Revised: May 23, 2006

Computational Intelligence in Product Design Engineering: Review and Trends

Andrew Kusiak[*] and Filippo A. Salustri[#]

[*]Intelligent Systems Laboratory
Department of Mechanical and Industrial Engineering
2139 Seamans Center
The University of Iowa
Iowa City, Iowa 52242 - 1527
andrew-kusiak@uiowa.edu

[#]Department of Mechanical and Industrial Engineering
Ryerson University
350 Victoria Street
Toronto, ON, M5B 2K3, Canada
salustri@ryerson.ca

Abstract

Product design engineering is undergoing a transformation from informal and largely experience-based discipline to a science-based domain. Computational intelligence offers models and algorithms that can contribute greatly to design formalization and automation. This paper surveys computational intelligence 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 computational intelligence. 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 computational intelligence concepts and approaches in product design engineering.

Keywords: Computational intelligence, engineering design, product engineering, decision making, design automation.

1. 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 results applied to product design engineering, and structure the computational intelligence approaches in a general unifying framework.

No survey of such a broad field can be complete. An attempt has been made to balance degree of detail against availability of literature sources. Any imbalance in the coverage is due to the availability of information rather than 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.

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

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

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, 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, 3, and 4 are examined.

2. Taxonomy of Computational Intelligence Algorithms, Techniques, and Tools

Many experts agree that computational intelligence (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 computational intelligence 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 1. Note that Category 1 of Fig. 1 has been excluded from Table 1 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 "x" was placed in the appropriate entry of Table 1.

Table 1. 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

The literature pertaining to each of the seven approaches of Table 1 is discussed in the sections that follow.

2.1 Ontologies

An *ontology* is an agreed upon set of terms and meanings that enables parties to share diverse knowledge through a common language (Gruber, 1992). "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 1).

Fowler *et al*. (2004) used ontologies of engineering features to develop software to check that product configurations satisfy both physical and organizational constraints.

Yoshioka *et al*. (2004) 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*. (2004) 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*. (2004) 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.

Cox (2003) 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*. (2003) developed agent-based systems using ontologies to support collaborative design processes by making diverse and distributed sources of knowledge appear homogenous and integrated to users.

The Enterprise Intelligence Laboratory at University of Toronto has developed an extensive ontology based capability (see Gruninger *et al*. 2000) to address modeling of an entire enterprise, including requirements, supply chain management, quality management, etc.

Lin and Ho (1999) used ontologies to analyze requirements in the domain of network management software.

2.2 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 1) as it creates design objects from the typically unstructured legacy data.

Kusiak *et al*. (2000) 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 (2001) applied data mining for knowledge acquisition from design activities involving a CAD system. They developed a method called *extended dynamic programming* to extract the knowledge. Romanowski and Nagi (2001) proposed a design system based on knowledge extracted from product life-cycle data.

Giess *et al.* (2002) 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 (2004) 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 (2004) used data mining to extract know-how of disciplines related to design processes for micro-technological products. Cascint (2004) uses data mining techniques to create TRIZ-based semantic portals to support the redesign of metal parts in plastic.

Terpenny *et al*. (2000) 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 semi-autonomous, agent-based tools using structured knowledge in design.

2.3 Evolutionary computation

Evolutionary Computation is concerned with the development of problem solving methods based on concepts from natural systems. A number of evolutionary computational methods have been developed, including genetic algorithms, genetic programming, evolutionary strategies, and evolutionary programming. All of these approaches are discussed next.

2.3.1 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 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 be 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.

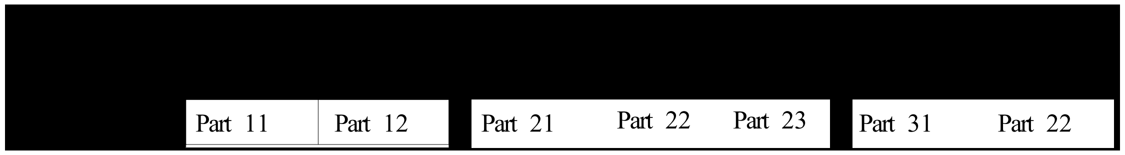

Figure 2. Genomic representation of a product.

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 (1995) applied a genetic algorithm 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 (2002) applied a genetic algorithm to learn the replication of human intent of interest to product design engineering. The user provided initial selections and the algorithm optimized the intent.

2.3.2 Genetic programming

Genetic Programming (GP): Genetic programming creates a computer program in the scheme computer language as the solution (Koza 1992 and Benzhaf *et al.* 1998). Zhang and Muehlenbein (1995) investigated the relationship between the performance and complexity of the evolved structures. Employing statistical search, Iba *et al.* (1995) introduced a new approach to genetic programming 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 (1998) showed that poor results with parsimony pressure are due to "failed" populations 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.* (1999) 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. Genetic programming has also been applied to multiobjective robust design by Forouraghi (2000).

Within the limits of their current applicability, genetic programming 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.

2.3.3 Evolutionary programming

Evolutionary Programming (known also as Evolutionary Algorithms) incorporates aspects of natural selection or survival of the fittest. An evolutionary algorithm (EA) maintains a population of structures (initially randomly generated) that evolve according to 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 evolutionary algorithms is provided in Fonseca and Fleming (1995), Bäck (1996), Coello (1999), and Van Veldhuizen and Lamont (2000). 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 Horn *et al.* 1994 and Moriarty and Miikkulainen 1998).

2.3.4 Evolutionary strategies

Evolutionary Strategy (ES): 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 (1997) extended the evolutionary strategy approach to multi-parent 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. Multi-parent diagonal crossover and uniform scanning crossover and a multi-parent version of intermediary recombination were considered in the experiment. Olafsson (1996) demonstrated the use of evolutionary game theory for allocation of service requirements on to an ensemble of heterogeneous network components. Schweitzer *et al.* (1997) applied Boltzmann and Darwin 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.* (1999) presented evolutionary strategy to design a reconfigurable controller. The designed product exhibit better properties than the one designed with conventional constraint based methods. Moriarty and Miikkulainen (1998) applied co-evolutionary 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 (1999) 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.

2.4 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 1).

Sim and Chan (1992) developed a knowledge-based system for rolling element bearing selection. They used heuristic knowledge supported by a manufacturer's catalogue to generate a solution. Stacey *et al.* (2000) reported on a computational intelligence 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 (2001) 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.* (2003) 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 (2004) 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.

2.5 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 (a) is representative of a particular class of common problems, and (b) 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 1) 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. Marling *et al.* (2002) presented the recent advances in CBR.

Scott and Cook (2004) used CBR in combination with context-free grammars to "emulate human reasoning" with respect to assessing product requirements. Morcous *et al.* (2002) 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 (2000) 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 developed by Leake and Wilson (2001) to support aerospace design.

Many applications of CBR in design have been restricted to relatively narrow domains. Lee and Luo (2002) 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.* (2003) 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 (2003) applied CBR to the design of mechanical bearings. Vong *et al*. (2002) used CBR to design the hydraulic circuits of production machines.

CBR has also been applied to broader cases, e.g., Chiu (2003) used CBR to studying cognitive processes of designers.

2.6 Qualitative reasoning

Qualitative reasoning allows developing models when the relationships between variables and parameters are not well established (Weld and de Kleer 1989). These methods seek ideal solutions to simplified or abstracted situations and therefore they fall in Category 4 (Table 1). 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 (Bratko 1994). Bond graphs are well suited to integrate the process modeling constructs. They provide means for unambiguous definition of the behavior of components by (Karnopp *et al.* 1990):

- 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;
- Allowing for ready formation and subsequent changes of the structure, important in the creative system design;
- 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.* (1990) 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 (2000) discussed bi-directional 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 (1998), but has also examined the role of other theories of qualitative reasoning in product design engineering such as analogical reasoning (see Sgouros 1998). Stahovich *et al.* (2000) used qualitative reasoning to develop a non-mathematical formalization of rigid-body mechanics.

2.7 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, 3, and 4 in Table 1, as they combine aspects of all the major approaches.
Chau and Albermani (2004) 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 advise in preliminary design as well as downstream design stages.

Lou *et al*. (2004) developed a new frame-rule structure for knowledge processing in mold design by incorporating features of product modeling, frame-based KBSs, case-based reasoning, and neural networks. They reported that design efficiency was significantly improved. Zhang *et al*. (2004) developed a system integrating blackboard architecture with case-based reasoning for stamping process planning in progressing die design. The advantage of the system is that case-based reasoning 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 Zadeh 1976; Karray and De Silva 2004).

Nursel (2003) reported an interesting use of a genetic algorithm to design neural network structures. The combined genetic algorithm and neural network approach are reported to reduce the computational complexity in design and manufacturing applications.

Evolutionary programming was used by Rosenman (2000) to adapt previously stored design solutions in a case-based reasoning system. It is argued that such “knowledge-lean” techniques are more broadly applicable than conventional case-based design approaches. Chan *et al*. (2000) 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.

3. 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 1.

3.1 Typological Characteristics of Processes

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 1) 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.

Platform orientation. Analogous to the product platform typology of Schuh *et al.* (2000), 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), adorning the process with other (possibly new) process components as needed. A process based on *common architecture* uses pre-defined 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.

Differentiation. Some processes, called here *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 (Kusiak, 1999). The latter two processes differ in the distribution of process activates. 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 1.

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 *re-engineering* method. Re-engineering 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.

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 apriori 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 1). 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).

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 1) 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.

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 Category 2 (Table 1). 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.

3.2 Selection of CI methods

The typology presented in Section 3.1 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 significant movement of personnel between the divisions. 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 above. 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 *case-based reasoning* 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 list above) 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 computational intelligence 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.

3.3 Evolutionary computation and process perspective

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

- Horizontal, and
- Vertical

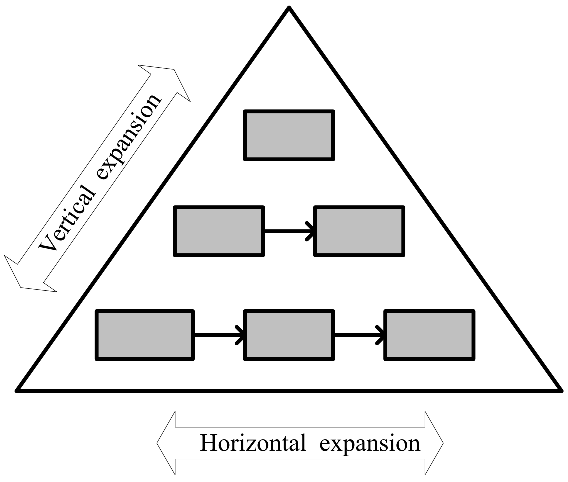

Figure 3. Horizontal and vertical expansion of a process model.

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).

To support the horizontal notion of process modeling concepts from evolutionary computation will be applied. The feasibility of applying evolutionary computation, in particular genetic programming 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 sub-model in Fig. 4(b).

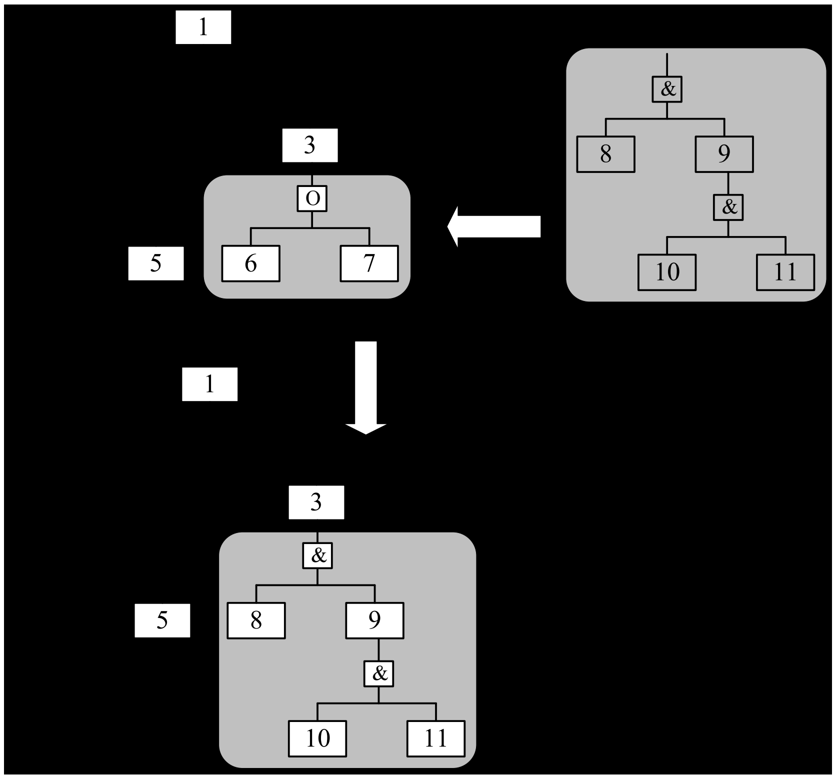

Figure 4. Illustration of the crossover operator in a process model.

The crossover operator demonstrated in Fig. 4 is one of many operators defined in genetic programming that can be applied in product design engineering (e.g., see operators defined in Hawley and Mori 1999).

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:

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

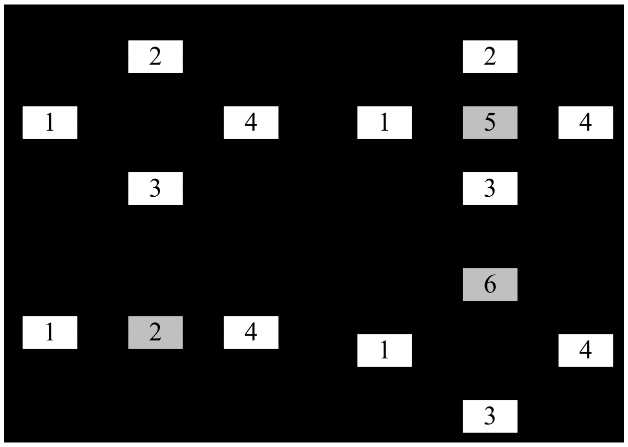

Figure 5. Activity model operators: (a) reference model, (b) generalization, (c) specialization, (d) mutation.

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 sub-process 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 adaptability of the search process of various design libraries and repositories.

4. 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:

- Innovation
- Conceptual design
- Standardization
- Modularity
- Design of product families
- Design complexity management

The relationship between each of these topics and the taxonomy presented in Section 2 is discussed below. For a researcher interested in a particular topic, this relationship allows identifying pertinent CI techniques and literature in Section 2. A practitioner may identify models and tools (Section 2) for a topic of interest.

4.1 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. The recently published "Innovate America" Report (NIIR 2004) has brought renewed interest in innovation.

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 (Kostoff 2003).

There are three basic approaches to innovate: structured, creative, and dynamic, producing either a sustaining or a disruptive product (Allen 2003). 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 (Report_1 2003). Creative innovation thrives more often in small organizations where focusing on the "big picture" is easier accomplished (Allen 2003; Shah 2004). 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 (Report_1 2003).

Sustaining innovations are built off previous innovations (Allen 2003), e.g., the palm PDA. The PDA been an innovative and successful device, however, its predecessor the Apple Newton (a disruptive innovation) has failed. Sustaining innovations tend to be more successful then 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 (TRIZ Journal 2005), the Osborn/Parnes creative problem solving (CPS) process (Daupert 2005), and the innovation technology (IvT) approach.

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 (Domb 2003). 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 (Siem 1996).

The CPS 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, IvT, approach. 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 (Report_2 2004).

Other innovation tools include CREAX (Report_3 2005), Visual Mind (Report_4 2005), and Pull Thinking (Report_5 2005).

Genetic algorithms (GAs) have been used to design new electronic circuits. In some cases Gas have outperformed the circuitry designed by humans (Ando *et al.* 2003 and Thompson *et al.* 1999). Some research, e.g., Deb (2003), suggests that genetic algorithms 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:

- Innovation is the engine of the global economy, accounting for some 50% of the economic growth (NIIR 2004).
- Innovation will mark the first economic revolution of the 21st century (Shah 2004).
- Innovation involves almost all aspects of life, yet the innovation process is not well understood.
- Innovation applies to the creation of methods used in industry, including the design of consumer goods and services.
- The increasing complexity of technologies, their interdependencies, and the rapidly expanding volume of data call for a paradigm shift to be led by innovation.

Innovation clearly belongs to Category 2 and 3 (Table 1).

4.2 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 1).

Zhang *et al.* (2005) 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 (2005) 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 (2003) 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 (2000) 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 (2000) 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 (AHP) methododolgy to select among possible physical laws, and a prescriptive design process. Cvetkovic and Parmee (2002) 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 (2001) developed a computer-based system using inductive learning to semi-automate concept design tasks. Ling *et al.* (2004) proposed case-based reasoning to represent function spaces. Kryssanov *et al.* (2001) suggested that semiotics could significantly improve our understanding of the creative process of designing and developing an applicable computational theory.

4.3 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 1).

Tarondeau (1998) discussed the impact of standardization on the number of components, number of reference points to be managed, and the manufacturing complexity. Lee and Tang (1997) developed a model optimizing the trade-off between the investment in standardization and the profit due to the economy of scale. Erol (1999) proposed a mathematical formulation for the standardization of low value components that was solved by Dupont *et al.* (1999).

Fouque (1999) 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 in 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 (Dupont 1998).

Kota *et al.* (2000) 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. Thoteman and Brandeau (2000) presented an approach for determining an optimum commonality among sub-products from the customer differentiation view point. For highly diversified products, standardization is not the best solution.

4.4 Modularity

A way to design products for highly diversified requirements is to apply modular design methods. These methods are to direct designers towards 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 (Agard and Kusiak 2004a).

Kusiak and Huang (1996, 1998) 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 in are presented in Kusiak (1999) and the recent product design engineering literature.

Product flexibility and the use of common components across various products are important in modular design (Gertosio and Dussauchoy 2004). 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 Dupont *et al.* (1999), 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 Erens and Verhulst (1997). Other examples of modular concepts are presented in Gertosio and Dussauchoy (2004).

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 Martin (1999), Newcomb *et al.* (1998), Simpson (2000), Erens and Verhulst (1997), Dahmus *et al.* (2001), Gonzalez-Zugasti *et al.* (1999), and Jiao and Tseng (1999).

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 (2000) used optimization methods to design modular chemical plant process flowsheets. Similarly, Smayling *et al*. (1999) developed modularized process elements to automate the design of electronic components.

Goel and Bhatta (2004) 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* (2003) 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 1).

4.5 Design of product families

While *modularity* (discussed above) 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 1).

To meet diversified product requirements, numerous strategies are available (Agard and Kusiak 2004b). It is conceivable that a standardized 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*. (2002) 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.* (2002) based their multi-objective approach on a utility-based compromise decision-support model.

Du *et al*. (2002) 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 (2001) and Corbett and Rosen (2004) 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) (Lee and Tang 1997). 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.* (1998), used the term postponement as a synonym of delayed differentiation.

4.6 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.

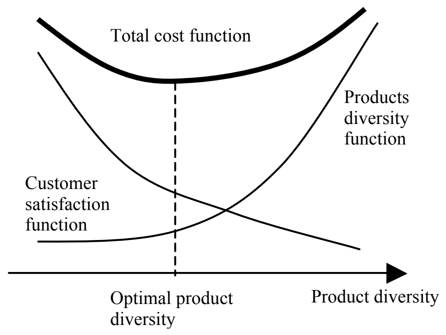

Figure 6. Product diversity cost (Tarondeau 1998).

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).

5. Conclusion

This paper has surveyed recent literature and generalized emerging concepts of computational intelligence 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 computational intelligence 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 3.2). 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 towards systems supporting many of the frameworks reviewed in this paper, including increased automation and enhanced innovation in product design engineering.

References

Agard, B. and A. Kusiak (2004a), Data mining for subassembly selection, *ASME Transactions: Journal of Manufacturing Science and Engineering*, Vol. 126, No. 3, 2004, pp. 627-631.

Agard, B. and A. Kusiak (2004b), A data-mining based methodology for the design of product families, *International Journal of Production Research*, Vol. 42, No. 15, pp. 2955-2969.

Aha, D.W. (1992), Tolerating noisy, irrelevant and novel attributes in instance-based learning algorithms, *International Journal of Man-Machine Studies*, Vol. 36, No. 2, pp. 267-287.

Albers, A. and J. Marz (2004), Restrictions of production on micro-specific product development, *Microsystems Technologies*, Vol. 10, No. 3, pp. 205-210.

Allen, K. (2003), *Bringing New Technology to Market*, Prentice Hall, Upper Saddle River, N.J.

Ando, S., Ishizuka, M., and H. Iba (2003), Evolving analog circuits by variable length chromosomes, *Advances in Evolutionary Computing: Theory and Applications*, ACM, pp. 643-662.

Armstrong, R., Gannon, D., Geist, A., Keahey, K., Kohn, S., McInnes, L., Parker, S., and B. Smolinski (1999), Toward a common component architecture for high-performance scientific computing, *Proceedings of High Performance Distributed Computing Conference HPDC'99*, `www.cs.utah.edu/sci/publications`.

Auer, P., Holte, R., and W. Maass (1995), Theory and application of agnostic PAC-learning with small decision trees, in A. Prieditis and S. Russell, Eds, *ECML-95:*

Proceedings of 8th European Conference on Machine Learning, Springer Verlag, New York.

Bäck, T. (1996), *Evolutionary Algorithms in Theory and Practice*, Oxford University Press, New York.

Bentley, P.J., and J.P. Wakefield, (1995), The table: An illustration of evolutionary design using genetic algorithms, *Proceedings of the 1st IEE/IEEE Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications*, Sep 12-14.

Benzhaf, W., Nordin, P., Keller, R.E., and F.D. Francone (1998), *Genetic Programming; An Introduction*, Morgan Kaufmann, San Francisco, CA.

Berrais, A. (2005), A knowledge-based expert system for earthquake resistant design of reinforced concrete buildings, *Expert Systems with Applications*, Vol. 28, No. 3, pp. 519-530.

Brown, D., Leal, D., McMahon, C., Crossland, R., J. Devlukia (2004), A web-enabled virtual repository for supporting distributed automotive component development. *Advanced Engineering Informatics*, Vol. 18, No. 3, pp. 173-190.

Byrne, R.P. and Bogle, I.D.L. (2000), *Global optimization of modular process flowsheets*, Industrial and Engineering Chemistry Research, Vol. 39, No. 11, pp. 4296-4301.

Carbonell, J.G. (Ed.) (1990), *Machine Learning: Paradigms and Methods*, MIT Press, Cambridge, MA.

Cascint, G. (2004), Plastics design: Integrating TRIZ creativity and semantic knowledge portals, *Journal of Engineering Design*, Vol. 15, No. 4, pp. 405-424.

Chao, K-M., Laing, C., Anane, R., Younas, M., and P. Norman (2003), Multiple evolutionary agents for decision support, *Transactions of the Society for Design and Process Science*, Vol. 7, No. 2, pp. 39-56.

Chan, F.T.S., Jiang, B., and N.K.H. Tang (2000), Development of intelligent decision support tools to aid the design of flexible manufacturing systems, *International Journal of Production Economics*, Vol. 65, No. 1, pp. 73-84.

Chau, K.W. and F. Albermani (2004), Hybrid knowledge representation in a blackboard KBS for liquid retaining structure design, *Engineering Applications of Artificial Intelligence*, Vol. 17, No. 1, pp. 11-18.

Child, P., Diederichs, R., Sanders, F.-H., and S. Wisniowski (1991), The management of complexity, *Sloan Management Review*, Fall, pp. 73-80.

Chiu, M-L, (2003), Design moves in situated design with case-based reasoning, *Design Studies*, Vol. 24, No. 1, pp. 1-25.

Cho, S.-B. (2002), Towards creative evolutionary systems with interactive genetic algorithm, *Applied Intelligence*, Vol. 16, No. 2, pp. 129-138.

Clark, P. and R. Boswell (1989), The CN2 induction algorithm, *Machine Learning*, Vol. 3, No. 4, pp. 261-283.

Coello, C.A.C. (1999), A comprehensive survey of evolutionary-based multiobjective optimization techniques, *Knowledge and Information Systems*, Vol. 1, No. 3, pp. 269-308.

Corbett, B., and Rosen, D.W., (2004), A configuration design based method for platform commonization for product families, *AIEDAM*, Vol. 18, No. 1, pp. 21-39.

Cox, S. (2003), Semantic support for grid-enabled design search in engineering, *Proceedings GGF9 Semantic Grid Workshop*, Chicago, USA.

Cvetkovic, D. and I. Parmee (2002), Agent-based support within an interactive evolutionary design system, *AIEDAM*, Vol. 16, No. 5, pp. 331-342.

Dahmus, J.B., J.P. Gonzalez-Zugasti, and K. Otto (2001), Modular product architecture, *Design Studies*, Vol. 22, pp. 409-424.

Danesh, M.R., and Y. Jin (2001), An agent-based decision network for concurrent engineering design, *Concurrent Engineering: Research and Applications*, Vol. 9, No. 1, pp. 37-47.

Daupert, D. (2005), *The Osborne-Parnes Creative Problem Solving Process Manual*, `http://www.ideastream.com/create`.

Deb, K. (2003), Unveiling innovative design principles by means of multiple conflicting objectives, *Engineering Optimization*, Vol. 35, No. 5, pp 445-470,

Domingos, P. and M. Pazzani (1996), Beyond independence: conditions for the optimality of the simple Bayesian classifier, *Machine Learning: Proceedings of the Thirteenth International Conference,* Morgan Kaufmann, Los Altos, CA, pp. 105-112.

Du, X., Jiao, J., and M.M. Tseng (2002), Product family modeling and design support: An approach based on graph rewriting systems, *AIEDAM*, Vol. 16, No. 2, pp. 103-120.

Dupont, L. (1998), *La Gestion Industrielle: Concepts et Outils*, Hermès, Paris, France.

Dupont, L., Erol, M., Cormier, G., and N. Turkkan (1999), La standardisation des composants: modèles et algorithmes, *3ème Congrès International de Génie Industriel*, Montréal, Canada, May, pp. 671-680.

Eiben, A.E. and T. Bäck (1997), Empirical investigation of multiparent recombination operators in evolution strategies, *Computational Intelligence*, Vol. 5, No. 3, pp. 347-365

Erens, F. and K. Verhulst (1997), Architectures for product families, *Computers in Industry*, Vol. 33, pp. 165-178.

Erol, M. (1999), Prise en compte de la flexibilité dans la planification dynamique, Ph.D. Thesis, Institut National Polytechnique de Grenoble, France.

Fellini, R., Kokkolaras, M., Papalambros, P.Y., and Perez-Duarte, A. (2002) Platform selection under performance loss constraints in optimal design of product families, *ASME Design Engineering Technical Conferences*, paper DETC2002/DAC-34099.

Fensel, D., Musen, M., Plaza, E., Schreiber, G., Studer, R., Wielinga, B., Motta, E., van Harmelen, F., Benjamins, V.R., Crubezy, M., Decker, S., Gaspari, M., Groenboom, R., and Grosso, W. (2003) *The unified problem-solving method development language UPML*, Knowledge and Information Systems, Vol. 5, No. 1, pp. 83-131.

Fonseca, C.M. and P.J. Fleming (1995), An overview of evolutionary algorithms in multiobjective optimization, *Evolutionary Computation*, Vol. 3, No. 1, pp. 1-16.

Forouraghi, B. (2000), A genetic algorithm for multiobjective robust design, Applied Intelligence, Vol. 12, No. 3, pp. 151-161.

Fouque, T. (1999), A la recherche des produits flexibles, *Revue Française de Gestion*, mars-avril-mai, No. 123, pp. 80-87.

Fowler, D. W., Sleeman, D., Wills, G., Lyon, T. and D. Knott (2004), The designers' workbench: Using ontologies and constraints for configuration, *Proc. 24th SGAI International Conference on Innovative Techniques and Applications of Artificial Intelligence*, Queens' College, Cambridge, UK.

Gertosio, C. and A .Dussauchoy (2004), Knowledge discovery from industrial databases, *Journal of Intelligent Manufacturing*, Vol. 15, pp. 29-37.

Giess, M.D., Culley, S.J., and A. Shepherd (2002), *Informing Design Using Data Mining Methods*, *Proceedings of the ASME DETC Conference,* Montreal, Canada.

Goel, A.K. and Bhatta, S.R. (2004), *Use of design patterns in analogy-based design*, Advanced Engineering Informatics, Vol. 18, pp. 85-94.

Gonzalez-Zugasti, J., Otto, K. and J. Baker (1999), Assessing value for product family design and selection, *Proceedings of the 25th ASME Design Automation Conference*, Las Vegas, Nevada, September, pp. 12-15.

Gruber, T. (1992), Ontolingua: A mechanism to support portable ontologies, Technical Report KSL91-66, Knowledge Systems Laboratory, Stanford University, Stanford, CA.

Gruninger, M., Atefi, K., and M.S. Fox (2000), Ontologies to support process integration in enterprise engineering, *Computational and Mathematical Organization Theory*, Vol. 6, No. 4, pp. 381-394.

Hamburg, L. (2004), Improving Computer supported Environment Friendly product Development by Analysis of Data, Proceedings of the 2nd European Conference on Intelligent Systems and Technologies Conference, Iasi, Romania.

Hawley, R.S. and C.A. Mori (1999), *The Human Genome: A User's Guide*, Academic Press, San Diego, CA.

He, D.W., A. Kusiak, and T.L. Tseng (1998), Delayed product differentiation: A design and manufacturing perspective, *Computer-Aided Design*, Vol. 30, No. 2, pp. 105-113.

Horn, J., Goldberg, D.E., and K. Deb (1994), Implicit niching in a learning classifier system: Nature's way, *Evolutionary Computation*, Vol. 2, No. 1, pp. 37-66.

Huang, C.C. and Kusiak, A. (1998), Modularity in design of products and systems, *IEEE Transactions on Systems, Man and Cybernetics*, Part A, Vol. 28, No. 1, pp. 66-77.

Iba, H., deGaris, H., and T. Sato (1995), A numerical approach to genetic programming for system identification, *Evolutionary Computation,* Vol. 3, No. 4, pp. 417-452.

Inmon, W.H., Terdeman, R.H., and C. Imhoff (2000), *Exploration Warehousing: Turning Business Information into Business Opportunity*, John Wiley, New York.

Ishino, Y. and Y. Jin (2001), Data Mining and Knowledge Acquisition in Engineering Design, in *Data Mining for Design and Manufacturing: Methods and Applications*, Braha, D. (Ed), Kluwer, Norwell, MA, pp. 145-160.

Jiao, J. and M. Tseng (1999), A methodology of developing product family architecture for mass customization, *Journal of Intelligent Manufacturing*, Vol. 10, pp. 3-20.

Karnopp, D.C., Margolis, D.L., and R.C. Rosenberg (1990), *System Dynamics: A Unified Approach*, John Wiley, New York.

Karray, F.O. and C.W. De Silva (2004), *Soft Computing and Intelligent Systems Design: Theory, Tools and Applications,* Addison Wesley, New York.

Kitamura, Y., Kashiwase, M., Fuse, M., and R. Mizoguchi (2004), Deployment of an ontological framework of functional design knowledge, *Advanced Engineering Informatics*, Vol. 18, No. 2, pp. 115-127.

Kosanke, K. and J.G. Nell (Eds) (1997), *Enterprise Engineering and Integration*, Springer, New York.

Kostoff, R.N. (2003), Role of technical literature in science and technology development and exploitation, *Journal of Information Science*, Vol. 29, No. 3, pp. 223-228.

Kota, S., Sethuraman, K., and R. Miller (2000), A metric for evaluating design commonality in product families, *ASME Transactions: Journal of Mechanical Design*, Vol. 122, pp. 403-410.

Koza, Z. (1992), *Genetic Programming*, MIT Press, Cambridge, MA.

Kryssanov, V.V., Tamaki, H., and S. Kitamura (2001), Understanding design fundamentals: how synthesis and analysis drive creativity, resulting in emergence, Artificial Intelligence in Engineering, Vol. 15, No. 4, pp. 329-342.

Kusiak, A. (1999), *Engineering Design: Products, Processes and Systems*, Academic Press, San Diego, CA.

Kusiak, A. (2000), *Computational Intelligence in Design and Manufacturing*, John Wiley, New York.

Kusiak, A. and C.C. Huang (1996), Development of modular products, *IEEE Transactions on Components, Packaging, and Manufacturing Technology*, Part A, Vol. 19, No. 4, pp. 523-538.

Kusiak, A., Kernstine, K.H., Kern, J.A., McLaughlin, K.A., and T.L. Tseng (2000), Data Mining: Medical and Engineering Case Studies, *Industrial Engineering Research Conference*, Cleveland, Ohio.

Leake, D.B., and D.C. Wilson (2001), A case-based framework for interactive capture and reuse of design knowledge, *Applied Intelligence*, Vol. 14, No. 1, pp. 77-94.

Lee, K.S. and C. Luo (2002), Application of case-based reasoning in die-casting die design, *International Journal of Advanced Manufacturing Technology*, Vol. 20, No. 4, pp. 284-295.

Lee, H.L. and C.S. Tang (1997), Modeling the costs and benefits of delayed product differentiation, *Management Science*, Vol. 43, No. 1, p. 40-53.

Lim, T.-S., Loh, W.-Y., and Y.-S. Shih (2000), A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms, *Machine Learning*, Vol. 40, pp. 203-228.

Lin, C-Y.I. and C-S. Ho (1999), A generic ontology-based approach for requirement analysis and its application in network management software. *AIEDAM*, Vol. 13, No. 1, pp. 37-61.

Lina, Y.J., and R. Farahati (2000), Optimum assembly design utilizing a behavioral modeling concept, *Assembly Automation*, Vol. 23, No.2, pp. 181-191.

Ling, W., Yan, J., Wang, J., and Y. Xie (2004), Case-based conceptual design, *Chinese Journal of Mechanical Engineering* (English Edition), Vol. 17, No. 1, pp. 73-77.

Lohn, J.D. and S.P. Colombano (1999), A circuit representation technique for automated circuit design, *IEEE Transactions on Evolutionary Computation*, Vol. 3, No. 3, pp. 205-219.

Lou, Z., Jiang, H., and X. Ruan (2004), Development of an integrated knowledge-based system for mold-base design, *Journal of Materials Processing Technology*, Vol. 150, No. 1-2, pp. 194-199.

MacDuffie, J.-P., Sethuraman, K., and M.-L. Fisher (1996), Product variety and manufacturing performance: Evidence from the international automotive assembly plant study, *Management Science*, Vol. 42, No. 3, pp. 350-369.

Marling, C., Sqalli, M., Rissland, E., Munoz-Avila, H., and D. Aha (2002), Case-based reasoning integrations, *AI Magazine*, Spring Issue, pp. 69-86.

Martin, M.V. (1999), Design for variety: A methodology for developing product platform architectures, Ph.D. Thesis, Stanford University, Stanford, CA.

Michalewicz, Z. and M. Schoenauer (1996), Evolutionary algorithms for constrained parameter optimization problems, *Evolutionary Computation*, Vol. 4, No. 1, pp. 1-32.

Ming, X.T., (2001), Inductive learning techniques in design process: A design concept learning system, *Integrated Computer-Aided Engineering*, Vol. 8, No. 2, pp. 171-186.

Morcous, G., Rivard, H., and A.M. Hanna (2002), Case-Based Reasoning System for Modeling Infrastructure Deterioration, *Journal of Computing in Civil Engineering*, Vol. 16, No. 2, pp. 104-114.

Moriarty, D.E. and R. Miikkulainen (1998), Forming neural networks through efficient ad adaptive coevolution, *Evolutionary Computation*, Vol. 5, No. 4, pp. 373-399.

Murthy, S.K. and S. Salzberg (1994), A system for the induction of oblique decision trees, *Journal of Artificial Intelligence Research*, Vol. 2, No. 1, pp. 1-33.

Mussi, S., (2004), Putting value of information theory into practice: a methodology for building sequential decision support systems, *Expert Systems*, Vol. 21, No. 2, pp. 92-103.

Newcomb, P.J., B. Bras, D.W. Rosen (1998), Implications of modularity on product design for the life cycle, *ASME Transactions: Journal of Mechanical Design*, Vol. 120, No. 3, pp. 483-490.

NIIR (2004), *Innovate America*, Council for Competitiveness, National Innovation Initiative Report.

Nursel, Ö. (2003), Use of genetic algorithm to design optimal neural network structure, *Engineering Computations*, Vol. 20, No. 8, pp. 979-997.

Olafsson, S. (1996), Resource allocation as an evolving strategy, *Evolutionary Computation*, Vol. 4, No. 1, pp. 33-55.

Parmee, I.C. and C.R. Bonham (2000), Towards the support of innovative conceptual design through interactive designer/evolutionary computing strategies, *AIEDAM*, Vol. 14, No. 1, pp. 3-16.

Pisan, Y., (1998), An integrated architecture for engineering problem solving, Doctoral dissertation, Northwestern University, Evanston, IL.

Pokojski, P., Okapiec, M., and G. Witkowski (2002), Knowledge-based engineering, design history storage, and case-based reasoning on the basis of car gear box design, *Proceedings of the Conference on Artificial Intelligence Methods*, Gliwice, Poland.

Qin, X., and Regli, W.C., (2003), A study in applying case-based reasoning to engineering design: mechanical bearing design, *AIEDAM*, Vol. 17, No. 3, pp. 235-252.

Report_1 (2003), Cheskin and Fitch: Worldwide, *Fast, Focused & Fertile: The Innovation Evolution*.

Report_2 (2004), Dodgeson, Gann and Salter, *Industrial Dynamics, Innovation and Development*, Elsinore, Denmark.

Report_3 (2005), The CREAX Innovation Suite 3.1, `http://www.creax.com/tools.htm`.

Report_4 (2005), Visual Mind, `http://www.visual-mind.com`.

Report_5 (2005), Pull Thinking, `http://www.pullthinking.com`.

Rivard, H. and S.J. Fenves (2000), SEED-Config: A case-based reasoning system for conceptual building design, *AIEDAM*, Vol. 14, pp. 415-430.

Romanowski, C.J. and R. Nagi (2001), A Data Mining for Knowledge Acquisition in Engineering Design, in *Data Mining for Design and Manufacturing: Methods and Applications*, Braha, D. (Ed.), Norwell, Boston, MA, pp. 161-178.

Rosenman, M. (2000), Case-based evolutionary design, *AIEDAM,* Vol. 14, No.1, pp. 17-29.

Schuh, G., Ley, W., Gruenenfelder, M.P., and A.P. Hofer (2000), The potential for product family management based on product platform concepts, in S. Sivaganathan and P.T. Andrews (Eds), *Design for Excellence*, Professional Engineering Publishing, London, UK, pp. 601-611.

Schweitzer, F., Rosé, H., Ebeling, W., and O. Weiss (1997), Optimization of road networks using evolutionary strategies, *Evolutionary Computation*, Vol. 5, No. 4, pp. 419-438.

Scott, W. and S.C. Cook (2004), A requirements assessment architecture that combines natural language parsing and artificial intelligence, Proc 14th Annual Symposium of the International Council on Systems Engineering, Toulouse, Paper Number 6.1.3.

Siem, P. (1996), *An Introduction to TRIZ: A Revolutionary New Product Development Tool*, Visions, January.

Seepersad, C.C., Mistree, F., and J.K. Allen (2002), A quantitative approach for designing multiple product platforms for an evolving portfolio of products, *ASME Design Engineering Technical Conferences*, Paper DETC2002/DAC-34096.

Sgouros, N., (1998), Interaction between physical and design knowledge in design from physical principles, *Engineering Applications of Artificial Intelligence*, Vol. 11, pp. 449- 459.

Shah, J. (2004), Engineering Design in 2030: An NSF Strategic Planning Workshop, (Workshop Chair: J. Shah), `http://dal.asu.edu/engdesign/index.html`.

Siddique, Z. and D.W. Rosen (2001), On combinatorial design spaces for the configuration design of product families, *AIEDAM*, Vol. 15, No. 2, pp. 91-108.

Sim, S.K. and Y.W. Chan (1992), A knowledge-based expert system for rolling-element bearing selection in mechanical engineering design*, Artificial Intelligence in Engineering*, Vol. 6, No. 3, pp. 125-135.

Simpson, T.W., J.-R.A. Maier, and F. Mistree (2000), Product platform design: Method and application, *Research in Engineering Design*, Vol. 13, pp. 2-22.

Smayling, M., Rodriguez, J., Young, A. and Ichiro, F. (1999), *Process synthesis using TCAD: a mixed-signal case study*, IEICE Transactions on Electronics, Vol. E82-C, No. 6, pp. 983-991.

Stacey, M., Clarkson, P.J., and C. Eckert (2000), Signposting: an AI approach to supporting human decision making in design, *Proceedings of the ASME Computers in Engineering Conference,* Paper CIE-14617.

Stahovich, T.F., Davis, R., H. Shrobe (2000), Qualitative rigid-body dynamics, *Artificial Intelligence*, Vol. 119, No. 1-2, pp. 19-60.

Soule, T. and J.A. Foster (1998), Effects of code growth and parsimony pressure on populations in genetic programming, *Evolutionary Computation*, Vol. 6, No. 4, pp. 293-309.

Tarondeau, J.C. (1998), *Stratégie Industrielle* (2nd Edition), Collection Gestion, Vuibert, France.

Terpenny, J.P., Strong, S., and J. Wang (2000), A methodology for knowledge discovery and classification, *Proceedings of the Tenth Flexible Automation and Intelligent Manufacturing Conference,* June, College Park, MD.

Thompson, A., Layzell, P., and R.S. Zebulum (1999), Exploration in design space: Unconventional electronics design through artificial evolution, *IEEE Transactions on Evolutionary Computation*, Vol. 3, No. 2, pp. 167-196.

Thonemann, U.W. and M. Brandeau (2000), Optimal commonality in component design, *Operations Research*, Vol. 48, No. 1, pp. 1-19.

Tor, S.B., Britton, G.A., and W.Y. Zhang (2003), Indexing and Retrieval in Metal Stamping Die Design Using Case-based Reasoning, *ASME Transactions : Journal of Computing and Information Science in Engineering*, Vol. 3, No. 4, pp. 353-362.

Tormey, D., Chira, O., Chira, C., Brennan, A., and T. Roche (2003), The Use of Ontologies for Defining Collaborative Design Processes, *Proceedings of the 32nd International Conference on Computers and Industrial Engineering*, University of Limerick, UK.

Tsumoto, S. (2000), Automated discovery of positive and negative knowledge in clinical databases, *IEEE Engineering in Medicine and Biology*, Vol. 19, No. 4, pp. 56-62.

Van Veldhuizen, D.A. and G.B. Lemont (2000), Multiobjective evolutionary algorithms: Analyzing the state-of-the-art, *Evolutionary Computation*, Vol. 8, No. 2, pp. 125-147.

Vong, C.M., Leung, T.P., and P.K. Wong (2002), Case-based reasoning and adaptation in hydraulic production machine design, *Engineering Applications of Artificial Intelligence*, Vol. 15, No. 6, pp. 567-585.

Weld, D.S. and J. de Kleer, Eds. (1989), *Readings in Qualitative Reasoning About Physical Systems*, Morgan Kaufmann, Los Altos, CA.

Yao, X., Liu, Y., and G. Lin (1999), Evolutionary programming made easier*, IEEE Transactions on Evolutionary Computation*, Vol. 3, No. 2, pp. 82-102.

Yoshioka, M., Umeda, Y., Takeda, H., Shimomura, Y., Nomaguchi, Y., and T. Tomiyama (2004), Physical concept ontology for the knowledge intensive engineering framework. *Advanced Engineering Informatics*, Vol. 18, No. 2, pp. 95-113.

Zadeh, L. (1976), A fuzzy-algorithmic approach to the definition of complex or imprecise concepts, *International Journal of Man-Machine Studies*, Vol. 8, pp. 249-291.

Zakarian, A. and A. Kusiak (2000), Analysis of process models, *IEEE Transactions on Electronic Packaging Manufacturing*, Vol. 23, No. 2, pp. 137-147.

Zavbi, R. and J. Duhovnik (2000), Conceptual design of technical systems using functions and physical laws, *AIEDAM,* Vol. 14, No. 1, pp. 69-83.

Zhang, B.-T. and H. Muehlenbein (1995), Balancing accuracy and parsimony in genetic programming, *Evolutionary Computation*, Vol. 3, No. 1, pp. 17-38.

Zhang, W.Y., Tor, S.B., and G.A. Britton (2004), A Hybrid Intelligent System for Stamping Process Planning in Progressive Die Design, *Innovation in Manufacturing Systems and Technology (IMST) Report*, MIT, `http://hdl.handle.net/1721.1/3905`.

Zhang, W.Y., Tor, S.B., and G.A. Britton (2005), A graph and matrix representation scheme for functional design of mechanical products, *International Journal of Advanced Manufacturing Technology*, Vol. 25, No. 3-4, pp. 221-232.