

Computational Intelligence in Product Design Engineering: Review and Trends

Andrew Kusiak, *Member, IEEE*, and Filippo A. Salustri, *Member, IEEE*

Abstract—Product design engineering is undergoing a transformation from informal and largely experience-based discipline to a science-based domain. Computational intelligence (CI) offers models and algorithms that can contribute greatly to design formalization and automation. This paper surveys CI concepts and approaches applicable to product design engineering. Taxonomy of the surveyed literature is presented according to the generally recognized areas in both product design engineering and CI. Some research issues that arise from the broad perspective presented in the paper have been signaled but not fully pursued. No survey of such a broad field can be complete; however, the material presented in the paper is a summary of state-of-the-art CI concepts and approaches in product design engineering.

Index Terms—Computational intelligence (CI), decision making, design automation, engineering design, product engineering.

I. INTRODUCTION

PRODUCT design engineering is a complex discipline; it draws upon and contributes to other disciplines, and it is not well formalized. This interdisciplinary nature of product design engineering has resulted in numerous computational approaches that have been reported in the literature. The goal of this paper is to discuss the recent computational intelligence (CI) results applied to product design engineering, and structure the CI approaches in a general unifying framework.

No survey of such a broad field can be complete. An attempt has been made to balance the degree of detail against availability of literature sources. Any imbalance in the coverage is due to the availability of information rather than the topic's importance. Such topics are included as they are relevant to the breadth of our survey.

A general taxonomy of models used in product design engineering is proposed in Fig. 1. Some formal approaches in product design engineering fall into the programming language category (category 1 in Fig. 1) equating designing mechanical components as coding. This approach is analogous to the hardware design language developed in electronics.

There is no indication that a widely accepted computer instruction approach to product design engineering will be realized in the near future, and therefore, category 2 in Fig. 1 presents a more realistic option of object-based design. This is a coarser

approach to product design engineering aiming at capturing higher level objects, i.e., parts and assemblies defined by a collection of functions. Category 3 in Fig. 1 introduces a natural system perspective to product design engineering. This perspective supposes that any product can be viewed as a genome, with subassemblies represented by chromosomes, and parts represented by genes. Genetic operators, decision rules, and other logic would govern the design and redesign of products. The nature-based perspective to mechanical design could be the most promising; as such, it will be a focal point of this survey.

Many researchers favor an optimization approach to product design engineering, shown as category 4 in Fig. 1. This approach usually describes some aspects of design with a constrained objective function. A multitude of different models reflects different facets of mechanical design under this approach. However, here, the term “optimization” is used in a broader sense than in mathematical programming. Once a basic design concept is established, the remainder of the development process can be regarded as a refining that concept into a real product. This sense of “refinement” is how the authors wish the reader to interpret the term “optimization.”

In summary, one may distinguish between categories 2, 3, and 4 in Fig. 1, as representing perspectives of informational structure versus evolutionary development and optimization, respectively.

In the next section, numerous approaches and methods that fall into categories 2–4 are examined.

II. TAXONOMY OF CI ALGORITHMS, TECHNIQUES, AND TOOLS

Many experts agree that CI will contribute greatly to design automation. Machine learning algorithms fuse historical design information distributed in space and time into coherent and understandable design knowledge. The only impediment here is in the representation of such information in a uniform way.

The CI approaches of potential use in product design engineering can be grouped into seven major classes. These classes are identified and related to the three categories of Fig. 1 in Table I. Note that category 1 of Fig. 1 has been excluded from Table I as the computer code-development approach requires an extensive coverage that could not be accommodated in this paper. The criterion used for matching a CI approach with a category is based on the literature coverage. If a substantive collection of research was found, then an “×” was placed in the appropriate entry of Table I.

The literature pertaining to each of the seven approaches of Table I is discussed in the sections that follow.

Manuscript received August 30, 2005; revised May 23, 2006. This paper was recommended by Associate Editor D. McFarlane.

A. Kusiak is with the Intelligent Systems Laboratory, Department of Mechanical and Industrial Engineering, University of Iowa, Iowa City, IA 52242 1527 USA (e-mail: andrew-kusiak@uiowa.edu).

F. A. Salustri is with the Department of Mechanical and Industrial Engineering, Ryerson University, Toronto, ON M5B 2K3, Canada (e-mail: salustri@ryerson.ca).

Digital Object Identifier 10.1109/TSMCC.2007.900669

Category 1: Computer code development	Category 2: Design objects	Category 3: Genetic analogy	Category 4: Optimization
---	-------------------------------	--------------------------------	-----------------------------

Fig. 1. Classification of modeling approaches in product design engineering.

TABLE I
CI METHODS AND CATEGORIES OF RESEARCH APPROACHES OF FIG. 1

	Category 2: Design Objects	Category 3: Genetic Analogy	Category 4: Optimization
Ontologies	x		
Data Mining	x		
Evolutionary Computation		x	
Decision Making			x
Case-Based Reasoning	x		x
Qualitative Reasoning			x
Hybrid Approaches	x	x	x

A. Ontologies

An *ontology* is an agreed upon set of terms and meanings that enables parties to share diverse knowledge through a common language [1]. “Intelligent” methods and algorithms use knowledge organized into ontologies. The nature and representation of ontologies are of importance to product design engineering. Ontologies are the formal underpinnings of all methods that define knowledge as complex structures, and they are fundamental to category 2 (design objects) approaches (see Table I).

Fowler *et al.* [2] used ontologies of engineering features to develop software to check that product configurations satisfy both physical and organizational constraints. Yoshioka *et al.* [3] developed a framework for knowledge-intensive engineering driven substantially by ontologies of physical concepts that constitute “pluggable” metamodels in the framework. These physical concept ontologies form the common basis to integrate diverse information sources.

Ontologies were used by Kitamura *et al.* [4] to build and successfully deploy a framework to capture and reuse knowledge about product functions in a large electric corporation. The authors reported that a key feature of their framework was its ability to make explicit the knowledge that designers would only use implicitly otherwise, and to help share the knowledge with team members.

Brown *et al.* [5] used ontologies to create a Web-based repository to support the distributed development of automotive components, using conventional Web technologies and standards. Leveraging the Web for such purposes allows users to add and search content using ubiquitous and robust systems that many industries already have in place.

Tao *et al.* [6] described the development of ontologies to facilitate searching design spaces based on the *semantic grid* concept of the Web. The reported work is part of the Geodise project (<http://www.geodise.org>) intended to provide a complete Web-enabled knowledge-based system (KBS) for design and optimization involving fluid dynamics.

Tormey *et al.* [7] developed agent-based systems using ontologies to support collaborative design processes by making diverse and distributed sources of knowledge appear homogeneous and integrated to users.

The Enterprise Intelligence Laboratory at the University of Toronto has developed an extensive ontology-based capability to address modeling of an entire enterprise, including requirements, supply chain management, quality management, and so on [8].

Lee and Luo [9] used ontologies to analyze requirements in the domain of network management software.

B. Data Mining

The volume of “legacy data” collected by industry is growing at an unprecedented rate. Such large quantity of data is usually difficult to process and analyze, yet is likely to be a source of valuable knowledge. Data mining, also called *knowledge discovery*, provides algorithms for searching and summarizing the legacy data in a usable form. Data mining can be combined with other approaches to develop intelligent systems. It falls into category 2 (Table I) as it creates design objects from the typically unstructured legacy data.

Kusiak [10] proposed a data mining system for predicting product cost using historical design data. A rough-set theory algorithm was used to extract the decision-making knowledge. Ishino and Jin [11] applied data mining for knowledge acquisition from design activities involving a computer-aided design (CAD) system. They developed a method called *extended dynamic programming* to extract the knowledge. Romanowski and Nagi [12] proposed a design system based on knowledge extracted from product life cycle data. Giess *et al.* [13] mined manufacturing and assembly data of gas turbine rotors to establish and quantify relationships between the balance and vibration data, which, in turn, improved component tolerance designs. Hamburg [14] applied a decision-tree algorithm to support product development by analyzing high-level data such as market position, strategy, philosophy, and culture of the manufacturing and customer behavior. The extracted knowledge was to be integrated in the product development process. Albers and Marz [15] used data mining to extract know-how of disciplines related to design processes for microtechnological products. Cascint [16] uses data mining techniques to create TRIZ-based semantic portals to support the redesign of metal parts in plastic.

Terpenney *et al.* [17] developed a methodology to assist in the discovery, classification, and capture of design knowledge. The methodology was intended to guide industries in developing taxonomies and ontologies in a practical way. By providing such guidance, the paper demonstrated that it was possible to build semiautonomous, agent-based tools using structured knowledge in design.

C. Evolutionary Computation

Evolutionary computation is concerned with the development of problem-solving methods based on concepts from natural

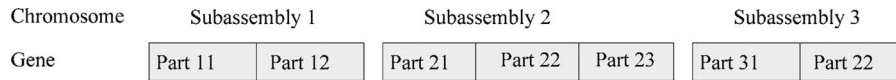


Fig. 2. Genomic representation of a product.

systems. A number of evolutionary computational methods have been developed, including genetic algorithms (GAs), genetic programming (GP), evolutionary strategies (ES), and evolutionary programming. All of these approaches are discussed next.

1) *Genetic Algorithms*: A human organism in its all complexity can be represented by approximately 30 000 genes expressed as a vector of four genetic letters A, C, G, and T, or just a series of zeros and ones, if one prefers the binary representation. Any two humans differ only in a small percentage of their “genetic vectors” regardless of their phenotypic differences (looks). How long would the vector of “product letters” need to represent a watch, a bicycle, or a space station? One could argue that such vectors of characters can be handled by the modern computer hardware and software. Thinking of a product design in a bottom-up (genetic like) rather than the top-down (the way the products are often viewed today) leads one to believe that intelligent systems may dramatically change the course of product design engineering. The time could be ripe to explore the genetic design paradigm.

Fig. 2 shows a “genomic-like” representation of a product, where the genes represent parts, and the chromosomes represent subassemblies. The same product could be represented at finer level by assigning genes to its design features.

Bentley and Wakefield [18] applied a GA to design a table. Using primitive shape representations, tables were designed to satisfy the size constraints, the distribution of mass, and the resulting stability. Cho [19] applied a GA to learn the replication of human intent of interest to product design engineering. The user provided initial selections and the algorithm optimized the intent.

2) *Genetic Programming*: GP creates a computer program in the scheme computer language as the solution [20], [21]. Zhang and Muehlenbein [22] investigated the relationship between the performance and complexity of the evolved structures. Employing statistical search, Iba *et al.* [23] introduced a new approach to GP by integrating a GP-based adaptive search of tree structures and a local parameter tuning mechanism. In traditional GP, recombination can cause frequent disruption of building blocks, or mutation can cause abrupt changes in the semantics. Soule and Foster [24] showed that poor results with parsimony pressure are due to “failed” pmpulations that overshadow the results of the populations that successfully incorporate the parsimony pressure. Additionally, they showed that the effect of parsimony pressure could be measured by calculating the relationship between program size and performance within the population. This measure can be used as a partial indicator of success or failure of individual populations. Yao *et al.* [25] proposed a fast evolutionary programming algorithm using as the primary operator Cauchy rather than Gaussian mutation. The authors showed the relationship between the search step

size and the probability of finding a global optimum. GP has also been applied to multiobjective robust design in [26].

Within the limits of their current applicability, GP algorithms have typically been able to generate and evaluate far more design alternatives than any team of designers. As such, these algorithms can be effective design tools, especially in developing new and innovative design alternatives.

3) *Evolutionary Programming*: *Evolutionary programming [also known as evolutionary algorithms (EAs)]* incorporates the aspects of natural selection or survival of the fittest. An EA maintains a population of structures (initially randomly generated) that evolve according to the rules of selection, recombination, mutation, and survival, referred here as genetic operators. A shared “environment” determines the fitness (performance) of each individual in the population. The fittest individuals are more likely to be selected for reproduction (e.g., retention, duplication), while recombination and mutation modify the individuals, yielding potentially superior ones.

The background on various implementations of EAs is provided in [27]–[30]. The last paper provides a comprehensive typology of EAs. A promising alternative in solving difficult and dynamic problems is the coevolutionary algorithm, which is a variation of EA where each individual represents only a partial solution to the problem (see [31] and [32]).

4) *Evolutionary Strategies*: ES is an algorithm where individuals (potential solutions) are encoded by a set of real-valued “object variables” (the individual’s ‘genome’). For each object variable, an individual has a “strategy variable” determining the degree of mutation to be applied to the corresponding object variable. The strategy variables mutate, allowing the rate of mutation of the object variables to vary. The population size, the number of offspring produced in each generation, and whether the new population is selected from parents and offspring or only from the offspring characterize an ES.

Eiben and Bäck [33] extended the ES approach to multiparent recombination involving a variable number of parents to create an individual offspring. The extension was experimentally evaluated on a test suite of functions of different modality and separability and the regular/irregular arrangement of their local optima. Multiparent diagonal crossover and uniform scanning crossover and a multiparent version of intermediary recombination were considered in the experiment. Olafsson [34] demonstrated the use of evolutionary game theory for the allocation of service requirements onto an ensemble of heterogeneous network components. Schweitzer *et al.* [35] applied Boltzmann and Darwin’s and mixed strategies to find differently optimized solutions (graphs of varying density) for the road network, depending on the degree of frustration. They showed that the optimization process occurs on two different time scales. In the asymptotic limit, a fixed relation between the mean connection distance (detour) and the total length (costs) of the network

exists that defines a range of possible compromises. Thompson *et al.* [36] presented ES to design a reconfigurable controller. The designed product exhibit better properties than the one designed with conventional constraint-based methods. Moriarty and Miikkulainen [32] applied coevolutionary search to design a neural network. The designed network was robust due to neurons assuming overlapping roles as well as increased diversity. Lohn and Colombano [37] presented an evolutionary search method for automatic generation of circuit designs. They used a set of circuit primitives that were synthesized in valid circuits. The algorithm allows for the evolution of the circuit size, circuit topology, and device values.

D. Decision Making

Product engineering has a significant decision-making element. Intelligent decision-support systems are especially useful in product design engineering because of high complexity associated with the decisions and of the risks associated with making wrong decisions. Decision-making algorithms optimize various design outcomes, and therefore, naturally fall in category 4 (Table I).

Sim and Chan [38] developed a KBS for rolling element bearing selection. They used heuristic knowledge supported by a manufacturer's catalog to generate a solution. Stacey *et al.* [39] reported on a CI approach, called *signposting*, to support decision making in design. Signposting provides both inference knowledge and strategic problem-solving knowledge by focusing on the dependencies between design parameters. Danesh and Jin [40] created an agent-based decision network to support decision making in collaborative design. Each designer is represented with a software agent in an objective-based negotiation environment.

Agents were also used by Chao *et al.* [41] to model interactions between design systems used by multiple teams working on large-scale, complex-design problems. An evolutionary approach was used to automate negotiation between the agents in exchanging design solutions from different systems.

Mussi [42] presented a method for building decision-support systems based on decision theory using value of information. The method accounts for the vagueness of information derived from tests needed to validate hypotheses crucial to task completion, which is typical of product design engineering situations.

E. Case-Based Reasoning

Case-based reasoning (CBR) is an approach that attempts to mimic the human capacity to adapt and reuse solutions from known problems to new ones. It assumes that similar problems can be solved with similar solution approaches. A *case* is a description of a problem and its solution. New problems are analyzed and compared to known cases until a best match is found. The solution of the matching case is used (and sometimes adapted) to solve the new problem. CBR performs best when the library of known cases is such that each case: 1) is representative of a particular class of common problems and 2) has some similarity to a few other cases in the library. The major operational elements of CBR systems include gathering

and analyzing cases, establishing a "similarity measure" for new problems, and adapting known solutions to new problems. The structuring of cases places CBR to the design object (category 2, Table I) approach, while its aspects of space searching indicate its membership in category 4 (optimization).

CBR has been used to build software of known solutions. Depending on the way the similarity is defined, it is possible to apply CBR in quite innovative settings. Scott and Cook [43] used CBR in combination with context-free grammars to "emulate human reasoning" with respect to assessing product requirements. Morcoux *et al.* [44] applied CBR to model infrastructure deterioration in civil engineering structures using a large volume of data (i.e., large number of cases) on the strength and deterioration of structures.

Rivard and Fenves [45] developed a CBR system for conceptual design of buildings. The system supports the hierarchical decomposition of design cases, offers multiple views, and encapsulates the outcome of the design. Multiple case retrieval methods are available, and case adaptation is done by a "replay" method of existent processes. Note that adaptation is generally a parametric operation requiring a parameterized model of the object being designed. Such parametric models may themselves be the object of intelligent systems.

Concept maps were used to navigate and manipulate cases and their adaptations in the CBR system presented in [46] to support aerospace design.

Many applications of CBR in design have been restricted to relatively narrow domains. Lee and Luo [9] developed a CBR system for the design of die-casting dies. The system logs how humans use it and trains itself to new cases, thus improving its performance that is transparent to its user community. Tor *et al.* [47] applied a two-stage similarity algorithm to control the size of the search space in the CBR design of a stamping die. Their solution is demonstrated to noticeably speed up die design. Qin and Regli [48] applied CBR to the design of mechanical bearings. Vong *et al.* [49] used CBR to design the hydraulic circuits of production machines.

CBR has also been applied to broader cases, e.g., Chiu [50] used CBR to studying cognitive processes of designers.

F. Qualitative Reasoning

Qualitative reasoning allows developing models when the relationships between variables and parameters are not well established [51]. These methods seek ideal solutions to simplified or abstracted situations, and therefore, they fall in category 4 (Table I). While they are not able to operate with highly detailed "real life" information, they are able to guide design engineers in general terms. The qualitative reasoning approach integrates well with the knowledge to be extracted from the data sets [52]. Bond graphs are well suited to integrate the process modeling constructs. They provide means for unambiguous definition of the behavior of components by [53] the following.

- 1) Use of a limited number of versatile general terms and symbols to provide a rational graphical structure describing the presence and the interaction of effects impacting the dynamic performance of the system.

- 2) Allowing for ready formation and subsequent changes of the structure, important in the creative system design.
- 3) Use of the model structure to formally prepare a rational and adequately complete set of equations suitable for computer simulation of the system.

Karnopp *et al.* [53] described applications of bond graphs in engineering systems using the same set of ideal elements and provided standard techniques for translating these into a simulation model. Zakarian and Kusiak [54] discussed bidirectional reasoning of interest to product design engineering.

Other research in qualitative reasoning has focused on the application of qualitative physics to engineering in general—such as Pisan [55], but has also examined the role of other theories of qualitative reasoning in product design engineering such as analogical reasoning (see [56]). Stahovich *et al.* [57] used qualitative reasoning to develop a nonmathematical formalization of rigid-body mechanics.

G. Hybrid Approaches

As individual methods and techniques have matured, an interest in combining them has emerged. Combinations of different methods have led to hybrid approaches that could mitigate the shortcomings of the elemental methods. Obviously, hybrid approaches span all of categories 2–4 in Table I, as they combine aspects of all the major approaches.

Chau and Albermani [58] developed a hybrid system including production rules, object-oriented programming, and procedural methods to express engineering heuristics in a blackboard KBS for designing liquid retaining structures. The system can provide advice in preliminary design as well as downstream design stages.

Lou *et al.* [59] developed a new frame-rule structure for knowledge processing in mold design by incorporating features of product modeling, frame-based KBSs, CBR, and neural networks. They reported that design efficiency was significantly improved. Zhang *et al.* [60] developed a system integrating blackboard architecture with CBR for stamping process planning in progressing die design. The advantage of the system is that CBR can be used with past data as well as other reasoning methods. Many hybrid approaches fall in the category of soft computing methods and are discussed at numerous conferences and publications. One of the major drivers of soft computing is the fuzzy set theory (e.g., see [61] and [62]).

Nursel [63] reported an interesting use of a GA to design neural network structures. The combined GA and neural network approach are reported to reduce the computational complexity in design and manufacturing applications.

Evolutionary programming was used to adapt previously stored design solutions in a CBR system [64]. It is argued that such “knowledge-lean” techniques are more broadly applicable than conventional case-based design approaches. Chan *et al.* [65] used the *analytical hierarchy process* methodology jointly with expert systems, fuzzy systems, and neural networks to develop a decision-support tool for designing flexible manufacturing systems.

III. PROCESS VIEW OF PRODUCT DESIGN ENGINEERING

Product development processes can be considered as artifacts sharing commonality with the products themselves. In this section, a categorization of design processes based on similarities between the process and product characteristics is presented. The categorization in this section refers to the three main categories of Table I.

A. Typological Characteristics of Processes

1) *Modularity*: The first and most obvious characteristic is that of *modular* versus *platform-based* processes (in analogy to product development). Modular processes are composed of “ready-made” elements that can be assembled into an overall process. The functional nature of the modules makes optimization (category 4, Table I) approaches likely to be used.

Platform-based processes use common bases that are modified to suit specific needs. Since they depend on well-established bases, there is ample opportunity to perform data mining. However, platform-based processes are easy to institutionalize but harder to adapt to corporate and technological changes.

2) *Platform Orientation*: Analogous to the product platform typology discussed in [66], processes can be based on *standard components*, *basic components*, *common architecture*, and *standard interfaces*. Processes based on *standard components* are built up from specific component processes; the overall structures of these processes are defined each time a new process is developed. A process based on *basic components* reuses certain fundamental process components connected in well-understood ways (within a particular company or setting), enhancing the process with other (possibly new) process components as needed. A process based on *common architecture* uses predefined overall process structure (e.g., a generalized workflow-like arrangement) and fleshes out process components as required by the application of the architecture to a particular situation. Finally, a process based on *standard interfaces* develops process architecture and components.

3) *Differentiation*: Some processes, called here the *standardized processes*, are based on common process models intended to meet specific goals (such as best practices). Otherwise, processes can be *early differentiation* processes or *delayed differentiation* processes [67]. The latter two processes differ in the distribution of process activities. The early differentiation process shapes the unique design at the beginning of the process, while late differentiation process makes design unique at the final stage. Therefore, the late differentiation processes are more likely reusable across different products. This characterization is concerned primarily with optimization, i.e., finding an ideal process based on corporate and other constraints, and thus, fall in category 4 of Table I.

4) *Modification*: Processes can be modified generally for three reasons. Modifications may be done to *customize* the process for reuse in a new setting. Processes may also be modified for improvement, either *gradually* (e.g., continuous improvement) or by more radical *reengineering* method. Reengineering of processes is usually undertaken only when substantial

changes are essential. This characterization belongs to category 4 as it deals with optimization and goal attainment.

5) *Customization*: Processes can be characterized by the type of their reuse. *Unique processes* that are not expected to be reused as opposed to the processes developed using principles of *mass customization*. For the latter, we intend that specific process aspects are identified *a priori* as variable and that are expected to change based on how the process will be applied. Considering process elements is analogous to the design objects of category 2 (Table I). However, allowing for customization of processes in response to the environment in which the process is to be used, such processes associate with category 3 (e.g., if evolutionary computation algorithms are used to perform the customization) or category 4 (e.g., if heuristic/deterministic algorithms are used to perform the customization).

6) *Construction*: Another characteristic that can distinguish processes is the method by which they are constructed, in analogy to the generally accepted kinds of product design: innovative design, variant design, or redesign. *Generative* methods develop new processes “from scratch” based on prior knowledge. *Variant* methods develop processes from existent ones by modifying existent processes. *Reverse-engineered* processes are those developed by dissecting existent processes, usually to address identified shortcomings. We consider all three of these methods as category 4 (Table I) because the development of the new process is primarily oriented to goal satisfaction. Additionally, instances of the variant and reverse-engineered methods may also be category 3 if they depend on evolutionary analogies.

7) *Evolution*: One may also characterize processes by the way they change over time. The basic division in this case is between changes in *procedural elements*—the tasks and activities that occur in the process—and changes in the nature of the *information objects* used by those activities and tasks. Since both procedural elements and information elements can be treated as design objects, this characteristic associates with category 2 (Table I). If the specific changes over time take advantage of the evolutionary analogy, then such processes belong to category 3.

The above-defined elements will make up the library of constructs proposed in this paper. This library will be an important element of a product design engineering cyber infrastructure.

B. Selection of CI Methods

The typology presented in Section III-A can be used to guide the selection of CI methods. For example, consider the following hypothetical company.

- 1) The company is well established, with significant corporate design knowledge stored in various conventional databases, e.g., CAD database.
- 2) The company is a consumer goods producer with small profit margins, and it cannot afford substantive process overhaul. That is, the company prefers small and continuous process improvements.
- 3) The company designs a broad range of consumer products under a single brand.
- 4) The company has a number of distinct divisions, but there is a significant movement of personnel between the divi-

sions. This suggests a preference for design processes that can vary between divisions but that has a common base to leverage worker expertise.

- 5) The company has developed pockets of procedural expertise that are not systematically connected. The company plans using this expertise for process improvements.
- 6) There are no known serious process problems in the company, but process effectiveness is noted as slowly decreasing.
- 7) The company’s structure includes fairly independent groups. The interactions between the groups are defined by the corporate leadership.

One might identify the company as seeking *gradual, platform-based modularity* using *variant* design methods, *mass customization*, and *standardized interfaces* to address the identified process problems. Solutions will be based on changes to *procedural elements* because of procedural expertise being exchanged between groups with movements of personnel.

Based on this description, one may then consider various CI methods based on the mapping between process characteristics and the categories outlined earlier. The hypothesis is that the CI methods identified in this way would be best suited for use by the company.

For example, one could propose a CBR system to support this company. CBR works best with a broad range of slightly similar products, and can help discover new variations on existent products. The movement of personnel between divisions provides a “vector” to distribute new knowledge and tools (such as CBR) throughout the company, so a phased implementation seems possible. Furthermore, the “pockets” of expertise suggests that a *knowledge-acquisition* system with *data mining support* could be useful to pull the knowledge from the “pockets” and eventually redistribute it to other workers.

Clearly, not enough information is presented here to allow a detailed and reliable selection of specific CI methods for specific companies. However, we believe we have shown the potential of this approach. Indeed, one can envision a three-dimensional matrix aligning company characteristics (as outlined in the earlier list) on one axis, against process characteristics (e.g., platform-based modularity) on a second axis, and computational technologies on the third axis. One might then use the matrix as a guide to identify what technologies of CI might be proposed for specific industrial settings. An exploration of such a scheme is, however, beyond the scope of this paper and is deferred to a future publication.

C. Evolutionary Computation and Process Perspective

Process modeling involves two notions (see Fig. 3).

- 1) Horizontal.
- 2) Vertical.

A process model is seldom developed at one level rather it is built vertically and horizontally. The top node in the hierarchy denotes the overall process that is decomposed into lower level components. The most granular model is usually a network of activities (the horizontal notion).

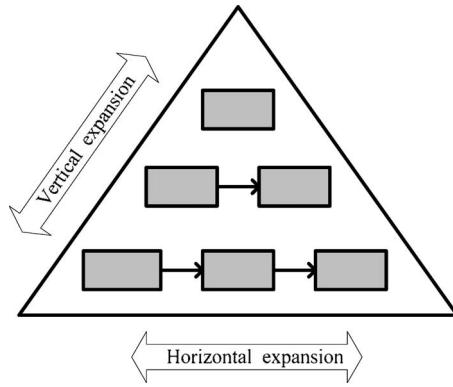


Fig. 3. Horizontal and vertical expansion of a process model.

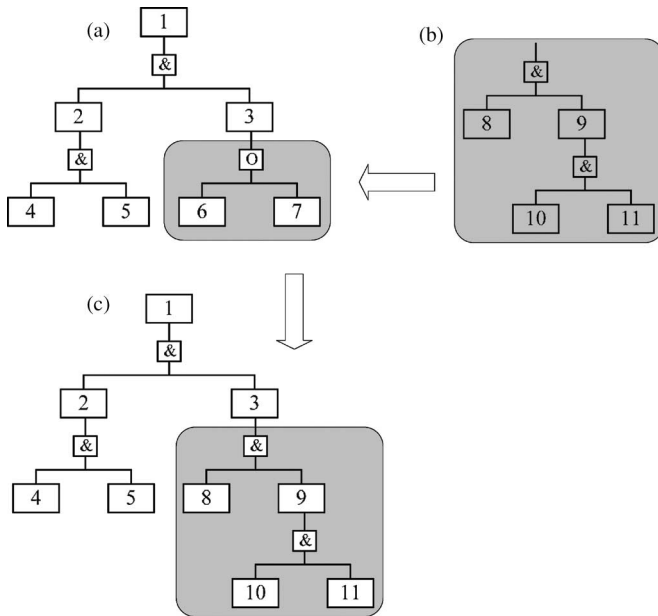


Fig. 4. Illustration of the crossover operator in a process model.

To support the horizontal notion of process modeling concepts from evolutionary computation will be applied. The feasibility of applying evolutionary computation, in particular GP is illustrated in Fig. 4. The crossover operator applied to the process model in Fig. 4(a) produces the model in Fig. 4(c) by using the submodel in Fig. 4(b).

The crossover operator demonstrated in Fig. 4 is one of many operators defined in GP that can be applied in product design engineering (e.g., see the operators defined in [68]).

The evolutionary computation concepts can be applied to support the horizontal notion of process modeling. The use of evolutionary computation in horizontal process modeling is illustrated with the following three activity operators.

- 1) Specialize.
- 2) Generalize.
- 3) Mutate.

To demonstrate these operators, consider the model in Fig. 5(a). The generalization operator transforms the model in Fig. 5(a) in the model in Fig. 5(b) by incorporating activity 5.

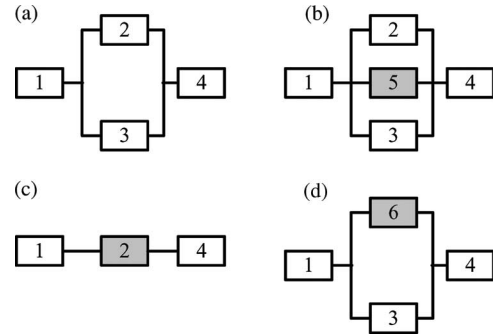


Fig. 5. Activity model operators. (a) Reference model. (b) Generalization. (c) Specialization. (d) Mutation.

Similarly, the specialization and mutation operations are illustrated in Fig. 5(c) and (d).

In addition to the activity operators, algebra for inputs, outputs, controls, mechanisms, and logical connectors can be defined. For example, the generalize operator applied to an EXCLUSIVE OR connector would transform it into an OR-connector.

One of the tools that can contribute to increasing the autonomy of process models is data mining. For example, a decision rule derived by a data mining algorithm may select in the model in Fig. 5(a) the path {1–2–3} based on real-time data.

The subprocess in Fig. 5(a) could be a fragment of the active Web search process. At present, the information on the Web is organized and largely searched hierarchically. In the near future, the process of retrieving Web information will be active. One way to make the Web active is to introduce process models that would adapt depending on the arising conditions, e.g., changing user's profile, the domain search frequency, changing the domain content. Data mining agents could track and increase the adaptability of the search process of various design libraries and repositories.

IV. PROBLEM VIEW OF PRODUCT DESIGN ENGINEERING

The review of the literature points to numerous topics of interest to both research and industry. The most urgent topics include:

- 1) innovation;
- 2) conceptual design;
- 3) standardization;
- 4) modularity;
- 5) design of product families;
- 6) design complexity management;

The relationship between each of these topics and the taxonomy presented in Section II is discussed next.

A. Innovation

The term *innovation* is widely used in a broad range of settings; however, analysis of the engineering literature indicates that the knowledge about the underlying science of innovation is limited.

Innovations in any domain can be enhanced by principles and insights from disparate disciplines. However, the process of

identifying the linkages between the disparate disciplines and the target domain is not well developed [69].

There are three basic approaches to innovate: structured, creative, and dynamic, producing either a sustaining or a disruptive product [70]. Structured innovation spawned during the industrial era was engineered to be highly efficient and replicable by innovating within set guidelines. It has been primarily used in large corporations, and it emphasizes internal leadership, strategic planning, effective execution of ideas, shareholder pressure, and financial resources more than other approaches, while placing less emphasis on a creative environment [71]. Creative innovation thrives more often in small organizations where focusing on the “big picture” is easier accomplished [70], [72]. The greatest advantage to the creative approach is the process itself. Dynamic innovation is a blend of both the structure and creative innovation approaches. Businesses of all sizes, from small to large, have used the dynamic approach to produce successful innovations. Dynamic innovation has taken on the aspects of structured innovation that embody strategic thinking and planning, along with the need for execution of projects [71].

Sustaining innovations are built off previous innovations [70], e.g., the personal digital assistant (PDA). The PDA has been an innovative and successful device, however, its predecessor the Apple Newton (a disruptive innovation) has failed. Sustaining innovations tend to be more successful than the disruptive ones. The sustaining innovation follows the incumbent, and therefore, it is easier to develop and market.

Several tools have been developed in support of innovation in engineering, including TRIZ—a Russian algorithm for theory of inventive problem solving [73], the Osborn/Parnes creative problem solving (CPS) process [74], and the innovation technology (IvT) approach [75].

TRIZ was developed to aid innovation by studying the patterns of problems and solutions, rather than relying on the spontaneous creativity of individuals or groups. This is done by focusing on a problem in its basic form while simultaneously understanding that this problem is rarely the actual problem to be solved. TRIZ handles three basic problems: the technical conflict and physical contradiction problem in which a solution creates another problem; the inventive problem where before a problem is solved, the solution of the conflict must be resolved; and the creation of the ideal machine/process in which something simplistic is constructed from a concept [76].

The CPS [74] is an “as-needed” problem solver for a generation of innovative solutions. The process greatly increases the chances of creating useful and unique solutions to almost any problem applied by groups or individuals. During the working process, combining convergent and divergent thinking is used to generate numerous potential solutions, while the user imagination is used freely to aid in the creation of innovative and working solutions.

Another approach used by engineers is the innovation technology is the IvT approach [75]. It involves various tools for problem solving, e.g., modeling, simulation, virtual reality, data mining, artificial intelligence, rapid prototyping, high throughput chemistry, and high throughput screening.

Other innovation tools include CREAX [77], Visual Mind [78], and Pull Thinking [79].

GAs have been used to design new electronic circuits. In some cases, GAs have outperformed the circuitry designed by humans [36], [80]. Some research, e.g., [81], suggests that GAs can help discover “innovative principles” of designing.

It is expected that in the near future, evolutionary computation algorithms will become embedded in software and integrated with other systems to support innovation. The ramification and use of the existing methodologies, e.g., group thinking and brainstorming, will be better understood, and new progressive methodologies will be developed.

Some of the drivers for the development of innovation science are as follows.

- 1) Innovation is the engine of the global economy, accounting for some 50% of the economic growth [82].
- 2) Innovation will mark the first economic revolution of the twenty first century [72]. Innovation involves almost all aspects of life, yet the innovation process is not well understood.
- 3) Innovation applies to the creation of methods used in industry, including the design of consumer goods and services.
- 4) The increasing complexity of technologies, their interdependencies, and the rapidly expanding volume of data call for a paradigm shift to be led by innovation.
- 5) Innovation clearly belongs to categories 2 and 3 (Table I).

B. Conceptual Design

Conceptual design is the early stage of design where general notions of a product are developed. To date, no computer-based system has been developed that can actually *perform* conceptual design. However, there are a number of systems that have been proposed to *assist* human designers in this task. These systems generally involve knowledge management in the areas of expected and desired function and behavior of products. Conceptual design falls in category 2 (Table I).

Zhang and Muehlenbein [22] developed a graph and matrix representation for the functional design of mechanical products. The system assists in performing design tasks that involve reasoning about function and behavior of products. Berrais [83] reported on a KBS that is used as an interactive design tool for all stages of design and analysis of earthquake-resistant reinforced concrete buildings, paying special attention to the preliminary stages, by imposing a predefined design methodology. Lina and Farahati [84] developed a KBS for assembly design of blade and shell assemblies that focuses on the early stages of product development, before actual component shapes have been determined. Experimental validation of two cases indicates satisfactory results.

Parmee and Bonham [85] used evolutionary techniques for quick identification of regions of complex design spaces that contain high-performance solutions. The results of such searches stimulate human designers in an iterative process of solution space refinement. Test results indicate that such an approach can stimulate innovation. Zavbi and Duhovnik [86]

developed a KBS using physical laws to identify key behaviors in technical systems and assists designers in establishing behavioral models of products from their expected functions. The system uses the analytical hierarchy process methodology to select among possible physical laws, and a prescriptive design process. Cvetkovic and Parmee [87] used several types of agents (search agents, interface agents, and information agents) to develop an evolutionary conceptual design system. A special type of agent to capture preference was also developed to account for qualitative and experiential knowledge of the designers.

Ming [88] developed a computer-based system using inductive learning to semiautomate concept design tasks. Ling *et al.* [89] proposed CBR to represent function spaces. Kryssanov *et al.* [90] suggested that semiotics could significantly improve our understanding of the creative process of designing and developing an applicable computational theory.

C. Standardization

Standardization is the activity of developing uniform products or product components that can be used in different settings. Standardization of products and components has been discussed in the literature from different viewpoints. Because standardization regards goal attainment and optimization (e.g., lowering part counts, increasing production runs) falls in category 4 (Table I).

Tarondeau [91] discussed the impact of standardization on the number of components, the number of reference points to be managed, and the manufacturing complexity. Lee and Tang [92] developed a model optimizing the tradeoff between the investment in standardization and the profit due to the economy of scale. Erol [93] proposed a mathematical formulation for the standardization of low-value components that was solved in [94].

Fouque [95] discussed different scenarios for the standardization of two components (C1 and C2) into one (C), namely, an increase in the service level of component C1 and/or C2, a decrease in the correlation between the demand for C1 and C2, an increase in the uncertainty of demand for C1 and/or C2, similar costs of the components C1 and C2, and a low demand for the two components. Standardization aggregates the risk and reduces the uncertainty of the standardized component C with respect to the uncertainty of each component C1 and C2. In addition, the level of in-process inventory may be reduced and the productivity and service level may increase [96].

Kota *et al.* [97] proposed a measure that captures the level of commonality in a product family, i.e., the potential of the part family to divide the elements and to reduce the total number of parts. This measure allows the comparison of design alternatives. Thonemann and Brandeau [98] presented an approach for determining optimum commonality among subproducts from the customer differentiation viewpoint. For highly diversified products, standardization is not the best solution.

D. Modularity

A way to design products for highly diversified requirements is to apply modular design methods. These methods are to direct

designers toward products including functional units (modules) that are often interchangeable and reusable over different products or product variants.

Modules imply standard interfaces allowing for their use across different products. To implement the modular design concept, it is necessary to partition a product into semi-independent or mutually separable elements. It then becomes possible to design, manufacture, and service the modules independently. The differentiation of products is accomplished at the assembly stage by the selection of modules and their location in a product [99].

Kusiak and Huang [100] and Huang and Kusiak [101] discussed modular design aimed at the production of a wide variety of products at low cost. A matrix representation of the product allowed the identification of modules sharing different characteristics. Numerous applications of the product modular concept are presented in [67] and the recent product design engineering literature.

Product flexibility and the use of common components across various products are important in modular design [102]. The flexibility of a module (the number of its uses) depends on its functionality and the required standard interfaces.

Prior research on modular design has emphasized consistency of the design process and manufacturing. For example, the taboo search algorithm presented in [94] aimed at the design of an assembly system for modular products. A modular design methodology intended to produce a large variety of products at a low cost is discussed in [103]. Other examples of modular concepts are presented in [102].

Modular design leads to a large number of different products using a limited number of modular components. One aspect of product modularity, the design product families, has been discussed in [103]–[109].

Modular process design with CI has been pursued in disciplines where processes are inherent elements of products, such as in chemical and electrical engineering. Byrne and Bogle [110] used optimization methods to design modular chemical plant process flowsheets. Similarly, Smayling *et al.* [111] developed modularized process elements to automate the design of electronic components.

Goel and Bhatta [112] used *design patterns* as starting points to modularize design activities involving analogical reasoning, and develop computable models of limited domains based on their approach. Fensel *et al.* [113] reported on their Unified Problem-Solving Method Description Language (UPML), which used modularized process elements to implement reusable methods, applied to simple design problems.

Modularity is associated with category 4 (Table I).

E. Design of Product Families

While *modularity* (discussed earlier) treats the identification, specification, and design of modules, a separate issue—design of product families—builds upon modularity concepts, taking a more holistic perspective on product development. Like modular design, product family belongs to category 4 (Table I).

To meet diversified product requirements, numerous strategies are available [114]. It is conceivable that a standardized

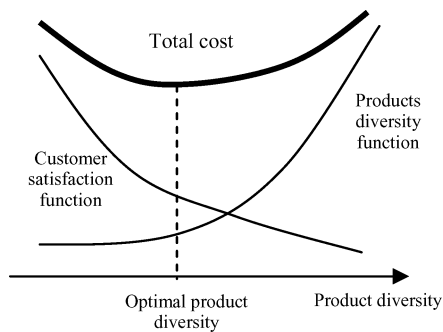


Fig. 6. Product diversity cost [91].

product would satisfy many customers, as well as the requirements of a single customer. A cost-based compromise between these two strategies is of interest. Therefore, single and multi-objective models are of interest. Fellini *et al.* [115] addressed performance losses of product families with respect to individually designed products, arising from commonality constraints. This is done by a user-specified performance loss tolerance on an optimization of choice of components. Seepersad *et al.* [116] based their multiobjective approach on a utility-based compromise decision-support model.

Du *et al.* [117] used graph-rewriting techniques to create hierarchies of graph schema for different product families, which can provide an interactive environment for customers to make choices among product offerings. Siddique and Rosen [118] and Corbett and Rosen [119] developed constraint-based methods for combining design configuration spaces that model design requirements for physical connections, module partitions, and assembly sequences for product families. They also presented a new designer-guided method, called the partitioning method, for decomposing configuration design problems hierarchically to enable significant reductions in design space sizes.

The delayed product differentiation concept implies delaying the point of differentiation of the product or the process (in which a product acquires its identity) [92]. The goal of the delayed product differentiation is to maximize the use of standard elements and to push back, to the latest time possible, the point when each product differs from another. Some authors, e.g., He *et al.* [120], used the term postponement as a synonym of delayed differentiation.

F. Design Complexity Management

To meet the customer needs, product diversity tends to grow over time, and therefore, a suitable management strategy is needed. The cost of offering a large portfolio of products should not exceed gains obtained by satisfying the customer needs. It is essential to determine the level of diversity that minimizes the total cost (see Fig. 6).

The major challenge is how to offer a large diversity of products while managing a limited diversity of components. Different approaches have been used to address this challenge, e.g., standardization, modular design, design of product families, and product delayed differentiation. Assemble-to-order is a policy that links modular design and product delayed differentiation.

According to this policy, modules are built from basic parts and stocked, the final assembly is done after an order has been confirmed.

The large apparent diversity for the customers is enabled by a combinatorial association of basic parts.

The major component of diversity is not visible to the customers. It is actually created by the evolution of components (changes in technology) or the creation of new versions (product upgrades).

V. CONCLUSION

This paper has surveyed recent literature and generalized emerging concepts of CI in product design engineering. The research covered in this paper is being vigorously pursued and no survey as broad as this one could be complete. Rather than considering all papers published, a representative “slice” of recent research has been described. The reviewed literature indicates certain trends that are briefly summarized next.

Some recent research combined multiple approaches to develop new tools. As the new techniques become better understood (e.g., CBR), they are used as building blocks upon which more powerful systems are constructed. This is a characteristic of the area that was not evident a decade ago.

Another somewhat paradoxical trend that can be observed is a tendency to include a human in the “loop” of the intelligent system. This may be an indication that developing totally autonomous and thinking software is not feasible (at least, given the current understanding of the computing science and the human mind). Usable intelligent systems of systems involving humans are likely to emerge in time.

A third trend is the emergence of the WWW as a component of the CI landscape. Whether by using Semantic Web technologies or just using browsers as user interface tools, the Web continues to be a growing application platform.

The role of data, data analysis, knowledge extraction, and knowledge management in product design engineering is gaining momentum. As sufficient volume of information surrounding the design process will be captured, design may become process driven. Dynamically induced knowledge and models could guide the design of innovative artifacts.

Another conclusion one may draw is that there appears to be some correlation between characteristics of corporate settings and the kinds of CI tools that could be most beneficial in those settings (see Section III-C). The authors have not gathered “hard” data on this matter, but instead, we suggest that it may be a fruitful avenue for future work.

Finally, evolutionary computation algorithms could pave way toward systems supporting many of the frameworks reviewed in this paper, including increased automation and enhanced innovation in product design engineering.

REFERENCES

- [1] T. Gruber, “Ontolingua: A mechanism to support portable ontologies,” Knowl. Syst. Lab., Stanford Univ., Stanford, CA, Tech. Rep. KSL91-66, Nov. 1992.
- [2] D. W. Fowler, D. Sleeman, G. Wills, T. Lyon, and D. Knott, “The designers’ workbench: Using ontologies and constraints for configuration,” in

- Proc. 24th SGA Int. Conf. Innovative Tech. Appl. Artif. Intell.*, Cambridge, U.K., 2004, pp. 209–221.
- [3] M. Yoshioka, Y. Umeda, H. Takeda, Y. Shimomura, Y. Nomaguchi, and T. Tomiyama, "Physical concept ontology for the knowledge intensive engineering framework," *Adv. Eng. Inf.*, vol. 18, no. 2, pp. 95–113, 2004.
 - [4] Y. Kitamura, Y. Kashiwase, M. Fuse, and R. Mizoguchi, "Deployment of an ontological framework of functional design knowledge," *Adv. Eng. Inf.*, vol. 18, no. 2, pp. 115–127, 2004.
 - [5] D. Brown, D. Leal, C. McMahon, R. Crossland, and J. Devlukia, "A web-enabled virtual repository for supporting distributed automotive component development," *Adv. Eng. Inf.*, vol. 18, no. 3, pp. 173–190, 2004.
 - [6] F. Tao, L. Chen, S. J. Cox, N. R. Shadbolt, C. Puleston, and C. Goble, "Semantic support for grid-enabled design search in engineering," presented at the GGF9 Semantic Grid Workshop, Chicago, IL, 2003.
 - [7] D. Tormey, O. Chira, C. Chira, A. Brennan, and T. Roche, "The use of ontologies for defining collaborative design processes," presented at the 32nd Int. Conf. Comput. Ind. Eng., Limerick, Ireland, 2003.
 - [8] M. Gruninger, K. Atefi, and M. S. Fox, "Ontologies to support process integration in enterprise engineering," *Comput. Math. Org. Theory*, vol. 6, no. 4, pp. 381–394, 2000.
 - [9] K. S. Lee and C. Luo, "Application of case-based reasoning in die-casting die design," *Int. J. Adv. Manuf. Technol.*, vol. 20, no. 4, pp. 284–295, 2002.
 - [10] A. Kusiak, "Data mining: Manufacturing and service applications," *Int. J. Prod. Res.*, vol. 44, no. 18/19, pp. 4175–4191, 2006.
 - [11] Y. Ishino and Y. Jin, "Data mining and knowledge acquisition in engineering design," in *Data Mining for Design and Manufacturing: Methods and Applications*, D. Braha, Ed. Norwell, MA: Kluwer, 2001, pp. 145–160.
 - [12] C. J. Romanowski and R. Nagi, "A data mining for knowledge acquisition in engineering design," in *Data Mining for Design and Manufacturing: Methods and Applications*, D. Braha, Ed. Norwell, MA: Kluwer, 2001, pp. 161–178.
 - [13] M. D. Giess, S. J. Culley, and A. Shepherd, "Informing design using data mining methods," in *Proc. ASME DETC Conf.*, Montreal, Canada, 2002, pp. 207–215.
 - [14] L. Hamburg, "Improving computer supported environment friendly product development by analysis of data," presented at the 2nd Eur. Conf. Intell. Syst. Technol. Conf., Iasi, Romania, 2004.
 - [15] A. Albers and J. Marz, "Restrictions of production on micro-specific product development," *Microsyst. Technol.*, vol. 10, no. 3, pp. 205–210, 2004.
 - [16] G. Cascint, "Plastics design: Integrating TRIZ creativity and semantic knowledge portals," *J. Eng. Des.*, vol. 15, no. 4, pp. 405–424, 2004.
 - [17] J. P. Terpenney, S. Strong, and J. Wang, "A methodology for knowledge discovery and classification," in *Proc. 10th Flexible Autom. Intell. Manuf. Conf.*, College Park, MD, Jun. 2000, pp. 22–32.
 - [18] P. J. Bentley and J. P. Wakefield, "The table: An illustration of evolutionary design using genetic algorithms," in *Proc. 1st IEEE/IEEE Conf. Genetic Algorithms Eng. Syst.: Innovations Appl.*, Sep. 1995, pp. 412–418.
 - [19] S.-B. Cho, "Towards creative evolutionary systems with interactive genetic algorithm," *Appl. Intell.*, vol. 16, no. 2, pp. 129–138, 2002.
 - [20] Z. Koza, *Genetic Programming*. Cambridge, MA: MIT Press, 1992.
 - [21] W. Benzhaf, P. Nordin, R. E. Keller, and F. D. Francone, *Genetic Programming: An Introduction*. San Francisco, CA: Morgan Kaufmann, 1998.
 - [22] B.-T. Zhang and H. Muehlenbein, "Balancing accuracy and parsimony in genetic programming," *Evol. Comput.*, vol. 3, no. 1, pp. 17–38, 1995.
 - [23] H. Iba, H. deGaris, and T. Sato, "A numerical approach to genetic programming for system identification," *Evol. Comput.*, vol. 3, no. 4, pp. 417–452, 1995.
 - [24] T. Soule and J. A. Foster, "Effects of code growth and parsimony pressure on populations in genetic programming," *Evol. Comput.*, vol. 6, no. 4, pp. 293–309, 1998.
 - [25] X. Yao, Y. Liu, and G. Lin, "Evolutionary programming made easier," *IEEE Trans. Evol. Comput.*, vol. 3, no. 2, pp. 82–102, Jul. 1999.
 - [26] B. Forouraghi, "A genetic algorithm for multiobjective robust design," *Appl. Intell.*, vol. 12, no. 3, pp. 151–161, 2000.
 - [27] C. M. Fonseca and P. J. Fleming, "An overview of evolutionary algorithms in multiobjective optimization," *Evol. Comput.*, vol. 3, no. 1, pp. 1–16, 1995.
 - [28] T. Bäck, *Evolutionary Algorithms in Theory and Practice*. New York: Oxford Univ. Press, 1996.
 - [29] C. A. C. Coello, "A comprehensive survey of evolutionary-based multiobjective optimization techniques," *Knowl. Inf. Syst.*, vol. 1, no. 3, pp. 269–308, 1999.
 - [30] D. A. Van Veldhuizen and G. B. Lemont, "Multiobjective evolutionary algorithms: Analyzing the state-of-the-art," *Evol. Comput.*, vol. 8, no. 2, pp. 125–147, 2000.
 - [31] J. Horn, D. E. Goldberg, and K. Deb, "Implicit niching in a learning classifier system: Nature's way," *Evol. Comput.*, vol. 2, no. 1, pp. 37–66, 1994.
 - [32] D. E. Moriarty and R. Miikkulainen, "Forming neural networks through efficient and adaptive coevolution," *Evol. Comput.*, vol. 5, no. 4, pp. 373–399, 1998.
 - [33] A. E. Eiben and T. Bäck, "Empirical investigation of multiparent recombination operators in evolution strategies," *Comput. Intell.*, vol. 5, no. 3, pp. 347–365, 1997.
 - [34] S. Olafsson, "Resource allocation as an evolving strategy," *Evol. Comput.*, vol. 4, no. 1, pp. 33–55, 1996.
 - [35] F. Schweitzer, H. Rosé, W. Ebeling, and O. Weiss, "Optimization of road networks using evolutionary strategies," *Evol. Comput.*, vol. 5, no. 4, pp. 419–438, 1997.
 - [36] A. Thompson, P. Layzell, and R. S. Zebulum, "Exploration in design space: Unconventional electronics design through artificial evolution," *IEEE Trans. Evol. Comput.*, vol. 3, no. 2, pp. 167–196, Sep. 1999.
 - [37] J. D. Lohn and S. P. Colombano, "A circuit representation technique for automated circuit design," *IEEE Trans. Evol. Comput.*, vol. 3, no. 3, pp. 205–219, Sep. 1999.
 - [38] S. K. Sim and Y. W. Chan, "A knowledge-based expert system for rolling-element bearing selection in mechanical engineering design," *Artif. Intell. Eng.*, vol. 6, no. 3, pp. 125–135, 1992.
 - [39] M. Stacey, P. J. Clarkson, and C. Eckert, "Signposting: An AI approach to supporting human decision making in design," presented at the ASME Comput. Eng. Conf., Baltimore, MD, 2000, Paper CIE-14617.
 - [40] M. R. Danesh and Y. Jin, "An agent-based decision network for concurrent engineering design," *Concurrent Eng.: Res. Appl.*, vol. 9, no. 1, pp. 37–47, 2001.
 - [41] K.-M. Chao, C. Laing, R. Anane, M. Younas, and P. Norman, "Multiple evolutionary agents for decision support," *Trans. Soc. Des. Process Sci.*, vol. 7, no. 2, pp. 39–56, 2003.
 - [42] S. Mussi, "Putting value of information theory into practice: A methodology for building sequential decision support systems," *Expert Syst.*, vol. 21, no. 2, pp. 92–103, 2004.
 - [43] W. Scott and S. C. Cook, "A requirements assessment architecture that combines natural language parsing and artificial intelligence," presented at the 14th Annu. Symp. Int. Council Syst. Eng., Toulouse, France, 2004, Paper 6.1.3.
 - [44] G. Morcous, H. Rivard, and A. M. Hanna, "Case-based reasoning system for modeling infrastructure deterioration," *J. Comput. Civil Eng.*, vol. 16, no. 2, pp. 104–114, 2002.
 - [45] H. Rivard and S. J. Fennes, "SEED-Config: A case-based reasoning system for conceptual building design," *AIEDAM*, vol. 14, pp. 415–430, 2000.
 - [46] D. B. Leake and D. C. Wilson, "A case-based framework for interactive capture and reuse of design knowledge," *Appl. Intell.*, vol. 14, no. 1, pp. 77–94, 2001.
 - [47] S. B. Tor, G. A. Britton, and W. Y. Zhang, "Indexing and retrieval in metal stamping die design using case-based reasoning," *ASME Trans.: J. Comput. Inf. Sci. Eng.*, vol. 3, no. 4, pp. 353–362, 2003.
 - [48] X. Qin and W. C. Regli, "A study in applying case-based reasoning to engineering design: Mechanical bearing design," *AIEDAM*, vol. 17, no. 3, pp. 235–252, 2003.
 - [49] C. M. Vong, T. P. Leung, and P. K. Wong, "Case-based reasoning and adaptation in hydraulic production machine design," *Eng. Appl. Artif. Intell.*, vol. 15, no. 6, pp. 567–585, 2002.
 - [50] M.-L. Chiu, "Design moves in situated design with case-based reasoning," *Des. Stud.*, vol. 24, no. 1, pp. 1–25, 2003.
 - [51] D. S. Weld and J. de Kleer, Eds., *Readings in Qualitative Reasoning About Physical Systems*. Los Altos, CA: Morgan Kaufmann, 1989.
 - [52] A. Kusiak and Z. Song, "Combustion efficiency optimization and virtual testing: A data-mining approach," *IEEE Trans. Ind. Inf.*, vol. 2, no. 3, pp. 176–184, Aug. 2006.
 - [53] D. C. Karnopp, D. L. Margolis, and R. C. Rosenberg, *System Dynamics: A Unified Approach*. New York: Wiley, 1990.
 - [54] A. Zakarian and A. Kusiak, "Analysis of process models," *IEEE Trans. Electron. Packag. Manuf.*, vol. 23, no. 2, pp. 137–147, Apr. 2000.
 - [55] Y. Pisan. (1998). An integrated architecture for engineering problem solving. Ph.D. dissertation, Northwestern Univ. Evanston, IL [Online]. Available: <http://www-staff.it.uts.edu.au/ypisan/research/publications/thesis/index.html>.

- [56] N. Sgouros, "Interaction between physical and design knowledge in design from physical principles," *Eng. Appl. Artif. Intell.*, vol. 11, pp. 449–459, 1998.
- [57] T. F. Stahovich, R. Davis, and H. Shrobe, "Qualitative rigid-body dynamics," *Artif. Intell.*, vol. 119, no. 1/2, pp. 19–60, 2000.
- [58] K. W. Chau and F. Albermani, "Hybrid knowledge representation in a blackboard KBS for liquid retaining structure design," *Eng. Appl. Artif. Intell.*, vol. 17, no. 1, pp. 11–18, 2004.
- [59] Z. Lou, H. Jiang, and X. Ruan, "Development of an integrated knowledge-based system for mold-base design," *J. Mater. Process. Technol.*, vol. 150, no. 1/2, pp. 194–199, 2004.
- [60] W. Y. Zhang, S. B. Tor, and G. A. Britton, "A graph and matrix representation scheme for functional design of mechanical products," *Int. J. Adv. Manuf. Technol.*, vol. 25, no. 3/4, pp. 221–232, 2005.
- [61] L. Zadeh, "A fuzzy-algorithmic approach to the definition of complex or imprecise concepts," *Int. J. Man-Mach. Stud.*, vol. 8, pp. 249–291, 1976.
- [62] F. O. Karray and C. W. De Silva, *Soft Computing and Intelligent Systems Design: Theory, Tools and Applications*. New York: Addison-Wesley, 2004.
- [63] Ö. Nursel, "Use of genetic algorithm to design optimal neural network structure," *Eng. Comput.*, vol. 20, no. 8, pp. 979–997, 2003.
- [64] M. Rosenman, "Case-based evolutionary design," *AIEDAM*, vol. 14, no. 1, pp. 17–29, 2000.
- [65] F. T. S. Chan, B. Jiang, and N. K. H. Tang, "Development of intelligent decision support tools to aid the design of flexible manufacturing systems," *Int. J. Prod. Econ.*, vol. 65, no. 1, pp. 73–84, 2000.
- [66] G. Schuh, W. Ley, M. P. Gruenenfelder, and A. P. Hofer, "The potential for product family management based on product platform concepts," in *Design for Excellence*, S. Sivaganathan and P. T. Andrews, Eds. London, U.K.: Professional Engineering, 2000, pp. 601–611.
- [67] A. Kusiak, *Engineering Design: Products, Processes and Systems*. San Diego, CA: Academic, 1999.
- [68] R. S. Hawley and C. A. Mori, *The Human Genome: A User's Guide*. San Diego, CA: Academic, 1999.
- [69] R. N. Kostoff, "Role of technical literature in science and technology development and exploitation," *J. Inf. Sci.*, vol. 29, no. 3, pp. 223–228, 2003.
- [70] K. Allen, *Bringing New Technology to Market*. Upper Saddle River, NJ: Prentice Hall, 2003.
- [71] A. Kusiak, "Innovation science: a primer," *Int. J. of Computer Applications in Technology*, vol. 28, no. 2–3, pp. 140–149, 2007.
- [72] A. Kusiak and C.-Y. Tang, "Innovation in a requirement life-cycle framework," in *Proc. 5th Int. Symp. Intell. Manuf. Syst. (IMS'2006)*, Sakarya, Turkey, pp. 61–67.
- [73] A. Kusiak and M. Smith, "Data mining in design of products and production systems," *IFAC Annual Reviews in Control*, vol. 31, no. 1, pp. 147–156, 2007.
- [74] D. Dauert. (2005). The Osborne–Parnes creative problem solving process manual, [Online]. Available: <http://www.ideastream.com/create>.
- [75] "Dodgeson, Gann and Salter Report," Presented at the DRUID Summer Conf. Ind. Dyn., Innovation Dev., Elsinore, Denmark, 2004.
- [76] P. Siem, "An introduction to TRIZ: A revolutionary new product development tool," *Visions*, Jan. 1996.
- [77] CREAX Innovation Suite 3.1 [Online]. Available: <http://www.creax.com/tools.htm>.
- [78] Visual Mind [Online]. Available: <http://www.visual-mind.com>.
- [79] Pull Thinking, [Online]. Available: <http://www.pullthinking.com>.
- [80] S. Ando, M. Ishizuka, and H. Iba, "Evolving analog circuits by variable length chromosomes," in *Advances in Evolutionary Computing: Theory and Applications*. New York: Springer-Verlag, pp. 643–662.
- [81] K. Deb, "Unveiling innovative design principles by means of multiple conflicting objectives," *Eng. Optim.*, vol. 35, no. 5, pp. 445–470, 2003.
- [82] "Innovate America," National Innovation Initiative Report, Council for Competitiveness. Washington, DC, 2004.
- [83] A. Berrais, "A knowledge-based expert system for earthquake resistant design of reinforced concrete buildings," *Expert Syst. Appl.*, vol. 28, no. 3, pp. 519–530, 2005.
- [84] Y. J. Lina and R. Farahati, "Optimum assembly design utilizing a behavioral modeling concept," *Assem. Autom.*, vol. 23, no. 2, pp. 181–191, 2000.
- [85] I. C. Parmee and C. R. Bonham, "Towards the support of innovative conceptual design through interactive designer/evolutionary computing strategies," *AIEDAM*, vol. 14, no. 1, pp. 3–16, 2000.
- [86] R. Zavbi and J. Duhovnik, "Conceptual design of technical systems using functions and physical laws," *AIEDAM*, vol. 14, no. 1, pp. 69–83, 2000.
- [87] D. Cvetkovic and I. Parmee, "Agent-based support within an interactive evolutionary design system," *AIEDAM*, vol. 16, no. 5, pp. 331–342, 2002.
- [88] X. T. Ming, "Inductive learning techniques in design process: A design concept learning system," *Integr. Comput.-Aided Eng.*, vol. 8, no. 2, pp. 171–186, 2001.
- [89] W. Ling, J. Yan, J. Wang, and Y. Xie, "Case-based conceptual design," *Chin. J. Mech. Eng. (English Edition)*, vol. 17, no. 1, pp. 73–77, 2004.
- [90] V. V. Kryssanov, H. Tamaki, and S. Kitamura, "Understanding design fundamentals: How synthesis and analysis drive creativity, resulting in emergence," *Artif. Intell. Eng.*, vol. 15, no. 4, pp. 329–342, 2001.
- [91] J. C. Tarondeau, *Stratégie Industrielle*, 2nd ed. Vuibert, France: Collection Gestion, 1998.
- [92] H. L. Lee and C. S. Tang, "Modeling the costs and benefits of delayed product differentiation," *Manage. Sci.*, vol. 43, no. 1, pp. 40–53, 1997.
- [93] M. Erol, "Prise en compte de la flexibilité dans la planification dynamique" Ph.D. dissertation, Inst. Nat. Polytech. Grenoble, Grenoble, France, 1999.
- [94] L. Dupont, M. Erol, G. Cormier, and N. Turkkan, "La standardisation des composants: modèles et algorithmes," in *Proc. 3ème Congr. Int. Génie Ind.*, Montreal, Canada, May 1999, pp. 671–680.
- [95] T. Fouque, "A la recherche des produits flexibles," *Rev. Française Gestion*, no. 123, pp. 80–87, Mar.–May 1999.
- [96] L. Dupont, *La Gestion Industrielle: Concepts et Outils*. Paris, France: Hermès, 1998.
- [97] S. Kota, K. Sethuraman, and R. Miller, "A metric for evaluating design commonality in product families," *ASME Trans.: J. Mech. Des.*, vol. 122, pp. 403–410, 2000.
- [98] U. W. Thonemann and M. Brandeau, "Optimal commonality in component design," *Oper. Res.*, vol. 48, no. 1, pp. 1–19, 2000.
- [99] B. Agard and A. Kusiak, "Data mining for subassembly selection," *ASME Trans.: J. Manuf. Sci. Eng.*, vol. 126, no. 3, pp. 627–631, 2004.
- [100] A. Kusiak and C. C. Huang, "Development of modular products," *IEEE Trans. Compon., Packag., Manuf. Technol. A*, vol. 19, no. 4, pp. 523–538, Dec. 1996.
- [101] C. C. Huang and A. Kusiak, "Modularity in design of products and systems," *IEEE Trans. Syst., Man, Cybern. A Syst. Humans*, vol. 28, no. 1, pp. 66–77, Jan. 1998.
- [102] C. Gertosio and A. Dussauchoy, "Knowledge discovery from industrial databases," *J. Intell. Manuf.*, vol. 15, no. 1, pp. 29–37, 2004.
- [103] F. Erens and K. Verhulst, "Architectures for product families," *Comput. Ind.*, vol. 33, pp. 165–178, 1997.
- [104] M. V. Martin, "Design for variety: A methodology for developing product platform architectures" Ph.D. dissertation, Stanford Univ., Stanford, CA, 1999.
- [105] P. J. Newcomb, B. Bras, and D. W. Rosen, "Implications of modularity on product design for the life cycle," *ASME Trans.: J. Mech. Des.*, vol. 120, no. 3, pp. 483–490, 1998.
- [106] T. W. Simpson, J. R. A. Maier, and F. Mistree, "Product platform design: Method and application," *Res. Eng. Des.*, vol. 13, no. 1, pp. 2–22, 2000.
- [107] J. B. Dahmus, J. P. Gonzalez Zugasti, and K. Otto, "Modular product architecture," *Des. Stud.*, vol. 22, pp. 409–424, 2001.
- [108] J. Gonzalez Zugasti, K. Otto, and J. Baker, "Assessing value for product family design and selection," in *Proc. 25th ASME Des. Autom. Conf.*, Las Vegas, NV, Sep. 1999, pp. 12–15.
- [109] J. Jiao and M. Tseng, "A methodology of developing product family architecture for mass customization," *J. Intell. Manuf.*, vol. 10, no. 1, pp. 3–20, 1999.
- [110] R. P. Byrne and I. D. L. Bogle, "Global optimization of modular process flowsheets," *Ind. Eng. Chem. Res.*, vol. 39, no. 11, pp. 4296–4301, 2000.
- [111] M. Smayling, J. Rodriguez, A. Young, and F. Ichiro, "Process synthesis using TCAD: A mixed-signal case study," *IEICE Trans. Electron.*, vol. E82C, no. 6, pp. 983–991, 1999.
- [112] A. K. Goel and S. R. Bhatta, "Use of design patterns in analogy-based design," *Adv. Eng. Inf.*, vol. 18, pp. 85–94, 2004.
- [113] D. Fensel, M. Musen, E. Plaza, G. Schreiber, R. Studer, B. Wielinga, E. Motta, F. van Harmelen, V. R. Benjamins, M. Crubezy, S. Decker, M. Gaspari, R. Groenboom, and W. Grosso, "The unified problem-solving method development language UPML," *Knowl. Inf. Syst.*, vol. 5, no. 1, pp. 83–131, 2003.
- [114] B. Agard and A. Kusiak, "A data-mining based methodology for the design of product families," *Int. J. Prod. Res.*, vol. 42, no. 15, pp. 2955–2969, 2004.
- [115] R. Fellini, M. Kokkolaras, P. Y. Papalambros, and A. Perez Duarte, "Platform selection under performance loss constraints in optimal design of product families," *ASME Des. Eng. Tech. Conf.*, 2002, Paper DETC2002/DAC-34099.

- [116] C. C. Seepersad, F. Mistree, and J. K. Allen (2002), "A quantitative approach for designing multiple product platforms for an evolving portfolio of products," presented at the ASME Des. Eng. Tech. Conf., Paper DETC2002/DAC-34096 [Online]. Available: <http://www.srl.gatech.edu/~carolyn/pdf/dac34096.detc02.pdf>.
- [117] X. Du, J. Jiao, and M. M. Tseng, "Product family modeling and design support: An approach based on graph rewriting systems," *AIEDAM*, vol. 16, no. 2, pp. 103–120, 2002.
- [118] Z. Siddique and D. W. Rosen, "On combinatorial design spaces for the configuration design of product families," *AIEDAM*, vol. 15, no. 2, pp. 91–108, 2001.
- [119] B. Corbett and D. W. Rosen, "A configuration design based method for platform commonization for product families," *AIEDAM*, vol. 18, no. 1, pp. 21–39, 2004.
- [120] D. W. He, A. Kusiak, and T. L. Tseng, "Delayed product differentiation: A design and manufacturing perspective," *Comput.-Aided Des.*, vol. 30, no. 2, pp. 105–113, 1998.



Andrew Kusiak (M'89) is a Professor in the Department of Mechanical and Industrial Engineering, University of Iowa, Iowa City. His current research interests include applications of computational intelligence in automation, energy, manufacturing, product development, and healthcare. He is the author or coauthor of numerous books and technical papers published in journals sponsored by professional societies, such as the Association for the Advancement of Artificial Intelligence, the American Society of Mechanical Engineers, the Institute of Industrial Engineers,

the International Federation of Information Processing, the International Federation of Automatic Control, the Institute for Operations Research and the Management, the International Society for Productivity Enhancement, and the Society of Manufacturing Engineers. He speaks frequently at international meetings, conducts professional seminars, and consults for industrial corporations. He has served on editorial boards of more than 30 journals. He is currently the Editor-in-Chief of the *Journal of Intelligent Manufacturing*.

Prof. Kusiak is a Fellow of the IIE.



Filippo A. Salustri (M'92) is an Associate Professor of mechanical engineering at Ryerson University, Toronto, ON, Canada. He has been teaching, researching, and practicing design engineering for 17 years. He has worked with Ford Motors, Spar Aerospace, and many other companies. His current research interests include formal and informal methods of designing, information visualization, and Web-based design tools.

He is a member of the Design Society, the Design Research Society, the American Society of Mechanical Engineers, the Canadian Society for Mechanical Engineering, and the International Council on Systems Engineering. He is a founding member of the Canadian Design Engineering Network.