Zhenjun Ming
Beijing Institute of Technology,
Beijing 100081, China

Anand Balu Nellippallil
School of Aerospace and Mechanical Engineering,
University of Oklahoma,
Norman 73019, OK

Yan Yan
Beijing Institute of Technology,
Beijing 100081, China

Guoxin Wang[1]
School of Mechanical Engineering,
Institute for Industrial Engineering,
Beijing Institute of Technology,
No. 5 Zhongguancun South Street,
Haidian District,
Beijing 100081, China
e-mail: wangguoxin@bit.edu.cn

Chung Hyun Goh
Department of Mechanical Engineering,
The University of Texas at Tyler,
Tyler 75799, TX

Janet K. Allen
School of Industrial and Systems Engineering,
University of Oklahoma,
Norman 73019, OK

Farrokh Mistree
School of Aerospace and Mechanical Engineering,
University of Oklahoma,
Norman 73019, OK

PDSIDES—A Knowledge-Based Platform for Decision Support in the Design of Engineering Systems

We hypothesize that by providing decision support for designers we can speed up the design process and facilitate the creation of quality cost-effective designs. One of the challenges in providing design decision support is that the decision workflows embody various degrees of complexity due to the inherent complexity embodied in engineering systems. To tackle this, we propose a knowledge-based Platform for Decision Support in the Design of Engineering Systems (PDSIDES). PDSIDES is built on our earlier works that are anchored in modeling decision-related knowledge with templates using ontologies to facilitate execution and reuse. In this paper, we extend the ontological decision templates to a computational platform that provides knowledge-based decision support for three types of users, namely, template creators, template editors, and template implementers, in original design, adaptive design, and variant design, respectively. The efficacy of PDSIDES is demonstrated using a hot rod rolling system (HRRS) design example. [DOI: 10.1115/1.4040461]

Keywords: platform, engineering design, knowledge, decision making, decision support problem, ontology

1 Frame of Reference

Design of engineering systems is increasingly recognized as a decision-making process [1–4]. We believe that the principal role of a human designer is to make decisions. Providing decision support is of critical importance for augmenting this role, by speeding up the design process and generating quality designs. One of the challenges in providing decision support in the design of engineering systems, especially complex systems that are, by definition, made up of inter-related subsystems [5], arises because of the complexity embodied in the decision workflows that embody multiple coupled decisions networked in various degrees of complexity. The networked decision workflows may include different types of decisions, e.g., selection of design alternatives and improvement of an alternative considering multiple goals. The decisions are coupled together due to the dependency existing among systems and subsystems. The different types of decisions and their associated dependencies in the decision workflows make it difficult to provide appropriate decision support.

Decision-making is a knowledge-intensive process, with knowledge playing a significant role in speeding up and affecting decisions. Design knowledge representation for conceptual and detailed design has been area of interest in knowledge-based design and engineering for many decades. However, most of the works on knowledge representation deal with design in general (CAD oriented), not in the context of supporting decisions. For example, Shah and Mäntylä [6] introduced the parametric and feature-based methods, which have specific data structures and algorithms embedded to facilitate rapid and reusable three-dimensional geometric model generation. While, the parametric, feature-based procedure knowledge representations introduced in Ref. [6] cannot be (at least not directly be) applied to represent the human decision-making processes in design. Coyne et al. [7] propose a prototype-centric framework for the development of knowledge-based design systems. In their framework, prototypes can be generated, refined, and adapted to create novel designs. However, the design decision-making processes are not addressed in their work. Finger and Dixon [8] reviewed many descriptive, prescriptive, and computer-based models of design processes in the late 1980s with the aim to create intelligent CAD expert systems. Human decision-making process is not emphasized and well analyzed but just lightly mentioned as "concept selection" with no detailed information in their review. Verhagen et al. [9] analyzed a total of 50 research contributions in the area of knowledge-based engineering (KBE), pointed out the challenges, and suggested some future research opportunities in the field. However, the goal of the total 50 KBE research contributions, as stated by the authors, is to automate the product design and development process, but not support designers making better decisions. Similarly, Rocca [10] provided an extensive review of KBE from a language-based technological perspective, the aim being to

[1]Corresponding author.

Contributed by the Computers and Information Division of ASME for publication in the Journal of Computing and Information Science in Engineering. Manuscript received June 10, 2017; final manuscript received May 25, 2018; published online July 3, 2018. Assoc. Editor: Yan Wang.

understand what the technological fundamentals of KBE are and how it can be used to automate large portions of the design process. One thing in this paper that is related to decision-making is that KBE is used to develop multimodel generators in MDO, but the compromise decision (i.e., the tradeoff) among multidisciplinary models is not discussed. Jakiela and Papalambros [11] from University of Michigan introduced a prototype "intelligent" CAD system, in which decision-making process during the conceptual design is programmed using production rules to automatically generate three-dimensional models. While this system provides knowledge-based automatic decision-making in design, the limitation is that it only accounts for geometrical modeling. Sapuan [12] presented a knowledge-based system for material selection. However, the decision process and associated knowledge representation language are domain specific and thus not reusable and extensible.

Despite the fact that many knowledge-based systems have been developed to support engineering design, the challenge of supporting the decision workflow in the design of complex engineering systems is not yet well addressed, mainly, for the following reasons:

(1) Lack of both reusable and executable decision knowledge representation schemes. Knowledge reusability is critical for adaptive and variant design wherein only a small portion of the original decision workflows needs to change while the rest remains the same and can be reused. Some authors have proposed to represent decision knowledge as ontologies (e.g., Ref. [13]), but they mainly focus on capturing the semantic information of design decisions while failing to represent the execution process information, which is necessary for effecting new decisions, especially in a computational environment whereby some degree of automation is realized.
(2) Lack of a classification of users for decision support. The needs of designers for decision support vary according to how much novelty is involved in the design and how much knowledge they have about the design process. For example, an expert has much knowledge about design and can perform the decision-making process independently; thus, the support this designer needs from the computer system is very different from a novice designer who only has the basic knowledge about design and needs to get most of the knowledge from the system. Very few of knowledge-based systems recognized this difference and provided appropriate decision support.

To address the aforementioned needs, we propose a knowledge-based Platform for Decision Support in the Design of Engineering Systems (PDSIDES). The new contributions embodied in this paper are summarized as follows:

- We integrate the decision-related knowledge that is modeled as decision support problem (DSP) templates and represented using ontologies in our earlier works [14–16] into a computational platform (PDSIDES) to facilitate extensive reuse and execution. Our earlier works are about information modeling; this paper is about platformization.
- We define three types of users, namely, *template creators, template editors*, and *template implementers*, who perform *original design, adaptive design,* and *variant design* respectively in PDSIDES.
- We provide customized decision support for human *template creators, template editors*, and *template implementers* during their design of engineering systems in PDSIDES.

Overview. The paper is organized as follows: In Sec. 2, we introduce the primary constructs used in PDSIDES by referencing our previous work to provide the context. In Sec. 3, we describe the design of PDSIDES, including platform overview, users and working scenarios, knowledge-based decision support. The technical implementation of PDSIDES is introduced in Sec. 4. In Sec. 5, we illustrate the efficacy of PDSIDES using a hot rod rolling system (HRRS) design example. In Sec. 6, we offer some closing remarks and enumerate future research opportunities.

2 Primary Constructs

2.1 Decision Support Problem. Platform for decision support in the design of engineering systems is designed from a decision-based design (DBD) perspective, wherein decisions serve as markers to identify the progression of a design from initiation to implementation to termination [17]. We recognize that the implementation of DBD can take many forms, such as Ref. [18]; our implementation being the DSP technique [19]. Key to the DSP Technique is the notion that there are two types of decisions, namely, selection and compromise, and that a complex design can be represented by modeling a workflow of compromise and selection decisions. The selection DSP (sDSP) [20] involves making a choice among a number of alternatives taking into account a number of measures of merit or attributes, while the compromise DSP (cDSP) [21] involves the improvement of an alternative through modification by making a trade-off among multiple design objectives. The sDSP and the cDSP are two fundamental decision-making constructs in PDSIDES.

The design of complex systems may require the formulation and resolution of a series of coupled decisions, in which case the hierarchical DSP construct based on the sDSP and the cDSP is used as the model to support hierarchical decision-making, for the detailed mathematical model see Refs. [22] and [23]. Key to the hierarchical DSP is the combination of all the DSPs (including sDSPs and cDSPs) simultaneously by reformulating the DSPs into a single cDSP. Hierarchical DSPs are generally multiobjective, nonlinear, mixed discrete-continuous problems. A tailored computational system known as DSIDES [24] is integrated into PDSIDES to solve such problems.

2.2 Decision Template. One of our primary goals in designing PDSIDES is that designers can rapidly create decision models for the specific design problems they have by using the DSP constructs, and making decisions, and finally the produced decision knowledge can be stored and reused by other users for similar designs. To achieve this goal, the DSPs are represented as computational decision templates in PDSIDES. Decision templates, originally proposed by Panchal et al. [25], make it possible to model the compromise DSP so that the template is reusable and computer interpretable. We extend the idea to model the selection DSP and hierarchical DSP as templates in our earlier work [14,15]. Key to the computational DSP templates is the modularization of the DSP constructs and the separation of declarative and procedural knowledge, which allows both to be reused across problems.

In PDSIDES, all the DSP template modules including the sDSP template modules such as alternatives and attributes, the cDSP template modules such as constraints, variables, etc. and the hierarchical DSP template modules are managed in the module repository, as shown in Fig. 1. It is noted that the sDSP template and the cDSP template are also defined as a particular type of module since they comprise the key "building blocks" of a decision hierarchy and can be linked together using the *interface* and *process* modules; see Ref. [14] for details. Template modules represent the declarative knowledge in PDSIDES, which embodies problem specific information and can be reused in the instantiation of DSP templates (the wired "boards") to support a designer making selection, compromise, and hierarchical decisions. The procedural knowledge denotes how specific information is processed to reach a decision, and is archived in the templates (the printed "wiring" between different modules) for the execution of decisions. The separation of these two types of knowledge makes it fairly easy for designers to reconfigure existing templates, which is critically important in adaptive and variant designs where design

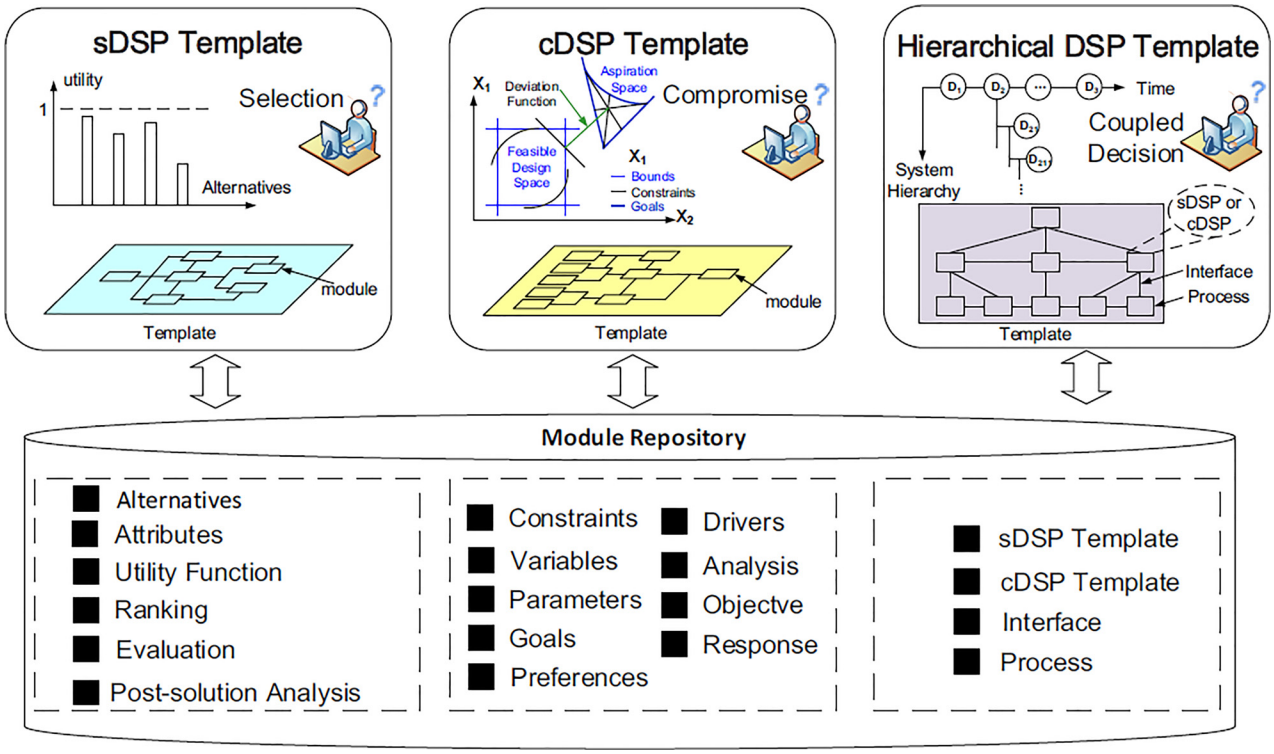

Fig. 1 DSP templates and their associated modules

consideration changes and the original decision model needs to be modified. Template modification is discussed in Sec. 3.

2.3 Ontology. In order to store and reuse the knowledge archived in the DSP templates in a computational environment, there needs to be a formal representation scheme. Ontologies are defined by Gruber [26] as explicit formal specifications of terms and relations among them are increasingly used for knowledge modeling in engineering design, such as Refs. [13] and [27]. In PDSIDES, ontology is used to formally represent the knowledge (including declarative and procedural knowledge) archived in the DSP templates. Key elements of an ontology are terms and relations. Terms represent the components of a domain, which refers to the modules of the DSP templates in this paper. According to Li et al. [28], the grain sizes of terms in an ontology are determined by the consideration of the need for an application or computational complexity. In PDSIDES, to comprehensively capture the semantics of the DSPs, we introduce some additional terms, such as *coefficient, and utility calculation* to the sDSP template ontology [15] and *quantity, function* to the cDSP template ontology [16]. Relations in an ontology represent the connections of a term to other terms (e.g., the connecting a *goal* to a *variable* using relation *function-of*), that provide the context of the terms and make them easy-to-comprehend and facilitate communication. The terms and relations in an ontology capture the declarative knowledge, which is domain-specific, while some attached elements such as rules, axioms, or Java function calls capture the procedural knowledge, which is domain-independent. There are two popular paradigms for ontology formalism, namely, web ontology language and frame [29]. The frame paradigm is chosen because it is based on a closed-world assumption wherein everything is prohibited until it is permitted, which is suitable for modeling the highly constrained DSPs. In frame-based ontology, terms are defined as `classes` and relations are defined as `slots`. With `classes` and `slots`, ontologies in PDSIDES are defined as shown in Fig. 2. On the left-hand side and right-hand side are the cDSP and the sDSP ontologies respectively, which are integrated by the hierarchical DSP ontology in the middle for capturing knowledge related to hierarchical decision workflows. For detailed specification of the `classes` and `slots`, see our earlier works [14–16]:

The advantages of the use of ontology in PDSIDES are summarized as follows:

- *Facilitate knowledge sharing*. This is embodied in two aspects, namely, knowledge sharing among different users in PDSIDES and knowledge sharing between PDSIDES and other product lifecycle management platforms. The DSP ontologies represent the common language used for design decision-making in PDSIDES, and thus users from different design disciplines (e.g., thermal, structural, dynamic, etc.) can easily understand, communicate knowledge such as *variables, goals, and constraints*, with each other. Meanwhile, the explicit, formal specifications of the terms of the DSP ontologies enable PDSIDES the ability to exchange knowledge with other product lifecycle management platforms such as product data management systems and simulation-based analysis systems.
- *Facilitate knowledge population*. In order for the computational templates defined in Sec. 2.2 to execute and effect real decisions, the modules of the templates must be populated with specific knowledge (or information). The DSP ontologies are the abstractive representations of the templates, which is very convenient for instantiating different instances with specific information.
- *Facilitate knowledge retrieval*. One of the prerequisites for the reuse of templates and the associated modules is that they can be retrieved from the repository (knowledge base) when needed. The DSP ontologies capture the complex semantic relationships among the modules and templates, which allows it to support semantic-based retrieval that can respond to comprehensive query needs. For detail about semantic retrieval, see Ref. [30].
- *Facilitating consistency maintaining*. Modification of the original templates usually happens in adaptive or variant design, which may lead to inconsistency of the modified templates since the arrangement or values of the modules are

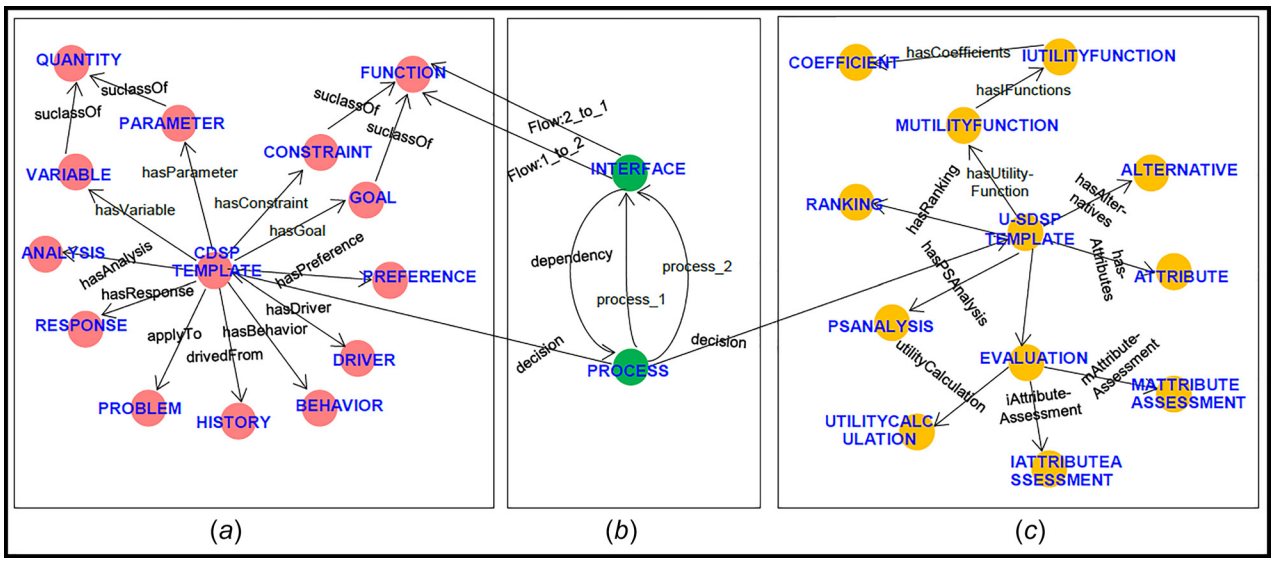

Fig. 2 Ontologies in PDSIDES: (*a*) cDSP ontology [16], (*b*) hierarchical ontology [14], and (*c*) sDSP ontology [15]

changed. The DSP ontologies support rule-based reasoning and appropriately handle the inconsistency, which is discussed in Sec. 3.3.

3 Design of Platform for Decision Support in the Design of Engineering Systems

Based on the primary constructs introduced in Sec. 2, the design of Platform PDSIDES is introduced in this section. First, an overview of PDSDES is presented, and then the platform users and their associated working scenarios are defined and described. Finally, we discuss how knowledge-based decision support is provided for different types of users.

3.1 Platform Overview. An overview of PDSIDES is illustrated in Fig. 3. PDSIDES is divided into three parts: knowledge, users, and decision-based design. What follows is the description of the platform from the bottom-up that includes how these three parts are connected to enable the functionalities.

At the bottom of PDSIDES, decision-related knowledge is stored in the knowledge base. The knowledge including declarative knowledge such as *problem statement, alternatives, attributes, variables, parameters, and constraints*, and procedural knowledge such as consistency rules and computing codes (for calculating, e.g., expected utility of a sDSP template) are organized by a holistic ontology, which is the combination of the three ontologies shown in Fig. 2. In the middle part are the three types of users, namely, the template creator, template editor, and template implementer, which will be formally defined in Sec. 3.2. The three types of users embody three different levels of knowledge (represented by the stairs in Fig. 3). The top level is the template creator who is responsible for creating the DSP templates, the middle level is the template editor who is responsible for editing DSP templates, and the bottom level is the template implementer who is responsible for implementing the DSP templates. The interactions among the three types of users are a closed loop, where the template operational guidance is passed downward from the creator to the editor then to the implementer and the feedback of operating the templates is sent upward from the implementer to the editor and then to the creator. The creation, edit, and implementation of the DSP templates are all facilitated using the holistic ontology. The top part of PDSIDES is about decision-based design. In PDSIDES, design is classified into three types, namely, original design, adaptive design, and variant design; all are realized from a decision-based perspective using the DSP templates. In

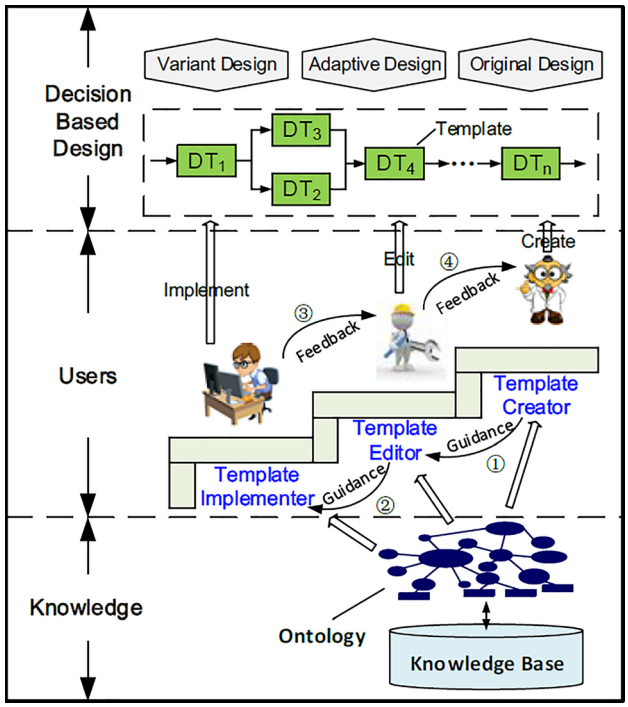

Fig. 3 PDSIDES overview

specific design cases, the underlying decision workflow is represented by networked DSP templates that can be exercised by three types of users through creating, editing, and implementing.

3.2 Users and Working Scenarios. The definitions of three types of users are introduced and their associated working scenarios are described in detail in this section.

3.2.1 Template Creator. Template Creators are domain experts and are responsible for creating DSP templates for original design that calls for new concepts. Original design usually needs the working principle of the system to be setup. In PDSIDES, to do original design template creators, first we need to determine what type of decision needs to be made since different types of decisions require different knowledge. For selection decisions,

creators need to come up with the alternatives for selection, attributes to evaluate the alternatives, and utility functions to measure the performance of the alternatives, etc. For compromise decisions, creators need to identify the variables that represent the features of the system, constraints and bounds that confine the feasible design space, and goals and preferences that determine the aspiration space etc. For hierarchical decisions, in addition to the determination of the "nodes" (which may be selection or compromise) in the decision workflow, creators also need to identify the dependency and the associated information flows between different "nodes." The knowledge can be of the creators' previous experience, prediction, or results from simulation analysis, etc. With this knowledge, template modules are created and assembled to form decision templates that then are tested and stored for reuse.

3.2.2 Template Editor. Template Editors are senior designers who have sufficient knowledge and experience in a specific domain and are responsible for editing (or tailoring) existing decision templates in adaptive design; this requires the original templates to be adapted for new applications. Adaptive design stands for those design cases in which the working principle of the system remains the same while some design consideration varies due to the evolution of the requirements. For example, a pressure vessel may need to be redesigned to adapt to a new goal of minimizing the economic cost because of the intensive market competition. In PDSIDES, to perform adaptive design, template editors need to modify existing DSP templates to reflect the change of design consideration. For the sDSP templates, the modification includes adding/removing *alternatives* and *attributes*, reconfiguration of the *utility functions*, etc. For the cDSP templates, the modification includes adding/removing *variables, constraints, goals,* etc. For the hierarchical DSP templates, modification includes three aspects: the first is about modifying the modules within the DSP templates in a decision workflow, the second is about modifying the number DSP templates (adding/removing sDSP or cDSP templates), the third is about modifying the arrangement (sequence, information flow, etc.) of the DSP templates. The editor's knowledge related to the modification is captured in the newly modified DSP templates, which are stored and used for new applications.

3.2.3 Template Implementer. Template Implementers are designers who have basic knowledge and typically little knowledge or interest in the analysis embodied in the template; they are responsible for executing existing decision templates that result in variant designs that require only parametric changes to the original decision templates. Variant design usually happens when the values of some original design parameters vary. For example, assuming that the original material of a pressure vessel is replaced by some new materials with different *density* and *strength*, the values of parameters *density* and *strength* of the original design model (e.g., the cDSP) need to be updated to reflect the change that will result in a different dimension of the pressure vessel. In PDSIDES, to perform variant design template implementers can change the values of the DSP template parameters including: (1) bounds of the sDSP *attributes* or cDSP *variables*, (2) cDSP parameters and targets, (3) relative importance of the sDSP *attributes* and cDSP *goals*. With the change of parameters values, Template Implementers can execute the DSP templates and get variant designs.

It is noted that in PDSIDES, users with access to higher knowledge levels also have the access to perform the operations that are defined for users of lower knowledge levels. For example, a template creator can be an editor or implementer, and an editor can also be an Implementer. With decisions modeled as DSP templates and users classified into three types, the process of decision-based design in PDSIDES is shown in Fig. 4. A user (e.g., a domain expert) first starts with a problem statement to describe the design problem he/she is faced with, then searches PDSIDES for a DSP template to support the design. In PDSIDES, DSP template searching is a query-based process where a problem statement (a short text) is used as the input and a documented DSP

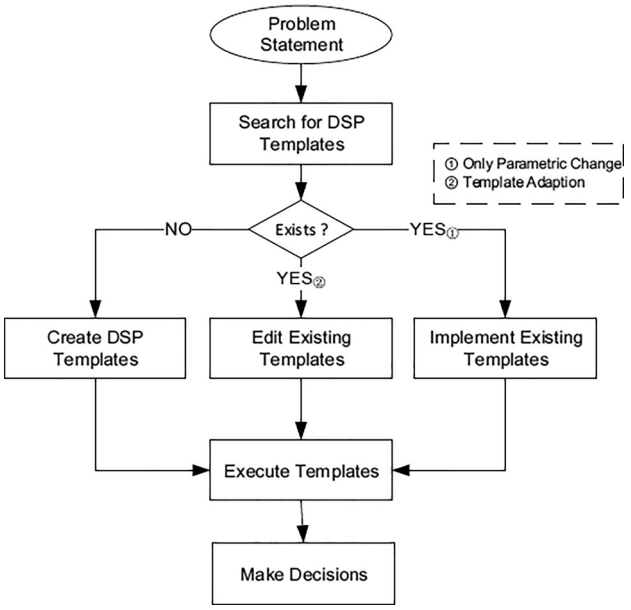

Fig. 4 Flowchart of decision-based design in PDSIDES

template instance is the output. Both the problem statement and template instances are mathematically represented using the bag-of-word approach [31] during the query process. The similarity between the problem statement and different template instances is measured by a cosine coefficient as shown in Eq. (1). As this is not the key focus of this paper, readers are referred to Ref. [32] for detail.

$$\text{sim}\left(\vec{A}, \vec{B}\right) = \frac{\vec{A} \times \vec{B}}{|\vec{A}| \cdot |\vec{B}|} \tag{1}$$

$\vec{A}$ and $\vec{B}$ are two n-dimensional vectors that represent the word frequencies for the given problem statement and a specific template instance, respectively. It should be noted that the *bag-of-word* characterizing the template instance not only includes words from the textual slots such as "name" and "description," but also words from the structural slots such as "variables" and "constraints," etc. which will make the instance more comprehensive and easier to be matched. If no DSP template instance is matched, then a new template needs to be created, to be executed, and to make the decision. If there exists some template(s), the designer needs to further determine how much modification needs to be made to the template. If only a change in the nature of a parameter is needed, then the designer just resets the parameter values, executes the template, and makes a decision. If more adaption is needed, then the designer needs to do the editing before executing the template and make a decision.

3.3 Knowledge-Based Decision Support. The core of PDSIDES is the ontology that integrates the knowledge to support the three types of designers, namely, template creators, template editors, and template implementers. In Fig. 5, we represent how knowledge-based decision support is provided to the three types of designers in their associated working scenarios (taking the cDSP templates as an example).

3.3.1 Template Creators. Template creators provide the vocabulary to them for modeling decisions and capture knowledge from them. Template creators need a formal language to help them describe and model the decisions for original design. The DSP ontologies in PDSIDES can provide them with the vocabulary to model their decisions. For example, The term *variable* is defined as a `Class` with several `slots` including *upper bound, lower bound, unit, value*, etc., which will help specify the module

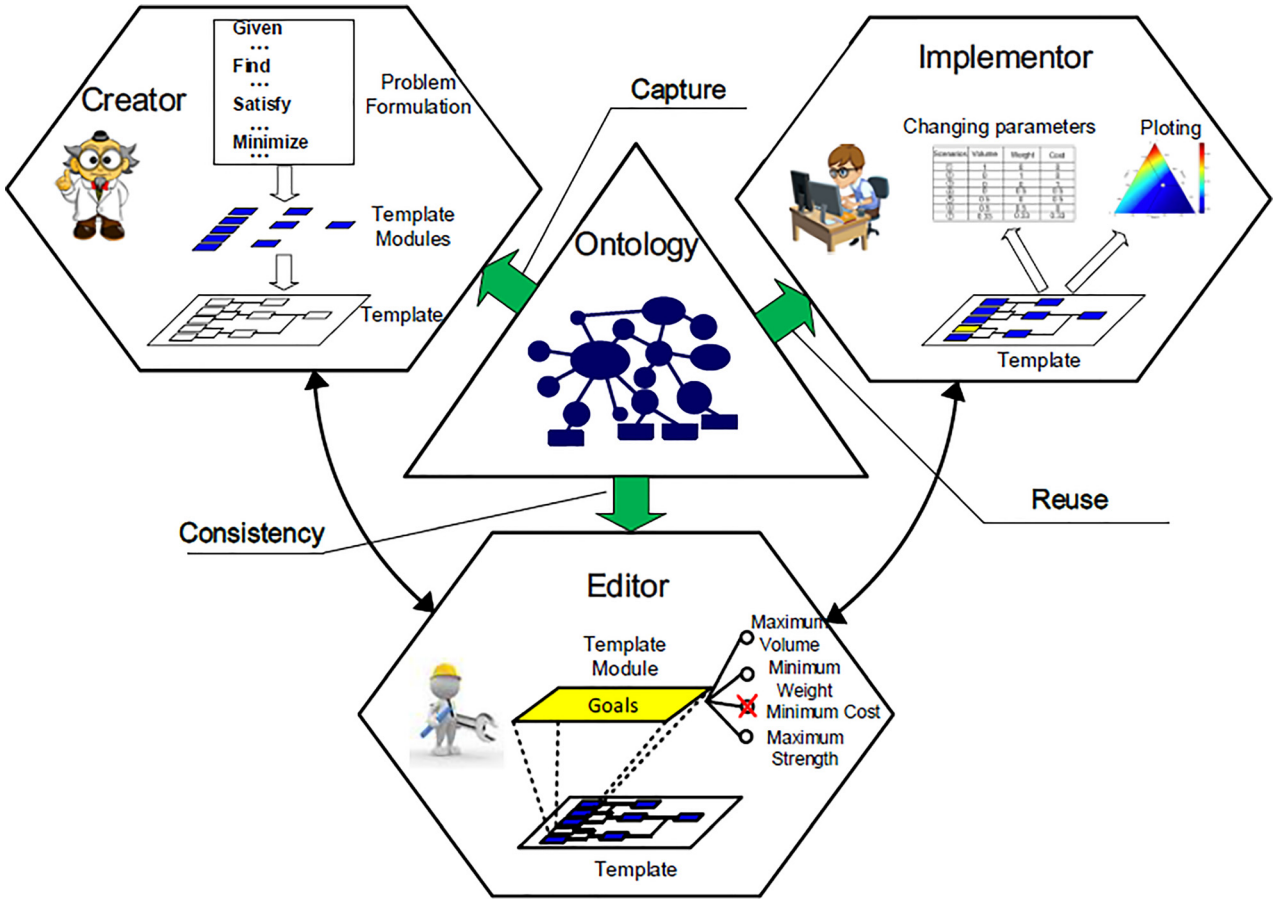

Fig. 5 Knowledge-based decision support in PDSIDES

"*variables*" of the cDSP template. Using the `classes` and `slots` defined in the ontology, DSP templates can be quickly instantiated as instances, which are captured and stored in the database for reuse, as shown in the top-left picture of Fig. 5.

3.3.2 Template Editors. Template editors ensure consistency for editing. As mentioned earlier, modification of existing DSP templates may incur inconsistency, especially when the template is highly complex (e.g., tens of *variables, constraints* or *goals*, etc.) and the editor who modifies the template is not the original creator and does not have the full knowledge about the template. Therefore, what they need is a consistency checking mechanism to identify the potential inconsistency. A rule-based reasoning mechanism is attached to the DSP ontologies in PDSIDES to provide consistency checking service to the template editors. The rules are extracted from the DSP constructs, such as the sum of the weights assigned to the *goals* must be equal to 1. An example that a template editor removes an existing goal (minimum cost) from the cDSP template is shown at the bottom of Fig. 5; PDSIDES will check if this brought inconsistency and inform him.

3.3.3 Template Implementers. Template implementers reuse the documented knowledge and perform postsolution analysis. As we state in Sec. 3.2, template implementers are those who have little knowledge or interest in the analysis embodied in the templates; what they need is information that helps them exercise the template and make the decision. In PDSIDES; the knowledge provided to the template implementers includes both the declarative knowledge and procedural knowledge. The former is captured from template creators and editors, and the latter is built in the platform such as design space exploration algorithms, and plots, which are hard-coded and can be invoked when needed. The picture on the top-right in Fig. 5 represents a template implementer who is changing the weights assigned to different goals and using the ternary plot to identify the insensitive weight sets in order to make a robust decision, during which process the knowledge documented in the template is reused.

4 Implementation of Platform for Decision Support in the Design of Engineering Systems

Platform for decision support in the design of engineering systems is implemented as a two-tier client-server architecture to provide knowledge-based decision support with web browser-based graphical user interfaces (GUI) over the internet, as shown in Fig. 6. In the client-server architecture, applications of PDSIDES are deployed to a web application server (marked as "knowledge server" in Fig. 6) and provide remote user accesses using browsers such as Internet Explorer and Google Chrome. Due to the easy access through web browsers, PDSIDES can reach out to a rich amount of users to get them involved in the decision template creating, editing, and executing process for engineering system design, which is also a knowledge capturing, evolution, and reuse process over the internet. The maintenance and upgrades for PDSIDES in a client-server architecture are fairly convenient since the application package is deployed in one web server instead of distribution to a wide range of client computers. The client side of PDSIDES is the user interaction GUI including template searching and browsing GUI which are designed for locating the wanted DSP templates and presenting them, template creating, and editing GUI, which are designed based on the DSP template structures for the purpose of instantiation and modification of the DSP templates, the template execution, and analysis of GUI, which are designed for executing DSP templates and performing postsolution analysis. The GUI is allowed to communicate with

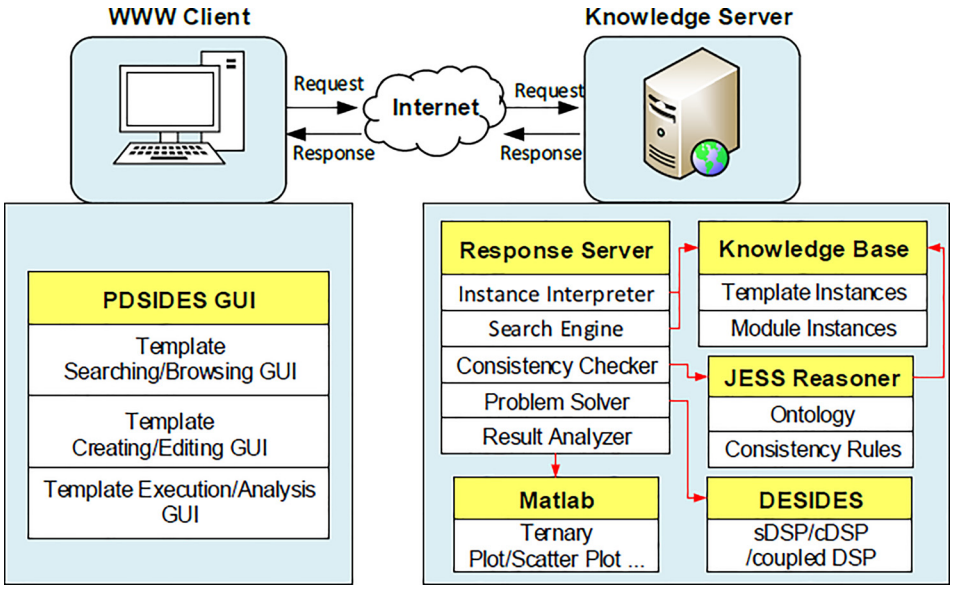

Fig. 6 System architecture of PDSIDES

PDSIDES knowledge server by a request-response mode using the hypertext transfer protocol. PDSIDES knowledge server includes five main parts, namely, response server, knowledge base, JESS reasoner, DSIDES, and MATLAB. The response server is the central "brain" that integrates other four parts for responding to requests. The response sever itself has five components including a search engine, an instance interpreter, a consistency checker, and a problem solver. The instance interpreter is for interpreting the data collected from the template creators (or editors) and formatting it into DSP Template instances according to the DSP ontologies, the generated template instances and module instances are stored in the knowledge base. The search engine is connected to the knowledge base to provide ontological semantic-based knowledge retrieval. Consistency checking is facilitated through a consistency checker together with the JESS reasoner—the rule engine for the JAVATM platform [33], which can provide rule-based intelligence inference. The problem solver is connected to DSIDES for solving the DSPs; it is invoked when a template executer executes a template. The result analyzer is to help users especially template implementers analyze the results produced by the problem solver. MATLAB has a strong capability in providing data visualization tools such as ternary plots and scatter plots; therefore this feature is integrated to PDSIDES.

The front-end (i.e., the GUI) of PDSIDES is realized by JavaScript that can be embedded in the web pages. The development process is facilitated by the Sencha Inc.'s GXT [34]. GXT is a comprehensive Java framework that uses the Google Web Toolkit compiler [35], allowing developers to write applications in Java and compile their codes into highly optimized JavaScript that supports feature-rich web applications. Particularly, in order to enable graph-based interaction in terms of the operation of the DSP networks that may have multiple DSP templates and associated connections involved, Apache Flex [36]—a rich internet application developing framework is integrated to GXT to facilitate the creation of web-based diagrams. A DSP template such as a cDSP template may be very complex and have tens of *variables, parameters, constraints, goals,* etc., which usually makes data transmission overloaded between the front end and the back end. To address this issue, JSON [37]—a lightweight data-interchange format, is used as the data transmission scheme together with the hypertext transfer protocol.

The back end (i.e., the sever side) of PDSIDES is written in Java to enable interoperability among different applications and cross-platform deployment. Many back-end applications, such as the instance interpreter, search engine, consistency checker, and JESS reasoner, are heavily dependent on the DSP ontologies. As mentioned earlier, the DSP ontologies are formalized using the frame-based paradigm that contains `classes` and `slots`; the realization of this paradigm using the frame language is presented in this section, as shown in Fig. 7. The top box in the figure represents the definition of `class` "*SystemGoal*" in the cDSP ontology, which includes definitions of `slots` such as *target, linearity, and equality*, and the associated `facets` such as *type, cardinality, and allowed values*. The development of the DSP ontologies is facilitated using the protégé tool [38], released by Stanford University which provides an environment for modeling the frame-based ontologies and web ontology language ontologies. The frame-based ontology is actually an object-oriented mechanism based on which lots of instances can be populated. Two boxes at the bottom of Fig. 7 represent two instances (i.e., *volume goal* and *weight goal*) of `Class` "*SystemGoal*" represented using frame language. The specific data in the `slots` of the instances are first collected using the template creating/editing GUI, then processed by the instance interpreter, and finally persisted in relational databases (in PDSIDES we use ORACLE). Instances are treated as facts that are processed in the consistency checking process. In the JESS reasoner, all the `facts` are matched to the consistency rules and take certain actions if the corresponding rules are triggered. An example of the consistency rules is as follows:

```
(defrule MAIN::rule_5.1
   (object (is-a cDSPTemplate) (OBJECT? a))
   =>
   (bind? k (Sum (slot-get? a preference) ONE))
   (if (and (<>? k 1.0) (<>? k 0.0)) then (printout t "MESSAGE: the sum
of all the preferences is not 1.0!" crlf)))
```

The rule means that if the sum of the all preferences (i.e., the weights) in any instance of Class cDSPTemplate is not equal to 1, the reasoner will send a message about this inconsistency to the user who is operating the template instance.

The portal of PDSIDES is shown in Fig. 8. A user can log in to PDSIDES through a web browser using a username and a password. Template creators, editors, and implementers are three roles that are assigned to users of PDSIDES according to the knowledge they have in a specific domain. One designer can have more than one role. Each role has its particular view in the platform; the portal is the view shared by all three. The portal includes two main parts: the left-hand side is the navigation panel and the right-hand side is statistical information panel. The former represents the key functionalities of PDSIDES including

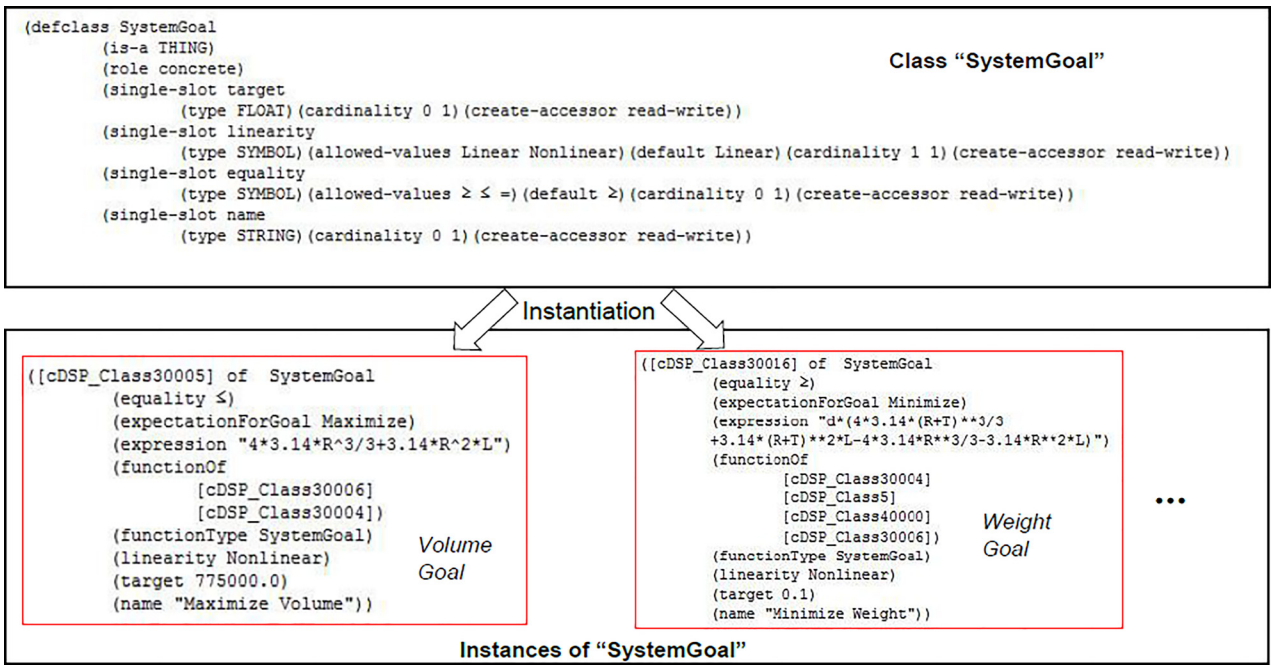

Fig. 7 Frame-based realization of the ontology and associate instances

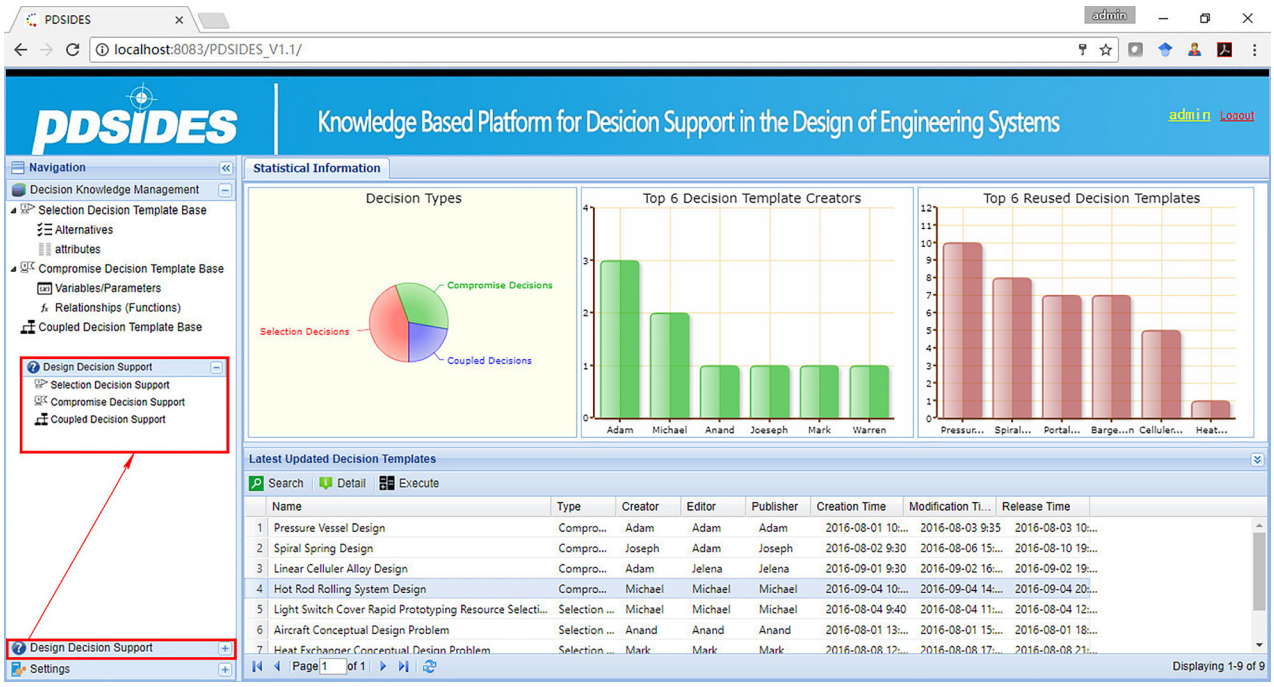

Fig. 8 PDSIDES portal

the decision knowledge management portion (managing knowledge about selection, compromise, and hierarchical decisions. Access is assigned to creators and editors), the design decision support portion (providing DSP template executing and analysis service, access is assigned to Implementers), and the settings portion (purview management, access is assigned only to system administrators). The latter presents the charts and tables in terms of the decision-related knowledge and users. Users can see the number and the distribution of DSP templates in PDSIDES, the ranking of Creators who contribute their knowledge to PDSIDES, the ranking of templates that are reused frequently, and the latest updated DSP templates. They can also search, browse, and execute certain templates.

5 A Test Example for Platform for Decision Support in the Design of Engineering Systems

In this section, the performance of platform PDSIDES is tested via a gear manufacturing process design problem—a complex system design that calls for a series of decisions to be made. The foundational problem is contributed by our industrial partner—the Tata Consultancy Services in India [39]. From the raw material to the final gear product, the material goes through multiple unit operations such as casting, rolling, cooling, forging, and machining, which are some of the processes in the steel manufacturing process chain. In order to obtain the desired end properties of the gear produced, proper decisions need to be made about the

process control parameters (*set points*) at each of these processes. A large number of plant trials involving time and cost are needed to identify these operating set points. An alternative to this is to exploit the advancements in modeling tools and frameworks to carry out the design of the system to realize the end product. To couple the material processing-structure-property-performance spaces, we need to achieve the vertical and horizontal integration of models which further allows us to carry out the integrated decision-based design of the manufacturing processes to realize the end product [40–42]. Decisions to be made at each manufacturing unit are formulated as cDSPs and linked as a decision network (mathematically modeled as coupled cDSPs) using a goal-oriented, inverse decision-based design method [40]. In this paper, the hot rod rolling system design problem addressed by Nellippallil et al. [40,43] is used as an example to test the performance of PDSIDES. As mentioned earlier, the problem includes multiple stages. We frame a boundary within the cooling stage and the end rod product requirements for the sake of simplicity.

5.1 Original Design. In original design, the template creator (domain expert) formulates in PDSIDES the cDSP for the problem boundary framed within the hot rod rolling process chain problem by taking into account the complete information flow across models thereby establishing relationships. The relationships established in the original design cDSP are the end mechanical properties of the product; *YS* (yield strength), *TS* (tensile strength), *ITT* (impact transition temperature), and *HV* (hardness) as a function of the system variables that are the output after rolling and input to cooling stage. The output parameters after cooling like FGS (ferrite grain size, D_α), X_f (phase fractions of ferrite), S_0 (pearlite interlamellar spacing), and composition variables that defines the end mechanical properties are defined as constraints in the cDSP formulated. The end product mechanical property goals, for example, maximizing *YS*, *TS* and minimizing *ITT* along with the goal for managing banding by maximizing ferrite fraction are captured in the cDSP. These goals are controlled by the independent system variables of this problem namely *CR* (cooling rate), *AGS* (grain size after rolling), *C* (carbon), and *Mn* (manganese). The upper and lower limits for the system variables and the maximum and minimum values for certain cooling stage parameters as well as for the mechanical properties are defined in the cDSP as bounds and constraints. The target values for the goals are defined as $YS_{\text{Target}} = 400\,\text{MPa}$, $TS_{\text{Target}} = 780\,\text{MPa}$, $ITT_{\text{Target}} = -90\,^\circ\text{C}$, $X_{f\,\text{Target}} = 0.8$. The original design cDSP reads as follows:

Given

(1) End requirements identified for the rod rolling process
- Maximize Yield Strength (Goal)
- Maximize Tensile Strength (Goal)
- Minimize ITT (Goal)
- Maximize Ferrite Fraction (Goal)
- Maximize Hardness (Requirement)

(2) Well established empirical and theoretical correlations, RSMs and complete information flow from the end of rolling to the end product mechanical properties (more description in reference [40])

(3) System variables and their ranges

Table 1 System variables and ranges for CDSP

Sr. No	System variables	Ranges
1	X_1, CR	11–100 K/min
2	X_2, AGS	30–100 μm
3	X_3, the carbon concentration ([*C*])	0.18–0.3%
4	X_4, the manganese concentration ([Mn])	0.7–1.5%

Find

System Variables

- X_1, Cooling Rate (CR)
- X_2, Austenite Grain Size (AGS)
- X_3, the carbon concentration ([*C*])
- X_4, the manganese concentration ([Mn])

Deviation Variables

$$d_i^-, d_i^+, \quad i = 1, 2, 3, 4$$

Satisfy

System Constraints

- Minimum ferrite grain size constraint
- Maximum ferrite grain size constraint
- Minimum pearlite interlamellar spacing constraint
- Maximum interlamellar spacing constraint
- Minimum ferrite phase fraction constraint (manage banding)
- Maximum ferrite phase fraction constraint (manage banding)
- Minimum manganese concentration constraint (manage banding)
- Maximum manganese concentration constraint (manage banding)
- Maximum carbon equivalent constraint (manage banding)
- Minimum yield strength constraint
- Maximum yield strength constraint
- Minimum tensile strength constraint
- Maximum tensile strength constraint
- Minimum hardness constraint
- Maximum hardness constraint
- Minimum ITT constraint
- Maximum ITT constraint

System Goals

Goal 1:

- Maximize yield strength

$$\frac{YS(X_i)}{YS_{\text{Target}}} + d_1^- - d_1^+ = 1$$

Goal 2:

- Maximize tensile strength

$$\frac{TS(X_i)}{TS_{\text{Target}}} + d_2^- - d_2^+ = 1$$

Goal 3:

- Minimize ITT

$$\frac{ITT_{\text{Target}}}{ITT(X_i)} - d_3^- + d_3^+ = 1$$

Goal 4:

- Maximize ferrite fraction

$$\frac{X_f(X_i)}{X_{f\,\text{Target}}} + d_4^- - d_4^+ = 1$$

Variable bounds

Defined in Table 1

Bounds on deviation variables

$$d_i^-, d_i^+ \geq 0 \text{ and } d_i^- * d_i^+ = 0, \text{i} = 1, 2, 3$$

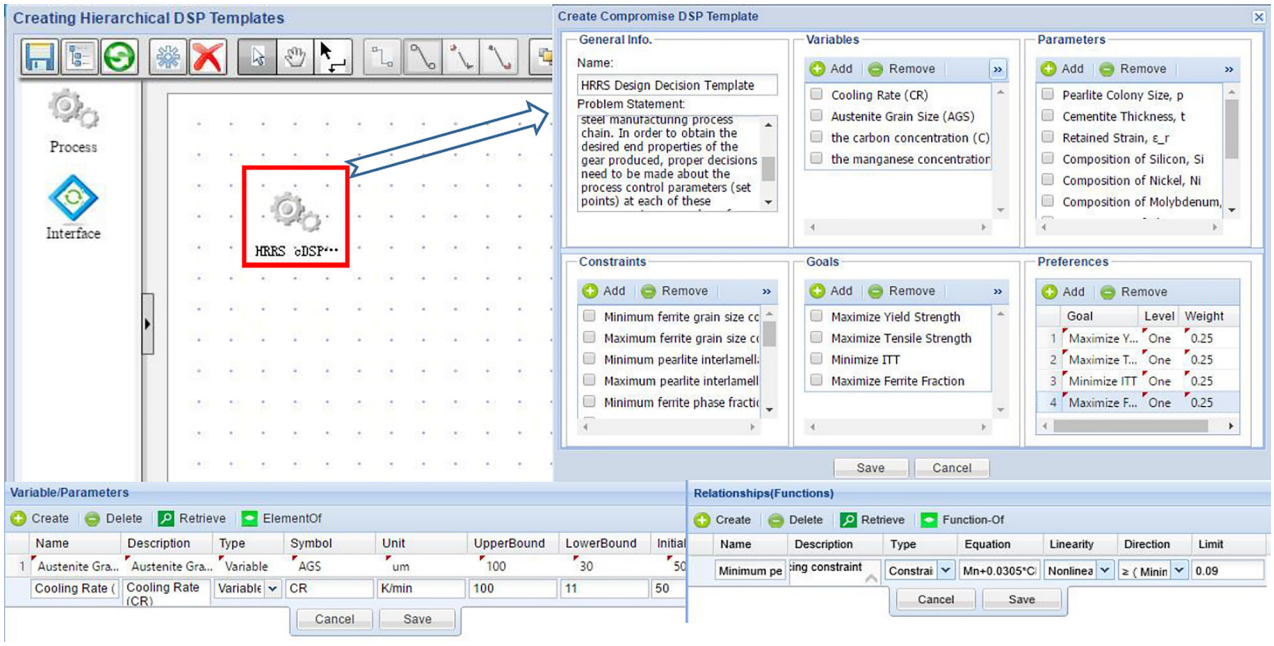

Fig. 9 Creating the HRRS design decision template in PDSIDES

Minimize

We minimize the deviation function

$$Z = \sum_{i=1}^{4} W_i \left(d_i^- + d_i^+\right); \quad \sum_{i=1}^{4} W_i = 1$$

By the formulation of cDSP, knowledge associated with the following inverse problem is captured: *Given the end product mechanical properties of a new steel product mix, what should be the microstructure after rolling and design set points for cooling stage that satisfies the requirements identified?* To facilitate knowledge capturing process in the computational environment, PDSIDES provides the GUI for the template creator to create DSP templates, as shown in Fig. 9.

On the left-hand side of the canvas are the building blocks, including *Process* and *Interface*, which are formally defined in the ontology for the purpose of creating decision network templates (hierarchical DSP templates). Since there is only one cDSP formulated for the original design of HRRS, the template creator can simply instantiate a *process* on the canvas, and embody it with a cDSP template. The cDSP template is created in the "compromise decision template base" portion of PDSIDES. As shown in the window on the top right of Fig. 9, the template creator can instantiate the HRRS cDSP template by specifying the slots including *name, problem statement, variables, parameters, constraints, goals, and preferences* using data such as cooling rate, austenite grain size (AGS), and carbon concentration, of the HRRS cDSP. Facet information of the `slots`, such as *symbol, unit* of a *variable* and *equation*, *limit* of a *constraint*, are further specified using the GUI designed for the instantiation of template modules, as shown in the two panels on the bottom. When the HRRS cDSP template is populated with specific information, it is sent to the knowledge server for consistency checking, calculation of results, persistence in the knowledge base, and is ready for future reuse in adaptive and variant designs.

5.2 Adaptive Design. In adaptive design, the template editor (senior designer) modifies the existing original design cDSP template according to new requirements. In the hot rod rolling problem addressed, the cDSP template of the original design relates the end product mechanical properties as a function of microstructure factors after rolling and the cooling stage operating parameters. The intermediate factors, for example, the ferrite grain size after cooling and the pearlite interlamellar spacing, which directly influence mechanical properties, are defined as constraints. Suppose, a situation arises that the designer is interested in knowing the range of microstructure factors after cooling that will satisfy a given end mechanical property requirements. In such a situation, new decision models need to be created by considering the microstructure factors after cooling as independent variables to define the end mechanical properties. This requirement can be easily satisfied by editing the existing formulated original design cDSP template in PDSIDES. The editing involves two major steps: step 1, decompose the original cDSP template into two separate cDSP templates, and step 2, link the two separate cDSP templates using an *Interface*.

The process of the first step is shown in Fig. 10. The original cDSP is decomposed into two cDSPs, namely, *cDSP 1* and *cDSP 2*. *cDSP 1* relates the end mechanical properties as a function of microstructure factors (D_α, X_f, S_0, *Mn*, Si, *N*) after cooling. The combination of microstructure factors after cooling that best satisfies the end requirements are identified by exercising this sub-cDSP. While, *cDSP 2* has the best combination of microstructure factor values after cooling identified from *cDSP 1* as goals. Using *cDSP 2*, the relationship between the microstructure factors after cooling with the microstructure after rolling and the cooling stage operating parameters (*AGS*, *CR*, *C*, *Mn*) is established. To realize the decomposition, modification of the original cDSP is as follows:

- For cDSP 1:
- Set ferrite grain size (D_α), phase fraction of ferrite (X_f), pearlite interlamellar spacing (S_0), manganese concentration ([Mn]), the composition of Si ([Si]), and the composition of

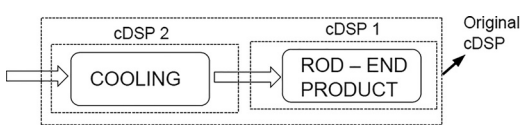

Fig. 10 Decomposition of the original CDSP

N ([N]) which are system constraints of the original cDSP, to be system variables.
- Keep the rest constraints and goals the same as the original cDSP.
- For cDSP 2:
- Keep the system variables, namely, CR, AGS, the carbon concentration ([C]), and the manganese concentration ([Mn]), the same as they are in the original cDSP.
- Set ferrite grain size (D_α), phase fraction of ferrite (X_f), and pearlite interlamellar spacing (S_0), which are system variables of *cDSP 1*, to be system goals.
- Set the final values of D_α, X_f, and S_0 obtained from *cDSP 1*, to be the targets of the system goals of *cDSP 2*.

The connection between *cDSP 1* and *cDSP 2* is that the output (i.e., the final values of the system variables) of *cDSP 1* comprises the input (i.e., the targets of the system goals) of *cDSP 2*. This connection represents the information workflow that links two cDSPs, which maps to step 2 mentioned earlier for editing the original cDSP template. On the platform, the editing and the associated consistency checking process is shown in Fig. 11. The template editor can instantiate two new cDSP templates on the canvas, as highlighted by two rectangles marked as "End Product cDSP" and "Cooling cDSP" that represent *cDSP 1* and *cDSP 2*, respectively. The instantiation of these two cDSP templates is the same as that is shown in Fig. 9. It is noted that many modules of the original cDSP template are reused due to the modularization during the instantiation process of the two new cDSP templates. The link between two cDSP templates is captured by the instantiation of an *Interface* marked as "exchange" that is highlighted in the circle. Configuration of the Interface is performed in the right window, where information in terms of interface type, strength, information flow, etc. is specified. According to the interaction between the two cDSP templates, the information flow is weak (one-way), sequential and flows from *cDSP 1* to *cDSP 2*. The contents of the flow are the values of the five system variables of *cDSP 1*. Before executing the edited decision templates, the editor needs to check if there is any inconsistency due to the editing. The consistency checking process is shown in the panel on the bottom of Fig. 11. Consistency rules can be dynamically defined and added into the reasoner for reasoning. If no rule is violated, the newly edited cDSP templates would be ready for execution, storage, and reuse.

The results are obtained after exercising the *cDSP 1* and *cDSP 2* and are provided in reference [40] and are not repeated here.

5.3 Variant Design. In variant design, the template implementer makes parametric modifications to the already developed decision templates and executes the templates for different scenarios. In this paper, we showcase variant designs by executing the cDSP template of the original design for different scenarios identified by assigning weights to the deviations associated with each goal. We also illustrate the efficacy of ternary plots in PDSIDES to support the template implementer in exploring the solution space of variant designs to make appropriate design decisions. For the problem formulated in original cDSP, the template implementer is interested in accomplishing the following goals: maximizing ferrite fraction (to manage banding), maximizing tensile strength, maximizing yield strength, and minimizing impact transition temperature. To visualize the goals in ternary space, it needs the template editor to first edit the original cDSP template to remove the goal on impact transition temperature and assign it as a constraint with minimum and maximum value. This is carried out because it is known that the impact transition temperature is directly influenced by changes in weights to other goals and hence need not be considered as a direct goal. Thus, the variant design cDSP has three goals—maximizing ferrite fraction, maximizing tensile strength, and maximizing yield strength. Having developed the variant design cDSP, the next step for the template implementer is to identify design scenarios for execution.

On the platform, the identification of design scenarios is facilitated by the panel shown in Fig. 12. The template implementer can specify several weight combinations (each combination stands for one scenario) for goal deviations using the table on the top, PDSIDES will calculate the result with respect to each of the weight combination. In this paper, 19 different scenarios are identified, for more information on identifying scenarios, see Ref.

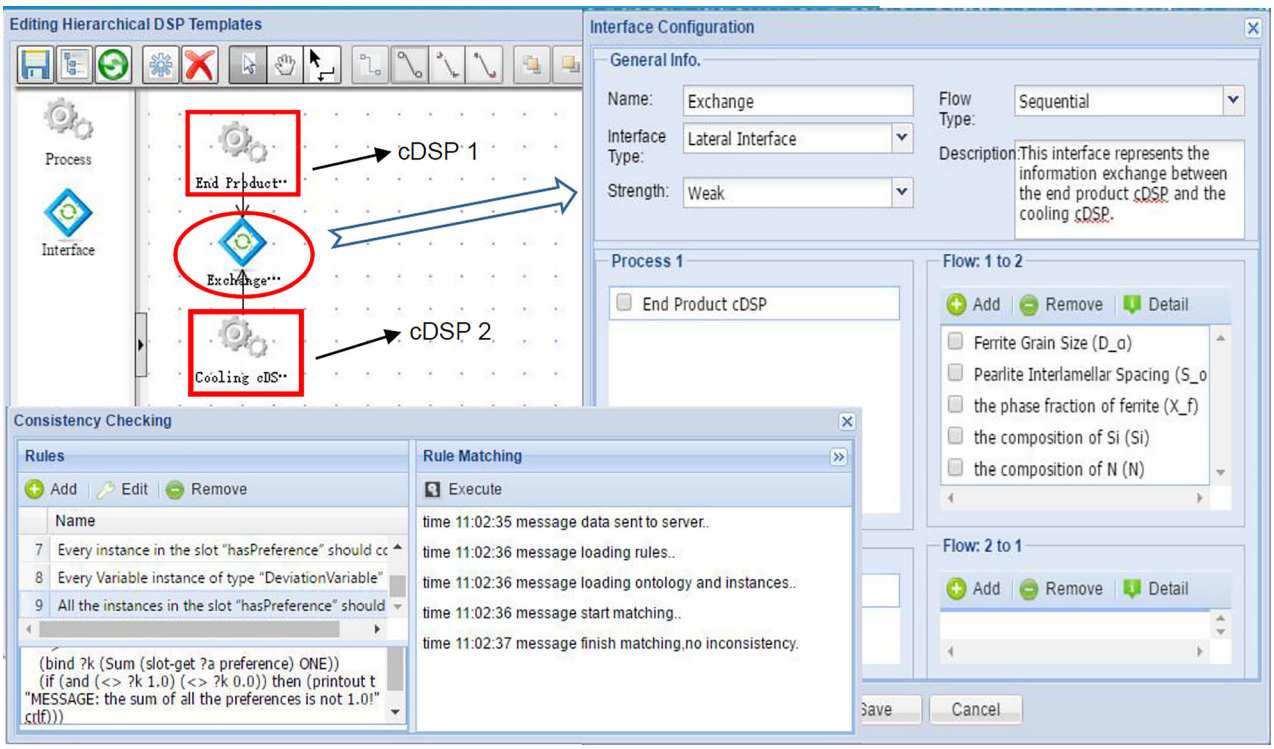

Fig. 11 Editing the HRRS design decision template in PDSIDES

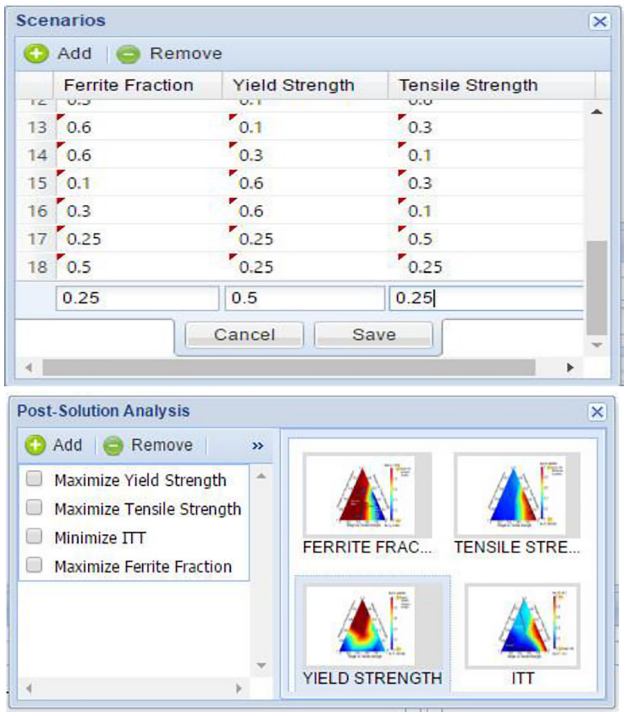

Fig. 12 Exercising the HRRS design decision template in PDSIDES

[41]. The template implementer exercises the original cDSP template in variant design scenarios and the results obtained are sent to MATLAB (at the back-end of PDSIDES) to plot as ternary plots shown in the bottom panel of Fig. 12. The template implementer identifies regions (weight combinations) that satisfy the requirements from the ternary plots. More information on the creation of ternary plots and the evaluation of the same is available in Ref. [41].

The ternary plot for ferrite fraction is shown in Fig. 13. The requirement for the template implementer is to maximize ferrite fraction to a value of 0.8 and the maximum value achieved on exercising the cDSP is 0.7116 identified by the light dashed line in the high ferrite region region of Fig. 13. Any weight combination of goals in this region achieves high ferrite fraction. Similarly, the high pearlite fraction region is identified by the high pearlite region in Fig. 13. The highly banded ferrite–pearlite microstructure region is identified in the boundary between these two regions. The same method is extended to identify the regions that satisfy the requirements of tensile strength, yield strength, and impact transition temperature.

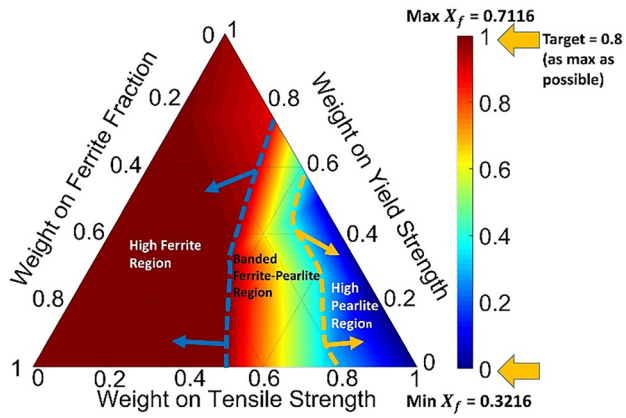

Fig. 13 Ternary plot for ferrite fraction

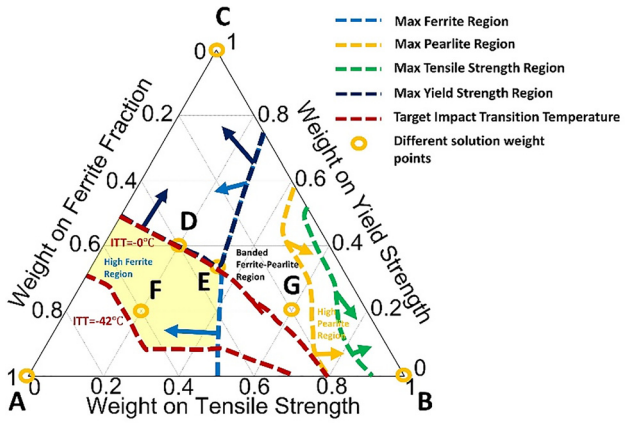

Fig. 14 Superimposed ternary plot

Table 2 Identified solution points after exploration

Sol. Pt	CR (K/min)	AGS (μm)	C (%)	Mn (%)	X_f	YS (MPa)	TS (MPa)	ITT (°C)
A	16.5	99.9	0.18	0.7	0.71	232	487.7	−26
B	99.9	30	0.29	1.5	0.32	248	662	99
C	22.8	30	0.18	1.5	0.7	284	526	3.5
D	11	30	0.18	1.5	0.71	283	519	0
E	11	30	0.18	1.5	0.71	283	519	0
F	11	30	0.18	0.7	0.7	244	513	−42
G	62	30	0.19	1.5	0.65	281	547	15

Since the template implementer's interest is to identify a common region that satisfies all the goals, a superimposed ternary plot having all the goals is generated as showcased in Fig. 14. From the superimposed ternary plot, several solution weight points (A, B, C, D, E, F, G) are identified and analyzed. The results associated with these solution weight points are summarized in Table 2. On analyzing Fig. 14 and Table 2, it is seen that the light banded ferrite-pearlite region satisfies all the requirements for managing banding (high ferrite), maximizing yield strength, maximizing tensile strength, and minimizing *ITT* in the best possible manner. However, the requirements for high tensile strength and high yield strength are compromised to satisfy requirements like managing banding and minimizing *ITT*. It is also observed that a high ferrite region supports the maximization of yield strength and minimization of *ITT*. The maximization of tensile strength, however, is supported by high pearlite fraction. Point F out of these multiple solutions listed in Table 2 is picked as F satisfies all the requirements in the best possible manner.

By reusing the knowledge archived in the original HRRS design cDSP template for execution and utilizing the ternary plot for post-solution analysis, the template implementer explores the solution space of variant designs and makes appropriate design decisions.

6 Closing Remarks

Engineering system design is fundamentally a decision-making process and knowledge plays a critical role in facilitating decision making. In this paper, we present a Knowledge-Based Platform for Decision Support in the Design of Engineering Systems. In PDSIDES, decision-related knowledge is modeled as modular, computational templates based on the DSP constructs using ontology to facilitate execution and reuse. In order to provide users of different knowledge levels with a proper decision support, we define three types of users, namely, *template creators*, *template editor*, and *template implementers*, who perform original design, adaptive design, and variant design respectively. The unique advantage of PDSIDES is that it provides the functionality to

capture knowledge when *template creators* create decision templates in original design, maintain consistency when *template editor* modifies decision templates in adaptive design, and provide a package of documented knowledge when *template implementers* execute decision templates in variant design.

Distributed information control is not yet considered in the current version of PDSIDES. Future research opportunities lie in enabling the negotiation of collaborative decisions that are controlled by different stakeholders. For example, in the HRRS design, example process designers at different stages such as rolling and cooling may not be willing to share the full information in their own decision-making process, and then the negotiation of a collaborative decision is needed. Providing the functionality for negotiating collaborative decisions would be of great potential for the application of PDSIDES in a supply chain environment, where the decision makers are distributed.

Acknowledgment

Zhenjun Ming acknowledges the support from China Scholarship Council grant 201406030014 for visiting the System Realization Laboratory at the University of Oklahoma and carrying out this research. Yan Yan and Guoxin Wang acknowledge the support from National Natural Science Foundation of China grant 51375049. The authors thank TRDDC, Tata Consultancy Services, Pune for supporting this work (Grant No. 105-373200). Janet K. Allen and Farrokh Mistree gratefully acknowledge funding from the John and Mary Moore Chair and L.A. Comp Chair at the University of Oklahoma respectively. This paper is an outcome of the International Systems Realization Partnership between the Institute for Industrial Engineering @ The Beijing Institute of Technology, The Systems Realization Laboratory @ The University of Oklahoma and the Design Engineering Laboratory @ Purdue.

Nomenclature

cDSP =	compromise DSP, a domain-independent decision construct for formulating compromise decisions
DSP =	decision support problem, the construct for formulating decisions encountered in engineering design. Independent of the domain of application
DSP Template =	the domain-independent DSP construct instantiated with the infusion of domain knowledge and is computer interpretable
Decision Workflow =	a set of interconnected decisions that are represented by a number of DSP templates either serially or hierarchically
PDSIDES =	knowledge-based platform for decision support in the design of engineering systems
sDSP =	selection DSP, a domain-independent decision construct for formulating selection decisions
Template Creators =	users of the PDSIDES platform who create DSP templates (original design)
Template Editors =	users of the PDSIDES platform who edit or tailor existing DSP templates (adaptive design)
Template Implementers =	users of the PDSIDES platform who execute DSP existing templates (variant design)

References

[1] Soria, N., Colby, M. K., Tumer, I. Y., Hoyle, C., and Tumer, K., 2017, "Design of Complex Engineered System Using Multiagent Coordination," ASME J. Comput. Inf. Sci. Eng., **18**(1), p. 011003.
[2] Berg, L. P., and Vance, J. M., 2016, "An Industry Case Study: Investigating Early Design Decision Making in Virtual Reality," ASME J. Comput. Inf. Sci. Eng., **17**(1), p. 011001.
[3] Afshari, H., Peng, Q., and Gu, P., 2016, "Design Optimization for Sustainable Products Under Users' Preference Changes," ASME J. Comput. Inf. Sci. Eng., **16**(4), p. 041001.
[4] Daskilewicz, M. J., and German, B. J., 2012, "Rave: A Computational Framework to Facilitate Research in Design Decision Support," ASME J. Comput. Inf. Sci. Eng., **12**(2), p. 021005.
[5] Kuppuraju, N., Ganesan, S., Mistree, F., and Sobieski, J. S., 1985, "Hierarchical Decision-Making in System-Design," Eng. Optim., **8**(3), pp. 223–252.
[6] Shah, J. J., and Mantyla, M., 1995, *Parametric and Feature-Based CAD/CAM: Concepts, Techniques and Applications*, Wiley, New York.
[7] Coyne, R. D. D., Rosenman, M. A., Radford, A. D., Balachandran, M., and Gero, J. S., 1990, *Knowledge-Based Design Systems*, Addison-Wesley Pub. Co, Boston, MA.
[8] Finger, S., and Dixon, J. R., 1989, "A Review of Research in Mechanical Engineering Design—Part II: Representations, Analysis, and Design for the Life Cycle," Res. Eng. Des., **1**(2), pp. 121–137.
[9] Verhagen, W. J. C., Bermell-Garcia, P., Van Dijk, R. E. C., and Curran, R., 2012, "A Critical Review of Knowledge-Based Engineering: An Identification of Research Challenges," Adv. Eng. Inform., **26**(1), pp. 5–15.
[10] Rocca, G. L., 2012, "Knowledge Based Engineering: Between AI and CAD. Review of a Language Based Technology to Support Engineering Design," Adv. Eng. Inform., **26**(2), pp. 159–179.
[11] Jakiela, M. J., and Papalambros, P. Y., 1989, "Design and Implementation of a Prototype 'Intelligent' CAD System," ASME J. Mech. Transm. Autom. Des., **111**(2), pp. 252–258.
[12] Sapuan, S. M., 2001, "A Knowledge-Based System for Materials Selection in Mechanical Engineering Design," Mater. Des., **22**(8), pp. 687–695.
[13] Rockwell, J. A., Grosse, I. R., Krishnamurty, S., and Wileden, J. C., 2010, "A Semantic Information Model for Capturing and Communicating Design Decisions," ASME J. Comput. Inf. Sci. Eng., **10**(3), p. 031008.
[14] Ming, Z., Wang, G., Yan, Y., Panchal, J. H., Goh, D., Allen, J. K., and Mistree, F., 2017, "Ontology-Based Representation of Design Decision Hierarchies," ASME J. Comput. Inf. Sci. Eng., **18**(1), p. 011001.
[15] Ming, Z., Wang, G., Yan, Y., Dal Santo, J., Allen, J. K., and Mistree, F., 2017, "An Ontology for Reusable and Executable Decision Templates," ASME J. Comput. Inf. Sci. Eng., **17**(3), p. 031008.
[16] Ming, Z., Yan, Y., Wang, G., Panchal, J. H., Goh, C. H., Allen, J. K., and Mistree, F., 2016, "Ontology-Based Executable Design Decision Template Representation and Reuse," Artif. Intell. Eng. Des., Anal. Manuf., **30**(4), pp. 390–405.
[17] Mistree, F., Smith, W. F., Bras, B. A., Allen, J. K., and Muster, D., 1990, "Decision-Based Design: A Contemporary Paradigm for Ship Design," Trans., Soc. Nav. Archit. Mar. Eng., **98**, pp. 565–597.
[18] Hazelrigg, G. A., 1998, "A Framework for Decision-Based Engineering Design," ASME J. Mech. Des., **120**(4), pp. 653–658.
[19] Muster, D., and Mistree, F., 1988, "The Decision Support Problem Technique in Engineering Design," Int. J. Appl. Eng. Educ., **4**(1), pp. 23–33.
[20] Mistree, F., Lewis, K., and Stonis, L., 1994, "Selection in the Conceptual Design of Aircraft," AIAA J., 5th Symposium on Multidisciplinary Analysis and Optimization, American Institute of Aeronautics and Astronautics, Panama City Beach, FL, Sept. 7–9.
[21] Mistree, F., Hughes, O. F., and Bras, B. A., 1993, "The Compromise Decision Support Problem and the Adaptive Linear Programming Algorithm," *Structural Optimization: Status and Promise*, AIAA, Washington, DC, pp. 247–286.
[22] Bascaran, E., Bannerot, R., and Mistree, F., 1987, "The Conceptual Development of a Methodology for Solving Multi-Objective Hierarchical Thermal Design Problems," ASME Paper No. 87-HT-62.
[23] Smith, W. F., 1985, *The Development of AUSEVAL: An Automated Ship Evaluation System*, M.S. dissertation, University of Houston, Houston, TX.
[24] Reddy, R., Smith, W., Mistree, F., Bras, B., Chen, W., Malhotra, A., Badhrinath, K., Lautenschlager, U., Pakala, R., and Vadde, S., 1996, "DSIDES User Manual," Georgia Institue of Technology, Atlanta, Georgia.
[25] Panchal, J. H., Fernández, M. G., Paredis, C. J. J., and Mistree, F., 2004, "Reusable Design Processes Via Modular, Executable, Decision-Centric Templates," AIAA Paper No. 2004-4601.
[26] Gruber, T. R., 1993, "A Translation Approach to Portable Ontology Specifications," Knowl. Acquis., **5**(2), pp. 199–220.
[27] Zhan, P., Jayaram, U., Kim, O., and Zhu, L., 2010, "Knowledge Representation and Ontology Mapping Methods for Product Data in Engineering Applications," ASME J. Comput. Inf. Sci. Eng., **10**(2), p. 021004.
[28] Li, Z., Raskin, V., and Ramani, K., 2008, "Developing Engineering Ontology for Information Retrieval," ASME J. Comput. Inf. Sci. Eng., **8**(1), p. 011003.
[29] Wang, H., Noy, N., Rector, A., Musen, M., Redmond, T., Rubin, D., Tu, S., Tudorache, T., Drummond, N., and Horridge, M., "Frames and OWL Side by Side," Presentation Abstracts, Stanford University, Stanford, CA, p. 54.
[30] Mocko, G. M., Rosen, D. W., and Mistree, F., 2007, "Decision Retrieval and Storage Enabled Through Description Logic," ASME Paper No. DETC2007-35644.
[31] Baeza-Yates, R., and Ribeiro-Neto, B., 2011, *Modern Information Retrieval: The Concepts and Technology behind Search*, Addison Wesley, Boston, MA.
[32] Salton, G., Wong, A., and Yang, C.-S., 1975, "A Vector Space Model for Automatic Indexing," Commun. ACM, **18**(11), pp. 613–620.
[33] Friedman-Hill, E., 2015, "JESS - the Rule Engine for the JavaTM Platform," Manning Publications Co., Shelter Island, NY, accessed June 16, 2018, http://herzberg.ca.sandia.gov/

[34] Sencha, 2018, "Sencha GXT," accessed June 16, 2018, https://www.sencha.com/products/gxt/#overview
[35] Google, 2018, "Google Web Toolkit," accessed June 16, 2018, http://www.gwtproject.org/overview.html
[36] Adobe, 2018, "Apache Flex," accessed June 16, 2018, https://www.adobe.com/devnet/flex.html
[37] JSON, 2018, "JavaScript Object Notation," accessed June 16, 2018, http://www.json.org/
[38] Stanford University, 2018, "Protégé 3.5," accessed June 16, 2018, https://protegewiki.stanford.edu/wiki/Protege_3.5_Release_Notes
[39] Allen, J. K., Mistree, F., Panchal, J., Gautham, B., Singh, A., Reddy, S., Kulkarni, N., and Kumar, P., 2013, "Integrated Realization of Engineered Materials and Products: A Foundational Problem," 2nd World Congress on Integrated Computational Materials Engineering (ICME), Salt Lake City, UT, July 7–11, pp. 277–284.
[40] Nellippallil, A. B., Vignesh, R., Allen, J. K., Mistree, F., Gautham, B. P., and Singh, A. K., 2017, "A Goal-Oriented, Inverse Decision-Based Design Method to Achieve the Vertical and Horizontal Integration of Models in a Hot-Rod Rolling Process Chain," ASME Paper No. DETC2017-67570.
[41] Nellippallil, A. B., Song, K. N., Goh, C.-H., Zagade, P., Gautham, B., Allen, J. K., and Mistree, F., "A Goal-Oriented, Sequential, Inverse Design Method for the Horizontal Integration of a Multi-Stage Hot Rod Rolling System," ASME J. Mech. Des., **139**(3), p. 031403.
[42] Nellippallil, A. B., Song, K. N., Goh, C.-H., Zagade, P., Gautham, B., Allen, J. K., and Mistree, F., "A Goal Oriented, Sequential Process Design of a Multi-Stage Hot Rod Rolling System," ASME Paper No. DETC2016-59402.
[43] Nellippallil, A. B., Vignesh, R., Allen, J. K., Mistree, F., Gautham, B. P., and Singh, A. K., 2017, "A Decision-Based Design Method to Explore the Solution Space for Microstructure After Cooling Stage to Realize the End Mechanical Properties of Hot Rolled Product," Fourth World Congress on Integrated Computational Materials Engineering (ICME 2017), Ypsilanti, MI, May 21–25, pp. 353–363.