Proceedings of IDETC/CIE 2005
ASME 2005 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference
September 24-28, 2005, Long Beach, California, USA
DETC2005-85561

A KNOWLEDGE-BASED SYSTEM ENGINEERING PROCESS FOR OBTAINING ENGINEERING DESIGN SOLUTIONS

Brian Prasad and Jeff Rogers
Control Systems Division, Parker Aerospace, Parker Hannifin Corporation,
14300 Alton Parkway, Irvine, CA 92618-1898 USA
bprasad@parker.com; jrogers@parker.com

ABSTRACT

Designing and developing highly engineered products requires direct (and more dynamic) associations between customers' specifications and product characteristics (or its behaviors). In order to meet the specified customer performance, cost, and integrity goals, a multitude of specialized analyses, heuristics, shortcuts, look-up tables, equations, algorithms, finite elements, and material substitution at multiple levels (system, subsystems, components and parts) are ought to be performed. The product geometries of such engineered product are often complex and many parts are designed interactively from scratch using a 3D commercial computer-aided design (CAD) -- lately often referred as Product Life-cycle Management (PLM) system. Today, this very ***"PLM-based"*** engineered product-design process is often "static", very "feature or geometry-dependent," "knowledge-intensive," and therefore, engineers often takes considerable time (months) to complete this manual process.

Today, more and more companies want to quickly reengineer a product from a multitude of family solutions (corresponding to various design trade-off studies). They are interested in some dynamic form of a decision-based system that could automatically filter through a multitude of historical product solutions and quickly reconfigure one that meets the customer requirements with the least cost, weight, and time investment. Such decision-based product automation is not an easy task by any means. Product definitions without knowing specific geometry are hard to conceive, capture generically, and reuse widely (via any generative tool). A typical product development process —by its nature—**is highly dynamic, nonlinear, discrete, feature-dependent, and part-dependent.** The solution is not easy, since problem formulation is time-bound, has numerous discrete inputs, topologies, and several mathematical discontinuities.

This paper discusses the system architecture of the ***Knowledge-driven Automation*** (KDA) program -- established at Parker in 2002. It addresses many of the above product development issues and problems. In particular, authors describe a ***Knowledge-based System Engineering Process for Obtaining Engineering Design Solutions in a Commercial PLM Setting.*** The architecture and solutions use a number of innovative knowledge-based engineering (KBE) concepts and procedures. Through strategic use of generative modeling, spreadsheet tables, part and assembly templates, system engineering concepts, and our proprietary "smart part concepts," authors were able to ***engineer-to-configure*** a family of hydraulic actuators automatically from their customer specifications using a set of PLM (CATIA V5 and its underlying knowledgeware) tools.

1.0 INTRODUCTION

Conventional product design and development practice in PLM often concentrates solely on constructing "end-use" descriptions of the product-- often called layout or detailed designs. Most layout descriptions fall in two categories: concept descriptions and math-based descriptions. Examples of concept descriptions are a concept paper, a product schematic sketch, a blueprint or drawing, and/or a physical model (mock-up). Many of virtual or math-based descriptions, in the layout stage, are explicit representations of geometry in either "flat files" (such as CAD file/2D parts geometry) or in "digitized 2D/3D form" (such as in popular CAD entities: arcs, lines, surfaces, etc.) In the preliminary stages of layout design, some of the tangible deliverables are: functional decomposition, device models, back of the envelope calculations, detailed or fixed dimensional geometry and drawings. This is shown by the shaded blocks in Figure 1. When a layout is "locked-in" in digitized forms, it is easy to create tool-path or NC data, post processing data (machine inspection) and prototype machining (streolithography), etc. It is also easy to create downstream results, such as bill-of-materials (BOM), process plan, products manual, users manual, service plan, etc. (see Figure 1. Some of the intermediate deliverables at this stage include: design, manufacturing specifications, tooling specifications, and even physical prototypes. These deliverables are for the "locked-in" form of CAD data (static form). This is because text and point data in CAD present no problem if the "original intent" of the design is not altered. However, via fixed-dimension modes, it is very difficult to transfer or reformat the intent of an engineering design to any other form or object. Intermediate deliverables are not the end product of the design process. Using a symbolic representation or a variable dimensional (parametric) mode, one can generate these intermediate deliverables easily for a variety of new or altered situations. Parametric and knowledge based CAD representations are

examples of such techniques that are based on capturing design intent not just capturing the results of CAD documentation.

It is also generally said that 70% of the product cost is "committed" or "locked-in" after the layout design is complete. This presents three dilemmas:

- First, layout designs (locked-in a fixed-digitized form) are not flexible, so engineers are unable to incrementally modify them as they move into other iterative life-cycle analysis phases.
- Second, even if such flexibility exists, no matter how many engineers would try, the expected gain will only be a fraction of the remaining 30%. The reason for this is because 70% of the designs were already frozen or committed at this point.
- Third, even if description of the design process at this late stage is captured parametrically or through similar means (of representing design rationale) it will not be of much use since most communications in the current process are carried out in explicit forms (such as blueprint or drawings, CAD geometry, etc., as explained above and in Figure 1).

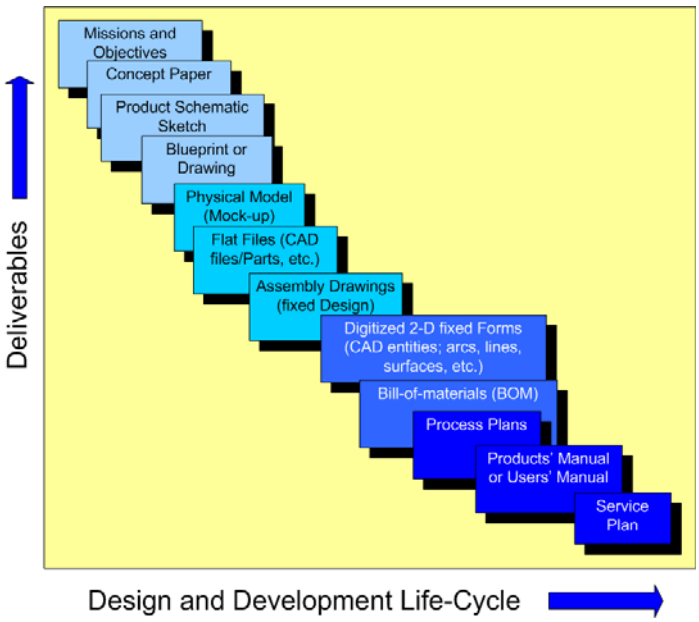

Figure 1: List of typical deliverables in a conventional design & development process

Knowledge-based product data management (PDM) and knowledge-based product lifecycle management (PLM) tools are emerging today as viable means to capture the best practices, analyses, sizing principles, and methods and communicate them across the various life-cycle functions of product development.

While designing an artifact, teams often forget that the product is a system. Products consist of subassemblies each fulfilling a different but distinct function. A subassembly is a group of two or more units that can be brought together (say for instance, shared axes or fit together). Relation information connects subassemblies, processes, design methods, and/or physical features. Subassembly information includes components, parts, and features (materials, attributes, parameters, and geometry) as well as other sub-systems. Process information includes assembly or manufacturing considerations such as joining faces, aligning bolts, common axes, welding lines, etc. Design methods include parts layout, design rules, analysis methods, packaging schemes, dimensional tolerance, material substitution, and form options. A feature is a representation of form, relevance, and intent to some aspect of part design of interest to either functionality (part features or so called form features) or manufacturability (e.g., DFX) [Nielsen, 1990]. *Form* signifies the attributes of the features present, their connectivity (topology) and geometry. *Relevancy* points the reason a feature is included in the design (e.g., issues related to functionality or DFX rules). *Intent* represents the imposed constraints by the concurrent engineering (CE) teams on the freedom of the parts' function or rules associated with a DFX principle. In order to achieve a truly **integrated product and process design (IPPD**), all the above needs have to be considered simultaneously. The core concept of IPPD is the system definition. Consider what Szucs [1980, pp. 42] stated: *The term system is used to denote an object composed of certain elements that are linked by well-defined (but not necessarily known) relationships. It is stipulated that the system may interact with its environment (receive external stimuli) and that its behavior comprises responses that are useful or profitable to the operator. Objects completely isolated from their environment (completely closed system) and phenomena that cannot be influenced by person are not regarded as systems in the technological sense.*

2.0 SYSTEM CONSTITUENTS

Before we take a close look at the requirements of a full IPPD system, let us review what constitutes a system. Truly, a system has three parts:

(a) ***Class Hierarchy (a structure definition)*:** Class hierarchy is a very efficient mechanism for system definition because one can use method and variable definitions in more than one subclass (such as sub-systems, components, etc.) without duplicating their definitions. For example, consider a system that represents various kinds of human operated vehicles (See Figure 2). This system would contain a generic class for vehicles, with subclasses for all the specialized types. Five major subclasses might include: auto, truck, bus, aircraft and land transport devices. The generic vehicle class would contain the methods and variables that were pertinent to all vehicle features, such as identification numbers, passenger loads, fuel capacity, etc. The subclasses, in turn, would contain any additional methods and variables

that were specific to the specialized cases [Taylor, 1993]. Truck may consist of pick-up, van, and a tractor-trailer. Land transport devices may include subclasses such as motorcycles, bicycles, skateboards and unicycles.

{Motorcycles }

{Bicycles}

{Train} ∈ [Land transport devices]

****

{Tram}

Where ∈ denotes an element of a set or member of the class hierarchy.

(b) ***Integration Hierarchy (an assembly definition)****:*

[Land transport devices] ⊃ {Motocycles, Bicycles, Train,…, Tram}
Where ⊃ denotes "is a member of super class set." The elements in the curly bracket are its member classes.

(c) ***Constancy-of-specifications, requirements, purpose or goals****:* This identifies the key product functions, requirements or constraints (RCs) of the product system, some of them are also common to its elements (product sub-classes). For example, general system-level RCs must gives rise to component (e.g., a department unit) level RCs. Constancy means carrying the same definition and the same sets of requirements hierarchy and management process throughout a product development process. Integrated product definition (IPD) teams must agree on a common set of engineering requirements that relate to the company's (high-level product development) goals and customer needs. Constancy of RCs also provides a common set of ground rules for mutual cooperation and communication throughout the product development community. If a product development process is "left to themselves in the Western world, components become selfish, competitive, independent profit centers [Deming, 1993]."

$RCs_{parts} \exists RCs_{components} \exists RCs_{subsystems} \exists RCs_{system} \exists RCs_{department} \exists RCs_{SBUs} \exists RCs_{enterprise}$

where, ∃ denotes an element of.

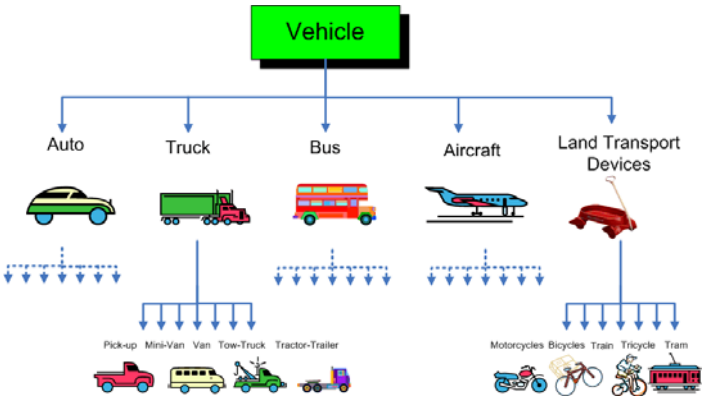

Figure 2: Class of human operated vehicles

Having a definition of what constitutes a system is the first step toward being able to manage an IPPD system. However, a product-breakdown structure or its definition alone is not enough. The design of a typical automobile, aircraft, helicopter or computer involves thousands of engineers making millions of design decisions over several years. Class hierarchy was one of the key elements of organizing a complex product as described by Prasad [Chapter 8, pp. 380-387]. Decomposing the product realization process into class hierarchy of discrete activities is another essential element of IPPD.

3.0 SYSTEM LEVEL ORGANIZATION OF INTEGRATED PRODUCT & PROCESS DEVELOPMENT

An important distinction in applying knowledge-based engineering to IPPD is to recognize that the artifact being designed is an assembly or a system (i.e., something which is composed of a hierarchy of sub-systems, components, parts, and templates). Templates form the lowest level of inputs for reconstructing the system. A number of reusable templates could define a generic part. A system level organization of product development activities is shown in Figure 3.

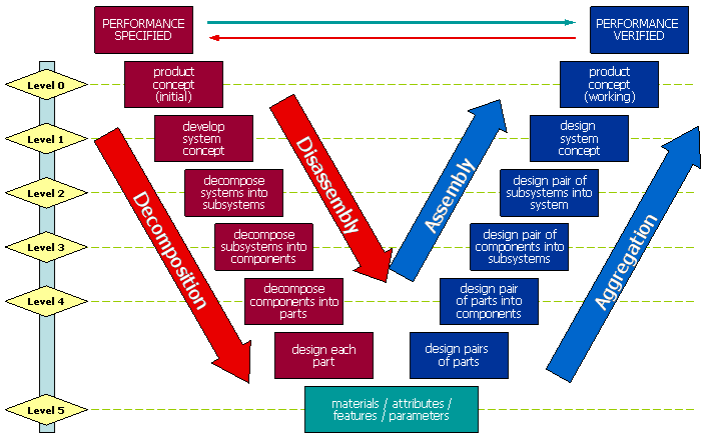

Figure 3: A system level organization of integrated product and process development activity

Organization of an IPPD process plays an important role in ensuring the effectiveness of the resulting production system. Organization of IPPD can be approached by carefully splitting the system-level problem into its mutually separable transformation states, followed by modeling of each state, then the reconstruction of a system definition from the aggregate definitions of its constituent states [see Prasad, 1977].

As shown in Figure 3, functional description of a product realization process thru IPPD consists of three major components:

- ***System Decompositions:*** During system decompositions, systems are broken into subsystems, subsystems into components, components into parts. Finally the parts are expressed in terms of base templates. The templates, in turn, consist of both geometric and non-geometric set of features and parameters. There arc many ways a product can be decomposed. In the Figure above, 5 levels of decomposition are shown. Level 0 is finished product solution. Level 5 represents specifications, BOPs, BOMs, etc. The number of levels depends upon the complexity of the problems and the desired aggregation schema used.

- ***Part Solutions: One key aspect of product realization is to recognize that a "part" can be made out of multiple templates.*** The solution for each template-based construct is sought concurrently having shared specifications being part of a global set of specifications stored at the system level. During solution iterations attempts are made to satisfy the various part-template goals. Normally, during a single pass of an "all" parts solution, only a portion of each part's goals are satisfied. The part's goals which are not met are passed onto the system's level goals. The design then enters into a system reconstruction loop.

- ***System Reconstruction:*** During system reconstruction, parts are assembled, inter-part constraints are applied, and its system performance is measured. Value engineering and continuous process improvement are considered part of this validation. If no improvement is desired, the artifact system is converged; otherwise, necessary improvements are incorporated through a secondary redesign bypass loop.

4.0 PRODUCT SYSTEM DECOMPOSITION

Product decomposition means viewing the product realization process as a part of the whole and then overlapping (aggregating) the decomposed sets to recreate or reconstruct the whole system from its constituent parts. In other words:

Product Realization $\Leftrightarrow$ [Decomposing (parts-from-the-whole) $\oplus$ Reconstructing (whole-from-the-parts)]

The term "whole" also includes multiple characteristics of life-cycle concerns (e.g., X-ability) that we may need to consider. Many products can be decomposed safely into sets or class hierarchy. However, not all such decomposed sets of life-cycle activities in a product exhibits independence.

Smith and Browne [1993] describe decomposition as a fundamental approach to handling complexity in engineering design. The two (decomposition + concurrency) allows one to identify activities that can be overlapped or performed simultaneously. It also allows one to formulate strategies leading to their separation, e.g., indexing, alternate decomposition, teaming, or restructuring. In good product decomposition, the decomposed class hierarchy is described in such a way that their interfaces are eliminated or minimized.

Attributes that are essential for successful product realization outcomes are:

- ***Minimize Interfaces***: This entails reducing all sorts of interfaces required for both decomposition and reconstruction parts of the "Product Realization Process" to a minimum.

  Number-of $P_{interfaces}$ $\Rightarrow$ Minimum

  These include the interface relationship between management and design, supplier interface, design development interface, design to assembly interface, design to manufacturing interface, production interface, etc. Such interfaces can be very long indeed and tend to depend upon the size of the company and the product and process complexity.

  Do all the decomposed parts exhibit independent or semi-independent characteristics? Due to increased global pressure to bring a product early into the marketplace,

parallel processing in CE is becoming a necessity. There are, however, many ways a product, process or work information can be decomposed and overlaid in parallel. If a product, process or work information activity does not affect other parameters or processes, it can be performed locally. If it does, it can be performed in a distributed fashion by the IPPD team. Local or distributed processing, to a large extent, depends on how a product structure is originally broken up or decomposed. Wherever there is a dependency, a dependent portion (for instance a part or a process) should not be designed in isolation from the rest of the product or process. This is derived from the proposition that if a component is sub-optimized in isolation from the rest, the whole may not be fully optimized. Good decomposition allows the scheduling of activities to proceed in parallel. For example, it is not necessary to delay the start of an activity if the information required for a decomposed activity is not dependent on the rest.

- ***Quick Processing***: means performing individual activities as fast as possible using knowledge-based productivity tools, lean process reengineering tools, design aids or knowledge-based engineering techniques. It also amounts to speeding up the preparation time in building up the information content before and after an execution of a task. This emphasizes the mandate for shortening the pre- and post-processing time and the time it takes for completing the decomposed tasks themselves.

  Quick Processing $\Leftrightarrow$ Minimize Lead-time-of $P_{task\text{-}i}$ for I=1, n

  Where, n is the number of tasks in the P-set.

- ***Virtual Communication***: This provides transparent communication among the individual activities that are partitioned (decomposed).

  Tasks $P_{communication} \Rightarrow$ Transparent

There is a difference between the complexity of the philosophies (such as product complexity, process complexity, enterprise complexity, and the philosophies of their virtual communications. An organization committed to making such complex products in the shortest possible time need pay equal attentions to providing virtual communication among the individual activities that are decomposed. This amounts to using inter-part and intra-part communication techniques that are method-dependent but part-independent.

Through a knowledge-based approach, engineers optimize the artifact simultaneously with the aid of knowledge and experience acquired from the experts. Engineers acquire the product and process knowledge and then use the power of intelligent multi-template systems to guide decision making. [Prasad, 1977]. In the decomposition schema introduced in this paper, a two level decomposition (Level 1 and Level 2) is used. The Actuator System Solution represents Level 1. SmartPart definition, described next, represents Level-2 decomposition.

5.0 DEFINITION OF A SMART PART

When a team develops a part, it specifies three types of information: inputs to a given baseline system, certain behaviors required from the existing system, and changes or restrictions on the future behaviors of the system. In this definition, they are characterized as inputs, requirements, and constraints, respectively.

***Inputs*:** Inputs are everything needed to carry out an IPPD process or a function.

Inputs $\{ I_i \}$ include data $\{ D_i \}$, knowledge $\{ K_i \}$, and process $\{ P_i \}$ at state i.

Inputs = Function-of (Data, Knowledge and Process)

or $$I_i \equiv f \left[\{ D_i \}, \{ K_i \}, \{ P_i \} \right]$$

Inputs at an intermediate state assume the values of the previous state inputs during transformation. Initial data is usually derived from environmental conditions, customer requirements, or voice of the customer. Data may also be present as capital investments, people, money, materials or equipment. Another element of input is the process P_i at state i. The process includes teams, skill, and leadership, as well as the state of domain, infrastructure, tools, organization, information, communications, technology, and other business processes. K_i is the product or process knowledge including scientific, technological, empirical, statutory, social, cultural, or environmental aspects.

***Requirements*:** These are defined as "necessary" statements of the characteristics of a domain object that are required (R_i) to describe the domain, its features, or its multiple behaviors. Requirements may take the form of industry standards, materials characteristics, aesthetic considerations, human factors, safety, ergonomics, regulatory, ecological or environmental conditions among others.

***Constraints*:** Constraints are the restrictions on the inputs and the requirements. Constraints C_i, at state i, restrict the behaviors of the domain object to function in a particular or specific way. Collectively, the constraints define what will be an acceptable outcome. The restrictions range from social, political, economic, technological, ecological, legal to random behaviors. The simplest form of constraint would be like assigning a value to a form or function feature. In the context of engineering design, a constraint can be thought of as a required relationship among design attributes and features.

The above framework items are collectively called herein a system specification set $\{ SS_i \}$:

$$\{ SS_i \} \equiv \cup \left[\{I_i \}, \{ R_i \}, \{ C_i \} \right]$$

To understand “IPPD process” definitions, it is useful to start with a definition of a “SmartPart” model. A SmartPart definition is shown in Figure 4 using a three-template schema: specifications, sizing, and geometry. [Prasad, 1977].

- ***Specification Template:*** The specification template captures both inputs and requirements for design. All attributes related to specification inputs, like materials data, fatigue life computation knowledge and specific processes for a particular part family are captured via knowledge rules and instructions. It contains no geometry.

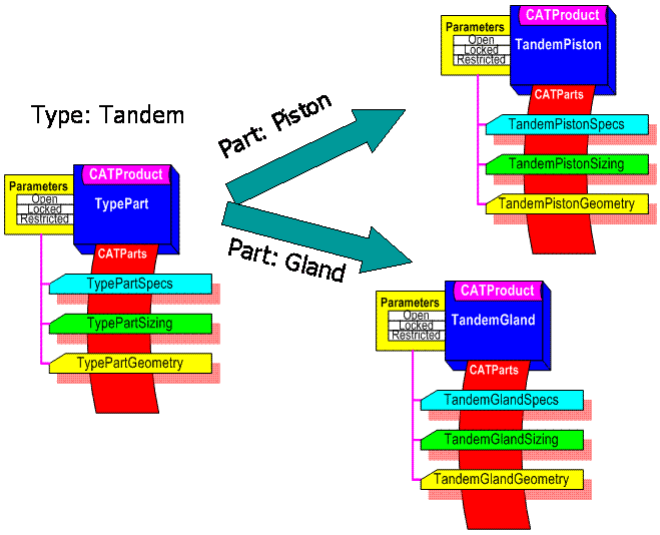

Figure 4: Definition of a SmartPart and how it is reused to create intelligent part families

- ***Sizing Template***: The sizing template capture the analytical basis of design and thus provide the constraints for the problem definitions. It serves as a modifier for the initial system specifications block. All inputs and knowledge related to sizing of the part and a decision-making process for making a trade-off against the captured constraints are modeled using knowledge rules and instructions. It also contains no geometry.

- ***Geometry Template***: The geometry template is a feature-based representation of the design intent. It takes the inputs from both specification and sizing templates and forms geometric definitions from those abstract specifications. This is commonly achieved using a CAD/CAM or PLM environment.

Such a three-template schema is interrelated, by design. It is effectively used to capture and define the artifact’s intent according to the perceived customer needs. Using this approach, the SmartPart model broadly defines the constraint boundaries, overall requirements of the product design, problem resolution definition, and life-cycle intent.

One of the most important steps in IPPD is the creation of a detailed taxonomy of the “product solution process” [Prasad, 1977]. Taxonomy allows concurrent realization of an artifact in an orderly fashion. The three-template schema (altogether called herein as SmartPart) serves as the set of reconfigurable building blocks for this taxonomy. Using the three-component construct, authors were able to transform the system specifications (inputs, requirements, and constraints) into a solution output (in the form of a physical artifact). The sizing template (a component of the SmartPart) serves a very useful purpose since it embodies a basis of comparison of prediction with measurements.

First, a SmartPart template captures the system’s behavior in some form of abstraction or formalism. Second, to use this formalism, it transforms some crude description of the parts’ desired behavior (called system specifications) into a customer requirements and finally into a physical description. Let us assume that the IPPD System problem at hand consists of “M” SmartParts. Figure 4 shows a mapping of the functional description of the product and all of its decomposed components (SmartParts) through a series of transformations leading to a physical description. The series exhibits a precedence relationship among its transformed states. Each state is governed by a set of SmartParts, which may be executed in any random order. Each smart part is driven by a set of specifications (via a specification template) and a set of sizing constraints (via a sizing template), which in turn produces a set of consistent geometry outputs (via a geometry template). Let us also assume that convergence of the product solution is achieved by repeating this transformation process “n” time.

Let us assume further that
$[<P1_0 >, <P2_0>, <P3_0>, \ldots, , <PX_k>, \ldots, <PM_0>]$
represents an activity plan for a set of SmartParts (X=1, M) at the initial state of transformation (k=0).

Thus, at an intermediate (say ith iteration point),
$[<P1_i >, <P2_i>, <P3_i>, \ldots, , <PX_i>, \ldots, <PM_i>]$
represents an activity plan for a set of SmartParts from X=1 to X=M.

And $[<P1_n >, <P2_n>, <P3_n>, \ldots, , <PX_n>, \ldots, <PM_n>]$
represents an activity plan for a set of SmartParts at the conclusion of nth state of transformation.

Let us also assume,

$[\{ S1_i \}, \{S2_i \}, \{S3_i \}, \ldots, \{ SX_i\}, \ldots, \{ SM_i \}]$ denotes a set of SmartPart specifications from X=1 thru X=M at a transformation state i.

The relations between the system Specifications { SS_i's },and SmartPart functional specifications
Namely, [{ $S1_i$ }, {$S2_i$ }, {$S3_i$ }, …, , { SX_i},..., { SM_i }]
is expressed as

{ SS_i's } == U [{ $S1_i$ }, {$S2_i$ }, {$S3_i$ }, …, , { SX_i},...,
{ SM_i }]

Where U indicates Union of individual specifications. Each of these templates is discussed next:

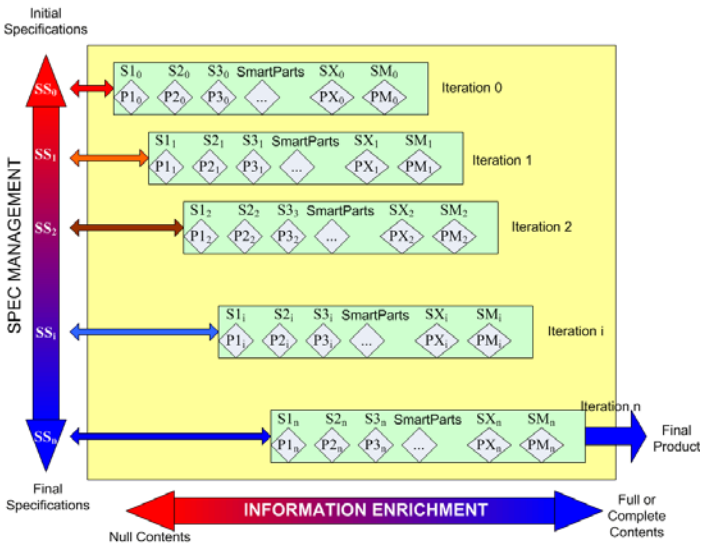

Figure 5: A transformation strategy for realizing a product system solutions from its decomposed SmartParts

5.1 Parts Specification Templates

A part's specification template provides a mechanism to pass the technical specifications from system-level tree to parts-level tree. Technical specifications help identify both the product's functional intent as well as its parts' characteristics. Technical specifications are the equivalent of the customer requirements that are stated in meaningful and quantifiable terms. "Meaningful" implies they are understandable to engineering communities; "quantifiable" indicates that they can be measured in physical parameters like force, distance, torque, acceleration, rates of change, and so forth. The customer requirement for an engine that accelerates fast might be translated into a technical requirement of "time it takes to reach a speed of 100 miles per hour under specific conditions." The customer requirement for a door that opens easily can be translated into technical requirements for "amount of force required for a given push or rotation depending upon the type of handles used." Technical requirements can also be established directly using the technical expertise of experienced engineers. It can also be taken from product acceptance standards if the work-group can intelligently reflect the customer needs or expectations.

This specification template converts a set of system requirements, SSs (such as those to fulfill the customer requirements (CtRs) into a set of part requirements (PtRs). Then development of the physical embodiment (enrichment of data, process and knowledge) takes place first in the part specification template and then in "sizing" and "geometry" templates.

$$[<PX_i>; X=1, M]_{specs} \ \{<CtRs_i>\} \rightarrow \{ PtRs_i \}$$

5.2 Parts Sizing Template

A sizing template is made out of two types of Part's specifications: behavioral and physical.

- ***Behavioral:*** Those specifications that describe the desired behavior of the overall system at an abstract level are called behavioral specifications. Examples of behavioral specifications include global behaviors, such as limits on overall frequency, stiffness, etc. Note that strength is not a system behavior because it is generally governed by "local characteristics" such as presence of notch, crack, etc.
- ***Physical***: Physical specifications place physical restrictions on the outputs, such as geometry, allowable sizes and weights, amount of scraps, etc.

Representations of the behavioral inputs and physical requirements must be linked to representations of their physical characteristics, which dictate the constraints. Constraints at different levels of abstraction guide the design process. RCs are also used to maintain consistency and propagate design decisions. The relationship may exist in an explicit form, like an analytical form (such as design equations.) A physical term, the weight of a helical coil spring, for example, is analytically expressed as the product of stiffness and the square of the allowable deflection, a behavioral term. Other forms in which this relationship exists are database entries, like a spreadsheet, or an implicit form, such as a finite-element method (FEM) or CAD geometry which relate geometric characteristics (such as weight) to behavior characteristics (such as stiffness). CATIA V5 provides a number of techniques for capturing that knowledge into equations and rules.

Here, a series of mapping for smart part sizing occurs transforming the part-requirements (PtRs) as described by Prasad [1977] into part-constraints, (PtCs):

$$[<PX_i>; X=1, M]_{sizing} \ \{<PtRs_i>\} \rightarrow \{ PtCs_i \}$$

5.3 Parts Geometric Template

Most of the activities in the product development cycle of mechanical products are centered on generating and designing geometric shapes that perform some specific functions. Geometrical requirements may include generation of geometry, topology, dimensions, and process tolerances.

There are two types of functional requirements (FRs) that are usually needed to model the product geometry adequately: quantitative FRs and intent FRs. A complete design of a product requires not only the FRs but also the geometric configurations that realize the FRs. This views an assembly as a hierarchy of standard class structures and uses a strategy of known classes to devise a plan for the whole assembly. This identifies a class hierarchy within each concept representation (such as behavior or physical models). For example, instead of viewing an assembly as a set of parts, one can view it as composed of a set of class structures. Structure description could be a class representing standardized relationships among a set of parts. These descriptions typically form a class, which is made out of other classes. For example, a door structure is composed of a latch structure and two door hinge structures. A door hinge structure is composed of a hinge pin structure and two multiple screw plate structures. The multiple screw plate would be composed of an odd number of simple screws and a solid circular pin. The simple screw would be composed of a variety of contact (seating and positioning) structures.

A series of mappings for part geometric template transforms the part constraints (PtCs) into the physical geometry domain (or the artifact itself).

$$[<PX_i>; X=1, M]_{\text{geometric}} \{ <PtCs_i> \} \rightarrow \{ PtGs_i \}$$

Where, $\{PtRs_i\}$, $\{PtCs_i\}$ and $\{PtGs_i\}$ represent the vector of part requirements, part constraints and geometry requirements, respectively.

The mapping process from the system to SmartPart, from SmartParts to SmartParts and finally from system specs to the physical artifact is not unique. There can be an infinite number of plausible solution paths.

6.0 A PRODUCT DEVELOPMENT TAXONOMY FOR SATISFYING REQUIREMENTS AND CONSTRAINTS

A taxonomy is the theory, principle, or process of classifying organisms in established categories (The American Heritage Dictionary, 1981). Long term success depends upon the need to create a product development taxonomy (a transformational strategy for product realization) in which product specifications (frames) can be transformed into a description of a series of sub-specifications (sub-frames) for the lower level transformations [Finger and Rinderle, 1989].

The product realization process can thus be viewed as a definition of loosely connected transformations converting the original specifications into an useful product or service (see Figure 5). Each transformation level (shown in Fig. 5) represents a subset of the product realization space at different levels of abstraction and/or granularity. The key to gaining an insight into any system is developing a functional model and defining a set of input specifications and output characteristics as a part of this system model. Comparisons to inputs (specifications) and outputs (characteristics) are, in fact, the constraints for the broad product development system model (see Figure 3). Output constraints are related to input specifications—a specification is broken up into several requirements. Specifications include both the loads and sizing parameters. Loads do not change but sizing parameters change if the model needs to be altered to satisfy the computed constraints. At the beginning of the transformation (stage 0), the design often exists in pure specification form (see Figure 5).

6.1 Beginning of Iteration (Stage 0):

At the beginning of iteration, at start

$\{ SS_0 \} \rightarrow \{ \varnothing \}$ is at Initial set of specifications (incomplete state), has an empty content.

And the set of SmartParts,

$[<P1_0>, <P2_0>, <P3_0>, \ldots <PX_0>, <PM_0>] \rightarrow [\varnothing]$ are at null state or has an empty content

During each stage, as described earlier, a three-step intra-part process is used for satisfactions of specifications and constraints. This is shown in Figure 6. Let us assume we have "M" SmartParts.

- **S2P Inter-Part Exchange:** First, specifications are passed from system-level master Specs (say SS) buckets to SmartParts specs (say SX) buckets.

  $\{ SS_0 \} \Rightarrow \{<S1_0>, <S2_0>, <S3_0>, \ldots <SX_0>, <SM_0> \}$ where X=1, M.

- **P2P Intra-Part Exchange:** SmartPart solutions are obtained for this modified specification set:

  $\{<S1_0>, <S2_0>, <S3_0>, \ldots <SX_0>, \ldots <SM_0> \}$

  Specifications are passed from specification templates to sizing templates. This may provide new values for the computed constraints requiring changing previous values of the sizing parameters meaning previous assumed values. This modified list of parameters shows up in the sizing templates. Meaning, from the definition of SmartParts, $[<P1_0>, <P2_0>, <P3_0>, \ldots <PX_0>]$, the old specification set is modified as follows:

  $<P1_0> <S1_0> \rightarrow <S1_1>$

  $<P2_0> <S2_0> \rightarrow <S2_1>$

  $<P3_0> <S3_0> \rightarrow <S3_1>$

  * * * * * * *

  $<PM_0> <SM_0> \rightarrow <SM_1>$

  Geometry templates are the by-product of this knowledge-based engineering process. Once the

specifications are set (based on the customer's inputs) and appropriate constraints are applied (to provide feasible designs) the display of the conforming design is carried out by the geometry templates. Geometry primitives of a CAD system are employed to display the computed design in 3D form. Since the information flows from specifications templates to sizing templates to finally geometry templates, we call this process a **top-down** process.

- **P2S Inter-Part Exchange:** The set of parameters, which are modified are passed back from sizing templates to system level master specification.

  $\{<S1_1>, <S2_1>, <S3_1>, \ldots <SX_0>, \ldots <SM_1>\} \rightarrow \{SS_1\}$, where X= 1, M.

The triad-tree structure (shown in Figure 6), specifications and goals—together provide a basis for system optimization.

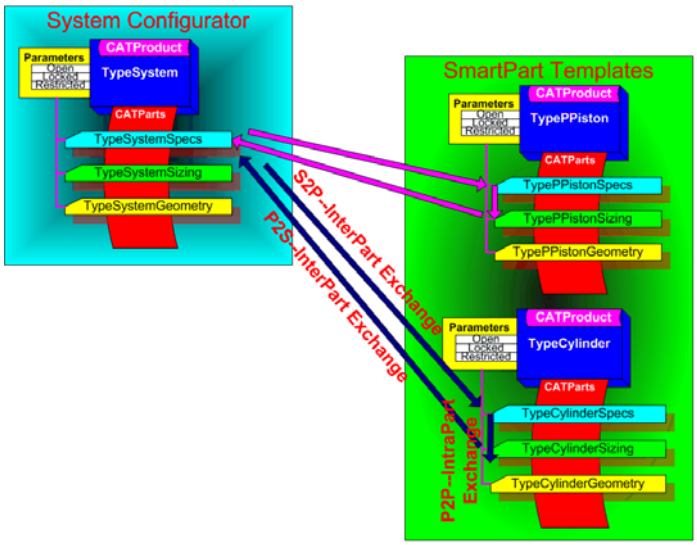

Figure 6: Inter-part and intra-part communications among white board for systems specs and SmartParts

As various tasks within this IPPD are performed the set of specifications changes to some realistic intended values. The corresponding designs (meaning product and/or process designs) begin to take shape.

6.2 Intermediate Stage (say ith Stage):

At the intermediate stage (say ith stage), the SmartParts modules, $[<P1_i>, <P2_i>, <P3_i>, \ldots <PX_i>, \ldots <PM_i>]$ transforms a set of specifications at state i from

$<P1_i> <S1_i>$ to $\rightarrow <S1_{i+1}>$;

$<P2_i> <S2_i> \rightarrow <S2_{i+1}>$;

$<P31_i> <S3_i> \rightarrow <S3_{i+1}>$ and finally

$<PM_i> <SM_i> \rightarrow <SM_{i+1}>$ respectively.

And the modified output of the previous state from sizing becomes the specification data for the new state.

that is,

$\{<S1_i>, <S2_i>, <S3_i>, \ldots <SX_i>, \ldots <SM_i>\} \rightarrow \{S_{i+1}\}$

6.3 At the end (say nth stage)

When iteration reaches the last (nth stage), hopefully, all specifications have been implemented. At that point all SmartParts designs reach at their "full content"

Stage n: $[<P1_0>, <P2_0>, <P3_0>, <PX_i>, \ldots <PM_n>] \rightarrow [\bullet]$ a Full Content

And $[<S1_n>, <S2_n>, <S3_n>, <SX_i>, \ldots <SM_n>] \rightarrow$ forms

$[\bullet]$ a Complete Set

The design can be thought of as having reached a "full or complete content" when the sets of specifications stabilize and when all the constraints have been satisfied. The artifact contains all the information (enterprise, requirements, product, process, and cognitive) needed to function as a unit as initially desired. The aim of a good transformation strategy is to uncouple the system so that each transformation state affects only one set of outputs. This is very similar to Suh's First Axiom [Suh, 1988] in the axiomatic design theory.

7.0 IMPLEMENTATION INTO A COMMERCIAL PLM TOOL

Parker Hannifin's Control Systems Division uses CATIA V5 PLM system for designing and developing all of its products. We, therefore, wanted to build a knowledge-based system engineering process inside of CATIA, because in doing so, all parameter definitions, naming conventions, captured rules, best practices knowledge and Microsoft Excel table links would be maintained by CATIA. We would not be required to maintain any external links or external knowledgebase interfaces since we didn't built any outside. We also wanted the system to be quite general purpose and the chosen system architecture to be highly generic and reusable across different product lines that Parker manufactures. In order to facilitate these requirements, we built two knowledge-based configurators

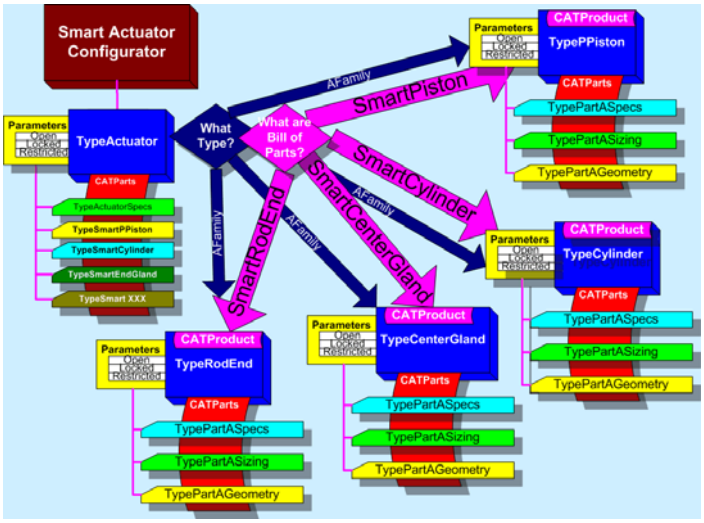

Figure 7: How Product Solution is achieved from its decomposed SmartParts during reconstruction loop

- A knowledge-based configurator for building the SmartPart from the rules and their corresponding part-templates (both captured in CATIA V5) (see Figure 7.)
- Another knowledge-based configurator for building the product solution. ProductConfigurator assembles a product solution from the set of decomposed constituents (called SmartParts). (See Figure 7.)

The product development taxonomy described in section 4 is implemented via the rule-bodies in CATIA V5 Knowledgeware tools. Three design tables are required to initiate a new design (a) a table to select bill-of-parts and appropriate materials, (b) a table to identify inputs, choose design loads and basic configuration, and (c) a table to specify the mating and assembly requirements for the SmartParts so configured. They are shown in Figure 8.

	A	B	C	D	E	F	G	H
1	Part_Name	Material_Type	Tempe	RequiredFati	RR	KT	KtCurv	KbrG
2	Cylinder	7075-T73 ALUMINUM ALLOY,BAR (T=2-3 in)	160	600000.001	0	2	5c4	4a0
3	PPiston	15-5 PH CRES, COND H1025,BAR & FORGING	160	50000.001	-1	2		
4	CEEndGland	ALUMINUM NICKEL BRONZE (C63000),BAR & SH	160	600000.001	0	3		
5	REEndGland	ALUMINUM NICKEL BRONZE (C63000),BAR & SH	160	600000.001	0	3		
6	EGLockNut	7075-T73 ALUMINUM ALLOY,BAR (T=2-3 in)	160	600000.001	0	3		
7	PRodEndAssy	15-5 PH CRES, COND H1025,BAR & FORGING	160	50000.001	0	3	5c4	4a0
8	Sleeve	4130 STEEL, 150 KSI HT,ALL WROUGHT FORMS	160	600000.001	0	2		
9	TailStock	7075-T73 ALUMINUM ALLOY,BAR (T=2-3 in)	160	600000.001	0.25	3		
10	REBearing	17-4 PH CRES, COND H1025,BAR (T LT 8 in)	160	50000.001	0.25	2.5		
11	RE2LVDTProbePin	15-5 PH CRES, COND H1025,BAR & FORGING	160	50000.001	0.25	2.5		
12	LVDTProbeLockNu	7075-T73 ALUMINUM ALLOY,BAR (T=2-3 in)	160	50000.001	0.25	2.5		
13	LVDTProbe	15-5 PH CRES, COND H1025,BAR & FORGING	160	50000.001	0.25	2.5		
14	LVDTBody	15-5 PH CRES, COND H1025,BAR & FORGING	160	50000.001	0.25	2.5		
15	LVDTGuideRing	15-5 PH CRES, COND H1025,BAR & FORGING	160	50000.001	0.25	2.5		
16	CEBearing	17-4 PH CRES, COND H1025,BAR (T LT 8 in)	160	50000.001	0.25	2.5		
17	ActuatorSystem							
18	Empty							
19	OutputLinkAssy	15-5 PH CRES, COND H1025,BAR & FORGING	160	50000.001	0.25	3		

Figure 8a: Bill-of-Parts and Materials Table.

4	ActuatorSystem_Type	AFamily	AFamily	AFamily
5	ActuatorConfig_Type	Simplex	Tandem	Tandem
6	ActuationBalanced	false	false	false
7	BalanceTubeExists	true	false	false
8	SleeveExists	false	false	false
9	CylEnd1_ConnectionType	Bearing	Bearing	Bearing
10	Nom_Extend_Sizing_Load (lbf)	104400.001	120000.001	51334.001
11	Nom_Supply_Pressure (psi)	4100.001	4100.001	4057.001
12	Nom_PTank_Pressure (psi)	175.001	150.001	600.001
13	Nom_STank_Pressure (psi)	100.001	150.001	600.001
14	Limit_Load_Compression (lbf)	120060.001	180060.001	61600.001
15	Limit_Load_Tension (lbf)	112140.001	112140.001	61600.001
16	Ultimate_Load_Compression (l	180090.001	180090.001	92401.001
17	Ultimate_Load_Tension (lbf)	168210.001	168210.001	92401.001
18	Ultimate_Load_Side (lbf)	800	800	400
19	Endurance_Load_Fatigue (lbf)	104400.001	104400.001	51334.001
20	Proof_Supply_Pressure (psi)	6000.001	6000.001	6500.001
21	Burst_Supply_Pressure (psi)	10000.001	10000.001	10820.001
22	Impulse_Supply_Pressure (psi)	6000.001	6000.001	6100.001
23	Proof_Tank_Pressure (psi)	4000.001	4000.001	1275.001
24	Burst_Tank_Pressure (psi)	6000.001	6000.001	2125.001
25	Impulse_Tank_Pressure (psi)	2000.001	2000.001	700.001
26	Impulse_Load_Cycles_Fatigue	1000.001	1000.001	5000.001
27	Stroke_Nominal (in)	9.487	9.487	8.592
28	Retract_Length (in)	24	34.957	33.394
29	Bearing_Friction_Coeff_Proof	0.15	0.15	0.15
30	Bearing_Friction_Coeff_Burst	0.2	0.2	0.2
31	Bearing_Friction_Coeff_Fatigu	0.1	0.1	0.1

Figure 8b: Table to Specify Inputs, Select Design Loads and Basic Configuration Data.

	B	C	D	E	F	G	H	I	J
1	Constraint Name	Type	Value (in	Constraint Orient	First Product	First Publication	Second Pro	Second Publication	Compute
2	CY2PP_Axial_Coincidence	Coincidence		CatCstOrientUndefine	Cylinder	Cylinder_AxisLine	PPiston	PPiston_AxisLine	Y
3	CY2REEG_Axial_Coincidence	Coincidence		CatCstOrientUndefine	Cylinder	Cylinder_AxisLine	REEndGland	REEG_AxisLine	Y
4	CY2REEG_Transverse_Parallel	Parallel		CatCstOrientSame	Cylinder	Cylinder_TransverseLine	REEndGland	REEG_TransverseLine	Y
5	CY2REEG_Contact	Coincidence		CatCstOrientOpposite	Cylinder	CylinderNutBoreThread_REEndGland_Conta	REEndGland	REEGMiddleRing_Cylinder_ContactPla	Y
6	CY2EGLockNut_Axial_Coincidence	Coincidence		CatCstOrientUndefine	Cylinder	Cylinder_AxisLine	EGLockNut	EGLockNut_AxisLine	Y
7	CY2EGLockNut_Transverse_Parallel	Parallel		CatCstOrientSame	Cylinder	Cylinder_TransverseLine	EGLockNut	EGLockNut_TransverseLine	Y
8	CY2EGLockNut_Contact	Contact		N/A	Cylinder	Cylinder_EGLockNut_ContactFace	EGLockNut	EGLockNut_Cylinder_ContactFace	N
9	CY2CEEG_Axial_Coincidence	Coincidence		CatCstOrientUndefine	Cylinder	Cylinder_AxisLine	CEEndGland	CEEG_AxisLine	Y
10	CY2CEEG_Transverse_Parallel	Parallel		CatCstOrientSame	Cylinder	Cylinder_TransverseLine	CEEndGland	CEEG_TransverseLine	Y

Figure 8c: Table to Specify Mating and Assembly Constraints Data.

8. RESULTS AND DISCUSSIONS

Parker designs a variety of products which control various moving surfaces on airplanes like ailerons, tails, flaps and rudders. In order to demonstrate “how the knowledge-based system functions” we chose two families of actuator solutions: simplex and tandem. The system specs data commonly provided by airplane manufacturers generally provides a majority of data for the design of these actuators. A list of engineering specs data for tandem and simplex are shown in two columns of an Excel table in Figure 8. The bill-of-parts and constraints are unique for simplex and tandem solutions since the name of decomposed parts and mating constraints are different. Thus we provided a different set of inputs for each.

The knowledge-based system engineering process architecture described in Sections 3-4 were employed. The ProductConfigurator ran through the SmartPart iteration cycles and provided us with a feasible solution in the end. At that point, we obtained a complete 3D solution in CATIA V5 for both the cases. The 3-D isometric views of the simplex

and tandem solutions and their cross-sections details are shown in Figure 9a and 9b, respectively (as examples).

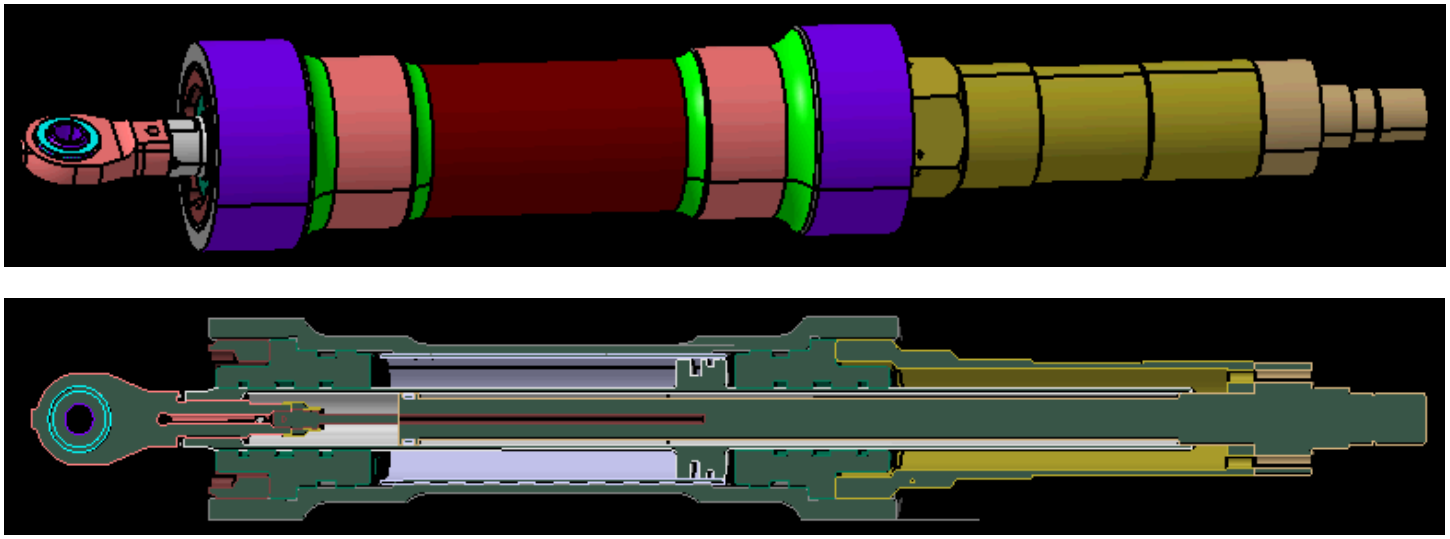

Figure 9a: Balanced simplex actuator with 3050 psi supply pressure and 3.89 inch stroke.

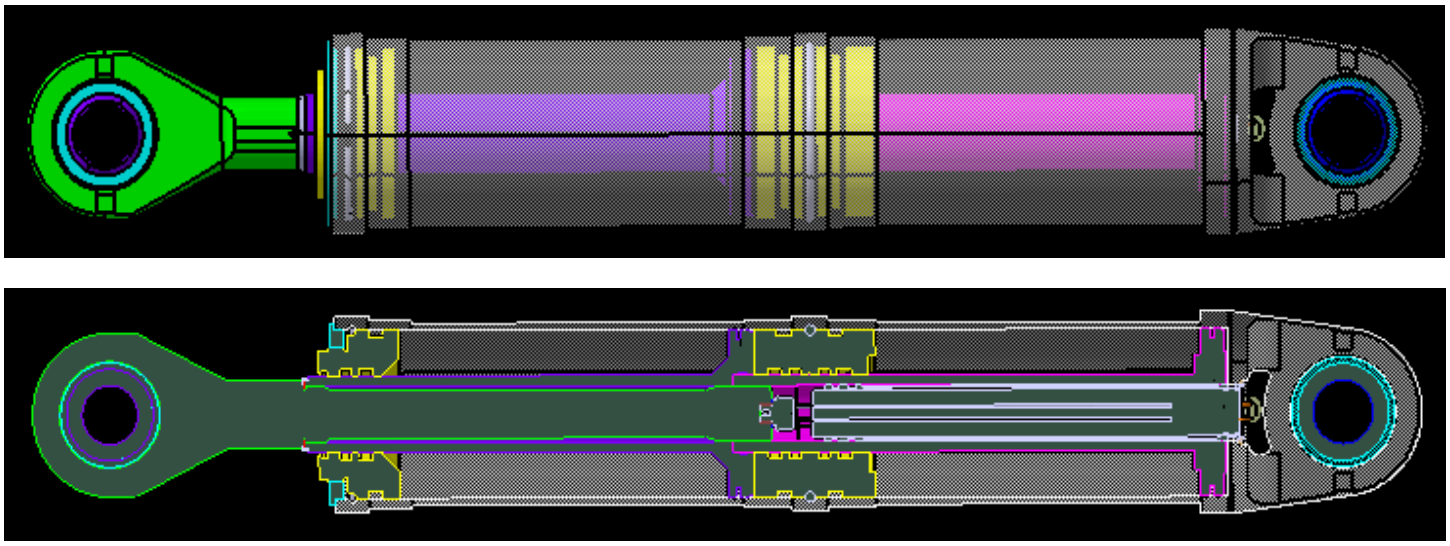

Figure 9b: Unbalanced tandem actuator with 4100 psi supply pressure and 9.49 inch stroke.

After checking through the calculations, we found that the final designs meet all load requirements, length constraints, and mating dimensions (including appropriate placements of O-rings based on computed ODs (outer diameters) for the moving parts).

9. CONCLUDING REMARKS

A knowledge-based system-engineering process has been developed at Parker to realize new product families from only raw specifications and reusable smart templates. The pertinent characteristics of this development were realized by **decomposing** the product in question into two levels: a system level and a series of semi-independent SmartParts levels, so that the necessary intra-part communication of common parameters (requirements and constraints) is performed locally during each SmartPart solution. The number of common (interface) parameters were small since the dependency of the decomposed parts in the product breakdown structure, and the dependency while further decomposing each part into a set of three SmartPart templates were minimized. By **organizing** specifications into a single level SmartPart sets in parallel with product system specs helped us to minimize non-compliance. **A** two-level (consistent) taxonomy for controlling and balancing the flow of specifications (inputs, requirements, and constraints) throughout the product development process were used.

In addition, we used a consistent naming convention for the template-parameters, so that they carry equal values if they

have identical assigned names**.** This made **passing** the specification values from system specs to SmartParts specs and among templates of the SmartParts themselves a very trivial problem. We piped two or more processes so that the next process can start just after the first one is finished generating the required information for the next process (just-in-time). This shifted the controls of the original product realization process to simply "**managing the individual template-based SmartParts**." SmartParts methodology described earlier took care of local satisfaction of constraints and essential communications among the three intra-part tracks represented by those templates. Only what has been modified at system level or changed during a SmartParts sizing-level template solutions were exchanged across templates. The methodology allowed us to solve the decomposed SmartParts in parallel to each other—in no specific order.

The result of this implementation has produced a tremendous savings in reusability of knowledge, reduced product lead time, reduced errors, quality improvements, and customer satisfaction. It has also helped us in driving down the costs of new product development.

10. ACKNOWLEDGEMENTS:

Authors would like to thank Parker Hannifin, Aerospace Group for providing R&D funding. Thanks to the Control Systems Division personnel for giving their valuable time for mining their product and process knowledge in order to built a prototype system for configuring a Hydraulic Actuator. The concept developed in this paper is quite general and could be applied to any product, process or system. The theoretical basis was based on some of the authors' prior works [Prasad 76, 77]. However, its implementation using KBE is completely new and was never tried before. Authors would like to thank Mark Czaja, Matt Ivary, Glenn Zwicker, Glenn Kirkendall, and a number of close associates of the authors including Valori Zaffino, Brian Tims, Scott Leland, Hector Espinoza and Constante Manapsal. Without their help, its successful implementation would not have been possible.

11. REFERENCES

1. Chen, B., and Menq, C.-H., 1992, "Initial Attempts On the Characterization of Functional Requirements of Mechanical Products," PED-Vol. 59, Concurrent Engineering, ASME WAM, 1992, Anaheim, CA, pp. 315-329.
2. Deming, W.E., 1993, The New Economics, Cambridge, MA, published by MIT Center for Advanced Engineering Study, November 1993.
3. Finger, S., and Rinderle, J.R., 1989, "A Transformational Approach to Mechanical Design Using A Bond Graph Grammar", ASME DE-Vol. 17, Design Theory and Methodology -DTM '89, Edited by Elmaraghy, Seering and Ullman, ASME Ist Int'l Conference on Design Theory and Methodology, Montreal, Quebec, Canada, Sept. 17-21, pp. 107-116.
4. Nielsen, E., 1990, "Designing Mechanical Components with Features", Ph.D. Thesis, University of Massachusetts, Amherst, MA.
5. Prasad, B., 1996, Concurrent Engineering Fundamentals: Integrated Product and Process Organization – Volume I, Prentice Hall PTR, Upper Saddle River, New Jersey.
6. Prasad, B., 1997, Concurrent Engineering Fundamentals: Integrated Product a Development – Volume II, Prentice Hall PTR, Upper Saddle River, New Jersey.
7. Prasad B., 1999, "Enabling principles of Concurrency and Simultaneity in Concurrent Engineering," Artificial Intelligence for Engineering Design, Analysis and Manufacturing, Vol. 13, pp. 185–204,Cambridge University Press 0890-0604099.
8. Pugh, S., 1991, Total Design, Addison-Wesley Publishers, Wokingham, UK.
9. Rogers, J. and Prasad, B. "Getting the Most Gains Out of Knowledge-based Engineering – Parker Aerospace Experiences", 2004 Annual Conferences & TechniFair, April 25-28, Fontainebleau Hilton Resort, Miami Beach, Florida, 2004.
10. Stauffer, L.A., and Slaughterbeck-Hyde, 1989, "The Nature of Constraints and Their Effect on Quality and Satisficing", ASME DE-Vol. 17, Design Theory and Methodology -DTM '89, Edited by Elmaraghy, Seering and Ullman, ASME Ist Int'l Conference on Design Theory and Methodology, Montreal, Quebec, Canada, Sept. 17-21, pp. 1-8.
11. Suh, N.P., 1988, The Principles of Design, Oxford University Press, Oxford, UK.
12. Szucs, E., 1980, "Similitude and Modeling", Elsevier Scientific Publishing Company, 1980, pp. 42.
13. Taylor, D.A., 1993, "Object-Oriented Technology - A Manger's Guide", Addison-Wesley Publishing Company, Reading, MA.
14. Thompson, J.B., and Lu, S.C.-Y., 1989, "Representing and Using Design Rationale in Concurrent Product & Process Design", in "Concurrent Product and Process Design", ASME Winter Annual Meeting, San Francisco, California, DE-Vol. 21, PED- Vol. 36, Dec. 10-15.