Artificial Intelligence for Engineering Design, Analysis and Manufacturing (2016), *30*, 390–405.
© Cambridge University Press 2016 0890-0604/16
doi:10.1017/S0890060416000378

Ontology-based executable design decision template representation and reuse

ZHENJUN MING,[1] YAN YAN,[1] GUOXIN WANG,[1] JITESH H. PANCHAL,[2] CHUNG-HYUN GOH,[3] JANET K. ALLEN,[4] AND FARROKH MISTREE[5]

[1]Beijing Institute of Technology, Beijing, China
[2]Purdue University, West Lafayette, Indiana, USA
[3]University of Texas at Tyler, Tyler, Texas, USA
[4]School of Industrial and System Engineering, University of Oklahoma, Norman, Oklahoma, USA
[5]School of Aerospace and Mechanical Engineering, University of Oklahoma, Norman, Oklahoma, USA

(Received October 1, 2015; Accepted May 31, 2016)

Abstract

In decision-based design, the principal role of a designer is to make decisions. Decision support is crucial to augment this role. In this paper, we present an ontology that provides decision support from both the “construct” and the “information” perspectives that address the gap that existing research focus on these two perspectives separately and cannot provide effective decision support. The decision support construct in the ontology is the compromise decision support problem (cDSP) that is used to make multiobjective design decisions. The information for decision making is archived as cDSP templates and represented using frame-based ontology for facilitating reuse, consistency maintaining, and rapid execution. In order to facilitate designers’ effective reuse of the populated cDSP templates ontology instances, we identified three types of modification that can be made when design consideration evolves. In our earlier work, part of the utilization (consistency checking) of the ontology has been demonstrated through a thin-walled pressure vessel redesign example. In this paper, we comprehensively present the ontology utilization including consistency checking, trade-off analysis, and design space visualization based on the pressure vessel example.

Keywords: Compromise Decision Support Problem; Decision-Based Design; Ontology; Reuse; Template

1. INTRODUCTION

Engineering design is increasingly recognized as a decision-making process (Mistree et al., 1990; Thurston, 1991; Hazelrigg, 1998; Lewis et al., 2006). Decision-based design is an approach to engineering design that recognizes the substantial role that decisions play in design (Lewis et al., 2006). Mistree et al. (1990) point out that the principal role of a designer is to make decisions, and decisions serve as markers to identify the progression of a design from initiation to implementation to termination. For augmenting human designers’ decision-making ability, thus generating quality designs, providing relevant decision support is of critical importance. Basically, decision support can be contributed from two perspectives, namely, the “construct” and “information” perspectives. The construct perspective emphasizes providing an analytical tool or approach in which decisions are mathematically formulated and rationally made based on the available information. Typical types of decision formulation include the utility-based decision making (Hazelrigg, 1998), which is based on objective rationality and seeks the best solution, and the decision support problem (DSP) Technique (Muster & Mistree, 1988), which is based on bounded rationality and seeks the “satisficing” (good enough) solutions. Based on these two types of decision formulations, researchers have developed a variety of approaches for dealing with uncertainty (Vadde et al., 1994; Resende et al., 2012), preferences (Fernandez et al., 2005; Kulok & Lewis, 2007), distributed design (Lewis & Mistree, 1998; Gu et al., 2002), and demand modeling (Wassenaar et al., 2005; Williams et al., 2007), which strengthen human designers’ ability of decision making in the design process.

Having analytical decision-making tools and approaches is not enough for making good decisions. The availability of necessary information is another factor that influences quality decision making, and it constitutes the information perspective for providing decision support. In most cases, much of

Reprint requests to: Janet K. Allen, School of Industrial and System Engineering, University of Oklahoma, 202 West Boyd Street, Suite 116, Norman, OK 73019, USA. E-mail: janet.allen@ou.edu

the information needed in current design decisions can be found from previous decisions, especially in those redesign (adaptive or variant) cases where the underlying design concepts are the same (Pahl et al., 2007) and significant amounts of information can be reused. Therefore, effectively organizing and reusing previous design decision knowledge is very important. Ontologies, which are explicit formal specifications of terms and relations among them (Gruber, 1993), are increasingly used for knowledge management in engineering design. For example, Li et al. (2008) created an ontology as a sophisticated indexing mechanism for structuring information repositories of unstructured documents in order to retrieve information with high precision and recall; Liu et al. (2013) modeled product family using ontology and suggested a framework for faceted information retrieval; Witherell et al. (2007) created an ontology for archiving and reusing optimization knowledge including assumptions, methods, and results; and Rockwell et al. (2008, 2009) documented the decision-related semantic information using ontology for sharing and reuse. These ontology-based methods do provide some degree of information-related support in decision making; however, the retrieved or instantiated information of the ontology is not computationally formulated thus is usually used as references that cannot be directly used to execute and generate a solution. There is a need for an ontology that captures information for decision making at the computational level that can be executed rapidly and facilitates designers efficiently reusing previous knowledge in decision making.

As to making decisions by reusing previous design knowledge, inconsistency is an issue that cannot be ignored. In many redesign (knowledge reuse) cases, partial modification of previous design information is often needed to reflect the changes, which may lead to inconsistency in the modified decision model and bad decisions. Therefore, in addition to the executability, the ontology should also support consistency checking to determine if the executed information is organized in a consistent manner. Furthermore, postsolution analysis, such as trade-off analysis and design space visualization, is also critical for rational decision making based on which designers can gain insights of their decisions and make sure that quality designs are eventually achieved.

To address these needs, we are developing a knowledge-based platform for decision support in the design of engineering systems (PDSIDES; Ming et al., 2015). This platform is supported by an ontology that incorporates both construct and information for providing decision support. The construct used in the platform is the compromise DSP (cDSP). The information is categorized into two types, namely, procedural and declarative information. The former is the meta-information represented by the elements and relations within the modular and executable cDSP template using ontology, while the latter is the domain-specific information populated according to the cDSP template ontology that can be reused in solving similar design problems. In this paper, we focus on development of the cDSP template ontology that serves as the core information model to provide decision support on PDSIDES and utilization of the ontology in practical engineering design, that is, identification of ways to reuse the populated template instances. The remainder of this paper is organized as follows. The foundations of this paper are discussed in Section 2, and the development of the cDSP template ontology is presented in Section 3. The cDSP model modification types are identified in Section 4 to facilitate the reuse of the populated template instances in future design scenarios. Section 5 is an example to demonstrate the usefulness of the ontology in practical engineering design. Section 6 is the conclusion.

2. FOUNDATIONS

In this section, foundations of this paper are presented, which include the mathematical decision model used as the construct for decision support, a "design for reuse" thinking of modeling decisions as templates, an ontology that serves as the representation scheme for organizing the information of decision templates.

2.1. Our fundamental decision model

In order to achieve rationality and obtain quality designs in decision making, decisions needs to be rigorously formulated. In this paper we adopt the DSP technique (Muster & Mistree, 1988) as the formalism to formulate decisions encountered in engineering design. The DSP technique seeks the bounded rationality and "satisficing" (good enough; Simon, 1976) solutions. In the DSP technique, two types of DSPs are identified: selection and compromise DSPs. Selection DSP deals with making a choice among a number of possibilities taking into account a number of measures of merit or attributes (Fernandez et al., 2005). cDSP deals with the determination of the "right" values (or combination) of design variables to describe the best satisficing system design with respect to constraints and multiple goals (Mistree et al., 1993). The cDSP has been adopted for solving many system design problems such as ships (Mistree et al., 1990), aircraft (Lewis & Mistree, 1995), materials (Seepersad et al., 2008), and so on. In this paper, we use the cDSP as our fundamental decision model for making decisions among multiple conflicting goals considering constraints.

The mathematical formulation of the cDSP is shown in Figure 1a. The system descriptors, namely, design and deviation variables, system constraints, system goals, bounds, and the deviation function, are described in detail in Mistree et al. (1993) and are therefore not repeated here. A two-dimensional problem shown in Figure 1b is used as an example to illustrate the features of the cDSP solution space. In the design space defined by variables (x_1 and x_2), feasible designs are confined in the feasible design space bounded by linear (C_1 and C_2), nonlinear (C_3) constraints, and the bounds (x_1^{lower}, etc.) on variables. Designers' task is to explore the feasible design space and find those designs that satisfy the multiple goals ($G_1 - G_4$) as much as possible, that is, to minimize the deviation (e.g., d_1^+, d_1^- referring to the over- and underdeviation

Given

An alternative to be improved, domain dependent assumptions

n	number of system variables
q	inequality constraints
p+q	number of system constraints
m	number of system goals
$g_i(X)$	system constraint function
$f_k(d_i)$	function of deviation variables to be minimized

Find

X_i	System variables	i = 1, …, n
d_i^+, d_i^-	Deviation Variables	i = 1, …, m

Satisfy

System constraints (linear, nonlinear)

$g_i(X) = 0 \quad i = 1, \ldots, p$

$g_i(X) \geq 0 \quad i = p+1, \ldots, p+q$

System goals (linear, nonlinear)

$A_i(X) + d_i^- - d_i^+ = G_i \quad i = 1, \ldots, m$

Bounds

$X_i^{min} \leq X_i \leq X_i^{max} \quad i = 1, \ldots, n$

$d_i^+, d_i^- \geq 0;\ d_i^+ \cdot d_i^- = 0$

Minimize

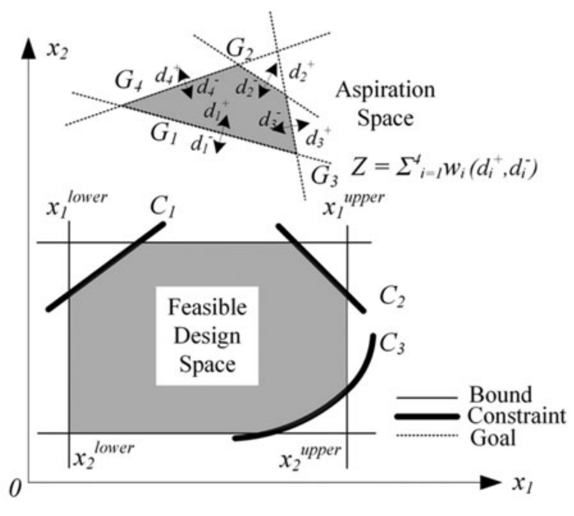
Deviation Function: Archimedean

$Z = \Sigma_{i=1}^{m} w_i(d_i^-, d_i^+)$

Deviation Function: Preemptive

$Z = [f_i(d_i^-, d_i^+), \ldots, f_k(d_i^-, d_i^+)] \quad i = 1, \ldots, m$

(a)



(b)

Fig. 1. Mathematical form and a two-dimensional illustration of a compromise decision support problem.

of the goal G_1) of the goals. In practical engineering design, the goals are usually conflicting (e.g., minimizing the weight and maximizing the volume of a container), which leads to the formation of the aspiration space (see the inner area framed by $G_1 - G_4$ where d_1^+, d_2^-, d_3^-, and d_4^- are preferred). When there is no overlap between the aspiration space and the feasible design space, designers need to compromise, that is, to find the shortest distance (represented by the deviation function Z) between the two spaces. The adaptive linear programing algorithm (Mistree et al., 1993), which has been implemented in the decision software DSIDES (Reddy et al., 1996), is used to solve cDSPs.

2.2. Decision template

Design reuse, aiming at maximizing customer satisfaction with minimal resources and cost and with minimal effort by the reuse of successful past designs, in part and in whole (Sivaloganathan & Shahin, 1999), has been well adopted in industry. Examples can be found in many product architectures where the modular, standardized components are designed once and reused in assemblies across products. Analogously to product architectures, Panchal et al. (2004) extend the reusability to the design process by proposing the concept of a modular, executable decision template from a decision-centric perspective. The key idea is that in a computational environment, design processes are composed by a series of templates representing decisions that can be instantiated with product-specific information and executed. Due to the modularity, these templates can be easily reused in similar product designs or product redesigns. The template is modeled based on the cDSP construct and analogous to the architecture of a printed wiring board with a number of electronic components, as shown in Figure 2. The key feature is the separation of "declarative" and "procedural" information: the former stands for the elements of the cDSP, such as variables, parameters, and constraints, and is represented by "chips" in the figure; the latter stands for the algorithm that solves the cDSP and is represented by the "wiring" and the "board." The reusability of the template extends to both the "declarative" and the "procedural" information.

Although the implementation by Panchal et al. (2004) enables greater reusability of information across design problems, the use of the extensible markup language (XML) as its underlying representation scheme limits the application of the cDSP template maintaining consistency. In many redesign cases, partial modification of the design model (cDSP template) is needed to reflect the evolution in the design consideration, which often leads to inconsistency in the modified model. These inconsistencies are even severe when the model is highly complex (e.g., tens of variables, constraints or goals) and the model modifier is not the original creator who has the full knowledge about the model. XML fails to avoid inconsistency because of the loose definition of items of the template and the lack of reasoning mechanism to support consistency checking; thus, XML is not suitable for representing the

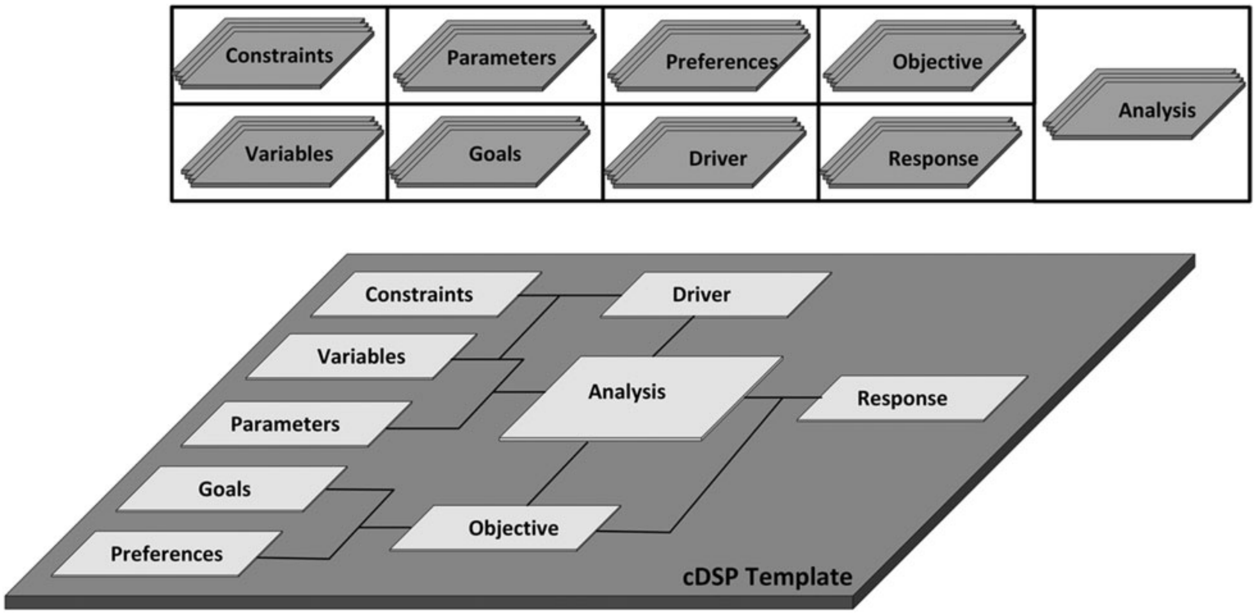


Fig. 2. Compromise decision support problem template (Panchal et al., 2004).

highly constrained cDSP model. In this paper, we adopt the idea of modeling cDSPs as executable template, rerepresent the templates using ontology, and demonstrate its consistency maintaining ability when the templates are modified.

2.3. Ontology

Ontology, known as explicit specification of a conceptualization, has many obvious advantages such as sharing information, integrating different applications, implementing interoperability, and reusing knowledge (Yang et al., 2008). Although the research on ontologies has its root in computer science, ontologies have been widely used for information modeling in engineering. Lee et al. (2012) proposed an ontology-based multilevel product-modeling framework, which is based on the Core Product Model 2 (Fenves et al., 2008) proposed by the National Institute of Standards and Technology (NIST), to address the requirement of semantic richness from different stakeholders across the product lifecycle. Lu et al. (2015) used ontology for capturing geometric constraint specifications to facilitate data exchange between different product development systems. Barbu et al. (2012) created OntoSTEP based on an ontology that can transform digital models with geometric information into semantically rich models that include function and behavior to facilitate manufacturing. Many other ontologies have been created by researchers and engineers based on their domain of interest. To systematically evaluate these ontologies, Chandrasegaran et al. (2013) recommended the establishment of a Global Center for Engineering Ontologies, similar to the National Center for Biomedical Ontologies (http://www.bioontology.org/) in the United States.

A foreseeable consequence of the creating of these ontologies is that a huge amount of product- or problem-specific, semantically rich information will be populated as instances and accumulated in the associated knowledge bases. From the perspective of design guidance, the primary requirement is the availability of critical information when it is needed. This need motivates a group of researchers to work on ontology-based knowledge archival (Witherell et al., 2007) and retrieval (Li et al., 2008; Liu et al., 2013), as mentioned in Section 1. However, from the decision support point of view, the retrieved information still cannot be effectively and efficiently reused for quality decision making because there is a lack of decision-making mechanism (construct) to utilize the information and generate a solution. In this paper, we create an ontology based on the cDSP construct and archive the information for decision making as templates that are executable and reusable. In terms of the ontology formalism, two popular paradigms exist: Web Ontology Language and Frame (Wang et al., 2006). We choose the Frame paradigm here because it is based on a closed-world assumption where everything is prohibited until it is permitted, which is suitable for modeling the highly constrained cDSP (Ming et al., 2015). In the following section, the development of the frame-based ontology for decisions is presented.

3. ONTOLOGY DEVELOPMENT FOR THE cDSP TEMPLATE

A complete frame-based ontology includes the concepts, relations among concepts, and consistency rules for keeping the populated instances consistent to what they are defined to be. In accordance with the cDSP construct, in this section we identify the key concepts and formally define them as classes, identify the relations among the concepts and formally define them as slots, and identify consistency rules and formally define them. Finally, the complete structure of the cDSP template ontology is presented.

3.1. Concept identification

Concepts of a domain are often described by a vocabulary of terms. Panchal et al. (2004) have identified a set of terms for modeling the cDSP template, which include *Constraint*, *Goal, Parameter, Variable, Preference, Analysis, Driver*, and *Response*. These terms are adopted and reused here by explicitly defining them as classes. Six additional classes, referred as *Problem, cDSPTemplate, Behavior, History, Function*, and *Quantity*, are introduced to capture the information that adds to the semantic richness and integrity of the ontology. The definitions of the classes are shown in Table 1.

3.2. Relation definition

The semantical relations among concepts are captured using slots in an ontology. Generally, there are two types of slots: data slots and object slots. Data slots are used to link concepts to nonobject data of which the types include *String, Float, Integer, Symbol*, and so on. For example, concept *Goal* may need a data slot to link itself to a target value with the type of *Float*. Objects slots are used to link concepts to object data of which the type is restricted to Class instance. For example, concept *Function* may need an object slot to link itself to a list of *Variable* instances. Based on the mathematical formulation of cDSP shown in Figure 1, the data slots and object slots of the cDSP ontology are defined as shown in Table 2 and Table 3, respectively.

3.3. Maintaining consistency

It is of critical importance for an ontology to restrict the populated instance in the manner as they are defined to be, that is, to keep consistency. As mentioned in Section 2.2, inconsistency is prone to occur when design consideration evolves, and the original cDSP template needs to be modified; for example, some parameters, goals, or constraints are added or removed (which is introduced in Section 4). Thus, detecting the inconsistency and informing designers to fix the inconsistency is very important. Rule-based reasoning is the method used for consistency checking in ontologies. In this paper, the rules for maintaining consistency in the cDSP template ontology are identified as shown in Table 4. Java Expert System Shell (*Jess*; Sandia National Laboratories, n.d.), a rule engine for the Java platform, is adopted as the reasoner. In order to be compatible with *Jess*, the rules are defined as the following (taking Rule 6 as an example):

```
(defrule MAIN::rule_6(object (is-a cDSPTemplate)
  (OBJECT ?y)) => (foreach ?x (slot-get ?y hasParameter)
  (if (neq (slot-get ?x lowerBound)(slot-get ?x upperBound))
  then(printout t WARNING_6crlf))))
```

It means that if any instance in the slot "hasParameter" has unequal values in terms of lower bound and upper bound, the reasoner will send a message about the inconsistency to the designer who is working on that cDSP template.

3.4. The complete structure of the ontology

The complete structure of the cDSP template ontology is shown in Figure 3. It is a network structure in which the gray nodes represent the classes and the black nodes represent the data. Different classes are linked by object slots that are represented by the dashed-line arrows, and the classes are linked to real data by data slots, which are solid-line arrows.

Table 1. *Classes of the cDSP template ontology*

Class	Definition
Problem	General information of a design problem such as the product to be designed and the functional requirements to be satisfied.
cDSPTemplate	Integrating all the template elements and the associated information, a formulation of a problem; a problem can be formulated as multiple cDSP templates.
Behavior	Behavioral information of the decision model represented by the template, such as sensitivity to the variation of parameters and converging tendency[a]
History	The evolution history of a template: (1) from which template this template is derived and (2) which template is developed based on this template.
Function	The general properties of a constraint and a goal
Quantity	The general properties of a parameter and a variable
Constraint	A function with a constant value that cannot be violated, subclass of function
Goal	A function with a target value that can be violated, subclass of function
Parameter	A quantity with a fixed value during the problem solving process
Variable	A quantity with a variable value during the problem solving process
Preference	Designers' preferences regarding the satisfaction of the system goals
Analysis	The information of inputs, outputs, and the associated analysis codes
Driver	The interface to problem solvers that run the analysis codes
Response	The actual response to a given specification of the template

[a]See the information captured by ternary plot and scatterplot in Section 5.

Table 2. *Data slots of the cDSP template ontology*

Slot Name	Definition	Type
name	Name of an instance	String
description	Description of an instance	String
quantityType	Type of a *Quantity* (“SystemVariable,” “SystemParameter,” or “DeviationVariable”)	Symbol
symbol	Symbol of a *Quantity*	String
value	Value of a *Quantity*	Float
unit	Unit of a *Quantity*	String
lowerBound	Lower bound of a *Quantity*	Float
upperbound	Upper bound of a *Quantity*	Float
expression	Expression of a *Function*	String
functionType	Type of a *Function* (“SystemConstraint” or “SystemGoal”)	Symbol
monotony	Monotony of a *Function* (“Minimize,” Maximize,” or “Force”)	Symbol
equality	Equality of a constraint *Function* (“≤,” “≥,” or “=”)	Symbol
linearity	Linearity of a Function (“linear” or “nonlinear”)	Symbol
target	Target of a *Function*	Float
weight	Weight of a *Preference* in an Archimedean form	Float
level	Level of a *Preference* in a preemptive form	Symbol
numberOfSamples	Number of samples for a Specific *Preference* in design experiments	Float
problemSolver	Problem solver that drives the template	Symbol
codeFileLocation	Storage path of the analysis code file	String
result	Result information of a *Response*	String
behavioralInfo	Behavioral information of a template	String
modification	Modification information of a template from its predecessor template	String

Hierarchical relations (e.g., the relation between classes *Constraint* and *Function*) among concepts are represented by arc arrows. The ontology modeling process is facilitated using the Protégé tool, developed by Stanford University (2013), which provides an environment for creating and editing ontologies as well as populating instances based on ontologies. The interface of Protégé and a cDSP template instance example are shown in a later section. Protégé also enables the development of plugins as extensions to the function of ontology. JessTab (Eriksson, 2008) is a plugin that attaches *Jess* to Protégé and is used for consistency checking in this paper. To facilitate the execution of the populated cDSP template instances, we also developed a plugin that link DSIDES to Protégé using Java function call for generating design solutions, which is not the main focus of this paper and thus not described in detail here.

Table 3. *Object slots of the cDSP template ontology*

Slot Name	Definition	Type
elementOf	Link a *Quantity* to a set of *Functions*	Instance
functionOf	Link *Function* to a set of *Quantities*	Instance
associatedGoal	Link a *Preference* to a *Goal*	Instance
input	Link an *Analysis* to a set of *Quantities* as input	Instance
output	Link an *Analysis* to a set of *Quantities* as output	Instance
hasVariable	Link a *cDSPTemplate* to a set of *Variables*	Instance
hasParameter	Link a *cDSPTemplate* to a set of *Parameters*	Instance
hasConstraint	Link a *cDSPTemplate* to a set of *Constraints*	Instance
hasGoal	Link a *cDSPTemplate* to a set of *Goals*	Instance
hasPreference	Link a *Goal* to a *Preference*	Instance
hasDriver	Link a *cDSPTemplate* to a *Driver*	Instance
hasAnalysis	Link a *cDSPTemplate* to an *Analysis*	Instance
hasResponse	Link a *cDSPTemplate* to a *Response*	Instance
hasTemplate	Link a *Problem* to a set of *cDSPTemplate*	Instance
applyTo	Link a *cDSPTemplate* to a *Problem*	Instance
hasHistory	Link a *cDSPTemplate* to a *History*	Instance
hasBehavior	Link a *cDSPTemplate* to a set of *Behaviors*	Instance
derivedFrom	Link a *cDSPTemplate* to another *cDSPTemplate*	Instance
evolveTo	Link a *cDSPTemplate* to another *cDSPTemplate*	Instance

Table 4. *Consistency rules of the cDSP template ontology*

Rule 1	Every instance in the slot “hasVariable” should be of type “SystemVariable.”
Rule 2	Every instance in the slot “hasParameter” should be of type “SystemParameter.”
Rule 3	Every instance in the slot “hasConstraint” should be of type “SystemConstraint.”
Rule 4	Every instance in the slot “hasGoal” should be of type “SystemGoal.”
Rule 5	Every instance in the slot “hasVariable” should comply to lowerBound $\leq$ value $\leq$ upperBound.
Rule 6	Every instance in the slot “hasParameter” should comply to lowerBound $=$ value $=$ upperBound.
Rule 7	Every instance in the slot “hasPreference” should comply to $0 \leq$ weight ≤ 1.
Rule 8	Every Variable instance of type “DeviationVariable” should comply to value ≥ 0.
Rule 9	All the instances in the slot “hasPreference” should comply to $\sum_{0}^{k} w_i = 1$.
Rule 10	Every Variable instance of type “DeviationVariable” should comply to $d_i^- \times d_i^+ = 0$.
Rule 11	All the Variable instance of type “DeviationVariable” should be included in the slot “output.”
Rule 12	All the Variable instance of type “SystemVariable” should be included in the slot “input.”

4. cDSP MODEL MODIFICATION TYPES

In many design (e.g., adaptive and variant) cases, as we mention in Section 1, a large part of information of previous decisions can be reused for making the current design decisions because the underlying design concepts are similar. In an ontological environment, this means that most of the populated instances can be reused in future designs. To successfully reuse these instances, partial modification is needed for the adapting to the design changes. The aim in this section is to determine the types of modification of the cDSP model that can be made in future design scenarios.

The cDSP construct (the “Given-Find-Satisfy-Minimize” structure as shown in Figure 1a) provides designers with

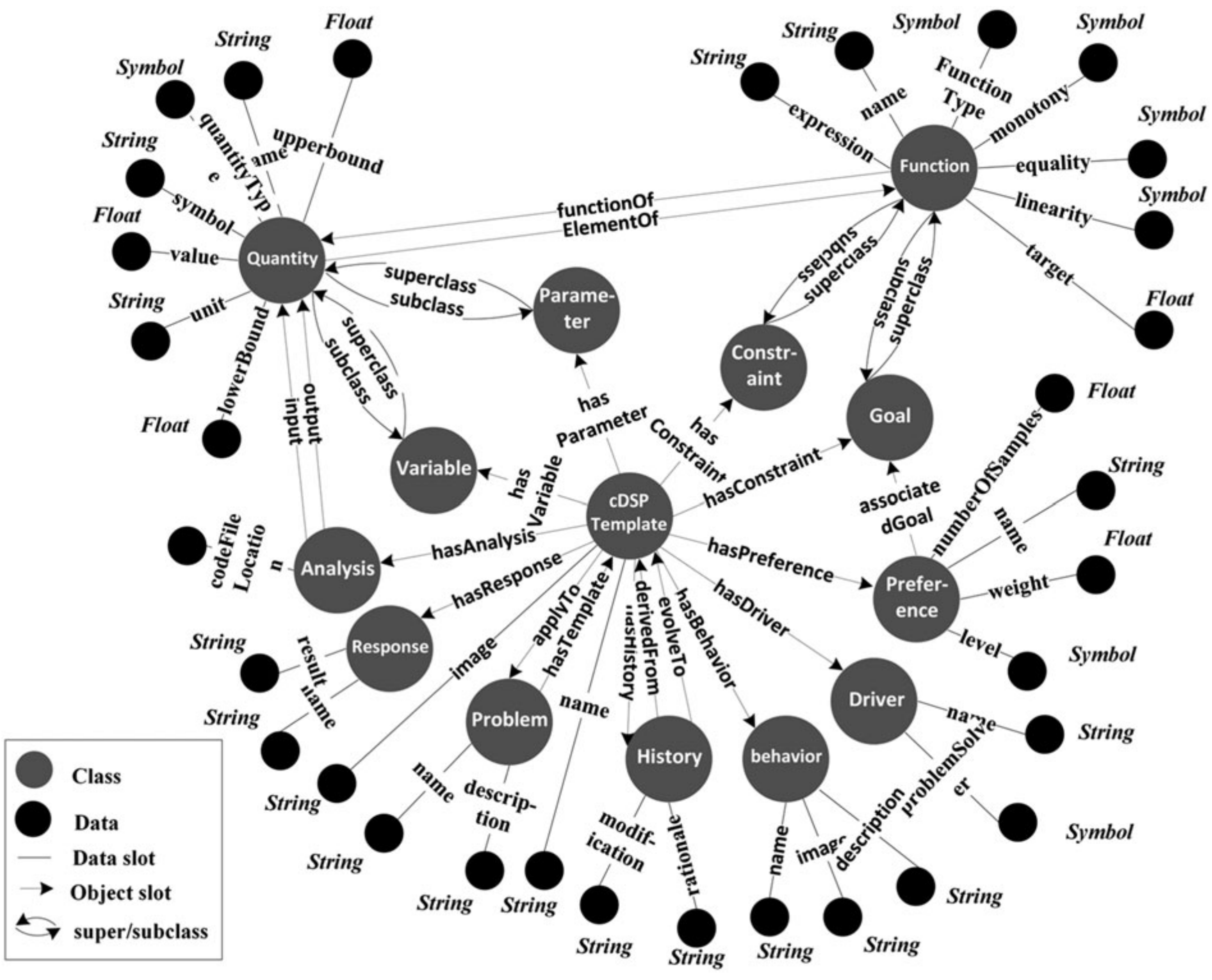


Fig. 3. Overview of the complete structure for the compromise decision support problem template ontology.

high flexibility to modify their decision model for specific problems. The main types of modification can be summarized as: modifying the number of design variables, modifying the values assigned to parameters, and modifying the number of goals and constraints. Detailed description of these three types of modification is given below, and the effects that each type of modification has on the design space are captured in Figure 4.

Type I: Modifying the number of design variables. This type of modification includes adding new variables and removing existing variables. The former can be done by introducing some existing parameters (with fixed value) to be variables; for example, in Figure 4, x_3 is previously a parameter and is introduced to be a variable, which results in that the design space transforms from a two-dimensional space to a three-dimensional space. While the latter can be done by assigning fixed value to existing variables, as shown in Figure 4, the value of the previous variable x_3 is fixed and the design space transforms from three dimensional to two dimensional.

Type II: Modifying the value assigned to parameters. This type of modification includes the following: modifying the values assigned to the weights of the goals (e.g., the value of w_i in the deviation function is modified in Fig. 4), which may have an influence on the achievements of the goals; modifying the values assigned to the coefficients of the equations (e.g., the value of coefficient a of goal G_1 and the value of coefficient D of constraint C_3 are modified in Fig. 4), which may result in the changes of curve shapes that frame the aspiration space and feasible design space; and modifying the values assigned to the bounds of the variables (e.g., the value of x_1^{lower} is modified in Fig. 4), which may lead to the shrinkage of the feasible design space.

Type III: Modifying the number of goals and constraints. This type of modification includes adding new goals as shown in Figure 4 where G_5 is added, removing existing goals (e.g., G_4 is removed from Fig. 4), adding new constraints (e.g., C_4 is added in Fig. 4), removing existing constraints (e.g., C_2 is removed from Fig. 4), and conversion between existing goals and constraints (e.g., constraint C_1 is converted to a goal in Fig. 4). Adding new goals and converting existing constraints to goals may result in shrinkage of the aspiration space. Removing existing goals and converting exiting goals to constraints may result in expansion of the aspiration space. Adding new constraints and converting existing goals to constraints may result in shrinkage of the feasible design space. Removing existing constraints and converting existing constraints to goals may result in expansion of the feasible design space.

In practical engineering design, different types of modifications may be needed simultaneously according to specific requirements. For example, in an adaptive design case, designers may need to set a previously fixed parameter to a variable, vary the bounds of some variables, and add new constraints to the problem, and thus, all the three cDSP model modification types may be needed simultaneously. In the ontological environment, the modification is facilitated by editing the template instances in terms of data slots and object slots.

5. EXAMPLE

In this section, a thin-walled pressure vessel design example is discussed to illustrate the utilization of the cDSP template ontology in terms of facilitating designers in making decisions by reusing previous design information. The example is an extension of the problem considered by Lewis and Mistree (1998), who used it to illustrate collaborations in decision-based design. An instance is created using the original design decision information in Section 5.1, and then three re-

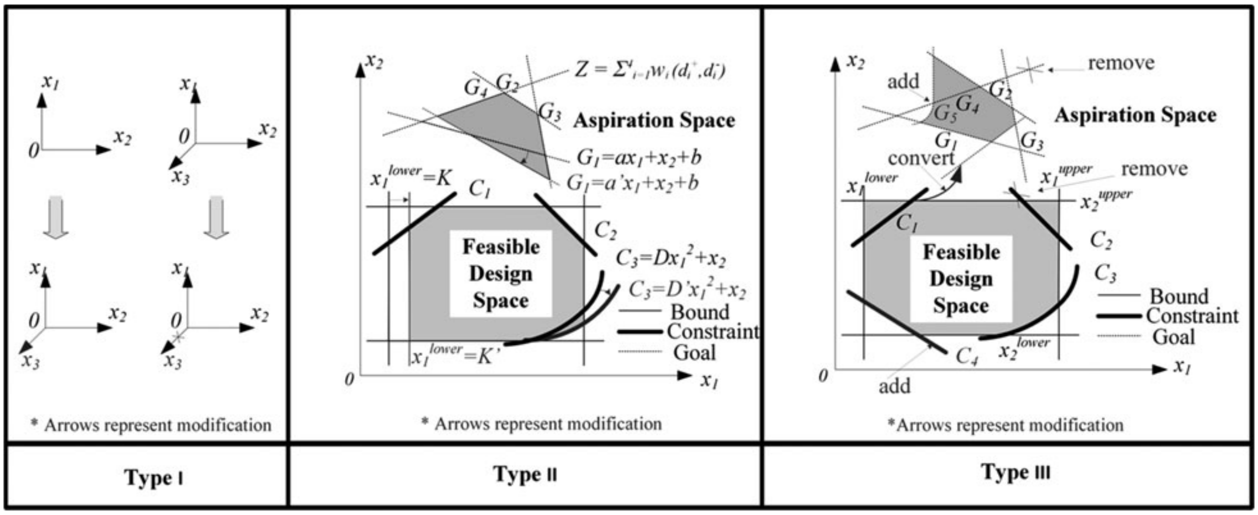


Fig. 4. Three types of the compromise decision support problem model modification.

design scenarios are presented in Section 5.2 for demonstrating the utilization of the ontological method. Finally, a summary and discussion are given in Section 5.3.

5.1. Populating an original design instance

The pressure vessel design variables are the radius R, the length L, and the thickness T, as shown in Figure 5. It is to withstand a specified internal pressure P, and the material is also specified. There are two objectives: to minimize the weight and to maximize the volume of the cylinder, both subject to stress and geometry constraints. The nomenclature for is example is presented in Table 5. The cDSP formulation is shown below:

1. **Given**
2. • Weight: $W(R,T,L)=\rho\left[\frac{4}{3}\pi(R+T)^3+\pi(R+T)^2L-\left(\frac{4}{3}\pi R^3+\pi R^2L\right)\right]$
3. • Volume: $V(R,L)=\frac{4}{3}\pi R^3+\pi R^2L$
4. **Find**
5. • System variables: thickness T, radius, R, and length, L
6. • Overachievement deviation variable associated with weight goal, d_w^+
7. • Underachievement deviation variable associated with volume goal, d_v^-
8. **Satisfy**
9. • Stress constraint: $c_1:\sigma_{\text{circ}}=\frac{PR}{T}\leq S_t$
10. • Geometric constraints: $c_2: 5T-R\leq 0$
11. c_3: $R+T-40\leq 0$
12. c_4: $L+2R+2T-150\leq 0$
13. • Bounds: $T_l\leq T\leq T_u$
14. $R_l\leq R\leq R_u$
15. $L_l\leq L\leq L_u$
16. • Weight goal: $W-d_w^+=W_{TV}$
17. • Volume goal: $V+d_v^-=V_{TV}$
18. **Minimize**
19. $W_{VOL}d_v^-+W_{WGT}d_w^+$

The specific input data for this problem is given in Table 6. Based on all the prepared information, a cDSP template instance for the design of the pressure vessel is instantiated in Protégé as shown in Figure 6. In Figure 6, the panel on the left-hand side is the class browser with all the classes listed, the panel in the middle is the instance browser listing all the instances associated with a selected class, and the panel on the right-hand side is the instance editor where a specific instance can be created and edited. In this case, all the slots (including data and object slots, e.g., "name," "hasParameter," "hasVariable") of the pressure vessel instance are created according to the structure defined in Section 4 and the data of the problem itself. When the instance is correctly created, the result is calculated using JAVA function calls that communicate with the problem solver DSIDES. The result is *(R, T, L) = (36, 4, 70), (Weight, Volume) = (39457 lb., 480385 in.^{3})* as shown in the front window in Figure 6.

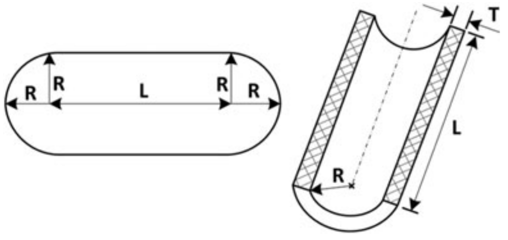


Fig. 5. Thin-walled pressure vessel with hemispherical ends.

Table 5. *Nomenclature for the pressure vessel example*

w	Weight of the pressure vessel (lbs)
V	Volume (in.^{3})
R	Radius (in.)
T	Thickness (in.)
L	Length (in.)
P	Pressure inside the cylinder (klb)
S_t	Allowable tensile strength of the cylinder material (klb)
ρ	Density of the cylinder material (lb/in.^{3})
σ_{circ}	Circumferential stress (lb/in.^{2})
TV	Target value for a goal
W_{VOL}	Weight associated with the weight goal
W_{WGT}	Weight associated with the volume goal

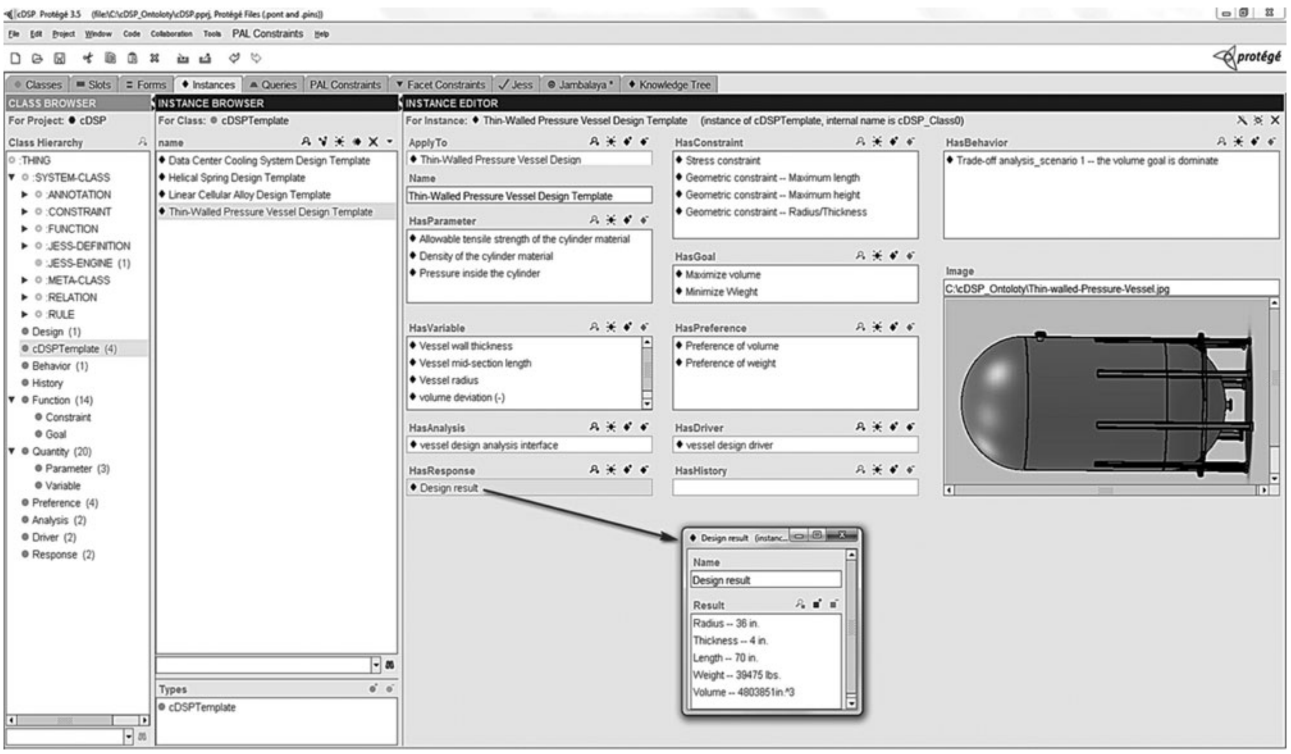
5.2. Facilitating decision making in redesign scenarios

5.2.1. Consistency checking

In this scenario, the original design variable R is assumed to be fixed to 20 in. due to manufacturing limitations, and thus modifications are required from the existing design to reflect the revised condition. The problem is closely related to

Table 6. *Pressure vessel input*

P	3.89 klb
S_t	35.0 klb
ρ	0.283 lb/in.^{3}
L_l	0.1 in.
L_u	140.0 in.
R_l	0.1 in.
R_u	36.0 in.
T_l	0.5 in.
T_u	6 in.
W_{TV}	0.1 lb
V_{TV}	775,000 in.^{3}
W_{VOL}	0.5
W_{WGT}	0.5


cDSP Protégé 3.5 (file:\C:\cDSP_Ontoloty\cDSP.pprj, Protégé Files (.pont and .pins))
File Edit Project Window Code Collaboration Tools PAL Constraints Help
protégé
Classes Slots Forms Instances Queries PAL Constraints Facet Constraints Jess Jambalaya * Knowledge Tree

CLASS BROWSER
For Project: ● cDSP
Class Hierarchy
- :THING
- :SYSTEM-CLASS
  - :ANNOTATION
  - :CONSTRAINT
  - :FUNCTION
  - :JESS-DEFINITION
  - :JESS-ENGINE (1)
  - :META-CLASS
  - :RELATION
  - :RULE
- Design (1)
- cDSPTemplate (4)
- Behavior (1)
- History
- Function (14)
  - Constraint
  - Goal
- Quantity (20)
  - Parameter (3)
  - Variable
- Preference (4)
- Analysis (2)
- Driver (2)
- Response (2)

INSTANCE BROWSER
For Class: ● cDSPTemplate
name
- Data Center Cooling System Design Template
- Helical Spring Design Template
- Linear Cellular Alloy Design Template
- Thin-Walled Pressure Vessel Design Template

Types
- cDSPTemplate

INSTANCE EDITOR
For Instance: ♦ Thin-Walled Pressure Vessel Design Template (instance of cDSPTemplate, internal name is cDSP_Class0)

ApplyTo
- Thin-Walled Pressure Vessel Design

Name
Thin-Walled Pressure Vessel Design Template

HasParameter
- Allowable tensile strength of the cylinder material
- Density of the cylinder material
- Pressure inside the cylinder

HasVariable
- Vessel wall thickness
- Vessel mid-section length
- Vessel radius
- volume deviation (-)

HasAnalysis
- vessel design analysis interface

HasResponse
- Design result

HasConstraint
- Stress constraint
- Geometric constraint -- Maximum length
- Geometric constraint -- Maximum height
- Geometric constraint -- Radius/Thickness

HasGoal
- Maximize volume
- Minimize Wieght

HasPreference
- Preference of volume
- Preference of weight

HasDriver
- vessel design driver

HasHistory

HasBehavior
- Trade-off analysis_scenario 1 -- the volume goal is dominate

Image
C:\cDSP_Ontoloty\Thin-walled-Pressure-Vessel.jpg

Design result (instanc...
Name
Design result
Result
Radius -- 36 in.
Thickness -- 4 in.
Length -- 70 in.
Weight -- 39475 lbs.
Volume -- 4803851in.^3

Fig. 6. Screenshot of the pressure vessel design compromise decision support problem template instance in Protégé ontology editor.

the suitable choice of L and ; it transforms from a three-dimensional $(L - R - T)$ problem to a two-dimensional $(L - T)$ problem as shown in Figure 7. In Figure 7, the previous feasible set is a solid vertical box in the positive orthant. One end of this vertical box is the polygon AJIH in the $L = 0$ plane, and the other end is formed by the inclined plane passing through the points marked EFG. When the new limitation is introduced, the previous feasible solution space is sliced by a plane vertical to the R axis at $R = 20$, and the cross section marked DBCK forms the new feasible solution space.

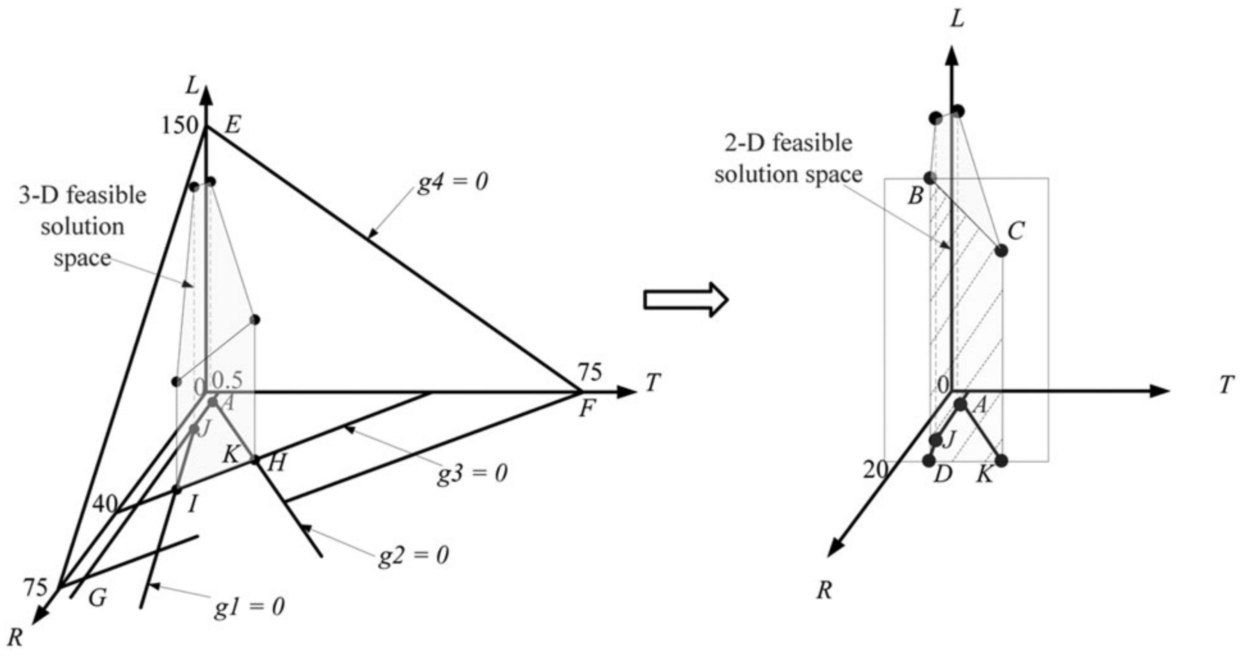


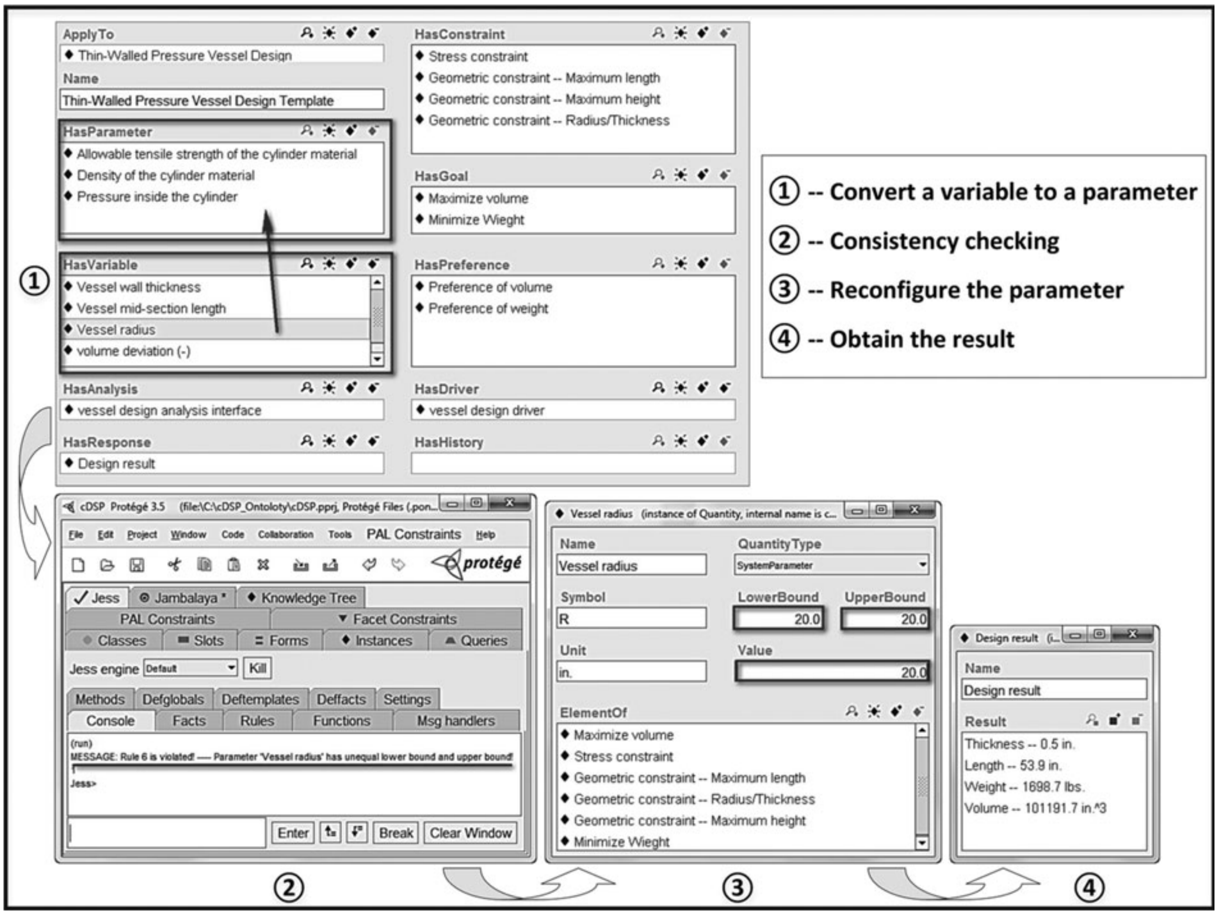
Fig. 7. Design space dimension change.

Facing the problem, designers need to properly adjust the original decision model to make a new decision. It is easy to recognize that the adjustment maps to modification Type I described in Section 4. In the ontology context, this adjustment is facilitated by a consistency mechanism shown in Figure 8. Specific steps of the process are as follows:

1. Convert a variable to a parameter: Move the previous variable "vessel radius" from the slot "hasVariable" to slot "hasParameter."
2. Check consistency: Check the consistency of the modified cDSP template instance using the JESS rule engine. A message pops up showing "Rule 6 is violated!—Parameter 'vessel radius' has unequal lower and upper bounds!"
3. Reconfigure the parameter: Reconfigure the parameter instance "vessel radius" as *lower bound = value = upper bound = 20 in.*
4. Obtain the result: Problem solver DSIDES will calculate and return the result according to the newly specified cDSP template configuration. The result is *(T, L) = (0.5, 53.9)*, *(Weight, Volume) = (1698.7 lb., 101191.7 in.^{3})*, which will be documented as a new instance.

5.2.2. *Trade-off analysis*

In this scenario, a new goal aimed at minimizing the cost is assumed to be considered in the design of the pressure vessel. The cost equation is derived from Sandgren (1990) as $C(R, L, T) = 0.6224RTL + 1.7881R^2T + 3.1611T^2L + 19.8621RT^2$, *and the target cost is set to \$0.1*. In the new problem adjustment needs to be made to the original decision model in in terms of adding a new goal and modifying the value assigned to the weights of the goals, which is the combination of modification Type II and III. We assume that this adjustment is well facilitated by the consistency-checking mechanism discussed in the previous section and focus on how trade-off analysis is facilitated under the ontological context in this section. Here trade-off analysis means analyzing the effects of different distribution of goal priorities on the actual performance of goals, which helps designers understand about the rationality of their decisions. A ternary plot (see Kulkarni et al., 2015; Shukla et al., 2015) has been used as a trade-off analysis tool for the three-goal cDSPs. The key idea is that the sample of the weighted value sets for the goals is generated, then based on the value sets the performance (represented as normalized deviation) of each goal is calculated, and finally, the relationship between the performance and the value sets is shown using a triangle with a color-bar as shown in Figure 9.



ApplyTo
- Thin-Walled Pressure Vessel Design

Name: Thin-Walled Pressure Vessel Design Template

HasParameter
- Allowable tensile strength of the cylinder material
- Density of the cylinder material
- Pressure inside the cylinder

HasVariable
- Vessel wall thickness
- Vessel mid-section length
- Vessel radius
- volume deviation (-)

HasAnalysis
- vessel design analysis interface

HasResponse
- Design result

HasConstraint
- Stress constraint
- Geometric constraint -- Maximum length
- Geometric constraint -- Maximum height
- Geometric constraint -- Radius/Thickness

HasGoal
- Maximize volume
- Minimize Wieght

HasPreference
- Preference of volume
- Preference of weight

HasDriver
- vessel design driver

HasHistory

① -- Convert a variable to a parameter

② -- Consistency checking

③ -- Reconfigure the parameter

④ -- Obtain the result

cDSP Protégé 3.5 (file:\C:\cDSP_Ontoloty\cDSP.pprj, Protégé Files (.pon...

```
(run)
MESSAGE: Rule 6 is violated! —— Parameter 'Vessel radius' has unequal lower bound and upper bound!
Jess>
```

Vessel radius (instance of Quantity, internal name is c...

Name	Vessel radius
QuantityType	SystemParameter
Symbol	R
LowerBound	20.0
UpperBound	20.0
Unit	in.
Value	20.0

ElementOf
- Maximize volume
- Stress constraint
- Geometric constraint -- Maximum length
- Geometric constraint -- Radius/Thickness
- Geometric constraint -- Maximum height
- Minimize Wieght

Design result

Name: Design result

Result
Thickness -- 0.5 in.
Length -- 53.9 in.
Weight -- 1698.7 lbs.
Volume -- 101191.7 in.^3

File Edit Project Window Code Collaboration Tools PAL Constraints Help
protégé
Jess
Jambalaya
Knowledge Tree
PAL Constraints
Facet Constraints
Classes
Slots
Forms
Instances
Queries
Jess engine Default
Kill
Methods Defglobals Deftemplates Deffacts Settings
Console Facts Rules Functions Msg handlers
Enter Break Clear Window
①
②
③
④

Fig. 8. Decision model adjustment facilitated by consistency checking.

The scales on the edges refer to the weights assigned to goals, the points within the triangle stand for different sets of weights [e.g., point *S* stands for set (0.2, 0.4, 0.4)], the color denotes the deviation of the associated points, and the corresponding values are indicated by the color bar. To perform trade-off analysis, designers need to prepare a large enough number of sample sets and run computing codes in DSIDES using each of the sample sets to generate the result, and then all the sample sets and the associated results are input to MATLAB to generate a ternary plot. This is usually a time-consuming process and calls for automation- or platform-related support in order to improve the efficiency.

In the ontology framework, the trade-off analysis includes three procedures: sample generation, communication to DSDIES for computing, and ternary plot generation. These three procedurals are all supported by Java function calls provided by Protégé. Sample generation plays a critical role for generating quality ternary plots. In the ontology, the sampling procedural is controlled by the data slot "numberOfSamples" (see Table 2), which defines the number of scales for the weight of each goal. At the computational level, it is implemented by a Java function that combines the scales to weight sets under the rule that the sum of scales in each set is 1, as shown in Figure 10a. The columns refer to goals, the numbers attached to the points in each column refer to scales of weights, and the colored lines that link the points stand for the weight sets. Here the slot "numberOfSamples" is set to three, and six sample sets are generated ("numberOfSamples" can be set to a larger number to generate more sets). Each sample set in Figure 10 is automatically sent to DSIDES to compute a result. Based on the sample sets and results, the ternary plot for the new goal "cost" is automatically generated and captured by the object slot "hasBehavior," as shown in Figure 10b. The blue area of the ternary plot in the figure means the preferred weight sets for minimizing the "cost" goal. Plots for the other two goals (the "weight" goal and the "volume" goal) are also captured for designers' reference.

5.2.3. Design space visualization

The assumptions used in the previous section are followed in this section to illustrate design space visualization. In the previous section, the ternary plot offers designers a means to analyze the trade-off among different goals by showing the relationship between the weight sets and the deviation of the goals. Trade-off analysis can lead designers to rational decisions, but the resulting designs may not necessarily be acceptable. For example, set point ($W_{VOL} = 0$, $W_{WGT} = 0$, $W_{COS} = 1$), which refers to preference for "cost" goal only, is located in the blue area in the ternary plot for the "cost" goal. Designers may choose it as the priority set for making the decision. However, the corresponding design ($R = 2.5,\ T = 0.5,\ L = 0.16$) may be unacceptable due to the small size and the obviously small volume. Therefore, informing designers about associated designs under certain priority sets is also very important for facilitating decision making. We implement this using scatterplots as shown in Figure 11. The three axes correspond to the three variables of the problem, points refer to sample design specifications generated in the feasible design space, and the color of a point denotes the deviation of the objective function (see Fig. 1) for the associated design. Points in blue mean lower deviation, and therefore are the superior designs desired by designers. By the plot, designers can have an intuitive view of the "outline" of superior designs based on where the blue points locate. For example, in Figure 11, superior designs under set point ($W_{VOL} = 0.33$, $W_{WGT} = 0.33$, $W_{COS} = 0.33$), which is an acceptable set in the ternary plot, are locating where the associated size seems large and more acceptable. Based on

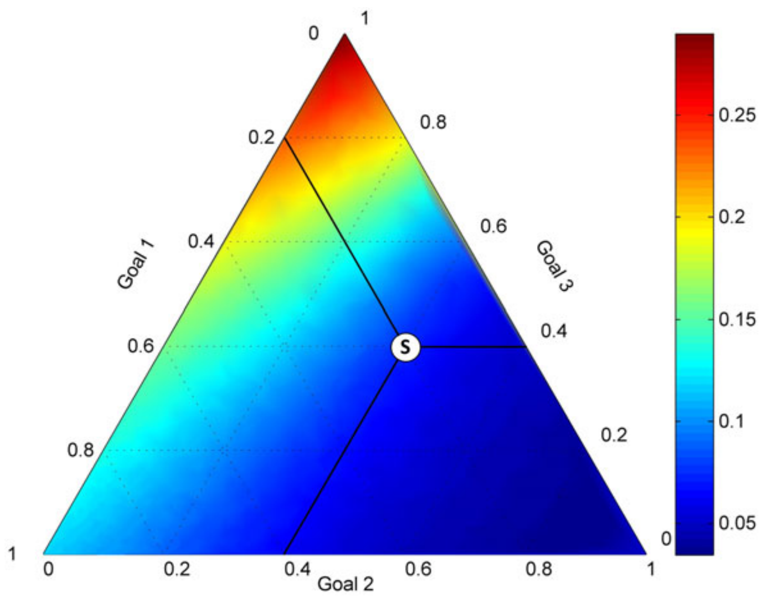


Fig. 9. Ternary plot for trade-off analysis.

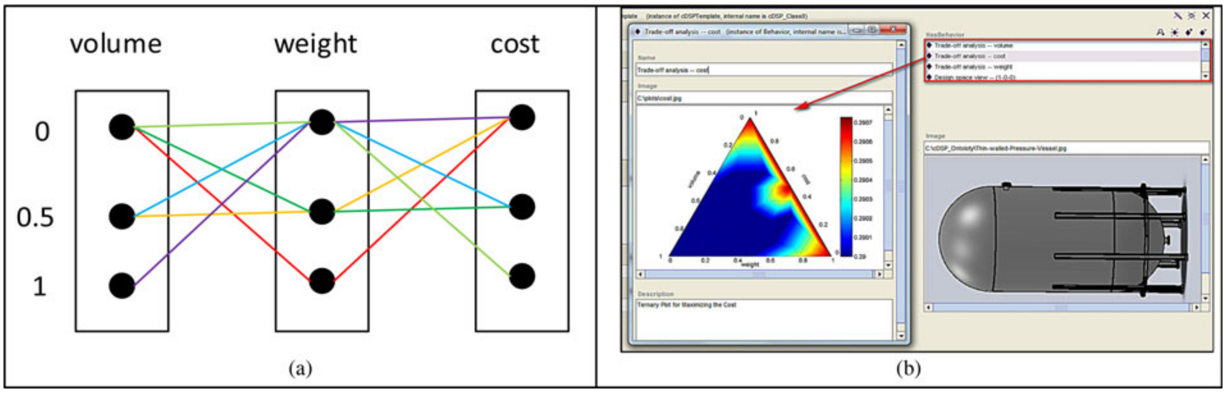

volume
weight
cost
0
0.5
1
(a)
(instance of cDSPTemplate, internal name is cDSP_Class0)
Trade-off analysis -- cost (instance of Behavior, internal name is...
Name
Trade-off analysis -- cost
Image
C:\plots\cost.jpg
Description
Ternary Plot for Maximizing the Cost
HasBehavior
Trade-off analysis -- volume
Trade-off analysis -- cost
Trade-off analysis -- weight
Design space view -- (1-0-0)
Image
C:\cDSP_Ontology\Thin-walled-Pressure-Vessel.jpg
(b)

Fig. 10. Trade-off analysis in the ontology.

visualization of the design space, designers will have more confidence in the decisions they make.

In the ontology context, design space visualization is implemented through Java function calls provided by Protégé. The function collects information in slots "hasVariable," "hasParameter," "hasConstraints," "hasGoals," and "hasPreference," and generates a scatter plot using samples in the feasible design space. The plot is finally captured as a behavior in the slot "hasBehavior" of a cDSP template instance for designers' reference, as the plot under priority set (0,0,1) shown in Figure 12. Plots under other priority sets such as (1,0,0), (0,1,0), and (.33,.33,.33) are also captured by the slot "hasBehavior," which are not shown in this paper.

5.3. Summary and discussion

Using the thin-walled pressure vessel example, we instantiated a cDSP template instance using the original design decision information, modified the number of design variables based on existing template instance for making new decisions, modified the number of goals and the value assigned to parameters (weights to goals) for making new decisions, performed trade-off analysis to facilitate decision making, and performed design space visualization to facilitate decision making. The reusability of the ontology-based cDSP template is verified by the reuse and modification of previously created instance, and the executability is verified by consistency checking, automatic generation of results, performing trade-off analysis, and design space visualization. From the results, we see that designers' decision making is facilitated in terms of

- editing the decision model (cDSP template instance) with consistency ensured,
- making rapid decisions with the help of automatic execution, and
- making rational decision supported by insightful information displaying (ternary and scatterplots).

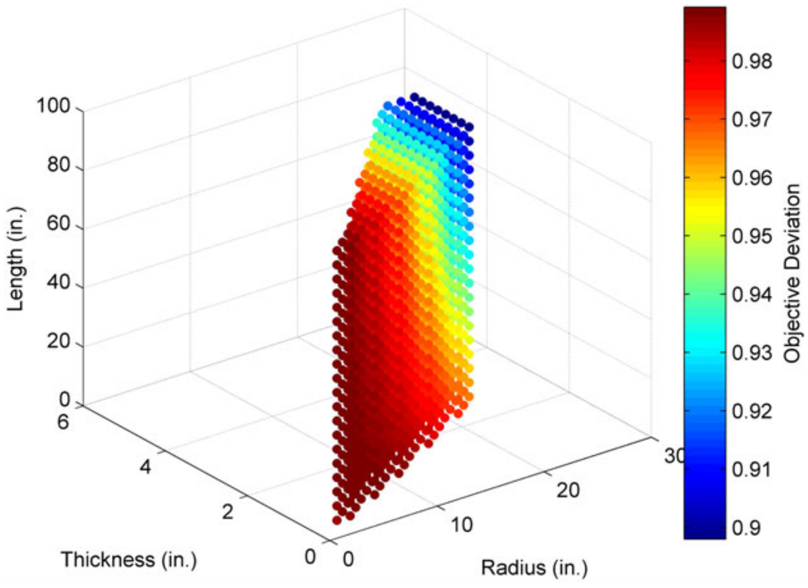
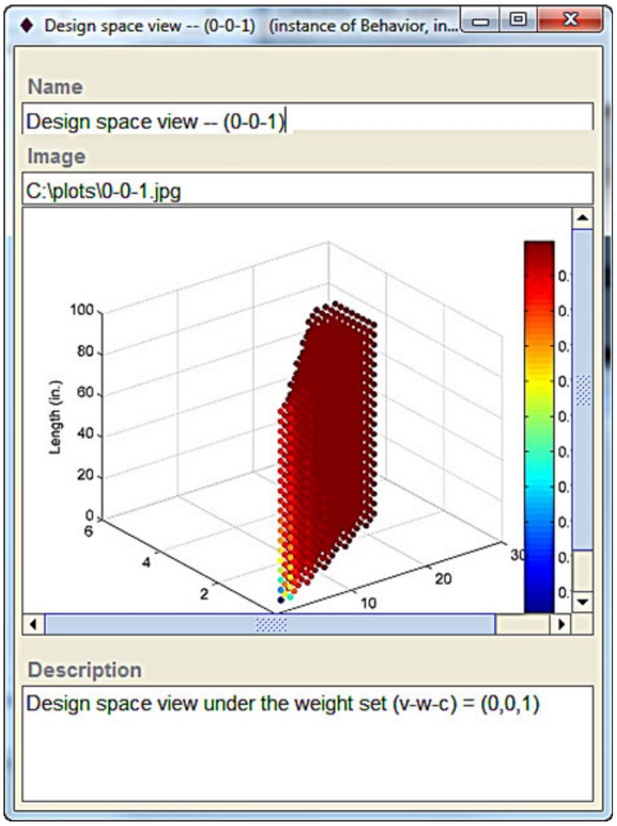


Fig. 11. Scatterplot.


◆ Design space view -- (0-0-1) (instance of Behavior, in...

Name

Design space view -- (0-0-1)

Image

C:\plots\0-0-1.jpg

Description

Design space view under the weight set (v-w-c) = (0,0,1)

Fig. 12. Design space visualization in the ontology.

6. CONCLUSION

Engineering design is increasingly recognized as a decision-making process, and the principal role of a designer is to make decisions. To augment this role of a decision maker, providing relevant decision support is crucial. In this paper, we represent the cDSP template using frame-based ontology that can provide decision support from two perspectives: the construct perspective and the information perspective. Regarding the construct, cDSP acts as the mathematical tool for formulating and making decisions among multiple conflicting goals under multiple constraints, which is frequently encountered in system designs. Regarding the information, the ontological template based method offers a means for archiving the critical decision-making information in an executable, reusable, and consistency-maintainable manner that facilitates rapid decision making. To facilitate designers' reuse of the populated cDSP template instances in future design decisions, we identified three types of modifications that can be made separately or in combination when design consideration evolves. Through a thin-walled pressure vessel redesign example, the efficacy of the approach proposed in this paper is demonstrated. It is shown that this approach supports designers in editing the decision model with consistency ensured, making rapid and rational decisions with automatic execution and insightful information visualization. The advantage of the ontology created in this paper over those semantic-centric ontologies lies in that it not only captures and documents the decision rationale that helps designers understand about how and why previous decisions are made but also facilitates designers reusing (by modification) previous information to effect new decisions when requirement changes.

In this paper, we have demonstrated the usefulness of the ontology-based decision template representation method based on the cDSP construct, and developed some decision support applications (graphical tools for information displaying) using the Java language. Due to the extensibility, computer interpretability, and inference ability possessed by ontology, the same idea can extend to the representation of other decision-making constructs such as the selection DSP and the utility-based approaches, the integration of other graphical information displaying tools, and the implementation in other languages such as C++. That is the direction our future work (the platform PDSIDES) is targeting.

ACKNOWLEDGMENTS

The authors gratefully acknowledge the China Scholarship Council for providing support (Grants 201406030014 and 201406035008) for carrying out this research. Janet K. Allen and Farrokh Mistree gratefully acknowledge NSF Grant CMMI-1440457.

REFERENCES

Barbau, R., Krima, S., Rachuri, S., Narayanan, A., Fiorentini, X., Foufou, S., & Sriram, R.D. (2012). OntoSTEP: enriching product model data using ontologies. *Computer-Aided Design 44(6)*, 575–590.

Chandrasegaran, S.K., Ramani, K., Sriram, R.D., Horvath, I., Bernard, A., Harik, R.F., & Gao, W. (2013). The evolution, challenges, and future of knowledge representation in product design systems. *Computer-Aided Design 45(2)*, 204–228.

Eriksson, H. (2008). *Jess Tab*. Accessed at http://www.jessrules.com/jess-wiki/view?JessTab on July 15, 2015.

Fenves, S.J., Foufou, S., Bock, C., & Sriram, R.D. (2008). CPM2: a core model for product data. *Journal of Computing and Information Science in Engineering 8(1)*, 014501.

Fernandez, M.G., Seepersad, C.C., Rosen, D.W., Allen, J.K., & Mistree, F. (2005). Decision support in concurrent engineering—the utility-based selection decision support problem. *Concurrent Engineering—Research and Applications 13(1)*, 13–27.

Gruber, T.R. (1993). A translation approach to portable ontology specifications. *Knowledge Acquisition 5(2)*, 199–220.

Gu, X.Y., Renaud, J.E., Ashe, L.M., Batill, S.M., Budhiraja, A.S., & Krajewski, L.J. (2002). Decision-based collaborative optimization. *Journal of Mechanical Design 124(1)*, 1–13.

Hazelrigg, G.A. (1998). A framework for decision-based engineering design. *Journal of Mechanical Design 120(4)*, 653–658.

Kulkarni, N., Gautham, B., Zagade, P., Panchal, J., Allen, J.K., & Mistree, F. (2015). Exploring the geometry and material space in gear design. *Engineering Optimization 47(4)*, 561–577.

Kulok, M., & Lewis, K. (2007). A method to ensure preference consistency in multi-attribute selection decisions. *Journal of Mechanical Design 129(10)*, 1002–1011.

Lee, J.H., Fenves, S.J., Bock, C., Suh, H.W., Rachuri, S., Fiorentini, X., & Sriram, R.D. (2012). A semantic product modeling framework and its application to behavior evaluation. *IEEE Transactions on Automation Science and Engineering 9(1)*, 110–123.

Lewis, K., & Mistree, F. (1995). Designing top-level aircraft specifications: a decision-based approach to a multiobjective, highly constrained problem. *Proc. 36th AIAA/ASME/ ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conf.*, Paper No. AIAA-95-1431-CP, pp. 2393–2405. New Orleans, LA, April 10–13.

Lewis, K., & Mistree, F. (1998). Collaborative, sequential, and isolated decisions in design. *Journal of Mechanical Design 120(4)*, 643–652.

Lewis, K.E., Chen, W., & Schmidt, L.C. (2006). *Decision Making in Engineering Design*. New York: ASME Press.

Li, Z., Raskin, V., & Ramani, K. (2008). Developing engineering ontology for information retrieval. *Journal of Computing and Information Science in Engineering 8(1)*, 011003.

Liu, Y., Lim, S.C.J., & Lee, W.B. (2013). Product family design through ontology-based faceted component analysis, selection, and optimization. *Journal of Mechanical Design 135(8)*, 081007.

Lu, W.L., Qin, Y.C., Liu, X.J., Huang, M.F., Zhou, L.P., & Jiang, X.Q. (2015). Enriching the semantics of variational geometric constraint data with ontology. *Computer-Aided Design 63*, 72–85.

Ming, Z., Yan, Y., Wang, G., Panchal, J.H., Goh, C.H., Allen, J.K., & Mistree, F. (2015). Ontology-based executable design decision template representation and reuse. *Proc. ASME Computers and Information in Engineering Conf.*, Paper No. DETC2015-46272, Boston, August 2–5.

Mistree, F., Hughes, O.F., & Bras, B.A. (1993). The compromise decision support problem and the adaptive linear programming algorithm. In *Structural Optimization: Status and Promise* (Kamat, M.P., Ed.), pp. 247–286. Washington, DC: AIAA.

Mistree, F., Smith, W., Bras, B., Allen, J., & Muster, D. (1990). Decision-based design: a contemporary paradigm for ship design. *Transactions, Society of Naval Architects and Marine Engineers 98*, 565–597.

Muster, D., & Mistree, F. (1988). The decision support problem technique in engineering design. *International Journal of Applied Engineering Education 4(1)*, 23–33.

Pahl, G., Pahl, G., Wallace, K., & Blessing, L.T.M. (2007). *Engineering Design: A Systematic Approach*. London: Springer.

Panchal, J.H., Fernández, M.G., Paredis, C.J.J., & Mistree, F. (2004). Reusable design processes via modular, executable, decision-centric templates. *Proc. AIAA/ISSMO Multidisciplinary Analysis and Optimization Conf.*, Paper No. AIAA-2-4-4601, Albany, NY.

Reddy, R., Smith, W., Mistree, F., Bras, B., Chen, W., Malhotra, A., Badhrinath, K., Lautenschlager, U., Pakala, R., & Vadde, S. (1996). *DSIDES User Manual*. Atlanta, GA: Georgia Institute of Technology, Woodruff School of Mechanical Engineering, Systems Realization Laboratory.

Resende, C.B., Heckmann, C.G., & Michalek, J.J. (2012). Robust design for profit maximization with aversion to downside risk from parametric uncertainty in consumer choice models. *Journal of Mechanical Design 134(10)*, 100901-1–100901-12.

Rockwell, J., Grosse, I.R., Krishnamurty, S., & Wileden, J.C. (2009). A decision support ontology for collaborative decision making in engineering design. *Proc. Int Symp. Collaborative Technologies and Systems*, Baltimore, MD, May 18–22.

Rockwell, J.A., Witherell, P., Fernandes, R., Grosse, I.R., Krishnamurty, S., & Wileden, J.C. (2008). A Web-based environment for documentation and sharing of engineering design knowledge. *Proc. 28th Computers and Information in Engineering Conf.*, Brooklyn, NY, August 3–6.

Sandgren, E. (1990). Nonlinear integer and discrete programming in mechanical design optimization. *Journal of Mechanical Design 112(2)*, 223–229.

Sandia National Laboratories. (n.d.). *Jess®, the Rule Engine for the Java™ Platform*. Accessed at http://herzberg.ca.sandia.gov/ on July 15, 2015.

Seepersad, C.C., Allen, J.K., McDowell, D.L., & Mistree, F. (2008). Multifunctional topology design of cellular material structures. *Journal of Mechanical Design 130(3)*, 031404.

Shukla, R., Kulkarni, N., Gautham, B., Singh, A., Mistree, F., Allen, J., & Panchal, J.H. (2015). Design exploration of engineered materials, products, and associated manufacturing processes. *Journal of the Minerals, Metals & Materials Society 67(1)*, 94–107.

Simon, H.A. (1976). *Administrative Behavior: A Study of Decision-Making Processes in Administrative Organization*. New York: Free Press.

Sivaloganathan, S., & Shahin, T. (1999). Design reuse: an overview. *Proc. Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture 213(7)*, 641–654.

Stanford University. (2013). *Protégé 3.5 Release*. Accessed at http://protege-wiki.stanford.edu/wiki/Protege_3.5_Release_Notes on July 15, 2015.

Thurston, D.L. (1991). A formal method for subjective design evaluation with multiple attributes. *Research in Engineering Design 3(2)*, 105–122.

Vadde, S., Allen, J.K., & Mistree, F. (1994). The Bayesian compromise decision-support problem for multilevel design involving uncertainty. *Journal of Mechanical Design 116(2)*, 388–395.

Wang, H., Noy, N., Rector, A., Musen, M., Redmond, T., Rubin, D., Tu, S., Tudorache, T., Drummond, N., & Horridge, M. (2006). *Frames and OWL Side by Side*, p. 54. Available at http://protégé.stanford.edu/conference/2006/submissions/slides/7.2wng_protege2006.pdf

Wassenaar, H.J., Chen, W., Cheng, J., & Sudjianto, A. (2005). Enhancing discrete choice demand modeling for decision-based design. *Journal of Mechanical Design 127(4)*, 514–523.

Williams, C.B., Allen, J.K., Rosen, D.W., & Mistree, F. (2007). Designing platforms for customizable products and processes in markets of non-uniform demand. *Concurrent Engineering—Research and Applications 15(2)*, 201–216.

Witherell, P., Krishnamurty, S., & Grosse, I.R. (2007). Ontologies for supporting engineering design optimization. *Journal of Computing and Information Science in Engineering 7(2)*, 141–150.

Yang, D., Dong, M., & Miao, R. (2008). Development of a product configuration system with an ontology-based approach. *Computer-Aided Design 40(8)*, 863–878.

Zhenjun Ming is a PhD candidate in the School of Mechanical Engineering at Beijing Institute of Technology (BIT). He is currently a Visiting Scholar at the School of Aerospace and Mechanical Engineering at the University of Oklahoma on a scholarship provided by the China Scholarship Council. Zhenjun earned his Bachelor degree in industrial engineering from BIT in 2011. His PhD dissertation is about designing a knowledge-based platform for decision support in the design of engineering systems. He has published one journal paper and two conference papers and is a winner of the 2015 NSF/ASME Design Essay Competition.

Yan Yan is a Professor in the School of Mechanical Engineering at BIT. She received her Bachelor and PhD from BIT. She is a member of the Science and Technology Committee in the State Administration of Science, Technology and Industry for National Defense (China). Professor Yan's research interests lie in knowledge-based engineering, information modeling in design, and manufacturing. She has coauthored two textbooks and over 60 peer-reviewed papers.

Guoxin Wang is an Associate Professor and the Vice Director of the Industrial Engineering Institute at BIT. He is a Senior Fellow of the Chinese Mechanical Engineering Society. Dr. Wang directs and has accomplished 15 projects from the National Nature Science Foundation of China, the National High-Tech. R&D Program, the National Defense Basic Scientific Research Foundation, and the National and International Enterprise Research Foundation. He has published over 40 papers and a book. His research interests include system modeling and simulation, knowledge-based engineering, and reconfigurable manufacturing systems.

Jitesh Panchal is an Associate Professor in the School of Mechanical Engineering at Purdue University. He received his BTech (2000) from the Indian Institute of Technology Guwahati and his MS and PhD in mechanical engineering from the Georgia Institute of Technology. Dr. Panchal's research interests are in computational design of complex engineering systems with focus on three areas: concurrent products and materials design, collective systems innovation, and cyberphysical systems for design and manufacturing. He is a coauthor of *Integrated Design of Multiscale, Multifunctional Materials and Products*. He is a recipient of the CAREER award from the National Science Foundation, the Young Engineer Award, two best paper awards from the ASME CIE division, and a university silver medal from the Indian Institute of Technology Guwahati.

Chung-Hyun Goh is a member of the Mechanical Engineering Faculty of the University of Texas at Tyler. Prior to joining the University of Texas at Tyler, he worked in the Systems Realization Laboratory at the University of Oklahoma from 2012 to 2015. He worked for the Korean government after he received his PhD at Georgia Institute of Technology. Dr. Goh is a member of ASME, ASEE, TMS, and the board of directors in the materials and fracture group in the Korean Society of Mechanical Engineers. He has published a 24 peer-reviewed journal and proceedings papers as well as a coauthored textbook.

Janet K. Allen holds the John and Mary Moore Chair and is Professor of industrial and systems engineering at the University of Oklahoma. Janet's research is in engineering design and especially the management of uncertainty when making design decisions. Her interest in managing uncertainty extends to robust design, uncertainty quantification, information economics, and statistical and computational methods to facilitate engineering design. She has an ongoing interest in simulation and modeling in support of design decision making with a particular interest in the design of complex systems. Dr. Allen is a Fellow of the ASME, a Senior Member of the AIAA, and an Honorary Member of the Mechanical Engineering Honor Society Pi Tau Sigma.

Farrokh Mistree holds the L. A. Comp Chair in the School of Aerospace and Mechanical Engineering at the University of Oklahoma. Farrokh's current research focus is model-based realization of complex systems by managing uncertainty and complexity. The key question he is investigating is the principles underlying rapid and robust concept exploration when the analysis models are incomplete and possibly inaccurate. Dr. Mistree has coauthored two textbooks, one monograph, and more than 400 technical papers dealing with the design of material, mechanical, thermal, and structural systems; ships; and aircraft. Farrokh is a Fellow of ASME and an Associate Fellow of AIAA.