A COMPUTATIONAL MODEL OF ENGINEERING DECISION MAKING

A Thesis
Presented to
The Academic Faculty

by

Collin M. Heller

In Partial Fulfillment
of the Requirements for the Degree
Master of Science in the
Daniel Guggenheim School of Aerospace Engineering

Georgia Institute of Technology
December 2013

A COMPUTATIONAL MODEL OF ENGINEERING DECISION MAKING

Approved by:

Professor Brian German, Advisor
Daniel Guggenheim School of Aerospace Engineering
Georgia Institute of Technology

Professor Karen Feigh
Daniel Guggenheim School of Aerospace Engineering
Georgia Institute of Technology

Professor Chris Paredis
School of Mechanical Engineering
Georgia Institute of Technology

Date Approved: 30 July 2010

ACKNOWLEDGEMENTS

I am very grateful to the many people who helped me reach this milestone. Specifically to my adviser, Dr. Brian German. His constant support, guidance, and patience have made the last two years a truly valuable experience. I owe many thanks to members of my committee, Dr. Karen Feigh and Dr. Chris Paredis, for their feedback and guidance.

I would also like to thank the members of the German Research Group (GRG); Matt Daskilewicz, who offered invaluable insight and assistance over the last two years, and who is always there to remind me of how little I know; Michael Patterson and David Pate, for their guidance and for putting up with my constant barrage of questions related to my research; Xiaofan Fei, who helped get me through my first year; Marc Canellas, who was always willing to listen and to share ideas. Also, a special thanks to Sean Chait for his assistance with navigating the literature.

This material is based upon work supported by the National Science Foundation under Grant No. 1130222. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the National Science Foundation.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS . . . . . **iii**

LIST OF TABLES **4**

LIST OF FIGURES **5**

SUMMARY **8**

I INTRODUCTION AND MOTIVATION **10**

1.1 Scope and Motivating Requirements 13

II LITERATURE SURVEY **17**

2.1 Design Theory 17

2.2 Empirical Studies of Designers 19

2.2.1 Designer Behavior 19

2.2.2 Designer Cognition 19

2.3 Modeling the Design Process in Organizations 20

2.3.1 Computational Organization Theory 20

2.3.2 Design Structure Matrices 21

2.3.3 Game Theory 21

2.3.4 Multidisciplinary Design and Optimization (MDO) Architectures . 22

2.4 Other Relevant Models 22

2.5 Gaps in the Literature 23

III MIMICKING ENGINEERING DECISION MAKING **25**

3.1 Decision Making Framework 25

3.2 Design Problem 27

3.2.1 Design Alternatives 28

3.2.2 Design Attributes and Incentives 28

3.2.3 Reality Model 29

3.3 Knowledge Model 29

3.3.1 Gaussian Process Models 30

3.3.2 Beliefs in Extrapolation 38

3.3.3 Transforming Beliefs into Gaussian Processes 44

3.3.4 Alternatives Knowledge Models 48
3.4 Preference Model 49
3.4.1 Utility Theory Background and Assumptions 49
3.4.2 Formulating a Designer Utility Function 54
3.4.3 Alternative Utility Function Forms 65
3.5 Integrating the Elements 67
3.5.1 Expected Utility Maximization 67
3.5.2 Alternate Forms 68
3.5.3 Approximating Exploratory Behavior 69
IV ISOLATED DECISION DEMONSTRATIONS 72
4.1 Single Dimension Decision 72
4.1.1 Decisions and Results 72
4.1.2 Analysis of Decisions 76
4.2 Multi-Dimension Decision 78
4.2.1 Baseline Designer 79
4.2.2 Experienced Designer 85
4.2.3 Inexperienced Designer 86
V SEQUENTIAL DECISION MAKING 90
5.1 Illustrative Example 92
5.2 Calculating the Utility of Actions 95
5.3 Generalizations of Alternatives and Attributes 100
5.4 Design Process Influences on Utility 102
5.4.1 Generalized Form 103
5.4.2 Example Utility Models of Time 107
5.4.3 Other Influences on Utility and Alternatives 111
VI IMPLEMENTATION AND DEMONSTRATIONS 113
6.1 Implementation 113
6.1.1 Expected Utility Calculation 113
6.1.2 Sequential Decision Making 114
6.2 Computational Issues 115

6.2.1 Adaptive Sampling . . . 116
6.2.2 Elimination of Dominated Alternatives . . . 118
6.2.3 Eliminating Duplicate Information State and Expected Utility Calculations . . . 119
6.3 Investigations of Time . . . 121
6.4 Strategy in Variable Fidelity Analysis . . . 125
6.4.1 Implementation of Demonstration Problem . . . 127
6.4.2 Results . . . 128
VII POTENTIAL VALIDATION METHODS . . . 132
7.1 Limitations . . . 132
7.1.1 Sequential Decision Making . . . 132
7.1.2 Knowledge Model . . . 135
7.1.3 Preference Model . . . 136
7.2 Usage Applications . . . 137
7.3 Experimental Validation . . . 139
7.3.1 Potential Pitfalls . . . 140
7.3.2 Suggested Guidelines Experiments . . . 143
VIII CONCLUSION AND FUTURE DIRECTIONS . . . 146
8.1 Summary of Contributions . . . 146
8.2 Directions for Future Work . . . 147
8.2.1 Model Improvements . . . 147
8.2.2 Organization Simulation . . . 147
8.2.3 Mechanism Design . . . 148
APPENDIX A — DERIVATION OF GAUSSIAN PROCESS MODEL COVARIANCE FUNCTION . . . 149
APPENDIX B — DERIVATION OF SEPARATED EXPECTED UTILITY EQUATION . . . 153

LIST OF TABLES

1 Designer's prior beliefs regarding the design alternatives 126

LIST OF FIGURES

1 Exponential growth in aircraft production cost over time 11
2 Comparison of weight growth in F-22 and F-18E/F 12
3 Network of 2566 engineering change requests 12
4 Engineering organization as a coupled dynamical system 13
5 Conceptual flowchart of decision making framework 26
6 A simple Gaussian process model with four training points 31
7 Each point in the Gaussian process model is a normal probability distribution 32
8 Random selection of functions from basis 33
9 All basis functions that do not pass through the training points are eliminated 33
10 Learning a new training point 35
11 Gaussian process with an exponential utility function 37
12 Designer's beliefs regarding how CFD error relates to mesh fineness 39
13 How might a designer extrapolate based on the given points? 40
14 Designer's beliefs in extrapolation 41
15 Effect of mean function on extrapolation 42
16 Custom covariance function designed for extrapolation 43
17 Gaussian process with noisy information 45
18 Gaussian process with slope information 46
19 Low fidelity model represented as a Gaussian process 47
20 Different information combined into a single Gaussian process model 48
21 A decision between two lotteries 50
22 Eliciting a utility function for money 51
23 Risk behavior 52
24 Demonstrating utility independence 54
25 Requirements based utility function 58
26 A very strict requirement 59
27 Risk behavior of utility function 60
28 Utility independence of requirements 61
29 Notional utility functions for requirements 66

30 Low fidelity model of y with respect to x 73
31 Expected utility of each alternative and outcome of decision 74
32 Expected utility and outcome of second decision 74
33 Expected utility and outcome of third decision 75
34 Comparision of low fidelity model and true function 76
35 Belief update after outcome of first decision is revealed 77
36 Belief update after outcome of second decision is revealed 78
37 Designer's utility function 80
38 Low fidelity model of TSFC 81
39 Comparison of beliefs and reality in regards to engine weight 81
40 Comparison of beliefs and reality in regards to production cost 82
41 Expected utility of the design space for the designer's first decision 83
42 Updated uncertainty in beliefs after result of first decision is revealed (shown in % of expectation) 84
43 Expected utility of the design space for the designer's second decision . . . 84
44 Expected utility of more experienced designer 85
45 Discrepancy between the inexperienced designer's beliefs and reality in regards to engine weight 86
46 Expected utility of inexperienced designer 87
47 Expected utility for designer's second decision and location of second, third, and fourth decisions 88
48 Expected utility fails to capture the best choice for exploration 90
49 Simple decision between two designs 92
50 Decision between two actions: (I) making a decision and (II) running a high fidelity model 93
51 Example of several possible decision trees which a designer might consider . 97
52 Two-level decision tree with three alternatives and two possible outcomes . 99
53 Utility independence of time and design performance is only valid if utility is calculated at the designer's final decision. 105
54 Time utility function for strict deadline with discounting 109
55 Time utility function for strict deadline with discounting 109
56 Time utility function misaligned with the interests of the organization . . . 110

57 Expected utility variation for a notional probability space with two requirements . . . 117
58 Coarse mesh of probability space . . . 118
59 Refined adaptive mesh of probability space . . . 119
60 Duplicate information state and expected utility calculations . . . 120
61 Designer beliefs based on low fidelity model . . . 121
62 Expected utility of x if the designer was given only one decision . . . 122
63 Expected utility of x for different values of γ . . . 123
64 Shape of expected utility curves . . . 124
65 Expected utility for each alternative available to the designer . . . 128
66 Designer's second decision based on the outcome of the low fidelity model . . . 130
67 Prior beliefs and new information . . . 136
69 Uncertainty is zero at reference point even in the absence of training data at the reference . . . 151

SUMMARY

The research objective of this thesis is to formulate and demonstrate a computational framework for modeling the design decisions of engineers. This framework is intended to be descriptive in nature as opposed to prescriptive or normative; the output of the model represents a plausible result of a designer's decision making process. The framework decomposes the decision into three elements: the problem statement, the designer's beliefs about the alternatives, and the designer's preferences. Multi-attribute utility theory is used to capture designer preferences for multiple objectives under uncertainty. Machine-learning techniques are used to store the designer's knowledge and to make Bayesian inferences regarding the attributes of alternatives. These models are integrated into the framework of a Markov decision process to simulate multiple sequential decisions. The overall framework enables the designer's decision problem to be transformed into an optimization problem statement; the simulated designer selects the alternative with the maximum expected utility. Although utility theory is typically viewed as a normative decision framework, the perspective in this research is that the approach can be used in a descriptive context for modeling rational and non-time critical decisions by engineering designers. This approach is intended to enable the formalisms of utility theory to be used to design human subjects experiments involving engineers in design organizations based on pairwise lotteries and other methods for preference elicitation. The results of these experiments would substantiate the selection of parameters in the model to enable it to be used to diagnose potential problems in engineering design projects.

The purpose of the decision-making framework is to enable the development of a design process simulation of an organization involved in the development of a large-scale complex engineered system such as an aircraft or spacecraft. The decision model will allow researchers to determine the broader effects of individual engineering decisions on the aggregate dynamics of the design process and the resulting performance of the designed artifact

itself. To illustrate the model's applicability in this context, the framework is demonstrated on three example problems: a one-dimensional decision problem, a multidimensional turbojet design problem, and a variable fidelity analysis problem. Individual utility functions are developed for designers in a requirements-driven design problem and then combined into a multi-attribute utility function. Gaussian process models are used to represent the designer's beliefs about the alternatives, and a custom covariance function is formulated to more accurately represent a designer's uncertainty in beliefs about the design attributes.

CHAPTER I

INTRODUCTION AND MOTIVATION

This thesis concerns the development of a computational framework for modeling engineering decision making. The effort to develop this framework was inspired by the needs of a separate research project to simulate the dynamics and performance of an engineering organization[1]. Since our models are designed to fit within the context of the organizational dynamics simulation, the requirements for these models are largely driven by the needs of the simulation. This section will outline the motivation behind the simulation which will in turn define the requirements for this research.

While the aerospace industry has developed numerous successful programs in the past several decades, it has also had its share of setbacks and problems. Cost and schedule overruns are becoming a ubiquitous part of large-scale aerospace development programs. Figure 1 shows Norman Augustine's famous plot of aircraft unit cost over time which has been updated with more recent information from the United States Naval Institute [7, 15]. The cost of military aircraft is growing exponentially, far outpacing inflation. A 2009 Government Accountability Office report found DoD acquisition programs to be an average of 42% over initial budget estimates and delayed by an average of 22 months. Commercial development programs share a similar trend in cost growth, albeit less pronounced than military programs. To maintain competitiveness, the aerospace industry needs to both better estimate and control cost and schedules.

The extant literature identifies many potential causes for these problems, and it is unlikely that all issues are attributable to just a single cause. One relatively unexplored area of research investigates the dynamics of large engineering organizations. Consider the weight growth of the F/A-22 and the F/A-18E/F development programs shown in Figure 2 [73]. The figure shows the percent increase in the aircraft weight from the original desired

[1]NSF Grant 1130222

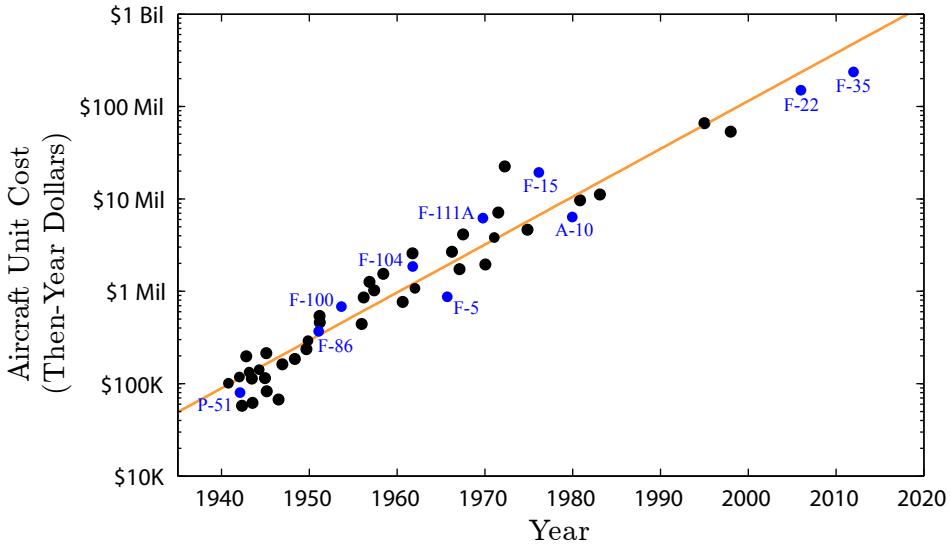


Figure 1: Exponential growth in aircraft production cost over time

takeoff weight over time. The square and circle denote preliminary (PDR) and critical design reviews (CDR), respectively. The dynamics of the two programs have some similarities but also some striking differences. The F-22 remains above the desired weight during the entire design phase and exhibits more pronounced fluctuations; notice the sharp decreases in weight just before PDR and the growth immediately after. The behavior of the F-18E/F remains closer to its target and is generally more tame. At the same time, both programs feature steady weight growth after CDR. These curves share many similarities to plots of a dynamical system's response over time. Theoretically, we could view an engineering organization from the perspective of a dynamical system. If we could identify the features of the program that led to a particular set of dynamics, perhaps we could "design" the organization to behave more like than F-18E/F development program than the F-22.

Modeling an organization in such a way can be challenging, since an organization is made up of numerous entities which are constantly interacting. Figure 3 shows a network of engineering change requests taken from a study regarding change propagation [27]. The research team studied change requests in a US government contract program involving a complex globally distributed system of both hardware and software. Overall the research

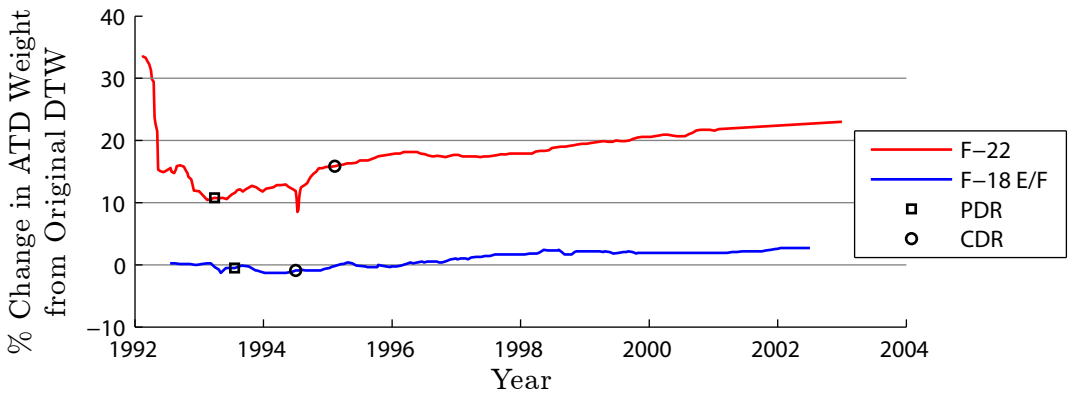


Figure 2: Comparison of weight growth in F-22 and F-18E/F

team documented over 41,500 change requests during the course of eight years. The diagram shows the largest network of change requests, a total of 2,566. This figure illustrates the complexity of large-scale engineering systems; even seemingly isolated design decisions can have extraordinary consequences. Yet, designers are likely unable to perceive the consequences of their own decisions far beyond their own discipline. The research team summed up the complexity of the problem with the following quote:

> *"Through the change network analysis we found that change propagation in large technical systems is actually much more complicated than we thought initially."*

Figure 3: Network of 2566 engineering change requests

The focus of the overarching organization simulation is represented graphically in Figure 4. The performance of an organization and the resulting value of the designed system

are influenced by three major elements: the design problem which includes requirements, organizational goals, and external influences; the decisions of the individual designers and/or design teams; and the dynamics of the organization. These elements, however, are not independent from one another; each is coupled and provides feedback to the others as shown by the bi-directional arrows in the figure. Our ultimate goal is to understand the center of Figure 4, the organization performance and the resulting system value. To enable this capability, this thesis will focus on developing a model of the decision element with the intention of later combining it with the other two elements. In order maintain flexibility within the simulation, the framework was developed to be independent of any specific design problem or design strategy. The simulated designer can make decisions under multiple objectives with or without uncertainty. The framework is robust enough to manage decisions with many feasible solutions as well as decisions without feasible solutions.

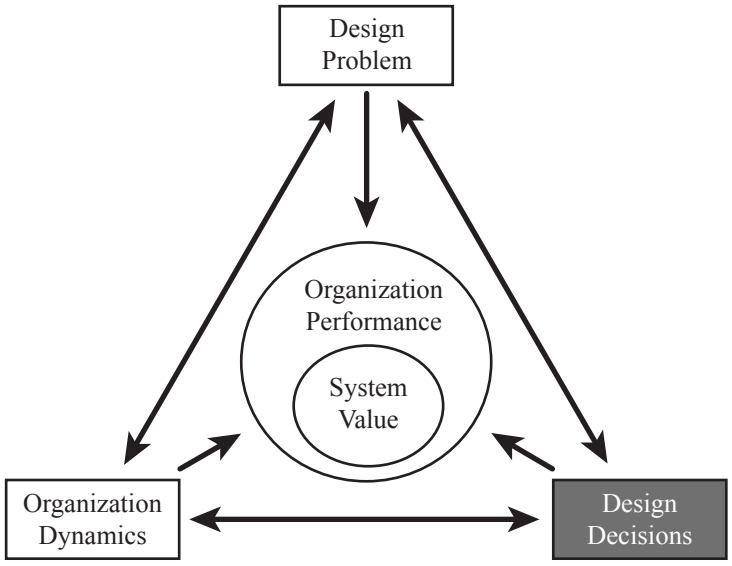


Figure 4: Engineering organization as a coupled dynamical system

1.1 Scope and Motivating Requirements

Before examining the details of the framework, I would like to explicitly define the intended scope. The models used in this framework are intended to be descriptive of human designers, where the word "descriptive" is used as an antithesis to normative and prescriptive. This

framework attempts to mimic the decisions that engineers make in reality, not necessarily the decisions that they should make. Similarly, the framework is not intended to be a formal prescriptive method for arriving at a decision to an actual design problem.

At the same time, this model is not intended to replicate a designer's cognitive reasoning process about the design problem. At no point do I claim that designers follow the process outlined in this thesis when making decisions, nor will I claim that the human brain operates in this manner. Rather, this model is intended to mimic the *outcome* of the reasoning process.

If this model does not accurately represent the manner in which humans reason, then of what use is the model for predicting decisions? In his book, *The Sciences of the Artificial*, Herbert Simon makes an important distinction between the inner environment and the outer environment of a system [64]. To borrow his example, consider both a grandfather clock and a digital wristwatch. These artifacts have completely different inner environments; the parts and mechanisms of one have almost no similarity to the other. However, at the interface of the inner and outer environment, where the user looks at the clock-face to read the time, both clocks provide the exact same functionality. Though they operate in entirely different ways, they are both capable of accurately displaying the time. As long as the inner environment of each clock is well suited for its outer environment, their function will be indistinguishable. This will not be the case at the extremes of their outer environment; for example, if we took both clocks out to sea in a violently rocking ship, the grandfather clock would struggle to keep proper time, while the digital wristwatch would hardly notice the difference. Simon's point is that our models do not have to accurately reflect the inner environment, so long as we are cautious when placing them in an appropriate outer environment.

For this framework, this "interface" between the inner and outer environment is the outcome of the designer's reasoning, his or her final decision which leads to an irrevocable allocation of resources. The outer environment consists of the designer's knowledge, incentives, and the details of the engineering organization. Two major challenges of this research are (1) to determine the appropriate outer environment for the model and (2) to

validate the behavior at the interface. For this reason, care will be taken to list all pertinent assumptions used when formulating the models and their fundamental limitations. To reiterate, the inner environment of this model is not intended to match the inner environment of human thought. This model is *at best* an approximation of decisions that one might plausibly expect from a decision maker.

In order to replicate human decision making for a variety of design problems, the simulation should share some of the characteristics and abilities of its human counterpart. The following requirements were identified as essential characteristics of the framework:

- Models decisions that a human designer would actually make, not necessarily the decision he or she should make (ideally, the framework would also allow for testing of both normative and descriptive models).
- Capable of including the influences of the design problem and organizational dynamics on the designer's decision.
- Independent of a particular ruleset or strategy employed by a designer.
- Of sufficient specificity for computation.
- Capable of specifying relative preferences for designs with multiple attributes. This is especially important when the designer is faced with trade-offs between attributes.
- Capable of storing relevant knowledge about the design alternatives.
- Capable of updating knowledge in the presence of new information.
- Capable of making a decision in both the presence and absence of a feasible design space.
- Capable of developing a strategy when making multiple sequential decisions.

The goal of this thesis is to formulate a framework with appropriate models that meet these requirements.

This thesis is organized as follows: Chapter 2 provides a brief overview of relevant literature on this topic. Chapter 3 introduces the proposed framework for isolated decisions and derives models for implementation. Chapter 4 then demonstrates the framework on two example problems. Chapter 5 extends the framework to account for sequential decisions. This extension is then tested on two sequential decision problems in Chapter 6. Chapter

7 explores the limitations of the model and provides recommendations for improving the model with a human subjects experiment. Finally, Chapter 8 summarizes the contributions and proposes directions for future research.

CHAPTER II

LITERATURE SURVEY

This chapter provides a summary of relevant literature. Our research encompasses a variety of topics from design theory to artificial intelligence. For this reason, a complete and detailed description of all the relevant literature falls beyond the scope of this document. Several pertinent topics are identified and their relationship to the research goal is established. The purpose of this chapter is to provide a concise overview of past research, illustrating the ways in which models from the literature fulfills aspects of the motivating requirements while also showing the ways in which these models are insufficient. After examining the literature I will identify noticeable gaps which this thesis intends to fill.

2.1 Design Theory

For this project, it is important to have a clear understanding of the nature of design and the processes it entails. Forming a clear definition allows for the development of a concise model while still encompassing all the pertinent characteristics of the design process. Although numerous definitions and theories abound, this section will focus only on the work of Hazelrigg, Gero, and Simon. For additional models, schema, and definitions of design, see References [59, 68, 70].

Hazelrigg defines design as a decision-making process [30]. This stands in contrast to other definitions of design which often characterize it as a problem-solving process. Hazelrigg argues that *problems* are independent of the designer's resources and preferences; the solution to a problem is only dependent on the problem statement. *Decisions*, on the other hand, incorporate human values in order to rank alternatives. Since designers must also manage their resources, they are more appropriately viewed as decision makers than problem solvers.

Gero defines design as a process to "transform requirements, which embody the expectations and purposes of the resulting artifact, into design descriptions [25]." Gero's definition

is based on his *Function*, *Behavior*, *Structure* model of design: the function represents the purpose of the artifact; the artifact's behavior encompasses the characteristics that allow it to perform the desired function; finally, the structure is a description of the artifact, usually in the form of engineering drawings. Hence, the purpose of design is to transform requirements (function) into a design description (structure). The difficulty of design lies in the mapping between these three constructs; while a structure maps to a behavior and a behavior fulfills a function, there typically is no direct mapping between function and structure. Gero proposes that designers use "prototypes," canonical mental models of designs that fulfill specific functions, to create this mapping.

Additionally, Gero makes a distinction between different types of design, which he denotes as *routine*, *innovative*, and *creative*. Routine design occurs in a well-defined space of potential designs, a space in which the designer has experience. Innovative design uses the same design variables as routine design, but considers values outside their typical range. Finally, creative design adds to the design space, incorporating entirely new design variables.

One can find similarities between Gero's classification and Simon's distinction between a well-structured and an ill-structured problem [63]. Rather than drawing a distinct boundary, Simon views the structure of problems as a continuum. Well-structured problems are usually characterized by their precisely defined components: the problem statement and goal are clear and explicit, and the problem-solver has a definite procedure for testing each alternative. Ill-structured problems are simply the opposite of well-structured problems. Simon considers most design activities to be closer to the ill-structured end of the spectrum.

Simon also introduces the concept of bounded rationality, the idea that humans often must make decisions with incomplete information, limited computation ability, and a finite amount of time [62]. Since exploring the design space requires allocation of resources, designers are not willing to search indefinitely for an optimal solution; often they know an optimal design is not necessary, and finding the true optimal may not be possible. Instead, Simon argues that humans often use *satisficing* over optimization; designers are interested in finding a satisfactory answer rather than the best answer. While there is much debate over the use of optimization versus satisificing, empirical evidence shows that practicing

engineers tend to satisfice [28, 29].

2.2 Empirical Studies of Designers

Numerous studies have been performed of practicing engineers and design teams to better understand their needs and behavior. This section highlights just a few relevant articles from this large body of literature. The literature has been divided into two very broad categories: designer behavior and designer cognition.

2.2.1 Designer Behavior

There is a large body of literature observing students and practicing engineers both in the field and in a laboratory setting. To give just a few examples, much research has focused on the differences between expert and novice designers [6, 4, 3, 2] and the effects of time and deadlines on performance [58, 51, 61, 49, 72]. Many of these studies do not attempt to explain the mechanisms behind the phenomena but rather give insights into the designers' needs with a goal of developing tools to assist the engineers in designing. Mehalik and Schunn provide a review of numerous empirical design studies [46].

2.2.2 Designer Cognition

In addition to observing designer and team behavior, researchers have also tried to understand the mental processes behind the designer's behavior. A well-known example is design fixation, a designers' tendency to replicate designs that they have already seen and resistance to departing from their original ideas [33, 54]. Other researchers have examined the designers' processes through diaries or think-aloud studies [8, 18]. Cross provides a broad review of research into the cognitive processes of design [17].

The design cognition and behavior literature provide many insights into specific behaviors of engineers. However, the literature is still far from developing a robust model of design decision-making. While this research can be used to validate certain aspects of a model, it likely cannot be used to formulate a decision-making model. For this reason, our models are developed at a higher level of abstraction than designer cognition; our framework does not model the designer's thoughts but artificially mimics the designer's behavior and the

resulting design outcome from this behavior.

2.3 Modeling the Design Process in Organizations

Many researchers have developed models of the design process in organizations using various theories and techniques. This section provides an overview of the most relevant models.

2.3.1 Computational Organization Theory

Perhaps the most relevant literature for this research comes from the field of *Computational Organization Theory* (COT). COT represents members of an organization as information processors, where the role of the members is to transfer information from one form to another. Several platforms have been developed for computationally modeling organizations; Virtual Design Teams [34], for example, was specifically designed to model engineering organizations. Carley and Gasser provide a thorough review of this subject [11].

Virtual Design Teams (VDT) is a computational platform that originated from Stanford in the 1980s [34], and it is likely the closest model to our research topic. VDT was designed with a conscious effort to maintain a relatively high-level of abstraction. Based on research by Galbraith [23], people inside the organization are modeled as information processing units, and the length of each of the organization's tasks is calculated by estimating the time for information processing. The model defines numerical parameters for capturing elements of the problem and organization such as "task complexity" and "skill of the designer". Tasks are encoded in terms of work volume, a unit that represents the amount of information processing work.

While VDT has been successful in modeling organizations, it is unable to capture specific design decisions. The model abstracts away the actual design problem and treats the design process simply as a task to be performed. Setbacks and rework are more or less seen as random events with no tenable cause. In this way, VDT is able to capture the overall dynamics of the organization but gives little insight into the performance of the designed system. The models also rely considerably on the modeler's ability to specify all the relevant tasks and their characteristics ahead of time.

2.3.2 Design Structure Matrices

Design Structure Matrices (DSM) are a visual representation of information dependencies and feedback in a system [21]. Tasks in the system are represented in both rows and columns of a square matrix. The diagonal matrix entries represent task completion times. Off-diagonal terms represent dependencies among tasks. Smith and Eppinger use the design structure matrix extensively to model sequential iteration in engineering design [65, 66]. Each task has a probability of repetition related by the strength of task coupling; this allows for an estimation of rework in an engineering design. DSM allows for the reordering of tasks and restructuring of design decomposition in a system to reduce feedback and can provide estimates of total development time [20, 52].

2.3.3 Game Theory

Certain authors have studied decentralized design in the context of game theory. Lewis and Mistree use various communication protocols from game theory such as cooperative, non-cooperative, and Stackelberg leader-follower [43, 42]. From these protocols, the authors simulate organizations and analyze characteristics such as convergence and performance of the resulting design. Given a non-cooperative design "game," authors have investigated the conditions under which the design problem will converge [12, 13].

The use of game theory seems intuitive, since the design process in a large organization is composed of many players with potentially conflicting interests, each bearing an influence on the final design. However, the validity of the use of game theory lies in the extent to which designers perceive their actions as a game. As Rasmusen states, "Game theory is not useful when decisions are made that ignore the reactions of others or treat them as impersonal market forces [55]." If designers do not perceive their actions as being part of game, then the resulting behavior is more typical of a dynamical system. In fact, the authors of this research consider only relatively naïve strategies and not optimal strategies since the designers are given incomplete information. While this research ostensibly captures all three elements in Figure 4, the current literature tends to ignore formulating how a designer comes to a particular solution and assumes arbitrary problems and objective functions.

2.3.4 Multidisciplinary Design and Optimization (MDO) Architectures

Several types of MDO architectures were inspired by the protocols and hierarchies of engineering organizations. Collaborative Optimization (CO) is an optimization scheme that divides the optimization along disciplinary lines, similar to the divisions in an engineering organization [38]. Each disciplinary analysis has its own optimizer and is only concerned with its local design variables and constraints. Coordination and consistency are maintained by a system level optimizer.

Analytic Target Cascading (ATC) is an optimization scheme similar to CO [40]. In this framework, a top-level optimizer sends children nodes a set of targets, both for objectives and consistency requirements. Each child node can run a unique optimizer, specifically tailored for its design sub-problem. The child nodes can also send targets to its own children, allowing for an unlimited number of hierarchical levels.

While not an MDO architecture, Compromise Decision Support Problem (DSP) is a optimization technique from the goal programming literature [47]. Compromise DSP uses goals rather than objectives, similar to the way that engineers are often given requirements. "Hard" goals, or rigid requirements, are used as constraints on the optimizer. "Soft" goals are then rank ordered by importance; the optimizer uses a lexicographic minimum to determine the "optimal" design.

2.4 Other Relevant Models

Researches have also constructed models of specific problem-solving behavior. For example, Cagan and Kotovsky observed people repeatedly playing Tower of Hanoi problems and used simulated annealing-based algorithms to mimic the observed behavior [10]. Olson used a combination of simulated annealing and genetic algorithms to create a multi-agent simulation of an organization based on observations of a design team [50].

The field of Artificial Intelligence was explored with mixed results. Machine learning techniques were found to be useful for certain aspects of the framework, as described in Section 3.3. Artificial intelligent systems are becoming more commonplace as decision

support systems. Expert systems were prolific in the 1980s and are highly effective for well-structured problems with a relatively narrow focus [45]. Schank and Abelson's *scripts*, *plans*, and *frames* construct for storing and recalling knowledge is still widely used in systems today [60]. For modern design support systems, the concept of *knowledge-based engineering*, or the process of integrating expert knowledge into computer systems, has become increasingly popular with tools being developed for industry such as Adaptive Modeling Language. In general, the Artificial Intelligence community has been most successful in decision making when focused on a relatively narrow and domain-specific problem. Few, if any, models exist for robust automated design decision-making for generalized design problems.

Perhaps the most significant unanswered question in the modeling literature is in regards to the designer's objectives. In almost all the examples, the model assumed an objective function was already given or ignored the concept of objectives altogether; in other words, the designer had already determined what the "best" design would be in terms of just a single numerical attribute. Each technique represents more or less a method for searching and moving through a design space. Most of the models described do not account for uncertainty and are only concerned with a single attribute, despite the fact that designers often face multiple objectives and an uncertain future.

2.5 Gaps in the Literature

Based on the analysis of the literature, we can identify several key areas where the extant research falls short of the motivating requirements:

- *Connection with the Design Problem*: Much of the outlined literature treats the design problem as an abstract task. This is especially true with VDT and Design Structure Matrices; in both these examples the tasks can consume a variable amount of time, but the outcome of the task is unknown. Since the details of the problem are not accounted for, we are unable to compare the quality of designs between simulations.
- *Appropriate Level of Abstraction*: Other areas of research examine design at a level too detailed for practical simulations. Gero's prototypes, for example, require very detailed knowledge of the designer's process; if one were to implement such a model,

they would need to develop a large number of prototypes, even for relatively simple design problems. The quality of the designer's decision would be highly dependent on the modeler's ability to create effective prototypes.

- *Capturing Designer Preferences*: For the literature that examines the quality of the resulting design, the preferences and/or objective function of the designer is either pre-defined or unaccounted for. None of the above examples detail how a designer's preferences are developed. Lewis's game theoretic approach, for example, assumes arbitrary objective functions in implementations. Much of the empirical research focuses on the designer's behavior but does not address what influences specifically drive his or her behavior.
- *Accounting for Uncertainty*: A practicing designer must sometimes make decisions without perfect knowledge of the future and the implications of their decisions. In almost all the models outlined above, the information available to the designer is assumed to be deterministic and to perfectly model reality.

CHAPTER III

MIMICKING ENGINEERING DECISION MAKING

From the literature survey, it is evident that the extant research is largely insufficient in satisfying all the needs of the simulation outlined in the motivation on page 15. The purpose of this chapter is to systematically develop a framework for mimicking decisions made in isolation. The first section of this chapter will outline the decision making framework along with the necessary models and information required for the simulation. Subsequent sections will expound on these models: Section 3.2 explores the relevant information needed about the problem to form a decision; Section 3.3 proposes a particular knowledge model and illustrates its potential uses; Section 3.4 leverages utility theory to derive a preference model for a requirements-driven organization. Finally, Section 3.5 synthesizes these models into a single equation. Chapter will 5 expand the framework to encompass situations in which a designer is able to make multiple sequential decisions where the result one decision may influence the next.

3.1 Decision Making Framework

In his book, *Fundamentals of Decision Making*, George Hazelrigg identifies three elements shared by all decisions [32]: (1) alternatives, (2) an expectation of the future, and (3) preferences. Fundamentally, the task of this thesis is to develop a framework that incorporates each of these elements. This thesis will take a modular approach to this problem; each element can substituted or refined without affecting the others. The following three sections will develop each of these elements further and propose a model to represent them. For the purposes of this thesis, I have redefined Hazelrigg's three elements as follows:

1. *The design problem*: Contains the alternatives, incentive structure, relevant design attributes, and any additional information which the designer can access.
2. *The designer's knowledge*: Provides a mapping between the alternatives and the designer's beliefs about the attributes of each alternative.

3. *The designer's preferences*: Captures the designer's risk behavior and controls how he/she manages trade-offs in conflicting objectives.

Representing each of these elements in a computationally tractable manner is the main challenge of developing this framework.

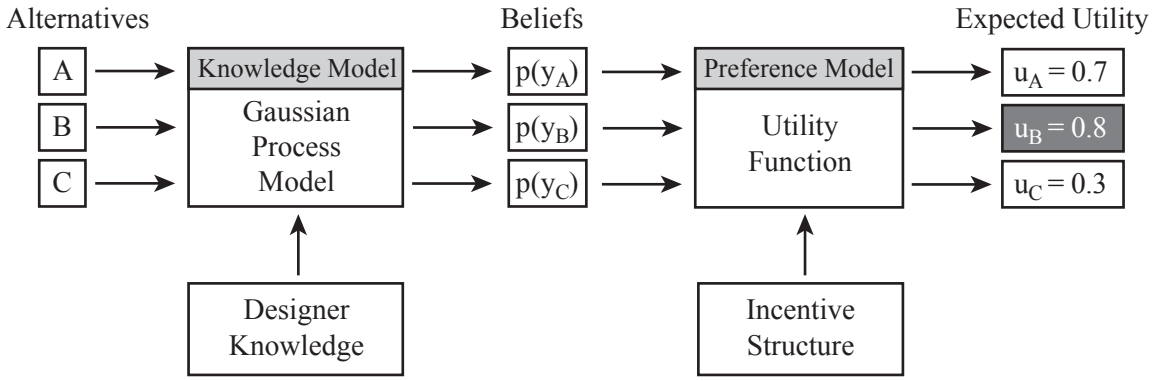


Figure 5: Conceptual flowchart of decision making framework

The overall premise of this chapter is represented by the flowchart in Figure 5. The task of the framework is to transform a set of design alternatives, which I will refer to as $\boldsymbol{x}$, into a design decision, π (shown in the figure as the shaded alternative). As Hazelrigg states, a designer's choice should be dependent on his or her expectation of the future and his or her personal preferences. These components comprise the two essential models in the decision-making framework: the *knowledge model* and the *preference model*. As the flowchart indicates, the purpose of the knowledge model is create a mapping between the alternatives and the designer's beliefs about their attributes given the designer's knowledge. If the designer's beliefs are uncertain, then the designer assigns a probability to each outcome, and his or her beliefs are represented by probability distributions; these probability distributions are subjective and based on the information and biases in the designer's knowledge. Once the beliefs are represented in this probabilistic form, they serve as inputs to the preference model to obtain the designer's decision based on the his or her incentives and values. The output of the preference model is a scalar value for each design alternative. The designer's decision, π, is the alternative with the highest value from the preference

model. I hypothesize that if appropriate models can be developed to represent knowledge and preferences, then human behavior can be approximated in this manner.

We can summarize this strategy in the equation of *subjective expected utility* [22]. Given a set of alternatives, $\boldsymbol{x}$, each with a set of possible outcomes, y, the decision maker will choose the alternative with the highest expected utility:

$$\mathrm{E}[u(\boldsymbol{x})] = \sum_i u(y_i)p(y_i) \tag{1}$$

where u is the decision maker's personal utility function and p is a subjective probability of achieving an outcome based on the designer's beliefs. From the equation, we can see that different designers can make different decisions if they have different preferences, $u(y_i)$, or different beliefs about the alternatives, $p(y_i)$. In this way, we can model designers on an individual level by uniquely specifying their personal beliefs and preferences.

Before proceeding, it is important to have a clear understanding of what decision the designer is actually making. In this chapter, the designer will be making a *final* design decision. In other words, after the designer makes his or her decision, he or she irrevocably allocates resources towards developing the chosen artifact. We could imagine that the design goes into production after the designer's decision or proceeds to the next phase of design. The designer does not anticipate that any changes will be made to the design after his or her decision. The derivation of sequential decision making in Chapter 5 will more clearly show why the assumptions in this chapter constrain the framework to only final decisions.

3.2 Design Problem

The first element of any design decision is the design problem itself. In this thesis, I will assume that the design problem is given to the modeler. As we will see in this chapter, this assumption greatly simplifies the framework, but also introduces some limitations. Since the design problem is given to the modeler, I will not develop any "model" of the problem; instead, I will identify the necessary information required to represent the problem in the framework.

3.2.1 Design Alternatives

The alternatives represent the choices available to the designer. In this chapter, our definition of alternatives will be limited strictly to discrete and continuous design variables. In the most general form of the framework, the alternative space can be much larger, encompassing the set of all possible actions that a designer could perform. However, in the context of this formulation, performing actions and receiving outcomes strictly applies to sequential decision making and falls outside the scope of isolated decisions. Therefore, I will defer a full discussion of alternatives until Section 5.3.

One limitation of this framework is that all the designer's choices must be predefined. The designer may choose a design in a rarely explored portion of the design space, but he or she is still confined to the design space and the parameterization embodied therein. To borrow Gero's terminology, this restriction limits the designer to *routine* and *innovative* designs while precluding *creative* designs.

3.2.2 Design Attributes and Incentives

The designer has preferences for one alternative over another based on the alternatives' attributes. The relevant attributes and the designer's preferences are driven by the designer's own goals and by the incentive structure of the organization. For example, an engineering organization often issues requirements for specific design attributes to its design engineers. The designer is incentivized to meet the requirements, and will therefore prefer alternatives that meet design requirements over alternatives that do not meet the requirements. Like the alternatives, attributes could also be indirectly related to the design. For example, the organization's management may give the designer a schedule deadline. Meeting the deadline is typically not considered an intrinsic property of a design, but the deadline may still influence which alternative the designer chooses. In this chapter, all attributes and preferences will be driven solely by requirements on the characteristics and performance of the design itself. In Chapter 5, the preference model is expanded to account for time and budget.

3.2.3 Reality Model

A reality model returns an outcome to a designer in response to an action. This model is only applicable in the context of sequential decision making introduced in Chapter 5 but is included here for completeness. This model returns a deterministic outcome for a given action. For example, if a designer runs an analytical model, the reality model will return the results of the analysis. If the modeler has a model of the designer's external environment, this model can be used to determine the final outcome of the design after multiple decisions.

3.3 Knowledge Model

In the context of this framework, we can view the simulated designers as information processors: they have information about the design alternatives before beginning the design process (their prior knowledge); they obtain knowledge throughout the design process (learning from models, experts, etc...); and they make design decisions based on the information available to them. In order to model a human decision maker, the framework requires a knowledge model that can store, retrieve, and make inferences from information. Specifically, a knowledge model should be capable of:

- *Representing the designer's beliefs including uncertainty*: The designer has beliefs about the relationship between alternatives and attributes based on his or her experience, education, and resources. These beliefs may be uncertain, especially if the designer lacks experience with a particular design problem. A designer might be more certain of one belief than another.
- *Inferring the properties of alternatives given information regarding similar alternatives*: A designer might not have specific knowledge of a particular design, but may have knowledge about similar designs. For example, if a designer has analyzed two airplane designs, one with a wing aspect ratio of 10 and another with an aspect ratio of 9, he will likely estimate that than airplane with an aspect ratio of 9.5 has properties in between the two analyzed designs, assuming all other properties are the same.
- *Updating knowledge in the presence of new information*: The design process is typically a learning process; the designer is constantly receiving and processing new

information that enables him or her to make a more informed decision. To accurately model the design process, the knowledge model should be capable of receiving new information and combining it with prior information to form inferences.

- *Storing knowledge obtained from multiple sources*: A designer may draw on multiple resources to obtain information about a design. These could include, among others, analytical and computational models, prior design experience, and expert opinions. A model should have the flexibility to store information obtained from these resources both individually and combined.

In this section, I propose the use of Gaussian process models as a technique for representing a designer's knowledge. Section 3.3.1 will introduce the fundamental concepts behind Gaussian process models. Section 3.3.2 will customize these models to more accurately represent a designer's beliefs. Section 3.3.3 will demonstrate how varying forms of knowledge can be encoded into a Gaussian process model. Finally, Section 3.3.4 will explore some limitations of the models and explore alternative candidates for representing knowledge.

3.3.1 Gaussian Process Models

One possible method for storing and learning information is through the use of *Gaussian process models*. In the optimization and geostatistics community this method is primarily known as *kriging*. However, since this research draws heavily on literature from the statistics and robotics communities, I will refer to these models as Gaussian process models. Although there are slight differences between the specific forms of kriging, these terms are generally interchangeable. Rasmussen and Williams have written a comprehensive book on Gaussian process models which serves as the primary resource for this section [56].

Rather than encode specific rules and equations of design analysis, Gaussian process models encode the designer's knowledge at a higher level of abstraction. When determining the designer's decision we are concerned with only two types of beliefs: (1) the designer's expectation of how the alternatives relate to the attributes and (2) the designer's confidence in his or her expectation.

The mapping between the alternatives and expectations is dependent on the designer's knowledge, available information, and the tools he or she can access. Using Gaussian process models, we can model different designers with varying levels of "experience". An expert designer will have more accurate expectations about the mapping between alternatives and attributes. Expert designers will also tend to have less uncertainty in their beliefs. Novice designers will have inaccurate expectations and large uncertainty. A novice designer might have inaccurate beliefs with low uncertainty, thinking their beliefs to be correct when, in fact, they are incorrect.

3.3.1.1 Model Basics

A Gaussian process model is a non-parametric, supervised-learning, regression technique. The goal of this model is to infer the value of a function, $f(\boldsymbol{x})$, at a set of test points, X, using a set of training data which I will denote as X_{tr} and $\boldsymbol{f_{tr}}$. Unlike many other regression techniques, Gaussian process models do not return a single value at a test point, but rather a probability distribution. Since the model assumes a normal (or Gaussian) distribution at every location, these models are referred to as a Gaussian process models.

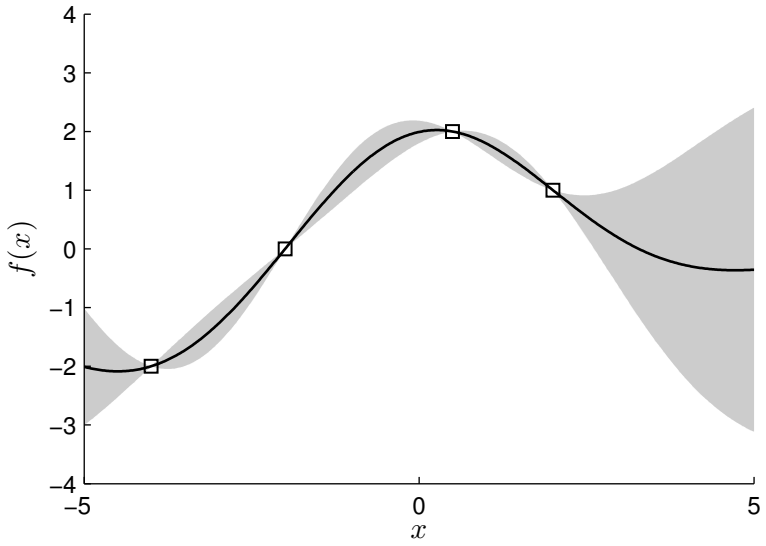


Figure 6: A simple Gaussian process model with four training points

Figure 6 illustrates a basic example of a Gaussian process model. Four training points

are used to define the model shown as the black rectangles. In between these training points, the Gaussian process model infers the shape of the function. Each point in the model is a normal probability distribution; the expected value is shown as the black line and a 95% confidence interval is shown in the shaded region. In Figure 7 the Gaussian process in Figure 6 has been rotated, and the normal distributions are drawn explicitly. Notice that the probability distributions at the training locations collapse to a single point as the values are known with certainty. In between training points, the uncertainty grows since the function value at these locations is known with less certainty.

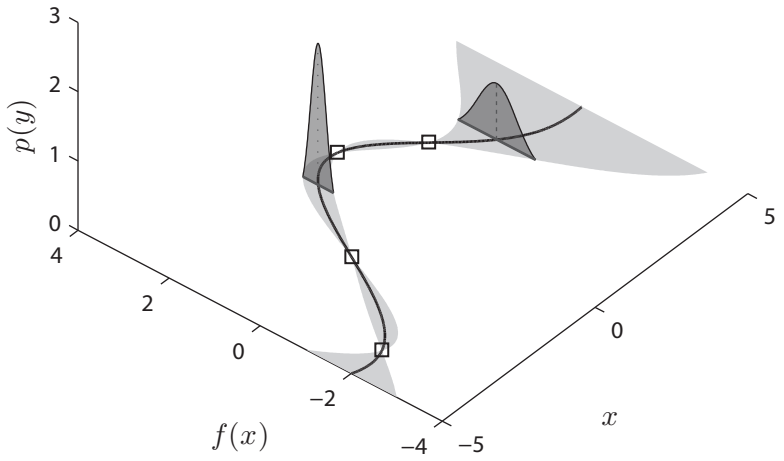


Figure 7: Each point in the Gaussian process model is a normal probability distribution

Rasmussen and Williams outline two interpretations of Gaussian process models [56]: the *function-space view* and the *weight-space view*. For this discussion, I will use the function-space interpretation as it more intuitively aligns with our purpose. Gaussian process models assume a prior distribution on a (often infinite) set of basis functions. Figure 8 shows numerous basis functions chosen at random from the infinite set. As in Figure 6, the shaded gray region represents a 95% confidence interval; in other words, 95% of all basis functions at any point will lie within the gray region. Any of these functions shown could potentially be the "true" underlying function. Depending on the prior chosen, certain functions are assumed to be more likely than others to be the true function. When training points are added to the model, all functions that do not pass through those particular points are

eliminated from the set of possible functions. This is done through a Bayesian inference; if a function does not pass through a training point, the probability of it being the true function is zero. The probability distribution for the remaining functions is updated via the Bayesian inference, yielding a probability distribution at each location. In Figure 9, three training points have been added, and all basis functions that do not pass through the training points have been eliminated. Notice that the confidence interval has been updated accordingly.

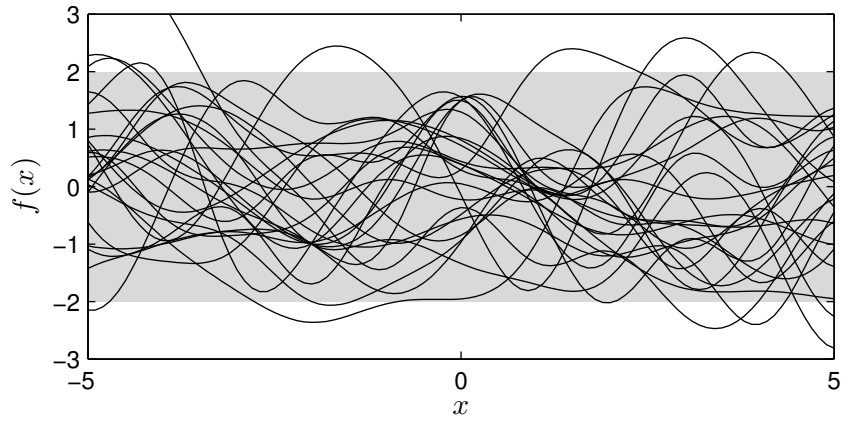


Figure 8: Random selection of functions from basis

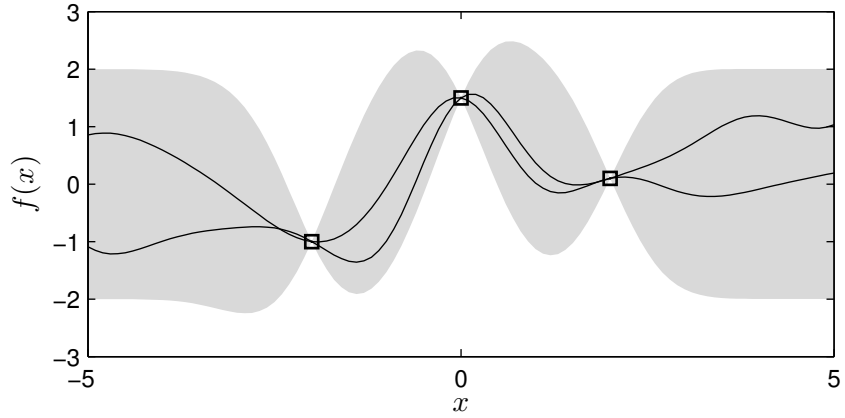


Figure 9: All basis functions that do not pass through the training points are eliminated

The function-space interpretation makes a convenient analogy with the knowledge and information that engineers typically use. Consider an engineer designing the wing of an

airplane. A designer may not know the precise relationship between aspect ratio and drag, but the designer probably knows that drag tends to decrease with increasing aspect ratio. Therefore, any function between these attributes should have a negative slope; we can eliminate the set of all functions that have positive slope (or consider them highly unlikely), because a positive slope does not fit the designer's "training data," his experience and education. See Section 3.3.3 for more examples of capturing types of knowledge.

The prior distribution of basis functions is entirely determined by a mean function and a covariance function. For our purposes, we will assume that the mean is always zero (this particular instantiation is known as *simple kriging*). Therefore, in the absence of any training information, the model would predict each test point to have a mean of zero and a variance given by the covariance function, $k(x, x^*)$. Loosely speaking, the covariance function specifies the relationship between the function value of two points based on their x-coordinate location. In Figure 6, a one-dimensional squared-exponential covariance function was used:

$$k(x, x^*) = \sigma^2 \exp\left(-\theta^2 (x - x^*)^2\right) \tag{2}$$

This function essentially states that the covariance between two points is a function of their Euclidean distance; the closer the two points, the more strongly correlated the function values are at those points. σ and θ are referred to as *hyperparameters*. A general discussion of hyperparamters and covariance functions is deferred to the next section.

A covariance matrix, K, is formed by determining the covariance between two sets of points. For example, the (i, j) element of $K(X, X^*)$ is equal the covariance of the i^{th} element of X and the j^{th} element of X^*. For concise notation the covariance matrix $K(X, X^*)$ will be written as K_{X,X^*}, and $K(X, X)$ will simply be written as K_X. A full derivation of the Gaussian process model equations is not contained in this text; however, it can be shown that any set of test points has the following distribution [56]:

$$\boldsymbol{f} \Big| X, X_{tr}, \boldsymbol{f_{tr}} \sim \mathcal{N}\left(K_{X,X_{tr}} K_{X_{tr}}^{-1} \boldsymbol{f},\ K_X - K_{X,X_{tr}} K_{X_{tr}}^{-1} K_{X_{tr},X}\right) \tag{3}$$

If the training points are not known with absolute certainty (such as in physical experiments

or robotic sensing applications), a noise parameter, σ_n, can be included in the formulation:

$$\boldsymbol{f}\Big|X, X_{tr}, \boldsymbol{f_{tr}} \sim \mathcal{N}\big(K_{X,X_{tr}}(K_{X_{tr}} + \sigma_n I)^{-1}\boldsymbol{f},\ K_X - K_{X,X_{tr}}(K_{X_{tr}} + \sigma_n I)^{-1}K_{X_{tr},X}\big) \quad (4)$$

where σ_n represents the standard deviation of the function from the training data. In optimization involving computer experiments, this term is often set to zero since the computer models are deterministic. However, including a small amount of noise at each point can greatly improve the numerical stability of the calculations.

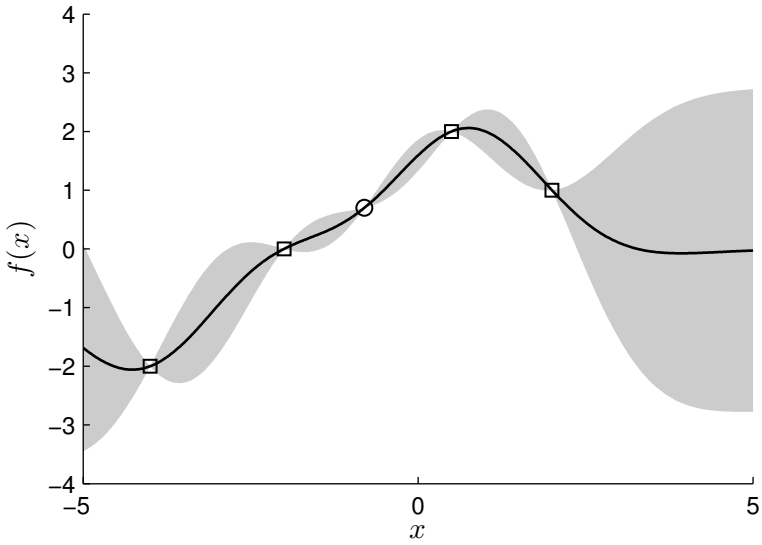


Figure 10: Learning a new training point

In Figure 10, another training point has been added to the original training set shown in Figure 6. Gaussian process models are quite flexible to adding information; although this point falls outside the original 95% confidence interval, the model adjusts to account for it. Notice that, in certain areas, the uncertainty has grown in the updated model even with the additional information. Since the new training point was much lower than predicted, the model has learned to expect the true function to have more variation than it originally believed.

From the examples shown, we can already see that Gaussian process models meet many of the requirements outlined at the beginning of this section:

- *Representing the designer's beliefs including uncertainty*: Since the output of a Gaussian process model is a probability distribution, the model is capable of representing the designer's subjective beliefs even if those beliefs are uncertain.
- *Inferring the properties of alternatives given information regarding similar alternatives*: Since Gaussian process models are a regression technique, the model can make inferences about designer variables for which it may not have explicit information. Furthermore, the model recognizes that there is uncertainty in its inferences.
- *Updating knowledge in the presence of new information*: As depicted in Figure 10, the model can update its beliefs if given new information.

Section 3.3.3 will demonstrate how knowledge can be stored from multiple sources.

3.3.1.2 Covariance Functions

As mentioned above, the covariance function is critical in determining the shape of the Gaussian process model. Each covariance function has specific properties that make it well-suited for modeling certain types of underlying functions and poorly suited for others. The covariance function chosen for a particular application depends on the modeler's prior knowledge about the underlying function.

One of the most commonly used covariance functions is the squared exponential covariance function shown earlier in Equation 2. This covariance function is an example of a *stationary* covariance function: stationary covariance functions depend solely on the distance between two points. In other words, they are invariant to translations in the design space. The most common non-stationary functions use the dot product of the training points; these functions are invariant to rotations about the origin.

The squared exponential covariance function also assumes a continuous and infinitely differentiable underlying function. The *exponential covariance function*, on the other hand, is a stationary covariance function that allows the slope to be discontinuous.

$$k(x, x^*) = \sigma^2 \exp\big(-\theta|x - x^*|\big) \tag{5}$$

Figure 11 shows this covariance function used for the same set of training points as in Figure 6. Notice that the expectation is not smooth and that the uncertainty is much

higher than in Figure 6. Reference [56] contains numerous examples of commonly used covariance functions.

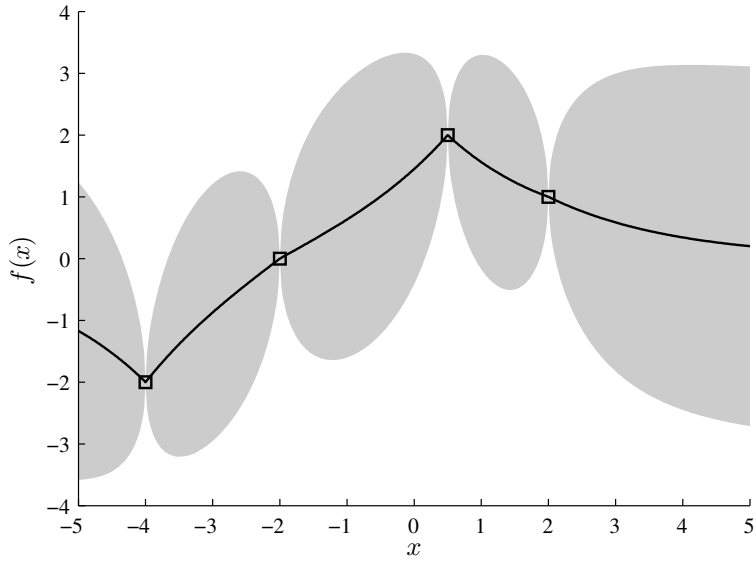


Figure 11: Gaussian process with an exponential utility function

In Equations 2 and 5, σ and θ are called *hyperparameters*. These terms control certain characteristics of the shape of the Gaussian process model. For the squared exponential covariance function, θ is a measure of how quickly the function can change and controls the non-linearity of the function; large values of θ indicate that the underlying function can have rapid curvature changes. σ sets the maximum possible variance at a point and controls how rapidly the variance grows. Since these parameters can have a strong influence on the shape of the Gaussian process model (and the resulting design decisions), we require a robust method of determining an appropriate value for them.

In the optimization community, a common method for setting hyperparameters is to maximize *marginal likelihood*. By maximizing marginal likelihood, we are finding the value of hyperparameters that best explains the training data. For numerical stability, many authors recommend maximizing the log likelihood [56], given by the equation below:

$$\log p(\boldsymbol{f}_{tr}|X_{tr},\Theta) = -\frac{1}{2}\boldsymbol{f}_{tr}^{T}K_{tr}\boldsymbol{f}_{tr} - \frac{1}{2}\log|K_{tr}| - \frac{m}{2}\log 2\pi \tag{6}$$

where m is the number of training points.

When selecting a method for setting hyperparameters, the fundamental questions to ask are, "To what extent does this method replicate the way humans transform information (training data) into beliefs?" and "To what extent do the value of the hyperparamters affect the design decision." Without validating data from a human subjects experiment, any argument for a particular method would be based largely on speculation. In the examples shown in this thesis, the value of the hyperparameters were set via maximizing marginal likelihood. This method was chosen since it naturally forms a balance between fitting accurately fitting the training data (the first term in Equation 6) and penalizing complexity (the second term in Equation 6). The method is also easily implemented computationally. For other methods of fitting hyperparameters, refer to Reference [56].

3.3.2 Beliefs in Extrapolation

When used in optimization, most optimization algorithms recommend a *warm start*: a space filling design of experiments to provide the Gaussian process model with training data [35]. In an actual design scenario, the designer may not have information over the range of design variables in question, especially at the outset of the design process. This is especially true when the designer is considering design variables outside a typical range (such as an turbofan designer developing an engine with a larger bypass ratio than previous models) or using novel design variables altogether (such as an airframe designer considering a blended-wing body design). If a designer has information over a subset of the range of design variables, how does this knowledge influence his or her beliefs over the unknown range?

The answer to this question depends largely on the designer and the specific situation. In many cases, the designer may have some general knowledge and intuition, even if he or she has not dealt with the specific design problem at hand. For example, a aerodynamics expert knows that increasing the fineness of a Computation Fluid Dynamics (CFD) mesh will result in a more accurate solution, and the expert can estimate this increase in accuracy. However, the designer likely knows that this trend will not continue indefinitely; at some point, the introduction of round-off error from the numerical approximation will dominate

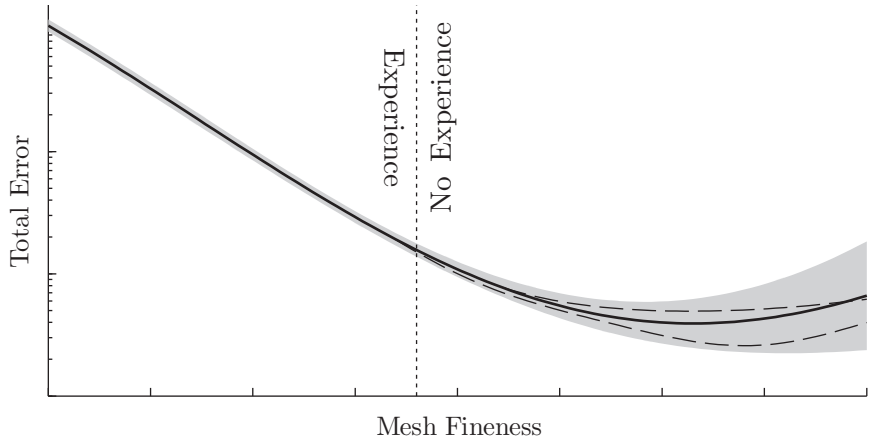


Figure 12: Designer's beliefs regarding how CFD error relates to mesh fineness

the reduction in error from using a finer grid. In other words, the designer knows that the relationship between error and fineness, while initially decreasing, will reverse. Perhaps the designer's beliefs can be represented by the plot in Figure 12; the designer has run the model for a different levels grid fineness and has seen the data converge. However, passed the dotted line, the designer is inexperienced. He knows that eventually round-off error will dominate, but does not know the exact shape of the error curve in Figure 12. The dashed lines show two possible shapes that the underlying function could have; the designer would consider each of these shapes plausible. In scenarios like the example given, the designer's knowledge in extrapolation can be encoded as noisy training data, representing an estimate of the underlying shape of the function, but with uncertainty regrading the true value of a function.

Consider, however, the scenario where the designer has absolutely no knowledge of the trend in extrapolation. Perhaps the relationship between the alternatives and attributes is a black box, so convoluted that the designer is unable to infer a general underlying relationship. An example is shown in Figure 13; the designer only knows the points shown by the squares and is trying to infer the value at $x > 3$ and $x < -4$. In this case, the designer might draw on the *Principle of Indifference*; when faced with n mutually exclusive and exhaustive possibilities, the probability of each possibility is equal to $1/n$

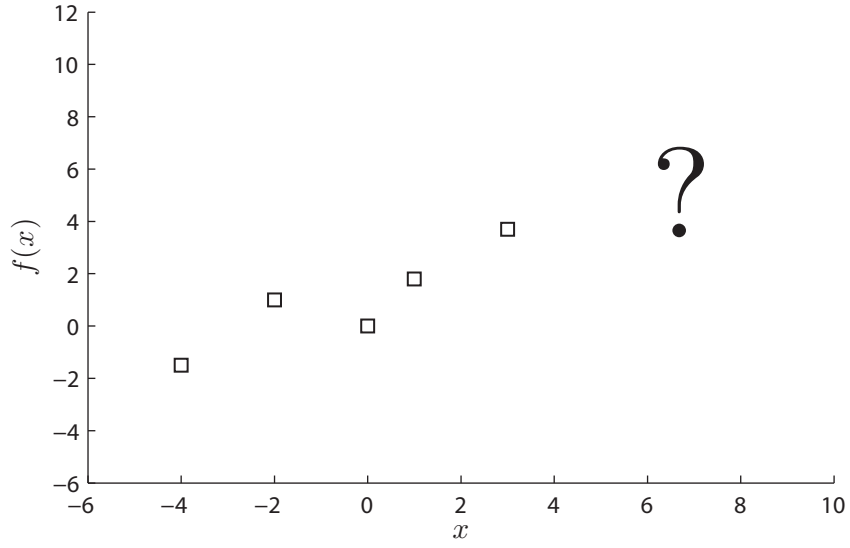


Figure 13: How might a designer extrapolate based on the given points?

in the absence of information to suggest otherwise. However, a designer would not be indifferent to the *numerical value* of an attribute; he or she still has knowledge of the numerical value of the function based on the training points and his or her continuity assumption. Instead, I hypothesize that the designer would be indifferent of the *trend*; it is possible that the trend continues, but it is also possible that the trend reverses. In the absence of information, the designer does not know whether the function is increasing or decreasing. Therefore, in extrapolation, the *slope of the expectation* should tend towards zero and the uncertainty in the function value should grow as the designs become further from the designer's knowledge. This is represented by the notional plot in Figure 14. The designer's knowledge and confidence interval is represented by the shaded region and the dashed lines represent possible function shapes that the designer would consider plausible. The designer believes that the function could continue to increase, but the designer also believes it possible that the function could reverse – or any combination in between.

3.3.2.1 Extrapolation Behavior of Gaussian Process Models

Ideally, we would like to be able to retain the previously described capabilities of Gaussian process models while replicating the belief structure shown in Figure 14. Unfortunately, the

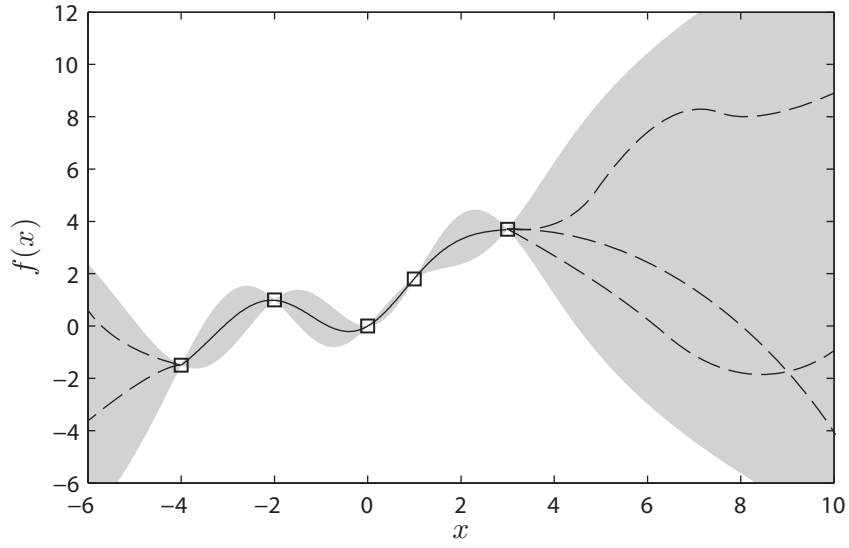


Figure 14: Designer's beliefs in extrapolation

behavior of most Gaussian process models departs radically from this concept. As stated earlier, the shape of the Gaussian process model is determined entirely by the training data, mean function, and covariance function. In many cases, the covariance function primarily affects the non-linearity of the Gaussian process; the behavior in extrapolation is more heavily influenced by the mean function. Figure 15 displays several Gaussian process models, each utilizing identical training information and the squared-exponential utility function but calculated with a different mean function.

Figure 15a shows a Gaussian process model with an assumed mean of zero. In the optimization community, this scenario is known as *simple kriging*. Notice the behavior of the function in the absence of training data; the mean rapidly returns to zero, and the variance asymptotically reaches a maximum value. The slope of the expectation does return to zero, but encountered a rapid trend reversal beforehand. From the perspective of designer beliefs, the designer has ruled out the possibility that the trend continues to increase, as shown by the dashed line. This belief would be especially problematic if the simulated designer were looking for the high value of y;. The designer would expect the highest value to be around $x = 3$ and would not look at higher values of x. In contrast, I believe many people would consider it reasonable that the highest value of y lies at high

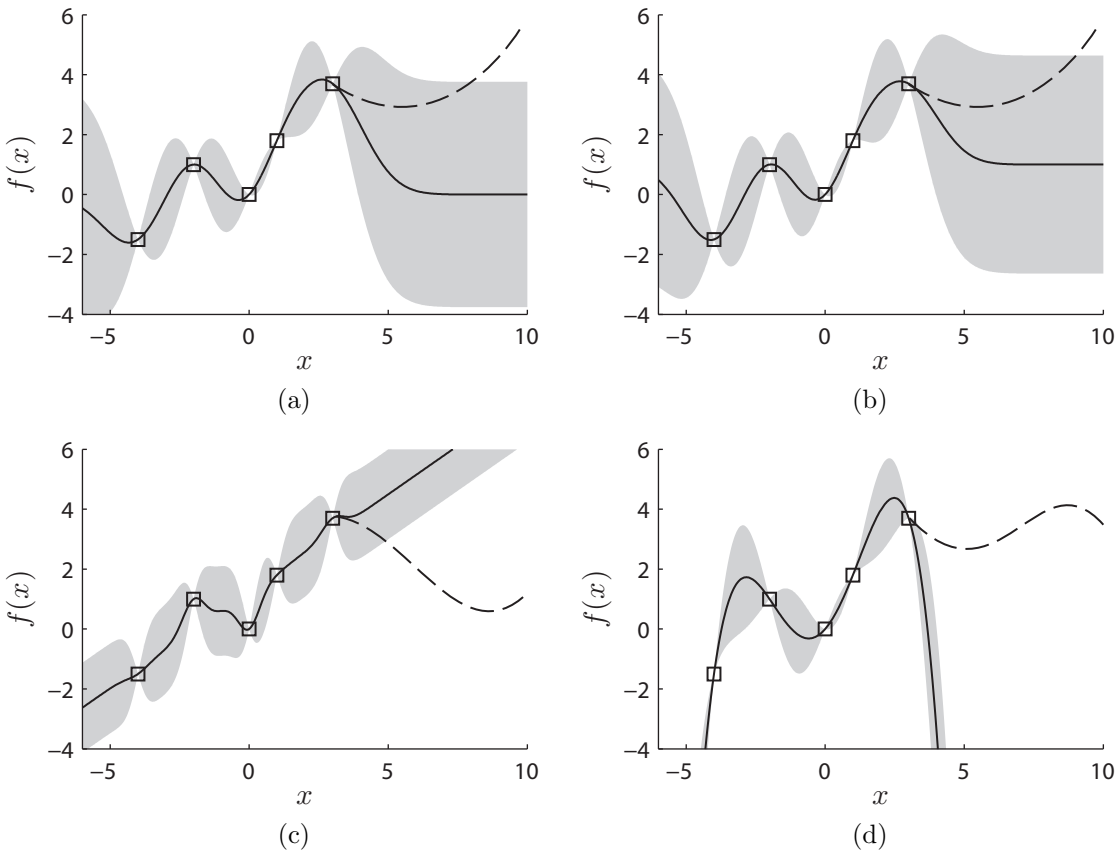


Figure 15: Effect of mean function on extrapolation

values of x.

One strategy for mitigating these problems is to consider alternative mean functions. Figure 15b displays a Gaussian process model where the mean is assumed to be stationary and is calculated with a maximum likelihood estimator. Unfortunately, this example suffers from similar problems as the example if 15a. The designer has ruled out the possibility of an increasing function, shown by the dashed line.

In *universal kriging*, the mean function is assumed to be a polynomial. Two examples are shown in Figure 15c and 15d of a linear and quartic mean, respectively. In the absence of training data, the Gaussian process follows the mean function, and the covariance tends towards its maximum value. While these assumptions have fixed some of the problems seen with simple and ordinary kriging, the have introduced additional inaccuracies. In the linear example, the designer assumes that the trend continues towards infinity, ignoring

the possibility shown by the dashed line. The quartic polynomial is perhaps the worst fit, assuming that the trend sharply decreases towards infinity.

Additional mean and covariance formulations were explored, each with similar problems. The resulting conclusion is that no combination of mean and covariance function currently exists that matches all the desired characteristics described in this section. Because of this, an effort was undertaken to derive a suitable combination with the desired properties. Due to the underlying mathematical depth of the covariance function, an exhaustive derivation is located in Appendix A.

To meet the desired characteristics outlined above, the following covariance function was derived:

$$\operatorname{cov}(y_i, y_j) = \frac{\sigma^2}{2}\left[\sqrt{\pi}\Big(2\alpha^m + \delta(0,\boldsymbol{x}) + \delta(0,\boldsymbol{x}^*) + \delta(\boldsymbol{x},\boldsymbol{x}^*)\operatorname{erf}\big(\delta(\boldsymbol{x},\boldsymbol{x}^*)\big)\Big) - \exp\big(-\delta(\boldsymbol{x},\boldsymbol{x}^*)^2\big) - 1\right] \quad (7)$$

The Gaussian process model is then calculated in the limit as α tends toward infinity. Therefore, Equation 3 is modified as follows:

$$\boldsymbol{f}|X, X_{tr}, \boldsymbol{f_{tr}} \sim \mathcal{N}\Big(\lim_{\alpha\to\infty} K_{X,X_{tr}} K_{X_{tr}} \boldsymbol{f},\ \lim_{\alpha\to\infty} K_X - K_{X,X_{tr}} K_{X_{tr}} K_{X_{tr},X}\Big) \quad (8)$$

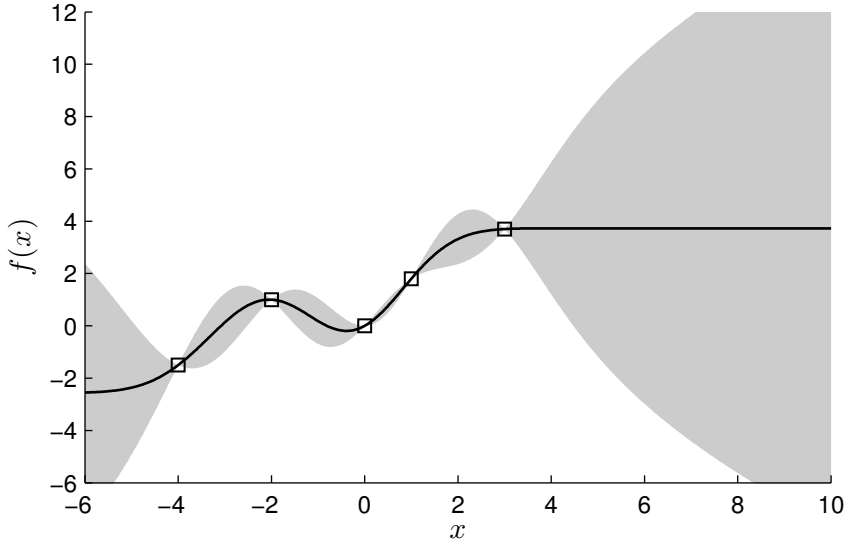


Figure 16: Custom covariance function designed for extrapolation

Figure 16 shows the derived covariance function in Equation 7 for the same training points in Figure 13. Notice that the resulting Gaussian process model has all the desired characteristics in extrapolation: the mean of the expectation tends towards zero, the uncertainty continues to grow as the distance from the training points increases, and the designer has not ruled out the possibility of the function continuing to increase or reversing to decrease. Additionally, this covariance function can be used in any number of dimensions and with any number of training points. This covariance function represents a novel contribution of this thesis.

3.3.3 Transforming Beliefs into Gaussian Processes

This section outlines several methodologies and capabilities for modeling potentially disparate types of knowledge about a design. The goal of this section is to illustrate methods for building Gaussian process models under a variety of situations. Some of these methods are further illustrated in the example problems in Chapter 4.

3.3.3.1 Past Experience: Point Designs

Known designs are the simplest information that can be encoded into a Gaussian process model. If the designer knows the attributes of a design point with certainty, this knowledge becomes a training point in the Gaussian process model. This data could represent the designer's past experience, a model that the designer trusts, or previous design information that the decision maker can access.

However, the designer does not necessarily need to know the function value at a point with certainty. In Figure 17, the training points in blue have noisy information, while the other training points are known with certainty. Notice that the uncertainty at the noisy locations does not collapse to zero. Noise in the data has two possible interpretations: either the designer is unsure of the accuracy of the information or he/she doubts the reproducibility of the information.

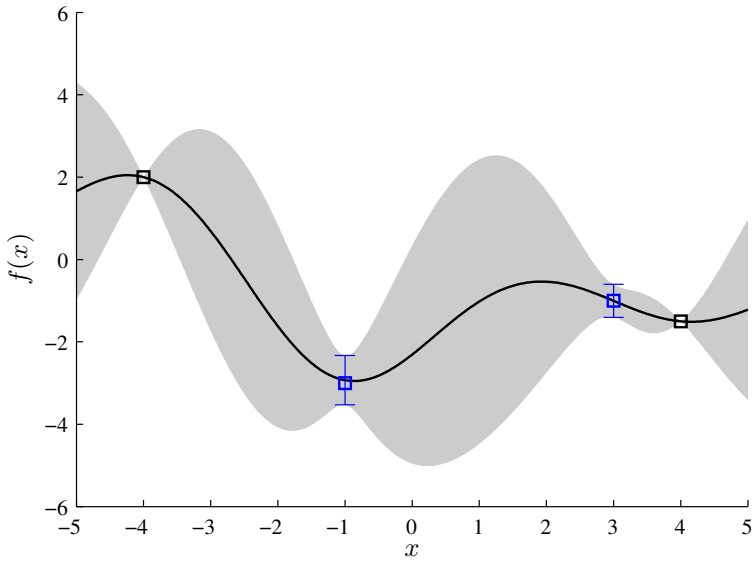


Figure 17: Gaussian process with noisy information

3.3.3.2 Trends: Slope Information

Since a linear transformation of a normal distribution is also a normal distribution, the derivative of a Gaussian process model is also a Gaussian process model. Solak et al. [67] derives a method for encoding derivative information into a Gaussian process model. Training points of slope information are treated exactly like a training point containing function values. The covariance between a training point with slope information and a regular training point is given by the following equation:

$$\operatorname{cov}\left(\frac{\partial f}{\partial x_i}, f\right) = \frac{\partial}{\partial x_i} k(x_i, x_j) \tag{9}$$

Similarly, the covariance between two training points containing slope information is given by the following equation:

$$\operatorname{cov}\left(\frac{\partial f}{\partial x_i}, \frac{\partial f}{\partial x_j}\right) = \frac{\partial^2}{\partial x_i \partial x_j} k(x_i, x_j) \tag{10}$$

Figure 18 shows a Gaussian process model with both function value information and slope information. The training points in blue have both sets of information, while the training point shown in black has only function value information. Notice that, in the presence of slope information, the uncertainty around the training point is much smaller

than when slope information is not present. Like the training points with function value information, the slope training points can contain noise. Therefore, if the designer has an approximate idea of what the slope is, that information can be encoded into the Gaussian process model. In this example, all the training points with slope information also have function value information, but this is not a general requirement; one could generate a Gaussian process model with only slope information and no function value information.

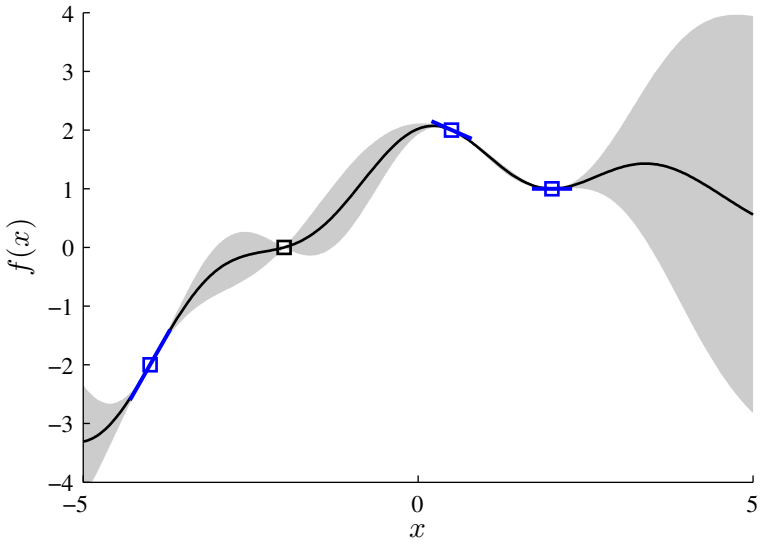


Figure 18: Gaussian process with slope information

Additionally, Riihimäki and Vehtari have developed a method for encoding monotonicity information into Gaussian process models [57]. To do this, they use training points with "virtual" slope information. At these points, the probability of a negative slope is either zero or very small (if the function is assumed to be monotonically increasing). By using enough virtual points, one can demand this behavior everywhere. This capability has many practical applications, since many engineering functions are monotonic over their useful range.

3.3.3.3 Analytic and Computer Models

In addition to having knowledge at specific design points, a designer might also have information over a continuous range of design variables. This data may be certain or uncertain.

Figure 19 represents an example of a low-fidelity design model. The expectation is the value that the model returns, while the error represents the designer's uniform uncertainty in the model. This model was created with numerous "virtual" training points with noise.

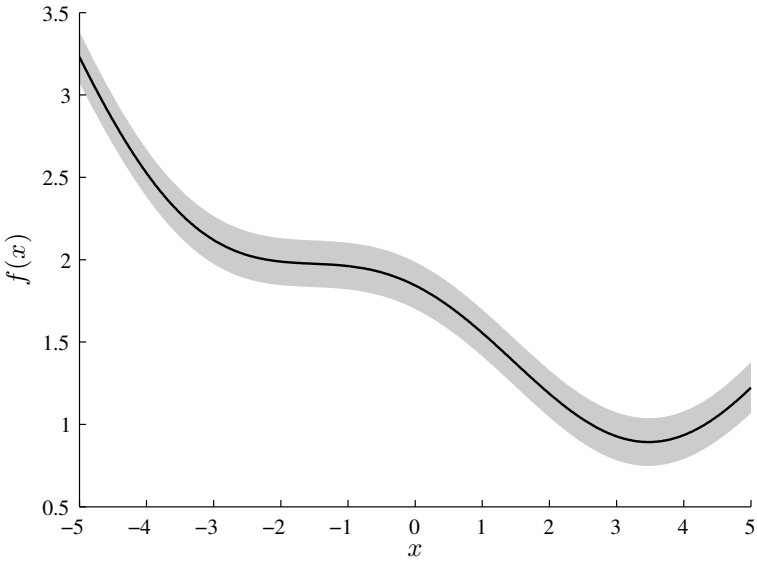


Figure 19: Low fidelity model represented as a Gaussian process

3.3.3.4 Combinations of Information

Often a designer has multiple sources of information; he or she uses knowledge and past experience in conjunction with computational models and empirical data. Perhaps the most useful feature of Gaussian process models is its ability to handle multiple types of information and make sense of it. In Figure 20a, the designer has three different types of information: a low-fidelity model with a range from -5 to 2.5, a known design point at $x = -2$, and a rough idea of the slope at two locations after the low-fidelity model ends. The low fidelity model is shown in gray, the known point in blue, and the slope information in red. Figure 20b shows a single Gaussian process model that was created with this suite of information. The known data point is able to correct the error in the low-fidelity model. The slope information guides the designer's beliefs in the absence of the low-fidelity model. Notice that the uncertainty begins to steadily grow after the designer can no longer use the low-fidelity model.

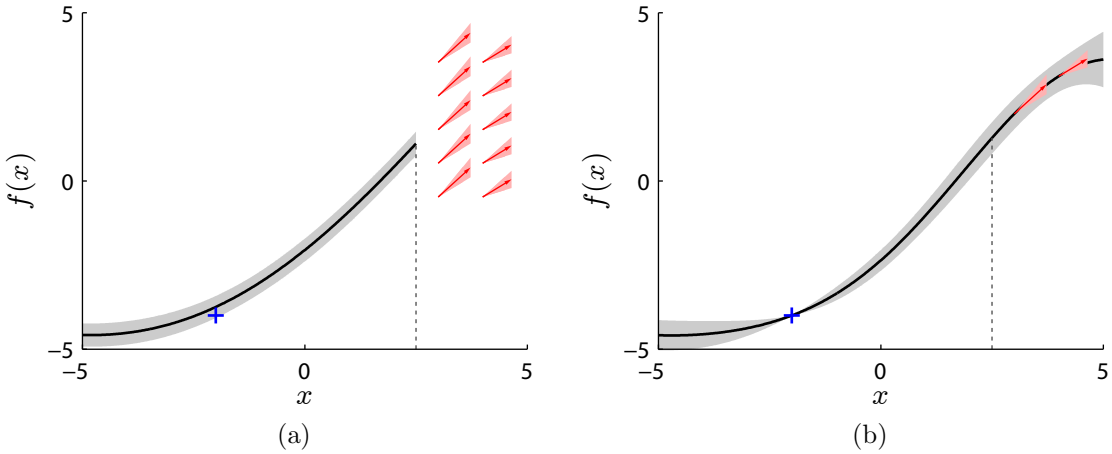


Figure 20: Different information combined into a single Gaussian process model

3.3.4 Alternatives Knowledge Models

In some cases, the use of the covariance function derived in Section 3.3.2 would not be appropriate for a particular simulation. For example, if the designer does not believe the underlying function to be continuous and smooth, then Equation 7 would be a poor choice for a covariance function. Fortunately, use of the framework does not rely on any particular covariance function; the modeler can choose a function which best matches the designer's beliefs. Since the designer has a unique Gaussian process model for each attribute, the modeler can choose different covariance functions for each attribute.

For certain classes of problems, the use of a regression technique itself might not be appropriate. This is especially true if the attribute in question is discrete. Fortunately, Gaussian process models can be reformulated for classification problems. Rasmussen and Williams [56] provide a derivation of this formulation. For brevity, it is not included here.

In general, the framework is not limited to the use of Gaussian process models. Any machine learning process that meets the requirements outlined at the beginning of this section would be a sufficient knowledge model. In its simplest form, a knowledge model is not required and the designer's beliefs can be represented by static probability distributions. This strategy is employed in the second example problem in Chapter 6.

3.4 Preference Model

The final enabling component of the framework is a method of encoding the preferences of the designer. Since the designer may not always know the exact consequences of his or her decisions, we require a method of eliciting preferences in the presence of uncertainty. We also require a method that allows the designer to negotiate trade-offs between preferences, since the designer is usually concerned with more than one objective.

This framework utilizes multi-attribute utility theory to capture the designer's preferences. The remainder of this section will address the methodology for developing a utility function. The first subsection outlines some fundamentals of utility theory: utility functions, risk, and multi-attribute utility theory. Specific attention is paid to the underlying assumptions and their validity in our context. The next section documents the derivation of a requirements-driven utility function: defining the underlying assumptions, forming a single attribute utility function, and combining multiple functions into a multi-attribute utility function.

3.4.1 Utility Theory Background and Assumptions

Utility theory is based on von Neumann and Morgenstern's four axioms regarding preferences under uncertainty [71]: *completeness*, *transitivity*, *continuity*, and *independence*. For brevity the mathematical definition of these axioms is not presented here; a thorough discussion of the axioms of utility theory can be found in Reference [39]. If a decision-maker's preferences conform to the these axioms, then he or she is said to be *von Neumann-Morgenstern rational*; it should be noted that this definition of rationality is different from definitions used in other contexts. Rationality is not concerned with what one specifically prefers, but rather the consistency of their preferences. Preferring cake to ice cream is not irrational; however, preferring cake to ice cream, ice cream to cookies, and cookies to cake is irrational.

If one's preferences align with the four axioms, the following theorem can be proved:

> *There exists a utility function, u, that assigns a real number to each outcome, such that*
>
> $$\mathrm{A} \succ \mathrm{B} \;\text{ iff }\; \mathrm{E}[u(\mathrm{A})] > \mathrm{E}[u(\mathrm{B})] \tag{11}$$

To put Equation 11 in words, the alternative A is strictly preferred to the alternative B if and only if the expected utility of A is greater than the expected utility of B. This utility function is unique up to a positive linear transformation.

A simple way to understand utility theory is through the use of lotteries. Figure 21 gives an example of two lotteries. In this document, a lottery will be denoted as a circle with the name of the lottery inscribed in the circle. Each branch of a lottery contains an outcome and the probability of that outcome. The square in Figure 21 represents a decision; in this case, the decision maker must choose lottery A or B. Equation 11 states that, if the designer conforms to the axioms on utility theory, then every lottery outcome has a definable expected utility. If the decision maker were indifferent between the two lotteries, then each lottery would have the same expected utility.

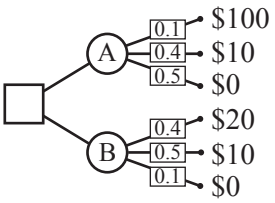


Figure 21: A decision between two lotteries

How might one determine which lottery has the highest expected utility? This choice depends entirely on the preferences of the designer. The axioms of utility theory tell us that we can replace any complicated lottery with an equivalent lottery containing only the best and worst alternatives. The simple lottery in Figure 22 can assist us with the more complicated decision in Figure 21. For this simple lottery, the decision maker is given a choice between lottery C and an outcome with absolute certainty, lottery D. Notice that we have used the best and worst outcomes from the lotteries in Figure 21 to populate lottery C. This decision enables us to determine the decision maker's utility function.

Since utility functions are only unique up to linear transformations, we can arbitrarily define the utility at two locations. Let us suppose that $u(\$0) = 0$ and $u(\$100) = 1$. We can then set the value of y to any number between 0 and 100. The pertinent question for the designer is, "What value of probability, p, would make you indifferent between lotteries

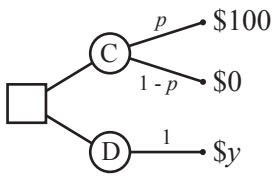


Figure 22: Eliciting a utility function for money

C and D?" Because we have rescaled our utility function between 0 and 1, whatever value of p the designer selects is equal to the utility of y. In this manner, we can determine the utility of \$10, \$12, and \$20, then calculate the expected utility of each of the lotteries in Figure 21. The lottery with the highest expected utility is the preferred lottery.

In Chapter 1, we identified this framework as a descriptive model of human decision-making. Since utility theory is typically considered a normative tool in decision making, is utility theory a valid choice for this model? Perhaps a more fundamental question is, to what extent do designer's preferences conform to the the fundamental axioms of utility theory? Empirical evidence shows large holes in this assumption; interested readers should consult the Allais paradox [36] and the Ellsberg paradox [19]. Fundamentally, the use of utility theory in a descriptive model is incorrect; humans do not tend to have consistent preferences that conform to the four axioms. However, utility theory can still be used as an approximation to human behavior; while the framework cannot exactly model decisions humans may make, in many cases it can provide similar decisions. At the same time, utility theory enables us to capture a variety of preference structures based on multiple incentives. If our ultimate goal is to understand how these incentives and their corresponding decisions affect the dynamics of a larger organization, this approximation may be sufficient. Regardless, it should remain clear that this is a fundamental limitation of the framework.

3.4.1.1 Quantifying Risk

The process outlined in the previous subsection for determining utility may seem rather trivial and unnecessary; one could just calculate the expected *value* of each lottery and choose the option with the highest expected value. In practice, however, people often do not reason in this manner. Consider again the set of lotteries shown in Figure 21. Both

lotteries have the same expected value of \$14. If we chose the lottery solely on expected value, the decision maker should be indifferent between the two lotteries. However, most people would select lottery B; lottery B has a 90% chance of making money, while lottery A only has a 50% chance. Utility theory does not claim that the choice of lottery B is irrational; it simply implies that utility is not linear with money. Consequently,

$$\mathrm{E}[u(y)] \neq u(\mathrm{E}[y]) \tag{12}$$

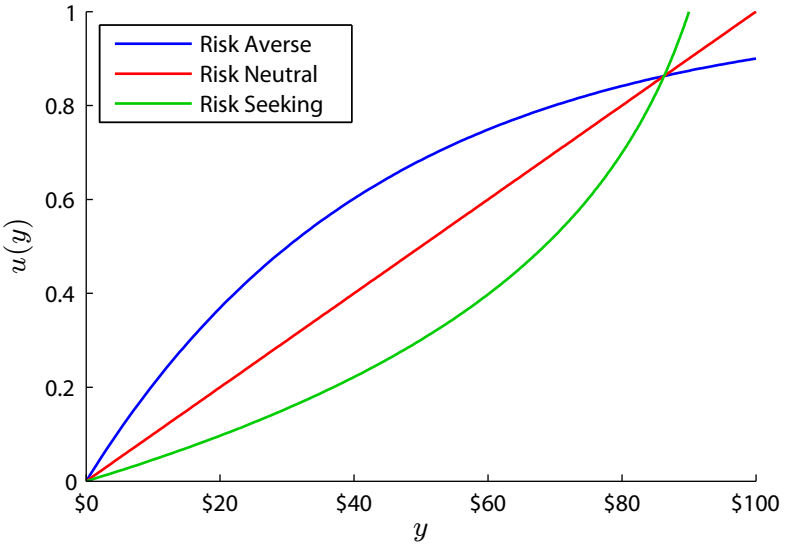


Figure 23: Risk behavior

Figure 23 shows an example of three different utility functions, each with different risk behavior. Risk averse behavior implies a concave utility function, while risk seeking implies a convex utility function. We can quantify this behavior using the Arrow-Pratt measure of absolute risk aversion [5, 53].

$$r(y) = -\frac{u''(y)}{u'(y)} \tag{13}$$

Positive quantities of r imply that the decision maker is risk averse, while negative quantities imply risk-seeking behavior. From Equation 13 and Figure 23 it is clear that curvature in a utility function determines the risk behavior and has important implications for decision making. The slope in the denominator of Equation 13 serves to normalize the risk value, since utility functions are invariant to linear transformations.

It is important to note that the risk behavior outlined in this section was originally derived for money; care must be taken when applying these principles to situations of risk that involve quantities that cannot be expressed monetarily. However, this theory is still valid for *monotonically increasing* utility functions [37]. For a discussion of risk with respect to deceasing or non-monotonic utility functions, see Reference [37].

3.4.1.2 Multi-Attribute Utility Theory

Suppose we are in a situation in which we care about more than one attribute, as is often the case with designers. Eliciting preferences consistently for such a complicated problem may be difficult. However, suppose we could identify a utility function for each attribute individually, assuming the other attributes remained unchanged. Is it possible to combine these utility function into an aggregate utility function that still reflects the designer's preferences?

Multi-attribute utility theory deals with the problem of multiple objectives. Since multi-attribute utility theory is an extensive subject area, this thesis will only cover a few select topics that are pertinent to our discussion. See Keeney and Raiffa's comprehensive book for more information on this subject [37]. For our purposes, creating this combined utility function is much simpler if we can demonstrate two properties of the designer's preferences: *mutual preference independence* and *mutual utility independence*.

Preference independence implies that the decision maker's preference for one outcome over another is consistent regardless of the other attributes. For example, I tend to prefer \$100 to \$10. This preference would be consistent on any day of the week. Therefore, my preference for money is preferentially independent of the day of the week.

While preferential independence is fairly easy to demonstrate, utility independence can be far more nuanced. Utility independence states that the utility of an outcome is the same regardless of the outcome of another attribute (up to a linear transformation). Consider the two decisions outlined in Figure 24; suppose that our risk averse decision maker is indifferent between lotteries A and B. Said another way, the utility of \$40 on Monday is equal to 0.6. Now consider lotteries C and D. If the probability, p, that would make the

decision maker indifferent between these alternatives is still 0.6, then the decision maker's preferences for money are utility independent of the day of the week. Why might p not equal 0.6? Suppose a decision maker must pay rent on Tuesday and finds that he is $100 short of making the payment. On Monday, the utility of $40 might have been quite high. On Tuesday, however, the utility of $40 may be quite low, since the designer might take a riskier option that gives him a chance a making the rent payment. Utility independence implies preferential independence, but preferential independence does not necessarily imply utility independence.

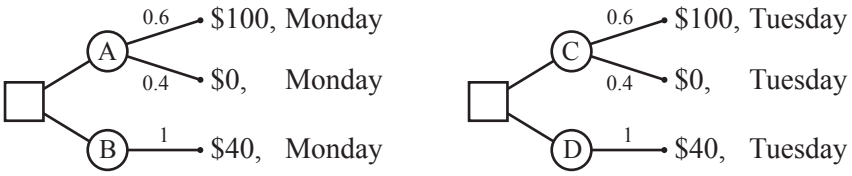


Figure 24: Demonstrating utility independence

If two attributes are mutually utility independent, then the risk behavior, r, of each individual utility function remains unchanged when combined into a multi-attribute utility function. Equation 13 implies that the most general form of a multi-attribute utility function with n mutually utility independent attributes is the multilinear form:

$$u_{\text{multilinear}} = \sum_{i=1}^{n} k_i u_i + \sum_{i=1}^{n-1} \sum_{j>i}^{n} k_{ij} u_i u_j + \sum_{i=1}^{n-2} \sum_{j>i}^{n-1} \sum_{l>j}^{n} k_{ijl} u_i u_j u_l + \ ... \ + k_{123...n} u_1 u_2 u_3 ... u_n \quad (14)$$

Therefore, if we can prove mutual utility independence for a set of attributes, then we know that the multi-attribute utility function must be of the form represented in Equation 14. The only remaining challenge is to determine the value of the unknown constants, k.

3.4.2 Formulating a Designer Utility Function

The following section documents the derivation of a utility function for a designer faced with a multivariate and multi-criteria decision in a requirements-driven organization. Implicit in this formulation are the following assumptions:

- The designer is only given inequality requirements
- The designer is influenced only by his/her incentive to meet the requirements

- The designer's preferences remain constant over time
- The designer does not anticipate that the requirements will change
- The designer does not believe that meeting the requirements is impossible

We assume the designer is a given a set of inequality requirements, $\boldsymbol{f}^*$, for a set of attributes, $\boldsymbol{f}$. We define a unitless variable, $\boldsymbol{g}$, such that

$$g = \begin{cases} \frac{f}{f^*} - 1 & \text{if } f \text{ is required to be greater than } f^*, \\ 1 - \frac{f}{f^*} & \text{if } f \text{ is required to be less than } f^* \end{cases} \tag{15}$$

Since $\boldsymbol{f}$ is a function of $\boldsymbol{x}$, $\boldsymbol{g}$ will also be a function of $\boldsymbol{x}$. g is a measure of how well a requirement is met; a requirement is met if $g > 0$ and unmet if $g < 0$. When used in lotteries, g will be shown as a percent.

For the remainder of this section, we will first investigate the form of a utility function for one requirement only as a function of g. We will then demonstrate that the mutual preference and utility independence properties from Section 3.4.1.2 appear to be reasonable assumptions for these decisions. Finally, we will illustrate how to combine individual utility functions into a unified multi-attribute utility function.

3.4.2.1 Single Requirement Utility Function

Given the assumptions above, we can postulate the form of a utility function. This utility function is for a single-requirement and is dependent *only* on how well this single requirement is met. We are conceptualizing how u_i changes with g_i as if g_i were completely independent of all $g_{j \neq i}$. In reality, all values of $\boldsymbol{g}$ are related by the alternatives, $\boldsymbol{x}$, they represent. This dependence will be captured in the aggregate multi-criteria utility function, not in the individual utility functions.

The form of the utility function used in the remainder of this document was constructed by first postulating the following list of general attributes of a designer's decision-making preferences given the assumptions listed at the beginning of this section.

1. A designer will always prefer a higher value of g.

   Justification: g is a measure of how well a designer meets the requirements. For $g < 0$ this assumption is obvious, since the designer would always prefer

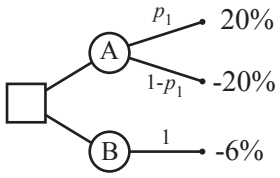
being closer to meeting a requirement. For $g > 0$, we suspect this to also be the case. A designer often prefers to have margin on requirements. Exceeding a requirement is often increasingly beneficial to the organization; for example, an organization would always prefer to come in under cost. One might disagree with this assumption on the basis that exceeding requirements usually implies additional unwanted costs. Recall, however, that we are examining this requirement in isolation, assuming all other requirements remain constant. If a designer can add margin to a design without incurring additional costs, it is likely that he/she would prefer to do so.

2. As $|g|$ becomes large, the designer becomes increasingly closer to being indifferent between alternatives.

   Justification: Consider the difference between a design where $g = -5\%$ and a design where $g = 0\%$. A designer would strongly prefer the latter, since it meets the requirements. The designer would be more indifferent between two designs with $g = 50\%$ and $g = 55\%$. Even though the designs in the two scenarios have the same Δg, both designs in the second scenario meet the requirements; therefore, the designer perceives these designs as being more equivalent than the other two.

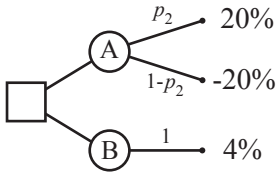
3. When g is near zero, the change in utility with g is higher than at values of g above zero.

   Justification: Suppose the best theoretically-possible design could exceed the requirement by 20% and the worst design would miss the requirement by 20%. One might imagine that this is a cost requirement, and the percentages represent how far under cost the project is. Consider the following decision:

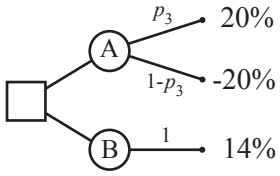


   For most people and requirements, the value of p_1 in this lottery is likely closer to 0.5 than it is to 1 or 0. The expected value of the lottery is 0%. If the

designer were risk neutral the value of p_1 would be 35%. However, the designer would likely be quite worried of coming in 20% over budget, so they might prefer the certain option which would make the value of p_1 higher. Now consider the following decision:



In this example, the value of p_2 is likely closer to 1 than it is to 0.5. The designer is guaranteed to meet the budget requirement, so the designer would require a fairly high probability of being under budget in order to take the risk. Now consider a third decision:



Once again, the value of p_3 is likely very close to 1. Since, the difference between p_3 and p_2 is smaller than the difference in p_2 and p_1 for the same change in g, the designer is becoming increasingly closer to indifference between the alternatives. A similar argument can be made for when g approaches -20%.

4. When g is near zero, the change in utility with g is higher than at values of g far below zero.

   Justification: See the justification for (3).

If we accept the premises above, this implies the following properties of the utility function:

1. u monotonically increases with g
2. $\lim_{g \to \pm\infty} \frac{du}{dg} = 0$
3. There exists a quantity g^* such that $g^* \leq 0$ and $\frac{du}{dg}\big|_{g=g^*} > \frac{du}{dg}\big|_{g>g^*}$
4. There exists a quantity g^* such that $g^* \leq 0$ and $\frac{du}{dg}\big|_{g=g^*} < \frac{du}{dg}\big|_{g<g^*}$

From the properties listed above, we can see that the designer's utility function is restricted to the class of sigmoid-like curves.[1] Specifically, I propose the following function to approximate the designer's preferences for meeting a requirement, shown in Figure 25 and represented by the following equation:

$$u = \left(\exp\left(\frac{g^* - g}{b} \right) + 1 \right)^{-1} \tag{16}$$

where g^* and b are designer-specific parameters described in detail below. This function satisfies all the postulates outlined above and has a convenient analytical form which assists in simplifying certain computations. It is likely that the shape that best represents a designer's preferences is more intricate and nuanced than the simple function shown here. However, for the examples and demonstrations performed during the research in this thesis, it was found that Equation 16 was sufficient in demonstrating plausible and reasonable designer behavior. See Section 3.4.3 for further discussion regarding the shape of a designer's utility function.

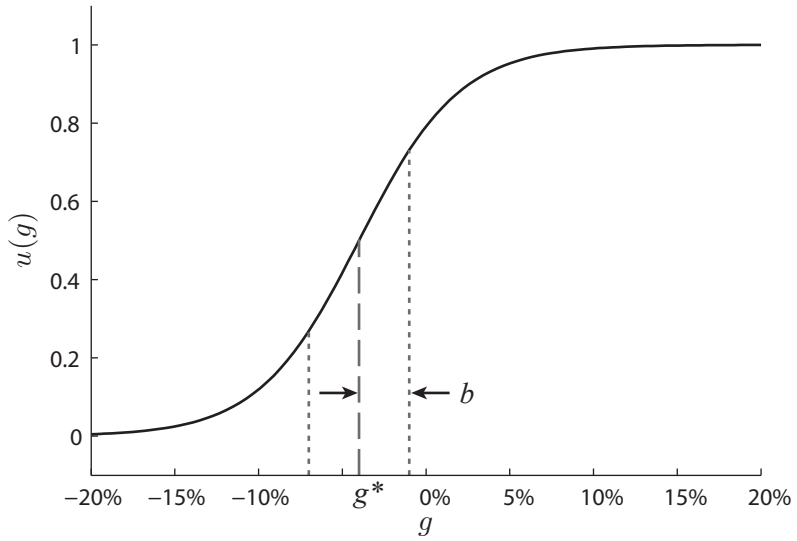


Figure 25: Requirements based utility function

The utility function in Equation 16 contains two parameters, g^* and b. These parameters control the shape of the designer's utility function and, therefore, can affect the designer's

[1] A true sigmoid function is assumed to be differentiable, which may not strictly be true for the designer's utility function. Hence, the phrase sigmoid-like is used to imply the general shape of the curve.

behavior. Mathematically, g^* is the inflection point in the curve and the value of g at which $u = 0.5$. The parameter b represents the distance from g^* to 73.1% of maximum utility.

From a behavioral perspective, the reader can imagine that g^* represents the value of g at which the design is close enough to the requirements that the design is "good enough." When combined with other requirements, g^* is usually the value of g at which the designer focuses his attention on other requirements that are performing poorly. In conjunction with g^*, b represents how tolerant the designer is to missing the requirement by small amounts. I will adopt the terms *strict* and *lax* to describe the shape of a particular utility function. For lax requirements, b is "large" and g^* is negative. For strict requirements, the value of b is very small and g^* is very near zero. In the limit as $b \to 0$ and $g^* \to 0$, the utility function becomes the Heaviside function shown in Figure 26.

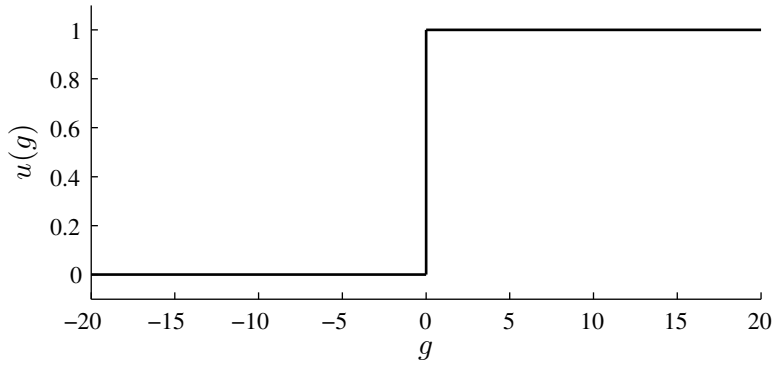


Figure 26: A very strict requirement

A strict requirement implies that the designer receives no reward for *that specific requirement* if the requirement is missed even slightly (the designer may still receive a reward for the other requirements). For a lax requirement, a designer will still receive a partial reward for coming close to the requirement. Strict and lax depend largely on the designer's motivation and perception of what management finds acceptable. For example, if an engine manufacturer has a contract with an airframe manufacture specifying a particular value for thrust-specific-fuel-consumption (TSFC) at a particular operating condition, the engineers employed by the engine manufacture will probably view the TSFC requirement as strict; missing the requirement can have severe consequences for their company. At the same time,

an engineer working on a specific part may view a weight requirement as lax; if the designer is slightly overweight, it might not severely affect his reward since the weight of his part will not contribute significantly to the overall weight of the engine. Note that strict and lax are not binary designations but represent a continuum of b values.

Using Equation 13, we can analyze the risk behavior of the designer which has been plotted in Figure 27. The designer is risk averse for values of $g > g^*$ and risk seeking for $g < g^*$. At $g = g^*$, the designer is risk neutral. This risk behavior for designers in the context of requirements has been observed by other researchers [16]. The maximum value of risk aversion is equal to the reciprocal of b.

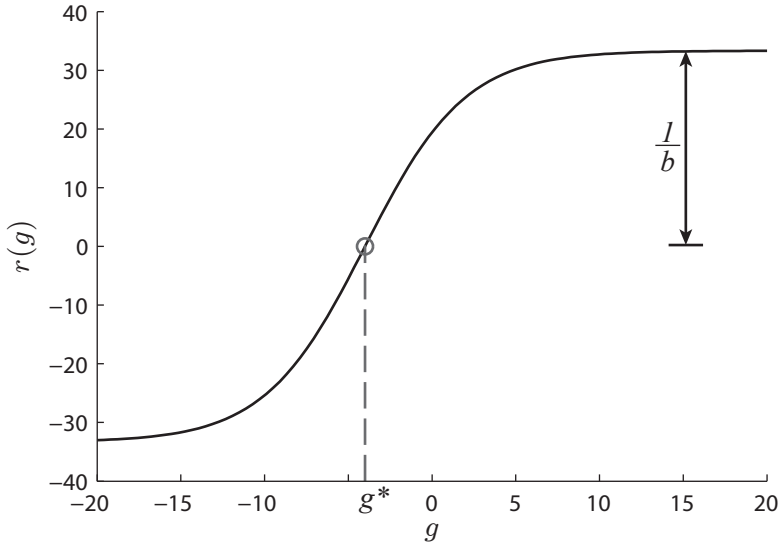


Figure 27: Risk behavior of utility function

3.4.2.2 Combining Utility

Designers are often concerned with more than one requirement. Given the requirement-based utility function derived above, the goal of this subsection is to determine: (1) under what conditions can the single-requirement utility function be aggregated into a multi-attribute utility function and (2) the form of a requirements-driven multi-attribute utility function.

For this discussion, we will make the distinction between *hard requirements* and *soft requirements*. A hard requirement is critical to the design; failing to meet a hard requirement would result in a non-functional artifact. A factor of safety requirement on a key structural member could represent a hard requirement. Soft requirements, while highly desired, may not critically impact the functionality of the design. Aircraft range might be considered a soft requirement, since failing to meet a range goal would not impair the aircraft's basic ability to fly. This distinction is similar to concepts of hard and soft goals in goal programming [47]. A more formal definition of soft and hard requirements will be derived in this section. Like the parameters in the utility function, whether a requirement is hard or soft is a subjective question; it depends entirely on the beliefs of the designer. Note that a requirement being *strict* or *lax* has no bearing on whether it is considered *hard* or *soft*. Strictness relates to the shape of the individual requirement's utility function, while hard and soft embody the requirements relationship to other requirements.

Preferential independence of requirements-based utility functions is an easy assumption to justify; a designer will always prefer higher values of g_i regardless of the value of g_j. Keep in mind that we are assuming that the value of g_j remains constant; any decision that might affect the value of g_i has no impact on g_j.

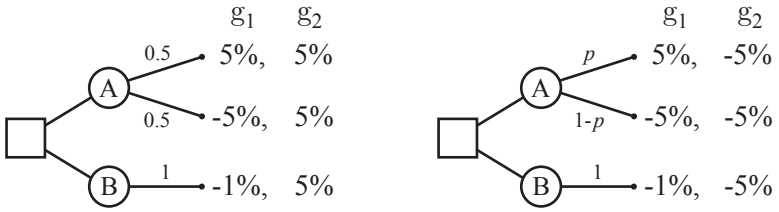


Figure 28: Utility independence of requirements

For utility independence, consider the decisions in Figure 28. Suppose that the designer is indifferent between lotteries A and B; when g_2 is 5% the designer is indifferent between a certain design that misses the requirement by 1%, and a design that has an equal probability of meeting or missing a requirement by 5%. If the requirements are mutually utility independent, then the value of p that would make the designer indifferent to lotteries C and D would remain 0.5. Recall that the choice of p has no bearing on the value of g_2; g_2 is fixed

for both lottery C and lottery D. In general, I believe the risk behavior of most engineers would not change due to independent changes in other attributes, especially if that decision has no impact on the other attributes.

If we accept the premise of mutual utility independence, the total utility function, u_{total}, is multilinear in its most general form. For reasons that will later become clear, we will now divide the requirements into soft and hard requirements. Without any loss of generality, we can group the soft requirement utility functions into an aggregate utility function, u_{soft}; the form of this function will be explored later. Suppose that the designer has n hard requirements and m soft requirements, the latter having been combined into u_{soft}. The multilinear form of u_{total} is

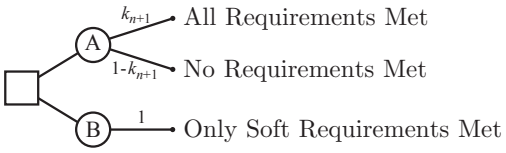
$$u_{total} = \sum_{i=1}^{n+1} k_i u_i + \sum_{i=1}^{n} \sum_{j>i}^{n+1} k_{ij} u_i u_j + \ ... \ + k_{123...n} u_1 u_2 u_3 ... u_n u_{n+1} \tag{17}$$

where $u_{n+1} = u_{soft}$. To simply the analysis, we will also make an approximation for the utility function derived earlier: we will consider any met requirement to have a utility of 1 and any unmet requirement to have a utility of 0 (this is equivalent to g^* and b being very close to zero).

Consider the scenario in which only the soft goals are met and none of the hard goals are met. Equation 17 simplifies to

$$u_{total} = k_{n+1} \tag{18}$$

We can analyze this same scenario through the use of lotteries, as shown below:



To determine the value of utility in Equation 18, we must postulate the probability,k_{n+1}, that would make the designer indifferent between the two lotteries. To give a more direct example, this is equivalent to asking "What is the utility of an airplane that can neither takeoff nor land, but can comfortably seat 300 passengers?" For the majority of decision makers, the value of k_{n+1} is approximately zero; most people would take any chance that

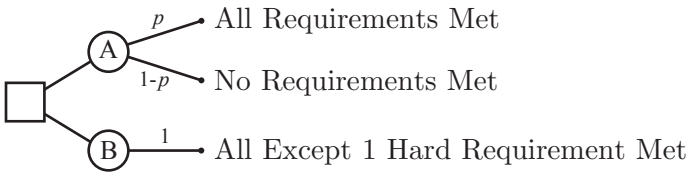
could yield a functional airplane over a design that would yield a useless airplane with certainty. Therefore, k_{n+1} must equal 0.

Now consider the scenario where only one hard requirement is met and all the remaining hard requirements are unmet. Once again, the utility function simplifies to

$$u_{total} = k_i \tag{19}$$

By similar logic, the utility of this design is very close to zero; an airplane that can takeoff but not land is equally as useful as the airplane that could neither takeoff nor land. Therefore, k_i must also equal 0.

We can extend this concept and consider the scenario where all requirements are met with the exception of a lone hard requirement.



Consider the example of an airplane that can theoretically takeoff, land, comfortably seat 300 passengers, but whose fuselage skin would fail under pressurization. As in the previous examples, I would contend that the utility of this design is also very small, and that we can assume that it is approximately zero. A designer would still take virtually any chance of a functioning airplane over an airplane that is known not to function. Making this an assumption further simplifies the analysis and leads to a more formal definition of a hard requirement: a requirement is considered "hard" if failing to meet said requirement results in a design of near-zero utility. Recall that the labeling of a requirement as hard or soft is up to the judgment of the designer being modeled. A designer may not consider any requirement to be a hard requirement, while another designer may consider all requirements to be hard requirements.

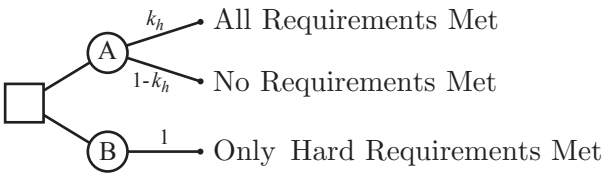
If we eliminate all the terms that do not include the utility of every hard requirement, we are left with the following equation

$$u_{total} = \left(k_{123\ldots n} + k_{123\ldots n+1} u_{\text{soft}}\right) \prod_{i=1}^{n} u_i \tag{20}$$

By constraining the range of the utility function between 0 and 1, we gain an additional requirement that $k_{123\ldots n}$ and $k_{123\ldots n+1}$ must sum to 1. Therefore, the final form of the total utility function is

$$u_{total} = \left[k_h + (1 - k_h)u_{\text{soft}}\right] \prod_{i=1}^{n} u_i \tag{21}$$

where k_h is the utility of meeting only the hard requirements and can be determined via the following lottery:



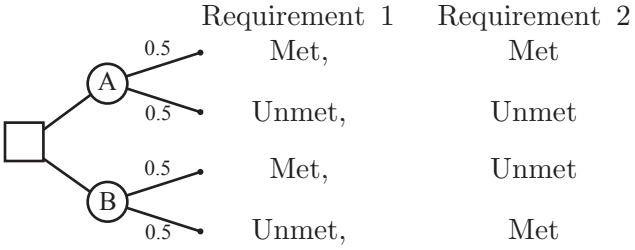
In practice, the value of k_h is likely very high.

While an equation incorporating the hard requirements has been derived, the functional form for u_{soft} has yet to be addressed. Unfortunately, without information specific to the design problem, little more can be said about the form of a soft requirement multi-attribute utility function. With soft requirements, the relation between them is more complex. There can exist complementary requirements in which the requirements have a net higher utility when both of them are met than the sum of their utilities when they are met individually. In its most general form, u_{soft} will be of a multilinear form since the requirements are mutually utility independent.

Two commonly used utility functions are the *additive* and *multiplicative* utility functions. Both are special cases of the multilinear utility function. The additive utility function is given by the following equation:

$$u(y) = \sum_{j} k_j u_j(y_j) \tag{22}$$

where k_j is the utility of meeting only that requirement. Note that the sum of all k_j must be equal to 1. The additive utility function has the most strict assumptions of all the utility function forms. If two utility functions are additive, it implies that the designer is indifferent to the following lotteries:



Obviously hard requirements cannot be additive since both outcomes in lottery B are useless to the designer. An additive utility function assumes that none of the requirements are complimentary; the reward the designer receives for meeting a particular requirement is the same regardless of whether the other requirements are met.

The multiplicative utility function is given by the following equation:

$$u(\boldsymbol{y}) = \frac{1}{k}\left[\prod_j \big(kk_j u_j(y_j) + 1\big) - 1\right] \tag{23}$$

where k is the value that satisfies the following relation:

$$k + 1 = \prod_j \big(kk_j + 1\big) \tag{24}$$

Each soft requirement has its own k_j which is approximately equal to the utility of only meeting that requirement. Once again, the particular form is dependent on the perceptions of how the designer views he will be rewarded; if neither of these equations capture the designer's preferences, then the modeler should refer back to the multilinear utility function and determine the values of the unknown constants through the use of lotteries.

3.4.3 Alternative Utility Function Forms

The utility function represented by Equation 16 was designed to be computationally simple yet flexible; it satisfies all the requirements outlined in Section 3.4.2.1 and can be adapted to different designers and requirements by varying the values of g^* and b. At the same time, this utility function may not accurately represent a designer's preferences in all situations. The utility function that best approximates a designer is likely more complex than the simple form given and may be highly dependent on the situation.

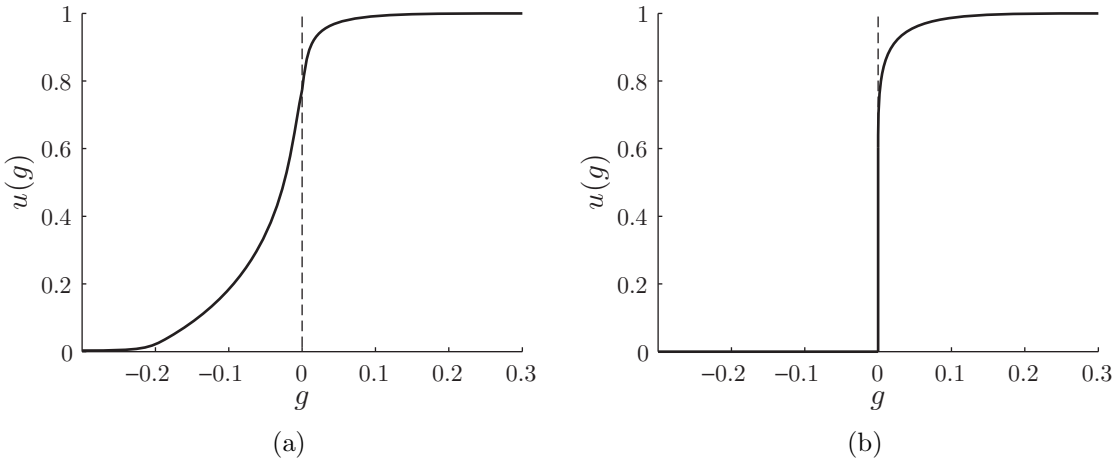


Figure 29: Notional utility functions for requirements

Figure 29 shows two notional examples of utility functions that might represent a particular designer's preferences. Unlike the utility function in Figure 25, Figure 29a is not rotationally symmetric; The slope is relatively gradual up to $g = 0$, but then rapidly asymptotes to 1 after meeting the requirement. This curvature implies that the risk seeking behavior is not symmetric; the designer is much more risk averse after meeting the requirement than he or she is risk seeking when not meeting the requirement.

Figure 29b shows a utility function that may be specific to certain classes of requirements. The utility function is equal to zero for $g < 0$, but then discontinuously jumps to a nominal value at $g = 0$. From there the designer exhibits risk averse behavior. This particular utility function could reflect a designer's preferences for a safety factor requirement. If the safety factor is below the requirement of 1, then the design has no utility. After meeting the requirement, the designer prefers high safety factor values, but approaches the problem from a risk averse perspective.

The modular structure of the framework gives the modeler immense flexibility when modeling a particular designer. As with the covariance function in the previous section, if the utility function demonstrated in this document does not accurately reflect the beliefs of the designer, then the modeler is free to substitute his or her own. As long as the utility function remains monotonically increasing, the equations derived for the multi-attribute utility function will still hold, regardless of the particular shape of the single attribute

utility function.

3.5 Integrating the Elements

In many real world applications, design is an under-determined problem; multiple designs exists that would satisfy the requirements. Rarely does the design process yield a single, obvious design point. In other cases, the design problem is over-determined, and no design exists that simultaneously satisfies all requirements. This presents a challenge when creating a decision algorithm. If the algorithm is given many feasible designs, it needs a logical method of choosing one of them. At the same time, if we restricted the designer's decision to only the feasible design space, the designer would not be able to make a decision when no feasible space existed.

Optimization can be viewed as a method of automating a decision. If the designer can express his preferences in the form of an objective function, he or she can run one of many optimization algorithms to determine the "best" alternative. Our approach has been to reformulate the designer's decision in the form of an optimization problem statement. The entire decision-making process is replaced with a single, albeit complex, objective function. Since the optimization problem is unconstrained, all alternatives have an objective function value; therefore, in the absence of feasible alternatives, the designer is still able to return a decision.

3.5.1 Expected Utility Maximization

Given a design alternative, the designer's beliefs about that alternative, and the designer's preferences, the expected utility of an alternative can be calculated using the following equation:

$$\mathrm{E}[u(\boldsymbol{x})] = \int_{-\infty}^{\infty} u(\boldsymbol{g}) p(\boldsymbol{g}|\boldsymbol{x}) \mathrm{d}\boldsymbol{g} \tag{25}$$

The designers decision, π, is the alternative with the maximum expected utility:

$$\boldsymbol{\pi} = \underset{\boldsymbol{x}}{\operatorname{argmax}} \left(\int_{-\infty}^{\infty} u(\boldsymbol{g}) p(\boldsymbol{g}|\boldsymbol{x}) \mathrm{d}\boldsymbol{g} \right) \tag{26}$$

Since the probabilities of each g are calculated in independent Gaussian process models, the probabilities of each attribute are independent. Therefore, the equation for expected

utility becomes:

$$\mathrm{E}[u(\boldsymbol{x})] = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\ldots\int_{-\infty}^{\infty} u(g_1, g_2, ..., g_n)p(g_1|\boldsymbol{x})p(g_2|\boldsymbol{x})...p(g_n|\boldsymbol{x})\mathrm{d}g_1\mathrm{d}g_2...\mathrm{d}g_n \qquad (27)$$

In reality, the designer may not perceive the probabilities in Equation 27 as independent, since certain attributes may be correlated. For example, consider a designer who is given a requirement for aircraft weight and range. The designer knows that if the weight of the aircraft is heavier than expected, then the range will probably be lower than expected. Gaussian process models can account for attribute correlations, and the interested reader is referred to the literature of *cokriging*.

Equation 27 brings together all three elements of the framework: the alternatives x and the relevant attributes g; the designer's knowledge of each alternative in the form of probabilities; and the designer's preferences in the form of a utility function.

Since all alternatives have an expected utility, we are able to see not only what the designer's best decision is but also how indifferent he is between the alternatives. If the expected utility of many alternatives is very close, then the designer would be largely indifferent between the alternatives. Additionally, we can examine the different components of the equation to deduce the designer's "reasoning" for choosing one alternative over another.

3.5.2 Alternate Forms

The functional form of Equation 27 is computationally troublesome, since no analytical solution exists for the derived utility function form. Furthermore, the expected utility calculation is a multi-dimensional integral with a dimension for each requirement; if a deterministic numerical approximation technique is used, then sampling a large multi-dimensional space can be computationally expensive since the sampling grid grows exponentially.

Fortunately, if Equation 21 is used to model the designer's preferences, then Equation 27 can be broken up as follows:

$$\mathrm{E}[u(\boldsymbol{x})] = \int_{-\infty}^{\infty}\big(k_h + (1-k_h)u_s(\boldsymbol{g}_s)\big)p(\boldsymbol{g}_s)d\boldsymbol{g}_s\prod_i\int_{-\infty}^{\infty}u(g_{h,i})p(g_{h,i})dg_{h,i} \qquad (28)$$

See Appendix B for a derivation. The first integral containing u_{soft} can also be further split into single integrals if multiple soft requirements exist. The particular equation, however,

depends on the function form of u_{soft}. If u_{soft} is of the multiplicative form of Equation 23, then the first integral is equivalent to:

$$\int_{-\infty}^{\infty} \big(k_h + (1-k_h)u_s(\boldsymbol{g}_s)\big)p(\boldsymbol{g}_s)d\boldsymbol{g}_s = \\ k_h - \frac{1-k_h}{k} + \frac{1-k_h}{k}\prod_j \int_{-\infty}^{\infty} (kk_j u(g_{s,j}) + 1)p(g_{s,j})dg_{s,j} \quad (29)$$

For additive form of Equation 22, the integral is equivalent to:

$$\int_{-\infty}^{\infty} \big(k_h + (1-k_h)u_s(\boldsymbol{g}_s)\big)p(\boldsymbol{g}_s)d\boldsymbol{g}_s = k_h + (1-k_h)\sum_j \Big(\int_{-\infty}^{\infty} k_j u(g_{s,j})d_g s, j \Big) \quad (30)$$

For large amounts of requirements, Equation 28 can drastically reduce the amount of computation time. Alternatively, a Monte Carlo integration technique can be used as the variance in the solution is independent of the number of dimensions.

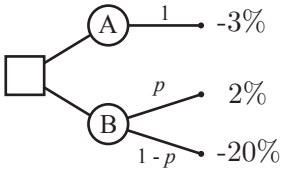
3.5.3 Approximating Exploratory Behavior

Without the sequential decision making technique described later in Chapter 5, the designer will always make a decision as if it were his final decision. In some cases, however, sequential decision making behavior can be approximated using only the techniques described in this chapter. Specifically, we can entice the designer to explore uncertain design alternatives by changing the lottery formulation.

Suppose a designer is choosing designs to analyze in order to determine the values of their attributes. One strategy that a designer might choose is to investigate the "best" alternative (i.e., the alternative with the highest expected utility) first. For a designer with a heavily discounted utility function, this could be a reasonable approximation. Were the designer to choose a second design point to analyze, one could adopt the same strategy: investigate the design with the highest expected utility given the information obtained from the first analysis. However, in many cases the design analyzed first will continue to have the highest expected utility even after the new information is available. This is often true due to the uncertainty in the other alternatives; the risk averse shape of the designer's utility function around $g = 0$ causes the designer to prefer designs will less uncertainty. Were one

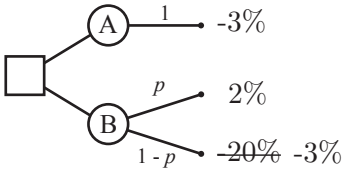
to adopt this strategy of choosing design points, the designer would analyze the same design point repeatedly.

In practice, a human designer would never pick to run the same point through an analysis twice (unless the designer believed the first analysis to be incorrect). The fundamental discrepancy between the human and simulation lies in the lotteries each is considering. Recall that the formulation in this chapter is for *final decisions*. The designer assumes that he is stuck with whatever he chooses. The designer is looking at his or her situation as if it were the following lottery:



The designer has already analyzed Design A, and knows its attributes with certainty, but, due to the formulation, the designer does not want to analyze Design B due to the possibility of receiving a design that misses the requirements by 20%.

However, when the designer is exploring, he can always fall back on a previous design if the one chosen proves to be poor. If Design B did miss the requirements by 20%, the designer could always choose Design A for his final decision. Therefore, the designer is actually faced with the following lottery:



Mathematically, we can represent this by altering the form of Equation 25. Suppose that $u_{\max}$ is a previously analyzed alternative with the maximum expected utility. If the design point with $u_{\max}$ has no uncertainty, then the expected utility of exploring any other design is equal to:

$$\mathrm{E}[u(x)] = \int_{-\infty}^{\infty} \max(u(g), u_{\max}) p(g) dg \tag{31}$$

If the outcome of an analysis does not have zero uncertainty, then the following equation should be used (assuming the beliefs are normal distributions):

$$\mathrm{E}[u(x)] = \int_{-\infty}^{\infty} \max(u(\mu + z\sigma), u(\mu_{\max} + z\sigma_{\max}))\phi(z)dz \tag{32}$$

where μ is the mean of the distribution of g, σ is the standard deviation, ϕ is the standard normal distribution, and z is a dummy variable. These equations state that if an alternative returns less utility than a point already run, then the designer can still select the previous design with higher utility.

It is important to note that there are several limitations to this utility function and many scenarios where this utility function will differ from human behavior. This method only applies to selecting design alternatives, since actions themselves do not have utility outside the sequential decision making framework that will be presented in Chapter 5. If the designer does not analyze the design with highest expected utility first, then the first decision will be inaccurate. In certain time critical situations with large uncertainty, this may be the case; a designer may examine a very uncertain design first in order to weed out candidates.

CHAPTER IV

ISOLATED DECISION DEMONSTRATIONS

In this chapter, we show two test examples of the decision-making framework described in the previous chapter: a single-attribute, unidimensional-design-space problem and a multi-attribute, multidimensional-design-space problem. The purpose of this chapter is to demonstrate an implementation of the framework and to show how the assumptions made lead to certain design decisions.

The first example involves a designer choosing the value of a single continuous variable in order to satisfy one requirement. For this example, we will adopt the perspective of the designer and will only view the information the designer is aware of. This allows us to analyze and evaluate the designer's decisions without any unfair bias. The second example problem involves the conceptual design of a turbojet engine. The designer can select values for overall pressure ratio (OPR) and the turbine inlet temperature (T_4). The designer must deal with often conflicting requirements on cost, weight, and thrust-specific fuel consumption (TSFC). For this example, we will assume an omniscient perspective and will be able to view "reality" in addition to the designer's beliefs.

4.1 Single Dimension Decision

Suppose a designer must choose a design variable x such that a design attribute y is greater than 0.91. Any physical meaning of x and y are purposefully omitted; this allows for a more objective analysis and prevents us from subconsciously reasoning with any additional information not available to the designer in our analysis.

4.1.1 Decisions and Results

The designer's source of knowledge for the decision is a low-fidelity model, shown in Figure 30. Note that Figure 30b is just a zoomed version of Figure 30a. The designer trusts the model for low values of x. One could imagine that past designs included low values of x,

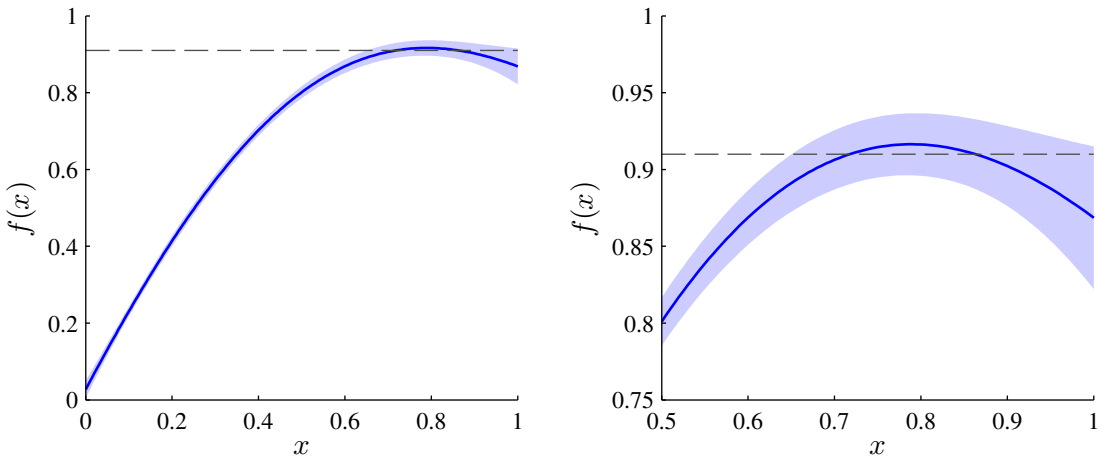


Figure 30: Low fidelity model of y with respect to x

and the model has been validated in that region. For high values of x, the uncertainty in the model grows. The requirement is shown by the dashed gray line. Note that the designer's uncertainty about the true function falls below the requirement; from the designer's perspective, it would be entirely plausible that the requirement cannot be met.

The designer also possesses a high-fidelity model. However, this model is incredibly expensive and time consuming to run. We will assume that the designer only has enough allotted time to run at most two high-fidelity cases before making a final decision. His task in this example is to choose which high-fidelity cases to run. To simplify the problem, we will assume that the designer trusts the high-fidelity model to represent reality with complete accuracy. For the first and last decisions, the standard expected utility formulation given by Equation 25 is used to calculate the decisions. For the designer's second decision, the exploratory approximation from Section 3.5.3 will be used since the designer has the opportunity to explore the design space.

Figure 31 demonstrates the designer's first decision and the designer's rationale behind the decision. Figure 31a shows the designer's expected utility at each x. The designer's knowledge has been superimposed for reference. The utility function matches our intuition about the problem. Low values of x have almost no utility, as they do not meet the requirements. Utility is highest in the region where the designer is most likely to meet the

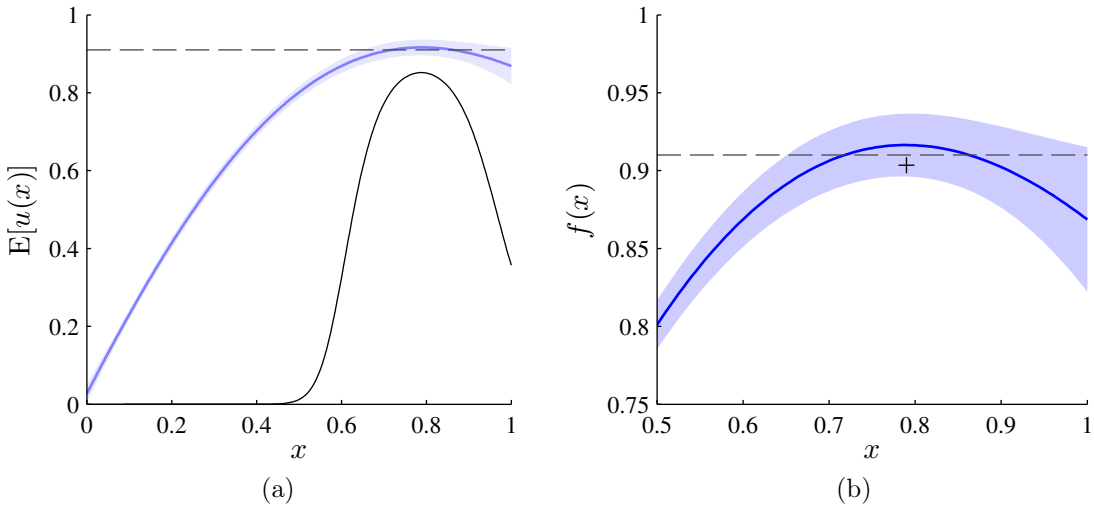


Figure 31: Expected utility of each alternative and outcome of decision

requirements based on the information given to him. As the low fidelity model passes below the requirement, utility drops rapidly. Notice that the utility function never reaches its maximum of one; the designer is not guaranteed that these designs will meet the requirement.

Figure 31b shows the result of the designer's decision; the designer has run the high fidelity model which has returned the new information shown by the plus sign in the figure. To the designer's dismay, the point he ran does not meet the requirement. However, the designer still has enough time to run one more high fidelity point.

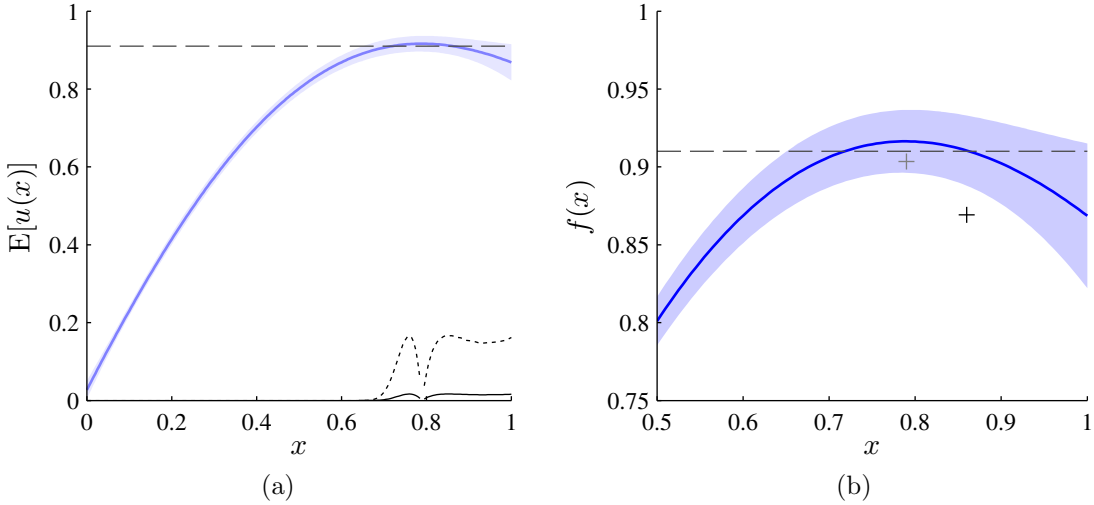


Figure 32: Expected utility and outcome of second decision

Figure 32 shows the designer's second decision. The expected utility is shown by the black line at the bottom of the figure. The dotted line is the same line, but has been magnified 10X to show detail. There are two interesting things to note about this plot. First, the utility function has many local maxima, all of which have very similar expected utility. This implies that the designer was largely indifferent between multiple options; specifically to the left and the right of his first decision. Second, the expected utility of all options is relatively low. This implies both that the designer does not believe it likely that any value of x will do better than his first decision, and that the designer is largely indifferent between his options. Slight perturbations to the utility function parameters could have resulted in a different decision.

Figure 32b shows the result of the designer's second decision. Notice that the high fidelity model returned a value outside of the designer's original beliefs about the model.[1] Put another way, the designer would be surprised by this result.

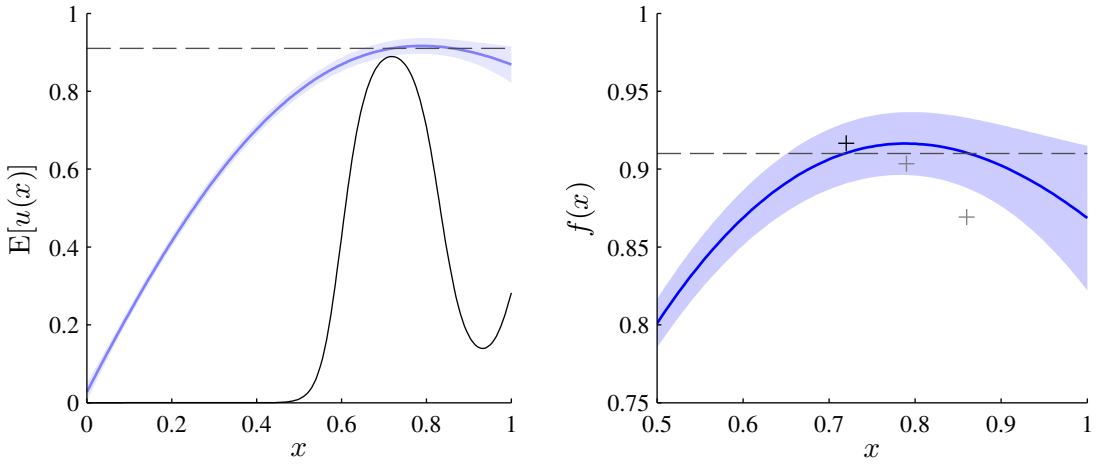


Figure 33: Expected utility and outcome of third decision

Although the designer's second decision was a worse design than his first, he can still use the new information for his final decision. Figure 33 shows the designer's final decision. The designer chose a point to the left of his original high-fidelity decision. As we can see in

[1]Note that "outside the designer's beliefs" is meant to imply outside the two sigma boundary. Since the designer's beliefs are normal distributions, nothing is outside the realm of possibility, but some outcomes are extremely unlikely.

the figure, this design was successful in meeting the requirements.

4.1.2 Analysis of Decisions

Figure 34 shows the true function compared to the designer's low fidelity model. As the designer expects, the true function matches the model for low values of x. However, his beliefs are mistaken at higher values of x; he trusts the model too much. In many regions, the true function falls well outside his belief structure.

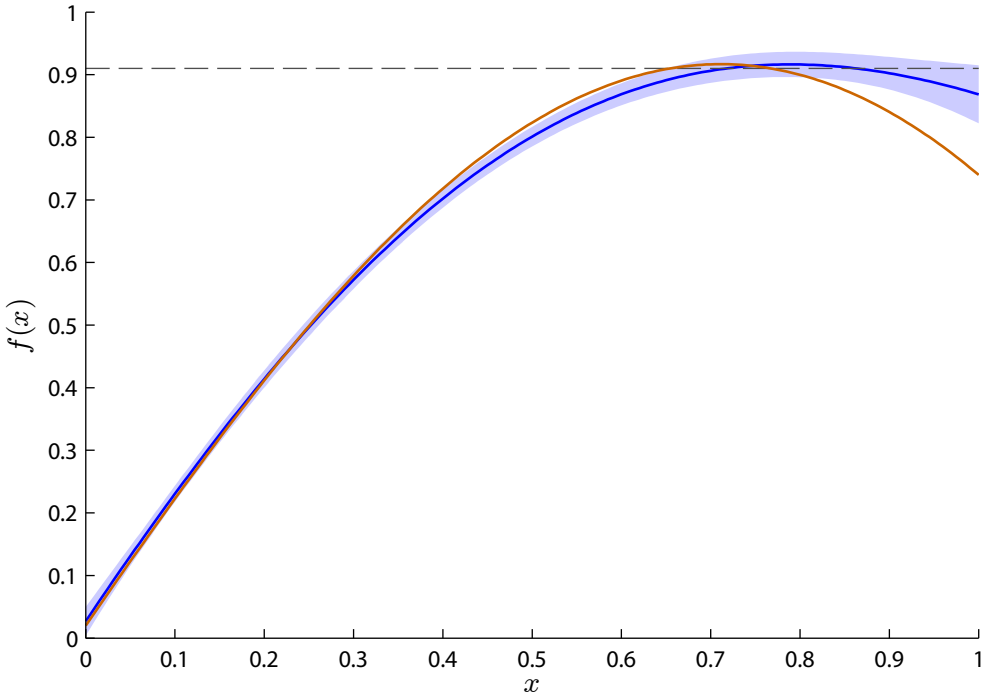


Figure 34: Comparision of low fidelity model and true function

The first decision occurred largely as one would expect; the low-fidelity analysis informed the designer of where good designs might exist, and he chose the design most likely (according to the model information) to meet the requirements.

In order to understand the designer's next decision, we need to understand how his belief structure updates in response to the new information. The first high fidelity point returns lower than his expectation, but still within the bounds of plausibility. Figure 35 shows in magenta the designer's updated belief structure after receiving new information. For comparison, the designer's original belief structure is still shown in blue. Notice that the

uncertainty collapses at the high fidelity point. The uncertainty in the region immediately surrounding the point has greatly diminished. Notice also that the expectation is now below the low fidelity model. Overall the designer believes that the low-fidelity model simply overestimated the original function; recall that the designer does not expect large deviations from the low-fidelity model (even though his model is grossly inaccurate in this region).

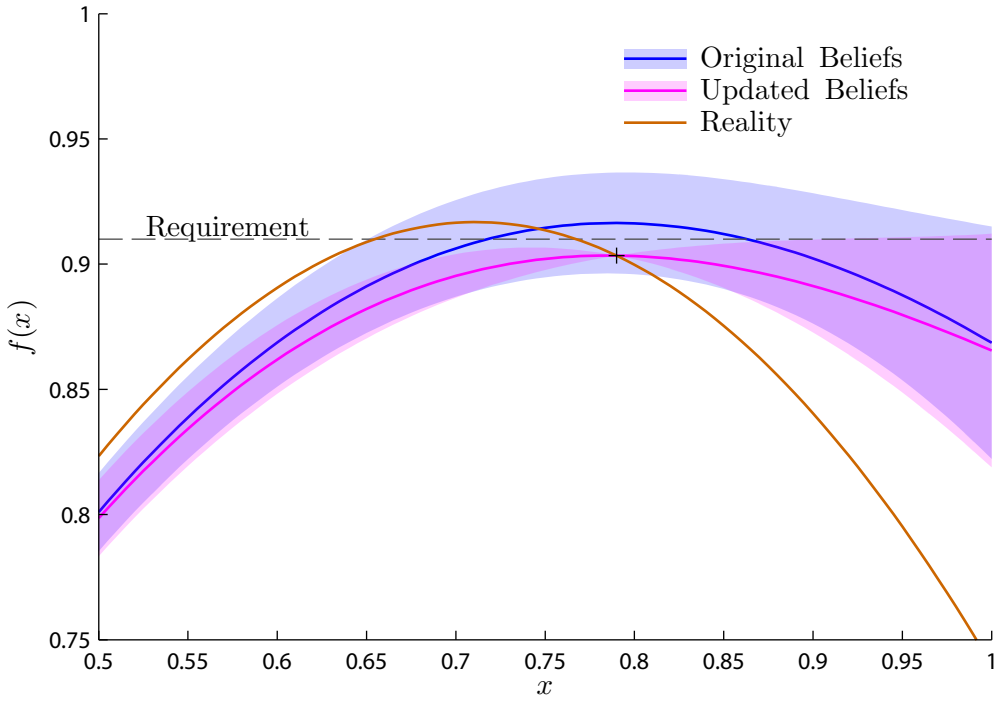


Figure 35: Belief update after outcome of first decision is revealed

To the left and right of his first decision, the designer believes it possible (but rather unlikely) that a design could meet the requirements. The shape of the Gaussian process model illustrates why the designer was relatively indifferent to a designs on both sides of the first decision: he believes that the "optimum" point in the low fidelity model is still the optimal point in reality.

Once the designer obtains his second piece of information, he realizes the inaccuracy of his low fidelity model. As can be seen in Figure 36, the designer has to make radical changes to his belief structure in order to adapt to new information. The designer is able to seamlessly combine the trusted information at low values of x with the certain information

at high values of x. This allows him to infer that the ideal design lies to the left of his original guess.

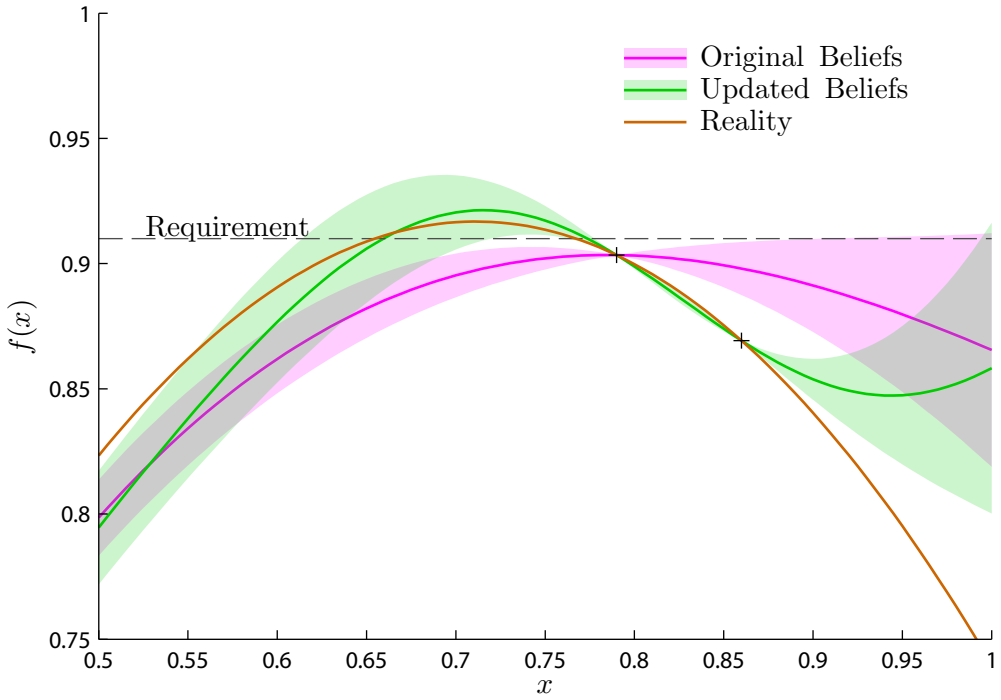


Figure 36: Belief update after outcome of second decision is revealed

Overall, I believe this example illustrates the robustness of the framework. Although he was given poor initial information, the designer was still able to recover and learn from mistakes. At the same time, each of his decisions was plausible; I believe many people given similar information would make similar decisions. Obviously this assertion requires further validation, which will likely require a more thorough visit of the psychology literature for empirical evidence of how decision makers behave in a similar context.

4.2 Multi-Dimension Decision

For the multidimensional decision problem we will adopt an omniscient perspective, capable of seeing both reality and the designer's beliefs simultaneously. This problem will be conducted three times with varying amounts of initial information given to the designer. The designer's preference structure will remain the same for each case. For brevity, we will analyze only the first case in detail, but the differences in behavior and their underlying

causes in the other examples will be discussed.

4.2.1 Baseline Designer

This example problem involves the conceptual design of a subsonic turbojet engine. The designer is tasked with selecting the design overall pressure ratio (OPR) and turbine inlet temperature (T_4) for the engine. The design has three requirements:

1. Thrust Specific Fuel Consumption (TSFC) $<$ 0.9 lbm/lbf/hr
2. Cost $<$ \$800,000
3. Weight $<$ 2,400 lbs

The engine is sized for a design thrust of 20,000 lbs. The "reality" models of engine performance were created using ENGGEN [24]. Surrogate models were created of these variables to smooth out any of the noise in the program. The production cost was calculated using Reference [48].

For this problem, the designer is given the following information:

- A low fidelity model of TSFC which the designer trusts to be accurate to within 5%.
- Weight data from historical designs of past engines. The designer trusts this information to be both accurate and reproducible.
- Production cost information from the aforementioned past designs. Since costs tend to change over time, the designer does not trust this information to be completely reproducible but trusts it as an estimate within $\pm$5%.

In order to examine sequential decisions and learning, the designer can obtain point design information after he has made a decision. One can imagine that the designer has access to a high fidelity model; alternatively, we could imagine that the designer is asking other engineers to perform an initial design and return performance and cost estimates. We will assume that obtaining this information is very expensive, and the designer wants to minimize the number of times he samples a point design.

Figure 37 shows a map of the design space overlayed with both the requirements and the designer's multi-attribute utility function. In this example, the designer perceives all the requirements as soft requirements. Since at least one requirement is being met in the

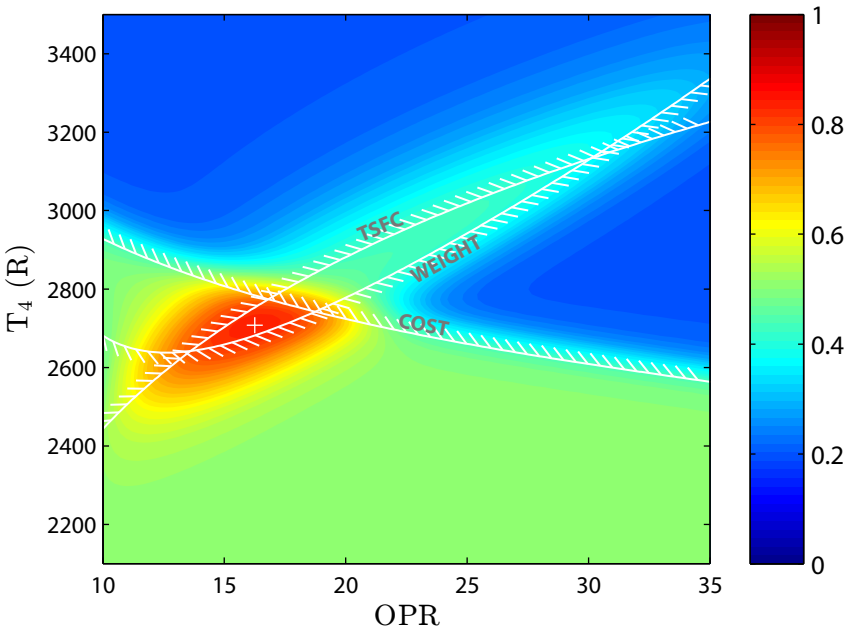
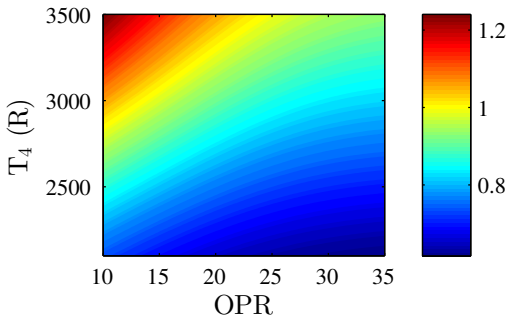


Figure 37: Designer's utility function

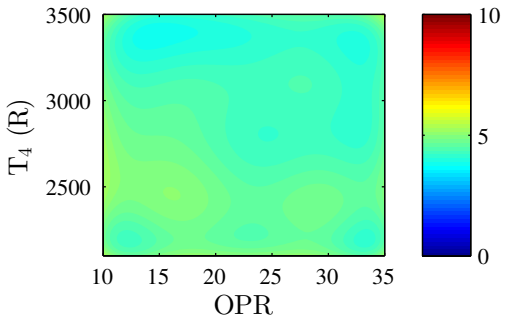
entire design space, no portion of the design space has zero utility. The actual requirements are shown as the labeled hatched lines; the overall feasible design space is relatively small. The optimal design is marked by the white plus sign. Were the designer omniscient of the entire space, this would be his selection. However, as we will see, the designer only has rough approximations of the design space, so his choice will be much more difficult.

Figure 38a shows the designer's low-fidelity model of TSFC. Figure 38b shows the 95% confidence interval in both dimensions. For ease of interpretation, the confidence interval is shown as a percentage of expectation. Therefore, for TSFC, the designer trusts the low-fidelity model nearly uniformly to be accurate to within ±5%. Figure 38c shows the actual error between reality and the model. While the model is better in some areas than others, the error stays relatively consistent between 2 and 4%. Therefore, unlike the last model, the designer's beliefs about the accuracy of this model are in line with reality.

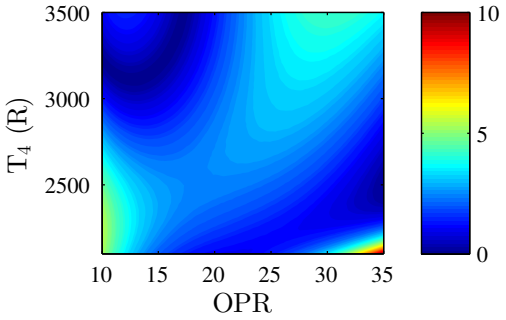
Figure 39a shows the "true" weight model for the engine (the scale is in 1,000 lbs). Figure 39b and 39c show the designer's beliefs about how weight varies with OPR and T4. The known past designs are indicated by the x's; note that at these points the uncertainty (Figure 39c) collapses to zero. Overall, the designer is able to form a reasonably accurate



(a) TSFC model (lbm/lbf/hr)

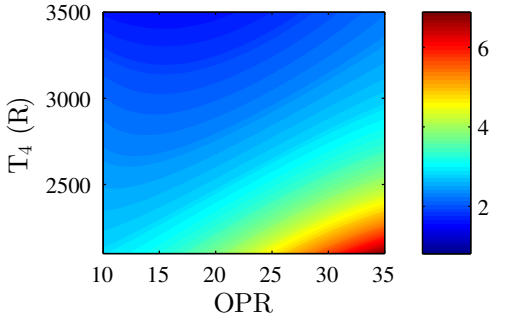


(b) TSFC uncertainty (% of expectation)

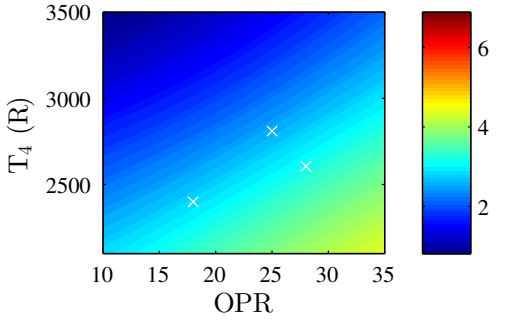


(c) Percent error between model and reality

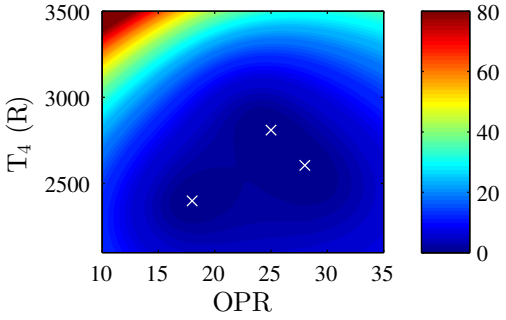
Figure 38: Low fidelity model of TSFC



(a) Weight reality (1,000 lbs)

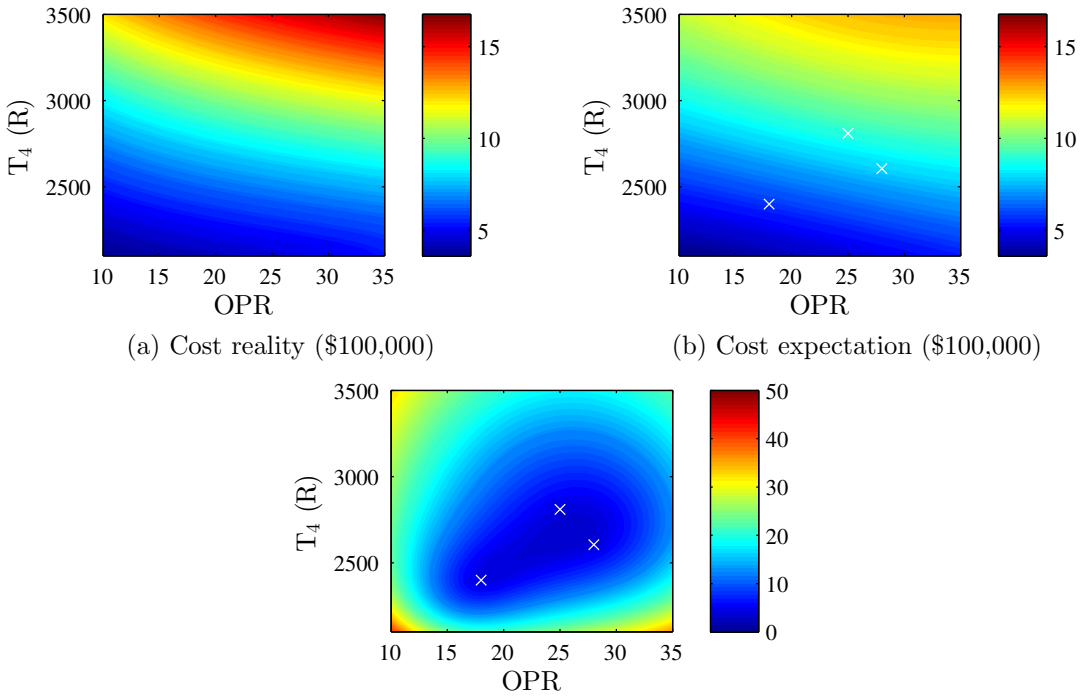


(b) Weight expectation (1,000 lbs)



(c) Weight uncertainty (% of expectation)

Figure 39: Comparison of beliefs and reality in regards to engine weight



(a) Cost reality ($100,000)

(b) Cost expectation ($100,000)

(c) Cost uncertainty (% of expectation)

Figure 40: Comparison of beliefs and reality in regards to production cost

model of reality in the region where he knows point designs. However, there are discrepancies, especially in regions outside the three known designs. Specifically, the designer underestimates the weight at very low T4 and high OPR. The designer also cannot infer the non-linearities in weight at low OPR and high T4.

Figure 40 shows a comparison of reality and the designer's beliefs about cost. Notice that the designer's uncertainty about his beliefs never decrease to zero, even at the "known" points. This is due to the designer's distrust of the cost figures; he believes them to be a good estimate, but not entirely accurate.

Figure 41 shows the designer's expected utility for his first decision. For reference, the actual constraints are still shown, but recall that the designer can not actually see the exact location of these constraints. The past designs with weight and cost information are marked with a black x. The designer's decision is marked by the circle. At the decision point, the designer's expected utility of 0.940 indicates a relatively high confidence that this point meets the requirements. In fact, the designer is very close; he barely misses the weight

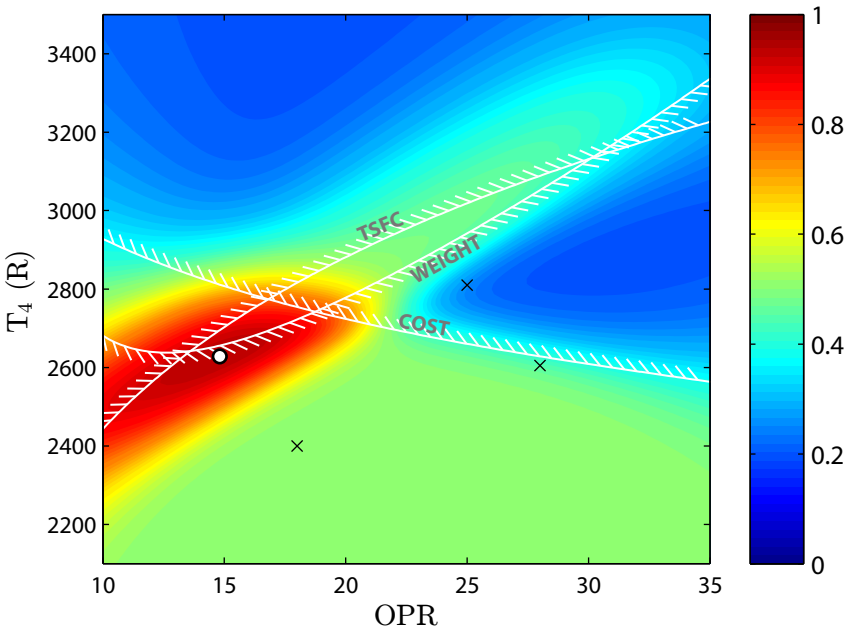
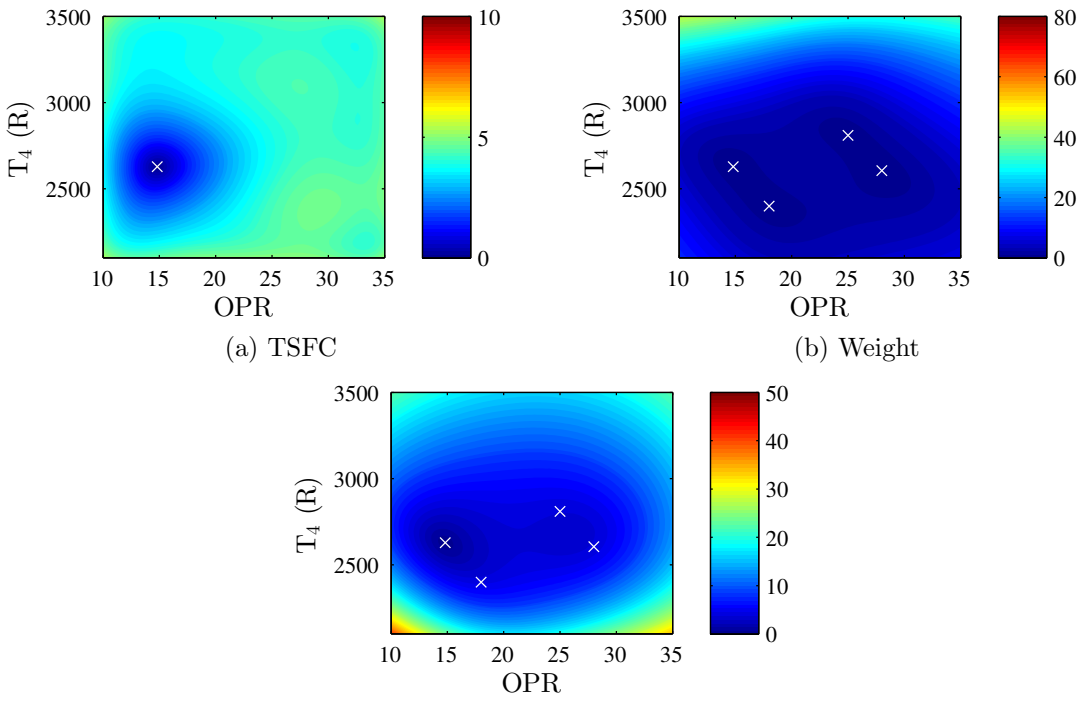


Figure 41: Expected utility of the design space for the designer's first decision

requirement by 1.1%.

If we compare Figure 41 with Figure 37, we can see that the designer's beliefs about reality do not differ drastically from reality itself. The designer's error generally lies in estimating the weight requirement; the designer is unable to infer the curvature of the iso-contour lines seen in Figure 39a. Hence, the region where the designer expects to meet all requirements is much larger than reality. The designer is able to get very near the TSFC constraint, because he knows this constraint with the most certainty. Notice that the designer chooses to stay away from the cost constraint, which is known with the least certainty. The designer believes that he would still meet the weight requirement in this area, but this belief is mistaken.



(a) TSFC

(b) Weight

(c) Cost

Figure 42: Updated uncertainty in beliefs after result of first decision is revealed (shown in % of expectation)

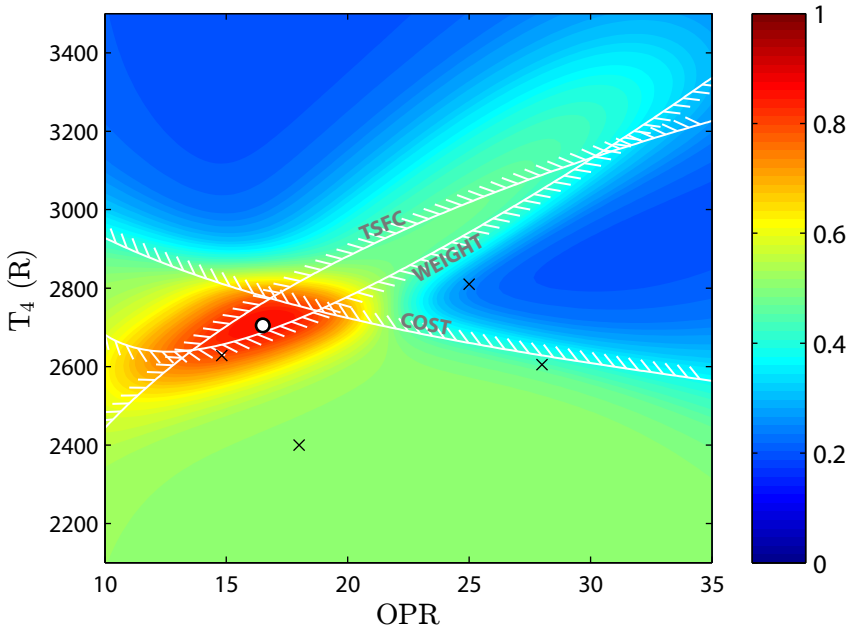


Figure 43: Expected utility of the design space for the designer's second decision

Since we have incorporated models of "reality", we can let the designer see the outcome of his decision and make a different selection. Figure 42 shows the designer's uncertainty with respect to each attribute after the first decision. For TSFC in Figure 42a, the Gaussian process model is able to seamlessly integrate the point-design information with the low-fidelity model. In the cost model in Figure 42c, the new data point is assumed to be fully reproducible, so the designer associates zero uncertainty with it.

Figure 43 shows the designer's second decision. With the new information, he realizes the error in his beliefs about the true weight function and updates his selection accordingly. As a result, his new decision meets all the requirements.

4.2.2 Experienced Designer

Figure 44 illustrates the designer's decision if we add another reference point for the cost and weight models. As mentioned in Section 3.3.1, the designer's experience is characterized by his accuracy in expectations and uncertainty in his beliefs. An additional design point will give him a more accurate representation of the design space and will reduce his uncertainty in the feasible region of the design space. This new data point is at a lower OPR and

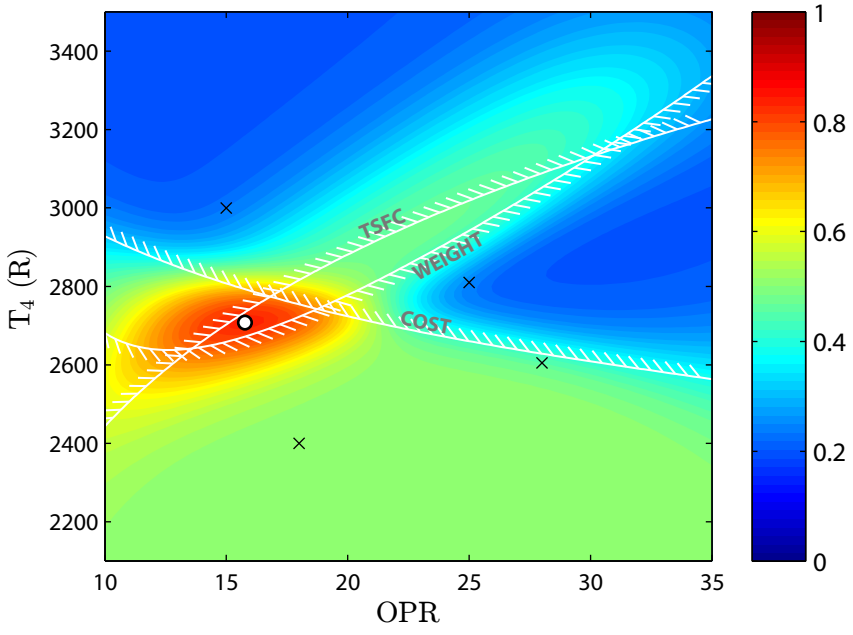
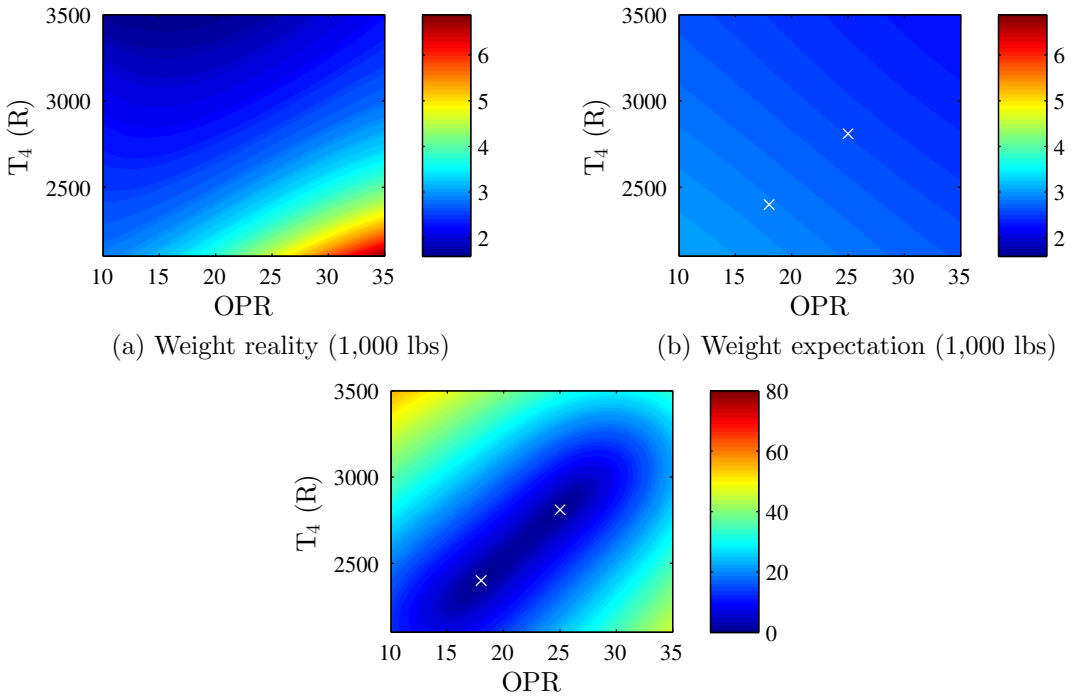


Figure 44: Expected utility of more experienced designer

higher T4 value than his previous data points. It is also incredibly useful, since it allows the designer to base his inferences on interpolation rather than extrapolation. The designer can make a better decision, since the feasible space is largely within his domain of knowledge. As the figure indicates, the designer is able to make a decision that meets all the requirements on his first attempt.

4.2.3 Inexperienced Designer

In this example, one of the designer's three original data points from the baseline example has been removed. Figure 45 shows the designer's belief about the variation in engine weight across the design space. The lack of information seriously inhibits the designer's ability to understand the shape of the design space, especially since the two points he has are nearly on an iso-weight line. Based on the available information, the designer is unable to determine if OPR and T_4 have any large effects on the weight of the engine. In fact, the designer has inferred a trend that is almost 90° out of sync with reality. The designer's beliefs regarding



(a) Weight reality (1,000 lbs)

(b) Weight expectation (1,000 lbs)

(c) Weight uncertainty (% of expectation)

Figure 45: Discrepancy between the inexperienced designer's beliefs and reality in regards to engine weight

cost are not seriously inhibited since the data points are nearly perpendicular to the iso-cost contour line. The designer also retains the low-fidelity model of TSFC.

Figure 46 shows the designer's expected utility of each point in the design space. Notice the striking difference between this figure and the baseline designer's utility in Figure 41. For clarity, the designer's decision has been marked by the arrow. In this figure, the utility at every point is generally low; no point has an expected utility of above 0.57. This demonstrates the designer's general uncertainty in his decision; unlike the previous two scenarios, at no point does he expect to meet all three requirements. For reference, this particular designer would have been indifferent between choosing this design, and a design where every requirement was missed by 1.2% with certainty. The designer would also be indifferent to a design where two requirements are barely met, and one is missed by 3%.

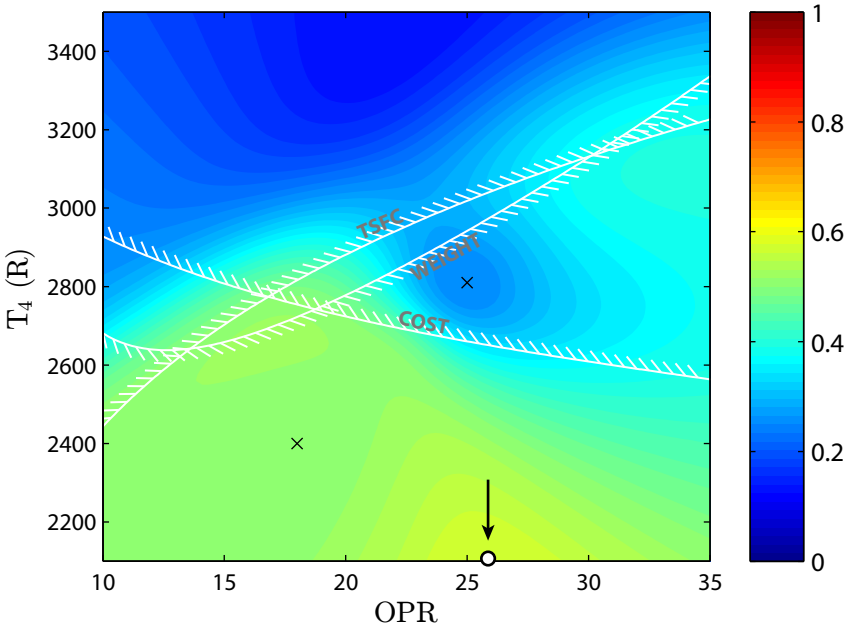


Figure 46: Expected utility of inexperienced designer

By examining the shape of the utility function, we can infer his reasoning for the choice he made. The designer is fairly confident about TSFC. We can see how the expected utility has a sharp drop that mirrors the TSFC requirement; the offset is due to the fact the low fidelity model underestimates the actual TSFC. At both known locations, the designer

knows the weight is well outside the requirements; the upper right point is 8% overweight and the lower left point is 18% overweight. Meanwhile, the designer is confident that he meets the cost requirement at the lower left point and knows that he misses it at the upper right point.

This puts the designer in conflict as he knows that if he chooses the lower left design point, he is virtually guaranteed to exceed two of the requirements by a wide margin: TSFC and cost. However, he is also guaranteed at this point to miss the weight requirement. Therefore, the designer decides to compromise; he will move away from the certain point to an area of uncertainty. The designer chooses a direction which he is fairly confident will still meet cost. It is possible in this region that the weight will get worse. However, the certain design is so far from meeting the weight requirement that being further away would not be too detrimental, and the designer is willing to take the risk.

Unfortunately for the designer, he completely misses the feasible space. Like the baseline designer, we can give this designer the information at the point he chose and see how he corrects himself. Figure 47 shows the designer's expected utility for his second decision

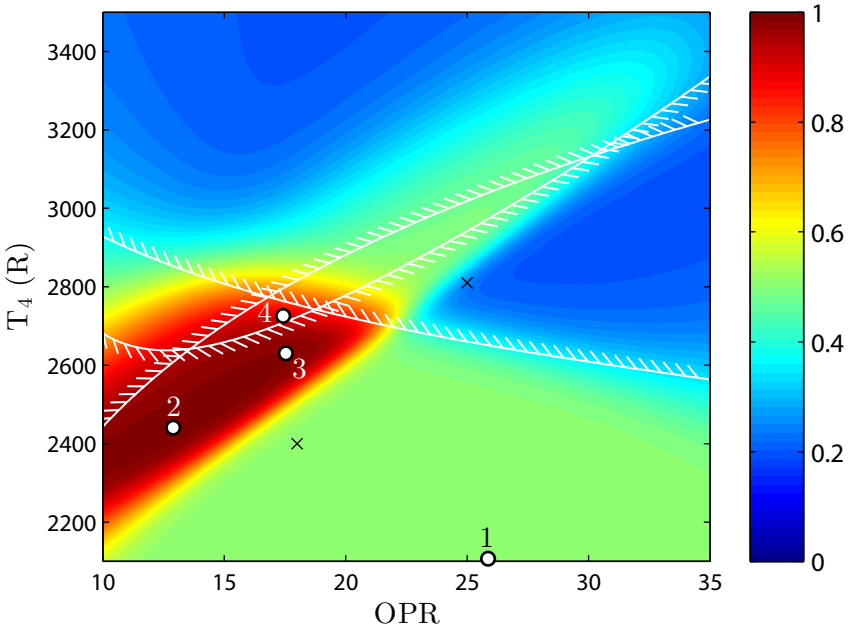


Figure 47: Expected utility for designer's second decision and location of second, third, and fourth decisions

and the results of his subsequent three decisions. He now has a better understanding of the variation of attributes with design variables, but the understanding is still weak; the information he has added is relatively far from the feasible design space. The designer's second decision is still outside the requirements. As the figure shows, it takes the designer a total of four decisions in order to find a feasible design. This behavior demonstrates the value of information close to the feasible design space. As in real design problems, the designer's information needs to be relevant to designs near the requirements if he is to effectively solve the problem.

CHAPTER V

SEQUENTIAL DECISION MAKING

The formulation outlined in Chapter 3 can be effective for capturing design decisions in isolation when the outcome of the decision has no bearing on the designer's future decisions. When future possibilities are considered, however, the model tends to yield inaccurate results. Consider the notional two-dimensional design space shown in Figure 48. The design space is composed of three distinct regions. Most of the design space, shown in white, is characterized by low expected performance. The dark gray region is characterized by moderate performance with low uncertainty. Perhaps the designer knows that all hard requirements in this region will be met, but not all soft requirements. The lighter gray region is characterized by high uncertainty; it is possible that this region performs better than dark gray region, but it is also possible that the region performs poorly. If the designer had to choose the "best" design based solely on the information above, most designers would choose the dark gray region; the expected utility would be highest in the dark gray region, significantly lower in the light gray region due to the large uncertainty, and near zero in the white region where poor performance is expected.

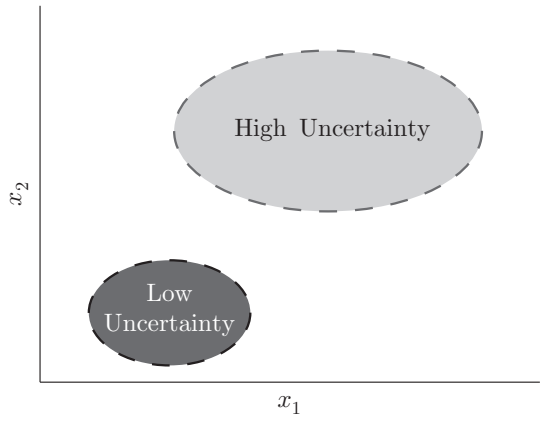


Figure 48: Expected utility fails to capture the best choice for exploration

Suppose, however, we gave the designer a perfectly accurate model which could sample the entire design space, and allowed the designer to run the model once at any point in the space. After receiving the new information from the high fidelity model the designer would then be asked for his or her final decision. Given the option to obtain new information, how would the designer choose?

In this scenario, a human designer may choose a design point that is very different from the region of highest expected utility. If the designer's uncertainty in the dark gray region is low enough, and he believes the light gray region might contain a better design, it may be more helpful to the designer to run the high fidelity model in the light gray (uncertain region). Were the designer to discover that the light gray region contained better designs than the dark gray region, the designer would choose a design in the light gray region. Otherwise, the designer could default back to the dark gray region. Running the high fidelity model in the dark gray region yields little to no new information; the designer already has high confidence in his beliefs in this area.

Thus we see a fundamental disconnect between the expected utility formulation and the reasoning of a human designer. As we will see in Section 5.2, the formulation in Chapter 3 assumes that the designer is making a final decision; there is no mechanism to account for the possibility of future actions. In this manner, the designer only maximizes the utility of each step as if it were his last. The goal of this chapter is to extend the expected utility formulation to account for long-term expected utility. To do this, the expected utility of actions will be calculated, and the utility function will be altered to account for time and budget. Section 5.1 will illustrate this next extension with a simple example designed to show how the underlying equations coincide with the reasoning a designer might use to make his or her decision. Section 5.2 formalizes the mathematics of the decision process. Section 5.3 will generalize the concepts of alternatives and attributes, freeing our models from the confines of simple design variables. Finally, Section 5.4 will address the inclusion of time in the utility function and proposes some example models.

5.1 Illustrative Example

We can develop a method for modeling sequential decision making behavior by examining how one might reason through the example shown in Figure 49. Suppose a designer were choosing between two designs, A and B. The designer knows with absolute certainty that Design A misses the requirement slightly. Design B is uncertain and could be significantly better or worse than Design A. If this were the only information available to the designer, we would simply calculate the expected utility of each alternative to determine which design the designer chose. Suppose, however, that the designer was given a third option; the designer could run a high fidelity model that would tell him or her the attributes of Design B with certainty. Under what conditions would the designer run the high fidelity model?

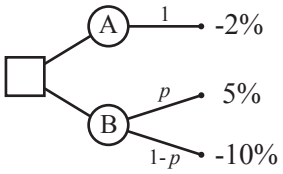


Figure 49: Simple decision between two designs

To answer this question, we will transform the decision between *designs* into a decision between *actions* as shown in Figure 50. In this thesis, actions in a decision tree are represented by diamonds. If the designer chooses Action I, he or she will make a decision without running the high fidelity model. Under Action II, he or she will run the high fidelity model and then make a decision. For now, we will assume that the cost of running the high-fidelity model is negligible. To determine which action the designer will choose, we will calculate the expected utility of each action.

If the designer performs Action I, he or she will choose the design with the highest expected utility based on his or her prior knowledge. Therefore, the expected utility of Action I is equal to the maximum of the expected utilities of the designs.

$$\mathrm{E}[u(\mathrm{I})] = \max\Big(\mathrm{E}[u(\mathrm{A})], \mathrm{E}[u(\mathrm{B})]\Big) \tag{33}$$

$$= \max\Big(u(-2\%), pu(5\%) + (1-p)u(-10\%)]\Big) \tag{34}$$

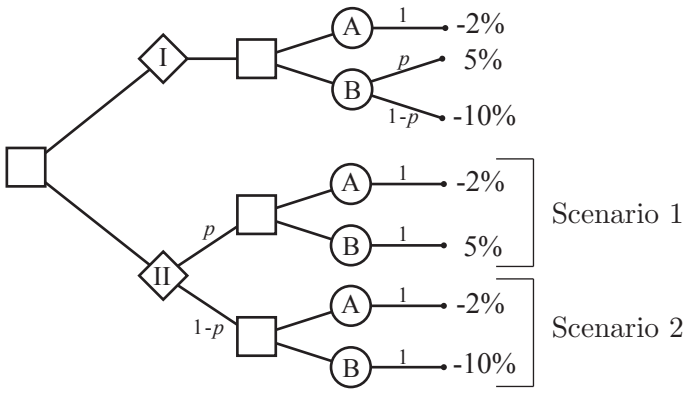


Figure 50: Decision between two actions: (I) making a decision and (II) running a high fidelity model

Which alternative has the highest expected utility depends on the specifics of the designer's utility function. For this example, we will not make any assumptions about the functional form except for monotoncity; this will allow us to generalize the results.

To calculate the expected utility of Action II, we will examine the ends of the decision tree and propagate the utility backwards. Consider the first possible decision shown after Action II, labeled Scenario 1; in this scenario, the designer ran the high fidelity model which determined Design B to exceed the requirements by 5%. Were this to occur, the designer will obviously choose Design B, regardless of the details of his or her utility function. In the alternative scenario (labeled Scenario 2), Design B was found to miss the requirement by 10%. In this scenario, the designer will obviously choose Design A. Note that the designer does not know which of these two scenarios will happen until after he or she runs the high fidelity model. Therefore, the designer must base the likelihood of these scenarios on his or her beliefs before deciding between the actions. In the designer's mind, the likelihood of the Scenario 1 is simply the likelihood of Design B exceeding the requirement, the probability p. The utility of Action II can be determined by calculating the utility of the final decision in each scenario multiplied by the probability of reaching that scenario.

$$\mathrm{E}[u(\mathrm{II})] = pu(5\%) + (1-p)u(-2\%) \tag{35}$$

Comparing Equations 34 and 35, it is clear that the expected utility of Action II will always be larger than the expected utility of Action I for any monotonically increasing utility

function. In other words, if the cost of running the high fidelity model is negligible to the designer (i.e. not included in the utility function), then the designer will always choose to run the high fidelity model. Similarly, if running the high fidelity model were negligible in terms of cost and time, we would expect a human designer to make the same decision.

If we remove this assumption of negligible model cost, we can examine the conditions under which the designer would and would not run the model. Suppose that the high fidelity model takes a very long time to run and that the designer's preferences are influenced primarily by this temporal effect. The cost does not strictly have to be temporal, but the mathematics will yield similar results regardless. Equations 34 and 35 now become functions of time.

$$\mathrm{E}[u(\mathrm{I})] = \max\Big(u(-2\%, t_1), pu(5\%, t_1) + (1-p)u(-10\%, t_1)]\Big) \tag{36}$$

$$\mathrm{E}[u(\mathrm{II})] = pu(5\%, t_2) + (1-p)u(-2\%, t_2) \tag{37}$$

If Action I is chosen, the designer will finish sooner at time t_1. If the designer chooses Action II, he or she will have to wait until time t_2 to make a final decision where $t_2 > t_1$. For simplicity, we will assume that the effect of waiting until t_2 is utility independent from meeting the requirements and is represented by a factor, $\gamma < 1$ (the utility independence assumption will be more thoroughly examined in Section 5.4):

$$u(g, t_1) = u(g) \tag{38}$$

$$u(g, t_2) = \gamma u(g, t_1) = \gamma u(g) \tag{39}$$

Combining Equations 36 and 37 with Equations 38 and 39, respectively, we arrive at a more generalized form of expected utility:

$$\mathrm{E}[u(\mathrm{I})] = \max\Big(u(-2\%), pu(5\%) + (1-p)u(-10\%)]\Big) \tag{40}$$

$$\mathrm{E}[u(\mathrm{II})] = \gamma pu(5\%) + \gamma(1-p)u(-2\%) \tag{41}$$

Examining these equations, it is now not always true that the utility of Action II is greater than the utility of Action I. We can identify three parameters that, in conjunction, would deter the designer from choosing Action II:

- $u(5\%) - u(\text{-}2\%) \approx 0$: If the designer's preferences for a $g = -2\%$ design and a $g = 5\%$ design are similar, then it likely that the designer will choose not to run the high fidelity model.
- Small values of p: If the possibility of achieving a 5% design is small, then the designer may prefer not to run the high fidelity model.
- Small values of γ: If the cost of running the high fidelity model is high, the designer may prefer not to run the model.

Each of these parameters represent a valid concern for the designer; we could imagine a human designer contemplating the same considerations when making this decision. Note that the sequential-decision making problem in this example enticed the designer to explore the design space, even though exploration was not an explicit parameter in the utility function. Exploration had inherent value to the designer because it assisted the designer in choosing a better design. This convenient feature allows the framework to retain the simple utility functions derived in Section 3.4 with only small modifications.

5.2 Calculating the Utility of Actions

Until this chapter, decisions have been viewed in isolation; the current decision was assumed to have no impact on future decisions, and no future decision could alter the outcome of the current decision. To look at this another way, the decisions examined have been formulated in the context of final decisions; the designer returns his or her decision and cannot perform any actions in the future that would change the final outcome the design. In this chapter, we will extend the framework to include all decisions. As we will see, this allows the framework to incorporate new alternatives which the framework was previously unable to model.

The methods described in this section are similar to normative methods developed in a variety of fields. Like the formulation in this chapter, many of these methods focus on determining the *value* of an action or activity. Many of these methods value activities based on the information they provide. *Information Economics*, for example, studies how information affects economic decisions. Loch and Terwiesch applied the findings of information economics to the design process [44]. In their paper, several heuristics are developed that

can guide an engineer's decision making process based on the cost of information. Browning et al. examine the value of activities in the product develop process [9]. Their *risk value method* assigns a value for an activity based on how the information reduces performance risk. Most closely related is work by Thompson, Lee and Paredis [69, 41]; decision trees are used to analyze the trade-offs between the quality of the artifact being designed and the cost of the design process itself. The formulation in this chapter will also incorporate decision trees in order to calculate the expected utility of actions.

The sequential decision making process can be generalized by viewing the design process as a decision tree and back-propagating expected utility. To simplify the analysis and to leverage the previously outlined models, two assumptions are required:

- *The designer is concerned with only the final outcome of the design*: The designer is making decisions based on trying to improve the performance of the final design and not any intermediate result or reward. If the designer had a performance review in the middle of a design, this assumption may be inaccurate.
- *The time required to decide between a set of actions is much smaller than the time it takes to perform those actions*: In other words, the designer is not making this decision under any time pressure.

The validity of the former assumption is analyzed in greater detail in Section 7.1.1. For most design situations the latter assumption is easily defensible.

At each decision, the designer is in an *information state*, s, which defines all the information the designer has obtained up until that point. In each state, the designer has a set of possible actions. These actions are not restricted to selecting a design variable; a designer can choose to meet with another team, run a computer model, perform an experiment, or any other action that influences his knowledge about the design alternatives. Each action may lead to one of several possible outcomes. In most cases, the designer does not know the outcome ahead of time with absolute certainty. This uncertainty is represented as a probability distribution of outcomes. In other words, there is a probability, $P(s, a, s')$, that action a in state s will lead to state s' Note that we are viewing these probabilities from a Bayesian perspective; each action may have a deterministic outcome, but the lack

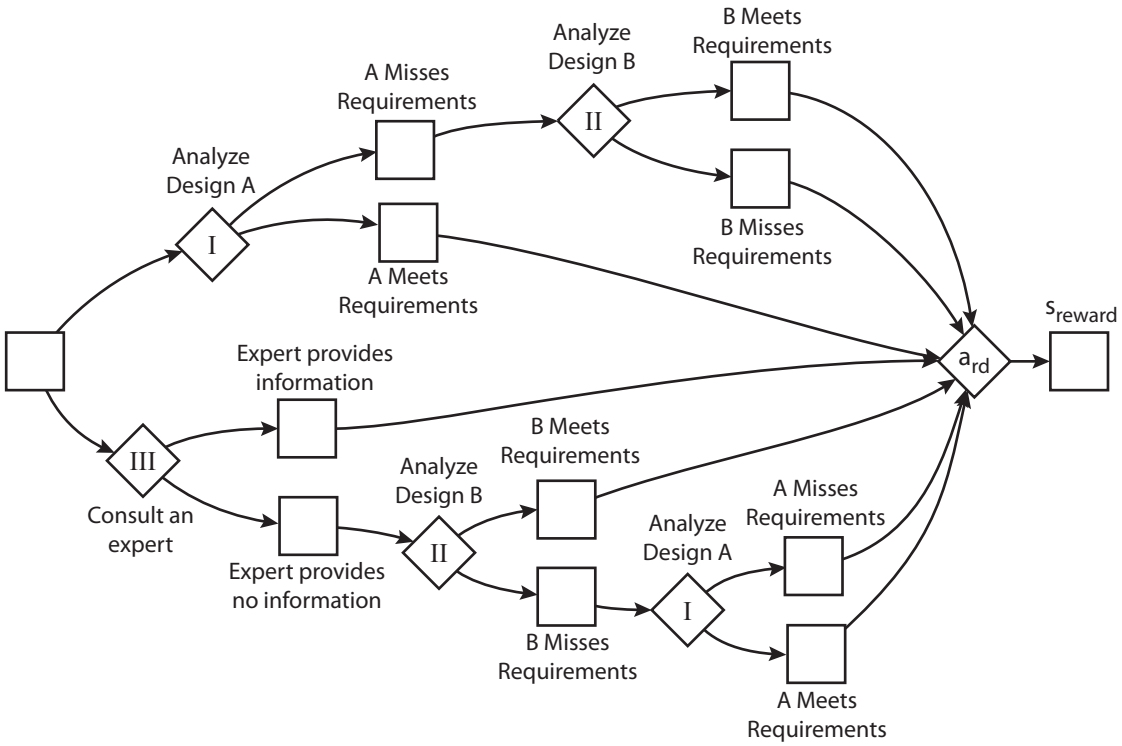


Figure 51: Example of several possible decision trees which a designer might consider

of knowledge regarding the outcome is represented by the designer's subjective probability distributions.

For every design problem, the designer has a final state which I will define as s_{reward}. Here the use of *reward* is synonymous with *utility*. The designer can enter this state at any time by selecting a final design. I will refer to this action as the *return decision* action, or a_{rd}. Based on the first assumption outlined above, expected utility is only calculated when the designer returns a decision.

Figure 51 shows a notional set of decision trees that a designer might consider.[1] In this figure, states (or decisions) are represented as squares, with actions again being represented by diamonds. Note that although s_{reward} is represented by a square, no decision is made in this state. Each state is uniquely defined by the time at which the designer enters the state, and the information which the designer possesses in that state.

[1]Note that numerous additional branches and states exist given this set of actions. I have only listed a few possibilities.

Since expected utility is calculated when the designer returns a decision, the expected utility of a_{rd} is simply equal to the expected utility given the information in the final state:

$$\mathrm{E}[u(a_{\text{rd}})] = \mathrm{E}[u(\boldsymbol{x}|s)] \tag{42}$$

If returning a decision is the only action available for the designer in a given state, then the expected utility of that state is the expected utility given by Equation 42. For all other states, the expected utility of that state is equal to action with the highest expected utility.

$$\mathrm{E}[u(s)] = \max_{a}\big(\mathrm{E}[u(a|s)]\big) \tag{43}$$

This assumes that the designer will always select the action with the highest expected utility at each stage of the design process. The expected utility of an action is calculated by summing the expected utility of each outcome state by the probability of that outcome occurring:

$$\mathrm{E}[u(a|s)] = \sum_{s' \in S} P(s, a, s')\mathrm{E}[u(s')] \tag{44}$$

Therefore, the expected utility of any state is given by the following recursive equation:

$$\mathrm{E}[u(s)] = \max_{a}\left(\sum_{s' \in S} P(s, a, s')\mathrm{E}[u(s')]\right) \tag{45}$$

The expected utility of actions is calculated in this manner for two reasons: (1) following this policy leads to logical decisions in regards to performing actions and managing the designers resources, and (2) this policy provides a clear, mathematically-defined mechanism for automating the decision process. In regards to the first reason, propagating the possible outcomes promotes a natural balance between exploration and exploitation. Notice that no "utility of exploration" parameter was added to the utility function; the designer explores when it is beneficial to the final design. These calculations also lead the simulated designer to use resources in a way that mimics human behavior. Given the option of a low fidelity model and a high fidelity model, the designer typically selects the low fidelity model to obtain useful information before running an expensive high fidelity model. In regards to the second reason, this mechanism provides a method of down-selecting a decision from a (potentially infinite) set of alternatives that is simple to define. In reality, a designer may

adopt a different strategy for choosing a design, and this strategy may be dependent on the particular design scenario. In my opinion, the use of this strategy provides a balance between realism and generalizability; the policy provides reasonable decisions and can be used for virtually any set of actions and outcomes.

In practice, one computes Equation 45 by propagating the designer's decision tree and working backwards. Consider the case where a designer is choosing to run an analysis on one of three different alternatives: x_1, x_2, and x_3. Suppose that the analysis returns only a binary result, α or β. We will assume that the designer has enough time to run the analysis on only one of the alternatives. After the designer receives information from the analysis, he or she will choose one of the design alternatives for the final design.

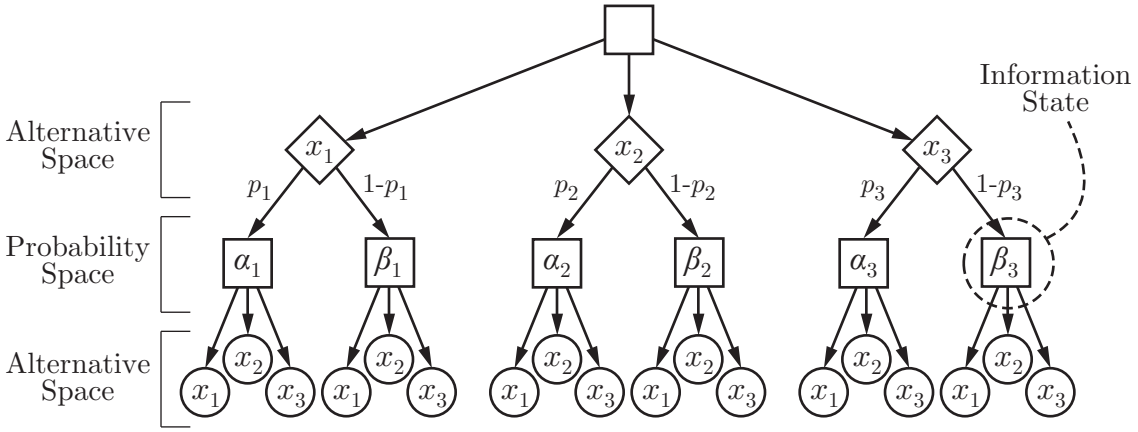


Figure 52: Two-level decision tree with three alternatives and two possible outcomes

The full listing of his or her possible decisions and outcomes is shown in Figure 52. Using this example, I will define several terms which I will use to describe different locations in the decision tree. The decision tree starts at the top of the figure, and the designer can choose to run the analysis on any one of the design alternatives (shown as the diamonds). When the designer is making a decision, I will refer to the set of all alternatives and information states as the *alternative space*. For the designer's first decision, there are three alternatives, each of which has the same information state (the designer has only his prior knowledge, since no other actions have been performed). For a particular alternative, the analysis can

return one of two possibilities, α and β (shown as the decision squares). I will refer to the set of all possible outcomes as the *probability space*. Note that each probability space has a parent alternative space. In this example, there are six different possible outcomes in the probability space. An *information state* is a unique set of information that the designer possesses based on his previous actions and the outcomes of those actions. For example, running an analysis of x_3 can lead to information state α_3 or β_3 depending on the information the analysis returns. Once the designer has this information, he or she chooses a final design alternative (shown as the circles). Since the designer is making a decision of which analysis to run and which design to choose, I will refer to this as a *two-level* decision problem where the level is the maximum number of actions that a designer could sequentially perform.

While Equation 44 and the example in Figure 52 are shown for discrete alternatives and outcomes, the framework is still effective for continuous alternatives and outcomes. In this case, Equation 44 becomes an integral which can be numerically approximated:

$$\mathrm{E}[u(a|s)] = \int_{-\infty}^{\infty} p(s,a,s')\mathrm{E}[u(s')] \tag{46}$$

5.3 Generalizations of Alternatives and Attributes

In Section 3.4, utility was defined only in terms of meeting requirements on the attributes of the design itself. This definition restricted the modeler's choice of alternatives to design variables, since only design variables could be related to requirements. In this section, the set of alternatives will be expanded to encompass the set of all actions that a designer can perform. The important question for the modeler is: for any given action, what information is required to accurately model the designer's preferences for that action?

We can determine the necessary information by examining the equations that lead to a decision, specifically Equations 25 and 45 which have been repeated below for clarity:

$$\mathrm{E}[u(\boldsymbol{x}|s)] = \int_{-\infty}^{\infty} u(\boldsymbol{g})p(\boldsymbol{g}|\boldsymbol{x},s)\mathrm{d}\boldsymbol{g} \tag{25}$$

$$\mathrm{E}[u(s)] = \max_{a}\left(\sum_{s'\in S} P(s,a,s')\mathrm{E}[u(s')]\right) \tag{45}$$

From the equations we can see that the designer can only be influenced in three ways:

- Changing the designer's utility function, $u(g)$.
- Changing the probability of reaching a state, $P(s, a, s')$.
- Changing the designer's beliefs about the relationship between alternatives and attributes, $p(g)$.

Altering the designer's utility function is the most rare influence that an action can exert. The utility function is typically derived from the incentive structure of the organization which is passed down to the engineer from management. For an action to influence the utility function, the action would have to interact with the incentive structure in some way. An example of this would be an engineer requesting more time. If management grants the engineer an extension, the designer's temporal utility function has changed.

Actions can also influence the probability of reaching a new state. Consider an engineer designing an artifact in a large engineering organization. Suppose that the engineer must obtain approval from his manager on his design ideas before moving forward in the design process. We can encode this into the framework by including "approved by manager" as a binary hard requirement in addition to the other performance requirements. In the designer's current state, he can perform the action "submit to manager for approval" and there is a certain probability, $P(s, a_{\text{submit}}, s_{\text{approved}})$, that the designer will move to the "approved" state and a $(1 - P(s, a_{\text{submit}}, s_{\text{approved}}))$ probability that the manager will not approve the design. The designer knows that the manager is much more likely to approve the design if he submits a technical drawing instead of a napkin sketch. Therefore, the action "draft a technical drawing" will increase the probability of moving to approved state. In other words, the drafting action will increase $P(s, a_{\text{submit}}, s_{\text{approved}})$. Therefore, actions can influence the probability of reaching a particular state.

Most often an action will influence the designer's beliefs about the relationship between alternatives and requirements. Mathematically, the action is changing the shape of the probability distribution. Actions will likely influence both the mean and uncertainty of $p(g)$. However, in most situations, the designer will only be able to specify ahead of time the change in uncertainty, not the change in mean. Consider the case of running a high

fidelity model; after the designer runs the model, the designer's uncertainty about the relationship between design alternatives and design attributes will decrease (if the designer trusts the model to be completely accurate, his or her uncertainty will go to zero). It is also likely that the high fidelity model will return results that are different than designer's expectation; this represents a mean shift in the designer's beliefs. However, the designer knows only the change in uncertainty ahead of time; the designer does not know what the high fidelity model will return.

In certain situations, the designer might have an idea of how an action will shift the mean of a distribution. Consider the scenario in which a designer on Team A works in tandem with design Team B. Suppose that Design Team B controls a design variable, z, that influences the performance of Team A's design; high values of z will have positive effects on Team A's design and low values of z could be detrimental to Team A's design. Team A knows from past experience that Team B typically picks low values of z, but does not know with certainty what Team B will choose in this case. If Team A were to meet with Team B and discuss the effects of z on performance, it is more likely that Team B will choose a higher value of z. Therefore, the action "meet with Team B" shifts the mean of Team A's beliefs to higher values of z. Team A may still be uncertain of what Team B will decide, but they now believe it to be more likely that Team B will choose a high value of z.

By viewing actions from an information perspective as outlined above, the modeler can incorporate virtually any action a designer can perform as long as the effect of the action can be expressed in one of these three ways. Analytic models and experimental tests are changes in mean and uncertainty. Requests for time extensions are potential changes to the utility function. Meetings with other engineering teams can result in changes in the mean and uncertainty of a distribution.

5.4 Design Process Influences on Utility

In Chapter 3, utility was specified only as a function of the attributes of the alternatives. In this section, I will refer to that utility function as a *performance utility function*, since it is based on the performance of the artifact. To account for the designer's decisions in a

sequential decision making scenario, the utility function must also depend on the attributes of the *design process* itself. Since time and budget are typically the most powerful influences in any organization, this thesis will focus specifically on these influences.

For designers in an engineering organization, time and resources can have a powerful effect on people's decisions. The specific effect of time on behavior depends largely on the incentive structure and the designer's perception of what constitutes an acceptable pace. In many cases, a designer might be faced with a deadline. In other cases, a designer might not have a specific deadline, and his temporal preferences are based on his own internal pacing schedule. The resulting conclusion is that a person's preferences in regards to time is both very specific to the problem and to the individual. This makes the formulation of a utility function with the inclusion of temporal influences difficult, and, unlike the requirements based utility function in Section 3.4, it is unclear whether the utility function can be summarized in a single specific form. However, the utility of time can still be examined in a general form, and many useful conclusions can be made. For this reason, I will propose several forms of a temporal utility function, two of which will be tested in Chapter 6.

5.4.1 Generalized Form

We will begin the derivation of a time-based utility function by defining a dimensionless variable, τ, which represents the fraction of time available to the designer to perform actions:

$$\tau = 1 - \frac{t}{t_0} \tag{47}$$

where t_0 is any arbitrary unit of time. If the designer is faced with a deadline, t_0 is the amount of time from the beginning of the decision making process to the deadline. In this manner, τ is equal to 1 at the beginning of the decision-making process, 0 at the deadline, and can theoretically range to $-\infty$. Note that τ is *decreasing* with increasing time. Similarly, we can define a parameter, β which represents the amount of budget remaining.

$$\beta = 1 - \frac{b}{b_0} \tag{48}$$

where b is the amount of cost incurred at a given time, and b_0 is the allotted budget given to the designer. As with g, I will commonly refer to τ and β in percentages. For most of this discussion, I will examine utility only as a function of τ. However, all the claims I make about τ will be equally valid for β.

Recall that utility is only calculated on the last transition when the designer chooses action a_{rd}. This implies that the utility function is valid only after the designer is locked into a final design. After reaching state s_{reward}, the designer is unable to perform any action that might change the design. While this might appear to be a subtlety of the framework, this fact greatly simplifies the formation of a utility function; we need only consider the effects of time, budget, and design performance at the end of a design, not at any intermediate time.

An alternative approach would be to define an inter-temporal utility function, one that is based on the designer's current state. At each action, the expected utility would be calculated based on a new utility function for that specific time. This complicates the form of the utility function, since the assumptions of utility independence are easily violated. Consider the following example shown in Figure 53; suppose a designer is deciding between Designs A and B in Scenario 1. Suppose that, at the deadline, the risk-averse designer is indifferent between the designs. Now consider the scenario in which the designer has a significant amount of time remaining. The designer might not be indifferent between these two designs, since the available time might allow the designer to further analyze Design B. In this scenario an assumption of utility independence between g and τ would not hold. However, if each of these decisions was a *final decision* and the designer could take no action that would provide further information about the two designs, utility independence would be a reasonable assumption since the designer cannot utilize the remaining time to improve the design. In other words, the designer's risk behavior regarding performance would be the same regardless of the time remaining, since any remaining time cannot be used once the designer returns his or her design. Therefore, the utility function is a reasonable assumption for the final transition into s_{reward}.

The above lottery illustrates that performance can be utility independent of time. To use

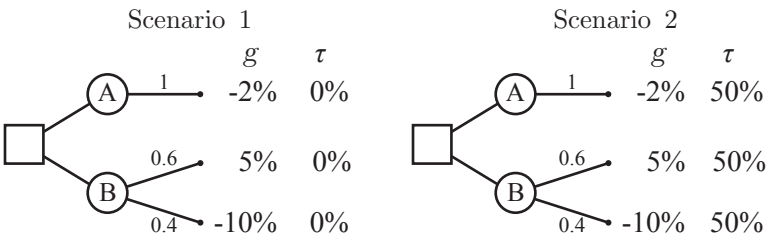


Figure 53: Utility independence of time and design performance is only valid if utility is calculated at the designer's final decision.

the multilinear utility function, however, *mutual* utility independence must be shown. In other words, time must be shown to be utility independent of performance. This assumption is more difficult to justify; imagine a designer faced with a decision between two actions, each lasting an uncertain amount of time. If time and performance are utility independent, then the designer's choice between actions should be the same regardless of the performance of the design. The designer's risk with respect to time may not always be independent of performance. One could imagine that a particular designer might be more willing to miss a deadline if he or she knows that the performance of the design is outstanding, and less likely if the performance is mediocre. However, I believe that this affect of performance on time is generally small, and that utility independence is a close approximation to a designer's behavior. This effect is probably most pronounced at the extremes, i.e. when the designer is missing multiple important requirements. For more realistic problems where most requirements are met, the designer's preference for time is likely very similar for a nominal range of performance.

If time and design performance are assumed to be mutually utility independent, we can examine the utility of τ in isolation from g. I will define the utility of time to be the function u_τ. Since a designer prefers to finish sooner than later (for the same design performance, g), the functional form of u_τ is restricted to monotonically increasing functions. Like the performance utility function, we can arbitrarily define the temporal utility function to be bounded between 0 and 1. As such, $u_\tau(1) = 1$ and $u_\tau(-\infty) = 0$. Unfortunately, little more can be said about the nature of u_τ without the context of a specific problem. For scenarios involving deadlines, it is likely that u_τ drops significantly across $\tau = 0$; however, this may

not be the case for all situations. Section 5.4.2 will explore possible functional forms of u_τ.

As previously mentioned, τ and g are assumed to be utility independent. Therefore, in its most general form, the multi-attribute utility function is of a multilinear form:

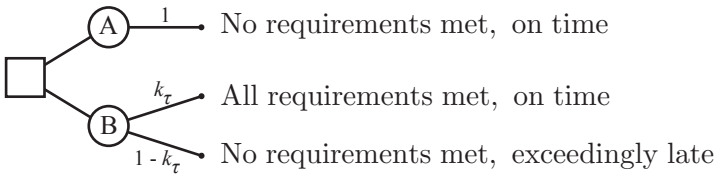
$$u_{\text{total}}(\boldsymbol{g}, \tau) = k_\tau u_\tau(\tau) + k_g u_g(\boldsymbol{g}) + k_{\tau g} u_\tau(\tau) u_g(\boldsymbol{g}) \tag{49}$$

Equation 49 has three unknown parameters: k_τ, k_g, and $k_{\tau g}$. However, since we can arbitrarily define u_{total} to range from zero to one, only two of these parameters are independent.

Suppose a designer returned a design decision instantaneously, but none of the requirements were met. In other words, $u_g \approx 0$. In this case, u_{total} simplifies to

$$u_{\text{total}}(\boldsymbol{g} \ll \mathbf{0}, \tau \approx 1) \approx k_\tau \tag{50}$$

which is equivalent to the following lottery:

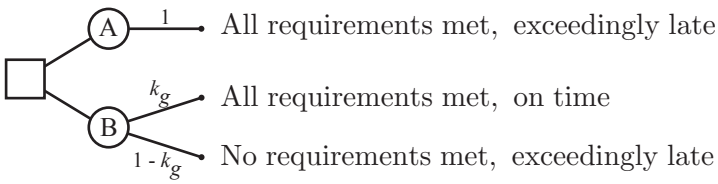


From Equation 50 and the lottery above, it is clear that $k_\tau \approx 0$; the designer would obtain no benefit from instantaneously picking a design of no value.

Similarly, we can consider the case when all requirements are exceeded, but the designer has drastically exceeded their allotted time. Equation 49 simplifies to

$$u_{\text{total}}(\boldsymbol{g} \gg 0, \tau \ll 0) \approx k_g \tag{51}$$

which is equivalent to the following lottery:



Once again, $k_g \approx 0$. The design is nearly useless if the designer has to wait forever to obtain it. The elimination of these terms leaves only the term containing both u_g and u_τ.

Since the utility function is defined between zero and one, $k_{\tau g} = 1$. Therefore, Equation 49 simplifies to:

$$u_{\text{total}} = u_\tau u_g \tag{52}$$

5.4.2 Example Utility Models of Time

As mentioned in the beginning of this section, the functional form of the utility function is highly dependent on the individual designer and design situation. Because of this, the class of temporal utility functions is likely very large, and exploring all possible functional forms in detail is beyond the scope of this thesis. In this subsection, I will propose functional forms of two utility functions that are simple to implement and have easily interpretable meanings. Additional forms will be explored, but are not implemented in this thesis.

Suppose a designer has been given a strict deadline; meeting the deadline is critical to the designer's preferences and a design returned after the deadline has zero utility. In the limit in which the deadline is the only influence on the designer's temporal preferences (and not his desire to finish early), the utility function for time becomes.

$$u_\tau = H(\tau) \tag{53}$$

where H is the Heaviside step function. Note that this utility function implies that the designer is indifferent to finishing early. In this situation, we can expect the designer to always finish at or very near the deadline[2]; since the actions available to the designer can only improve his or her final design decision, we can expect him or her to always choose to perform another action as long as it does not exceed the deadline.

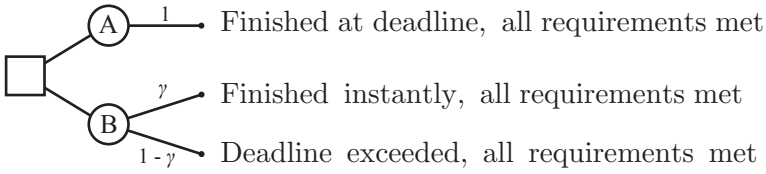
In reality, most designers would prefer to finish the design earlier if finishing early is not substantively detrimental to the performance of the design. If performing an action is expected to yield very small amounts of improvement, the designer may decide not to perform them. In this case, a designer's preferences could be represented by the following

[2]Note that this statement assumes the designer has perfect knowledge of how long a task will take. If there is uncertainty in task length (which, in a realistic scenario, there usually is), then the designer may finish early to avoid missing a deadline. Even with the Heaviside utility function, it is still possible for a designer to rationally make decisions that lead to a missed deadline. It is likely that most deadlines are missed due to the inability to perfectly account for task length, not because the designer is indifferent to missing a deadline.

equation:

$$u_\tau(\tau) = H(\tau)\gamma^{(1-\tau)} \tag{54}$$

where γ represents the temporal utility at the time of the deadline. This utility function is displayed in Figure 54 with γ set to 0.9 (note that the x-axis has been reversed to match the passing of time). The value of γ is equivalent to the following lottery:



Similar to k_h, the value γ is likely near one for most designers. Note the similarities between this utility function and the *satisficing* paradigm; recall that when a designer satisficies he or she is more interested in a satisfactory solution than the optimal solution. In the paradigm of this framework, this behavior arises from the passing of time (τ) and expense of resources (β) outweighing the expected improvement from continuing the design search. Given the formulation of the performance utility function outlined in Section 3.4, the designer knows that most of the utility will be achieved by just meeting all the requirements (a satisfactory answer). With a discounted temporal utility function such as Equation 54, the designer will only continue if the cost of the actions is smaller than the expected improvement in design. Both Equations 53 and 54 are explored in further detail in the demonstrations in Section 6.3. Discussion of their applicability to modeling human designers is deferred to that section.

For human designers in most design situations, the temporal utility function is likely more nuanced and detailed. In many situations, exceeding a deadline by a second, few hours, or even a few days may not be detrimental to the performance of the artifact. In this case, the deadline is not represented by the discontinuity of the Heaviside function. Instead, the slope is likely more gradual with the precise representation specific to the situation and the risk characteristics of the designer. The solid line in Figure 55 represents a notional example of this utility function. Note that the utility before the deadline is discounted, representing the designer's preference for a satisficing solution. At the deadline, the curve

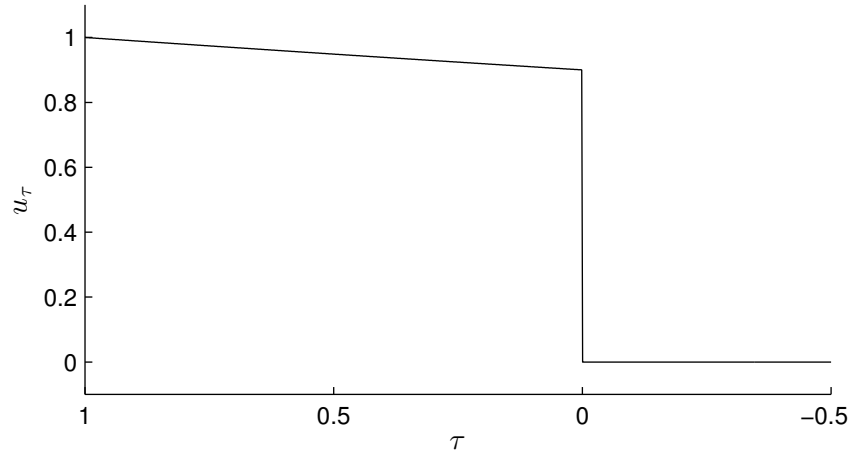


Figure 54: Time utility function for strict deadline with discounting

has an initially gradual but then steep slope. This represents the designer's tolerance for exceeding the deadline by small amounts but not large amounts. This utility function was created by combining the discounting function with a logistic function:

$$u_\tau(\tau) = \frac{\gamma^{(1-\tau)}}{\exp\left(\frac{\tau^* - \tau}{\beta}\right) + 1} \tag{55}$$

where γ, τ^*, and β are parameters to personalize the function to a specific designer.

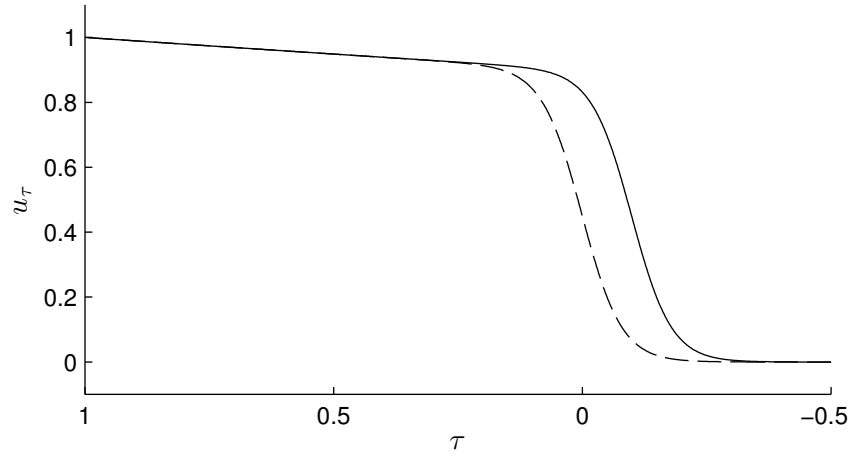


Figure 55: Time utility function for strict deadline with discounting

One could also imagine a very conservative designer who highly prefers finishing early and tends to avoid waiting until the deadline. Perhaps this designer's utility function is

represented by the dashed curve in Figure 55. Similar to the solid curve, this curve's steep descent begins before the deadline occurs. Mathematically, this is the same function as Equation 55 with a larger value of τ^*.

In practice, there may even be times when the designer's utility function is not aligned with the interests of the organization. In many organizations, an engineer's time is billed to a certain project. If the designer finishes his or her work early, he or she may not have a project to bill his or her hours. This might incentivize the designer to extend the work up until the deadline. Perhaps the designer's utility function is shaped like the curve in Figure 56. Note that this function violates the monotonically increasing assumption made in Section 5.4, so the lotteries would need to be changed in order to derive the same general form of the utility function (the conclusion, however, is identical in this case).

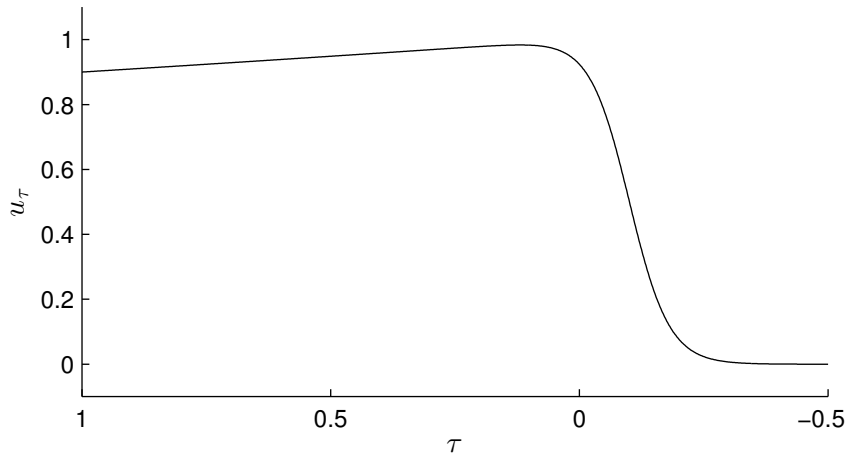


Figure 56: Time utility function misaligned with the interests of the organization

For a given design problem, a modeler can obtain a temporal utility function for a specific designer using lottery questions in the same way that one would create the performance utility function in Section 3.4. Once again, the accuracy of the utility functions is dependent on the human's ability to specify their preferences using lotteries and maintain consistency with these preferences in an actual design scenario. In practice, this can be very difficult for a human, and empirical evidence shows that humans are inconsistent as Section 3.4.1 mentioned. However, one can view these utility functions as an approximation of human

behavior. If we make a simplifying assumption that the designer will always finish within the deadline, then Equation 54 may be sufficient to model the human's behavior. If the modeler is more interested in the conditions that would cause a designer to miss a deadline, then Equation 55 is more appropriate. In general, the level of complexity of the utility function should be consistent with the detail of the behavior being modeled.

5.4.3 Other Influences on Utility and Alternatives

Fundamentally, the designer's utility function is driven by his or her incentives. In this thesis, the incentives examined were performance, time, and budget. While these are likely the strongest and most common influences on a designer's decision, other influences may exist.

One potentially strong influence may be a designer's desire to justify a decision. Since an engineer is held accountable for their decision, the engineer may see utility in having information to justify the decision. This could be represented by a utility for reduced uncertainty; even if a designer knows which uncertain alternative is the "best" from a performance standpoint, he or she may still desire to reduce the uncertainty in order to justify the decision to others. This could lead to the appearance of a confirmation bias; the designer performs the most analysis on the alternative he or she believes to be the best to raise total utility while disregarding other designs. Note that, given the shape of the requirements-based utility function, the designer is already incentivized to reduce uncertainty (he or she is risk-averse). The addition of justification would be an additional incentive for the designer.

In this thesis, designers are analyzed in isolation. In an engineering organization, however, designers must also interact with other people. This will likely introduced additional parameters into the utility function that influence communication. For example, another design team might request the designer's assistance; the designer may not see any benefit (from a design performance standpoint) of assisting the other team. However, from a social perspective, the designer may still help the other team in order to remain on good terms with them. These social influences could potentially have large effects on the designer's

ultimate decisions.

Finally, the sequential decision making framework was formulated under the premise that the designer always selects a final design. In reality, the designer always has the unspoken option of choosing none of the alternatives. The designer could inform management that the design is simply not feasible, or the designer could iterate, generating new alternatives and repeating the design process. The designer's decision depends ultimately on his or her beliefs about the success of pursing this final option. In its most simple form, this alternative could be represented by a utility threshold (recall that utility is always defined between 0 and 1 in this document). If no alternative meets a certain level of utility, then the designer does not choose any of them.

CHAPTER VI

IMPLEMENTATION AND DEMONSTRATIONS

This purpose of this chapter is to summarize the entire framework and demonstrate how it can be implemented. Section 6.1 will provide a clear methodology for implementing both the sequential decision technique in Chapter 5 and the expected utility calculations in Chapter 3. Section 6.2 will address several computational issues that can arise when implementing a sequential decision making problem. Several techniques will be recommended for mitigating these problems. The remaining two sections will demonstrate entire framework on two problems: first, the single dimension problem in Section 4.1 is revisited to show the effect of the framework on the designer's decision. Next, a designer is given a set of actions to choose from in a variable fidelity analysis problem.

6.1 Implementation

Since a description of the framework is developed incrementally in Chapters 3 and 5, this section is intended to prescribe a concise methodology for implementation. Regardless of whether or not sequential decision making is used, all steps under "Expected Utility Calculation" should be performed.

6.1.1 Expected Utility Calculation

Define the Problem

1. Define all design alternatives and requirements.
2. If desired, create a reality model.

Create a Utility Function

3. For each requirement, determine appropriate values of b and g^* based on the beliefs of the designer.
4. For each requirement, determine whether the designer views the requirements as hard or soft. If soft, use lotteries to determine a multilinear utility function. If the set of

requirements contains both hard and soft, use the lottery on page 64 to determine a value of k_h.

5. Construct a temporal utility function based on the designer's beliefs about the consequences of going past a deadline. If the modeler can assume that the designer will never exceed the deadline, Equation 54 might be an adequate utility function. If the designer would exceed a deadline to improve the design, a utility function similar to Equation 55 would be more appropriate.

Create a Gaussian Process Model

6. Determine the designer's prior knowledge regarding the relationship between alternatives and attributes. Represent this knowledge using a probability distribution or a Gaussian Process model using noisy training points to represent uncertainty. Refer to Section 3.3.3 for more information on constructing the designer's beliefs.
7. If Gaussian process models are used, select an appropriate covariance function based on the designers beliefs about the underlying relationship between alternatives and attributes. Determine values for the hyperparameters by maximizing marginal likelihood.

6.1.2 Sequential Decision Making

If the designer is making more than one decision, complete the following steps. If the designer is making only a final decision, perform only step 11.

8. Define all possible actions that a designer can perform.
9. Determine how each action will influence the designer's knowledge. Recall from Section 5.3 that an action can only affect the utility function, probability of reaching a state, and/or beliefs about the attributes of alternatives.
10. Given the set of actions, build a decision tree to determine the information states that the designer can reach. If continuous alternatives or outcomes are used, then the alternative space and probability space need to be discretized; determine the fineness of sampling for the alternative space and the probability space.
11. At s_{reward} of each branch of the decision tree, use Equation 28 on page 68 to calculate

the expected utility of each alternative.

12. Working backward through the decision tree, use Equation 45 on page 98 in each probability space to calculate the expected utility in the parent alternative space. In each alternative space, select the maximum expected utility to feed into the parent probability space.
13. Repeat step 12 until you reach the beginning of the decision tree (the point furthest from s_{reward}. At this level, an expected utility is calculated each action. The designer chooses the action with the maximum expected utility.
14. If a reality model exists, use the outcome from the reality model to determine the designer's next decisions.

6.2 Computational Issues

The greatest challenge of the sequential decision making technique outlined in the previous chapter is the computational effort required. A good estimate of the length of time to perform the calculation is to find the number of expected utility calculations; in practice, it was found that the majority of the computational effort was in numerically integrating Equation 28. Consider again the decision tree shown in Figure 52. For three alternatives each with two possible outcomes, we are required to perform eighteen expected utility calculations (three alternatives and six information states).[1] Were we to add the possibility of running a second analysis before making a decision to the previous example, the number of expected utility calculations increases by a factor of four (assuming the designer does not choose to run an analysis on the same alternative twice). A three level problem with four alternatives and three outcomes requires 432 expected utility calculations. From this example, it is evident that adding levels, alternatives, and outcomes can quickly increase the amount of expected utility by orders of magnitude.

In a continuous space, this is problem is even more pronounced. The inclusion of the *max* operator in Equation 45 precludes the possibility of finding an analytical solution for

[1] One might recognize that some of the expected utility calculations are the same and need not be repeated. In this case however, it is possible that all states are unique. For example, if running an analysis on x_1 provides the designer with some tangential information regarding x_2 and x_3, then all the calculations in the final alternative space might be unique.

all but trivial problems. In practice, Equation 46 is solved numerically by dividing the alternatives and outcomes into discrete possibilities. For realistic problems, this can require enormous computational effort if a fine mesh of the alternative and probability space is used. In general, computation increases in polynomial time with number of alternatives and possibilities and exponential time with number of levels. Equation 56 provides an estimate of the number of expected utility calculations required for a given problem with continuous design variables and outcomes:

$$\text{Expected Utility Calculations} = x^{nl} p^{r(l-1)} \tag{56}$$

where x is the discretization of the design space for each design variable, n is the number of design variables, l is the number of decision levels, p is the discretization of the probability space for each requirement, and r is the number of requirements. To give an example of the computational difficulty, a simple design problem with one continuous design variable and two continuous requirements analyzed at three levels with both the alternative and probability space discretized to 50 points requires about 780 billion expected utility calculations. For large values of any variable in Equation 56, the number of calculations can quickly become unmanageable. The goal of this section is to outline several strategies for minimizing the computational effort.

6.2.1 Adaptive Sampling[2]

For continuous probability spaces, finely sampling the entire space of outcomes can be computationally intractable. Luckily, in many scenarios, the variability in expected utility across the probability space is generally very small as the space divides into distinct clusters. Figure 57 shows an example of this phenomena; the x and y axes show the outcome of a running an analysis on a particular design for two attributes. Movement along each axis shows the deviation from the designer's expectation for each attribute in units of standard deviation. In other words, at the point(0,0) the analysis returns values for each attribute that perfectly match the designer's expectation. At (1,-2) the analysis returned a value for

[2]**This section was developed with substantial assistance from Dr. Matt Daskilewicz who recommended and programmed an adaptive sampling algorithm for this research.**

attribute 1 that was one standard deviation above what the designer expected and a value for attribute 2 that was two standard deviations below what the designer expected. Notice the two distinct regions in the probability space; most of the design space has constant utility, and only a small region in the upper right corner has any variability. In order to obtain an accurate measure of the probability space, one only needs to sample the transition.

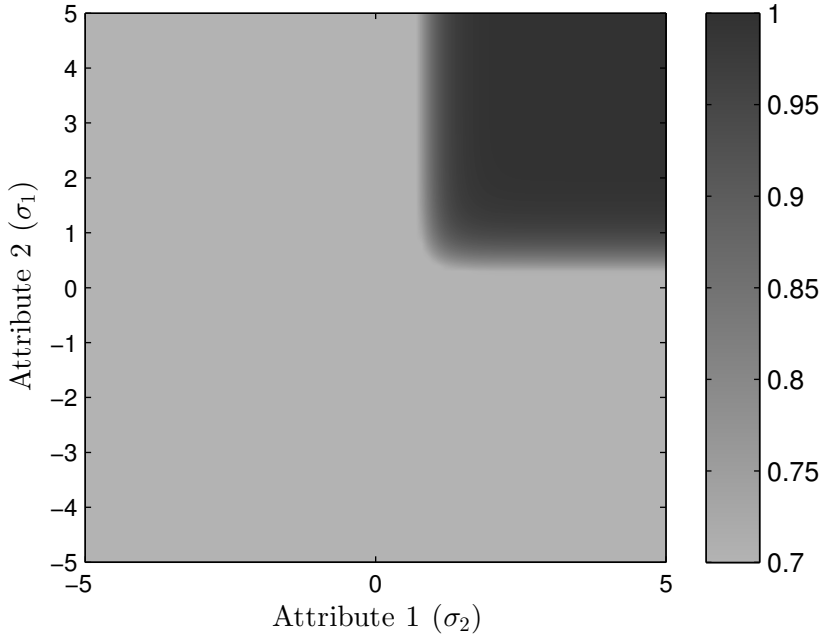


Figure 57: Expected utility variation for a notional probability space with two requirements

This phenomena has two causes: (1) the max operator in Equation 45 and (2) the localized effect of learning. For many actions, the outcome of the action will not affect the maximum expected utility. For example, suppose a designer is analyzing a design at a particular value of a design variable, $x = 1$. If this is far from the optimum value of the design variable, information obtained about the design at $x = 1$ will not have a significant influence on the attribute value at the optimum. Hence, the expected utility remains largely unchanged.

We can take advantage of this feature by only sampling areas of the probability space with variation. In this thesis, a simple adaptive sampling algorithm was implemented as follows:

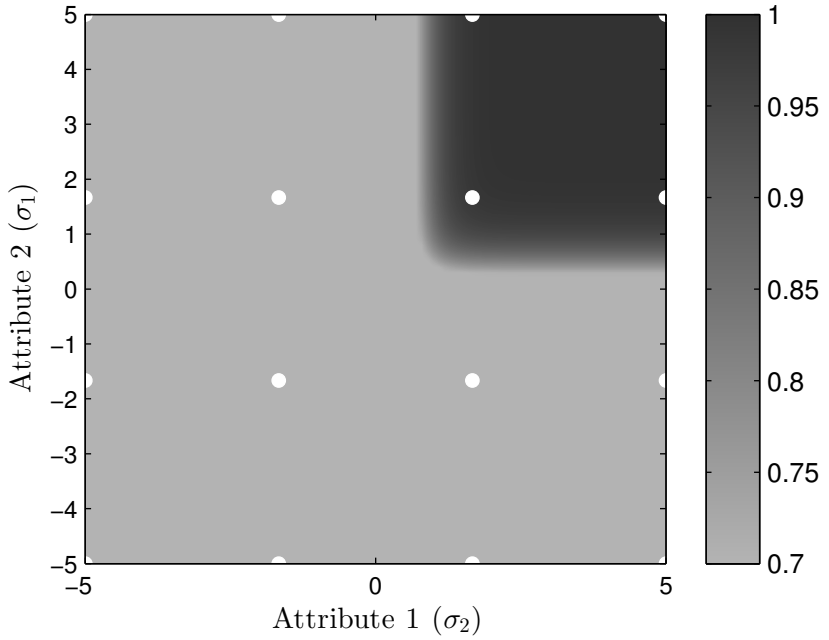


Figure 58: Coarse mesh of probability space

1. Sample the probability space with a course distribution of points, as shown by the white circles in Figure 58.
2. For each square of samples, refine the mesh if any variation exists within the square (i.e., if all vertices are not identical). In this case, the squares in Figure 59 represent a finer sampling where variation occurs.
3. Repeat step 2 for a pre-specified number of refinement levels. The triangles in Figure 59 represent a second level of refinement.

In the example shown, we were able to achieve a sampling equivalent to 13 sample in each dimension (169 total samples) using only 52 samples. Further savings could be achieved by using a more sophisticated adaptive sampling algorithm.

6.2.2 Elimination of Dominated Alternatives

When considering temporal effects, there are many times when stopping criteria can be determined without evaluating the remainder of a decision tree. One can compare the utility of choosing action a_{rd} to another action by comparing the expected utility of a_{rd} to the maximum possible expected utility of choosing another action. Consider the following

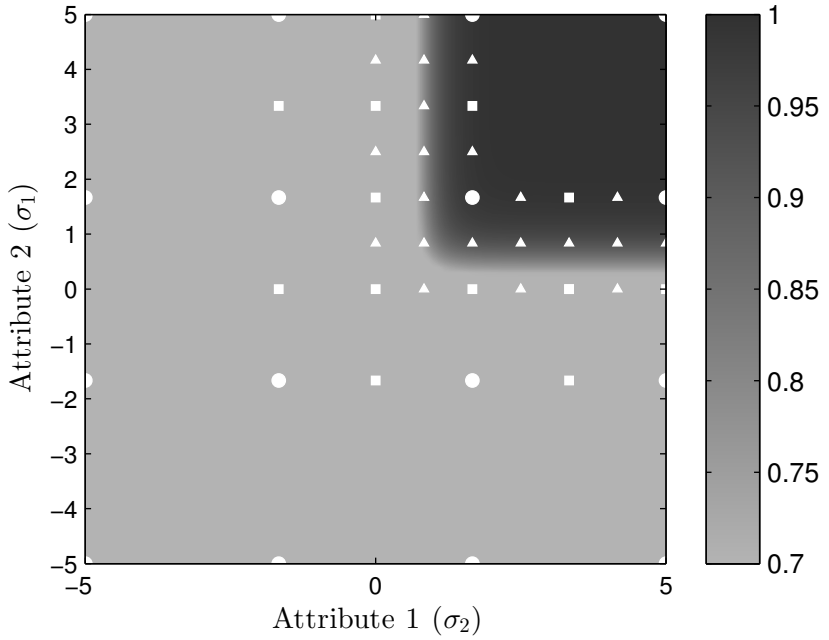


Figure 59: Refined adaptive mesh of probability space

example: A designer is at time step t_1. Suppose that if the designer were to return a decision at t_1, his maximum expected value of u_g would equal 0.9 and u_τ would equal 0.95, resulting in a total expected utility of 0.855. At time step t_2, u_τ would equal 0.8 based on his temporal utility function. Therefore, without calculating u_g for any other action, it is clear that a_{rd} would be the optimal decision since no value of u_g at t_2 could overcome the decrease in u_τ. This observation is especially useful near a deadline, where u_τ tends to decrease rapidly.

6.2.3 Eliminating Duplicate Information State and Expected Utility Calculations

When two actions are performed whose outcomes are not dependent on each other, the probability of reaching a particular information state is the same regardless of the order that the actions were performed. For example, consider a designer faced with two design alternatives, A and B, shown by the decision tree in Figure 60. For the designer's first action, he could choose to (1) analyze Design A, (2) analyze Design B, or (3) return a decision without analyzing a design. If the designer chooses to analyze a design, then he

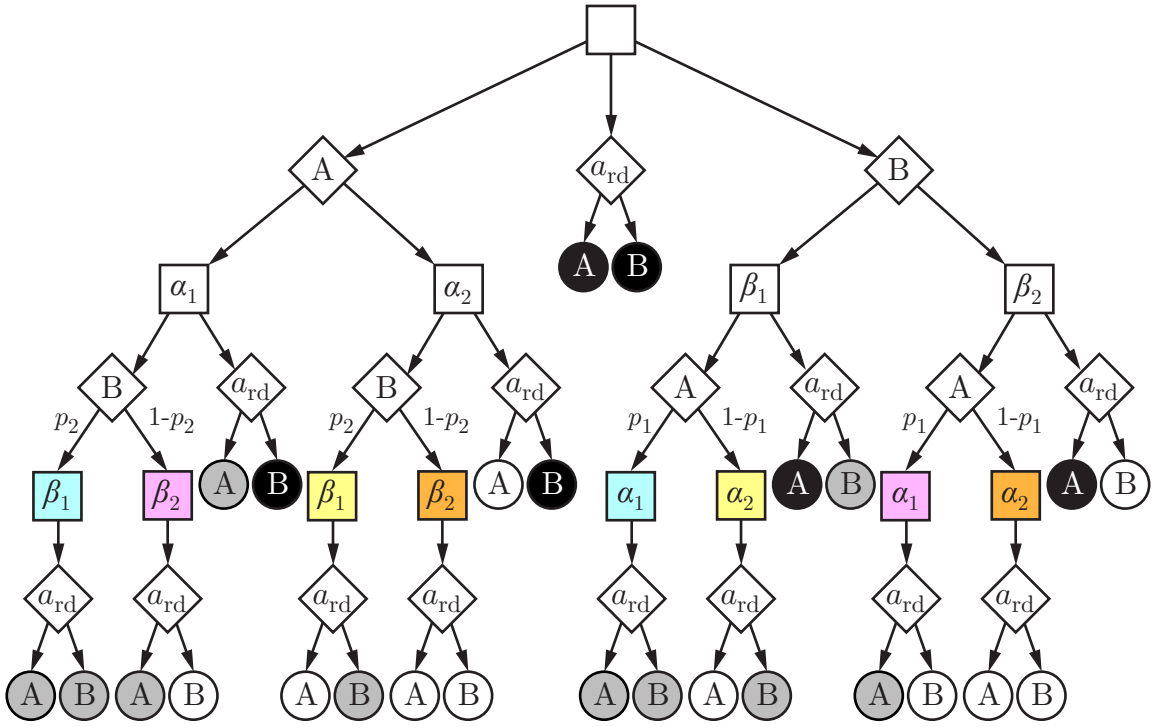


Figure 60: Duplicate information state and expected utility calculations

can (1) analyze the other design or (2) return a decision for his second action. Since the order that designs were analyzed has no bearing on the probabilities of their outcomes, the information states with the same color are identical. The expected utility calculations and the designer's decision in both situations would be identical. In Figure 60, the identical information states have the same color. Therefore, the expected utility at any orange information state is the same; these calculations need not be performed twice.

If the designer does not learn anything about Design A from analyzing Design B and vice versa, then the problem in Figure 60 can be simplified further by eliminating duplicate expected utility calculations. The expected utility calculations (which occur at the circles in the decision tree) have been shaded in the figure to show identical calculations. If every end of the decision tree were calculated, a total of 24 would be required; by eliminating duplicates, only 6 need to be performed.

Note that this simplification cannot always be performed. To borrow the example from Section 5.3, the actions "create technical drawing of design idea" and "submit design idea

to manager" are not independent. Creating the technical drawing and submitting it to the manager has a different probability distribution of outcomes than submitting a napkin sketch to the manager and then creating a technical drawing.

6.3 Investigations of Time

The first experiment was designed to fulfill three goals: (1) demonstrate the methods formulated in Chapters 3 and 5, (2) explore the effects of the temporal utility function on the designer's decision making behavior, and (3) compare the results of the sequential decision making algorithm to the approximation derived in Section 3.5.3 and demonstrated in Section 4.1.

Recall that, in this example, the designer is choosing a value of x such that y is greater than 0.91. The designer has been given a low fidelity model, shown again in Figure 61. The designer can run a high fidelity model at three different locations before making a decision.

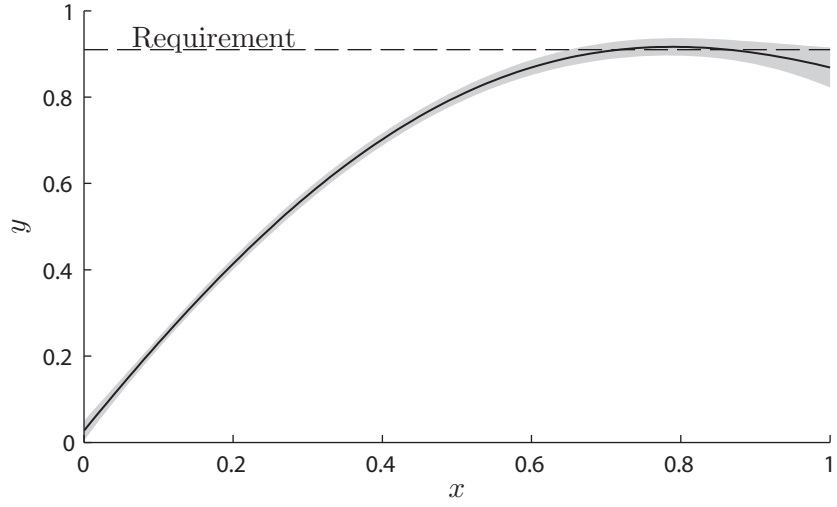


Figure 61: Designer beliefs based on low fidelity model

In this example, we will explore the effect of various temporal utility functions on the designer's behavior. Specifically we will calculate the expected utility of each alternative using Equation 54 with varying values for γ. As in Section 4.1, the designer's performance utility function follows the form of Equation 16 with $g^* = -0.02$ and $b = 0.03$. Since the designer is only concerned with one requirement, the concept of "hard" and "soft" are

meaningless in this example.

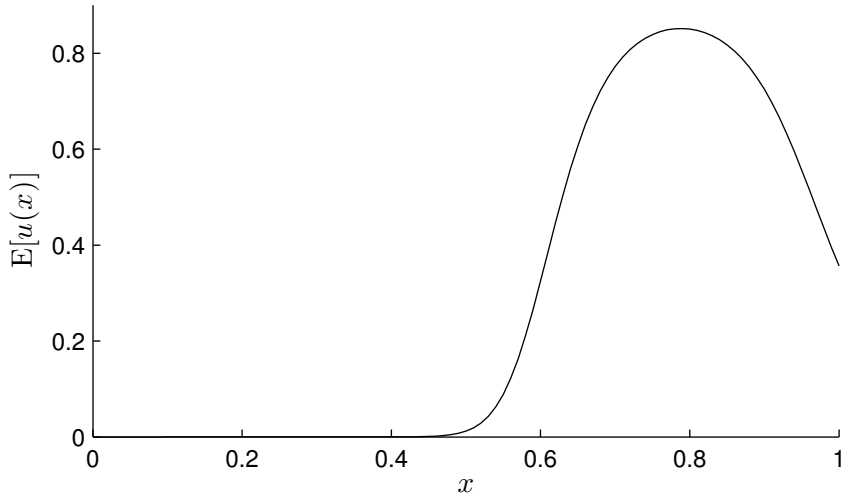


Figure 62: Expected utility of x if the designer was given only one decision

For comparison, Figure 62 shows the expected utility of each alternative if the designer could not run a high fidelity model and had to make a decision based solely on the low fidelity model. This is calculated using only the formulations in Chapter 3 and not including any of the sequential decision making formulations in Chapter 5. The result of this calculation is largely as expected; utility is highest in the region most likely to meet the requirement and near zero everywhere else. Were the designer to choose a design based on the low fidelity model, he would pick the peak of the low fidelity model.

In terms of the sequential decision making, this example problem is a 3-level problem. Since both the design variables and attributes are continuous, both the alternative space and probability space were discretized into 50 points. Through experimenting with different size grids, it was found that this level of accuracy filtered out almost all the numerical noise in the curves. However, accurate results can still be obtained in this problem with half the grid size.

Figure 63 shows the designer's first decision using the sequential decision making construct for various values of γ. The expected utility from Section 4.1 is represented by the dashed line. Perhaps the most striking feature of 63 is that all of the expected utility curves

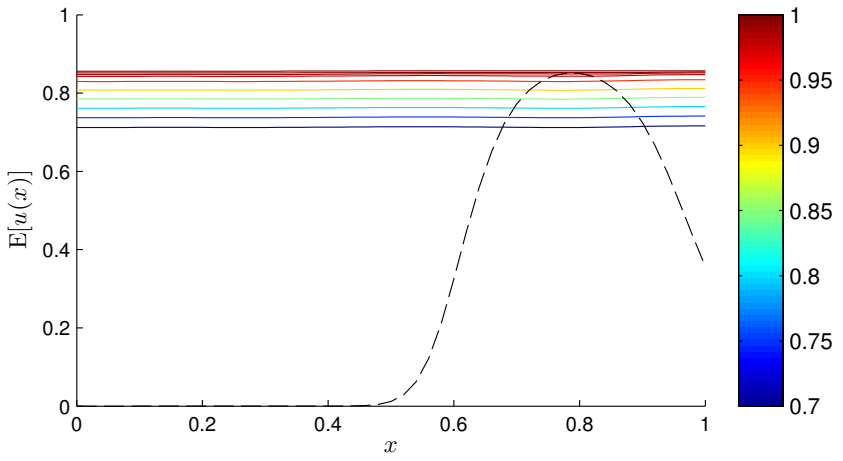


Figure 63: Expected utility of x for different values of γ

appear to be flat. This represents a general indifference by the designer between alternatives; each of them is approximately as good as the others. In Figure 64, we have zoomed in on the top four curves. The dotted line shows the maximum expected utility of the alternatives using the method from Chapter 3. We can interpret the line as a boundary for continuing the design search; if the expected utility of any curve falls below this dashed line, the designer would prefer to make a decision rather than run the model. We can see that the temporal utility function cannot be significantly discounted if the designer is going to run an analysis. This, however, is consistent with the nature of the problem; the designer has relatively narrow uncertainty bands about his beliefs. The low fidelity model indicates that he will achieve high expected utility. Therefore, the designer has little incentive to search. Were his uncertainty greater, he would likely be more inclined to search for a solution.

The designer's preferred value x to run in the analysis has been marked by the circles in Figure 64. Notice that the designer does not pick the optimum of the low fidelity model. Instead, the designer prefers very high x where the designer has the most uncertainty. This is a departure from the example in Chapter 4. In the sequential decision making algorithm, the designer is trying to pick the point that maximizes his information. By choosing a point all the way to the right, the designer can obtain a better sense of the true shape of the curve.

One might notice that the variation of the curves is very small. In fact, the expected utility of the most preferred and the least preferred differ only by about 0.005. The reader might interpret this to mean that the designer thinks that running points at low values of x is very beneficial, but this interpretation is incorrect. Instead, it implies that the designer does not think that running multiple points in general is beneficial. If the designer's prior beliefs are correct (recall from Chapter 4 that they are not), then the designer has little to gain from sampling the design space; his uncertainty is already very small and his utility function is lax. The designer in the next example has much greater uncertainty, and the differences in utility are more pronounced.

Finally, the demonstration shown here offers a glimpse of a way in which the framework can improve. The lowest curve in Figure 64 has a pronounced and curious shape. The optimal design point in the low fidelity model is a local minimum in expected utility with sequential decision making. When testing the sequential decision making algorithm on multiple test problems, this was a common occurrence; the point with the highest expected utility using the framework in Chapter 3 had the lowest expected utility using sequential decision making. Much of this phenomena can be attributed to the learning model; if the designer runs the high fidelity model on the "best" point and the design turns out to be poor, the designer expects the designs to be poor everywhere. However, if the designer

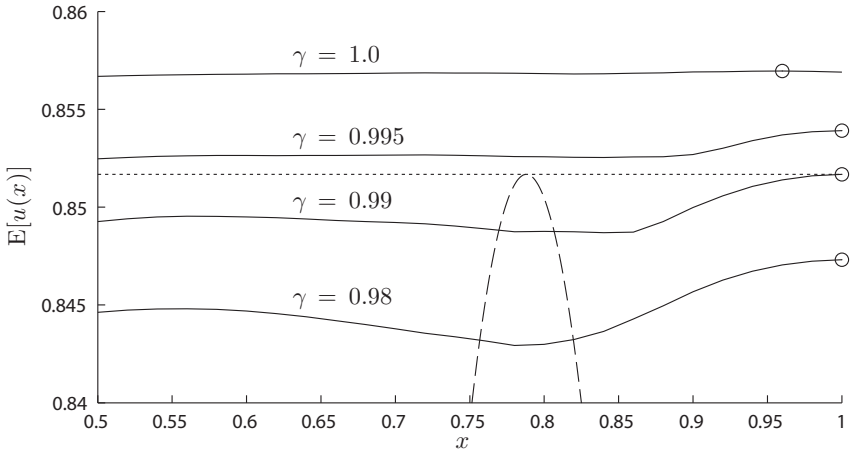


Figure 64: Shape of expected utility curves

runs an analysis to the left or the right of the best point and it turns out to be poor, the designer expects that the "best" point has not changed significantly. The knowledge model fails to capture the changes in beliefs of human beings. This behavior can be attributed to the noise in Equation 4. Even if the designer receives data that proves that his uncertainty estimation is wrong, the noise in the model currently does not update to reflect this. One potential solution is to treat the noise as a hyperparamter [56]. Then, when maximizing marginal likelihood, the noise should update to reflect the designer's increased uncertainty.

6.4 Strategy in Variable Fidelity Analysis

The second demonstration is designed to illustrate the framework's capability to develop a strategy for the designer. By *strategy*, I mean a decision for any given information state. Unlike the previous demonstration, this problem contains no model of "reality;" we are more interested in determining how different realities change the behavior of the designer.

In this design problem, the designer is faced with three alternatives which I will label A, B, and C. The designer is also given two requirements: both y_1 and y_2 must be above 1. We will assume that Requirement 1 is a soft requirement, and Requirement 2 is a hard requirement ($k_h = 0.8$). In other words, the designer should be more concerned with meeting Requirement 2 than Requirement 1. Both requirements are strict ($g^* = 0$, $b = 0.5\%$). The designer has varying levels of prior knowledge about the each of the alternatives; this knowledge is represented as normal probability distributions for each attribute. Table 1 shows the mean and standard deviation of each requirement. From a qualitative perspective, the simulated designer knows the least about Design A. The designer suspects that Design B meets Requirement 1, and that Design C meets Requirement 2. Were the designer to make a decision based on his prior knowledge, he would choose Design C which has an expected utility of 0.650. Note that Design C is the most attractive since it is the most likely to meet the hard requirement. For reference, Design A has an expected utility of 0.433, and Design B has an expected utility of 0.213.

The designer has at his disposal both a low fidelity and a high fidelity model of the relationship between each alternative and the requirements. Running the low fidelity model

Table 1: Designer's prior beliefs regarding the design alternatives

	Requirement 1		Requirement 2		
	$E[y_1]$	$\sigma(y_1)$	$E[y_2]$	$\sigma(y_2)$	$E[u(x)]$
A	0.90	0.30	1.00	0.30	0.433
B	1.10	0.15	0.85	0.20	0.213
C	0.90	0.15	1.15	0.20	0.650

costs the designer $\frac{1}{5}\tau$. The designer can run only one alternative through the low fidelity model at a time, but this model returns an estimate for both requirements. One might imagine that the low fidelity is similar to the aerospace software tool Flight Optimization System (FLOPS) which returns a estimate for many design attributes in short amount of time. We will assume that the designer believes the standard deviation of this model to be 0.1, and that this standard deviation remains constant regardless of the value the model returns. Each run of the high fidelity model costs the designer $\frac{1}{3}\tau$. Unlike the low fidelity model, this model returns a value for only one requirement. This is similar to a CFD or an FEA model which will return only the aerodynamic loads or the structural responses, respectively. The designer trusts the high fidelity model to be perfectly accurate.

Additionally, we will assume that the alternatives are so different that obtaining information about one gives no information about another. In other words, the designer does not learn anything about one alternative based on the properties of another. In regards to the models, we will assume that results from a high fidelity model will not influence the designer's trust in future runs of a low fidelity model. For example, if the high fidelity model returns a value that was significantly different from the low fidelity model, the designer will continue to believe the standard deviation of the low fidelity model to be 0.1. In other words, the designer suspects this inconsistency to be an outlier. Finally, we will assume that the designer has perfect knowledge of how long each analysis will take. It is important to note that these are simplifying assumptions intended to reduce the computational complexity and not limitations to the framework. Each of these details and nuances could be captured in the model. However, as we will see in the demonstration, the framework provides reasonable designer behavior even when these simplifying assumptions are made.

Equation 54 was used to model the designer's preferences for time. It is assumed that

the deadline given to the designer is very strict, implying no utility for $\tau < 0$. A discount factor of $\gamma = 0.9$ was used in Equation 54. Per the assumptions in Section 5.4.1, it is assumed that the time of making a decision is negligible compared to the time to run the models.

6.4.1 Implementation of Demonstration Problem

This demonstration is an example in which some of the strategies outlined in Section 6.2 can be used to minimize the computational load of calculating the expected utility of each alternative. The simplicity comes from the lack of unique information sets. The information state is defined by the actions of the designer and the results of those actions. In this case, however, the order of the actions does not influence the probability of a particular outcome. From an information state perspective, running the low fidelity on model on Design A and then B is identical to running the model on Design B and then A. The information obtained from Design A will not influence the information obtained from Design B. In this way, the overall number of expected utility calculations is relatively low. In addition, some information states are obviously poor choices; for example, if the designer ran a high fidelity model on both requirements for one design, he would not run a low fidelity afterwards; doing so would provide the designer with no new information. Therefore, we can eliminate sets of actions that would obviously return poor expected utility.

In the implementation, the expected utility was calculated for each unique information state. The maximum expected utility of the alternatives from each information state was then stored in memory. The algorithm then expanded the decision tree, found an identical information state in memory, and used the stored expected utility in its calculation. In this way, the most computationally expensive operation, calculating expected utility, is performed the minimum number of times possible.

The temporal utility function also played a role in simplifying the computation. Since the future is discounted, there are many instances in which the simulation can determine that returning a decision will be better than continuing without calculating the expected utility of continuing. In general, however, this effect was small compared to eliminating

duplicate information states.

6.4.2 Results

Figure 65 displays the expected utility of each possible alternative. From the graph, we can see some general trends that would likely be consistent with a human designer's reasoning. The observation that is perhaps the most obvious to the designer is that making a decision without running a model is the least attractive alternative. This is to be expected, since the designer has great uncertainty in his prior knowledge. A designer would find it more beneficial to obtain information at the cost of time, than to make a poor decision instantly.

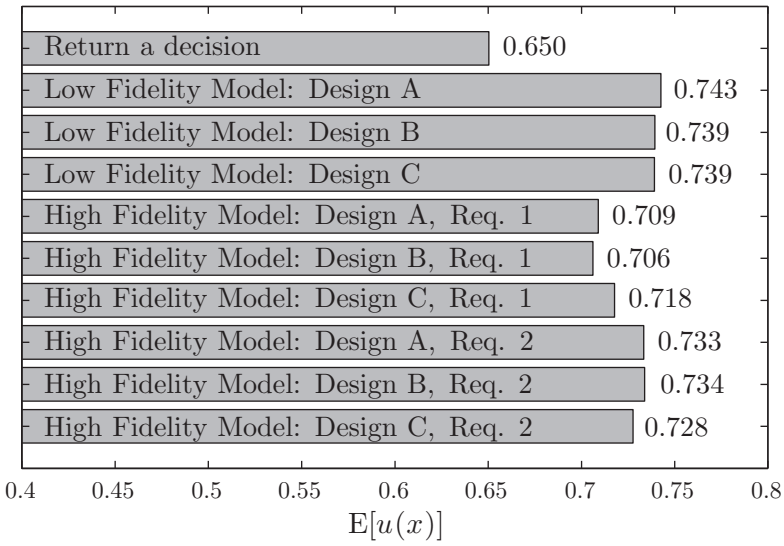


Figure 65: Expected utility for each alternative available to the designer

In general, the designer found running a low fidelity model to be more beneficial than a high fidelity model. In the design process, engineers often follow the same practice of using low fidelity models to learn which alternatives are worth investigating and which alternatives are inadequate. At the same time, running a high fidelity model on y_2 is very close in expected utility to running a low fidelity model. This can be attributed to y_2 being a hard requirement; the designer can obtain almost 80% of his utility be just meeting y_2. However, analyzing Design C has the lowest expected utility of the three alternatives; Design C has a 77% chance of meeting Requirement 2, so the designer does not want to

spend a third of his time confirming something he already believes to be true. As one would expect, running a high fidelity model on y_1 has the lowest utility of all the analyses. Even if a design meets Requirement 1, this information is virtually useless unless the design also meets Requirement 2. Design C is a slight exception to this rule; in this case, the designer already has some confidence that Design C meets Requirement 2, so investigating Requirement 2 first is not a waste of time.

In the end, the designer decides to analyze Design A with the low fidelity model. Of all the design alternatives, the designer has the least information about Design A. Recall that Design A has a 50% chance of meeting Requirement 1, so running a low fidelity model will allow the designer to determine if Design A is on par with Design C and worth pursuing.

Since the algorithm considers all outcomes of a designer's decision in order to calculate the expected utility, we can examine the designer's second decision based on the information received from analyzing Design A with the low fidelity model. Figure 66 displays the designer's second decision as a function of the outcome of the low fidelity model for both y_1 and y_2[3]. If the designer finds that y_2 of Design A is less than 1, the designer further analyzes Design C to confirm that y_2 does in fact meet the requirement; the designer has eliminated Design A as a potential candidate, and focuses his attention on the design most likely to meet the requirements.

If y_2 of design A is greater than 1, the designer's decision depends on the value of y_1. If y_1 does not meet the requirements, the designer chooses to run the low fidelity model on Design C to get an estimate of both attributes. This will allow the designer to compare Design A and C to better compare the designs. If y_1 is greater than 1 and y_2 is slightly greater than 1, the designer will run a high fidelity model on Design A's y_1 to confirm that the design does meet the requirements. If the low fidelity model returns a value of y_1 that is near 1, the designer will run a high fidelity model on that attribute. If however, both attributes exceed the requirements, the designer will simply pick Design A. Since his utility function is discounted, the designer decides to stop early since he has high confidence that

[3]The actual division of decisions does not fall into perfect rectangles. The sampling of this probability space was fairly sparse; therefore, the exact boundaries are difficult to ascertain from the data.

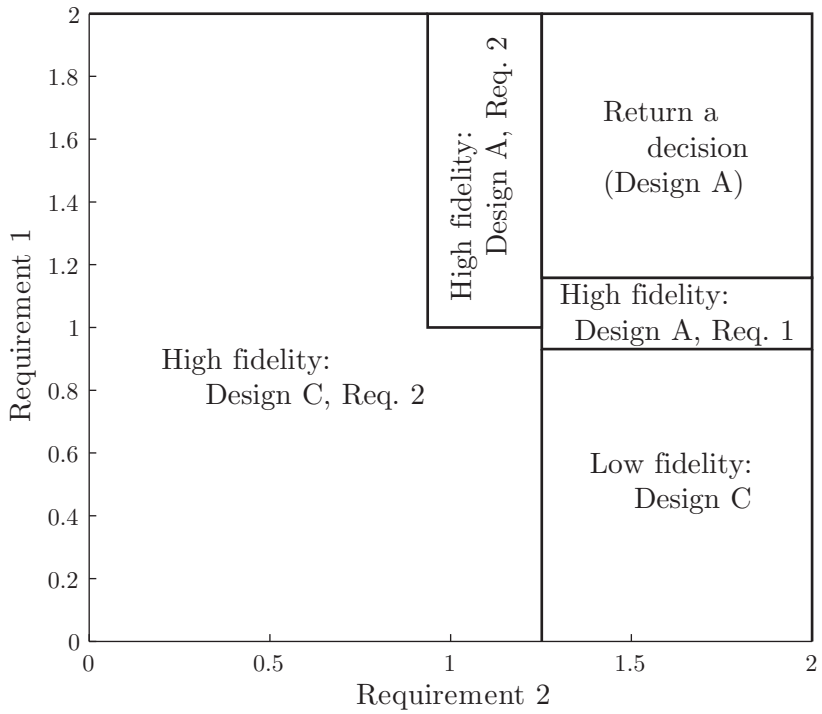


Figure 66: Designer's second decision based on the outcome of the low fidelity model

Design A meets both requirements.

Extending this idea, we can map out the designer's strategy for any possibility set of actions and outcomes independent of a single reality. The independence from a particular reality is a powerful tool of the sequential decision making method; it allows to consider a large set of realities and examine the conditions in which the designer would make poor decisions. For example, were the utility of time function formulated differently, we could examine the conditions under which the designer would go over the deadline. Note that this ability is contingent on two conditions: (1) that the designer considered the reality a possibility and (2) that the probability space is small enough to be stored on the computer. If reality procures an outcome which was outside the designer's realm of plausibility, then the next decision would have to be recalculated with the new information.

In this example, the framework was found to give reasonable results for formulating a strategy in variable fidelity analysis. The designer begins by running a low fidelity model to inexpensively obtain information about a highly unknown alternative. The designer then

uses the information to inform his next decision; running high fidelity models when he need more accurate information, and running low fidelity models when he needs a quick comparison.

At the same time, it is likely that many people would have made a different decision than the simulated designer. Some of this discrepancy can be explained by the parameters used in the utility function (g^*, b, γ). For any particular person, the parameters used in example may be very different than the parameters that best match the individual's preferences. Furthermore the differences in expected utility for the first decision are very small; in fact, the differences are very close to the numerical precision of the model and running the same scenario with less precision in the probability space can return a different decision. In an informal poll of several engineers, I received many different responses of first decisions. While not scientifically rigorous, this poll provides evidence that there are many "good" responses to this problem. The simulated designer would agree with that assessment, acknowledging that he is nearly indifferent to many possible alternatives. For a more thorough discussion on this topic, see Section 7.1.3.

CHAPTER VII

POTENTIAL VALIDATION METHODS

As mentioned in Chapter 1, the goal of this framework is to computationally model the decisions made by human designers. While the framework has been demonstrated on several examples, these results have only been compared to intuition and reasoning; a formal experiment comparing human decision making to the simulated decision making has not been performed. The purpose of this chapter is two-fold: (1) to explore the limitations and weaknesses of the framework in the context of its ability to replicate human decision making and (2) to explore import considerations for designing experiments to validate, invalidate, or improve the model.

7.1 Limitations

This section explores some theoretical and observed limitations of the model. This section has two purposes: (1) to identify conceptual differences between actual human beings and the simulation that could cause differences in design decisions and (2) to indicate areas of the model that could be improved.

7.1.1 Sequential Decision Making

For most sequential decision making problems, the most pressing limitation is the computational time and effort required to calculate expected utility. Although the integral in Equation 28 can be quickly computed with high accuracy, the number of expected utility calculations can easily become overwhelming even for relatively simple problems. This raises a fundamental question about the ability of the framework to model human decisions: If the framework requires such large computational power to simulate a decision that humans make in seconds, how can the framework claim to mimic the designer's decisions?

In reality, humans do not consciously examine every possible outcome to determine which action has the highest expected utility. Humans likely perform many simplifications

to the problem, substantially reducing the computational power required. Listed below are a few simplifications that likely cause differences between actual designers and the simulation:

7.1.1.1 Heuristics

The designer might be able to simplify the sequential decision making problem using heuristics learned through education or experience. For example, designer's usually learn to run low fidelity models early in the design process, and high fidelity models later in the process. From an information theory perspective, the sequential decision making algorithm comes to the same conclusion in most cases. Using simple rules such as these, the designer can easily eliminate several alternatives from the decision tree, substantially reducing the amount of computation required.

7.1.1.2 Depth of Search

Examining Equation 56, the most influential contributions to the computational effort is the number of levels to the decision problem. From a human's perspective, this is representative of how far into the future the human can reason. For certain classes of problems, such as picking specific design points to run in a high fidelity model, examining current decisions in the context of future results can be very difficult. In the high fidelity model example, the designer may disregard the value of information and simply run the points he or she believes to be most likely to return a good design. In Section 6.2.2 it was seen that the discount factor was one way of reducing number of levels that the designer was required to explore; for low values of γ, it could easily be shown that returning a decision had higher expected utility than continuing to search. In this way, the discount factor can represent the ability of a designer to see into the future. A highly experienced designer might have a high discount factor, representing his or her ability to intelligently reason how to extract the most useful information from his actions. A novice designer, on the other hand, may discount heavily, causing every decision to appear as his or her last without any consideration of the future.

7.1.1.3 Memory Recall

It is also possible that expert designers have stored combinations of good actions in their long-term memory and, rather than evaluating every possibility, draw on their past experience to make decisions. This phenomenon is similar to the behavior observed in chess players by de Groot and Simon [14]. In these experiments, chess players of varying skill levels were observed to compare cognitive differences of expert and novice players. Simon, summarizing the work of de Groot, write that

> *de Groot was unable to find any gross differences in the statistics of their thought processes; the number of moves considered, search heuristics, depth of search, and so on. [Chess] masters search through the same number of possibilities as weak players.*

Instead, Simon and de Groot find that chess masters possess a strong ability to recreate chess positions after observing them for only 5 seconds, something weak players were unable to do. However, this ability was only observed when the pieces were placed in "meaningful" positions; when the chess pieces were placed randomly on the board, the chess master was no better at reconstruction than the novice. Simon's hypothesis for what separates a master from a novice is their ability to draw from experience to recognize combinations of good moves: "behind this perceptual analysis, as with all skills, lies an extensive cognitive apparatus amassed through years of constant practice." Perhaps, then, the difference between the computational effort of the human and the machine is the human's ability to use experience to narrow down the best set of actions, whereas the computer examines the problem for the first time with no prior experience.

7.1.1.4 Inter-temporal Utility

One of the assumptions made in Section 5.2 was that the designer is only concerned with the utility of their final decision. In reality, this may not always be true. For human beings, it can be very difficult to consider decisions in regards to the final consequences, especially if those consequences occur at a much later time. Literature suggests that humans use transition points to help pace their work. For example, several authors have found

in experimental studies that teams given a deadline use the midpoint of their time as a transition point [26, 49]. This may be one possible way that humans are able to simplify this computationally intensive problem. By allocating certain tasks to separate blocks of time, the individual can focus on a small subset of the problem and plan accordingly. This seems to imply the creation of an inter-temporal utility function; the designer creates utility functions at certain transition points based on what they would like to accomplish at a certain time. By dividing the sequential decision making into smaller blocks with fewer levels, the amount of computation can be drastically reduced.

Overall, the framework's technique for choosing an alternative in a sequential decision making problem probably does not match the reasoning of human being. However, like the Gaussian process model and utility function in Chapter 3, the sequential decision making algorithm represents an approximation to the designer's reasoning. Recall that the "interface" of the model between the inner environment and outer environment is the designers' decisions and not their reasoning; as long as the simulation produces results similar to a human's decision, the lack of similitude in reasoning may not be significant.

7.1.2 Knowledge Model

In Chapter 3, Gaussian process models were used to model the designer's beliefs. These models have the benefit of being intuitive and computationally simple, they may not accurately reflect a designer's beliefs. One fundamental limitation of a Gaussian process model is the use of normal distributions as a representation of beliefs in all situations. In many cases, the designer's beliefs may best be reflected by another distribution. Consider the designer's beliefs shown in Figure 67. Suppose the designer analyzed a design at $x = 3$ and found that the $f(x)$ was lower than the designer's expected value. How do the designer's beliefs change?

If Gaussian process models are used, then the designer's updated beliefs will appear as Figure 68a; normal distributions with the mean generally shifted lower than the designer's original expectation. The black line shows the designer's original beliefs and the probability distributions have been drawn on the z axis. For some designers, however, Figure 68b may

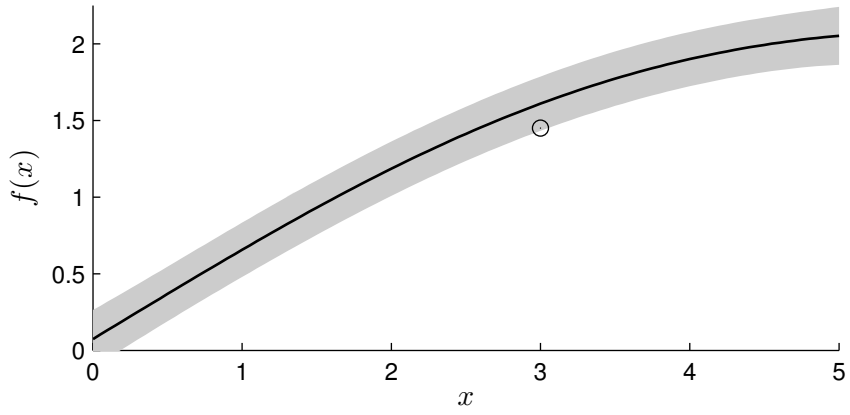


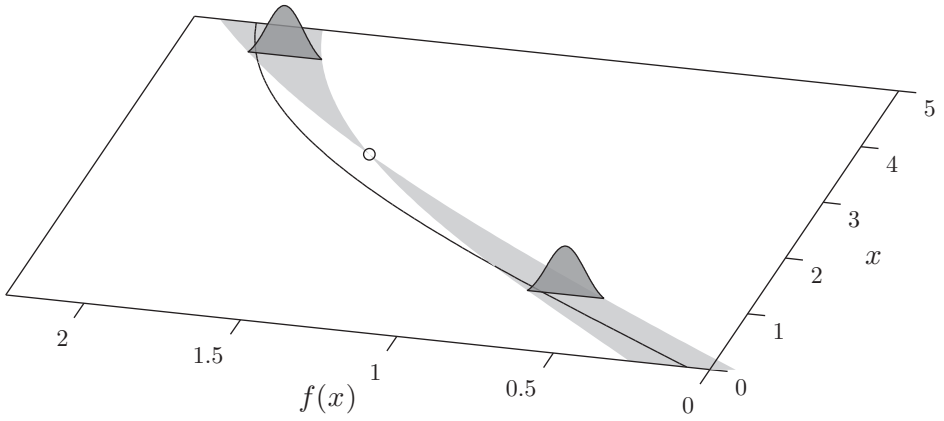
Figure 67: Prior beliefs and new information

be a more accurate reflection of the designer's actual beliefs; the designer may find it very unlikely that the underlying function is above his original expectation. At the same time, he or she may now find it plausible that the underlying function is much lower than expectation. Hence, the probability distributions have all be skewed towards lower values of $f(x)$. While an alternative probability distribution may more accurately represent a human's beliefs, implementing such a distribution is computationally difficult since Equation 3 has an analytic solution only for normal distributions.

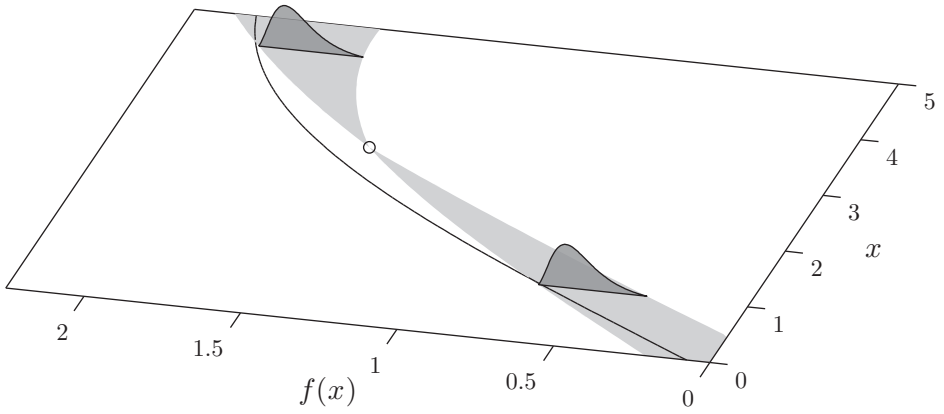
7.1.3 Preference Model

The limitations of utility theory for articulating human preferences have already been identified at the end of Section 3.4.1. As the text indicated, humans are known not to follow the four axioms of utility theory. However, it is possible that improvements can still be made to the implementation of utility theory in the framework. *Prospect theory*, for example, offers several modifications to utility theory that better mimic the behavior of humans [36]. For example, prospect theory has shown that humans often do not consider events that have very small probabilities and treat events with high probability as certain events. To model this, a *decision weight* function is multiplied by the value of each outcome; this weight function essentially modifies the tails of the probability distribution, making highly likely outcomes appear certain and unlikely outcomes appear as impossibilities.

In many of the examples used in this thesis, the utility calculations were carried out



(a) Beliefs represented by normal distributions



(b) Beliefs represented by skewed distributions

to several decimal places. In reality, a utility function for a human cannot be defined to such detail; humans would be unable to specify preferences consistently to that degree. The modeler should keep this in mind when viewing the designer's decisions. If the expected utility between two alternatives is very close, the modeler should expect that actual humans might be indifferent between the two options.

7.2 Usage Applications

At the end of Chapter 1, several characteristics were outlined that the model should possess. The first of these stated that the framework should model decisions that humans actually make, not necessarily the decisions they do make. The intention of this requirement is that the model should be predominately descriptive in nature as opposed prescriptive or

normative. With the details of the framework defined, we can now examine whether or not the framework meets this requirement.

One might notice that the framework contains many normative tools: utility theory is a normative model of decision-making; the sequential decision-making strategy is optimal for a given utility function; even the Gaussian Process models define the probabilities in a mathematically consistent manner. It has already been acknowledged that humans do not follow these algorithms when making decisions; they typically do not create utility functions, and they do not exhaust all potential possibilities. From this perspective, the decision making algorithms appear to be more normative; they represent the optimal decision given a set of beliefs and a utility function.

Two areas where the framework is descriptive lie in the Gaussian process models and the utility function. The simulated designer's Bayesian inferences depend upon the assumed prior distribution. In Section 3.3, a custom covariance function was developed to better represent the prior distribution that a designer might have. One can form an analogy between the prior distribution and the prior knowledge and biases that the designer possesses. By changing this distribution via variation of the hyperparameters (or the function itself), one can change the designer's knowledge and learning. In this manner, the designer's knowledge can be seen as descriptive; it may be possible to fit a function and hyperparameters to a human being in order to model his or her learning.

Similarly, the designer's utility function is adaptable to a specific human being. In this thesis, a utility function was derived based on a requirements-driven paradigm. If the preferences and risk behavior of the designer's utility function are aligned with that of the organization, then this representation is normative; the designer will make decisions that the organization desires. However, researchers have shown that representations similar to the one presented in this thesis can be misaligned with the risk preferences of the organization [16, 1]. This representation better aligns with the descriptive category; theoretically we could design a utility function that closely models the designer's actual preferences, whether aligned or misaligned with the organization. Like the Gaussian process models, the utility function in Equation 16 contains parameters that can be fit to a specific designer.

Based on the flexibility of the Gaussian processes and the utility functions, the decision making framework is perhaps best described as a continuum between normative and descriptive models; by choosing effective prior distributions and appropriate utility functions, the framework can provide the optimal decisions from the organization's perspective. At the same time, these models can be customized to represent a specific designer whose interests may not align with that of the organization's.

For the models to be truly descriptive, an element of the framework is likely missing: a search procedure. Designer's do not exhaustively search all possibilities, but likely use a simpler method of deciding which action to perform. This method may involve heuristics or a simpler a more limited search of the alternative space. The sequential decision-making technique presented here is only one option for making decisions. In my opinion, it is a robust method for representing reasonable behavior; at the same time, the computational costs of this method are far too high. For the model to reach an operational level suitable for simulation, more research should be invested in better understanding how a designer chooses between actions.

7.3 Experimental Validation

This section concerns human subjects experiments as a method of validating the framework and associated models. The goal of this section is not to design the details of an experimental plan, but rather to explore important considerations for creating an experiment and to suggest strategies for investigating the validity of the model. This section begins by exploring the underlying research question of the experiment. Section 7.3.1 identifies potential problems to avoid when creating an experiment. Finally, Section 7.3.2 suggests experiments for validation on a conceptual level.

Perhaps the most important question to ask before designing the experiment is "what is the experiment intending to test?" When validating this framework, the experimenter has at least these two options:

- Validate each of the models individually (i.e., the knowledge model, preference model, and sequential decision making algorithm)

- Validate the simulated designer's decisions as a whole

In my opinion, the main focus of the test should be on the latter and not necessarily test for similitude in the individual model. The former test relies on the assumption that an accurate knowledge model and preference model would lead to an accurate prediction in the designer's experiment. This would imply that designers use their beliefs and preferences and a mathematically consistent way, which may not be true. This question of scope hearkens back to the discussion in the motivation of the inner environment, outer environment, and interface. If the ultimate goal of the overarching simulation is to explore how the design problem, designer decisions, and organization dynamics interact to affect organizational performance and system value, then the most important aspect of the model is the final decision as this decision is the *interface* of this model. The framework is effective if it can return similar decisions as a human designer given the same prior knowledge, alternatives, and incentive structure, regardless of whether it is effective in modeling the specific beliefs and reasoning process of the designer.

At the same time, investigating how the designer views uncertainty and preferences could give insight into how to improve these models to make them more realistic. Data that allows for more accurate modeling of preference and knowledge would be incredibly useful. Instead of asking the question "can a utility function accurately model a designer's preferences under uncertainty?" perhaps a better question is "what utility function form best approximates a designer preferences?" Testing individual models can also give insight as to why the model might fail to capture human decision making. Experimenting with individual models would provide a degree of traceability and guide the modeler on which models need to be improved.

7.3.1 Potential Pitfalls

Throughout the development of the framework, the models and ideas were tested through informal polls and interviews with engineering graduate students. Although the results are not rigorously scientific, some of the observations can help inform the development of a more rigorous human subjects experiment. A major challenge in developing an experiment

is creating similar conditions in both the human subject and the computer simulation. This can be a more challenging problem than it initially appears; designers typically do not think in terms of uncertainty and utility nor do they describe their actions from an information perspective. Therefore, care must be taken when creating experiments involving human subjects as it is difficult to create the same initial conditions in the problem. Specifically, the experimenter should avoid the following common pitfalls that were found through the informal experiments I conducted.

7.3.1.1 Controlling Knowledge

As Equation 25 demonstrated, the designer's beliefs are contingent on his utility function and beliefs about the alternatives. Therefore, in order to create an experiment under the same conditions as a simulation, it is important that the designer and simulation begin the problem with the same knowledge as the simulation.

This can be accomplished in two different manners:

1. Eliciting the designer's prior knowledge about the design problem for use in the simulation: the human subject can be asked about their knowledge prior to performing the experiment; this knowledge can then be programmed into the simulation's Gaussian process model.
2. Experimenting with an unfamiliar design problem: if a design problem is chosen such that the designer has no prior knowledge, then the experimenter can easily control the information that the human subject has at the beginning of the design process.

The former option relies on the human's ability to specify their knowledge and uncertainty. This can be challenging, as designers may not think of their beliefs in terms of probability distributions. It is also contingent upon the ability to transfer that knowledge into the simulation. The latter option is likely the most effective. By using an unfamiliar problem, the modeler can accurately control both the designer's knowledge and the simulation's knowledge.

7.3.1.2 Unintended Uncertainty

In addition to controlling the designer's prior knowledge about a design, the experimenter should use caution when generating this prior knowledge. Practicing designers do not tend to view their beliefs as probability distributions. If a designer is given probability distributions as information to use in the experiment, then his or her beliefs can actually differ significantly from these probability distributions.

Before receiving the results of the computer simulation of the "Strategy in Variable Fidelity Analysis" example problem, several engineering students were given the same information in the problem statement and asked to make a decision. The decisions largely fell into two separate groups; while many chose to run the low fidelity on Design A, several people decided to analyze Requirement 2 of Design C with the high fidelity model, an alternative the simulated designer finds to be relatively poor (his 6^{th} preferred option). In my opinion, the discrepancy lies in the manner in which I facilitated the human's beliefs. When I inquired as to the reasoning behind the designer's decision, one subject responded that he would would run the high fidelity model on Design C which would "*tell me how correct my own beliefs were*." Already the designer has uncertainty is his beliefs given the probability distribution; however, the designer's comments imply an extra layer of beliefs which I refer to as "meta-beliefs." These meta-beliefs are beliefs about the prior knowledge and an acknowledgment that the designer does not trust the information given in the problem statement. These beliefs are problematic, since the designer's knowledge is no longer represented by the information in Table 1. Instead, the meta-beliefs imply a wider uncertainty over the distributions; while the simulation believes that the probability of Design C meeting requirement to is approximately 77%, the human now believes this probability to be smaller. Hence, the human designer makes different decisions than the simulation.

7.3.1.3 Eliciting Preferences

The modeler should use caution when eliciting a designer's preferences through lotteries. The ability to create an accurate utility function hinges on two assumptions: (1) that the designer will be able to determine which lottery he or she prefers and (2) that the designer

will have the same preferences when facing an actual design problem. The second assumption is likely the most problematic as designers typically think of designs as deterministic systems and not lotteries. In general, I discourage the use of pure lotteries with engineers, such as the one in Figure 22, as the concept is rather foreign. Instead, I recommend rephrasing lotteries in a way that designers would find more familiar. Some examples are given in the next subsection.

7.3.2 Suggested Guidelines Experiments

For any experiment, it is important that the incentive structure for the designer be clearly defined. This helps to remove ambiguity in defining preferences. For example, the human subjects can be given a reward (monetary or otherwise) depending on how well the subjects meet the requirements for the design. Using a set reward structure, the experiment can create the appearance of both hard and soft requirements, offering no reward if certain requirements are not met. The human subject should also have a clear understanding of the consequences of going over time. Alternatively, instead of time, the designer can be given a computational "budget". To model discounting, the designer might be allowed to keep the portion of the budget that he or she does not use.

At the same time, the modeler should use caution when creating a reward system; this is especially true if the modeler is trying to mimic an actual engineering organization. With the reward structure clearly represented, the human subject may start to view the design process as a game, and his or her behavior may depart from his or her behavior in a normal design situation. Often the rewards for an engineer are more implicitly understood by the engineer. For example, the designer knows that strong performance will likely lead to a promotion in the organization in the future; the designer is not given a specific monetary reward for a particular performance.

As mentioned above, giving subjects beliefs as probability distributions can create meta-beliefs and lead to unintended consequences. If any probabilistic data is given to the designer, it should be presented as *factual information* and not *beliefs*. In general, the use of the word "beliefs" should be avoided when presenting information as it can lead the human

subjects to develop meta-beliefs. Words can also be used that have an implied uncertainty. For example, one could tell the designer that induced drag "scales with" the reciprocal of aspect ratio and that a drag coefficient was "approximated" to equal 0.3. These words imply that uncertainty exists in the exact relation between alternatives and attributes.

Perhaps the most effective means for generating beliefs would be to let the designer experiment with information and form their beliefs organically. For example, the subject could be allowed to use a low fidelity model and a high fidelity model on a "practice" problem before the experiment begins. By comparing the differences between the models, the designer can obtain their own ideas about the uncertainty in the low fidelity model. In this way, the experiment avoids the use of probability distributions which tend to increase the uncertainty of the designer.

As mentioned earlier, the human subject should be unfamiliar with the problem in order to control their knowledge. A conceptual design problem would also likely be most appropriate, since detailed design problems tend to contain less uncertainty. However, the amount of information given to the designer should be as small as practical; the Gaussian process model is able to store and parse large amounts of information simultaneously, which an inexperienced human may not be able to match.

Using the methods described above, the modeler can create a simulation and human subjects experiment that have very similar initial conditions. This will be helpful when validating the designer's decisions. However, this will provide little insight into a designer's preferences and beliefs. Follow-up interview questions could be useful in better understanding the designer's reasoning. However, as mentioned earlier, more accurate results can likely be obtained if probability distributions and lotteries are avoided. Instead, these questions can be rephrased in ways which the designer might be more familiar. For example, rather than asking a designer to draw their beliefs and confidence intervals, the experimenter could ask the designer "At what values of x do you think it is likely that y is greater than 5?", "What is the highest value of y you would expect at this x" and "At what value of x do you trust the model the least?" If a particular distribution is assumed (such as a normal

distribution), these questions can give insight to the general shape. Using the first and second questions, the modeler has enough information to determine the mean and variance of the designer beliefs. The last question can help the modeler determine where the maximum variance in the model lies. While these questions may not give a precise distribution, they can provide insight into the shape of the designer's belief structure.

CHAPTER VIII

CONCLUSION AND FUTURE DIRECTIONS

8.1 Summary of Contributions

This thesis has outlined a framework for mimicking engineering decision making computationally. Chapter 3 described models for representing the designer's knowledge and preferences. A custom covariance function was derived to better represent the designer's beliefs when extrapolating information. A utility function was proposed for a designer in a requirements-driven organization. This utility function was then transformed into a multi-attribute utility function to represent the designer's preferences in the context of multiple objectives. This framework was demonstrated on both a single dimension and multiple dimension problem in Chapter 4.

Chapter 5 extended the framework to account for sequential decision making. Algorithms were derived to calculate expected utility for *actions* in addition to *alternatives*. Guidelines were given for encoding these actions into the framework. The utility function of Chapter 3 was extended to account for time and budget. In Chapter 6, the single dimension example from Chapter 4 was revisited to explore the effect of both the sequential decision making and utility of time algorithm. Additionally, the sequential decision making algorithm was demonstrated on a variable fidelity analysis problem involving three alternatives, two tools, and two requirements. Finally, the concept of validation with a human subjects experiment was analyzed with several recommendations made.

Overall, the demonstrations have the shown strong potential for the framework to mimic engineering decision making. The variable fidelity problem in Chapter 6 shows the designer develops a clear and logical strategy when choosing actions, accounting for contingencies in ways shared by its human counterparts. At the same time, there are still opportunities for significant improvement as demonstrated by the first example in Chapter 6; for example, improvements can be made to the learning model to better account for the way the designer

updates his or her beliefs when encountering "surprising" information. As Chapter 7 indicated, data from a human subjects experiment could lead to substantial improvements in the models.

8.2 Directions for Future Work

8.2.1 Model Improvements

Many improvements can be made to the models themselves to enhance the validity of the model. Some of these have already been identified: hyperparamter selection including noise can be investigated to determine a robust method for learning that mimics how humans actually learn. The shape of the utility function can be investigated to determine a more precise for functional form for replicating designer preferences.

In regards to sequential decision making, there is much room for improvement in terms of computation time. If robust heuristics can be determined that would simplify the decision tree, then huge gains could be made in speed of computation. Research could also explore the limits of humans ability to make sequential decision making to determine where the framework might differ from that of a human.

8.2.2 Organization Simulation

As mentioned in Chapter 1, then ultimate goal of this framework is to incorporate it into an organizational dynamics simulation. To do so, the simulated designer needs the capability to interact with other designers and management. Chapter 5 has already shown how actions such as consulting experts and meeting with other designers can be incorporated into the action space. However, questions of organizational hierarchy and communication still need to be investigated. For an individual designer, the framework is formulated in terms of information. When multiple designers are involved, this information is transferred to different teams through communication, which can alter the information in unintended ways. These concepts need to be investigated in order to accurately model real organizations.

The creation of such a simulation could lead to very insightful experiments. The sequential decision making algorithm's independence from a specific reality model is especially

powerful; running the simulation only once, the modeler can investigate under what conditions the designer would make certain decisions. This enables the framework to answer many interesting questions about the designer's behavior, such as:

- Under what conditions would designers go over their allotted amount of time? How do their interactions with other organization members influence their propensity to miss a deadline?
- How good are the designers' final designs compared to what the organization would consider the "best" design?
- Given multiple tasks, is a designer effective in managing his or her time?
- In what ways does communication between designers improve and impede the design process?

These are just some of the questions that could be investigated with a model of a full organization. Similarly, the modular structure of the framework allows the modeler to test normative models for the same design problem. For example, a normative value model could replace the designer's multi-attribute utility function; the results of the value model could then be compared to the other utility functions.

8.2.3 Mechanism Design

Given the same set of actions and identical prior knowledge about a design problem, the only thing which differentiates two designers is their utility function. Since the utility function is derived from the designer's incentive structure, one could ask the question "What organization structure and incentives leads to the best organization performance?" In the field of game theory, *mechanism design* attempts to design the best game in order return a desired outcome. Hazelrigg advocates the use of mechanism design in order to align the interests of the designer with the interests of the organization [31]. In this way, a designer could experiment with different utility functions to determine which one maximizes organization performance and system value.

APPENDIX A

DERIVATION OF GAUSSIAN PROCESS MODEL COVARIANCE FUNCTION

As mentioned in Section 3.3.2, current mean and covariance functions fail to capture the desired characteristics of beliefs in extrapolation. The purpose of this appendix to to explicitly derive the covariance function used throughout the thesis. The following assumptions were made about the designer's beliefs regarding the functional relationship between alternatives and attributes:

- The slope of the underlying function is smooth and infinitely differentiable (this implies that the underlying function itself is also smooth).
- In the absence of information, the expectation of the slope is stationary and assumed to be zero (principle of indifference).
- The variability in slope in the extrapolated region is consistent with the variability in slope in the interpolated region.

Since the assumptions are primarily concerned with the slope of the Gaussian process model, my strategy for derivation was to define the slope of the Gaussian process model with the desired properties and then to calculate the resulting mean and covariance functions. This strategy is effective due to the fact that derivatives are linear operators; since a linear operation on a normal distribution results in a normal distribution, the derivative of a Gaussian process is also a Gaussian process.

The second assumption states that the slope of the expectation of the Gaussian process should return to zero in the absence of information. If the derivative of a Gaussian process is also a Gaussian process, this implies that the mean function for the slope is equal to zero (since a Gaussian process returns to its mean function in the absence of information). With the mean function of the slope defined, the only remaining choice is a selection of a covariance function for the slope. For this derivation, the squared exponential covariance

was chosen to represent the slope. Theoretically, any covariance function could be used; the squared exponential was chosen primarily for its simple analytic form which can be easily integrated. This selection implies the first assumption.

Drawing on work by Solak et al. on incorporating derivative observations into Gaussian process models, I defined the covariance between two derivative observations as the squared exponential covariance function [67]:

$$\operatorname{cov}\left(\frac{dy}{d\boldsymbol{x}}, \frac{dy^*}{d\boldsymbol{x^*}}\right) = \sigma^2 \exp\left(-\delta\left(\boldsymbol{x}, \boldsymbol{x}^*\right)^2\right) \tag{57}$$

where δ is a weighted Euclidean distance between points:

$$\delta(\boldsymbol{x}, \boldsymbol{x}^*) = \left(\sum_{i=1}^{m} \theta_i^2 (x_i - x_i^*)^2\right)^{1/2} \tag{58}$$

Note that the validity of using the squared exponential covariance function is contingent on the designer's beliefs of the underlying function; specifically, that the slope is stationary and continuous. If the designer did not believe these assumptions about the underlying function then a different covariance function should be used.

The training data used in the Gaussian process models is often function value observations, not slope observations. Therefore, we require a formula for the covariance between two function value observations. This can be found by integrating Equation 57 twice (this is equivalent to treating the training points as integral observations on a squared exponential Gaussian process model):

$$\operatorname{cov}(y, y^*) = \sigma^2 \int_{-\alpha}^{\boldsymbol{x}} \int_{-\alpha}^{\boldsymbol{x}^*} \exp\left(-\delta\left(\boldsymbol{x}, \boldsymbol{x}^*\right)^2\right) d\boldsymbol{x} d\boldsymbol{x}^* \tag{59}$$

Since an integral is defined between two limits, an additional parameter, $-\alpha$, must be introduced into the equation. This parameter can be thought of as a reference point, a known value from which all the observations are based. In other words, all functions whose integral from $-\alpha$ to x does not equal the observation at x will be eliminated from the prior. Therefore, in order to make an observation at an x location, the function value at reference point must be fully defined. At this location, the Gaussian process assigns the mean value to the reference point.

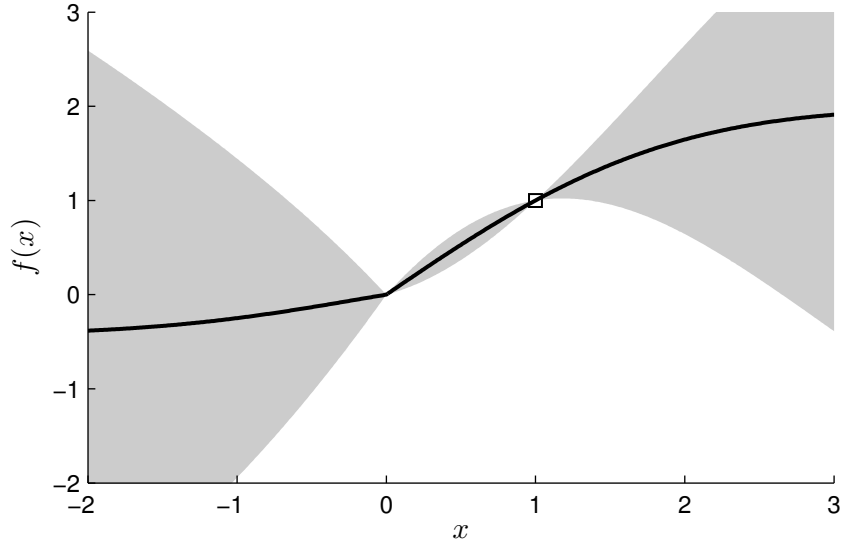


Figure 69: Uncertainty is zero at reference point even in the absence of training data at the reference

Choosing this reference point can be problematic, since no points may be known with certainty. In a way, the Gaussian process model is adding information that we may not have. Figure 69 represents this problem; The Gaussian process model is given only information at $x = 1$. Since the reference point α is set to 0, the Gaussian process assumes a function value at the reference. Note that this function value was not given as training data; the Gaussian process has added information to the model that was not explicitly defined.

Fortunately, as the reference point moves infinitely far way from the data, the effect of the reference point on the Gaussian process model tends towards zero. Therefore, by setting α to infinity, we can achieve the desired Gaussian process model without having to define a reference point. To do this, we leave α as variable and carry it through the integral in Equation 59. Finally, we take the limit of Equation 3.

$$\operatorname{cov}(y_i, y_j) = \frac{\sigma^2}{2}\Bigg[\sqrt{\pi}\Big(2\alpha^m + \delta(0,\boldsymbol{x}) + \delta(0,\boldsymbol{x}^*) + \delta(\boldsymbol{x},\boldsymbol{x}^*)\operatorname{erf}\big(\delta(\boldsymbol{x},\boldsymbol{x}^*)\big)\Big) - \exp\big(-\delta(\boldsymbol{x},\boldsymbol{x}^*)^2\big) - 1\Bigg] \quad (60)$$

$$\boldsymbol{f}|X, X_{tr}, \boldsymbol{f_{tr}} \sim \mathcal{N}\Big(\lim_{\alpha\to\infty} K_{X,X_{tr}}K_{X_{tr}}\boldsymbol{f},\ \lim_{\alpha\to\infty} K_X - K_{X,X_{tr}}K_{X_{tr}}K_{X_{tr},X}\Big) \quad (61)$$

Like the squared exponential function, this covariance function assumes that the underlying function is continuous and smooth. Although the equation appears otherwise, this covariance function is actually stationary; the terms that are not functions of $\delta(\boldsymbol{x}, \boldsymbol{x}^*)$ are eliminated after the limit is applied. Unfortunately, I have been unable to derive an analytical form of Equation 8 with the limit applied for n training points. Therefore, I am unable explicitly prove that the covariance function is stationary. However, I have derived an explicit form for up to three training points; since the mean and variance were stationary in these cases, I believe this is strong evidence that the function is stationary for any arbitrary number of training points.

The presence of the limit in Equation 8 can be computationally cumbersome, since most programming languages are unable to symbolically evaluate the limit as α goes to infinity. Instead, the Gaussian process model can be approximated by choosing a value of α that is much larger than the range of x. In general, two orders of magnitude is sufficient to produce results that are indistinguishable from evaluating the true limit. However, care must be exercised when choosing a value of α, since values that are too large will result in an ill-conditioned matrix. This can especially be an issue when maximizing marginal likelihood; as can be seen in Equation 6, the second term in the equation is the logarithm of the determinant of a covariance matrix. For ill-conditioned matrices, the determinant can be very close to zero, resulting in very large fluctuations in marginal likelihood. Because of this, optimizing marginal likelihood can sometimes be challenging. One strategy for overcoming this is to use a small amount of noise in all observations, even if the noise should be equal to zero. This effectively prevents the determinant of the covariance matrix from reaching zero but does not significantly affecting the shape of the Gaussian process model.

APPENDIX B

DERIVATION OF SEPARATED EXPECTED UTILITY EQUATION

This appendix will demonstrate how the multi-dimensional integral can be transformed into the multiplication of several single-dimension integrals. If numerical integration methods are used that involve a grid along each dimension, this alternative form can offer substantial computational savings.

Consider the following scenario, where the designer is faced with three requirements: u_1 and u_2 are hard requirements and u_s is a soft requirement. The expected utility of each alternative can be calculated using the following equation:

$$\mathrm{E}[u(\boldsymbol{x})] = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} u(g_1, g_2, g_\mathrm{s})p(g_1|\boldsymbol{x})p(g_2|\boldsymbol{x})p(g_\mathrm{s}|\boldsymbol{x})\mathrm{d}g_1\mathrm{d}g_2\mathrm{d}g_\mathrm{s} \tag{62}$$

The requirements-driven utility function from Equation 21 can be substituted into Equation 62:

$$\mathrm{E}[u(\boldsymbol{x})] = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \big(k_h + (1-k_h)u_\mathrm{s}\big)u_1u_2p(g_1|\boldsymbol{x})p(g_2|\boldsymbol{x})p(g_\mathrm{s}|\boldsymbol{x})\mathrm{d}g_1\mathrm{d}g_2\mathrm{d}g_\mathrm{s} \tag{63}$$

If g_1 is integrated first, then all variables in Equation 63 are treated as constants:

$$\mathrm{E}[u(\boldsymbol{x})] = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \big(k_h + (1-k_h)u_\mathrm{s}\big)u_2p(g_2|\boldsymbol{x})p(g_\mathrm{s}|\boldsymbol{x})\left[\int_{-\infty}^{\infty} u_1p(g_1|\boldsymbol{x})\mathrm{d}g_1\right]\mathrm{d}g_2\mathrm{d}g_\mathrm{s} \tag{64}$$

The term in brackets is simply the expected utility of x with respect to the first requirement. Since this is a definite integral, this term is a constant for the remainder of the integrals. Therefore, Equation 64 can be rewritten as:

$$\mathrm{E}[u(\boldsymbol{x})] = \left[\int_{-\infty}^{\infty} u_1p(g_1|\boldsymbol{x})\mathrm{d}g_1\right]\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \big(k_h + (1-k_h)u_\mathrm{s}\big)u_2p(g_2|\boldsymbol{x})p(g_\mathrm{s}|\boldsymbol{x})\mathrm{d}g_2\mathrm{d}g_\mathrm{s} \tag{65}$$

A similar argument can be made for the remaining hard requirements. Therefore, for n hard requirements, the expected utility equation becomes:

$$\mathrm{E}[u(\boldsymbol{x})] = \int_{-\infty}^{\infty} \big(k_h + (1-k_h)u_s(\boldsymbol{g}_s)\big)p(\boldsymbol{g}_s|\boldsymbol{x})d\boldsymbol{g}_s\prod_{i}^{n}\int_{-\infty}^{\infty} u(g_{h,i})p(g_{h,i}|\boldsymbol{x})dg_{h,i} \tag{66}$$

REFERENCES

[1] Abbas, A. E. and Matheson, J. E., "Normative target-based decision making," *Managerial and Decision Economics*, vol. 26, no. 6, pp. 373–385, 2005.

[2] Ahmed, S. and Wallace, K., "Identifying and support the knowledge needs of novice designers within the aerospace industry," *Journal of Engineering Design*, vol. 15, pp. 475–492, 2004.

[3] Ahmed, S. and Wallace, K., "Understanding the knowledge needs of novice designers in the aerospace industry," *Design Studies*, vol. 25, pp. 155–173, 2004.

[4] Ahmed, S., Wallace, K., and Blessing, L., "Understanding the differences between how novice and experienced designers approach design tasks," *Research in Engineering Design*, vol. 14, pp. 1–11, 2003.

[5] Arrow, K., "The theory of risk aversion," in *Essays in the Theory of Risk-Bearing*, Markham Publishing Company, 1971.

[6] Atman, C. J., Adams, R. S., Cardella, M. E., Turns, J., Mosborg, S., and Saleem, J., "Engineering design processes: A comparison of students and expert practitioners," *Journal of Engineering Education*, vol. 96, pp. 359–379, OCT 2007.

[7] Augustine, N., *Augustine's Laws*. American Institute of Aeronautics and Astronautics, 1997.

[8] Ball, L., Evans, B., and Dennis, I., "Cognitive processes in engineering design: A longitudinal study," *Ergonomics*, vol. 37, no. 11, pp. 1753–1786, 1994.

[9] Browning, T., Deyst, J., and Eppinger, S., "Adding value in product development by creating information and reducing risk," *IEEE Transactions on Engineering Management*, vol. 49, 2002.

[10] Cagan, J. and Kotovsky, K., "Simulated annealing and the generation of the objective function: a model of learning during problem solving," *Computational Intelligence*, vol. 13, no. 4, pp. 534–581, 1997.

[11] Carley, K. and Gasser, L., "Computational organization theory," *Multiagent systems: A modern approach to distributed artificial intelligence*, pp. 299–330, 1999.

[12] Chanron, V. and Lewis, K., "A study of convergence in decentralized design," in *ASME Design Engineering Technical Conference*, pp. 2–6, 2003.

[13] Chanron, V., Singh, T., and Lewis, K., "Equilibrium stability in decentralized design systems," *International journal of systems science*, vol. 36, no. 10, pp. 651–662, 2005.

[14] Chase, W. and Simon, H., "Perception in chess," *Cognitive Psychology*, vol. 4, pp. 55–81, 1973.

[15] Christie, J. D., "DOD on a glide path to bankruptcy.," *Proceedings of the United States Naval Insitute*, vol. 134, no. 1264, pp. 22 – 25, 2008.

[16] Collopy, P., "Adverse impact of extensive attribute requirements on the design of complex systems," in *7th Aviation Technology, Integration, and Operations (ATIO) Conference*, 2007.

[17] Cross, N., *Designerly Ways of Knowing*. Springer-Verlag London Limited, 2006.

[18] Dorst, K. and Cross, N., "Creativity in the design process: co-evolution of problemsolution," *Design Studies*, vol. 22, no. 5, pp. 425 – 437, 2001.

[19] Ellsberg, D., "Risk, ambiguity, and the savage axioms," *The Quarterly Journal of Economics*, vol. 75, no. 4, pp. pp. 643–669, 1961.

[20] Eppinger, S., Whitney, D., and Gebala, D., "Organizing the tasks in complex dseign projects: Development of tools to represent design procedures," in *NSF Design and Manufacturing Systems Conference*, 1992.

[21] Eppinger, S., Whitney, D., Smith, R., and Gebala, D., "A model-based method for organizing tasks in product development," *Research in Engineering Design*, vol. 6, no. 1, pp. 1–13, 1994.

[22] Fischhoff, B., Goitein, B., and Shapira, Z., "Subjective expected utility: A model of decision-making".," *Journal of the American Society for Information Science*, vol. 32, no. 5, pp. 391 – 399, 1981.

[23] Galbraith, J., *Organization Design*. Addison-Wesley Publishing Co., 1977.

[24] Geiselhart, K., "Enggen engine cycle analysis program release 4.0 user's guide," tech. rep., NASA Langley Research Center, 2009.

[25] Gero, J., "Design prototypes: a knowledge representation schema for design," *AI magazine*, vol. 11, no. 4, p. 26, 1990.

[26] Gersick, C. J. G., "Marking time: Predictable transitions in task groups," *The Academy of Management Journal*, vol. 32, no. 2, pp. pp. 274–309, 1989.

[27] Giffin, M., de Weck, O., Bounova, G., Keller, R., Eckert, C., and Clarkson, P., "Change propagation analysis in complex technical systems," *Journal of Mechanical Design*, vol. 131, p. 081001, 2009.

[28] Hayes, C. and Akhavi, F., "Design decision making: Adapting mathematical paradigms to fit designer's actual needs," in *Proceedings of the Human Factors and Ergonomics Society 52nd Annual Meeting*, 2008.

[29] Hayes, C. and Akhavi, F., "Combining naturalistic and mathematical decision aids to support product design," in *9th International Conference on Naturalistic Decision Making*, 2009.

[30] Hazelrigg, G. A., "A framework for decision-based engineering design," *Journal of Mechanical Design*, vol. 120, pp. 653–658, 1998.

[31] Hazelrigg, G., *Systems Engineering: An Approach to Information-Based Design*. Prentice-Hall, 1996.

[32] Hazelrigg, G., *Fundamentals of Decision Making for Engineering Design and Systems Engineering*. 2012.

[33] Jansson, D. and Smith, S., "Design fixation," *Design Studies*, vol. 12, no. 1, pp. 3–11, 1991.

[34] Jin, Y. and Levitt, R. E., "The virtual design team: A computational model of project organizations," *Computational & Mathematical Organization Theory*, vol. 2, pp. 171–195, 1996. 10.1007/BF00127273.

[35] Jones, D. R., Schonlau, M., and Welch, W. J., "Efficient global optimization of expensive black-box functions," *Journal of Global Optimization*, vol. 13, pp. 455–492, 1998.

[36] Kahneman, D. and Tversky, A., "Prospect theory: An analysis of decision under risk," *Econometrica*, vol. 47, no. 2, pp. pp. 263–292, 1979.

[37] Keeney, R. and Raiffa, H., *Decisions with Multiple Objectives*. Cambridge University Press, 1993.

[38] Kim, H. M., Michelena, N., Papalambros, P., and Jiang, T., "Target cascading in optimal system design," *Journal of Mechanical Design*, vol. 125, pp. 474–480, 2003.

[39] Kreps, D., *Notes on the Theory of Choice*. Westview Press, 1988.

[40] Kroo, I., Braun, R., Gage, P., and Sobieski, I., "Multidisciplinary optimization methods for aircraft preliminary design," in *5th Symposium on Multidisciplinary Analysis and Optimization*, 1994.

[41] Lee, B. and Pardis, C., "Accounting for the duration of analyses in design process decisions," in *2010 SAE World Congress*, 2010. Detroit, MI.

[42] Lewis, K. and Mistree, F., "Modeling interactions in multidisciplinary design: A game theoretic approach," *AIAA journal*, vol. 35, pp. 1387–1392, 1997.

[43] Lewis, K. and Mistree, F., "Collaborative, sequential, and isolated decisions in design.," *Journal of Mechanical Design*, vol. 120, no. 4, pp. 643 – 652, 1998.

[44] Loch, C. and Terwiesch, C., "Rush and be wrong or wait and be late? a model of information in collaborative processes," *Production and Operations Management*, vol. 14, pp. 331–343, 2005.

[45] McCorduck, P., *Machines who think: a personal inquiry into the history and prospects of artificial intelligence*. A.K. Peters, 2004.

[46] Mehalik, M. and Schunn, C., "What constitutes good design? a review of empirical studies of design processes," *International Journal of Engineering Education*, vol. 22, no. 3, p. 519, 2007.

[47] Mistree, F., Hughes, O., and Bras, B., "Compromise decision support problem and the adaptive linear programming algorithm," *Progress in Astronautics and Aeronautics*, vol. 150, pp. 251–251, 1993.

[48] Nelson, J. R. and Timson, F. S., "Relating technology to acquisition costs: Aircraft turbine engines," tech. rep., Rand Corporation, 1974.

[49] Okhuysen, G. A. and Waller, M. J., "Focusing on midpoint transitions: An analysis of boundary conditions," *The Academy of Management Journal*, vol. 45, no. 5, pp. pp. 1056–1065, 2002.

[50] Olson, J. T., *The Collective Potential: Achieving Organizational Potential by Design*. PhD thesis, Carnegie Mellon University, 2006.

[51] Perlow, L. A., "The time famine: Toward a sociology of work time," *Administrative Science Quarterly*, vol. 44, no. 1, pp. 57–81, 1999.

[52] Pimmler, T. and Eppinger, S., "Integration analysis of product decompositions," in *ASME Design Theory and Methodology Conference*, 1994.

[53] Pratt, J. W., "Risk aversion in the small and in the large," *Econometrica*, vol. 32, no. 1/2, pp. pp. 122–136, 1964.

[54] Purcell, A. and Gero, J., "Effects of examples on the results of a design activity," *Knowledge-Based Systems*, vol. 5, no. 1, pp. 82 – 91, 1992. ¡ce:title¿Artificial Intelligence in Design Conference 1991 Special Issue ¡/ce:title¿.

[55] Rasmusen, E., *Games and Information: An Introduction to Game Theory*. Blackwell Publishers, 1989.

[56] Rasmussen, C. and Williams, C., *Gaussian Processes for Machine Learning*. MIT Press, 2006.

[57] Riihimäki, J. and Vehtari, A., "Gaussian process with monotonicty information," in *13th International Conference on Artificial Intelligence and Statistics*, 2010.

[58] Robinson, M., "How design engineers spend their time: Job content and task satisfaction," *Design Studies*, vol. 33, pp. 391–425, 2012.

[59] Roozenburg, N. and Cross, N., "Models of the design process: integrating across the disciplines," *Design Studies*, vol. 12, no. 4, pp. 215 – 220, 1991.

[60] Schank, R. C. and Abelson, R. P., "Scripts, plans, and knowledge," in *Proceedings of the 4th international joint conference on Artificial intelligence - Volume 1*, (San Francisco, CA, USA), pp. 151–157, Morgan Kaufmann Publishers Inc., 1975.

[61] Seers, A. and Woodruff, S., "Temporal pacing in task forces: Group development or deadline pressure?," *Journal of Management*, vol. 23, no. 2, pp. 169 – 187, 1997.

[62] Simon, H. A., "A behavioral model of rational choice," *Quarterly Journal of Economics*, vol. 69, no. 1, pp. 99 – 118, 1955.

[63] Simon, H. A., "The structure of ill structured problems," *Artificial Intelligence*, vol. 4, no. 34, pp. 181 – 201, 1973.

[64] Simon, H. A., *The Sciences of the Artificial*. The MIT Press, 3rd ed., 1996.

[65] Smith, R. and Eppinger, S., "Identifying controlling features of engineering design iteration," *Management Science*, pp. 276–293, 1997.

[66] Smith, R. and Eppinger, S., "A predictive model of sequential iteration in engineering design," *Management Science*, vol. 43, no. 8, pp. 1104–1120, 1997.

[67] Solak, E., Murray-Smith, R., Leithead, W., Leith, D., and Rasmussen, C., "Derivative observations in gaussian process models of dynamic systems," in *Conference on Neural Information Processing Systems* (Becker, S., Thrun, S., and Obermayer, K., eds.), Advances in neural information processing systems 15, MIT Press, 2003.

[68] Takeda, H., Veerkamp, P., and Yoshikawa, H., "Modeling design process," *AI magazine*, vol. 11, no. 4, p. 37, 1990.

[69] Thompson, S. and Pardis, C., "An investigation into the decision analysis of design process decisions," *Journal of Mechanical Design*, vol. 132, 2010.

[70] Ullman, "A model of the mechanical design process based on empirical data," *Artificial Intelligence for Engineering Design, Analysis, and Manufacturing*, vol. 2, pp. 33–52, 1988.

[71] Von Neumann, J. and Morgenstern, O., *Theory of Games and Economic Behavior*. Princeton University Press, 3rd ed., 1953.

[72] Waller, M. J., Zellmer-Bruhn, M. E., and Giambatista, R. C., "Watching the clock: Group pacing behavior under dynamic deadlines," *The Academy of Management Journal*, vol. 45, no. 5, pp. pp. 1046–1055, 2002.

[73] Younossi, O., Stem, D., Lorell, M., and Lussier, F., "Lessons learned from the F/A-22 and F/A-18E/F development programs," tech. rep., Rand Corporation, 2005.