Proceedings of the ASME 2011 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference
IDETC/CIE 2011
August 28-31, 2011, Washington, DC, USA

DETC2011-47291

TOWARDS ENABLING VISUAL DESIGN EXPLORATION INVOLVING MULTIPLE ABSTRACTIONS OF DESIGN DESCRIPTIONS

Srikanth Devanathan
School of Mechanical Engineering
Purdue University
West Lafayette, IN 47907, USA

Karthik Ramani*
School of Mechanical Engineering
Purdue University
West Lafayette, IN 47907, USA

ABSTRACT

Designers use several visual tools for exploring and understanding design problems and solutions. House of Quality (HoQ), function-structure, Morphological matrices, concept selection tables, 2D drawings are some of the visual tools and representations used in mechanical design. In this article we attempt to connect these visual tools and their underlying models to support exploration in early design using a representation called the working knowledge model (WKM). We identify two key aspects in design that are important for establishing such connections: different abstractions are used to describe the same design element, and several alternatives are considered during exploration. The constituent elements of the visual tools such as engineering characteristics (ECs) in a HoQ are described using classes of the information model. A simple wiki-based implementation is described that allows the user to tag the wiki text with WKM classes, which is then extracted to populate a database. This information is used by visual tools that can be embedded within a wiki page; the decisions taken by the user using these visual tools are then incorporated back into the WKM database and the wiki is updated if needed. A case study of the design of an automotive flow control valve is described to demonstrate the prototype.

1 INTRODUCTION

The design process is an iterative map from the customer requirements to the final design of the product [1]. The engineering design process broadly follows the pattern of (1) task clarification / specification development, (2) conceptual design, (3) embodiment design, and (4) detailed design [1-7]. Conceptual design takes the statement of the problem and generates solutions that are called schemes [2], or principle solutions [6], or concepts. These concepts are elaborated and a selection is made among them for further refinement at the end of the embodiment design stage. The result is usually a set of general drawings and there is a great deal of feedback from this stage to the conceptual stage [2, 8]. During detailed design, a large number of small decisions are made so that every detail of the design is fixed before commencing production.

During such a design process, designers use several techniques, methods and tools for problem understanding, communication, exploration, and decision-making. Examples of such techniques include: Theory of Inventive Problem Solving (TRIZ), Quality Function Deployment (QFD), functional reasoning, sketching and optimization. These techniques make use of "visual" representation of design information (or knowledge) such as the House of Quality (HoQ) for QFD, function-structure diagrams for functional reasoning, or Pareto plots in optimization to focus the designer's attention on a particular aspect of the design. Such visualizations not only represent information about the design, but also the formulation of the simplified design problem being solved. According to Ullman [9], such simple design problems can be classified as (a) selection design, (b) configuration design, (c) parametric design, (d) redesign, and (e) original design. Computational approaches such as case-based reasoning, function-means (F/M) synthesis, simulation-based design and techniques based on constraint processing have been developed for modeling and supporting such problem-solving activities in early phases of the design.

Through an informal survey of several practicing engineers who were members of the Purdue Product Lifecycle Management (PLM) Center, we observed that these tools are not being used together during early design in a computational setting. Although interoperability issues among Computer-Aided Design (CAD) and Computer-Aided Engineering (CAE) tools are well known in PLM literature (see for example, [10, 11]), connections among "visual" tools and design problems are not available to the best of our knowledge.

In practice, design is an iterative process where these tools are used multiple times during a single design. More importantly, each of the design support tools mentioned above deals with a specific subset of the design problem, at a specific level of abstraction. Designers take decisions based on

* Author of correspondence; Phone: (765) 494-5725, Fax (765) 494-0539, Email: ramani@purdue.edu

information presented through such visual tools. Stjernfelt argues that in design "…the fact that the diagram displays the interrelation between the parts of the object it depicts is what facilitates its use in reasoning and thought processes, thus taking the concept of sign far from the idea of simple coding and decoding and to the epistemological issues of the acquisition of knowledge through signs" [13].

Our goal is to study the connection of information among different visual tools in the design process as well as between known problem formulations. This involves using a representation that accommodates different levels of abstraction of the various aspects of design. In the current work, we focus on the early stages of mechanical design process, i.e. from specification development until embodiment design. The contents of the visual tool (or the design problem) expressed using the elements of a single design model provides a mechanism for connecting such tools.

In this paper, we extend the NIST Core Product Model (CPM) [10] to handle additional aspects such as numerical constraints and mappings between descriptions of different abstractions, to introduce the working knowledge model (WKM). We describe the meaning of working knowledge, as we define it, in Section 2. Section 3 provides a brief overview of some of the visual tools used in design. A review of existing design knowledge and information models is provided in Section 4. A small subset of concepts described in existing literature is considered within WKM (Section 5). In Section 5 we also map the elements of the WKM onto the elements of the visual tools so as to facilitate bidirectional exchange of data between a design tool and WKM. Section 6 describes a prototype implementation where a Wiki is used in conjunction with WKM to capture, store and present design information visually; an example of a coolant valve design to illustrate the use of WKM along with visual tools is also described in Section 6. Discussion and conclusion are presented in Sections 7 and 8 respectively.

2 WORKING KNOWLEDGE IN DESIGN

In general, there are two common views about the nature of knowledge in general: structural view- knowledge as the content of a representation, and functional-knowledge as the capability to solve problems [14]. Restricting to design, Klein [15] identifies three main categories of design knowledge: (1) general domain knowledge, (2) case-specific object level knowledge, and (3) problem solving & control knowledge. In this view, general domain knowledge includes knowledge describing the relations between function, behavior, structure; case-specific knowledge consists of requirements and (possibly partial) design descriptions. During the design process, the domain knowledge of the designer engineers can be assumed to change little in comparison to case-specific knowledge [15]. In this work, we further restrict ourselves the "working" aspect of the design knowledge, i.e., only that which is known about the current design and can be expressed explicitly.

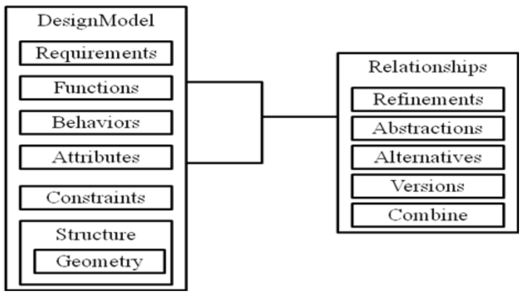


Figure 1. Concepts and relationships in the working knowledge model.

Working knowledge, as we define it, is the collection of design representations and their relationships accumulated from the commencement of the design process. Specifically, working knowledge consists of (Figure 1):

(1) Knowledge about requirements, objectives, and constraints that the design should satisfy. Requirement is a documented need that should be satisfied by the product. Objectives are the set of wants that the product should excel at. Constraints are restrictions on the product.

(2) Knowledge about function, form and behavior of the product being designed. Function is the activity or task that the design should perform. Behavior is the activity or task that the product performs in reality. Form is the structure of the product and may include geometry.

(3) Relationship between representations:

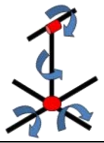
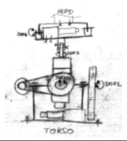
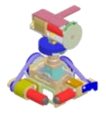
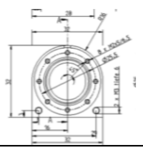
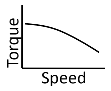
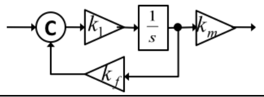
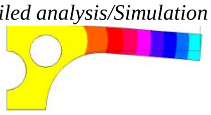
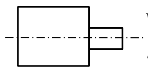
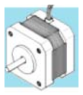
(a) The alternatives of entities that exist at each stage in the design process (expressed explicitly by the designer). For example, a function can be realized by several alternative embodiments; alternative requirements may be available at the initial stages of design; or even alternate representations may be available to describe a product's geometry. Additionally, alternative designs may be carried forward simultaneously during the design process.

(b) Representation of these entities in different levels of abstraction [16]. Examples for the different levels of abstraction for constraints, behavior, form and function are shown in Table 1. For example, relationships among parameters can be represented qualitatively (as in the roof of the house of quality, for example) or as analytical equations or using simulation models or even surrogate-models generated using simulation data. Similarly, geometry can be described as sketches, 2D drawings, or 3D models.

(c) And the sequence of changes (edits) made to the design, typically called version information.

Entities such as requirements, functions, from, and behavior can be captured using the Core Product Model (CPM) [10]. Additionally Product Data Management (PDM) systems capture version information. Even though design process is a refinement of abstract representations even for a particular design description such as behavior [9, 17], existing models do not capture the relationships among abstractions. In this paper, we extend the classes in CPM to capture relationship information; these relationships are useful when design information is accessed for exploration using visual tools.

Table 1. Levels of abstraction of design entities.

DESIGN ENTITY	LEVEL OF ABSTRACTION: ABSTRACT				CONCRETE
Geometry	*Line diagrams* 	*Rough sketches*	*2D Drawings* 	*3D CAD Models* 	*Detailed manufacturing drawings* 
Function/ Behavior	*Qualitative descriptions* “converts electrical energy to mechanical energy”		*Charts and tables* 		*Detailed Simulation* 
Constraint / Analytical relationship	*Qualitative relationships* $l \xrightarrow{+} \delta$& $h \xrightarrow{-}$ (read as δ decreases monotonically with h)		*Simple analytical relationships* $\delta = \frac{Wl^2}{8Eh}$	*Surrogate models* Response surfaces, Neural networks, etc.	*Detailed analysis/Simulation* 
Attributes/ Parameters	*Qualitative* [V] = +	*Order of magnitude* V~10^1	*Fuzzy* medium V	*Intervals* $V \in [10,15]$	*Quantitative* V=12
Objective	*Qualitative words* Should be responsive				*Quantitative* $\max V$
Material	*Generic class* Metal		*Material class* Aluminum		*Specific material* AlZnCuMg1.5
Design Description	*Solution principle* “Convert electrical energy to mechanical energy” 		*Concept* A “motor”  $\tau = f(V,s)$ $V = 12$ $s \in [0,1000]$ …		*Individual artifact* A specific stepper motor model “NEMA 17 S91” 

3 VISUAL TOOLS USED IN EARLY DESIGN

An important purpose of the working knowledge model is to enable iterative early design and acquire design knowledge through means of various visual tools. Table 2 (adapted from Hubka and Eder [18]) lists the different visual representations that are used in early design. In this work, we have implemented the following visual tools presented in [9,18,19]: Function-Structure diagrams, Morphological Matrix, House of Quality and a tool to author 2D drawings. Apart from these, we also include the SysML requirements and parametric diagrams due of a lack of equivalent visual representations for requirements and parametric equations.

Quality Function Deployment (QFD) [7] has become a widely accepted method for engineering design in industry and software programs for computer support of this method are available. The main visual model of the QFD method, the House of Quality (HoQ) [20] can be used very flexibly and it can capture various aspects of the working knowledge. The version of the HoQ that is used often relates customer requirements (“Voice of the customer”) in the interrelations matrix to the engineering characteristics. HoQ can also be used to model objectives, constraints, a relative ranking of requirements, and qualitative relationships among engineering characteristics and compare alternative or competing designs. In the version of HoQ presented in [9], engineering characteristics are quantitative properties of the design, i.e. the parameters of that design.

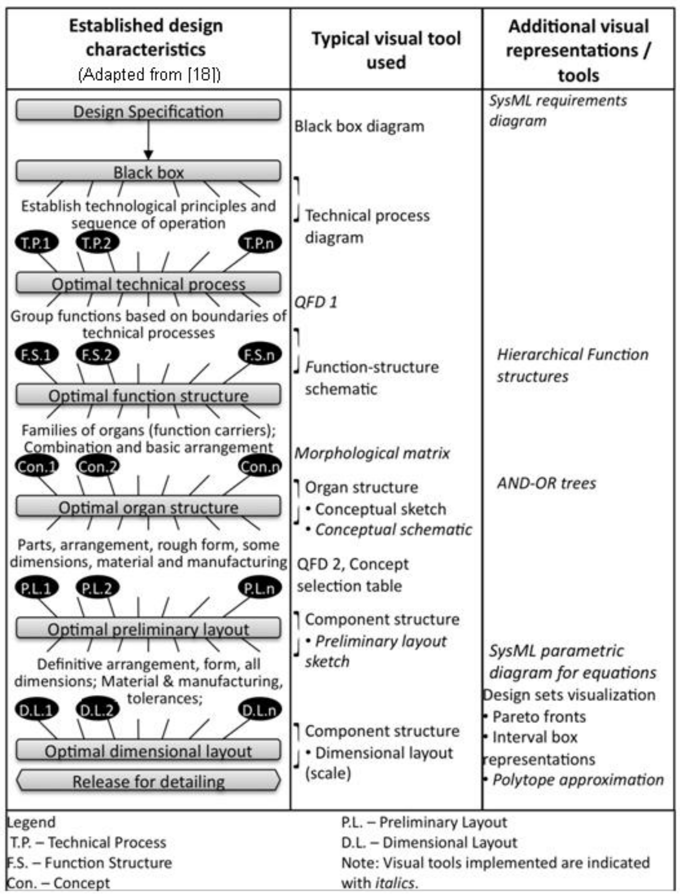
Knowledge about Function is often visualized in hierarchical or procedural function structure diagrams [3]. The hierarchical function structure diagram displays the functions that are to be performed by the system along with the sub-functions that are needed to achieve the overall function. The procedural function structure diagram is used to describe the sequence in which these functions are performed and the flow associated with each sub-function.

The morphological matrix [18, 19, 59] is a systematic method to combine solution principles based on the function decomposition. Here, the several possible “means” or “solution principles” are listed by the designer for each function that the system needs to perform. This work assumes that the designer provides the necessary information although it is conceivable that the various “means” or “solution principles” are populated from a database based on the function similar to the approach in [21] and [22].

Computer based tools for modeling product geometries deal with entities (like lines, arcs, surfaces etc.) and constraints (such as tangency, collinear, angle etc.) between these entities.

The Systems Modeling Language SysML [23] is a graphical modeling language that was developed to support systems engineering processes. The SysML Requirement Diagram is used for representing the relationships among various requirements for the design.

Table 2. Visual tools used during the design process.


Established design characteristics (Adapted from [18])
Typical visual tool used
Additional visual representations / tools
Design Specification
Black box diagram
SysML requirements diagram
Black box
Establish technological principles and sequence of operation
Technical process diagram
T.P.1
T.P.2
T.P.n
Optimal technical process
QFD 1
Group functions based on boundaries of technical processes
Function-structure schematic
Hierarchical Function structures
F.S.1
F.S.2
F.S.n
Optimal function structure
Families of organs (function carriers); Combination and basic arrangement
Morphological matrix
Con.1
Con.2
Con.n
AND-OR trees
Organ structure
• Conceptual sketch
• Conceptual schematic
Optimal organ structure
Parts, arrangement, rough form, some dimensions, material and manufacturing
QFD 2, Concept selection table
P.L.1
P.L.2
P.L.n
Component structure
• Preliminary layout sketch
Optimal preliminary layout
Definitive arrangement, form, all dimensions; Material & manufacturing, tolerances;
SysML parametric diagram for equations
Design sets visualization
• Pareto fronts
• Interval box representations
• Polytope approximation
D.L.1
D.L.2
D.L.n
Component structure
• Dimensional layout (scale)
Optimal dimensional layout
Release for detailing
Legend
T.P. – Technical Process
F.S. – Function Structure
Con. – Concept
P.L. – Preliminary Layout
D.L. – Dimensional Layout
Note: Visual tools implemented are indicated with italics.

4 BRIEF REVIEW OF COMPUTER REPRESENTATIONS OF DESIGN INFORMATION

Several models have been developed for capturing design information and knowledge as well as for supporting various types of problems encountered during design. It is not the intent of this paper to provide an exhaustive survey of such models but to provide a sampling of such representations. We restrict our scope to models represent the design information for documentation, knowledge capture and search. The working knowledge model is extended from one such representation.

Research in analogy based design synthesis support, such as those based on case-based reasoning (see for example, [24]) are the earliest efforts in developing product knowledge models. Notably, the models presented in KRITIK [25] and KRITIK2 [26], use Structure-Behavior-Function (SBF) formalism to represent the structure of the product, its behavior based on the structure and the function achieved through those behaviors. Another case-based-design tool CADAT [4] uses an AND/OR tree representation of design to support design decomposition. A comprehensive review of such systems can be found in [27]. Design catalogs have also been proposed (see for example, [28]) that contain objects, solutions or operations, allowing designers to explore variants [29]. Active catalogs (see for example, [30]) have been developed for selecting and evaluating electromechanical components and sub-systems.

In the context of information modeling in design, researchers at National Institute for Standards and Technology (NIST) provide the most comprehensive modeling framework up to date for representing 'Artifacts'. Sudarsan et al. [31] present the Core Product Model (CPM) and its extensions such as the Open Assembly Model (OAM) and Design-Analysis Integration Model (DAIM). The core product model focuses on artifact representation including function, form, behavior and material, physical and function decomposition, and relationships among these concepts [10]. The CPM also allows modeling other aspects such as rationale, requirements and product families. Xue et al. [32] introduce the concept of 'worlds' in capturing the operations performed on the product knowledge model during the design process as a part of an evolutionary design database. CPM forms the underlying product knowledge model, but is extended to include (1) arithmetic and temporal constraints, (2) tasks, and (3) attribute. The design repository hosted at Oregon State University [12] is based on a model that is similar to CPM but provides a documented interface to enter and retrieve product information using web forms or external applications. The ISO 10303 standard [33], also known as STEP, provides a detailed mechanism for specifying product data throughout the lifecycle in a implementation neutral fashion to enable file and data exchange between various applications used ding the design process such as Computer Aided Design, Computer Aided Manufacturing and Product Data Management systems.

Models and modeling languages have been developed for formally specifying the knowledge associated with systems design such as the Systems Modeling Language (SysML), which is based on the Unified Modeling Language (UML) [34, 35]. Apart from modifying existing diagrams from UML (Activity, Block Definition and Internal Block Diagrams) SysML also introduces new diagrams to model systems knowledge (Requirements and Parametric diagrams). Although SysML is powerful in expressing the result of a design activity, it does suffer from several weaknesses [36], most importantly lacking the ability to capture interconnections among diagrams.

Klein [15] presents a knowledge model as a part of the Methodology and tools Oriented to Knowledge-based engineering Applications (MOKA) that also uses UML-like formalisms to allow designers document explicit design knowledge. The MOKA language supports multiple views of the design such as geometric view, structural-view and kinematics-view. MOKA provides the ability to model constraints, illustrations, activity, rules, entities as well as products, design process and rationale. Like SysML, MOKA is a diagrammatic representation available to a knowledge engineer to express explicit design knowledge. They are not connected to existing visual representations used in design practice.

Tamburuni [41] presents an Analyzable Product Model (APM) that provides semantics for mapping geometry definitions and a generic description of analysis models. Depending upon the application, the model allows selection of the appropriate analysis model to support design; also, changes to the form can be propagated to analysis models that are coupled in order to maintain consistency within the design.

In the context of configuration design of complex products, Feldkamp et al. [39] have proposed a method and software tool, called System Design for Reuse (SyDeR), that combines structural model, a taxonomy based library of solutions, and, constraint propagation techniques. The notion of ports is used to encapsulate and model the hierarchical nature of the system being designed, and importantly, to keep track of alternatives solutions available for configuration.

There are several specialized models (such as for generative design) and other abstract representations (such as MOKA and SysML) for modeling design knowledge. On one end specialized models allow computational support but the concepts used in such models are not shared among other models; whereas on the other end, abstract languages may allow rich representations at the cost of useful computational support.

Even among somewhat generic models, there is ambiguity in definition of the modeling concepts. For example in the NIST Core Product Model [31, 32], on which the WKM is based, Artifact represents the actual artifact being designed. An artifact is composed of other artifacts leading to a natural extension of an assembly model. Although this modeling approach is sufficient for describing the final outcome of the design process, or, an intermediate description of a single design that is created as a succession of design operations, several representations are used during design to describe the same artifact. There is no clear definition of an artifact in literature when early stages of design are considered. That is, is solution principle [3], or scheme [2], or concept [21], or configuration [39], an artifact? They all describe the design at different levels of abstraction, but identifying a given description as an idea or concept or embodiment is subjective in nature. In this work, we overcome the ambiguous meaning by creating the notions of Design Model and Instance (both are extensions of the Artifact class in CPM). A design model can represent the design at any stage in the design process whereas an instance is a notional representation that is assembled from information available in the lowest level of abstraction, i.e. as precise as possible.

Maeda et al. [59] propose a framework to organize and acquire knowledge using visual representations. This work (similar to Electronic Design Notebook [60]) parses an arbitrary graph drawn by the designer expressing design knowledge in the form of decision tables and decision trees. Reich [14] presents a Computational QFD (CQFD) tool that uses a graph-based modeling environment to allow users to create diagrams such as affinity diagram, relation diagram and House of Quality; the graph model underneath these diagrams is used to maintain consistency between them. As far as we are aware, Design Scribe [60] is the only work that attempts to integrate some of the visual tools used in mechanical design; even in this case, a very specialized model is used to capture rationale using the House of Quality. Compared to these and similar approaches, the objective of this work is to enable sharing of design information (say, in different levels of abstraction) between the visual tools.

5 INFORMATION MODEL

5.1 Basic design elements in the model

Figures 2 and 3 show the elements of the working

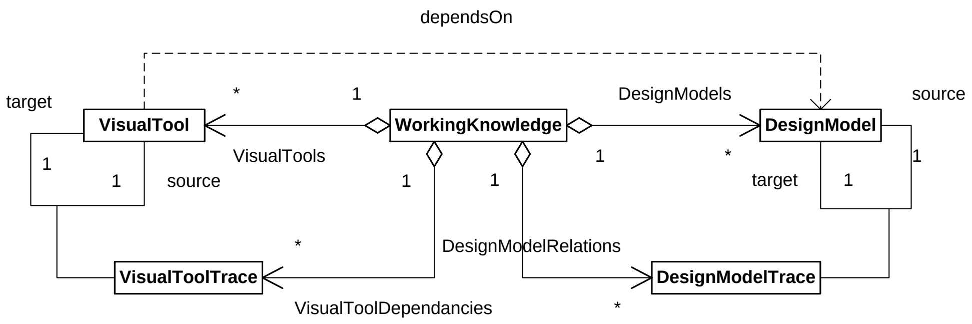


Figure 2. UML representation of the working knowledge model.

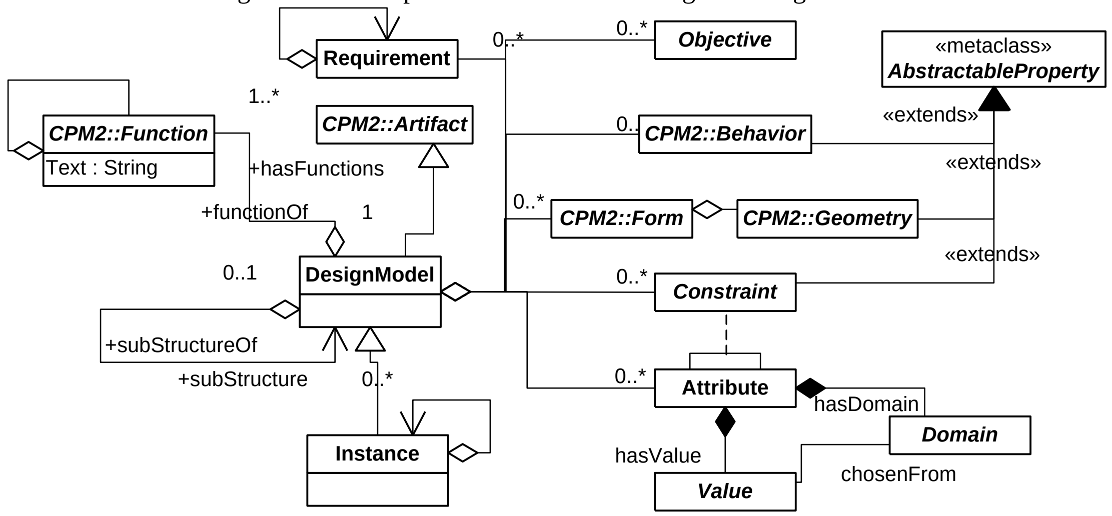


Figure 3. UML representation of DesignModel (Classes from CPM2 are also shown).

knowledge that are considered in this implementation along with the relevant classes from the CPM2 (the second version of CPM [44]).

A Design Model is the primary concept of the working knowledge model. A DesignModel represents the product being designed at a particular point in time in an appropriate abstraction. More specifically, a DesignModel is a placeholder for the final Instance of the product.

The dichotomy between a DesignModel and Instance is useful when configuration design is being performed. For example, in design of an industrial truss where the individual beams are selected from a catalog as shown in Figure 4, the DesignModel of the structure consists of other DesignModels for each beam and for each connector. The DesignModel for the beam can be instantiated using any of the pre-defined sections available in the catalog. Similarly, each DesignModel of the connector can be instantiated using those available in the catalog, each of which is an Instance. An instance of the overall DesignModel (i.e., the industrial structure) is created once an appropriate instance for each "beam" and "connector" is selected.

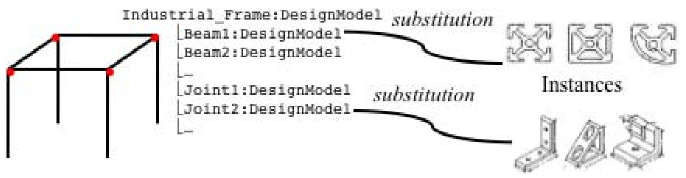


Figure 4: An example to illustrate the "substitute" relationship (Adapted from [59]).

Apart from catalog components, however, the distinction between a DesignModel and Instance cannot be clearly established. Moreover, several descriptions of the same design can be generated during product development depending upon the intended use as discussed in Section 4.3; manufacturing drawings, for example, are generated based on the designed geometry and provide additional process related information. To overcome the ambiguous definition, we consider an Instance to be a notional idea that is assembled from information available in the lowest level of abstraction (i.e., as precise as possible).

Structurally, a DesignModel (and Instance) can be composed of other DesignModels or Instances; Instances can only be composed of other Instances as illustrated in Figure 4. A DesignModel also consists of Attributes, Constraints, Geometries and Objectives. A DesignModel is associated with Requirements, Functions and Behaviors.

A design model consists of several Attributes that describe the model. Attribute describes the characteristic property of an artifact [45]. Parameters[1] form an important specialization of an attribute that takes numerical values (Figure 5). "Color" is an example for Attribute while "Cost", "Weight" and "Length" are examples for Parameter (which is a sub-class of Attribute). A Domain can be defined for a Parameter from which a Value is chosen. The set consisting of "blue", "red", "green", and "white" is an example for categorical domain of the attribute 'Color'; [-1.5,-1]U[1,1.5] is an example for IntervalDomain which is a union of AtomicDomains. Objectives are association of an Attribute along with a qualifier (min, max or target). A Parameter-Value pair can be designated as a Target.

Requirements represent the explicit conditions that final design should satisfy and are stated by the stakeholders in the design. These stakeholders include customers (e.g. 'should provide a particular functionality'), end users, manufacturers (e.g. 'should be manufacturable with the available resources') and design engineers themselves. Although requirements are described textually, we assume that the designer translates them into functional requirements, constraints and objectives during the course of the design process.

Constraints represent the computer-verifiable expressions associated with the design. Weilinga [46] differentiates constraints from requirements based on the "tone", i.e., positive or negative. Requirements are associated with a positive tone that represents what is needed from a design. Constraints, on the other hand, are associated with a negative tone and represent what is possible with the design. Constraints, in general, restrict the choice available to the designer [47]. However, in this work we do not use "tone" to differentiate between constraint and requirement because of its subjective nature. Instead, we use 'the ability of a computer system to interpret' as the criteria; this is because any constraint (in the sense of ref. [46]) that is not traced to the customer can be ascribed to requirements from any of the stakeholders including designers and manufacturers. For example, even constraints arising out of physical phenomenon, such as 'stress<yield_strength' can be traced to a requirement 'should not fail'. In other words, constraints can also be described as the mathematical translation of stakeholder requirements. Qualitative, Analytical and Geometric constraints can be represented in the current version of WKM (Figure 6). QualitativeConstraint in WKM captures positive and negative relationships between attributes (including parameters); AnalyticalConstraint captures algebraic and numerical relationships between parameters. GeometricConstraint captures the relationships between geometric entities in a drawing.

A design model can be associated with a collection of Geometry descriptions. In this version of the working knowledge model, geometry can be described as a Sketch, 2D Drawings or 3D_Models. Each 2D drawing consists of geometric entities (Points, Lines, Arcs, Circles, etc.) and geometric relationships between these entities. These geometric relationships may involve Parameters. The Sketch and 3D_Model classes in this implementation of WKM merely store a link to an external file.

5.2 Capturing relationships in WKM

The current version of WKM accommodates five types of operations with the design elements presented earlier. These are: (1) Edit, (2) Combine, (3) Set-up substitution relationships, (4) Refine, and its inverse, (5) Abstract.

[1] There is confusion in literature as to what the difference between a variable and a parameter is [46]. Variable seems to be a problem specific term, whereas a parameter can denote any numerical attribute of a product.

1. Editing, the simplest operation, involves changing the design model by adding, removing and modifying its constituent elements. For example, the designer may add a new requirement, change a constraint or remove a geometric feature from the design.

2. Combine is the operation where portions of more than one design models are used to create a new design model, such as those seen in conceptual design activities. For example, the best aspects of concept A and concept B can be combined to create a new concept C that is significantly different from both A and B. The combine operation is not illustrated in the example described in this paper.

3. Setting up substitution relationships involves enumerating an equivalence class of objects and identifying common elements among the members of that set, possibly by means of an exemplar. The equivalence class then signifies the alternatives based on relevant criteria. At a later stage in the design process, when a substitute is chosen, Edit operation is used to create a new design model using that substitution. For example several ‘Means’ that are specified for a particular function in a morphological matrix form an equivalence class. The set of individual motors in a catalog and a common geometry model are another example for the equivalence class and their exemplar. Yet another example can involve descriptions of a set of designs using ‘ports’ such as models presented in [39] and [41]. Equivalence among design element descriptions is also captured in WKM. For example, a geometric constraint can be converted into an analytical relationship to be solved by a numerical solver. Extending the idea further (although not implemented here), different formats for representing 3D models can be thought of as another equivalence class. Visual tools such as Morphological Matrices can create the equivalence class for substitution where each combination of means results in a different DesignModel; all these DesignModels are potential designs that could satisfy the requirements.

4. Refinement is the operation where an abstract representation is made more concrete. For example, the designer may refine a 2D drawing into a 3D CAD model. Although each class in WKM can be refined, DesignModel and Requirement are the important classes that are refined. For example, a DesignModel for a “Flow Control Valve” can be refined to a “Solenoid valve” which is further refined to a “12V DC Solenoid operated spool valve”. Similarly, a behavior model involving simple algebraic equations can be refined into a finite element analysis model.

5. Abstraction operation is the inverse of refinement. This operation is necessary for iterative design exploration where a design model (concept or idea etc.) is elaborated and the knowledge gained is taken back as abstractions and commence the next iteration. Although abstracting design models is a manual activity, some of the abstraction operations such as converting an analytical model to a quantitative model can be automated.

The Trace relationship defined in CPM2 is extended in WKM for capturing the above relationships. We further extend the Trace association into DesignModelTrace, GeometryTrace, ConstraintTrace, AttributeTrace, RequirementTrace, ObjectiveTrace corresponding to each class in WKM. Geometry, constraint and attribute traces consist of the type of operation and a set of <source,target> pairs indicating that the target is obtained from the source entity after the relevant operation is performed. A DesignModelTrace is composed of GeometryTrace, ConstraintTrace and AttributeTrace mimicking the composition of the DesignModel class.

Abstraction is supported in WKM by two metaclasses: AbstractableProperty, and AbstractionCreator. An AbstractableProperty signifies any of the following classes and their sub-classes: Behavior, Constraint and Geometry. For example, complex analysis and simulation models may be abstracted as analytical relationships using response surface approximations; analytical relationships can be abstracted further as qualitative relationships [36]. Figure 6 shows the different abstractions that are considered in the present implementation of the WKM.

5.3 Connecting visual tools to WKM

The elements of the HoQ can be mapped to WKM classes as shown in Figure 7. For example, engineering characteristics (as defined in [9]) correspond to Parameters in WKM. The parameters in the design are candidates for being included in

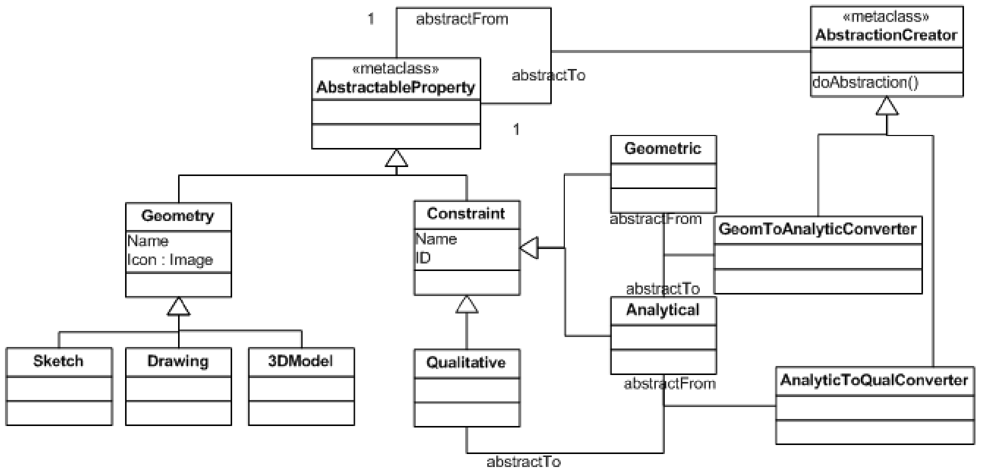


Figure 6. Abstractions in WKM (abstractions of Behavior is not shown but is similar to Constraint).

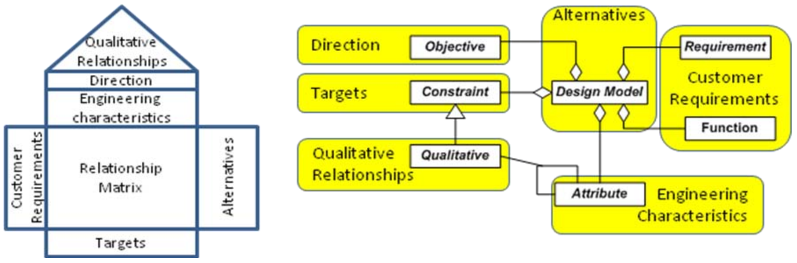


Figure 7. A typical HoQ (from [9]) and corresponding classes in WKM; the contents of the Relationship matrix are *Trace* relationships from Requirements to Constraints and Objectives (not shown).

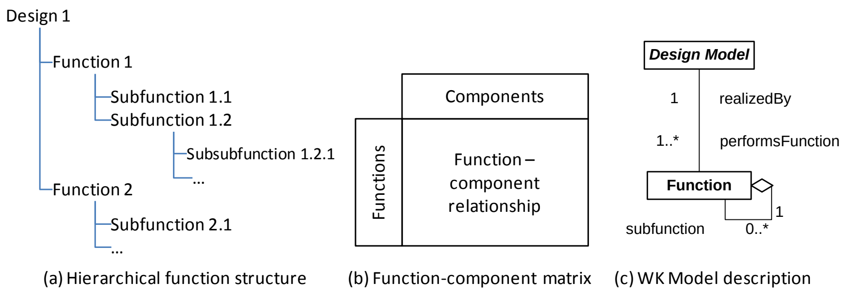


Figure 8. Model of hierarchical function structure and function component matrix using WKM elements.

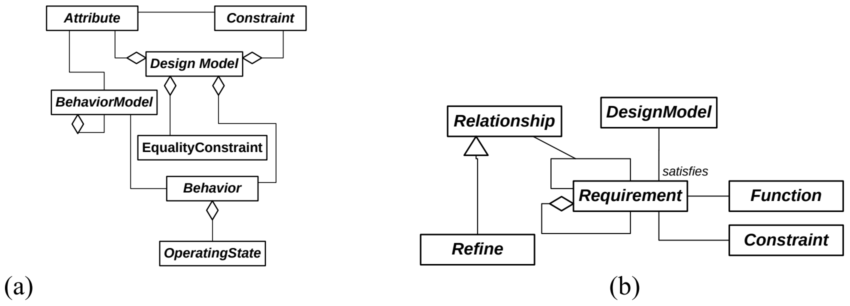


Figure 9. Partial descriptions of SysML (a) Parametric diagram and (b) Requirements diagram using WKM elements.

the HoQ as ECs; the ECs that are added to the HoQ are then added to the WKM as Parameters. The Constraints and Targets are set on Attributes within the HoQ. The WKM can be edited though the HoQ using these mappings.

Figure 8 describes the function structure diagram and the morphological matrix using WKM elements. Each Function can be hierarchically decomposed into its sub-function and a DesignModel can perform many functions.

Figure 9 describes the content of the SysML Parametric and Requirements diagrams using the elements of the WKM. The XML schema for representing SysML models is described in the STEP application protocol AP233 [48]. Parametric diagram can be used to describe Attributes and mathematical relations between these attributes that form the (equality) Constraints within WKM. Internal Block Diagrams are similar to class diagrams in UML2 [49] and are used to represent the structure of DesignModel.

The geometry of a DesignModel is described by a collection of drawings. A geometric parameter can be shared between multiple drawings. A Drawing is a collection of geometric entities, variable & constraints defined between these entities. For example, points, lines, arcs, and circles are geometric entities. The geometric constraints such as coincidence, parallel, perpendicular, fix and angle determine the topology of the drawing.

6 IMPLEMENTATION AND CASE-STUDY EXAMPLE

A wiki-based prototype was implemented to evaluate the working knowledge model. Wikis are freely-editable collections of web pages, exhibiting potential for a flexible documentation and communication tool for collaborative design tasks as well as support for team design thinking early in the design process [53]. Wikis store versions of each page along with the edits made to the page. They also allow the user to store, retrieve and manage versions of files. The content of a wiki page is stored in a database and each page can be created dynamically from its content. A wiki installation also includes a parser that can be extended for custom applications such as the current prototype named “DesignWiki”. The WMK classes were translated as database tables and stored in a MySQL database [63] that is shared with the wiki.

The role of the wiki in the prototype is to document known facts and assumptions whereas visual tools embedded within the wiki page support exploration. The annotated text in the wiki page is parsed to populate the WKM data model that is accessed by any visual tool used in the design; any new knowledge generated or added through the visual tool updates the WKM and creates a new version of the wiki page that reflects the change. This process is explained below.

Figure 10 shows the schematic of the prototype implementation. The customized media-wiki [63] serves at the documentation front-end for design knowledge. The user can annotate the wiki page with WKM classes that is then parsed and stored in the database. The WKM data is accessed and edited using SQL commands by visual tools and other wiki pages. Visual tools were implemented as JAVA applets and can be embedded within each wiki page using special extensions to the media-wiki syntax. The visual tools embedded in the wiki page can also be used to modify WKM objects. Depending upon the operation performed within the visual tool, the user may create new WKM objects or edit information contain within these objects. New versions for wiki pages referencing these objects are created automatically to reflect changed made using the visual tool. Wikis, by default, provide versioning ability and the built-in version management system is exploited in this implementation to support "edit" operations discussed in section 5.2.

The following visual tools have been implemented in JAVA as applets and can be embedded within any wiki-page corresponding to a DesignModel: (1) House of Quality, (2) Hierarchical function diagram, (3) Morphological Matrix, and (4) A tool that displays analytical relationships between parameters of the design in a manner similar to the SysML Parametric diagram.

Apart from the above tools, a simple editor was also used to create 2D drawings; Topcased [50], an open source SysML authoring tool was used to view and edit SysML requirement diagrams. A commercial tool Isight [58] was used to create abstractions of analytical models from simulation codes. The contents of the drawing file is also parsed to identify geometric constraints; these constraints are automatically converted into analytical constrains by another tool. The file-management ability of the wiki was used to store, parse, generate and retrieve the files needed for these external tools.

Each page in the wiki can represent a WKM class and a wiki page can reference WKM class objects within the text. Table 5 lists a limited set of tags and attributes corresponding to each class in WKM. These tags are used to both define as well as reference WKM objects. Tags can also be nested within each other.

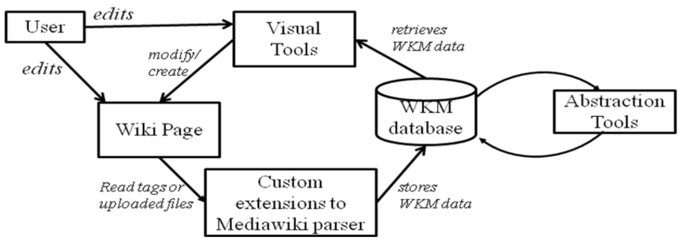
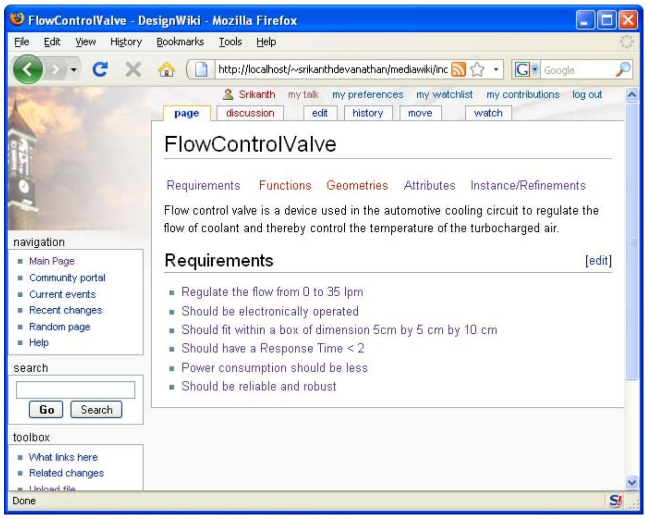


Figure 10. The system uses the mappings defined in section 5.3 to populate the visual tools and transfer the decisions back to the working knowledge model.

```
<WKMClass type="DesignModel" name="FlowControlValve"/>
Flow control valve is a device used in the automotive
cooling circuit to regulate the flow of coolant and
thereby control the temperature of the turbocharged
air.
==Requirements==
* <Requirement name="Regulate coolant flow"
type="functional">Regulate the flow from 0 to 35 lpm
</Requirement>
* <Requirement name="Electronic operation">Should be
electronically operated</Requirement>
* <Requirement name="Fit in box">Should fit within a
box of dimension 5cm by 5 cm by 10 cm</Requirement>
* <Requirement name="Response time">Should have a
<Constraint> <Parameter>Response Time</Parameter> < 2
</Constraint> <Requirement>
* <Requirement name="Less Power consumption"> Power
consumption should be less</Requirement>
* <Requirement name="Reliable and robust"> Should be
reliable and robust </Requirement>
```


FlowControlValve - DesignWiki - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://localhost/~srikanthdevanathan/mediawiki/inc
Google
Srikanth my talk my preferences my watchlist my contributions log out
page discussion edit history move watch
navigation
Main Page
Community portal
Current events
Recent changes
Random page
Help
search
Go Search
toolbox
What links here
Related changes
Upload file
Done

FlowControlValve

Requirements Functions Geometries Attributes Instance/Refinements

Flow control valve is a device used in the automotive cooling circuit to regulate the flow of coolant and thereby control the temperature of the turbocharged air.

Requirements [edit]

- Regulate the flow from 0 to 35 lpm
- Should be electronically operated
- Should fit within a box of dimension 5cm by 5 cm by 10 cm
- Should have a Response Time < 2
- Power consumption should be less
- Should be reliable and robust

Figure 12. An example of creating the wiki-page for a "Flow Control Valve".

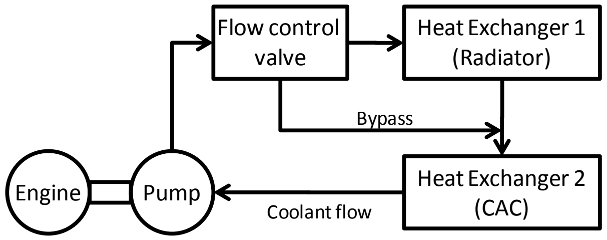
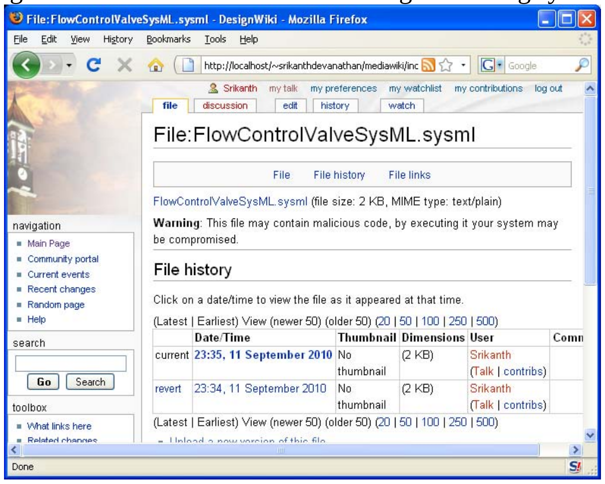


Figure 11. Schematic of automotive engine cooling system.


File:FlowControlValveSysML.sysml - DesignWiki - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://localhost/~srikanthdevanathan/mediawiki/inc
Google
Srikanth my talk my preferences my watchlist my contributions log out
file discussion edit history watch
navigation
Main Page
Community portal
Current events
Recent changes
Random page
Help
search
Go Search
toolbox
What links here
Related changes
Done

File:FlowControlValveSysML.sysml

File File history File links

FlowControlValveSysML.sysml (file size: 2 KB, MIME type: text/plain)

Warning: This file may contain malicious code, by executing it your system may be compromised.

File history

Click on a date/time to view the file as it appeared at that time.

(Latest | Earliest) View (newer 50) (older 50) (20 | 50 | 100 | 250 | 500)

	Date/Time	Thumbnail	Dimensions	User	Comm
current	23:35, 11 September 2010	No thumbnail	(2 KB)	Srikanth (Talk | contribs)	
revert	23:34, 11 September 2010	No thumbnail	(2 KB)	Srikanth (Talk | contribs)	

(Latest | Earliest) View (newer 50) (older 50) (20 | 50 | 100 | 250 | 500)

Figure 13. The SysML drawing is generated automatically and be downloaded from the wiki. The modified SysML file when uploaded is parsed and WKM is updated.

We illustrate the use of WKM and the wiki in supporting design through visual tools by means of a flow-control valve that is used in an automotive cooling circuit. Figure 11 shows a schematic of automotive engine cooling system. A centrifugal pump driven by the engine circulates the coolant fluid through the heat exchanger called Charge Air Cooler (CAC) where the heat from the supercharged air is transferred to the coolant. The hot coolant is passed through a second heat exchanger (radiator) where the heat from the coolant is transferred to the environment. Varying the rate coolant flowing through the radiator controls the temperature of the supercharged air. The amount of coolant flowing through the radiator is varied by means of a flow control valve as shown. This valve is operated using a pulse-width modulated (PWM) signal. The system consists of four main sub-systems: 2-way valve, the radiator, the engine and the pump.

The design of the flow-control valve used in the cooling system follows a design process where the initial requirements are available in the form of a document. This document is used to create the initial wiki page as shown in Figure 12 a. The special tag
`<WKMClass type=”DesignModel” name=”FlowControlValve”>`
in line 1 indicates that this page describes a DesignModel called “FlowControlValve”. The text in lines 7-20 lists the annotated requirements on the flow control valve. Requirement “Response Time” in lines 14-16 illustrates how other tags such as “Constraint” and “Parameter” can be embedded within the “Requirement” tag. When the user completes editing this page, links to the list of attributes, functions, requirements, constraints, objectives, geometry, and the latest instance are added automatically to the header of the wiki-page as shown in Figure 12 b.

The user can view the requirements by clicking the “Requirements” link at top of the wiki page (Figure 12 b). This link opens a dynamically-generated page “FlowControlValveRequirements” that lists the requirements of the flow control valve and also allows the user to download the “sysml” and “sysmldi” files (Figure 13) that are used by Topcased (Figure 14). The user adds additional requirements (“Continuous operation”, “Tolerate contaminants” and “Easy to control”) using Topcased and uploads the newer version to the wiki. The system automatically parses the “sysml” file and updates the working knowledge model.

The user then revisits the “FlowContolValve” page and adds the following lines to the end of the text:
```
==House of Quality==
<visualtool type=”HOQ”/>
```
This command adds the House of Quality visual tool to the page. Figure 15 shows the screenshot of the wiki page after the user has added “Engineering Characteristics” i.e. Parameters to the model. Note that the newly added requirements are also listed in the HoQ. The user also sets targets or adds constraints to the parameters using this tool. Once the house of quality has been edited, a new version of the wiki page is automatically created by the visual tool to capture the edit operation. We note that the user has also set Objectives and Constraints on the design.

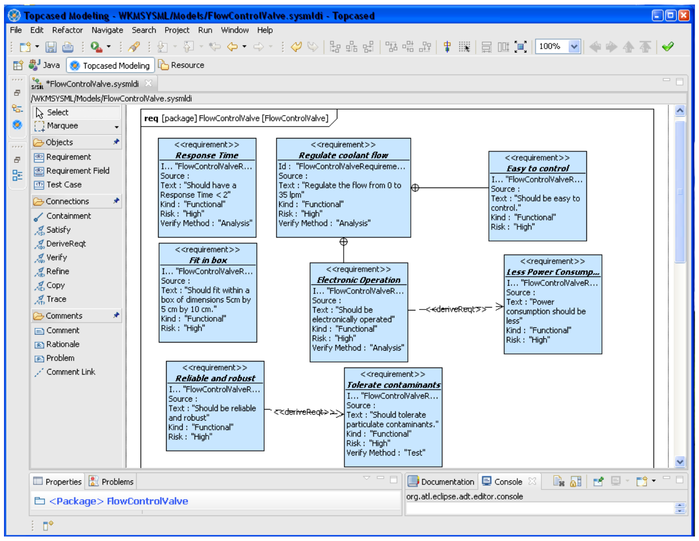
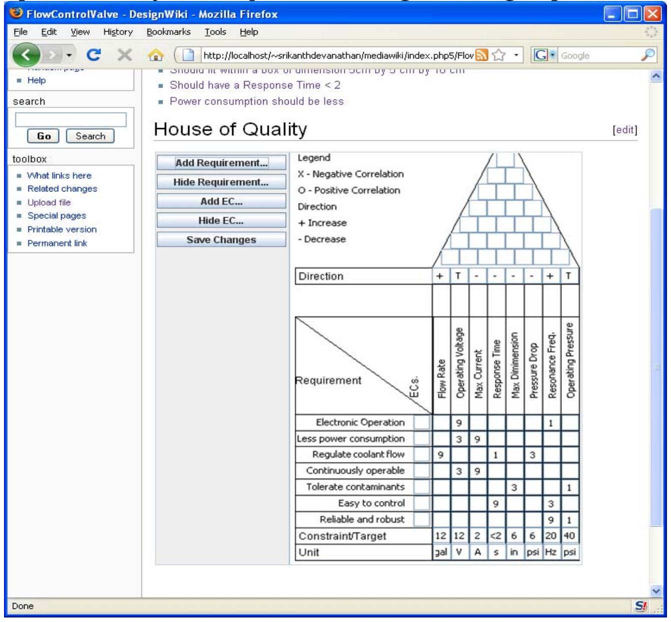

Topcased Modeling - WKMSYSML/Models/FlowControlValve.sysmldi - Topcased
File Edit Refactor Navigate Search Project Run Window Help
Java
Topcased Modeling
Resource
*FlowControlValve.sysmldi
/WKMSYSML/Models/FlowControlValve.sysmldi
Select
Marquee
Objects
Requirement
Requirement Field
Test Case
Connections
Containment
Satisfy
DeriveReqt
Verify
Refine
Copy
Trace
Comments
Comment
Rationale
Problem
Comment Link
req [package] FlowControlValve [FlowControlValve]
<<requirement>> Response Time
I... "FlowControlValveR...
Source :
Text : "Should have a Response Time < 2"
Kind : "Functional"
Risk : "High"
Verify Method : "Analysis"
<<requirement>> Regulate coolant flow
Id : "FlowControlValveRequireme...
Source :
Text : "Regulate the flow from 0 to 35 lpm"
Kind : "Functional"
Risk : "High"
Verify Method : "Analysis"
<<requirement>> Easy to control
I... "FlowControlValveR...
Source :
Text : "Should be easy to control."
Kind : "Functional"
Risk : "High"
<<requirement>> Fit in box
I... "FlowControlValveR...
Source :
Text : "Should fit within a box of dimensions 5cm by 5 cm by 10 cm."
Kind : "Functional"
Risk : "High"
<<requirement>> Electronic Operation
I... "FlowControlValveR...
Source :
Text : "Should be electronically operated"
Kind : "Functional"
Risk : "High"
Verify Method : "Analysis"
<<deriveReqt>>
<<requirement>> Less Power Consump...
I... "FlowControlValveR...
Source :
Text : "Power consumption should be less"
Kind : "Functional"
Risk : "High"
<<requirement>> Reliable and robust
I... "FlowControlValveR...
Source :
Text : "Should be reliable and robust"
Kind : "Functional"
Risk : "High"
<<deriveReqt>>
<<requirement>> Tolerate contaminants
I... "FlowControlValveR...
Source :
Text : "Should tolerate particulate contaminants."
Kind : "Functional"
Risk : "High"
Verify Method : "Test"
Properties
Problems
<Package> FlowControlValve
Documentation
Console
org.atl.eclipse.adt.editor.console

Figure 14. The requirements for the flow control valve are depicted as a SysML requirements diagram using Topcased.



- Should fit within a box of dimension 5cm by 5 cm by 10 cm
- Should have a Response Time < 2
- Power consumption should be less

House of Quality

Legend
X - Negative Correlation
O - Positive Correlation
Direction
+ Increase
- Decrease

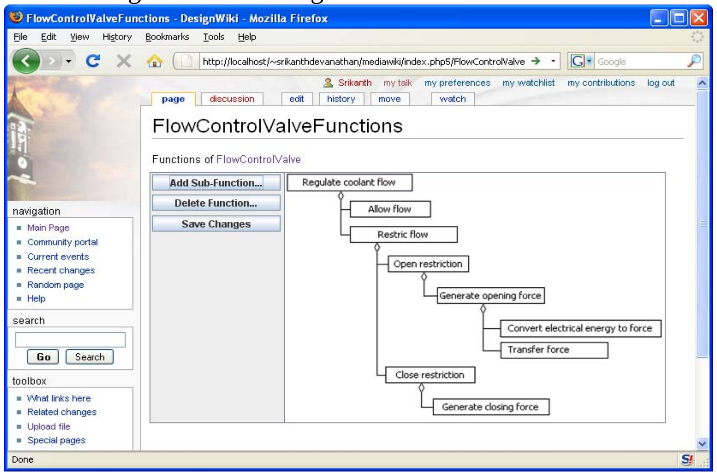
Direction	+	T	-	-	-	-	+	T
Requirement \ ECs.	Flow Rate	Operating Voltage	Max Current	Response Time	Max Dimimension	Pressure Drop	Resonance Freq.	Operating Pressure
Electronic Operation		9					1	
Less power consumption		3	9					
Regulate coolant flow	9			1		3		
Continuously operable		3	9					
Tolerate contaminants					3			1
Easy to control				9			3	
Reliable and robust							9	1
Constraint/Target	12	12	2	<2	6	6	20	40
Unit	gal	V	A	s	in	psi	Hz	psi

Figure 15. Screenshot of the HoQ tool.

The user then clicks the “Functions” link at the top of the wiki page (Figure 12 b). This link opens another dynamically generated page that lists the functions of the flow control valve and includes the Hierarchical Function Structure widget as shown in Figure 16. The user manually adds the sub functions such as “Allow Flow”, “Restrict Flow” and “Generate opening force” as shown in Figure 16. Similar to the house of quality tool, the hierarchical function structure tool also creates a new version of the wiki page “FlowControlValveFunctions” to capture this edit operation.

Upon re-visiting the “FlowControlValve” page, the user adds the following lines to the end of the text:
```
==Morphological matrix==
<visualtool type=”MorphologicalMatrix”/>
```
This command adds the Morphological matrix visual tool to the page. Figure 17 shows the screenshot of the wiki page after the user has manually added the means of a “Solenoid”, “Motor”, “Spool

Valve", "Butterfly Valve" and "Spring". These DesignModels were created separately in the wiki with their sketch and analytical behavior model (equations). The Morphological matrix tool only shows the sketch and name of these "means". The user creates two DesignModels using the Morphological Matrix, one called "SolenoidValve" (that uses a solenoid, a spool valve and a spring) and another called "MotorButterflyValve"(that uses a motor to actuate a butterfly valve). A new wiki page for each DesignModel is created automatically; a category called "FlowControlValveInstancesRefinements" is created in the wiki and the new wiki pages are added to this category. The user can access these DesignModels by clicking the "Instance/Refinements" link at the top of the page (Figure 12 b). This link opens the page shown in Figure 18 which lists all the DesignModels in the category "FlowControlValveInstancesRefinements", thereby allowing the user to navigate to these designs.

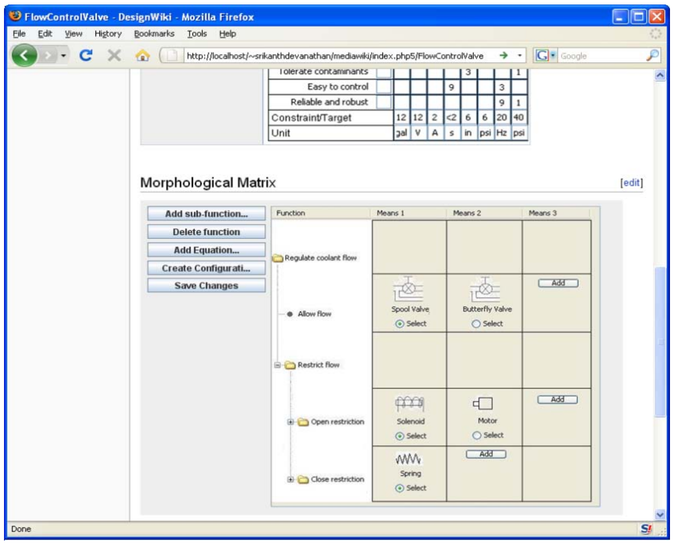

FlowControlValveFunctions

Functions of FlowControlValve

Figure 16. Screenshot of the page "FlowControlValveFunctions" showing the hierarchical function structure visual tool.

Clicking the "Behaviors" link on top of the "SolenoidValve" page (Figure 19) opens the "SolenoidValveBehaviors" page as shown in Figure 20. In this version of the prototype, SemanticBehaviorModel, although not created for the SolenoidValve example, are listed as plain text when available.

Having created an analytical model for the solenoid valve, the user now adds the house of quality visual tool to the "SolenoidValve" page (Figure 23). This house of quality visual not only contains the requirements and engineering characteristics that were defined for the "FlowControlValve", but also includes the qualitative relationships that were abstracted from the analytical behavior models. These relationships were abstracted from geometric as well as analytical relationships that were added using other tools. This demonstrates how different representations of the same design description element (relationship, here) are used in design; and more importantly, how they can be supported for iterative design exploration.

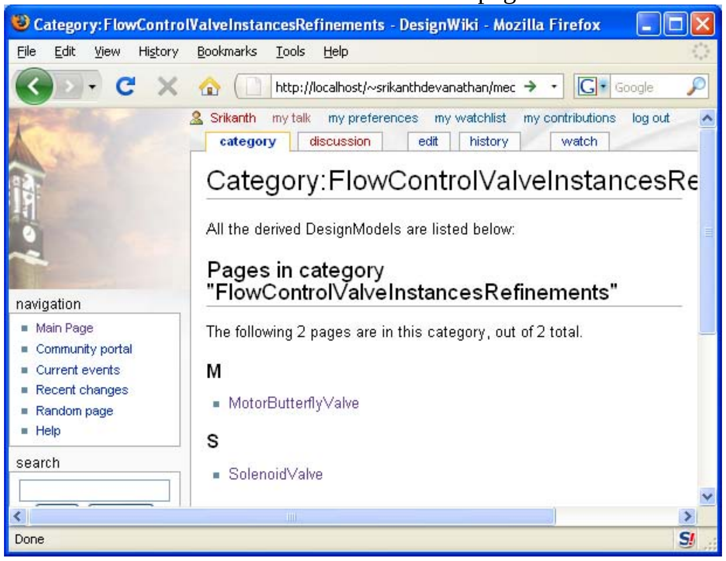

Tolerate contaminants					3				1
Easy to control				9			3		
Reliable and robust							9	1	
Constraint/Target	12	12	2	<2	6	6	20	40	
Unit	gal	V	A	s	in	psi	Hz	psi	

Morphological Matrix [edit]

Function	Means 1	Means 2	Means 3
Regulate coolant flow			
Allow flow	Spool Valve ◉ Select	Butterfly Valve ○ Select	Add
Restrict flow			
Open restriction	Solenoid ◉ Select	Motor ○ Select	Add
Close restriction	Spring ◉ Select	Add	

FlowControlValve - DesignWiki - Mozilla Firefox
http://localhost/~srikanthdevanathan/mediawiki/index.php5/FlowControlValve
Add sub-function...
Delete function
Add Equation...
Create Configurati...
Save Changes

Figure 17. Screenshot of the morphological matrix tool embedded within the wiki page.


Category:FlowControlValveInstancesRe

All the derived DesignModels are listed below:

Pages in category "FlowControlValveInstancesRefinements"

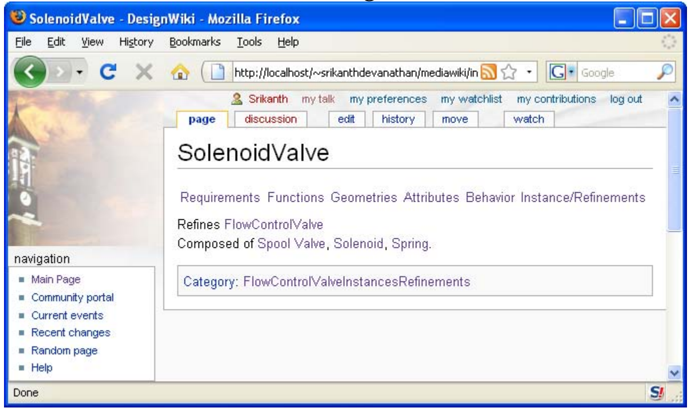
The following 2 pages are in this category, out of 2 total.

M
- MotorButterflyValve

S
- SolenoidValve

Category:FlowControlValveInstancesRefinements - DesignWiki - Mozilla Firefox
http://localhost/~srikanthdevanathan/mec
category discussion edit history watch
navigation
Main Page
Community portal
Current events
Recent changes
Random page
Help
search

Figure 18. Clicking the "Instance/Refinement" link in the "FlowControlValve" wiki page allows the user to navigate to other DesignModels.


SolenoidValve

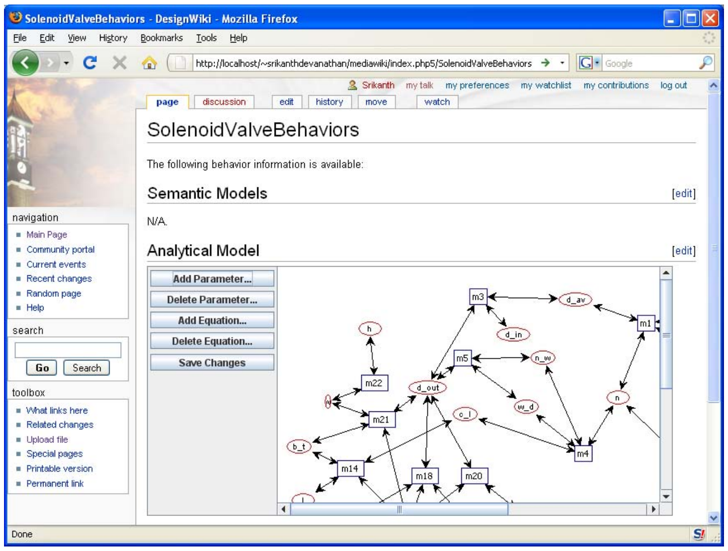
Requirements Functions Geometries Attributes Behavior Instance/Refinements

Refines FlowControlValve
Composed of Spool Valve, Solenoid, Spring.

Category: FlowControlValveInstancesRefinements

SolenoidValve - DesignWiki - Mozilla Firefox
http://localhost/~srikanthdevanathan/mediawiki/in
page discussion edit history move watch
navigation
Main Page
Community portal
Current events
Recent changes
Random page
Help

Figure 19: The wiki page created by after using the Morphological Matrix tool



SolenoidValveBehaviors

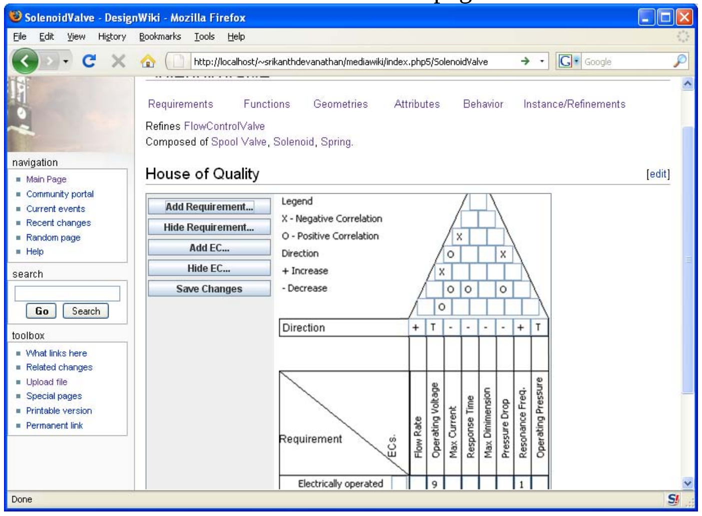
The following behavior information is available:

Semantic Models [edit]

N/A.

Analytical Model [edit]

Figure 20. The behavior models for the solenoid valve are shown in the wiki page.



Refines FlowControlValve

Composed of Spool Valve, Solenoid, Spring.

Figure 21. The roof of HoQ after abstracting the analytical constraints into qualitative relationships.

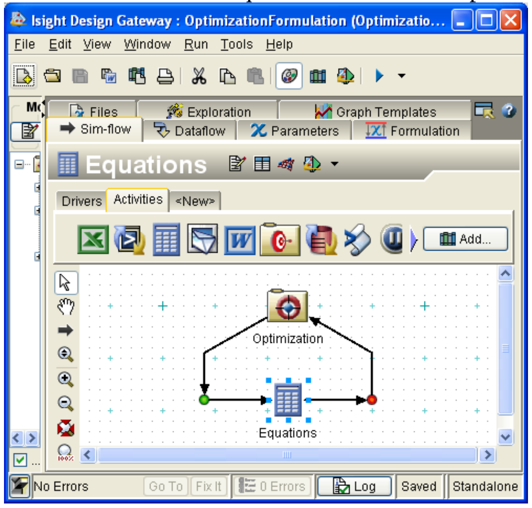


Figure 24. The optimization problem is formulated for the SolenoidValve design.

7 DISCUSSION

The DesignModel of the Solenoid valve, at this stage in design, contained the parameters, analytical equations, parameter definitions, as well as constraints and objectives. It was therefore deemed natural to allow the user to setup and run an optimization problem using this information. A separate tool was implemented that collates parameters, equations, constraints and objective definitions and creates an Isight [58] optimization model shown in Figure 22. In the current implementation, all the parameters and equations were added to the Isight model for optimization; the Isight model had to be manually edited to remove unnecessary variables, constraints and equations. The use of the optimization model is to suggest that the information available in WKM can be used to formulate known problems in design. Future work will include extensions to other design problems such as configuration design.

The working knowledge model and the prototype described in this paper is part of an ongoing effort to create a tool that can be used by designers at least by students in an education setting. Based on the effort in developing the case study the following observations can be made:

1. The wiki provides an accessible medium for storing WKM data. The set of features provided by a wiki are sufficient to allow the user to tag portions of the text to acquire WKM data.
2. The ability to embed visual tools directly in a wiki page allows the page to behave as a "live" design document collating all related information as well as allowing the user to explore the design using visual tools.
3. The process of manually tagging the wiki text is laborious and prone to error. In general, the wiki syntax can obfuscate the text [54]. In order for the prototype to be useful to designers, a What You See Is What You Get (WYSIWYG) editor is needed.
4. Visual tools such as the Morphological Matrix, automatically create wiki pages with important tags and statements. These statements and tags are visible to the user when the page is edited and are prone to deletion. In the current implementation, it is difficult to rectify such deletions without special scripting.
5. A typical installation of Mediawiki caches each page in the database to be delivered when the page is requested; this behavior is efficient when the pages are assumed to change little with time, as in an encyclopedia. However, in the current implementation, since the design information changes frequently we disabled the caching ability so that changes made in other related pages are reflected when the page is loaded. The process to generate the page consumes processing time on the wiki server and is therefore not scalable with users.
6. Another important aspect of such design support tools is restricting the wiki pages accessible to users. Although the wiki provides the ability to manage user access, certain operations such as creating wiki pages that were performed by visual tools required administrative privileges not available to all users; we therefore had to ensure all users had administrative privileges. This is a serious security concern that will have to be

addressed in future before being made available to general users.

7. The number of visual tools implemented in this prototype is limited. Other visual representations such as Technical process diagrams and concept selection tables need to be implemented for the prototype to be effective. The visual tools were implemented in the prototype to merely demonstrate the feasibility of using WKM to transfer from one tool to the other. Identifying and implementing the effective set of visual tools that are optimal for a particular design is beyond the scope of the current work.

8 CONCLUSIONS

Mechanical design is a complex iterative and cognitively challenging activity. Designers reduce this complexity by focusing on a narrow aspect of the design problem using visual tools. Such diagrams reduce the cognitive load on the designer to understand the relationships between the diagram entities. Design can also be characterized as a divergent-convergent exploration process. The key aspects of such a design exploration process are the reasoning about alternatives and describing these alternatives at different levels of abstraction.

In this article, an information-model to support such a design process is presented. This model, called the working knowledge model defines a minimal ontology that is needed to capture the relationships and describe both the product knowledge as well as the model of several visual tools. This model was developed by identifying a minimal set of design knowledge concepts that are required to describe the set of visual tools used in early phases of mechanical design. A wiki-based prototype system was implemented to demonstrate the connection among visual tools to the working knowledge model and thereby support early design exploration. A sketch-based design exploration tool was also implemented based on the working knowledge model to support embodiment design exploration. The geometric constraints and the analytical constraints that were added to the design were abstracted as qualitative relationship thereby allowing exploration at different levels of abstractions.

ACKNOWLEDGMENTS

Devanathan acknowledges the support provided by Purdue Discovery Park Center for Advanced Manufacturing (CAM) during initial stages of the work presented

REFERENCES

[1] N. P. Suh, Axiomatic Design: Advances and Applications, Oxford University Press, New York, USA, 2001.

[2] M. J. French, Conceptual Design for Engineers, Butterworth, London, 1985.

[3] G. Pahl, and W. Beitz, Engineering Design: A systematic approach, Second ed., Springer, 1999.

[4] D. Braha, and O. Maimon, A Mathematical Theory of Design: Foundations, Algorithms and Applications vol. 17, Kluwer Academic Publishers, Dordrecht, The Netherlands, 1998.

[5] J. S. Gero, and T. McNeil, An Approach to the Analysis of Design Protocols, Design Studies, vol. 19, (1998), pp. 21-61.

[6] G. Pahl, and W. Beitz, Engineering Design: A Systematic Approach, Springer, 1996.

[7] G. E. Dieter, Engineering Design: A Materials and Processing Approach, 3 ed., McGraw Hill, New York, USA, 1999.

[8] N. Cross, Engineering Design Methods: Strategies for Product Design, John Wiley & Sons Litd., West Sussex, England, 2000.

[9] D. G. Ullman, The Mechanical Design Process, Third ed., McGraw-Hill, 2003.

[10] S. Rachuri, Y.-H. Han, S. Foufou, S. C. Feng, U. Roy, F. Wang, R. D. Sriram and K. W. Lyons, A Model for Capturing Product Assembly Information, ASME Journal of Computing and Information Science in Engineering, vol. 6, (2006), pp. 11-21.

[11] S. Szykman, S. J. Fenves, S. B. Shooter and W. Keirouz, A foundation for interoperability in next-generation product development systems, Computer Aided Design, vol. 33, (2001), pp. 545-559.

[12] M. R. Bohm, R. B. Stone, T. W. Simpson and E. D. Steva, Introduction of a data schema to support a design repository, Computer Aided Design, vol. 40, (2008), pp. 801-811.

[13] F. Stjernfelt, Diagrammatology an investigation on the borderlines of phenomenology, ontology, and semiotics, Springer, Dordrecht, 2007.

[14] Y. Reich, "Synthesis and theory of knowledge: general design theory as a theory of knowledge, and its implication to design," in Engineering design synthesis, A. Chakrabarti, Ed., ed London: Springer-Verlag, 2002.

[15] R. Klein, "Knowledge modelling in design–the MOKA framework," in Artificial Intelligence in Design'00, Dordrecht, 2000, pp. 77-102.

[16] S. Devanathan, C. Sauter, A. Albers and K. Ramani, "A Working Knowledge Model for Supporting Early Design Through Visual Tools," in 17th International Conference on Engineering Design (ICED'09), Stanford, CA, 2009, pp. 299-310.

[17] V. Goel, Sketches of thought: a study of the role of sketching in design problem-solving and its implications for the computational theory of the mind, University of California at Berkeley, Berkeley, CA, USA, 1991.

[18] V. Hubka, W. E. Eder, Design Science: Introduction to the Needs, Scope and Organization of Organization of Engineering Design Knowledge, second ed., Springer, 1996.

[19] K. Otto, K. Wood, Product design: techniques in reverse engineering and new product development, Prentice Hall, Upper Saddle River, NJ, 2001.

[20] J. R. Hauser, D. Clausing, The House of Quality, Harvard Business Review, (1988), pp. 63-73.

[21] S. Wilhelms, Function- and constraint-based conceptual design support using easily exchangeable, reusable principle solution elements, Artificial Intelligence for Engineering Design, Analysis and Manufacturing, vol. 19, (2005), pp. 201-219.

[22] B. O'Sullivan, "Constraint-Aided Conceptual Design," Ph. D. thesis, Department of Computer Science, University College Cork, Cork, Ireland, 1999.
[23] http://www.omg.org/cgi-bin/apps/doc?formal/07-09-01.pdf. (2007, June). OMG SysML Specification v 1.0. Available: http://www.omg.org/cgi-bin/apps/doc?formal/07-09-01.pdf
[24] M. L. Maher, M. B. Balachandran and D. M. Zhang, Case-based reasoning in design, Lawrence Erlbaum Associates, Mahwah, NJ, USA, 1995.
[25] A. K. Goel, S. R. Bhatta and E. Stroulia, "Kritik: An Early Case-Based Design System," in Issues and Applications of Case-Based Reasoning in Design, M. L. Maher P. Pu, Eds., ed Mahwah, NJ: Erlbaum, 1997, pp. 87-132.
[26] A. K. Goel and S. R. Bhatta, Use of design patterns in analogy-based design,, Advanced Engineering Informatics, vol. 18, (2004), pp. 85-94.
[27] J. E. Fowler, "Variant Design for Mechanical Artifacts - A State of the Art Survey," National Institute of Standards and Technology, Gaithersburg, MD 208991993.
[28] H. J. Franke, S. Loffler and M. Deimel, "Increasing the efficiency of design catalogs by using modern data processing technologies," in International Design Conference, Dubrovnik, 2004, pp. 853-858.
[29] K. Roth, "Design Catalogues and their usage," in Engineeriing Design Synthesis, A. Chakrabarti, Ed., ed London: Springer-Verlag London Limited, 2002.
[30] J. Kim, P. Will, S. R. Ling and B. Neches, Knowledge-rich catalog services for engineering design, Artificial Intelligence for Engineering Design, Analysis and Manufacturing, vol. 17, (2003), pp. 349-366.
[31] R. Sudarsan, S. J. Fenves, R. D. Sriram and F. Wang, A product information modeling framework for product lifecycle management, Computer-Aided Design, (2005), pp. 1-13.
[32] D. Xue, H. Yang and Y. L. Tu, Modeling of Evolutionary Design Database, Journal of computing and information science in engineering, vol. 6, (2006), pp. 22-32.
[33] International Standards Organization, ISO 10303 - Automation systems and integration — Product data representation and exchange .
[34] R. S. Peak, R. M. Burkhart, S. A. Friedenthal, M. W. Wilson, M. Bajaj and I. Kim, "Simulation-Based Design Using SysML—Part 1: A Parametrics Primer," presented at the INCOSE International Symposium, San Diego, 2007.
[35] R. S. Peak, R. M. Burkhart, S. A. Friedenthal, M. W. Wilson, M. Bajaj and I. Kim, "Simulation-Based Design Using SysML—Part 2: Celebrating Diversity by Example," presented at the INCOSE International Symposium, San Diego, 2007.
[36] K. Fogarty, M. Austin, "System Modeling and Traceability Applications of the Higraph Formalism," University of Maryland, College Park, MD. ISR Technical Report 2007-21, 2007.
[37] R. D. Sriram, Intelligent systems for engineering: a knowledge based approach, Springer-Verlag London Limited, London, 1997.
[38] R. H. Bracewell, and J. E. E. Sharpe, Functional Description Used in Computer Support for Qualitative Scheme Generation - Schemebuilder, Artificial Intelligence for Engineering Design, Analysis and Manufacturing, vol. 10, (1996), pp. 333-346.
[39] F. Feldkamp, M. Heinrich and K. D. Meyer-Gramann, SyDeR—System design for reusability, Artificial Intelligence for Engineering Design, Analysis and Manufacturing, vol. 12, (1998), pp. 373-382.
[40] H. Leemhuis, R. Baumann, U. Kaufmann, F. Swoboda, T. Kuhn and R. Zbigniew, "Function oriented product modeling based on feature technology and integrated constraint management," in Proc. 11th Symp. Product Data Technology, Sandhurst, UK, 2002.
[41] D. R. Tamburini, "The Analyzable Product Model Representation to Support Design-Analysis Integration," Ph. D., Georgia Institute of Technology, Atlanta, Georgia, 1999.
[42] F. Andersson, K. Sutinen and J. Malmqvist, "Product Model for Requirements and Design Concept Management: Representing Design Alternatives and Rationale," presented at the International Conference on Systems Engineering, SE-412 96 Göteborg, Sweden, 2003.
[43] S. R. Bradley and A. M. Agogino, Design capture and information management for concurrent design, International Journal of System Automation: Research and Application (SARA), vol. 1, (1991),
[44] S. J. Fenves, S. Foufou, C. Bock and R. D. Sriram, CPM2: A core Model for Product Data, Journal of computing and information science in engineering, vol. 8, (2008), p. 014501.
[45] A. Messac and W. Chen, The Engineering Design Discipline: Is its Confounding Lexicon Hindering its Evolution?, Journal of Engineering Evaluation and Cost Analysis, Decision-Based Design: Status & Promise, vol. 3, (2000), pp. 67-83.
[46] B. J. Wielinga and J. M. AkkermansA. T. Schreiber, "A Formal Analysis of Parametric Design Problem Solving," in In Proceedings of the 9th Banff Knowledge Acquisition Workshop (KAW-95), ed, 1995.
[47] U. Roy, N. Pramanik, R. Sudarsan, R. D. Sriram and K. W. Lyons, Function-to-form mapping: model, representation and applications in design synthesis, Computer-Aided Design, vol. 33, (2001), pp. 699-719.
[48] www.ap233.org, STEP System Engineering Project, accessed February 2011.
[49] www.uml.org. Object Management Group, accessed February 2011.
[50] www.topcased.org, TopCased Editor, accessed February 2011.
[51] www.jgraph.com, JGraph Library, accessed February 2011.
[52] C. M. Hoffmann, Summary of basic 2D constraint solving, Int. J. Product Lifecycle Management, vol. 1, (2006), pp. 143-149.
[53] C.J. Walthall, S. Devanathan, K. Ramani, E.D. Hirleman, L.G. Kisselburgh and M.C.Yang, “Evaluating wikis as a communicative medium for collaboration with co-located

and distributed design teams," ASME J. Mech. Des. (2011), in press.
[54] C.J. Walthall, S. Devanathan, T. Deigendisch, C.Sauter, A.Albers and K.Ramani, "Survey of wikis as a design support tool", in 17th International Conference on Engineering Design (ICED'09), Stanford, CA, 2009.
[55] C.J. Walthall, S. Devanathan, L.G. Kisselburgh, K. Ramani and E.D. Hirleman, "A framework for evaluating wikis as a medium for communication within engineering design teams," ASME International Design Engineering Technical Conferences IDETC/CIE 2009, San Diego, CA, 2009.
[56] http://www.simulia.com/products/isight.html
[57] N. Titus and K. Ramani, "Design space exploration using constraint satisfaction," Nineteenth International Joint Conference on Artificial Intelligence IJCAI, July-August 2005, pp. 31-37.
[58] F. Zwicky, Discovery, Invention, Research through the morphological approach, MacMillan, New York (1969).
[59] Y. Maeda, Y., Koseki, Y. Koike and M. Tanaka, "Design knowledge organization and acquisition through visual representation,"in Knowledge Intensive CAD volume 1, T. Tomiyama, M. Mantyla, S. Funger eds, Chapman & Hall, New York, NY (1996).
[60] F. Lakin, J. Wambaugh, L. Leifer, D.M. Cannon and C. Sivard, "The electronic design notebook: Performing medium and processing medium," Visual Computer, vol. 5, pp. 214-226, (1989).
[61] www.mediawiki.org, Mediawiki, accessed February 2011.