. Lala (1991) observed that redundancy by itself only guarantees an increase in the arrival rate of faults and that a carefully planned redundancy management strategy is necessary to achieve an increase in system dependability compared to a non-redundant system of the same functionality.

Meyer (1975) showed that the probability of fault detection for a component can be made equal to unity only if the detector is as complex, in terms of the number of states, as the component being monitored. A self-checking pair configuration (Nelson, 1990; Hammett, 2002) consisting of two identical components connected to a simple comparison-based error detector is a common arrangement to achieve fail-passive response to physical faults. For communication system components, including links and switches whose only function is to transport data messages between communicating entities, it is common practice to rely on message encoding and other forms of state information embedded in the messages (such as a message sequence number) to achieve fail-passive behavior. Such approaches are in use even for safety-critical applications regardless of the fact that they violate the assumptions in the calculations of communication service integrity. Paulitsch, et al. (2005) recommend more technically sound replication techniques for intermediate communication stages like switches in order to guarantee

compliance with communication integrity and availability requirements.

Table 2: Required processor redundancy for fail-operational fault response as a function of processor and communication failure modes

Number of Faults	Processor Failure Mode	Communication Failure Mode	Minimum Processor Redundancy
1	Passive	Passive	2
	Active	Passive	3
	Active	Active	4
2 Sequential	Passive	Passive	3
	Active	Passive	4
	Active	Active	5

Fault-tolerance techniques against logic design errors are similar to the techniques for physical faults. The main difference is that failure independence and non-coincidence for logic components are predicated on design dissimilarity. Torres-Pomales (2000) describes a number of fault-tolerant architectural configurations applicable to logical component faults. The development assurance level (DAL) of a logic component is a qualitative measure of the required development rigor, which is assumed to be negatively correlated with the likelihood of residual design errors and positively correlated with the cost of development. ARP4754A (SAE, 2010) describes an approach for DAL assignment that takes into consideration fault severity mitigation using architectural features.

4.2. Design Evaluation

In evaluating alternative system architectures, we seek an explicit and quantitative method that can measure the relative merit of the alternatives, rank them, and identify the best choice. The system evaluation problem can be divided in two parts: evaluation of individual criteria and aggregation of criteria into an overall value metric.

The design evaluation criteria (or objectives) include availability, integrity, weight, power, and cost. All these system attributes can be obtained from attributes of the lower-level components and models of the system architecture. Availability and integrity can be evaluated using Reliability Block Diagrams, Fault Trees and Markov Models (Geist & Trivedi, 1990; Reibman & Veeraraghavan, 1991; Butler & Johnson, 1995; NASA, 2002). Representative values for component fault rate, weight, power, and cost can be obtained from vendors and published sources, including (Hodson, et al., 2011). Laprie, et al. (1990) present a life-cycle cost model for various software-fault tolerant configurations. We will assume that all hardware components use commercially available (COTS) products. Notice that all the design alternatives must satisfy minimum safety requirements.

Buede (2009) and Callopy (209) list a number multi-attribute value analysis methods, including analytical hierarchy process (AHP), percentaging, fuzzy algorithm, quality function deployment (QFD), and Pugh Matrix. According to Buede, these methods are either not well founded or other analytical concerns have been raised about them. Marler and Arora (2004) surveyed the multi-objective optimization methods and classified them based on the articulation of preferences among objectives: a priori, a posteriori, and no articulation. Only a-priori methods are of interest here, as we want our metric of overall value to reflect the relative importance of the system attributes. A-priori methods include the

weighted global criterion method, weighted sum method, lexicographic method, weighted min-max method, and exponential weighted criterion method among others. The most common multi-objective optimization method is the weighted sum method, in which the global objective function is the sum of the weighted value of the individual criteria. This is the value function we have chosen to evaluate the alternative system architectures. The formula for the overall value is:

$$v(\mathbf{x}) = \sum_{i=1}^{n} w_i v_i(x_i)$$

where $\mathbf{x}$ is the vector of individual system evaluation criteria (or objectives), x_i is the absolute value the i-th objective, v_i is the relative value or utility of x_i, and w_i measures the relative importance of objective x_i. The w_i weights are commonly normalized to sum to 1, and the v_i functions are defined on a common normalized range from 0 to either 1, 10, or 100.

Buede (2009) describes the linear, concave, convex, and S-curve general forms of the value functions $v_i(x_i)$ that represent the change in utility of x_i as it varies in the range from the threshold to the goal value specified by the system requirements.

Meya and Swoy (1992) describe two different utility (i.e., value) functions: endpoint and ratio. The endpoint function is similar to Buede's linear value function and consists of a linear interpolation between the utilities assigned to the endpoints of x_i. For the ratio function, the utility v_i of a beneficial attribute is defined as $\log(x_i/x_{i,min})$, where $x_{i,min}$ denotes the minimum value of x_i. The value of a detrimental attribute is defined as $\log(x_{i,max}/x_i)$, where $x_{i,max}$ is the maximum value of x_i. The ratio function has the advantage that it scales x_i based on order of magnitude relative to the actual range of values of x_i and no other subjective utility function is needed.

If the system requirements do not specify the threshold or the goal for some attribute x_i, the endpoints of x_i could be taken as the minimum and maximum over all the alternate system designs in order to enable a meaningful and balanced evaluation using the value curves described by Buede (2009).

Meya and Swoy (1992), Buede (2009), and Callopy (2009) acknowledge the hierarchical nature of value in the sense that the top-level value of a system can be expressed as a composition of increasingly refined evaluation criteria. Meya and Swoy (1992) refer to this as an attribute tree. From this perspective, the value of a composite criterion is defined in terms of the aggregate value of its sub-criteria.

Callopy (2003) considered the impact of technical risk on the overall value of a system. Technical risk is the uncertainty in the performance of a system or its components. System value risk is related to the sensitivity of the objective hierarchy to uncertainty in lower objectives. The topic of risk is outside the scope of this paper and is not part of the system evaluation criteria. However, a simple sensitivity analysis will be performed.

5. Alternative Architectures

Looking at Figure 1, the Input and Output Modules are given, and the design problem is to determine the configuration for the processors and the communication network. To satisfy the deterministic safety requirements, the processors and the network must incorporate redundancy and fault tolerance capabilities for error containment and recovery. It is given that the links have passive failure modes. The switches

are logically simple devices, which means that they are assumed not to have logical design errors and to fail only due to physical faults. Thus, simple replication-based redundancy can be used to realize switch-fault tolerance. The processors are modeled as consisting of an application layer, a processing platform layer including the computation hardware and an operating system, and a communication end system element to handle the interaction with the network. The processor hardware and software are assumed to be complex, and thus, they can fail due to residual design errors. The hardware components can also fail due to physical faults. The end systems are assumed to be logically simple components. There are two main approaches to deal with design errors: acceptance tests and diversified design (Laprie, Arlat, Beounes, & Kanoun, 1990). Based on the results presented by Hammet (2002) and Meyer (1975), the only way to ensure near perfect error detection coverage and containment is to have a detector that is as complex as the monitored system. This essentially demands the use of full component replicas, or in the case of design errors, component variants with similar functionality but dissimilarity in requirements, design, or implementation. Thus, dissimilar redundancy will be used to realize fault tolerance for the applications and the processing platform.

Laprie, et al. (1990) and Hodson (2011) identify two basic forms of redundancy management approaches: voting based and pairwise-comparison based. In both cases, it is assumed that the redundant replicas or variants receive the same input sequence and are required the produce the same or equivalent outputs. The voting approach uses a majority voter to decide the final correct output for a set of replicas or variants (Torres-Pomales, 2000). For the pairwise comparison approach, pairs of replicas or variants are compared and only pairs with agreeing outputs are assumed to be correct. The final output for a configuration with pairwise comparison is taken from one of these agreeing pairs.

Four different fault tolerant configurations were considered in the generation of alternative architectures. For redundancy with (identical) replicas to tolerate only physical faults, there is the *voted replica* (VR) configuration in which a voter is used to decide the final output from multiple replicas, and the *self-checking replica* (SR) configuration consisting of pairwise comparisons and the logic for selecting a good output from a pair of agreeing replicas. For redundancy with dissimilar variants, the *voted variant* (VV) configuration uses voting as the decision logic, and the *self-checking variant* (SV) configuration uses pairwise comparison and selection. Thus, the VR and SR configurations are applicable to the network switches, and the VV and SV configurations are relevant to the applications and the processing platform.

For the definition of alternative system architectures, it was assumed that different physical faults or design errors are triggered (i.e., cause the generation of data errors) sequentially such that, at any time, the system has to deal with at most one active component failure and can recover from it before another fault is triggered. This is a critical assumption that is given in the problem statement and is consistent with the definition of deterministic safety requirements.

In a voting configuration, the individual redundant elements can be fail-active as the voter acts to both contain and mask component failures. The realization of a fail-operational, fail-passive response with voting fault tolerance requires a minimum redundancy of three, such that the voter can mask the first component failure (i.e., fail-operational response) and the system gets reduced to two operational components, and then on the second failure, the voter detects a discrepancy and stops the generation of outputs (i.e., fail-passive response).

For a self-checking configuration, a fail-operational, fail-passive response requires two redundant pairs. The first component failure causes a redundant pair to fail passive and the output decision logic to select the other pair, thus realizing the fail-operational response. If the second component failure affects

the second redundant pair, the decision logic will detect the discrepancy and stop the generation of output as required for fail-passive response. Note that voting and selection of redundant data are performed at the receivers of the data rather than at the sources. This ensures failure independence between the redundant sources and the decision logic for redundancy management.

Table 3 shows the specifications of the basic components used for the system architectures. The cost, weight and power specifications were obtained from equipment manufactures' websites for representative off-the-shelf devices. The costs for applications and processors are for high-assurance designs, and the values were determined using the cost multipliers for step increases in development assurance levels given in (HighRely, Inc., 2009):

Cost(DAL D) = 1.05 x Cost(DAL E, not safety relevant)

Cost(DAL C) = 1.30 x Cost(DAL D)

Cost(DAL B) = 1.15 x Cost(DAL C)

Cost(DAL A, highest safety criticality) = 1.05 x Cost(DAL B)

The hardware-software cost ratios given by Dörenberg (1997, p. 33) and Spitzer (2007, pp. 14-2) were used to estimate the cost of software:

Software cost = 3.35 x Hardware cost

For evaluating the architectures, 300-ft point-to-point data communication links were assumed for all the connections to the network switches. The failure rate for the software applications was taken from a paper by Bleeg (1988) on fly-by-wire architectures. All other failure rates were taken from the report by Hodson, et al. (2011) on avionics architectures.

Table 3: Attributes of system components

Component	Cost ($)	Weight (lbs)	Power (W)	Failure Rate (failures per hour)
Application (Single variant)	85,158.94	0.0	0.0	1.00×10^{-7}
Processor (Hardware and Software)	24,723.56	2.0	38.0	3.33×10^{-5}
End System	7,000.00	0.5	6.5	5.04×10^{-6}
Switch	25,000.00	8.0	28.0	5.04×10^{-6}
Link (300 ft)	500.00	6.6	0.0	1.30×10^{-6}

Table 4 lists the alternative architectures generated for evaluation. The abbreviations VV, SV, etc. correspond to the fault tolerant configurations used for the corresponding components as indicated in the table. The subscripts x,y indicated the number of replicas or variants per redundant channel and the number of channels in the configuration. For example, for $VV_{1,4}$ there was one variant per channel and 4 channels in the configuration, for a total of 4 variants of the corresponding component. Architecture SA7 had 4 processing channels, each with one processor variant and 3 application variants with a voted configuration. On architecture SA7, the same set of application variants was replicated on each processing channel to take advantage of the low failure rate of individual application variants and to limit the total cost of the applications to only three variants.

In each configuration, every processor, Input Module and Output Module had one bidirectional link to each switch.

Table 4: Alternative System Architectures

Architecture	Applications	Processing	Switches
SA1	$VV_{1,4}$	$VV_{1,4}$	$VV_{1,4}$
SA2	$VV_{1,4}$	$VV_{1,4}$	$SR_{2,3}$
SA3	$SV_{2,3}$	$SV_{2,3}$	$VV_{1,4}$
SA4	$SV_{2,3}$	$SV_{2,3}$	$SR_{1,3}$
SA5	$VV_{3,1}$	$SV_{2,3}$	$VV_{1,4}$
SA6	$VV_{3,1}$	$SV_{2,3}$	$SR_{2,3}$
SA7	$VV_{3,1}$	$VV_{1,4}$	$SR_{2,3}$

The architectures were modeled using reliability block diagrams. All faults were assumed permanent, and fault recovery was assumed to be instantaneous until the point of total system failure due to exhaustion of operational resources. With this assumption, the evaluation of availability is simply an evaluation of reliability.

Reliability was evaluated using a commercial off-the-shelf tool (PTC, 2013). All the architectures required a higher degree of redundancy to meet the probabilistic safety requirements than what was strictly needed to meet the deterministic requirements. Furthermore, every model was too complex to complete the full-system reliability analysis without running out of memory resources. The main source of model complexity comes from the need to accurately describe the conditions under which proper operation is preserved as link failures occur. Instead of computing the reliability for whole systems, multiple overlapping sections of the models were analyzed separately to achieve some confidence that the overall design met the availability and integrity requirements. An implication of this is that the actual probabilities for these architectures are not available. This was taken into consideration in the evaluation and comparison of the architectures as described in Section 8. Section 6 provides additional information about the reliability analyses.

The structure of the fault tolerant configurations used in the alternative architectures, combined with the assumption of sequential fault triggering, ensures that satisfying the reliability requirement will also satisfy the integrity requirement.

6. Reliability Analysis

Reliability block diagrams (RBD) were used to compute the reliability of the alternative architectures. RBD models capture the inter-components dependencies that must be satisfied for the system to remain in an operational state. A *series* configuration with two or more components means that all of those components must remain operational in order for the system to remain operational. A basic 1-of-m *parallel* configuration means that at least one component must remain operational. In general, an n-of-m parallel configuration represents a dependence relation such that the system will remain operational if at least n of the m redundant paths (or channels) are operational.

Figure 2 shows the RBD model for architecture SA7. The RBD is composed of sections for the

various architectural components: applications (App); processing (Proc); switches (SW); S1, S2, and S3 input modules; and A1, A2, A3, A4, and A5 output modules. Most of the complexity in the model is due to the need to accurately capture the conditions that guarantee successful communication between network terminals (i.e., processors and input and output modules). Four operational internal data flows are required in order for the system to remain operational: from input modules to processors, from output modules to processors, from processors to processors, and from processors to output modules. As links fail, only certain combinations of connections between the terminals and the switches can guarantee successful end-to-end dataflow. As the system is intended for safety-critical applications, the reliability calculation (and thus the RBD) must be made with respect to conditions for which an operational state can be guaranteed (i.e., must do a pessimistic, worst-case analysis).

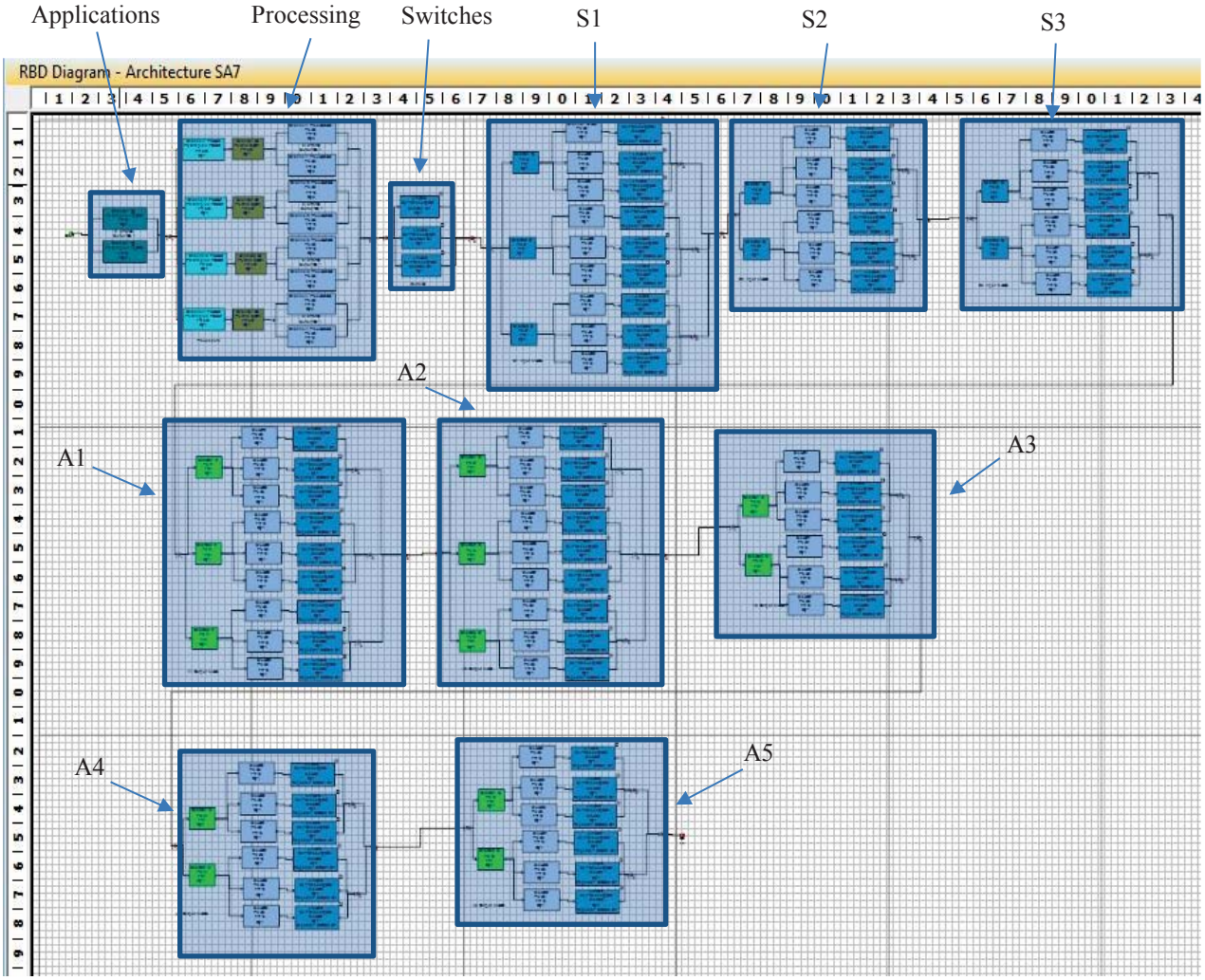

Figure 2: Reliability Block Diagram for Architecture SA7

The reliability analysis tool was unable to complete the analysis of a full architecture before running out of memory resources. Because of this, it was decided instead to compute the reliability for sections of

the model. Table 5 shows the calculated unreliabilites (= 1.0 – reliability) for each of the alternative architectures. Only the mantissas are given, and a factor of 10^{-10} is implicit. The switches had an insignificant contribution to the overall system unreliability. It is likely that most of the unreliability contribution comes from the processors and the valid patterns of degradation of the network paths.

Table 5: Unreliabilities for Various Sections of the Alternative Architectures ($x10^{-10}$)

Model Section	SA1	SA2	SA3	SA4	SA5	SA6	SA7
App-Proc	3.32	3.03	4.57	4.34	4.56	4.34	3.04
SW	0.00	0.01	0.00	0.01	0.00	0.01	0.01
App-Proc-SW	3.33	3.04	4.57	4.34	4.56	4.35	3.05
SW-S_{all}	5.01	5.01	5.01	5.01	5.01	5.01	5.01
SW-A_1	3.01	3.01	3.01	3.01	3.01	3.01	3.01
SW-A_2	3.01	3.01	3.01	3.01	3.01	3.01	3.01
SW-A_3	1.01	1.01	1.01	1.01	1.01	1.01	1.01
SW-A_4	1.01	1.01	1.01	1.01	1.01	1.01	1.01
SW-A_5	1.01	1.01	1.01	1.01	1.01	1.01	1.01

7. Evaluation Criteria

The original intent was to compare the architectures based on the criteria of availability, integrity, cost, weight and power. However, because it was not possible to get the actual availability and integrity values, these were not used in the overall evaluation. Instead, the availability and integrity requirements were simply taken as constraints that must be satisfied by the architecture, and any excess availability or integrity beyond the minimum constraints were not taken into account as beneficial to the overall value of the architectures. Thus, only the detrimental attributes of cost, power, and weight were considered in the overall evaluation.

The cost of a fault-tolerant configuration must take into consideration the life cycle cost, including requirements, specification, design, implementation, validation, verification, and maintenance. Laprie, et al. (1990) present a simple cost model for dissimilar redundancy configurations. The cost model gives the maximum, minimum, and average cost multipliers for various fault tolerant configurations. That model was applied by using the average multiplier for variants and the minimum multiplier for replicas. Table 6 shows the multipliers used to estimate the cost of redundant configurations.

Table 6: Multiplicative Cost Factors for Redundant Configurations

Redundancy	3	4	6
Voted Variants	2.25	3.01	--
Self-Checking Variants	--	3.01	4.63
Voted Replicas	1.78	2.24	--
Self-Checking Replicas	--	2.24	3.71

The value (or utility) function for each criterion was defined over the range of all the alternatives as the problem statement set no constraints or goals for any of the detrimental criteria. The value functions

were defined as linearly decreasing from 1.0 at the minimum value of the criterion to 0.0 at the maximum value of the criterion. Let $x_{i,min}$ and $x_{i,max}$ denote the minimum and maximum values for the i-th criterion. The value of x_i is given by:

$$v_i(x_i) = (x_{i,max} - x_i)/(x_{i,max} - x_{i,min}).$$

The criteria were divided in two groups: weight and power, and cost. This is consistent with the typical program objectives of performance and cost (as well as schedule) and allows the examination of the trade-off between performance and cost for the set of alternative architectures. The relative importance of cost versus power and weight was varied from 0.0 to 1.0, and the relative importance between power and weight was held constant at 0.5 each.

8. Evaluation of Alternatives

Figures 3, 4, and 5 show the total cost, power, and weight for the alternative architectures. The general cost patterns are that self-checking switches increase the cost of an architecture by about 5%, and using both self-checking processors and switches increases the cost by about 42%. Self-checking switches also increase the power by about 17%, and self-checking processors and switches increase the power consumption by about 43%. The upside for self-checking switches is that they tend to reduce the weight by about 20%, but the use of self-checking processors and switches reduces the total weight only 3% more (23% total).

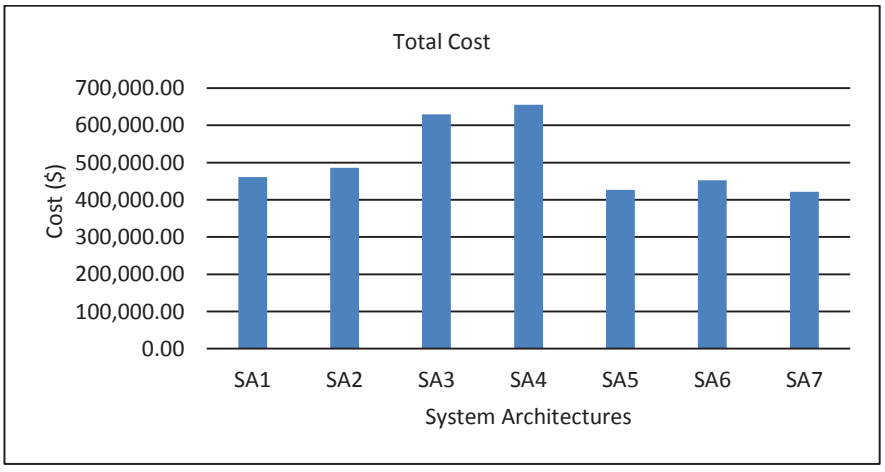

Figure 3: Total Cost for Alternative Architectures

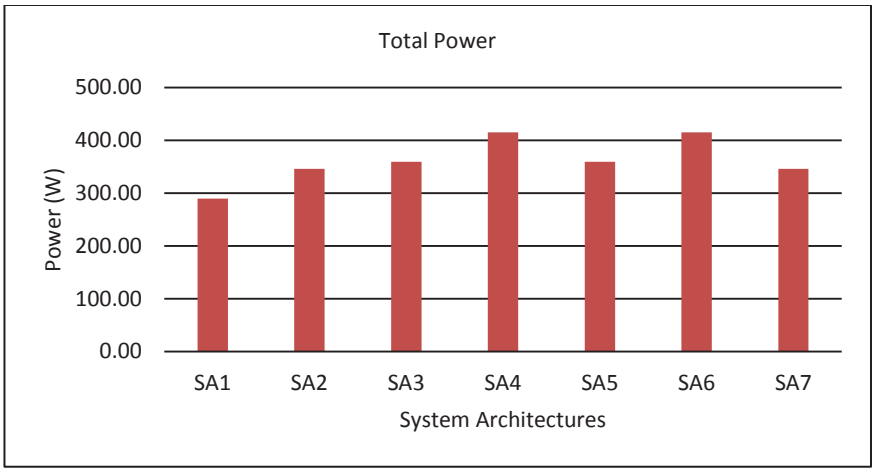

Figure 4: Total Power for Alternative Architectures

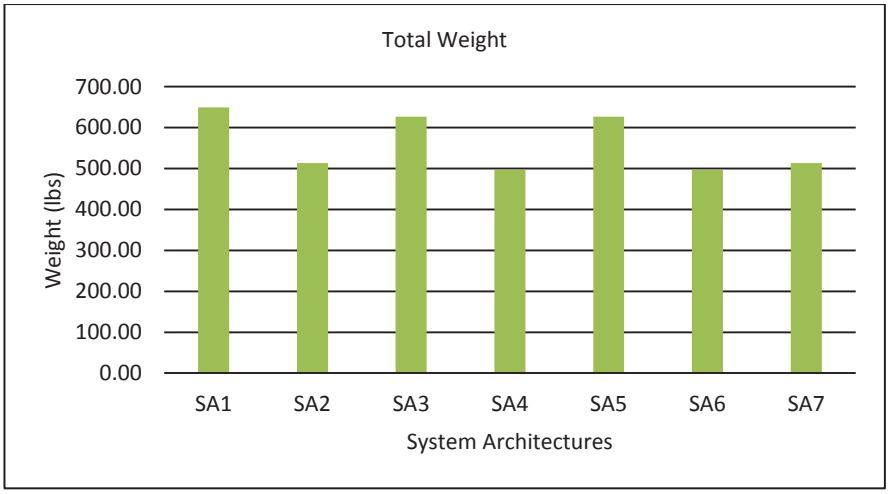

Figure 5: Total Weight for Alternative Architectures

Figure 6 shows the overall value of the configurations as the relative importance of cost varies from 0.0 (not important) to 1.0 (only important criterion). Architectures SA3 and SA4 with self-checking applications and processors decrease in value as the importance of cost increases, while all other architectures show an increase in value. An interesting result is that the value of SA2 remains constant. This is an indication of a good balance between voting configurations for applications and processors and a self-checking configuration for the switches. Architecture SA7, which was deliberately added to the set of alternatives after examining the results for the other architectures, takes this balance one step further by using the $VV_{3,1}$ application configuration from SA5 and SA6 to reduce the cost of the applications using voting of three variants within each processing channel and replicating the application variants on each channel. This application configuration increases the reliability and integrity of each channel. The drawback of the $VV_{3,1}$ application configuration is that the minimum required processing capacity per channel is the largest of all. This is not reflected in any of the criteria or the overall value calculation.

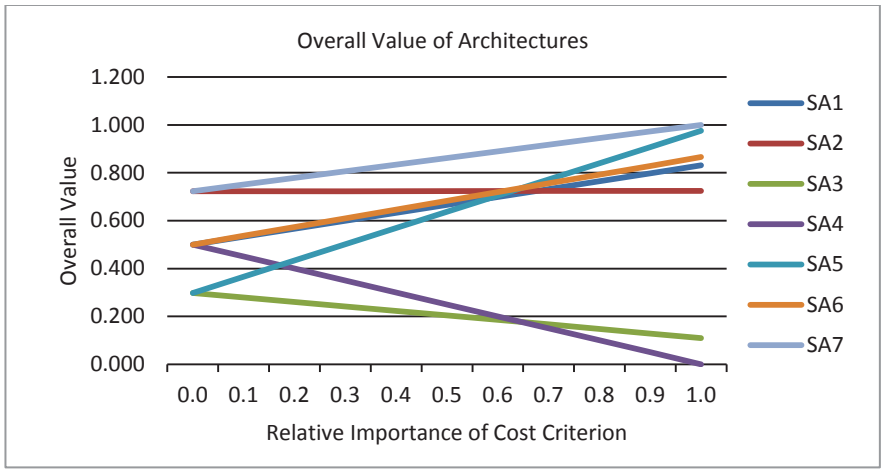

Figure 6: Overall Value for Alternative Architectures

Figure 7 shows the change in the unreliability (= 1.0 – reliability) for the combination of the processors and switches of architecture SA7 when the failure rates of the end system, processing, and switch components are reduced or increased by a factor of 10. The Input and Output Modules are not included in this analysis due to the limitations of the reliability analysis tool. Figure 7 shows that the failure rate of the processing components is the most important determinant of system reliability for this architecture.

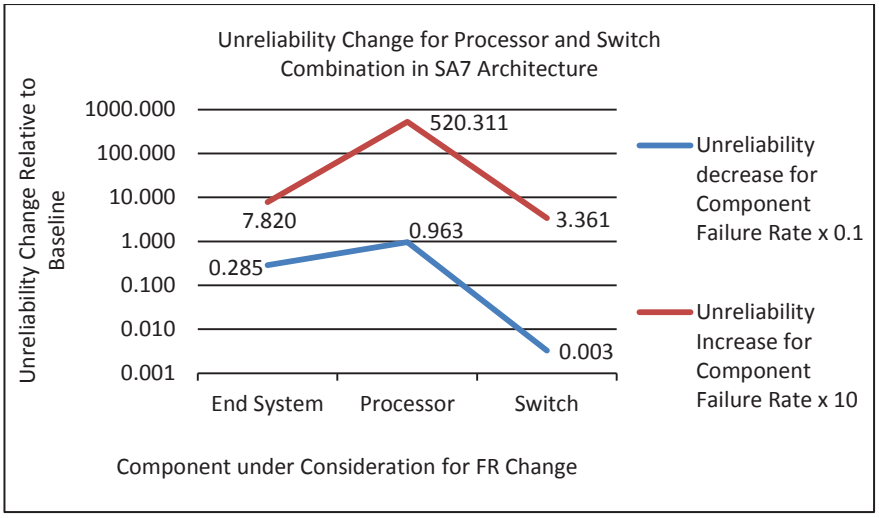

Figure 7: Sensitivity of System Reliability to Component Failure Rate

9. Concluding Remarks

An example application of multi-criteria decision analysis has been presented for the selection of an architecture for a safety-critical distributed computer system with power, weight, and cost considerations. Seven alternative architectures were produced in an iterative process that leveraged the overall multi-criteria quantitative evaluation to deliberately generate an architecture that is superior to all others. A reliability sensitivity analysis of the selected architecture showed that the failure rate of the processing platform, including the hardware and operating system, is the most important component that should be targeted for further development to enhance the system reliability.

This example was a high-level analysis performed on abstract system models that did not include details that could be significant in the validity of the analysis for an actual realization of the system. The models for cost, weight, and power would normally be refined and validated before making a final architecture selection. Additionally, the reliability models would also need to be simplified (to enable a full analysis) and validated. Finally, the trade-off between using standard off-the-shelf components and high quality components would need to be examined to determine if any architectural simplification achievable with high quality components can be justified in a multi-criteria decision process.

References

Anderson, D. R., Sweeney, D. J., Williams, T. A., Camm, J. D., & Martin, K. (2011). *An Introduction to Management Science: Quantitative Approaches to Decision Making, Thirteenth Edition.* USA: South-Western Cengage Learning.

Black, R., & Fletcher, M. (2005, December). Next Generation Space Avionics: Layered System Implementation. *IEEE Aerospace and Electronic Systems Magazine*, pp. 9 - 14.

Bleeg, R. (1988). Commercial Jet Transport Fly-By-Wire Architecture Considerations. *9th AIAA/IEEE Digital Avionics Systems Conference.* AIAA/IEEE.

Buede, D. M. (2009). *The Engineering Design of Systems: Models and Methods.* John Wiley & Sons.

Butler, R. W., & Johnson, S. C. (1995). *Techniques for Modeling the Reliability of Fault-Tolerant Systems With the Markov State-Space Approach.* NASA Reference Publication 1348, National Aeronautics and Space Administration.

Collopy, P. (2003). Balancing Risk and Value in System Development. *AIAA Space 2003 Conference & Exposition.* AIAA Paper 2003-6376: American Institute of Aeronautics and Astronautics.

Collopy, P. D. (2009). Aerospace System Value Models: A Survey and Observations. *AIAA SPACE 2009 Conference & Exposition.* AIAA Paper 2009-6560: American Institute of Aeronautics and Astronautics.

Dörenberg, F. M. (1997, February). *Integrated and Modular Systems for Commercial Aviation.* Retrieved from http://www.nonstopsystems.com/radio/article-IMA97.pdf

FAA. (1988). *AC 25.1309-1A - System Design and Analysis.* Federal Aviation Administration.

Geist, R., & Trivedi, K. (1990, July). Reliability Estimation of Fault-Tolerant Systems: Tools and Techniques. *Computer, 23*(7), 52 - 61.

Hammett, R. (2002, April). Design by Extrapolation: An Evaluation of Fault-Tolerant Avionics. *IEEE Aerospace and Electronic Systems Magazine*, pp. 17 - 25.

Hasson, J., & Crotty, D. (1997). Boeing's safety assessment processes for commercial airplane designs. *AIAA/IEEE Digital Avionics Systems Conference (16th DASC)*, *1*, pp. 4.4 - 1-7.

HighRely, Inc. (2009). DO-178B Costs Versus Benefits. *Online whitepaper*. Retrieved from http://highrely.com/whitepapers.php

Hodson, R. F., Chen, Y., Morgan, D. R., Butler, A. M., Schuh, J. M., Petelle, J. K., . . . Nguyen, H. D. (2011). *Heavy Lift Vehicle (HLV) Avionics Flight Computing Architecture Study.* NASA/TM-2011-217168, National Aeronautics and Space Administration.

Johnson, B. W. (1989). *The Design and Analysis of Fault Tolerant Digital Systems.* USA: Addison-Wesley.

Lala, J. H., Harper, R. E., & Alger, L. S. (1991, May). A Design Approach for Ultrareliable Real-Time Systems. *Computer, 24*(5), 12 - 22.

Laprie, J.-C., Arlat, J., Beounes, C., & Kanoun, K. (1990, July). Definition and analysis of hardware- and software-fault-tolerant architectures. *Computer, 23*(7), 39 - 51.

Marler, R., & Arora, J. (2004, April). Survey of multi-objective optimization methods for engineering. *Structural and Multidisciplinary Optimization, 26*(6), 369 - 395.

Meya, R. D., & Swoy, L. F. (1992). A Method of Evaluating Candidate Spacecraft Data System Architectures. *IEEE/AIAA 11th Digital Avionics Systems Conference* (pp. 457 - 463). Institute of Electrical and Electronics Engineers.

Meyer, J. F., & Sundstrom, R. J. (1975, May). On-Line Diagnosis of Unrestricted Faults. *IEEE Transactions on Computers, C-24*(5), 468 - 475.

Miner, P. S., Malekpour, M., & Torres, W. (2002). A Conceptual Design for a Reliable Optical Bus (ROBUS). *The 21st Digital Avionics Systems Conference. 2*, pp. 13D3-1 - 13D3-11. Institute of Electrical and Electronics Engineers.

NASA. (2002). *Fault Tree Handbook with Aerospace Applications, Version 1.1.* National Aeronautics and Space Administration.

NASA. (2011). *NASA System Safety Handbook: Volume 1, System Safety Framework and Concepts for Implementation.* NASA/SP-2010-580, National Aeronautics and Space Administration.

Nelson, V. P. (1990, July). Fault-Tolerant Computing: Fundamental Concepts. *Computer, 23*(7), 19 - 25.

Paulitsch, M., Morris, J., Hall, B., Driscoll, K., Latronico, E., & Koopman, P. (2005). Coverage and the Use of Cyclic Redundancy Codes in Ultra-Dependable Systems. *International Conference on Dependable Systems and Networks (DSN 2005)* (pp. 346 - 355). Institute of Electrical and Electronics Engineers.

Powell, D. (1992). Failure Mode Assumptions and Assumption Coverage. *Twenty-Second International Symposium on Fault-Tolerant Computing (FTCS-22)* (pp. 386 - 395). Boston: Institute of Electrical and Electronics Engineers.

PTC. (2013). *Widnchill Quality Solutions*. Retrieved from http://www.ptc.com/products/windchill/quality/

Reibman, A. L., & Veeraraghavan, M. (1991, April). Reliability Modeling: An Overview for System Designers. *Computer, 24*(4), 49 - 57.

SAE. (2010). *Aerospace Recommended Practice (ARP4754) - Guidelines for Development of Civil Aircraft and Systems, Revision A.* Society of Automotive Engineers.

Spitzer, C. R. (Ed.). (2007). *Digital Avionics Handbook* (Vols. Avionics: Elements, Software and Functions). CRC Press.

Torres-Pomales, W. (2000). *Software Fault Tolerance: A Tutorial.* NASA/TM-2000-210616, National Aeronautics and Space Administration.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
01- 04 - 2014	Technical Memorandum	

4. TITLE AND SUBTITLE

Selecting an Architecture for a Safety-Critical Distributed Computer System with Power, Weight and Cost Considerations

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-20379

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-2014-218242

12. DISTRIBUTION/AVAILABILITY STATEMENT

Unclassified - Unlimited
Subject Category 62
Availability: NASA CASI (443) 757-5802

13. SUPPLEMENTARY NOTES

14. ABSTRACT

This report presents an example of the application of multi-criteria decision analysis to the selection of an architecture for a safety-critical distributed computer system. The design problem includes constraints on minimum system availability and integrity, and the decision is based on the optimal balance of power, weight and cost. The analysis process includes the generation of alternative architectures, evaluation of individual decision criteria, and the selection of an alternative based on overall value. In this example presented here, iterative application of the quantitative evaluation process made it possible to deliberately generate an alternative architecture that is superior to all others regardless of the relative importance of cost.

15. SUBJECT TERMS

Architecture; Decision analysis; Reliability; 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	24	(443) 757-5802