An Object-Oriented Approach to the Concurrent Engineering of Electronics Assemblies

Wen-Yau Liang\textsuperscript{1} Peter O’Grady\textsuperscript{2}

1 Department of Information Management
Da-Yeh University
112, Shan-Jeau Rd., Fu Hsing, Ta Tsun,
Changhua, Taiwan, R.O.C.
Tel: 886-4-8528469
Fax: 886-4-8528466
E-mail: wyliang@mail.dyu.edu.tw

2 Department of Industrial Engineering
University of Iowa
Iowa City, Iowa 52242, USA
Tel: +1 319 358 2979
Fax: +1 319 359 2454
E-mail: peter-ogrady@uiowa.edu

An Object-Oriented Approach to the Concurrent Engineering of Electronics Assemblies

1. ABSTRACT

An electronics assembly (EA) can be regarded as the backbone of electronic or Electro-mechanical products, where its functions are implemented by combining components and their interconnections on a substrate plate. The design of EAs is relatively complex and encompasses the consideration of many diverse considerations. This paper is concerned with the central area of electronic assemblies component selection (EACS), and with considering constraints at this stage to avoid multiple repetitions of the design process. The main task is to take the requirements and constraints, together with a set of possible electronics components, and then to select a subset of these components to satisfy the requirements (functional, physical…) and constraints, while minimizing or maximizing the objective function. The use of such EACS promises substantial benefits but its successful implementation is hampered by the lack of a suitable formalism, particularly where, as is often the case, participants are geographically separated. The work presented in this paper is focused on how to represent EACS, where the participants may be remote, and how to implement it in a formal and systematic way. A representation formalism is proposed for that purpose. This paper first overviews the EACS process. A formalism for EACS is then presented. An Internet network based implementation of this formalism is described that uses the Internet to support EACS, and an example is used to illustrate the implementation. The main contributions of this paper are threefold. First, the design domain of electronics assemblies is described. Second, a formalism for EACS is presented. Third, the use of this formalism is illustrated with an Internet-based implementation showing how the formalism can be used for a specific problem.

2. SYMBOLS AND NOTATION

{ } set
[ ] subset
[ ] feasible subset
<> order set
{ }u unsatisfied set
{ }s satisfied set
C_i constraints
k generation/iteration
z location of design object
\( o_{ij}^l (z) \) design object i with attribute j from location z selected at design model l
\( M_{ij}^l (z) \) design module i with attribute j from location z selected at design model l
n number of design objects in \( S_i^k \)
m number of design models in population
\( R_o^p \) functional requirement o with attribute p
3. INTRODUCTION

An Electronics Assembly (EA) is a part of an electronic or electro-mechanical system. An electronics assembly is a backbone of electronic or electromechanical products, where its functions are implemented by combining components and their interconnections (conductors) on a substrate plate. It provides a rigid base for mounting electronics components and conductors for connecting them. The most significant role of EAs is that they separate electrical functions from other parts of the system, giving modularity to the system. The design of EAs is relatively complex and encompasses the consideration of many diverse considerations and this complexity is compounded by the relatively high rate of technology change in EAs where new technologies continue to emerge at a seemingly increasing rate.

A typical EA design process can be described as follows [Ohr, 1990; Byers, 1991], Figure 1. A designer starts by analyzing a schematic diagram (*engineering analysis*). The schematic symbols in the diagram are converted to physical components (*component selection*) and the components are placed on a board (*placement*). Based on the selected components, board characteristics such as size, thickness, and type are determined (*board selection*). Then, conductors are routed between the components and other elements (*conductor routing*). The next step is to finally check the design (*evaluation and checking*). Ideally at each stage, the design is checked against a set of constraints that concern various Concurrent Engineering activities such as assembly, printing, drilling, and testing. If some design rules are violated, the design is modified and the design rules are checked again. This iterative process continues until the result is satisfactory.

![Figure 1: The EA Design Process](image)

The primary input to EA design is a schematic diagram, which is a schematic capture of the circuit information. The main output is a master artwork, which is a scaled configuration of the EA and from which the master pattern is photographically produced [Lindsey, 1979]. Most commercial CAD
system (Such as Mentor by Mentor Graphics, CADSTAR for Windows and VISULA NT by Zuken-Redac,) can fully perform this design process and manually select the parts/components from component library, which intend to be genetic. Recently, Nakamura and Kikuno (1998) and Smith etc., (1998) also have done related research. A simple transformation mechanism and a large number of alternatives characterize the transformation process. However, the number of possible mapping may be large since several vendors often supply various types of components which perform the same functionality. Typically, the solution space is continuous in most dimensions and the solution points are densely populated. However, the shape of the solution space is usually complicated by a large number of manufacturing constraints, and designers will then have to return to earlier design stages to make some necessary changes. One repercussion of this is that design can therefore involve a significant number of iterations of the design process shown in Figure 1 so that considerable time and effort is expended in achieving a satisfactory design.

One way of avoiding this is for such considerations to be brought to bear earlier in the design process, such as at the component selection stage. The area of component selection is therefore of considerable importance, and an effective component selection process will assist in (Turino, (1990):

1. Developing a product that can be readily accessed serviced and tested.
2. Lowering the cost of developing special test equipment, adapters, and fixtures.
3. Reducing the number of design iterations.
4. Avoiding unnecessary production test and field service time and cost.

This paper is concerned with this area of component selection, and with considering constraints (note that the constraints are not only the functional such as performance, cost, reliability but also physical such as board size and temperature) at this stage to avoid multiple repetitions of the design process shown in Figure 1. For example, one design challenge is making sure that test pins are accessible by the test equipment. This can cause problems if this aspect is only considered at later stages (usually in the Evaluation and Checking stage of the design process shown in Figure 1) since if there is a problem with accessibility then the designer has to return to earlier design stages to rectify the problem. The aim of the work in this paper is to consider, as far as possible, such aspects as part of the EACS problem, and therefore to minimize unnecessary design iterations.

The main task of this enlarged component selection can be thought of as to take the requirements and constraints, together with a set of possible electronics components, and then to select a subset of these components to satisfy the requirements and constraints, while minimizing or maximizing an objective function. The main research issue addressed in this paper is thus to develop a formalism to address the fundamental problem:

Given a set of candidate electronics components \( \{q_i(z)\} \), at design iteration \( k \), produce a design \( S^k \) that is composed of a subset of the candidate electronics components and which satisfies both a set of functional requirements \( \{R_t\} \) and a set of constraints \( \{C_i(z)\} \) while minimizing (maximizing) an objective function \( V^k \), where \( V^k = f(S^k) \).
This design problem can be termed *electronics assemblies components selection (EACS)*.

The requirements for such a formalism include a complete, integrated, and systematic sharing, in real-time, of design, engineering, and manufacturing knowledge (Guichelear, 1993; Kusiak, 1993). The key in determining the needs of an EACS System are related to the availability of applications that are platform independent. In general, an effective EACS System should run on platforms with the capabilities of replicated and linked multimedia information, responsive and reliable communication networking, discussion management over time and space, and common and consistent user-interface characteristics (Marca and Bock, 1992). Thus another important element in an EACS system is related to the fact that data and information can be distributed over the entire system. An effective EACS system would have a formalism that recognizes the location of the data and information.

A promising approach to the research issue above 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" (Booch, 1994).

Work done on object-oriented design systems includes that by Veth (1987), Tomiyama et al. (1989), Ullman et al. (1988), Takeda et al. (1992), Kusiak et al. (1991), Dopplick (1995), Fauvel (1994), Moriwaki and Nunobiki (1994), Hakim and Garrett (1997). While the potential advantages of object-oriented design has been recognized (Coad and Yourdon, 1991; Liu (1996)), the previous object-oriented design systems appear to have been built for the most part in an *ad-hoc* manner for a specific design domain and are not based on a formal methodology. Therefore, the results are difficult to extend to other design domains.

The work presented in this paper is focused on EACS, where the participants may be remote, and how to implement it in a formal and systematic way. A representation formalism is proposed for that purpose. This paper first overviews the EACS process. A formalism for EACS is then presented. This formalism is derived from an object-oriented approach in what is called Design with Objects (DwO), where electronics components are represented as objects. The implementation of this approach to EACS is then overviewed by a description of a system, conveniently called EACS, that uses the above formalism. The EACS system uses an Internet-based implementation to allow for geographically dispersed participants.

**4. ELECTRONICS ASSEMBLIES DESIGN**

As indicated above, a typical EA design process is shown in Figure 1. Much EA design is driven by various Concurrent Engineering issues (the constraints in Figure 1) which include assemblability, solderability and testability. These issues are summarized in Ginsberg (1990), Skinowicz (1981) and Traister (1990), Wakerly (1994), Abadir and Kapur (1997), Bhavsar and Edmondson (1997). This section gives an overview of the important area of testability and how consideration of testability leads to a number of design rules and Concurrent Engineering constraints.
4.1. Electronics Assemblies Test

Testing is one of the important steps of the manufacturing process of EA board. Indeed, it has been estimated that the cost of testing a typical EA board is about one-third of the total manufacturing cost (Arabian, 1989). It is generally recognized that any defects in EA design should be found in the earlier stages of a manufacturing process. The reason for the early fault detection requirements is the economics of test and repair. It is costly to let a fault pass on to later stages of product manufacturing or support. The philosophy is "fix it now, or suffer far more expensive problems in the downstream" (Parker, 1989). For example, to find a defective part at incoming inspection might cost about $0.50. But, the cost of finding and replacing the same part, once the board is passed on to the final system assembly, would amount to $50.00 (GenRad, 1985). Further testing cost information can be found in Al-hayek et al. (1997) and Pynn (1997).

In order for an EA board to function properly in the final assembly to which it is inserted, the components and their interconnections should be free of defects. In addition, since the components work together with other components to achieve the functionality that a board designer intended, the interactions between them should be also fault-free. The testing process associated with EA board manufacture occurs at a variety of stages including components test, bare-board test, and loaded-board test. One aim, as stated earlier, is to identify faults as early as possible to reduce the cost of rectifying the problem. In the stage of components test, the logical behavior of an active device, such as an integrated circuit, is tested and this is usually done in incoming inspection. The pins of the component are connected to the access channels of Automatic Test Equipment (ATE), and a set of input signals are applied (Williams and Parker, 1982; Gloster and Brglez, 1989). In the bare board test, several properties of the conductive paths such as the width of traces and trace spacing are checked. Although the bare board test has been traditionally performed by human inspectors or a simplified in-circuit tester, AOI (Automatic Optical Inspection) systems have been developed and used with the advances in machine vision technology (Ye and Daniellson, 1988; Hara et al, 1983; Griffin et al, 1990). For loaded boards, two test methods have been widely used: a functional test and an in-circuit test (ICT). Recently the use of an alternative approach called Boundary Scan Architecture (BSA) has been promoted. The functional test verifies that a board works properly. Connecting its edge connectors to the test equipment, and applying a set of input signals does this test. Although this test can cover all types of board faults from manufacturing faults to the errors in logic design, one serious drawback is its poor diagnosibility. In other words, it is very difficult to pinpoint the exact cause of the fault down to the component level. This means that the functional test is ideal for pass/fail test, but not for repair driving or for process monitoring and this has resulted in the popularity of ICT.

4.2. Design-for-Testability Approaches

An ICT helps to ensure that there are no shorts or opens on the EA board and that the components function as intended. However, the increasing density of EA boards has resulted in accessibility to test
points has become a major problem. This has prompted efforts to ensure that test considerations are incorporated directly into the components. Also, to ensure the thermal and mechanical properties of the board the components selection is critical important before actually put them on the board. Therefore, it may be practical to check the design with design rules during the design process (components selection) and making recommendations on improvements that can be made to the design to improve its testability and assemblability. Since the main testability problem is accessibility, most design rules concern providing reliable test points or access points. These rules can be found from several sources (Kim, 1994; Barnes, 1983; Kakani, 1987; Bullock, 1987; Solberg, 1989; Conti, 1987a, 1987b; Turino, 1990; Abramovici et al., 1990; Aitken, 1995; Wakerly, 1994). It should be noticed that the design rules are heavily dependent on the test equipment and methodology (Wakerly, 1994; Bhavsar and Edmondson, 1997; Abadir and Kapur, 1997). Table 1 shows some example of EA design rules that apply to component selection and placement.

<table>
<thead>
<tr>
<th>No.</th>
<th>Design Rules</th>
<th>Reason/Explanation</th>
<th>Area</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Components should not be located within 1 inch area from the guide edge.</td>
<td>Various methods used to locate printed circuit boards on the work board holder</td>
<td>Component Placement</td>
</tr>
<tr>
<td>2</td>
<td>Smaller case sizes (&lt; 0.12 inch by 0.060 inch) should be avoided.</td>
<td>It is desirable to minimize pick and place difficulties due to small size.</td>
<td>Components Selection and Placement</td>
</tr>
<tr>
<td>3</td>
<td>Components should be located with the same orientation if possible.</td>
<td>It is desirable to minimize changing equipment.</td>
<td>Component Placement</td>
</tr>
<tr>
<td>4</td>
<td>Pads should be at least 0.2 inch away from the board edge.</td>
<td>For easy handling and good plating and etching</td>
<td>Conductor Routing</td>
</tr>
</tbody>
</table>

Table 1: Example EA Design Rules

5. DWO: THE DESIGN MODEL AND DESIGN PROCESS

This section describes a proposed object-oriented formalism, called DwO, that can be used for design problems such as that for EACS. The proposed DwO approach and the use of EACS are described. This leads to a process model for EACS that aims to show how EACS can be carried out. We can view all entities associated with design as objects, design thus becomes Design with Objects (DwO). This means that the disparate entities associated with design, including design artifacts, design processes, design algorithms, and design data, are each regarded as objects. Each design object can contain methods (procedures that act on design objects), data (a named attribute of a design object) and interfaces (the basic structure and interface of a design object).

Design objects are related to each other by the following three object operators which are Inheritance, Import and Message passing.

Such a use of design objects has several potential advantages. The first potential advantage is that of computability where design can be thought of a computable design process model. The implication 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. The second potential advantage is that of reusability where, once a design object has been established, it can be reused. This is allied with the third potential advantage namely that of exchangeability where the modular nature of objects allows for an object to be replaced by another object with a similar

5.1. The Design Model S and Other Design Objects

DwO views the design model (S) as a central element of design. S is the representation of an artifact to be designed and represents a model that consists of one or more design objects, o_i, that can be physical or non-physical objects. From a DwO perspective:

\[ S = [o_i(z)] \]

Where z is the location of the item and is a variable of the form of a numeric, symbolic and/or Boolean data type.

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

\[ S^k = [o^k_1(z), o^k_2(z), o^k_3(z), \ldots, o^k_n(z)] \]

Where S^k represents the design model l at generation k, and o^k_i(z) represents a design object from location z included in design model l at generation k, while n represents the number of design objects in the design model. Note that the design objects may come from differing locations, z.

5.2. DwO Methods and Design Process Models

A DwO method is a mechanism for changing or creating one or more design objects. In order to apply a DwO method, both the design object(s) to be manipulated and the model to which the method is applied should be specified. Two basic DwO methods can be defined, namely add and delete, and 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 design objects in DwO can be used to develop a design process model for a particular design problem and domain. Example design process models are described in Liang and O'Grady (1997, 1998) and O'Grady and Liang (1997)
5.3. Object Decomposition

The structure in Figure 2 shows how a particular instance of a design model, $S^k$, 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, $S^k$, 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, $S^k$, of this design model (Liang and O’Grady, 1998).

An important point to note is that additional objects can be defined within the overall architecture shown Figure 2.

5.3.1. Using DwO for EACS

The above has described DwO in terms of the basis of design objects, features, design object methods and the structure of DwO. From 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. In the following problem, we regard electronics components $\{q_i(z)\}$ as a design object $o_i(z)$.
The central problem of EACS, as stated in the Introduction, is:

Given a set of candidate electronics components \{q_{i}(z)\}, at design iteration \(k\), produce a design \(S_{l}^{k}\) that is composed of a subset of the candidate electronics components and which satisfies both a set of functional requirements \(\{R_{o}\}\) and a set of constraints \(\{C_{i}(z)\}\) while minimizing (maximizing) an objective function \(V_{l}^{k}\), where \(V_{l}^{k} = f(S_{l}^{k})\).

Where the constraints \(\{C_{i}(z)\}\) which encompass the wide range of considerations and constraints that have to be brought to bear in the EACS problem, as indicated in the Introduction. These considerations and constraints include such diverse elements as:

- Board characteristics such as size, thickness, type.
- Engineering requirements such as operating temperature, cost, and reliability.
- Concurrent Engineering constraints such as those that affect assemblability and testability of the EA.

For example the accessibility requirement may involve not placing components in the outer inch of a board where test equipment may be unable to reach. This can be stated, as in design rule No.1 of Table 1, as:

"Components should not be located within 1 inch of the guide edge"

This can be expressed as the following if-then rule:

\[
\text{If component. } x \text{ position-1/2*component.width} \leq 1 \text{ then}
\]

\[
\text{Violate Concurrent Engineering constraint D102}
\]

These Concurrent Engineering constraints then can be evaluated during the EACS process (see Step 3.5 below), and any violations rectified at this stage.

For this work, the design model \(S_{l}^{k}\) is made up of a sub-set of electronics components \[q_{i}(z)\]. The problem then becomes one of selecting the sub-set of electronics components \[q_{i}(z)\], subject to the set of constraints \(\{C_{i}(z)\}\) and to minimize an objective function \(V_{l}^{k}\).

5.4. EACS Design Process Model

This section describes one instance of a EACS design process model, using DwO as described above, which consists of Steps 1 - 3 (with associated sub-steps), which are described in this section. Note that this is one possible design process model, and the purpose of this design process model is to show how such models can be formulated.

Step 1 (Refinement)

Refine the set of initial design specification into an equivalent set of functional requirement \(\{R_{o,p}\}\) such that each \(R_{o,p}\) is ready to be mapped into a subset of electronics components \(\{q_{i}(z)\}\). \(R_{o,p}\) is functional requirement o with attribute p. The details of this step is described in Liang and O’Grady (1997, 1998) and O’Grady and Liang (1997).
Step 2 (Design Initialization)

design algorithm A imports evaluation schema E

design algorithm A gets data from requirements R

design algorithm A gets data from constraints C. The design algorithm, A, used in this design
process model described below.

The design model, S, consists of a design object which, in turn, consists of a set of electronics
components \{q_{i,z}\}.

instance of design model \(S^k\) inherits from design model S

instance of electronics component \(q_{i,z}\) inherits from design object O

\{R_{o,}\}_u = \{R_{o,}\}

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

Step 3 (Selection Object P)

In this step, Object P (Figure 2), containing the selection algorithm, is invoked. A variety of selection
procedures are possible. Below one particular selection procedure is described which is encapsulated
in selection object P. Object P involves selecting a functional requirement and then selecting a
design object that satisfies the requirement. 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_{o,}) from \{R_{o,}\}_u

Step 3.2 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 (Sue, 1990) where attribute \(j\) in design
electronics component \(q_{i,j}\) satisfies attribute \(p\) in requirement R_{o,p}

Step 3.3 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.

Step 3.4 Selection object P is invoked which ranks the set of electronics components \{q_{i,z}\}
into an ordered set electronics components \(\langle q_{i,z} \rangle\), according to the evaluation
scheme E, such that the \(q_{i,z}\) with higher ranking comes first. This ranking is based
on the objective function \(V_i^k\), where \(V_i^k = f(S_i^k)\). Select the highest ranked \(q_{i,z}\).
Using add object methods (\(\Phi_{add}\)), add the selected electronics component \(q_{i,z}\) into
the current instance of the design model (\(S_i^k\)). This results in a new design model
(new \(S_i^k\)).
Step 3.5 This sub-step checks that none of the set of constraints \( \{C_i(z)\} \) are violated when the method is applied. This procedure is called validity-preservation. Assign the same orientation to the component as that of similar components in the design model. If no such type has been selected, randomly assign orientation of unmarked orientation of the component to the component and marked this (this is according to design rule No.3 of Table 1). Check that model satisfies the Concurrent Engineering and overall constraints \( \{C_i(z)\} \).

Step 3.6 If the model violates any of the constraints \( \{C_i(z)\} \) and both orientations of the component have been marked, undo Step 3.4 and delete \( q_{i,z} \) from the ordered set of design objects \( <q_{i,z}> \). Then Repeat from Step 3.4 and choose \( q_{i,z} \) with the next rank. Otherwise, Go to 3.5

Step 3.7 Delete the functional requirement \( (R_{o,z}) \) from the set of functional requirements \( \{R_{o,z}\} \). Go to 3.1.

Steps 1-3 above, with associated sub-steps, constitutes the design algorithm A that underlies the EACS 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 \( \{C_i(z)\} \).

5.4.1. Search Strategies

The EACS design process model with selection object \( P_1 \), described above in Step 3.5, involves a search process that combines depth-first and breadth-first methods. Depth-first search is generally appropriate when a tree structure is not too deep, while breadth-first is more suitable when the number of alternatives at each level of the tree structure is not too large.

The choice between depth-first and breadth-first method is complicated by the presence of constraints, since only a portion of the components will be viable in the resulting design. If we consider selection object \( P_1 \) and assume a fraction \( b \) of feasible components at each level are viable in the design solution, then the expected number of iterations of the design process with selection object \( P_1 \) will be equal, in the worst case, to \( X*(Y*b+1) \), while the total number of iterations for complete enumeration is equal to \( X*(Y^X) \). Table 2 shows the effects of different values of \( X \) and \( Y \) for \( b = 0.5 \).
An Object-Oriented Approach to the Concurrent Engineering of Electronics Assemblies

<table>
<thead>
<tr>
<th>Number of Requirements (X)</th>
<th>Feasible Components for Each Requirement (Y)</th>
<th>Expected Number of Iterations Using Selection Object $P_1$</th>
<th>Expected Number of Iterations for Complete Enumeration</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>6</td>
<td>40</td>
<td>604661760</td>
</tr>
<tr>
<td></td>
<td>12</td>
<td>70</td>
<td>6.192E+11</td>
</tr>
<tr>
<td></td>
<td>18</td>
<td>100</td>
<td>3.57E+13</td>
</tr>
<tr>
<td>30</td>
<td>6</td>
<td>120</td>
<td>6.632E+24</td>
</tr>
<tr>
<td></td>
<td>12</td>
<td>210</td>
<td>7.121E+33</td>
</tr>
<tr>
<td></td>
<td>18</td>
<td>300</td>
<td>1.366E+39</td>
</tr>
<tr>
<td>50</td>
<td>6</td>
<td>200</td>
<td>4.041E+40</td>
</tr>
<tr>
<td></td>
<td>12</td>
<td>350</td>
<td>4.55E+55</td>
</tr>
<tr>
<td></td>
<td>18</td>
<td>500</td>
<td>2.901E+64</td>
</tr>
</tbody>
</table>

Table 2: Expected Number of Iterations for Selection Object $P_1$ with $b = 0.5$

The figures in Table 2 show that the use of selection object $P_1$ reduces the expected number of iterations substantial compared with a complete enumeration. However, the use of selection object $P_1$, results in only obtaining a feasible solution, and 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. The considerable drawback to such an complete enumeration approach is that of the excessive computation required. For example, for even relatively small design problems ($X=30$) with fairly loose constraints ($Y = 12$) then the expected number of total enumeration is very large ($7.121E+33$). The repercussion of this is that for such problems, generating good or optimal solutions will lead to a high degree of inefficiency. There is therefore a need to devise algorithms that generate better solutions but without such a large computational burden, and this is considered later in Section 6.4

5.4.2. Evaluation Schema

In the previous EACS design process model, the objective function $V_i^k$ is used to rank components and design models. A weighted attainment objective function certainly can be used, which enables the determination of compromise solutions. This will facilitate the satisfactory fulfillment of different departmental aspects. The objective function can be formulated as follows:

$$V_i^k = f(S_i^k) = a \cdot q_{i,aaa} + b \cdot q_{i,bbb} + \ldots + r \cdot q_{i,rrr}$$

Where $a+b+\ldots+r=1$

$q_{i,rrr}$ is attribute of $q_i$, such as cost, reliability etc.

Due to the different scale units is $q_{i,rrr}$, there is a need to normalize each $q_{i,rrr}$ before it added to the objective function.
6. IMPLEMENTATION OF EACS

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 Section 5.3.1. 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 \( \{R_n\} \) and a set of constraints \( \{C_i\} \) while minimizing (maximizing) an objective function \( V^k \), where \( V^k = f(S^k) \).

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.

6.1. The Architecture of EACS Design System

The overall architecture of the EACS design system, shown in Figure 2, is based on the Object-Oriented Design System (OODS) (Liang and O’Grady, 1997, 1998; O’Grady and Liang, 1997). For the EACS design system each object is created as ActiveX object, which can obtain data over the Internet so that data can be in different locations, and can be updated by the different functional departments. 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). For this example system, the users can access the EACS system through a conventional web browser.

![Figure 3: The Architecture of the EACS Design System](image)

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.

6.2. 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 been completed (Figure 1) so that the broad specification of each component has been determined (Liang and O'Grady, 1998). 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 \( \mu \)F) and semiconductors (including a 555 timer) as shown in Table 3. 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 3, 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 \( V_{l}^{k} = f(S_{l}^{k}) \).

<table>
<thead>
<tr>
<th>Component</th>
<th>Type</th>
<th>Specification</th>
<th>X Position</th>
<th>Y Position</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Resistor</td>
<td>200 (Ω)</td>
<td>1.2</td>
<td>0.5</td>
</tr>
<tr>
<td>2</td>
<td>Resistor</td>
<td>400 (Ω)</td>
<td>2.1</td>
<td>1.6</td>
</tr>
<tr>
<td>3</td>
<td>Capacitor</td>
<td>0.1 (µF)</td>
<td>1.4</td>
<td>2.3</td>
</tr>
<tr>
<td>4</td>
<td>Capacitor</td>
<td>0.01 (µF)</td>
<td>2.4</td>
<td>2.1</td>
</tr>
<tr>
<td>5</td>
<td>Capacitor</td>
<td>0.47 (µF)</td>
<td>3.3</td>
<td>2.6</td>
</tr>
<tr>
<td>6</td>
<td>Semiconductor</td>
<td>Timer (555)</td>
<td>2.4</td>
<td>1.5</td>
</tr>
<tr>
<td>7</td>
<td>Semiconductor</td>
<td>emitting diode</td>
<td>3.1</td>
<td>1.1</td>
</tr>
</tbody>
</table>

Table 3: Electronic Components in Pulser/Flasher

The main objects involved are as follows:

The requirements \( \{R\} \) are of the form:

\[
\{ \text{component } i.\text{type}, \text{component } i.\text{spec.}, \text{component } i.\text{x position}, \text{component } i.\text{y position} \}
\]

These are directly determined from Table 3.

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 inches of the board edge (to allow for the equipment to work properly)

This can be expressed as an if-then rule as follows:

\[
\text{(component. x position-1/2*component. length} \leq 0.2") \quad \text{or}
\]
\[
\text{(component. x position+1/2*component. length} \geq \text{board width - 0.2")} \quad \text{or}
\]
\[
\text{(component. y position-1/2*component. width} \leq 0.2") \quad \text{or}
\]
(component. y position-1/2*component. width >= board length-
0.2") then

\textbf{Violate} Concurrent Engineering Constraint E224

Other concurrent engineering constraints are formulated in a similar manner.

\textbf{The design model}, \( S \), includes the following design objects:

\[ \text{[component } i \text{id}, \text{ component } i \text{orientation}, \ldots, \text{ component } i \text{id}, \text{ component } i \text{orientation}] \]

Where component \( i \text{id} \) is the identification of the component and where component \( i \text{orientation} \) 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.

\textbf{The evaluation schema (E)} is taken to be, for this example, as described in Section 5.4.2.

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 4.

\begin{table}[h]
\centering
\begin{tabular}{|c|c|c|c|c|c|c|c|}
\hline
\textbf{ID} & \textbf{Spec.} & \textbf{Length(\text{")}} & \textbf{Width(\text{")}} & \textbf{Max.} & \textbf{Cost} & \textbf{Vendor} & \textbf{…} & \textbf{…} \\
\hline
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 & … & \\
\hline
\end{tabular}
\caption{Resistors: A Segment of the Aggregated Component Database}
\end{table}

\begin{table}[h]
\centering
\begin{tabular}{|c|c|c|c|c|c|c|c|}
\hline
\textbf{ID} & \textbf{Spec.} & \textbf{Length(\text{")}} & \textbf{Width(\text{")}} & \textbf{Max.} & \textbf{Cost} & \textbf{Vendor} & \textbf{…} & \textbf{…} \\
\hline
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 & … & \\
\hline
\end{tabular}
\caption{Capacitors: A Segment of the Aggregated Component Database}
\end{table}

\begin{table}[h]
\centering
\begin{tabular}{|c|c|c|c|c|c|c|c|}
\hline
\textbf{ID} & \textbf{Spec.} & \textbf{Length(\text{")}} & \textbf{Width(\text{")}} & \textbf{Max.} & \textbf{Cost} & \textbf{Vendor} & \textbf{…} & \textbf{…} \\
\hline
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 & … & \\
\hline
\end{tabular}
\caption{Semiconductors: A Segment of the Aggregated Component Database}
\end{table}

\textbf{6.3. 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 degrees while the minimum operating temperature is 20 degrees. The user can enter this data on the EACS user interface, which uses a standard web browser (Figure 4). The link between the user's input screen and the overall architecture of EACS is shown in Figure 3.
Figure 4: The Input Screen of the EACS System

Besides the parameters shown in the Figure 4, the design requirements, obtained from Table 3, can be placed into a database, which has been pre-connected to the EACS system using the CS object. The board size is taken to be 3” by 4”. In practice such design requirements are usually generated by ECAD software.

Given the above scenario, example results using selection object $P_1$ (step 3.4 above) are shown in Figure 5, with, for this example, a cost of $2.34 and a reliability of 0.952.

Figure 5: Example Results Obtained by EACS Using Selection Object $P_1$
6.4. The Use of an Evolution Search Selection Object $P_2$

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.

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.

Selection object $P_2$ consists of a constrained genetic algorithm search mechanism (Liang and O'Grady, 1997; O'Grady and Liang, 1997). 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 section 4.7. To avoid confusion with $P_1$, $P_2$ sub-steps will be indicated by ($P_2$) attached to each sub-step.

**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 that stated above)

a) If $\{R_{o,}\}_{u} = \text{Null}$, then set $\{R_{o,}\}_{u} = \{R_{o,}\}$ and $\{R_{o,}\}_s = \text{Null}$. 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 (\( \Phi_{\text{add}} \)), add the selected electronics component into the current instance of the design model \( (S_l^k) \). This results in a new design model \( (\text{New } S_l^k) \).

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 \( \{R_o,..\} \). 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 \( V_l^k \), where \( V_l^k = f (S_l^k) \).

c) Use an appropriate selection strategy to select the new population.

**Step 3.3(P_2) Crossover**

Apply the constrained crossover method (O'Grady and Liang, 1998) 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 (O'Grady and Liang, 1998) to the design models that are selected for mutation. In this proposed constrained mutation method, a design model \( S_l^k \) 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 \( S_l^k \) 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 6, indicate that a family of design models \( (S_l^k) \) are obtained. As the number of members in a population increase or as the number of generations increase, then the population tends to converge. For this example, with 40 in the population and with 30 generations, the result is a convergence to a single population member with a cost of $1.70 with a reliability of 0.956. This is an improvement over the results 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 \( P_2 \). It means that the \( P_1 \)
can be replaced by the $P_2$. In this case, we use genetic algorithm search algorithm as $P_2$ but other search algorithm such as branch and bound can also be introduced into the system.

6.5. Results from the Example Implementation of EACS

The example implementation of EACS described above is useful in illustrating some of the main points about the approach used. These include the issues surrounding computability, and reusability & exchangeability as follows:

**Computability:** 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 ($S_i^k$) in the design process. The ready translation from Figure 2: The Overall DwO Structure to Figure 3 illustrates this feature of the approach used.

**Reusability & Exchangeability:** Because the design objects in this approach are modular, they can be replaced by another design object, provided the interfaces of the two objects are similar. This would allow, for example, for the relatively straightforward upgrading of portions of a computerized design system. As shown in this implementation, the selection objects $P_1$ and $P_2$ have been exchanged in a straightforward manner.

7. SUMMARY AND CONCLUSIONS

The work presented in this paper is focused on Electronics Assemblies Components Selection (EACS) where participants may be geographically separated, and how to implement EACS in a formal and systematic way. A representation formalism is proposed for that purpose. This paper first overviews the EACS process and identifies the knowledge gaps in existing EACS support. A new formalism for EACS is then presented. This formalism uses an object-oriented approach, called Design with Objects.
(DwO), where electronics components are represented as objects. An Internet network based implementation of this formalism is described that uses the Internet to support EACS. Perhaps the main contribution of this work is that it describes a formalism that can be used for EACS. So that it could be used where participants are geographically scattered. The implementation described uses the Internet as a vehicle but the formalism should also be applicable to any other computer network.

8. REFERENCES


Booch, G., 1994, Object Oriented Design with Applications, Benjamin/Cummings.


Coad, P. and Yourdon, E., 1991, Object-Oriented Design, Yourdon Press, NJ.


GenRad, 1985, Introduction to In-Circuit Testing, GenRad


Gloster, C.S., and Brglez, F., 1989, "Boundary Scan with Built-In Self-Test", IEEE Design & Test of Computers, February, pp. 266-274


Solberg, V., 1989, "Go Team, Go!,” *Circuits Manufacturing*, February, pp. 53-55


