MODELING AND ANALYSIS OF COMPLEX SYSTEMS DESIGN PROCESSES

A Dissertation

Submitted to the Faculty

of

Purdue University

by

Kushal A. Moolchandani

In Partial Fulfillment of the

Requirements for the Degree

of

Doctor of Philosophy

December 2018

Purdue University

West Lafayette, Indiana

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF DISSERTATION APPROVAL

Dr. Daniel A. DeLaurentis, Chair

School of Aeronautics and Astronautics

Dr. William A. Crossley

School of Aeronautics and Astronautics

Dr. Jitesh H. Panchal

School of Mechanical Engineering

Dr. Tahira Reid

School of Mechanical Engineering

Approved by:

Dr. Weinong Chen

Head of the School Graduate Program

ACKNOWLEDGMENTS

My time as a PhD student has taught me a lot and was full of experiences that I will remember forever. This journey would not have been possible without the care and support of the many people who played a role.

First, I would like to thank the members of my committee, Dr. Daniel A. DeLaurentis, Dr. William A. Crossley, Dr. Jitesh H. Panchal, and Dr. Tahira H. Reid, for their guidance throughout the program and taking the time to review this dissertation. I am especially thankful to my advisor, Dr. Daniel A. DeLaurentis, for his patience and guidance at all stages of my research at Purdue.

I thank all my friends and colleagues at Purdue, and especially those in the Systems-of-Systems laboratory for their moral support and assistance. I also want to thank Dr. Bernard Tao, with whom I consulted on many matters concerning research and career.

Lastly, I thank my parents, Mr. Ashok and Mrs. Reshma Moolchandani, and my sister, Harsha Moolchandani, for their unwavering love and patience during all my endeavors. In all my accomplishments, I owe the most to them.

TABLE OF CONTENTS

Page

LIST OF TABLES vii

LIST OF FIGURES viii

SYMBOLS x

ABSTRACT xi

1 INTRODUCTION 1
- 1.1 Problem Scope, Proposed Research Goal, and Objectives 4
- 1.2 Heilmeier Catechism: Contributions Of This Research 7
- 1.3 Summary and Dissertation Organization 8

2 LITERATURE REVIEW 9
- 2.1 Research on Modeling Complex Systems Design Processes 12
- 2.2 Recognized Challenges to Complex Systems Engineering 15
  - 2.2.1 Values, Incentives, and Hierarchy 17
  - 2.2.2 Organizations and Teams 21
  - 2.2.3 People and Machines in Systems Engineering 26
- 2.3 A Brief Discussion of Decision Theory 30
- 2.4 Selecting a Design Process Policy 33
- 2.5 Summary 36

3 DESIGNING THE DESIGN PROCESS 37
- 3.1 Problem Context and Definition 37
  - 3.1.1 Problem Definition 39
  - 3.1.2 Assumptions in Modeling 40
- 3.2 Model of Agents' Decision-Action Behavior 42
- 3.3 Valuation: Setting Up the Agent Rewards 44
- 3.4 Planning: Selecting an Optimal Policy 47
- 3.5 Guiding the Design Process Using Principal-Agent Theory 49
- 3.6 Setting Up The Modeling Framework 51

4 FEATURES OF THE DESIGN PROBLEM 57
- 4.1 The synthetic design problem 59
- 4.2 Case 1: Interacting agents who seek to arrive at a common value of design outcome 59
- 4.3 Case 2: System-level designer provides a target value 64
- 4.4 Case 3: Teams add bias to the value of design variable 67

Page

4.5 Case 4: Teams learn to adapt their behavior . . . 69
4.6 Case 5: The system-level designer formulates heuristics . . . 73
4.7 Summary of Case Studies with Framework . . . 76

5 DEMONSTRATION OF FRAMEWORK USING AN AIRCRAFT DESIGN PROBLEM . . . 79
5.1 Demonstration Problem Setup . . . 80
5.2 Results from Framework Versus Optimization Algorithm . . . 83
5.3 Effect of Organization Restructuring on Design Outcome . . . 88
5.4 Discussion of Compensation Given to the Teams by the Principal . . . 91
5.5 Summary and Discussion of Demonstration Problem . . . 94

6 DICUSSION OF DESIGN PROCESS MODELING . . . 97
6.1 Role of Design Process Models . . . 99
6.1.1 Setting Up An Organizational Structure . . . 99
6.1.2 Flexibility In Modeling Design Processes . . . 101
6.1.3 Using Value-based Approach In Complex Systems Design . . . 102
6.1.4 Examples of Questions a Model of Design Process Can Answer 104
6.2 Qualitative Research in Engineering Design . . . 106
6.2.1 Conducting Qualitative Research . . . 108
6.3 Summary of Modeling Framework . . . 110

7 CONCLUSIONS AND FUTURE WORK . . . 113
7.1 Future Work . . . 115

REFERENCES . . . 117

VITA . . . 122

LIST OF TABLES

Table		Page
1.1	Research questions, hypotheses, and proposed approach	5
2.1	Attributes of complex systems (from [12])	10
2.2	Axioms of Rationality	31
4.1	Comparison of results from cases 2, 4, and 5	76
4.2	Summary of cases simulated using proposed framework.	78
5.1	Bounds on design variables from aircraft design application problem	81
5.2	Subsystem teams in the aircraft design application problem	82
5.3	Design requirements and constant parameters of modeling.	84
5.4	Performance requirements for aircraft design problem.	84
5.5	Results comparison between framework discussed and simulated annealing optimizer.	86
5.6	Comparison of variables between framework and simulated annealing optimizer.	87
5.7	Results comparison between baseline case and when performance team's objectives merge with principal.	90
5.8	Comparison of design variables between baseline case and when performance team's objectives merge with principal.	90
6.1	Percentage of times teams choose among available actions	100
7.1	Research questions and outcomes of this research	114

LIST OF FIGURES

Figure Page

1.1 The "traditional" sequential view of design process. 2

1.2 An hierarchical and iterative view of design process. 3

1.3 A schematic of organization structure showing hierarchy and informal interactions. 4

2.1 Classification of complex systems design research [13]. 13

2.2 Proposed areas of research to advance complex systems design [3]. 16

2.3 The three dimensions of human behavior modeling (from Ref. [30]). 25

2.4 Different types of risk profiles: The concave curve is for a risk averse agent, the straight line is risk neutral, and the convex curve is risk prone. 32

2.5 A design structure matrix. 34

2.6 The extended design structure Matrix from [43]. 35

3.1 Schematic of the process-system as a bi-level framework. 41

3.2 Model of agents' decision-action behavior. 43

3.3 Flowchart of design framework for complex systems design. 53

4.1 Pareto-optimal frontier of the comet problem. 60

4.2 Flowchart for Case 1: Teams which arrive at common value of design. . . . 62

4.3 Case 1: Pareto-optimal frontier of the comet problem with design solutions explored by the teams. 63

4.4 Flowchart of Case 2: The principal provides the teams with a target value to work towards. 66

4.5 Case 2: Pareto-optimal frontier of the comet problem with design solutions explored by the teams when guided by the principal. 67

4.6 Flowchart for Case 4: When teams learn and adapt their behavior. 70

4.7 Case 4: Pareto-optimal frontier of the comet problem with design solutions explored by the teams when they evaluate future discounted values. 73

Figure Page

4.8 Case 5: Pareto-optimal frontier of the comet problem with design solutions explored by the teams when they are guided by the principal. 76

5.1 Schematic of organization structure for demonstration problem. 81

5.2 Breakdown of Aircraft Direct Operating Costs from [53]. 83

5.3 Schematic of organization structure for demonstration problem. 89

5.4 Cumulative compensation received by structures and aerodynamics teams. 93

5.5 Compensation received by structures and aerodynamics teams in each iteration. 94

5.6 Cumulative compensation received by structures and aerodynamics teams. 95

6.1 The product-process-organization nesting of complex systems design process.98

SYMBOLS

x	design variable
v	value
a	action (or task)
π	design policy

Subscripts:

i	current state
k	agents

ABSTRACT

Moolchandani, Kushal A. Ph.D., Purdue University, December 2018. Modeling and Analysis of Complex Systems Design Processes. Major Professor: Daniel A. DeLaurentis.

This work proposes a framework for modeling an organization as a network of autonomous design agents who collectively work on the design of a complex system. The research objective is to identify a design process policy which best suits the current organization evaluated on the basis of the value that it provides to the organization. Consequently, the research question is, "How does an organization comprised of autonomous design teams select a design process policy which provides the highest value?" The proposed framework models design teams as agents who adapt their behavior using information on design variables available from other teams and the incentives in form of rewards from a system-level designer.

While extant literature on complex systems design has proposed several models of design processes, there is still a need for models that are versatile enough to represent different types of purposes and scopes of hierarchical levels. Further, models still do not account for the social, cultural, and political aspects of design. Due to the invariably long development times of a complex system, the environment's dynamics such as changing requirements would require all design teams to update their models and decisions during the process. They have to do this while accounting for the decisions of the other teams. The system-level designer, on the other hand, has to ensure that the design teams' decisions are in the best interest of the organization, which is to maximize value. The work proposed in this research addresses these issues by taking a bottom-up approach to modeling this complex, dynamic and uncertain design environment, where organizational-level outcomes are modeled as a result of decisions of individual teams who respond to local incentives.

The system-level designer and the subsystem design teams, are modeled to interact with other agents with whom they share design variables. The subsystem teams first solve their local design problems, and then exchange the results of these problems with other teams. The proposed modeling is versatile to represent human behaviors such as their adding of margins to design variables during the process of information exchange. In each interaction, the receiving teams make decisions to update their local variable values with the one newly available or to continue to use their own value. They make these decisions on the basis of which decision leads to the highest utility measured by a predefined value function. Thus, each team acts in its self-interest and maximizes its local value. In case they do not arrive at a common design, the system-level designer attempts to assign rewards which incentivize the teams to update designs such that they are compatible with the other teams. In such cases, the teams would be willing to forgo a portion of their utility obtained from the design outcome if they are compensated for this loss by the system-level designer. Therefore, the task of a system-level designer is to solve a compatibility problem which trades off between different subsystems outcomes and arrives as the final design while maximizing the organization's value.

The framework is developed and then described through a series of increasingly complex design cases using a synthetic optimization problem. Following this, an aircraft design problem serves as a demonstration of application of this framework. The results obtained from both the synthetic and the demonstration problem then inform the discussion of various characteristics of a complex systems design process.

1. INTRODUCTION

As the capability requirements of modern engineered systems grows, so does their complexity. One of the sources of this complexity is the presence of a large number of interacting parts that together provide capabilities which are greater than the sum of those parts, and this "emergent" behavior is a key distinguishing feature of *complex systems* [1]. A consequence of the increase in complexity is that the number of people involved during development and the amount of information they exchange also increase. This means that the development process of complex systems, besides being untenable to the reductionist approach of traditional systems engineering, faces challenges pertaining to the social, economic, and political interactions during complex systems development [2]. Unfortunately, the methods and approaches proposed for managing the increasing complexity of design processes have not kept pace with the complexity of systems being developed [2], leading to the now familiar problems of cost and schedule overruns, or performance deficits [2–4]. In other words, there exists *a need for approaches for modeling not just the systems being developed, but also the processes utilized in doing so.*

Systems engineering, which was developed to manage systems throughout their life cycles, has traditionally viewed design processes as a series of sequential steps, or phases, starting with specification of system requirements, followed by identification and analysis of alternatives, and the eventual selection and detail design of the preferred alternative (Fig. 1.1). All design processes could be divided into these broad phases of increasing information and the number of people involved, though the exact definitions of the phases could differ [5].

This sequential view of design is different from the paradigm of concurrent engineering and integrated product and process design (IPPD), both of which were later advancements and recognized that an actual design process is iterative, hierarchical,

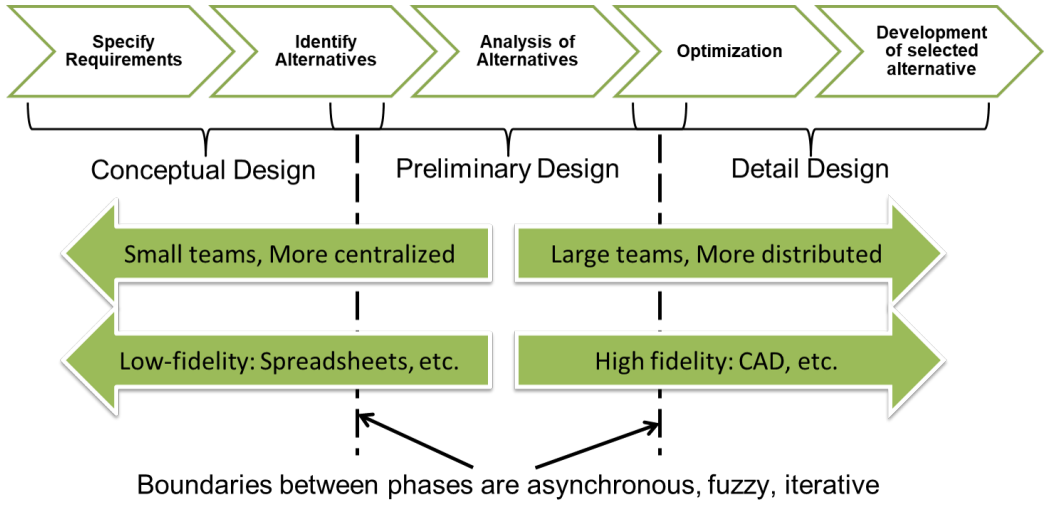

Figure 1.1.: The "traditional" sequential view of design process.

and many of the steps involved are done in parallel. Figure 1.2 shows such an hierarchical view with multiple design teams each with their local processes and some features of the design process such as the delegation of tasks across levels of hierarchy. Under this view, complex systems are designed in an environment of geographically separated, semi-autonomous teams and organizations who interact, and may use different set of tools and techniques in their processes. Together these constituents can, in turn, be considered to constitute a dynamic system. As the design evolves from concept to integration, such a 'process-system' will display dynamics of addition of more designers over time, the gradual change in modeling techniques used from low-fidelity ones to high-fidelity ones, etc.

Further elaborating on this view, an organization along with its designers and their tools, all interacting with one another, provides the environment in which complex systems are designed. The schematic in Fig. 1.3 shows an organizations split into disciplinary teams which interact along hierarchical communication links, indicated by solid bidirectional arrows for interactions across levels of hierarchy and dashed arrows for interactions at the same level. Accompanying such orderly and efficient

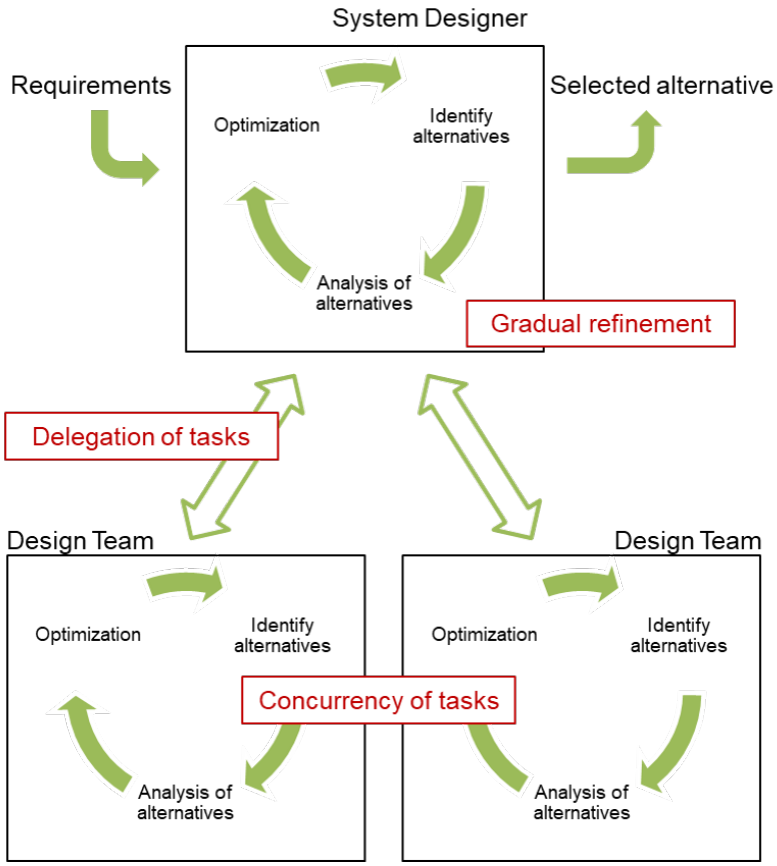

Figure 1.2.: An hierarchical and iterative view of design process.

communication channels could be various other informal interactions among members of different teams, shown in the figure by dotted arrows, which are the information exchanges among teams which are frequently not captured by hierarchical interaction models [6]. This view of system development means that design becomes the task of "organization and management of people and information they develop in the evolution of a product" [7].

The design processes setup for the purpose of complex systems design are customized to the particular system being developed and the organization developing them. Different projects invariably require different processes and even these would change during the course of development. Thus, the *objective* of this work is *to pro-*

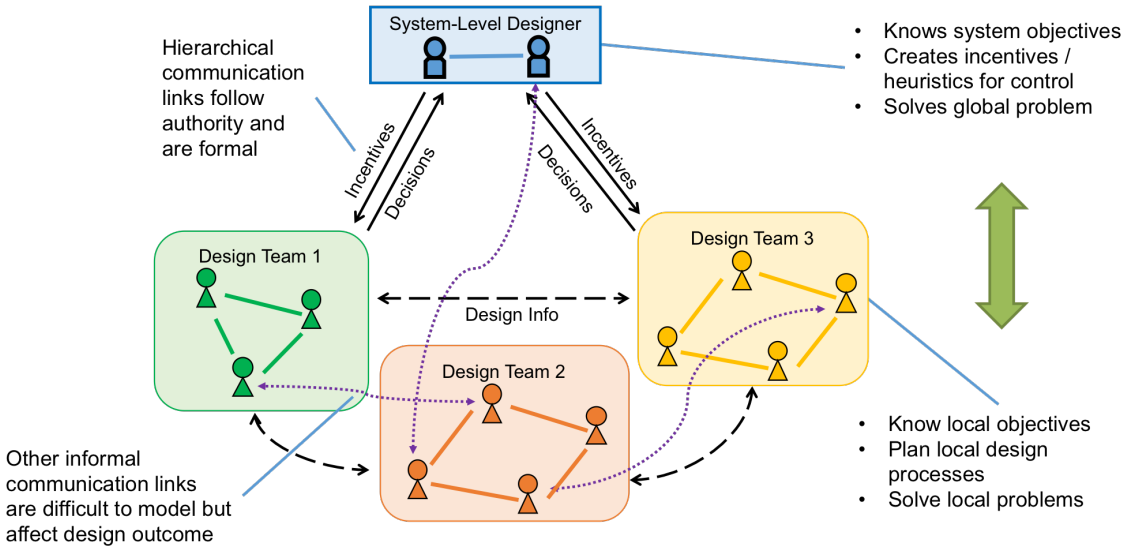

Figure 1.3.: A schematic of organization structure showing hierarchy and informal interactions.

pose a computational framework for modeling and analysis of complex systems design processes which takes into account the particular structure of an organization.

1.1 Problem Scope, Proposed Research Goal, and Objectives

The primary research question we address here is, *"How can an organization comprised of autonomous teams identify an appropriate design process policy for design of complex systems so as to maximize its value?"* Implicit in this question is that an organization evaluates its value by taking into account its own structure and objectives. Table 1.1 shows the two questions we answer in an effort to meet the objective. The first question sets up a model of design teams as adaptive agents who solve subsystem design problems. The second question recognizes that because designing a complex system requires information exchange and coordination among numerous independently operating designers, a facilitator of this system, in absence of any direct control, needs to find ways to influence their decisions. For this purpose, Principal-

Table 1.1.: Research questions, hypotheses, and proposed approach

Research question:	**How can we model the effects of available information on the design teams behavior?**
Hypothesis:	Designers decision-making can be studied by modeling them as learning agents that update their design policies in response to available information.
Proposed approach:	Value Iteration, a method of dynamic programming, can be used to model designers selection of optimal design process strategy.
Expected outcome:	A model for designers decision-making behavior during design process.
Research question:	**How can a system-level designer identify heuristics for control of design teams?**
Hypothesis:	The system-level designer can setup rewards for meeting requirements, which work as incentives to encourage behavior in favor of global objectives.
Proposed approach:	Solving a Principal-Agent model which maximizes value of both design teams and system-level designer, can be used to identify rewards for teams.
Expected outcome:	A model for selection of design heuristics based on giving rewards to teams.

Agent models provide a means to identify design heuristics that can be employed by the system-level designer to guide the decisions of subsystem teams.

Consider two different views of complex systems development processes, viz., a *product-view* and a *process-view*. While the product-view of complex system design focuses on maximizing or minimizing a measure of performance of the system being

developed, the process-view considers design to be a set of decisions, and seeks to improve the decision-making by maximizing or minimizing a measure of performance of the design process. There exists research which suggests that considering the process along with the product early in the development process can lead to shorter system development time-lines and lower costs [7, 8], pointing to a need for modeling design processes. The process-view of complex systems development becomes even more imperative when we consider that the involvement of a large number of technical teams and their interactions have an effect on the outcome of the design process [9]. Therefore, in keeping with the research objective, we take a process-view of complex systems development in this work.

The expected outcome is an approach for modeling design process-systems. Such a model would support studies of effects of organizational structure and the selected design process on product architecture and performance. A key benefit of this model would be that by using the above process-system model different heuristics can be compared for this suitability to the given context. This is important because any change to design requirements usually requires a new set of heuristics because they are usually suitable only for the particular context for which they are established. Thus the process-system model developed in this work will give organizations an ability to evaluate different heuristics before they select one which is suitable for their needs.

The context mentioned above includes the elements of the process-system such as system requirements and the semi-autonomous subsystem teams over which a system-level designer has limited control authority. While the organization seeks to maximize its value by meeting customer requirements, the teams are concerned with solving local subsystem design problems. In this scenario, the proposed heuristics would be selected such that they provide incentives to guide the decision-making of the teams in a manner which favors the system-level objectives. We will consider just two levels of organizational hierarchy, though the model of system-subsystem interaction can be applied recursively at multiple levels within the organization.

1.2 Heilmeier Catechism: Contributions Of This Research

This work proposes a simulation framework useful for selection of a design process policy, which is a particular sequence of set of tasks. Such a model will be useful to an organization engaged in development of complex engineered systems for maximization of value obtained from systems development. The environment in which complex systems are developed is characterized by features such as teams which are not under direct authority of a system-level developer, the competing objectives of the teams, and long development time-scale which makes the identification of a design policy both important and difficult.

Previous studies have proposed methods for selection of design processes using representations such as design structure matrices which show interdependence among tasks, optimization algorithms such as genetic algorithms which help in selection of an optimal policy, etc. However, despite the numerous studies on design process modeling that exist, there is still a need to account for multi-level models which account for designers' behaviors and the dynamic nature of design environment.

This research proposes an approach wherein design teams within an organization are modeled as learning agents that adapt their behavior to changing information. This model would be able to represent designers' behaviors and provide a bottom-up analysis of the process as a result of designers' actions. This model of design process will be useful to any organization engaged in complex systems development. Its utility to the organizations would be twofold, helping in the identification of heuristics for influence over teams' behaviors and in evaluating how organization restructuring done so as to make teams' interactions more efficient would lead to improvement of value obtained from the process.

Assessment of this approach will be based first on comparison with existing computational design approaches and then an analysis of results with those reported in literature which are relevant to the decision-making modeling done in this work.

1.3 Summary and Dissertation Organization

The box below summarizes the objective of this work along with its expected contributions.

Research objective: Propose a framework for selection of complex systems' design process policy which is flexible to account for changing information and which provide highest value to the organization.
Expected outcome: A model of design teams decision-making that can simulate flexible interactions and information exchange.

The dissertation is organized as follows. Chapter 2 presents a review of literature on complex systems design and its development processes along with the shortcomings of current research and the proposed future directions. In particular, this chapter presents a brief review of value-driven design, modeling of design organizations, and decision theory, each of which informs modeling in this dissertation.

Chapter 3 describes the proposed complex systems design process including the terminology used in this work. It establishes the context, states the assumptions made, and presents background on the value iteration method of dynamic programming and principal-agent models both of which are used in this work.

Chapter 4 builds up the framework proposed in this work to be used for selection of a design policy through a series of increasingly complex modeling cases. The steps of the process are elaborated in this chapter along with demonstration of the framework on a synthetic design problem.

Chapter 5 presents an application of the proposed framework to an aircraft design problem.

Chapter 6 presents a discussion of results from the simulations and compares them with the discussion of complex systems design processes presented in literature.

Finally, chapter 7 concludes along with discussion of potential future work in the area of complex systems design.

2. LITERATURE REVIEW

A complex system can be defined as "an assembly of interacting members that is difficult to understand as a whole" [10]. In this definition, 'interacting members' identifies the interdependence of system components, while 'difficult to understand as a whole' recognizes that for such systems the system-level behavior is greater than the sum of the behaviors its of components. The subsystems' interactions are the source of both the complex systems' capabilities as well as the observed complexity. We can break down the architecture of any complex system into several layers of hierarchy and find that these interactions exist across several layers with the result that the subsystems' behaviors depend on those of their neighbors, which are the other subsystems they interact with, and can exhibit sensitivity to even small perturbations [11]. A typical complex system, which can also be referred to as a large-scale system (LSS), has attributes such as uniqueness, lengthy installation times, presence of policy component, etc. [12], see Table 2.1.

Let us first look at the concept of complexity using the example given by Moffat [11]. Consider the phenomenon of heating a fluid between two infinite flat plates. As heat is applied to the bottom plate, the heating and resultant expansion of the fluid sets up a movement in form of convection currents. Up to a certain threshold rate of heat application, the system, when left long enough, will tend to a state of equilibrium such that small perturbations introduced at the equilibrium will have no lasting influence. As the rate of application of heat is increased, there comes a critical point beyond which the movement of the fluid elements no longer remains uniform and instead displays a certain complexity for which the existing models become inadequate. At this stage, the fluid moves in structures called Bernard cells whose movements, because the fluid is no longer uniform, are dependent on their neighboring cells. In this case, as is usually the case, the *complexity* of the system arises

Table 2.1.: Attributes of complex systems (from [12])

Policy component	Technical analysis may be insufficient to inform their operations and they invariably need subjective judgment.
High order	There are many subsystems which together constitute the whole system and they are interdependent for their operation.
Complex to describe	Their modeling and analysis pose challenges which mean they may not be easy to model analytically.
Lengthy installation	Long time is required for their design and deployment.
Unique	This is often the case for LSS.
Prior complete testing impossible	Due to the large size and complexity, testing of the complete system before their operationalization is unlikely.

mainly due to the interactions among its constituent subsystems, here, the Bernard cells. At this stage, we say that it is a complex system.

Analogous to the flow of heat in the above example, engineered systems handle the flow of information among its sub-systems. Over time, engineered systems have been required to provide increasing amount of capabilities both due to the competitive nature of the markets as well as the growing needs and expectations of the society. These increased capability requirements are met with higher numbers of subsystems that are interdependent for their operation, which results in higher information flow among the subsystems. While the upside of this increased complexity is that such systems have capabilities that are greater than the sum of individual subsystems, the

downside is that the design and development time, resources, and investment required increase simultaneously.

The objective of modeling design processes is to come up with a representation of the process and then use it to enable designers make better decisions. The proposed representations rely on the fact that, like complex systems, design processes also share a number of common characteristics. For example, the following is a non-comprehensive list of these characteristics, based on those specified in Refs. [4, 8] among others:

1. Complex systems design is usually done in a multidisciplinary environment by a number of teams.

2. Design teams interact by exchanging information in an iterative manner.

3. System design is gradually refined with the fidelity increasing as time passes.

Systems engineering research has continually tried to keep pace with the growing complexity of the systems it is used to design by collectively focusing on one or more of the common underlying themes listed above. However, as systems' complexity grows, the tools employed for their development have to address a much wider set of challenges than the traditional systems engineering was developed for, including considering social and political phenomena in design organizations. In this chapter, we will discuss various methods in systems engineering research by classifying such them into a number of themes identified by the research community. Alongside, we will discuss the further advancements that can be done to the modeling of design processes.

The rest of this chapter begins with a discussion of various studies on complex systems design process modeling in section 2.1, followed by a discussion of currently recognized challenges and proposed future research directions 2.2. Section 2.3 briefly discusses decision theory, which provides some useful contributions to the modeling within this dissertation. Section 2.4 discusses methods for selection of a design process

policy. Finally, section 2.5 concludes this chapter by highlighting the contributions that this dissertation will make to complex systems design process modeling.

2.1 Research on Modeling Complex Systems Design Processes

In a recent survey of literature, Wynn and Clarkson proposed *scope* and *type* as the two dimensions for categorizing process models for design and development of engineered systems [13]. Along the former dimension are the micro-, meso-, and macro-level models depending on their breadth of coverage in design context, while along the latter dimension are divisions based on the overall purpose into four categories of procedural, analytical, abstract, and management science / operations research (MS/OR); Fig. 2.1 summarizes these dimensions. Broadly speaking, the scope dimension spans from low-level models which focus on individual steps to the high-level models which focus on entire project structures and the context. For example, the Function-Behavior-Structure (FBS) framework [14] fits within the category of micro-level abstract models, particularly because without resorting to mathematical formulation it suggests that all designs can be represented in terms of their functions, which describe what the design is for, behaviors, which describe what it does, and structures, which describe what it is. At the meso level, for example, are the task precedence models such as PERT and task dependency models such as the design structure matrix (DSM). Agent-based models such as the virtual design team (VDT) [15] also appear at this level within the analytical type category. Finally, at the macro level are models such as the integrated product and process approaches within the procedural type dimension, or system dynamics models within the analytical dimension.

Two reasons can be used to justify the large number of design process models and the need to further develop more of them: one, these models are invariably an abstraction of the context that they are intended to be used in, and two, their form is influenced by the modeler themselves, particularly with regards to the research

	Procedural	Analytical	Abstract	MS / OR
Micro	Provide prescriptive guidelines for the design and problem-solving activity throughout a project	Provide formalisms to assist in the modelling of design knowledge from a process perspective	Concerns domain-dependent insights on forms of reasoning, elementary activities, and/or types, structures, and evolutions of knowledge that occur during design.	Concerns models which apply mathematical or computer analysis to generate general insights from representative or synthetic situations.
Meso	Aim to support the effective generation of good designs by prescribing a systematic design process.	Concern with the specific steps that do or should occur within a company and/or design context.	Provide conceptual frameworks for understanding how meso-level process flows, or models of them, relate to the design's progression.	Provide mathematical or computational tools for research in which representative or synthetic cases are analyzed to extract general insights.
Macro	Provide graphical depictions and explanations of the contextual issues that need to be addressed during design, or prescribe management structures and philosophies.	Investigate and address the impact of a process' context.	Focus on clarifying the design process' overall form and how design processes interact with their context.	Concerns computational or mathematical studies of factors governing processes on the macro-level.

Figure 2.1.: Classification of complex systems design research [13].

question that the modeler poses. Like the systems themselves, the design processes of complex systems also show significant elements of novelty, complexity, and iteration. Consequently, the task of process models is to organize both the creative work done at any stage, and especially the initial stages, alongside the more routine activities that individuals involved in development have to accomplish. The payoff to be obtained with an appropriate choice of models is that organizations can potentially streamline planning and information exchange required during the process.

Yet, while models can help in analysis of design situations for which they are setup and under the assumptions that they make, in actual practice models, where reliable ones exist, may have limited utility especially in the early stages of design, where designers have to balance exploration of new alternatives with exploitation of available information to further refine the already developed alternatives. The

designers are also confined to using the tools and models provided to them by their organization, which in some cases may not be appropriate for their context or need. This is why there is a need to examine the properties of network of interactions between the models and their users and how these properties can affect the utility of such models [13].

The interactions between multiple teams can be examined by representing a design organization as a kind of distributed network, with the agents in this network being self-interested and who respond to local incentives. Klein et al. [16] discussed the important properties of collaborative design dynamics, stating that, "complex systems design then becomes a collaborative activity done by self-interested agents who interact with one another and behave in accordance with their local objectives." In particular, a central focus of complex systems research is the dynamics of distributed networks, i.e., networks in which there is no centralized controller and global behavior emerges solely as a result of concurrent local actions, whereas a central concern of negotiation research is designing the rules of encounter between interdependent nodes such that each node is individually incentivized to make decisions that maximize social welfare, i.e., maximize the global utility of the collected set of local decisions. Information systems are increasingly becoming the medium by which design participants interact, and this fact can be exploited to help monitor the influence relationships between them.

In the work done in this dissertation, we discuss a framework which models the subsystem teams as agents who have their local objectives and a system-level manager whose task is to formulate incentives and heuristics to guide the teams' decisions in favor of the system-level objectives. Thus this work addresses both the dynamics of distributed networks of self-interested design teams and the negotiation among them and across levels of organizational hierarchy in form of incentives exchange. Using the classification scheme in Fig. 2.1, work presented in this dissertation falls within the meso-level analytical type because it focuses on end-to-end flow of tasks and provides insights based on the particular situation of an organization comprised

of semi-autonomous teams which interact to produce the final system design; we will further discuss the context and assumptions made in a later section. However, there is a broader scope to the proposed framework because it crosses the meso-level dimension to include context in form of organizational structure and the managerial issue of guiding the decisions of self-interested teams, and both these latter tasks are found at the macro-level dimension.

2.2 Recognized Challenges to Complex Systems Engineering

In 2010, a series of workshops on engineering design identified a number of open research questions still facing the community [2–4]. For example, in the workshop on design of complex systems conducted by the NSF [3], the community recognized that an ability to effectively design complex systems will give us many high quality and technologically advanced systems. Further, the real payoff would be more on the methods, tools, and processes, especially those that can be done away with. In addition to development of tools and methods is the recognition that of the five identified overarching themes that are necessary for advancement of the multidisciplinary optimization theory, one is the need to put humans "back in the loop" [4], because while computational analysis capabilities are getting increasingly more sophisticated, human decision-making is still important, especially for conceptual design. Figure 2.2 shows the five areas of future research including research on organizational modeling, uncertainty and decision-making, and metrics that this workshop highlighted. These workshops also identified the need for more research and development on topics such as decomposition and organization of information flow during the design process. A joint NSF and NASA workshop on design of complex engineered systems identified that the development process of such systems needs as much attention as the systems themselves [2], further stating that while the physics is generally well modeled, the social, economic, and political interactions need further research.

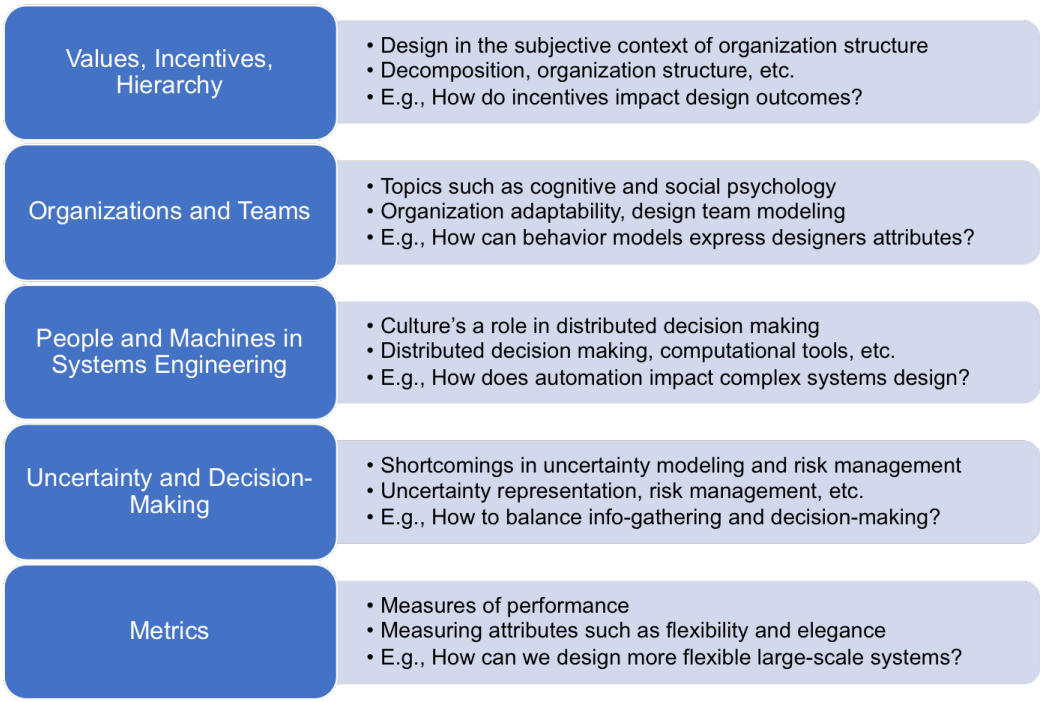

Figure 2.2.: Proposed areas of research to advance complex systems design [3].

More than one of the recognized research areas are related to role of organizational structure and design team modeling within the complex system design process. Examples of questions in the context of organizational modeling include, "How do two or more groups share and update their knowledge (tacit or explicit) when they reside in different organizational systems?," "How does one group incentivize another to cooperate?," and "How can we arrange for information flow and collaborative learning from one group to another without control of the information becoming an organizational objective? [2]" Clearly there is ample potential for further research and advancement in the development of complex systems.

For the rest of this section, we will use the first three of these areas of research to discuss literature spanning the space of systems engineering research and link the works cited to the classification scheme discussed in the previous section. The fourth area of uncertainty and decision-making discusses the shortcomings of systems engineering models in representing uncertainty and risk management, including such

questions as to how to model different types of uncertainties in design, how uncertainties propagate through system hierarchy, and how do we balance gathering more information with the need to make decisions with already available information. The latter question regarding balance between information gathering and using is the decision of exploration of design space versus exploitation of already available information; we will discuss more about exploration versus exploitation later in the dissertation.

The fifth area, metrics, looks at quantification of the measures of performance of the system, including measuring attributes which are traditionally difficult to objectify, for example, elegance and flexibility. As an example, the definition of an elegant design states that it has the four properties of effectiveness, i.e., it works, robustness, efficiency, and that it minimizes unintended consequences [17].

Note that while we attempt to classify literature into one of the three areas below, some of it may span more than one area. Nonetheless, each of the following cited works relates to some phase of design process modeling, and our primary objective is to discuss how their ideas can be extended towards our development of process modeling framework proposed in this dissertation.

2.2.1 Values, Incentives, and Hierarchy

This area of research places design in the subjective context of organization structure, and includes questions on how decomposition, selection of contracting procedures, etc., affect design outcomes.

One of the characteristics of complex systems design that we stated above is that it is done by a multiple teams acting semi-autonomously and in coordination with one another. The division of design tasks among the teams requires the decomposition of design problem into smaller subproblems which can then be solved in parallel by separate teams. Careful decomposition of the design problem is essential because it affects how information flows in the organization, dependent upon the rules of interaction among design teams; later in this section we will discuss studies which look at the

effects of rules of information exchange on design outcomes. In brief, the interaction networks have an influence on the product architecture because they constraint the flow of information in the organization, limiting the space of solutions that the organization can explore. Naturally, the formation of teams would be influenced by how the problem is decomposed, which makes understanding decomposition strategies an imperative.

Wagner discussed four forms of decomposition – object, aspect, sequential, and model-based decomposition [18]. Object decomposition is wherein the system is divided into subsystems which are to be integrated, thereby requiring system modularity. For example, object decomposition in case of an aircraft would be divisions into teams which develop engines, wings, fuselage, etc. Aspect decomposition is that which is done by disciplinary specialties, for example into teams of aerodynamics, propulsion, structures, etc. Finally, sequential decomposition is in chronological order of design tasks, and model-based decomposition is by the models that are used during the design process. In aerospace design, it is not unusual for the organizational hierarchy to follow disciplinary lines, which in above terminology is aspect decomposition [19], and the requirements allocation mirror disciplinary boundaries.

Many different methods have been proposed for decomposition in literature. For example, Michelena and Papalambros [20] presented a method for decomposition using hypergraph partition of the problem into model-based components. Representing the design problem as a hypergraph, which is a generalization of a graph where a single edge can connect multiple nodes, they used spectral graph-partitioning methods to decompose the problem. As another example, Kusiak and Wang [21] used incidence matrices representing relationships between tasks and design parameters to partition the design process into clusters of tasks which can be completed in parallel. In a later section we will look at other studies which have used these incidence matrices, called design structure matrices, in the modeling of design processes.

On the issues of value and incentives, we can discuss the differences that may arise while taking a value-based versus optimization-based approach to design because such

a distinction between approaches could be useful in analyzing how the designers' attitudes plays a role in their decisions. For example, Collopy [22] discussed modeling of designers' behavior stating that the designers' tolerance to risk decreases once the specified requirements are met, thereby inhibiting further search for better solutions. When operating in constrained space, performance and cost are traded off with other system attributes, with the result usually being a decrease in performance along with an increase in cost. In contrast, in an optimization-based approach, defining a scalar objective function converts the teams' extensive attributes to a score on which the teams select the alternative with highest value. This objective function inherently contains the system trade-offs and represents the teams' utility for each of the alternatives. In a value-driven design paradigm, this function would be expressed in monetary units and would be the measure against which, besides system alternatives, the tools, processes, and methods can be assessed. Thus, the benefits of value-driven design include that it enables optimization, prevents design trade conflicts, and avoids cost growth and performance erosion [23].

While the paradigm of value-driven design is useful because it brings together economic theories and systems engineering principles and provides decision-makers with an objective basis to optimize the system [23], it does not provide the designers with any directive, and rather the objective functions flow down to each team. To use the value-driven paradigm for guiding design processes, we have to incorporate it within the design problem itself, which we discuss next.

In the previous chapter we briefly contrasted between a product-view and process-view of complex systems development; these views are based on the notion of value-driven decision-making that is artifact-, process-, and organization-focused as proposed by Lee and Paredis [8]. A value-driven approach to design states that all design is the process of maximizing value of a product, and this value can be given as:

$$\mathcal{A} : \max_{a \in A} \pi_A(a) \tag{2.1}$$

Here, the designer seeks to maximize the profit π_A of a system a, chosen from the set of alternatives $\mathcal{A}$. In this form, design amounts to being a selection problem, where the objective is to select an alternative which provides the highest utility to the user. The challenge facing all systems development programs is that of identifying a feasible design space and then searching for the best alternative from within it. Optimization algorithms fulfill this role by searching for the optimal value of the objective function over the space of feasible designs. These algorithms differ in the heuristics they employ for search depending on the features of these spaces, which is the reason not all algorithms are suitable for all problems. In other words, the process of search over the design space affects the outcome of optimization algorithm.

Thus, we can extend the above selection problem to state that all designs are outcomes of a selected process, so that, in order to maximize the value of the product, we need to maximize the value of the process defined as a function of the product:

$$\mathcal{A} : \max_{a \in A} \pi_A(a, t(\mathcal{A})) - C(\mathcal{A}) \tag{2.2}$$

In this equation, the value is a function of both the chosen alternative, and a time component, $t(\mathcal{A})$, which will take into account the temporal nature of the process. This value is reduced if the cost of this process, $C(\mathcal{A})$ is high. Reference [8] avoids the self-referential nature of the above objective function by re-framing it as:

$$\mathcal{P} : \max_{p \in P} \pi_P(a(p), t_p(p)) - C_p(p) \tag{2.3}$$

Here, $\mathcal{P}$ is the set of decisions that the designers can take, and the objective is to maximize the value of the process-system, π_P, which is a function of chosen alternative, a time component, and the cost of a particular set of decisions, p, called a policy. C_p specifies the cost of these decisions. This formulation provides the motivation for attempting to maximize the value of the process. The result of foregoing discussion is that organizations have an incentive to set up appropriate processes at the beginning of all complex system design programs.

We can further develop this line of thought to account for the semi-autonomous nature of teams involved in design. Being semi-autonomous means that the system-

level designer has no direct authority over the teams' decision-making, and can influence their decision-making by providing them with appropriate incentives, which could take the form of monetary rewards, as an example. Discussion of incentives comes from mechanism design theory, which is the reverse of game theory. Being semi-autonomous, the teams have local value functions and make decisions to maximize those, sometimes to the detriment of the system-level value function. Because for an organization the ultimate goal is to maximize its own value, in the absence of any direct authority, the organization's value depends on how it devises an incentive scheme i from the set of available options I:

$$\mathcal{O} : \max_{i \in I} \pi_O = \max_{i \in I} \pi_A(a(p(i)), t_p(p(i))) - C_i(i) \tag{2.4}$$

This equation is a modification of equation 2.3 to make process as a function of an incentive scheme, and this means that the task before the organization is to select an incentive scheme that ultimately leads to maximization of its value, π_O. Note that in this discussion we have used notation consistent with reference [8]; our use of notation is different which we will define in the next chapter.

An example of related work includes that by Cheung et al. [24] who discussed the application of value-driven design approach for the design of a jet engine. Collopy [25] presented a survey of value models in aerospace systems development. In this dissertation, we take a value-based view of design and aim to select a design process policy which maximizes the organization's value function. Also, in this dissertation, setting incentives amounts to the organization's selection of heuristics for control of design teams. We will do this using Principal-Agent theory, discussed further in section 3.5.

2.2.2 Organizations and Teams

This area of research raises questions on topics such as organization adaptability, design team modeling, etc. and extends design research into topics such as cognitive and social psychology.

An hypothesis called the "mirroring hypothesis," also called the Conway's law, states that organizational structure both influences and is influenced by the architecture of the product being designed [9, 26]. We noted earlier that with growing system complexity, the number of teams and even organizations involved in their development increase, leading to personnel and information management challenges. If the network of interactions among teams affects system design it will ultimately affect the performance of the final system produced. This effect gives rise to a need to study the interplay between organizational structures and system architectures. Large organizations which are structured hierarchically can be represented using tree diagrams because such representation are easy to study. However, real communication networks within organizations are seldom such cleanly structured and real people interactions are messier.

Flynn [6] studied organizational structure considering both the formal and informal interactions, and claimed that "an all encompassing analysis of the organization is possible and would result in useful output that would allow managers to make deliberate and effective design decisions in the continuous improvement process." In his modeling he made use of the methods of social network analysis and group modeling techniques. The model served as a mapping tool useful for identifying the various formal and informal relationships that exist within the organization though it did not provide the ability for any performance analysis. In the classification scheme we discussed in section 2.1, this work falls within the macro level analytic type; its lack of ability to provide guidance for organization design is the reason it would not be within the procedural type category.

In contrast, the viable systems model (VSM) prescribes a way to structure an organization for improved efficiency. As opposed to the usual hierarchical structural representation of an organization which relies on strict authority, this model presents a more organic approach which enables an organization to be flexible and democratic. This model represents an organization using five layers inspired by the human brain. At the lowest level are the individual designers who perform design activities. There-

after, at each successively higher level are those that perform conflict resolution and stabilization activities, internal regulation and optimization activities, adapting and forward planning activities, and command and policy determination activities. This is an example of macro level procedural type model.

Jin and Levitt [15] presented a discrete event model of organizational design teams called the Virtual Design Team (VDT) model. Building on organizational contingency theory, VDT modeled actors, their activities, communication tools, and organizational structures. The model made use of a simulation approach to study such questions as how does the decentralization of decision-making impact project performance, or how does reciprocal interdependency between two design teams affect their workload? It split all tasks into two parts, viz., production work which adds value to the system being developed, and coordination work which facilitates the completion of production work. Having done this, the two requirements of the VDT model were to, first, capture both the work contents and activity dependencies, and second, to be able to map the computational model to real world data so that insights from the simulation can provide practical utility. Like VSM, this model falls within the macro level but in the MS/OR category because it provides agents models of an organization for simulation and study.

The above discussion is on studies of how organizational structures can be represented and how decision-making within these structures can be modeled; let us now look at how information flow within organizations can be handled. With multiple autonomous teams working independently and in parallel, the information they generate and share with other teams is certain to contain inaccuracies and, especially in early stages of design, be of lower fidelity in nature. Aggregation and a "cleaning up" of such information becomes imperative before it is used to inform their decision-making. Predd [27] proposed a "scalable aggregating algorithm (SAA)" for aggregating the information from incoherent or abstaining human judges, in this case experts. This can be employed for cases where teams exchange information but provide either biased or incomplete information. Then using partially available information, aggregated data

can be generated, helping speed up the process of design. In the work in this dissertation, we will set up design agents to evaluate information available from all sources independently and then select the one which provides highest utility to the team. This approach is more useful in the early stages of design exploration. Aggregating disparate information into a single dataset may, however, be an alternative approach, especially in later stages of design when available information is being refined.

Lastly, two further aspects come under the purview of this topic area. Alexandrov and Lewis [28, 29] proposed a method to make the multidisciplinary problem itself reconfigurable in a bottom-up manner, where, using strategies such as lexical analysis and grammatical analysis, computational components of a design problem can be identified and assembled into various different MDO formulations and solution algorithms. Practically this means that the designers can specify the inputs and outputs of their disciplines, and the method would then suggest alternative formulations of the design and optimization problem to the user. Their proposed method is intended to ease system synthesis and integration by automatically performing tasks such as error checking, formulation and reformulation of optimization problem formulation, etc.

Finally, in keeping with the optimization community's recognition of inclusion of humans within the design process modeling, it would be useful to model human behavior in the context of organizational structure. Wise et al. [30] categorized the various approaches to modeling human behavior along the three dimensions which indicate the depth and flexibility of behaviors, the temporal dimension from reacting to current situations to anticipating and planning for future scenarios, and complexity of the models themselves (see Fig. 2.3). Studies and application areas with regards to human behavior modeling can be then mapped to these three dimensions. Despite the breadth of surveyed methods, employing quantitative approaches requires making some abstractions, even for approaches narrowly focused in a context. These abstractions along with the measures of performance are a result of the researcher's choice and therefore implicitly incorporate bias. Later, in Chapter 6 we will discuss using

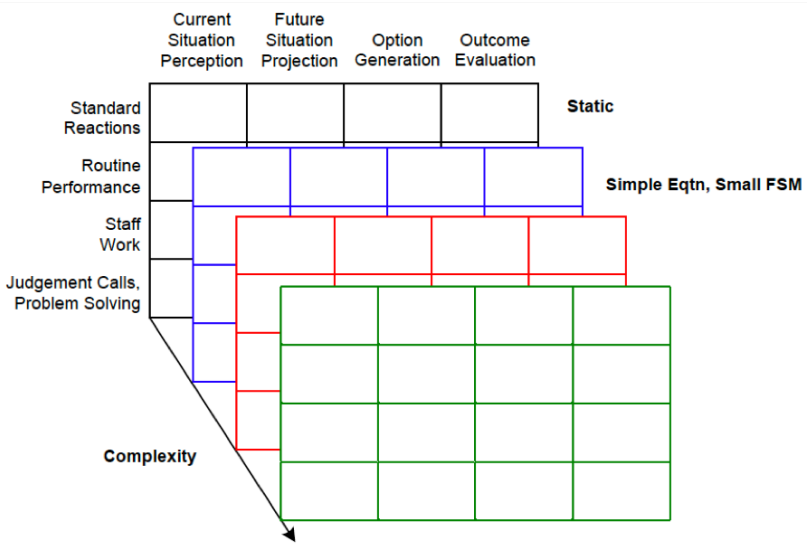

Figure 2.3.: The three dimensions of human behavior modeling (from Ref. [30]).

qualitative approaches, which provide dense descriptions of phenomena as opposed to the highly abstracted quantitative approaches and work as complements to the quantitative approaches.

Clearly research in the organization of design teams is vast with potential for many more questions to be asked and answered. In this dissertation, we look at a model of an organization where the design teams adaptively make decisions using information available at any given point in the process. The design process arrives at a final system architecture as a result of the collective decisions of the teams guided by the incentives provided by the system-level manager. Like the studies by Flynn [6] and Jin and Levitt [15] above, the interaction network within the organization will evolve as teams make decisions on which other team to interact with at any time. In other words, the interaction network within an organization for a system design process is an outcome of the framework presented in this dissertation rather than an input to the modeling.

2.2.3 People and Machines in Systems Engineering

This area addresses questions relating to how culture plays a role in distributed decision making and how computational tools share decision-making responsibilities with the humans.

Each team involved in a complex systems development project identifies its value functions and behaves in accordance with them. The local value functions take precedence over global objectives at the team-level of an organization with the global objectives more likely to be satisfied when they provide a higher value to the teams. It is the system-level designer's task to provide not just a set of requirements but also incentives which coax the teams to make decisions in favor of the system. The set of well-formulated incentives cause the teams' value functions to align closer to that of the system-level objectives. The challenge for the system-level designer is that he does not know of these value functions and can only form incentives based on his observations of teams decision outcomes. Under the setup just discussed, requirements flow downwards while the objectives flow upwards. This problem of guiding teams working independently under guidance of a system-level manager is studied in Principal-Agent theory where the principal can observe the outcomes of agents' decisions but not the workings of their internal decision-making.

Vermillion and Malak [31] presented a study in which they made use of the Principal-Agent theory to solve a problem of task delegation. In their framework they incorporated models of human behavior including their risk profiles and studied the effects of two different incentive schemes on design outcomes using an example of vehicle engineering. In the work in this dissertation, we will not study a principal's task delegation because we assume the teams decide their set of tasks locally and select from among them. Rather, in the principal-team interaction, the principal will setup rewards to be given to the teams while the teams will report design variable information. We will further discuss Principal-Agent models as used in this work in a later section (see section 3.5). Lee and Paredis [8] have also proposed us-

ing Principal-Agent models in cases of distributed system design to answer questions such as "How should we assign authority and responsibility?" and "How should we provide incentives?"

Disregarding informal interactions between designers, teams involved in distributed design interact with either the other teams or the system-level designer subject to the rules of communication. The nature of problem decomposition, for example, could dictate the communication links within an organization. Braha and Bar-Yam [32] developed a model of an organization as a network whose topology changes with time due to evolving information exchange requirements. They modeled the nodes to represent tasks and the links to represent information flow among the tasks, and the existence of a link was a function of time with a certain predetermined probability function. They studied a design process from the perspective of network theory and did not account for human behavioral attributes when drawing conclusions. However, considering both the human behavioral and cultural aspects of design would be a factor with direct relevance on the rules set up for information exchange among designers.

While the network of communication links within the organization could be based on factors such as problem decomposition, the designers may interact informally with members of other teams, especially in cases where the teams are collocated and personal interactions among members are possible. Such informal interactions allow for rapid exchange of information among designers and could foster innovation in an organization. We can conclude, therefore, that what information flows on the communication links, whether formal or informal, has an effect on the design outcome. Some relevant questions to consider for the purpose of setting up communication rules include:

1. What communication links or rules of information exchange should exist? Who should be involved in every exchange of information?

2. How should the incentives (rewards) for meeting objectives be set up?

In a study on the effect of communication within an organization on design, Honda et al. [33] evaluated the effects of different information passing strategies on the final outcome in a multi-level design problem. They made use of a game theoretic and a modified game theoretic framework in a bi-level system design problem and compared the results of both approaches to coordination among teams. They arrived at a number of conclusions: the choice of system variables affects the optimality of design outcomes, perfect information passing may not decrease the number of iterations, and the design cycle stability depends on the amount of information passed. Similarly, Ciucci et al. [34] investigated the effects of passing different types of information on the ability of subsystem design teams to converge on Pareto-optimal designs. They found that passing more information generally results in converging on Pareto-optimal sets even though the cost of doing so increases.

To reiterate an earlier point, design process models are setup to aid designers' decision-making. Before moving to a discussion of decision-making in the next section, let us discuss some studies on how computational methods can help model human decisions. The following two studies are based on the dynamic programming approach. Boukhtouta and Powell [35] proposed a dynamic programming based approach for coordination among multiple independently acting agents. Their framework solved a problem of resource distribution in a complex network with multiple autonomously acting agents. These agents exchanged information with one another, sometimes with distortion, which could lead to suboptimal utilization of resources. They presented an algorithm wherein the agents adaptively learn and modify their communication behavior so as to improve the final allocation outcome. They also addressed features of such a coordination problem as possibility of misinformation during communication, and showed that the benefits of communication include that final solution approaches an optimal value that could be found if the problem were to be solved by a single agent controlling all variables.

Fang et al. [36] proposed an approximate dynamic programming (ADP) approach combined with a transfer contract mechanism approach via a value function approx-

imation to solve the problem of architecture selection for a system-of-systems. In an acknowledged SoS, where the stakeholders have to share resources to provide capabilities but without the direct authority of a central controller, the transfer contract mechanism provides an internal market for the participants to share products and services with one another. The proposed ADP model extended decision-making to multiple stages over a long-term time horizon. The use of an approximate value functions avoided the necessity of collecting complete information for an optimization problem. The authors, like in the previous study, also showed the benefits of communication and adaptation on the final solution outcome.

Renner and Schmedders [37] discussed how a dynamic principal-agent problem can be solved using the dynamic programming method of value iteration together with a polynomial approximation technique. They modeled continuous choices sets for both the principal and the agent and transformed the bi-level optimization problem to a standard nonlinear program, and then presented a recursive solution technique.

Olewnik et al. [38] proposed a framework which combines concepts from multi-objective optimization, consumer choice theory, and utility theory to develop systems which can adapt their functionality to changing requirements. In their approach, they sought to maximize corporate utility, thereby looking beyond just product level optimization, when selecting a system architecture. The role of designers' subjective decisions was recognized by the fact that decision makers need to specify the points on Pareto front. Their work falls into a category of approaches which propose frameworks for decision support for complex systems design.

Our proposed framework, discussed in the next chapter, will make use of dynamic programming, particularly the value iteration approach, to model agent decision-making. The organization is setup as a network of interacting agents where links between two agents exists when they share variables. The agents then use value iteration to calculate the discounted future rewards and make decisions from which the system design evolves. Before proceeding, we briefly discuss the basics of decision

theory. The concepts discussed next are relevant to selecting such features of our modeling such as risk behavior of designers, utility functions for the teams, etc.

2.3 A Brief Discussion of Decision Theory

Decision theory is the study of an agent's choices, and it is concerned with goal-directed behavior in the presence of options. It is studied in many fields including economics, statistics, psychology, political science, etc. Whereas game theory addresses the question of how do interactions affect agents decisions?, decision theory studies how individual agents make choices? There are two "kinds" of decision theory, viz., *normative*, which is theory of how decisions should be made, and *descriptive*, which is the theory of how decisions are actually made.

The decisions that an agent makes depends on its abilities, its observations, and its preferences. A *rational* agent will make decisions based on its preferences over the possible outcomes of its actions. An agent's rationality, in turn, depends on the axioms given in Table 2.2. In the notation of this table, "o_1," "o_2," etc., indicate the outcomes as a result of agents' decisions, and these axioms relate to how the agents make a decision by evaluating the expected outcomes and choosing the one which leads to the outcome over which it has the highest preference.

For the purpose of modeling, we need a convenient way of providing information about the preferences, and this comes in the form of a value function. This function assigns a real number to each of the possible outcomes called its *value*. Then combining the values with probabilities of occurrence of outcomes, we get a probability-weighted value for each possible state of the world. This linear sum of probability-weighted values is called the *expected utility* or simply *utility* and a rational agent will pick the option which gives it the highest utility. At this point, it will be useful to distinguish and clearly define the terms to be used in our discussion – the following definitions are based on those given by Collopy [25]. A *value* is a numerical encoding of preference, such that, saying that an agent prefers outcome o_1 over o_2 is equivalent to saying that

Table 2.2.: Axioms of Rationality

Completeness	An agent has preferences between all pairs of outcomes: $o_1 \geq o_2$ or $o_2 \geq o_1$
Transitivity	Preferences must be transitive: if $o_1 \geq o_2$ and $o_2 \geq o_3$ then $o_1 \geq o_3$
Monotonicity	An agent prefers a larger chance of getting a better outcome than a smaller chance of getting the better outcome. That is, if $o_1 \geq o_2$ and $p \geq q$, then $[p : o_1, (1-p) : o_2] \geq [q : o_1, (1-q) : o_2]$, where p and q are probabilities.
Decomposability	An agent is indifferent between lotteries that have the same probabilities over the same outcomes.
Continuity	Suppose $o_1 > o_2$ and $o_1 > o_2$, then there exists a $p \in [0, 1]$ such that o_2 $[p : o_1, (1-p) : o_3]$
Substitutability	If $o_1 \geq o_2$ then the agent weakly prefers lotteries that contain o_1 instead of o_2, everything else being equal. That is, for any number p and outcome o_3: $[p : o_1, (1-p) : o_3] \geq [p : o_2, (1-p) : o_3]$

the value of outcome o_1 is greater than that of o_2. *Utility*, on the other hand, is as we defined above – a linear weighted sum of values of each of the possible outcomes multiplied by their associated probabilities of occurrence. The concept of utilities is central to the theory of decision-making under uncertainty, and in our approach of value-based decision-making, we will evaluate the expected utility of each decision that an agent can make.

While it is useful to assign a single numerical value to each possible state, the actual value functions differ from person to person. Behavioral attributes such as risk-taking ability play a role in an agent's value function. For example, Fig. 2.4

distinguishes between the risk-taking behavior of different agents by showing three types of agents: the risk averse, risk prone, and risk neutral. Other theories such as Prospect Theory have suggested that agents are more sensitive to losses than to gains in value, and thus, the risk profiles are not symmetric between the two; for further information see reference [39] and a brief discussion in reference [31]. Of relevance to our discussion is the fact that the choice of value functions will affect agents' decision-making which, in turn, will influence not just the final design outcomes but also whether an organization can successfully develop systems within the cost and time budgets.

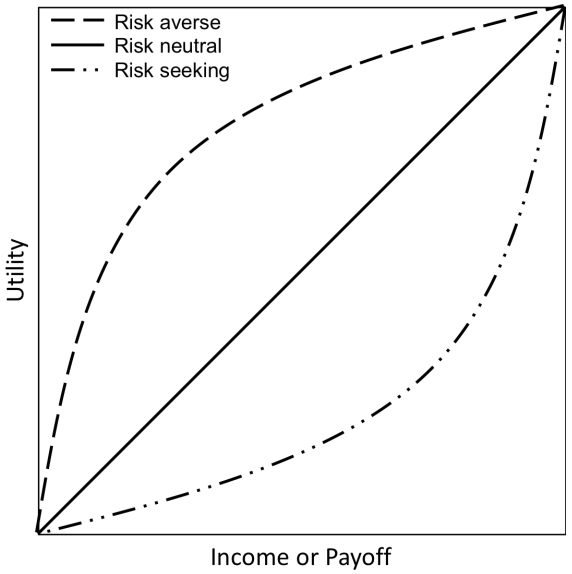

Figure 2.4.: Different types of risk profiles: The concave curve is for a risk averse agent, the straight line is risk neutral, and the convex curve is risk prone.

As stated above, a rational agent will select a decision to maximize its expected utility. Suppose O is a possible set of outcomes specifying a state of the world. Then the choice to select an expected utility maximizing decision can be mathematically this can be expressed as $\mathcal{E}(u|d \in D) = max \sum u(\omega) * P(\omega)$, where d are the decisions that an agent makes from the possible set of decisions D, and u and P are the utility function and the probabilities of occurrence of a specific outcome ω, respectively.

If the agent had to make only a single decision, and assuming that the probabilities of outcomes can be calculated, the agent would search over its decision space and select the set of decisions which leads to maximum expected utility. In general, however, the agent would have to make a decision based on its current observation and available options, then observe the effects of the selected action, make another decision, and so on, in what is called a sequential decision problem. A *policy* is a mapping from states to actions, meaning that it specifies what the agent should do in each of that states that it visits. Similar to maximizing the utility of a decision, we can say that a rational agent will select a policy with maximum expected value, and this can be written as $\mathcal{E}(u|\pi) = max \sum u(\omega) * P(\omega)$, where π stands for the agent's selected policy and all other variables mean the same as before.

The above was a brief discussion of decision processes and utility theory; for further details see Refs. [40–42]. In the next chapter, we will use discussion from this section to elaborate on how a design process can be represented as a Markov decision process and how an organization can select a policy which provides it with the highest value. In the next section, we discuss some of the current literature on selection of design process policies.

2.4 Selecting a Design Process Policy

One way to study team interactions is to develop models of systems and the constituent subsystems being designed and use the common design variables among the subsystem models as a surrogate for team interactions. The design structure matrix (DSM) is a common representation employed in systems engineering which takes this approach. DSM lays out the subsystems or tasks along the first row and column of a matrix and indicates interdependencies between them by marking the cells of the matrix. Figure 2.5 shows an example of a DSM where all design tasks are listed in the first row and column. The off-diagonal cells indicate interactions among tasks. The cells in the upper triangular region indicate feedback dependencies while

	Task 1	Task 2	Task 3	Task 4	Task 5	Task 6
Task 1		X				X
Task 2			X	X		
Task 3		X			X	
Task 4		X				
Task 5				X		
Task 6	X					

Figure 2.5.: A design structure matrix.

those in the lower triangular region are feed-forward dependencies. The schematic shown here shows binary links with an '×' indicating a dependency while an empty cell shows a lack thereof; alternatively these dependencies can take numerical values to show their strength.

A proposed extension to the DSM, called extended design structure matrix, or XDSM, builds on the DSM's capabilities of showing interconnections among system components, by adding the ability to show data dependencies and process flow [43]. The XDSM has been used to represent the solution strategies of all existing MDO architectures. Figure 2.6 shows the XDSM for the Gauss-Seidel MDO algorithm for a three subsystem problem [43]. Other approaches to sequencing design tasks have made use of approaches such combining simulated annealing optimization algorithm which is a meta-heuristic approach with risk analysis [44].

A tool called Design Manager's Aid for Intelligent Decomposition (DeMAID), released in 1989, provides capabilities such as reducing feedback couplings in complex systems design, sequencing the design processes, grouping processes into iterative sub-cycles, etc. The utility of such a tool is that once designers understand the flow

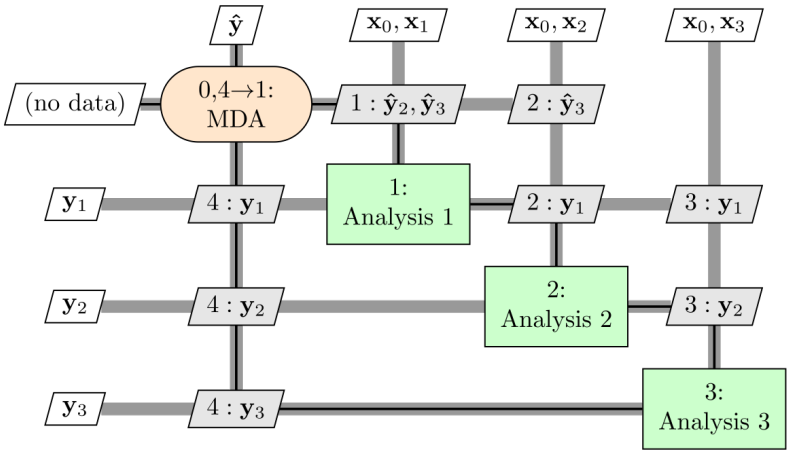

Figure 2.6.: The extended design structure Matrix from [43].

of design processes and effects of iterative sub-cycles, the time and cost of systems development can be better managed [45]. This tool makes use of the DSM for representing coupling between tasks, and makes use of strengths of these coupling to determine the best sequence of tasks. Later enhancements made use of time and cost information along with use of genetic algorithm approach to optimize the design sequence.

Other approaches to design process sequencing have also made use of the DSM, combining it with either simulation technique [46] or by the use of heuristics to find subproblems [47]. In the former study, Cho and Eppinger, used a simulation based approach for modeling and analysis of design processes wherein they used a DSM to capture information flows between tasks. They captured features such as information transfer patterns, uncertain task durations, resource conflicts, overlapping and sequential iterations, and task concurrency. Using various measures that represent realistic features of design processes, they proposed a method for project planning and control including sequencing of tasks iteratively. These measures included those such as rework probability, which indicates the probability that a task has to do rework when it is affected by another task that it is interdependent on, and rework impact, which is a measure of level of interdependency between tasks in sequential iteration.

On the other hand, Qian et al. [47] developed a new hybrid optimization approach for a large DSM-based activity sequencing problem with the objective of minimizing the sum of superdiagonal numbers. They established several simple rules for reducing the sum of superdiagonal numbers in a DSM and proposed a heuristic for obtaining good feasible solutions.

2.5 Summary

In summary, we reiterate some of the future research questions that still need to be addressed for complex systems design. First, we need to account for social and human factors and how they affect the design process. The key impact that such factors have are that the interactions in an organization are messy and often informal as opposed to the strict formality and structure of a tree-type hierarchy, and that this leads to inaccuracies in data exchange whether it is due to designer biases or their intention to protect their objectives. Next, we need research on how design tools and methods are actually utilized in an organization [13]. In other words, we need to study how the models map to reality and how they can improve an organization's performance. Lastly, we need to study how design process can be identified which are optimal in the given context. This includes elucidating the set of tasks to accomplish and the order in which they should be done, their allocation to teams, etc.

The work in this thesis will address the last of these research questions, i.e., how do we identify design processes suitable to the context. Rather than provide a prescriptive theory, we will set up a simulation framework of distributed, semi-autonomous design teams' whose collective decision-making will result in a design outcome. The framework will be set up with agent models of the teams, and from simulations using framework, we will be able to record team interactions and the sequence of their decisions. In the next chapter, we discuss the background theory used in our modeling.

3. DESIGNING THE DESIGN PROCESS

In this chapter we will define the complex systems design problem and discuss the context of the problem along with the assumptions we make. In section 3.2, we discuss the agent model that we will use for both design teams and the system-level designer. In sections 3.3 and 3.4 we cover how the design teams assign values to design tasks and how they use the value iteration algorithm to select the best policy, respectively. Following this, in section 3.5, we discuss the principal-agent model, which the system-level designer uses to select heuristics for guiding teams' decisions.

3.1 Problem Context and Definition

First, we define some terminology used in our discussion below. The design space of a system can be defined using ranges of values of attributes that specify that system, and each instantiation of attribute values defines an alternative design. From Collopy and Hollingsworth [23], we distinguish between attributes and design variables – attributes are such things as weight of the system, its cost, etc. which are used to connect to the customer, whereas the design variables are properties of the system itself such as length of a component, the diameter of a cutout, etc. In this work, we assume that customer requirements are already specified and the modeling in this work is concerned with the design teams' decisions, and hence, we will hereafter use the term *variables* to describe the system properties that the teams control.

A *design process* is the task of searching through the space of feasible alternatives and selecting the best from among them. Thus, the design process proceeds through a series of transitions within the space of feasible alternatives, so that in the process view of design, each alternative can be considered to be one feasible state of the

process and the many alternatives generated and evaluated on the way to a final design are the various states explored.

A feature of complex systems development is that for such systems the design problem is decomposed into several subproblems so as to generate multiple design subspaces which are subsets of system attributes and which can then be explored in parallel by different design teams. In other words, in complex systems design, each of the subsystem teams controls a subset of all design variables, some of which are shared among multiple teams. When in a given state, the teams may choose to further *explore* the design space by changing variable values in search for better alternatives or *exploit* their current knowledge to refine the design they have already. They could work with local information or communicate with other subsystem design teams to get more information. In this way, an information-theoretic view of design considers each action that a team takes as an information processing unit in which the designers observe the current state information and act on it to produce an output which is the next state reached as a result of a chosen action. It is the role of a design heuristic to present the designers with a set of alternative actions and help select the one which is best suited to the current state.

Henceforth, we interchangeably refer to the subsystem design teams as either *agents* or *teams* and the system-level designer as the *principal*. All the design tasks that the principal or agents do are referred to as either *actions* or *tasks*. In any state, a *heuristic* is a rule which helps the agents select an action from the available options. A particular sequence of selected actions which constitute the design process is a *policy*. Thus, a heuristic is a function which takes the current state as an input and produces a policy as an output. With every action that an agent takes, it will incur a cost but will change its state, preferably taking it closer to the final outcome, and thus provide it with some payoff. Under the value-based view of design, these payoffs can be quantified in monetary units, though they could take any other form of compensation. In our discussion henceforth, we will refer to payoffs as *rewards*.

3.1.1 Problem Definition

Formally, we can define a design problem with the following elements:

$$\begin{aligned}
&\text{Set of states} && : \mathcal{X} = \bigcup_k x_k && : \forall k \\
&\text{Set of actions} && : \mathcal{A} = \bigcup_{i,k} a_{i,k} && : \forall i,k \\
&\text{Set of rewards} && : \mathcal{R} = \bigcup_{i,k} r_{i,k} && : \forall i,k \\
&\text{Design policy} && : \pi
\end{aligned}$$

In this definition, $\mathcal{X}$ is the feasible space of designs defined in terms of system variables. $\mathcal{X}_0$ are the variables under control of the principal, while $\mathcal{X}_k$ are those under control of each of the teams k. Some of the design variables may be shared by more than one team and also the principal, and hence, the design space is specified by the ranges of values of the union of all variables. Similarly, the set of actions $\mathcal{A}$ is the union of all actions available to all teams k in all states i and which, when done, take the agents from one state to the next. The rewards $\mathcal{R}$ is the set of immediate rewards or payoffs that an agent obtains on taking a particular action in a particular state. Finally, a specific sequence of actions is a design policy, π. In this form, the design process is similar to a Markov decision process, although in the latter, the set of actions are specified as probabilities of transition from one state to another on taking an action. Here we assume that the teams select one of the possible actions in any state and can observe the state it leads to.

In the planning method described in the following sections, the teams initially know only the states and actions available to them while they observe rewards as a result of taking action. The agents' rewards on completion of each action depend on the increase in value that the agent can obtain from that action, the costs associated with the action, and any form of payoff provided by other teams or principal. Agents observe the rewards that they obtain by completing each of the actions, and use the value of discounted cumulative rewards to select a design policy.

Figure 3.1 shows the schematic of a bi-level organization, which we will use throughout the rest of our discussion. The solid bidirectional arrows in this figure

indicate information exchange between the principal and teams while dotted arrows show the same among teams. The nature of information communicated on these links may differ based on who the communicating agents are and may form part of the definition of a heuristic. In a complex systems design process, the principal does not directly engage in low-level tasks such as design space exploration, and rather guides the teams' decision-making by negotiating with them an incentive scheme. We note that rewards are an outcome of an *incentive scheme*. Just as a policy tells us which action to take in any state, an incentive scheme tells us the reward associated with an action.

The teams exchange design variable values and do so with more frequency than their interactions with a system-level designer. Thus, the teams may iterate among themselves and only communicate with the principal once they have locally converged to a mutually agreed solution. Those teams which interact with greater frequency, may then be put in close proximity to aid their decision-making. Further, the links among teams may be more informal, meaning that they may arise and disappear as is suitable to the needs of the teams, whereas the links between teams and the principal may be more structured and formal.

The proposed approach of modeling teams as learning agents is applicable to each of the blocks in Fig. 3.1. The differences between modeling a principal and each of the agents would be the local tasks that each of the agents solve as well as the information they process and their local value functions. The objective of value maximization is applicable at all levels of organizational hierarchy.

3.1.2 Assumptions in Modeling

The possible different structures of an organization and the processes that these organizations employ are numerous and perhaps as diverse as the systems being developed themselves. To keep the scope manageable, we restrict to modeling an organization with two levels of hierarchy, with a single system-level designer at the higher

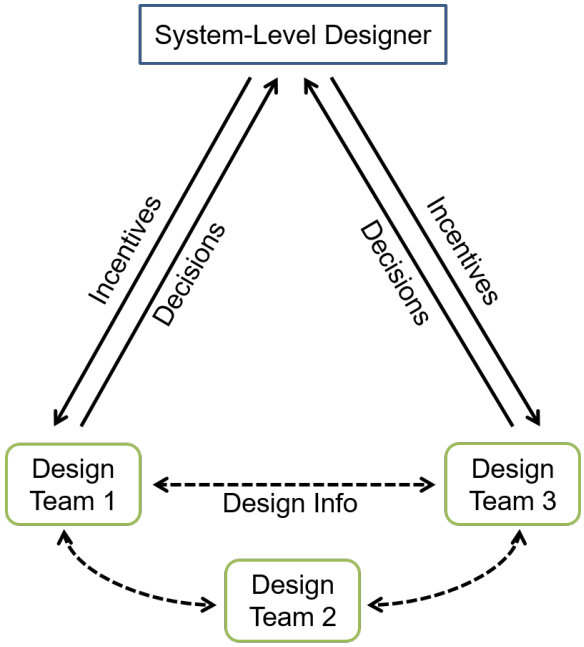

Figure 3.1.: Schematic of the process-system as a bi-level framework.

level, and multiple fixed number of design teams at the lower level. We assume that initial problem decomposition is completed and all teams have clearly defined local objective functions before the start of our process when then remain fixed throughout the process. The structure of the organization defined as the links between agents also remains fixed throughout the course of process, although an outcome of our modeling is an observation on how many times do agents exchange information on each of the links; this will be discussed further in later chapters.

The design environment in which the teams operate is observable, deterministic, and static. *Observable* means that the team can fully observe its current state when making any decision. However, because the other teams are also in the process of designing their subsystems, no team can observe the full design state of the other subsystems. *Deterministic* means that any action that the team takes maps to one future state. This assumption is important in the setup of the planning algorithm described below; here it does not mean that the models used by the team are deterministic, rather it means that regardless of the model employed, the team can settle on a single

estimate of the future state of the world given the application of an action. *Static* means that the state of the world does not change during the course of evaluation.

Design teams have the most flexibility during the conceptual phase, which is when a small number of designers work in close proximity. They are divided into teams along disciplinary boundaries, and the communication links between them are based on shared variables.

3.2 Model of Agents' Decision-Action Behavior

Figure 3.2 shows our model of an agent for representing both the design teams and the principal, broken down into four constituent blocks. The specification of *agent behaviors* involves three things. First, the agent knows its goals and preferences. This is in accordance the assumption stated above that the agent has been provided with its local objectives. Second, it has prior knowledge of the environment, which means that the agent knows which other agents it has to interact with and what information it can exchange because both the organizational structure and rules regarding information exchange have been preset. Lastly, the agent is aware of its abilities, meaning that it has models of the system that it is designing available from the outset and knows what actions are available for it to take.

The *information acquisition* block is where the agent obtains data from other agents that it interacts with. From here, the information passes on to the *agent actions* block, which as two activities – *valuation* and *planning*. In our implementation, this is where the value iteration algorithm selects an optimum policy based off the supplied information. This block updates its estimate of optimum policy as soon as new information becomes available.

Let us highlight the roles of the Valuation and Planning blocks. The Valuation block calculates the payoff to be obtained from each of the available tasks. This block makes use of the value function and information on the contract with the Principal to arrive at the expected reward of each possible action. The Planning block then

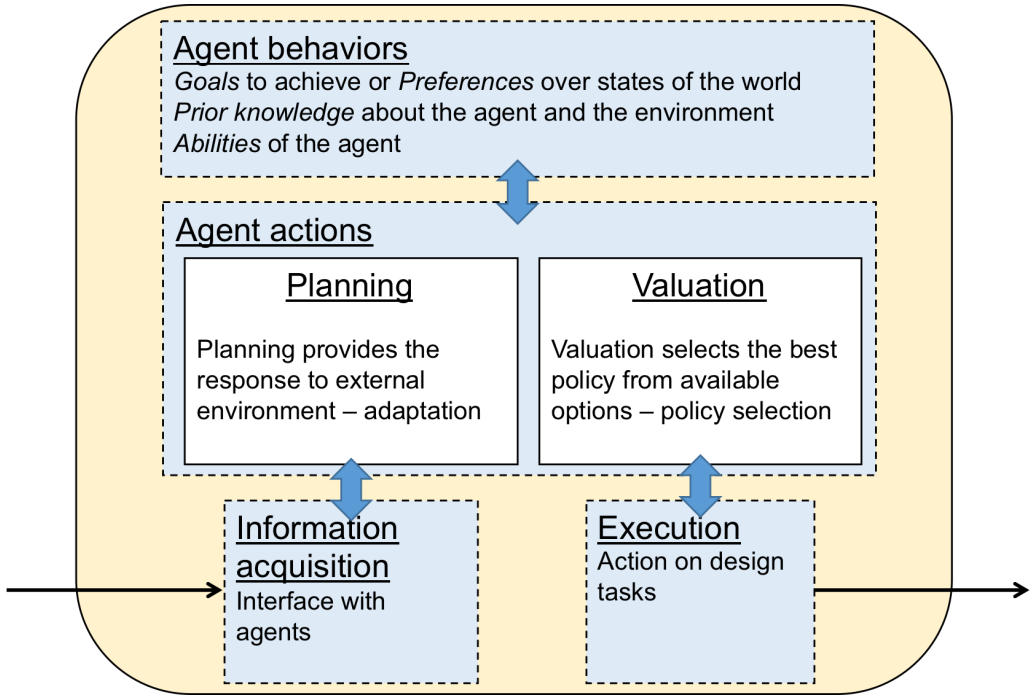

Figure 3.2.: Model of agents' decision-action behavior.

receives this rewards information and selects an action to take in the current state. Thus, this block selects the design policy. Finally, the selected process is passed on to the Execution block where the actual design is done.

Dividing the agent model into four blocks of behaviors, actions, information acquisition, and execution agrees with other models in literature on both organizational structuring and human information processing. For example, in a model of human information processing, Parasuraman and Sheridan [48] have used a four-stage model, where the stages in order are information acquisition, information processing, decision selection, and action implementation. Our model of the team agent closely resembles this model, opening the discussion of modeling teams as "thinking agents." Another, slightly contrasting view of an agent is the Viable Systems Model (VSM) which breaks down the organization along five tasks that it needs to accomplish at any level of hierarchy. Starting from the lowest, these five tasks are action implementation, stability of tasks, optimization of actions, adaptation to environment, and policy selection. In

our analogue, the first two of these tasks are together done by the Execution block, while optimization of actions, in our approach can be assigned to the Valuation block and the last two tasks can be done by the Planning block. Note that our representation is that of a 'team agent,' which could be an individual or a team of humans. Thus, this segregation is functional and not a personnel assignment and the exact specification of agent behaviors would depend on the composition of the team.

3.3 Valuation: Setting Up the Agent Rewards

An action, seen as an information processing unit, is an operator on the design variables:

$$a_{i,k} = O(x_s, x_l) \tag{3.1}$$

Here, x_s indicates a variable shared with other teams and x_l indicates a local variable. The approach of bi-level integrated systems synthesis (BLISS) [49] assumes that when solving the local optimization problem, a design team keeps all shared variables fixed, and only changes values of local variables. In our approach, this is a choice the teams must make at every stage during the design process – they may change values of all variables under their control or only a subset of those, for example, only the ones which are local and not shared.

In order to select an action at each state, the teams evaluate the discounted future rewards obtained from the outcomes of each of the available actions. The rewards that the teams receive are a function of inherent value of the action and any compensation from the principal as a result of a negotiated contract. The inherent value of an action can be the increase in value that the agent obtains on taking that action. The compensation received from the principal could be in form of a large reward associated with attaining a final goal.

Suppose that the teams receive rewards $r_1, r_2, \ldots$ for each of the future states reached on following a particular policy. Design can be seen as an indefinite horizon problem, where even though there is a goal to achieve, the agents do not know how

many actions they will have to take before they achieve it. Thus, the above sequence of rewards is infinite and the teams calculate the value of future discounted rewards as:

$$\begin{aligned} V_i &= \sum_i \gamma^{i-1} r_i \\ &= r_1 + \gamma r_2 + \gamma^2 r_3 + \dots \\ &= r_1 + \gamma V_{i+1} \end{aligned} \tag{3.2}$$

which is the value of state i and γ is the discount factor.

Henceforth, we will indicate the value obtained by the principal until state i with $V_{0,i}$, where the first subscript, 0, indicates the principal and the value obtained by an agent k until state i by $V_{k,i}$. We will indicate the value obtained in a particular state i with $v_{0,i}$ for the principal and with $v_{k,i}$ for an agent k.

To calculate inherent value of an action, an agent needs two inputs – the current state of design as specified by design variable values, whether locally calculated or provided by the other design agents, and values of sensitivities of the value function with respect to the those variables. Using this, the agent can calculate the change in value it can obtain for a change in variable values:

$$\Delta v_{k,i} = \frac{\partial v_{k,i}}{\partial x} \Delta x \tag{3.3}$$

With this formulation we are assuming that a change in variables leads to a linear change in value. While this may not be true in general, in the early stages of exploration of the design space this assumption serves as a useful simplification and provides the teams with discrete alternatives on a value scale to choose from. As design progresses and begins to converge, the search will proceed in smaller steps where this approximation will become more reliable. This definition of value change is also consistent with the discussion by Collopy [25] who states that a scale of value unique to an affine transformation can be used within the expected utility theory. Further, surplus value theory, such as employed by Cheung et al. [24] also uses a similar form of value function in its calculations.

The choice of variables x that the team chooses to modify in any state depends on which action it chooses in its current state. For example, in an approach similar to BLISS, the team may assume that it keeps the variables received from other teams fixed and changes only the local variables, or it may decide to change all variables which affect its value. In the next section we will discuss how the teams calculate discounted values for each possible action and thereby select the one with the highest.

The agent's value function also depends on the compensation it receives from the principal. To understand how we can set up the compensation that a principal gives to an agent, we have to look at their utility functions separately. The objective behind providing compensation is for the principal to guide a team's decisions towards the objective favorable to the system-level objectives. Thus, while the principal is concerned with maximizing the system-level value, it is dependent on the agents' decisions regarding system design variables to achieve its objective.

The principal's value function depends on the outcome and the compensation it provides to the teams, i.e., $V_{0,i} = v_{0,i}(y_i, c_i)$, where y_i is the observed output and c_i is the compensation provided both in state i. Since each team can be expected to provide a different output and receive a different compensation, this value function will also depend on the teams that the principal interacts with. Here, we assume that at any state, the principal has chosen the one outcome which provides the highest value, and hence we do not distinguish his value function with respect to teams. Since the principal tries to maximize the discounted sum of values over the time horizon, the principal's objective function is:

$$max \sum_i \gamma^{i-1} v_{0,i}(y_i, c_i) \tag{3.4}$$

Similarly, the agent's objective is:

$$max \sum_i \gamma^{i-1} v_{k,i}(a_i, c_i) \tag{3.5}$$

which depends on the action a_i which it chooses and the compensation c_i in state i. In both the principal and agent's objective functions, γ is the discounting rate, which we will continue to assume is the same for both.

Together, the value change obtained by an agent as a result of its action in a given state and the compensation it receives from the principal make up the reward that it receives in that state:

$$r_i = \frac{\partial v_{k,i}}{\partial x}\Delta x + v_{k,i}(a_i, c_i) \tag{3.6}$$

The first term in this equation is the inherent value change that the agent obtains by taking an action, and the second term is the compensation that the agent receives from the principal for that action. The agent will use this form of reward to calculate its discounted future rewards and select a policy, which we discuss in the next section.

In the value iteration algorithm which we discuss below, the payoffs (rewards) stays fixed during an "episode," during which a policy is selected. Then the selected action is carried out, followed by an iteration where the process of value calculation and action selection repeats. Through a series of iterations any two teams arrive at a mutually agreed upon value of shared variables, and though they start with requirements provided by a system-level designer, the consensus design they arrive at may not meet the given requirements. In the next iteration, they would try and move the mutually agreed value closer to requirements. The system-level designer can play a role by modifying their rewards, or even weights of variables in order to encourage the teams to match requirements when they do not do so. This is where the utilities of the principal and the teams differ. For the principal, a good design solution is one which provides the highest system-level value and where all teams arrive at a consensus design fulfilling its requirements. For the teams, the local objective functions take precedence. Thus, while for the principal the contract would be based on how close the teams arrive to its Pareto front, the incentives for the teams would be based on their outcomes which is their design state.

3.4 Planning: Selecting an Optimal Policy

The task for the agent is to select an optimum policy given the decisions of the other agents and the system-level designer. A design policy is what specifies the action

an agent should take in any particular state, and, under the value-based view of design, every agent selects the policy which provides it with the highest discounted reward. We will model the learning behavior of teams using the dynamical programming technique of value iteration which we discuss now.

A function called the Q-function defines the value of a starting in state i, taking action a and then following policy π. The value of following policy π in the current state, given as $V^{\pi}(i)$, is defined iteratively with the Q-function, given by $Q^{\pi}(i, a)$:

$$Q^{\pi}(i, a) = r_i(a) + \gamma \sum_{y} V^{\pi}(i) \tag{3.7}$$

$$V^{\pi}(i) = Q^{\pi}(i, \pi(i)) \tag{3.8}$$

In the search for an optimal policy, we define the optimum Q-function, $Q^{*}(i, a)$, by using the value of $V^{*}(i)$, where the latter is defined as:

$$V^{*}(i) = \max_{a} Q^{*}(i, a) \tag{3.9}$$

We use the value iteration algorithm to find the optimal policy. This algorithm starts at an arbitrary end and works backwards recursively refining the values of V and the Q-function until some termination criterion is met. Algorithm 1 shows our implementation of the value iteration algorithm. A design team is in one state at any given them, and hence this algorithm is shown for one state with a predefined set of actions. This algorithm converges regardless of the initial value of V and hence we can arbitrarily initialize it. Note that the same method works for both the principal and the agents.

In the algorithm, each team defines its own set of available actions to choose from. The state is defined as the current instantiation of local attributes while the rewards are calculated as discussed in the previous section. We use a condition on the marginal increase in value function as the termination criterion, such that if the increase in V from one iteration to the next is below a threshold ϵ, the algorithm stops.

Algorithm 1 Value iteration algorithm based on Q-values.

Input: Current state $\mathcal{X}$, set of actions $\mathcal{A}$, and rewards $\mathcal{R}$

Output: Design policy π and value function

Initialize v arbitrarily and k = 0
Specify γ and set $done = FALSE$
for action $a \in \mathcal{A}$ **do**
    Evaluate reward r_a
end for
while $done = FALSE$ **do**
    k = k + 1
    $v_{old} = v$
    for action $a \in \mathcal{A}$ **do**
        $Q_a = r_a + \gamma \times v_{old}$
    end for
    $v = \max_a Q_a$
    if $|v - v_{old}| \leq \epsilon$ **then**
        $done = TRUE$
    end if
end while

3.5 Guiding the Design Process Using Principal-Agent Theory

In this section we discuss the principal-agent models which can be used to represent interactions across hierarchies in the bi-level organization that we have assumed for our complex systems design problem. The key idea of the principal-agent model is that the agents carry out the tasks to complete the work that the principal needs to get done and receive a payoff from the principal in return. The principal, in turn, does not see the internal decision making of the agents and can only observe the outcomes of the agents' actions. For more information on principal-agent models see Refs. [31, 37, 50].

Based on our assumptions, we model the principal and agent interactions over an indefinite time horizon. The agents who carry out the design activities choose, in each state, the action to perform which results in an output. They report these outputs to the principal, who then provides the agents with a compensation which depends on the observed outcome. Let us assume that the principal's value is given by V_0 and it depends on the value of the outcomes and the compensation given to the agents. Let us also assume that the agents do not have any other source of compensation besides that provided by the principal, and the value of agent $k = 1, 2, \ldots$ is the function V_k which depends on the compensation received and the cost of the effort they make for generating the outcomes.

In the scenario of system development, the principal and the agents will agree to a contract at the beginning. Thereafter, the principal interacts with each of the agents in discrete states i. In between those interactions, the agents select an action or actions to take and report the outcome to the principal. Note that as per our assumption of deterministic outputs, the teams know exactly the outcome of their actions at each interaction; this is unlike some of the literature on principal agent models in which the outcomes have an associated probability distribution. Since both the principal and the agent will evaluate their utilities over a time horizon we will assume that they calculate discounted values and that they both apply the same value of discounting rate.

Vermillion and Malak [31] provide the following details of setting up a principal-agent model based on Myerson [50]. The principal has a set of potential actions it can take, $\mathcal{A}_0$, and a value function V_0. The agents, in turn, have their own set of local actions $\mathcal{A}_k$ and value functions V_k. In addition, the agents also have local information θ_i, while the principal holds a belief over the agents' behaviors, ϕ_0. In general, the interaction between the principal and the agents is represented as:

$$\Gamma = (\mathcal{A}_0, V_0, \langle \mathcal{A}_k, \theta_k, V_k \rangle_{k=1}^{n}, \phi_0) \quad \forall k \tag{3.10}$$

Under this framework, the objective of each of the agents and the principal is to maximize their respective utilities. In other words, the decision problems of the principal and agents are, respectively:

$$\pi_0^* = arg\max_{d_0 \in D_0} V_0 \tag{3.11}$$

$$\pi_k^* = arg\max_{d_k \in D_k} V_k \tag{3.12}$$

The outcome of this decision problem τ is a decision profile $\pi^* = (\pi_0^*, \pi_1^*, \ldots, \pi_n^*)$, which leads to the next state of design.

In our setup of the principal-agent interactions, both the principal and all agents solve the above equations in each state, and their respective value functions, $V_{0,i}$ and $V_{k,i}$, represent the discounted rewards that they obtain in state i. Thus, from equations 3.4 and 3.11, the principal's decision problem becomes the following optimization problem:

$$\max_{y,c} V_0 = \sum_i \gamma^{i-1} v_{0,i}(y_i, c_i) \tag{3.13}$$

Similarly, from equations 3.5 and 3.12, the agents' decision problems become the following optimization problems:

$$\max_{a,c} V_k = \sum_i \gamma^{i-1} v_{k,i}(a_i, c_i) \tag{3.14}$$

These value functions are then used in the value iteration algorithm we discussed above.

In the next section we will discuss the steps involved in setting up the problem for use in our proposed framework.

3.6 Setting Up The Modeling Framework

Figure 3.3 shows the flowchart of the framework; briefly, the key steps are:

1. Before the start of the design process, the principal and the teams select their respective value functions. This is a modeling choice to be made in the description of the agent models. Another modeling choice is to select the set of actions that each of the agents can choose from.

2. In the first step of the process, the teams select an initial design and evaluate their local value functions. The principal evaluates his value function for each teams' initial design.

3. The principal selects the best design from the available choices and provides incentives to the teams to encourage convergence to a solution which favors system-level objective. The principal informs the teams of the current best design.

4. The teams choose from an available set of actions while interacting with one another to improve their local value functions. When they fail to improve value, they report their design to the principal.

5. The previous steps iterate until the improvement in principal's value function falls within a tolerance of desired value; this value could be one on the Pareto-front of the principal's objective.

The framework assumes that the complex systems design problem has already been decomposed into subsystem design problems with one team assigned to each subproblem. Each team is given or identifies its local objective functions and a value function based on its objectives. Thus, before the start of the design loop in the framework, each team and the principal has their objectives and value functions. These objectives are chosen by the modeler. The value function is defined as discussed in the 'Valuation' section above. Furthermore, the communication links and the rules of information exchange are set up at the beginning based on the variables shared between teams. Here, we set up a fully connected network, so that any two agents who share variables can exchange information over a bidirectional link. In fact, the

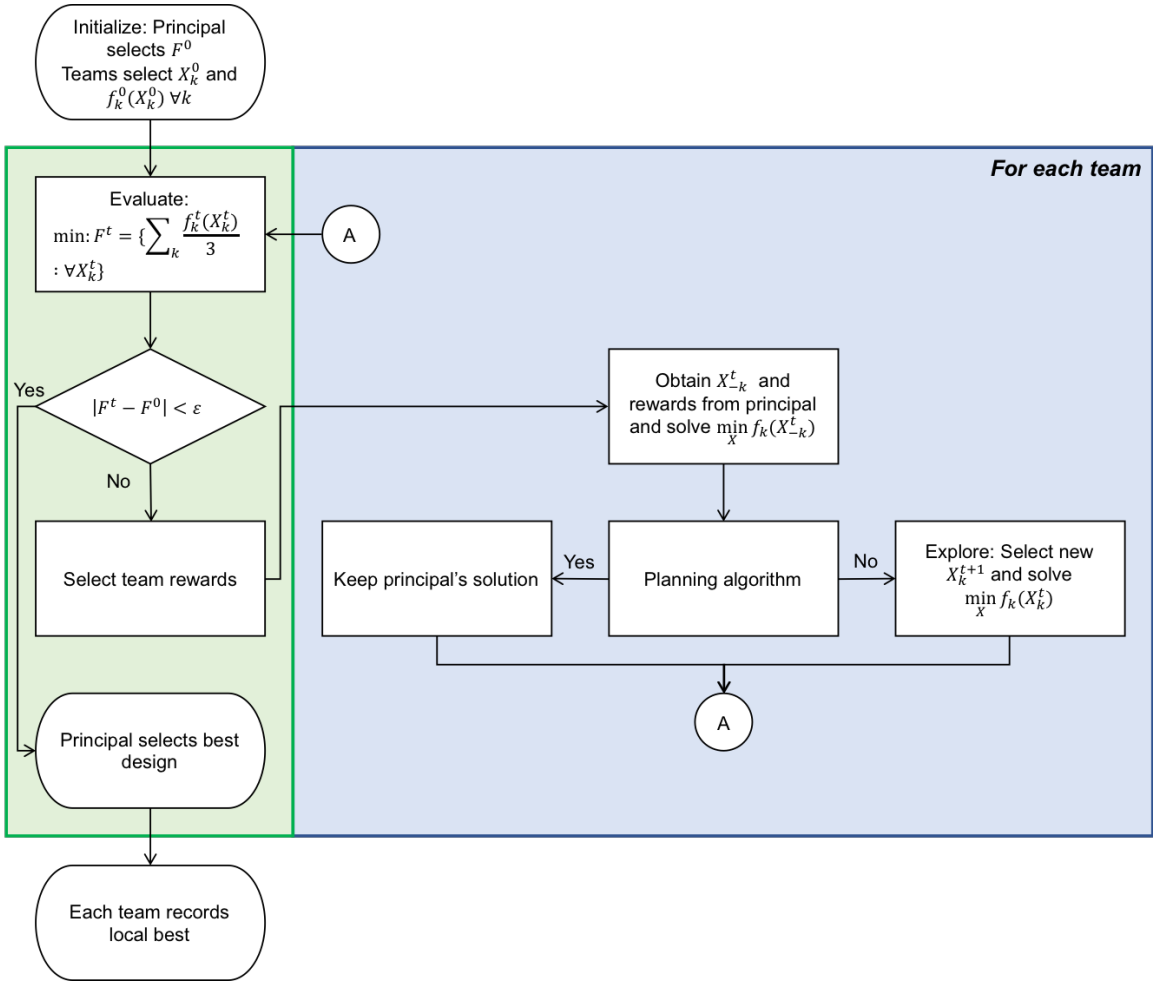

Figure 3.3.: Flowchart of design framework for complex systems design.

frequency and direction of exchange over any link, measured by how often an agent uses the information received from a neighbor is an output from this modeling useful for setting up communications within an organization.

The modeler also has to specify the agents' risk profiles. We discussed three risk profiles in the previous chapter. The modeler can specify any of those three or more complex profiles as part of the agent model. Use of survey data or qualitative analysis is one way in which risk profiles can be determined.

Finally, the modeler selects a set of possible actions for each agent, although this selection can be a responsibility of the agents. In our modeling, the following are the

available actions for each subsystem agent, and this set remains constant throughout the process:

1. Explore design space further
2. Use neighbor's design and explore using local variables
3. Use principal's design and explore using local variables
4. Do nothing

We will discuss these alternatives along with a synthetic demonstration problem in the next chapter. Also, the set of actions available to the principal include choosing from among the subsystem design alternatives or doing nothing. This is because the principal does not explore the design space himself.

The principal can select a target value in two possible ways. If it knows the Pareto front, it can select a design on the front. In our simulation, the principal selects the best of the subsystem designs in the initial iteration as the target. Thereafter, during each iteration the principal evaluates the value function of all subsystem design and terminates the process if it is within a tolerance of its optimal value, or if the marginal gain in value is below a threshold. Also, during each iteration, the principal passes the current best design and a value of rewards to each of the teams.

The teams, on receiving information from the principal, make local decisions until they converge to their best values which they communicate back to the principal. They also exchange information with their neighbors during design. Thus, the teams' interactions are more frequent than those between the teams and the principal.

Note that, even though all agents have the same four available options of actions in our modeling, we can easily set them to have different sets of actions. One final note is on the principal's selection of a design from the available options in each iteration. While the principal evaluates the designs from different teams, it will evaluate all designs where the value of any variable is available from multiple teams. Thus, from the principal's point of view, the teams compete against one another over

the variables that they share. For the teams, however, since they evaluate whether to use a neighboring team's design or to use their own, the choice of competition versus cooperation is local, and made in each state of the process.

In the next chapter, we will elaborate on the process steps using a synthetic design problem including how the teams select between alternative actions by calculating discounted rewards.

4. FEATURES OF THE DESIGN PROBLEM

In this chapter, we will discuss increasingly more complex models of the design process, building up gradually from teams as agents in a network without an hierarchy which collaborate while searching the design space to a bi-level organizational model where the agents learn and adapt their behavior while being guided by the rewards provided by the principal. The following are some features of this organization to bear in mind when considering the discussion which follows. This organization has two levels – at the upper level is a system designer, the principal, who selects the rewards it provides to the teams and maximizes the organization's utility, while at the lower level are the design teams who solve their local problems and maximize their own utilities. The principal tries to achieve compatibility among the teams. It has no direct authority over the actions of the design teams; rather, it attempts to influence their decisions by providing them with appropriate incentives in form of rewards which are proportional to how closely they achieve the principal's targets. The teams are semi-autonomous and they act in self-interest with the objective to maximize their local utility. They coordinate with other teams by exchanging information such as design variable values.

The following are the layers of complexity we will consider; note that each case builds on top of its previous one:

Case 1: Teams interact to arrive at common design. In this first case, all agents know their local objectives and solve their design problems in parallel. They exchange shared design variables with one another and cooperate to arrive at local designs. Without a system-level goal, the teams in this case explore the design space as long as doing so increases their utility.

Case 2: A system-level designer provides target values. In this case, the principal provides a design which optimizes the system-level value function as a goal for the design teams to arrive at collectively. The teams exchange information with the principal directly and may receive rewards proportional to how close they get to the target value.

Case 3: Teams add bias to the value of design variable. With a given target value, the teams add margins to their design variables to maintain flexibility for future iterations. Adding biases is the teams' attempt to get other teams to modify their behavior without compromising on their own value.

Case 4: Teams learn to adapt their behavior. Using the information received from the other teams, the teams now compare alternative actions available to them and select the one which maximizes their local objective using the value iteration algorithm discussed earlier. Thus, this case discusses the first research task of this work.

Case 5: The system-level designer formulates heuristics. The principal receives teams' design outcomes and selects, using the value iteration algorithm, the one which provides the highest system-level utility. He then provides rewards to teams to guide their decisions towards the selected design. This case discusses the second research task.

The next section presents the synthetic design problem we will use for the rest of this chapter, and the five cases follow thereafter.

4.1 The synthetic design problem

The "design" problem in this chapter is a synthetic multi-objective problem, named the comet problem for the shape of its Pareto-optimal front, from Ref. [51]:

$$\begin{aligned}
&\underset{x_1}{\text{minimize}} \quad f_1(x) = (1 + g(x_3))(x_1^3 x_2^2 - 10x_1 - 4x_2) \\
&\underset{x_2}{\text{minimize}} \quad f_2(x) = (1 + g(x_3))(x_1^3 x_2^2 - 10x_1 + 4x_2) \\
&\underset{x_3}{\text{minimize}} \quad f_3(x) = 3(1 + g(x_3))x_1^2 \\
&\qquad\qquad 1 \leq x_1 \leq 3.5 \\
&\qquad\qquad -2 \leq x_2 \leq 2 \\
&\qquad\qquad g(x_3) \geq 0
\end{aligned} \tag{4.1}$$

Figure 4.1 shows the Pareto frontier for this problem solved using a genetic algorithm approach. Starting from a wide region at one end, this frontier progressively narrows towards the other end giving its 'comet' shape. This profile makes it a difficult frontier for any multi-objective algorithm because only a narrow region of the space dominates the rest, thereby lying on the Pareto frontier. In the problem above, we have chosen $g(x_3) = x_3$ and $0 \leq x_3 \leq 1$. Note that in Fig. 4.1 we have plotted the negative of the three objective functions for ease of visualization, similar to that done in Ref. [51].

4.2 Case 1: Interacting agents who seek to arrive at a common value of design outcome

In this first case, teams seek simply to come to a commonly agreed value of design variables while solving their local optimization problems. We assume that each one of the functions in equation 4.1 is assigned to one of the teams; thus objective function 1 belongs to team 1, function 2 belongs to team 2 and function 3 belongs to team 3. Note that, in all cases in this chapter, we are using the objective functions as the teams' value functions, and hence we will refer to teams' decision-making on the basis

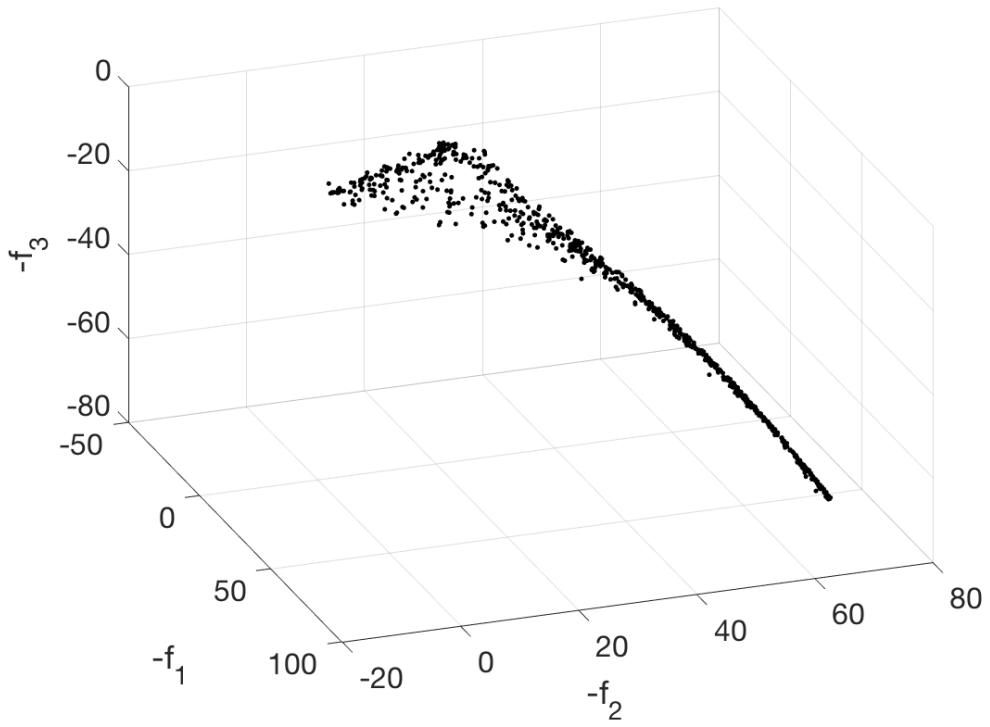

Figure 4.1.: Pareto-optimal frontier of the comet problem.

of their objective function maximization. In general, a value function would be some combination of multiple objectives that the teams have. The teams solve their design problems in parallel, and then exchange the design variable values they obtain with their neighbors, which are the teams with which they share communication links. Here, we assume cyclicity, such that team 1 provides its output to team 2, who, in turn, provides its output to team 3, who provides it to team 1; thus each team has one neighbor that it receives information from and one that it gives information to. Finally, not all teams control all variables even though teams 1 and 2 share all three variables. Arbitrarily, for the purpose of this demonstration, we assume that team 1 searches over design variables x_1 and x_2, team 2 searches over x_2 and x_3, and team 3 searches over x_3 and x_1.

The teams begin with an initial design point and evaluate their objective function values. They then pass on their design point to their neighbor (cyclically as noted above). In turn, upon receiving input from another team, they evaluate their objective function at the received design point. If this new design point improves their objective function value, they discard their own solution in favor of this new one, else they ignore this shared solution and explore further. At the end of one exchange of design variables, the teams share their local objective function values and calculate the mean value of all three objective functions to decide if design should proceed with another round of iteration. If the change in this mean value from the value obtained during previous iteration is within a predetermined tolerance, or if the number of iterations has exceeded the maximum allowed, the design process stops, else the teams continue their search. Note that all three teams are attempting to minimize their objective functions and because their objective function values are of the same order of magnitude scaling is not necessary in this case.

Figure 4.2 shows the flowchart for this case; all of the process steps are executed by all teams in parallel. X indicates the design variable, with subscript k indicating a team and superscript t indicating the iteration number. Initially, all teams select a design X_k^0 and evaluate their local objective functions $f_k(X_k^0)$. They share this objective function values and calculate the initial mean of these functions, F^0. Thereafter, each team receives its neighbor's design X_{-k}^t, where subscript $-k$ indicates a neighbor, and decides whether to keep the neighbor's design or to explore the space further. Next, the teams collectively calculate the mean of their respective objective functions, $F^{t+1} = \sum_k \ f_k^t/3$, and in the following decision block, they decide to terminate the process when the marginal gain in value falls below a fixed tolerance.

Figure 4.3 shows the results of this case together with the Pareto front. We restart the process multiple times and at each convergence the blue "+" are the teams' best designs while the red "$*$" indicate all the design points that, at the end of each rerun, the teams selected as the best design based on the evaluation of F.

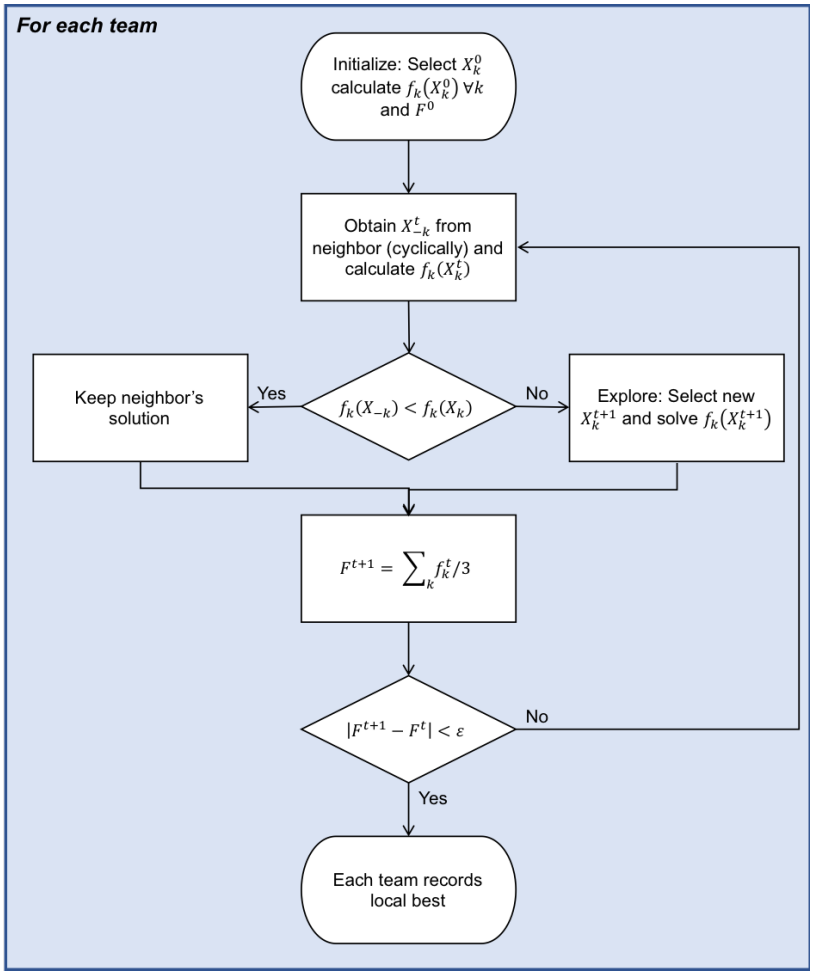

Figure 4.2.: Flowchart for Case 1: Teams which arrive at common value of design.

From this simplistic implementation of complex problem solving case, we demonstrate two features of the process. First, since the agents do not have any goal in this case, with repeated trials, they are able to collectively explore the entire design space; we can see this from the nearly even spread of design points along the Pareto front in Fig. 4.3. However, the process stops as a result of either reaching the limit on number of allowable iterations or finding no significant increase in utility function during the previous iteration, and not as a result of obtaining a global objective.

Second, we can see from Fig. 4.3 that in most cases, the teams' local best design points are not the same as the one they selected collectively. In other words, in order

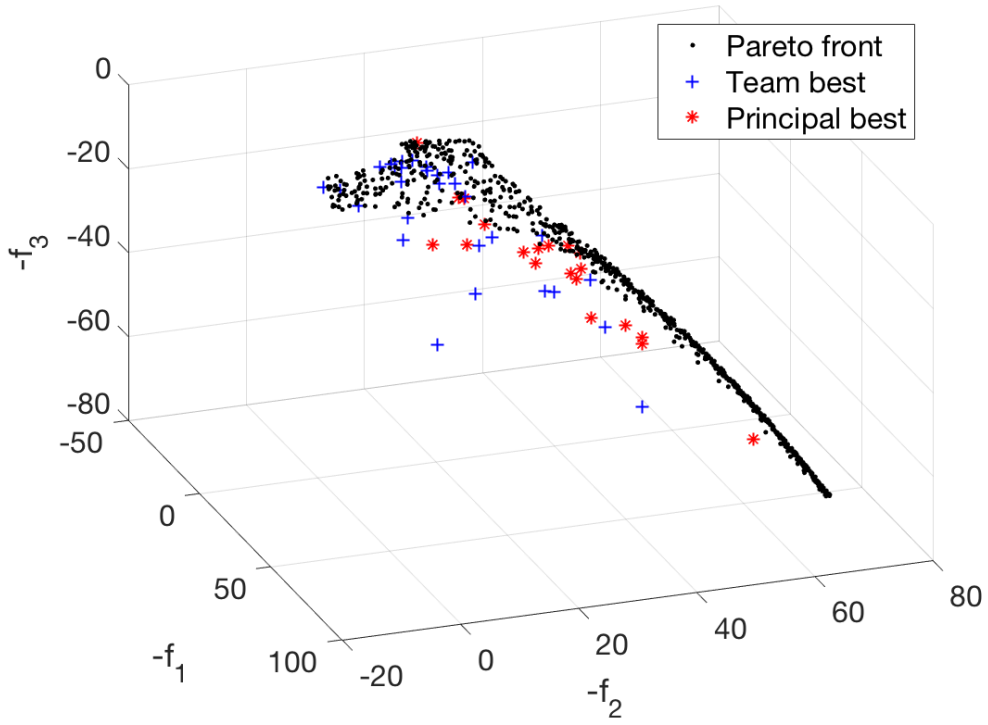

Figure 4.3.: Case 1: Pareto-optimal frontier of the comet problem with design solutions explored by the teams.

to cooperatively agree on a design at the end of each rerun, either one or two teams have to agree to compromise on their obtained utility. Thus, this process needs a mechanism to compensate the teams which do not achieve local optimality at the end of the process.

There are certain features we have not modeled within this case. First, the teams search through the design space by randomly generating new points. They do not make use of such information as the local gradient when selecting a direction of search. Second, if the design they receive from their neighbor does not improve their objective function value, they will always explore further. Thus, the teams do not account for the cost of further search, and a limit is imposed only the the maximum number of allowed iterations.

Summary of Case 1: The teams explore design space collectively. They collaboratively decide when to stop the design process on the basis of low marginal increase in selected value function. Because there is no system-level designer providing them with target values, the search proceeds at random, and, at the end, not all teams achieve highest local utility measured by their respective value functions.

4.3 Case 2: System-level designer provides a target value

In the previous case, the teams solved local problems and exchanged design variable information to arrive at a set of mutually agreed upon design points, but their search was not guided by a goal even though the decision on whether the process should continue or terminate was mutually agreed. In case 2, a principal both provides the teams with a goal and plays the role of a mediator who, after he has received the initial set of design points from the teams, makes decision on whether the process should proceed for another iteration or terminate.

The process is as follows. First, the principal optimizes his value function to determine the Pareto-front in the design space. In this case, since we continue to use the problem formulation from equation 4.1, we assume that the principal knows its value function and can optimize it. For simplicity, and to compare with the previous case, we will assume that the mean of the three objective functions in the comet problem, i.e., the function $F = \sum_k \ f_k/3$ discussed in the previous case is the principal's value function; then the principal selects a random point on the Paretro front as the target value for the teams.

The teams start from an initial design within the feasible space to solve their local problem and provide this design to the principal, who uses its value function to determine the best of the set, i.e., the one which provides it with the highest value. Note that, in this case, even though we assume that the principal uses a composite of the teams' value function as its own, in general the principal's value function may

be different from those of the teams, and indeed the principal may not even know of the teams' value functions which may be private information.

Once the principal receives designs from all three teams, it selects the one which lies closest to the target design as the current best design, and passes this back to all other design teams. Thus, we assume that for the principal the proximity of a solution to the selected target value, measured as the Euclidean distance, is the criterion for the selection of best design and for stopping further search. If this distance is less than a tolerance value, the process stops, else the principal implores the teams to explore further. The team whose design point was selected during the current iteration may choose to explore further if it expects an improvement in utility by doing so. Clearly, the process differs from the previous case: while in the previous case, we assumed a cyclicity among teams for information exchange, here the principal ensures that the best selected design point at every iteration is visible to all teams. The teams do not interact with one another directly.

Figure 4.4 shows the flowchart for this case. The steps in the green box are the principal's tasks who evaluates its value function F and decides if further design space exploration should continue. The teams' tasks in the blue box are similar to that in Case 1, which is to solve their local design problem. Once the allowable number of iterations is exceeded or the principal's criterion met, the search stops, and the principal and all teams select their best design.

The results of several different reruns of this case are shown in Fig. 4.5. Even in this case, the teams explore a wide range of the design space and not all arrive at a single design point at each rerun, since during any rerun of the process the principal may decide to stop the process and the teams will have to accept the current best system-level design. Once again, the blue "+" signs are the teams' best designs at the end of each rerun, while the red "*" are the best designs accepted by the principal.

The most important note on this figure is that it shows very few design solutions. This is because we limit the number of iterations to 1000; this limit is common to all cases run in this chapter. If the stopping criterion is not met within the limit of

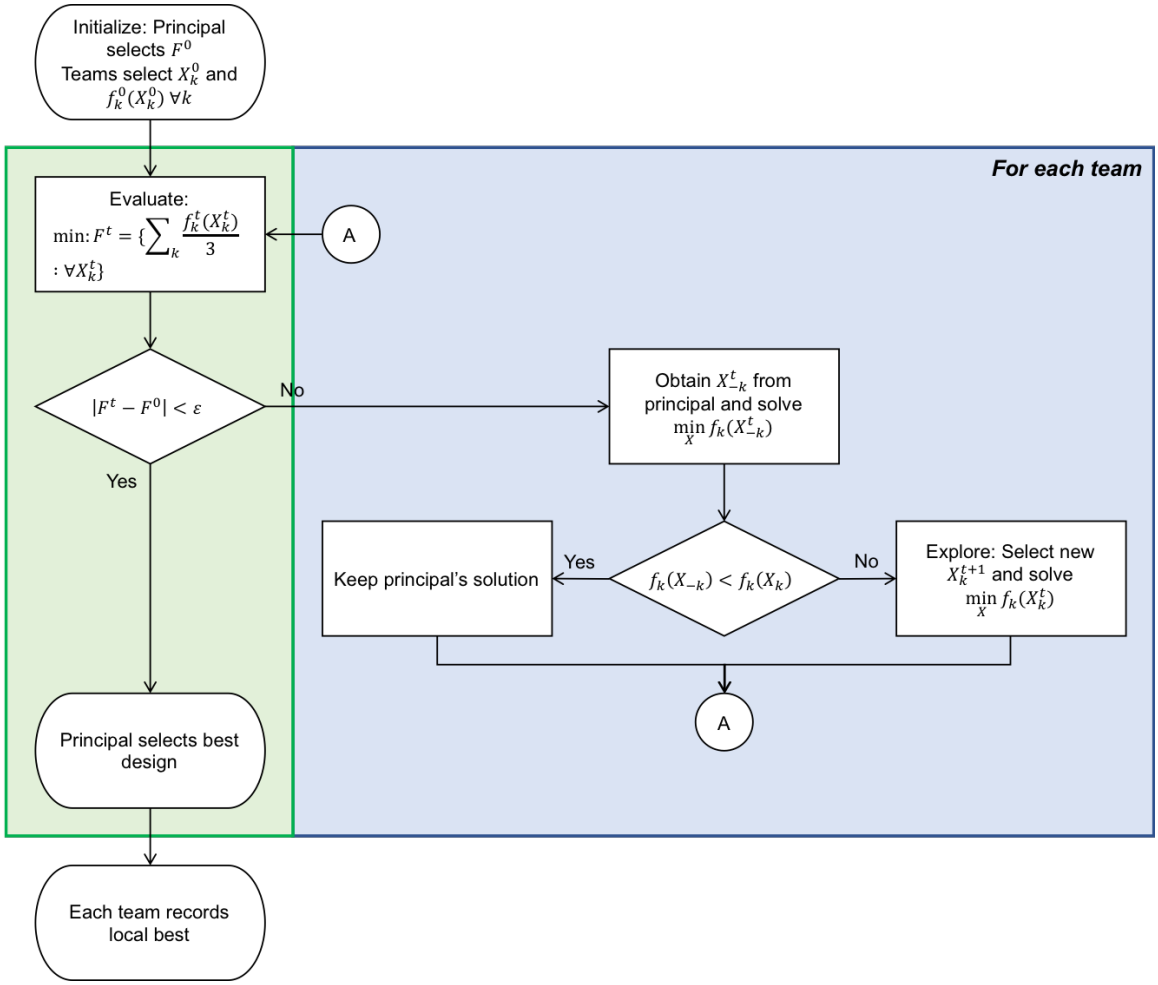

Figure 4.4.: Flowchart of Case 2: The principal provides the teams with a target value to work towards.

number of iterations, the results of the process are not recorded. In the result shown in Fig. 4.5, only four out of 20 reruns of the process converged to a solution within the limit of number of iterations. Later, when the principal provides the teams with rewards to make decisions in favor of the system-level objective, we will expect the process to terminate faster.

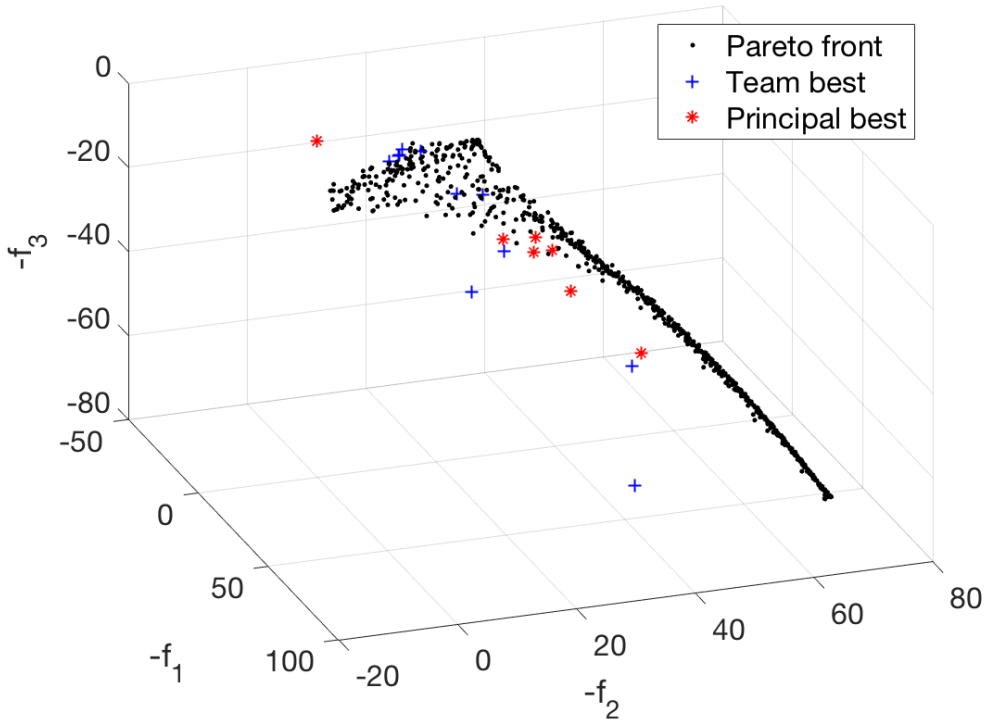

Figure 4.5.: Case 2: Pareto-optimal frontier of the comet problem with design solutions explored by the teams when guided by the principal.

Summary of Case 2: With a given target, the teams now make choices to work towards the principal's goal if it improves their local utility. Because there is no penalty for deviation from a common design and the principal does not yet solve a compatibility problem, the process still proceeds as random search. Not all reruns of the process converge within the limit of number of iterations.

4.4 Case 3: Teams add bias to the value of design variable

In this case, the complexities of behaviors begin to manifest. This case builds on the previous two by adding one feature – that the teams no longer convey honest information to their neighbors. In a complex systems design process, the design teams may add margins to the value of variables that they pass to other teams for reasons

such as maintaining flexibility for future changes. While such an act may help the team's local objectives, it turns out to be detrimental to the design process itself by requiring higher budget expenses while still producing sub-optimal designs [52]. Even so, this practice is common, and needs to be studied and accounted for during the selection of a design process.

Austin-Breneman et al. [52], studied the effect of biased information passing between design teams on the final outcome of a complex system development process. Particularly, the authors modeled a series of optimization problems to study the effect of addition of margins by the design teams on the information they pass to other teams, and showed that this practice negatively affects both the program time and optimality of the final design. This behavior of addition of margins is based on interviews with systems engineers involved in complex systems development projects, and is a result of subsystem design teams retaining margins to allow for future flexibility of their designs. The practice of adding margins turns out to be detrimental to the project because it adds to the costs and leads to sub-optimal designs.

Our implementation of biases are similar to those suggested in the above study. The teams add a 30% margin to their designs before passing them to either the principal or other teams; this margin remains constant throughout the process, in contrast with that reported in reference [52] which states that the teams progressively reduce margins as the process advances. Thus, the teams maintain two values of each design variable – one which is true and local to them, and the other which is biased and is shared with the other teams. Only when the design process stops do the teams convey true values of their variables to the principal. The flowchart for this case is the same as that for case 2 (Fig. 4.4), with the sole difference that the teams now report designs with margins. Because, as in the previous case, the search relies on random search, there are no notable visual differences in the results obtained. The teams do not measure the costs of the process which stops when the the principal achieves its objective or when the number of iterations reaches the limit. In the next two cases, when we model teams as agents who evaluate discounted future rewards

of their actions, we will again compare the results of biased information passing with the cases where teams pass true information.

Summary of Case 3: Passing of biased information results in an increase in the costs of design process and the solution quality deteriorates.

4.5 Case 4: Teams learn to adapt their behavior

This case discusses the first of the two research tasks, which concerns the teams' adaptive behavior based on the information available to them from their neighbors; in this case we do not model teams' interactions with the principal during the course of the process, which we will discuss in the next case. In the previous two cases, when the teams were faced with choices, they evaluated the value obtained from the designs proposed by the principal and their neighbors and selected the one which gave them higher value than what they had already, or continued with further exploration otherwise. Similar to Case 2, the principal selects a target design at the start of the process and decides when to stop the process based on its value function. However, rather than simply consider the instantaneous value gain, the teams observe the rewards obtained from each of the possible set of actions available and choose the one which gives them the highest discounted value gain. After each iteration, the teams share their local design alternative with the highest discounted value with the principal then uses a system-level value function to select the current best design. This case will investigate whether using discounted value approach leads the teams to arrive at better design outcomes and faster.

Figure 4.6 shows the flowchart for this case. This is similar to the previous case, except that the teams' decision block is now replaced with the planning algorithm. The principal still evaluates the marginal gain in value over consecutive iterations to decide when to stop the process. Thus, except for the teams' approach for making a choice, the procedure for this case is similar to that in Case 2.

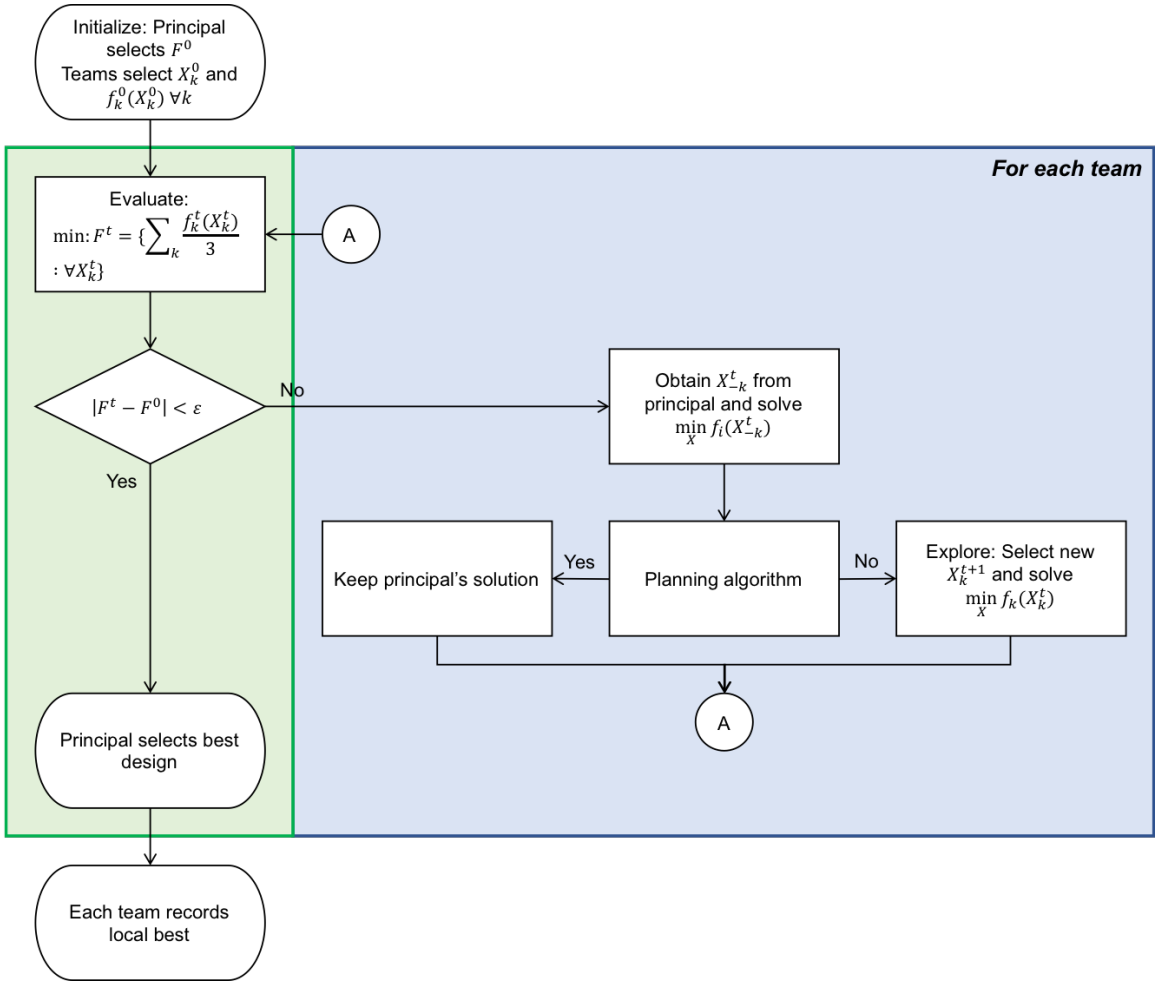

Figure 4.6.: Flowchart for Case 4: When teams learn and adapt their behavior.

In the previous case, the teams evaluated the design provided by either their neighbors or the principal and if it improved their objective, they retained the provided design, or they explored further to find a new design. In this case, the teams calculate the reward they can get from each action using Eq. 3.6 and then use these rewards in the value iteration algorithm to determine which action to take. The agents' Valuation block updates the values of rewards every time they receive new designs from the other agents, but the Planning block runs the value iteration algorithm until convergence and only updates at the end of each convergence. This means that the agents

may asynchronously receive updates from other agents but will alternate between using new information and taking an action.

The teams initially select a set of available actions. In our modeling, we assume that the same set of actions are available to all teams, though this does not need to be true in general. The following are the actions that all teams can take during each stage of the iteration:

1. *Use principal's design:* Similar to case 1, the team receives and holds constant the values of all variables provided to it by the principal and which are not local to its subsystem, and explores the design space by searching on the local variables. For example, team 1 will use the principal's value of variable x_3 and explore on x_1 and x_2.

2. *Use neighbor's design:* The team takes the value provided by the neighbor for the variables it shares with the neighbor and explores on other variables. For example, in the interaction between teams 1 and 2, team 1 will explore on variables x_1 and x_3 but use the value of x_2 provided by team 2 because this latter variable is shared between them.

3. *Explore design space:* Here the team explores on all its variables including the ones it shares. It ignores any variables received from its neighbors and the principal.

4. *Do nothing:* This option has the default payoff of zero reward. In case the rewards from other options is negative, the team will choose to do nothing and wait until the next iteration when it will have new information. In this way, the teams avoid cost of evaluating the other teams' designs when doing so provides no benefit.

Note that the role of the principal in this case is to provide the teams with a target value, and then decide when it has been achieved. The principal does not attempt to influence teams' behavior by the use of incentive, which we will look at in the next case.

During each state of design, the teams implement value iteration algorithm discussed in the previous chapter. When the planning step is reached within the framework, the starting state for each team is the current state it is in, and the terminal or goal state is always the satisfaction of principal's objective. The purpose of planning during each iteration is to choose which of the available options provides it with the highest payoff. This means that the chosen action may not necessarily move the team towards the principal's target.

Figure 4.7 shows the results of run from this case; the symbols mean the same as before. Two things become quickly apparent from the figure. First, with the limit of 1000 iterations which is the same as that imposed in Case 2, the teams converge to a solution in more number of cases. Thus, by evaluating the potential payoffs from actions, the teams converge quicker even when all other conditions are the same as in Case 2. Second, because the decision to continue the process further is local to the teams, the resultant 'best' solutions are not all close to the Pareto front. The principal's role as the one who provides rewards as incentives will address this latter issue in the next case.

When the same case was run but with addition of margins by the teams, we observe that the teams, on average, need more iterations in order to arrive at a final design. This means that with an iteration limit of 1000, the teams were less likely to converge to a final design if they add margins when passing information to their neighbors. In 100 reruns of the process, the teams converged on 55 occasions without addition of margins, whereas with addition of margins they converged on 26 occasions.

In a bi-level organizations, the teams exchange information with higher frequency among themselves than they do with the principal who is a level up in hierarchy. We also record the number of designs that the teams explore among themselves before they communicate with the principal. In this example, the teams explored, on average, approximately 1500 designs without bias, while with bias they explored over 3500 designs. We will further discuss this fact in the next case.

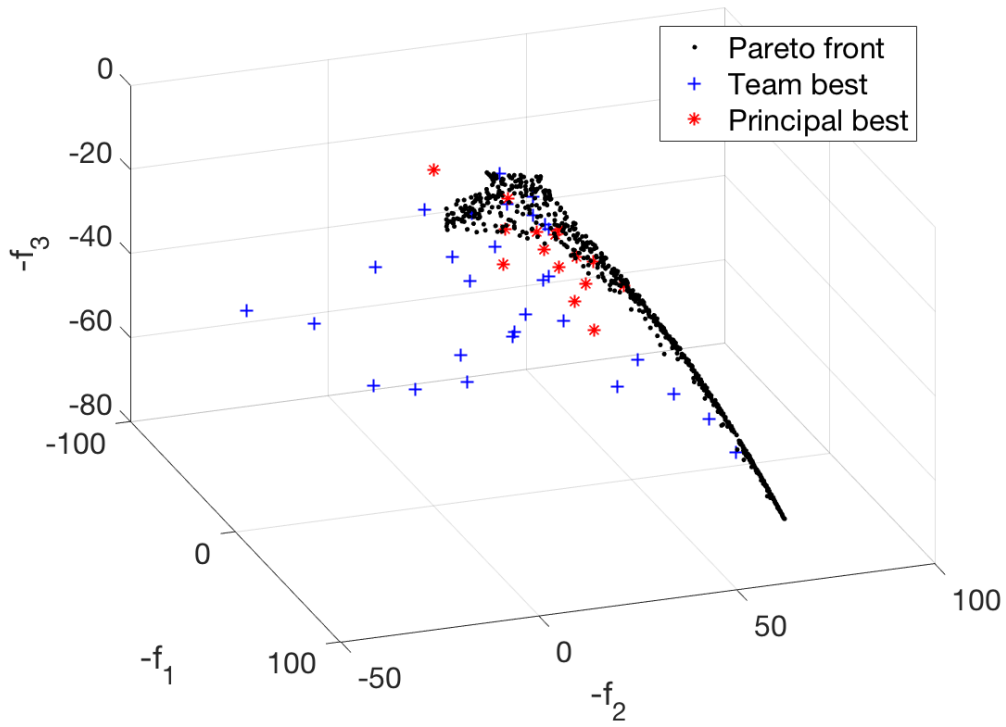

Figure 4.7.: Case 4: Pareto-optimal frontier of the comet problem with design solutions explored by the teams when they evaluate future discounted values.

Summary of Case 4: When considering the future discounted rewards available from the set of possible actions, the teams converge to a solution in more number of cases, though they do not arrive at the Pareto front in all those cases. Addition of margins by the teams leads to fewer converged solutions with a higher average number of iterations.

4.6 Case 5: The system-level designer formulates heuristics

In the previous case, the foremost objective of the design teams was to maximize their own value; they cooperated only when it was beneficial to do so. Building on the previous case, in this case the principal provides teams with rewards which are proportional to how close they get to its target. The additional rewards are the means

by which the principal can influence the teams' decisions such that they may choose alternatives which favor the system-level objectives over local objectives. However, because the teams evaluate alternative actions of using their neighbor's designs or exploring towards system-level objectives, an effect of provision of compensation by the principal could be that the teams may be more likely to compete rather than cooperate to achieve convergence of designs. To avoid this effect, the principal continually adjusts these rewards based on observed values of design variables during the course of the process. Furthermore, the principal will also update its targets if a new design with higher value is discovered. The teams, in turn, will adjust to these changes by modifying their strategy accordingly.

In Case 2, we observed that the final converged design outcomes were close to the Pareto front. Two features made this possible. First, because the teams never considered the cost of exploration and compared only two options – whether a new design improves their value or not – they would, given sufficient number of iterations, always reach at the Pareto front. This is because the decision on continuing the design process rested with the principal who, being aware of the Pareto front, could always make the process continue until the designs were close to optimal. Consequently, such a random search by the teams took a large number of iterations before convergence. In the present case, the 'reward' from each of the actions includes a term which reduces the payoff in proportion to the difference between the state the teams achieve and the state the principal desires. This term functions as a cost on any action which is not goal-directed towards a system-level objective. An explicit cost term could itself be a part of the teams' value function but is not modeled in the current case.

Second, we assumed in Case 2 that the principal could solve the system-level problem to find a Pareto front. If, however, the principal is not aware of its value function, then searching for the Pareto front is not possible. In such a case the principal will begin with the designs explored by the teams to decide on which ones give it the highest value. Thus, we now assume that the principal can no longer solve the system-level problem. Instead, it evaluates the designs obtained from each of

the teams to determine which one provides it with the highest value. During each iteration, it considers the state change as proposed by each of the team and selects the one which gives it an increase in value. It stops further design process when the marginal value gain during two consecutive iterations falls below a tolerance value. The flowchart for this is the final version in Fig. 3.3. This figure shows that now even the principal uses the value iteration algorithm, in the block labeled 'Planning algorithm', to evaluate discounted rewards and select the current best design.

Figure 4.8 shows the result of this case. First, we observe that the principal achieves designs which lie very close to the Pareto front even though it has not solved a system-level problem. This means that multiple teams making local decisions and being guided by a principal can achieve designs with high system-level value functions. Second, even in this case not all teams get close to the Pareto front at the end of the process. This is because the principal stops the process in lesser number of iterations than in the previous cases on the basis of value gain at the system-level. The teams are then compensated by the principal for their lost value at the termination of the process.

Table 4.1 shows the comparison between this case and cases 2 and 4 for the number of times that the teams converged to a final design and the average number of iterations that they took to do so. The improvement due to the teams' accounting for discounted rewards is already significant when comparing cases 2 and 4. Case 5 further shows the effect of principal's rewards. In Case 5, all 100 reruns lead to a converged design and the average number of designs explored is reduced from 967 in Case 4 to 14.

Summary of Case 5: Since teams will now be compensated by the principal, they converge to target design quickly. The principal ensures that the total compensation is less than the value obtained from the final design. Thus, the principal is responsible for guiding the process, while the teams make decisions on subsystem design.

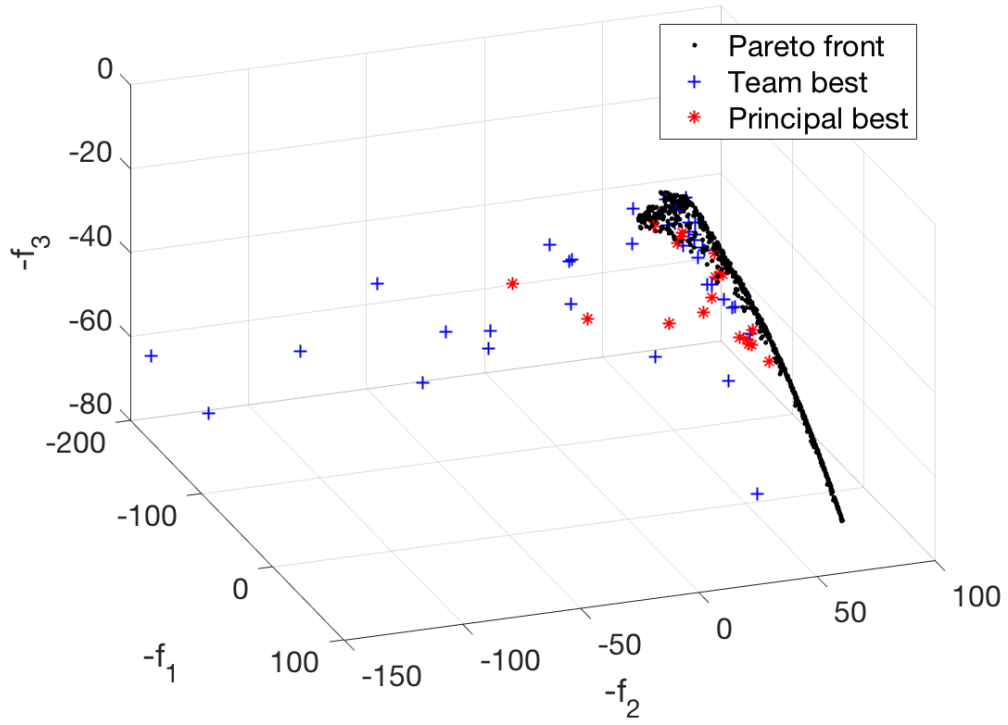

Figure 4.8.: Case 5: Pareto-optimal frontier of the comet problem with design solutions explored by the teams when they are guided by the principal.

Table 4.1.: Comparison of results from cases 2, 4, and 5

Case	Number of times converged	Average number of iterations
Case 2	36	2587
Case 4	71	967
Case 5	100	14

4.7 Summary of Case Studies with Framework

Table 4.2 provides a summary of all five cases simulated in this chapter; the key results and insights are stated in this table.

In this chapter, we built increasingly more complex scenarios of system design and simulated in the proposed framework. The final scenario captured all the features of this framework. Highlights of this modeling include:

- All teams at all levels of hierarchy are set up as agents following the model discussed in chapter 3. Particularly, define the value functions of each of these agents and the set of their actions.

- The agent which models the principal is similar to all other agents except that it does not search over the design space and instead provides rewards to the teams in order to coax them to make decisions in favor of the system-level objective. In this manner, it attempts to achieve compatibility among teams.

- The results of the simulation including the number of interactions among agents and their choices of which action to select during each state are recorded and used for analyses such as how to structure the communication links within the organization.

We will further elaborate on the last point in the above list in chapter 6, where we will discuss how the network of interactions among the teams can be assessed using this framework. Before that discussion, the next chapter will demonstrate an application of this method to an aircraft design problem.

Table 4.2.: Summary of cases simulated using proposed framework.

Case	Key Result	Insights
Case 1	The teams engage in random search over their local design space. They collectively end process when a commonly agreed system-level value function has low marginal gain in value.	Without a system-level target, the process will converge on a solution that provides high payoff to some but not all teams. Also, the final designs are not always optimal at the because no agent carries out system-level trade-offs.
Case 2	The teams converge on a principal-provided target. The number of convergences when limited by number of iterations reduces.	When a single agent knows a Pareto front of a multi-objective problem he can choose to continue process until his target is achieved. However, lack of consideration of cost makes this procedure unrealistic.
Case 3	When teams add margins to the information they pass, the process takes longer to complete and the quality of final solution declines.	Addition of margins by design teams is detrimental to the process but this effect is hard to observe when there is no system-level policy guiding search.
Case 4	The teams now achieve higher number of converged design solutions because they use a value iteration approach.	By accounting for future costs and benefits via a discounted rewards approach, the teams converge to a solution faster.
Case 5	The principal successfully incentivizes teams to converge to designs closer to system-level optimum.	When a principal accounts for future discounted value, he is able to achieve high system-level value even though he is not aware of a Pareto front.

5. DEMONSTRATION OF FRAMEWORK USING AN AIRCRAFT DESIGN PROBLEM

We demonstrate the application of the proposed process modeling framework using an example of civil jet aircraft design problem. In the synthetic problem of the previous chapter, all three subsystems shared all of the design variables, although we assumed that each of the teams controlled only two of their three variables. This assumption no longer holds in the problem which follows because not all variables are common to all teams and some variables are local to just one team. Further, we assumed that the principal's value function in the synthetic design problem was a composite of the teams' value functions. This is no longer the case in our demonstration problem. Here, the principal has the objective of minimizing the aircraft operating cost, while the subsystem teams have their respective local objective functions which we use as their value functions. None of the agents know the value functions of any other agent. The only interaction among the teams is by the design variable information they send to their neighbors, and between the teams and the principal are the variables and rewards they exchange.

In the section that follows (Sec. 5.1) we present the set up of this problem including the subsystems we include in our modeling, the variables that each of them control and their value functions. Similar to the synthetic problem of previous chapter, we will solve the same design problem using our framework and then an optimizer, in this case, using the simulated annealing algorithm. Thus we will compare the outcomes from using an optimization approach where an algorithm considers trade-offs between all subsystems simultaneously versus a distributed design scenario where every agent can make choices only for local design variables; Sec. 5.2 presents these results. Following this, Sec. 5.3 shows the effect of restructuring the organization, and

Sec. 5.4 discusses the compensation that the principal gives to the teams on the basis of the selected incentive scheme. Finally, Sec. 5.5 summarizes our studies within this chapter.

5.1 Demonstration Problem Setup

The aircraft design problem consists of three teams of aerodynamics, structures, and propulsion, and they are guided by a system-level cost minimization agent. Figure 5.1 shows the bi-level architecture of this problem. We do not initially specify any communication links, rather the links exist wherever two teams share variables. As the three teams exchange information with one another as well as the principal during the process, we record the teams' choices of which source of data to accept during each state of the process, which indicates the links that the teams need during each state of the process. Then, the sum of total number of interactions indicates the rate information exchange over each link. In our modeling, all three teams as well as the principal have only one objective function, except in Sec. 5.3 where we combine the principal and the performance team into one agent. We discuss the specifics of teams' decision-making next.

The problem has six design variables: total take-off weight (W_{TO}), wing aspect ratio (AR), sea-level thrust (T_{sl}), wing area (S_{wing}), coefficient of lift at take-off ($C_{L,TO}$), and the coefficient of lift at landing ($C_{L,land}$). Table 5.1 shows the bounds on the variable values.

The structures team can modify the first four of these design variables and calculates such attributes of the aircraft as the take-off weight, fuel weight, and wing loading (W/S). The weights calculations are based on the the mission segment weight fraction method. In our demonstration, this team's objective is to minimize the total take-off weight which becomes its value function, though in general, it can have multiple objectives which would require the value function to be a composite of all of its objectives with a scalar output.

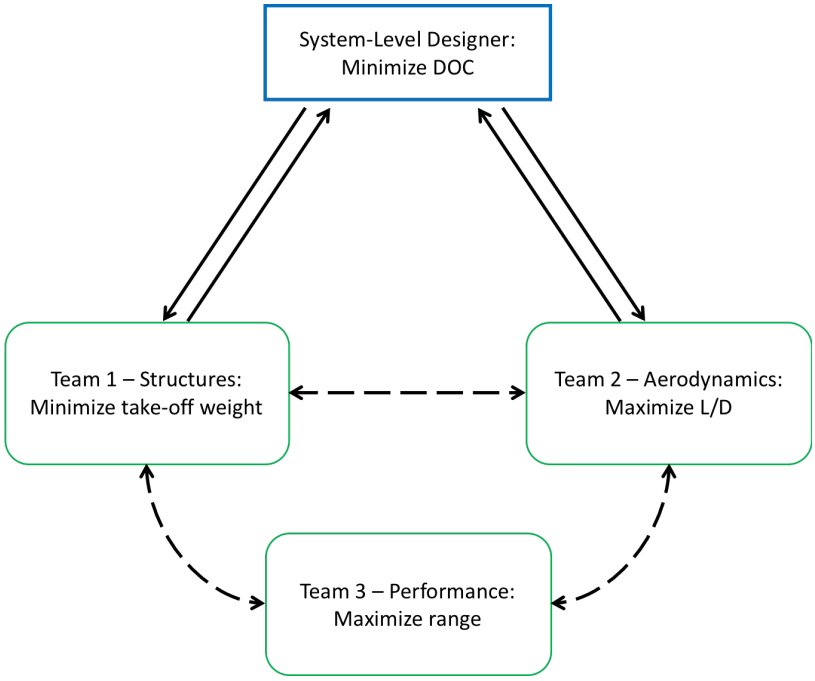

Figure 5.1.: Schematic of organization structure for demonstration problem.

Table 5.1.: Bounds on design variables from aircraft design application problem

Variable	Lower bound	Upper bound
Total take-off weight (W_{TO}, kg)	60000	75000
Wing aspect ratio (AR)	5.50	9.00
Sea-level thrust (T_{sl}, kN)	10	16
Wing area (S_{wing}, m^2)	70	120
Coefficient of lift at take-off ($C_{L,TO}$)	1.20	2.00
Coefficient of lift at landing ($C_{L,land}$)	2.00	3.00

The aerodynamics team can modify the values of wing aspect ratio and both the lift coefficients. This team's objective is to maximize the value of lift to drag (L/D) ratio. Finally, the performance team calculates the flight Mach number and the range of the aircraft. This team receives the design variables from the weights and

aerodynamics teams and evaluates those designs to ensure that any range requirement is satisfied and selects the design which gives the highest range. Note that since this team does not search over the design space, it does not make use of the value iteration algorithm that we discussed. It simply selects the alternative with a higher range as its preferred design.

Table 5.2 summarizes the objectives of each of the teams and the variables they control. The 'Variables Controlled' column is empty for the performance team because they do not modify any of the design variables. This team can force the other teams to discard an existing design and explore the space further if the design does not satisfy the range requirement. Similarly, the principal does not modify any of the design variables. It chooses between the designs provided by either the structures or the aerodynamics teams and maintains the best design as the current solution. Once he sees that the marginal improvement in value function over consecutive iterations is below a threshold value, it stops the process.

Table 5.2.: Subsystem teams in the aircraft design application problem

Subsystem	Objective	Variables Controlled
Structures	Take-off weight (kg)	$W_{TO}, AR, T_{sl}, S_{wing}$
Aerodynamics	L/D ratio	$AR, C_{L,TO}, C_{L,land}$
Performance	Range (km)	–
Principal	Direct operating cost ($/seat-nm)	–

The principal's objective of minimizing operating cost uses the cost estimation method given by Jenkinson et al. [53]. In this method, the direct operating costs (DOC) of an aircraft is divided into three main components – standing charges, maintenance costs, and flying costs, which include the airport, crew and fuel cost; Fig. 5.2 shows the breakdown of aircraft direct operating costs. The trip DOC is seen to vary with maximum take-off weight (W_{TO}), range, cruise Mach number, cruise

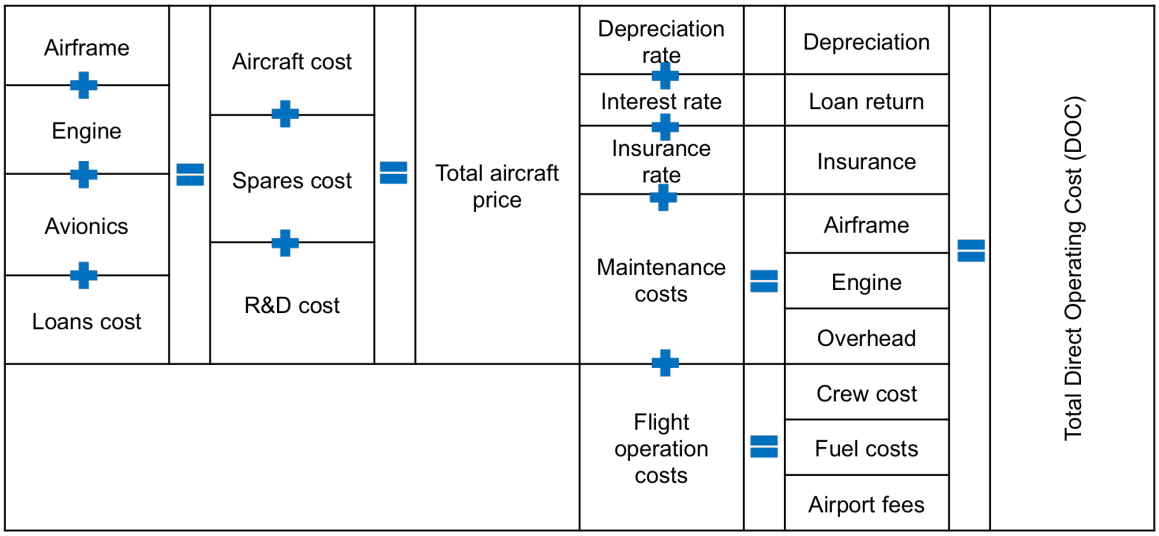

Figure 5.2.: Breakdown of Aircraft Direct Operating Costs from [53].

altitude, and the mission fuel. The trip DOC divided by number of passengers and the range results in DOC per seat nautical mile, and this is the metric that the principal uses as objective for minimization.

Table 5.3 shows some of the design requirements and constant parameters of modeling. The "range with max payload" constant is the performance team's minimum range target; this team rejects any design which does not provide at least this range with the result that all other teams are forced to search further. Finally, Table 5.4 shows some additional performance requirements for which this aircraft was sized.

5.2 Results from Framework Versus Optimization Algorithm

This section discusses the results of the above design problem solved in a manner similar to Case 5 from the previous chapter. The results obtained from the framework are compared with those obtained from a simulated annealing algorithm. The previous section discussed the initial steps of problem decomposition, specification of design variable bounds, and the constant design requirements and cost parameters. In the framework, the maximum number of iterations is restricted to 1000 similar to

Table 5.3.: Design requirements and constant parameters of modeling.

Design constant	Value
Design requirements	
Range with max payload	2500 km
$W_{payload}$	15000 kg
Number of crew	2 (flight) + 4 (cabin)
Number of Passengers	100
For Direct Operating Cost calculations	
Aircraft utilization	4200 hours/year
Depreciation	16% per annum
Aircraft residual value	10%
Investment interest rate	5.4%
Insurance rate	0.5%

Table 5.4.: Performance requirements for aircraft design problem.

Parameter	Performance requirement
Landing field length	$\leq$ 750 m at W_{TO}
Landing field distance	$\leq$ 1000 m
Take-off field distance	$\leq$ 1000 m
Second segment climb gradient	3.0%
Missed approach climb gradient	2.7%

the cases in the previous chapter. The following discussion does not account for team bias in the framework.

Specification of the problem setup also requires specifying the set of action choices that the agents have. As noted earlier, the performance team does not make use of the value iteration algorithm, so we need not specify its action set. For the structures and aerodynamics teams, the action set is similar to that discussed in the previous chapter and includes the following four alternatives: (1) use principal's design, (2) use neighbor's design, (3) explore space, and (4) do nothing. The principal's action set involves comparison of designs received from the structures and aerodynamics teams and selection of the one which gives better value. The principal's value function is DOC per seat nautical mile based on Ref. [53] as discussed above.

The simulated annealing algorithm, which also solves this problem, provides a comparison of results. Our implementation of the optimization is based on a code developed by Goffe et al. [54] which is based on the simulated annealing optimization algorithm developed by Corona et al. [55]. In this optimizer, the initial temperature is set as 10^9 and it is reduced by a factor of 0.95 in each iteration. The difference between the optimizer and the framework is that, while the framework simulates a distributed design problem, the optimizer sees all functions and, thus, can search for a globally optimum solution.

Table 5.5 shows a comparison of results obtained from the framework with those from the simulated annealing optimization. A few observations are readily apparent. First, the structures team, whose team-level objective is to minimize total take-off weight, achieves a better solution with our framework as compared with the results from the simulated annealing optimizer. The aerodynamics team, whose team-level objective is to maximize the lift-to-drag ratio, on the other hand obtains a slightly lower L/D ratio in the framework-selected design than in the optimizer results. Note that both of these teams use the discounted value approach to identify which design gives them a better value, and their value functions were dependent on their single objective as defined in this study. Particularly, the structures team achieves higher value as it reduces weight whereas the aerodynamics team can improve value by increasing the L/D ratio.

Table 5.5.: Results comparison between framework discussed and simulated annealing optimizer.

Subsystem	Objective	Framework results	Optimizer results
Structures	Take-off weight (kg)	61288	62250
Aerodynamics	L/D ratio	14.12	14.33
Performance	Range (km)	2971	2857
Principal	DOC ($/seat-nm)	15.17	15.57

In contrast, the performance team, which does not use the value iteration method and evaluates simply whether the current design meets a minimum range, achieves a higher range from the framework. Note that, as indicated in Table 5.3 above, the range requirement, in this case, is set as a constraint, such that only designs with range greater than 2500 km are accepted. Finally, the principal gets a better total direct operating cost. Thus, the principal obtains a better result even while evaluating a value function which depends on only one of the six design variables, viz., the total take-off weight, W_{TO}.

Table 5.6 shows the final values of design variables obtained by the framework and optimizer. We observe that using the framework results in an aircraft which is lighter than that obtained by the optimizer. However, it also has a smaller wing, which results in a higher wing loading of 824.9 kg/m^2 with the framework versus 803.23 kg/m^2 with the optimizer.

The above table also shows that the framework results in a lower wing aspect ratio and an engine with smaller sea-level thrust. Both these factors influence performance with the result that not only does the aircraft obtained from the framework fly further, it also does so at a higher cruise Mach number of 0.80 versus 0.78 from the optimizer. Note that, the cruise altitude is calculated and aircraft is allowed to fly at different altitudes. The aircraft obtained from the framework carries a slightly higher fuel by

Table 5.6.: Comparison of variables between framework and simulated annealing optimizer.

Variable	Framework results	Optimizer results
Total take-off weight (W_{TO}, kg)	61288	62250
Wing aspect ratio (AR)	5.80	6.03
Sea-level thrust (T_{sl}, kN)	10.52	10.90
Wing area (S_{wing}, m^2)	74.29	77.5
Coefficient of lift at take-off ($C_{L,TO}$)	1.33	1.32
Coefficient of lift at landing ($C_{L,land}$)	2.70	2.15

weight fraction of 0.167 versus 0.163 of the optimizer. Fuel can be a significant portion of the direct operating cost, if the fuel price is high. This can potentially change the value calculations; here, all parameters, including fuel price are based on those given by Jenkinson et al. [53]. Finally, note that a combination of lower take-off weight, smaller aspect ratio, and higher wing loading together require higher lift coefficients in the framework results to meet takeoff and landing constraints.

Two key outcomes from this modeling will be useful in analysis of a design process. First, by observing how many times does information exchange take place on each of the communication links within an organization, we can identify which teams should be placed in close proximity so as to encourage easier exchange of information which would be beneficial for the process outcome. In the next chapter, we will discuss using such observations from simulation in more detail. In the next section, however, we will discuss how we can set up the same problem but by restructuring the organization. Specifically, in the next section we discuss the effect on design outcome when we merge the performance team with the principal.

Second, this framework can help evaluate the effects of different incentive schemes on subsystem teams' decision-making. Later in this chapter we will discuss how we

can record and evaluate the compensation that the principal provides to each of the teams during the course of the project based on the incentive scheme set up for the aircraft design problem.

5.3 Effect of Organization Restructuring on Design Outcome

We have noted that a system decomposed into a set of subsystems is a prerequisite to the use of proposed framework. This means that, even though the frequency of interactions among teams is an outcome from the simulation, the particular set of agents we choose to model and their available actions is a modeling choice to be made before we can use this framework. Here, we demonstrate the effect of an alternative organizational structure on design. Particularly, since the performance team does not search over the design variables, we now make two changes to problem set up: first, the performance team is merged with the principal, and second, the merged agent will use a discounted value approach with a linear value function comprising two objectives.

Figure 5.3 shows the modified structure of this organization. Note the principal's block at the top of the figure. The principal retains its objective of minimization of DOC per seat nautical mile, but in addition it now subsumes the performance team's objective of meeting a minimum range. Additionally, it adds Mach number as a linear term within its value function, which then it uses in its value iteration algorithm. In the previous section, neither the principal nor the performance team search over the design variables, and this holds true in this case. Thus, the principal, while it now includes the performance team's objective within its own value function, does not use Mach number as a design variable to search over, and instead obtains it based on the design proposed by the two remaining subsystem teams.

The Mach number term is evaluated as a difference from a minimum requirement of Mach 0.8, which means that any design which gives a lower Mach number receives a penalty. Also, the range requirement is no longer a threshold condition like in the

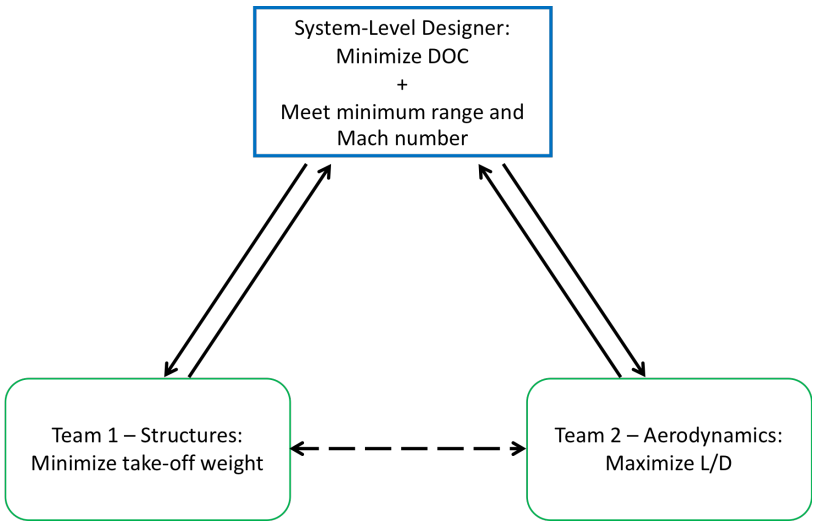

Figure 5.3.: Schematic of organization structure for demonstration problem.

previous case, rather now it contributes to the value function of the principal, such that any designs below the 2500 km range requirement are penalized and reduce the principal's value. Thus, the principal's value function is now a linear combination of its DOC per seat nautical mile, range requirement, and the Mach number it obtains, and all these three terms are added together with equal weights.

Table 5.7 shows the comparison of results when performance was a separate team in the baseline case discussed in Sec. 5.2 and those obtained in the present case with the performance team merged with the principal. We observe that while the aircraft is now slightly heavier, it flies slightly less distance and has a lower DOC per seat nautical mile.

Table 5.8 shows the comparison among design variable values for the baseline case with this case. However, the most important effect of this restructuring of the organization is that in the present case the aircraft flies at a Mach number of 0.85 versus 0.80 in the baseline case. The effect of adding maximization of range and Mach number to the principal's value function results in the aircraft flying faster for nearly the same range as in the baseline case.

Table 5.7.: Results comparison between baseline case and when performance team's objectives merge with principal.

Subsystem	Objective	Baseline case	Principal + performance team
Structures	Take-off weight (kg)	61288	61568
Aerodynamics	L/D ratio	14.12	13.77
Performance	Range (km)	2971	2940
Principal	DOC ($/seat-nm)	15.17	14.55

Table 5.8.: Comparison of design variables between baseline case and when performance team's objectives merge with principal.

Variable	Baseline case	Principal + performance team
Total take-off weight (W_{to}, kg)	61288	61568
Wing aspect ratio (AR)	5.80	5.87
Sea-level thrust (T_{sl}, kN)	10.52	10.63
Wing area (S_{wing}, m^2)	74.29	75.23
Coefficient of lift at take-off ($C_{L,to}$)	1.33	1.69
Coefficient of lift at landing ($C_{L,land}$)	2.70	2.23

Finally, we note that the aircraft design obtained in this case where the objectives of the performance team are merged with the principal differs primarily in that it flies faster and is very close to the design obtained in the baseline case for all design variables. The key notable difference is that the aerodynamics team suffers in its value function obtained. We observe this from Table 5.7 which shows that the aircraft in this case has a reduced L/D ratio and from Table 5.8 which shows that while the lift coefficient at take-off condition is higher, the lift coefficient at landing is lower.

Thus, with this case we have demonstrated that problem decomposition has an effect on the design outcome. In large organizations, with many more subsystem teams than were studied in this chapter, it would be difficult to evaluate all organizational structures. However, as we noted previously, in complex systems design, organizations are frequently decomposed along disciplinary boundaries. We could begin by modeling an existing organization within our proposed framework, and by observing the results of the simulation we can evaluate one or a few alternative structures. Then we can choose the one which would provide the best trade-off between design obtained and the cost expended in doing so. This is one of the capabilities provided by the proposed modeling framework.

5.4 Discussion of Compensation Given to the Teams by the Principal

This section discusses the compensation that the principal pays out to the teams in order to coax them to make decisions in favor of the system-level objectives. This compensation could be a monetary payoff that the principal gives to the teams to compensate them for any loss in value they may have for complying with the principal's objectives. Thus, the teams are incentivized to make decisions in favor of system-level objectives even if doing so would make them lose value because they know that the principal would help them recover their losses.

The principal's value function consists of two components, viz., the inherent value it obtains from the system it designs, which is the outcome from the subsystem teams' decisions, and the compensation that it pays to the teams. The compensation functions as a cost to the principal, and it wants to restrict the total payout to all teams to within its budget. In each step of the iteration, the principal selects which of the available designs is best as measured by the discounted future rewards which are the sum of its inherent value of the system design and the compensations it pays out to the teams.

For the team whose design is selected, the compensation is zero, while for the other teams it is in proportion to their deviation from the optimal. This form of compensation would seem to encourage teams to move away from system-level objectives because then they would be given a higher reward. However, note that this is a compensation paid to a team for loss of their value when they comply with the system-level objectives at the expense of their local objectives. The teams' highest value would still be calculated from the subsystem design which gives the optimal value of their local value function, and if the principal requires the teams to deviate from their local best designs, it will compensate them for the difference between their best value and the value they obtain for the principal's proposed design. In each state, the principal will limit the amount of payoff it can give, so that its sum over the entire process is within the budget. Thus, there is an upper limit on what the teams can obtain as payoff. Note again, that we had made the assumption that the only source of payoff to the teams is from the principal.

Figure 5.4 shows the cumulative compensation received by both the structures and aerodynamics teams. In the first step, the principal chooses the design of structures team and compensates the aerodynamics team to coax them to move towards the same design. Hence, the aerodynamics team begins with a high value of compensation at the beginning. The value function of the principal accounts for all these payoffs.

Vermillion and Malak [31] discussed using an incentive scheme for task allocation wherein a subsystem team was given a reward only if its design was above a certain threshold. Such a threshold can also be applied in case of our aircraft design problem. In such as case, the teams will receive rewards only if they are within a certain bounds from the principal's best design. This is important because those subsystems which deviate by large margins from the principal's best design can be identified as those subsystem which impose high costs (in form of the compensation), and this could prompt further exploration of the design for alternatives.

Figure 5.5 shows the compensation received by the two teams in each iteration. Note that, this figure only shows the last 21 of a total of 31 iterations that this

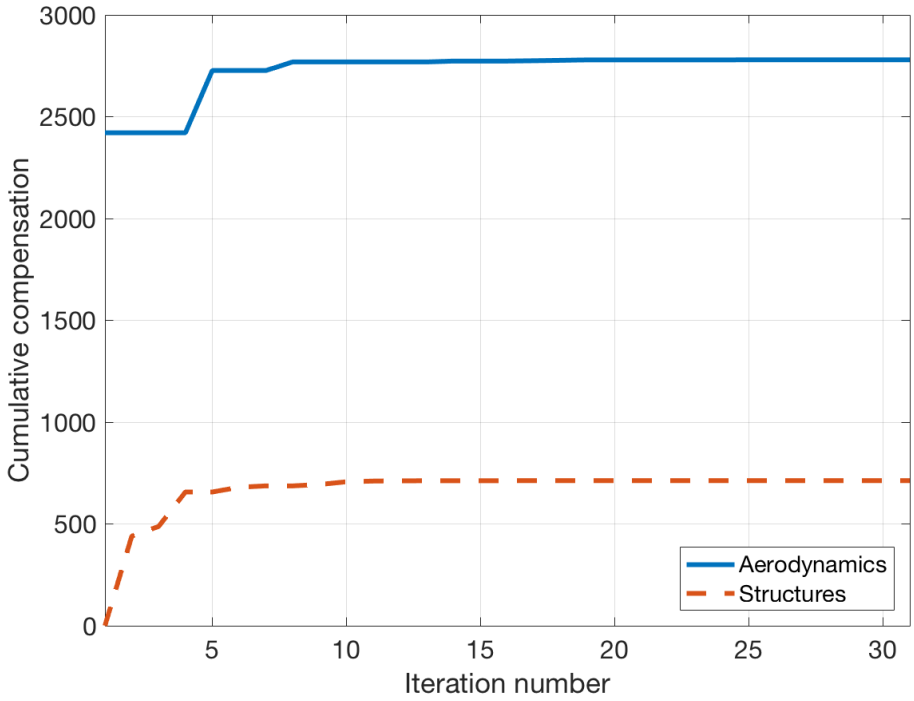

Figure 5.4.: Cumulative compensation received by structures and aerodynamics teams.

simulation ran for; these amounts are smaller than those given in the first 10 iterations. Figure 5.6 shows the cumulative compensations received by the two teams in the last 21 iteration. In this figure, the compensation for the aerodynamics team is shown on the left axis, while that for the structures team is on the right axis.

Finally, we note that the compensation structure discussed in this section is based on the Euclidean distance between each team's design variables and the principal's selected design variable. Hence, the actual values shown in the above two figures are representative only of the scale of compensations. In an actual process, the principal will set up mapping from these values to a monetary amount to be paid out to the teams.

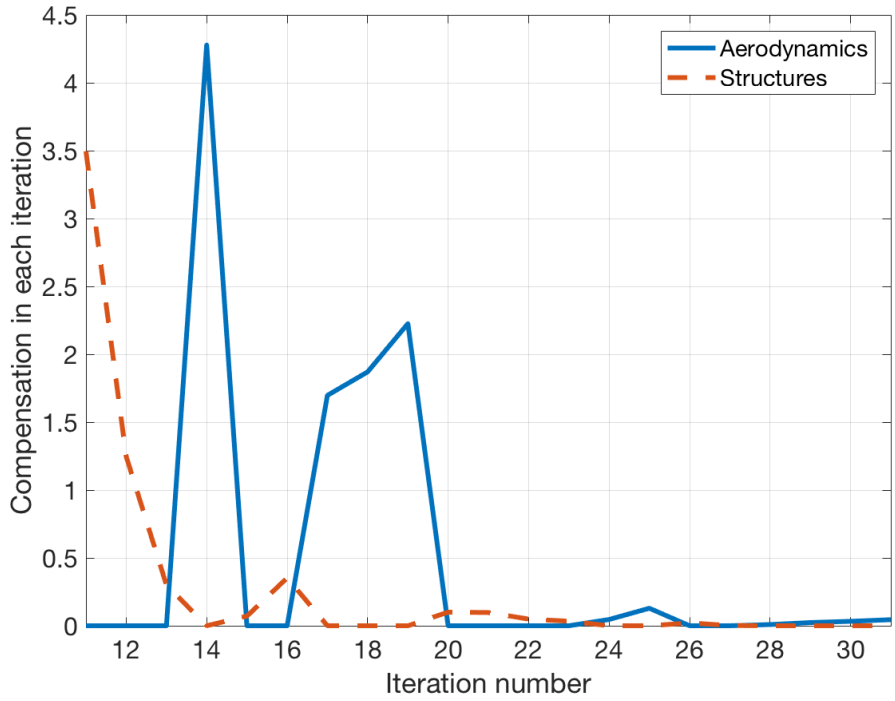

Figure 5.5.: Compensation received by structures and aerodynamics teams in each iteration.

5.5 Summary and Discussion of Demonstration Problem

Our use of a principal-agent model is justified because the division of responsibilities is such that the task of searching over design space falls on the subsystem teams, and the task of ensuring that the teams arrive at a compatible final design falls on the principal. The above selection of subsystem objectives is a modeling choice we make for the purpose of this demonstration and these objectives can be easily changed, such as our demonstration in Sec. 5.3. Even the hierarchy within the process can be changed, for example, by setting any of the other subsystem teams as the principal, or by combination of two or more teams. A record of the compensation payoffs to the teams by the principal is the incentive scheme, and different schemes can be evaluated using this framework.

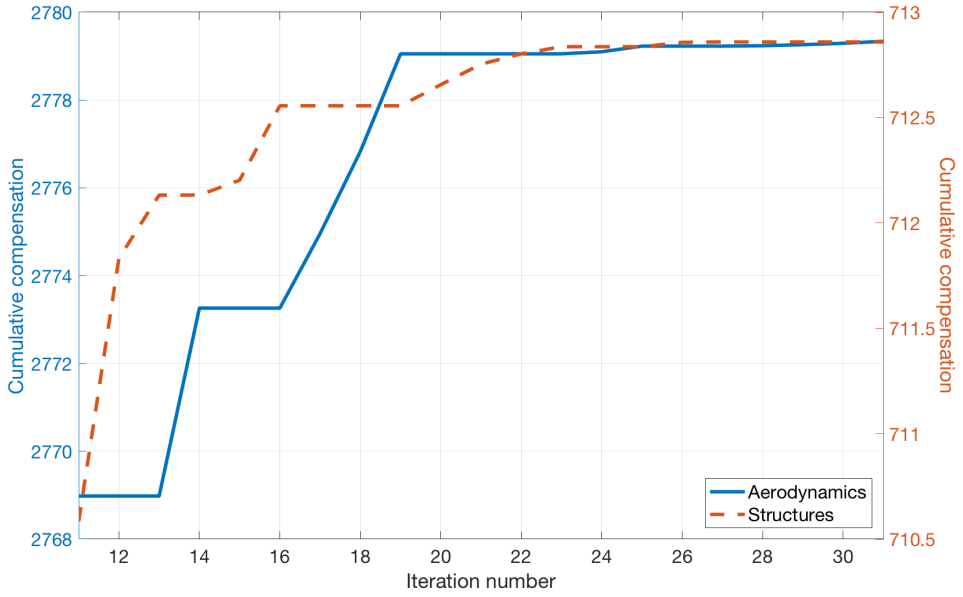

Figure 5.6.: Cumulative compensation received by structures and aerodynamics teams.

Clearly, with distributed design, the simulation in the framework makes different trade-offs than an optimization algorithm, even though they both use the same objective function of minimization of direct operating cost. The differences are due to a combination of decision-making being distributed among different subsystem teams rather than a system-level optimizer, the set of available actions, and our selected incentive scheme. Changes that may be made to any of these features to account for the organization's context will result in different results than what we have obtained. In the next chapter, we will discuss how we can use the results of simulation obtained from this framework for selection of an organization structure.

6. DICUSSION OF DESIGN PROCESS MODELING

Referring again to the classification scheme proposed by Wynn and Clarkson [13] (Fig. 2.1), the work we have discussed in this dissertation falls within the analytical type at the meso level because this work "concerns the specific steps that should occur within a design context." Because the proposed framework allows for modeling and simulation of several cases within a design process, such as we discussed in chapter 4, this work also covers the management science / operations research type category. Therefore, this work spans multiple categories, as was suggested to be a need by Ref. [13].

The motivation of modeling complex systems design processes arises from the fact that in such processes an organization needs to coordinate a large number of design teams and their activities and ensure that the outcome meets requirements while maximizing system-level value. However, the system-level designer lacks direct control over the actions of subsystem teams. Further complexity of the process results from the presence of social phenomena such as an organization's culture, attitudes of the human designers involved, economic factors such as different subsystem value models, etc.

Chapter 2 discussed a few of the many models developed for study of such processes. Each model has its limitations and is suitable to particular problems, which means that the organization has to choose an appropriate model for its purpose. The dynamics of the process-system, therefore, would also include the changing models at each stage of design. Our approach has been distribute the decision-making among the teams so that a system-level designer is tasked with management of information and teams rather than make low-level design decisions. In turn, the teams, tasked with making design decisions, learn from available information and adapt their behavior to obtain the best value. In this chapter, we will discuss the outcomes of

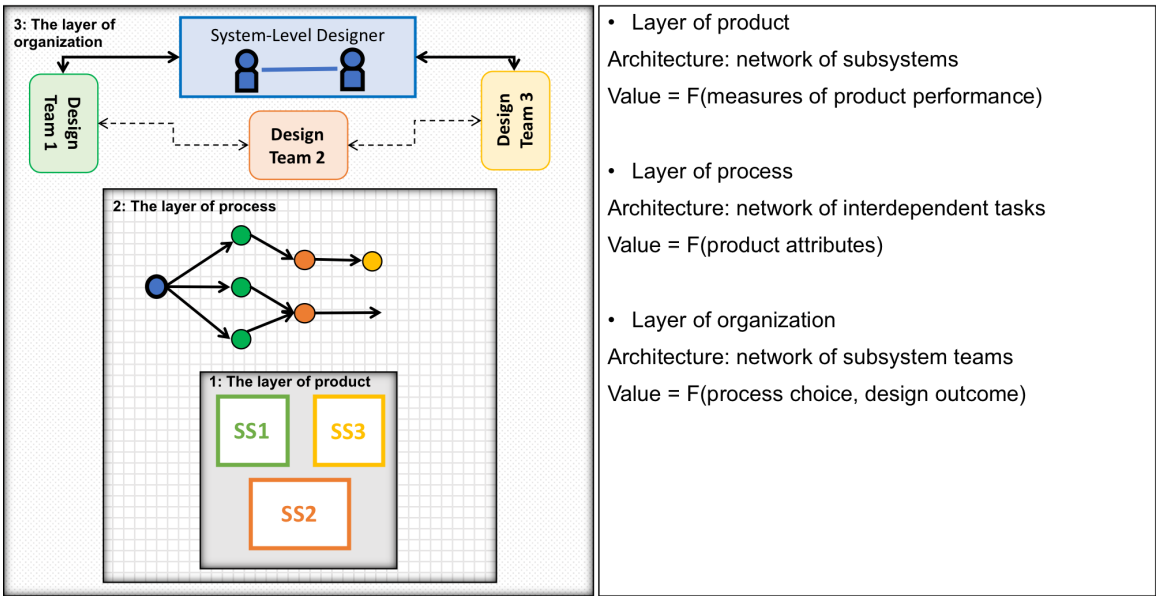

Figure 6.1.: The product-process-organization nesting of complex systems design process.

our simulation alongside discussion from literature and suggest further advancements that are essential for progress in design process modeling.

Figure 6.1 shows the product, process, and organization levels nested within one another and identifies the architectural components, value functions, and the roles of each level; we discussed how a value-driven design problem can be set at the three levels in chapter 2. At the innermost level of the product, the architecture consists of network of subsystems and their components. Any value function of a product will be function of its attributes, for example, the range of an aircraft. The role of this level is to provide the capabilities required by a customer. At the outermost level of the organization is the network of subsystem teams. Their value depends on the choice of processes they employ and the design outcomes they achieve. The role of this level is to design the system which provides the capabilities while maximizing system-level value.

Finally, the layer of the design process lies in between the product and organizational layers and provides the interface between them. The architecture of the process

level is a network of interdependent tasks. The value function of a process depends on the costs of the actions that the teams take and the payoff they receive in return.

In the discussion which follows, we discuss how a simulation of the process layer can be used to set up an organizational structure. We also discuss value functions particularly from the point of view of modeling human risk behaviors. We discuss some of the questions that can be posed to our proposed framework. While quantitative models of human behavior exist, we also discuss how qualitative models can complement them by presenting descriptive analysis of design processes.

6.1 Role of Design Process Models

In this section, we will discuss the following outcomes from models of complex systems design processes:

1. Simulating a design process can be a way to identify the organizational structures that would lead to desired outcomes. We use the number of interactions among teams based on our simulation to determine how the structure can be setup.

2. Models of design processes need to be flexible to adapt to various different organizational structures and the change in the requirements of the process change over time. They can also be used for informing an appropriate organizational structure setup.

3. A value-based approach is a useful means of comparing between different processes. This approach can take into account features of human behavior such as their biases, their preferences, risk tolerance, etc.

6.1.1 Setting Up An Organizational Structure

We set up a fully connected network of agents at the start of our synthetic problem in chapter 4. Let us now discuss how simulation using the proposed framework can

be used to determine the structure of the network of agents in this problem. Each team in the synthetic problem knows its value function which it tries to maximize. Further, each team chooses from four available actions – take the values of variables from neighbors and explore on local variables, explore on all variables to search in direction of the principal's target, or to do nothing. Their decisions during the course of exploration are recorded as outcome, and this information can be used to determine which links among teams to establish.

Table 6.1 shows the results of team interactions in the problem. In this table, the values within the cells indicate the percentage of total interactions that a team in a row used information received from the team in a column. Thus, for team 1, in row 1 of the table, on 35% occasions it chose to continue exploration using local information, on 24% occasions it used the information obtained from team 2, while on 41% occasions it used information from team 3. From these values, we can see that teams 1 and 3 need to be placed such that they can communicate quickly because while team 1 used information from team 3 most often, team 3 also used either its local information or that received from team 1 on most occasions.

Table 6.1.: Percentage of times teams choose among available actions

	Team 1	Team 2	Team 3
Team 1	0.35	0.24	0.41
Team 2	0.14	0.45	0.41
Team 3	0.33	0.28	0.39

Though this problem is small, with only three subsystem teams, the same methodology can be used in larger problems with more number of teams. With a bigger matrix, methods of decomposition which use design structure matrices, such as those discussed in chapter 2, can be used to set up organizational network.

6.1.2 Flexibility In Modeling Design Processes

Design progresses in jumps with short periods of rapid growth in design knowledge alternating with periods of highly structured actions. This manner of progress, with alternating periods of rapid knowledge growth and refinement, can be called as epoch-shock sequence in which epochs are the "valleys" of largely consistent behavior which are separated by the "peaks" of shock [56]. The transition between epochs may result from technological, mission, context, or policy shocks. For example, while all system designs would begin with subsystem teams exploring alternatives, a major technological breakthrough or a new mission opportunity may prompt them to begin the exploitation of current knowledge. As design progresses, subsequent epochs indicate both increasing technical sophistication as well as increasing cost of activities. The management of these epochs will influence the cost of complex systems development and their ability to meet mission requirements.

In our model, the transitions could be the result of the principal changing its targets during the process. Because the principal does not know of its Pareto front, it may, on the discovery of a new design choose to update targets if this new design provides an increase in value which is enough to offset the cost of doing so. When discussing the results of Case 5 in our synthetic problem, we found that this case took, on average, 14 iterations to converge. Consider the first 10 iterations of one of the runs of this problem:

$$\boxed{1 \to 2 \to 2 \to 1 \to 1 \to 2 \to 1 \to 3 \to 3 \to 2}$$

This sequence indicates that initially, the principal accepts the design from team 1 as its best design and hence its target. In the next iteration, it changes its target to that of team 2, then back to team 1 in fourth iteration, and so on. This is an hypothetical scenario in which the principal does not have any requirements of its own and does not solve for its value function besides evaluating it for the designs provided by the teams. In reality, the principal would have a set of targets which

may remain fixed or change during the process in response to some shocks such as new customer requirements or availability of new technology. The cost to the principal is that when changing targets it would have to compensate the teams for their loss of value. The message here is that with appropriate incentives, design teams can collectively achieve system-level targets even if these targets are changed during the course of the process.

6.1.3 Using Value-based Approach In Complex Systems Design

Value-based models of design provide an organization with the ability to select a process with the highest payoff. However, not only are models of value estimation still lacking (see section 2.2), during the process of design, estimates of actual value will fluctuate depending on factors such as agents' decisions, sudden changes in requirements, etc. Phenomena such as delays in communication, inaccuracies in data, etc. further complicate value estimation. In response to these shortcomings, designers have to frequently update their value estimates. This flexibility of value estimation means that the designers have to continually update their policies in response to changing information; in other words, design teams have to behave as adaptive agents who update their policy whenever doing so results in an increase in value.

We discuss value-based approaches from two perspectives – one, the risk-taking ability of human designers, since the risk-aversion of designers is a key feature argued in favor of using value-driven design over requirements-driven design, and two, a principal's ability to influence teams' value models by changing rewards functions.

A representation of human behavior, through simulation and analysis, helps to identify factors that will improve the humans' decision-making. Such modeling must be cognizant of the environment in which the designers operate. However, human behavior modeling faces challenges such as a lack of common vocabulary across various disciplines within physical and social sciences, the domination of modeling by physical scientists as opposed to social scientists, and no common authority of data

management [57]. A lack of consistent data also makes verification and validation activities difficult. In brief, social modeling poses new challenges which require a rethinking of modeling, analysis, and validation approaches.

An attribute of human behavior used to argue in favor of value-driven design is the human designers' risk tolerance. Collopy [22] examined the effect of assigning objectives on decision-making stating that if the current design is likely to achieve the requirements then the designers are like to become risk averse whereas if they are likely to miss requirements then they will prefer risk. Vermillion and Malak [31] also modeled risk behavior within a principal-agent model of task allocation. They state that modeling risk behavior is a factor when determining an incentive scheme because the probability of meeting requirements is dependent on the effort expended on the task. We modeled all agents as risk-neutral, i.e., they follow the linear curve in Fig. 2.4 because such an agent simply wants to maximize his payoffs. However, as an alternative to these utility curves, one can use prospect theory as a basis for setting up risk behaviors. Arguably, the performance team in the aircraft design example discussed is an extreme version of use of this theory because this team, without evaluating discounted values like other teams, simply rejected any design which did not meet requirement of minimum range, whereas it was indifferent to any design which exceeded its desired range.

When using the value-driven approach to decision-making, an organization has to ensure that it sufficiently explores the design space before advancing to more detailed design. Our modeled approach of value iteration can be replaced with other methods from reinforcement learning in order for the agents to consider discounted future rewards for their decision-making. Particularly, the approaches of Q-learning and SARSA, both based on Q-values, have an additional parameter of learning rate which can help control the amount of exploration an agent carries out. When the learning rate is high, the teams give more importance to the future rewards and thus explore more of the design space because they consider more alternatives, whereas when the rate is low, the agents use only the most recent information and exploit

it for design refinement. This is similar to the role that the "temperature" setting in simulated annealing plays. The message is that organizations need to both define appropriate value functions and how they will use these functions for decision-making.

6.1.4 Examples of Questions a Model of Design Process Can Answer

Lee and Paredis [8] posed some questions that can be asked when setting up heuristics for design. For a process-system, setting up simulation models is a useful way to answer at least a few of those questions. Consider, for example, some of the following questions:

1. Questions on setting up an organizational structure:

    - How does the structure of organization affect the architecture of the product being developed?

      This effect is called the mirroring hypothesis and can be assessed by comparing different organization structures and their resultant products and looking for correlation between their structures.

    - If the structure of an organization affects product outcome, how do we set up an organization?

      We proposed an approach in this work. Use simulation to identify which links among designers have higher frequency of information exchange and place those designers in close proximity.

2. Questions on setting up a design process

    - How does the choice of process affect the product being developed?

      A model of the design process-system can be integrated with models of the product being designed and thereby allow for the simultaneous optimization of both the product and process architectures; we discussed this in chapter 2.

- How quickly does a change in the process being used affect the architecture of the product? Do some processes make the product more flexible than others? Can a process-system model identify the optimal path of design refinement?

  Organizations may modify their processes to affect change in product architecture, and studying the dynamics of process-systems may help evaluate how quickly the product can evolve. Comparing the value of alternative process-system models will help identify the best choice of processes to utilize.

3. Questions on setting up heuristics for control of process

   - How can heuristics used to control the decisions of the designers be identified?

     The efficacy of heuristics in design can be studied by comparison of scenarios which simulate use of heuristics versus those that do not; the comparison would be based on the measures of performance of the systems being developed.

   - How does the system-level designer set incentives for the design teams?

     By setting a rewards policy. This could be used to control flow of information among teams and to incentivize decision-making which favors system-level objectives. Our modeling achieved this by giving rewards to teams in proportion to how closely they met system-level targets.

   - How does the system-level designer and the design team balance exploration with exploitation?

     We used a discount factor when modeling teams' assessment of value of available tasks. This factor can be controlled by the system-level designer to encourage exploration with the objective of long-term gains versus exploitation to provide more immediate solutions. Provision of rewards can

also be a means to achieve this as large rewards can be given for a specific solution rather than design space exploration.

6.2 Qualitative Research in Engineering Design

So far, the discussion has centered on use of computational models as a means to simulate and analyze alternative processes for design. Such quantitative approaches as value-based design provide an objective basis when making decisions. However, the models used in quantitative analysis are not free of the modeler's subjectivity who ultimately decides on the form of the models and the questions posed to them. Quantitative models that are frequently utilized in modeling of design processes accept this necessary compromise in order to convey the results succinctly. However, in some cases, quantitative models may not even be available. For example, Price el al. [58], while discussing the application of systems engineering to aircraft design, point out that several of the disciplines involved may not have quantitative models available. This is particularly true for the non-technical disciplines for which modeling methods are lacking. One of the non-technical disciplines for which quantitative analysis is difficult is that of human behavior modeling. Techniques for human behavior modeling evaluate one or a few attributes of decision-making, making use of such mathematical tools as uncertainty modeling or Bayesian inference.

In our Case 3 in the synthetic problem, the addition of margins by teams was based on a study by Austin-Breneman et al. [52], who modeled designers biases in form of addition of margins during decision-making in a simulation and concluded that the addition of margins is detrimental to the final design outcome. They note that it is common for initial margins to be of an order of 30% and to gradually reduce as design progresses. The quantification of amount of margin is based on their survey of real systems engineering practitioners. Thus, they relied on qualitative analysis to inform their quantitative modeling. In this section, we will first discuss, briefly, the basics of qualitative analysis, followed by its utility in engineering design applications.

During complex systems design, the social aspects of interactions among engineers can be as important as the technical ones [59,60], and poorly structured organizations can inhibit information exchange and therefore affect the quality of design outcome. Qualitative analysis is particularly suitable for studies of social aspects within a design organization because whereas quantitative analysis seeks to describe design process objectively and imposes implicit bounds on description because of limitation to our ability to measure things, qualitative discussion is able to capture a broader range of phenomena with dense and descriptive accounts of an organizations culture. A form of qualitative analysis called inductive analysis, in particular, does not even start from preset attributes of the subject under study and identifies them during the process of analysis. The dense descriptions of qualitative analysis also require fewer abstractions.

Qualitative models have other advantages over quantitative ones. First, quantitative models may be sensitive to their parameter settings so that small changes in the initial conditions may lead to large differences in the results. Qualitative models, on the other hand, are more robust to such changes mainly because they involve longer descriptions of the context. Second, as we add more details to quantitative models, we may also be able to make more precise conclusions, and in the extreme cases, addition of fidelity to models may lead to completely different conclusions. Qualitative models are more likely to hold on to their conclusions even when further details are added.

Daly et al. [60] noted that the goals of qualitative research are illumination, understanding, and extrapolation of findings to other similar situations, and that these emerge gradually during the process of analysis. Unlike quantitative methods, not only are the research questions of qualitative analysis grounded in literature and available data, but the researcher is also a part of the "instrumentation" and the resultant analyses are reported as rich and thick descriptions. Due to these differences, qualitative analysis can help identify new phenomena that quantitative analysis could not, and thus serves as a useful counterpart to quantitative inquiry.

Szajnfarber and Gralla [56] focused on theory building and laid out a process for using qualitative methods to study engineering systems. The authors contended that as the scope of systems engineering research expands to include human and organizational issues, qualitative methods – which are well-established in other disciplines – will need to play an increasing role. As with quantitative research, validity plays a critical role in qualitative research, but the same standards need to be interpreted and implemented slightly differently.

6.2.1 Conducting Qualitative Research

The key benefit and role of qualitative research is that it can be useful in identifying values of parameters of quantitative models, such as the degree of risk tolerance of human designers. Qualitative research is inductive in nature and the research questions, hypotheses, method of analyses, etc. develop during the process of conducting research. Thus, the researcher has to maintain flexibility to change any component of research during the course of analysis. Marshall and Rossman [61] identified characteristics of qualitative research including that it takes place in the natural world, uses multiple methods that are interactive and humanistic, focuses on context, is emergent rather than tightly prefigured, and is fundamentally interpretive.

The key distinguishing feature of qualitative study, as opposed to quantitative studies is that it is reflexive in nature, which means that the various activities such as collecting and analyzing data, developing a theory, identifying research questions or modifying them, etc. go on simultaneously and affect one another. Similar to design of complex systems, this process iterative with no single prescribed order of tasks that may lead from beginning to end. We can consider a model of qualitative research to be comprised of five components [62]:

1. *Goals:* which identify the objectives, contributions, and the motivations of the study

2. *Conceptual framework:* which is the identification of context, background knowledge, experiences, etc. which inform the conduct of study

3. *Research questions:* which elaborate on the desired insights from the study

4. *Methods:* which are the approaches and techniques to be employed in the conduct of the study

5. *Validity:* which establishes link between theory and reality along with identifying the potential failures of the model and alternative interpretations

Each of these factors influences and is influenced by one another. A researcher clarifies the goals of qualitative research by addressing the question of "why do this research?" The questions such as "what do we want from this study?" and "what do we need to do for this study?" are addressed by the next three components of conceptual framework, research questions, and methods. And finally, the validity component addresses the how of qualitative study by asking, "how can we be wrong (or right)?"

While such a categorization can provide a useful starting point, any qualitative research is designed specific to the context for which it is setup and the purpose that a researcher seeks to answer. Further details on qualitative methods of study can be found in references [61–63].

While qualitative studies are not included as part of work presented herein, such studies can be taken up as a complement to modeling of design teams. Surveys and other such instruments of qualitative analysis can be used to derive such features of an organization as the preference structures of the teams which would be used to select value models, and the risk behaviors of the designers involved. In turn, the needs of the modeling can be used to frame the research questions to be answered by qualitative research. Thus, qualitative research has the potential to contribute to the advancement of design process modeling and analysis.

6.3 Summary of Modeling Framework

All models make some simplifying assumptions, and the most useful models capture the most important features of the system they are modeling with the highest possible accuracy. A model which captures multiple levels of hierarchy considers a broader context at once, allowing for simultaneous analysis of people at the various levels of the organization. Such models can potentially provide a template for representation of decision problems that can face designers at multiple levels of hierarchy, with the result that solutions which answer questions at one level, can adapted at another level with modifications.

We setup a bi-level organization model by using a principal-agent model, where the division of tasks between a principal and the agents parallels discussion of decomposition of the system itself. For comparison, the Bi-level Integrated System Synthesis (BLISS) is a method for optimization of engineered systems by decomposing the design problem into multiple autonomous local optimization problems [49]. The problem is setup such that the system-level optimizer operates on a smaller set of global design variables whereas the subsystem problems can potentially handle larger number of detailed design variables. Because coordination occurs by exchange of optimum sensitivity information, the algorithm can fit a variety of organization structures. The performance of system analysis at each iteration means that the procedure can be terminated at any point of the designer's choosing, similar to our approach. However, in BLISS, the system-level and subsystem problems alternate with the latter problem being solved first followed by the former system-level problem performing the task of integration. Each team is thus bound by a strict schedule of their local problem-solving. Further, the use of gradient methods means that any non-convexity would make the problem highly dependent on the starting point of optimization and that every iteration leads to an improvement in design. Both of these features are different in our model of adaptive agents. The agents are free to continue solving their local problem and use information from any other agent only when it is

beneficial for them to do so, and the result of such distributed decision-making is that at the system-levels, the decisions of any one agent can lead to lower value during the course of the process.

In summary, our framework addresses all three roles of a design process model that we discussed in section 6.1. First, simulation using the proposed framework gives an organizational structure as an output. This means that we do not have to specify a strict hierarchy as an input to our model. Second, because we set up a fully connected network of agents who make independent decisions on which of their neighbors to interact with and when, our framework is flexible to any organizational structure and changes to this structure that needs to be modeled. Finally, our set up of agents using discounted rewards is a way to use value-based approach for decision-making. The proposed agent model provides the flexibility to specify behavioral factors such as preferences, risk tolerance, etc.

7. CONCLUSIONS AND FUTURE WORK

We have presented a framework for modeling a complex systems design process in a bi-level organization of a single system-level designer and multiple subsystem teams. The teams all functioned as semi-autonomous agents who made decisions to maximize their local utilities while being guided by the system-level designer by provision of rewards.

Our model demonstrates how design teams can learn from available information such as the current designs of other teams and make decisions that lead to highest value obtained from the process. Using the approach of comparing between alternative actions on the basis of future discounted rewards obtained from the outcomes of each of the actions, we do not need a model of dynamics of the process to be able to estimate values. Instead, the value models of teams are functions of their local subsystem's attributes such that maximizing value would lead to improved system performance measures. The organization's value, therefore, results from an outcome of the collective decisions of the teams' local decisions.

We demonstrated the application of this framework to an example problem of aircraft design where a system-level designer had the objective of operating cost minimization while three subsystem teams minimized weight and maximized the lift-to-drag ratio and mission range. We compared the results of using our framework to this problem with the results from using simulated annealing algorithm to solve the same problem.

Thus, we have developed and demonstrated the following capabilities and features of a complex systems design problem. One, we showed modeling of design teams as semi-autonomous agents with their local preferences. This model can accommodate the teams' behavioral attributes such as their risk tolerance and their decision-making on the choice from among an available set of actions. Two, we set up a network of

interaction between multiple teams and system-level designer which is flexible with respect to the interaction links which exist within the organization. Thus this network evolves as design progresses and can be set up to represent different organizational architectures. Finally, we showed how a system-level designer can guide the decision-making of design teams by the use of rewards as incentives. The proposed framework can also accommodate other incentive schemes for guiding design decisions.

Table 7.1 summarizes the research questions and their outcomes from this dissertation.

Table 7.1.: Research questions and outcomes of this research

Research question:	**How can we model the effects of available information on the design teams behavior?**
Hypothesis:	Designers decision-making can be studied by modeling them as learning agents that update their design policies in response to available information.
Work done:	We modeled and simulated the distributed decision-making of teams who use the value iteration algorithm to select the optimal design process strategy.
Research question:	**How can a system-level designer identify heuristics for control of design teams?**
Hypothesis:	The system-level designer can setup rewards for meeting requirements, which work as incentives to encourage behavior in favor of global objectives.
Work done:	We solved a Principal-Agent model for interactions across levels of organizational hierarchy and which we used to identify rewards for teams.

Ultimately, the process-system comprised of a network of autonomous and self-interested design teams and the tools and methods that they use for systems development is itself complex. Its management needs a systems-thinking approach which is holistic, non-linear, and iterative in nature. This is in contrast with the linear, procedural systems engineering principles used for the development of the products. Modeling and simulation frameworks such as those presented in this work are a way to study the dynamics of design process-systems. Particularly, we have shown, that *a model of design teams as learning agents who interact and update their behavior using design variable and payoff information, when guided by appropriate heuristics from a system-level designer, simulates the dynamics of a design process.*

7.1 Future Work

Several advancements can be made to modeling done in this dissertation. In our modeling, while the teams chose from a predefined set of available actions in each state, they made their decisions under a single objective function. Future additions can make teams' value function a multi-objective function which the teams can use to either calculate a single scalar value or can choose to use one from among many different value functions. For example, early in the design process, when the teams are exploring the design space using low fidelity models, computational cost may not be the most important value function because initially achieving feasibility may be the only requirement. Later, computational cost may be more useful as a value function. In other words, both the value functions and the set of available actions to the teams can change as the process advances.

A good modeling approach will serve to provide explanation of why certain policies work while other do not. Thus decision-makers can evaluate several from a set of policies to see which one works best for their circumstances. Further, as the organization collectively gathers experience with complex systems design, the model can adapt to reflect the accompanying behavior changes. This would be yet another

feature of such modeling approach, and one which will provide context-specific decision support to the organization in which it is used. Ultimately, in enabling learning, the modeling can help the organization at all levels to quickly adapt to changes in the environment, which all designers perceive as changes in input information, with the result that the best possible response can be decided and acted on quickly. This would be another valuable contribution of this modeling approach to the complex systems design process.

This dissertation proposes a computational model of a design organization including the teams involved, their interactions, and decision-making. As with all modeling, several assumptions were necessary to make this study tractable. Further, validation of this work was done mainly by comparing with an established optimization method. However, future work can look at use of real-world data while modeling designers' behaviors and decision-making. The analysis of this data can be qualitative in nature, such as we discussed in brief in a preceding chapter. The, using the results of such data analysis, more refined models of human behavior and decision-making can be set up to simulate the dynamics of complex systems design processes.

REFERENCES

REFERENCES

[1] Mark EJ Newman. Complex systems: A survey. *arXiv preprint arXiv:1112.1440*, 2011.

[2] Christina L Bloebaum, Paul D Collopy, and George A Hazelrigg. Nsf/nasa workshop on the design of large-scale complex engineered systemsfrom research to product realization. In *Proceedings of the 14th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, Indianapolis, IN, Paper No. AIAA-2012-5572*, 2012.

[3] Abhijit Deshmukh and Paul Collopy. Fundamental research into the design of large-scale complex systems. In *13th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, Fort Worth, TX, AIAA, AIAA-2010-9320*, pages 486–494, 2010.

[4] Timothy W Simpson and Joaquim RRA Martins. Multidisciplinary design optimization for complex engineered systems: report from a national science foundation workshop. *Journal of Mechanical Design*, 133(10):101002, 2011.

[5] INCOSE Handbook. Incose systems engineering handbook: A guide for system life cycle processes and activities, version 4. *International Council on Systems Engineering*, 2014.

[6] DN Flynn. Building a better model: a novel approach for mapping organisational and functional structure. *Procedia Computer Science*, 44:194–203, 2015.

[7] David Ullman. *The mechanical design process*. McGraw-Hill Higher Education, 2015.

[8] Benjamin D Lee and Christiaan JJ Paredis. A conceptual framework for value-driven design and systems engineering. *Procedia CIRP*, 21:10–17, 2014.

[9] Alan MacCormack, Carliss Baldwin, and John Rusnak. Exploring the duality between product and organizational architectures: A test of the "mirroring" hypothesis. *Research Policy*, 41(8):1309–1324, 2012.

[10] James T Allison. Complex system optimization: A review of analytical target cascading, collaborative optimization, and other formulations. *MS thesis, Department of Mechanical Engineering, University of Michigan, Ann Arbor, MI*, 2004.

[11] James Moffat. *Complexity theory and network centric warfare*. DIANE Publishing, 2010.

[12] J.E. Gibson, W.T. Scherer, and W.F. Gibson. *How to Do Systems Analysis*. Wiley Series in Systems Engineering and Management. Wiley, Hoboken, N.J., 2007.

[13] David C Wynn and P John Clarkson. Process models in design and development. *Research in Engineering Design*, pages 1–42, 2017.

[14] John S Gero. Design prototypes: a knowledge representation schema for design. *AI magazine*, 11(4):26, 1990.

[15] Yan Jin and Raymond E Levitt. The virtual design team: A computational model of project organizations. *Computational & Mathematical Organization Theory*, 2(3):171–195, 1996.

[16] Mark Klein, Hiroki Sayama, Peyman Faratin, and Yaneer Bar-Yam. The dynamics of collaborative design: Insights from complex systems and negotiation research. In *Complex Engineered Systems*, pages 158–174. Springer, 2006.

[17] Michael D Griffin. How do we fix systems engineering? In *61st International Astronautical Congress*, pages 1–9, 2010.

[18] Terrance Carl Wagner. *A general decomposition methodology for optimal system design.* PhD thesis, 1993.

[19] Paul D Collopy. Economic-based distributed optimal design. *AIAA Paper*, 4675:2001, 2001.

[20] Nestor F Michelena and Panos Y Papalambros. A hypergraph framework for optimal model-based decomposition of design problems. *Computational optimization and applications*, 8(2):173–196, 1997.

[21] Andrew Kusiak and Juite Wang. Decomposition of the design process. *Journal of Mechanical Design*, 115(4):687–695, 1993.

[22] P Collopy. Adverse impact of extensive attribute requirements on the design of complex systems. *7th AIAA Aviation Technology, Integration, and Operations (ATIO) Conference*, 2007-7820, 2007.

[23] Paul D Collopy and Peter M Hollingsworth. Value-driven design. *Journal of aircraft*, 48(3):749–759, 2011.

[24] Julie Cheung, James Scanlan, James Wong, Jennifer Forrester, Hakki Eres, Paul Collopy, Peter Hollingsworth, Steve Wiseall, and Simon Briceno. Application of value-driven design to commercial aeroengine systems. *Journal of Aircraft*, 49(3):688–702, 2012.

[25] Paul Collopy. Aerospace system value models: A survey and observations. In *AIAA Space 2009 Conference & Exposition*, volume 2009-6560, Pasadena, California, 2009. American Institute of Aeronautics and Astronautics.

[26] Lyra Colfer, Carliss Y Baldwin, et al. The mirroring hypothesis: Theory, evidence and exceptions. *Harvard Business School Finance Working Paper*, (10-058), 2010.

[27] Joel B Predd, Daniel N Osherson, Sanjeev R Kulkarni, and H Vincent Poor. Aggregating probabilistic forecasts from incoherent and abstaining experts. *Decision Analysis*, 5(4):177–189, 2008.

[28] Natalia Alexandrov and Robert Lewis. Reconfigurability in mdo problem synthesis, part 1. In *10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, page 4307, 2004.

[29] Natalia Alexandrov and Robert Lewis. Reconfigurability in mdo problem synthesis, part 2. In *10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, page 4308, 2004.

[30] Ben P Wise, Mary McDonald, Lisa M Reuss, and Jesse Aronson. ATM human behavior modeling approach study, 2001.

[31] Sean D Vermillion and Richard J Malak. Using a principal-agent model to investigate delegation in systems engineering. In *ASME 2015 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, pages V01BT02A046–V01BT02A046. American Society of Mechanical Engineers, 2015.

[32] Dan Braha and Yaneer Bar-Yam. The statistical mechanics of complex product development: Empirical and analytical results. *Management Science*, 53(7):1127–1145, 2007.

[33] Tomonori Honda, Francesco Ciucci, Kemper E Lewis, and Maria C Yang. Comparison of information passing strategies in system-level modeling. *AIAA Journal*, 53(5):1121–1133, 2015.

[34] Francesco Ciucci, Tomonori Honda, and Maria C Yang. An information-passing strategy for achieving pareto optimality in the design of complex systems. *Research in Engineering Design*, 23(1):71–83, 2012.

[35] Abdeslem Boukhtouta, Jean Berger, Abraham George, and Warren B Powell. An approximate dynamic programming approach for semi-cooperative multi-agent resource management. *Journal of Artificial Intelligence and Soft Computing Research*, 2, 2012.

[36] Zhemei Fang, Navindran Davendralingam, and Daniel DeLaurentis. Multistakeholder dynamic optimization for acknowledged system-of-systems architecture selection. *IEEE Systems Journal*, 2018.

[37] Philipp Renner and Karl Schmedders. Dynamic Principal-Agent Models. *SSRN Electronic Journal*, 2016.

[38] Andrew Olewnik and Kemper Lewis. A decision support framework for flexible system design. *Journal of Engineering Design*, 17(1):75–97, 2006.

[39] Daniel Kahneman and Amos Tversky. Prospect theory: An analysis of decision under risk. In *Handbook of the fundamentals of financial decision making: Part I*, pages 99–127. World Scientific, 2013.

[40] Sven Ove Hansson. Decision theory: A brief introduction, 2005.

[41] David L Poole and Alan K Mackworth. *Artificial Intelligence: foundations of computational agents*. Cambridge University Press, 2010.

[42] D Warner North. A tutorial introduction to decision theory. *IEEE transactions on systems science and cybernetics*, 4(3):200–210, 1968.

[43] Joaquim RRA Martins and Andrew B Lambe. Multidisciplinary design optimization: A survey of architectures. *AIAA journal*, 51(9):2049–2075, 2013.

[44] Hisham ME Abdelsalam and Han P Bao. Re-sequencing of design processes with activity stochastic time and cost: An optimization-simulation approach. *Journal of Mechanical Design*, 129(2):150–157, 2007.

[45] J. Rogers. DeMAID/GA - An enhanced design manager's aid for intelligent decomposition. In *6th Symposium on Multidisciplinary Analysis and Optimization*, Bellevue,WA,U.S.A., 1996. American Institute of Aeronautics and Astronautics.

[46] Soo-Haeng Cho and Steven D Eppinger. A simulation-based process model for managing complex design projects. *IEEE Transactions on engineering management*, 52(3):316–328, 2005.

[47] Yanjun Qian, Jun Lin, Thong Ngee Goh, and Min Xie. A novel approach to dsm-based activity sequencing problem. *IEEE Transactions on Engineering Management*, 58(4):688–705, 2011.

[48] Raja Parasuraman, Thomas B Sheridan, and Christopher D Wickens. A model for types and levels of human interaction with automation. *IEEE Transactions on systems, man, and cybernetics-Part A: Systems and Humans*, 30(3):286–297, 2000.

[49] Jaroslaw Sobieszczanski-Sobieski, Jeremy S Agte, and Robert R Sandusky. Bilevel integrated system synthesis. *AIAA journal*, 38(1):164–172, 2000.

[50] Roger B Myerson. Optimal coordination mechanisms in generalized principal–agent problems. *Journal of mathematical economics*, 10(1):67–81, 1982.

[51] Kalyanmoy Deb, Lothar Thiele, Marco Laumanns, and Eckart Zitzler. *Scalable Test Problems for Evolutionary Multiobjective Optimization*. Springer-Verlag, London, 2005.

[52] Jesse Austin-Breneman, Bo Yang Yu, and Maria C Yang. Biased information passing between subsystems over time in complex system design. *Journal of Mechanical Design*, 138(1):011101, 2016.

[53] Lloyd R Jenkinson, Paul Simpkin, and Darren Rhodes. *Civil Jet Aircraft Design*. Arnold London, 1999.

[54] William L Goffe, Gary D Ferrier, and John Rogers. Global optimization of statistical functions with simulated annealing. *Journal of econometrics*, 60(1-2):65–99, 1994.

[55] Angelo Corana, Michele Marchesi, Claudio Martini, and Sandro Ridella. Minimizing multimodal functions of continuous variables with the simulated annealing algorithm corrigenda for this article is available here. *ACM Transactions on Mathematical Software (TOMS)*, 13(3):262–280, 1987.

[56] Zoe Szajnfarber and Annalisa L Weigel. A process model of technology innovation in governmental agencies: Insights from nasas science directorate. *Acta Astronautica*, 84:56–68, 2013.

[57] SK Numrich and Andreas Tolk. Challenges for human, social, cultural, and behavioral modeling. *SCS M&S Magazine*, 1(1):2010–01, 2010.

[58] M Price, S Raghunathan, and R Curran. An integrated systems engineering approach to aircraft design. *Progress in Aerospace Sciences*, 42(4):331–376, 2006.

[59] Anna-Maria Rivas McGowan, Panos Y Papalambros, and Wayne E Baker. A framework of working across disciplines in early design and R&D of large complex engineered systems. In *International Conference on Engineering Design (ICED) 2015*, Milan, Italy, 2015.

[60] Shanna Daly, Anna McGowan, and Panos Papalambros. Using qualitative research methods in engineering design research. In *DS 75-2: Proceedings of the 19th International Conference on Engineering Design (ICED13), Design for Harmonies, Vol. 2: Design Theory and Research Methodology, Seoul, Korea, 19-22.08. 2013*, 2013.

[61] Catherine Marshall and Gretchen B Rossman. *Designing qualitative research*. Sage publications, 2014.

[62] Joseph A Maxwell. Designing a qualitative study. *The SAGE handbook of applied social research methods*, 2:214–253, 2008.

[63] Johnny Saldana. *Fundamentals of qualitative research*. Oxford University Press USA, 2011.

VITA

VITA

Kushal Moolchandani earned his Bachelor of Engineering in Aeronautical Engineering from PEC University of Technology, Chandigarh, India, and his Master's degree in the School of Aeronautics and Astronautics at Purdue University, under Dr. Daniel A. DeLaurentis. His research interests include aircraft design and optimization, complex systems design, air transportation systems, and systems engineering.

In the course of Bachelor's degree, he worked on the conceptual design and sizing of a 100 seat commercial transport aircraft aimed at serving the Indian market. He also worked on the design and sizing of an unmanned aerial vehicle for high altitude surveillance. At Purdue, he worked on assessment of new aircraft technologies on aviation's environmental impact. This project formed the core of his Master's thesis and led to the development of the Fleet-level Environmental Evaluation Tool (FLEET). Additionally, he worked on a project to study the effects on network performance as a result of the preferences and consequent decisions of the autonomous nodes.

Besides research, Kushal enjoys swimming, reading, and traveling. He also practices Karate and ballroom dancing during his free time.