Function-to-Form Mapping: Model, Representation and Applications in Design Synthesis

U. Roy*, N. Pramanik
Knowledge Based Engineering Laboratory
Dept. of Mech, Aerospace, Manufacturing Engineering
Syracuse University
Syracuse, NY 13244-1240
Email: {uroy, pramanik}@ecs.syr.edu

R. Sudarsan§, R. D. Sriram and K. W. Lyons
Design and Process Group
Manufacturing Systems Integration Division
National Institute of Standards and Technology
Gaithersburg, MD 20899
Email: {sudarsan, sriram, klyons}@cme.nist.gov
§ visiting from George Washington University, Washington, DC 20052

Abstract

Design of a new artifact begins with incomplete knowledge about the final product and the design evolves as it progresses from the conceptual design stage to a more detailed design. In this paper, an effort has been made to give a structural framework, through a set of generic definitions, to product specification, functional representation, artifact representation, artifact behavior and tolerance representation. A design synthesis process has been proposed for evolution of a product from the product specification. The proposed design synthesis method is a mapping from the functional requirements to artifacts, with multi-stage constrained optimization during stages of design evolution. Provisions are kept to augment and/or modify the product specification and domain knowledge during stages of development to guide the design process. Physical and structural details of an artifact are captured as abstract sketches during conceptual design stage and as CSG/B-Rep solid models during the detailed design evaluation stages. An example artifact library has been created to show the effectiveness of the proposed design synthesis process with a simple design element. An overall design scheme has been presented.

Keywords: Design synthesis, function-to-form mapping, CAD, object-oriented representation, conceptual design, Constructive Solid Geometry (CSG), Boundary Representation (B-Rep)

1. Introduction

Although considerable advances are made over the last decade in the development of function-to-form mapping systems, there is not very much progress on systems that really aid the designer in converting functional specifications to concept design. A comprehensive exploration of non-geometric concepts in the conceptual design phase and creation of rough realization of physical forms from loosely (and often vaguely) stated design goals and required functions is still not well established. Whitney [1] attributed this to "the lack of basic engineering knowledge that can link form and function, the lack of a mature concept of a product data model, and/or the lack of a mature concept of the product design process." Most of the current function-to-form mapping systems are very domain-specific and lookup domain-specific rules. In our approach, we have tried to present a set of generic definitions for the entities that play a major role in a design environment and have used these definitions to drive a design synthesis process, which is fairly generalized.

2. Review Of Related Works

In the past, the research efforts involving part functions were mainly focussed in four major areas: (i) development of standard vocabularies for part functions; (ii) development of ontologies for functions; (iii) conceptual design with abstract part functions; and (iv) design with spatial relationships.

* corresponding author

To fill up the gap between the concept design (from a given set of functional descriptions) and actual geometry-based CAD, researchers were first trying to create a computer-aided design system that would help designers explore non-geometric concepts and to create rough realizations. Pahl and Beitz [2] suggested a procedure for deriving the overall functions of a design from the given design problem statements; and then decomposing and recomposing individual sub-functions to a hierarchical functional structure that could be mapped to appropriate physical elements. Though the method provides useful suggestions regarding the function decomposition process, it however, does not relate the functions to design geometry.

Kota [3] viewed mechanical designs as being synthesized from conceptual building blocks that perform specific kinematic functions. The motion synthesis approach provides a method for recognizing a given behavior in terms of known primitive behaviors. This is one of the first formalized ways of viewing design as the synthesis of kinematic processes; however, the approach is limited to a fixed set of primitives. Hundal [4] and Iyengar [5] have developed Function Based Design (FBD) Systems in which a configuration of parts is determined on the basis of the user specified functions (input and output quantities of a part). The FBD systems are only useful in the conceptual design stage and do not take into account the interaction between the parts at the geometric level. Schmekel [6] has presented a formal language which consists of a symbolic functional model to describe the functions of a product. The functions of the product are decomposed into relationships which are mapped onto functions of standard components. It, however, only deals with standard machine elements. Kannapan and Marshek [7] have developed a design reasoning system for developing design concepts based on functional and geometric requirements of mechanical systems built using different machine elements.

Rinderle with Hoover [8] and Finger [9] has described a procedure in which design specifications are transformed into an actual physical artifact using the bond graph technique. This bond graph technique has also been adopted by a number of other researchers because of its flexibility of modeling hybrid systems, such as electro-hydraulic electro-mechanical, etc. using the same symbols and mathematics throughout. However, bond graphs cannot be used for representing detailed part functions as they abstract a number of functional details and do not account for spatial relationships. Ulrich and Seering [10] have used bond graph techniques to transform a graph of design requirements into functionally independent physical components. Their technique is useful in the domain of conceptual design but cannot be used for detailed part design. Bracewell [11] has extended the bond graph based conceptual design scheme by coupling it with a parametric 3D geometric modeler. Gui and Mantyla [12] have studied a behavior modeling technique for components in assemblies. They have developed a set of behavioral specifications which can be used to specify the inter-relationships between sub-components and have focused on the issue of representing these relationships.

There are also several reported research programs [13-26] on the development of system frameworks for product modeling where function, behavior and product modeling have been discussed. [13,14] and [15-18] are the most important works that can be useful for our tolerance design purposes. MOSES is a research program being jointly undertaken by the Loughborough Institute of Technology and University of Leeds. Their works [13, 14] have explicitly focussed on the development of product models for representing different types of past attributes such as part function, manufacturing details and assembly. Sriram and Wong [15, 16] have developed the DICE (Distributed and Integrated environment for CAE) system with a view of addressing the coordination and communication problem in the product design process. Sriram's earlier work on the development of a flexible knowledge-based framework (CONGEN-CONcept GENerator) provides methods for generating concepts during the initial stages of the design [16,18]. CONGEN consists of a layered knowledge base (including domain independent knowledge sources like, synthesizer, geometric modeler, constraint manager, evaluator, etc.), a context mechanism, and a user-friendly interface. The architecture of CONGEN could be enhanced to address the life-cycle issues of a product and to consider the entire design/manufacturing processes of a product that is still in the preliminary stages of design. In [26], Szykman, Racz, and Sriram have proposed a generic scheme for representation of functions and associated flows. The scheme provides a mechanism for mapping from function domain to physical domain through references to artifacts. It supports both decomposition of functions into sub-functions and functions with multiple input and output flows.

However, what is missing from the previous works is an integrated approach to the modeling, representation and decomposition of part functions into a useful format that can be used for conceptual design as well as for detailed design including addressing issues involving tolerance synthesis, manufacturability and assembliability.

3. Schematic Representation of Product

A scheme for the representation of the product / artifact is defined below (figure 2) in a simplified tabular form. This is a generic class definition aimed at capturing the necessary details of customers' specification and conversion of the customers' specification into technical specification by the designer. The scheme is implemented in an object-oriented environment using Unified Modeling Language (UML), C++/Java. In this paper, we extend the object oriented class definition developed at NIST [24,26].

Name	String
Inputs	{ [**Input**]}
Outputs	{ [**Output**]}
Function	{ [**Function**]}
Artifact	Reference to **Artifact** through which Functional requirements are achieved
Relations	{[**Constraint**] } defined over the inputs, outputs and internal attributes of the function.
OptimalityMeasure	{ [**Constraint**] } Goal

Figure 2. Product Specification

The product specification consists of a set of product attributes (Inputs, Outputs), constraints associated with the product and optimization goals. Each attribute has its own identifier (name), type, category, relevance factor, unit of measurement and a value range.

Category of the input/output is a physical or abstract measurable entity in the design domain like (mass, length, force, torque, velocity, angular_velocity, electric_current, magnetic_flux, area, volume, voltage, etc.) Each category may have further sub-attributes to describe them.

A range of values is an interval (pair of numbers indicating the lower and the upper bound). All finite intervals are considered as closed intervals. Open intervals like >10 would be represented as [10, Inf] where Inf stands for infinity.

Sometimes, only a qualitative value may be known for an attribute. Thee qualitative values would be represented as the grade of the attribute from the set (VH HI MO LO VL) where VH = very high, MO = moderate, LO = low, VL = very low.

Constraints are restrictions on the possible variation of attribute values and are represented as a. set of relations amongst the attributes. Three types of constraints are considered here: relational constraints, causal constraints and spatial constraints. Relational constraints are direct functions of attributes. For example: in a rotary motion transformation, a global constraint requiring a speed ratio (assuming ω_I as input and ω_O as output rotary speed) could be represented as: ω_O.value / ω_I.value = [5,6]; speed increase by a factor of 5 to 6.

Causal constraints indicate dependency of one attribute on other attributes but the exact functional relation may not be known. These types of dependency would be useful mainly in studying the qualitative behavior of an artifact.

Spatial constraints are form-dependent functional relations. These are separated from the main relational constraints for ease of treatment. These constraints impose some form of geometric restrictions amongst the attributes of an artifact/device. Some examples of these types of constraints, for a chair, could be (‘arm’ parallel_to ‘seat’), (‘backrest’ perpendicular_to ‘seat’), (‘seat’ distance_from ‘base’ 2ft), etc.

Goals in the product specification are global optimality evaluation functions. In most of the cases, these optimizations can only be performed after a shape and size for the product has been evolved based on the product functional requirements. An example of a global goal could be: minimize total_weight ().

It has to be emphasized here that evaluation of global goals may not be feasible immediately after a feasible solution has been arrived at. After a feasible solution has been arrived at by satisfying the input/output requirements (functional requirements) and the constraints, the solution has to be converted into a physical solution by detailed design (sizing), allocating tolerances where possible, and then the global goals could be evaluated.

4. Functional Model And Its Representation

In the early phase of design, most of the design decisions taken are concerned with the desired characteristics and the overall functions of the assembly. By “Function” we mean an abstract formulation (or definition) of a task that is independent of any particular solution. In this phase, the abstract functional specification of an artifact is transformed into a physical description. In the later phases of design, the physical decisions that are made in the earlier phases are elaborated to ensure that they satisfy the specified functional requirements and life cycle evaluation criteria. To manipulate the “function” information, a “functional data model” (that describes the functional information throughout the design cycle) is needed so that appropriate reasoning modules can interrogate and extract functional information during the decision-making processes (as the geometric reasoning modules query data from the “product data model” (CAD model) during the shape design process).

Depending on the design phase, different types of function exist at different levels of abstraction. In the conceptual design phase, functions are usually independent of working principle, whereas in later design phases, when the functions are detailed, they become increasingly dependent on the working principle that has been selected. Therefore, we need to adopt such a formal function representational scheme that will be helpful when modeling the overall function of the assembly in the conceptual design stage, and also be useful to model smaller and smaller sub-assembly, particularly as the component and feature levels are approached. The representation should also have unambiguous semantics to perform analysis on (i.e. to manipulate) the functional description.

In the present paper, an attempt has been made to give a definition of a function, which captures some of the basic features at the abstract level, as well as detailed design level. Functions defined here are also having a mechanism to incorporate functional equivalence classes associated with a function. This means that a function will have references to other functions that can collectively be considered equivalent to the function. The collective equivalence could be of a combination of functions connected by ‘AND’ and ‘OR’ or combinations there of. Domain-specific knowledge could be used to define such equivalence classes. This will be a fundamental feature for functional decomposition during the design process.

As for example:

function ('rotary_motion to rotary_motion') could be expressed as equivalent to:
function('rotary_motion to linear_motion') AND
function('linear_motion to rotary_motion')

A definition for the function in a tabular form is given in Figure 3.

Name	String
Input	{ [**Input**]}
Output	{ [**Output**]}
FunctionofArtificat	Reference to **Artifact** through which Functional requirements are achieved
Relations	{ **Constraint**} defined over the inputs, outputs and internal attributes of the function.
SubFunctionOf	{ [**Function**] }
SubFunctions	{ [**Function**] }
OptimalityMeasure	{ [**Constraint**] } Goal

Figure 3. Functional Representation

5. Behavioral Model And Its Representation

The functional model as discussed in the earlier section is useful for conceptual design where the design concept is evolved on the basis of the specified abstract functions. However, the functional model is not sufficient to synthesize assembly/components behavior. This is because of the fact that functional models do not adequately capture the interactions of forces and kinematic motions between the part geometry. For instance, the fit condition between a shaft and a bore cannot be expressed by a spatial relationship since it does not provide functional design details such as contact pressure, contact force, rotational torque, rotational speed, etc. at the shaft-bore interface. To synthesize product/assembly behavior and geometric tolerances, full behavioral models of the involved components in the assembly (including both the structural and kinematic behavioral models) are required. In this section, we will discuss the structural behavioral model.

Behavior of a function is defined to be the set of values of parameters (which are related causally) of the function either at a specified time or a series over a period of time [27]. Behavior of a function is context sensitive and as such, behavior comes into play only in the context of a form. A function defined as an abstract object often can be achieved through different forms / devices and the function will have different behavior under each separate form / device context. In the present study, the following general class of behavior model (Figure 4) is proposed for the representation of behavior of a component.

Name	String
StateVariable	Variables of interest { [**Variable**] }
CausalLink	**{ [**DependsOn**] | [**AffectsVariable**] |** Null**}**
	DependsOn Set of variables that influence this variable. { [**Variable**] } AffectsVariable Set of variables that are influenced by this variable. { [**Variable**] }
ReferenceValue	**{ Tabulated | Procedural | External}**
	Tabulated Reference to a set of (time, value), at different times. Pair of time, value) can be read or interpolated directly. **Procedural** A method to compute the value at a specified time. Either a closed form parametric equation of time: x = x(t) or a procedure for compute the value of the variable at a specified time t >= 0. **External** A value coming as input and needs no computation.
SubBehaviorOf	{ [**Behavior**] }
SubBehaviors	{ [**Behavior**] }
BehaviorOfArtifact	Reference to **Artifact** for which the Behavior is computed.

Figure 4. Behavioral Representation

6. Artifact Model And Its Representation

The modeling of an artifact and its components for use in conceptual design as well as in detailed design stage, including synthesis and analysis of geometric tolerances, requires a high-level description which can be represented by a set of attributes. This attribute/sub-attribute set forms a symbolic representation of the artifact that can be manipulated by the design synthesis process as described in Section 7.

6.1 Artifact Classification

To facilitate efficient retrieval, artifacts are grouped by a set of known functionality. Following Thornton's list [28], we adopt the following groups (Figure 5) that imply a particular mechanical functionality and sometimes possess some kind of generic shapes. Please note that in the following list, many of these names cover groups of components. For instance a gear could be a spur gear or a bevel gear, and could transmit torque (rotation) between parallel or perpendicular shafts.

Group	Artifacts
Containers/Reference Frames	Cover, Housing, Plate, Duct, Pipe
Controllers	Valve, Gauge, Knob, Pointer, Nozzle
Fasteners	Clip, Bolt, Nut, Strap, Rivet Coupling, Pin
(Pl. note that these Fasteners are Joint Elements as discussed in section 6.3)	
Load Bearers	Bracket, Brace, Bearing, Web, Pillar, Rod, Spring
Locators	Joint, Spacer, Pivot, Pad, Key, Pin
Power Transmitters	Chain, Cable, Shaft, Pulley, Cam, Gear, Piston
Seals	Gasket, O-ring, Sleeve

Figure 5. Artifact Groups

6.2 Artifact Representation

The artifact representation model consists of three basic parts: a functional representation, a structural representation and a behavioral representation. These are discussed in the following subsections.

Form Representation

The form of an artifact is expressed in terms of its constituent components and sub-components, and the interactions between them. The form of each artifact representation consists of information about

- (i) the component/sub-component structure of the artifact,
- (ii) typical shape with the critical dimension outlined,
- (iii) rules for selecting and sizing the artifact,
- (iv) alternative shapes (and/or overall dimensions),
- (v) list of additional features and services which would be required to make the artifact work in a real-life environment,
- (vi) the possible modes/situations in which the artifact might fail, and
- (vii) material properties.

Components in a composite artifact (assembly) are of type Artifact and the assembly contains references to these components. Components can be of two different varieties: either primitive or composite. Composite components are those whose internal substructure is represented explicitly by a set of more detailed, lower-level sub-components, whereas primitive components are simple.

The **Artifact** data model contains artifact-specific information that not only helps in design synthesis but also in studying its behavior in an assembly. Along with the key artifact characteristics and component/sub-component structural description, the data model must contain information on artifact tolerances and its kinematic variations. A detailed definition of the artifact is presented below:

Generic Definition of the Artifact

The term *artifact* in this report is used synonymously with physical object, device, form, component, and assembly/sub-assembly of components. In the object-oriented approach, all entities like artifact, function, device, etc, will be defined as classes.

An **Artifact** will essentially have its functional details, form/structural details and behavior model along with links to other artifacts/functions.

To capture the essential information of artifacts so that a general mapping procedure could be adopted to evolve a design from a product specification, including satisfaction of constraints, a generic definition of the artifact is presented in figure 6 below:

.

Name	String
Purpose	String
ArtifactGroup	Artifact **Group**. The **Group** classifies the artifact into a category by the type of application it can perform
ArtifactAttributes	List of attributes similar to those defined for a Product. Attributes are of type (**Function, Artifact, Input, Output**, internal) and

	would have their own details exactly as defined in product specification	
Requires	**{ [Artifact] [**Location] **[**Orientation] **} |{[Function}.** This is a link to other artifacts or functions that are required by this artifact	
	Location Location of the component **Artifact** w.r.t. local coordinate systems of **Artifact .** The location or position of the component in the main assembly where this part will fit with respect to a local coordinate system on the main assembly. Orientation Orientation of the Artifact w.r.t. other **Artifact** The orientation along with the location or position of the component in the main assembly will define the structural links between the main assembly and the part.	
Form	{ Name **, Sketch [Parameter**] **[Behavior>] [Constraint] [Feature] [FormTolerance], { [Material] }** Form/Structure is defined here as a class that incorporates both abstract structure as well as physical shapes of **Artifact**s. The **Sketch** entity in this structure is used to represent both of the two-structure types.	
	Feature Subclass of **Artifact.** **Part** Subclass of **Artifact.**	**SubAssembly** Subclass of **Artifact.** **Assembly** Subclass of **Artifact**
	Sketch	**{ AbstractSketch [CADSketch]}**
	AbstractSkecth { **SketchNode IndividualBehavior SketchTolerance** }. . .	The abstract sketch is a schematic representation of the bare minimal structural information required during conceptual design. The **SketchNode** entities are the nodes on the structure where we focus our attention for input/ output/ other special features.
	SketchNode {	**Coordinates DegressOfFreedom AssociatedVariables Input Output }**
	CADSketch	CSG / B-Rep representation of **Artifact,** CAD model file reference.
Relations	{ [**Constraint**] } defined over the inputs, outputs and internal attributes of the **Artifact**.	
Optimality Measures	{ [**Constraint**] } Goal	
ArtifactBehavior	{ [**Behavior**] }	
	FormBehavior	Behavior of the **Form.**.
	IndividualBehavior	Behavior of the individual **SketchNode**
ArtifactFunction	**{ [Function] }**	
ArtifactTolerance	{ [**Tolerance**] }	
	ArtifactTolerance SubClass of **Tolerance.** Tolerance class defined for the overall artifact. **FormTolerance** SubClass of **Tolerance.** Tolerance class defined for the structure **SketchTolerance** SubClass of **Tolerance**. Tolerance class defined for the individual sketch elements.	

Figure 6. Artifact Definition

It is assumed that each artifact will have at least one relational constraint of the form: $\mathbf{C_0(I,O,\beta) = 0}$, where **O** is the output, **I** is the input and β is an n-vector internal parameter of the artifact. $\beta = (\beta_1, \beta_2, \ldots \beta_n)$. The vector β is the internal parameters defined in terms of physical attributes and laws governing the performance of the system. It is also assumed that the input to an artifact as well as each component of the parameter β will have a known feasible range of values. These limits are the physical ranges within which the artifact can operate. Each component of β can be expressed in parametric form as:

$$\beta_k = \beta_{k\_Low} + (\beta_{k\_High} - \beta_{k\_Low}) * \theta_k : \quad \theta_k \in (0,1), \quad k \in (1,n)$$

where β_{k_Low} is the lower limit and β_{k_High} is the upper limit of value of β_k .

The above relational constraint, $\mathbf{C_0(I,O,\beta) = 0}$, could be solved for **O,** giving $\mathbf{O = f(I, \beta)}$. This representation shows the main causal link between the input and the output of the artifact.

As for example, a helical spur gear box to reduce/increase the speed would have an internal parameter R which is the ratio of the number of teeth of the two gears and the main constraint $\mathbf{C_0 (N_i , N_0 , R) \equiv N_0 - R * N_i = 0}$ where $\mathbf{N_i}$ is the input speed**,** $\mathbf{N_0}$ is the output speed and **R** is the speed ratio. **R** could have a range of, say, [0.25, 4], indicating that the gear box could be used for reduction of speed of up to 1/4th as well as for increasing the input speed by 4 times. Depending on specific requirements dictated by the solution, suitable value will be selected.

Apart from the main causal link,$\mathbf{C_0}$ between the input and the output, there could also be several other constraints associated with an artifact that has the general form of:
$\mathbf{C_j(I, O, \beta)}$ **<rel_opr> <value>, j >0** where **<rel_opr>** is the relational operator (one of **{LT LE EQ GE GT NE}**) and **<val>** is a numeric value. In some cases, these relations would be converted to the standard equality form $\mathbf{C_j(I, O, \beta) = 0}$, by introducing additional variables for the cases where the **<rel_opr>** is not "**EQ**"). As for example, the constraint, $\mathbf{C_j(I, O, \beta) \equiv \beta_1}$ **GE 5.00** would be converted to: $\mathbf{C_j(I,O,\beta) \equiv \beta_1 - t - 5.00 = 0}$, where t (>=0) is an auxiliary variable.

Along with all the information as described in figure 6, each component of an artifact needs to be modeled as being composed of a single material. Materials are to be represented as material class (remark: we have included some reference to the material properties as attributes in the example definition of the artifacts; a separate material class definition will be introduced at a later stage of this work to represent the entire material related aspects). A list of possible candidate materials and notes on manufacture need to be included as well.

The artifact definition should also have an attribute linked to a database of manufacturing "know-how" (which has not been included in the present artifact definition). It should contain information on what shapes can be made of what materials and what processes could be used (including specific process capabilities and characteristics).

To illustrate the conceptual design procedure, we classify some of the motion-transmitting artifacts against a set of kinematic functions. A sample artifact library (**ARTL**) has been developed for use in the design synthesis example (**Appendix – I**). Examples of representation of functions and artifacts using the proposed definitions are shown in **Appendix – II.**

The Example – 4 in Appendix – II elaborates the representation of artifacts. We have shown two artifacts. The first one is a primitive (a spur gear) and the second one is a compound artifact (a gearbox consisting of six primitive artifacts: two spur gears, two shafts and two keys). While both the representations use the same artifact definition given in 6.2, the second artifact gearbox uses other artifacts through the "*requires*" parameter.

The representation for the spur gear (primitive artifact) begins with a list of physical properties/attributes like material, diameters, thickness, etc. (we can add/modify as many parameters as would be necessary to define the artifact). The *'input' and 'output'* are then listed. The *'requires, 'constraints', 'form/structure' and 'behavior'* parameters then follow. In this case, there was no constraint specified. The 'behavior' has been defined as a set of state-space variables associated with the artifact.

For the gearbox (compound artifact), the representation follows a similar logic, however, in this case, the '*requires'* clause uses the spur gear primitive (defined earlier) along with shaft and key elements. Also, in this case a constraint has been added as a relationship between the output and the input. This forms the main constraint $\mathbf{C_0}$ defined earlier in this section. The behavior also reflects the outcome of the combined elements.

6.3 Artifact Link/Joint Model And Its Representation

Physical connections between artifacts are modeled as link/joint artifacts, with each high-level connection being elaborated in terms of lower-level connections. In the assembly level, the relationship of an artifact with other artifacts has two main attributes: relationship type and mating characteristics. A relationship type can be simple "contact", "detachable type rigid attachment", "constraint" etc. Mating characteristics are described by: (i) mating subassembly/part identification number, (ii) identification numbers of mating features, and (iii) mating conditions. Identification of mating features are done by (i) feature I.D. and (ii) mating face I.D. Each face contains information about face_type (planar, cylindrical, etc.), dimension, size_tolerance, form_tolerance, normal vector, process model, force_d.o.f., kinematic_d.o.f., surface_roughness and face_interaction_list [29].

Mating conditions are described by mating type, joint characteristics, spatial constraints, rules, and attributes. Mating type can be any of these: Fit (i.e., sliding, clearance, transition, and interference), Fasteners, Gear, Bearing, etc., which can be further described in terms of some atomic relationships: orientation, primary mating (no_contact, touch_contact and overlap_contact) relations, size relation, intersects, between, containts , and connect.

7 Design Methodology: An Approach To Function-To-Form Mapping

The design of an artifact to satisfy the product specification (PS) is a complicated process. The design process is considered evolutionary in nature. We start with incomplete knowledge to look for suitable and/or functional entities in the corresponding library to arrive at a design starting point. At this stage, some of the attributes specified in PS may have been found and some of the constraints may have been satisfied. To proceed further, more knowledge is required to be injected into the system and the set of specifications are needed to be transformed for subsequent enhancement of the initial solution. Here the design of an artifact is represented as D = {<PS><Art_Tree>} where <PS> is Product Specification, <Art_Tree> is the artifact tree (a tree structured list of artifacts).

Initially, the artifact tree is empty. Subsequently, when suitable artifacts are mapped to perform a desired functionality, these artifacts are added to the artifact tree. Outputs from an artifact that are not in the PS go as inputs to the next artifacts. Outputs that are mapped in the PS are terminals. Also, the designer may designate an output as terminal so that further mapping of this output as input to a new artifact is not required. Above approach for design synthesis generates stages of (sequence of) of partial solutions as shown below.

$D_0 = \{<PS_0><Art\_Tree_0>\}$
$D_1 = \{<PS_1><Art\_Tree_1>\}$
$D_2 = \{<PS_2><Art\_Tree_2>\}$
…
$D_n = \{<PS_n><Art\_Tree_n>\}$

Where, at the beginning of the design process, $<Art\_Tree_0>$ is NULL.

It has been detailed later in this section that, at each stage, the partial solutions are checked for convergence to the desired output specified in the PS. This checking is performed using two basic criteria: a constrained norm minimization process involving the relational constraints associated with the product specification ant the individual artifacts. The norm (defined later in this section) is the 'distance' of the partial solution from the desired output. After the minimization, spatial constraints are checked. Based on the above two process, the set of candidate artifacts that are identified, are graded from 'best' to 'worst' at that particular stage. A *design alternatives control parameter,* $\mathbf{N}_{alt}$ as the number of artifacts (that are most desirable in the graded list) to be considered for the next stage is used to reduce the search space. This implies that, for example, if in a stage 10 artifacts are mapped and $\mathbf{N}_{alt}$ has been set by the designer as 3, only the 'best' three of these 10 will be used as possible candidates in this stage and searching would continue from those 3 only for the next stage. It may be noted here that as $\mathbf{N}_{alt}$ increases, possibilities of more diverse solutions increase, which is a desirable feature, since more alternative design solutions can be explored. However, there is a cost associated with the increase in $\mathbf{N}_{alt}$ in terms of computation time and storage requirements. In the proposed system, we have planned to keep this design control parameter $\mathbf{N}_{alt}$ as a designer selectable value.

The design synthesis process at some intermediate stage will have at most $\mathbf{N}_{alt}$ branches from each of the artifacts in that particular stage. The process of expanding a particular branch will terminate when one of the following conditions has been reached.

i) A feasible solution satisfying the output specification, relational constraints as well as spatial constraints are satisfied. This means that the minimization process discussed earlier has resulted in an acceptable distance between the desired output in PS and the partial solution. We designate this acceptable distance as a *convergence criterion*, $\in_0$. Thus $\mathbf{d} <= \in_0$ is the termination criteria.

ii) The search for a suitable artifact from the artifact library failed to map at least one artifact and hence the design synthesis process cannot proceed further.

There are some basic considerations in the design evolution process depicted above which needs further investigations. These are: transformation of PS_n to PS_{n+1}, including attribute transformation, constraint transformations, and variation of internal parameters of each artifact for searching a solution as a minimization of the above mentioned. These are discussed in the following sections, before the design synthesis procedures are presented.

7.1 Product Specification (PS) Transformations

In this subsection we discuss the details of Product Specification transformations which are required at each stage of the design synthesis process. The Product Specification transformation consists of Attribute Transformations, Constraint Transformations and the Variation of internal parameters. These are discussed in the following sections.

7.1.1 Attribute Transformation

The product specification $\mathbf{PS_0}$ contains the initial specification with $\mathbf{PS_0.Inp}$ and $\mathbf{PS_0.Out}$ as sets of input and output specifications, respectively. Assuming that at stage **j,** a sub-set of these sets of requirements are satisfied, $\mathbf{PS_j}$ is transformed into $\mathbf{PS_{j+1}}$ as described below.

Let us assume that an artifact, $\mathbf{Art_{jk}}$ has been found in the design stage **j** with some elements of $\mathbf{Art_{jk}.Inp}$ are in $\mathbf{PS_j.Inp}$ and some elements of $\mathbf{Art_{jk}.Out}$ are in $\mathbf{PS_j.\ Out}$. We can present as a union of two mutually exclusive sets

$$\mathbf{Art_{jk}.Inp = Art_{jk}.Inp_1 \cup Art_{jk}.Inp_2}$$
$$\mathbf{Art_{jk}.Out = Art_{jk}.Out_1 \cup Art_{jk}.Out_2}$$

where $\mathbf{Art_{jk}.Inp_1 \subseteq PS_j.Inp}$ and $\mathbf{Art_{jk}.Inp_2 \not\subset PS_j.Inp}$
where $\mathbf{Art_{jk}.Out_1 \subseteq PS_j.Out}$ and $\mathbf{Art_{jk}.Out_2 \not\subset PS_j.Out}$

If $\mathbf{Art_{jk}.Inp_2}$ is NULL then all input requirements of the artifact $\mathbf{Art_{jk}}$ are in the product specification $\mathbf{PS_j.Inp}$ and this artifact needs no further artifacts whose output should be mapped to inputs. Otherwise, we transform the inputs to a new set of outputs for some artifact to be searched with:

$$\mathbf{PS_{j+1}.Out = PS_j.Inp \cup Art_{jk}.Inp_2}$$

If $\mathbf{Art_{jk}.Out_2}$ is NULL then all outputs of the artifact $\mathbf{Art_{jk}}$ are in the product specification $\mathbf{PS_j.Out}$ and the outputs of this artifact need not be mapped as input to some other artifact. Otherwise, we transform these outputs to a new set of inputs for some artifacts. Here, the designer can accept some of these outputs as by-products to the environment and treat them as already satisfied. The remaining outputs are then transformed into a set of new input specification as:

$$\mathbf{PS_{j+1}.Inp = PS_j.\ Out \cup Art_{jk}.Out_2}$$

7.1.2 Constraint Transformation

Constraints play a major role in any design by restricting the design search space from an open-ended search to a more restrictive (and hopefully, of polynomial time) search. In other words, constraints could be thought of as a guiding mechanism for evolving a design along some restricted path.

In the present case, constraints are categorized into three separate categories for ease of treatment/management. These are: <relational>, <causal> and <spatial> .

Relational constraints are functions relating attributes (or parameters of attributes) according to some physical law or some other restrictions.

<relational> ::= **f**(<attribute_name>[,<attribute_name>]…) EQ <value_range>
The function **f** could be of three types: explicit, implicit or parametric.
<explicit|implicit> ::= $f(X) \in \mathbf{R}$, $X \in \mathbf{R}^n$
<parametric> ::= $f(X(t)) \in \mathbf{R}$, $t \in \mathbf{R}^n$: $t_j \in (0,1)$ & $X_j = X_{j0} + t_{j*} (X_{j1} - X_{j0})$

If **f** is a vector valued function, it could be treated as a set $(f_1,f_{2,\ldots}f_n)$ of *n* scalar functions such that $f_j \in \mathbf{R}$, $j\in(1,n)$.

For example, in a rotary motion transformation, a global constraint requiring a speed ratio (assuming omegaI as input and omegaO as output rotary speed) could be:

PS.omegaI.value/PS.omegaO.value EQ (5/6) ; a reduction of 5 to 6 is desired

We have mentioned during discussion on artifact representation that a range (interval) will be accepted as a possible value for any parameter. Since relational constraints are functions involving such parameters, we have used standard interval arithmetic to treat these types of values.

A constraint defined by $f(y_1,y_2, y_3, \ldots y_n) = 0$ would be converted to a set of n equations, by solving for each y_j in terms of the others.

$$y_1 = f_1(y_2, y_3, y_4 \ldots)$$
$$y_2 = f_1(y_1, y_3, y_4 \ldots)$$
$$y_3 = f_1(y_1, y_2, y_4 \ldots)$$
$$\ldots.$$

If such an explicit representation is not possible, the constraint may have to be represented in a different way, either by linearizing about some operating point, or by approximating into simpler forms.

If an attribute of an artifact is linked/related to another attribute in a linked artifact, two possible cases occur: an output attribute goes as an input to the next artifact or an input attribute comes out as an output. In either case, we use the corresponding component of the constraint and solve for the new range for the parameter. This new range accompanies the attribute as a constraint to the next artifact. In the next artifact, there may be a priori knowledge about the range of an attribute within which that artifact operates. In order that the incoming attribute value range is acceptable, an intersection of the two intervals is performed as: $P_{in} \cap P_{allowable.}$ If the intersect is NULL, there is a contradiction and the constraints associated with the incoming attribute P makes the new artifact unsuitable for a possible element of the artifact tree.

Spatial Constraints

These constraints are applicable to attributes that belong to the structural aspects of artifacts. These would represent spatial (structural) relationship between attributes having shape/size/orientation -related properties.

```
<spatial> ::= <attribute> <spacial_relationship> [<attribute>] [a_value]
<spatial_relationship> ::= <orientation><position><connection>
<orientation> ::= <direction cosine of major axis of attribute1 w.r.t. that of some attribute2> .
<position> ::= co-ordinate of center of attribute1 w.r.t. center of  attribute2
<connection> ::= <connection_type><contact_details>
<connection_type> ::= <point2point|point2surface|surface2surface|etc…>
<contact_details> ::= set of points, surface, and common dof of connection.
```

Some common orientations are: horizontal, vertical, perpendicular_to, parallel_to, distance_from, etc. As for example, for a chair, we might have following spatial constraints:

```
(‘arm’ parallel_to ‘seat’ )
(‘backrest’ perpendicular_to ‘seat’)
(‘seat’ horizontal_to ‘base’)
(‘seat’ distance_from ‘base’ 2 ft)
              ….
```

7.1.3 Variation of Internal Parameters of Artifacts for Selecting an Artifact

As it has been pointed out in earlier discussion in this section, artifacts are searched from the artifact library by matching input parameter types for possible candidates in the solution. However, a suitable measuring and optimizing criteria would be required for guiding the solution. In other words, some criteria for selecting the ‘best’ possible candidate at each stage from a possible set of artifacts have to be formulated.

We define a ‘distance’ type norm for measuring the proximity between the desired output (as specified in PS) and the partial solution reached at some stage j, as:

$d(a,b) = ((a_{low}-b_{low})^2 + (a_{high}-b_{high})^2)^{1/2}$
where a and b are two variables representing intervals $a = [a_{low}, a_{high}]$ and $b = [b_{low}, b_{high}]$

Above definition satisfies properties of a norm:

$d(a,b)=0$ iff $a_{low} = b_{low}$ and $a_{high} = b_{high}$
$d(a,b)>0$ for $a != b$
$d(a,b) = d(b,a)$

We would, sometimes, use a parametric form to represent intervals a and b.
As for example, $a = a_{low}+(a_{high}-a_{low})*\theta : \theta \in [0,1]$

While the range of feasible variations of the input will be used to check for suitability of accepting an artifact, the variations allowed in the internal parameters of the artifact would be used to minimize the ‘distance’ between the desired output as specified in PS and the output (partial solution) at the present stage. The minimization scheme is formulated as below:

Minimize: $d(O_j, O_0)$

where, O_0 is the output specified in the PS and O_j is the output from the artifact j in an intermediate stage of the design.

The partial solution O_j is given by: $O_j = f_j (I_j, \beta_j)$, which is derived from the main constraint C_0 (relationship between the input and the output of the artifact j, vide section 6.2), by solving for O_j from $C_{j0} (I_j, O_j , \beta_j) = 0$.

The parameter β (where $\beta_j = (\beta_{j1,} \beta_{j2,\dots} \beta_{jn})$) is the internal parameter of artifact j. The parameter β is expressed in parametric form as: $\beta_{jk} = \beta_{jk\_Low} + (\beta_{jk\_High} - \beta_{jk\_Low})* \theta_{jk}$: $\theta_{jk} \in (0,1)$, $k \in (0, n)$. The subscripts Low and High indicate the lower and upper bounds of the interval for β_{jk}.

It is also possible that apart from the C_o constraint, an artifact may have additional relational constraints associated with it. These relational constraints are expressed as: $C_k(I, O, \beta)$ <rel_opr> <value> , k>0 and <rel_opr> is the relational operator (one of {LT LE EQ GE GT NE}), and <val> is a numeric value. For the optimization scheme, these relationships are converted to the standard equality form $C_k(I, O, \beta) = 0$, by introducing additional variables for the cases where the <rel_opr> is not "EQ").

The input to the artifact *j*, I_j is equal to the output from the previous artifact *j-1* and so on. These gives rise to the chain of linked equations and the optimization scheme becomes:

Minimize: $d(O_j, O_0)$

subject to:
constraints associated with artifact j
$C_{j0} (I_j, O_j , \beta_j) = 0$
$C_{j,1} (I_j, O_j, \beta_j) = 0$
…
$C_{j,c(j)} (I_j, O_j, \beta_j) = 0$

$I_j = O_{j-1}$
; constraints associated with artifact j-1
$C_{j-1,0} (I_{j-1}, O_{j-1,} \beta_{j-1}) = 0$
$C_{j-1,1} (I_{j-1}, O_{j-1}, \beta_{j-1}) = 0$
…
$C_{j-1,c(j-1)} (I_{j-1}, O_{j-1}, \beta_{j-1}) = 0$

$I_{j-1} = O_{j-2}$
; constraints associated with artifact j-2
$C_{j-2,0} (I_{j-2}, ,O_{j-2} , \beta_{j-1}) = 0$
$C_{j-2,1} (I_{j-2}, O_{j-2}, \beta_{j-2}) = 0$
…
$C_{j-2,c(j-2)} (I_{j-2}, O_{j-2}, \beta_{j-2}) = 0$
…

$I_2 = O_1$
; constraints associated with artifact 1

$C_{1,0}(I_1, O_1, \beta_1) = 0$
$C_{1,1}(I_1, O_1, \beta_j) = 0$
…
$C_{1,c(1)}(I_1, O_1, \beta_1) = 0$

Above minimization scheme could be solved using Lagrange multiplier scheme by including the constraints into the main optimization function as:

$$\begin{aligned} d_j = \quad & d(O_j, O_0) + \\ & \Sigma_{n \in (1,j)} \Sigma_{k \in (1, c(n))} (\mu_{n,k} * C_{n,k}(I_n, O_n, \beta_n)) + \\ & \Sigma_{p \in (1, j-1)} (\lambda_p * (I_{p+1} - O_p)) \end{aligned}$$

where c(n) is the number of constraints associated with artifact n, and μ's and λ's are Lagrange multipliers.

The minimization of d_j produces a set of parameters (β^*_n), for each artifact n ($n \in (1, j)$), which makes the present solution closest to the desired solution. We denote by d_j^* and O_j^* the corresponding optimal distance and output. If the value of $d_j^*(O_j^*, O_0)$ is within a specified value $\in_0$ (convergence criterion), we can accept the current design solution given by D_j = {<PS$_j$><Art_Tree$_j$>} as a feasible solution. However, if the distance d_j^* is not within acceptable limit, the solution at this stage represents a partial (an incomplete) solution i.e. the desired output value has not yet been achieved yet .

The above minimization process deals with the relational constraints only. After the minimization has been performed, (irrespective of the solution whether an acceptable feasible solution or a partial solution), the spatial constraints are then checked. There can arise four situations after the spatial constraints are applied.

i) A feasible solution has been achieved and the spatial constraints are all satisfied.
ii) A feasible solution has been achieved and all the spatial constraints are not satisfied.
iii) An incomplete solution has been achieved and the spatial constraints are all satisfied.
iv) An incomplete solution has been achieved and not all the spatial constraints are satisfied.

The case i) represents a complete solution and the corresponding branch of the tree can be terminated without further growth. The rest of the three cases are incomplete and the branching / growth of the solution tree continues to the next stage.

7.2 Design Synthesis Process

The basic procedure for the design synthesis is as follows:

0. Develop design domain specific artifact library (ARTL), functional equivalence library (FUNL) and domain knowledge base (DK). For the time being, we assume that the DK is specified in the form of constraints and relations in the PS itself. However, these could be separated out for treating them in a generic way.

1. Start with a product specification PS.

2. Locate suitable artifacts from the ARTL mapping the input parameters from the product specification with those of the artifacts having same input type. If no artifacts are found, go to step 8.

3. Check whether the type of output from some of these artifacts matches the output types specified in PS. Divide the artifacts into two sub-groups: one with artifacts whose output matches the desired output (D_M) and other where such a match is not found (D_{NM}).

4. a) With D_M

   Generate the distance function between the output and the desired output in PS and minimize the distance along with the constraints associated with the attributes.

   1. If the distance for some of the artifacts is within a specified acceptable value, a possible solution has been found.
   2. Apply the spatial constraints to these artifacts. If these constraints are satisfied, go to step 10
   3. If the distance is not within the acceptable value, only a partial solution has been found. Take the top N_{alt} artifacts nearest to the solution. Go to step 5.

   b) With D_{NM}

   In this case, the minimization criteria can not be applied yet since the output type did not match the desired output type in the PS. The minimization scheme can only be applied when a match for the desired output has been found.

5. Generate new sets of attributes and transform the constraints to augment the PS so that additional attributes associated with the selected artifacts could be taken into account.

6. Repeat steps 2 to 5 with transformation of the product spec PS.

7. Continue until such time as all the attribute requirements are satisfied or some attributes could not be mapped.

8. At any stage, if some attributes could not be mapped, there would be three alternatives: look for a possible functional equivalence class and modify the PS accordingly and continue search. If such a functional equivalence class is not found, consult with the designer to acquire new attributes, knowledge, constraints and/or modify existing specification. Repeat steps 2-5 after such modifications. If above steps still fail to map some attribute requirements, the designer needs to add new artifacts in ARTL and/or add new functional equivalence classes in Function Library. After this step, repeat again.

9. In case all options are exhausted at an intermediate stage, consider the possibility of going back one step and consider other paths with artifacts with lesser matches.

10. After a feasible solution has been found, a tentative sizing of the components of the artifacts is carried out by using the attribute values specified and by applying the physical laws governing the behavior of the artifact. If during this process, some parts could not be sized within acceptable range of values, consider possible change of the PS and go to step 7.

11. Introduce tolerance models associated with each artifact in the artifact tree and carry out tolerance analysis. If during this process, tolerance requirements for some parts are not feasible, consider changing PS and go to step 7.

12. Consider manufacturability of the artifacts in the design solution. Apply criteria for manufacturability. If during this process, some manufacturing requirements for some parts are not feasible, consider changing PS and go to step 7.

13. Consider global goals and constraints associated with the product specification. If the global constraints are satisfied, initiate global optimization processes and consider changing the PS again to achieve some global goals and go to step 7.

14. A feasible design has been found.

The iterative design synthesis process will terminate when one of the following is satisfied.

1. All the attributes in the PS are found and the desired output value level has been achieved. In this case, a feasible solution has been found. The global constraints and goals can now be evaluated.

2. Some of the attributes are yet to be found and no further artifact could be located in ARTL. In this case, either the designer will provide some more domain-specific knowledge in the PS_n or some new artifacts would be added to proceed further. However, to explore other possible solutions, we may backtrack one step to D_{n-1} and consider other less favorable possibilities.

General observations on the design synthesis process

In general, an artifact may have more than one input and output attributes and constraints associated with them. To consider the artifact as an element of the solution, these attributes are also to be considered as part of the design specification. Thus, we need to augment the design specification with the unsatisfied attributes of this artifact.

If some of the input and output attributes are already in PS, we mark them as found and the remaining attributes need to be satisfied. Since these inputs and outputs were not in the original product specification, they are not desirable from the product specification requirement. However, these must be mapped to other artifacts. We would put a negative weight to these attributes (undesirable?) and augment the PS with these new sets of attributes along with associated constraints.

With this augmented PS, we will now search for artifacts from the ARTL. The input attributes must come as output from some other artifact or from a *terminal* which we also consider as artifact with no input and one output (like an electricity supply point as a terminal that supplies electric energy and need no further input)

The output attributes must either be accepted as an undesirable byproduct to the environment and no further exploration would be required or the output must be mapped as in input to some other artifact.

As an example, the above proposed design procedure has been applied to a simple design problem and the steps are elaborated in the next section.

7.3 Design Synthesis Example

Design problem statement: Design "a device to increase the rotational speed by a factor of 10.

The product specification is as below:

PS = (
- (function rotation to rotation)
- (input w_I rotary_motion rad/sec 0 w_{i0})
- (output w_o rotary_motion rad/sec 0 w_{o0})
- (constraint relational $w_o = w_i / 10$)
- (constraint spatial (axis(w_I) parallel_to axis(w_o))
- // some other attributes, not essential for the example are left out.

)

Let $PS_0 = PS$. The design is started with PS defined above and solution $D_0 = (\{PS_0\}, \{null\})$

To make the example easier to follow, we will use $A_{i,j}$ to indicate artifact A_i as used in stage $_j$

The attribute $PS_0.w_I$ is an 'input' and the type of the 'input' is 'rotary_motion'. Searching the artifact library **ARTL**, we find artifacts $\{A_1, A_2, A_3, A_4, A_5, A_6, A_7, A_8\}$ all having the same 'input' category with 'rotary_motion' as type. Hence $D_0 = (\{PS_0\}, \{A_1, A_2, A_3, A_4, A_5, A_6, A_7, A_8\})$. The set of artifacts $\{A_1, A_2, A_3, A_4, A_5, A_6, A_7, A_8\}$ can satisfy the 'input' requirement. Thus we set

$D_0=(\{PS_0\}, \{A_{1,0}, A_{2,0}, A_{3,0}, A_{4,0}, A_{5,0}, A_{6,0}, A_{7,0}, A_{8,0}\})$

There are no more 'input' category attributes. We now check whether the outputs of these artifacts match the output specified in the PS. We find that artifacts $\{A_{4,0}, A_{6,0}, A_{7,0}\}$ meet this requirement. Thus at this stage, we have two distinct subsets of D_0 , $D_{01}= (\{DP_0\}, \{A_{4,1}, A_{6,1}, A_{7,1}\})$ and $D_{02} = (\{DP_0\} \{A_{1,1}, A_{2,1}, A_{3,1}, A_{5,1}, A_{8,1}\})$. While each artifact in D_{01} can meet both input and output requirements of PS, those in D_{02} only meet the input requirement.

The two sets D_{01} and D_{02} require different treatments and we split the design synthesis into two segments.

Segment #1: With D_{01}

Since both input and output requirements have been found, we apply the constraints and see how far the constraints are satisfied:

For artifact $A_{4,1}$:

Its internal relationship is: $w_O = w_I * R$, where $R=[0.25,4.0]$
In parametric form, it becomes $R = 0.25+3.75*\theta$, where $\theta \in (0,1)$.
Minimization scheme: minimize $d(PS_0.w_o, D_1.A4.w_o)$
$d = d(10*w_I, (0.25+3.75*\theta)*w_I)= w_I*(9.25-3.75*\theta)$
$\Rightarrow \theta^* = 1.0$ for minimum $d = 4*w_i$.

With this θ^*, d is not yet zero, so the artifact A41 can only meet the output requirement partially. We need to add some other artifacts to the artifact 4 to reach the desired output. In this stage, we have $(\theta^*, w_o^*) = (1, 4.0 * w_i)$

Similar computations with artifact $A_{6,1}$ and $A_{7,1}$ also leads to incomplete fulfillment of the desired output level, which are $(\theta^*, w_o^*) = (1, 3.0 * w_i)$, $(\theta^*, w_o^*) = (1, 4*w_I)$ respectively.

Now, applying the spatial constraints, we see that:
Art $A_{4,1}$: axis(w_1) parallel_to axis(w_2) IS TRUE
Art $A_{6,1}$: axis(w_1) parallel_to axis(w_2) IS TRUE
Art $A_{7,1}$: axis(w_1) parallel_to axis(w_2) IS FALSE

Thus, at this stage, by considering the above two constraints (both relational and spatial) satisfaction criteria, we grade artifact $A_{4,1}$ as the most appropriate artifact, followed by artifact $A_{6,1}$ and artifact $A_{7,1}$.

For the next stage, to search for new artifacts that can take outputs from the above 3 artifacts (artifacts $\{A_{4,1}, A_{6,1}, A_{7,1}\}$), we transform their corresponding outputs to the desired inputs for the new artifacts.

The spatial constraint has been satisfied for artifacts $A_{4,1}$ and $A_{6,1}$ and would need no further transformation; however, the spatial constraints for artifact $A_{7,1}$ has not been satisfied and needs to be transformed as: axis(w_1) perpendicular_to axis(w_2). This is because the current unsatisfied constraint axis(w_1) parallel_to axis(w_2) (an angle of zero) could not be met. The constraint axis(w_1) perpendicular_to axis(w_2) (an angle of 90^0) is converted to axis(w_1) perpendicular_to axis(w_2) (an angle of 90^0), so that the combined effect of another artifact which meets this requirement would produce 0 or 180 as parallel, the final requirements.

With these modified specifications, we carry out the search again.

First we take the case of artifact $A_{4,1}$ and find that all the artifacts A1 - A8 can take the corresponding output as input.

Applying the same procedure as it was followed in the previous stage, we come across artifacts D_2= $\{A_{4,2}, A_{6,2}, A_{7,2}\}$ as possible candidates. Applying the norm minimization criteria:

For Artifact $A_{4,2}$:

Its internal relationship is: $w_O = w_I * R$, where R=[0.25,4.0]
In parametric form, this becomes $R = 0.25+3.75*\theta$, where $\theta \in (0,1)$.
Minimization scheme: minimize $d(PS_0.w_o, D_2.A4.w_o)$
$d = d(10*w_I, (0.25+3.75*\theta)*4*w_I)= w_I*(10-1-3.75*4*\theta)$
$\rightarrow (\theta^{*,} w_o^{*}) = (0.6, 2.5 * w_I)$ for minimum d=0.

Thus, $D_2 = (\{\{PS_1\}, \{A_{4,1}\}\}, \{\{PS_2\}, \{A_{4,2}\}\})$ is a solution that meets the output requirements.
Applying similar optimization criteria to the other two artifacts $A_{6,1}$ & $A_{7,1}$, we get optimal solution as: $(\theta^{*,} w_o^{*}) = (1.0, 2.0 * w_I)$ and $(0.6, 2.5 * w_I)$ respectively. The corresponding distances are $2*w_I$ and 0 respectively. Now applying the spatial constraints,

Art $A_{4,2}$: axis(w_1) parallel_to axis(w_2) IS TRUE
Art $A_{6,2}$: axis(w_1) parallel_to axis(w_2) IS TRUE
Art $A_{7,2}$: axis(w_1) parallel_to axis(w_2) IS FALSE

Thus after this stage, one feasible solution ($A_{4,1} \rightarrow A_{4,2}$) and a partial solution ($A_{4,1} \rightarrow A_{6,2}$) are found. For the third combination ($A_{4,1} \rightarrow A_{7,2}$), though the distance is zero, the spatial constraint can not be satisfied.

The same procedure with artifacts $A_{6,1}$ and $A_{7,1}$ from stage 1 leads to the following situations: ($A_{6,1} \rightarrow A_{4,2}$), ($A_{6,1} \rightarrow A_{6,2}$), ($A_{6,1} \rightarrow A_{7,2}$) as a partial solution and ($A_{7,1} \rightarrow A_{7,2}$) as a feasible solution. The process could be continued till the desired output criteria are met.

A part of the design process is shown in figure 7 where A_{nj} indicates the artifact number n in the **ARTL** as used in design stage j. Some of the solutions found are shown in figure 8.

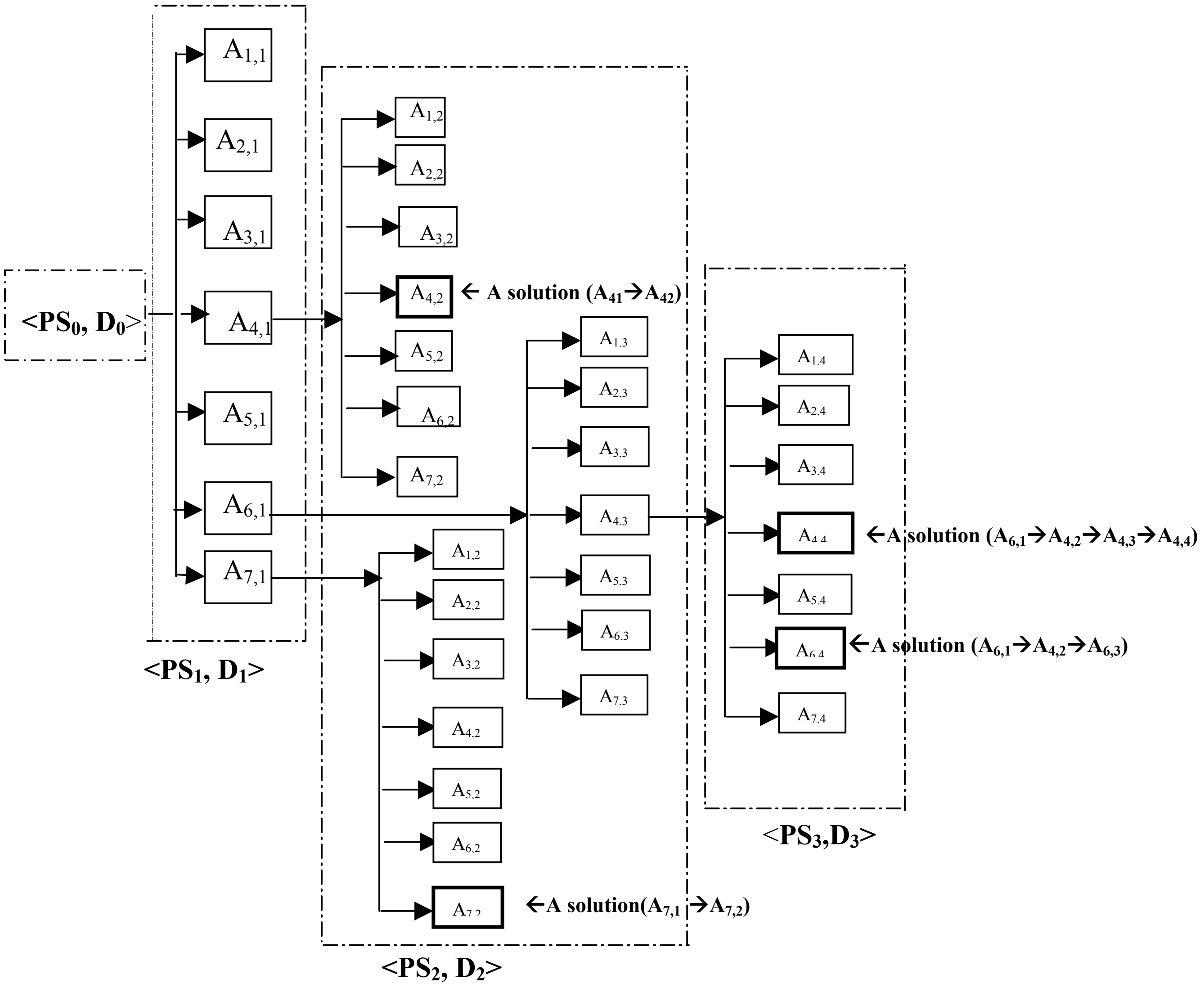
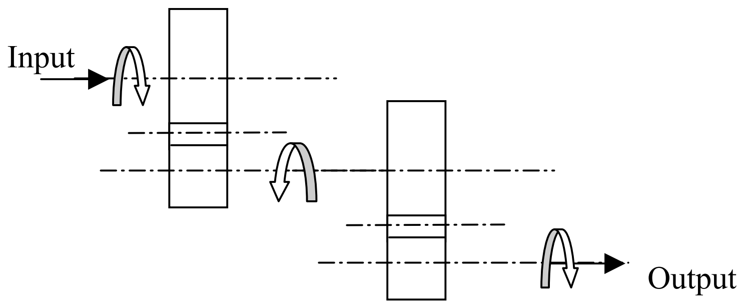
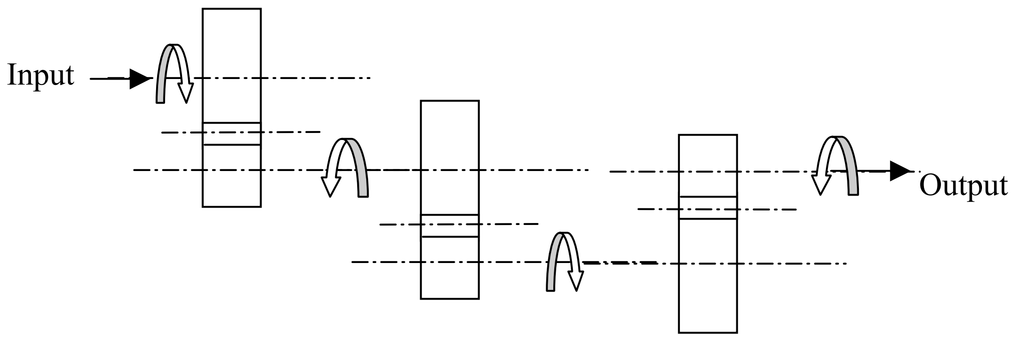

<PS₀, D₀>
<PS₁, D₁>
<PS₂, D₂>
<PS₃,D₃>
A1,1 A2,1 A3,1 A4,1 A5,1 A6,1 A7,1
A1,2 A2,2 A3,2 A4,2 A5,2 A6,2 A7,2
A1,3 A2,3 A3,3 A4,3 A5,3 A6,3 A7,3
A1,4 A2,4 A3,4 A4,4 A5,4 A6,4 A7,4
← A solution (A41→A42)
←A solution (A6,1→A4,2→A4,3→A4,4)
←A solution (A6,1→A4,2→A6,3)
←A solution(A7,1→A7,2)

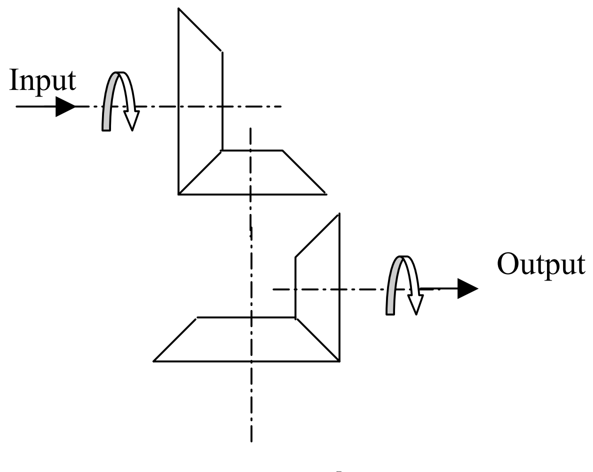
Figure 7. Some Branches of the Example Design Process up to Stage 3.



Solution $A_{4,1} \rightarrow A_{4,2}$



Solution $A_{4,1} \rightarrow A_{6,2} \rightarrow A_{4,3}$



Solution $A_{7,1} \rightarrow A_{7,2}$

Figure 8. Typical Solutions from the Design Example

Segment #2: With D_{02}

The next process is applied to D_{02}, where the output is not the desired output specified in PS. In our example, we have the set $D_{02} = (\{DP_0\} \{A_{1,1}, A_{2,1}, A_{3,1}, A_{5,1}, A_{8,1}\})$.

Since these artifacts do not produce the desired output type (i.e. rotary_motion) as specified in PS, we can not apply the relational constraints specified in the PS. However, we can apply the spatial constraints as well as any local constraint associated with the artifacts, to eliminate some of the artifacts or to assign lesser importance to the artifacts of the set D_{02}. After the constraints are checked, we transform the outputs and the spatial constraints to search for artifacts which can take these outputs as input attributes. If some artifacts could be found, we proceed as it has been stated in Segment #1. However, if nothing could be found, we consult the functional equivalence knowledge base library to check if suitable functional equivalence could be identified which can decompose this output into a collective group of functions. If such functions are found, we continue the search with the new set of inputs to locate some artifacts. If some artifacts are found, we proceed as before. If no such function is found, the design process stops at this stage and further input is required from the designer. There are three possibilities: inject some more design specific domain knowledge either in the PS or in the functional equivalence class library, or add new artifacts.

It may be noted here that the sample artifact library and the functional equivalence class library developed for this design example have no such artifacts. And the design process in this Segment ended here without any feasible solutions with $D_{02.}$

It is postulated that after a finite number of steps, we will get a set of artifact chains, the final element of which has the desired output type. Now, we are in a position to apply the relational constraints and optimize the internal parameters of each artifact to see if the desired level of output has been reached. This problem then becomes a multivariate minimization with constraints.

The minimization process assigns optimum values for the internal parameters of each artifact. After the minimization, if the distance d is within a specified value of $\in_0$, the solution has converged to a feasible solution. However, if d is still not within the range, we continue to add another possible chain of artifacts, and optimize. The process repeats till the desired level has been reached.

8.0 Conclusion

The objective of this study has been to evolve an object-oriented generic approach for design of products encompassing the complete product life cycle from product specification to conceptual design to detailed design and global goal optimization. We have presented a workable scheme to represent product specification, functional requirements, artifact representation, artifact behavior, tolerance representation, synthesis and analysis, and study of kinematic behavior of artifact assemblies. The proposed system has been verified for the design synthesis process with a simple example based on a sample artifact library developed for the design example. However, there are several aspects of the proposed system which need further study/research. Some of the issues are as follows:

i) Development/ adoption of a suitable model to convert natural language product specification to the proposed formal product specification through natural language modeling interface.
ii) Detailed study of artifact functional behavior (both qualitative and quantitative) as well as kinematic behavior using suitable behavior modeling tools.
iii) Further study of tolerance synthesis and analysis schemes for allocation of tolerances and the effect of such tolerance allocation on the product manufacturability, assembliability and cost.
iv) Schemes for optimization of global goals associated with the final product to further improve the design.

v) Study of computational aspects associated with the proposed system including development of object classes, development of a suitable user interface and inclusion of some expert system tools for qualitative reasoning as well as assisting the design synthesis process.

vi) Development of mathematical theorems to establish convergence criteria for the proposed design synthesis process and to evaluate the complexity of the proposed algorithms.

vii) To apply the proposed design synthesis procedure to more complex design problems taken from various design domains to establish the efficacy of the system in handing a general class of product design. This task would involve development of suitable artifact and function libraries for each design domain.

Since this report is devoted to the study of function-to-form mapping in the product development context, large scale assembly issues including the intricate problem of evolving both the assembly structure and its associated tolerance information simultaneously have not been addressed. In this report, we have mainly concentrated on a single artifact-pair relation. Though our proposed model is generic, we need to address system issues such as assembly-oriented tolerance synthesis & analysis, and tolerance propagation.

Acknowledgments

The authors thank the anonymous referees for their valuable suggestions for the improvement of the paper.

This work is sponsored by the SIMA (Systems Integration for Manufacturing Applications) program in NIST and the RaDEO (Rapid Design Exploration and Optimization) program at DARPA (Defense Advanced Research Project Agency).

References

[1] Daniel E. Whitney, *Electro-mechanical Design in Europe: University Research and Industrial Practice.* The Charles Stark Draper Laboratory, Inc. Cambridge, MA 02139, October, 1992.

[2] G. Pahl, and W. Beitz. *Engineering Design*. Springer-Verlag, 1984.

[3] S. Kota. *A Qualitative Matrix Representation Scheme for the Conceptual Design of Mechanisms.* In Proc. of ASME Design Automation Conference (21st Biannual ASME Mechanisms Conference), pp. 217-230, 1990.

[4] M. S. Hundal, and J. F. Byrne. *Computer-Aided Generation of Function Block Diagrams in a Methodical Design Procedure.* In Proc. of Design Theory and Methodology- DTM'91 Conference, Volume DE-Vol.27, ASME, pp. 251-257, 1991.

[5] G. Iyengar, C-L Lee, and S. Kota. *Towards an Objective Evaluation of Alternate Designs*. ASME Journal of Mechanical Design, Vol. 116, pp. 487-492, 1994.

[6] H. Schmekel. *Functional Models and Design Solutions*. In Annals of CIRP, Volume 38, pp. 129-132, 1989.

[7] S. M. Kannapan, and K. M. Marshek. *Design Synthesis Reasoning: A Methodology for Mechanical Design*. Research in Engineering Design, Vol. 2, No. 4, pp 221-238, 1991.

[8] S. P. Hoover, and J. R. Rinderle. *A Synthesis Strategy for Mechanical Devices*. Research in Engineering Design, Vol. 1, No. 2, pp 87-103, 1989.

[9] S. Finger, and J. R. Rinderle. *A Transformational Approach to Mechanical Design Using Bond Graph Grammar*. In Proc. of 1st ASME Design Theory and Methodology Conference, pp. 197-216, ASME, 1989.

[10] K. T. Ulrich and W. P. Seering. *Synthesis of Schematic Descriptions*. Research in Engineering Design, 1:3-18, 1989.

[11] R. H. Bracewell, R. V. Chaplin, P. M. Langdon, M. Li, V. K. Oh, J. E. E. Sharpe, and X. T. Yan. Integrated Platform for AI Support of Complex Design. In AI System Support for Conceptual Design (Editor: J. E. E. Sharpe), Springer-Verlag, 1995.

[12] J. K. Gui, and M. Mantyla. *Functional Understanding of Assembly Modeling*. Computer Aided Design, Vol. 26, No. 6, pp. 435-451, 1994.

[13] J. E. Baxter, N. P. Juster, and A. de Pennington. *Verification of Product Design Specifications Using a Functional Data Model*. MOSES Project Research Report 25, University of Leeds, April 1994.

[14] B. Henson, N. P. Juster, and A. de Pennington. *Towards an Integrated Representation of Function, Behavior and Form.* MOSES Project Research Report 16, University of Leeds, April 1994.

[15] A. Wong, and D. Sriram. *SHARED: An Information Model for Cooperative Product Development*. Research in Engineering Design, 1993.

[16] D. Sriram, R. Logcher, A. Wong, and S. Ahmed. *An Object-Oriented Framework for Collaborative Engineering Design*. Computer-Aided Cooperative Product Development (Editors: D. Sriram, R. Logcher and S. Fukuda), Springer-Verlag, New York, 1991.

[17] D. Sriram, K. Cheong, and M. Lalith Kumar. *Engineering Design Cycle: A Case Study and Implications for CAE.* Chapter 5, Knowledge Aided Design, Knowledge-Based Systems, Vol. 10 (Editor: Marc Green), Academic Press, New York, pp. 117-156, 1992.

[18] S. R. Gorti, and Ram D. Sriram. *From Symbol to Form: A Framework for Conceptual Design*. J. Computer-Aided Design, Vol. 28, No. 11, 1996, pp. 853-870.

[19] F. L. Krause, F. Kimura, T. Kjellberg, and S. C-Yu Lu. *Product Modeling.* Annals of CIRP, Vol. 42, No. 2, pp. 695-706, 1993.

[20] S. Y. Reddy. *Hierarchical and Interactive Parameter Refinement for Early-Stage System Design*. PhD Thesis, University of Illinois at Urbana-Champaign, 1994.

[21] S. Finger, M. S. Fox, F. B. Prinz, and J. R. Rinderle. *Concurrent Design*. Applied Artificial Intelligence, Vol. 6, pp. 257-283, 1992.

[22] S. Finger, M. S. Fox, D. Navinchandra, F. B. Prinz, and J. R. Rinderle. *Design Fusion: A Product Life-cycle View for Engineering Designs*. Technical Report EDRC 24-28-90, EDRC, CMU, 1990.

[23] M. R. Cutkosky, R. S. Engelmore, R. E. Fikes, M. R. Genesereth, T. R. Gruber, W. S. Mark, J. M. Tenenbaum, and J. C., Weber. *PACT: An Experiment in Integrating Concurrent engineering Systems.* IEEE Computer, pp. 28-37, 1993.

[24] U. Roy, R. Sudarsan, R. D. Sriram, K. W. Lyons, and M. R. Duffey, "Information Architecture for Design Tolerancing: from Conceptual to the Detail Design," accepted for presentation and publication in the Proc. of DETC'99, 1999 ASME International Design Engineering Technical Conferences, September 12-15, 1999, Nevada, Las Vegas, USA.

[25] U. Roy, R. Sudarsan, Y. Narahari, R. D. Sriram, K. W. Lyons, and N. Pramanik. *Information Models for Design Tolerancing: From Conceptual to the Detail Design.* Technical Report, National Institute of Standards and Technology, 1999 (in preparation).

[26] Szykman, S., J. W. Racz and R. D. Sriram, "The Representation of Function in Computer-based Design," Proceedings of the 1999 ASME Design Engineering Technical Conferences (11th International Conference on Design Theory and Methodology), Paper No. DETC99/DTM-8742, Las Vegas, NV, September, 1999.

[27] B. Chandrasekaran, and J. R. Josephson. *An explication of Function.* Laboratory for AI Research, The Ohio State University, Columbus, OH, January 1996

[28] A. C. Thornton. *Genetic Algorithms Versus Simulated Annealing: Satisfaction of Large Sets of Algebraic Mechanical Design Constraints*. Artificial Intelligence in Design'94, (Editors: J. S. Gero and F. Sudweeks), Kluwer Academic Publishers, The Netherlands, 1994.

[29] B. Bharadwaj. *A Framework for Tolerance Synthesis of Mechanical Components*. Master's Thesis. Syracuse University, Syracuse, NY, 1995.
.

APPENDIX – I

Sample Artifact Library (ARTL)

Artifact	**Cam Follower (Lin)**	**Cam Follower(Osc)**	**Rack & Pinion**	**Spur Gear Box**
id	**A1**	**A2**	**A3**	**A4**
figure ref	figure 1-2	figure 1-1	figure 1-4	figure 1-6
Function	rotary_to_linear	rotary_to_rotary	rotary_to_linear	rotary_to_rotary
purpose	rotary to linear oscillating	rotary to rotary oscillatory	rotary to linear	rotary to rotary
type				
equivalence				
Input	w_In	w_In	w_In	w_In
category	rotary_motion	rotary_motion	rotary_motion	rotary_motion
weight	1	1	1	1
unit	rpm	rpm	rpm	rpm
from val	100	100	100	100
to val	300	300	300	300
Internal Par	E (eccentricity)	R (range_angle)	R (pinion radius)	R (speed_ratio)
category	-	-	-	-
weight	1	1	1	1
unit	mm	degree	-	-
from val	0	0	?	0.25
to val	?	60	?	4
Output	x(x_range)	w_Out	x_vel	w_Out
category	linear_oscillatory	rotary_oscillatory	linear_motion	rotary_motion
weight	1	1	1	1
unit	mm	cpm	mm/sec	rpm
from val	unknown			
to val				
Constraint	C0	C0	C0	C0
type	relational	relational	relational	relational
expression	x = 2*E	w_out = w_in	x_vel=w_in*R	w_out = w_in*R
Constraint	C1	C1	C1	C1
Type	spatial	spatial	spatial	spatial
Expression	axis(w_in) _|_ axis(x)	axis(w_in) || axis(w_out)	axis(w_in) _|_ dir(x_vel)	axis(w_in) || axis(w_out)
Artifact Ref	cam	cam	rack	spur_gear
Artifact Ref	follower	follower	pinion	spur_gear
Behavior	x(t)=2*E*cos(w_in*t)	w_out(t)=w_in(t)	x_vel(t)=w_in(t)*R	w_out(t)=w_in(t)*R

Artifact	**4-Bar Linkage**	**Belt Pulley**	**Bevel Gear Box**	**Nut & Bolt**
id	**A5**	**A6**	**A7**	**A8**
figure ref	-	-	figure 1-5	figure 1-3
Function	rotary_to_rotary	rotary_to_rotary	rotary_to_rotary	rotary_to_linear
purpose	to transmit rotary motion	rotary to rotary	rotary to rotary motion	rotary to linear
type				
equivalence				
Input	w_In	w_In	w_In	w_In
category	rotary_motion	rotary_motion	rotary_motion	rotary_motion
weight	1	1	1	1
unit	rpm	rpm	rpm	rpm
from val	100	100	100	100
to val	300	300	300	300
Internal Par	Lengths of the bars	R(speed_ratio)	R(speed_ratio)	P (pitch)
category	-	-	-	-
weight	1	1	1	1
unit	mm	-	-	mm
from val	?	0.5	0.25	1
to val	?	2	4	5
Output	w_Out	w_Out	w_Out	x_vel
category	rotary_motion	rotary_motion	rotary_motion	linear_motion
weight	1	1	1	1
unit	rpm	rpm	rpm	mm/sec
from val				
to val				
Constraint	C0	C0	C0	C0
type	relational	relational	relational	relational
expression	w_out = w_in*R	w_out = w_in*R	w_out = w_in*R	x_vel = w_in*P
Constraint	C1	C1	C1	C1
type	spatial	spatial	spatial	spatial
expression	axis(w_in) || axis(w_out)	axis(w_in) || axis(w_out)	axis(w_in) _|_ axis(w_out)	axis(w_in) || dir(x_vel)
Artifact Ref	spur_gear	belt	bevel_gear	nut
Artifact Ref	spur_gear	pulley, pulley	bevel_gear	bolt
Behavior	w_out(t)=w_in(t)*R	w_out(t)=w_in(t)*R	w_out(t)=w_in(t)*R	x_vel(t)=w_in(t)*P

Note: This table is a simplified version of the actual artifact representation as objects. Only the basic features required to follow the design synthesis example given in this paper are shown here.

Sketches of some of the above artifacts are given in the following page.

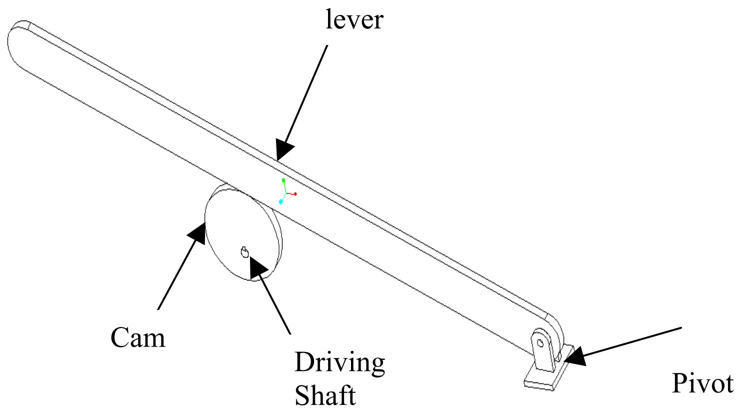


Figure 1-1. Cam and Follower (Oscillating)

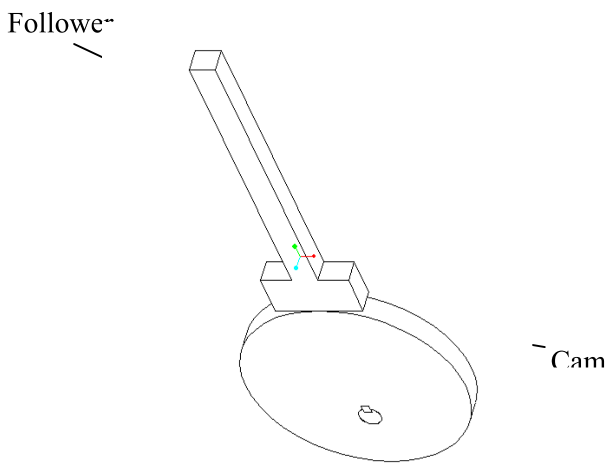


Figure 1-2. Cam and Follower (Linear)

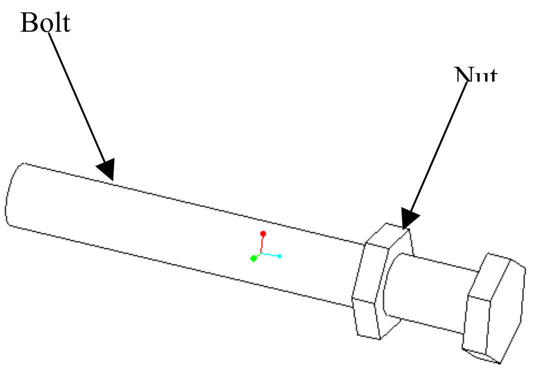


Figure 1-3. Nut and Bolt

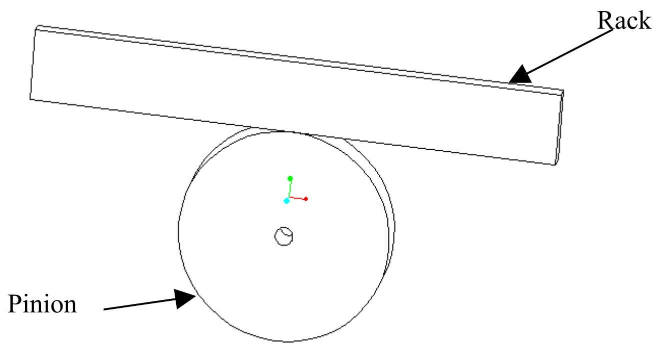


Figure 1-4. Rack and Pinion

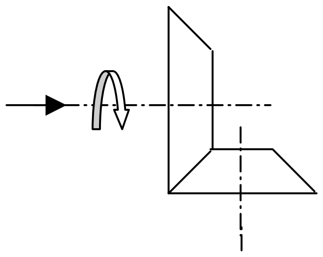


Figure 1-5. Bevel Gear Box

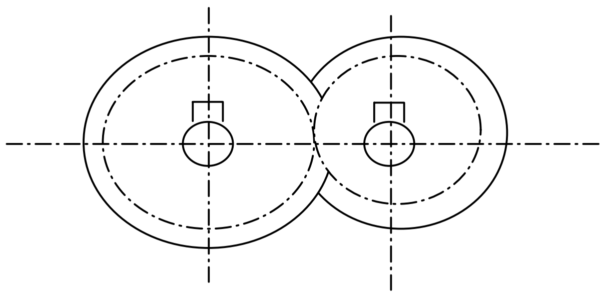


Figure 1-6. Spur Gear Box

APPENDIX – II

Representation of Products and Artifacts in Object-oriented Form.

Example - 1: A product specification using definition given in section 3, for designing "a device to increase the rotational speed by a factor of 10" could be represented as shown below:

```
PS: a_product =  // a device to increase rotary speed
(
        (attr (name fff) (type function) (rf 1) (category rotation_to_rotation) )
        (attr (name omegaI) (type input) (rf 1) (category rotary_motion) (unit rpm) (value (range 200  300))
        )
        (attr (name omegaO) (type output) (rf 1) (category rotary_motion) (unit rpm))
        (attr (name speed_ratio)(type internal) (category NONE) (unit NONE)     (value (range 10 10))
        )
        (constraint (relational (omegaO/omegaI EQ speed_ratio)))
        (constraint (spatial (axis(omegaI) parallel_to axis(omegaO)))
        (goal (minimize weight()))
        ….
  )
```

Example – 2: Functional representation

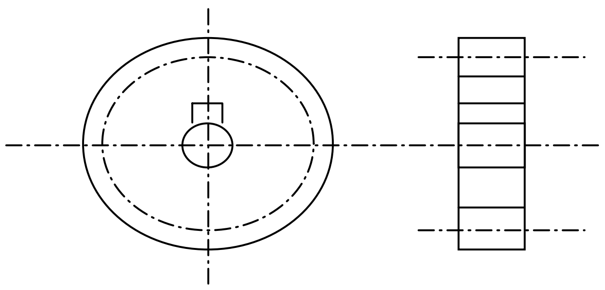
```
        Function: rot2osc =  // function to convert  rotary motion to oscillatory motion
        (
                (fid rotary_to_linear_oscillatory)
                (input (iid omegaI)(category  rotary_motion)(weight 1)(unit rpm) (value_range UNKNOWN UNKNOWN)
                )
                (output (oid )(category  oscillatory_motion)(weight 1)(unit cycles/sec) (value_range UNKNOWN UNKNOWN)
                )
                (artifact UNKNOWN) // not yet set in the generic definition
                (constraint NONE)
(eqv (function rot_to_linear AND function linear_to_oscillatory))
                (goal UNKNOWN)
        )
```

Example – 3: Representation of behavior

```
Behavior: gearbox // behavior of  a gear box
        (
                (state_var omegIn
                        (causal_link
                                (depends_on NONE) // its an input
                                (affects_var omegaOut)
                        )
                        (val_ref  external)
                )
                (state_var speedRatio  // internal time-independent parameter
                        (causal_link NONE)
                        (val_ref  NONE)
                )
                (state_var omegOut
                        (causal_link
                                (depends_on omegaIn)
                                (affects_var NONE) // it's an output
                        )
                (val_ref  procedural
                        ( EQ omegaIn * speedRatio )
                )
        )
```

Example – 4: Representation of an artifact. First, the representation of a single spur gear is given, followed by a spur gear box that uses the artifact spur gear in its definition.

A spur gear:



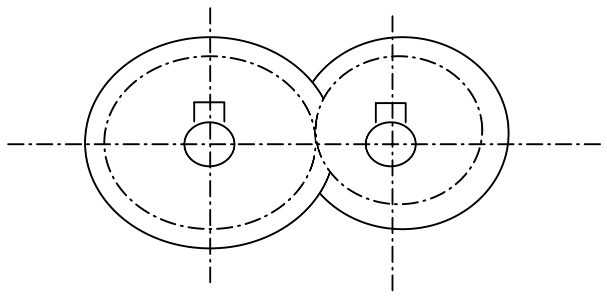
```
Artifact   gear (
            (aid gear)
            (attr shaft_dia ) // diameter
            (attr width) // length
            (attr pitch_dia) // pitch diameter
            (attr n_teeth) // number of teeth
            (attr sb_max) // max allowable bending stress
            (attr sc_max)  // maximum allowable compressive stress
            (attr st_max)  // max allowable torsional stress
            (attr E)   // material property: mod of elasticity
            (attr G) //   material property: mod of rigidity
            (attr rho) // material property: density
            ….
            (purpose transmit_torque)
            (purpose transmit_rotation)
            (purpose change_speed)
            (requires (art  shaft))
            (requires (art  key))
            (requires (art housing))
            (input  torque_in torque 1.0 kgm)
            (input   omega_in  angular_velocity 1.0 rad/sec)
            (output torque_out torque 1.0 kgm)
            (output omega_out  angular_velocity 1.0 rad/sec)
            (constraints NONE)
            (goal NONE) // no goals defined yet!
            (structure
                        (sid  spur_gear')
                        (ctrl_var (dia leng torque_in torque_out omega_in omega_out)
                        (sketch  (abst_sketch
                                                (node 1
                                                            (point)
                                                            (coord (0,0,0) // center of gear
                                                            (dof (0,0,0,1,0,0)
                                                            (input omega_in)
                                                            (output NONE)
                                                            )
                                                (node 2
                                                            (circle) // pitch circle
                                                            (center (coord (0,0,0))
                                                            (dia pitch_dia)
                                                            (input NONE)
                                                            (output torque_out)
                                                            (output omega_out)
                                                            )
                                                (node 3
                                                            (circle) // outer circle
                                                            (center (coord (0,0,0))
                                                            (dia outer_dia)
                                                            (input NONE)
```

```
                                        (output NONE)
                                        )


                                (cad_sketch (file “spur_gear_01.dwg”))
                                (constraint
                                        (C0: torque_out = torque_in)
                                        ( C1: <contact stress> LE <allowable stress>))
                                )
                        (behavior
                                        (state_var torque_in
                                                (depends_on NONE)
                                                (affects torque_out)
                                                (val_ref  INPUT))
                                        (state_var omega_in
                                                (depends_on NONE)
                                                (affects omega_out)
                                                (val_ref  INPUT))
                                        (state_var torque_out
                                                (depends_on torque_in)
                                                (affects NONE)
                                                (val_ref  procedural (EQ torque_in)))
                                        (state_var omega_out
                                                (depends_on omega_in)
                                                (affects NONE)
                                                (val_ref procedural (EQ omega_in)))
                                        (state_var angular_twist
                                                (depends_on NONE)
                                                (affects NONE)
                                                (val_ref procedural
                                                (EQ <appropriate eqn>)))
                                        (state_var torsional_Stress
                                                (depends_on NONE)
                                                (affects NONE)
                                                (val_ref procedural
                                                (EQ (<appropriate eqn>)))

        )
```

A spur gear box to change rotational speed:



```
Artifact   gear_box (
        (aid gear_box)
        (attr shaft_dia_in ) // input gear diameter
        (attr shaft_dia_out ) // output gear diameter
        (attr pitch_dia_in) // input gear pitch diameter
        (attr n_teeth_in) // number of teeth input gear
        (attr n_teeth_out) // number of teeth output gear
        (attr trans_ratio) // speed ration (output_speed/input_speed)
```

```
                      //  EQ n_teeth_in/n_teeth_out
(attr sb_max) // max allowable bending stress
(attr sc_max)  // maximum allowable compressive stress
(attr st_max)  // max allowable torsional stress
(attr E)   // material property: mod of elasticity
(attr G) //   material property: mod of rigidity
(attr rho) // material property: density
….
(purpose transmit_torque)
(purpose change_rotational_speed)
(requires
        (art gear)
        (location (coord (node 1)))
        (orientation (direction_cosine (cx,cy,cz))
)
(requires
        (art gear)
        (location (coord (node 2))
        (orientation (direction_cosine (cx,cy,cz))
)


(requires
        (art  shaft)
        (location (coord (node 1)))
        (orientation (direction_cosine (cx,cy,cz))
)
(requires
        (art  shaft)
        (location (coord (node 2)))
        (orientation (direction_cosine (cx,cy,cz))
)
(requires
        (art  key)
        (location (coord (node 4)))
        (orientation (direction_cosine (cx,cy,cz))
        )
(requires
        (art  key)
        (location (coord (node 5)))
        (orientation (direction_cosine (cx,cy,cz))
        )
(requires
        (art housing)
        (location (coord (node 1)))
        (orientation (direction_cosine (cx,cy,cz))
        )
(input  torque_in torque 1.0 kgm)
(input   omega_in  angular_velocity 1.0 rad/sec)
(output torque_out torque 1.0 kgm)
(output omega_out  angular_velocity 1.0 rad/sec)
(constraints
        (C0: omega_out = trans_ratio * omega_in)
        // pcd of two gears should meet
        // others
)
(goal NONE) // no goals defined yet!
(structure
        (sid  gear_box)
        (ctrl_var (torque_in torque_out omega_in omega_out)
        (sketch  (abst_sketch
                        (node 1
                                (point)
                                (coord (0,0,0) // center of gear in-gear
                                (dof (0,0,0,1,0,0)
                                (input omega_in)
                        (output NONE)
                        )
                (node 2
                        (point // contact point on pcd of both
                                (coord (0,0,0)))
                        (input NONE)
                        (output torque_out)
                        (output omega_out)
                        )
                (node 3
                        (point // center of out-gear
                                (coord (0,0,0)))
                        (input NONE)
                        (output omega_out)
                        (output torque_out)
                        )
                (node 4
                        (point // key hole location
                                (coord (0,0,0)))
                        (input NONE)
                        (output omega_out)
                        (output torque_out)
                        )
                (node 5
                        (point // key hole location
                                (coord (0,0,0)))
                        (input NONE)
                        (output omega_out)
                        (output torque_out)
                        )

                (cad_sketch (file "spur_gear_box_01.dwg"))
                (constraint
                (stress_ok := <contact stress > LE <allowable stress>)
                )
        (behavior // of combined gears
                        (state_var torque_in
                                (depends_on NONE)
                                (affects torque_out)
                                (val_ref  INPUT))
                        (state_var omega_in
                                (depends_on NONE)
                                (affects omega_out)
                                (val_ref  INPUT))
                        (state_var torque_out
                                (depends_on torque_in)
                                (affects NONE)
                                (val_ref  procedural (EQ torque_in)))
                        (state_var omega_out
                                (depends_on omega_in)
                                (affects NONE)
                                (val_ref procedural
                                (EQ omega_in*trans_ratio)))
                        (state_var angular_twist
                                (depends_on NONE)
                                (affects NONE)
                                (val_ref procedural
                                (EQ <appropriate eqn>)))
                        (state_var torsional_Stress
                                (depends_on NONE)
                                (affects NONE)
                                (val_ref procedural
                                (EQ (<appropriate eqn>)))
)
```