Electronic Delivery Cover Sheet

SHADY PEYVAN
MILLIKAN MEMORIAL LIBRARY
CALIFORNIA INST OF TECHNOLOGY
1200 CALIFORNIA BLVD
PASADENA, CA 91125
UNITED STATES

ORDER NUMBER:	**CI-02704011-5**
Account Number:	DD701890
Delivery Mode:	ARI
Delivery Address:	131.215.226.3
Submitted:	2002/04/10 16:31:49
Received:	2002/04/10 16:31:49
Printed:	2002/04/12 09:09:48

Direct	**Periodical**	**OCLC**	**UNITED STATES**
$75.00			
$ 75.00			

Client Number:	6368842 /PRASAD, BRIAN
Title:	**SYSTEMS ENGINEERING : THE JOURNAL OF THE INTERNATIONAL COUNCIL ON SYSTEMS ENGINEERING.**
Vol./Issue:	VOL 5 NO 2
Date:	2002
Pages:	123–144
Article Title:	BRIAN PRASAD: BUILDING BLOCKS OF A DECISION–BASED INTEGRATED PRODUCT DEVELOPMENT AND SYSTEM REALIZATION PROCESS
Report Number:	OCLC 38498986
Series Title:	WILEY SERIES IN SYSTEMS ENGINEERING
Publisher:	NEW YORK, NY : WILEY, C1998–
Information Source:	<TN LENDER: *CAI,CAI

INSTRUCTIONS: PLEASE ARIEL 131.215.226.3

Estimated cost for this 22 page document: $9.5 document supply fee + $30 copyright = $39.5

Phone/Téléphone: 1-800-668-1222 (Canada - U.S.) (613) 993-9251 (International)
Fax/Télécopieur: (613) 993-7619 www.nrc.ca/cisti producthelp.CISTI@nrc.ca

Regular Paper

Building Blocks for a Decision-Based Integrated Product Development and System Realization Process

Biren (Brian) Prasad

California Institute of Technology, Engineering and Sciences Division and Spec2market Solutions, KBE *Product Business Unit,* P.O. *Box* 3882, *Tustin,* CA 92781-3882

Received 16 May 2000; Accepted 3 October 2001

ABSTRACT

A key requirement, in a distributed product development environment at General Motors, was to provide a series of quantitative and qualitative mechanisms for integrating competing information from distributed agents. Such mechanisms must also account for provisions for combining different opinions, for resolving conflicts, and for finding a feasible or optimal solution at the end. We, at Electronic Data Systems (EDS), General Motors Account, developed a Decision-based Integrated Product Development (DIPD) methodology to capture a system-level optimization formulation as part of a product design, development and delivery (PD^3) process. The paper describes this methodology in the context of system-level optimization. DIPD employs the inputs, requirement, constraints, and output conventions to formulate the product realization problem in a distributed manner. The purpose of this DIPD methodology is to improve the performance characteristics of the product, process, and organization (PPO) relative to automobile consumer needs and expectations. DIPD builds the theory through a systematic revision and extension of the paradigms introduced earlier by optimization experts and practitioners including this author [Prasad, 1996]. The eight parts of this DIPD methodology, called building blocks, are discussed at length in this paper. The first four blocks, 1–4, provide a conceptual framework for understanding the challenges and opportunities in DIPD. The last four parts, 5–8, of this methodology provide the building blocks for an analytical and conceptual framework for decision-making, PPO improvements, and a large-scale system optimization. © 2002 Wiley Periodicals, Inc. Syst Eng 5: 123–144, 2002

E-mail: Prasadb1@cox.net

Systems Engineering, Vol. 5, No. 2, 2002

Key words: concurrent design; integrated product development; decision-based system; system optimization; product and process realization and improvements

1. INTRODUCTION

The product environment in modern manufacturing is commonly very complex. It consists of many components of products, processes, organization (PPO) and services, including information services (hardware, communication network and CAX (computer-aided "X"-ing software). Here, X (in CAX) can take several values like "analysis," "Design," "Engineering," "Manufacturing," etc. If X = "Design," then CAX means CAD software. If X = "Engineering," it is CAE software. If X = "Manufacturing," then CAX becomes CAM. The design of an automobile at General Motors, for example, contains 2000–3000 parts, and involves thousands of engineers making millions of design decisions over its life cycle. None of these parts are designed and developed in isolation from each other [Eppinger et al., 1994].

Figure 1 shows a process analogy of a conventional product design, development, and delivery (PD^3) system. It compares a conventional process of product realization with a process of fluid flow running through a maze of pipes. Each pipe of the piping assembly for fluid flow represents a part or an information buildup activity in a conventional PD^3 process. Serial engineering process (as opposed to concurrent engineering process) involves a number of connected parts or repeated activities of an assembly, such as plan, redo, download,

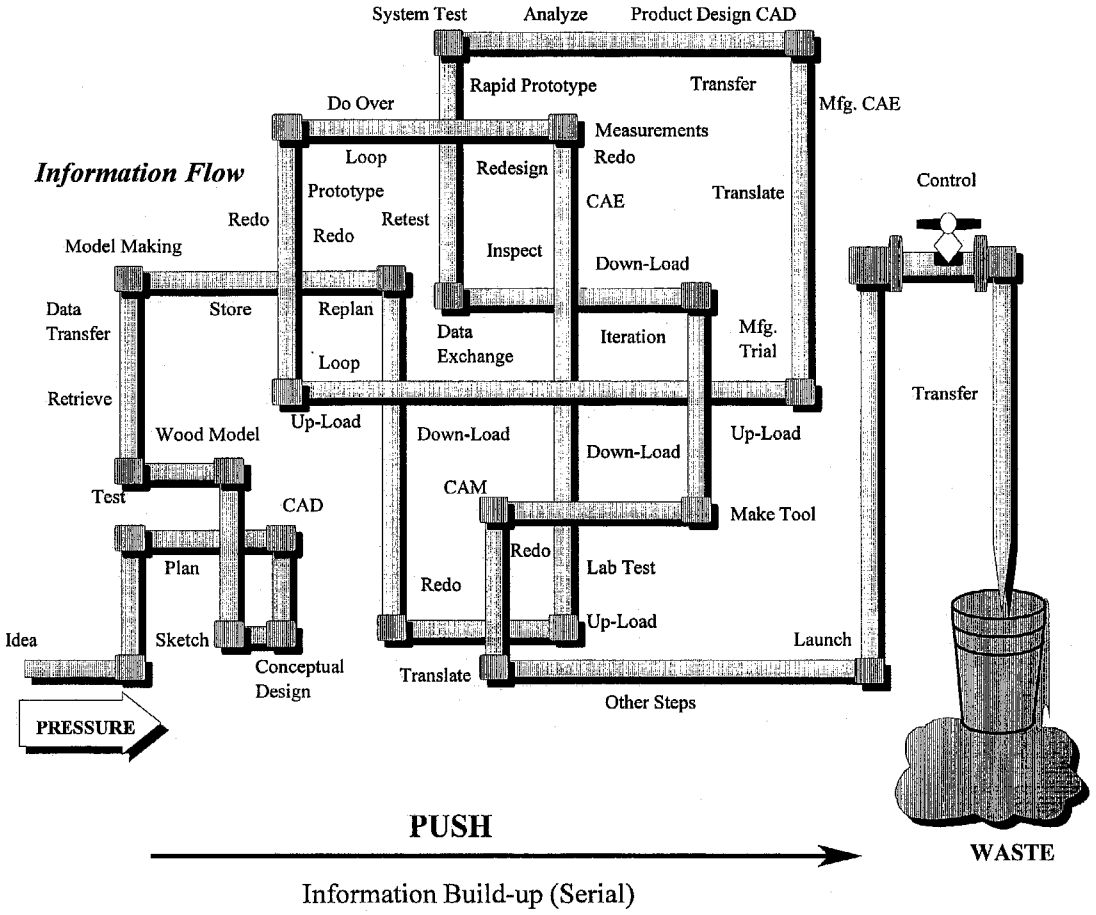


Figure 1. An analogy for a serial PD^3 process.

upload, iteration, retrieve, store, etc., which must be performed in the proper sequences.

There are a lot of similarities between a physical fluid flow and information flow. The "*fluid*" flowing through the pipes denotes "*information*" flow of a PD^3 process. The "*fluid pressure*" is equivalent to needs for "*information buildup*." The activities or parts, to be designed, are represented as straight pipes. The "*cross-section*" of each pipe represents the corresponding "*design parameters*." A typical conventional decision-making step is shown in Figure 1 by a pipe elbow or an end-coupling. Similar to how an end-coupling changes the direction of the fluid flow, decision-making in the conventional serial process changes the steps or parts required for subsequent information buildup. The length of each pipe in the assembly denotes the time or efforts it takes to complete the current task or buildup the necessary information for the next serial step of a PD^3 process. Each design decision is a tradeoff affecting many other design parameters. Such a traditional breakdown of design tasks, even though it resembles a hierarchical pattern, is repetitive and inefficient. Decision-making in the conventional process therefore can be very difficult and total lead-time in a PD^3 process could be very large considering the magnitude and complexity (including coupling) of the products and processes that need to be addressed [Lewis and Mistree, 1998].

Beyond concurrency of the decomposed tasks, integration of distributed PPO (product, process, and organization) environments seeks to offer better life-cycle alternatives and product realization potentials. The product realization taxonomy described in Prasad [1996] establishes a methodology of systematically organizing the process necessary for new product realization and for developing future product upgrades [Berger et al., 1989], although this process taxonomy is useful for formulating and decomposing a system [Kusiak and Wang, 1993]. However, the taxonomic concepts do not take the work-groups to the next step, i.e., to synthesize or to optimize the system with respect to an identified set of Product, Process, and Organization (PPO) constraints. Such a formulation is called herein PPO design system. Taxonomy characterizes the PPO design system problem into a well-structured set of decomposed tasks. This means a taxonomy use converts the PD^3 process into a topology of networks showing how tasks are interconnected or even coupled [Prasad, 1996]. However, such taxonomy does not show how the network of tasks will be solved.

Well-structured tasks are amenable to a variety of solution techniques such as analysis, simulation, sensitivity analysis, optimization, mathematical programming, and other weak numerical techniques [Prasad and Emerson, 1984]. Weak numerical techniques are approximate representations of the problem domain, like graphical integration methods, idealization of the problems (e.g., beam's equations) and Taylor's based approximations (e.g., approximation concepts in optimization) or iterative-based equation solvers. Decision-based design problems are generally coupled problems [Lewis and Mistree, 1998]. There are two types of complexity in a PPO design problem, one is "*structural complexity*" and the other is "*computational complexity*" [Prasad, 1984a, 1984b]. The two are not the same. Structural complexity is resolved through "*problem decoupling and simplification*," [Eppinger et al., 1994] whereas "*computational complexity*" is resolved through innovative use of "*problem solving*" techniques [e.g., Thurston and Locascio, 1993]. Computational models and structuring techniques help design work-groups to reduce the "*structural complexity*." For example, de*coupling* of the tasks provides a structured road-map outlining problem solving steps [Prasad, 1985; Steward, 1981]. With predetermined computational models for PPO, work-groups could simply enter the specifications and the PPO synthesis model "*forward-solves*" the problem-set to provide the anticipated results. This process is very similar to how an engineering spreadsheet works. In PPO synthesis or system design optimization, unlike the spreadsheet, the computational model or work-group can "*back-solve*," entering the desired result and making the synthesizer find suitable values for the input specification. AI-based techniques are demonstrated to perform well in a closed environment (well-structured symbolic or rule-based domain) that uses weak methods of problem solving. There are few commercial tools that can accept a set of mathematical equations, which are pre- or user-defined, and analyze or synthesize the problem on a case-by-case basis. However, most design synthesis tools require the problems to be explicitly defined or have their characteristics explicitly known.

The paper presents a conceptual framework incorporating a system optimization methodology that can be used in the context of systematic improvement of a PPO design. System optimization is employed here to indicate a process of finding interdependency among distributed agents, defining strategies, and finding a set of feasible or a consensus-based solutions that resolve the multidisciplinary conflicts.

2. DECISION-BASED INTEGRATED PRODUCT DEVELOPMENT (DIPD) METHODOLOGY

DIPD is described as the process of going from a set of incomplete and inconsistent product requirements to

realizing a physical product [Prasad, 1997]. As stated earlier, this methodology has eight parts to it. Each part contributes to the overall effectiveness of the DIPD process. These parts are listed here and shown graphically in Figure 2.

1. Product Requirements Planning & Management
2. Work Structuring & CE Team Deployment
3. Methodology Systematization
4. Product and Process Systematization
5. Problem Identification and Support System
6. Integrated Problem Formulation
7. Collaboration and Cross-functional Problem Solving
8. Continuous Monitoring and Knowledge Upgrade

The first four blocks involve the construction and definition of problem statements to pin-point the artifact's functions, components, subfunctions, and their interconnections. The first building block of DIPD is the determination of "*product requirements planning and management.*" The product requirements in the beginning are often incomplete and may be inconsistent. As the product moves into subsequent steps, it tends to become complete and consistent. The second building block is "*work structuring and CE team deployment.*" Structuring of work facilitates the integration of complementary engineering expertise [Browning, 1999]. This section discusses the dilemma of dividing responsibilities so that each work-group can work in parallel and yet collaborate for joint decision-making. The first and the second building blocks together are called the planning phase. The third building block, "*Methodology systematization,*" outlines the basic conceptual framework for DIPD methodology [Chandrasekaran, 1989]. The fourth building block, "*Product and Process systematization,*" lays down the taxonomy of product and process transformation leading to a physical artifact. The third and fourth building blocks together are called the systematization phase. This phase implies understanding the ability of the manufacturing process and communication of upstream

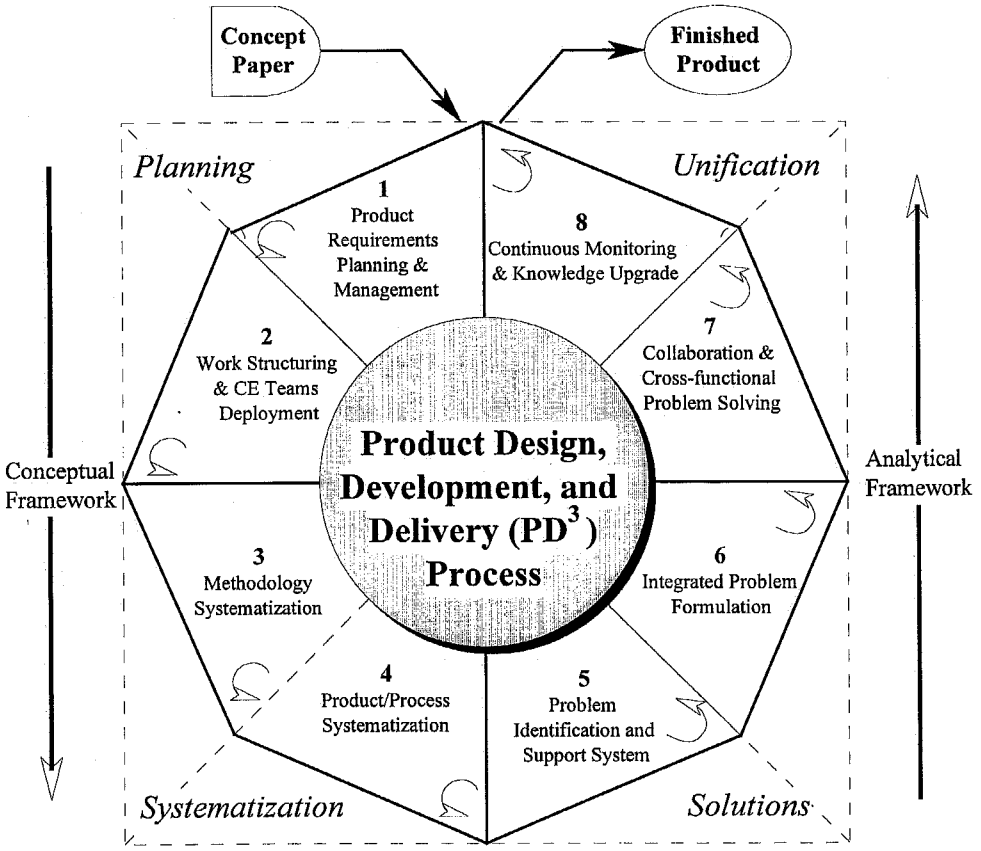


Figure 2. Building blocks of an integrated PD^3 process.

and downstream concerns to produce consistent production parts. While the systematization phase of DIPD may consume only 5–15% of the enterprise resources, it casts a long shadow. When this phase of DIPD is complete, most manufacturing expenses, as well as value characteristics of the so-called "would be product" have already been committed [Prasad, 1996]. This represents, therefore, an important step in the PD^3 process. The key strategy of the third and fourth building blocks is to predict problems associated with the methods and product systematization [Clark and Fujimoto, 1991]. This way the problems can be dealt with adequately or changed, if needed, when the initial cost of their modifications is reasonable.

The remaining four building blocks outline a methodology for arriving at an optimized design or a consensus-based design alternative. The eight steps together describe solution strategies based on what is algorithmically feasible (from the contributions of blocks 5 and 6) and what is possible through a consensus-building approach (from the contribution of blocks 7 and 8). The fifth and sixth building blocks together are called "*solutions*." In the sixth block, "*an integrated problem formulation approach*," the methodology determines what constraints are violated, in addition to giving an alternate set of optimized solutions. In the seventh block, *consensus-based approach*, the methodology combines different opinions, which may or may not be analytically based. The consensus resolves key conflicts and provides a set of possible solutions in addition to the conflicts that cannot be resolved. Collaboration means coordination of work-group problem solving abilities. The last building block, "*continuous monitoring and knowledge upgrade*," emphasizes the need for continuous improvement in PPO. Knowledge upgrade refers to updating the product and process knowledge captured in prior steps or previous iterations. The structuring of work and the need for redefinition of the PPO methodology within the Concurrent Engineering framework (taxonomy) was discussed in Prasad [1996] at length.

Interdependency can be classified as competitive, complementary and cooperative. A consensus-based solution, then, entails a cooperation of multiple competing perspectives maneuvering through all considerations of the product's life-cycle values. Building blocks 7 and 8 together are called "*unification*." Although this definition of unification will streamline the product realization process during the search for the best design, a solution cannot be arrived at without a good *problem identification and support system* (block 5) followed by *integrated problem formulation* (block 6). Therefore, all eight facets must be used within the context of a cross-functional team as a part of a simultaneous or concurrent engineering effort. In reality, most of the above eight steps are often dependent upon each other. This could require the cross-functional teams to repeat one or more of the above steps.

2.1. Product Requirements Planning and Management

Effective identification of design variables, constraints and objectives at each level of transformation requires a high level of intelligence on the part of the product development team (PDT). Human resources and expertise are needed to analyze adequately the transformation process and identify the optimization model from the prescribed set of inputs, requirements and constraints (RCs), and outputs. It is still beyond the current capabilities of modern computers to automate these functions. Various types of optimization models may be necessary to satisfy RCs such as incorporating life-cycle values or assessments [Thurston and Locascio, 1993]. The types of assessments are established in the ***Measures of Effectiveness*** (MOEs). The unknowns are the priority, the distribution of these MOEs into objectives, and distribution of specifications into design variables and constraints. The latter may be looked upon as a type of transformation from a "*conceptual model*" state to an "***optimization model***" state. The conceptual model refers to a state where requirements are fuzzy and problem at hand could not be represented on firm mathematical footing. The optimization model refers to representing the same using mathematical programming or equivalent means. This is not an easy process. The model depends upon many factors: prior design history, life-cycle priorities, and other value implications. A team of work-groups with different expertise may be required to formulate the optimization model that contains all essential elements of customer needs and RCs.

A systematic classification scheme for distributing RCs during product development must be designed. This is viewed in Figure 3 as a graphical pyramid. The pyramid has three incline triangular ceilings and a base. Each triangular ceiling is supported by three sides belonging to RCs for product, process, and enterprise, respectively. In modern manufacturing, product, process, and enterprise are constantly entwined. The base of the pyramid is the foundation—the CE philosophy of product realization on which the remaining three sides stand. In order to support cooperation amongst teams, generally a variety of team attributes are often required to support the CE infrastructure and provide an overall enterprise-wide "consistency of purpose." Prasad [1996] has proposed a set of team attributes and collectively name it as 7Cs (*Collaboration*, *Commitment*,

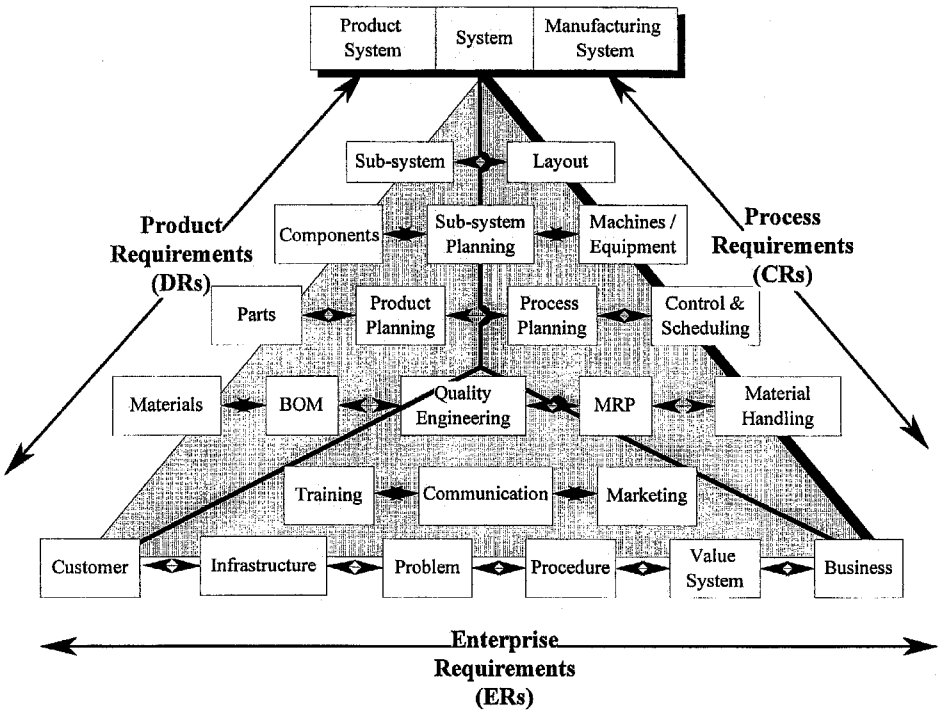


Figure 3. A systematic scheme for classifying requirements and constraints (RCs).

Communications, *Compromise*, *Consensus*, *Continuous Improvement*, and *Coordination*). Consistency of purpose cannot be accomplished without a standardized set of operating principles and adequate resources to conduct the business operation. Such business principles are categorized by Prasad [1996] as 3Ps (*policies*, *practices*, and *procedures*), and operating resources by 4Ms (*money*, *machines*, *manpower*, and *management*). In addition, the individual strength of each team comes from its constituencies. Prasad [1996] proposes a set of seven qualifying characteristics to judge its strength, namely, *talents*, *tasks*, *teams*, *techniques*, *technology*, *time*, and *tools* and refers to them collectively as 7Ts. The CE infrastructure gets its strengths from these 7Ts, which form the base of the pyramid. The foundation base touches the ceiling sides and supports them all. Corresponding to each side of the foundation, RCs are shown classified into three distinct forms corresponding to the three major subdomains:

- Product requirements (TRs) for product design subdomain
- Process requirements (SRs) for production process subdomain
- Enterprise requirements (ERs) for customer and business subdomains.

Along each base side, the RCs are further branched into independent block structures to generate the next level of transformation, which can best serve the needs of different life-cycle subdomains. The pyramid model shows perfect symmetry with equilateral sides, but it is unlikely that this will happen in all cases. There are some CRs, DRs, and ERs that are common; e.g., manufacturing tolerances may appear at all places. The RCs are directly related to the geometrical form of an artifact, which can be used to infer the functional form of a product at each level. In order to simplify the reasoning process and to better characterize the relationship between RCs and geometrical forms, these RCs can be represented as a hierarchical tree-type structure [Steward, 1981]. This hierarchical structure is based on the assumption that RCs can be propagated into a lower level RCs at each loop level (see Prasad [1996]) independently.

In order to maintain a "constancy-of-purpose" it is essential that the product realization process be branched into its smaller pieces. This results into multiple loops of the product and process design (see Prasad, 1996: Fig. 8.16). The loops are actually a part

of a larger system called product realization taxonomy, which is discussed in greater detail in Chapters 8 and 9 in Prasad [1996]. The existence of loops is mentioned herein to point out their dependence on the "whole system" concept. Interaction takes place between the teams and the loops, and amongst the loops themselves. The two key characteristics of this loop methodology are "iteration" and "integration." The process of iteration is a good classical feature of many "implicit-type" problem formulation and iterative solving methodologies. RCs at each loop level can then be mapped into their corresponding form features through a mapping process. By backtracking (concatenating the RCs), as we move from the bottom up, the actual functional form of the artifact can be inferred.

At the beginning of the DIPD process, initial specifications represent the highest level of abstractions representing a complete and consistent set of specifications for PPO (product, process, and organization) design. As various tasks within this taxonomy are performed, designs (meaning product and/or process designs) begin to take shape. The set of specifications changes over time. At the end (when time is up) all specifications have been implemented. At that point, the designs are at the "full content," and specifications are at the "null state."

2.2. Work Structuring and CE Team Deployment

Work structuring defines a work breakdown structure (WBS). A WBS and its interrelationships allow for the structuring of tasks at different breakdown levels. This is useful in product realization because different alternatives can potentially be generated and evaluated by using the system to execute WBS networks and by changing the 7Ts deployment and work structuring [Prasad, 1996]. Team or work-group deployment defines a PDT organizational structure [Steward, 1981] and a cooperative project team structure [Browning, 1999]. Work structuring includes timing for the various transformations applied to the PPO design cycles. Choices about the network at each level of the hierarchy—that is, what the activity relationship will be—is a tough problem [Browning, 1999]. Facilities to manage and organize the WBS at both computational and human levels are an important part of DIPD. Prasad describes a program timing schedule for 1-T loop, 2-T loop, and 3-T loop transformations [Prasad, 1996] including a set of taxonomy for loop transformations. In 1-T loop, the overlap between the tracks, due to concurrency, also has no significance since possible advantages from their interactions cannot be realized. In real situations, interactions among two or more of these parallel tracks do take place. Many work-groups and organizations recognize these interactions by activities differing in the level of interactions between the tracks.

Each loop represents an interaction between one or more essential tracks of product development. 1-T designation indicates interactions take place within the track itself. 2-T designation indicates that looping or interactions take place between two parallel tracks. Similarly, 3-T designation means that there are interactions between at least three parallel tracks. Depending upon the product type, its complexity, and the organizational setup, a series of these 2-T or 3-T loops can be employed to reach a final product realization state [Prasad, 1996].

Methodology systematization, on the contrary, has nothing to do with timing. Systematization sets a common methodology for problem solving that can be applied to any transformation.

2.3. Methodology Systematization

A product realization process usually involves a large number of design and analysis activities that need to be managed. The quality or depth of information about a PPO design solution evolves during this realization process. Without any methodological decomposition, it is not possible to process the design constraints of a subassembly or an individual part since at that point, many of its details are unknown. It has long been recognized that problems, no matter what their size or complexity, can best be solved by working through a sequence of steps [Warfield and Hill, 1972]. Steps systematize the methodology of problem solving, which, in turn, helps to prevent adverse situations. Researchers now recognize systematization as a fundamental approach to understanding and controlling the interaction between the constituent elements [Ulrich and Eppinger, 1994]. Another important reason why designers work hierarchically is that an individual person or a work-group is not able to process large numbers of constraints simultaneously [Clark and Fujimoto, 1991; Clausing, 1994]. Systematization offers a powerful tool to reduce the inherent complexity of the problem domain [Steward, 1981]. Systematization methodology ensures that everything possible will be done to apply the 7Ts resources in the most effective manner. One such act of systematization in product design is to apply decomposition—branching the PPO design into loops, loops into activities, and finally activities into tasks. Stefik [1981] describes further motivations for decomposing PPO design and manufacturing activities:

- The apparent structural and computational complexity of a PPO design and manufacturing problem is often reduced as a result of decomposition.
- If the decomposition is done with a view to minimizing interdependence, while the activities are split into tasks, each discrete task can then run in parallel.
- The PPO problem is reduced to a series of self-contained smaller activities or tasks. For example, most of the details of a subsystem are irrelevant when the design problem is dealt with at the system level.
- The talent and expertise of designing sets of decomposed problems can be divided among the area specialists. " Each team of work-groups can be assigned to work on each decomposed set concurrently.
- This enhances concurrency of the product realization process.

There are many levels of abstractions in systematization. In problems as complex as DIPD, systematization starts with a PPO (product, process, and organization) management. In Prasad [1996], a loop concept was introduced to manage the interactions between different life-cycle phases. Each loop consisted of five major components: a "baseline system," inputs, outputs, constraints, and requirements (see Prasad [1996]). In general, the solution to problems of this type can be approached by a four-stage systematization process as shown in Figures 2 and 4:

Stage 1. Planning: This is the initial stage when system specification is defined. It may not be possible to prescribe a complete system specification during planning; some requirements are often incomplete. Partial ordering of intermediate goals is, therefore, identified and a view to early determination of gross features is conceived.

Stage 2. Systematization: There are two kinds of PPO systematization: methodology systematization and product and process systematization. Systematization is the

- systematic decomposition of the PPO problem into discrete subproblems,
- decomposition of product and process specifications into different levels of abstractions,
- description of this abstractions in terms of the functional and/or physical elements,
- identification of the interactions that may occur between these elements, and then

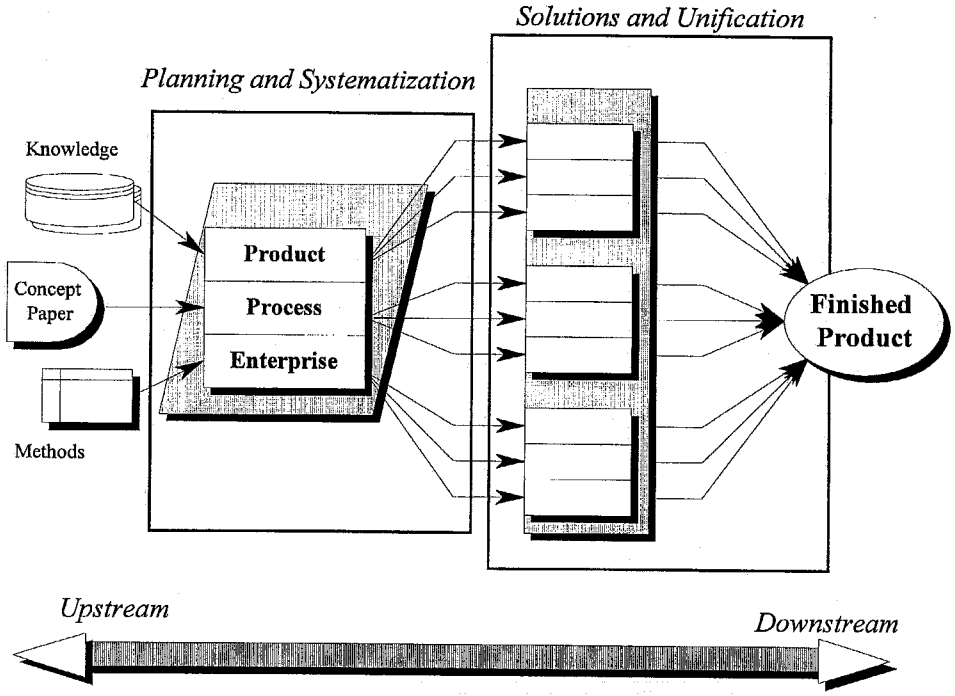


Figure 4. DIPD methodology systematization.

- aggregation of the discrete constituents back into their high level original definitions.

Stage 3. Solutions: After Stage 2 is completed, it is followed by a series of solutions of the decomposed subproblems, specifications, or constituents. In this stage, a number of alternatives or options are obtained for each sub-problem, and the best solution is selected in each case.

Stage 4. Unification: The fourth stage is aggregation or reconstruction of an overall system solution from the various solutions to the subproblems. Unification is an important step in solving DIPD problems while using the decision-making process. Unification may involve system optimization and may consist of a higher level of decision-making and complex reasoning. Unification begins with the "outside-in," where a pair of parts is designed from basic information (such as materials, features, etc.). Later, it is rolled up into larger and larger parts, components, subsystems, and then into a system via a design aggregation process. During this process the decomposed parts are assembled to form components. Components are brought together to form subsystems. Subsystems are assembled to form systems; and finally, systems are brought together to construct the artifact as a whole. Unification, thus, helps eliminate the constraint violations and yields to refining the PPO for interface minimization considerations.

The relationships between the four stages of systematization and the corresponding steps of the building blocks mentioned earlier are given in Table I. The above four-stage process highlights the major steps to be undertaken in tackling a DIPD problem. The process can be regarded as a continuous cycle of improvement. Systematization is an important step in yielding a faster and better solution. It reduces number of searches and makes product realization more efficient. If the PPO decomposition were not disjoint, the subproblems would neither be discrete nor their attribute sets necessarily uncoupled. It is likelihood that such decomposition of the initial problem into a set of subproblems may suffer from excessive interdependence. It is possible to have a large number of constraints and design variables, which could be common to these subproblems. A few (one or two) iterations of the above four-stage process may not therefore yield a "good" solution. Several of the constraints could be in conflict. It may require an excessively large number of iterations. In such cases, PPO decomposition may not have many real benefits. Real benefits are obtained when the four-stage process results in significant savings in time and effort, compared to solving the original system problem as a "*huge global optimization problem.*"

2.3.1. Branching and Bounding Methodology

A branching and bounding methodology has been used here to first branch the product and process subdomains into loops. Later, the methodology bounds these loops into a "subdomain" and then enfolds them back into a complete system. The notion of branching allows for the exploration of "*possibilities*," whereas bounding provides a way for the concurrent workgroups to judge the intermediate outputs.

- ***Branching:*** Branching of each domain into loops is carried out uniformly as shown in Prasad [1996]. A consistent representation is used throughout the branching and bounding process. In each group, a "*baseline system*" can be further branched into independent subunits, which can better serve the needs of these loops. The three loops—feasibility synthesis, design synthesis, and process planning synthesis—provide a basis for satisfying product-oriented requirements giving rise to the so-called product-oriented loops (see Fig. 5). The other three loops—process planning execution, production synthesis, and opera-

Table I. Relationship Between Methodology Systematization Stages and Eight Steps

Four Stages and 8 Steps	Name of Stages	Corresponding Steps
Stage 1	Planning	• Product Requirements Planning & Management
		• Work Structuring & CE Team Deployment
Stage 2	Systematization	• Methodology Systematization
		• Product and Process Systematization
Stage 3	Solutions	• Problem Identification and Support System
		• Integrated Problem Formulation
Stage 4	Unification	• Collaboration and Cross-functional Problem Solving
		• Continuous Monitoring and Knowledge Upgrade

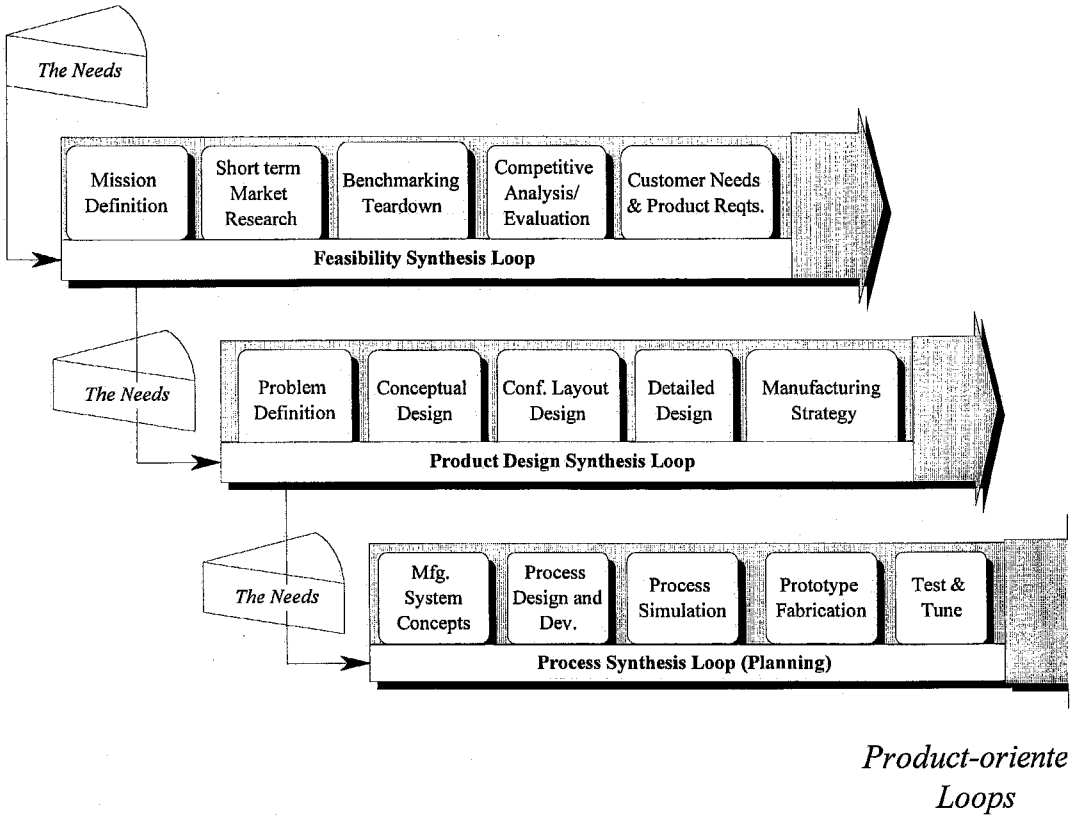


Figure 5. Concurrent elements of product-oriented loops.

tion synthesis—provide a basis for satisfying process requirements giving rise to the so-called process-oriented loops [Prasad, 1996]. The second level and third level breakdowns of the product-oriented loops are also shown in Figure 5. The constraints in each of the individual loops provide a basis for determining the "goodness" of a candidate baseline system.

- ***Bounding:*** This provides a mechanism to evaluate the goodness of a baseline or a candidate system with respect to meeting a common set of requirements. During this bounding phase, the system computes the "*goodness value*" based upon the constraints that are still unsatisfied up to this stage. The common PPO constraints in the two half-domains provide a basis for determining the "*goodness*" of the total system [Prasad, 1996]. The system value—a cumulative index on the measure of PPO violations—envisions the trade-off possibilities or further exploration of the problem domain. This may require a comparison of each synthesis loop's outputs to the system's goals and objectives, refinement or reallocation of requirements, reevaluation of lower level objectives or reconfiguration of goals. Bounding occurs through the use of concurrent function deployment, or similar techniques, for simultaneous consideration of a series of competing requirements and objectives.

The branching and bounding methodology provides the ability to refine successively the "*goodness or fitness*" of a baseline concept as one proceeds from one nested product realization loop to the other [Prasad, 1996: Chap. 8]. The domain of product realization process is, however, evolutionary. During a loop, the design or concept formulation is not fixed; rather it reflects the CE teams' understanding of the PPO design problem and the environments spanned by its full specification sets. As the satisfaction of the specification continues during each loop, the work-groups learn more about the forthcoming baseline model. Similarly work

group learn more about the output (solution) state as new aspects of its (each loop's) behavior inherent in the formulation are revealed. As a result, work-groups may gain new insight into the behavior of the model (and/or the solution output-state) that may have an effect in reformation of a new set of specifications or changing the baseline system concept. This process of learning and reformation can continue until one or more of the following conditions are met:

- The incremental change in behavior of the baseline system concept due to change in inputs becomes insignificant (produce no change in outputs).
- The requirement sets are empty; no more requirement is left to be satisfied.
- The incremental satisfaction of the constraints becomes contradictory, or results in a concept that is too costly or cannot be manufactured.

It is important to be able to use the prescribed evaluation criterion for each candidate baseline system in order to guide the PPO redesign process as the product evolves from problem or customer needs to an instance of an artifact. Each branch and bound procedure inherent within each loop provides a mechanism to select an increasingly good PPO design or concept. Any intermediate trial designs based on such bounding procedure are subject to iterative rework, or can potentially be discarded. This is quite natural. This has always been the case in the conventional process too when someone chooses to design a part from an incomplete or uncertain data. So what is different with DIPD? DIPD employs a taxonomy-based realization process. Without the taxonomy (ability to classify the PD^3 process) in a conventional process, one is forced to instantiate all possible configurations and check the compliance with respect to all possible value characteristic requirements. It may not be cost effective to carry out this instantiation process every time taking account of the inherent complexity of the product. Another alternative is to ignore many possible configurations, or consider a subset of inputs, requirements or constraints one at a time. However, this could result in a series of PPO design concepts that are suboptimal in some way. A taxonomy-based methodology acknowledges the risk up front and provides a taxonomy-based procedure to manage this risk appropriately. The four-stage DIPD methodology thus provides a built-in risk management technique. It balances the needed reduction in time-to-development against the risk of concept changes in the form of iterations and loops.

2.4. Product and Process Systematization

Concurrency of tasks can be exploited by differentiation, followed by systematization—organizing the information in a hierarchical way [Prasad, 1996]. In most product systems, there are complex interactions among many of its constituents: subsystems, components, parts, materials, features, etc. Product systematization is a technique of handling a larger class of problems, or a product, by decomposing and then concatenating the results of its behavior through a smaller set of problems or hierarchical organization [Koch, Peplinski, Mistree, and Allen, 1996]. Examples of an automobile, an aircraft, and a helicopter are shown in Prasad [1996], respectively. Similarly, a process can be decomposed into activities. A group of activities aggregated into a high-level activity group is called a scenario [Kusiak and Wang, 1993]. By systematization of process in the early stages of product development, the work-group can compare various scenarios. Concurrency can be affected by studying the dependency of the decomposed set or scenario. If one is able to reduce the dependencies among the decomposed sets or scenarios, concurrency can be increased. Concurrency can also be increased, and interdependency reduced, if one is able to maintain precedence between the consecutive decomposed sets or scenarios. Since the effect of maintaining precedence between tasks is reduced interdependence, the degree to which concurrency can be affected depends upon the mode of product decomposition into constituents or process decomposition into activities. Figure 6 shows an example of two scenarios of the same process. The process shown in scenario X has been decomposed into five activity-groups A–E as shown in scenario Y. Such scenarios are said to be "serially decomposable." The constraint equations between the scenario Y activities can be solved serially, yielding the value of one new activity for each constraint evaluation. When a set of constraint equations is not serially decomposable, other ordering methods are used [Navinchandra, Fox, and Gardner, 1993] to avoid solving a large set of equations simultaneously. Activities within a group can run in parallel. Using the discretization proposed here, there are two parallel activities in activity group A, four in group B, and three in groups C, D, and E (see Fig. 6). In this discretization, please note each activity has a starting point and an end point. Results would look different, if one chooses other methods of discretization. The activities within Groups A–D can be overlapped if the dependencies of the interfaces are not very strong. As differentiation proceeds, emphasis changes to interfaces between the decomposed sets: between system and subsystems, between subsystems and components, between components and parts, and between

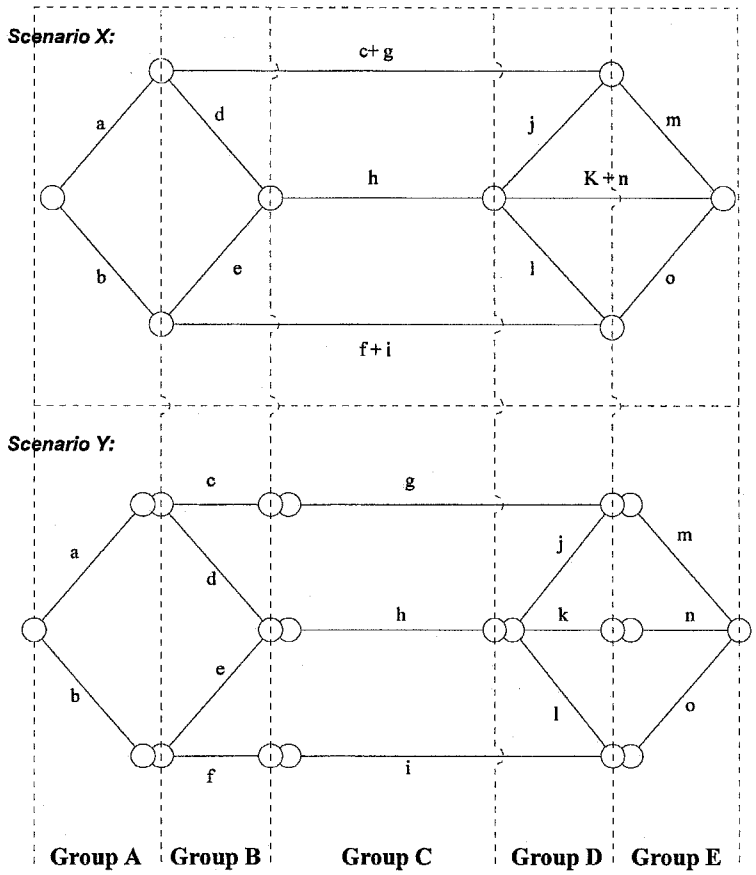


Figure 6. Decomposition of a scenario into a "serially decomposable" activity groups.

the constituents themselves (see Prasad [1996]). Even though there has been a number of recent works on multilevel optimization [Sobieszczanski-Sobieski, 1993] and decomposition [Browning, 1999; Ulrich and Eppinger, 1994], the field of automated assembly, or product system realization, is still new and growing [Prasad, 1997].

2.5. Problem Identification and Solving Methodologies

There are two steps to solving a typical design problem: (i) developing a problem identification scheme [Agrell, 1994] and (ii) identifying a suitable method of solution [Dowlatshahi, 1992]. The substeps involved in developing a problem identification scheme for a design concept are:

- Identification of parameters or design variables for input modifications,
- Identification of criteria for generation of objectives, requirements, and evaluation of constraints,
- Generation of modification,
- Selection of a suitable PPO—product, process or organization—model and executing it,
- Reevaluation of constraints and further problem identification (such as sensitivity analysis),
- Selection of parameters based on a particular solution methodology (such as trade-off or optimization) that satisfies the imposed constraints and meets the stated objectives [Pugh, 1991].

Problem solving methodologies are discussed next.

In DIPD schemes, work-groups require multiple problem solving methodologies so that concurrent teams can move from one level of abstraction to another, or from one viewpoint to another, as the design concept evolves [Dowlatshasi, 1992]. This may consist of developing a number of alternate design schemes and possible alternate arrangements. Members of the CE

work-groups can employ one or more of the following methods of solutions depending upon the problems at hand:

- *Performance Improvement:* Here the objective is to find a set of parameters that will improve some performance characteristics that are monitored. The goal is to move the product performance characteristics to lie within an acceptable range. The current design details and parameter values and bounds are analyzed or tested against acceptable criteria and to determine possible performance violations.
- *Parameter Design:* Here the objective is to minimize the impact of external uncontrolled processes or minimize the impact of process variations, which cannot be controlled.
- *System Design:* Here the objective is to configure a system with an alternate sets of design alternatives, an alternate set of material possibilities, and/ or a new method (process), which will provide a similar (close to previous values) or equivalent performance. The interactions and relationships between subparts of a product are measured with respect to a global set of system criteria.
- *Mathematical Programming:* Here the objective is to find a set of parameters that will satisfy a performance objective subject to a given set of constraints. The performance objective is a single function.
- *Multicriterion Optimization:* Here the objective is to find a set of parameters that will satisfy a set of performance objectives, subject to a given set of constraints [Agrell, 1994]. An example is a min-max problem.
- *Heuristic-Based Satisficing:* This employs heuristic based rules to aid in estimating RCs. The objective is to narrow down the solution field by successive introduction of heuristic-based RCs until only one solution is left, or the solution converges to an interesting part of the design space.

Most books on optimization, for example, concentrate on how to solve an optimization problem if it can be expressed in a mathematical form (such as a linear or a nonlinear function of design variables) [Prasad and Magee, 1984]. Such formulations are often of a closed type. The work-groups often find no difficulty in arriving at a suitable solution since all the necessary information about the problem is given or known. Work-groups also know when they are finished with the problem and generally know if it can be solved correctly. The most general statement of *optimization* problem posed is to:

Find a vector of design variables $v \in D$ that minimizes or maximizes

$$\text{a set of value characteristics (objective functions)} \quad VC_i(v), \tag{1}$$

while satisfying a set of constraint equations:

$$C_{ij}(v) \geq 0. \tag{2}$$

D is the design space in which the solution lies. The design space is actually an intersection of three sets: a set of requirements, a set of constraints and a set of design variables. Most optimization problems have four parts:

1. A transformation system
2. A set of objectives (a function or a criterion)—$VC_i(v)$
3. A set of design variables, v_i
4. A set of constraints, C_{ij}

- ***Transformation System:*** In most cases this is a part of the problem definition and, therefore, hidden. In books, this is often identified in an explicit form:

$$[T]\mathbf{v} = [O], \tag{3}$$

where T stands for transformation and O stands for output. Both are characteristic matrices and $\mathbf{v}$ is a vector of design variables. Such explicit forms define the problem—how the objective functions and how the constraints are related to design variables.

$$VC_i = f(\mathbf{v}) \tag{4}$$

$$C_{ij} = g(\mathbf{v}). \tag{5}$$

- ***Objectives:*** The objective is a function or criterion that characterizes the aspect of design to be improved. It is commonly posed as a single criterion problem, such as cost, weight, strength, stiffness, etc. At times, the objectives are to optimize under the presence of a set of multiple criteria and task levels [Prasad and Emerson, 1984]. The technical merits are established by evaluating key performance, reliability, structural integrity, and economy criteria and comparing these to the current best values forecast of these criteria [Agrell, 1994]. The problem of multicriterion optimization is to minimize or maximize a set of merit functions, simultaneously:

$$VC_i(\mathbf{v}), \quad \text{for } i = 1, 2, \ldots, qc\text{-max.} \tag{6}$$

For example, the design of a machine tool involves many aspects, such as transmission, control system, hydraulic components, power utility, and body frame. This is a case of configuration optimization. Configuration optimization is the process of first identifying the best from a group of configurations, and then embodying the configuration to provide the highest possible technical merit values, such as performance, reliability, durability, and economy. If there are several performance attributes, they can be combined using multiattribute utility theory [Keeny and Raiffa, 1976] to build a duty index. Similarly reliability attributes can be combined to obtain a "reliability index" and economy attributes can be combined to obtain a "cost index." Each index can be normalized with respect to a carefully selected benchmark solution—a datum value for comparison. Normalization scales the parameters and provides a better numerical stability and convergence.

- ***Design Variables:*** These represent those input parameters of a problem that are subject to change. The design variable is usually a vector, where

$$v = [\{v_{\text{sizing}}\}, \{v_{\text{shape}}\}, \{v_{\text{topology}}\}, \{v_{\text{knowledge}}\}]^T \tag{7}$$

There are four classes of design variables commonly used:

- *Sizing Variables*—$\{v_{\text{sizing}}\}$: These include variables like thickness (for thin-walled sections) and areas (for solid objects) that can be changed.
- *Shape Variables*—$\{v_{\text{shape}}\}$: These involve changing the configuration points or the geometry of the part that is represented such as length, width, height, coordinates, etc.
- *Topology Variables*—$\{v_{\text{topology}}\}$: These define parameters that actually determine where material should or should not be removed. As long as the topology change can be represented parametrically in the CAD system, the model can be optimized. Topology optimization allows feature patterns, such as how many bolts are needed to hold down a given part, or how many ribs provide a given stiffness.
- *Process Variables*—$\{v_{\text{process}}\}$: These involve changing the rules concerning the part's forming or processing needs, that have effect on changing the part's size, shape, topology, or functions themselves.
- ***Constraints***: These are the response parameters (state variables) of the model used to evaluate the design based on the criteria that limit how it should function or behave. They are usually specified in an equality or inequality equation form:

$$C_{ij}(v) \geq 0, \quad \text{for } j = 1, 2, \ldots, c\text{-max.} \tag{8}$$

Often such constraints also include limits on design variables:

$$\{V_{\min}\} \leq \{v\} \leq \{V_{\max}\}, \tag{9}$$

$\{V_{\min}\}$ and $\{V_{\max}\}$ vectors denote lower and upper bounds on design variables.

This completes the definition of a system-level optimization formulation. The transformation system ties the optimization model of the product with objective functions, optimization constraints and design variables in some mathematical or conceptual forms (see Fig. 7). The objective functions are derived from performance type outputs of the transformation state. The set of constraints and the objective functions is derived from behavior type outputs. Examples of behavior type outputs include deflection, noise, vibration, frequency, stiffness, strength, etc. Design variables are derived from specification attributes (inputs, requirements, and constraints). Note the subtle differences between the input and output constraints. Transformation state constraints are the inputs to the baseline state whereas the optimization constraints are outputs from the baseline state. Books on optimization seldom focus on the transformation system. They assume it to be given or explicitly known. However, most physical problems cannot be modeled purely in an explicit form. For example, in a minimum structural design problem, the transformation system exists in a finite element model or a similar form. The relationship between constraints (such as stress and deflections) and objective functions (such as weight) is tied to the stiffness matrix of a FEA model. Consequently, in most cases, design tradeoffs are made tacitly and implicitly. An implicit statement of a problem at a system level is shown in Figure 8. The solution of the problem consists of:

a. Minimizing a set of functions
b. Maximizing another set of functions and at the same time
c. Desensitizing some parameters of the problem.

Most product designers in industry are not familiar with how to express the transformation system as an optimization model, explicit or implicit, so that the problem can be optimized. An open-ended optimization problem usually takes longer to solve than a closed form optimization problem. Teams must evaluate the formu-

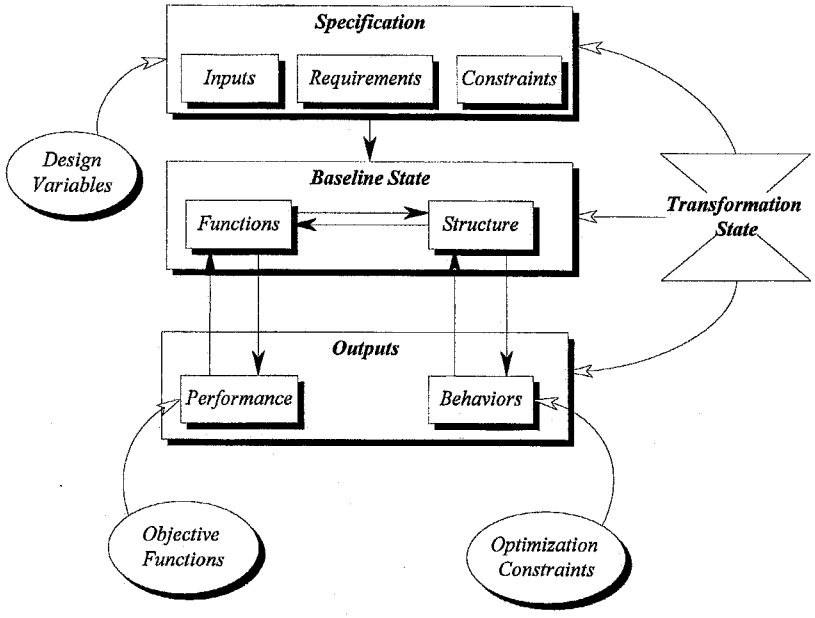


Figure 7. Relationship between a transformation state attributes and an optimization model variables.

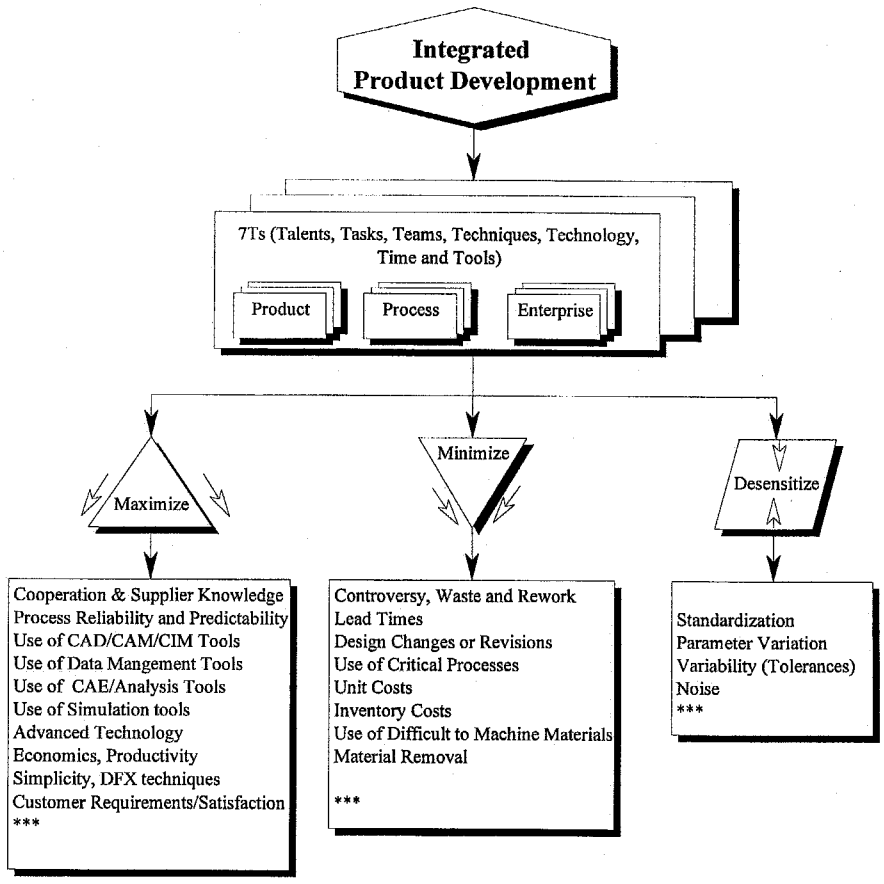


Figure 8. Definition of a global DIPD solution.

lation, the validity of the assumptions, the credibility of MOEs and other criteria, the mechanics of analysis, and the reasonableness of decisions. Here, formulation goes beyond its mathematical sense to modeling of the elements of design variables, constraints, and objectives, as work-groups move from one stage of product realization to the other. One of the important consideration in product realization process is output modeling. Output modeling consists of determining what outputs are important for the problem at hand and how to accurately represent them in quantifiable terms. Performance modeling is one type of output modeling. Performance modeling provides the transformation from specifications to the outputs that are performance-based. One way to represent a performance-based transformation model is to use a set of artificial symbol structures for the description of the incoming specifications and to use simulation or analyses to predict output behaviors or performance [Sobieszczanski-Sobieski, 1993]. In performance modeling, one may use performance metrics, (such as manufacturability, assemblability, reliability, etc.) to simulate, quantify or analyze the outcome. Analysis or simulation is not the only mechanism to capture outputs of a transformation system. Genetic algorithms are being explored with CAD systems to generate designs in a generate-and-test approach called "*conceptual interpolation*." In conceptual interpolation a number of conceptual operators provide a genetic basis for generating interpolate designs. In genetic algorithms, interpolate designs exist in generations. Within each generation, designs can mate and produce offspring according to some measure of their value characteristics (a fitness function). The conceptual operators amplify the power of design work-groups by allowing them to work at higher conceptual levels. Other techniques, such as use of fuzzy set theory, AI approach and "simulation-based generate and test" approach, are commonly used to alleviate formulation or trade-off difficulties [Saaty, 1978]. For example, if a set of fuzzy goals is modeled according to the life-cycle issues of the product, goals can be interpreted as a set of criteria. A premise of fuzzy set theory is that the overall preference of a product design alternative is represented well by aggregating the individual goals with respect to the criteria [Parsaei and Sullivan, 1993]. In this context, a variety of fuzzy set connectives can be used to form the framework on which the aggregation process of product realization goals can be based. Using this or similar methodology, future design evaluations are based on accumulated knowledge. If any of the elements of the fuzzy set or optimization model are incorrectly specified or are inappropriate, the resulting design would be incorrect, too. Thus, it is normally not sufficient to have a sound mathematical basis or to have the world's best algorithm. Formulating the transformation system and identifying a consistent "fuzzy set or optimization model" at each step of this transformation is critical to efficient product realization. A rational prediction to product realization requires building in a sound analytical or algorithmic methodology or a computer-based procedure at each step of this transformation.

Finding a product design or a concept that satisfies all the constraints is possible only when the constraint network represents all design alternatives, is complete and consistent, and results in a unique solution. These conditions are rarely, if ever, met. If the constraint network is over-constrained, no solution exists, and some constraints must be relaxed or a portion of the goals or objectives, which are conflicting, must be modified. If the network is underconstrained, many possible solutions exist, and additional constraints or goals must be added so that the resulting design or concept converges to some interesting domain.

2.6. Integrated Problem Formulation

Integrated problem formulation commonly involves two steps: understanding the structure [Steward, 1981] and intent of the problem and understanding the solution of the problem. In Prasad [1996], a taxonomy for product realization was established. A taxonomy plan driven by a complex knowledge-base can be used to carry out the above two steps. There is a whole body of work in decision-based design that is based on Von Neumann–Morgenstern/Arrow (vN-M/A) framework, which has axiomatic underpinnings [Hazelrigg, 1999, 1998]. Theses could also be part of this taxonomy plan. A taxonomy plan commonly consists of a typical choice of product realization loops (from all possible loop-types, like 1-D, 2-D, or 3-D tracks) [Prasad, 1996]. Taxonomy plan is commonly used as a part of a product realization process that transforms a set of raw product specifications to a physical part (product). Part of identification of criteria is determined through the use of specifications. The inputs represent the source of information that is specified at every step during product-enrichment. Another major source of criteria is a repertoire of CE measures of effectiveness (MOEs), such as simplicity, use of standard parts, reduction of materials, etc. PD^3 processes appear to be iterative at all life-cycle stages. The best results are obtained by generating, evaluating, and optimizing several alternative options at each stage of the PD^3 process. However, there appear to be numerous approaches to problem formulation and problem solution [Browning, 1999; Clausing, 1994]. Investigations to test the efficacy of various approaches have been done on embodiment, configuration, and structural (FEA/FEM) problem for-

mulation using knowledge-based engineering (KBE), or smart models [Navinchandra, Fox, and Gardner, 1993]. However, little progress has been reported on types of solution techniques or modeling approaches. The coupling of FEA/FEM with optimization has been found useful for structural design problems [Prasad, 1985]. The coupling of enterprise modeling, simulation, and statistical analyses (mean values and statistical distributions, etc.) has been found useful for studying the organizational structures [Steward, 1981] and for verifying the dynamic interactions of processes. For configuration and design embodiment cases [Pugh, 1991], bond graph based techniques have been useful to represent and connect functional requirements. For process design, Taguchi's method is most common to minimize variations (design for robustness) and assign tolerance limits [Clausing, 1994].

2.6.1. Method of Solution

Problem solving depends not only on inferences (decision-making ontology, such as algorithmic or heuristics types) but also on methods. Most numerical computational tools contain inferences that are algorithmic types. In such tools, decision-making is supported by some sensitivity or mathematical programming techniques, such as continuous or discrete optimization. Artificial intelligence, on the other hand, emphasizes symbolic processing and nonalgorithmic inferences, such as rule-based, model-based, or case-based reasoning. Most expert systems are heuristic-types. Real design problems require the use of both algorithmic and heuristic inferences. Combining an expert system approach (e.g., knowledge-based) with algorithmic inference (such as parametric, variational, or feature-based CAD tools) can make these CAD tools more powerful for solving complex product realization problems. A problem solving method is a form of representation, through which the method describes the dynamic knowledge that is applied to a static knowledge ontology. In Prasad [1997], two types of problem solving methods (in the form of constraint-based and knowledge-based) were discussed. Typical types of expert softwares that result from using methods based on knowledge-based representation are diagnostics, designers, planners, configurers, classifiers, assistants, etc. Problem solving methods are therefore useful in linking knowledge acquisition to problem-solving algorithms such as types of solutions for constraint-satisfaction problems (CSP).

Knowledge acquisition is meant here to indicate some sort of template that is to be filled in during knowledge modeling.

2.7. Collaboration and Cross-Functional Problem Solving

The problem solving is a process of finding a set of alternative solutions by applying the cross-functional teams and collaboratively improving the quality of decisions. Some simple methods of finding solutions are reasoning, calculations, table look-up, graphics, as well as different types of knowledge representation—heuristics, algorithmic, etc. Before we outline any strategy for solving problems, let us examine the cross-functional contributions. Most human beings, like any natural thing, choose a path of least resistance. Humans have a limited ability to entertain more than a few ideas at a time [Newell and Simon, 1972]. For example, if several alternatives exist for satisfying a constraint, most people "will satisfy rather than optimize." This observation in mechanical design is well noted by many [Stauffer and Slaughterbeck-Hyde, 1989; Newell and Simon, 1972]. H.A. Simon was the first to coin the term "satisficing" to describe this phenomenon. Pearl described "satisficing" as "discovering any qualified object with as little search effort as possible [Pearl, 1984]." Stauffer describes it as reducing the time and effort in searching for a solution by utilizing the first acceptable solution rather than searching for the best or optimal solution [Stauffer and Slaughterbeck-Hyde, 1989].

When faced with a number of alternatives, designers often employ a simplifying—elimination by aspect—strategy, choosing only a few critical attributes that is the most heavily weighted [Steward, 1981]. The above are some examples of ways a human mind reacts [Thurston, 1991]. Instead of fighting with these cognitive limitations of human beings, the strategy ought to minimize its effect. A taxonomy commonly employed in product realization [Prasad, 1996] represents such an strategy. Taxonomy streamlines the total PD^3 process into a limited number of loops [Prasad, 1996]. If the problem is classified in such a way that the entire problem is built from its constituents, then the problem reduces to applying "elimination by aspect" to each constituent-built, giving rise to "satisfaction by taxonomy." By consecutive use of "elimination by aspect," along with problem solving methodologies (see Fig. 8), cross-functional teams are more likely to converge to a "near-optimal" solution. Depending upon the situation and the team focus, a number of alternatives may exist for problem formulation. Teams may use an analytical technique, such as a formal optimization, or a nonlinear programming techniques, if the problem is well posed and can be quantified. If the tasks involve non-numerical or nonalgorithmic information, expert system techniques can be used to represent domain expertise in terms of heuristics [Thurston, 1991; Prasad, 1984]. For

other problems, requiring both heuristics and algorithmic aspects, the two methods can be combined [Prasad, 1985]. The selection of the redesign method depends on:

- ***Modeling Source:*** The types of individual models chosen for a loop evaluation and their potential for relative impact on improving the product performance, such as product cost.
- ***Time of Each Iteration:*** How long it takes to perform each iteration for a single configuration and how many configurations ought to be considered. The number of times a nested iterative subloop evaluation is performed depends on the chosen algorithm and its convergence criteria.
- ***Number of Iterations:*** How many evaluations are performed per iteration and how many iterations are required for convergence. If the same "redesign method" is repeated more than twice in a row to the same subloop, this may be a case of possible floundering in its strategy.

Alternatively, a work-group may use a line of experimentation techniques if the problem is difficult to quantify. Many approaches for problem formulation are contained in this paper. The solution techniques and tools are discussed later in this paper.

2.8. Continuous Monitoring and Knowledge Upgrade

In the world of global manufacturing a competitive advantage is often short-lived. As soon as a management becomes satisfied with the results of what process models have achieved, the company may again start losing ground to competitors. Competitors continually strive to improve their position in the world marketplace. In order to stay competitive, these process models must have a provision for upgrading knowledge as the process matures. An example of a typical process model for Continuous Knowledge Monitoring and Upgrade, CMKU, is shown in Figure 9:

$$\text{CMKU} = f[\text{Monitor, Select, Analyze, Contain, Correct, and Prevent}], \quad (10)$$

where f denotes the function. The CMKU process consists of the following six steps:

- ***Monitor:*** During monitoring if discrepancies are noted, the suspected work-groups are interviewed and "as-is" information is gathered.
- ***Select:*** Here a portion of the as-is process is identified and the information is flow-charted.
- ***Analyze:*** From the information flow charts, opportunities or bottlenecks are identified.

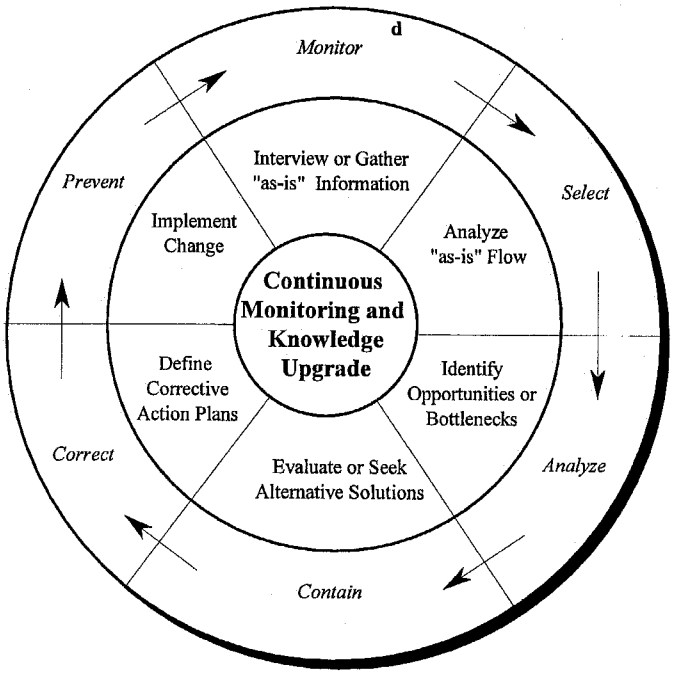


Figure 9. A typical process model for continuous monitoring and knowledge upgrade.

- ***Contain:*** Alternative solutions are sought or evaluated to contain the problem.
- ***Correct:*** A corrective action plan is determined and the work-group's cooperation is sought.
- ***Prevent:*** Changes are introduced in the process to prevent reoccurrence of the same or similar problems, concepts, or designs.

Normally during a process monitoring, there are two types of changes involved: scheduled changes (also referred as revisions) and unscheduled changes. Some changes may produce far-reaching influences. Revisions introduced into the way a product definition is performed, whether in the traditional sense or in the Concurrent Engineering mode, or even somewhere in between is painstaking. As a company migrates from one life-cycle aspect to the other, it may induce unscheduled changes in other functions in the company. Through this process of continuous monitoring and

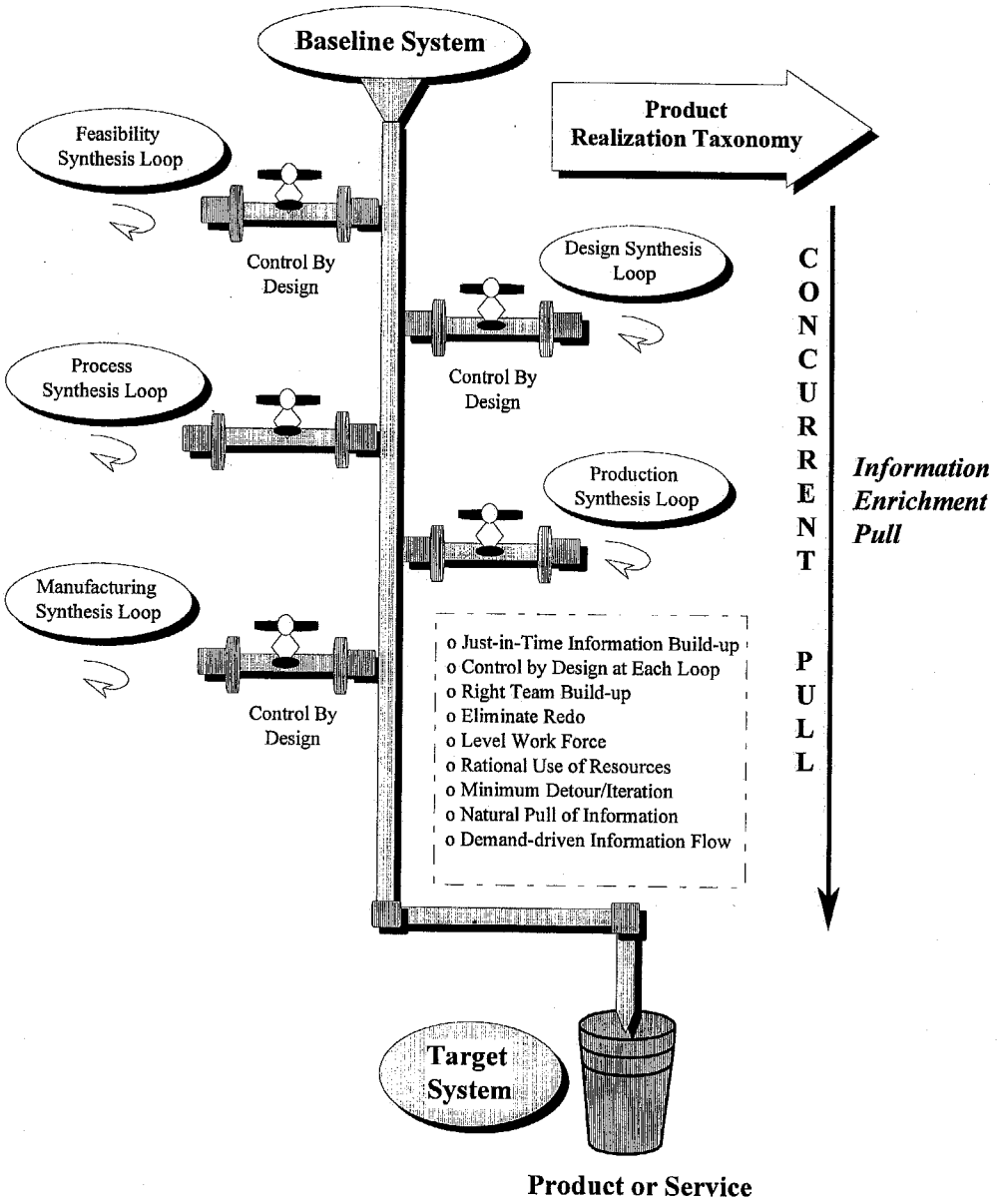


Figure 10. An analogy for a concurrent DIPD process.

upgrading it is possible to prevent some of these unscheduled changes.

3. CONCURRENT DIPD METHODOLOGY

In Figure 1, "a fluid flow through a pipe analogy" was shown for a serial process. The same analogy has been redesigned now in Figure 10 for a concurrent PD^3 process. Instead of placing the control at the end of the pipe assembly as in serial engineering, the control is now placed at each loop level of the concurrent DIPD process. This is called *control by design* also commonly called a *pull system* at each loop level. While there is a quite a bit of literature on the merits of pull system for manufacturing processes [Clausing, 1994; Eppinger, Whitney, Smith, and Gebala, 1994], this is perhaps the first attempt to show that a pull system is also good for a problem solving and information creating endeavor such as product design and development.

The configuration of the pipe connections and their relative positions along the vertical direction could be governed by a taxonomy of product realization [Prasad, 1996]. Even though pull system represented by Figure 10 seems to exhibit many of the good characteristics of product realization, it is not obvious that design processes would always be best run as pull systems. The numerous bends and elbow-connections have been replaced in Figure 10 by loops tapped in at designated points (governed by the corresponding taxonomy) along the vertical tube. The fluid pressure in the pipe would thus be created naturally due to the gravitational force. In concurrent DIPD process, this is equivalent to "just-in-time" information build-up for each loop. Five loops are, therefore, shown running concurrently in Figure 10. The amount of information buildup at each loop would thus be governed by a natural pull of the information rather than a force "push" found in serial engineering. Even though pull system seems to exhibit ample advantages over push system in a product realization process, in general, there are still some issues that remain unresolved. It is not clear what exactly a pull system would mean in product design and development. How is it different from a push system in common practice? What creates a pull in specific design situations?

4. CONCLUDING REMARKS

The paper describes a DIPD process, used at Electronic Data Systems (EDS), General Motors Account, of going from a set of incomplete and inconsistent functional requirements to realizing a physical product. This methodology is founded on eight building blocks. The first building block of DIPD is determination of "product requirements planning and management." The second building block is "work structuring and CE team deployment." Structuring of work facilitates the integration of complementary engineering expertise. The third building block, "methodology systematization," outlines the basic conceptual framework for DIPD methodology. The fourth building block, "product and process systematization," lays down the taxonomy of product and process transformation leading to a physical artifact. The remaining four building blocks outline a methodology for arriving at an optimized design or a consensus-based alternative. In the sixth block, "an integrated problem formulation approach," the methodology determines what constraints are violated, in addition to giving an alternate set of optimized solutions. In the seventh block, consensus-based approach, the methodology combines different opinions, which may or may not be analytically based. The last building block, continuous monitoring and knowledge upgrade, emphasizes the need for continuous improvement in PPO.

The four-stage DIPD methodology can be applied to any new product introduction, or to any problem set having a deviation from its initial specifications. This can also be used to tackle a continuous process improvement opportunity. There are many advantages associated with this proposed concurrent DIPD methodology. The methodology recognizes that PPO data in the early stage of product development is fuzzy, incomplete, and often uncertain [Wesner, Hiatt, and Trimble, 1994]. Concurrent DIPD provides a taxonomy-based concurrent engineering process to sort through this fuzzy set of information to establish rationally what will work and what will not. The eight-step DIPD methodology balances the needed reduction in responsiveness (with respect to time-to-market) against the risk of PPO design changes by using incomplete or uncertain information upfront in a taxonomy-supported PD^3 process. DIPD methodology, thus, provides an integral mechanism to manage the risk appropriately at each step. In comparison to serial process for product development, concurrent DIPD has the potential to eliminate excessive redo, minimize detour and iterations, and work with a leveled work force. The DIPD methodology is based on rationally utilizing the available resources and demand driven is the main accessing mechanism for pulling information. The work presented outlines a building blocks for realizing products. Due to publication restrictions, author could not include some useful industrial examples. From the publication of this paper, it is hoped that others would try out these concepts and approaches advocated in the paper and report findings in future research and publications.

REFERENCES

P.J. Agrell, Multicriteria approach to concurrent engineering, Int J Prod Econ 34(1) (1994), 99–113.

S. Berger, et al., Towards a new industrial America, Sci Am (June 1989), 39–47.

T.R. Browning, Designing system development projects for organizational integration, Syst Eng 2(4) (1999), 217–225.

B. Chandrasekaran, A framework for design problem solving, Res Eng Des 1(2) (1989), 75–86.

K.B. Clark and T. Fujimoto, Product development performance: Strategy, organization, and management in the world auto industry, Harvard Business School Press, Boston, 1991.

D. Clausing, Total quality development: A step by step guide to world-class concurrent engineering, ASME Press, New York, 1994.

S. Dowlatshahi, Product design in a concurrent engineering environment: An optimization approach, Int J Prod Res 30(8) (1992), 1803–1818.

S.D. Eppinger, D.E. Whitney, R.P. Smith, and D.A. Gabala, A model-based method for organizing tasks in product development, Res Eng Des 6 (1994), 1–13.

G. Hazelrigg, A framework for decision-based engineering design, J Mech Des 120 (December 1998), 653–658.

G. Hazelrigg, An axiomatic framework for engineering design, J Mech Des 121 (September 1999), 342–347.

R.L. Keeny and H. Raiffa, Decisions with multiple objectives, Wiley, Chichester, 1976.

P.N. Koch, J.D. Peplinski, F. Mistree, and J.K. Allen, "Configuring systems using available assets: A conceptual decision-based perspective," Mechanical design theory and methodology, M. Waldron (Editor), Springer-Verlag, New York, 1996, pp. 127–160.

A. Kusiak and J. Wang, Decomposition of the design process, J Mech Des 115 (1993), 687–695.

K. Lewis and F. Mistree, Collaborative sequential and isolated decisions in design, ASME J Mech Des 120 (1998), 643–652.

C.A. Mcmohan, M. Xianyi, K.N. Brown, and J.H.S. Williams, A parallel multi-attribute transformation model of design, Proc Des Eng Tech Conf, September 17–20, 1995, Boston, MA, DE-Vol. 83, ASME Press, New York, 1995, Vol. 2, pp. 341–350.

D. Navinchandra, M.S. Fox, and E.S. Gardner, "Constraint management in design fusion", Concurrent engineering: Methodology and applications, P. Gu and A. Kusiak (Editors), Elsevier, Amsterdam, 1993, pp. 1–30.

A. Newell, and H.A. Simon, Human problem solving, Prentice-Hall, Englewood Cliffs, NJ, 1972.

J. Pearl, Heuristics: Intelligent search strategies for computer problem solving, Addison-Wesley, New York, 1984.

G. Pahl and W. Beitz, 1991, Engineering design: A systematic approach, K. Wallace (Editor), Springer-Verlag, New York, 1991.

H.R. Parsaei and W.G. Sullivan (Editors), Concurrent engineering: Concurrent issues and modern design tools, Chapman & Hall, New York, 1993, pp. 207–230.

B. Prasad, An integrated system for optimal structural synthesis and remodeling, Comput Struct 20(5) (1985), 827–839.

B. Prasad, Novel concepts for constraint treatments and approximations in efficient structural synthesis, AIAA J 22(7) (1984a), 957–966.

B. Prasad, Explicit constraint approximation forms in structural optimization—Part 2: Numerical experiences, Comput Methods Appl Mech Eng 46(1) (1984b), 15–38.

B. Prasad, Concurrent engineering fundamentals, Volume I: Integrated product and process organization, PTR Prentice Hall, Upper Saddle River, NJ, 1996.

B. Prasad, Concurrent engineering fundamentals, Volume II: Integrated product development, PTR Prentice Hall, Upper Saddle River, NJ, 1997.

B. Prasad and J.F. Emerson, Optimal structural remodeling of multi-objective systems, Comput Struct 18(4) (1984), 619–628.

B. Prasad and C.L. Magee, "Application of optimization techniques to vehicle design—a review," Recent experiences in multidisciplinary analysis and optimization, NASA CP 2327, NASA Langley Research Center, VA, April 1984, Part 1, pp. 147–171.

S. Pugh, Total design—integrated methods for successful product engineering, Addison-Wesley, Wokingham, UK, 1991.

T.L. Saaty, Exploring the interface between hierarchies, multiple objectives and fuzzy sets, Fuzzy Sets Syst 1 (1978), 57–68.

L.A. Stauffer and R.A. Slaughterbeck-Hyde, The nature of constraints and their effect on quality and satisficing, 1st Int Conf Des Theory Methodology, September 17–21, DE-Vol. 17, ASME, New York, 1989, pp. 1–7.

M. Stefik, Planning with constraints (MOLGEN: Part I and Part II), Artif Intell 16 (1981), 111–169.

D.V. Steward, The design structure system: A method to managing the design of complex systems, IEEE Trans Eng Manage 28(3) (1981), 71–74.

D.W. Steward, Systems analysis and management: Structure, strategy, and design, PBI, New York, 1981.

J. Sobieszczanski-Sobieski, Multidisciplinary design optimization: An emerging new engineering discipline, Proc World Cong Optimal Des Struct Syst, Rio de Janeiro, August 2–6, 1993.

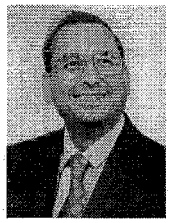
D.L. Thurston, A formal method for subjective design evaluation with multiple attributes, Res Eng Des 3 (1991), 105–122.

D.L. Thurston and A. Locascio, "Multivariable design optimization and concurrent engineering," Parsaei, Hamid R., and W.G. Sullivan, ed. Concurrent engineering: Contemporary issues and modern design tools, H.R. Parsaei and W.G. Sullivan (Editors), Chapman & Hall, New York, 1993, pp. 207–230.

K.T. Ulrich and S.D. Eppinger, Product design and development, McGraw-Hill, New York, 1994.

J. Warfield and J.D. Hill, 1972, A unified systems engineering concept, Battle Monograph, B.B. Gordon (Editor), No. 1, June 1972.

Wesner, J.W., J.M. Hiatt, and D.C. Trimble, Winning with quality: Applying quality principles in product development, Addison-Wesley, Reading, MA, 1994.



Biren (Brian) Prasad is the Visiting Professor at California Institute of Technology, Pasadena, CA. He served as the Director of University of California Extension, Engineering, Information Technology and Sciences Unit in Irvine, CA until 2001. Prior to that he was director of Knowledge-Based Engineering Product Business Unit at Unigraphics Solutions (UGS) in California, USA. Before joining UGS in 1998, he was the Principal Consultant and Director of Concurrent Engineering Services at Electronic Data Systems (EDS) (an ex-subsidiary of General Motors), where he was in charge of Automated Concurrent Engineering consulting Group since 1985. He has written or co-authored over 100 technical publications, including 70 archival papers and a dozen books. He wrote a new textbook on *Concurrent Engineering Fundamentals*—a two volume set—published by Prentice-Hall. The textbook is followed in many universities. His edited text is *Modern Manufacturing: Information Management and Control* (1994), published by Springer-Verlag. In 1989, he edited a set of three-volume book entitled *CAD/CAM, Robotics and Factories of the Future*, 1989, Springer-Verlag. He has served as editors for several additional texts, monographs, and proceedings. He supports professional societies in several editorials and organization roles. He has received three awards: AIAA's Survey Paper Citation Plaque & Award (1982), a NASA Award and a Certificate for Creative Development of a Technical Innovation on *"PARS"—Programs for Analysis and Resizing of Structures* (1981), and the ABI (American Biographical Institute) Commemorative Medal of Honor (1987). Dr. Prasad is the Managing Editor for the *International Journal of Concurrent Engineering: Research & Applications* (CERA). Dr. Prasad earned his Ph.D. from Illinois Institute of Technology, Chicago, a Degree of Engineer from Stanford University, California. He received an M.S. degree from Indian Institute of Technology, Kanpur and a B.E. Degree from Bihar College of Engineering, Patna, both from India. Dr. Prasad has served on numerous committees for ASCE, ASME, and SAE. He is the Chairman of the SAE's Computer Readers' Committee. His professional honors include: Associate Fellow of AIAA, Fellow of ASME, ASCE, SAE, AAAI, and Fellow and Life Member of ISPE.