Systems Research and Behavioral Science
Syst. Res. **18**, 523–543 (2001)
DOI: 10.1002/sres.354

■ *Research Paper*

Product, Process and Methodology Systematization to Handle Structural and Computational Complexity in Product Realization

Biren Prasad*

Knowledge-based Engineering, Unigraphics Solutions, CERA Institute, West Bloomfield, Michigan, USA

This paper establishes a concept for Product, Process, and Methodology (PPM) systematization to handle both structural complexity (often referred as *hard complexity*) and computational complexity (also referred as *soft complexity*) in product realization. A systematization methodology is proposed to handle both these complexities through a four-stage process: Planning, Systematization, Solution and Unification. **Planning** is the first stage where system specification is defined. **Systematization** is a systematic decomposition of the problem into a set of discrete sub-problems. There are three kinds of systematization that can be employed in product realization: Methodology systematization, Product systematization and Process systematization. During the **Solution stage**, a number of computational alternatives, solution strategies or topological options for handling *soft complexity* are obtained for each sub-problem, and the best solution is selected. **Unification** is an aggregation or a reconstruction of an overall product solution from various alternative solutions to the sub-problems.

A branching and bounding methodology has been employed here first to branch the complex product and process into *'sub-domains' and 'loops'* and later bound these *loops* back into *sub-domains* and then further bound those *sub-domains* back into an integrated product development system. The branching and bounding methodology provides the product development teams the ability to refine successively the 'goodness or fitness' of a baseline product concept as teams proceed from one nested loop to another. PPM systematization is a convenient way of organizing the process necessary for a new product realization or for developing future product upgrades. Copyright © 2001 John Wiley & Sons, Ltd.

Keywords product systematization; product and process realization; structural complexity; computational complexity; concurrent engineering; planning; optimization; solution

*Correspondence to: B. Prasad, Knowledge-based Engineering, Unigraphics Solutions, CERA Institute, PO Box 250254, West Bloomfield, MI 48325-0254, USA.

Received 24 February 1998
Accepted 21 December 1999

INTRODUCTION

The product environment in modern manufacturing is very complex. It consists of many components of products, processes, and services (Fleischer and Liker, 1997), including information technology (IT) services (hardware and software) (Bajgoric, 1997). The design of an automobile, for example, involves 2000–3000 parts, and calls for thousands of designers and engineers making millions of design decisions over its life cycle (Clark and Fujimoto, 1991). None of these parts are designed and developed in isolation from each other. The combination of new and old practices, such as old-fashioned habits, new life cycle environment, human factors, ergonomics, environmental changes, and mounting safety and international regulations, has also increased the complexity of product development efforts (Kamath and Liker, 1994). The complexity, as defined by Galbraith (1973), results from five main sources. They are inherent product complexity (Krishnan, 1993), process complexity (Magrab, 1997), organizational complexity (Checkland, 1981) (such as team cooperation and communication), computer and network complexity (Bajgoric, 1997), and a maze of specification complexity (Dong, 1995) including international regulations and safety (Fleischer and Liker, 1997).

Over the past several years the diversity, variety and complexity of New Product Introduction (NPI) have grown manifold—from 'very simple' to 'very complex', while at the same time the time-to-market aspect has shrunk (Prasad, 1996). The changing market conditions (such as global manufacturing, global economy and new technological innovation) (Clark and Fujimoto, 1991) and international competitiveness (Kamath and Liker, 1994) are making the time-to-market a fast shrinking target. Today, an automobile, with complexity several times higher than before, can be manufactured in less time (often less than three years) (Clark and Fujimoto, 1991). The same product, about half a decade ago, used to take over five years to bring into the marketplace, whereas, the complexity of such products 10 years ago—by today's standard—could be characterized only as 'very simple' (Kamath and Liker, 1994). The hardware workstation market is another good example. With new innovation in chip technology, workstation computer companies have continually shortened the time between new product introductions. In 1985, when a new computer central processing unit (CPU) was introduced, it was quite innovative—but was nowhere close to today's standard in complexity. Every 18 months thereafter, a new CPU twice as complex was introduced, with twice the performance at roughly half the price. In 1988, four-times complex and four-times faster CPU was introduced at a quarter of the price in a 12-month period. In 1990, the development cycle for a new 16-times faster CPU was introduced in only a six-month time span, nearly at 1/16 of its 1985 price (Fleischer and Liker, 1997). Such trends go on. The average development time for a compact disc (CD) player, today, is nine months; that for a personal computer is 14 months (Parsaei and Sullivan, 1993). Amongst the web of such complexities, it is easy to overlook that customer requirements are also constantly changing. The customer is also becoming more sophisticated. Each time a company fulfills the customer's requirements in a product, the level of customers' expectations also moves up a notch (Clausing, 1994). They demand customized products more closely targeted to their personal, social and cultural tastes (Tsuda, 1995). Checkland (1984a) identifies three types of situations for *hard complexity problems*. A type 1 situation is characterized by interconnections, which are part of the regularities of the universe. Decomposition of the product, process and work and their breakdown structures are examples of type 1 complexity. A type 2 situation is characterized by interconnection logic, which derives from the functions they perform. A type 3 situation is dominated by the meanings attributed to their (product, process and work) perceptions by autonomous observers. Global manufacturing, global economy and international competitiveness are examples of a type 3 situation. There is also an example of soft complexity (Checkland, 1984b, p. 65) in product realization since a group of problem solvers in concurrent engineering also generate soft complexity (Ho and Sculli, 1995). The contribution made in that paper for managing complexity is grounded on both *hard and soft complexity*. An open system view of organization (Checkland, 1981 and 1984a) recognizes that any type of complexity (for example, product, process, enterprise, IT services (Bajgoric, 1997), or a set of specifications) is a system, which has elements (both hard and soft complexity types) that influence each other, and the set in the complexity influences and is influenced by the broader environmental context (Checkland, 1984b).

Figure 1 compares the process of product design, development and delivery (PD^3) with a process of fluid flow analogy through a maze of pipes (Prasad, 1996). Each pipe of an assembly represents a part or an information build-up activity in a conventional PD^3 process. A serial engineering process involves a number of connected parts or repeated activities of an assembly, such as plan, redo, download, upload, iteration, retrieve, store, which must be performed in the proper sequence. The '*fluid*' flowing through the pipes denotes '*information*' flow of a PD^3 process. The '*fluid pressure*' is equivalent to need for '*information build-up*.' Straight pipes represent the activities or parts to be designed. The '*cross-section*' of each pipe represents the corresponding '*design parameters*.' A typical conventional decision-making step is illustrated 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

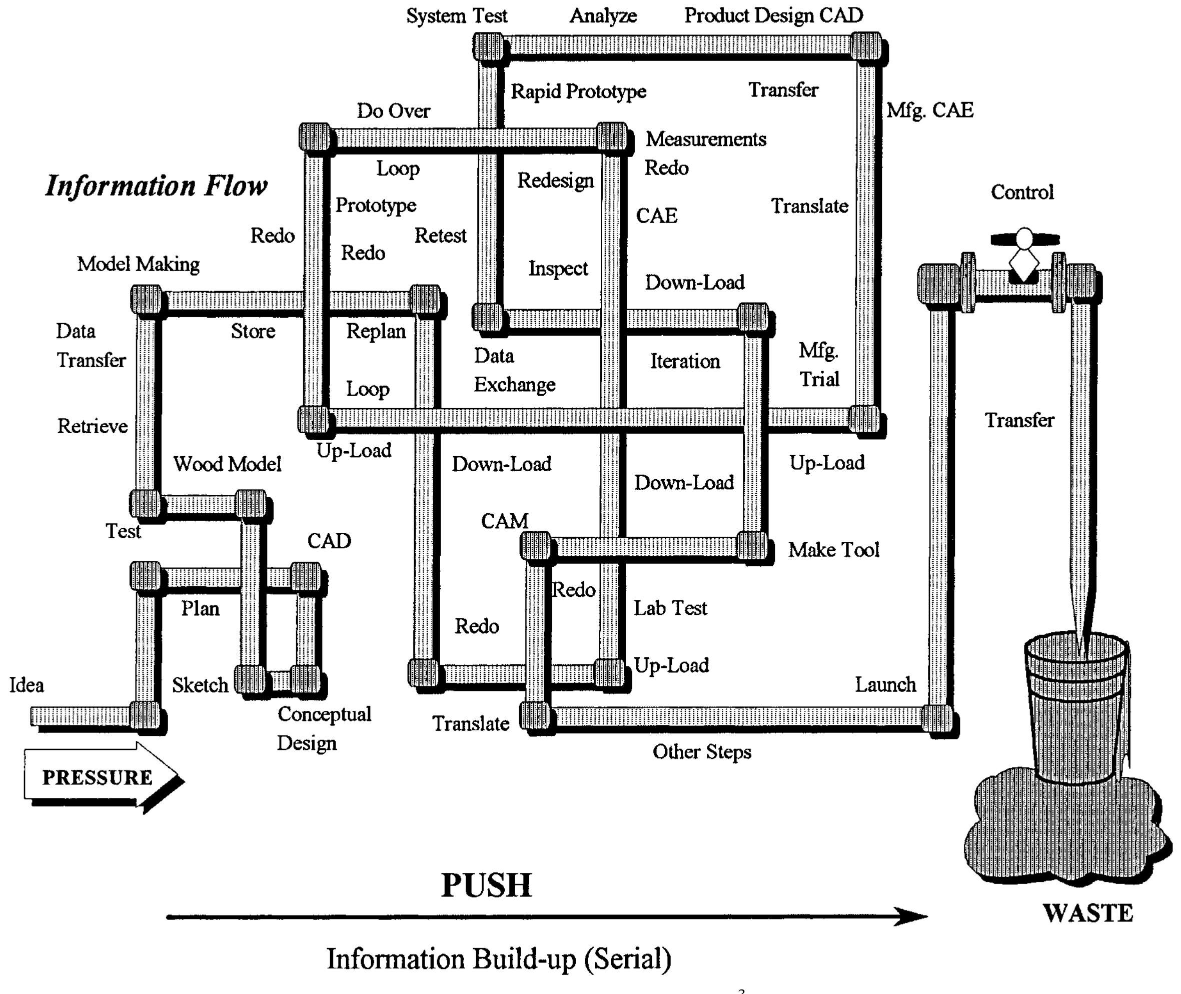

Figure 1. An analogy for a serial PD^3 process

for subsequent information build-up. The length of each pipe in the assembly denotes the time it takes to complete or build the necessary information for the next serial step of a PD^3 process. Each design decision is a trade-off affecting many other design parameters. Such a traditional breakdown of design tasks, even though it resembles a hierarchical pattern, is repetitive and inefficient (Kusiak and Wang, 1993). Decision-making in the conventional process therefore can be very difficult and total lead time could be very large considering the magnitude, types, and complexity of the products and processes (Checkland, 1984a) that need to be addressed (Ulrich and Eppinger, 1994). These complexities are often compounded by the presence of the following factors (Suri, 1988):

- *Large interconnected components*: There is a high stake on decisions that must be made simultaneously. In modern manufacturing, where materials, parts and information all move rapidly through the plant (Hardwick *et al.*, 1990), a small change (say a material change) at the design end of a PD^3 process can have a significant impact on the production end. This is likely whether or not parts are stamped, machined or injection moulded. Most changes, static or dynamic, must be managed in real time (Stark, 1992):

$$\frac{\Delta\ \text{Production}}{\Delta\text{Design}} \Rightarrow \text{Large (a factor of 100 or more)} \tag{1}$$

- *Limited resources*: Most modern manufacturers have downsized their resources (7*T*s, namely talents, tasks, teamwork, techniques, technology, time, tools (Prasad, 1996), as shown in Figure 2) to a bare minimum (Clark and Fujimoto, 1991). Resources are shared amongst the internal and external workgroups to contain costs. Repetitive demand of shared resources increases the burden of managing them efficiently:

$$\{T\} \leqslant \{T_{\max}\} \tag{2}$$

Where the set $\{T\} \equiv$ [talents, tasks, teamwork, techniques, technology, time, tools] and $\{T_{\max}\}$ is the allowable stretch of $\{T\}$.
- *Geographical distributions*: Manufacturing is global; it is distributed over a network of vast geographical areas. For example, a part may be designed in Detroit, manufactured in

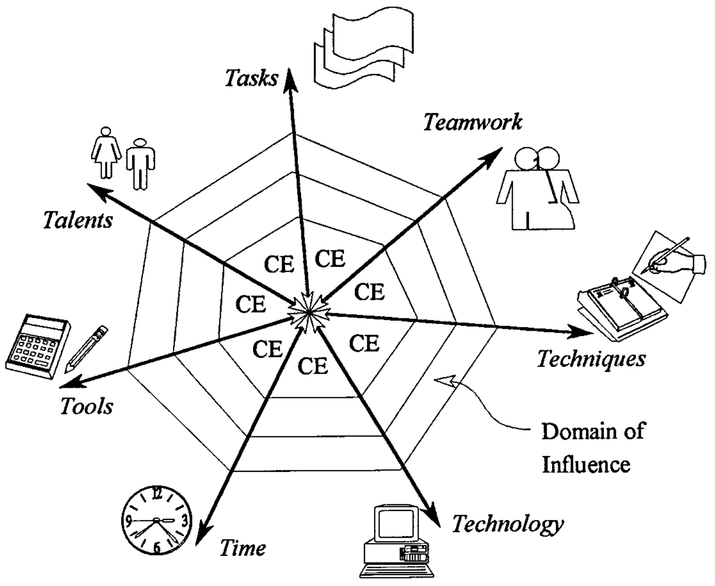

Figure 2. 7Ts: seven influencing agents of CE

Kentucky, and assembled in Korea or Mexico. Thus, the costs of travel, transportation, relocation, communication, currency exchange and labour agreements are some of the additional parameters that are factored into the cost equation (Daft, 1995):

$$\text{Cost} = f\ [\text{Travel, Transportation, Relocation, Communication}, \ldots, \text{etc.}] \quad (3)$$

- *Many goals and objectives*: In most large companies, there are sets of independent goals and objectives, F_{ui}, developed by each independent department or *u*nit. Not all of these goals and objectives are in agreement with the enterprise goals, mission statements or its vision, F_{ej}, assuming the latter exists (Hales, 1993). There is often no constancy of purpose between these independently specified goals. The situation gets worse if there is more than one strategic business unit (SBU), each having its own set of independent vision or mission statements. If we define

$$\text{Goals that may be in conflict} = \cup\,(F_{ui},\ F_{ej}) \quad (4)$$

where u stands for a unit, and e stands for an enterprise. The symbol ∪ means 'union', then goals that are good candidates for constancy of purpose are those for which intersections of F_{ui} and F_{ei} are non-zero.

$$\text{Constancy-of-purpose goals} = \cap\,(F_{ui}, F_{ei}) \quad (5)$$

The symbol in Equation (5), ∩, means 'intersection'. The terms u and i in F_{ui} take a value of

$$1 \leqslant u \leqslant \text{number of SBUs}$$

and

$$1 \leqslant i \leqslant \text{number of unit goals.}$$

The subscript j in F_{ej} takes a value of

$$1 \leqslant j \leqslant \text{number of enterprise goals} \quad (6)$$

INTEGRATED PRODUCT DEVELOPMENT PROCESS

Beyond concurrency, integration of distributed PPO (product, process and organization) seeks to offer better life cycle alternatives and realization potentials (Bajgoric, 1997). A taxonomy (Prasad, 1996) establishes a methodology of systematically organizing the process necessary for new product realization (Magrab, 1997) and for developing future product upgrades (Pahl and Beitz, 1991). Although this process taxonomy is useful for formulating and decomposing the system (Kusiak and Wang, 1993), it does not take the CE workgroups to the next step, i.e., to synthesize or optimize the design system with respect to the identified constraints. Taxonomy characterizes the design system problem into a well-structured set of decomposed tasks. This converts the integrated product development (IPD) process into a topology of networks showing how tasks are interconnected (Harr *et al.*, 1993). However, the taxonomy does not show how the network of tasks will be solved. This is referred to in this paper as '*soft complexity*' issues. Well-structured tasks are amenable to a variety of solution techniques such as analysis, simulation, sensitivity, optimization, mathematical programming and other weak numerical techniques (Prasad and Emerson, 1984). There are two types of complexity in a design problem: one is structural complexity (Smith and Browne, 1993) also called '*hard complexity*' (Checkland, 1984a); the other is *computational complexity* (Brandimarte and Villa, 1995; Prasad, 1984a, 1984b), which is also referred as 'soft complexity' (Ho and Sculli, 1995). The two are not the same concept (Checkland, 1984b). *Hard or structural complexity* is resolved through problem description, interconnections (Checkland, 1984a), etc., whereas *soft or computational complexity* is resolved through problem solving (Smith and Browne, 1993). When formulating a problem, a basic trade-off must be carefully considered between the need for representing the problem faithfully and the need for formulating a computationally tractable model (Brandimarte and Villa, 1995). Models and structuring techniques help design workgroups to reduce the structural complexity of the tasks and provide a structured roadmap that accompanies problem-solving steps (Prasad, 1985). With predetermined computational models, workgroups simply enter the specifications and the model forward-solves to provide the results. This process is very similar to a spreadsheet. In synthesis or design optimization, unlike the spreadsheet, the model or work group can back-solve entering the desired result and making the synthesizer find suitable input specification values. Artificial Intelligence (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 (Stefik, 1981). 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 need basis. However, most tools require the problems to be explicitly defined or have their characteristics explicitly known (Masud and Dean, 1993).

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 non-linear function of design variables (Lootsma, 1994; Prasad and Magee, 1984). Such formulations are often of a closed type. The workgroups 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 (Nemhauser, 1994) posed is to:

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

a set of value characteristics (referred as objective functions), $\mathrm{VC}_i\ (v)$ (7)

while satisfying a set of constraint equations:

$$C_{ij}\ (v) \geqslant 0 \tag{8}$$

D is the design space in which the solution lies.

IPD FORMULATION AT A SYSTEM LEVEL

If we employ optimization as a basis for formulating the above IPD problem, the most general statements will have the following four parts:

1. *A 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 (Brandimarte and Villa, 1995):

$$[T]v = [O] \tag{9}$$

Where T stands for transformation and O stands for output. Both are characteristic matrices and 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.

$$\mathrm{VC}_i = f(v) \tag{10}$$

$$C_{ij} = g(v) \tag{11}$$

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 Figure 3)

2. *Objectives*: The objective is a function or criterion that characterizes the aspect of design to be improved. The problem of multi-criterion optimization is to minimize or maximize a set of merit functions (Prasad and Emerson, 1984) simultaneously:

$$\mathrm{VC}_i(v); \quad \text{for } i = 1, 2, \ldots q\mathrm{c}_{\max} \tag{12}$$

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 (Nemhauser, 1994).

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

$$v = [\{v_{\text{sizing}}\}, \{v_{\text{shape}}\}, \{v_{\text{topology}}\}, \{v_{\text{knowledge}}\}]^T \tag{13}$$

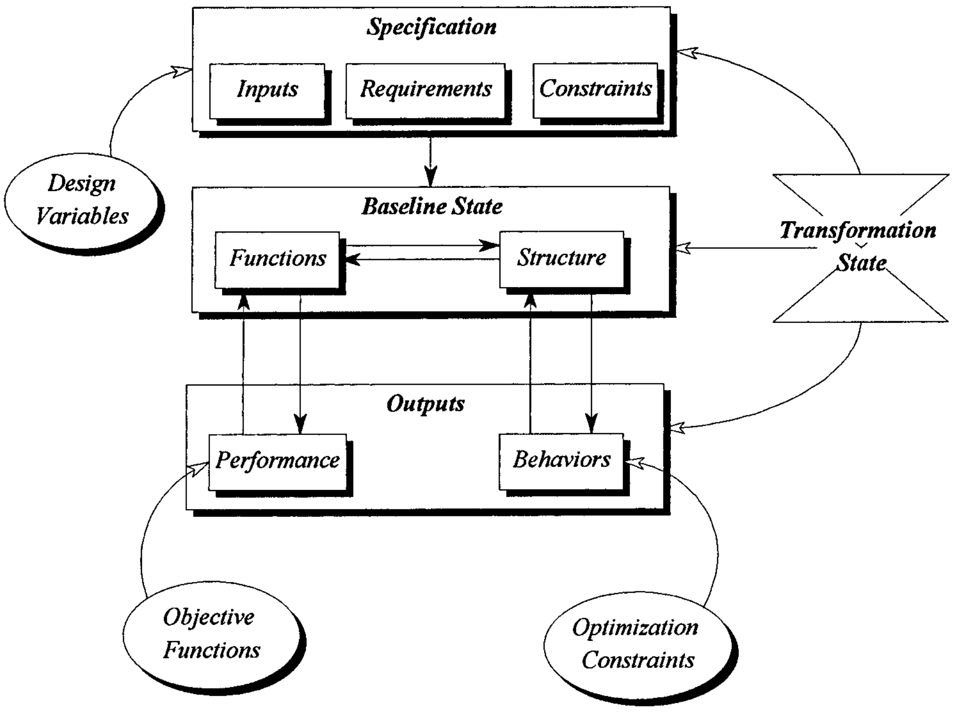

Figure 3. Relationship between transformation state attributes and optimization model variables

There are four classes of design variables commonly used (Prasad, 1996):

- $\{v_{\text{sizing}}\}$ – *sizing variables*
- $\{v_{\text{shape}}\}$ – *shape variables*
- $\{v_{\text{topology}}\}$ – *topology variables*
- $\{v_{\text{process}}\}$ – *process variables*. These involve changing the rules concerning the part's forming or processing needs, which have an effect on changing the part's size, shape, topology or functions themselves.

4. *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 (Navinchandra *et al.*, 1993):

$$C_{ij}(v) \geqslant 0; \text{ for } j = 1, 2, \ldots c_{\max} \quad (14)$$

Often such constraints also include limits on design variables:

$$\{V_{\min}\} \leqslant \{v\} \leqslant \{V_{\max}\} \quad (15)$$

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

This completes the definition of an optimization-based IPD formulation. Books on optimization seldom focus on the transformation system (Brandimarte and Villa, 1995). Impressive progress has been reported in the applications to real-world problems (Nemhauser, 1994). They assume it to be given or explicitly known. However, most physical problems cannot be modelled purely in an explicit form (Prasad and Magee, 1984). 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 finite element analysis (FEA) model. Consequently, in most cases, design trade-offs are made tacitly and implicitly. An implicit statement of a problem at a system level is shown in Figure 4. The solution of the problem is governed by (Prasad, 1996):

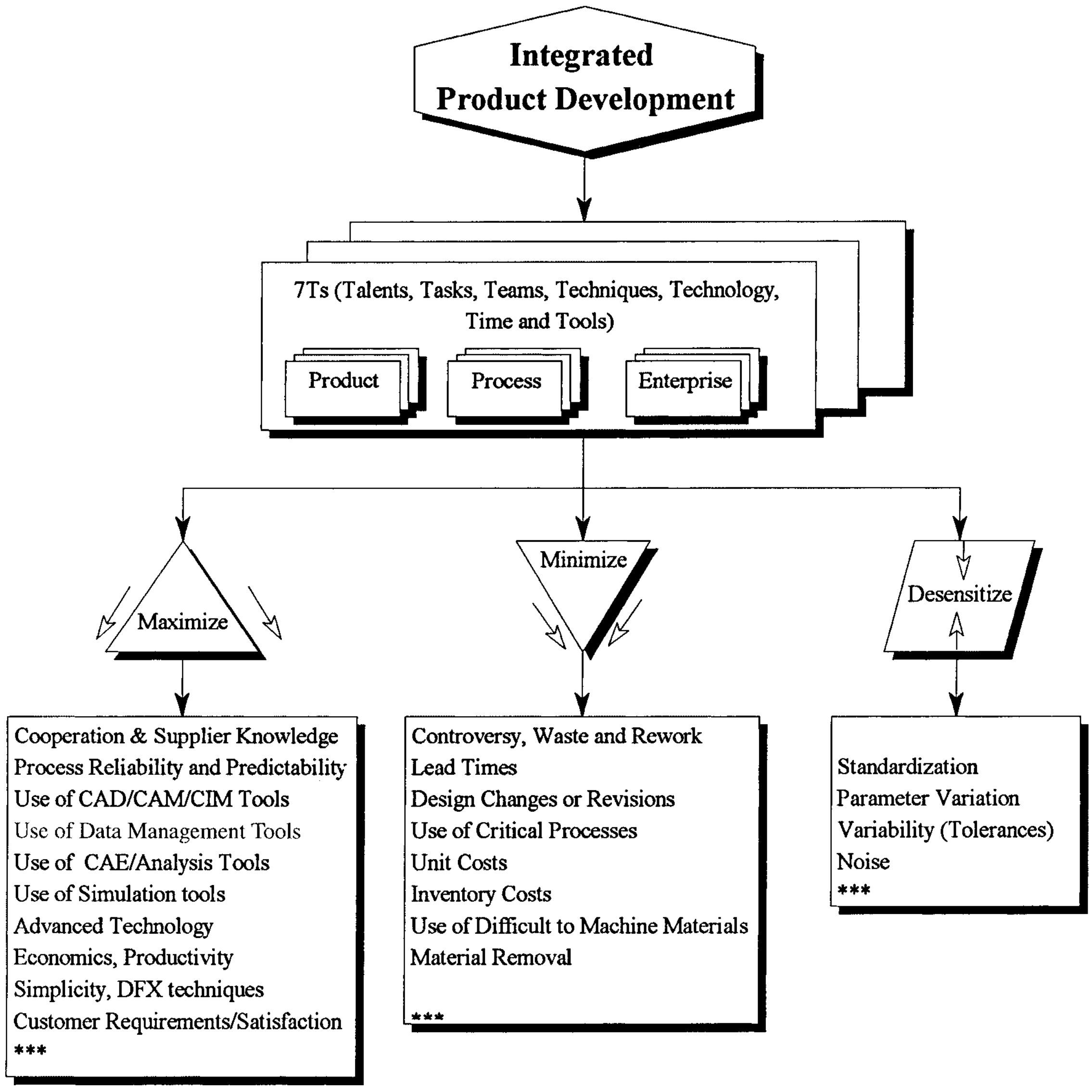
Integrated Product Development
7Ts (Talents, Tasks, Teams, Techniques, Technology, Time and Tools)
Product
Process
Enterprise
Maximize
Minimize
Desensitize
Cooperation & Supplier Knowledge
Process Reliability and Predictability
Use of CAD/CAM/CIM Tools
Use of Data Management Tools
Use of CAE/Analysis Tools
Use of Simulation tools
Advanced Technology
Economics, Productivity
Simplicity, DFX techniques
Customer Requirements/Satisfaction
***
Controversy, Waste and Rework
Lead Times
Design Changes or Revisions
Use of Critical Processes
Unit Costs
Inventory Costs
Use of Difficult to Machine Materials
Material Removal
***
Standardization
Parameter Variation
Variability (Tolerances)
Noise
***

Figure 4. Definition of a global IPD solution

(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 (Prasad, 1985). An open-ended optimization problem usually takes longer to solve in iteration than a closed-form problem (Brandimarte and Villa, 1995). Mathematical modelling of managerial problems does not produce universal laws as in natural sciences, but tentative representations of relationships which are felt to be relevant to the analysis of a given problem, in a particular period of time, and in a particular context (Lootsma, 1994). Teams must evaluate the formulation, the validity of the assumptions, the credibility of measures of merits (MOMs) and other criteria, the mechanics of analysis, and the reasonableness of decisions. Modelling is the art of selective simplification of reality (Brandimarte and Villa, 1995) Here, formulation goes beyond its mathematical sense to modelling of the elements of design variables, constraints and objectives, as workgroups move from one stage of product realization to the other. One of the important considerations in product realization process is an output modelling. Performance modelling is one type of output modelling. Analysis or simulation is not the only mechanism to capture outputs of a transformation system. Genetic algorithms are being explored with computer-aided design 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 modelled according to the life cycle issues of the product, goals can be interpreted as a set of criteria (Masud and Dean, 1993). 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 (Brandimarte and Villa, 1995). Formulating a transformation system involves finding a suitable trade-off between the need for faithfully representing the manufacturing system and the production scenario, while keeping data and computational requirements as low as possible (Brandimarte and Villa, 1995). 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. A right blend of knowledge and common sense is needed in order to build a methodology or a procedure, solve it, validate it, and successfully exploit the results. Systematization or system modelling in this paper has a different meaning than system engineering process (Hales, 1993) for project management. The systematization concepts presented here are *conceptual* in nature and aim at helping the product realization process. They are not aimed at stating universal law, or to help in understanding systems behaviour problems.

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 design solution evolves during this realization process (Zhang and Zhang, 1995). Before any methodological decomposition, it is not possible to process the design constraints of a sub-assembly or an individual part since, at that point, many of its details are unknown (Navinchandra *et al.*, 1993). 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). Group technology is an example of 'systematization'. Another important reason why designers work hierarchically is that an individual person or a workgroup is not able to process large numbers of constraints simultaneously (Hales, 1993). Systematization offers a powerful tool to reduce the inherent complexity of the problem domain (Checkland, 1984). Methodology ensures that everything possible will be done to apply the *7Ts* resources (Prasad, 1996) in the most effective manner. One such act of systematization in product design is to apply decomposition—branching the design into loops, activities and tasks (Magrab, 1997). Stefik (1981) describes further motivations for decomposing design and manufacturing activities:

- The apparent structural and computational complexity (Checkland, 1984b) of a design and manufacturing problem is often reduced as a result of decomposition (Kusiak and Wang, 1993).
- 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 (Ulrich and Eppinger, 1994).
- The 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 (Stark,1992). Each member of the workgroups can be assigned to work on each decomposed set concurrently (Zhang and Zhang, 1995).
- This enhances concurrency of the product realization process (Magrab, 1997).

There are many levels of abstractions in systematization. In problems as complex as IPD, systematization starts with product/process management (Harr *et al.*, 1993). Previously, 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 (Prasad, 1996). In general, the solution to problems of this type can be approached by a four-stage systematization process as shown in Figures 5 and 6:

- *Stage 1. Planning*: This is the stage when system specification is defined. Partial ordering of intermediate goals is identified and a view to early determination of gross features is conceived.
- *Stage 2. Systematization*: There are two kinds of systematization: methodology systematization and product and process systematization. Systematization is the
  - systematic decomposition of the problem into discrete sub-problems;
  - 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
  - aggregation of the discrete constituents back into their high-level original definitions.
- *Stage 3. Solution*: After Stage 2 is completed, it is followed by solutions to the series of sub-problems, specifications or constituents. In this stage, a number of alternatives or options are obtained for each sub-problem, and the best solution is selected.
- *Stage 4. Unification*: The fourth stage is aggregation or reconstruction of an overall product solution from the various solution alternatives to the sub-problems. Unification is an important step in solving IPD 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 helps eliminate the constraint violations and yields to refining the product for interface considerations.

The above four-stage process highlights the major steps to be undertaken in tackling an IPD problem. The process can be regarded as a continuous cycle. Systematization is an important step in yielding a faster and better solution.

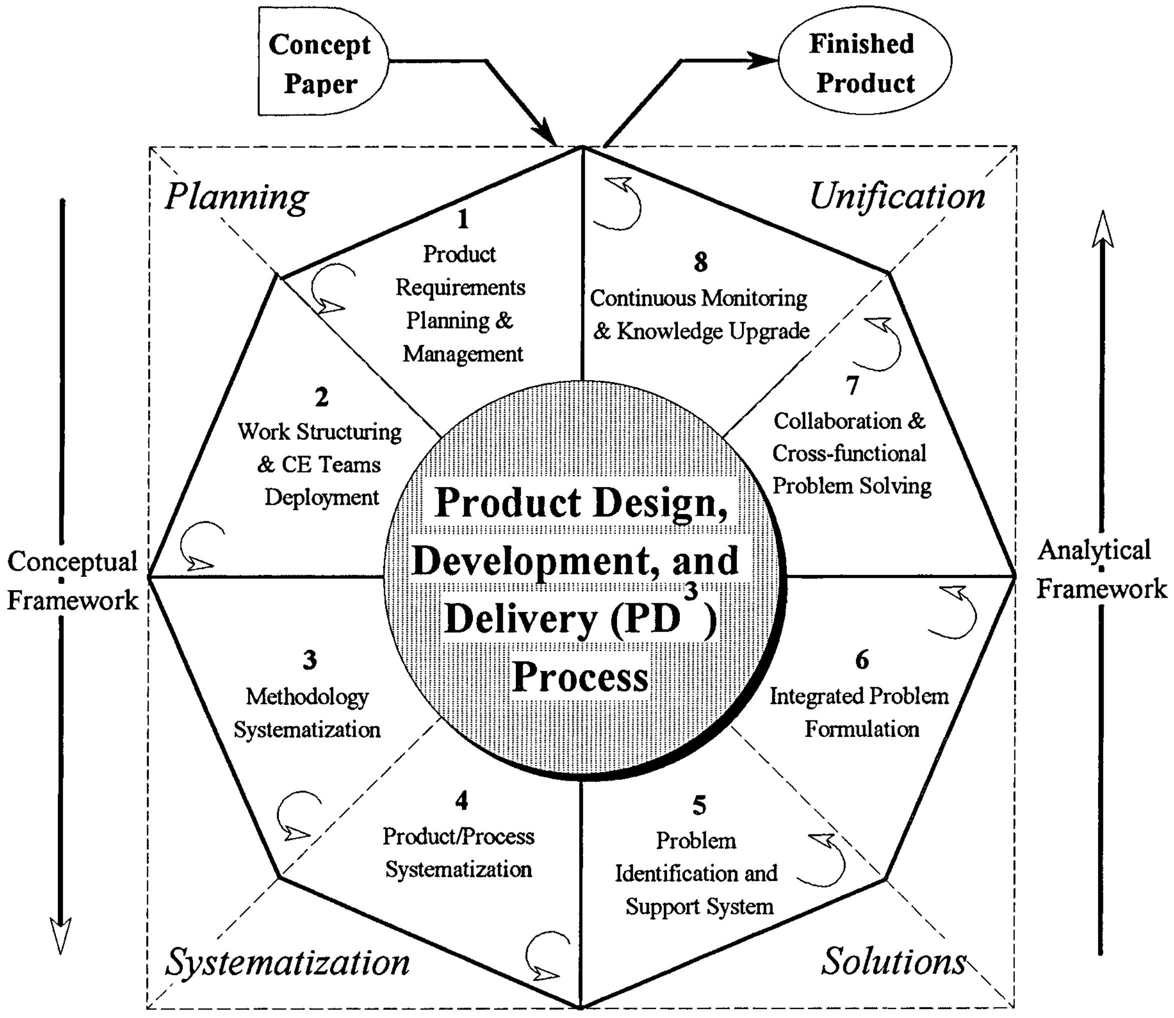
Concept Paper
Finished Product
Planning
Unification
Systematization
Solutions
1 Product Requirements Planning & Management
2 Work Structuring & CE Teams Deployment
3 Methodology Systematization
4 Product/Process Systematization
5 Problem Identification and Support System
6 Integrated Problem Formulation
7 Collaboration & Cross-functional Problem Solving
8 Continuous Monitoring & Knowledge Upgrade
Product Design, Development, and Delivery (PD3) Process
Conceptual Framework
Analytical Framework

Figure 5. Building blocks of an integrated PD^3 process

It reduces search and makes product realization more efficient. If the decomposition were not discrete, the sub-problems would neither be compatible nor necessarily complementary. The sub-problems will suffer from excessive interdependence. There will be a large number of constraints and design variables that are common to these sub-problems. One or two iterations of the above four-stage process would not lead to a 'good' solution. Several of the constraints would be in conflict. It may require an excessively large number of iterations. In such cases, decomposition may not have many real benefits. Real benefits are obtained when the four-stage process results in significant saving in time and effort, compared to solving the original system problem (Nemhauser, 1994) as a *'large-scale global optimization problem'*.

Branching and Bounding Methodology

A branching and bounding methodology has been used here to first branch the product and process sub-domains into loops, later bound these loops into a 'sub-domain' and then finally enfold it back into an IPD system. The notion of branching allows for the exploration of 'possibilities',

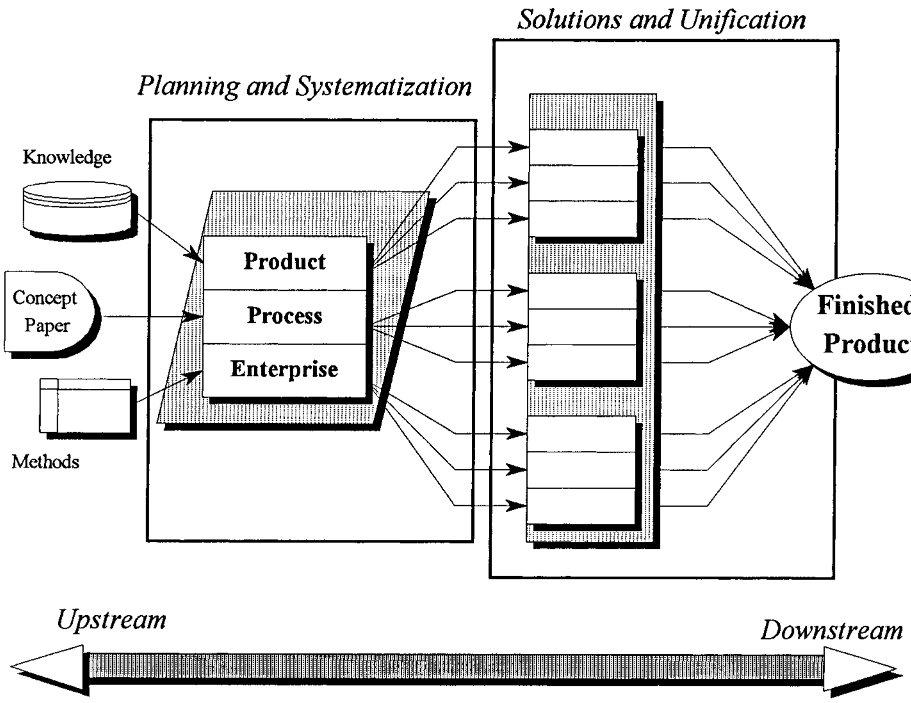

Figure 6. IPD methodology systematization

whereas bounding provides a way for the concurrent workgroups to judge the outputs (Prasad, 1997).

- *Branching*: Branching of each domain into loops is carried out uniformly (see Figure 7). A consistent representation is used throughout the branching and bounding process. In each group, the 'baseline system' can be further branched into independent sub units, 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 Figure 8). The other three loops—process planning execution, production synthesis and operation synthesis—provide a basis for satisfying process requirements giving rise to the so-called process-oriented loops (Figure 7). The second-level and third-level breakdowns of the product-oriented loops are also illustrated in Figure 8. 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 candidate system with respect to meeting the common set of requirements. During the bounding phase, the system computes the 'goodness value' based upon the constraints that are still not satisfied up to that stage. The common product and process constraints in the two half-domains provide a basis for determining the 'goodness' of the total system (Figure 7). The system value—a cumulative index on the measure of 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, re-evaluation of lower-level objectives or reconfiguration of goals. Bounding occurs through the use of concurrent function deployment, design function deployment or similar techniques (Evbuomwan and Sivaloganathan, 1994), for simultaneous consideration of a series of

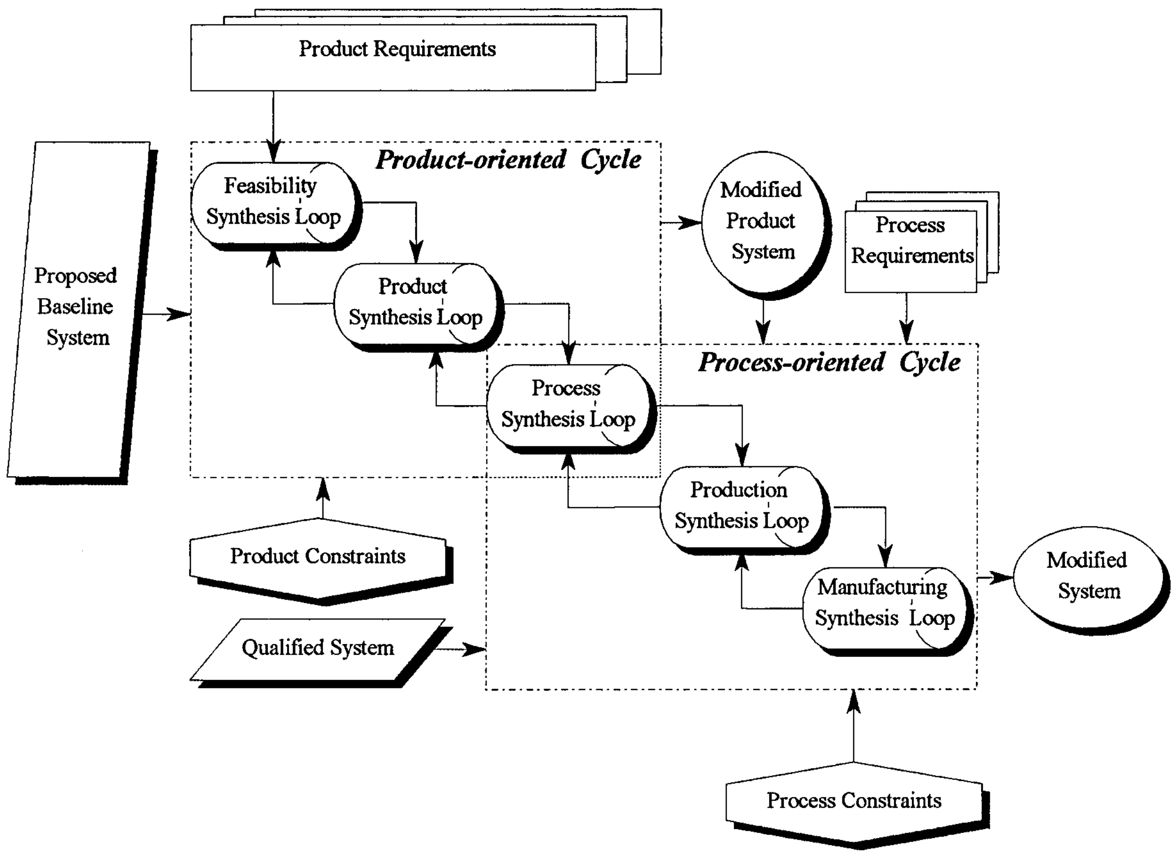

Figure 7. Branching of product and process management cycles into loops

competing requirements and objectives during an IPD process (Zhang and Zhang, 1995).

The branching and bounding methodology provides the ability to refine successively the 'goodness or fitness' of a baseline concept (Prasad, 1996) as one proceeds from one nested loop to the other (Figure 7). The domain of product realization process is, however, evolutionary. During a loop, the design or concept formulation is not fixed, rather it reflects CE teams' understanding of the design problem and environments spanned by its specification sets. As the satisfaction of the specification continues during a loop, the work groups learn more about the forthcoming baseline model and output (solution) state as new aspects of its behaviour inherent in the formulation are revealed. As a result, work groups may gain new insight into the behaviour of the model (and the solution output state). This 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 (Magrab, 1997):

- The incremental change in behaviour 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 redesign process as the product evolves from problem or customer

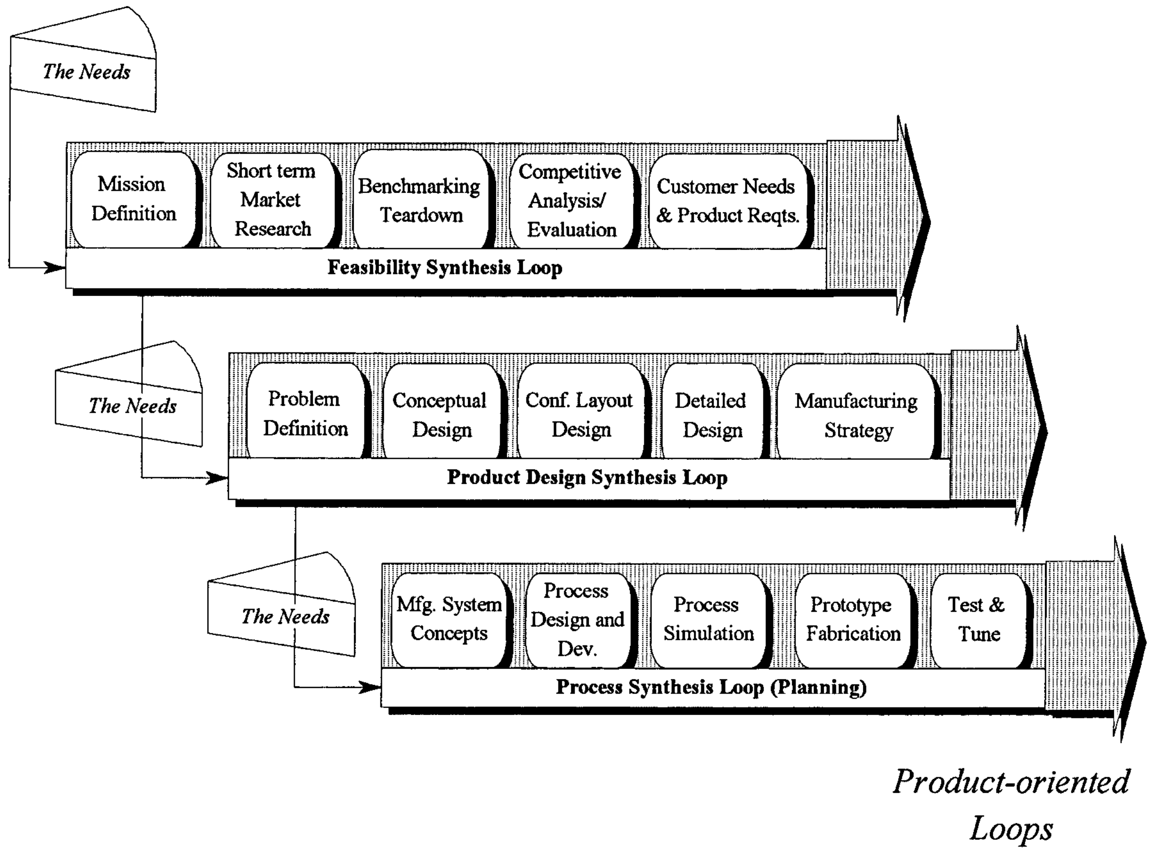

Figure 8. Concurrent elements of product-oriented loops

needs to an artifact instance. Each branch and bound procedure inherent within each loop provides a mechanism to select an increasingly good design or concept. Any intermediate trial design based on such bounding procedure is 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 a taxonomy-based CE process? Without the taxonomy (ability to classify the PD^3 process), one is forced to instantiate all possible configurations and check the compliance with respect to all possible value-characteristic requirements (Magrab, 1997). It may not be possible 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 (Stefik, 1981). This often results in a series of design concepts that are suboptimal in some way.

PRODUCT AND PROCESS SYSTEMATIZATION

Concurrency can be exploited by differentiation, followed by systematization—organizing the information in a hierarchical way (Figure 9). In most product systems, there are complex interactions among many of its constituents: subsystems, components, parts, materials, features, etc. Product systematization is a technique for handling a larger class of problems, or a product, by decomposing and then concatenating the

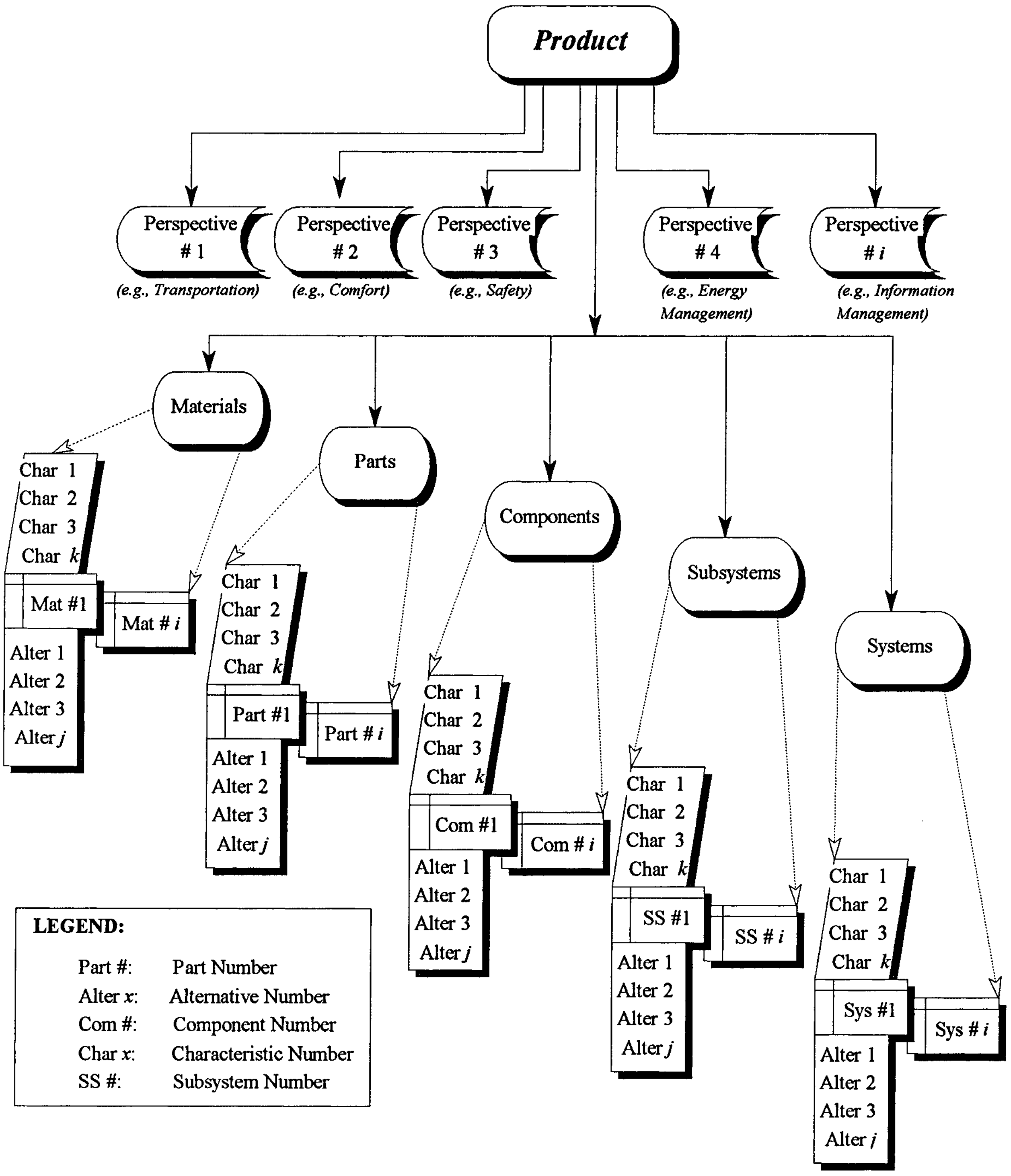

Figure 9. Areas of concurrency during product synthesis (bottom-up representation)

results of its behaviour through a smaller set of problems or hierarchical organization (Krishnan, 1993). Examples of an automobile, an aircraft and a helicopter are shown (Prasad, 1996). Similarly, a process can be decomposed into activities. A group of activities aggregated into a high-level activity group is called a scenario. By systematization of process in the early stages of product

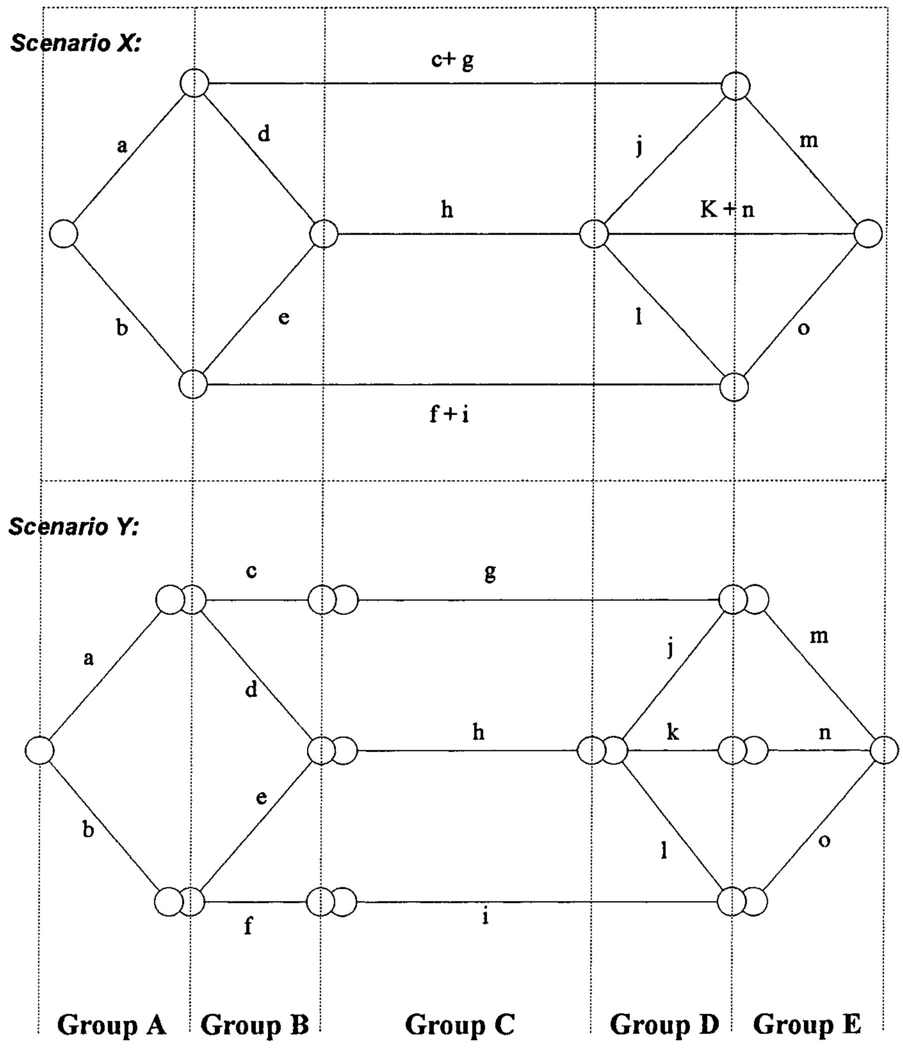

Figure 10. Decomposition of a scenario into a 'serially decomposable' activity groups

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 10 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 are not serially decomposable, other ordering methods are used (Navinchandra *et al.*, 1993) to avoid solving a large set of equations simultaneously. Activities within a group can run in parallel. There are two parallel activities in group A, four in group B, and three in groups C, D and E (see Figure 10). 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 the constituents themselves (see Figure 9). Besides some recent works on multi-level optimization and decomposition, the field of assembly, or system, is still new and growing. There are three possible ways in which product and process systematization can occur.

Through Problem Structure Recognition

Many researchers look for commonalities between the structures of the organization or the structures of the problem, or both (Galbraith, 1973). An engineering design can be divided into logically distinct modules that describe a portion of that design. This concept is known as 'modular design.' similarly, requirements and constraints (RCs) for each module can be broken down into manageable components. The idea is similar to what is done in software design. The computer program is divided into subroutines or procedures, which may be represented in pseudo code or an actual code. In modular design, we have modules of design that relate to each other. Alternate designs are created by giving values to their design modules. Each set of values given to a design module is called a version. Design session is a process of designing parts, and the version is an output of a design session. Versions of design modules can be related to one another in many ways. Several versions relate to a perspective, several perspectives relate to an assembly, several assemblies relate to a product. For example, there may be versions of the same design module belonging to the same product, or describing the same assembly, but from a different perspective (Hardwick *et al*., 1990).

Through Decomposition and Partitioning of Problems

Concurrency can be exploited by viewing the product as a complex system which can be decomposed or partitioned into a series of subproblems, each with its own set of RCs. Each subproblem can be solved in parallel with a subset of the RCs from the original problem and the results brought back to satisfy the coupling from the remainder of the RCs. The complexity of the products and the processes forces the work-groups to look for their breakdown structures and exploit any inherent independence so that the identified sub-problems can be solved somewhat independently from each other. Product decomposition thus simplifies the complexity of the original problem that would otherwise have to be alternatively dealt with. The original problem would have required consideration of all the requirements and constraints at one time. The use of smart modules alleviates this complexity by capturing the RCs in the form of product structure decomposed into smaller sizable chunks. It contains rules for reconfiguring or changing the product structure when there are new inputs.

Through Product or Process Organization

Products, in mechanical design, are divided into systems, subsystems, components, parts, and materials (Figure 9). In electrical design, circuits are divided into smaller subcircuits. Typically, most designs can be divided into perspectives as well as into configurations (or mock-ups) (Prasad, 1996). The focus of CE teams, during the initial realization process, is on the holistic elements of the PD^3 process design. The intent is to design the configuration (conceptual mock-up) first, before the perspectives are addressed. In automobiles, aeroplanes or helicopters, for example, perspectives might include aerodynamics, weight, strength, aesthetics, vibration, noise, etc. In mechanical design, a part might be described as a function or as a drawing; and in electrical design, a circuit might be described as a schematic or as a physical layout. It is important

Table 1. Organization by table

Part number	Cost	Price	Lead time	BOM routing	Planner code
ABC123	$30.00	$23.00	2 Months	SAE1030	Drill, Machine
ABC456	$67.00	$45.00	1 Week	Titanium	EDM

to note that such decompositions are not discrete. They are divided on the basis of:

- degree of independence;
- degree of compatibility;
- ability to provide complementary functions.

By decomposing a product into its constituents, and a process into its activities in such a way that their inherent dependencies are minimized, the level of concurrency can be maximized. Independence means that the strong interactions occur within an individual constituent or an activity itself. The weaker interactions occur across its decomposed constituents or activities. Pahl and Beitz (1991) and Suh (1990) used such concepts in modelling the functional requirements of product design. If the constituents or activities are inextricably related, such product or process divisions are not good. They are neither inherently compatible nor necessarily complementary.

Through Part/Activity Classifications

Classification is a basic concept. Once information is categorized, it can easily be found or tracked. A classic example is a periodic table. By categorizing the metals by their molecular weights, an understanding of physical laws of action and reaction is simplified. Another classical example of this is the library. All books in a library are categorized. This makes it easy to find a book on any topic, even though there are millions of books stored on the shelves. Manufacturers commonly keep track of thousands, even hundreds of thousands of parts, yet the vast majority of companies do not have a classification system. Classification is one of the basic scientific methods that can be used to understand a true product mix.

Methods of Classifications
Two most common methods of classifications are:

(a) Organization by tablet: In this method of classification, the variation of each item is captured in a tabular form (see Table 1). The advantage of this approach is that the table provides many different forms of sorting possibilities. For instance, a column of the table can be sorted based on a specific criteria. However, it results in a large number of possibilities (e.g., part number) if each item is varied independently.

(b) Organization by Properties: Generic properties provide a logical group of common characteristics (e.g., size, color, finish, tint) of a part, whose individual properties differ only in amount or types (e.g., cost, price, lead times) A template is defined for each property, which uniquely defines its characteristics using a set of conditional rules or equations. Using this template, a specific selection or variation for a property is configured and evaluated as in Pugh (1991). The template serves as a master database from which specific versions are generated on demand. Creators can control names, descriptions and placement of properties. In addition, one can link properties, thus allowing a way of grouping that makes most sense. There is no need to store all possible variation of the properties as in the table in the previous example.

Once someone starts to understand the product-mix that a company possesses or markets, they can also begin standardizing the design and process plans. Group Technology (GT) classification provides an important organizational structure for process data. An article written by E. M. Fisher, in *Datamation Magazine*, included this quotation from B. Chandrasekharan, 'classification (of data) is perhaps the most universally applied generic task. By classifying, an expert relates a single situation to a larger group of like cases. Instead of remembering what to do in each individual instance, you need only to remember what to do in each class of situation.' The two techniques for classifying objects are discussed in the following subsections.

Classification Techniques

(a) Through group technology: Group Technology (GT) is one of the oldest, yet most powerful, classification techniques known to the manufacturing community. For manufacturers, GT categories are based upon a combination of the following:

- the part's design geometry (what the part looks like);
- functional descriptions (what the part does);
- manufacturing processes (how to make the part).

GT classification techniques: Over the last two decades, many different techniques have been adopted in applying GT to manufacturing. The goals, however, have always been consistent. The basic methods of classifying parts are: monocodes, polycodes, and hybrids.

1. *Monocodes* are hierarchical systems that build relationships between features and attributes. For example, valves can be divided into subgroups, or types of valves (i.e., butterfly, gate, etc.). These subgroups (types) are related to the main group (valves).
2. *Polycodes* are not based on direct relationships. An example of this is an 'attribute,' such as material, which can be applied to all groups regardless of what the part is—whether it is a valve or an electronic circuit card assembly.
3. *Hybrid* system effectively combines Monocode and Polycode techniques. The team only has to enter, maintain and update the logical relationships that are needed in an entire hierarchical system. Hence, it is easier and faster to implement or change a hybrid system.

Group technology classification and coding schemes can also be used as a basis for a design retrieval system. This way, a GT retrieval system can provide a unified classification scheme for both design and manufacturing situations. However, GT provides more benefits to the manufacturing community than what design standardization provides to the design community. With GT, both parts and assemblies can be categorized. It is easy for anyone in the company to retrieve and utilize the historical information originally used to make or assemble a part. The benefits are real. This is because, irrespective of the techniques used, productivity improvements have been realized by utilizing the company's existing information, even without building any database or employing an experienced work group.

(b) Through design standardization: Standardization is another form of classification used by design teams to improve the consistency of parts being designed (Hales, 1993). Like GT, with design standardization, workgroups can frequently find and reuse existing designs rather than redesigning the same part. Even when the existing designs are not exactly right, design standardization allows work groups to locate similar designs that can be used as a starting point or for knowledge gathering. This can dramatically reduce the total design effort. Further, it is not necessary to recheck the design for all X-ability considerations since the standards ensure compliance with respect to many features that are in common. Thus, this standardization allows a concurrent team to share quickly proven design alternatives with other workgroups (Zhang and Zhang, 1995), and redesign the rest more quickly. From a CE perspective, both approaches of classification provide a means of standardization. GT is a tool primary known for manufacturing standardization. Parts whose features are the same need not be reanalyzed or redesigned. Similarly, if a manufacturing process plan is obtained for a part, no further work is required. The process plan stays the same for the entire group of similar parts. Standardization provides substantial benefits:

- eliminates unnecessary duplication of parts;
- controls excessive proliferation of new parts and manufacturing processes;

- reduces set-up time, work-in-process inventory;
- reduces scraps, while improving the quality of parts.

GT classification and design standardization are prerequisites for concurrent product design (Zhang and Zhang, 1995). With the help of these tools, the problem is broken down into family of subproblems, taking all active design and manufacturing requirements into consideration. Breaking down the problem into sub-problems makes it easier to solve, because it is easier for someone to deal with and understand a set of a hundred parts than thousands of parts.

CONCLUDING REMARKS

The paper presents a number of product, process, and methodology (PPM) systematization techniques to reduce the inherent complexity of the product development process. A taxonomy-based CE process for systematization, proposed in the paper, simplifies the problem domain into a smaller number of concurrent sets, which are easy to deal with and solve. The four-stage PPM methodology explained in the body of this paper helps to manage the specifications towards an integrated product development. It balances the optimal use of *7Ts* (talents, tasks, teamwork, techniques, technology, time and tools) resources against the risk of changing the concepts through iterations and loops while the product is being evolved.

The four-stage methodology can be applied to any new product introduction scenario, or to any problem set having a deviation from its initial specifications. This can also be used to tackle a continuous improvement opportunity. PPM systematization ensures that everything possible will be done to apply the *7Ts* resources in the most effective manner.

REFERENCES

Bajgoric N. 1997. Organizational systems integration: management information systems perspective,' *Concurrent Engineering: Research and Applications* **5**: 113–122.

Berger S, *et al*. 1989. Towards a new industrial America. *Scientific American*. June: 39–47.

Brandimarte P, Villa A. 1995. *Advanced Models for Manufacturing Systems Management*. CRC Press: Boca Raton, FL.

Checkland P. 1981. *Systems Thinking, Systems Practice.* John Wiley and Sons, Ltd. New York.

Checkland P. 1984. Rethinking a system approach. In *Rethinking the Process of Operational Research and System Analysis*, Tomlinson R, Kiss I (eds). Pergamon Press: New York.

Clark KB, Fujimoto T. 1991. *Product Development Performance: Strategy, Organization, and Management in the World Auto Industry*. Harvard Business School Press: Boston, MA.

Clausing DP. 1994. *Total Quality Development: A Step-by-step Guide to World-Class Concurrent Engineering*. ASME Press: New York.

Dong J. 1995. Organization structures, concurrent engineering, and computerized enterprise integration. *Concurrent Engineering: Research and Applications* **3**(3): 167–176.

Daft R. 1995. *Organization Theory and Design*, (5th edn). West Publishing: New York.

Evbuomwan NFO, Sivaloganathan S. 1994. The nature, classification and management of tools and resources for concurrent engineering. In *Proceedings of the 1st International Conference on Concurrent Engineering: Research and Applications*, Paul AJ, Sobolewski M (eds). Pittsburgh PA, 29–31 August; 119–126.

Fleischer M, Liker JK. 1997. *Concurrent Engineering Effectiveness: Integrating Product Development across Organizations*. Hanser Gardner: Cincinnati OH.

Galbraith JR. 1973. *Designing Complex Organizations*. Addison-Wesley: Reading MA.

Hales C. 1993. *Managing Engineering Design*. England: Longman: London/Wiley: New York.

Harr S, Clausing DP, Eppinger SD. 1993. Integration of quality function deployment and the design structure matrix. Working paper no. LMP-93-004, Laboratory for Manufacturing and Productivity, Massachusetts Institute of Technology, Cambridge, MA.

Hardwick M *et al*. 1990. ROSE: A database system for concurrent engineering applications. In *Proceedings of the Second National Symposium on Concurrent Engineering*, Morgantown, WV, 7–9 February; 33–65.

Ho JKK, Sculli D. 1995. System complexity and the design of decision support systems. *Systems Practice* **8**(5): 505–516.

Kamath RR, Liker JK. 1994. A second look at Japanese product development. *Harvard Business Review*, November–December: 154–170.

Krishnan V. 1993. Design process improvement: sequencing and overlapping activities in product development. D.Sc. thesis, Massachusetts Institute of Technology, September 1993.

Kusiak A, Wang J. 1993. Decomposition of the design process. *Journal of Mechanical Design* **115**: 687–695.

Lootsma FA. 1994. Alternate optimization strategies for large-scale production allocation problems. *European Journal of Operational Research* **75**: 13.

Magrab EB. 1997. *Integrating Product and Process Design and Development: The Product Realization Process*. CRC Press: Boca Raton, FL.

Masud ASM, Dean EB. 1993. Using fuzzy sets in quality function deployment. In *Proceedings of the 2nd Industrial Research Conference*, Mitta DA *et al*. (eds). Industrial Engineering and Management Press: Norcross, GA.

Navinchandra D, Fox MS, Gardner ES. 1993. Constraint management in design fusion. *Concurrent Engineering: Methodology and Applications*, Gu P, Kusiak A (eds). Elsevier:Amsterdam; 1–30.

Nemhauser G. 1994. The age of optimization: solving large-scale real-world problems. *Operations Research* **42**: 1–12.

Pahl G, Beitz W. 1991. In *Engineering Design: A Systematic Approach*. Wallace K (ed.). Springer: New York.

Parsaei HR, Sullivan WG. 1993. *Concurrent Engineering: Contemporary Issues and Modern Design Tools*. Chapman & Hall: London.

Prasad B. 1985. An integrated system for optimal structural synthesis and remodelling. *Computers and Structures*, **20**(5): 827–839.

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

Prasad B, Emerson JF. 1984. Optimal structural remodelling of multi-objective systems. *Computers and Structures* **18**(4): 619– 628.

Prasad B. 1984b. Explicit constraint approximation forms in structural optimization. Part 2: Numerical experiences. *Computer Methods in Applied Mechanics and Engineering*, **46**(1): 15–38.

Prasad B, Magee CL. 1984. Application of optimization techniques to vehicle design: a review. *Recent Experiences in Multidisciplinary Analysis and Optimization*. NASA CP 2327, Part 1, April 1984. NASA Langley Research Center, VA; 147–171.

Prasad B. 1996. Concurrent Engineering Fundamentals, Vol I: *Integrated Product and Process Organization*. Prentice Hall: Upper Saddle River, NJ.

Prasad B. 1997. Concurrent Engineering Fundamentals, Vol II: *Integrated Product and Process Organization*. Prentice Hall: Upper Saddle River, NJ.

Pugh S. 1991. *Total Design: Integrated Methods for Successful Product Engineering*. Addison-Wesley: Wokingham, UK.

Saaty TL. 1978. Exploring the interface between hierarchies, multiple objectives and fuzzy sets. *Fuzzy Sets and Systems* **1**: 57–68.

Smith GF, Browne GJ. 1993. Conceptual foundations of design problem solving. *IEEE Transactions on Systems, Man, and Cybernetics* **23**(5).

Stark J. 1992. *Engineering Information Management Systems: Beyond CAD/CAM to Concurrent Engineering*. Van Nostrand Reinhold: New York.

Stefik M. 1981. Planning with Constraints (MOLGEN: Part I and Part II). *Artificial Intelligence* **16**: 111–169.

Suri R. 1988. A new perspective on manufacturing system. In *Design and Analysis of Integrated Manufacturing Systems*, Compton WD (ed.). National Academy Press: May, Washington DC.

Suh WP. 1990. *The Principles of Design*. Oxford University Press: Oxford.

Tsuda Y. 1995. QFD models for concurrent engineering development processes of automobiles. *Concurrent Engineering: Research and Applications* **3**(3): 213–220.

Ulrich KT, Eppinger SD. 1994. *Product Design and Development*. McGraw-Hill: New York.

Warfield J, Hill JD. 1972. *A unified systems engineering concept*. Gordon BB (ed.). Battele Monograph, no. 1 (June) New York, NY.

Zhang HC, Zhang D. 1995. Concurrent engineering: an overview from manufacturing engineering perspectives. *Concurrent Engineering: Research and Applications* **3**(3): 221–236.