EXAMINING INTERACTIONS BETWEEN SOLUTION ARCHITECTURE AND DESIGNER MISTAKES

Erich Devendorf
Graduate Research Assistant
Department of Mechanical and Aerospace Engineering
University at Buffalo – SUNY
Buffalo, NY 14260
edd4@buffalo.edu

Kemper Lewis
Professor
Department of Mechanical and Aerospace Engineering
University at Buffalo – SUNY
Buffalo, NY 14260
Corresponding Author: kelewis@buffalo.edu

ABSTRACT
When designing complex systems, it is often the case that a design process is subjected to a variety of unexpected inputs, interruptions, and changes. These disturbances can create unintended consequences including changes to the design process architecture, the planned design responsibilities, or the design objectives and requirements. In this paper a specific type of design disturbance, mistakes, is investigated. The impact of mistakes on the convergence time of a distributed multi-subsystem optimization problem is studied for several solution process architectures. A five subsystem case study is used to help understand the ability of certain architectures to handle the impact of the mistakes. These observations have led to the hypothesis that selecting distributed design architectures that minimize the number of iterations to propagate mistakes can significantly reduce their impact. It is also observed that design architectures that converge quickly tend to have these same error damping properties. Considering these observations when selecting distributed design architectures can passively reduce the impact of mistakes.

1 INTRODUCTION
Decision Based Design (DBD) [1] and more recently Design for Market Systems (DFMS) [2] support some of the fundamental decisions that designers must make in the design and development of engineered products. DBD asserts that decisions are fundamental design constructs that can be classified into two categories: selection decisions and compromise decisions [1, 3]. DFMS expands this concept and emphasizes the role of engineers in big picture decisions, considering input from social, economic and environmental sources before making a decision [4-5]. Both DBD and DFMS focus on providing formalisms for modeling a design process as a decision making process [6-7]. The strength of these perspectives is they provide a context within which to rationally analyze, model and support decisions in design. However, the implementation of these perspectives is not without challenge.

For instance, DBD has been broadly summarized by Hazelrigg as a two step process [6] and although this process is an elegant representation of the designer’s work, completing either of these two steps is a challenging process [8]. To alleviate a portion of the burden on the designer a number of decision support tools have been developed [9-11]. Design tools all have their own individual limitations, a fact that is noted in [12]. If designers performed tasks flawlessly, then design would only be limited by the design tools themselves and the information available for use in those tools. However, other limitations have been discussed in [8], including limitations on the designers themselves.

Decision maker (DM) limitations have been discussed by Hey [13], who asserts DM’s often make irrational decisions. An irrational decision is one which does not optimize the designer’s utility [14] and the impact of this mistake is not always restricted to the DM. The discussion of rationality and mistakes in design is an important research topic, but this paper restricts itself to the interpretation provided in the literature review of mistakes found in Section 2. Designer mistakes can propagate to other designers and there has been significant research investigating how deliberate design changes can have unintended consequences as they propagate across a design
system [15-16]. The major difference between the propagation of changes and mistakes is the input in change propagation is a deliberate design change, while mistakes are unintentional and often undesirable.

While design mistakes can have a significant impact on the design process, there are other inputs that influence the design process [17]. This paper restricts itself to those inputs defined as designer mistakes and examines how design process architectures can be selected to mitigate the impact of a mistake in distributed multi-subsystem design. More specifically, the focus is on solution architectures in coupled optimization problems where ‘architecture’ refers to the ordering or organization of the subsystems.

One of the benefits of this work is it provides guidance in selecting appropriate design architectures. These architectures provide a passive mechanism to mitigate the impact of mistakes. This guidance will enable designers to reach iterative solutions quicker, in a more consistent manner that meets design deadlines and fulfills product development timetables. The next section provides background on the presence and role of mistakes in the engineering design process.

2 MISTAKES IN DESIGN

Traditional behavioral models assume DM’s are capable of perfectly executing their preferences and always make decisions that maximize their expected utility [18]. When DM’s choose alternatives that do not maximize utility based on the information available when the decision is made, it has been called an irrational decision. It is certain that DM’s intend to make decisions that result in the best possible outcome, but there is a substantial body of evidence suggesting there is a disconnect between these intentions and the ability of DM’s to perfectly execute their preferences.

The hypothesis that DM’s do not make perfect decisions has been put to the test by Hey [13], Sopher and Narramore [19]. In experiments using simple pair wise questions to elicit preference responses subjects were found to execute their preferences with 60% to 80% consistency. If the consistency rate was near 50% in these experiments, it would suggest DM’s have stochastic preferences. However, in all cases DM’s demonstrated some consistency in their responses. Their consistency suggests that the DM’s had a set of deterministic preferences, but exercised them stochastically.

Prospect Theory was proposed to explain some of these observed tendencies in a DM’s choices, notably when considering gains versus losses [20]. Prospect Theory is a descriptive model for decisions that attempts to predict real choices, instead of optimal choices [21]. Inconsistency is not viewed as irrational in Prospect Theory as it would be in Utility Theory. Instead it is modeled as a natural part of decision making. Through a set of surveys Kahneman and Tversky have demonstrated this principle and have expressed unequal utility changes in equation form [22].

While Prospect Theory does alleviate some inconsistencies in decision making, assuming DM’s have deterministic preferences and make stochastic choices can resolve the remaining inconsistencies. This idea is called bounded rationality and it is gaining increasing acceptance in a wide range of fields. Medicine [23], law [24] and economics [25] are all examples of disciplines that have benefited from incorporating bounded rationality into their theoretical formulations.

In engineering design the theory of stochastic choices has received broad implementation. MacDonald examined consumer decision making with regard to environmental product choices and found inconsistency between consumers’ stated preferences and purchasing decisions [26]. Designers making stochastic choices was also observed in [8] and used in [27] to model designers in an approach to separate design problems. An approach to remedy observed preference inconsistency in the Hypothetical Equivalents and Inequivalents Method [9] and elicit true preferences has also been created [28].

Approaches have also been developed to prevent designer mistakes and address the existence of stochastic design choices in design. The most well developed of these methods are Design Failure Modes and Effects Analysis [29] and the Risk in Early Design approach [30]. In most cases the approaches that incorporate mistakes in design have sought to reduce their overall impact on the design process. However, there has also been some work that leverages mistakes as a means to design innovation. In [31-32] designer mistakes are shown to sometimes have a beneficial influence in the design process. There are also numerous instances of products that owe their existence to mistakes supporting these approaches [33].

Although there is potential for mistakes to be beneficial in design and some error proofing approaches have incorporated this in their formulation [34], the overwhelming majority of design research focuses on mitigating uncertainty in design. Mitigating risk in the context of the final design has been the topic of Robust Design [35], which seeks to create final designs that are insensitive to uncertainty and change. In the same way a final design can be created with minimal sensitivity to changes, a design system can be configured to reduce the impact of uncertainties, in this case modeled as mistakes. In Section 3, multidisciplinary design optimization is discussed and distributed design is presented as a design framework that can be minimally sensitive to design mistakes.

3 MULTIDISCIPLINARY OPTIMIZATION

Complex system design often requires the involvement of a variety of experts. These experts may originate from different disciplines or even the same discipline. As products become increasingly sophisticated the number of disciplines required for a particular design task will continue to increase. For example, Ford’s Model T was composed of roughly 700 unique parts. In contrast the modern automobile has more parts in their radio alone and the Boeing 777 has over 3,000,000 unique parts provided by over 300 different suppliers [36]. With an increase in system size, there has also been an increase in the number of engineering disciplines required to complete design tasks. For example, drive by wire technology originally
developed for space shuttle has seen application in consumer automobiles [37].

By their nature, experts typically fully understand and are responsible for only a very specific part of the aggregate design problem. One way to facilitate communication between these experts is using one of the variety of multidisciplinary design optimization (MDO) frameworks that have been created.

Regardless of the framework chosen, the amount of time required to complete a design task is of critical importance. This was recently illustrated by the struggles of Boeing during development of the 787 Dreamliner. Its rival, Airbus, faced similar struggles developing the A-380 [38] and it is estimated that delays cost Boeing at least 100 orders. Some of that business was lost to their chief competitor Airbus. A major mitigating factor for Boeing was Airbus faced a similar problem as they did. Had Boeing been poised to seize the initiative it could have resulted in significant profits and gains in market share [39].

Even in less extreme examples, time remains a critical resource in the design process, whether it is speeding time to market or meeting an important deadline [40]. MDO frameworks provide the organizational structure necessary to meet these deadlines and coordinate among a variety of disciplines. In Section 3.1 these frameworks are broadly discussed and the distributed design framework is the topic of Section 3.2.

3.1 MDO Frameworks

MDO problems have two primary classifications based on the process applied to complete the design [41]. From a structural perspective the simplest method to solve an MDO problem is to apply an all-at-once approach. In an all-at-once approach designers from different disciplines work as system analysts to determine objective function and constraint values for a single optimization problem [42-43]. There are significant advantages to organizing an MDO system to solve a single centralized optimization problem. In a centralized problem all designers are working towards the same objective, information about the entire system is available to all designers and any optimal solution found is optimal to the global system.

Although these advantages make centralization attractive, it is almost impossible to centralize the design of complex systems. As technology enables designers to extend their capability and productivity it may be possible to centralize a greater number of design activities in the future. However, increases in engineering capability often result in increases in the complexity of design problems. Recognizing this, it is likely decomposition will remain an important and necessary aspect of product design processes in the near future.

Fortunately for designers a wide range of approaches have been developed to aid in system decomposition. The system decomposition process can be broken into two fundamental steps:

1. Identify the necessary subsystems,
2. Create a framework for communication between subsystems.

The first step in this process is not the topic of this paper, but it is by no means a trivial task. Subsystems can be created based on object decomposition, aspect decomposition, sequential decomposition and model based decomposition [44].

The second step is the creation of the design framework. In this paper “framework” is always be used to denote a specific MDO approach used to solve a design problem. MDO design frameworks specify the mechanics of how the design problem will be solved. These mechanics include determining the subsystem objective functions, establishing communication protocol, assigning control of design variables and addressing the other assorted protocols required for the decomposed subsystem to effectively iterate to a solution. There are a wide range of MDO frameworks which provide for these requirements while making guarantees about system convergence and the optimality of the final converged solution. These approaches include Analytical Target Cascading [45], Concurrent Sub Space Optimization [46], Bilevel Integrated System Synthesis [47] and Collaborative Optimization [48].

Using one of these frameworks has several advantages which depend on the framework chosen. For example, Analytic Target Cascading has been proven to guarantee the distributed system converges and that the converged value is a globally optimal solution [45]. Additionally, its hierarchy allows for traceability of the design process and provides for integration of marketing and business systems while establishing clear relationships between design subsystems. In spite of the advantages offered by existing MDO frameworks, there are many cases when a formal framework is not used.

There are several reasons why a formal MDO framework may not be applied to a design problem. Applying a framework to a decomposed problem requires a significant level of coordination between subsystems and a high level of management expertise. Further, the designers must all “buy in” to the proposed decomposition and framework. There are also some cases that do not naturally lend themselves to formal decomposition or where the parties involved cannot agree on an appropriate framework. When no formal framework is chosen or when the framework does not specifically proscribe subsystem interactions, the design problem may simply become a distributed design problem. The assumptions and mechanics governing distributed design problems are discussed in Section 3.2.

3.2 Distributed Design

Distributed design problems are a specific type of MDO problem, the mechanics of which are an ongoing area of research [31, 49, 50]. Distributed design problems are non-hierarchical with a set of different designers, or subsystems, each seeking to optimize their own individual objectives. When distributed design problems are cooperative they are similar to the all-in-one approaches to solving MDO problems. However, when the problems are non-cooperative they have more in common with decomposition approaches to MDO. It is important to note that the frameworks outlined for MDO decomposition approaches are not necessarily non-cooperative.
For example, Concurrent Sub Space Optimization and Collaborative Optimization more closely resemble cooperative processes [46, 48]. Non-cooperative distributed design problems are common in design, and have been widely studied. The requirements for non-cooperative behavior in an engineering design problem are [51]:

1. Designers have knowledge of only their own local objectives,
2. Designers act unilaterally to minimize their objective function,
3. Designers have complete control over specific local design variables,
4. Designers communicate by sharing the current value of their local design variables.

Distributed design problems are often sub problems that result from applying one of the design frameworks discussed in Section 3.1. Within a formal framework there are typically groupings of design subsystems in a feedback loop. These subsystems must iterate to find a solution and often fulfill the assumptions of non-cooperation. When this occurs, the grouped subsystems can be analyzed as a distributed design problem. For example, one approach to system decomposition is the Design Managers Aid for Intelligent Decomposition (DeMAID) [52]. In DeMAID the overall system is shown using a Design Structure Matrix with the goal of eliminating iterative loops to reduce redesign [53].

There are times, however, when all iterative loops between subsystems cannot be eliminated. These iterative loops maintain the assumptions of non-cooperation for a distributed problem because the system level optimizer is responsible for setting up the design framework but not supervising its execution. Convergence equilibrium and convergence rate are the two fundamental concerns when designers iterate under non-cooperative assumptions. These concerns are discussed in the following two sections, 3.2.1 and 3.2.2.

### 3.2.1 Convergence Equilibrium

Determining if a design system converges to a solution is of critical importance to understanding the system. Convergence equilibrium in distributed problems has been a topic of research for some time with the first work being performed by Vincent [54] for a simple two designer, two design variable problems. Vincent’s primary contribution was the introduction of Game Theory to model designer behavior, which was investigated further in [55]. In a game theoretic approach to design, the designers are modeled as players in an iterative game. In Vincent’s work each player alternates minimizing their local objective function value and communicates the associated design variables to the other player. After repeated playing of the sequential game the players either converge to a solution or diverge and continue playing indefinitely. When the players converge, they converge to a specific point called the Nash, or non-cooperative equilibrium [56].

Key to the understanding of the Nash Equilibrium is the concept of a player’s rational reaction set (RRS). In unconstrained problems the rational reaction set is the partial derivative of the designer’s objective function with respect to their local design variables. While determining designer RRS’s is not a trivial task, methods have been developed to approximate them for large systems [57]. The intersection of the RRS’s for both players in a two person game is by definition the Nash Equilibrium. For the simple case in Eqs. 1, two designer RRS’s taken from Vincent [54] are plotted with respect to the design variables \(x\) and \(y\) in Figure 1.

\[
\begin{align*}
F_1 &= x^2 + xy - 3x \\
\frac{\partial F_1}{\partial x} &= 2x + y - 3 = 0 \\
F_2 &= 0.5y^2 - xy \\
\frac{\partial F_2}{\partial y} &= y - x = 0
\end{align*}
\]  

As seen in Figure 1 the repeated plays of the sequential game converge to the Nash Equilibrium at \((x,y) = (1,1)\). This work was extended for sequential games to examine convergence more generally when there are more than two players controlling multiple design variables [58]. In these engineering examples it was shown again that convergence is a function of the relative slope of the designer’s rational reaction sets and control theory was applied for large scale problems [58-59]. A case for nonlinear rational reaction sets was also investigated [60]. While Chanron’s work dealt with sequential games, work by Smith and Eppinger demonstrated a similar principle for games with simultaneous play [61].

A recent extension of this convergence work was performed by Gurnani who demonstrated that the introduction of “mistakes” into the design process could cause some non-convergent problems to converge to a solution [31]. Past work has primarily emphasized determining if a system will converge or diverge and determining the equilibrium point for convergence. However, convergence and equilibrium point provide no information about the designer’s convergence rate. Understanding convergence rate is important in establishing distributed design architectures that converge quickly and consistently to the desired solution.
3.2.2 Convergence Rate

The second major concern for distributed design problems is determining the time required for designers to iterate to a solution. While significant effort has been focused on the convergence characteristics of distributed design problems, much less effort has been applied to the convergence time. One of the major contributions to examining convergence time was an extension of the DeMAID method to reduce the time required for designers to converge to a solution. In this extension, Rogers utilized the global sensitivity equations [62] with a weighting scheme to predict an optimal ordering of designers [63].

The approach taken by Rogers succeeded in reducing the overall design time required by reducing the number of iterative loops. Although there are other alternatives, in DeMAID all designers are ordered sequentially. Another approach to order designers in a purely sequential architecture was presented in [64] which used a genetic algorithm to prescribe the optimal designer ordering.

In contrast to the sequential investigations above, Smith and Eppinger applied a similar approach to minimize the amount of rework required in parallel architectures using an eigenvalue analysis and work transformation matrix [61]. While this approach examines the basic mechanics of the distributed system, it makes no recommendation for the order in which subsystems should iterate to reach equilibrium as quickly as possible. It is demonstrated in Section 3.2.3 that the rate of convergence and number of iterations required to converge depends on the design architecture chosen.

3.2.3 Solution Process Architecture

In this paper the “architecture” refers to the ordering of the solution process. It does not refer to the product architecture, which is an independent and significant area of design research [65]. Instead it refers to the structure of the solution process itself. This structure includes both sequential and simultaneous solution processes. In Figure 2, a simple diagram is shown illustrating the iterative process for a purely sequential architecture, a purely simultaneous architecture and a hybrid approach utilizing both sequential and parallel elements.

Chanron defined convergence criteria for sequential design processes which have the architecture shown in Figure 2. The converged solution is the same regardless of the ordering of the two designers. For parallel design systems, Smith and Eppinger derived the same convergence criteria as Chanron. These results, while not proven explicitly, are assumed to be applicable to hybrid architectures and systems with \( n \) subsystems.

Although there was an assumption in both derivations of a design architecture, the convergence criteria defined hold true regardless of the design system’s configuration. This independency makes them generally applicable, but also means they do not provide significant insight into what influence architecture may have on the rate at which the system converges.

For example the system shown in Eqn. 1 and Figure 1 is configured as a sequential system. The system is said to converge when the minimum change in objective function value is less than 1% and convergence takes 26 iterations. The alternative architecture for a two designer system is a parallel system. Beginning at the same starting point, \((x,y) = (1,4)\) a convergence plot of the design space is shown for the parallel system in Figure 3.

Even for a simple two designer system there is a significant difference in the rates at which process architectures converge to a solution. This simple example demonstrates the influence the choice of process architecture can have on convergence rate.

In a two designer system there are only two potential process architectures. However, as the number of designers increase, there are a large number of potential process architectures that fall into the category of hybrid. For larger
systems there are a wide range of architecture options and very different convergence rates. This can be demonstrated by considering the same problem used by Chanron in his convergence study for large systems [58]. This is an unconstrained five designer problem with sixteen unique design variables. The objective functions and controlled design variables are summarized in Figure 4. The breakdown in convergence times shown on the bars indicates the number of design architectures with a specified range of convergence times.

In Figure 5, the architectures are grouped into bins based on the number of iterations required to converge. The height of the bars indicates the number of design architectures with a specified range of convergence times shown on the x-axis. Figure 5 demonstrates the wide range of convergence rates that can be generated by changing the design architecture. The mean convergence time for the simulated architectures is 25 iterations. There are some cases when minimizing the number of iterations to converge to a solution does not directly correlate to the minimum absolute time to converge to a solution. These scenarios emerge when design tasks take a different amount of time to complete. Investigating how process architecture is influenced by the difference in time required for subsystems to converge is an important area of future research.

There were two unique, but similar, architectures that shared the fastest convergence time of 14 iterations and are shown in the top of Figure 6. The slowest converging architecture took 43 iterations to reach the Nash solution and is also shown in the bottom of Figure 6.

**Designer 1** – $x_1, x_2$

$f_1(x) = 9.41x_1^2 + 1.80x_2^2 + 6.06x_1 + 1.62x_2 + 8.16x_3 + 2.82x_4 + 6.20x_5 + 9.82x_6 + 4.26x_7 + 2.37x_8 + 1.25x_9 + 2.23x_{10} + 3.47x_{11} + 0.23x_{12}$

**Designer 2** – $x_3, x_4, x_5$

$f_2(x) = 6.55x_3^2 + 7.57x_4^2 + 5.68x_3 + 4.29x_5 + 8.84x_6 + 5.25x_7 + 2.13x_8 + 5.34x_9 + 3.58x_{10} + 2.34x_{11} + 4.57x_{12} + 4.12x_{13}$

**Designer 3** – $x_6, x_7$

$f_3(x) = 7.81x_6^2 + 5.49x_7^2 + 5.43x_6 + 7.51x_7 + 7.87x_8 + 4.57x_9 + 4.52x_{10} + 1.23x_{11} + 2.12x_{12} + 3.26x_{13}$

**Designer 4** – $x_8, x_9, x_{10}, x_{11}, x_{12}$

$f_4(x) = 9.88x_8^2 + 9.86x_9^2 + 6.49x_{10} + 9.48x_{11} + 6.4x_{12} + 5.43x_8 + 1.23x_9 + 7.89x_{10} + 0.32x_{11} + 5.43x_{12} + 1.30x_{13} + 1.94x_{14} + 5.75x_1 + 4.32x_{16} + 0.12x_{10} + 4.56x_{11} + 3.26x_{12} + 4.89x_8 + 2.3x_9 + 1.51x_{10} + 3.20x_{11}$

**Designer 5** – $x_{13}, x_{14}, x_{15}, x_{16}$

$f_5(x) = 9.52x_{13}^2 + 7.75x_{14}^2 + 9.93x_{15}^2 + 8.12x_{16} + 2.79x_{13} + 0.5x_{14} + 9.37x_{15} + 6.53x_{16} + 5.43x_{13} + 6.41x_{14} + 0.12x_{15} + 6.27x_{16} + 5.43x_{13} + 1.23x_{14} + 6.77x_{15} + 1.38x_{16} + 4.31x_{10} + 2.64x_{10} + 3.41x_{10}$

**FIGURE 4: FIVE DESIGNER PROBLEM**

**FIGURE 5: FIVE DESIGNER CONVERGENCE TIME BREAKDOWN**

In Figure 5, the architectures are grouped into bins based on the number of iterations required to converge. The height of the bars indicates the number of design architectures with a specified range of convergence times shown on the x-axis. Figure 5 demonstrates the wide range of convergence rates that can be generated by changing the design architecture. The mean convergence time for the simulated architectures is 25 iterations. There are some cases when minimizing the number of iterations to converge to a solution does not directly correlate to the minimum absolute time to converge to a solution. These scenarios emerge when design tasks take a different amount of time to complete. Investigating how process architecture is influenced by the difference in time required for subsystems to converge is an important area of future research.

There were two unique, but similar, architectures that shared the fastest convergence time of 14 iterations and are shown in the top of Figure 6. The slowest converging architecture took 43 iterations to reach the Nash solution and is also shown in the bottom of Figure 6.

**FIGURE 6: CONVERGENCE ARCHITECTURES**

One interesting result of the architectures shown in Figure 6 is the relative strength in the links between the different designers. Conventional wisdom has held that designers with strong links to one another should be arranged sequentially while designers with weak links are arranged in parallel [63]. For example, the other four designers are all strongly linked to Designer 2. However, in the fastest converging architectures, Designer 2 is not arranged sequentially as previous work has suggested is appropriate. Designer 4 is also linked to the other four designers, but in both cases is arranged in parallel with a majority of them. Further, Designer 3 controls design variables influencing Designer 1 and 4 but is not linked to Designer 5. In spite of this, one of the fast converging architectures places it in parallel with Designer 5 and one does not.

Even in the slowest converging architecture, it would be expected that since Designer 2 and Designer 4 are configured sequentially and all other designers depend on Designer 2 and 4’s variables this would be advantageous. Instead it results in an architecture that has the slowest convergence time.

Choosing the correct architecture becomes more difficult when disturbances, in the form of mistakes, are introduced to the system. Further, even if an appropriate architecture is chosen, it may be the case that designers do not rigidly adhere to its structure. In the five designer problem with mistakes studied in the next section, it is assumed that designers do follow the prescribed architecture, even though this may not be the case for real world design problems.

**4 DISTRIBUTED DESIGN CASE STUDY**

In this section a set of simulations are performed using the five designer problem and a variety of different, randomly
created design architectures. These simulations are performed using simulated mistakes as defined by Gurnani [31]. For all the simulations, the normal distribution’s mean was set to be the mistake free design variable value and the standard deviation and probability varied based on the simulation. If a mistake is found to have occurred based on the probability for the simulation, a random variable is generated using the standard deviation and normal distribution to replace the original design variable before it is passed to the other designers. Since the process has stochastic properties, a minimum of 75 trials were performed for all architectures and experiments. Statistical analysis of the samples suggested that this resulted in the actual mean and standard deviation for the process architectures at a 98% confidence level.

In order to extract simple insight into ordering distributed design architectures, a set of different experiments are performed. These experiments do not necessarily correlate to a specific type of mistake, but are meant to broadly capture the overall response of the process architecture to a wide range of potential disturbances. In Section 4.1, the five designer system is simulated with constant mistake properties. In Section 4.2 the same system is simulated with an increasing probability for all designers to make a mistake. For Section 4.3 the designers have a constant mistake probability but the severity is increased for each set of trials. Finally, simulations are performed for the case when only a single designer makes mistakes in Section 4.4. Based on these results some conclusions are drawn about the influence of process architecture on systems with mistakes.

4.1 Constant Errors Simulation

This simulation serves as a baseline to determine if there are any differences between the fastest converging architectures (as shown in Figure 6) for the error free and error inclusive process. It will also provide a benchmark to examine the simulations performed with increasing severity, probability and a single error prone designer. In this simulation the probability of an error occurring was set to 10% and the standard deviation of the normally distributed error was set to 10% of the current design variable value. The results for all the simulations are plotted in Figure 7.

![FIGURE 7: CONSTANT ERROR RESULTS](image)

Similar to Figure 5 there is a large range of convergence rates for the architectures with a general trend that as the mean time increases, so too does the standard deviation. The architecture with the minimum convergence time is circled and took 16.98 iterations with a standard deviation of 3.7 iterations. The fastest and slowest converging architectures are shown in Table 1.

<table>
<thead>
<tr>
<th>Design Architecture</th>
<th>Mean # of iterations to converge</th>
<th>Standard Deviation</th>
</tr>
</thead>
</table>

Although the fastest converging architecture appears to be better than the rest of the architectures, the result is not statistically significant when compared to the second architecture in Table 1 (p-value = 0.703, $\alpha=0.05$). However, when these architectures are compared to any of the others they converge statistically faster (p-value = 0.001, $\alpha=0.05$). It is interesting to note that only one of the two architectures shown in Figure 6 remains the fastest converging architecture, and it suggests that the first architecture in Table 1 may have properties make it better suited at handling designer mistakes. These same properties may be exhibited by the second architecture in Table 1.

The slowest architecture without mistakes is not the same as the slowest architecture with mistakes. Although this result was not significant with $\alpha=0.05$, the difference between the results was appreciable and demonstrates that architectures do have different sensitivity to mistakes. In the next section the error severity is increased incrementally to determine if certain architectures are more robust to changes in error severity.

4.2 Increasing Severity Simulation

In this simulation, the influence of increasing the mistake severity on architecture convergence time is investigated. At each level of mistake severity the rate at which mistakes occurred is set to 10%. Increasing the severity of a mistake is modeled by increasing the standard deviation of the normal distribution used to model the mistake. For the first severity level the standard deviation is set to 20% of the mistake free design variable value (assumed to be the distribution mean). The standard deviation is then incremented by 10%, with the maximum standard deviation being 40% of the design variable’s mistake-free value. The resulting non-dominated architectures identified in this simulation are summarized in Table 2.
Unlike the last experiment, the same architecture performed best for each error level in both mean and standard deviation. As expected, increasing the severity of a mistake also caused an increase in the mean and standard deviation, as more severe mistakes pushed the system further from equilibrium when they occurred. When compared to the experiment with constant mistake rates, summarized in Table 1 the architecture in both Table 1 and Table 2 dominated the other architecture in all three trials in both mean and standard deviation. Future work will focus on developing methods to determine why architectures perform well with low severity levels for mistakes but relatively poorly when the severity is increased.

### 4.3 Increasing Probability Simulation

In this simulation the influence of increasing error probability on the architecture choice is investigated. A constant mistake severity is applied for this problem and modeled as a normal deviation centered at the current design variable with a standard deviation equal to 10% of its value. Three error levels are investigated in this simulation, beginning with a base probability of 20% and incrementing it to 30% and 40%. These levels are chosen to provide a wide view across the spectrum of possible mistake rates and because it is determined that lower mistake rates did not have a significant impact on the overall system behavior. As expected the mean convergence time and the standard deviation increase with an increasing mistake rate. The results for the non-dominated architectures at each error level are summarized in Table 3.

In all the experiments, one of the architectures in Figure 6 performed significantly better than the other in spite of their similar performance for the mistake free case. To gain greater insight into these results only a single mistake is introduced into the system in Section 4.4. Using this mistake the convergence path taken by each architecture is more closely studied to determine why these two architectures are effective at both mitigating mistakes and converging quickly to a solution.

### 4.4 Single Error Prone Designer

The previous three simulations have all suggested that the fastest converging design architecture is the best choice to minimize the impact of errors. To better understand how mistakes propagate in distributed design, in this simulation designer 5 is the only designer that makes mistakes. Designer 5 is chosen as the mistake making subsystem because it is strongly linked to two of the other designers, designer 2 and 3, and completely independent of designer 1 and 4. This provides opportunities to group it with subsystems it is independent of if that results in a faster convergence time. It is also the only subsystem arranged differently for the two fastest converging architectures identified in Figure 6.

For the simulation, Designer 5 was assigned a 10% chance to make a mistake, again modeled as a normal distribution, centered at the mistake free value of the design variable and having a standard deviation equal to 10% of its total value. The simulation in other respects is identical to the two previous simulations for convergence criteria and determining the number of simulations. The simulation results are shown in Figure 8.
For the results shown in Figure 8 there is a grouping of designers with a low standard deviation and low convergence time, but one architecture, circled, dominates the other alternatives in both objectives. This architecture is shown in Table 4, along another architecture that is used for comparison purposes.

The dominant architecture for this simulation is the same as one of the two shown in the top of Figure 6 and is the first entry in Table 4. This architecture has also repeatedly appeared in the other simulations in Sections 4.1, 4.2 and 4.3. In the following discussion this architecture is referred to as Arch-35, because it groups Designers 3 and 5 together. The other architecture in the bottom of Table 4 is referred to as Arch-5. Although both Arch-5 and Arch-35 have identical convergence times for the mistake free system, shown in Figure 6, when mistakes are included Arch-35 converges more quickly and with a smaller standard deviation, Table 4. A statistical analysis of the simulation results is performed using a two sample t-test indicated the difference between the two is statistically significant $\alpha=0.02$.

In the mistake free simulation, both architectures converge at the same rate, 14 iterations. The key difference between these architectures is the placement of designer 3 in parallel with designer 5 in Arch-35. Intuitively it might make sense to isolate the designer making a mistake, but this assumption is what we test in this section.

To better determine how the difference in information flow influences the convergence behavior, Arch-35 and Arch-5 are compared using a single predetermined mistake occurring on the sixth design iteration. This mistake was simulated by adding a randomly determined value, 1.86, to all the design variables controlled by designer 5 at the same iteration for both architectures.

To visualize the impact of this mistake on the system, the Euclidean distance between the current value of the design variables not controlled by designer 5 and the Nash Equilibrium are aggregated using an $L_2$ norm. This aggregation is plotted in Figure 9.

The dominant architecture for this simulation is the same as one of the two shown in the top of Figure 6 and is the first entry in Table 4. This architecture has also repeatedly appeared in the other simulations in Sections 4.1, 4.2 and 4.3. In the following discussion this architecture is referred to as Arch-35, because it groups Designers 3 and 5 together. The other architecture in the bottom of Table 4 is referred to as Arch-5. Although both Arch-5 and Arch-35 have identical convergence times for the mistake free system, shown in Figure 6, when mistakes are included Arch-35 converges more quickly and with a smaller standard deviation, Table 4. A statistical analysis of the simulation results is performed using a two sample t-test indicated the difference between the two is statistically significant $\alpha=0.02$.

In the mistake free simulation, both architectures converge at the same rate, 14 iterations. The key difference between these architectures is the placement of designer 3 in parallel with designer 5 in Arch-35. Intuitively it might make sense to isolate the designer making a mistake, but this assumption is what we test in this section.

To better determine how the difference in information flow influences the convergence behavior, Arch-35 and Arch-5 are compared using a single predetermined mistake occurring on the sixth design iteration. This mistake was simulated by adding a randomly determined value, 1.86, to all the design variables controlled by designer 5 at the same iteration for both architectures.

To visualize the impact of this mistake on the system, the Euclidean distance between the current value of the design variables not controlled by designer 5 and the Nash Equilibrium are aggregated using an $L_2$ norm. This aggregation is plotted in Figure 9.

In this experiment Arch-35 converged in 22 iterations and Arch-5 converged in 26 iterations. Examining Figure 9, Arch-35 has slightly less overshoot than Arch-5 and it progresses smoothly towards the Nash Solution once it reaches its peak distance away from the Nash at 10 iterations. Both Arch-5 and Arch-35 reach a peak distance away from the Nash solution at the same point, but there is a brief increase in distance from Nash by Arch-5 at iteration 13 before it begins converging again. This jump costs Arch-5 several iterations and is a result of the way the changes in design variables flow through the designers in Arch-5. To more easily discuss these flows, the links between each design subsystem are shown in Figure 10.
additional iteration to receive input. This delays the rate at which information flows through designer 3 and into designer 1 for Arch-5 as compared to Arch-35. This difference accounts for the behavior observed in Figure 9 where Arch 5 has a slight increase in the distance to the Nash solution on the 12th iteration.

One implication of the results summarized in Figure 9 is that that to minimize the impact of errors it is important to establish architectures with the smallest maximum distance for an error to propagate to the other designers. This theory can be further supported by comparing a purely sequential process to a purely parallel process. A plot of the Euclidean distance between the design variables not controlled by designer 5 and the Nash Equilibrium are aggregated using and L2 norm. This aggregation for the two systems is shown in Figure 11.

FIGURE 11: DESIGN VARIABLE RESPONSE WITH ERROR

Similar to the previous simulation, a single mistake is introduced into the system. In this case it is introduced on iteration 5 for both the sequential and parallel designer. In the parallel architecture the mistake results in a peak in the distance from the Nash on the 7th iteration and there are no longer large jumps in the distance to the Nash after the 9th iteration. In the sequential process the mistake takes several iterations to complete its propagation and the peak in the error occurs on the 11th iteration, which is a complete cycle through the sequential process. Although more complete cycles through the parallel system are required for the error to reach all designers, the shorter cycle time leads to a drastically lower number of total iterations for the system to adjust to the mistake. This result supports the observation made from analyzing Arch-35 and Arch-5 that reducing the number of iterations required for mistakes to propagate across a design system can reduce the impact of mistakes. In the next section the results from the simulations performed in Section 4 are summarized and several areas of future work are identified.

5 CONCLUSIONS

In this paper designer mistakes are examined in a distributed design system within the context of design process architecture. It is demonstrated that the system architecture can have a significant impact on the overall convergence rate when designers are modeled using the idea of bounded rationality. One of the major contributions of this work is the identification of a controlling factor for architecture convergence rate in distributed design with mistakes. For the design system studied, it is found that the connectivity of the designers is critical to the propagation of mistakes. Mistakes tend to compound themselves in process architectures that take a relatively larger number of iterations to exchange information between their member subsystems. Additionally, it is found that architectures that converge quickly tended to propagate design variable changes quickly across subsystems.

To generalize this conclusion further, experimentation is needed on a variety of different distributed design systems. In these experiments a key concern will be examining the scalability of the conclusions drawn in this paper and investigating if meaningful interactions exist between mistake probability and severity using a two factor experiment. Furthermore, a major goal of this research is to provide guidance to designers selecting design process architectures. It has been demonstrated that a key consideration in this choice is the system connectivity, but an approach to quantitatively evaluate and compare different process architectures is an important area of future research to provide a formal process architecture evaluation.

A potential tool to quantitatively evaluate different process architectures is network theory. In future work, network theory will be used to better understand how the system connectivity can be quantitatively evaluated. The connections in this network based approach are determined by the design variables, but changing design architecture has been shown to influence the rate at which information is shared across the design system. Applying network theory to design teams is not a new idea [67], but it has not been applied in the context of analyzing distributed design process architecture choices. Applying network theory to distributed design and performing a set of experiments similar to those presented in Section 4 using a wide variety of different distributed design systems and process architectures, will enable the validation of the initial conclusions about design process connectivity presented in this paper.

Another area of future work is in examining factors other than mistakes that influence the design process. These disturbances include mistakes, but it will be important to identify other design disturbances that impact a design process. These disturbances can originate from a variety of events. For example, if a designer is unavailable due to a natural disaster, how does this influence the other designers? The ability to map particular events to the disturbances they will cause and apply them to the design system will have many benefits. It will provide a tool for the analysis of real systems and enable more accurate predictions on how events influence distributed design. This knowledge will enable design systems to be established with architectures that can minimize these disturbances impacts through passive rather than active mechanisms. In the same way mistakes, a design disturbance, were damped through architecture choices in this investigation, additional disturbances may be able to be similarly mitigated.
ACKNOWLEDGMENTS

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

REFERENCES


