Axiomatic Design Process for Disaster Tolerance

Diana EASTON
Systems Engineering
Southern Methodist University
Dallas, TX 75275, USA

and

Mitchell A. THORNTON
Computer Science and Engineering
Southern Methodist University
Dallas, TX 75275, USA

and

V. S. Sukumaran NAIR
Computer Science and Engineering
Southern Methodist University
Dallas, TX 75275, USA

ABSTRACT

Axiomatic Design (AD) is a methodology that utilizes customer needs as input and produces functional requirements, design parameters, and process variables through the use of matrix methods. AD is based on two design axioms; the independence and the information axiom. These two design axioms and the AD approach in general seem to be well-suited for the design and analysis of large-scale computer and communication infrastructure systems. This paper describes the application of ideas from AD for

Keywords: Disaster Tolerance, Axiomatic Design

1. INTRODUCTION

Disaster Tolerance is the characteristic attributed to a system that can withstand a catastrophic failure and still function with some degree of normality [1,14]. We differentiate between the terms "fault tolerance" and "disaster tolerance". Fault tolerant system design has been studied for the last few decades and is usually intended to tolerate natural failures due to hardware wear and tear, software design errors, and unintentional user errors. Typical strategies to provide fault tolerance at the physical level include the incorporation of redundant system components and a voting mechanism [2]. Other methods include the use of error-checking and correcting methods, hot-swap support, and special "watchdog" types of software.

There exists a fairly substantial commercial market for disaster tolerant and disaster recovery techniques and mechanisms for the IT industry such as that in [12]. These methods typically employ remote data storage centers and associated policies. In our work, we take a much broader view of disaster tolerance and we examine the problem from a design and retrofit perspective. Based on this viewpoint, we define disaster tolerance as follows:

Disaster tolerance is a superset of fault tolerance in that:

(a) a disaster may be caused by natural failures as well as intentional malicious and planned sabotage of the system;
(b) natural laws of low probability of simultaneous occurrence of multiple failures are not applicable;
(c) failures could escalate into a wide catastrophic system failure.

We are most interested in very large systems such as a large distributed computing network that would preclude the use of physical component redundancy due to prohibitive cost. As an example, a single network router may have a degree of fault tolerance by the incorporation of redundant output line drivers; however, a data network that is distributed over a large geographical area is likely to be impossible to replicate fully and thus redundancy is not a singularly valid method for incorporation of disaster tolerance. Even if such a

geographically wide-spread system were replicated, communications channels for synchronization and disaster detection would also be required, possibly with inherent redundancy, to achieve any degree of disaster tolerance.

We differentiate between *disaster tolerant systems* and *disaster recovery systems*. Disaster recovery is the ability to resume normal operations after a disaster has occurred while disaster tolerance is the ability to continue operations in an uninterrupted manner despite the presence of a disaster. This implies that the main difference between disaster recovery versus tolerance is one measured in terms of the delay that occurs after a disaster and before normal operations resume.

Whether a typical implementation is a disaster recovery or tolerance method depends upon the application. In past work in fault tolerance several approaches have been employed including redundancy [5,6], module replacement [7], error correction and detection [8], rollback [9], and self-repair [10,11].

Two basic approaches to fault recovery are termed forward and backward recovery. In the case of forward recovery real-time performance can be achieved. We propose to generalize and augment these approaches for large-scale system disaster recovery and to apply either forward or backward recovery methods where appropriate and, also, to implement these mechanisms in systems that are already in existence but have poor or no recovery ability at the present time.

The use of these concepts is generalized for disaster tolerant systems. This new area of disaster tolerance has several features and characteristics that differentiate it from classical fault tolerant results. Some of the chief differences are:

- The granularity of the underlying fault resulting in a system disaster,
- un-natural laws or causes of fault occurrence resulting in a disaster,
- and, the widely distributed and enormous size of the systems undergoing a disaster.

Because of these different characteristics and properties, classical fault detection and recovery techniques are, in general, not applicable to disaster tolerance. In classical fault tolerance, a great deal of effort is expended for fine-grained random error detection. In the large-scale systems we are interested in, such errors are manifested as erasures and thus may not qualify or cause widespread system failure. Furthermore, the incorporation of redundancy in large systems must be cleverly managed so that we do not create 100% (or more) overhead that would be impractical to implement.

Finally, there is a tight relationship with system security and availability of the system. These properties are generally not considered in classical fault tolerant design approaches and we believe that consideration of these aspects will lead to fundamentally new and different methods for the design and implementation of new systems. Furthermore, many existing large-scale systems present in the infrastructure would be impractical to redesign and deploy, thus it is necessary to determine new techniques to incorporate disaster tolerance as an augmentation of existing systems.

Furthermore, many existing large-scale systems present in the infrastructure would be impractical to redesign and deploy, thus it is necessary to determine new techniques to incorporate disaster tolerance as an augmentation of existing systems.

2. AXIOMATIC DESIGN BACKGROUND

Axiomatic Design (AD) is a new rigorous systematic paradigm developed for the purpose of transforming a set of customer needs, that are often loosely or ill-defined into functional requirements, process variables, and design parameters [3,4]. This technique is named for its use of two fundamental design axioms:

1) *Independence*: This axiom maintains and promotes the independence of various functional requirements, such that specific design parameters may be modified to satisfy a particular requirement without affecting other functional requirements.

2) *Information*: This axiom states that the information content of alternative designs should be minimized, thus maximizing the success of the design.

By avoiding complex functional requirements and focusing on simplified requirements with minimal information, the realization of a design adhering to the requirement is easier to achieve.

The AD process adheres to the two axioms through a rigorous dependence matrix formulation that uncouples (promotes independence) among the requirements. While this approach is suggested for use for use in the design of disaster tolerant systems, we also believe these principles can be adapted for modeling and simulation of the extremely large systems of interest, in terms of disaster tolerance.

As an example, system-wide simulation in the presence (or absence) of an event that potentially could cause a disaster is impractical due to the complexity of the system. By using the AD approach to determine uncoupled (or independent) subsystems, the subsystems can be independently modeled; both with and without the presence of a potentially disaster-causing event, and the law of superposition can then be used to infer overall system response.

The ability to model large complex systems in a normal and disaster mode is crucial for defining disaster models. Disaster models are important elements needed for the design of disaster tolerant systems, since they will be purposefully injected into robust system models to

gauge the degree of disaster tolerance present in the system.

Whether designing or simulating large systems, the "customer needs" that the system is intended to satisfy must be identified as a first step to develop a model. Because models are used for both the synthesis and analysis problems, we shall henceforth describe the AD approach with respect to large-scale system modeling with the understanding that applicability is present for both engineering tasks. In the AD approach, needs are mapped to functional requirements through the formulation of a dependency matrix. In the case of large-scale system design, it is typical that customer needs are not rigorously defined and indeed it is the job of the systems engineer to transfer these needs to a technically detailed set of functional requirements. In the case of simulation of an existing large-scale system, it is important to re-establish the customer needs from a technologically neutral point of view. This neutral point of view is essential to ensure that the resulting model is not dependent upon the particular technology in use for the system implementation.

The identification of the customer needs is often difficult, but also insufficient for formulating a model of the system. The development of technically detailed functional requirements is essential and it is in this phase of the system modeling process that we believe the AD approach is particularly advantageous for large-scale computer and communications systems.

In adherence with the design axiom of independence,

3. AD APPROACHES FOR DISASTER TOLERANCE

Our methodology utilizes system models to simulate large systems in normal operating modes and in disaster modes. The ideas of AD will be used to divide the simulations into suitable independent sub-system simulations that can be linearly combined to infer the entire system simulation result. The behavior of a system undergoing a disaster will be abstracted into a simple disaster model using catastrophe theory or cellular automata abstractions that can be easily injected into a system model, much in the same way that traditional fault modeling of integrated circuits is performed.

Once appropriate disaster models are identified, modifications to the disaster-free system design are identified that allow the system to operate in a disaster tolerant or disaster recovery mode. The disaster models will then be injected into these augmented system models and simulated again for the purposes of validating or gauging the effectiveness of the disaster tolerant system.

Our overall approach involves three aspects:

1) modeling large-scale systems both in the presence and absence of a disaster
2) formulating system enhancements that allow for disaster tolerance and/or recovery
3) validating the effectiveness of the system enhancements in terms of tolerance and/or recovery in the presence of a disaster

The block diagram in Figure 1.0 illustrates the various phases of our research approach to disaster tolerance and recovery.

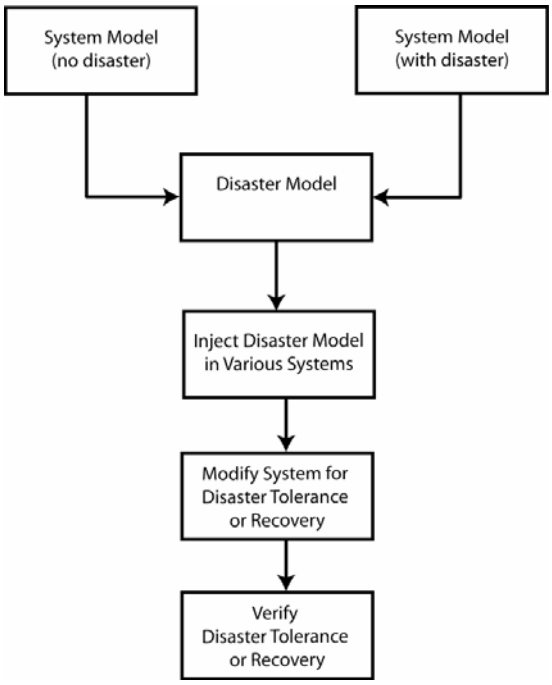

Figure 1.0: Block Diagram of Disaster Tolerance Research Approach

A key issue is the methodology used for modeling large-scale systems. Because the systems of interest are so large, we formulate abstract models that can be used to simulate only the characteristics of interest. Furthermore we decompose the system model into independent sub-models that may be simulated separately and combined using superposition to achieve the entire system response. We use ideas from the axiomatic design process to determine the appropriate decoupling of the system model into independent sub-components.

Initially, systems are modeled both in the presence of a disaster and in a disaster-free state. Based on the comparison of these two simulation responses, we will characterize an independent disaster model that may be injected into other system models to simulate behavior in the presence of a disaster.

Next, robustness features are added to the critical elements of the system model to allow for proper operation in the presence of a disaster (i.e. add disaster tolerance) or to allow for graceful recovery to proper operation in the presence of a disaster. After these,

modifications are made to the system model, we verify the disaster tolerance or recovery through the injection of various disaster models into the modified system model and use simulation to determine behavior in the presence of a disaster.

The AD approach is critical in this application since the systems we model are very large and have many interdependencies that may not be obvious. By formulating the design matrix, different subsystems may be parameterized and simulated separately. In the terminology of AD, the design process is envisioned as being composed of mappings among different domains. Initially, customer needs are formulated in the "customer domain" which are then mapped to the "functional domain", followed by a mapping to the "physical domain", and ultimately a mapping to the "process domain". These design domains can vary depending on the system of interest. For disaster tolerance, our initial domain is the "robustness domain" where needs for disaster tolerance are specified. This is followed by the same three domains as mentioned previously. Figure 2.0 contains a diagram of these domains and their relationship for disaster tolerance.

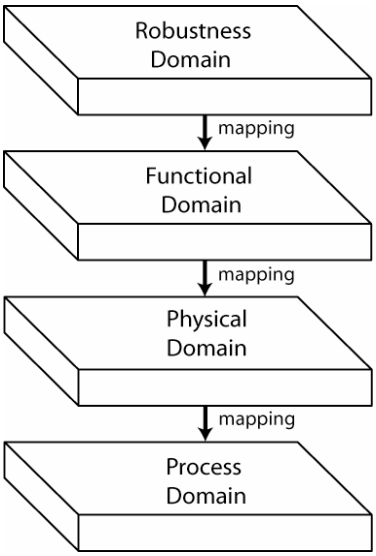

Figure 2.0: AD Domains for Disaster Tolerant System Design

The domain in the top of Figure 2.0 represents the "desired characteristics" of the disaster tolerant system whereas the bottom domain is the design solution. The two middle domains consist of formally specified functional requirements in order to achieve the desired characteristics and these requirements are mapped into the "physical domain" consisting of design parameters. These mappings are performed in accordance with the two axioms of design independence and information minimization.

In the first stage of mapping from the robustness domain to the functional domain, care is taken to ensure the mapping is performed in a design-neutral manner so that the design solution space is not constrained early in the process. The disaster tolerant robustness requirements are typically loosely defined and the designer must transform these needs into a more rigorous form that contains range values and tolerances. Also, aspects such as environmental influences and noise must also be incorporated at this time. In formulating the functional requirements, the philosophy of AD is to ensure that such requirements are as independent as possible; that is, no single functional requirement supersedes or influences another. At this stage of the design, the independence enforced is in terms of the functions not the physical parts of the system.

The next phase of the mapping involves transforming the functional requirements into design parameters. Conceptually, this is the portion of the design that transforms the "what it does" (or functionality), to the "how it does" (or implementation). In AD theory, this mapping can be formulated as a matrix equation that maps the set of functional requirements, $\{FR\}$, to the corresponding design parameters, $\{DP\}$. The linear transformation matrix is referred to as the *design matrix*, **A**. This relationship is given in the following equation.

$$\{FR\} = \mathbf{A}\{DP\}$$

As an example, consider the simple case where we have identified 3 functional requirements and 3 design parameters. In general, the design matrix has elements $[a_{ij}]=\mathbf{A}$ and the functional requirements are related to the design parameters as:

$$FR_1 = a_{11}DP_1 + a_{12}DP_2 + a_{13}DP_3$$
$$FR_2 = a_{21}DP_1 + a_{22}DP_2 + a_{23}DP_3$$
$$FR_3 = a_{31}DP_1 + a_{32}DP_2 + a_{33}DP_3$$

In order to satisfy the independence axiom, the design matrix **A** must be diagonal (all $a_{ij}=0$ unless $i=j$) or lower or upper triangular. In a lower triangular matrix all $a_{ij}=0$ for $i<j$ and an upper triangular matrix is one where all $a_{ij}=0$ for $i>j$. When the design matrix is diagonal, each FR is satisfied by an independent DP and the resulting design is referred to as an *uncoupled* design. A triangular design matrix indicates that FRs are independent if, and only if, a proper sequence of DPs is determined. Triangular design matrices represented *decoupled* designs. Any other form of the design matrix is referred to as a *coupled* design. It is thus a goal to specify the set of FRs and DPs such that an uncoupled or decoupled design matrix can be formulated.

From a mathematical point of view, it is possible to diagonalize a matrix through a coordinate transformation. Unfortunately, for the AD design matrices, such transformations can lead to DPs that have no real meaning and thus this purely mathematical approach is insufficient. In order to determine a design hierarchy that adheres to the independence principle, the

concept of zigzagging is employed. Zigzagging refers to the idea of crossing through to different domains rather than attempting to perform hierarchical decomposition within a single domain. It is often the case that high-level FRs cannot be decomposed into simpler independent FRs until decisions have been made regarding the DPs that will be used.

Another important concept with regard to the design matrix is considerations about its rank. In an ideal design, there are an equal number of FRs and DPs leading to a square matrix. Furthermore if the matrix represents an uncoupled or decoupled design, the matrix is necessarily of full rank. If the number of DPs is less than the FRs, the design is a coupled design. Alternatively, when the number of DPs is greater than the FRs, a redundant design results. In this latter case, it is possible to achieve an uncoupled or a decoupled redundant design depending on which of the DPs are fixed and in the order in which they are fixed.

Given these basic ideas in AD, the problem typically becomes one of decoupling a coupled design. In the context of modeling large-scale systems for disaster tolerance, it is generally easy to formulate a few high-level functional requirements, such as "in the event of a disaster, the system shall recover to be fully functional within five minutes". However, these few high-level FRs and the usually enormous number of DPs associated with a large-scale system place us in the unenviable situation of a redundant design. IT is for this reason, that we are focusing on the hierachical decomposition of the FRs through zigzagging so that an uncoupled system can be formulated. Whether the ultimate goal is for disaster tolerant system design or simulation, this decomposition task appears to be crucial in order to independently design or model critical subsystems.

4. CONCLUSION AND FUTURE EFFORT

We have described an approach for using ideas from the systems engineering design methodology known as axiomatic design for disaster tolerant, large-scale systems design and analysis. We believe such an approach is advantageous since AD provides a technique for automatically determining subsystem independence and minimization of conflicting needs and requirements. The enormously large scale of many critical infrastructure systems; particularly in communications and computing necessitate an approach for independent subsystem modeling and design in order for the problem to become feasible.

5. REFERENCES

[1] S.A. Szygenda and M.A. Thornton, Disaster Tolerant Computing and Communications, In Proceedings of the *International Conference on Cybernetics and Information Technologies, Systems and Applications* (CITSA 2005), and *International Conference on Information Systems Analysis and Synthesis* (ISAS), July 14-17, 2005, pp. 171-173.

[2] D.K. Pradam, Ed., **Fault-Tolerant Computing – Theory and Techniques**, Prentice-Hall, 1986.

[3] N. P. Suh, **The Principles of Design**, Oxford University Press, Oxford Series on Advanced Manufacturing, New York, New York, ISBN 0-19-504345-6, Oxford Series on Advanced Manufacturing, February 1990.

[4] N. P. Suh**, Axiomatic Design: Advances and Applications**, Oxford University Press, Oxford Series on Advanced Manufacturing, New York, New York, ISBN 0-19-513466-4, May 2001.

[5] J.C. Tryon, Quadded Logic, in Redundancy **Techniques for Computing Systems**, W.C. Mann and R.C. Wilcox, Ed., Washington DC: Spartan, 1962, pp. 205-228.

[6] R.E. Lyions and W. Vanderkulk, The use of triple modular redundancy to improve computer reliability, *IBM J. Res. Develop*., vol. 7, pp. 200-209, 1962.

[7] T. Bloom, Dynamic Module Replacement in a Distributed Programming System, *Ph.D. dissertation*, Massachusetts Institute of Technology, 1983.

[8] E.R. Berlekamp, **Algebraic Coding Theory**, McGraw-Hill, New York, 1968.

[9] R.E. Ahmed, R.C. Frazier, et al., Cache-aided Rollback Error Recovery (CARER) Algorithms for Shared-Memory Multiprocessor Systems, in Proc. of *Int. Symp. on Fault Tol. Comp. Sys*., 1990, pp. 82-88.

[10] S.A. Szygenda and M.J. Flynn, Failure Analysis of a Memory Organization for Utilization in a Self-Repair Memory System, *IEEE Trans. on Reliability*, R-20(2), May 1971, pp. 64-70.

[11] S.A. Szygenda and M.J. Flynn, Coding Techniques for Failure Recovery in a Distributive Modular Memory Organization, in Proc. of *American Federation of Information Processing Societies*, 1971, pp. 459-566.

[12] F. Chang, M. Ji, S.-T. Leung, J. MacCormick, S. Perl, and L. Zhang, *Myriad: Cost-effective Disaster Tolerance*, Proceedings of the *USENIX Conference on File and Storage Technologies*, January 2002.

[14] M. A. Harper, C. M. Lawler, and M. A. Thornton, IT Application Downtime, Executive Visibility and Disaster Tolerant Computing, In Proceedings of the *International Conference on Cybernetics and Information Technologies, Systems and Applications* (CITSA 2005), and *International Conference on Information Systems Analysis and Synthesis* (ISAS), July 14-17, 2005, pp. 165-170.