Computer-Aided Design, Vol. 30, No. 12, pp. 943–956, 1998

0010-4485/98/$ - see front matter

PII: S0010-4485(98)00050-5

Design with objects: an approach to object-oriented design

Wen-Yau Liang and Peter O'Grady*

This paper is concerned with object-oriented design and aims to describe an object-oriented approach [called design-with-objects (DwO)] that encompasses both the fundamentals of object-oriented design and the use of object-oriented design in the development of design process models. The major design objects are described, together with their main structure, the message passing and inheritance involved, and the computability and exchangeability of the approach. The main potential advantages of such an approach are *computability*, in that a design process model obtained using DwO is not just a descriptive model but a computable model; *reusability*, in that once a design object in DwO has been established, it can be used repeatedly; and *exchangeability*, in that objects with similar interfaces can be readily exchanged in a modular manner. This paper reviews the background to design process models and object-oriented design and then describes the basis of DwO. Design models, DwO methods and an architecture for DwO are described together with an example design process model formalism that uses DwO. The implementation of this example DwO design process model formalism for the problem of electronics assemblies components selection is overviewed.

Keywords: Design process, Electronics, Electronics assembly, Internet , Object-oriented design

NOMENCLATURE

{ }	set
[ ]	subset
⟨⟩	order set
$\{\}_u$	unsatisfied set
$\{\}_s$	satisfied set
C_i	constraints
k	generation/iteration
$o^l_{i,j}$	design object i with attribute j at design model l
n	number of design objects in S_l^k
$R_{o,p}$	functional requirement o with attribute p
S	design model S
V_l^k	objective function, where $V_l^k = f(S_l^k)$.
ξ	Object operator: inheritance
ψ	Object operator: import
ω	Object operator: message passing
O	Object: physical design objects
A	Object: design algorithms
R&C	Object: requirements and constraints
E	Object: evaluation schema

*Corresponding author. Tel.: 001 319 358 2979; Fax: 001 319 359 2454; E-mail: peter-ogrady@uiowa.edu
Department of Industrial Engineering, University of Iowa, Iowa City, IA 52242, USA
Paper Received: 12 March 1998. Revised: 3 July 1998. Accepted: 13 July 1998

INTRODUCTION

The fundamentally important nature of design has prompted much work into design and into the design process. The *design process* is the process by which design activities are performed to produce a design model that satisfies some given specifications. Research in studying design process, design theory, and design methodology aims to understand:

- what the design process is;
- how the design process proceeds;
- how the alternatives in design are produced and evaluated;
- how the design objectives are translated into a physical form;
- how the designers search or explore the alternative designs;
- how the design history should be managed;
- and how the design is represented as it progresses.

However, the most fundamental question is 'how do we design?' and this question concerns the nature of the activities that constitute the design process. Modeling the design process can be accomplished with a proper understanding on how we do design. This section focuses on the issues related to the *progress* of the design process, which includes the generation and selection of alternatives in design, and the representation of design as it evolves through the design process.

The following are some of the definitions of design:

> Design is an activity for converting specification descriptions to design descriptions ready for implementation and, in general, includes within it the concerns of manufacturing (materials, processing, and equipment) and also concerns felt further down the

life cycle of the product (e.g., maintenance and disposal).[1]

Design is the process of attaining wide latitude and narrow variance.[2]

Design is a search process in which a satisfactory design solution is produced from a number of alternatives.[3]

Design involves a continuous interplay between what we want to achieve and how we want to achieve it.[4]

Suh[4] states that the objective of design is always stated in the functional domain, whereas the physical solution is always generated in the physical domain. The design procedure involves inter-linking these two domains at every hierarchical level of the design process. In this view, these two domains are inherently independent of each other but are related by the design.

Part of the design could be thought of as a creative process. But the answer to the question 'How does one become creative?' is difficult to define. A number of people have tried to address this question.[5] Much work has been done to understand the creative process and develop a design methodology that can systematize the design process.[6–9] Harrisberger[10] lists a number of techniques which can be used to aid the design process, including the trigger-work technique, the checklist technique, the morphological technique, the attribute-seeking technique, the Gordon technique, and the brainstorming technique.

Other authors present rules for design in various situations[11,12] as well as general methodologies.[13–16] However, these techniques apply design rules or classification methods to a specific situation, or are not generalizable.[4]

The work on studying the nature of the design process, described in the following section, aims to obtain the necessary insights to develop design process models, which can then act as the base for improving the process, or for, for example, computerization of the process.

One promising approach is that of *object-oriented design*:

Object-oriented design is a method of design encompassing the process of object-oriented decomposition and a notion for depicting both logical and physical as well as static and dynamic models of the system under design.[17]

The emphasis of the object-oriented design is *decomposition* and *representation*. Decomposition is the breakdown of the functions of a product so that the lowest level of the function structure consists exclusively of functions that cannot be sub-divided further, while remaining generally applicable. Decomposition differentiates object-oriented design from conventional structured design. Representation is concerned with defining an object and organizing objects. It is generally thought that a good representation leads to a successful object-oriented design.

Work done on object-oriented design systems includes that by Kusial et al.[18], Takeda et al.[19], and Dopplick.[20] The potential advantages of object-oriented design have been recognized,[21,22] but the implementation path for such an approach has received little attention.

This paper aims to describe an object-oriented approach [called design-with-objects (DwO)] that encompasses both the fundamentals of object-oriented design and the use of object-oriented design in the development of design process models. The major design objects are described, together with their main structure, the message passing and inheritance involved, and the computability and exchangeability of the approach. The main potential advantages of such an approach are *computability*, in that a design process model obtained using DwO is not just a descriptive model but a computable model; *reusability*, in that once a design object in DwO has been established, it can be used repeatedly; and *exchangeability*, in that objects with similar interfaces can be readily exchanged in a modular manner.

While the DwO approach possesses some important potential advantages, there are several research issues that remain to be addressed before such an approach can be used. These research issues are related to the design objects, design object methods and the design process model formalism and this paper explores potential approaches to address these research issues.

The format of this paper is as follows: the background to design process models and object-oriented design is reviewed and this leads to a description of the basis of DwO. This basis of DwO rests on the concept of a design object which can represent not only physical entities, such as parts or components, but also non-physical entities, such as the design history or vendor information. Each design object can have methods, data (or property) and an interface (or structure). Design models, DwO methods and an architecture for DwO are then described. This is followed by an example design process model formalism that uses DwO. The implementation of this example DwO design process model formalism for the problem of electronics assemblies components selection (EACS) is then described.

THE DESIGN PROCESS

The differing views of the design process may be classified according to the activities involved,[23] their perspectives,[24] the nature of the design process[1] or else as a transformational view in which engineering design is regarded as a transformation from function to form.[25,3] Alternatively, design may be viewed as a problem-solving process, involving searching through a state space, where the states represent the various design solutions.[26] Bardasz and Zeid[27] classify the design process into two classes: creating new design models for new problems, and modifying old design models to fit new problems. Classifying design as routine design and innovative design, parametric design and configuration design, and variant design and creation of new design is suggested.[28,3,1,29] *Figure 1* depicts the relationships between these various classifications of design, although the borderline between the various classifications can be blurred.

Design process models

It has been suggested that design process models, in essence, are required to answer the following three basic questions.[31]

(1) How does the model incorporate the representation and reasoning mechanism about various forms of design knowledge?
(2) How well does the model describe the design process itself as it proceeds?
(3) How can the design process model be integrated into a system for design support?

Smithers[31] also identifies a set of basic types of knowledge applying and generating activities, which are typically found in all kinds of engineering design, and includes

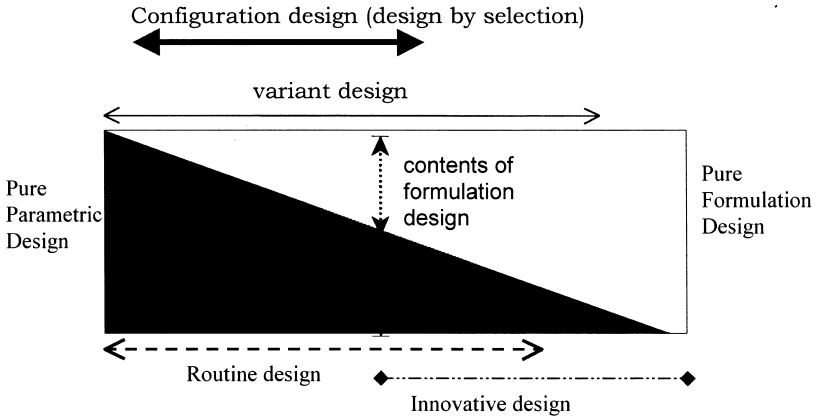


Figure 1 Classifications of design (adapted from Kim[30])

requirement description, decomposition, strategy planning, detailing and parameterization, synthesis, simulation, analysis, optimization, and documentation.[32,33] Though all of these activities are not needed in all cases of engineering design, it is generally accepted that the design process is composed of these or similar activities.

We can define design as the creation of synthesized solutions in the form of products, processes or systems that satisfy perceived needs through mapping between functional requirements (FRs) in the functional domain and the design objects (DOs) in the objects domain, through the proper selection of DOs that satisfy FRs (adapted from Suh,[4]), i.e.

$$[FR] = [A] \cdot [DO]$$

Where [A] is the design matrix. The relationship between requirements and functions is discussed in Kota and Ward,[34],and Kusiak and Szczerbicki.[35]. The meaningful and compatible combination of sub-functions into overall function produces a *functional structure*.[36]

Classifications of design process models

The design process models can be classified on a number of bases, each of which is based on its own view of design, thus leading to a variety of design process models. The research done would indicate that design process models can be classified as follows:

- **Perspectives**: Finger and Dixon[37] classify design process models as being either a descriptive model or a prescriptive model.
- **Nature of the design process**: Kannapan and Marshek[1] view the design process as either a natural process or an artificial process.
- **Transformational view**: Kott and May[38] classify the design process as being either a decomposition model or a transformation model.
- **Problem-solving process**: Yang and Rozenblit[39] see design as a search process and classify design process models as being either a computational model for hierarchical mechanical system design, an optimal directed design model, or a knowledge-based creative design model.

None of the categories in any classification scheme is exclusively able to explain the design process as a whole. It may well be said that these classifications reflect various aspects of the design process and its models.

Issues in design process models

Representation and evolution of design models

The concept of the design stages is often used to describe the progress of design.[40] Several researchers (including Yang and Rozenblit[39], Mantyla[40], and Hubka[41]) indicate that the design process consists of stages of progressive details. The results of a design process are represented in a design model. Eastman et al.[42] state that the information in the design model should include the representation of function, form, and physical properties; support for various levels of abstraction and for the various phases of the product life-cycle; the representation of semantics for all uses and validation; and extendibility.

Transformation from design objectives to a design model

Design objectives can be thought of as the explicit or implicit intent of design described in the form of functional requirements, function, user specification, and/or design constraints. The terms 'functional requirements', 'function', and 'constraints' can often loosely mean 'design objective'. However, there is not clear agreement on the definitions of function, functional requirements, and constraints, about how they are represented, and how they can be utilized for reasoning.[43] A commonly accepted notion is that the functions have relations to the form of the artifact being designed, and one of the primary tasks of the design process is to translate functional requirements into specific forms of the artifact being designed. It is still a point of argument whether the function to form relationship is 1:1, 1:n or n:m.[44,45] Methodologies for the transformation of design objectives to the attributes in design models have been reported. Suh[4] defines the design's objectives in terms of functional requirements (FRs) in the general domain. Then, to satisfy these functional requirements, a physical embodiment characterized in terms of design parameters (DPs) must be created. The design process involves relating these FRs of the functional domain to the DPs of the physical domain. Fenves and Baker[46], Lai and Wilson[47], Ulrich and Seering[48], and Paul and Beitz[36] have reported methodologies for the transformation of design objectives to the attributes in the design model, albeit in a restricted domain.

Design alternatives can be thought of as the result of the different interpretations and transformations of design objectives into design models, and the progress of design through the design stages can be seen as the search process for alternatives. However, it still remains to be argued whether the search process is strictly top down or bottom up or opportunistic.[49]

Design data modeling tools

There are two main tools that have been developed for modeling design data, namely STEP/EXPRESS and IDEF1X.

STEP/EXPRESS. STEP is emerging as an international standard under the auspices of the International Standard Organization (ISO) 10303, and it has begun to replace other standards.[50,51] The STEP standard also includes a part dealing with the EXPRESS language.[51] EXPRESS is a data descriptive language designed to allow partitioning of the diverse materials in STEP. It is both human and machine readable and aims to enhance human understanding and the generation of machine-interpretable applications. A schema, defined in the EXPRESS language, can be mapped to a physical to facilitate exchanging data between computer aided systems, and can be transformed to a database schema. EXPRESS provides a syntax for defining classes of entities (which may be information, resources, material, products, etc.) that support abstraction. However, dynamic behavior cannot be modeled.[52]

IDEF1X. IDEF1X is based on the entity-relationship approach to semantic data modeling developed by Chen[53] and the relational database theory of Codd[54]. The IDEF1X methodology shares many of the same constructs found in entity-relationship models: entities, attributes, and relationships between these entities.[55]

Both of these approaches are concerned with data modeling and can therefore be used to model the data flow in an object-oriented approach.

OBJECT-ORIENTED DESIGN

The previous section has described the background to design process models and the disconcertingly large number of different views about the design process, and hence about design models. The main issues that arise from the various views of the design process have been described while some of the background to object-oriented design has been described in the Introduction. This section further describes the background to object-oriented design and why it is potentially important.

Objects

Objects have a very complex history. However they somewhat resemble frames, introduced by Minsky.[56] Frames were proposed to represent knowledge units that are larger than the constants, terms, and expressions offered by logic. A frame represents a concept in multiple ways, and typically can be divided into descriptive as well as a behavioral components. Objects and frames share the property that they bring descriptive and behavioral features closely together. In software development, objects are most often used to help understand operations.[57] For example, the Simula[58] programming language is another, even older, historical root of objects. Simula was aimed at supporting simulation activities and supported parallelism, via the approximation of co-routines, allowing for many interacting entities in a simulation.

Several have proposed differing definitions of 'object', including:

> A visible or tangible thing of relative stable form; a thing that may be apprehended intellectually; a thing to which thought or action is directed.[59]
>
> An object has identity, state and behavior.[17]
>
> An object is a unit of structural and behavioral modularity that has properties.[60]

Champeaux et al.[57] define an object as a conceptual entity that:

- is identifiable;
- has features that span a local state space;
- has operations that can change the status of the system locally, while also inducing operations in peer objects.

Elements of the object

The main purpose of elements of the objects is to manage complexity.[17,21] Coad and Yourdon[21] suggest principles for managing complexity — including the following: abstraction, encapsulation, inheritance, association, communication with message, pervading methods of organization, scale, and behavior classification. Booch[17] claims that there are four major elements in an object, which includes abstraction, encapsulation, modularity, and hierarchy. He also suggests another three minor elements: typing, concurrency and persistence. These elements of the object form the basis of object-oriented design and lead to a discussion of why object-oriented design matters and the process of object-oriented design.

The process of object-oriented design

The object-oriented design process can be described or/and prescribed by classifying activities into various phases and components. Nearly all research makes at least the coarse distinction between analysis, design, and implementation phases of development.[57] Booch[17] suggests the object-oriented design process can be divided into a micro-development process and a macro-development process. Although it has been said that the object-oriented paradigm is changing the classic distinctions between analysis, design and implementation,[57] intrinsic differences among phases cannot be forgotten. The aim of analysis can be thought of as clarifying what the task is all about, while design and implementation take the task specification as a given and then work out the details of the solution. Another point of view is that objects in the analysis realm differ significantly from objects in the implementation phase, since an object in analysis is independent and autonomous but an object in an object-oriented programming language usually shares a single thread of control with many or all other objects.[57] Hence, the design phase plays a crucial role in bridging between these different computational objects.

Previous work on object-oriented design

Veth[61] and Tomiyama[62] suggest that there are two categories of design knowledge. One is design object

Table 1 Summary of previous work related to object-oriented design in engineering design

Domain	Description	Reference
Design knowledge	Suggests that there are two categories of design knowledge: design object knowledge and design process knowledge.	61,62 63,64
VLSI CAD applications	Deal with design objects that have an interface description and an implementation description. Identify modeling concepts and semantics of VLSI CAD design objects.	65
Pegasus	An object-oriented model targeted to support CAD applications.	66
Applications in engineering and computer science	Provides overall concepts of design methodologies and systems of object-oriented information management for advanced applications.	67
Design process model	This model has two levels of inferences; i.e. an action level and an object level.	19
CAD system	An object-oriented system for conceptual design.	18
Design for manufacture (DFM)	Applying the object-oriented design for manufacture.	68
Design support system	Applying an object-oriented intelligent design support system which is intended to assist in the basic design of machine tools.	69
Earth observing system (EOS)	Using an object-oriented design methodology, the design is iteratively refined with progressively more definition of subsystems, lower-level objects, and associated interfaces.	20
Data modeling	An object-centered approach for modeling engineering design products, which combining description logic and object-oriented modeling.	29

knowledge. Examples of this include research in geometric modeling and product modeling. The other is design process knowledge. Examples of this research include Yoshikawa[63] who propose a general design theory that discusses the static aspects of design based on axiomatic set theory. Ullman et al.[64] conduct an empirical study on design and show that the protocol analysis method was useful. Takeda et al.[19] propose a computable design process model based on an analysis of design protocols. This model has two levels of inferences: namely an action level and an object level. In this model, a design object is described in first order predicate logic at the object level, while a design process is represented and navigated by a sequence of design operations, including modification and evaluation of the design object, in the action level. In other words, the design process knowledge is operational knowledge about the design object knowledge. Kusiak et al.[18] present an object-oriented system for conceptual design that is applied to a CAD system. Dopplick[20] adapts the object-oriented design concept into engineering design. Using an object-oriented design methodology, the design is iteratively refined with progressively more definition of subsystems, lower-level objects, and associated interfaces. *Table 1* shows a summary of the previous work related to object-oriented design in engineering design. Each would appear to have been built for a particular domain and may therefore be difficult to extend to other design domains.

A PROPOSED DESIGN-WITH-OBJECTS APPROACH TO DESIGN

This section describes a proposed object-oriented design approach, termed design-with-objects (DwO). The concept of a design object is first detailed and the research issues associated with DwO are then described. A formalism for DwO is described in a following section.

The concept of a design object

The proposed DwO approach views design objects as being objects that represent not only physical entities, such as parts or components, but also non-physical entities, such as the design history or vendor information. Each design object can have the following features:

- **Methods**: which are procedures that act on data or on design objects.
- **Data (or property)**: which is a named attribute of a design object. Data (or property) describes design object characteristics such as the size, the weight, the state of a design object, or the state of progress of the design model.
- **Interface (or structure)**: this defines the basic structure and interface of a design object.

Given the above view of a design object, the relationship between design objects can be established by using the following object operators:

- **Inheritance (ξ)**: which creates an instance B from a design object A. The instance B then can be thought of another design object which inherits the basic structure and methods of A but which can be altered and which can contain its own data.
- **Import (ψ)**: this allows design object B to import methods from another design object A. The design object B can now use these imported methods in a similar manner to A.
- **Message passing (ω)**: this allows data to be passed from one design object B to another design object A. This could involve the data or methods associated with A and/or B.

These operators are shown diagrammatically in *Figure 2*.

The use of design objects in DwO has several potential advantages:

(1) **Computability**: design can be thought of a computable design process model. The implication for DwO is that a design process model obtained using DwO is not just a descriptive model but a computable model, which computes new stages in the design process using the object operators described above.
(2) **Reusability**: once a design object in DwO has been established, it can be used repeatedly.
(3) **Exchangeability**: because the design object in DwO is modular in concept, it can be replaced by another design object, provided the interfaces of the two objects are

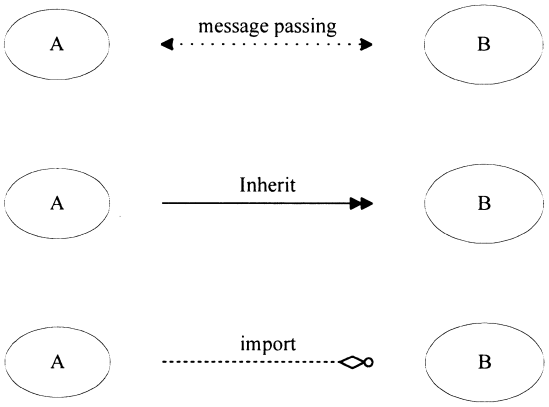


Figure 2 Object operators

similar. This would allow, for example, for the relatively straightforward upgrading of portions of a computerized design system.

While the above has described the underlying concepts behind design objects in DwO, such concepts are of limited use unless they can be incorporated into a design process model. Such a DwO design process model would have the potential advantages of design objects, described above, but would also possess some other potential advantages:

(1) **Improved Communication**: a design process model can potentially reduce the effort of translating thoughts from one design to another, and therefore reduce misunderstandings as an idea passes from one person to the next. This could result in an increase in productivity and quality.
(2) **Modeling the design process**: such modeling would allow for a greater understanding of the process involved and would therefore make the design task somewhat easier.

Research issues associated with the proposed design-with-objects approach

While design objects, and a DwO design process model, possess some important potential advantages, there are several research issues that remain to be addressed before such an approach can be used. These research issues are associated with the design objects (including the taxonomy, the content of the object library, the organization of design objects, and the issue of user-defined design objects), design object methods and the design process model formalism (adapted and extended from Kim[30]). Each of these issues is somewhat interrelated and is described in turn in this section.

Design objects

There are several research issues concerned with the selection and management of design objects. The first is concerned with the design object taxonomy and with the content of the design object library. Since the size of a design object library is limited by various factors, selecting an appropriate set of design objects is a research issue. Aspects to be considered include: which features should be included, how they should be organized, and whether user-defined features should be supported. A second research issue is associated with the organization of the objects to allow for their efficient use. An additional research issue is associated with the user-defined design objects. User-defined design objects would allow for a more flexible framework but present problems regarding validation and links with other objects.

Design object methods

An object design method is a mechanism for changing a design model. In order to apply an object design method, both the object to be manipulated and the model to which the method is applied should be specified. The research issue is therefore to specify the basis of the algebra inherent in methods, define basic methods, and show how methods can be defined, used and maintained.

A design process model formalism for design-with-objects

Perhaps the key research issue in DwO is associated with the question:

How can we use DwO to carry out design?

In other words, the key research issue is to show how a design process model can be formulated using DwO. Such a formalism would allow for the open representation of the design process and would be able to be computerized fairly readily. The design process model formalism would be built on a definition of a design model, design objects and design methods.

A DESIGN MODEL AND A DESIGN PROCESS MODEL FORMALISM FOR DwO

The above section has identified significant research issues in DwO associated with design objects, design object methods and in determining a suitable design process model formalism. This section outlines the basis for the DwO approach and this basis encompasses design objects, design object methods, the overall architecture for DwO and a design process formalism using DwO.

The design model S

A design model (S) is the representation of an artifact to be designed. For a DwO process then:

$$S = [o_i, .]$$

S represents a model that consists of one or more design objects. These design objects can be physical or non-physical objects as described above.

The exact form of S will vary with the design process model and with the artifact being designed. For example, for evolutionary design where the design goes through a number of generations and where there may be several design models at each generation, then

$$S_l^k = [o_{i,}^l .]^k = (o_{1,}^l ., o_{2,}^l ., o_{3,}^l \ldots\ldots o_{n,}^l .)^k$$

where S_l^k represents the design model l at generation k, and $o_{i,}^l$. represents a design object included in design model l at generation k, while n represents the number of design objects in the design model.

Design-with-objects methods

A DwO method is a mechanism for changing or creating one or more design objects. In order to apply a DwO method,

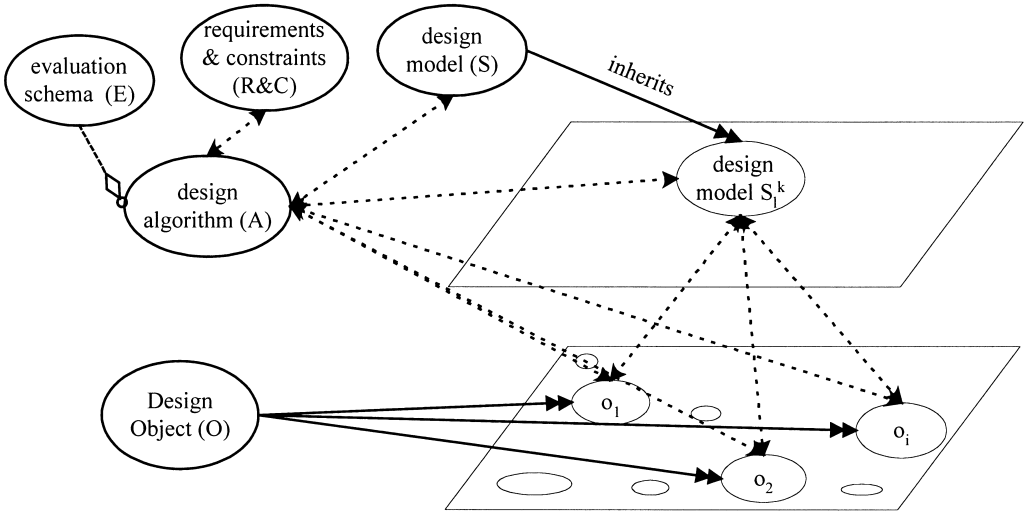


Figure 3 The overall architecture of DwO

both the design object(s) to be manipulated and the model to which the method is applied should be specified. We can define two *basic* DwO methods, namely *add* and *delete*, and these are defined as follows:

Definition (*Add* Method Φ_{add}): An *add* design object DwO method on a model S is a unit operator which adds exactly one design object o_i,. to the design model S.

$$\Phi_{add}(o_i, .) : S \rightarrow \text{New S where New } S = S + o_{i,} .$$

Definition (*Delete* Method Φ_{del}): A *delete* design object DwO method on a model S is a unit operator which deletes exactly one design object o_i,. from S

$$\Phi_{del}(o_{i,} .) : S \rightarrow \text{New S where New } S = S - o_{i,} .$$

It is readily apparent that any change in the design model S can be obtained by applying a finite number of these two basic DwO methods to S.

The flow of design operations in the proposed DwO approach is, in essence, represented in terms of DwO methods. A combination of DwO methods is an ordered set of DwO methods that are successively applied to the model.

The overall architecture of DwO

The central design process inherent in DwO can be represented as the architecture, shown diagrammatically in *Figure 3*, with five main types of objects involved: namely design models (S), design objects (O), design algorithms (A), requirements and constraints (R&C), and the evaluation schema (E). Object operators can express the relationship between these objects as follows:

Relation 1: instance of design model S_l^k ξ design model S
Relation 2: design algorithm A ω design model S
Relation 3: design algorithm A ψ evaluation schema E
Relation 4: instance of design object o_i ξ design object O
Relation 5: design model S_l^k ω o_i
Relation 6: requirements and constraints R&C ω design algorithm A
Relation 7: design algorithm A ω o_i
Relation 8: design algorithm A ω design model S_l^k

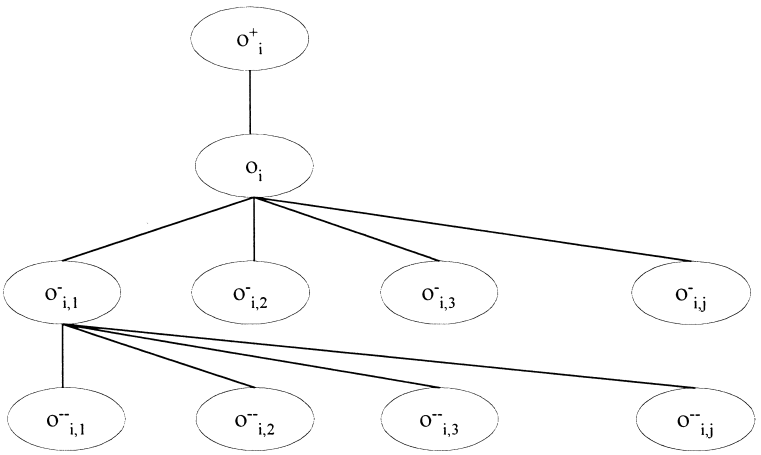


Figure 4 Hierarchy of design object instances

The architecture in *Figure 3* shows how the particular instance of a design model, S_l^k, is obtained from the design algorithm, evaluation schema, requirements, constraints and the design model object.

For pure formulation design (or creative design) a new design model object S is defined that describes the form of the model. A specific instance, S_l^k, of this design model can then be created. For pure parametric design then the design model object S has already been defined and the design process therefore only involves the determination of a specific instance, S_l^k, of this design model.

An important point to note is that additional objects can be defined within the overall architecture shown in *Figure 3*. For example, in the example below, the design algorithm object, A, includes a selection object, P. In particular, a design object, o_i, can be part of a tree structure with sub-objects, o_i^- and super-objects o_i^+ as shown in *Figure 4*. Each instance of the design object in the tree can have the DwO architecture, shown in *Figure 3*, associated with it. In this manner there is a hierarchy of instances of design objects with a DwO architecture potentially operating at each level of the hierarchy. It should also be noted that the sub-objects need not be separate physical objects but can be elements, such as features, of the instance of the design object.

Using DwO

The above has described DwO in terms of the basis of design objects, design object methods and the architecture of DwO. Form this basis it can then be possible to formulate different design process models. By doing this, the inference is that the resulting design process models would have the advantages described above, namely: computability, reusability, exchangeability, and improved communication.

With the design objects (with relations) and design methods defined, the DwO design process model can be thought of a *computable* model. The implication of this is that the design process model is not just a descriptive model but a computable model that computes new stages in the design process using the design methods described above. This will be illustrated by an example in the next section.

AN EXAMPLE DESIGN PROCESS MODEL FORMALISM USING DwO

As stated above, there are many differing views concerning the nature of the design process model. These views can be classified according to the activities involved, their perspectives, the nature of the design process, or the transformational view in which engineering design is regarded as a transformation from function to form. Alternatively, the design process may be viewed as a problem-solving process by searching through a state space, where the states represent the design solutions.

This section describes the application of the DwO approach to a specific example problem and uses a specific design process model. It should be noted that space does not permit a description of DwO applied to all possible classes of problems. Instead this section describes one example, but shows the procedure used so that it could be applied, with varying degrees of difficulty, to other design problems.

The example design problem considered

The design problem considered in this example design process model formalism is concerned with the designer selecting a combination of physical design objects from a set of candidate physical design objects to produce a design. This would occur, for example, where a designer had to choose from a number of available parts, subassemblies or modules to produce a design.

This can be stated formally as follows:

Given a set of candidate physical design objects $\{o_{i,}.\}$, at design iteration k, produce a design S_l^k that is composed of a subset of the candidate physical design objects and which satisfies both a set of functional requirements $\{R_{o,}.\}$ and a set of constraints $\{C_i\}$ while minimizing (maximizing) an objective function V_l^k , where $V_l^k = f(S_l^k)$.

Example design process model

Given the above problem definition, a number of design process models are possible that would aim to produce satisfactory designs. This section describes one instance of a DwO design process model, which consists of four steps (with associated sub-steps).

Step 1 (refinement)

Refine the set of initial design specification into an equivalent set of functional requirement $\{R_{o,p}\}$ such that each R_o,. is ready to be mapped into a subset of objects. $R_{o,p}$ is functional requirement o with attribute p. This step is described in Liang and O'Grady[70] and O'Grady and Liang[71].

Step 2 (design initialization)

A ψ E	(design algorithm A imports evaluation schema E)
A ω R&C	(design algorithm A gets data from requirements and constraints R&C)

The design algorithm, A, used in this example is described below.

The design model, S, for this example consists of a set of physical design objects [o_i,.].

$S_l^k \xi S$	(instance of design model S_l^k inherits from design model S)
$o_i \xi$ O	(instance of design object o_i inherits from physical design object O)

The physical design object, O, consists of a description of the artifact.

$$\{R_{o,}.\}_u = \{R_o,.\}$$

$$\{R_{o,}.\}_s = \text{Null}$$

Step 3 (function-object transformation)

In this step, each functional requirement is mapped into a set of design objects and the design objects are added to the instance of the design model by applying object design methods. This step involves selecting a functional requirement and then selecting a design object that satisfies the requirement. Let us assume for this example, that the

function to form relationship is 1:1.[44] When all functional requirements have been satisfied then the main design stage has been completed.

Step 3.1 If the set of unsatisfied requirement $\{R_o,.\}_u$ is empty, stop since the design is finished. Otherwise, arbitrarily select one requirement (R_i) from $\{R_o,.\}_u$.
Step 3.2 Find a set of design objects $\{o_i,.\}$ which satisfies the selected functional requirement ($R_{o,}.$) with each attributes. The selection can be done by an attribute matching process in a 1:1 relationship where attribute j in physical design objects $o_{i,j}$ satisfies attribute p in requirement $R_{o,p}$.
Step 3.3 If the set of design objects $\{o_i,.\}$ is empty, stop. In this case there is no design objects that will satisfy the selected functional requirement ($R_{o,}.$), and the design process can only proceed if either the selected functional requirement is relaxed or if alternative physical design objects can be made available.
Step 3.4 Object P is invoked. Object P contains the selection algorithm. For this example object P_1 is used which ranks the set of design objects $\{o_i,.\}$ into an ordered set $\langle o_i,.\rangle$, according to the evaluation scheme E, such that the $o_i,.$with higher ranking comes first. This ranking is based on the objective function V_l^k, where $V_l^k = f(S_l^k)$. Select the highest ranked $o_i,.$ Using add object methods (Φ_{add}), add the selected design objects into the current instance of the design model (S_l^k). This results in a new design model (New S_l^k).
Step 3.5 Checks that none of the set of constraints $\{C_i\}$ are violated when the method is applied. This procedure is called validity-preservation.
Step 3.6 If the model violates the constraints, undo Step 3.4 and delete $o_i,.$ from the ordered set of design objects $\langle o_i,.\rangle$. Repeat from Step 3.4 and choose o_i with the next rank.
Step 3.7 Delete the functional requirement ($R_{o,}.$) from the set of functional requirements $\{R_o,.\}_u$. Go to 3.1.

Steps 1–3 above constitutes the design algorithm A that underlies the design process model. The result of this process is that the instance of the design model obtained will be one that does not violate the set of constraints $\{C_i\}$, but the instance of the design model obtained may not have a high ranking based on V_l^k. A search mechanism is used in a later section to attempt to identify higher ranked designs.

EXAMPLE IMPLEMENTATION OF DwO — ELECTRONICS ASSEMBLIES

The DwO design process model described above can now be applied to an example product design problem as a particular instance of the more general design problem described in the previous section. The problem selected is that of electronics assemblies components selection (EACS), where it is desired to select a set of components that satisfies both a set of functional requirements $\{R_o,.\}$ and a set of constraints $\{C_i\}$ while minimizing (maximizing) an objective function V_l^k, where $V_l^k = f(S_l^k)$.

The work presented here is focused on how to represent EACS, where the participants may be remote, and how to implement it in a formal and systematic way.

The architecture of EACS design system

For the EACS design system each object in the overall architecture of DwO (*Figure 3*) is created as an ActiveX object, which can receive data over the Internet so that data can be in different locations, and updated by the different functional departments. The resulting overall architecture of the EACS design system is shown in *Figure 5* and is based on the object-oriented design system (OODS).[72] The implementation consists of a server with links to databases servers and clients that communicate with the server over the Internet. This particular implementation was carried out using a test-bed in the Iowa Internet laboratory (IIL) with multiple operating systems and computer platforms. For this example system, the users can access the EACS system through a conventional web browser.

The above has described how the main operational objects in OODS are defined. In addition to these operational objects, two further objects are defined in OODS, namely the user interface (UI) and configuration system (CS) objects. The UI object provides different interfaces for different design problems while the CS object allows the system to be customized for different design problems and domains.

Design scenario

Let us consider a scenario where a pulser/flasher electronic assembly is being designed and where the design has reached the stage where the schematic design aspect has

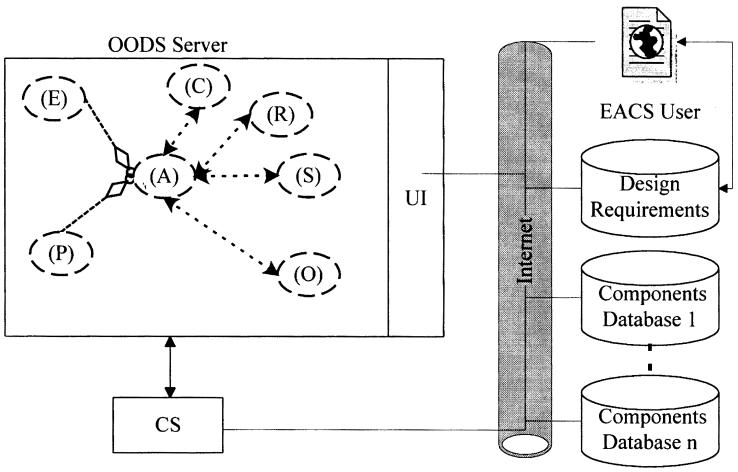


Figure 5 The architecture of the EACS design system

Table 2 Electronic components in pulser/flasher

Component	Type	Specification	X Position	Y Position
1	Resistor	200 (Ω)	1.2	0.5
2	Resistor	400 (Ω)	2.1	1.6
3	Capacitor	0.1 (μF)	1.4	2.3
4	Capacitor	0.01 (μF)	2.4	2.1
5	Capacitor	0.47 (μF)	3.3	2.6
6	Semiconductor	Timer (555)	2.4	1.5
7	Semiconductor	Emitting diode	3.1	1.1

been completed so that the broad specification of each component has been determined. In practice, these would usually be obtained from a computer system that develops a schematic outline of the electronics assembly. This pulser/flasher is composed of seven components including resistors, capacitors (for example 0.1 μF) and semiconductors (including a 555 timer) as shown in *Table 2*. The design problem now becomes that of EACS, as described above, to determine the set of specific components that meet the overall specification shown in *Table 2*, which form a set of design requirements, while still satisfying a set of constraints $\{C_i\}$ and while minimizing (maximizing) an objective function V_l^k, where $V_l^k = f(S_l^k)$.

The main objects involved are as follows:

The requirements (R) are of the form:

{component $_{1.\text{type}}$, component $_{1.\text{spec.}}$, component $_{1.x\ \text{position}}$, component $_{1.y\ \text{position}}$ …… component $_{i.\text{type}}$, component $_{i.\text{spec.}}$, component $_{\text{i.}x\ \text{position}}$, component $_{\text{i.y position}}$}
These are directly determined from *Table 2*.

The set of constraints {C} include such aspects as the maximum operating temperature, minimum operating temperature, component type, specified cost, and specified reliability. Also included in {C} are concurrent engineering constraints such as
Components should not be within 0.2 in. of the board edge (to allow for the equipment to work properly).
This can be expressed as an if-then rule as follows:

(component. x position-1/2 × component. length < = 0.2") or
(component. x position + 1/2 × component. length > = board width -0.2") or
(component. y position-1/2 × component. width < = 0.2") or
(component. y position-1/2 × component. width > = board length- 0.2") then
Violate Concurrent Engineering Constraint E224

Other concurrent engineering constraints are formulated in a similar manner.

The design model, S, includes the following design objects:

[component $_{1.\text{id}}$, component $_{1.\text{orientation}}$,…… component $_{\text{i.id}}$, component $_{\text{i.orientation}}$]
Where component $_{1.\text{id}}$ is the identification of the component and where component $_{\text{i.orientation}}$ indicates the orientation of the component. This is expressed as a Boolean variable of 1,0 indicating an orientation of 0 or 90 degrees respectively to allow for the usual restrictions imposed by electronic assembly equipment.

The evaluation schema (E) is taken to be, for this example, to find the lowest cost.

The set of design objects $\{o_{i,}.\}$ includes data on design objects and this is contained in vendor databases that may be scattered over the Internet. A segment of the aggregated component database (design object O) is shown in *Table 3*.

Example illustration

For the following example, let us consider a scenario where the overall constraint is represented by a budget of $2.8 with minimum reliability of 0.95. The maximum operating temperature is over 100° while the minimum operating temperature is 20°. The user can enter this data on the EACS user interface, which uses a standard web browser (*Figure 6*). The link between the user's input screen and the overall architecture of EACS is shown in *Figure 5*.

The design requirements, from *Table 2*, are placed in a database, which has been pre-connected to the EACS system using the CS object. The board size is taken to be 3 × 4 in.

Given the above scenario, example results using selection object P_1 (step 3.4 above) are shown in the screen of

Table 3 A segment of the component database

ID	Spec.	Length (in.)	Width (in.)	Max. Temp.	Cost	Vendor	…	…
Resistors								
R20030	200	0.3	0.09	100	0.25	…		
R20055	200	0.55	0.18	100	0.2	…		
R40055	400	0.55	0.18	100	0.3	…		
R20090	200	0.9	0.27	100	0.15	…		
Capacitors								
C01070	0.1	0.7	0.3	100	0.4	…		
C00180	0.01	0.8	0.25	100	0.15	…		
C04765	0.47	0.65	0.2	100	0.65	…		
Semiconductors								
SED039	Emitting diode	0.3	0.09	100	1.2	…		
SSD1142	Signal diode	0.55	0.18	100	0.8	…		
ST2206	Timer	0.8	0.8	100	0.55	…		

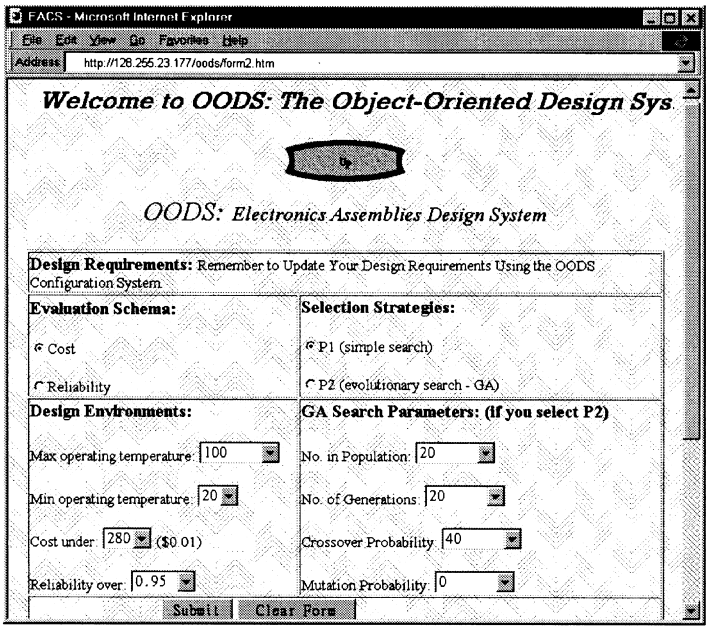


Figure 6 The input screen of the EACS system

Figure 7, with, for this example, a cost of $2.34 and a reliability of 0.952.

The use of an evolution search selection object P_2

Perhaps the main potential advantage of the DwO approach inherent in EACS is the exchangeability of an object, namely one or more objects can be easily replaced provided the interfaces remain with the same specification. To demonstrate this, in this section the original selection object P_1 is replaced with another selection object P_2 which uses an evolutionary search approach to attempt to identify a good, though not necessarily optimal, solution.

As indicated above, the use of selection object P_1 results in only obtaining a feasible solution, which may not be a particularly good solution since there is no attempt to achieve a good or optimal solution. In order to obtain a good or optimal solution, the use of complete enumeration might be considered but at the expense of a significant computational burden.

Selection object P_2 consists of a constrained genetic algorithm search mechanism.[70,33]

The nature of DwO allows for the exchangeability of objects so that we can simply exchange selection object P_2 for selection object P_1. This section describes a new selection object P_2, which can be used to replace P_1 for Step 3 during the design process model described in the previous section. To avoid confusion with P_1, P_2 sub-steps will be indicated by (P_2) attached to each sub-step.

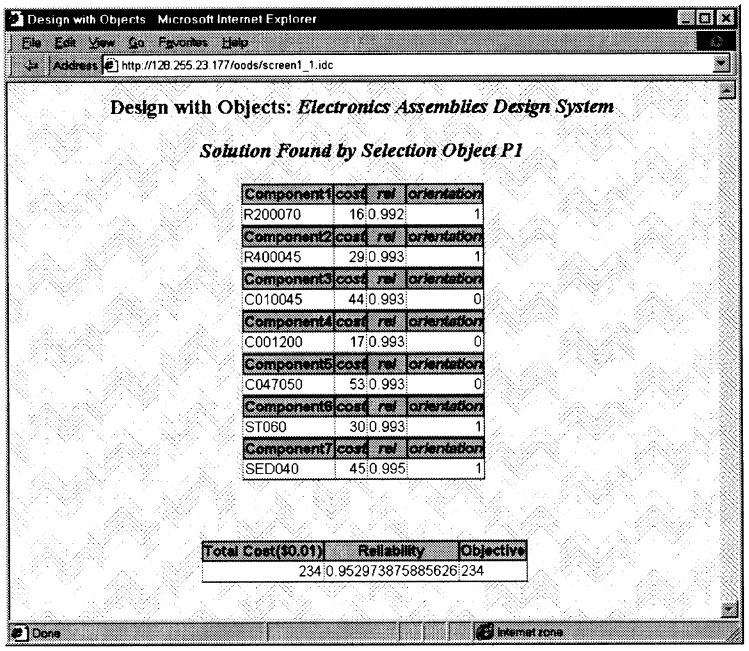


Figure 7 Example results obtained by EACS using selection object P_1

Step 3.1(P_2) Initialization

A variety of initialization schemes are possible, including random and user invoked schemes. The one presented here is based on P_1, with changes to ensure a random selection of electronics components. (Note that this step is different from Step 2 (Design Initialization) stated above. This step is to try to generate the initial population of design model. However before this step, Step 1 and Step 2 should be conducted in the exact same way as that stated above).

a) If $\{R_{o,}.\}_u$ = Null, then set $\{R_{o,}.\}_u = \{R_{o,}.\}$ and $\{R_{o,}.\}_s$ = Null. Add 1 to population number.
b) If population size = m + 1 (m is from user input), then go to Step 3.2(P_2).
c) Arbitrarily select one requirement ($R_{o,}$.) from $\{R_{o,}.\}_{u.}$ Find a set of electronics components $\{q_{i,}.(z)\}$ which satisfies the selected functional requirement ($R_{o,}$.) with each attributes. The selection can be done by an attribute matching process in a 1:1 relationship where attribute j in electronics component $q_{i,}.(z)$ satisfies attribute p in requirement $R_{o,p}$.
d) If the set of electronics components $\{q_{i,}.(z)\}$ is empty, **stop**. In this case there are no electronics component that will satisfy the selected functional requirement ($R_{o,}$.), and the design process can only proceed if either the selected functional requirement is relaxed or if alternative electronics components can be made available.
e) Randomly select one $q_{i,}.(z)$ from the set of electronics components $\{q_{i,}.(z)\}$. Using add object methods (Φ_{add}), add the selected electronics component into the current instance of the design model (S_1^k). This results in a new design model (New S_1^k).
f) Checks that none of the set of constraints $\{C_i(z)\}$ are violated when the method is applied. This procedure is called validity-preservation.
g) If the model violates the constraints, undo **e)** and delete $q_{i,}.(z)$ from $\{q_{i,}.(z)\}$. Then Repeat from **e)** and choose another $q_{i,}.(z)$. Otherwise go to **h)**.
h) Delete the functional requirement ($R_{o,}$.) from the set of functional requirements $\{R_{o,}.\}_u$. **Go to Step 3.1(P_2).**

Step 3.2(P_2) Selection

a) Add 1 to the generation number. If the generation number is greater than k, **stop**.
b) Evaluate each design model using the objective function V_1^k, where $V_1^k = f(S_1^k)$.
c) Use an appropriate selection strategy to select the new population.

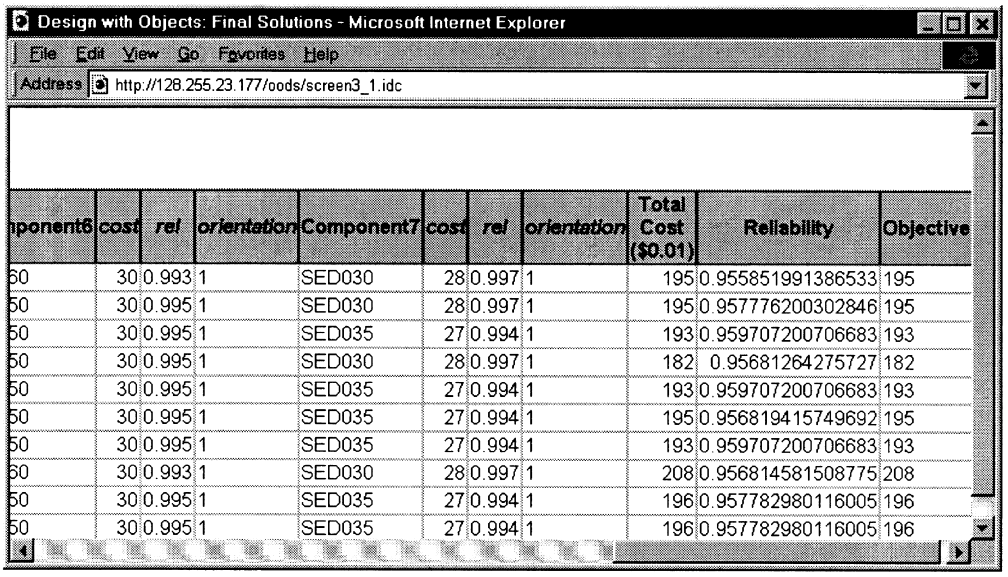
Step 3.3(P_2) Crossover

Apply the constrained crossover method[73] to the design models that are selected for crossover. In this proposed constrained crossover method, a crossover point is selected and components are exchanged between model one at a time and the validity is checked after each component is exchanged. If the result is infeasible then the exchange is cancelled and another component is selected. In this manner the final result will be a population of design models that are feasible.

Step 3.4(P_2) Mutation

Apply the constrained mutation method[73] to the design models that are selected for mutation. In this proposed constrained mutation method, a design model S_1^k and mutation point are selected. A member of the feasible subset of components then replaces the selected component and the validity of the resulting new design model is checked. If it is invalid then the replacement is cancelled and a new design model S_1^k and mutation point are selected. Go to **Step 3.2(P_2)**.

The result of using selection object P_2, shown in the screen of *Figure 8*, indicate that a family of design models (S_1^k) are obtained. As the evolutionary search mechanism goes though multiple iterations, there is the tendency to home in on improved component selections and, for this example, after 30 iterations the population



nponent6	cost	rel	orientation	Component7	cost	rel	orientation	Total Cost ($0.01)	Reliability	Objective
60	30	0.993	1	SED030	28	0.997	1	195	0.955851991386533	195
60	30	0.995	1	SED030	28	0.997	1	195	0.957776200302846	195
60	30	0.995	1	SED035	27	0.994	1	193	0.959707200706683	193
60	30	0.995	1	SED030	28	0.997	1	182	0.95681264275727	182
60	30	0.995	1	SED035	27	0.994	1	193	0.959707200706683	193
60	30	0.995	1	SED035	27	0.994	1	195	0.956819415749692	195
60	30	0.995	1	SED035	27	0.994	1	193	0.959707200706683	193
60	30	0.993	1	SED030	28	0.997	1	208	0.956814581508775	208
60	30	0.995	1	SED035	27	0.994	1	196	0.957782980116005	196
60	30	0.995	1	SED035	27	0.994	1	196	0.957782980116005	196

Figure 8 Portion of the example results obtained by EACS using selection object P_2

converges to a single type with a cost of $1.82 with a reliability of 0.957, which is an improvement over those obtained by selection object P_1. However these figures are illustrative only: the main principle is that of exchangeability in that the selection object P_1 has been readily exchanged with selection object P_2 .

This example has illustrated the application of DwO. An example DwO process model has been described and the implementation of this process model with different selection objects P_1 and P_2 has been overviewed. The implementation would indicate that the DwO is a computable model with reusable objects and that it allows for the ready exchange of objects.

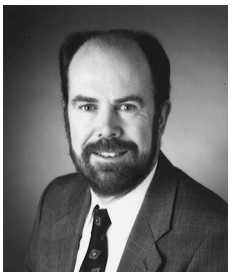
SUMMARY AND CONCLUSIONS

This paper has reviewed the background to design process models and object-oriented design and has then described the basis of DwO. Design models, DwO methods and an architecture for DwO have been described, together with an example design process model formalism that uses DwO. The implementation of this example design process model formalism that uses DwO for the problem of EACS has been described. This example uses the Internet to communicate between objects. The implementation would indicate that the DwO is a computable model with reusable objects and that it allows for the ready exchange of objects, although care should be exercised in extrapolating from a particular example to the general case. While much research remains to be done, the object-oriented approach used in DwO would appear to offer potential advantages in computability, reusability and exchangeability.

REFERENCES

1. Kannapan, S.M. and Marshek, K.M. Engineering design methodologies: a new perspective. In *Intelligent Design and Manufacturing*, ed A. Kusiak. Wiley, New York, 1992, pp. 3–38.
2. Bebb, H.B., Quality design engineering: the missing link to US competitiveness, keynote address. In *Proceedings of the NSF Engineering Design Conference*, Amherst, June 1989.
3. Gero, J.S., Design prototypes: a knowledge representation schema for design. *AI Magazine*, Winter 1990, 26-36.
4. Suh, P., *The Principles of Design*. Oxford, NY, 1990.
5. Shaw, M.C., Creative design . *CIRP Annals 35*, 1986, **35**(2), 461–465.
6. Asimow, M., *Introduction to Design*, Prentice-Hall, Englewood Cliffs, NJ, 1962.
7. Buhl, H.R., *Creative Engineering Design*. Iowa State University Press, IA, 1962.
8. Fasal, J., *Force Decisions for Value*. Product Engineering, McGraw Hill, NJ, 1965, pp. 84–86.
9. Zwicky, F., The morphological method of analysis and construction. *Courant*, Anniversary volume, Interscience Publishers, New York, 1948, pp. 461–470.
10. Harrisberger, L., *Engineeringship: A Philosophy of Design*. Brooks/ Cole, Belmont, CA, 1966.
11. Glegg, G.L., *Design of Design*. Cambridge University Press, New York, 1960.
12. Woodson, T.T., *Introduction to Engineering Design*. McGraw-Hill, New York, 1969.
13. Alger, R.M. and Hays, C.V., *Creative Synthesis in Design*. Prentice-Hall, Englewood Cliffs, NJ, 1964.
14. Hill, P.H., *The Science of Engineering Design*. Holt, Rinehart and Winston, New York, 1970.
15. Ostrofsky, B., *Planning and Development Methodology*. Prentice-Hall, Englewood Cliffs, NJ, 1997.
16. Starr, M.K., *Product Design and Decision Theory*. Prentice-Hall, Englewood Cliffs, NJ, 1963.
17. Booch, G., *Object Oriented Design with Applications*. Benjamin/ Cummings, New York, 1994.
18. Kusiak, A., Szczerbicki, E. and Vujosevic, R., Intelligent design synthesis: an object-oriented approach. *International Journal of Production Research*, 1991, **29**(7), 1291–1308.
19. Takeda, H., Tomiyama, T. and Yoshikawa, H., A logical and computable framework for reasoning in design In *Design Theory and Methodology — DTM'92*, ASME, 1992.
20. Dopplick, T., *A Science User's Guide to the EOSDIS Core System (ECS) Development Process, Technical Paper 160-TP-003-001*. Science Office, EOSDIS Core System Project, 1995.
21. Coad, P. and Yourdon, E., *Object-Oriented Design*. Yourdon Press, New Jersey, 1991.
22. Liu, C., *Smalltalk, Objects, and Design*. Hardbound, Greenwich, CT, 1996.
23. Whitney, D.E., Designing the design process. *Research in Engineering Design*, 1990, **2**, 3–13.
24. Waldron, M.B., *Modeling of the design process, intelligent CAD, I: proceedings of the IFIP TC 5/WG 5.2*, Workshop on Intelligent CAD, ed H. Yoshikawa, D. Gossard. Boston, MA, October 1987, pp. 13–27.
25. Akman, V., Hagen, P.J.V.ten and Tomiyama, T., A fundamental and theoretical framework for an intelligent CAD system. *Computer-Aided Design*, 1990, **22**(1), 352–367.
26. Simon, H.A., *The Sciences of the Artificial*. MIT Press, Cambridge, MA, 1969.
27. Bardasz, T. and Zeid, I., Cognitive model of memory for mechanical design problems. *Computer Aided Design*, 1992, **24**(6), 327–342.
28. Finger, S., Fox, M.S., Prinz, F.B. and Rinderle, J.R., Concurrent design. *Applied Artificial Intelligence*, 1992, **6**, 257–283.
29. Hakim, M. and Garrett, J., An object-centered approach for modeling engineering design products: combing description logic and object-oriented modeling. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 1997, **11**(3), 187–198.
30. Kim, C., *A representation formalism for feature-based design as a formal concurrent engineering approach*, Ph.D. Thesis. North Carolina State University, Raleigh, NC, 1994.
31. Smithers, T., AI-Based design versus geometry-based design or why design cannot be supported by geometry alone. *Computer-Aided Design*, 1989, **3**, 141–150.
32. Maher, M.L., Process Models for Design Synthesis. *AI Magazine*, Winter, 1990, 49–58.
33. Silverman, B.G. and Mezher, T.M., Expert critics in engineering design: lessons learned and research needs. *AI Magazine*, Spring 1992, 45–62.
34. Kota, S. and Ward, A. C., Functions, structures, and constraints in conceptual design. In *Proceedings of the 2nd International Conference on Design Theory and Methodology*, ed I. Rinderle. ASME Press, New York, 1990, pp. 239–250.
35. Kusiak, A. and Szczerbicki, E., A formal approach to specifications in conceptual design. *ASME Journal of Mechanical Design*, 1992, **114**(12), 659–666.
36. Pahl, G. and Beitz, W., *Engineering Design*. Springer-Verlag, New York, 1988.
37. Finger, S. and Dixon, J.R., "A Review of Research in Mechanical Engineering Design. Part 1: Descriptive, Prescriptive, and Computer-Based Models of Design Processes", *Research in Engineering Design*, 1989, **1**, 51–67.
38. Kott, A.S. and May, J.H., Decomposition vs. transformation: case studies for two models of the design process. In *Proc. of ASME Computers in Eng Conf., Assoc. of Mechanical Engineers, vol. 1*. New York, 1989, pp. 1–8.
39. Yang, J. and Rozenblit, J.W., Case studies of design methodologies: a survey. In *Proceedings of AI, Simulation and Planning in High Autonomy Systems*, ed J. Rozenblit. March 1990, pp. 136–141.
40. Mantyla, M., A modeling system for top-down design of assembled products. *IBM Journal for Research and Development, Sept 34*, 1990, **5**, 636–659.
41. Hubka, V., In *Principles of Engineering Design*, trans. ed Eder, W.E. Butterworth Scientific, London, 1982.
42. Eastman, C.M., Bond, A.H. and Chase, S.C., A formal approach for product model information. *Research in Engineering Design*, 1991, **2**, 65–80.
43. Libardi, E.C., Dixon, J.R. and Simmons, M.K., Computer environments for the design of mechanical assemblies: a research review. *Engineering with Computers*, 1988, **3**, 121–136.
44. Suh, N.P., Basic concepts in design for producibility, keynote paper. *Annals of the CIRP*, 1988, **37**(2), 559–567.
45. Hoover, S.P. and Rinderle, J.R., A synthesis strategy for mechanical devices. *Research in Engineering Design*, 1989, **1**, 87–103.
46. Fenves, S.J. and Baker, N.C., Spatial and Functional Representation Language for Structural Design. In *Expert Systems in Computer-Aided Design*, ed J. Gero. Elsevier Science Publication, North Holland, IFIP, 1987.

47. Lai, K. and Wilson, W.R.D., FDL: A language for function description and rationalization in mechanical design. In P*roceedings of Computers in Engineering Conference*, New York, 1987, pp. 87–94.
48. Ulrich, K. and Seering, W., In *Conceptual Design: Synthesis of Systems Components, Intelligent and Integrated Manufacturing Analysis and Synthesis*, ed C.R. Liu, A. Requicha, S. Chandrasekar. ASME, New York, 1987, pp. 57–66.
49. Westerberg, A.W., Piela, P., Subrahmanian, E., Pondar, G. and Elm, W., A future computer environment for preliminary design. In *Foundations of Computer-Aided Process Design*, ed I. Siirola, J. Grossman, G. Stephanopoulos. Elsevier, Amsterdam, 1990.
50. Gu, P. and Chan, K., Product modeling using STEP. *Computer-Aided Design*, 1995, **27**(3), 163–179.
51. ISO TC184/SC4, STEP Part 11, 21, 42, 203, ISO 10303, 1994.
52. European Committee for Standardization (ECN), An evaluation of CIM modeling constructs: evaluation report of constructs for views according to ENV 40 003. *Computers in Industry*, 1994, **24**(2–3), 159–236.
53. Chen, P.P.-S., The entity-relationship model-toward a unified view of data. *ACM Transactions on Database Systems*, 1976, **1**, 9–36.
54. Codd, E.F., A relational model of data for large shared data banks. *Communication of the ACM* , 1970, **13**, 377–387.
55. Kusiak, A., Letsche, T. and Zakarian, A., Data modelling with IDEF1x. *International Journal of Computer Integrated Manufacturing*, 1997, **10**(6), 470–486.
56. Minsky, M., A framework for representing knowledge. In *The Psychology of Computer Vision*, ed P. Winston. McGraw-Hill, New York, 1975.
57. Champeaux, D. de, Lea, D. and Faure P., *Object-Oriented System Development*. Addison Wesley, Reading, MA, 1993.
58. Dahl, O. and Nygaard, K., Simula: an algol-based simulation language. *Communication of the ACM*, 1966, 9.
59. *The Random House College Dictionary*. Random House, New York, 1975.
60. Buhr, R., *Machine charts for visual prototyping in system design*. Technical Report 88-2, Carlton University, 1988.
61. Veth, B., An integrated data description language for coding design knowledge. In *Intelligent CAD Systems I — Theoretical and Methodological Aspects*, ed P.J.W. ten Hagen, T. Tomiyama. Springer, Berlin, 1987.
62. Tomiyama, T., Kiriyama, H., Takeda, H. and Xue, D., Metamodel: a key to intelligent CAD systems. *Research in Engineering Design*, 1989, **1**(1), 19–34.
63. Yoshikawa, H., General design theory and a CAD system. In *Man-Machine Communication in CAD/CAM, Proceedings of the IFIP Working Group 5.2 Working Conference*, ed T. Sata, E.A. Warman. North-Holland, Amsterdam, 1981.
64. Ullman, D.G., Dietterich, T.G. and Stauffer, L.A., A model of the mechanical design process based on empirical data: a summary. In *Artificial Intelligence in Engineering Design*, ed J.S. Gero. Elsevier, Amsterdam, 1988.
65. Batory, D.S. and Kim, W., Modeling concepts for VLSI CAD objects, In *Proceedings of ACM-SIGMOD International Conference on Management of Data*, TX, USA, 1985. p. 446.
66. Biliris, A., *A data model for engineering design objects*, 2nd Intl. Conference on Data and Knowledge Systems for Manufacturing and Engineering, IEEE, 1989. pp. 49–58.
67. Kemper, A. and Moerkotte G., *Object-Oriented Database Management*. Prentice Hall, Englewood Cliffs, NJ, 1994.
68. Fauvel, O., Object-oriented design for manufacture. *Journal of Intelligent Manufacturing*, 1994, **5**, 1–11.
69. Moriwaki, T. and Nunobiki, M., Object-oriented design support system for machine tools. *Journal of Intelligent Manufacturing*, 1994, **5**, 47–54.
70. Liang, W.Y. and O'Grady, P., Genetic algorithm for design for assembly: the remote constrained genetic algorithm. *Computers and Industrial Engineering*, 1997, **33**, 593–596.
71. O'Grady, P. and Liang, W. Y., Remote Collaborative Design and Manufacture With Modules: Internet-Based Collaboration. In *Proceeding of 4th International Conference on Concurrent Enterprising*, October 1997, Nottingham, UK, pp 315–326.
72. Liang, W.Y. and O'Grady, P.J., *OODS: An Object Oriented Design System*, technical report. The Iowa Internet Lab, University of Iowa, 1998.
73. O'Grady, P. and Liang, W.Y., *An Object Oriented Approach to Design with Modules*, technical report. The Iowa Internet Lab, University of Iowa, 1998.



Peter O'Grady is a Professor and Chair of Industrial Engineering at University of Iowa. He received his bachelors degree from University of Cambridge and his Ph.D from University of Nottingham. His main areas of expertise are in Manufacturing Engineering and include Manufacturing Analysis, Manufacturing Systems, Computer Integrated Manufacturing, Manufacturing Control, and Concurrent Engineering. He has substantial management and administrative experience. He has been Editor, Design, Transactions of Institute of Industrial Engineers; and USA Editor of Journal of Design and Manufacturing; and is Associate Editor of Journal of Manufacturing Systems; a member of the Editorial Board of International Journal of Production Planning and Control; a member of the Editorial Board of Journal of Computers and Industrial Engineering; and a member of the Editorial Board of Journal of Engineering Design and Automation.

Dr. Wen-Yau Liang is currently Post-Doctoral Research Fellow of the Iowa Internet Laboratory with active research activities in collaborative systems for design and manufacturing, Electronic Commerce and large-scale logistics. His research interests are remote collaborative design for modular products, distributed design, object-oriented design, and product visualization. He received Ph.D. in industrial engineering from the University of Iowa in 1998 and MBA in industrial management from the National Cheng-Kung University of Taiwan in 1992. He has several papers published in related journals.