Unlocking Organizational Potential: A Computational Platform for Investigating Structural Interdependence in Design

Jesse Olson
Northrop Grumman Corporation,
Johnstown, PA 15901
e-mail: jesse.olson@alumni.cmu.edu

Jonathan Cagan
Department of Mechanical Engineering,
Carnegie Mellon University,
Pittsburgh, PA 15213
e-mail: cagan@cmu.edu

Kenneth Kotovsky
Department of Psychology,
Carnegie Mellon University,
Pittsburgh, PA 15213
e-mail: kotovsky@cmu.edu

A team's design—the structuring of its resources and flows of knowledge—is an important element determining its effectiveness. An essential element in achieving a team's problem-solving potential is the role that interdependence, in both the task and the organization, plays in determining the dynamic and emergent system-level properties of the organization. In this paper, we present a computational platform for experimentally investigating the influence of informational dependencies found in the design of a complex system for exploring their role in determining system behaviors and performance. The approach presented in this paper is a multiagent simulation of the conceptual design of space mission plans by Team X, an advanced project design group at NASA's Jet Propulsion Laboratory. The algorithm is composed of rich descriptive models of both the team-types and timing of interactions, collaborative methods, sequencing, rates of convergence- and the task-primary variables, their behaviors and relations, and the approaches used to resolve them. The objective is to create an environment of interaction representative of that found in actual design sessions. Better understanding how the dynamics arising from organizational and domain interdependencies impact an organization's ability to effectively resolve its task should lead to the development of guidelines for better coping with task complexities, suggest ways to better design organizations, as well as suggest ways for improving the search for innovative solutions. [DOI: 10.1115/1.3066501]

1 Introduction

So much of a team's problem-solving effectiveness hinges on its design—its context, culture, processes, and models. Changing a team's design changes what, how, and when information will be related—effectually changing the organization's problem representation [1]. Unfortunately, the ways in which a team's design may influence its collective representation are numerous, and the manner in which these changes may influence design effectiveness is not always recognized nor is the extent of the potential impact typically perceived [1].

In this paper, we present a computational platform designed to experimentally investigate the implications of these informational dependencies. This platform is designed to simulate both the complexity and dynamics of information interaction in a real world design environment: a collaborative design group at NASA's Jet Propulsion Laboratory (JPL), Team X, and the conceptual design of a space mission plan. Accordingly, the models presented are imbued with social and domain definitions and structures analogous to those populating the actual design environment, each with the specific intent of creating both the behavioral richness of individual parameters and an environment of interaction that is analogous to that occurring in Team X design sessions. They include thousands of domain variables, dozens of problem-solving approaches, and numerous methods, individualized to each agent, governing the types and timing of social interactions occurring among agents. This type of representational depth is rare in computational organization studies but is an essential element to the utility of this simulation as it enables a more critical look at the complex set of interactions occurring in real world design.

The goal in creating this platform is not to find the optimal way of solving this particular task, nor is it to simulate humans or theorized cognitive processes. Rather, it is to enable theory building and hypothesis generation in regard to the patterns of interaction arising in design contexts through controlled experimental exploration of a team-based computational representation, to allow us to study the impact that team and domain structures have on the design process, and, in the long term, to identify methodologies for effectually managing these organizational/domain complexities.

The object of this study is to move toward a more rigorous understanding of organizational problem solving to better understand how to more accurately describe, predict, and ultimately design an organization's fitness for resolving a given task. The focus of our efforts centers on identifying the mechanisms and patterns underlying organizational problem-solving behaviors, the aim of which is to gain a more precise understanding of the basic principles that govern organizational problem-solving potential and to develop a set of generalizable metrics for capturing the performance implications of organizational designs. The approach taken is based on the premise that parallels in the structure of design organizations and the design task can be revealed and that if the simulation of a complex task using this approach is successful, it should provide a basic approach as well as a potential tool for effectively structuring design organizations in relation to the design task in an optimally directed manner.

This paper highlights key features of this simulation.

Contributed by the Design Theory and Methodology Committee of ASME for publication in the JOURNAL OF MECHANICAL DESIGN. Manuscript received November 28, 2007; final manuscript received December 2, 2008; published online January 26, 2009. Review conducted by Janet K. Allen. Paper presented at the ASME 2006 Design Engineering Technical Conferences and Computers and Information in Engineering Conference (DETC2006), Philadelphia, PA, September 10–13, 2006.

1.1 Motivation: Interdependencies in Organization and Task. Because of the fundamentally interdependent nature of organizational constituents, organizations are frequently viewed as inherently complex systems [2]. Indeed, interdependence is central to the conception of what an organization is and is capable of [3]. Through nonsimple patterns of combination, organizations frequently display dynamic and nonlinear behaviors. These behaviors make analytical analyses of organizations intractable and imprecise [4]. However complicating, the complex characteristics of organizations also give rise to important organizational properties. Increasing organizational complexity enables more complex organizational level behaviors, resulting in a corresponding rise in the diversity of organizational behaviors possible [5,6]. Correspondingly, increases in organizational performance have been observed with increasing complexity, emergent properties in some cases enabling performance potentials greater than and unique from the aggregate of individuals [7,8]. The essential behaviors and performance of organizations, therefore, are both complicated by and reliant on the nonlinearities arising from interdependence.

Complex engineering domains are also characterized by the existence of strong interdependencies. These can make the process of designing such a system quite challenging. Nonlinear relations and compounding coupling effects can make the system behavior unpredictable. System level, even metasystem, trade-offs may be difficult to track and evaluate. At times, these interdependencies can even make it difficult to converge to an internally consistent solution, notwithstanding one that meets the myriad design requirements prescribed. But, in a manner similar to that discussed for organizations, the complex relationships of a design space may not only be required but desirable in that they may lead to a system functionality not achievable in less complex design instances.

Beyond the relationships between organizational or between domain parts, there is an important underlying structural correspondence between the structure of a problem domain and the design of an organization [9,10]. A central determinant of the effectiveness of an organizational design is the structure of the organization's task. There is a strong bilateral relation between the structure of a task and the design of an organization, product interdependence influencing the degree and location of organizational interaction [11–15], and organizational interdependence correspondingly influencing the interdependencies of the solution architecture [16,17]. This relation is especially pronounced and its impact on the organization's effectiveness particularly compelling as the complexity of the task rises. While the parallel nature of organizational and problem structures has been studied (e.g., [18]), this paralleling of organization and task in the presence of complex, non-linear properties has been largely neglected in organizational research [17,19], and methodologies taking advantage of that correlation in order to design and manage more effective organizational networks are accordingly limited. For these reasons, an integrated approach is necessary to capture the intricate interplay between the two.

1.2 Simulating the Complex Interactions in Design. Better understanding how the dynamics arising from organizational and domain nonlinearities impact an organization's ability to effectively resolve its task should lead to the development of guidelines for better coping with task complexities as well as suggest ways for improving the design of an organization and its search for innovative solutions. With this objective in mind, a computational design algorithm simulating a collaborative design environment was developed and is summarized in this paper. The subject of our computational simulation is a design group affiliated with NASA's JPL known as Team X. Team X is the conceptual design group responsible for creating JPL's space mission proposals. Benchmarking Team X and space mission plans provides a medium wherein to define and verify a system whose behaviors are representative of the intricate set of interactions found in complex engineering design.

This simulation is an emulative model developed using a systems theory approach, where myriad individual social and domain properties were designed and set in an environment of interaction to collectively produce the more complex systemic properties of the simulated environment. Then, interwoven into this fabric of interacting elements is a network-based representation identifying the relationships between elements of both the social and domain spaces. In this approach, the elements of both social and domain spaces are represented in the form of a network graph or relational matrix. This type of representation is most commonly employed in the study of social systems, where it is used to analyze organizational properties and behaviors [7,20]. It is also used to analyze product architectures. The primary methodology in this domain is known as the design structure matrix (DSM), and it has found application as both a systems analysis tool, visualizing and discussing important relationships for product analysis and reengineering, and as a product management tool, tracing the impacts of decisions and as a consensus document [17,21].

Like our approach, virtual design team (VDT) looked at a computational model for organization design, also using DSM as one of the foundational tools [18]. VDT allocates resources, aligning an organizational chart with design activities through analysis of time, cost, and quality. Our work builds on this foundation but seeks to align the structure of a design team with the structure of the task at hand, in our case a very complex task, associating the cross communication of parts of the team with the design task requirements and execution. Our approach uses intelligent agents, each able to simulate and reason about their domain of expertise within the design team. This approach allows for a generalizable model and implementation.

The primary contribution of network methodologies is that they enable the investigator to move beyond simple statements about individual or small groups of parameters and to focus on the pattern of connections in the network as a whole [20]. This is central to the description of complex systems as the constraining power of such systems is not mediated solely through direct links but is a function of the whole [7]. In this simulation this network representation is used to both characterize and set the existence and distribution of relationships in the system to provide a basis for generating experimental permutations and a basis for quantitative analysis.

Viewing the organizational and domain spaces at a scale where all-important factors are abstracted to knowledge elements and their interaction, such as the above-described network representation, is akin to representing these spaces by their most rudimentary constituents. At this scale, the distinction between product architecture and social structure is removed, and the functional relationship between the two is inherent in the description of knowledge networks. An organization's social, resource, and decision driven dependencies—its individuals' expertise, collaborative procedures, communication technologies, authority and technical structures, and even personality and culture directed toward the characterization and resolution of its given problem—may all be represented in these rudimentary networks. These organizational knowledge networks may be compared directly to corresponding networks of the (perceived) reality of the problem domain. This knowledge-level view and network modeling of organizational problem solving was employed in this simulated approach for its promise in condensing an organization's many complexities into its core, essential relationships such that they may be correlated directly with the corresponding problem representation. In this way, the simulation may be used to explore the essential organizational properties leading to improved problem solving within the context of the domain being worked, and then to provide simple extendable measures from which to define

methods and tools to design into an organization its desired problem-solving properties.

There are several reasons for choosing a computational approach in investigating this relationship. First, in a computational approach the targeted organizational and domain behaviors may be generated and studied through the definition of basic organizational and problem building blocks, situating them in an environment of interaction wherein the behaviors arising from interdependencies may emerge naturally. Second, by taking a computational approach the structural characteristics of the two may be experimentally controlled and investigated. As in many other empirical and computational studies representing a single organization, we cannot claim the generality of our models' predictive potential prior to completing supplemental investigations into other industries and environments. However, we would expect that the system's behaviors resulting from the structural dependencies present in this test case would be analogous to those found in other design organizations. The models presented here provide a method to experimentally investigate and to design complex design environments and, we believe, are a significant step toward understanding the role of structural dependencies on design performance.

2 Case Study: Team X

2.1 Motivation. In April 1995, NASA's JPL formed an advanced project design team, Team X. The aim in forming the team was to improve and accelerate JPL's conceptual development of new missions through the employment of a collaborative dedicated process. There are several aspects of JPL's Team X and their work that make them an enticing candidate for this study. First, the team's task space is inherently highly coupled, and to capture the essential trade-offs and to validate the design require the consideration of a rich set of interdependencies. This drives the team's interactions and results in a process that is highly interactive and fast paced. Second, while the details of analyses and interactions vary from mission to mission, the process and structure of interaction remain predominately consistent. The dedicated procedures and personnel, collocated efforts, and concise time frame make the process repeatable and particularly amenable to observation. As Team X and space mission plans are used as both a framework for creating our base model and to ground the resulting simulation, we will now briefly describe the two.

2.2 Overview: Team X and Space Mission Plans. The Team X process is team intensive and fast paced, in which participating engineers from each of the representative subdivisions are collocated and work concurrently and collaboratively to design a space mission plan's defining characteristics. The team consists of a group of approximately 20 individuals—16 engineers with expertise in a particular subsystem such as propulsion or programmatics, a systems engineer, a team leader, and customer representatives (see Table 1 for a complete listing). The pool of subsystem designers used in Team X sessions typically have several years of experience in their field and participate regularly in the team's proposals. The process generally occurs over the course of 1 week (three sessions of 3 h each, with some pre- and postwork, held in a week's time), in which time the team conceptually defines and scopes a space mission plan.

The designs produced by the team are conceptual plans defining system architecture, designating resources, and scoping costs and technologies. Although conceptual, significant attention to the analytical trajectories of and interaction between subsystem parameters is required to ensure that the plan is feasible. The number of input variables transferred via the internal network, for example, ranges from a dozen to nearly 600 per subsystem. The representational space that Team X is working in is rich and highly interdependent. The types of studies undertaken range in type from interplanetary missions to telescopes, and their customers range from internal to JPL to private organizations and universities.

Table 1 Space mission plan subsystem divisions

Subsystems	Stewardship
Attitude control	Determine instruments and thrusters to achieve pointing requirements
Computer system	Choose computer equipment and determine computer system requirements
Configuration	Roughly layout subsystem components
Cost	Evaluate cost structure
Ground	Determine ground support needs: communication, infrastructure, and hardware
Instruments	Determine instruments and properties
Mission design	Determine trajectory and compute "delta v"
Power	Determine power system configuration and budgets
Propulsion	Choose and size propulsion system
Programmatics	Determine the development schedule and determine programmatic costs and risks
Science	Evaluate potential for scientific contribution
Software	Determine the required lines of code and determine software development costs
Structures	Estimate structural masses and identify special structural requirements
Telecom	Size and sequence telecom equipment
Thermal	Determine thermal system configuration and budgets
Trajectory Visualization	Create trajectory visualizations and compute geometric details of trajectory

A typical proposal proceeds as follows: The beginning of the first session begins with the customer representative who presents the mission scope and objectives. The team leader then starts outlining system-level design issues including the general approach, which increases in focus and detail over the sessions. This defines the major direction or architecture of the solution. From this, individual subsystem designers begin to define their subsystems using estimates derived from past studies and heritage missions. Subsystem designers then work continuously throughout the remainder of the process, listening and participating in the team leader discussion when it applies to their subsystem. There are also frequent pair-ups and conversations between subsystem designers, participants moving regularly back and forth between individual subsystem work, small group work, and orchestrated team discussions. Each subsystem designer maintains a spreadsheet containing core analyses and vital subsystem statistics. These spreadsheets are connected to an internal network so that each subsystem can send required outputs and can receive pertinent inputs from other subsystems throughout the process. The process culminates in a final report and frequently a presentation of the final results that outlines the major conceptual features and considerations of the mission design.

As a result of the interdependence between subsystems, the process is iterative. Once individuals have set up and initialized essential preliminary subsystems definitions, they enter an iterative loop updating, reconfiguring and further defining their subsystems as they receive updated system and subsystem information. As this process continues, the team progresses toward converging on a solution, the magnitude of design changes decreasing over time. Near the end of the study, the team moves into a rapid iteration, wrap-up phase, quickly updating to make the final fine scale modifications and finally converging to an internally consistent solution.

The team rate of convergence will vary from mission to mission. More routine missions may be essentially completed early in the third session or late in the second, whereas a more demanding or unique mission may extend beyond the typical three sessions of 3 h each. Likewise, individuals' requirements and pacing vary significantly in this iterative process. Subsystems vary in the degree to which they are coupled with others, vary in the scope and magnitude of their analytical requirements, and vary in their sys-

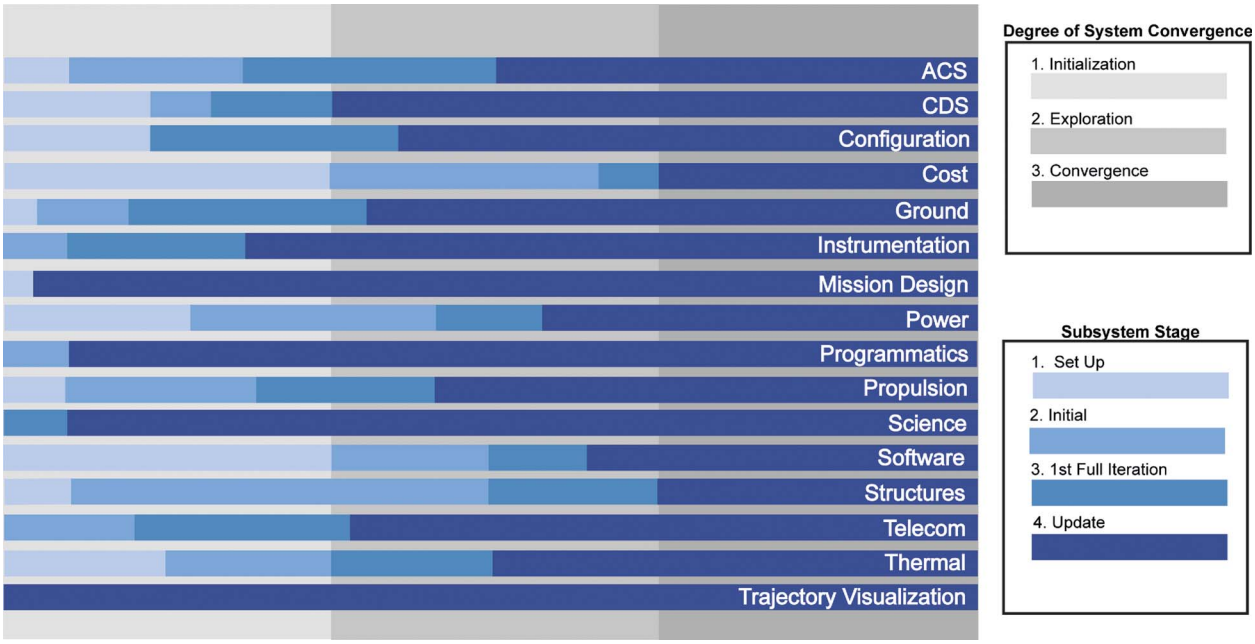

Fig. 1 Relative rates of progression of individual agents described in team interviews

tem roles, some being primary system drivers and others system recipients. The result is that individual subsystems converge at substantially different rates; some completing their initial subsystem definition early in the first session while others would not complete it until well into the second. Figure 1 shows the relative rates of convergence observed for each Team X participant that we obtained through interviews of the team members (normalized to the system rate of convergence that has been subdivided into three sections characterizing the types of activities taking place).

The process and environment are very information rich and collaborative. By concurrently and collaboratively developing the space mission plans, the collocated team is able to resolve trade issues in real-time and come to early agreements on design features. The process is further facilitated by dedicated facilities, tools, and consistent familiar procedures. Aside from an individual's own information, there are multiple sources of information that each individual monitors throughout the process. These include public displays (typically publishing system sheets with aggregate measures of the ship such as mass and power requirements), periodic subsystem updates via the internal network, background conversations/noise, and group discussions. To facilitate the most common communication interfaces, the room is organized to seat together individuals who interact the most frequently; however, it is not uncommon for discussion groups from throughout the room to take place. The environment is often described as managed chaos, and individuals selectively attend to the ambient information influx.

The types and extent of interaction among these subgroup discussions vary. We observed three general types of design related interaction: information updating, collaborative interchange of preferences, and iterative interchange of preferences. Information updating is the simple request and transfer of information related to the current and projected state of individuals' designs. The collaborative interchange of preferences is the open discussion of design options of mutual interest and impact, typically a qualitative discussion of the relative merits and trade-offs of potential solutions. Finally, the iterative exchange of preferences is where collaborating members will quantitatively analyze and re-analyze potential solution options until they come to an agreement on the best mutual solution.

3 Computational Implementation

3.1 Simulation Overview. The algorithm presented here is a computational simulation of JPL's Team X and the conceptual design of interplanetary space mission plans. The implementation, as are all models, is an abstraction of the actual team design environment. As such, there are many possible approaches in modeling the behaviors targeted. In our implementation, the team environment is achieved in a multiagent platform through the distribution and interaction of social and task niches. Each agent, 17 in all representing each subsystem engineer and the systems engineer, is patterned with a representative set of domain models and collaborative schemata as found in their corresponding Team X analog.

The algorithm is composed of descriptive models of both the team—types and timing of interactions, collaborative methods, sequencing, rates of convergence- and the task-primary variables, their behaviors and relations, and the approaches used to resolve them. The objective of these models is to create an environment of interaction representative of that observed in Team X. They were designed to create an environment in which the rich interplay between the task and team structures could emerge naturally.

Altogether, the domain models include over 1000 distinct variables that together describe a space mission. As in the actual domain the problem definitions contain a breadth of problem behaviors and requirements and correspondingly require a breadth of approaches to resolve them. Problem-solving strategies implemented in the algorithm range from the search of databases to analogical comparisons against heritage missions to iterative solver loops and multivariate optimization routines. The objective in putting together these models was to create rich, detailed task models representative of the primary set of behaviors and interactions characteristic of the actual domain space.

Similarly, team interactions analogous to those observed in Team X were created. The team models provide the system context for the domain models, determining how they interact to form and converge on a system solution. Each agent maintains its own methods and variable sets. They also maintain individualized schemata for determining the timing and choice of activities as

Table 2 Case study methods review

Method	Study subject	Data type
Session observations	Four separate design cycles, with project scopes ranging from the evaluation of in mission benefits of an experimental propulsion system to the plan design of a deep space science mission	Observed the team's procedures, their environment, and individuals' activities and documented interaction types and behaviors, the social structure as a function of time, the convergence characteristics of individuals and of the team, and key team issues
Team interviews	Representatives from each participating subsystem chair, systems engineer, team leader, and several customer representatives	Discussions targeted individuals' domain and social representations, including individuals' key considerations—primary parameters, their interactions and behaviors, as well as when and how they are approached—and with whom the individual communicates as well as when and why
Technical domain review	Reference [22], public mission documents	Supplemental technical review of relevant domain representations and analytical tools, and the mapping of the issues, models, and tools from those sources to the described and observed team behaviors (required because of the proprietary nature of actual Team X models)

well as collaborative preferences. As they interact rich ecologies of relationships among and between agents' actions and task related decisions result.

These model definitions were arrived at through an extensive study of the procedures, domain models, and individual schemata represented in the team. This case study consisted of a combination of information gathering methods, including: observational analyses, participant interviews, and an in-depth technical review. See Table 2 for a review of these methods and a summary of the data obtained.

All of the simulation features presented in this paper were designed with the intent of creating behaviors and interactions analogous to those observed in Team X sessions. There are five ways in which the simulation accomplishes this. They are through

1. distribution of social and task constituents
2. procedural analogies
3. information concurrency
4. structuring of knowledge flow
5. definition of domain behaviors

The remainder of this section highlights the main features and functioning of the algorithm.[1] This includes

1. the overall designed process including the iterative structure, concurrency simulation, definition of structural relations through network inputs, and agent schemata types
2. aspects of the task definitions including mission scope, variable types, varying the structures of influence, and the approaches used to resolve variables
3. and, finally, aspects of the team interactions including information updates, sequencing and timing of activities, agent collaborations, and methods for team facilitating

3.2 The Design Process

3.2.1 System Setup. Prior to beginning the actual design process, agents' memory matrices—used in the algorithm to dictate heuristic preferences and estimations—are initiated in a high-iteration, explorative design loop. In this stage the agent team is made to search the problem space through a series of randomly generated missions, the intent of which is to prepare agents with a learned generalization of the domain's characteristics and behaviors. At the completion of each mission, agents individually update their preferences based on the state of the resulting design (for more information on the setup stage or agent's memory matrices see Ref. [1]).

3.2.1.1 Mission definition and design initiation. To begin the simulation, a mission file is read in. This file defines the mission parameters: planetary target; targeted mass, power, and cost caps; suggested payload, etc. This represents the types of direction and scope provided by customers prior to beginning actual Team X design sessions. Once this file is loaded into the system, the design simulation begins.

3.2.1.2 Main design loop. The overarching structure of the simulation is iterative, the core element of which is a short design loop that calls each agent in turn. The structure of the core design loop itself is straightforward. Each loop typically contains two agent calls: a systems agent call and a subsystem agent call. The systems agent is called each design loop to update and publish the information that, in an actual Team X session, is typically displayed to the team throughout the sessions. Following the systems agent update, a subsystem agent is selected and that agent then chooses and executes an action based on its progress in completing and converging on its subsystem design.

Each agent maintains a time stamp designating its time progress through the sessions. Following the completion of a task, agents augment their time stamp a designated amount of time. The subsystem agent call in each design loop is made to the agent with the lowest time.

The loop then restarts and this continues until the system converges or the session times run out. In this way agents are kept at the same approximate point in the session, thereby simulating concurrency.

3.2.1.3 Problem resolution loop and collaboration loops. There are a few exceptions to the design loop's basic flow. For example, if the systems agent (in its turn) determines that the team is not satisfactorily meeting the customer's requirements (desired: launch vehicle, cost, risk, power requirements, ship mass, and/or science objectives) it may initiate a problem resolution loop. As with the primary design stages discussed above, each agent maintains a collection of activities specific to this system initiated stage as well. When the problem resolution loop is initiated, then the system loops through each agent, each of which in turn completes all of its activities associated with that stage. This loop is intended to model team discussions initiated by the systems engineer or team leader.

Another exception, or more accurately an extension, occurs when an agent's chosen activity is a collaborative one. When this is the case, the agent calls those agents indicated as having an

[1]The algorithm was implemented in JAVA in just over 100,000 lines of code. There are 1120 domain methods and nearly 1000 domain variables (on the order of 10,000 when unpacking the arrays). It takes 1–2 min on a 2.80 GHz Intel Celeron to run a complete design proposal.

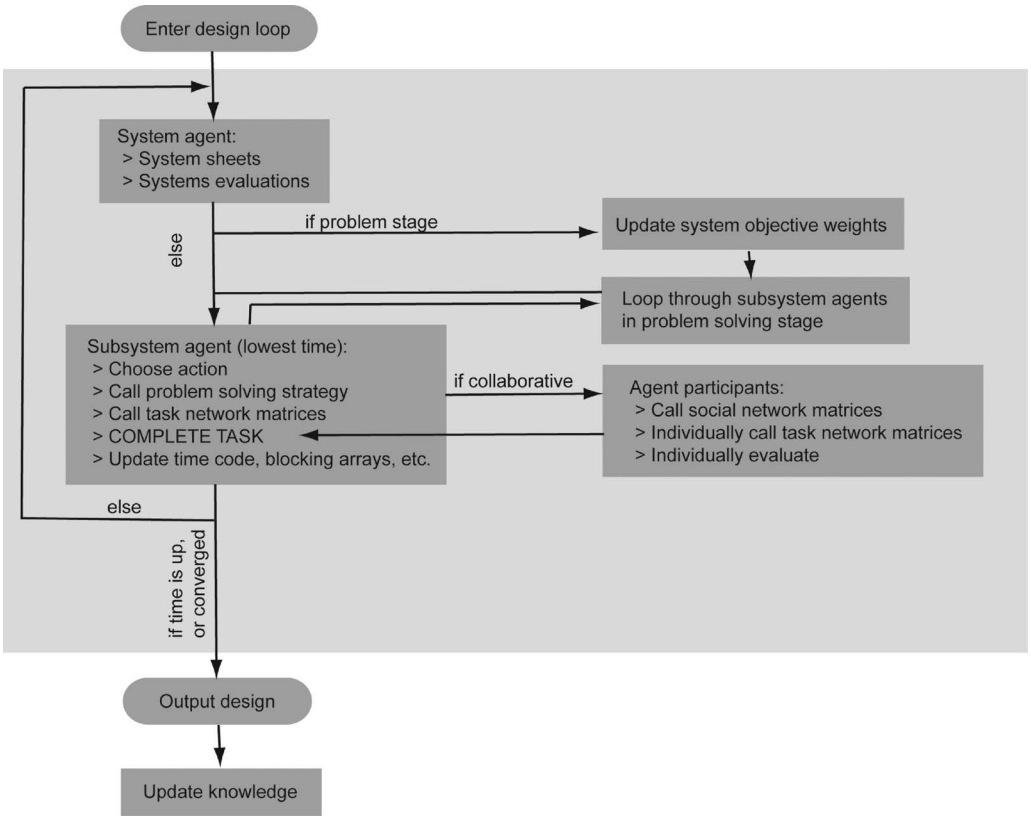

Fig. 2 Schematic overview of the algorithm's design loop

interest in the parameters being defined, and the group of them collectively determines the parameters' values (discussed in Sec. 3.5). Figure 2 shows a flowchart outlining the design loop.

3.2.1.4 Stages of progression. Agent schemata are defined such that agents progress through their individual collections of activities. This is done, for each agent, in stages—each with a unique collection of activities associated with it. In the early stages of a design proposal, agents will typically start with a set up stage, in which agents will set up databases and parameters. Agents then move to the next stage in which important subsystem parameters are initialized with preliminary estimates. In the following stage agents complete their first full iteration defining all subsystem parameters. Finally, in the last stage agents loop through the requisite activities updating their subsystem definitions as needed. Through this process, agents progress from the definition of customer requirements and rough projections of system requirements at the start and gradually converge[2] to an internally consistent solution.

3.2.1.5 Individuals' rate of progression. Although at approximately the same point in session, individual agents, as in actual Team X sessions, progress at different rates. Figure 1 shows the relative rates of progression for each subsystem participant observed in our case study. To capture these rates of progression and to temporally align subsystems' actions within the simulation, each action at each design stage was measured against our case study models and assigned a duration commensurate with its duration in Team X sessions. This keeps the time apportioned to parameter assignment for each agent roughly proportional to that of Team X. As these agents follow patterns of activity by design similar to that of Team X, then they progress through the process maintaining corresponding action to point-in-session relationships.

Agent concurrency and task apportionment are important aspects of this algorithm. They set up the structure to moderate fine scale action concurrency and to control the state of development of information across the system at any point in time. They are important, therefore, in ensuring relational ecologies as a function of time correlative to those found in Team X.

3.3 Network Formulation. An important feature of the algorithm is a set of network models used essentially as agent representations of interdependencies. These networks are used to determine with whom and on what agents collaborate as well as which variables to include in their various analyses, thus controlling social and task relational structures. This network-based platform enables the control of the team's structural properties and provides the basis from which to experimentally modify and analyze the interaction between task and team structures.

In this study, these relational networks are formulated as matrices, the subjects of the rows and columns are the nodes or parts that make up the network in question, and the cells of the matrix correspond to the relationships between those nodes. This formulation is consistent with commonly used techniques for the analysis of social networks (e.g., Refs. [23,24]) and the relationships of product architectures (e.g., Ref. [17]). Matrices for both social and problem relationships found in Team X were created, done by mapping out the relationships between all of the system's variables, tasks, and agents.

At the beginning of the program, the algorithm reads in these matrices and parses them into submatrices befitting each individual agent. Matrices created for input include a *system variable-to-variable* matrix, an *agent task-to-system variable* matrix, and

[2]It may be worthwhile noting here that the process is not guaranteed to converge. In the more extreme cases when a system driver, such as ship mass or cost, grows too large the process will diverge. To avoid this, these drivers are capped at a prespecified maximum value.

Mission Class:	Interplanetary orbiter
Target:	Saturn's moon, Enceladus, and neighboring rings
Science Objective:	Determine the geological history of the moon

Fig. 3 Algorithm base-case mission scope

an *agent-to-agent task* matrix. These, then, define the agents' networks of interdependence. When an agent chooses a collaborative problem-solving method, it refers to the agent-task matrix to identify which other agents are linked to that task and calls them to participate in resolving it. Likewise, each time an agent calls a domain method, it refers to the variable-variable matrix to identify which of the required variables are assigned to influence the method's return value. If the matrix indicates a relationship, the variable value is used. If not, a constant average value is used instead (so that the current value of the variable has no effect on the outcome of the output variable).

In the actual Team X design, individuals interact based on their perception of interdependence. If they believe that parameters from another subsystem will significantly affect the quality of their design or vice versa, they will initiate a discussion with that person. Likewise, analytical and heuristic methods for determining the values of their parameters themselves are defined (and assumptions made to simplify) based on the perception of variable interdependence. The network formulation is intended to simulate these perceptual webs.

The benefit of this methodology is that it allows us to easily extend the base simulation model to alternate network configurations—be they patterns for agent interaction or domain representations. Thus, an experimental set consisting of varying network configurations may be easily controlled and executed, enabling the investigation of the effects of and interactions between task and organizational structural properties on the design process. Once a matrix has been transcribed a range of mathematical operations can then be applied to it to determine a variety of local and global relational attributes. Examining the network properties of a system in this way can be a very informative approach in understanding its behavior. This network formation is a key aspect of the simulation's utility in studying the impact that team and domain network configurations have on the design process and in identifying ways to quantify and manage those impacts.

3.3.1 Agent Schemata. There are four areas, in determining what to do and how to do it, within which agents employ individualized schemata. These are as follows:

1. Choosing what to solve based on system stage, agent stage, previously accomplished tasks, and a learned order preference
2. Choosing whom to collaborate with based on internal representation of agent interdependencies in the form of social and task network matrices
3. Identifying what parameters to include in the analysis based on internal representation of agent interdependencies in the form of social and task network matrices
4. Deciding when to update others based on individual timing schemata identified in interviews

3.4 Task Representation. The domain models represented herein are defined around a single overarching mission objective. In this simulation, all models are defined within the scope of a deep space interplanetary orbiter. Furthermore, the primary objective of the orbiter, as defined in the simulation, is to assess the geological history of a designated planetary target, and so only payload options contributing to this objective are included. The base mission definition used in these tests is an orbiter mission to Saturn's moon Enceladus to determine its geological history (see Fig. 3).

This mission definition was used to establish the scope of the simulation's design process and to determine the types of analyses that required. Modeling the process to a specific class of mission (deep space orbiter, earth orbiter, etc.) corresponds to actual Team X procedure and in many ways is requisite (in both simulation and actual Team X design sessions) given the complexity and scope possible. In Team X design sessions, mission class significantly influences analyses, expectations, and team dynamics. In fact, Team X participants maintain different sets of parameters, tools, and databases corresponding to various mission classes, and load those utilities prior to beginning a design session.

Despite the simulation's definitional focus on a deep space geological orbiter, the simulation retains much flexibility in defining the task characteristics. For example, such large influencers as planetary target, trajectory plan, what payload sensors to include (including the use of a separable probe), and others are left variable. Some of these are defined up front in the process as part of the mission file, and others are determined in the course of the design process. Additionally, via the network formulation discussed above, the simulation allows us to define which domain parameters to include or abstract out of the design process (and by which agent) as well as the relationships between each parameter. So, there are a good deal of potential missions we can produce and numerous representations—from simple to complex—of each of those.

3.4.1 Definition. Within the scope, then, of an interplanetary geological orbiter, the main parameters and methods for designing each subsystem have been identified and representative models created. In this simulation, each agent maintains its own individual domain knowledge, designed and distributed in a manner synonymous to that observed in Team X. The main direction for defining subsystem models was established from the data collected from team member interviews. These data were used to identify the main system and subsystem parameters, trades, objectives, and the methods used to evaluate them, and attention was given to the approach taken by team members in defining each of these parameters. These data, though, were primarily descriptive and so, as the actual models were not available, the mathematical definitions and databases used in subsystems' implementations were taken from texts on space mission plans. The primary sources of data were Wertz and Larson [22] and public documentation of past space missions.

To better reflect the detail and complexity of team members' responsibilities, these models were supplemented in two ways. First, several additional subparameter definitions were included to capture important input variables of those main parameters identified in the interviews. Second, many of the important relations and considerations from Wertz and Larson [22], as from the interviews, were descriptive in nature and, so, to include these interdependencies many of the model equations were parametrically enriched to reflect the relation described.

3.4.2 Variable Types. There are two types of variable used in this algorithm. One type, which we will classify as *analytic* variables, has specific domain functions defining variables' behavior. These variables are deterministic; so given the state of their inputs, their behavior is fixed. Approximately 85% of the algorithm's variables fall within this category. The second variable type, which we will classify as *decision* variables, is forward influencing variables. While the state of related variables influences the quality of their chosen values, decision variables' definitions allow them to be set anywhere in an entire range of values without being mathematically inconsistent. It is to these variables that heuristic methods from individual learning schemas and collaborative decision making are applied. Although fewer in number, because of the flexibility in setting their values in contrast to the deterministic nature of the analytic variables, decision variables may be viewed as the primary system drivers, and they accordingly regulate the overall system state.

The algorithm was designed with the two types of variables to add an extra dimension of correlation to the actual work of Team X designers and to enable the instantiation of collaborative methods. In regard to correlating the algorithm to Team X, the two variable types are meant to be analogous to the same general classes of decisions engineers make in their design, decisions based on methodological analysis and decisions based largely on experiential or intuitive understanding of the system. For example, this may, perhaps, be most readily observed as engineers move from individual detailed analysis of their subsystem to a group discussion of trades on a shared parameter. In regard to algorithm collaboration, decision variables provide a medium in which agents may negotiate these shared parameters based on their influence on the agents' respective subsystems. They, then, may find a balanced value between their subsystems' needs while conserving mathematic consistency.

3.4.3 Variable Inputs. To enable some experimental control of task relationships, the domain models used in this algorithm are linked to the variable-variable matrices input at the start of the program. Where the matrix indicates a relationship, the designated variable is "perceived" to influence the parameter being defined. Where the matrix indicates that there is no relationship, the designated variable does not influence the parameter.

In defining the set of possible relationships and use of related variables, because of the difference in definition, the approach differs between analytic and decision variables. For analytic variables, the task models discussed above, designed to represent Team X models, are formulated as the base models. They are taken to be the most connected. So, the maximum number of inputs for each parameter is defined in those models, and the network with all the possible relationships between them is representative of Team X's models. For this variable type, when the matrix indicates a relationship from among the set of possible inputs, that variable value is passed into the method and used as normal. When the matrix indicates that one of those possible inputs is not related, in contrast, a constant average value is passed in (the median value in the range of possible values). This is done so that the current variable value has no effect in influencing the parameter one way or another, and the parameter value is kept consistent and reasonable. For decision variables, because of their flexible definitions, any parameter is capable of being an input. For these variables, all related variables are used in setting the learning arrays that are used to determine the variable's setting. For more detail on variable types and agent learning, see Ref. [1].

3.4.4 Problem-Solving Strategies. As in most complex design environments, the problem sets addressed in Team X sessions require many different analytical approaches. Each approach evaluates the influence of related variables in different ways. In keeping with our objective to reflect the approaches and parameter behaviors described in our case study of Team X, the algorithm also includes several different analytical approaches (in addition to the individual mathematical definitions used to describe parameters' trajectories of behaviors). There are database search modules that identify solutions from a set of discrete possibilities given constraints and objectives. There are modules that analyze a set of heritage mission characteristics, determine the degree of similarity, and set variables from the comparison. There are simple optimization routines (quadratic interpolation) and heuristic solvers for nonlinear multivariate (and multiagent) "optimization." There are two iterative loop solvers: one that loops until it converges to a solution that is mathematically consistent and another that loops through modification methods until a set of constraints is satisfied. Finally, there is a multivariate negotiation module that iterates between agents until they converge. The heuristic solver and negotiation module are discussed more in Sec. 3.5. In all cases, models were designed to reflect the approaches described by team members.

3.5 Team Representation. The team environment, in this algorithm, is achieved through the distribution of social and task niches among computational agents, each of which represents an analog Team X participant. Each agent maintains its own methods and variable sets. They also maintain individualized schemata for determining the timing and choice of activities as well as collaborative preferences. As they interact, then, a rich ecology of relationships among and between agents' actions and task related decisions results. These patterns of interaction arise from the combination of individual's schemata and the interactions among the tasks they choose, and they in turn lead to a more complex system behavior characterizing the overall search and convergence properties of the system. In this section several key aspects of the team interactions implemented in this simulation are discussed, including information updating, sequencing of activities, collaborative methods, and methods of team facilitation.

3.5.1 Interaction Overview. As in actual Team X sessions, the agents in this algorithm communicate subsystem information in essentially two ways. The first is a simple update, a noncollaborative transmittal of some portion of the subsystem's current design state. In Team X sessions this occurs primarily via the internal network, sending and receiving preset inputs throughout the sessions. It also occurs, though, as information is projected onto the front screens or announced to the team. The second is through collaborative dialog, mutual interchange of preferences and progression toward a commonly agreed upon state. Sidebar discussions may be observed frequently throughout Team X sessions as individuals work on subsystem trades of mutual interest.

In regard to agent updates, each agent maintains its own separate variable list containing all of the parameters used and/or set by that agent. Although separate, agents' lists may contain overlapping variables. That is, the same variable may be contained in more than one agent list. However, while several different agents may use a variable, only one has primary stewardship for setting that variable's value. The values of that variable in the other agents' lists are received from the one agent responsible for setting it. Their values for that variable are renewed each time the one agent responsible for it updates their list, and it remains at that state until it is updated again.

The decision of which agents to update is derived from Sec. 3.3. From the agent-task and task-variable matrices that are input into the system, an agent-variable network may be derived. This matrix is then used by the individual agents to determine who requires updates on each of "their" variables.

The decision of when to update agents' lists is based on individualized schemata designed to approximate the timing and intent of the corresponding Team X engineer. This information was identified in team member interviews. These schemata may be to update others at the completion of certain activities at different stages in the design process or at rough intervals of time given the design stage.

In regard to agent collaborations, when an agent chooses an activity that requires collaboration it calls all other agents associated with that activity. This determination is set in the agent-task matrix, and those relationships were identified in team member interviews as members identified with whom they worked with and on what. In the process of collaborating with each other, all the variables used by each participant in determining the targeted variable(s) are updated in each of the others' lists.

For both updates and collaborations, the webs of time dependent relationships are the result of the combination of individual interests and schemata, as described in team member interviews. So, the system-level patterns are a result of these finer scale concerns. It is in this way, by defining individuals and placing them in a specific environment of interaction, that the patterns of information flow and concurrency, team interaction, and system convergence are simulated.

3.5.2 Task Sequencing. Task layouts were created for each agent to facilitate the decision of which task to work on and when. Both the sequencing of tasks and the time allotted for task completion are assigned in these layouts for each agent and each subsystem stage—setup, initialization, first full iteration, update, and problem stages. To define task precedence, tasks were laid out in blocks and sub-blocks denoting the temporal sequencing of the tasks. This blocking structure, then, defines which actions are to precede which others, and which may be executed interchangeably with which others at a given point. The determination of this structure was made from the logical sequences required by the activities. Each activity, then, in each subsystem stage was assigned a time required to complete it. This determination was made in consideration of both the activity difficulty and the timing constraints imposed by the observed subsystem rates of progression (refer to Fig. 1). Between the two, task and time blocking, the ordering and timing of agent activities are made to align roughly with those observed in our case study.

A simple learning scheme is applied to agent blocking as well to aid in the choice of activities that may be performed interchangeably at a point in time. From analyses of final designs discovered during the system's exploration stage, agents draw an association between the order of activities taken and the subsystem's final utility value and number of iterations required to converge on the final design solution. From these analyses, a preference weight is assigned to activities in the block. When an agent reaches a point where it must choose between two or more activities that may be performed interchangeably at that point, it uses this heuristic to decide the ordering of its activity calls.

3.5.3 Collaborative Strategies. There are, of course, many intricacies in the ways individuals collaborate to resolve problems. In the algorithm, there are two methods used to simulate the types of collaborations observed in our case study of Team X: 1. an iterative negotiation model and 2. a direct negotiation model. The direct method is intended to approximate discussions in which participants openly present their preferences, weigh alternatives, and, from that information, come to an agreement directly. The iterative method is intended to approximate discussions in which participants offer solutions and countersolutions until they converge to a shared solution. Both methods are extensible to multiple variable analyses (from 1 to 16 variables in our implementation) and permit the participation of multiple agents.

3.5.4 Method 1: Direct Method. Our approach to collaborative agreement is to recognize that many of the collaborative decisions made in the early or conceptual phases of design are based largely on experiential or intuitive understanding of the system rather than methodological analyses. Individuals discuss their preferences, the relative impacts on their portion of the problem, and trade-offs qualitatively to get a "feel" for the shared effects of design decisions. This greatly simplifies required analyses and enables team members to come to a rapid agreement on shared parameters to accelerate convergence. This was a common form of interaction observed in Team X sessions.

To accomplish this type of collaborative interaction, participating agents review potential solution options and heuristically assign a utility value to them. These heuristic evaluations are based on the agent learning schemes (see Ref. [1] for more detail). Individual utility values are then aggregated, and the solution option with the best overall utility is used to set the variables values.

The procedure for this collaborative method begins when an agent calls this method, identifying the variable(s) to be determined. Based on the subsystem's agent-task matrix, participating agents are called, and each of the participating agents then individually evaluates the solution option, assigning it utility values for each of the decision variables being evaluated. Once each agent has assigned utility values for each decision variable under consideration, they are aggregated to form a total utility value. If this loop occurs during the system's exploration stage an additional term is added to the aggregate utility value to encourage divergence. That is, the aggregated utility value is weighted to encourage the selection of variable ranges not previously explored. In the end, the result is an aggregate utility value used to judge the overall fitness of the variables' settings in view of all participating agents. This process is continued to evaluate the overall utility for each variable-range configuration. If the resulting utility value is the best found to this point in the process, the decision variables are assigned a value within their designated ranges. An overview of the solver method is shown in Fig. 4. For more detail see Ref. [1].

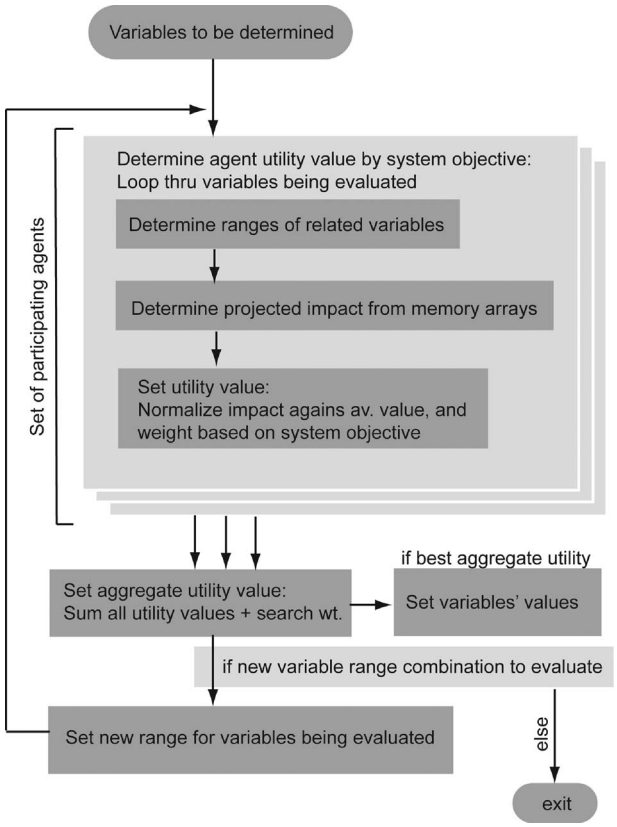

Fig. 4 Schematic overview of the direct heuristic solver method

3.5.5 Method 2: Multivariate Iterative Negotiation. In contrast to the heuristic evaluations of the direct agreement method discussed above, the iterative negotiation method is intended to approximate collaborations in which subsystem trade-offs are based on methodological analyses. Rather than qualitatively discussing the pros and cons of solution options, individuals analyze the problem and present solution options. Following which the other agents will do likewise, making some concessions based on others' requirements, and will return with an alternate solution option. This process continues until an agreement is reached. This method of interaction was also observed in Team X sessions.

In this method, the evaluation methods used by participating agents may be different for each agent. They may be individualized optimization routines, rule sets, or even the heuristic method described above. In this approach, participating agents individually analyze and optimize countersolutions that make concessions designed to step individual solutions toward convergence.

Following method initialization, the process begins with each participant making an initial assessment—that is, determining an "optimal" value or combination of values—of the chosen variable(s). Agents are then called individually to evaluate and propose revised variable values. In doing so, agents make conces-

sions in their choices of values to increase the overall "closeness" of values between participating agents. The closeness of values is based on the percentage difference between agents' values given by

$$\Delta\% = 1 - \frac{\sum_{i=1}^{n} \frac{V_{i,\min}}{V_{i,\max}}}{n} \tag{1}$$

where $\Delta\%$ is the percentage difference, $V_{i,\min}$ is the minimum proposed value for variable i, $V_{i,\max}$ is the maximum proposed value for variable i, and n is the total number of variables.

Concession requirements, then, are given by a step size (defined on a simple linearly decreasing schedule) that dictates the change in $\Delta\%$. This gives agents flexibility in setting the values of each variable. They may, for example, set one value to converge with other agents' and the other values to diverge, or they may set all values to converge with other agents' as long as the resultant value for $\Delta\%$ decreases by the set amount. In this algorithm, the optimization routines used in conjunction with the iterative negotiation method are programed with a preference for adjusting values toward those of other agents. Nevertheless, because of the interdependence between variables, adjusting some values toward other agents' will require setting others further away. These interrelationships may be intricate and complex, and the balance of values between variables affects the utility of each agent differently. This approach to concessions allows agents flexibility in pursuing the balance of variable values that best optimizes their individual utilitie, and ultimately allows for a balance between agents' preferences to emerge.

Agents continue to individually revise their proposed variable values until the set of values falls within the specified range of proximity. In our implementation, when $\Delta\% < 0.1$ the loop closes and the values for all participants are set to that of the final agent. An overview of the iterative solver is given in Fig. 5.

3.5.6 Team Facilitating. There are three ways in which the systems engineer directs team progression in this algorithm. The systems engineer sets the system stage, regulates the system objective weights, and initiates the problem loop. These three systems roles are meant to approximate the primary ways that the team leader and systems engineer were observed facilitating team processes and directing individuals' foci.

The first method is through the designation of the system stage. Just as each subsystem agent keeps track of its individual stage in the design process, the systems engineer keeps track of the team's point in the process. The system stage is used by individual agents in their determinations of what actions to choose as well as in the systems agent's determination of whether or not to initiate the problem loop.

The second is through the designation of objective weights. The system objective weights are used to focus agents' attentions on achieving specific system objectives, such as driving down the mass or cost of the ship. To accomplish this, they are used to weight the utility values derived for each subsystem objective in agents' heuristic evaluations of decision variables. This gives preference to the optimization of particular objectives in these evaluations. At the start of the design process, the default settings for system preferences are for maximizing subsystem objective utilities and minimizing cost. As the team progresses and the systems engineer identifies problem areas, areas in which the design is deficient and/or persists in being deficient in meeting the customer's requirements, the weights are adjusted to give increased preference to those areas. The systems engineer can, in this way, direct the team toward better meeting the customer's mission requirements.

Finally, the third method is through initiating the team problem loop; the purpose of which is to get each agent to work in unison in driving the system solution to better meet the customer's requirements. The purpose of the loop is to simulate team wide discussions that are initiated when the solution needs to be redirected or a focused team effort is needed to drive the solution in a certain direction. Periodically throughout the design process the systems engineer evaluates the state of the mission solution vis-à-vis the customer's requirements. It does this by checking the system status on each of the system objectives against predefined value limits. These objective value limits specify how much the system is allowed to exceed the customer's requirements before initiating the loop and are initially wide in the design's earlier stages and gradually tighten as the end of the session nears and/or the team nears convergence. If one or more of the objectives exceed the limit, the systems engineer updates the system objective weights and initiates the problem loop. Once the loop is initiated, each agent's stage is set to the problem stage and the system loops through each agent, which in turn completes each of their problem stage activities. In this way, the entire agent team makes a concerted effort to reposition the solution to better meet the customer's mission requirements.

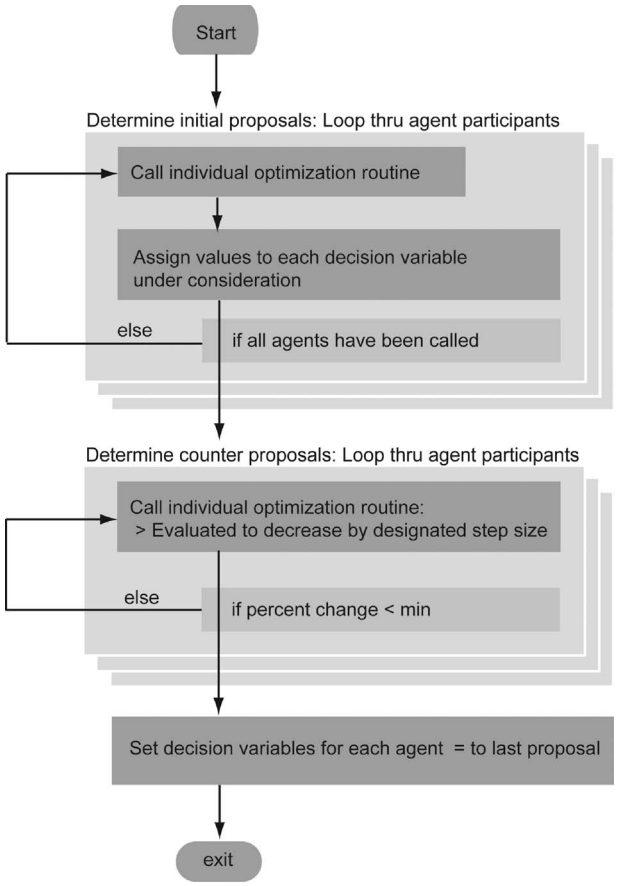

Fig. 5 Schematic overview of the iterative negotiation method

4 Model Verification

The objective in creating these computational models is to create an environment that simulates the trajectory of behaviors and interactions representative of those that occur in actual Team X design sessions. Extensive efforts were made to design and verify the fine scale behaviors—those of individual parameters and agents—such that they correspond to the behaviors described and observed in our case study. It is from the interaction of these finer scale behaviors, then, that the system-level patterns of behavior result. Therefore, to confirm that the computational simulation exhibits comparable ecologies of relationships, an examination of the system-level trajectories of behavior was conducted. The ques-

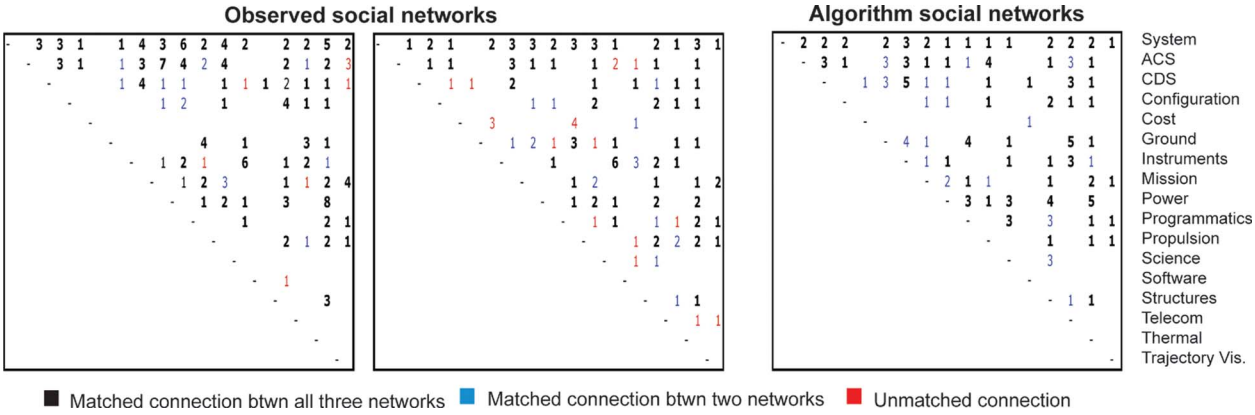

Fig. 6 The observed and algorithm social network ties for all sessions in each proposal. Note that the data in these matrices are weighted (the weight indicates the amount of time individuals interacted, 1=10 min) and are nondirectional (this is because the exchange of information in these interactions was dominantly collaborative (two way)).

tion being asked is do the fine scale interactions indeed lead to systemic patterns that are representative of those observed in Team X? To do this, an examination of the two foci of this study—the social and task domains—was performed and is discussed.

To make this comparison, algorithm output from 100 design runs was collected and averaged. Mission input parameters for each of these runs were randomly generated within the scope of the defined mission. These data were then contrasted against observed data and data collected about actual space missions.

The verification presented in this paper is an initial examination to ensure that the fine scale definitions lead to the system behaviors desired. For social behaviors, the network patterns of interactions from agents' combined collaborations are examined. For task behaviors, the primary system drivers (as identified in Team X member interviews) and subsystem mass distribution are examined (chosen for the availability of this information). Further verification assessing the algorithm's predictive accuracy will follow in succeeding studies.

4.1 Social Behaviors. To verify that the resultant environment of social interaction from our computational models is comparable to that of Team X, a comparative investigation of the relation of Team X's social networks to those of our simulation was performed. The point of this examination of social structure is simply to verify that the simulation demonstrates similar structures of interaction. That is, do the combinations and timing of agent activities lead to team social networks similar to what may be observed in actual sessions?

Observations of two separate and significantly different—in both difficulty and subject—Team X proposals were used for this comparison. The design related interactions occurring throughout these observed sessions were documented, from which social network matrices were created describing the team interactions at various points throughout the process. The matrices were then analyzed to identify recurring social interactions. For the algorithm, equivalent information on agent interaction was outputted and averaged for 100 design runs.

Figure 6 shows the overall social networks—the total interactions occurring during all three sessions—for the two observed proposals and the algorithm's output. The three cases—the two observed cases and the algorithm's average output—were compared. All of the algorithm's resultant social ties corresponded to at least one of the two observed social networks and of those ties found to be consistent between the two observed cases 95% were also found in the algorithm's network, meaning that the same subsystem agents in the algorithm were interacting as those that were observed to interact in the actual design environment. Furthermore, these matched ties consisted of 75% of the network's total ties where matched ties between the observed sessions consisted of 73–76% of the ties in those networks. The overall network of agent interaction, therefore, corresponded quite closely to that of the observed cases. In addition to the existence of relationships, the strength of the ties also seemed, in the main, representative of those observed.

Agent interactions during three intermediate stages of the design—corresponding to the stages of system convergence designated in Fig. 1—were likewise compared against the observed cases. The distribution among the three groups for each design stage was found to be similarly correlated (for more information see Ref. [1]).

4.2 Task Behaviors. The algorithm's domain outputs were also investigated to verify that the networks of task interactions from our computational models are representative of those of actual interplanetary space mission plans. The question being asked is do system parameters resulting from the algorithm's designs exhibit similar magnitudes to those of real space missions and are they similarly apportioned relative to each other? The individual fine scale subsystem parameters were, as described in Secs. 1–3, carefully designed to simulate the behaviors and to include the interactions identified in our case study of Team X. As a preliminary check, these were examined to verify that, in combination with all the other system variables in a design, they continued to fall within the range of values for which they were intended. Having then verified the individual fine scale behaviors, the next question is do they combine and interact to produce aggregate system-level values that are comparable to those found in real space missions?

The first examination made is of the top system drivers. The system drivers used in this comparative analysis were those identified in team member interviews and represent the top five most frequently identified system drivers. They are mission target, mission life, cost, mass, and power. The algorithm's values were averaged over 100 design runs. Data for these system parameters are given in Fig. 7 for the Cassini mission, the most closely related to our task definitions, and for the algorithm.

The comparison of the results is somewhat qualitative; however, the algorithm's results do fall closely within ranges comparable with those of the heritage missions. Recall that the domain models used in the algorithm were formulated within the scope of a planetary orbiter sent to Saturn's moon, Enceladus, and its neighboring rings. The algorithm's prescribed mission requirements, therefore, fall most closely with those of the Cassini mission, also an interplanetary orbiter dispatched to the moons and rings of Saturn. So, they share the same general target and similar science objectives (the algorithm even included analysis for an additional probe as with Cassini, albeit the science undertaken by

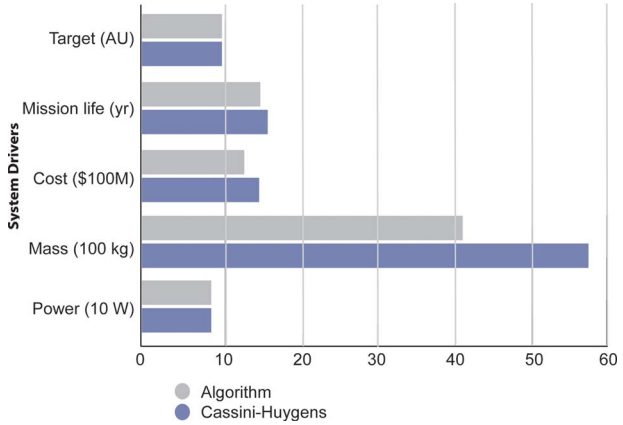

Fig. 7 Magnitudes of the primary system drivers for Cassini and the algorithm

the algorithm's mission definition was rather less extensive than that of Cassini). Correspondingly, the average values for mission lifetime, cost, mass, and power are seen to approach, while remaining somewhat less, those of the Cassini mission. The combination and interaction of individual variables in the algorithm's designs of its space mission led to system values corresponding nicely to those of Cassini, a comparable mission.

The second examination looks at the percent distribution of subsystem masses. The mass distributions for actual space missions were taken from Wertz and Larson [22], in which they analyzed 24 different space missions. These space missions were predominately earth orbiters, and so some deviation for interplanetary missions is to be expected. For example, many of the earth orbiter missions evaluated had a negligible or nonexistent propulsion system. To account for this, both actual and algorithm percentages have been adjusted to not include propulsion masses for this comparison. An increased mass apportioned to payload is another deviation that is to be expected, the range of payload masses for interplanetary missions differing significantly from mission to mission. Nevertheless, despite the expected deviations, the mass distribution between subsystems is representative roughly of space missions generally and the algorithm's results aligned very nicely with those of the actual missions. Figure 8 shows the resulting mass distributions for the algorithm, again averaged over 100 design runs, and the space missions examined

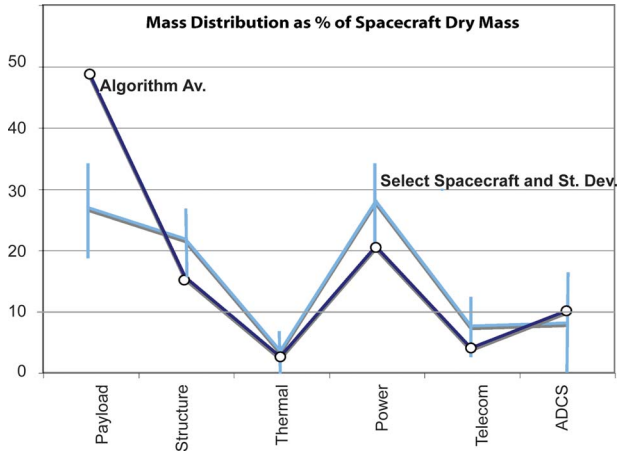

Fig. 8 Percent mass distribution for 24 space missions (light line) taken from Wertz and Larson [22] and the algorithm (dark line)

Table 3 Small sampling of the simulation's design output

Parameter	Value
Launch vehicle	AR 44 I
Launch date	Aug. 2024
Bus cost	$348 M
Orbital distance around Enceladus	1.01×10^7 miles
Min tech risk level	3
Trajectory duration: phase 2	6.7 years
Eclipse time	7.14 h
Orbital eccentricity	1.65
Flight software language	FORTRAN
Flight software length	300 KLOC
Computer	SWRI SC-2A
Data volume	1.2 Gbits
Throughput	71.4 Mips
IR ground resolution	59.4 m
IR aperture	0.40 m
Communication duration	2.68 h
Phase D length	6.2 years
Ground team size	48
Antenna No. 1 gain	42.4 dBD
RTG mass	60 kg
Power BOL	6040.6 W
Propulsion system	GDAIS ISE
Propellant consumed: phase 2	255.4 kg
Solar pressure (av)	76.2 N/m^2
Angular impulse: phase 2	507.6 N m s
Torque dump	0.67

in Ref. [22]. As an indication of design output, a small extract of typical design outputs is given in Table 3.

For more detailed information of simulation methods and verification testing see Ref. [1].

5 Discussion and Conclusions

5.1 Methodology Review. The simulation approach presented in this paper is designed as a platform to investigate the complex and dynamic environment of interaction associated with the design of complex systems. The simulation is designed to emulate the complex and dynamic characteristics of the real world design environment of NASA/JPL's Team X. Benchmarking Team X provides a medium wherein to define and verify realistic and representative design models. Using Team X as a point of reference in model creation provides insight into a range of problem-solving requirements, collaborative drivers, issues in the sequencing and timing of activities, and how people bring them together.

Our approach is to create rich and detailed models of these myriad interdependencies. For the problem domain, this includes identifying the primary parameters, their behaviors and relationships, and then distributing them appropriately throughout the system—both physically (knowledge and stewardship of the parameters) and temporally (coordinating the time function of interaction). For the team, this includes defining the primary types, timing, and distribution of interactions among agents—capturing the knowledge flow, collaborative decision making, and team procedures found in actual design sessions. And then, through the system environment created in the distribution and interaction of these team building blocks, complex behaviors and characteristics representative of the actual environment emerge. The result is a system that more closely approximates the complex characteristics typical of the real design environment.

There are a few distinctive features of this algorithm and approach that make it particularly well suited to the investigation of the interplay between the task and organizational interdependencies frequently found in collaborative design. First are the detailed task models. The primary parameters—their behaviors and relationships—of the actual solution space were thoroughly modeled in this simulation. Few computational organization studies

incorporate detailed task models exhibiting the intricate behaviors and interactions characteristic of real world design. These models enable a richness of interaction not achievable in simpler domain representations. Second is the breadth of collaborative and problem-solving strategies employed by agents in this algorithm. Information is transferred, both directly and indirectly, through variable updates, heuristic collaborations, and iterative negotiations. Parameters are resolved through direct mathematical definitions, database searches, analogical comparisons, iterative solvers, linear optimizations, and multivariate heuristic optimizations. The range of task and agent interaction types lends to the richness of the environment of interaction and is an essential characteristic of complex design. Third and lastly are the flexible network-based definitions of the task and agent structures. These make it possible to easily adjust the base simulation model to alternate task and/or social network configurations, thus enabling the experimental control and investigation of the effects of and interactions between task and organizational structural properties on the design process.

5.2 The Simulation as a Design Tool. Although the primary goal in developing this simulation was the creation of a platform to enable investigation of design interdependencies, it may also be viewed as an exploration and demonstration of a methodological approach to the computational design of complex problem domains. Using the distributed collaborative approach described above, this platform has been shown to be effective at producing realistic space mission plan designs. It successfully synthesizes a rich and diverse set of domain interests, and it successfully brings together the distributed and individualized interests and problem-solving approaches developed for and infused in the current approach.

Effectively extending this approach to practical design use would require some adjustments to the presented algorithm. Two important limitations in the current algorithm design are (1) the efficient generation of rich domain models and (2) the extension of the agent infrastructure to alternate design domains. Regarding the first limitation, in the presented implementation domain models were custom designed as representative approximations of existing models and thus constituted a significant labor burden. Solutions to mitigate this cost may include the development of interfaces to existing domain models and/or tools to quickly build the desired models. Regarding the second limitation, the simulation presented was designed for simulative performance as opposed to computational efficiency and statically coded to suit the targeted domain. A possible redesign may modularize possible agent interaction and capabilities to enable customization and reuse.

5.3 Concluding Remarks. There is an enormous potential for enhanced efficiency and robustness through the structuring of available resources, a potential that is increasingly pronounced as the complexity of the task rises. Better understanding the role of relational properties among and between task and organizational domains should improve our ability to predict and design for favorable emergent and dynamic organization properties, ultimately leading to methodologies that better account for task complexities and improve the search for and evaluation of innovative solutions.

The simulation presented in this paper provides a flexible and robust instrument for investigating properties characteristic of the complex set of interactions inherent in the collaborative design of a complex system. The study presented in this paper is a foundational part of a larger research agenda, the aim of which is to move toward a more precise and rigorous understanding of collaborative design, and its impact on the way we approach design management, teamwork, and computational design.

Acknowledgment

The authors would like to thank Team X at the Jet Propulsion Laboratory for assistance in this work. This research was sponsored by the Air Force Office of Scientific Research, Air Force Material Command, USAF, under Grant No. FA9620-04-1-0201.

References

[1] Olson, J., 2006, "The Collective Potential: Achieving Organizational Potential by Design," Ph.D. thesis, Carnegie Mellon University, Pittsburgh, PA.
[2] Carley, K., and Hill, V., 2001, "Structural Change and Learning Within Organizations," *Dynamics of Organizations: Computational Modeling and Organizational Theories*, A. Lomi, ed., MIT Press, Cambridge/AAAI Press, Live Oak, CA.
[3] Anderson, P., 1999, "Complexity Theory and Organizational Science. (Perspective)," Org. Sci., **10**(3), pp. 216–232.
[4] Carley, K. M., and Gasser, L., 1999, "Computational Organization Theory," *Distributed Artificial Intelligence*, G. Weiss, ed." MIT Press, Cambridge, MA, Chap. 7.
[5] Bar-Yam, Y., 2001, *Introducing Complex Systems*, NECSI Press, Boston, MA.
[6] Levinthal, D. A., and Warglien, M., 1999, "Landscape Design: Designing for Local Action in Complex Worlds," Org. Sci., **10**(3), pp. 342–357.
[7] Carley, K., 2003, "Intra-Organizational Computation and Complexity," http://www.heinz.cmu.edu/wpapers/retrievePDF?id=2000-15, Accessed 2/13/2003, Working Paper, CMU, Pittsburgh, PA.
[8] Krackhardt, D., and Carley, K., 1998, "A PCANS Model of Structure in Organization," Proceedings of the 1998 International Symposium on Command and Control Research and Technology, Monterey, CA, pp. 113–119.
[9] Alexander, C., 1964, *Notes on the Synthesis of Form*, Harvard University Press, Cambridge, MA.
[10] Levchuk, G. M., Levchuk, Y. N., Luo, J., Pattipati, K. R., and Kleinman, D. L., 2002, "Normative Design of Organizations—Part II: Organizational Structure," IEEE Trans. Syst. Man Cybern., Part A. Syst. Humans, **32**(3), pp. 360–375.
[11] Thompson, J. D., 1967, *Organizations in Action*, McGraw-Hill, New York.
[12] Galbraith, J. R., 1973, *Designing Complex Organizations*, Addison-Wesley, Reading, MA.
[13] Adler, P. S., 1995, "Interdepartmental Interdependence and Coordination: The Case of the Design/Manufacturing Interface," Org. Sci., **6**, pp. 147–167.
[14] Smith, R. P., and Eppinger, S. D., 1997, "Identifying Controlling Features Of Engineering Design Iteration," Manage. Sci., **43**, pp. 276–293.
[15] Loch, C., and Terwiesch, C., 1998, "Communication and Uncertainty in Concurrent Engineering," Manage. Sci., **44**(8), pp. 1032–1048.
[16] Henderson, R., and Clark, K., 1990, "Architectural Innovation: The Reconfiguration of Existing Product Technologies and the Failures of Established Firms," Adm. Sci. Q., **35**(1), pp. 9–30.
[17] Sosa, M., Eppinger, S., and Rowles, C., 2004, "The Misalignment of Product Architecture and Organizational Structure in Complex Product Development," Manage. Sci., **50**, pp. 1674–1689.
[18] Kunz, J. C., Levitt, R. E., and Jin, Y., 1998, "The Virtual Design Team: A Computational Simulation Model of Project Organization," Communications of Association for Computing Machinery, **41**(11), pp. 84–91.
[19] Krishnan, V., and Ulrich, K., 2001, "Product Development Decisions: A Review of the Literature," Manage. Sci., **47**(8), pp. 1032–1048.
[20] Barnes, J. A., 1969, "Graph Theory and Social Networks: A Technical Comment on Connectedness and Connectivity," Sociology, **3**(2), pp. 215–232.
[21] Sosa, M. E., Eppinger, S. D., and Rowles, C. M., 2003, "Identifying Modular and Integrative Systems and Their Impact on Design Team Interactions," ASME J. Mech. Des., **125**(2), pp. 240–252.
[22] Wertz, J. R., and Larson, W. J., 2003, *Space Mission Analysis and Design*, 3rd ed., Microcosm, Torrance, CA/Kluwer Academic, Dordrecht.
[23] Panzarasa, P., Carley, K., and Krackhardt, D., 2001, "Modeling Structure and Cognition in Organizations: A Meta-Network Computational Approach," Casos Conference, Carnegie Mellon University, PA.
[24] Wasserman, S., and Faust, K., 1994, *Social Network Analysis: Methods and Applications*, Cambridge University Press, Cambridge.