ABSTRACT

Mistakes in the design process have been recognized as a major source of product quality loss. There are several methods currently used to identify and quantify these mistakes. However, these methods typically do not provide a useful context within which to quantitatively incorporate mistakes into the design process in a beneficial way. This paper presents an approach to determine when it is appropriate to perform error checking to eliminate a potential mistake. The proposed approach is intended to be used when time is a limited design resource and design goals are technically attainable. It is proposed that the cost of a mistake can be quantified as the amount of time a mistake adds or subtracts from the overall time required to achieve the design’s objectives. To determine this, an optimization problem is formulated which minimizes time spent in the design process. In this optimization problem the design variables are the binary choice whether or not to perform an error check. The approach is demonstrated in two case studies, one a simple theoretical design problem and the other the design of an I-beam. The results of these case studies demonstrate the approach’s effectiveness, and present several avenues for future work.

1 INTRODUCTION

There is a growing recognition in industry and academia that at its core engineering design is a decision making process. This work is largely classified under the auspices of Decision Based Design (DBD) whose basic concepts were first presented by Shupe [1], and later turned into a framework by Hazelrigg [2]. Hazelrigg claimed decisions are primary design constructs and proposed a two step process can describe any design process: (1) generate alternatives and (2) choose the best one. There are many challenges to successfully implementing this two step process [3], but the elegance of this approach is the designer controls the process by choosing the appropriate idea generation and selection/refinement tools. The responsibility for each decision, therefore, is squarely placed with the designer and the DBD paradigm places the designer as the controller of the design process.

If designers could make perfect decisions, then the only limitations to a DBD approach would be the limitations inherent to the tools used during step 1 and 2. Developing validation criteria for the more commonly used decision support tools is an ongoing and important research topic [4, 5]. However, this research is focused on the decision tools themselves and not their end user, the designer. There is a growing body of research suggesting designers cannot flawlessly utilize these tools. Furthermore, this work [6] suggests that people are not flawless decision makers, and in some cases may even be poor decision makers. This paper presents an approach designed to augment designer decision making by quantitatively incorporating error checking into the design process.

2 BACKGROUND

Wassenaar and Chen acknowledged that when several design alternatives have utilities that are close together, there is a chance that the designer will choose an alternative without the highest utility [7]. In fact, the probability that the designer will make a mistake increases as the difference between the two alternatives expected utility decreases. Chen models these mistakes by assigning a deterministic and stochastic portion to a designer’s utility function. The deterministic portion is the designer’s true utility function, while the stochastic portion represents the designer’s inability to exercise that function. Mistakes are possible when the difference between two design
alternative’s deterministic portions is less than the sum of the stochastic contributions.

In the economics community the idea of exercising deterministic preferences and making stochastic choices is not a new idea. As early as 1985, Machina was investigating the validity of the claim that decision makers exercise their preferences with stochastic choices [8]. His work and more recent experimentation by Hey, Sopher and Narramore provide additional evidence that supports the theory of stochastic choices [9, 10].

One key similarity between the experiments conducted by Hey, Sopher and Narramore is that they invariably use simple questions to elicit preference responses from a decision maker. For example, in Hey’s experiment subjects were given the same test twice, separated by 3-5 days. The test had 80 pair-wise comparison lottery style questions that asked the subject to choose between two options [6]. These types of questions are much simpler than the complex design decisions engineers must make. However, it may be that engineers are better conditioned at making decisions than the general populace [11].

Unfortunately, the same behavior was demonstrated by engineers in case studies on preference consistency by Kulok [12]. In one case study a decision maker (DM) was asked to rank order a set of drills. When the consistency of his preferences was checked, it was discovered that one of his choices did not match the others. After asking the DM if the response should be different he confirmed that he had indeed made a mistake. These results beg the question: If a designer cannot trust themselves to make decisions, what can they trust?

2.1 Error Proofing

Proponents of design process error proofing would say designers can trust themselves to make decisions when they apply error proofing techniques to help prevent design mistakes. The term error proofing originally emerged during the 1960’s and is associated with the work of Shingo [13]. Shingo created Zero Quality Control that strove to completely eliminate the potential for factory workers to make mistakes or to catch mistakes through inspection. The two mechanisms identified by Shingo are the basic techniques for design error proofing as well. The designer can prevent errors though either prevention error proofing or detection error proofing [14].

2.1.1 Prevention Error Proofing

The guiding principle behind prevention error proofing is to prevent any actions that could result in an error. In manufacturing this means using guide pins, limit switches or proximity detection sensors. In design there are some obvious errors that can occur, D-FMEA has identified five common error classifications and provided a set of questions associated with each designed to help focus the designers on which errors may

Although prevention error proofing is largely impractical with respect to design decisions, there are places in the design process where it could be applied. The most potential of prevention error proofing is for it to be used to eliminate communication errors and so called “careless” mistakes. For example, in 1999 NASA’s Mars Climate orbiter crashed into Mars’s atmosphere and disintegrated. One of the root causes of the disaster was a communication mistake when altitude correction calculations were expressed in the wrong units [16]. Mandating the same units be used across all NASA and subcontractor calculations would have been an effective error proof. Another prevention error proof is requiring a computer password to be typed twice, preventing a typographical error from locking a user out of their machine.

2.1.2 Detection Error Proofing

In detection error proofing a process is constantly monitored to prevent deviation from the desired parameters. When deviation is detected, the process is stopped to prevent the errors from propagating further. The problem with implementing detection error proofing in the design process is it requires additional time during each design iteration be spent checking the designers output. However, unlike prevention error proofing, it is feasible to incorporate detection error proofing directly into the design process.

In the previous example from Kulok, a preference consistency check was used to guarantee the DM had correctly specified their preferences [12]. Kulok’s method fulfills the requirements to be a detection error proofing process because it checks the designer’s output for mistakes. However, the process is only applicable to stated preferences between two alternatives since it was specifically created to be used in the context of the Hypothetical Equivalents and Inequivalents Method [17]. More general methods of error proofing are necessary when examining the design process as a whole.

2.2 General Error Proofing Methodologies

There are two primary general design process error proofing methods in the literature today. These methods are intended to be used early in the design process, where detecting errors can help prevent them from cascading through the problem or to other sub systems. The basic premise for both methods is a two step process: (1) identify all potential errors and (2) rank the errors based on their severity. The final output for both processes is similar, but they each achieve that output in a different manner.

2.2.1 Design Failure Modes and Effects Analysis

Design Failure Modes and Effects Analysis (D-FMEA) uses expert systems to identify potential errors in the design process [18]. To assist experts in determining which errors may occur, D-FMEA has identified five common error classifications and provided a set of questions associated with each designed to help focus the designers on which errors may
occur. These classifications are Knowledge, Analysis, Communication, Execution, Change and Organization.

Once the designer has qualitatively generated a set of potential errors, D-FMEA uses a Risk Priority Number (RPN) to quantitatively rank them. The RPN is the multiplicative of the error’s chance of occurrence, chance of detection and severity. The chance of occurrence and detection are estimated by the designer conducting the D-FMEA. The severity, however, is calculated using two Houses of Quality [19] to map the design errors to customer requirements.

The major criticism for D-FMEA is it requires a large time investment to generate the initial set of potential errors. Even if the designer manages to create an exhaustive list of potential errors, the ranking system used by D-FMEA is not always accurate. The first issue with RPN is it is the product of three uncertain quantities, which through multiplication makes the end result even more uncertain. Additionally, the primary tool to determine severity, the House of Quality, has been shown by Olewnik to produce fairly arbitrary results when used as a quantitative tool [20]. These shortcomings of D-FMEA are also noted by Lough and Stone [21]. They propose an alternative solution to the problem of design process error proofing.

### 2.2.2 Risk in Early Design

An alternative method to error proofing the design process is the Risk in Early Design (RED) method [21]. In RED an initial set of potential mistakes is generated using the Function-Failure Design Method (FFDM) [22]. In FFDM a functional model of the design is cross referenced with a failure model for the components performing those functions. The effectiveness and results of this referencing is dependent both on the language used to describe the functional model and failure model and on the completeness of the design repository the data is taken from. Using historical failure data, each component’s failure mode is assessed for overall risk and the results are communicated to the designer using risk fever charts.

The RED method does offer some significant advantages over D-FMEA. It effectively automates the mistake generation process and it provides a metric to measure risk based on historical risk data. However, for unique designs the RED method does not provide the same insight an expert might on potential mistakes. Furthermore, the RED method presupposes the existence of historical failure data recorded in the appropriate failure taxonomy. Another limitation to the RED method is the information it provides is not in units familiar to a designer. Instead it is a relative rank ordering of potential failures sorted by severity and likelihood rated on a 0-5 scale. In addition, D-FMEA and RED both do not consider the potential for mistakes to have a positive impact on design and do not provide a clear mechanism to integrate information in the form of risk rankings/fever charts into the design process. These limitations are discussed in more detail in Section 2.3.

### 2.3 Current Method Limitations

#### 2.3.1 Are Mistakes Bad?

One of the presuppositions to both of these methods is that mistakes in the design process are a bad thing. There is no potential for mistakes to be encouraged through a favorable ranking in either method. It may seem counter intuitive to encourage mistakes in the design process, but there are a growing number of products whose existence can be traced directly back to a mistake. Scotchgard, Silly Putty and Coca-Cola are all examples of accidental creations that may not have occurred in a mistake free design environment [23]. Further, some engineers might argue they need to be free to make mistakes. Recent research by Gurnani supports the claim that perhaps mistakes can be advantageous and are a necessary part of design [24]. Gurnani demonstrated that for some problems it was possible to improve the overall solution quality and make divergent distributed design problems converge by allowing designers to make mistakes. This potential for mistakes is not accounted for in either D-FMEA or RED, and neither method can identify instances when it may be appropriate to make a mistake. In fact, neither method prompts the designer to carry out any specific action during the design process. This leads to the second limiting factor for these methods: they make no mention as to how to effectively implement an error proofing strategy.

### 2.3.2 Implementation

D-FMEA and RED provide no strategy to the designer on how to best utilize their respective outputs. In a vague sense it is implied that a designer should try to pinpoint if one of the errors has occurred, but how much effort should be spent error proofing the process? Additionally, the scales used by both D-FMEA and RED have little physical meaning to the designer. A designer cannot examine a potential mistake and conclude the real cost of it, instead the designer just knows in some way the mistake will hurt the product. Furthermore, without a useful scale to measure the cost of mistakes, there is no way to determine if the cost of making the mistake and continuing is greater than the cost of fixing the mistake itself. It is logical that only mistakes whose cost is greater the cost to fix them should be addressed. The approach presented in this paper addresses both of these challenges and is presented in Section 3.

### 3 PROPOSED APPROACH

#### 3.1 Assumptions and Applicability

The proposed two phase approach is meant to incorporate existing error proofing techniques like D-FMEA and RED in its first phase, and then quantitatively apply those techniques’ outputs in the second phase. The goal of the approach is to provide designers with a clear set of the error checks to be performed after each design iteration to guarantee that a mistake has not occurred. In this paper the term design iteration refers to the complete process a single subsystem
carries out to determine the values of their design variables that are passed on to the other subsystems. As it is currently formulated, the approach is intended to be used in distributed design environments where designers communicate primarily using design variable values. There are also two assumptions inherent to the proposed method:

1. Time is the limiting factor in the design process,
2. The designer is only trying to meet a predefined and technically achievable performance target.

Initially these conditions may seem to restrict the method’s scope of applicability, but there are many design scenarios where these conditions are already met. Typically in design product optimization does not stop when the system reaches convergence, but when the designer runs out of time. Any design scenario with a deadline for a deliverable will have time as a limiting factor in the design process. Part of the motivation to formulate the approach in a distributed design environment is that design scenarios with strict time and performance requirements are often found in these environments. In these types of design problems there is often a single project manager that sets both time and performance goals for their suppliers. In these cases the supplier has very little freedom, or motivation, to push the design window outward and any delays can result in the loss of a contract, a customer and their reputation. Therefore, it is assumed that failing to provide a design to the project manager is not an option and the fastest converging design process is desirable.

Additionally, when a project manager sets performance goals it is reasonable to assume the project manager would demand the best level of performance from its suppliers, but would set goals that are technically feasible. This assumption establishes an end point for where the design process should be stopped and a design submitted to the project manager. There are also an increasing number of design scenarios that meet the criteria for the proposed approach. One such scenario is the growing trend to outsource much of a product’s design work to a network of distributed suppliers. These suppliers must then assume the risks involved in producing an effective design on time [25]. In order for suppliers to remain successful it is important that they are able to meet their client’s goals within the specified time and performance constraints.

3.2 Approach Outline

As established above, the concept of time is fundamental to implementing the proposed approach in a design process. The approach is broken into two phases: an Error Generation Phase (EGP) and an Error Quantification Phase (EQP). The EGP is carried out before any design iterations have begun during the early stages of design, while the EQP outlines a process to be used by each designer during design iterations. The overall flow of the design process incorporating these two phases is shown in Figure 1.

In order to clarify the approach a case study is presented in conjunction with an explanation of the EGP and EQP. The formulation for this case study is presented in Section 3.2.1.

Following this the EGP is briefly discussed and the results of its application to the case study are shown. Since the EQP is the primary contribution of this work a step by step presentation of it is demonstrated in conjunction with the case study in Section 3.2.3.

**Figure 1. Proposed Design Process Flow Chart**

### 3.2.1 Problem Formulation

The problem to be presented in conjunction with the approach outline is a simple unconstrained, two designer, distributed design problem. The objective functions for this problem are shown in Eqn.1.

\[
F_1 = x^2 - 3x + xy \\
F_2 = 0.5y^2 - xy
\]  

(1)

To maintain the assumption of non-cooperation designer \(F_1\) controls design variable \(x\) and designer \(F_2\) controls design variable \(y\). As originally formulated this problem does not include mistakes or an associated time to conduct the design process [26]. This required some assumptions to be made to incorporate these into the problem. These assumptions are summarized in Figure 2.

**Figure 2. Two Designer Problem Summary**

### Objective Functions

<table>
<thead>
<tr>
<th>(F_1)</th>
<th>(F_2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(x^2 - 3x + xy)</td>
<td>(0.5y^2 - xy)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Time Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>One Iteration: 10 hours</td>
</tr>
<tr>
<td>Total Design Time: 160</td>
</tr>
</tbody>
</table>

**Design Targets**

\(F_1=0.5\) \(F_2=-0.5\)

For this problem all time units were assumed to be in hours, although another time metric could have been chosen provided consistency was maintained. Since this problem has no predetermined time limitations the choice of time scale is arbitrary. The time cost per iteration and total time were chosen based on observations of its behavior to allow sufficient time for the system to converge. With the problem formulated as a distributed design problem with time conditions, the EGP can be performed. The EGP is only addressed in broad terms in Section 3.2.2, but a more in depth explanation is available [27].
3.2.2 EGP

The purpose of the EGP is to provide the basic error data required to perform the EQP. This data includes a set of errors that may occur during the design process, the likelihood of each error occurring, and an error proofing procedure to identify and rectify the error. The first two pieces of data are outputs of the RED and D-FMEA approaches. It is recommended that one of these existing approaches be used to generate the set of potential errors. Neither of these approaches guarantees all the potential errors will be found, and there may be cases when an error is discovered during design that was previously unknown. In that case the designer is in no worse a state than if this approach had not been used.

The third piece of data must be independently determined and depends on the errors identified. However, error checking procedures are often created when developing validation and verification procedures and these procedures can be leveraged to provide the required data. In order to be useful in the context of the EQP each error checking procedure also needs to have an associated time cost. This time cost is the amount of time required to perform the particular error check.

Although the example problem is theoretical, and not a physical problem the potential errors proposed are all derived from a set of real potential mistakes. These errors, their realization, their likelihood and the time to correct them are summarized in Table 1.

<table>
<thead>
<tr>
<th>Units Error</th>
<th>Probability</th>
<th>Realization</th>
<th>Check Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.15</td>
<td>Designer expresses $x/y$ in inches instead of centimeters</td>
<td>8 hours</td>
<td></td>
</tr>
<tr>
<td>Decimal Misplaced</td>
<td>0.25</td>
<td>A decimal point is shifted one position up/down in $x/y$</td>
<td>4 hours</td>
</tr>
<tr>
<td>Modeling Error</td>
<td>0.05</td>
<td>A sign in the objective function’s cross term is reversed</td>
<td>6 hours</td>
</tr>
<tr>
<td>Premature Convergence</td>
<td>0.10</td>
<td>Designer terminates optimization algorithm before convergence is reached</td>
<td>5 hours</td>
</tr>
</tbody>
</table>

Table 1. Two Designer Problem Errors

To determine the values for each error probability a random number was generated from a uniform distribution between 0 and 0.35. There are many well developed tools, like probability encoding [28], that can be used to generate a probability from a data set. However, a uniform distribution was used because it is simple and there was initially no error data for this case study. For the Check Time, a random value from a uniform distribution between 1 and 10 was generated. If this approach were applied to a real problem, these numbers would be determined during the EGP of the approach.

After identifying an error check and its associated time cost thoroughness requires an additional piece of information. There may be some cases where error checking procedures are similar enough that the time required to perform several checks is less than the time required to perform each individual check. For example, if an automobile engine prototype is constructed then it could be used to test both the maximum horsepower and the level of emissions using the same prototype. Where these relationships exist they should be quantified and recorded for use during the EQP. For the example problem relationships between error checks were randomly established with a 35% chance of existing. Where a relationship was determined to exist, the amount of shared time was created from a uniform random distribution between 1 and 3. With the data from Table 1 and Table 2 all the data is available to begin the EQP phase of the approach.

### Table 2. Two Designer Problem Shared Costs

<table>
<thead>
<tr>
<th>Units Error</th>
<th>Decimal Misplaced</th>
<th>Modeling Error</th>
<th>Premature Convergence</th>
</tr>
</thead>
<tbody>
<tr>
<td>Units Error</td>
<td>-</td>
<td>1 hour</td>
<td>2 hours</td>
</tr>
<tr>
<td>Decimal Misplaced</td>
<td>-</td>
<td>2 hours</td>
<td>-</td>
</tr>
<tr>
<td>Modeling Error</td>
<td>1 hour</td>
<td>2 hours</td>
<td>-</td>
</tr>
<tr>
<td>Premature Convergence</td>
<td>2 hours</td>
<td>-</td>
<td>1 hour</td>
</tr>
</tbody>
</table>

3.2.3 EQP

Figure 3. EQP Flow Chart
The EQP is an eight step process whose purpose is to quantitatively incorporate the error data generated in the EGP into the design process. Using this data, designers are provided with a recommended set of error checks to perform in order to best help achieve their design targets. To demonstrate each EQP step, at the end of its explanation the step is applied to the example problem from Section 3.1. The EQP steps are outlined in Figure 3.

**EQP Step 1**

EQP Step 1 has two parts. In the first part the designer minimizes their local objective function. In the second part the designer maps that objective function to a time value representing how far along they are in the design process. There are several challenges associated with creating a relationship between the objective function value and time elapsed in the design process. While significant work has been conducted on convergence behavior [29-33], there is no widely accepted method to determine convergence rates or how close a distributed design system is to convergence. Additionally, the existing methods provide very little insight into the mechanisms governing the rate at which the problem converges. In this paper it is proposed that the relationship between the time passed in the design process and the current objective function value can be approximated using a function with a shape similar to the curve in Figure 4.

![Figure 4. Proposed Convergence Shape Plot](image)

The curve in Figure 4 has been purposely normalized on the y-axis to make it applicable regardless of the actual objective function values. The important characteristics of a function representing the curve in Figure 4 is that it should have a value of zero at time zero and a value of one when the process has reached the final convergence time. Additionally, gains in objective function should be increasingly difficult to obtain, or its second derivative should be negative. These requirements suggest that the best function to map time to objective function is an exponential function shown in Eqn. (1a).

\[
C = 1 - e^{-at} \quad \text{(1a)}
\]

For this equation \(C\) is the normalized objective function value, \(a\) is a time constant to be solved for, \(\delta\) is a small number to prevent \(\ln(0)\) and \(t\) is the current time since the design process began. Since we want to map objective function values to specific design times, Eqn. (1a) is solved for \(t\) as shown in Eqn. (1b). To determine the constant \(a\), it is recognized that \(a\) is analogous to the time constant in control theory which can be determined using the settling time if it is assumed that the total design time, \(\tau\), is the time required to reach 98% convergence. Therefore \(a\) can be determined using the expression:

\[
a = \frac{4}{\tau} \quad \text{(2)}
\]

Clearly this model will not be a perfect fit for every distributed design process. However, it is a first approximation for many systems and the criteria used in its determination are reasonable observations about the design process. It also reflects an intuitive understanding of the design process, since gains in performance become increasingly difficult to obtain as proximity to the desired objective increases. Further, this function also makes sense within the context of Chanron’s work on system convergence [29]. Chanron used linear control theory to model distributed design processes in state space and the solution to state space equations are by definition exponential functions [34]. The existence of exponential solutions, however, has some additional implications that are discussed in greater detail in Section 5.

Since the y-axis in Figure 4 is normalized, an approach is needed to normalize the objective function. The chosen normalization scheme is shown in Eqn. 3.

\[
C = \frac{f(x_0) - f(x_i)}{f(x_0) - f(x_{goal})} \quad \text{(3)}
\]

In Eqn. 3, \(f(x_0)\) is the initial objective function value, \(f(x_i)\) is the current objective function value and \(f(x_{goal})\) is the target objective function value. \(C\) is the normalized objective function convergence value. One implication in using this normalization scheme is that it is possible to achieve a \(C\) value greater than 1 when \(f(x_i)\) is less than \(f(x_{goal})\). To address this, it is assumed that regardless of whether the function approaches the target from a superior or inferior objective function value, it converges towards the target the same way, shown in Figure 5.

To accommodate the cases shown by the orange line in Figure 5, Eqn. 1b is modified to produce Eqn. 4.

\[
t = -\frac{\ln(1 - C + \delta)}{a} \quad \text{(4)}
\]
Using these equations the objective function can be mapped to the time domain by first normalizing the objective function using Eqn. 2 and then transforming it to time using Eqn. 1b and Eqn. 4. While it is relatively simple to translate the objective function values after each design iteration to the time domain, assessing an error’s cost presents additional challenges.

**EQP Step 2**

In EQP Steps 2 through 4 the goal is to evaluate an error’s cost, first in terms of the overall objective function and then in terms of time. It is generally desirable that the objective function cost is determined without additional function evaluations and that is accomplished by using the partial derivative of each designer’s objective function with respect to the design variables they control. While derivative information is not always available in design for a broad scope of design problems the objective functions have continuous derivatives that are readily available. The assumption that derivative information is available is also made by Chanron [32] and expected utility formulations employed by DBD to quantify DM preferences typically are continuously differentiable functions as well. Therefore, in EQP Step 2 the partial derivative of the designer’s objective function is taken with respect to the design variables they control. That derivative is then evaluated using the designer’s current design variables. This provides a slope that reflects the objective function’s sensitivity to changes in each design variable.

Designer 1 controls design variable \( x \), which means the partial derivative of the objective function evaluated at the current design point (1.2691, 2). The value \( y = 2 \) is the initial \( y \) design point, which remains unchanged since Designer 2 has not yet performed a design iteration. The evaluated partial derivative is shown in Eqn. 5.

\[
\frac{\partial f}{\partial x} = 2x - 3 + y = (2)(1.2691) - 3 + 2
\]

\[
\frac{\partial f}{\partial y} = 1.538
\]
In EQP Step 2 to determine the change in the overall objective function as shown in Eqn. 6.

\[
obj_{\text{sim}} = (x_{\text{sim}} - x) \frac{\partial f}{\partial x}
\]  

(6)

In Eqn. 6, \(obj_{\text{sim}}\) is the objective function value if the assumed error had not occurred, \(x_{\text{sim}}\) is the simulated design variable value, and \(x\) is the current design variable value. \(\frac{\partial f}{\partial x}\) is the partial derivative with respect to design variable \(x\).

To apply Eqn. 6 to the example problem an error is first simulated. For simplicity a units error is reversed by dividing \(x\) by 2.54 cm/in to yield \(x_{\text{sim}}\) equal to 0.50. Applying Eqn. 6, the approximated the objective function results in a change in objective function of -1.183.

**EQP Step 4**

The change in objective function determined in EQP Step 3 is be added to the current objective function value to determine what the new potential objective function could be if a mistake had not occurred. This value is then mapped to the time domain using Eqn. (1b). The current objective function and error are now expressed in the same units, time.

For the example problem adding the change in objective function value and the current objective function results in an approximated objective function value of 0.159. This value is then normalized by applying Eqn. 3 to yield 0.921 and converted to time using Eqn. 1b. In this case the result is 101.5 hours. This time value indicates that according to the model the design system would normally expect to iterate for 101.5 hours on an error free problem before achieving this value.

**EQP Step 5 and 6**

Using the current objective function and the objective function with an error’s time the cost of an error can be determined by subtracting the two values. This difference is then multiplied by the error’s probability to determine the expected value of the error’s cost. These results are saved with the associated mistake for use in EQP Step 7. For stochastically generated errors it is be desirable to use Monte Carlo simulation to determine the error’s expected value over a large number of trials. Once the simulation results are completed for all mistakes EQP Step 7 is used.

In the example problem a deterministic mistake was chosen, which means a single iteration is all that is necessary to determine the error’s potential impact. Subtracting the initial time value and the simulated value and multiplying by the probability for a units error in Table 1 results in an expected error cost of -12.8 hours. A negative time cost means if a mistake did occur it made the objective function worse than the true objective function value. However, since error checking a process takes time the improvement needs to be greater than the time to check it. For errors with a positive expected time cost it means the error actually improved the objective function, and there will never be an advantage to checking these errors. Once again, this assumes the system converges monotonically to the final value.

**EQP Step 7**

In this step the outputs from the EGP are integrated with the output from EQP Step 5 to formulate an optimization problem with time as the objective function and the decision to check each error as the design variables. The optimization problem to be solved is stated in Eqn. 7.

\[
\text{Minimize : } f(x) = \sum_{i=1}^{n} x_i \epsilon_i + \sum_{i=1}^{n} (1 - x_i) \delta_i - \sigma
\]

(7)

In this formulation the \(\Sigma\)'s denote summations over every potential error, \(x_i\) is either 0 or 1 to determine if the \(i^{th}\) error is checked or not, \(\epsilon_i\) is the cost of the \(i^{th}\) error, \(\delta_i\) is the cost of the \(i^{th}\) error check and \(\sigma\) is the cost savings for overlapping error checks. The design variables are restricted to two potential values, 0 or 1. This is done because an error check is either performed or it isn’t; it can’t be half performed. The objective function itself has three parts. The first part is the summation of the expected cost for each error that is not checked. Since this is a sum of expected values, it is necessary that the errors be independent of one another.

The second part of the objective function is the cost to implement an error checking process. These values are predetermined from EGP Step 3. The last part of the expression takes into account dependencies on the cost of testing mechanisms as outlined in EGP Step 5. The final output of the optimization is a vector of 0’s and 1’s for each potential error test. A 0 indicates the error check needs to be performed, while a 0 means the cost to test for the error is greater than the error’s cost.

To execute this optimization problem for the example problem, all four of the potential errors would have to be simulated and their potential impact evaluated in the same manner the units error was simulated. For any stochastic error, like the premature convergence error, several simulations are necessary to determine an accurate expected value. In the next section the results for continuing the EQP process until convergence and for all four errors are summarized and discussed.

### 3.2.4 Problem Results

Since the error proofing process is stochastic in nature a single simulation cannot capture the overall system behavior. Furthermore, since the approach uses an expected value to determine if an error proof should be performed there are times when the approach will perform worse depending on which side of the probability the mistake takes. In order to account for this, 150 simulations were run for several different design scenarios in order to provide a benchmark to evaluate the approach’s effectiveness. The first scenario is a simulation with no error checking and no errors by designers. The second
is a simulation with errors and no error checking performed and is representative of how a normal design process would proceed. The third benchmark is performing all the error checks every iteration. Finally, the approach is applied to a system with errors. Table 3 shows the results for these simulations.

<table>
<thead>
<tr>
<th>Design Scenario</th>
<th>Error Checking</th>
<th>Average Time</th>
<th>Standard Deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error Free</td>
<td>None</td>
<td>60 hours</td>
<td>0 hours</td>
</tr>
<tr>
<td>Errors</td>
<td>None</td>
<td>161 hours</td>
<td>108 hours</td>
</tr>
<tr>
<td>Errors</td>
<td>All Checked</td>
<td>162 hours</td>
<td>0 hours</td>
</tr>
<tr>
<td>Errors</td>
<td>EQP</td>
<td>133 hours</td>
<td>45 hours</td>
</tr>
</tbody>
</table>

The average time is the time required for the system to converge to 2% of the desired value over all the trials. It is important that in this case the results are not normally distributed around the mean, so the standard deviation is used only as a measure of the dispersion of the results.

Overall for the scenarios where errors were incorporated into a design process the approach yielded a better convergence time, reducing the time required to reach the desired solution by approximately 15%. Furthermore, 75% of the time the approach converged to its solution in less than 160 hours, or the average time for the two other simulations. There were even some occasions when the approach reached the desired solution faster than the error free process when favorable mistakes were made by the simulated designers.

4 CASE STUDY

In this section a case study is performed to demonstrate the approach’s application to an engineering problem. This case study is a nonlinear two designer I-beam design problem. Since the case study was converted into a distributed design problem from a multiobjective problem [35], some assumptions needed to be made with respect to the time and error data normally gathered during the EGP. The primary reason for this is time data is not typically available with a problem because it is either not recorded or it is proprietary in nature. To remove any bias, when data needed random numbers generated and where possible reasonable estimates were made.

For the remainder of this section the general problem outline is shown in Section 4.1, the required assumptions are shown in Section 4.2 and the results are presented and discussed in Section 4.3.

4.2.1 Problem Formulation

The second case study presented is the design of a 100 cm long cantilevered I-beam. This problem was originally formulated by Gold and Krishnamurty [35] as a constrained multi-objective design problem. For this case study, however, it was decomposed into a two designer problem with one designer attempting to minimize the I-beam’s weight and the other its deflection. Two of the four design variables were assigned to each designer and the problem maintains the assumptions of non cooperation. The problem is summarized in Figure 7.

<table>
<thead>
<tr>
<th>Designer 1: Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>Controls: x1,x3</td>
</tr>
<tr>
<td>Minimize: ( f_1(x) = \rho(2x_1x_3 + x_4(x_1 - 2x_4))^2 )</td>
</tr>
<tr>
<td>Subject to: ( \sigma_1 \leq 16 \leq 0 )</td>
</tr>
<tr>
<td>( 10 \leq x_1 \leq 80 )</td>
</tr>
<tr>
<td>( 0.9 \leq x_4 \leq 10 )</td>
</tr>
<tr>
<td>( 0 \leq x_1 \leq 10 )</td>
</tr>
<tr>
<td>( 0 \leq x_4 \leq 10 )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Designer 2: Deflection</th>
</tr>
</thead>
<tbody>
<tr>
<td>Controls: x2,x4</td>
</tr>
<tr>
<td>Minimize: ( f_2(x) = \frac{(\text{EI})^2}{(\text{Pl})^2} )</td>
</tr>
<tr>
<td>Subject to: ( \sigma_2 \leq 16 \leq 0 )</td>
</tr>
<tr>
<td>( 10 \leq x_1 \leq 50 )</td>
</tr>
<tr>
<td>( 0.9 \leq x_2 \leq 10 )</td>
</tr>
</tbody>
</table>

Figure 7. I-beam Design Problem

For this case study the I-beam was assumed to be constructed from ordinary 1040 Steel and to be loaded with 600 kN of force at the free end of the cantilevered beam. To accommodate the problem’s constraints an exterior penalty function was used to convert it to an unconstrained optimization problem. The geometric relationship between the design variables \( x_1, x_2, x_3, \) and \( x_4 \) are shown in Figure 8.

Figure 8. I-Beam Geometry

Once again this problem did not originally incorporate time data into its formulation. Some initial trials indicated the system without error converged in approximately 7 iterations to the desired objective function value. Since the process with errors will take longer to converge than the error free process, a little over twice the overall design time is allotted, with 160 hours being allowed at 10 hours allotted per iteration. This value was chosen based on the previous case study which had similar initial convergence behavior and once again the time unit hours is arbitrary for this case study provided consistency is maintained. It was also assumed that the same set of potential errors as found in the first case study, Table 1, are present in this simulation with the same error checking time values.

Finally an appropriate design target value needed to be determined. Previously this target was chosen to be near the problem’s Nash Equilibrium [26] to guarantee convergence. A
Nash point was chosen because a simple iterative process naturally converges to one of these equilibrium [26]. To achieve different targets, more sophisticated design techniques are required [36]. However, this problem presents a unique challenge since there are multiple equilibriums depending on the starting location, as shown in Figure 9.

![Figure 9. I-Beam Equilibrium Behavior](image)

Therefore, in this case the starting location and the target had to be chosen or the system would never reach the desired target because different portions of the design space converge to different Nash Equilibriums. The starting location was arbitrarily chosen in the middle of the design space and the target was set to the location’s equilibrium value, Table 4.

### Table 4. I-Beam Problem Parameters

<table>
<thead>
<tr>
<th>Design Variable</th>
<th>Initial Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1$</td>
<td>40 cm</td>
</tr>
<tr>
<td>$x_2$</td>
<td>25 cm</td>
</tr>
<tr>
<td>$x_3$</td>
<td>2.5 cm</td>
</tr>
<tr>
<td>$x_4$</td>
<td>2.5 cm</td>
</tr>
<tr>
<td>Mass Target</td>
<td>368 kg</td>
</tr>
<tr>
<td>Deflection Target</td>
<td>5.5 cm</td>
</tr>
</tbody>
</table>

Using these values as starting points and targets the same four design scenarios were simulated to provide a benchmark for the approach’s performance on this problem. Each design scenario was run for 150 iterations and the results of these simulations are summarized in Table 5.

### Table 5. I-Beam Design Problem Results

<table>
<thead>
<tr>
<th>Design Scenario</th>
<th>Error Checking</th>
<th>Average Time</th>
<th>Standard Deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error Free</td>
<td>None</td>
<td>70 hr</td>
<td>0 hr</td>
</tr>
<tr>
<td>Errors</td>
<td>None</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Errors</td>
<td>All Checked</td>
<td>202 hr</td>
<td>0 hr</td>
</tr>
<tr>
<td>Errors</td>
<td>EQP</td>
<td>288 hr</td>
<td>119 hr</td>
</tr>
</tbody>
</table>

In spite of the approach underperforming a process where all errors are checked, it does have some advantages over that error proofing strategy. It was found during the simulation this problem was extremely sensitive to small changes in design variable values, and the problem did not converge to the equilibrium if the design variable values were truncated at four decimal places. Further trials were not conducted to determine how many decimal places needed to be maintained. In spite of this, the approach was able to converge to the desired solution for all 150 trials. Also, when varying the starting location, the approach was able to achieve the desired solution in less than 300 minutes 50% of the time, where the process checking all errors never reached the desired solution. These results suggest that the approach has some additional robustness, which is not surprising given its stochastic nature. An additional case study for a large distributed design system with five designers and sixteen design variables can be found in [27].

### 5 CONCLUSIONS

The unique contributions of this paper include the concept of using time as a cost metric for mistakes in distributed design and providing an approach that supports this metric. The success of the approach in both case studies demonstrates its potential usefulness to designers and the advantages to using time as a cost metric when time is a limited design resource.

Although improvements are necessary to make the approach more effective, it leverages the two primary existing error checking methodologies, D-FMEA and RED. It also addresses the two primary limitations of the methodologies: they do not provide conditions for implementation and they do not consider the potential for mistakes to improve the design process. The approach outlined addresses both of these issues while incorporating the strengths of both D-FMEA and RED’s error generation methods.

In spite of this success this approach faces some challenges and has several limitations. When time is not an important
design constraint it is not as useful and it is designed only to be applied in a distributed design framework using an existing design process. There is also no provision in the approach for improving performance or for identifying times when it is advantageous to continue the optimization process to improve the solution. Furthermore, this approach presupposes the designer has access to derivative information, which is not available to many systems and it also limits the designer to specific design targets.

Finally, one of the implications of assuming an exponential convergence pattern is the potential for the design problem to converge sinusoidally. The challenge presented in these cases is the objective function cannot be mapped monotonically to a specific time value, which results in several potential time values for any given objective function value. Future work will focus on addressing the problem of sinusoidal convergence and refining a metric to measure convergence rates for distributed design theories. Additionally, a generalized method to estimate the impact of a mistake on the objective function is needed for systems without readily available derivative information.

ACKNOWLEDGMENTS

We would like to thank the National Science Foundation, grant DMII-0322783 and the Moog Corporation for their support of this work.

REFERENCES


