Addressing Deep Uncertainty in Space System Development through Model-based Adaptive Design

Mark Chodas
NASA Jet Propulsion Laboratory
4800 Oak Grove Dr.
Pasadena, CA 91109
mark.chodas@jpl.nasa.gov

Rebecca Masterson
Department of Aeronautics and Astronautics
Massachusetts Institute of Technology
77 Massachusetts Ave.
Cambridge, MA 02139
becki@mit.edu

Olivier de Weck
Department of Aeronautics and Astronautics
Massachusetts Institute of Technology
77 Massachusetts Ave.
Cambridge, MA 02139
deweck@mit.edu

Abstract—When developing a space system, many properties of the design space are initially unknown and are discovered during the development process. Therefore, the problem exhibits deep uncertainty. Deep uncertainty refers to the condition where the full range of outcomes of a decision is not knowable. A key strategy to mitigate deep uncertainty is to update decisions when new information is learned.

In this paper, the spacecraft development problem is modeled as a dynamic, chance-constrained, stochastic optimization problem. The Model-based Adaptive Design under Uncertainty (MADU) framework is presented, in which conflict-directed search is combined with reuse of information to solve the development problem efficiently in the presence of deep uncertainty. The framework is built within a Model-based Systems Engineering (MBSE) paradigm in which a SysML model contains the design, the design space, and information learned during search. The development problem is composed of a series of optimizations, each different than the previous. Changes between optimizations can be the addition or removal of a design variable, expansion or contraction of the domain of a design variable, addition or removal of constraints, or changes to the objective function. These changes are processed to determine which search decisions can be preserved from the previous optimization.

The framework is illustrated on a case study drawn from the thermal design of the REgolith X-ray Imaging Spectrometer (REXIS) instrument. This case study demonstrates the advantages of the MADU framework with the solution found 30% faster than an algorithm that doesn’t reuse information. With this framework, designers can more efficiently explore the design space and perform updates to a design when new information is learned. Future work includes extending the framework to multiple objective functions and continuous design variables.

REFERENCES .............................................. 15

1. INTRODUCTION

NASA has established processes designed to meet the challenge of developing systems that accomplish difficult tasks in the unforgiving environment of space. These processes have evolved over the years and have produced many successful space missions. Still, some missions struggle with cost and schedule overruns and technical failures [1]. The space system development process is a series of decisions, made over time, with an increasing level of detail and maturity in the design [2] [3]. The process is highly iterative with decisions made at higher levels being used to frame the set of options at lower levels. Decisions made later in the lifecycle benefit from the additional knowledge about the design space that is gained as the system is designed.

Uncertainty is prevalent in space system development [4]. System performance is difficult to measure before launch and so tests and models are used to raise the likelihood of the system performing as intended after launch. These tests and models are imperfect approximations of the actual operational environment and therefore estimates are uncertain. Space systems involve many tightly integrated components and therefore uncertainty arises from the difficulty in accurately quantifying all of the system interactions. Alternatively, system properties may not be accurately known and therefore cannot be modeled precisely. Many techniques exist to develop space systems under these types of uncertainty [5] [6] [7]. However, these approaches make the assumption that all uncertainty can be modeled a priori and this assumption does not always hold as the probability of some events that can occur during the development process cannot be accurately estimated. This type of uncertainty is called deep uncertainty [8].

Many events that may occur during the development process fall into the category of deep uncertainty. For example, a component may break during handling, a new technology may not work as intended, or unexpected interactions between system elements may emerge. These events are difficult or impossible to predict beforehand with confidence.

This paper presents the Model-based Adaptive Design un-
der Uncertainty (MADU) framework for use in developing space systems [9]. This framework adaptively updates the system design to both improve past design decisions when new information is learned and to handle unforeseen events. The framework is enabled by Model-based Systems Engineering (MBSE) where system information is captured in a descriptive system model [10] [11]. MBSE is a key enabler of MADU because it supports descriptive modeling of the system, the design space, and decisions made about the system design. These pieces of information are all needed to re-evaluate decisions when new information is learned. Frequent redesigns can be wasteful and so incremental search techniques based on conflict learning are used to perform the design update efficiently. With this framework, uncertainty can be more completely accounted for during the design process, saving rework, improving system performance, and lowering cost. This paper presents the motivation for such a framework, reviews relevant background topics, presents the methodology used by the framework, and uses a case study to illustrate the value of the framework.

2. MOTIVATION

Space systems commonly experience cost overruns and programmatic delays [1] [12]. The reasons for such extensive programmatic breaches include repeated changes to requirements, reliance on immature technology, and reliance on immature software [13]. These root causes are all manifestations of deep uncertainty. If requirements constantly change, constant rework is needed to update the system design to ensure that it satisfies the latest set of requirements. This difficulty is a form of deep uncertainty because the development team does not have enough knowledge to bound the correct set of requirements. Relying on immature technology or software increases the likelihood that unforeseen issues arise during development, a manifestation of deep uncertainty.

Specific examples of how deep uncertainty can cause cost and schedule overruns can be found in the James Webb Space Telescope (JWST) project. JWST is a next generation space telescope that will greatly expand humanity’s knowledge of the universe [14]. However, it has experienced unprecedented extensive delays during development [15] [16]. JWST has struggled with mismanagement, workmanship issues, and integration and test issues [17] [18] [19]. All three of these issues are instances of deep uncertainty. Mismanagement occurred because the project could not accurately predict the effort required to complete the telescope. Workmanship issues are difficult to predict because humans can make mistakes in unpredictable ways. The duration of integration and test tasks is difficult to accurately model for one-of-a-kind systems. The traditional method of building margin into the schedule to mitigate unforeseen delays failed for JWST because the delays could not be accurately predicted.

3. BACKGROUND AND LITERATURE REVIEW

The MADU framework is upon several foundational concepts. Previous work has established different levels of deep uncertainty and presented a variety of strategies for dealing with deep uncertainty. The constraint satisfaction literature provides the underpinning problem framework and the algorithmic concepts used to solve the problem.

Deep Uncertainty

Deep uncertainty refers to the situation where the probability of some future events cannot be accurately estimated. The framework developed in this paper is focused on addressing deep uncertainty in the space system development process and utilizes the following definition of deep uncertainty taken from Lempert et al. [8] [20]:

"the condition in which analysts do not know, or the parties to a decision cannot agree on, (1) the appropriate models to describe the interactions among a system’s variables, (2) the probability distributions to represent uncertainty about key variables and parameters in the models and/or (3) how to value the desirability of alternative outcomes.”

Deep uncertainty is related to the economic concept of Knightian uncertainty which refers to the inability to quantify all factors that affect a decision [21]. Deep uncertainty is commonly discussed as on a spectrum between total certainty and total ambiguity [22] [20]. Total certainty is the situation where everything is precisely known. Total ignorance is the situation where nothing is known. Neither extreme is realistic but they serve as two ends of a scale. Courtney defines four levels of uncertainty between these two extremes. Level one uncertainty is the situation where slight uncertainty exists but has a negligible impact on the system. Level two uncertainty is the situation where uncertainty exists and can be described statistically. Level three uncertainty is the situation where a range of possible outcomes exist but only a representative set of outcomes can be identified and quantified, with other outcomes not ruled out. Level four uncertainty is the situation where a large variety of outcomes exist, with no understanding of where a more likely outcome lies within the range of possibilities. The full range of possibilities may be unknowable. Level three and level four uncertainties are collectively referred to as deep uncertainty [23]. Strategies to make decisions under deep uncertainty utilize one or more of four principles: resistance, resilience, robustness, or adaptation [24] [20]. For level four deep uncertainty, adaptive methods are used because they don’t require an accurate model of future events. Therefore, the MADU framework pursues an adaptive strategy for handling deep uncertainty.

Constraint Satisfaction Problems

A constraint satisfaction problem (CSP) is a mathematical problem in which each variable within a set of variables must be assigned a value such that all constraints present in the problem are satisfied [25]. The MADU framework builds on algorithms that solve CSPs in order to efficiently solve the spacecraft design problem in the presence of deep uncertainty. Constraint problems are typically solved using an algorithm that incrementally constructs a series of candidate solutions. Using propagation methods, after a variable has been assigned a value, constraints can be used to prune the set of allowed assignments to unassigned variables. If all possible values are pruned from the domain of an unassigned variable, then the search algorithm backtracks by unassigning variables and proceeds down a different branch of the search tree. The algorithm terminates when a satisfying solution is found.

A particularly useful concept for efficient search algorithms is that of conflict learning. A conflict is a partial set of variable assignments that cannot be extended into a full assignment that satisfies all constraints [26]. Conflicts can be used during search to identify and avoid unsatisfiable regions of the search space. Algorithms that are able to identify and use conflicts
during search such as conflict-directed A* have proven to be efficient at solving CSPs [27]. To support adaptive design under deep uncertainty, efficient search techniques are needed to minimize the work needed when new information is learned. Incremental search algorithms are designed to be efficient when relatively small changes are made to a problem and therefore perform well within an adaptive design framework. The MADU framework combines the conflict-directed A* algorithm with explanation reuse-based incremental algorithms for dynamic constraint satisfaction problems in order to solve the spacecraft design problem in the presence of deep uncertainty [28] [29] [30].

4. METHODOLOGY

The Model-based Adaptive Design under Uncertainty (MADU) design framework models the spacecraft design problem as a dynamic, chance-constrained, stochastic optimization problem and solves that problem using incremental, informed, conflict-directed search that performs efficient updates to the optimal solution when changes to the problem occur. Furthermore, the framework uses a descriptive system model to store the system architecture and to express how the system may vary. The framework requires that design decisions are always revisited when new information is learned. In order to minimize the work needed to re-analyze decisions when new information is learned, MADU only revisits decisions whose outcome is affected by the new information. The decisions that are affected are identified automatically based on the characteristics of the new information. The MADU framework can be applied to guide a design team during the development process or it can be implemented computationally as an automated optimization algorithm. This paper describes a computational implementation of the framework detailed in [9].

Problem Formulation

The space system design problem is defined as a series of finite domain, chance-constrained, stochastic optimization problems. Changes to problem structure can occur between solutions of these problems. The goal for each individual optimization problem is to find the best solution that satisfies all constraints and minimizes the objective function. Nothing is known about the form of future problems, modeling level four deep uncertainty where future events cannot be accurately predicted. The formulation for an individual optimization is shown in equation 1. The goal of the problem is to optimize the assignments to a set of design variables while meeting constraints and considering uncertainty in the value of the design variables.

\[
\begin{align*}
\min_{x} & \quad f(x, y) \\
x & \in \{x_1, \ldots, x_m\} \\
v_i & \in \{v_{i1}, \ldots, v_{in}\} \\
x_i & \in v_i \\
c & = \{c_1, \ldots, c_k\} \\
h(x, y) & = 0 \\
P(g(x, y) < \alpha) & > p \\
p & \in [0, 1] \\
a(x) & = \{v_{ij}, \ldots, v_{mz}\} \\
g(x, y), h(x, y), a(x) & \in c \\
y & = \{y_1, \ldots, y_w\}
\end{align*}
\]

The set \( x \) represents the set of design variables that define the design of the system. The set \( v_i \) represents the domain for a design variable \( x_i \). The set \( v_i \) is made up of the design variable alternatives \( v_{ij} \). Each design variable alternative \( v_{ij} \) is a random variable which models the inability of the design team to precisely set system properties. There are no restrictions on the types of distributions that can be used for each design variable alternative. The set \( v_i \) must be finite. Each design variable must be assigned a random variable from its domain that satisfies all constraints in the set of constraints \( c \) while minimizing the objective function \( f \).

The set of constraints \( c \) is divided into equality constraints \( h(x, y) \), inequality constraints \( g(x, y) \), and set constraints \( a(x) \). Each equality constraint is defined by a function \( h(x, y) \) that must be equal to zero. Each inequality constraint is made up of a function \( g(x, y) \) and an upper limit \( \alpha \), such that \( g(x, y) \) must be less than \( \alpha \). The value \( p \) represents the minimum acceptable probability of meeting any inequality constraint. Therefore, the inequality constraints in this formulation are chance constraints. Set constraints are used to represent valid combinations of design variable alternatives with set notation, as opposed to the algebraic notation used in equality and inequality constraints, and are defined over a subset of the design variables. Each set constraint \( a(x) \) is a set of sets. Each inner set contains several design variable alternatives, one for each design variable that is in the scope of the set constraint. The inner sets define acceptable assignments to the design variables in the scope of the set constraint. If a design variable is assigned an alternative that is contained in an inner set of a set constraint, then the other design variables within the scope of that set constraint must be assigned the alternatives within that same inner set. If a design variable is assigned a value that is not contained in any inner set of any set constraint, then that assignment doesn’t place any additional constraints on the set of satisfying assignments to the other design variables. The set \( y \) is the set of parameters for the problem. Parameters are constants used in the problem formulation that cannot be altered by the optimizer but affect the value of the objective function and the satisfaction of constraints. The objective function is represented by the function \( f(x, y) \). The MADU framework only supports problems with a single objective function.

Solution Methodology

The MADU framework has four steps as shown in Figure 1. In step one, system information is captured in a system model. In step two, information is extracted from the system model and used to perform a design optimization. The optimization records the rationales for certain design decisions.
The rationale for each decision is recorded so that decisions can be revisited if necessary. In step three, the system model is updated with the optimal design and the decision rationales. Adhering to the single-source-of-truth maxim for MBSE, the decisions and rationales are stored in the system model alongside the design. The single-source-of-truth maxim dictates that all system information be stored in the system model so that retrieval of information is simplified and duplication of information is prevented. The results of the optimization are used by the design team to inform the development of the system. Next, if system development is complete and the design can no longer be changed, then the framework is exited. If development is not complete, then step four is triggered when the design team learns new information. In step four, the system model is updated to reflect the new information. Any number of changes can be made to the model before completing step four. Next, a design update is performed by returning to step two. Information from the previous problem is reused in order to improve the efficiency of finding the optimal solution to the changed problem. Once the new optimal design is found, step three is repeated and the optimal design, along with the decisions that lead to it, are once again captured in the system model.

**Step One: Construct system model**

The first step of the MADU framework is to capture system information in a system model. The system model must contain the following pieces of information:

- A description of the design of the system
- Elements within the design that can be varied and a set of alternatives for each variable
- Constraints defining the numerical or logical relationships within the system
- A measure of effectiveness to be optimized

The MADU framework uses SysML to construct the descriptive system model [31]. Once each of these pieces of information is captured in the system model, step two is triggered.

**Step Two: Optimize design while recording rationales**

In step two, information from the system model is extracted and used to find the optimal design. The optimization is performed while recording the rationale for any choices made during the optimization. The MADU framework uses conflict-directed search to find the optimal design. The design decisions made in the optimization are assignments to the design variables. Two types of rationales are identified and utilized during search: conflicts and satisfying states. A conflict is a set of design variable alternatives that cannot be extended into any full assignments that satisfy a set of constraints [26]. For each conflict, the rationale for the conflict is defined as the minimal set of constraints that entail the conflict. That set of constraints is recorded as the conflict explanation. A satisfying state is a partial set of design variable assignments that can be extended to at least one full set of design variable assignments that satisfies all constraints. In other words, a satisfying state is a state that has been proven to not be a conflict nor a superset of a conflict. The MADU framework doesn’t record rationales for each satisfying state because the maximal set of constraints that imply that a state is satisfying contains all constraints present in the problem.

The algorithm to find the optimal solution follows the architecture of the conflict-directed A’* algorithm in which candidate solutions are generated with a relaxed problem and a satisfiability checker determines if those candidate solutions meet constraints. This architecture is shown in Figure 2. Conflicts are stored in the list $C$ while satisfying states are stored in the list $T$.

**Relaxed Problem**—In the first part of step two, the set of design variables $x$, the design variable domains $v_i$ for each $x_i$, the set constraints $a$, the objective function $f$, and the set...
of known conflicts $C$ are used to solve a relaxed problem. A relaxed problem is a problem that has fewer restrictions on design variable assignments than the full problem [25]. The relaxed problem is used in order to compute an admissible heuristic, equivalently a bounding function, for the cost of the solution to the full problem. An admissible heuristic is an estimate of the cost of the solution to the problem that is guaranteed to be an underestimate [25]. Such heuristics are useful because they can be used to provide a guarantee that an optimal solution can be found [32]. The relaxed problem ignores the equality constraints $h(x,y)$ and inequality constraints $g(x,y)$ but resolves all known conflicts. With fewer restrictions on the output of the relaxed problem, the solution that is generated by the relaxed problem will have a cost that is less than or equal to the optimal solution to the full problem.

The goal of the relaxed problem is to find a full assignment of all design variables that satisfies the set constraints $a(x)$ and is not a superset of any known conflict. The solution approach to the relaxed problem can be thought of as a tree search as shown in Figure 3. Each layer of the tree corresponds to one design variable, each node in the tree represents an assignment to a design variable, and a path from the root of the tree to a leaf node represents a complete assignment to all design variables. Conflicts are depicted as dark red nodes in the tree. Each conflict prunes all child nodes from the search space (shown in light red) as any superset of a conflict is guaranteed to be unsatisfiable. Therefore, the algorithm must avoid any portion of the tree with conflicts and might instead choose the green set of assignments as a candidate solution. If conflicts are known from the previous problem, portions of the search space explored in the relaxed problem can be pruned a priori.

The algorithm to solve the relaxed problem is shown in Figure 4. It takes as inputs the set of design variables $x$, the set of domains for the design variables $v$, all known set constraints $a(x)$, the objective function $f$, and the set of all known conflicts $C$. Because the value of each design variable alternative cannot be precisely chosen, each design variable alternative is a random variable. Therefore, uncertainty must be considered in the relaxed problem. The MADU framework uses a Monte Carlo sampling strategy where samples from the uncertainty distributions for each design variable alternative are repeatedly drawn and an optimization is performed with each set of samples. This strategy means that each individual optimization can ignore uncertainty. Samples continue to be drawn until the probability that the lowest cost solution has been found meets a chosen threshold. To determine the number of samples that are required, the algorithm applies a Bernoulli trial framework. There are three conditions that must be met for a Bernoulli trial framework to apply [33]:

- Each trial results in either success or failure
- The probability of success remains constant across the trials
- The trials are independent

This application meets the first condition because a successful trial is defined as finding a new solution with a cost lower than the currently-known lowest cost. The second condition is met because the values for each design variable alternative are sampled using a consistent methodology and the current lowest cost remains fixed across the trials. The third condition is met because the samples for each variable are independent within each trial and each trial is independent from the preceding and following trials. To estimate the binomial proportion in situations where a number of trials have been performed without any successes, the "rule of three" is used [34]. The "rule of three" is an approximation of the upper 95% confidence bound when estimating a binomial proportion after a number of samples in which no successes have been observed. It states that, after $n$ trials without any successes, it can be concluded with 95% confidence that the probability of success lies below $3/n$. Once the lowest cost solution is likely found, the set of design variables assignments comprising the candidate solution $x_c$ is passed to the second half of step two.

Satisfiability Testing and Conflict Extraction—In the second part of step two, the candidate solution generated by the relaxed problem $x_c$ is checked for satisfiability. If it is satisfiable, then the candidate solution $x_c$ is returned as the optimal solution to the full problem $x^*$. If it is unsatisfiable, then conflicts are extracted as the set $\hat{C}$, added to the set of known conflicts, and the relaxed problem is restarted. Conflicts are extracted in a black box manner by testing each member of the power set of the candidate solution as shown in Figure 5. The figure shows an example for a problem with three design variables. The root node in this diagram represents the candidate solution generated by the relaxed problem which contains an assignment to all design variables. The other nodes in the diagram are all subsets of the set of assignments to the design variables in the candidate solution. All subsets must be tested in order to find all conflicts for a given candidate solution. However, if a subset is shown to be satisfiable, then large portions of the search space can be pruned because all subsets of a satisfying state are also satisfiable. For example, the root of the graph is shown in red in the figure to indicate that it is unsatisfiable. However, if one of its children is found to be satisfiable, as shown by the dark blue color, then all subsets of that node do not need to be tested because they are guaranteed to be satisfiable. The members of the power set that can be pruned based on the satisfying state are colored light blue. The conflict extractor continues to test other subsets of the candidate solution until all subsets have been determined to be unsatisfiable, been determined to be satisfiable, or are a subset of a satisfiable state. If satisfying states are known from the previous problem, portions of the conflict extraction search space can be pruned without any satisfiability testing, increasing search efficiency.

The algorithm to test satisfiability and extract conflicts is shown in Figure 6. The algorithm takes as inputs the candidate solution from the relaxed problem $x_c$, the set of design

![Figure 2. The architecture of the MADU optimization algorithm.](image-url)
The relaxed problem portion of the algorithm can be viewed as tree search. The algorithm must assign a value to each design variable while avoiding conflicts. Each layer of the tree corresponds to one design variable, each node in the tree represents an assignment to a design variable, and a path from the root of the tree to a leaf represents a complete assignment to all design variables. Conflicts are shown as dark red nodes in this picture and the portion of the search tree pruned by the conflicts is shown by the light red nodes. The chosen set of assignments is shown in green.

The algorithm used to solve the relaxed problem in order to compute an admissible heuristic for the cost of the solution to the full problem.

The conflict extraction portion of the algorithm viewed as search through the power set of a candidate solution. Satisfiable states (shown in dark blue) can prune large portions of the search space (shown in light blue) because all subsets of a satisfiable state are guaranteed to be satisfiable and therefore are not conflicts.

variable alternatives \( \nu \), the set of set constraints \( a(x) \), the set of equality constraints \( g(x) \), the set of inequality constraints \( h(x) \), the set of known conflicts \( C \), the set of known satisfying states \( t \), and the minimum acceptable probability of meeting any inequality constraint \( p \). Like the first part of step two, the uncertainty in the problem is dealt with using a Monte Carlo sampling strategy. First, a sample is taken from each design variable alternative in the candidate solution \( x_c \) to produce a new set of assignments to the set of design variables \( x_1 \). \( x_1 \) is placed at the head of a first-in-last-out (FILO) queue. A loop is then entered where the set of assignments to the design variables at the head of the queue is removed from the queue and passed to a satisfiability solver. If the set of assignments to the design variables is satisfiable, then that set of assignments is added to the set of possible satisfying states \( t_{\text{poss}} \). \( x_1 \) cannot be guaranteed to be a satisfying state because the satisfiability test is based on one sample taken from the distributions of the design variable alternatives. If \( x_1 \) is unsatisfiable, then the satisfiability checker returns a conflict and the explanation for that conflict. This conflict is added to the set of possible conflicts \( C_{\text{poss}} \). Again, that set of assignments is only possibly a conflict because it is
based on one sample taken from the distributions of the design variables. Following the identification of a conflict, new sets of assignments to the design variables that resolve the newly identified conflict are produced by unassigning variables involved in that conflict. Those new sets are added to the queue only if they are not a subset of a satisfying state. This loop continues until the queue is empty. Both $k_{cross}$ and $C_{cross}$ track the number of times a given possible satisfying state or possible conflict are detected in a sample from the candidate solution and use those numbers to compute a probability across all the samples performed so far that a given set of assignments is either a possible satisfying state or a possible conflict. Samples are repeatedly taken from the candidate solution $x_c$ until the probabilities of all possible conflicts and possible satisfying states stabilize. The probabilities are updated after each sample using either the “rule of three” or the Agresti-Coull interval [34] [35]. The “rule of three” is used if the conflict or satisfying state has been detected in all samples taken so far. Otherwise, the Agresti-Coull interval is used. After sufficient samples have been taken that the probabilities of each possible satisfying state or possible conflict are stable, those probabilities are used to determine if each possible satisfying state is a true satisfying state for the whole problem and likewise with the set of conflicts. This determination is based on comparing the probability of each possible satisfying state or conflict against the minimum acceptable probability of meeting any inequality constraint $p$. If the calculated probability of a possible conflict is greater than one minus $p$, then that conflict is added to the set of known conflicts for the overall problem. If the calculated probability of a possible satisfying state is greater than $p$, then it is added to the set of satisfying states for the overall problem. The new set of conflicts $C$ is passed back to the relaxed problem to continue searching for the optimal solution.

The loop shown in Figure 4 is repeated until a candidate solution produced by the relaxed problem is found to be satisfactory. At that point, step two is complete and the framework moves onto step three. The output from step two is the optimal solution $x^*$, the list of satisfying states $t$, and the list of conflicts $C$.

**Step Three: Update system model with optimal design and rationales**

In step three, the system model is updated with the results of the optimization. The optimal design is stored in the system model as well as the list of satisfying states and conflicts. The optimal assignments to design variables are represented by assigning default values to each of the Value Properties that represents a design variable. A conflict is represented by a Block composed of two parts. One part represents the set of conflicting design variable assignments and is modeled with a Block. This Block is tied to all Enumeration literals that represent a conflicting design variable assignment using a custom relationship called variableAssignments that extends the Dependency metaclass to define a set of design variable assignments. The other part of the conflict represents the set of Constraint Blocks that make up the conflict justification and is also represented by a Block. Each Constraint Block in the conflict justification is tied to the Justification Block using an association relationship. Satisfying states are modeled with a similar but simpler pattern. A satisfying state is represented by a Block with one or more variableAssignment relationships between that block and the Enumeration literals that represent the design variable alternatives that are part of that satisfying state. After updating the system model, if system development is complete, the framework terminates. If development is ongoing, then the framework waits until new information is learned and then proceeds to step four.

**Step Four: Update system model with new information**

When new information that may impact the system design is learned, the system model is updated to incorporate that new information. The new information must fall into the categories of acceptable changes defined in Table 1. The system model is kept updated as new information emerges and may be updated any number of times before the next optimization. The more changes to the problem that occur between optimization, the less likely it is that a given conflict or satisfying state survives the pruning that will occur when step two is repeated. Therefore, optimizations should be performed on as frequent a basis as possible to maximize information reuse.

Firstly, a design variable can be added or removed from the set of design variables $x$. In the MADU framework, the only way to add a design variable to the problem is to transform a parameter $y_{old}$, whose value is already specified in the problem formulation, into a design variable. The domain of this new design variable must only have a single member, a number that is equal to the value of the old parameter ($v_{new} = y_{old}$). In such a transformation, the solution to the problem isn’t changed. Once a new design variable has been introduced, other types of changes can be used to further change the problem by modifying the domain of the new design variable. A design variable can be removed from the problem only if it only has a domain with a single value and that value has no uncertainty. In that case, the design variable can be removed from the set of design variables $x$ and added to the set of model parameters $y$. The new parameter must have the same value as the single alternative of the old design variable. This transformation also means that the solution to the problem isn’t changed. Other types of changes are used to restrict the domain of the design variable to prepare it for removal.

Secondly, the domain of a design variable can be extended or contracted by adding or removing an alternative from the set $v_{i}$. Sequential contractions and extensions can be used to affect a change to a design variable alternative.

Thirdly, a constraint can be added to or removed from the set of constraints $c$. Most changes to a constraint should be modeled as a removal of the old version of that constraint and an addition of the new version of that constraint. However, certain changes in inequality constraints are handled by a different process. An inequality constraint can be tightened by decreasing its limit $\alpha$ or by increasing the minimum probability of satisfaction $p$. Similarly, an inequality constraint can be relaxed by increasing its limit $\alpha$ or by decreasing the minimum probability of satisfaction $p$. These changes can be handled in one step because their effect on the set of possible solutions is straightforward. In contrast, a general change to a constraint may have subtle effects on the set of possible solutions and so must be handled through the removal of the old version of the constraint and addition of the new version of the constraint.

Fourthly, the objective function can be changed to a different functional form. Within the MADU framework, there are no constraints on the form of the objective function.
Figure 6. The conflict extraction portion of step two iteratively checks satisfiability of the power set of the candidate solution \(x_c\). Unsatisfiable sets of assignments to design variables are recorded as conflicts in the set \(\hat{C}\) while satisfying set of design variable assignments are recorded as satisfying states in the set \(t\).

Table 1. The different types of changes that may be made to an optimization problem within the MADU framework.

| Type of Change                  | Problem Formulation Change | Example                                      |
|--------------------------------|----------------------------|----------------------------------------------|
| Addition of Design Variable    | \(x_{\text{changed}} \leftarrow x_{\text{orig}} \cup y_i \land y_{\text{changed}} \leftarrow y_{\text{orig}} \setminus y_i\) | Addition of gravitational slingshot maneuver to trajectory |
| Removal of Design Variable     | \(x_{\text{changed}} \leftarrow x_{\text{orig}} \setminus x_i\) | Removal of redundant component               |
| Addition of Design Variable Alternative | \(v_{\text{changed}} \leftarrow v_{\text{orig}} \cup v_{\text{new}}\) | New surface coating developed                |
| Removal of Design Variable Alternative | \(v_{\text{changed}} \leftarrow v_{\text{orig}} \setminus v_{\text{rem}}\) | Material found to violate contamination requirements |
| Addition of Constraint         | \(g_{\text{changed}} \leftarrow g_{\text{orig}} \cup g_{\text{new}} \lor h_{\text{changed}} \leftarrow h_{\text{orig}} \lor h_{\text{new}} \lor a_{\text{changed}} \leftarrow a_{\text{orig}} \lor a_{\text{new}}\) | Compatibility with additional ground stations desired |
| Removal of Constraint          | \(g_{\text{changed}} \leftarrow g_{\text{orig}} \setminus g_{\text{rem}} \lor h_{\text{changed}} \leftarrow h_{\text{orig}} \setminus h_{\text{rem}} \lor a_{\text{changed}} \leftarrow a_{\text{orig}} \setminus a_{\text{rem}}\) | Payload does not require high data rate interface |
| Tightening of Constraint       | \(\alpha_{\text{changed}} < \alpha_{\text{orig}} \lor p_{\text{changed}} > p_{\text{orig}}\) | Customer requires tighter pointing requirements |
| Relaxation of Constraint       | \(\alpha_{\text{changed}} > \alpha_{\text{orig}} \lor p_{\text{changed}} < p_{\text{orig}}\) | Launch vehicle capability increased          |
| Change Objective Function      | \(f \leftarrow f_{\text{new}}\) | Customer changes priority from cost to schedule |
**Repeat Step Two: Re-optimize design while recording rationales**

In a repeat of step two, decisions are revisited in light of new information learned in step four. Given the approach in the original step two, the decisions that need to be revisited are the identification of conflicts and satisfying states. The types of changes made to the model in step four to capture new information determine whether a conflict or satisfying state continues to apply to the problem. The rules for pruning conflicting and satisfying states are shown in Table 2. Conflicts and satisfying states are pruned from the set of known conflicts or satisfying states whenever it cannot be proven, without explicitly testing the satisfiability of the state, that the conflicts or satisfying states hold in the new problem.

Adding a design variable to the problem has no effect on the problem because the design variable simply replaces a parameter that was already in the problem. Therefore, neither the conflict list nor the satisfying list need to change. Similarly, removing a design variable from the problem doesn’t change the list of conflicts or satisfying states because the variable is transformed into a parameter and so the set of solutions to the problem doesn’t change.

Adding a design variable alternative has no effect on the list of satisfying states but may resolve some conflicts. Any constraint whose domain includes the design variable whose domain was expanded may now be satisfied with sets of previously unidentified variable assignments. Therefore, any conflict $C_i$ whose explanation $E(C_i)$ contains a constraint $c_i$ whose domain includes the variable $x_i$ whose domain was expanded with a new design variable alternative $v_{new}$ needs to be removed from the list of known conflicts unless the conflict contains a different design variable alternative $v_{ik}$ within the domain of the design variable $x_i$. Those conflicts will remain conflicts as they are not affected by any new variable combinations that may now be possible with the introduction of the new design variable alternative.

Removing a design variable alternative has no effect on the set of conflicts but may result in some satisfying states becoming conflicts. All satisfying states except those satisfying states $l_j$ that contain the design variable $x_i$ whose domain was reduced must be removed from the list of satisfying states.

Adding a constraint has no effect on the set of conflicts since the set of solutions to the problem will not expand when a constraint is added. However, adding a constraint means that all satisfying states must be removed since any satisfying state may no longer be satisfiable after the addition of the new constraint. Removing a constraint has no effect on the set of satisfying states since the set of solutions to the problem will not shrink but it does have potential effects on the set of conflicts. A conflict $C_i$ that contains the removed constraint $c_{rem}$ in its explanation $E(C_i)$ must be removed from the list of known conflicts.

Tightening a constraint is not will increase the set of solutions to the problem and so it has the same effect on the set of satisfying states and the set of conflicts as adding a constraint to the problem. Similarly, relaxing a constraint will not decrease the set of solutions to the problem so it has the same effect on the set of satisfying states and the set of conflicts as removing a constraint from the problem. A change in objective function only changes which satisfying states are preferred and so therefore has no effect on the set of conflicts or the set of satisfying states.

Once the set of conflicts and set of satisfying states have been appropriately pruned, the optimization can be performed again. This time, the set of conflicts and satisfying states that are passed to the optimization algorithm may not be empty. If some conflicts or satisfying states are preserved from the previous problem, the optimization algorithm can leverage this knowledge to find the new optimal solution faster than an algorithm with an identical search strategy that doesn’t have these pieces of information.

The principle of reusing conflicts from previous problems is an established technique for solving dynamic CSPs [36] [37] [38] [30]. In this algorithm, conflicts known before search begins improve the output of the relaxed problem. The initial solution generated by the relaxed problem will resolve all known conflicts and will therefore resolve more conflicts than an optimization that has to first identify conflicts. Therefore, fewer changes need to be made to the initial candidate solution to find the optimal solution, resulting in lower search time.

The reuse of satisfying states to improve conflict extraction is novel. The MADU framework uses a black box search strategy for extracting conflicts in which variables are gradually unassigned from an unsatisfying candidate solution. This process can be made more efficient by avoiding parts of the conflict extraction search space known to not have any conflicts. All satisfying states and subsets of satisfying states are guaranteed not to be conflicts. Therefore, satisfying states passed from the previous problem can be used to prune the conflict extraction search space. When extracting conflicts, child candidate solutions are only added to the queue if they aren’t already known to be satisfying. Therefore, reused satisfying states will prevent some child candidate solutions from being added to the queue and fewer child candidate solutions will need to be explicitly tested before the conflict extraction process is complete.

**Repeat Step Three: Update system model with new optimal design and new rationales**

This step is carried out in an identical manner to the original step three. The optimal design, conflicts, and satisfying states are stored in the system model. The results of the optimization are used to inform the system being designed. After completing this step, the decision node again determines if the framework continues to be followed or whether it should terminate. The loop between steps two, three, and four is continued until system development is over.

**5. REXIS THERMAL DESIGN CASE STUDY**

To walk through the MADU framework and explore its benefits, this case study solves a design problem based on the thermal subsystem of the REgolith X-ray Imaging Spectrometer (REXIS) spectrometer. This case study illustrates the capabilities of the MADU framework on a simple problem that can be intuitively understood. Deep uncertainty is introduced into the problem by adding a new design variable alternative after the problem has been initially solved. The new design variable alternative simulates information being learned during the development process.

**REXIS Overview**

REXIS is an instrument on board NASA’s Origins Spectral Interpretation Resource Identification Security Regolith EXplorer (OSIRIS-REx) spacecraft [39]. The OSIRIS-REx
mission will explore the near-Earth asteroid Bennu and return a sample of asteroid regolith to Earth [40]. REXIS contributes to the mission by observing the asteroid in the soft X-ray band in order to measure elemental abundances and ratios. REXIS was built by students at MIT, Harvard, and other universities. The overall goal of the REXIS instrument is to provide students with hands-on experience on a NASA mission working with NASA and industry professionals.

Figure 7 shows a CAD model of the REXIS instrument. REXIS is composed of two subassemblies: the main spectrometer containing the charge coupled devices (CCDs) and electronics and the Solar X-ray Monitor (SXM). The base of the main spectrometer contains the three electronics boards that run the instrument, communicate with the spacecraft, and regulate voltages. A thermal isolation layer separates the warm electronics box from the colder upper section of the spectrometer. The Detector Assembly Mount (DAM) contains the four detectors as well as several radioactive $^{55}$Fe calibration sources. To accurately measure the energy of each incident X-ray photon, the detector array is passively cooled below $-60^\circ$C. The cooling is accomplished by isolating the detectors from the warmer electronics box through low conductivity standoffs and connecting the detectors to a radiator using a high-conductivity thermal strap [41].

Problem Definition

This case study is motivated by the thermal design necessary to cool the REXIS detectors to below $-60^\circ$C. The design of the instrument was largely driven by this thermal design. In this case study, the size and material of the isolation layer and thermal strap are chosen to meet this temperature requirement while satisfying a maximum mass constraint.

A schematic of the portion of the REXIS instrument is shown in Figure 8. The detectors are housed in the Detector Assembly Mount (DAM). It is connected to the Detector Assembly Support Structure (DASS) through the Thermal Isolation Layer (TIL) and to the Radiator through the Thermal Strap.

The topology of the thermal system used in this problem is shown in Figure 9. Three nodes are present in the problem: the DASS, the DAM, and the Radiator. The DASS is modeled with a temperature $T_{DASS}$ of $-20^\circ$C (253 K). The DAM has an unknown temperature $T_{DAM}$ but has a maximum temperature limit of $-60^\circ$C (213 K). A heat load of 1.5 W is present on the DAM. This heat load $q_{DAM}$ comes from dissipation by detectors themselves, as well as by thermal radiation from surrounding, warmer components. Radiative heat transfer is not modeled outside of this heat load. The Radiator is assumed to have a temperature $T_{rad}$ of $-65^\circ$C (208 K). Two edges connect the three nodes. The lefthand edge represents the TIL which connects the DASS to the DAM. The TIL is made up of four standoffs. Each standoff has an unknown radius $r_{TIL}$. The TIL is modeled as a single edge, lumping the four standoffs into one when calculating the cross sectional area of the TIL $A_{TIL}$. The TIL also has an unknown conductivity $k_{TIL}$ and an unknown length $L_{TIL}$.

Accounting for bolted joint conductivity is an important consideration for accurately predicting the temperature of the DAM [42]. The bolted joint connecting the DASS to the TIL and the bolted joint connecting the DAM to the TIL both have a conductivity of 3.2 W/K.

The righthand edge represents the Thermal Strap which connects the DAM to the Radiator. The Thermal Strap has an unknown conductivity $k_{TS}$ and an unknown cross sectional area $A_{TS}$. The length of the Thermal Strap $L_{TS}$ is equal to 0.1 m. The bolted joint connecting the DAM to the Thermal Strap has a conductivity of 0.78 W/K while the bolted joint connecting the Thermal Strap to the Radiator has a conductivity of 0.40 W/K. The model parameters are

| Type of Change                  | Effect on Conflict List                                                                 | Effect on Satisfying List  |
|---------------------------------|-----------------------------------------------------------------------------------------|---------------------------|
| Addition of Design Variable     | None                                                                                    | None                      |
| Removal of Design Variable      | None                                                                                    | None                      |
| Addition of Design Variable     | if $v_{new} \in C_{i}$ and $c_{i} \in E(C_{i})$ then $C \setminus C_{i}$ unless $\exists v_{ik} \in E(v_{new}) \land v_{ik} \in v_{i}$ | None                      |
| Removal of Design Variable      | None                                                                                    | $\emptyset$ unless $\exists v_{ik} \in t_{i} \setminus v_{rem} \land v_{ik} \in v_{i}$ |
| Addition of Constraint          | None                                                                                    | $\emptyset$                |
| Removal of Constraint           | if $c_{rem} \in E(C_{i})$ then $C \setminus C_{i}$                                    | None                      |
| Tightening of Constraint        | None                                                                                    | $\emptyset$                |
| Relaxation of Constraint        | if $c_{rel} \in E(C_{i})$ then $C \setminus C_{i}$                                    | None                      |
| New Objective Function          | None                                                                                    | None                      |

Table 2. Summary of the effects of different changes on the list of conflicts and satisfying states.
TIL radius can take any value between 0 and 0.001 m in steps of 0.0001 m. Each geometric design variable alternative is modeled as a random variable with an uncertainty of 5% around its central value. In total, the problem has nine design variables.

**Equality Constraints**—The primary consideration for designing this thermal system is the temperature of the detectors. That temperature is calculated using the one-dimensional conductive heat transfer equation shown in equation 3. Conductive heat transfer \( q \) is determined by the temperature on the two ends of the interface \( T_a, T_b \), the length of the interface \( L \), the cross-sectional area of the interface \( A \), and the thermal conductivity of the material across which heat is transferred \( k \). The DAM temperature can be calculated by applying this equation to calculate the heat transfer across the TIL and across the Thermal Strap.

\[
q = \frac{kA}{L} (T_a - T_b) \tag{3}
\]

The DAM temperature is calculated using equations 4, 5, and 6 by applying the conductivity equation to calculate the heat transfer across the TIL and across the Thermal Strap. The first equation shows how the length of the Thermal Strap \( L_{TS} \), the conductivity of the Thermal Strap \( k_{TS} \), the cross sectional area of the Thermal Strap \( A_{TS} \), and the Thermal Strap bolted joint conductances \( k_{DAM-TS} \) and \( k_{TS-rad} \) are used to calculate the equivalent conductivity of the Thermal Strap \( \hat{k}_{TS} \). The second equation shows how the length of the TIL \( L_{TIL} \), the conductivity of the TIL \( k_{TIL} \), the cross sectional area of the TIL \( A_{TIL} \), and the TIL bolted joint conductances \( k_{DASS-TIL} \) and \( k_{TIL-DAM} \) are used to calculate the equivalent conductivity of the TIL \( \hat{k}_{TIL} \). The third equation shows how the equivalent conductivities of the TIL and Thermal Strap \( \hat{k}_{TIL} \) and \( \hat{k}_{TS} \), the temperature of the DASS \( T_{DASS} \), the temperature of the Radiator \( T_{rad} \), and the heat load on the DAM \( q_{DAM} \) are used to calculate the temperature of the DAM \( T_{DAM} \).

\[
\hat{k}_{TS} = \frac{1}{1/k_{DAM-TS} + 1/k_{TS-rad} + \frac{L_{TS}}{k_{TS}A_{TS}}} \tag{4}
\]

\[
\hat{k}_{TIL} = \frac{1}{1/k_{DASS-TIL} + 1/k_{TIL-DAM} + \frac{L_{TIL}}{k_{TIL}A_{TIL}}} \tag{5}
\]

\[
T_{DAM} = \frac{\hat{k}_{TIL}T_{rad} + \hat{k}_{TS}T_{DASS}}{\hat{k}_{TS} + \hat{k}_{TIL}} + q_{DAM} \tag{6}
\]

Additional equality constraints are used to calculate mass and cost of the design. Mass is calculated individually for the TIL and Thermal Strap as shown in equations 7 and 8 and then summed to calculate the total mass of the design as shown in equation 9. Similarly, cost is calculated individually for the TIL and Thermal Strap as shown in equations 10 and 11 and then summed to calculate the total cost of the design as shown in equation 12.

\[
m_{TS} = A_{TS}L_{TS}\rho_{TS} \tag{7}
\]

\[
m_{TIL} = A_{TIL}L_{TIL}\rho_{TIL} \tag{8}
\]
Table 3. The set of model parameters $y$ for the REXIS Detector thermal design problem

| Parameter Name                        | Parameter Symbol | Parameter Value |
|---------------------------------------|------------------|-----------------|
| DASS Temperature                      | $T_{DASS}$       | −20°C           |
| DAM Heat Load                         | $q_{DAM}$        | 1.5 W           |
| Radiator Temperature                  | $T_{rad}$        | −65°C           |
| DASS to TIL Bolted Joint Conductivity | $k_{DASS-TIL}$   | 3.2 W/K         |
| TIL to DAM Bolted Joint Conductivity  | $k_{TIL-DAM}$    | 3.2 W/K         |
| Thermal Strap Length                  | $L_{TS}$         | 0.1 m           |
| DAM to Thermal Strap Bolted Joint Conductivity | $k_{DAM-TS}$     | 0.78 W/K        |
| Thermal Strap to Radiator Bolted Joint Conductivity | $k_{TS-Rad}$     | 0.40 W/K        |

Table 4. The set of material alternatives for the Thermal Strap and TIL.

| Material                        | Conductivity W/(m K) | Density kg/m$^3$ | Volumetric Cost $/$cm$^3$ |
|---------------------------------|-----------------------|------------------|---------------------------|
| Aluminum 6061-T6 [43] [44]     | 167                   | 2700             | 0.177                     |
| 316 Stainless Steel [45] [46]   | 15.9                  | 7920             | 0.400                     |
| OFHC Copper [47] [48]           | 391                   | 8940             | 0.727                     |
| Titanium 6AL-4V [49] [50]       | 6.70                  | 4430             | 1.81                      |
| Torlon 5030 [51] [52]           | 0.360                 | 1610             | 3.66                      |

Table 5. The set of design variables for the REXIS thermal design problem, with the set of alternatives for each design variable listed.

| Name                        | Symbol | Units | Uncertainty | Alternatives               |
|-----------------------------|--------|-------|-------------|----------------------------|
| Thermal Strap Conductivity  | $k_{TS}$ | W/(m K) | 0.05        | 167, 15.9, 391, 6.70, 0.360 |
| Thermal Strap Cross Sectional Area | $A_{TS}$ | m$^2$ | 0.01        | $1 \times 10^{-4}, 1.5 \times 10^{-4}, 2 \times 10^{-4}$, $2.5 \times 10^{-4}, 3 \times 10^{-4}, 3.5 \times 10^{-4}$, $4 \times 10^{-4}, 4.5 \times 10^{-4}, 5 \times 10^{-4}$ |
| TIL Conductivity            | $k_{TIL}$ | W/(m K) | 0.05        | 167, 15.9, 391, 6.70, 0.360 |
| TIL Length                  | $L_{TIL}$ | m     | 0.01        | 0.005, 0.006, 0.007, 0.008, 0.009, 0.010, 0.011, 0.012 |
| TIL Radius                  | $r_{TIL}$ | m    | 0.01        | 0.004, 0.005, 0.006, 0.007 |
| TIL Density                 | $\rho_{TIL}$ | kg/m$^3$ | 0.05        | 2700, 7920, 8940, 4430, 1610 |
| TIL Volumetric Cost         | $s_{TIL}$ | $/$cm$^3$ | 0.05        | 0.177, 0.400, 0.727, 1.81, 3.66 |
| Thermal Strap Density       | $\rho_{TS}$ | kg/m$^3$ | 0.05        | 2700, 7920, 8940, 4430, 1610 |
| Thermal Strap Volumetric Cost | $s_{TS}$ | $/$cm$^3$ | 0.05        | 0.177, 0.400, 0.727, 1.81, 3.66 |

$m_{tot} = m_{TIL} + m_{TS}$ (9)

$S_{TS} = A_{TS}L_{TS}s_{TS}$ (10)

$S_{TIL} = A_{TIL}L_{TIL}s_{TIL}$ (11)

$S_{tot} = S_{TIL} + S_{TS}$ (12)

$m_{tot} = m_{TIL} + m_{TS}$ (9)

Constraint is shown in equation 14. The minimum probability of satisfying any inequality constraint $p$ is set at 0.95.

$T_{DAM} < −60$ (13)

$m_{tot} < 0.35$ (14)

Inequality Constraints—Two inequality constraints define acceptable solutions. The first inequality constraint expresses the requirement on DAM temperature $T_{DAM}$. That temperature must be less than $−60^\circ$C (213 K). The first inequality constraint is shown in equation 13. The second inequality constraint defines a maximum mass to replicate a mass allocation made by the design team. The total mass of the TIL and Thermal Strap cannot exceed 0.35 kg. The second inequality constraint is shown in equation 14. The minimum probability of satisfying any inequality constraint $p$ is set at 0.95.

Set Constraints—Set constraints are needed to ensure consistency in material selection across design variables. For example, the algorithm should not be able to choose the conductivity of copper (391 W/(m K)) and the cost of aluminum ($0.177/$cm$^3$). Therefore, two set constraints, each with multiple sets of satisfying design variable assignments, are included in the problem. The first set constraint enforces consistency for the Thermal Strap while the second set constraint enforces consistency for the TIL. The satisfying sets of design variable alternatives for each set constraint correspond to the rows in Table 4.
Figure 10. A radar plot showing the optimal value for five design variables after the initial solve of the REXIS detector thermal design problem.

Problem Size—The REXIS detector thermal design case study has 9 design variables, 2 inequality constraints, 9 equality constraints, and 2 set constraints. Accounting for the set constraints, there are 7200 possible solutions to the problem.

Solving the REXIS Thermal Design Problem

Step One: Construct system model—A SysML model is built that contains the design variables, design variable alternatives, constraints, and objective function defined in the previous section. Design variables are modeled in SysML as Enumerations. Each Enumeration owns a set of literals that define the set of alternatives for that variable. The uncertainty of each design variable is modeled using a Value Property owned by the Enumeration. Therefore, the uncertainty for each design variable alternative is the same. Constraint Blocks are added to the model to express the equality, inequality, and set constraints. The objective function is modeled by assigning a Value Property the measure of effectiveness Stereotype («moef»).

Step Two: Optimize design while recording rationales—With the system model constructed, the algorithm described in section 4 is run to find the optimal design. In this example problem, the optimal solution is found after checking 8 candidate solutions and identifying 29 conflicts. A total of 5735 calls to the optimizer and satisfiability checker are required and the total run time is 278.6 s. The optimal design is shown in Figure 10 and Table 6 and consists of a copper thermal strap with a cross sectional area of $3 \times 10^{-4}$ m$^2$ and a Torlon TIL with a radius of 0.004 m and a length of 0.012 m. Unsurprisingly, the optimizer chose a low conductivity material for the TIL and a high conductivity material for the Thermal Strap. Additionally, the TIL is made as tall and thin as possible to reduce conductivity and cost, while the Thermal Strap is made as thin as possible to reduce cost while remaining thick enough to meet the temperature constraint.

The algorithm records conflicts found during search. Table 7 shows two of the twenty-nine conflicts that were identified. The table shows design variable alternatives involved in the conflict and the explanation for the conflict. The first conflict is the selection of the 391 W/(mK) design variable alternative for the TIL conductivity $k_{TIL}$. The explanation for this conflict is the maximum DAM temperature constraint.

Intuitively, if a high conductivity is selected for the TIL, the Radiator cannot reject enough heat from the DAM to meet the maximum DAM temperature constraint. Therefore, setting the $k_{TIL}$ to such a high conductivity makes the problem unsatisfiable. The second conflict is the selection of the 2700 kg/m$^3$ density for the Thermal Strap. This selection implies that the Thermal Strap is made of aluminum. The explanation for the conflict is both the maximum DAM temperature constraint and the Thermal Strap set constraint. Intuitively, the conductivity of aluminum is not high enough to reject enough heat from the DAM to meet its temperature constraint. The conductivity of copper is sufficient to meet the DAM temperature constraint, as shown in Table 6 but the optimizer can’t simultaneously select the density of aluminum and the conductivity of copper because of the set constraint. The algorithm also records satisfying states found during search. In this case, only one satisfying state is found, the optimal solution.

Step Three: Update system model with optimal design and rationales—In step three, the system model is updated to capture the optimal design, set of conflicts, and set of satisfying states using the patterns described in section 4. See [9] for examples of how the optimal design, conflicts, and satisfying states are modeled.

Step Four: Update system model with new information—In step four, a new design variable alternative is introduced to simulate information being learned during the development process. The new alternative for TIL Length is added to the problem. The new alternative is a uniform random variable with a central value equal to 0.013 m and an uncertainty of 1%. This new design variable alternative is larger than any other alternative for TIL Length and so reflects a situation where the TIL can now be taller than its previous maximum height. Perhaps another component has been removed from the system, making more room for the TIL and DAM. This addition to the problem is an example of deep uncertainty because the change to the problem was not predicted in the initial problem formulation. To implement this change in the SysML model, a new Enumeration literal is added to the Enumeration that represents the $L_{TIL}$ design variable.

Repeat Step Two: Re-optimize design while recording rationales—Returning to step two, the information from the system model is extracted, changes to the problem are identified, the remaining conflicts and satisfying states are fed into the optimization algorithm, and the problem is re-solved. With the addition of a design variable alternative, all previously satisfying solutions remain satisfying. A conflict is pruned if a constraint in its explanation has the altered design variable in its domain, except in the case where the set of design variable alternatives that comprise the conflict contains a design variable alternative from the altered design variable. After applying these rules, the one satisfying state and the conflicts shown in Table 8 remain. These conflicts remain conflicts because they all involve assignments to the $L_{TIL}$ design variable. As a result of these conflicts, almost all of the alternatives for TIL Length can be ruled out before optimization begins.

A new solution is found after checking 9 candidate solutions, identifying 34 conflicts in addition to the 7 that were known from the previous problem, and making 5516 calls to the optimizer and satisfiability checker. The solution is found in 291.6 s. As a comparison, the same problem was run without utilizing any conflicts or satisfying states from the previous problem and it checked 10 solution candidates, made 6853
Table 6. The set of optimal design variables alternatives for the REXIS thermal design problem.

| Name                              | Symbol | Units   | Optimal Value |
|-----------------------------------|--------|---------|---------------|
| Thermal Strap Conductivity        | \( k_{TS} \) | W/(m K) | 391           |
| Thermal Strap Cross Sectional Area| \( A_{TS} \) | m²      | \( 3 \times 10^{-4} \) |
| TIL Conductivity                  | \( k_{TIL} \) | W/(m K) | 0.360         |
| TIL Length                        | \( L_{TIL} \) | m       | 0.012         |
| TIL Radius                        | \( r_{TIL} \) | m       | 0.004         |
| TIL Density                       | \( \rho_{TIL} \) | kg/m³   | 1610          |
| TIL Volumetric Cost               | \( s_{TIL} \) | $/cm³  | 3.66          |
| Thermal Strap Density             | \( \rho_{TS} \) | kg/m³   | 8940          |
| Thermal Strap Volumetric Cost     | \( s_{TS} \) | $/cm³  | 0.727         |

Table 7. Two of twenty-nine conflicts identified while searching for the optimal solution.

| Conflict | Explanation                                   |
|----------|-----------------------------------------------|
| \( k_{TIL} = 391 \) | Maximum DAM Temperature Constraint           |
| \( \rho_{TS} = 2700 \) | Maximum DAM Temperature Constraint, Thermal Strap Material Constraint |

Table 8. The full conflict set after accommodating the newly TIL Length design variable alternative.

| Conflict | Explanation                                      |
|----------|-------------------------------------------------|
| \( L_{TIL} = 0.005 \) | Maximum DAM Temperature Constraint               |
| \( L_{TIL} = 0.006 \) | Maximum DAM Temperature Constraint               |
| \( L_{TIL} = 0.007 \) | Maximum DAM Temperature Constraint               |
| \( L_{TIL} = 0.008 \) | Maximum DAM Temperature Constraint               |
| \( L_{TIL} = 0.009 \) | Maximum DAM Temperature Constraint               |
| \( L_{TIL} = 0.010 \) | Maximum DAM Temperature Constraint               |
| \( L_{TIL} = 0.011 \) | Maximum DAM Temperature Constraint, Thermal Strap Material Set Constraint, Maximum Mass Constraint |

Calls to the optimizer and satisfiability checker, and took 418.0 s to find the optimal solution, demonstrating the value of information reuse. The new optimal design is shown in Figure 11 and Table 9 and consists of an aluminum thermal strap with a cross sectional area of \( 4.5 \times 10^{-4} \) m² and a Torlon TIL with a radius of 0.004 m and a length of 0.013 m. The new alternative for TIL Length is chosen because it enabled the Thermal Strap to be changed to aluminum, saving cost.

Repeat Step Three: Update system model with new optimal design and new rationales—After the optimization is completed, the SysML model is updated once again with the optimal design, conflicts, and satisfying states. All old conflicts and satisfying states are deleted from the model and the new ones are created per the patterns described in section 4.

Summary
This case study illustrates how the MADU framework can solve spacecraft design problems efficiently in the presence of deep uncertainty. Using a simplified model of the thermal constraints on the REXIS detector, the MADU framework solved a new optimization problem formed through a change to the original optimization in 30% less time than an optimization algorithm that doesn’t reuse information. The time savings arose because of the reuse of information from the original optimization problem enabled by use of a system model. In other work, a larger set of cases is examined to determine how the MADU framework performs for other types of changes and for multiple changes made between optimizations [9]. The average savings found across all of those cases is a 58% reduction in search time.
Table 9. The set of optimal design variable alternatives for the changed REXIS thermal design problem, after the addition of a new design variable alternative for TIL Length. The old optimal values are shown for comparison.

| Name                        | Symbol | Units     | Old Optimal Alternative | New Optimal Alternative |
|-----------------------------|--------|-----------|-------------------------|-------------------------|
| Thermal Strap Conductivity  | $k_{TS}$ | W/(m K)   | 391                     | 167                     |
| Thermal Strap Cross Sectional Area | $A_{TS}$ | m²        | 3 x 10⁻⁴                | 4.5 x 10⁻⁴              |
| TIL Conductivity            | $k_{TIL}$ | W/(m K)   | 0.360                   | 0.360                   |
| TIL Length                  | $L_{TIL}$ | m         | 0.012                   | 0.013                   |
| TIL Radius                  | $r_{TIL}$ | m         | 0.004                   | 0.004                   |
| TIL Density                 | $\rho_{TIL}$ | kg/m³     | 1610                    | 1610                    |
| TIL Volumetric Cost         | $s_{TIL}$ | $$/cm³   | 3.66                    | 3.66                    |
| Thermal Strap Density       | $\rho_{TS}$ | kg/m³     | 8940                    | 2700                    |
| Thermal Strap Volumetric Cost | $s_{TS}$ | $$/cm³   | 0.727                   | 0.177                   |

6. CONCLUSION

The optimization framework presented in this paper is able to perform system design in the presence of deep uncertainty. The framework solves a dynamic, chance-constrained stochastic optimization problem using conflict-directed best-first search. Because the framework reuses information in an adaptive manner, it performs better than algorithms that don’t reuse information while not needing precise estimations of the future course of development of a space system.

The limitations of the framework include that it is designed for finite domain, single-objective, fixed architecture problems. While most engineering systems are best described with a mix of finite and continuous variable domains, the assumption made in the MADU framework of only finite variable domains is less limiting than it may appear. In aerospace systems, some system properties are unambiguously finite, such as the number of reaction wheels in a spacecraft. Other properties may be constrained to a finite set because of physical or financial limitations. Finally, variables that are truly continuous can be discretized for analysis. Handling continuous variables is challenging because a conflict only prunes an infinitesimally small part of the domain of a continuous variable. One avenue of future work is to remedy this limitation by employing techniques like interval arithmetic to make conflict-directed search efficient for continuous variables [53]. The choice for MADU to support only a single objective function is made for simplicity. The MADU framework could be extended to consider multiple objectives in the future. The choice of a fixed architecture is also made for simplicity. While MADU can transform model parameters into design variables and vice versa, no type of change defined in Table 1 enables the addition of model parameters. Future extensions to the MADU framework may remove this limitation.

Additional avenues of future work include improving the performance of the framework through better information storage and evaluation techniques, incorporating behavioral modeling in the system model to enable more complex optimizations to be performed, and application to other complex systems. In particular, the Starshade mission will be used as a case study to evaluate the capabilities and benefits of the MADU framework on a realistic architecture study. The Starshade mission aims to directly image exoplanets by using a space telescope with an external, formation flying occulter [54]. Starshade provides a good test of the MADU framework because its complex and unique mission design means that it will experience deep uncertainty and so the MADU framework will be particularly valuable in efficiently reacting to information that arises during development.

ACKNOWLEDGMENTS

This research was supported by a NASA Space Technology Research Fellowship (NNX14AL57H) and by the REXIS project (NN12FD70C, PO365358). This work was conducted while Mark Chodas was a Research Assistant at the Massachusetts Institute of Technology.

REFERENCES

[1] Emmons, Debra and Lobbia, Marcus and Radcliffe, Torrey and Bitten, Robert, “Affordability Assessments to Support Strategic Planning and Decisions at NASA,” in 2010 IEEE Aerospace Conference, 2010.
[2] NASA, Ed., NASA Systems Engineering Handbook Rev. 2. NASA, 2018.
[3] Hazelrigg, George, “A Framework for Decision-Based Engineering Design,” Journal of Mechanical Design, vol. 120, no. 4, pp. 653–658, 1998.
[4] Thunnissen, Daniel, “Propagating and Mitigating Uncertainty in the Design of Complex Multidisciplinary Systems,” Ph.D. dissertation, California Institute of Technology, 2005.
[5] Chen, Wei and Allen, Janet K and Tsui, Kwok-Leung and Mistree, Farrokh, “A Procedure for Robust Design: Minimizing Variations Caused by Noise Factors and Control Factors,” Journal of Mechanical Design, vol. 118, no. 4, pp. 478–485, 1996.
[6] Suh, Eun Suk and de Weck, Olivier and Chang, David, “Flexible Product Platforms: Framework and Case Study,” Research in Engineering Design, vol. 18, no. 2, pp. 67–89, 2007.
[7] Dezfui, Homayoon and Benjamin, Allan and Everett, Christopher and Maggio, Gaspare and Stamatelatos, Michael and Youngblood, Robert and Guarro, Sergio and Rutledge, Peter and Sherrard, James and Smith, Curtis and Williams, Rodney et al., “NASA Risk Management Handbook. Version 1.0,” 2011.
[8] Lempert, Robert, Shaping the Next One Hundred Years: New Methods for Quantitative, Long-Term Policy Analysis. RAND, 2003.
[9] Chodas, Mark, “Addressing Deep Uncertainty in Space Systems Management,” doctoral dissertation, Massachusetts Institute of Technology, 2012.
System Development through Model-based Adaptive Design," Ph.D. dissertation, Massachusetts Institute of Technology, 2019.

[10] Friedenthal, Sanford and Griego, Regina and Sampson, Mark, “INCOSE Model Based Systems Engineering (MBSE) Initiative,” in INCOSE 2007 Symposium, 2007.

[11] Fisher, Jerry, “From the Editor: Model-Based Systems Engineering: A New Paradigm,” INSCIGHT, vol. 1, no. 3, pp. 3–4, 1998.

[12] GAO, “Space Acquisitions: Stronger Development Practices and Investment Planning Needed to Address Continuing Problems,” GAO-05-891T, Tech. Rep., 2005.

[13] National Research Council, Pre-Milestone A and Early-Phase Systems Engineering: A Retrospective Review and Benefits for Future Air Force Acquisition. The National Academies Press, 2008.

[14] Gardner, Jonathan and others, “The James Webb Space Telescope,” Space Science Reviews, vol. 123, no. 4, pp. 485–606, 2006.

[15] GAO, “NASA: Assessments of Major Projects,” GAO-18-280SP, Tech. Rep., 2018.

[16] NASA, “NASA Completes Webb Telescope Review, Commits to Launch in Early 2021,” https://www.nasa.gov/press-release/nasa-completes-webb-telescope-review-commits-to-launch-in-early-2021, 2018, Accessed 04-03-2019.

[17] Casani, John and others, “James Webb Space Telescope (JWST) Independent Comprehensive Review Panel (ICRP) Final Report,” JWST Independent Comprehensive Review Panel, Tech. Rep., 2010.

[18] GAO, “James Webb Space Telescope: Project Facing Increased Schedule Risk with Significant Work Remaining,” GAO-15-100, Tech. Rep., 2014.

[19] ——, “James Webb Space Telescope: Project Meeting Cost and Schedule Commitments but Continues to Use Reserves to Address Challenges,” GAO-17-71, Tech. Rep., 2016.

[20] Walker, Warren and Lempert, Robert and Kwakkel, Jan, “Deep Uncertainty,” in Encyclopedia of Oper. Res. and Management Science 3rd Ed. Springer, 2013, pp. 395–402.

[21] Knight, Frank, Risk: Uncertainty and Profit. Houghton Mifflin Company, 1921.

[22] H. Courtney, 2020 Foresight: Crafting strategy in an uncertain world. Harvard Business Press, 2001.

[23] Cox, Louis , “Confronting Deep Uncertainties in Risk Analysis,” Risk Analysis, vol. 32, no. 10, pp. 1607–1629, 2012.

[24] Walker, Warren and Marchau, Vincent and Swanson, Darren, “Addressing deep uncertainty using adaptive policies: Introduction to section 2,” Technological Forecasting and Social Change, vol. 77, no. 6, pp. 917–923, 2010.

[25] Russell, Stuart and Norvig, Peter, Artificial Intelligence: A Modern Approach, 3rd ed.. Prentice Hall, 2010.

[26] Stallman, Richard and Sussman, Gerald, “Forward Reasoning and Dependency-Directed Backtracking in a System for Computer-Aided Circuit Analysis,” Artificial Intelligence, vol. 9, no. 2, pp. 135–196, 1977.

[27] Williams, Brian and Ragno, Robert, “Conflict-directed A* and its role in model-based embedded systems,” Discrete Applied Mathematics, vol. 155, no. 12, pp. 1562–1595, 2007.

[28] Jussien, Narendra, “The versatility of using explanations within constraint programming,” Université de Nantes, Tech. Rep., 2003.

[29] Dechter, Rina and Dechter, Avi, “Belief Maintenance in Dynamic Constraint Networks,” in Proceedings of the Seventh AAAI National Conference on Artificial Intelligence, 1988, pp. 37–42.

[30] Schiex, Thomas and Verfaillie, Gérard, “Nogood Recording for Static and Dynamic Constraint Satisfaction Problems,” in Proc. of 1993 IEEE Conference on Tools with AI, 1993, pp. 48–55.

[31] OMG, “OMG System Modeling Language v1.5,” 2015, https://www.omg.org/spec/SysML/1.5/PDF. Accessed 01-30-2019.

[32] Hart, Peter and Nilsson, Nils and Raphael, Bertram, “A Formal Basis for the Heuristic Determination of Minimum Cost Paths,” IEEE Transactions on Systems Science and Cybernetics, vol. 4, no. 2, pp. 100–107, 1968.

[33] Blitzstein, Joseph and Hwang, Jessica, Introduction to Probability. Chapman and Hall/CRC, 2014.

[34] Hanley, James and Lippman-Hand, Abby, “If Nothing Goes Wrong, Is Everything All Right? Interpreting Zero Numerators,” JAMA, vol. 249, no. 13, pp. 1743–1745, 1983.

[35] Agresti, Alan and Coull, Brent, “Approximate is Better than "Exact" for Interval Estimation of Binomial Proportions," The American Statistician, vol. 52, no. 2, pp. 119–126, 1998.

[36] Béssiere, Christian, “Arc-Consistency in Dynamic Constraint Satisfaction Problems,” in Proceedings of the Ninth National conference on Artificial intelligence, vol. 1, 1991, pp. 221–226.

[37] Debruyne, Romuald, “Arc-consistency in dynamic CSPs is no more prohibitive,” in Proceedings 8th IEEE Int. Conf. on Tools with AI. IEEE, 1996, pp. 299–306.

[38] Surynek, Pavel and Barták, Roman, “A New Algorithm for Maintaining Arc Consistency After Constraint Retraction,” in International Conference on Principles and Practice of Constraint Programming, 2004, pp. 767–771.

[39] Masterson, Rebecca and Chodas, Mark and Bayley, Laura and Allen, Branden and Hong, JaeSub and Biswas, Pronoy and McMenamin, Conor and Stout, Kevin and Bokhour, Ed and Bralower, Harrison and others, “Regolith X-Ray Imaging Spectrometer (REXIS) Aboard the OSIRIS-REx Asteroid Sample Return Mission,” Space Science Reviews, vol. 214, no. 1, p. 48, 2018.

[40] Laurentta, Dante and others, “OSIRIS-REx: sample return from asteroid (101955) Bennu,” Space Science Reviews, vol. 212, no. 1–2, pp. 925–984, 2017.

[41] Stout, Kevin and Masterson, Rebecca, “Thermal Design and Performance of the REgolith X-ray Imaging Spectrometer (REXIS) Instrument,” in Modeling, Systems Engineering, and Project Management for Astronomy VI, vol. 9150, 2014, p. 91501J.

[42] Gilmore, David, “Spacecraft Thermal Control Handbook Volume I: Fundamental Technologies,” 2002.
[43] MatWeb, “Aluminum 6061-T6; 6061-T651,” http://www.matweb.com/search/DataSheet.aspx?MatGUID=b8d536e0b9b54bd7b69e4124d8f1d20a, 2019, Accessed 04-08-2019.

[44] OnlineMetals.com, “Aluminum Round Bar 6061-T651-Cold Finish, 72” length,” https://www.onlinemetals.com/en/buy/aluminum/aluminum-round-bar-6061-t651-cold-finish/pid/18037, 2019, Accessed 04-08-2019.

[45] MatWeb, “Eagle Brass 316 Austenitic Stainless Steel, 1/2 Hardened,” http://www.matweb.com/search/DataSheet.aspx?MatGUID=0dd32ed08cbd48f488708d9f6e89665f, 2019, Accessed 04-08-2019.

[46] OnlineMetals.com, “Stainless Round Bar 316 Annealed Cold Finish, 72” length,” https://www.onlinemetals.com/en/buy/stainless/stainless-round-bar-316-annealed-cold-finish/pid/4481, 2019, Accessed 04-08-2019.

[47] MatWeb, “Eagle Brass 101 OFE Copper (Certified), 1/2 Hard,” http://www.matweb.com/search/DataSheet.aspx?MatGUID=8e2aa1a2ef74ba08293f70a378fcede6f, 2019, Accessed 04-08-2019.

[48] OnlineMetals.com, “Copper Round Bar 101-H04, 72” length,” https://www.onlinemetals.com/en/buy/copper/copper-round-bar-101-h04/pid/13898, 2019, Accessed 04-08-2019.

[49] MatWeb, “Titanium Ti-6AI-4V (Grade 5), Annealed,” http://www.matweb.com/search/DataSheet.aspx?MatGUID=a0655d261898456b958e5f825ae85390, 2019, Accessed 04-08-2019.

[50] OnlineMetals.com, “Titanium Round Bar 6al-4V, 72” length,” https://www.onlinemetals.com/en/buy/titanium/titanium-round-bar-6al-4v/pid/4736, 2019, Accessed 04-08-2019.

[51] MatWeb, “Solvay Specialty Polymers Torlon® 5030 Polyamide-imide (PAI), 30% Glass Fiber,” http://www.matweb.com/search/DataSheet.aspx?MatGUID=e7f9f5bcb90c49f799d46944ea581383, 2019, Accessed 04-08-2019.

[52] Drake Plastics, “Torlon Rod,” https://drakeplastics.com/torlon-rod/, 2019, Accessed 02-28-2019.

[53] Rossi, Francesca and Van Beek, Peter and Walsh, Toby, Handbook of Constraint Programming. Elsevier, 2006.

[54] Seager, Sara and others, “The Exo-S Probe Class Starshade Mission,” in Techniques and Instrumentation for Detection of Exoplanets VII, vol. 9605, 2015, p. 96050W.

**Biography**

**Mark Chodas**

Mark Chodas is a Systems Engineer at the NASA Jet Propulsion Laboratory. He graduated from the Massachusetts Institute of Technology with an S.B. in Aerospace Engineering in 2012, an S.M. in Aerospace Engineering in 2014, and a PhD in Aerospace Engineering in 2019. His research explores new capabilities enabled by model-based systems engineering. While at MIT, Mark was the instrument systems engineer for the REgolith X-ray Imaging Spectrometer (REXIS) instrument on the OSIRIS-REx mission.

**Rebecca Masterson**

Rebecca Masterson is the Co-Director of the MIT Space Systems Laboratory and Principal Research Scientist at the Massachusetts Institute of Technology. She has over 20 years of experience in spacecraft design and development including structural design, control structure interactions, system engineering, program management, and integration and test. She was the Instrument Manager for the OSIRIS-REx Student Collaboration Experiment, the REgolith X-ray Imaging Spectrometer (REXIS) and the integration and test lead for TESS (Transiting Exoplanet Survey Satellite). Dr. Masterson’s current research interests include model-based systems engineering, engineering management, uncertainty analysis as applied to engineering design, and integrated modeling. She teaches Satellite Engineering and Systems Engineering in the Aeronautics and Astronautics Dep. at MIT and holds B.S., M.S. and Ph.D. degrees in Mechanical Engineering, all from MIT.

**Olivier de Weck**

Olivier de Weck is Professor of Aeronautics and Astronautics and Engineering Systems at MIT where he teaches Satellite Engineering and System Optimization. He has authored over 300 publications (12 best paper awards since 2004) and is a Fellow of INCOSE, Associate Fellow of AIAA and Senior Member of IEEE. He worked with NASA’s Office of Emerging Space to develop the new Commercial Space Technology Roadmaps in 2018 and served as Editor-in-Chief of the journal Systems Engineering from 2013-2018. He is a former Senior Vice President of Technology Planning and Roadmapping at Airbus.