

A Knowledge-Based Object Oriented DFM Advisor for Surface Mount PCB Assembly

S. Krishnan and K. Srihari

Department of Systems Science and Industrial Engineering, State University of New York, Binghamton, NY, USA.

The volume and complexity of printed circuit boards (PCBs) that use surface mount technology has increased tremendously over the past decade. The intricacy of the design and manufacturing functions associated with this domain is continuously increasing. Several researchers have documented the significance of the design functions on product life cycle costs. In the surface mount PCB assembly area, the integration of the design and manufacturing activity is hampered by the lack of sufficient, in-depth manufacturing knowledge among the PCB designers. Also, the heuristic nature of manufacturing knowledge coupled with the frequent advances in the state-of-the-art has resulted in the lack of widespread in-depth process knowledge.

This research focused on the design and development of a knowledge-based methodology which can be used to assist the design engineer in the surface mount PCB assembly domain. The rules relate primarily to the solderability, cleanability and reflow aspects of the design. An object oriented programming (OOP) methodology is used. The OOP framework allows for structured and modular development and implementation of the knowledge-based design advisor. It allows for the easy maintenance and update of the software along with the integration of knowledge bases and databases. The system has been partitioned into three major components – the user interface, the update mechanism along with the inference engine, and the database management utility. An important feature of this research has been the focus on updateability of the design advisor with respect to data and knowledge (rules). The logic or the rules within the system have been developed in an updateable format. The calculations required during the firing of the rules are performed by dynamic linked libraries (dlls). The design advisor has been developed in the Windows operating system using the C++ language. The database manipulation functions are handled by CodeBase 5.0 libraries.

Keywords: Concurrent engineering; Design for manufacturing; Object oriented programming; Printed circuit boards

1. Introduction to Research

The need for design engineers to analyse their designs from a manufacturing perspective has been well recognised. Though the most effective method of attaining a design for manufacturability (DFM) perspective may be the formation of design teams which include experts from various facets of manufacturing and design, more often, such an endeavor fails owing to complex logistical problems within an organisation. In most large manufacturing organisations, the design and manufacturing facilities are located so far apart (sometimes in different cities) that it becomes virtually impossible to put together a design team. Such a scenario mandates the need for electronic communication and decision support systems which will provide the designers with a manufacturing perspective.

The lack of computer-based decision support that could assist a design engineer with manufacturing knowledge often causes the absence of a DFM review of a design [1]. This is especially true in the printed circuit board (PCB) design and assembly domain. A tremendous amount of segregation of design and manufacturing knowledge has occurred leading to a communication problem [2]. The adverse effect of the absence of the DFM perspective becomes obvious when it is realised that a majority of the manufacturing decisions are directly determined by the product's design [1]. Compounding this problem is the fact that the complexity and density of new PCB designs are on the increase [3] and there has been a lack of widespread expertise in the relevant manufacturing processes. The net result has been reduced productivity, and increased product cost.

The development of support systems, however, requires a great deal of caution. Frequently support systems fail owing to an improper integration of the associated databases [2]. This is especially true if several independent knowledge-based systems are initially developed and integration is attempted at a later stage. It is therefore important to follow a consistent method of representation of data and knowledge. This paper describes the design and development of an integrated expert system based DFM environment for the surface mount PCB assembly domain. This object oriented programming (OOP) based system is updateable and easily expandable.

Correspondence and offprint requests to: S. Krishnan, Department of Systems Science and Industrial Engineering, State University of New York, Binghamton, NY 13902, USA.

2. Related Concepts

The concepts used in this research include:

Electronics packaging.

PCB assembly using surface mount technology (SMT).

PCB design.

Design for manufacturing.

Artificial intelligence and expert systems:

Object oriented programming.

Blackboards.

Knowledge representation.

Heuristic search mechanisms.

2.1 Electronics Packaging

Electronics packaging or the packaging of electronic circuits is the science and art of establishing interconnections and suitable operating environments for electrical circuits to process and store information [4]. During the 1980s, PCB assembly using SMT was the fastest growing technology in small systems and for consumer electronics, accounting for nearly 20% of all consumer and low end packages. The use of SMT in PCB assembly is expected to reach 55% by the early 1990s and to be over 70% by the late 1990s [3]. Surface mount components are designed to be soldered to the surface lands (or pads) on the PCBs. The advantages of SMT use over conventional through-hole technology include size, interconnectivity, cost, performance, and quality [3, 5]. The typical process sequence for the assembly of single-sided surface mount PCBs consists of the pre-assembly operations on components for the purposes of trimming and forming leads and other modifications to vendor supplied components, attachment or mounting of components onto the board, soldering, cleaning, and testing.

Soldering is the joining of metals by causing a lower-melting point metal to wet or alloy with the joint surfaces and then to freeze them in place. Solders are used to establish reliable electrical connections and to physically hold the parts together. The most commonly used solders are alloys of tin and lead that melt at about 183°C. To achieve good wetting at the time of soldering, the surface to be joined as well as the solder must be free from oxide or other impurities. Fluxes are usually used to assist in the removal of the oxides. Two commonly used technologies for soldering components are plated through hole (PTH) technology and surface mount technology (SMT). However SMT is fast replacing PTH technology owing to [3, 5]:

Increased density.

Reduced weight.

Reduced space for raw materials and finished products.

Fewer assembly machine types required.

Reduced work-in-process (WIP).

Better performance characteristics as the interconnection paths between active integrated circuit (IC) chips are shorter.

Increased system reliability as the number of PCBs and connectors decrease.

Reduced electrical noise.

Smaller boards with fewer drilled holes cost less.

Depending on the method by which surface mount components (SMCs) combine with the PTH components in an assembly, the assembly technique is determined. PCBs are typically classified as Type I, Type II, or Type III boards [6]. A Type I board consists exclusively of SMCs on one or both sides of the PCB while a Type II board is a mixed technology board with PTH components on the top side of the PCB while SMCs could be placed on one or both sides of the PCB. A Type III board is a subset of Type II. In a Type III board, the top side of the PCB is populated with PTH components, and only passive SMCs such as capacitors, resistors, and small outline devices that are adhesively attached and wave soldered populate the underside of the PCB assembly.

2.2 PCB Design

PCB design is a complex process involving substantial insight and skill coupled with knowledge of electronic circuit design. It is a linear rigidly structured process that includes the following steps [1]:

Developing the circuit design.

Creating a schematic diagram using a computer.

Verifying the circuit design using simulation software.

Defining the board size and shape.

Placing components on a PCB using a combination of automatic and manual techniques.

Using autorouters to connect the appropriate pins of the packages.

Checking the board for wiring errors.

Sending the PCB design drawing for fabrication.

Sending the board along with the required components to be assembled.

From an end product application (or usage) perspective, the success of a PCB's design relies substantially upon the environmental conditions it operates under such as the ambient temperature, heat generated by the components and the ventilation provided for heat dissipation [8]. A PCB's circuit design is also assessed by considering two factors, namely reliability and producibility. It should be designed to meet the end item requirements of life-cycle, environment and function. Producibility is a difficult factor to assess. It needs to be considered along with product reliability needs and acceptable product cost. A PCB's design could further be affected by applicable manufacturing processes and constraints.

The design of a PCB assembly may be broadly partitioned into two divisions: mechanical configuration and electrical configuration. Typically the first mechanical constraint considered is the space available for the circuit which determines the number of layers and component densities of the board. The second constraint is the operation of the circuit. This

may require the use of heat sinks, etc. to manage elevated temperatures or other operational effects. The final constraint is the operating (usage) environment constraint like temperature, pressure, shock, vibration, etc. Very rarely is any consideration given to the economic manufacture of the assembly within the limitations of the manufacturing equipment.

The package dimensions, heat transfer capability, pinout, complexity and testability all limit the number of packages that can be put on the PCB. Components are usually selected for their electrical, thermal and mechanical characteristics. The material composition, finish, and geometry of the package body and the leads are critical factors that impact the PCB assembly method. Component selection depends upon electrical characteristics and performance, environmental requirements, manufacturing methods, thermal requirements, reliability, placement and other manufacturing equipment constraints, cleaning and corrosion, etc.

2.3 Concurrent Engineering and Design for Manufacture

A design helps to identify the processes required to convert raw materials into a finished product. This process requires the use of technical and managerial skills [9]. While it is possible to visualize the design process in traditional terms as several separate tasks, in reality it is a single process made up of many related tasks such as product definition, idea generation, etc. The traditional design process is a serial process in which an initial design is analysed by various modules [9]. The input to this process is a set of marketing needs and an approximation of the market value. The effective output of the traditional design process is an estimated production cost for the initial design. If the estimated production cost is greater than acceptable, then there is a need to repeat the process by modifying the design at one or more of the intermediate stages. The process of re-design causes substantial delay in the design-to-produce phase as almost invariably the domain experts of each phase are physically located in different departments and communication among them is a cumbersome process.

Concurrent engineering (CE) has as its goal the development of a detailed design while simultaneously developing the production capability and quality to manufacture the design [9]. It consists of a methodology using a multi-disciplined approach to achieve concurrency. CE tools exist in the form of algorithms, techniques and software other than the expertise and judgement of the relevant domain experts. DFM is specifically the integration of the product design and process planning within the realm of concurrent engineering. The goal is to design a product that is easily manufacturable. Consideration of manufacturability guidelines is an essential part of DFM. DFM guidelines are usually statements of good design practice that have evolved from many years of design and manufacturing experience.

A key to successful DFM practice is that manufacturers should make designers aware of process related constraints during the design process before the design is irreversibly put down on paper. This includes guidelines or rules of thumb

specific to a particular industry. The designer should consult these rules during the design process before it is too late or expensive to incorporate DFM considerations. Design rules and other techniques can be implemented on a computer as a toolkit for design engineers. These, however, are manufacturing method specific because a great deal of knowledge is required to advise a designer on just one facet of manufacturing.

2.4 Artificial Intelligence and Expert Systems

In recent years, the number of applications of expert systems has been on the rise [10]. An expert system is a computer based tool which has the capability to understand and use domain knowledge intelligently to suggest alternate paths of action [11]. The major components of an expert system are described below [12].

The user interface is a system dependent feature that insulates the user from the architecture or working of the software programs. The common forms of interfaces utilised are question and answer sessions, menus, or a natural language interface. The user interface must strive to make the entire system user friendly. The user of an expert system may be a "tester" where the user verifies the validity of the system's behaviour, a tutor where the user adds or modifies the knowledge in the system, and a pupil where the user gains personal expertise and knowledge from the system. A user of the system who is a customer uses the system for a real task.

The heart of an expert system is the knowledge base which usually contains the body of facts and general information related to the domain in a symbolic form. Heuristics, intuition and experiential knowledge about the problem area are also included in the knowledge base. There are several techniques employed in the representation of knowledge among which frame-based systems, object-based systems, and rule-based systems are the most frequently used. The principle of inheritance is an invaluable tool available in an object-based system. It allows the knowledge to be built in a hierarchical fashion and owing to its inherent modularity provides for greater efficiency and better performance.

The inference engine performs three basic functions [13]. They are the identification of applicable rules, conflict resolution, and rule execution. These three are continuously applied in a recognise-act control cycle. The available rules are used until there are no more rules to execute [11]. The procedures that implement the control cycle are independent of the production rules. The inference engine consists of a search strategy that is used to locate knowledge and infer new knowledge from the knowledge base. The advantages of separating the knowledge base from the inference engine in an expert system include:

1. Expert system builders can focus solely on the building of the knowledge base.
2. Changes can be made to one part of the knowledge base without substantially affecting other parts.
3. The same inference engine may be used with different knowledge bases.

4. Different inference mechanisms can be tried with a particular rule base.
5. An efficient update mechanism can be effected only by this separation.

An explanation subsystem allows the programmer to explain the reasoning or logic of the program to the user. These explanations are typically justifications for a conclusion, explanations of why a particular piece of data is required, and some deeper theoretical justifications of the program's actions. The knowledge update facility or the ability to modify/add new knowledge is absolutely essential for domains that are rapidly expanding or changing. The most primitive method of update would be the manual one wherein the knowledge engineer updates the files that make up the knowledge base. Alternatively, the update may be performed by the domain expert without any help from the knowledge engineer. This is usually done through a series of questions and answers. It is more sophisticated than the manual procedure. The most sophisticated knowledge update procedure would be learning where new knowledge would be generated from past experiences automatically. The system in this case learns by itself and is self-updating.

A blackboard architecture is an expert system framework in which several independent knowledge bases of segmented knowledge, called knowledge sources, communicate via a multi-dimensional global data structure called a blackboard [12]. An agenda-based control scheme is used to continually examine all the possible actions and then make the selection. A scheduler is often used to perform the agenda-based control. Each knowledge source is an expert system for solving a particular part of a multi-disciplined problem.

In an OOP system, everything is defined as an object or a system of interacting objects [11]. Individual objects are implemented as frame-like structures whose slots contain either variables or procedural information. Relationships between objects and the overall problem specification can be implemented as messages between objects. Objects are organised into a hierarchy of classes allowing the inheritance of properties and procedures. An OOP system allows the definition of computational objects or arbitrarily complex internal structure which may be thought of subsequently as a complex entity. These computational objects encapsulate both state and behaviour [11]. Their behaviours are implemented as attached procedures, called methods. These are invoked through messages sent to the object by the user or by other objects. On receiving a message, an object executes the designated method and responds accordingly. Objects have characteristics of both data and programs since they retain the state variables and react procedurally in response to appropriate messages. The properties of OOP make it a powerful and natural way to organise large and complex software implementations. The message passing aspect of OOP systems simplifies the representation of interacting components.

A critical aspect of a knowledge-based system is the knowledge base and its representation. Associated with this is the adoption of a suitable search strategy for the problem state space. Knowledge representation is usually classified as [11]:

Logical representation schemes wherein formal logic is used to represent the knowledge base.

Procedural schemes wherein the knowledge is represented as a set of instructions for problem solving.

Network schemes where the nodes are concepts of the domain and the arcs the relationships between them.

Structured schemes which are extensions of the network scheme and allow each node to be a complex data and knowledge structure.

3. Problem Statement and Research Objective

The use of SMT in PCB assembly has increased tremendously over the past decade. The complexity of the design and manufacturing functions associated with this domain is continuously increasing. Several researchers have documented the significance of the design functions on product life cycle costs. In the SMT domain, the integration of the design and manufacturing activity is hampered by the lack of sufficient, in-depth manufacturing knowledge among the PCB designers [1]. Also, the heuristic nature of manufacturing knowledge has resulted in the lack of widespread in-depth process knowledge.

The problem's significance becomes apparent when the two primary factors that combine to create economic problems for producers of PCBs are considered. Forecasts for component development show that within the last decade the use of SMCs has risen substantially. This trend is expected to continue in the 1990s. The growth in the use of SMCs has generated the need for the design engineer to design for their use in PCB assemblies. At the same time there has been a second force at work. The design engineer is being asked to take a greater responsibility for the "design for manufacturing". Consequently a growing technology and an expanded responsibility have created a need for support systems [1].

Though computers and automation have been used extensively in the surface mount and other forms of PCB assembly for quite a while, the integration of stand alone automated systems has not been done effectively. Schemes for the integration of the production and design process have been proposed [7]. The rate of success has been limited. Consequently there is a need to research, design, and develop an integrated knowledge-based environment to support and link the design and manufacturing activities for PCB assemblies using SMT (similar to Fig. 1).

While Fig. 1 identifies the global structure of a DFM system for the surface mount PCB assembly domain, the current scope of this research is limited to the development of the solderability, reflow and cleanliness advisors. Once the PCB's design is generated, the input from the computer aided design (CAD) system is provided to the system (the design advisor). The three modules access the CAD data as well as the user input information to perform the "design checking" operation. The modules communicate through one another through the blackboard whenever the need arises. A design cost evaluation module is currently being integrated within the design advisor.

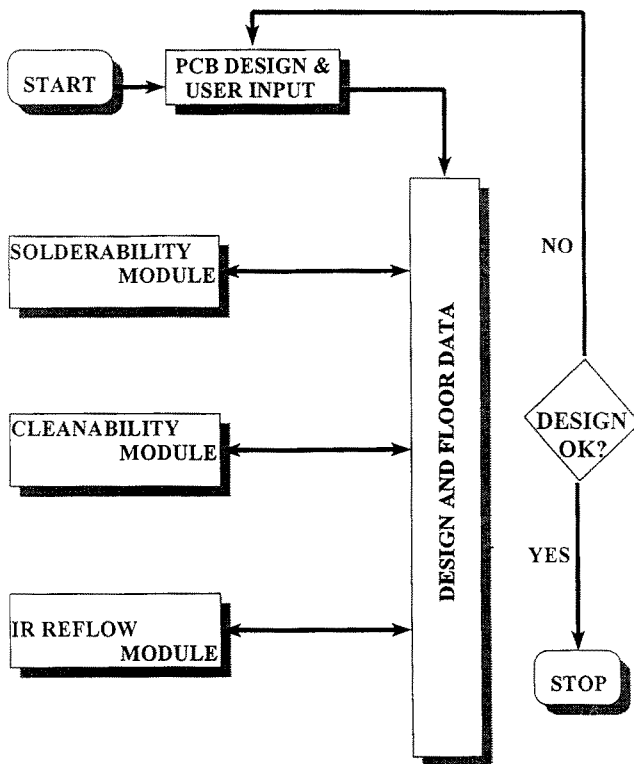


Fig. 1. System architecture.

If a specific design is found to be acceptable from the manufacturing perspectives (Fig. 1), the design could then be released for actual manufacture.

The principal function of the design advisor is to provide the PCB assembly designer (the intended user) with a feel for the manufacturing related problems. It will help in reducing the re-design of the PCB after the design is sent to the manufacturing floor. The prototype system designed and developed through this research provides the designer with "advice". It suggests possible refinements needed in the design from a manufacturing perspective. If a need to improve the PCB is identified and the improvements are done, the design advisor is then used to evaluate the redesign. If, on the other hand, the design is accepted, the designer would transmit the design to the manufacturing floor for production. An important feature of this research is the method in which the knowledge is stored in the knowledge base. All the rules have been separated from the source code of the program and are stored in databases. The specification of the rule structure and associated data structures is as specified in [14] and will be discussed later.

4. Research Methodology

Typically the design sequence in a manufacturing organisation is as follows:

1. Product requirement definition.
2. Preliminary logic design/design of schematic.

3. Selection of packages and substrate.
4. Floor planning of the packages/package layout.
5. Detailed logic design.
6. Module layout/printed wiring board design.
7. Module design.
8. Build board on the manufacturing floor.

Usually, a DFM environment performs the design checking operation between the stages (3) and (4), i.e. just after the packages are selected and are about to be placed. In fact, it mainly checks whether the placement of the components on the board is acceptable from a manufacturing point of view. It is important to note that any given functional objective of the circuit can be achieved through different combinations of packages and substrates. The designer is, however, faced with a number of alternatives during the selection of packages and most often makes the selection without any regard to manufacturability constraints. The design advisor developed in this research provides this critical manufacturability perspective to the designer during the early design phase (during and between stages 2 and 3 mentioned above). Fig. 2 provides a flowchart of the operation of the system. It is assumed that the intended user of this design advisor is a design engineer with a limited knowledge of the intricacies of manufacturing and populating a PCB.

The review of a design from a manufacturing process perspective can broadly be classified into two categories: manufacturing problems due to thermal considerations and manufacturing problems due to location considerations [5]. With respect to thermal considerations, the PCB design needs

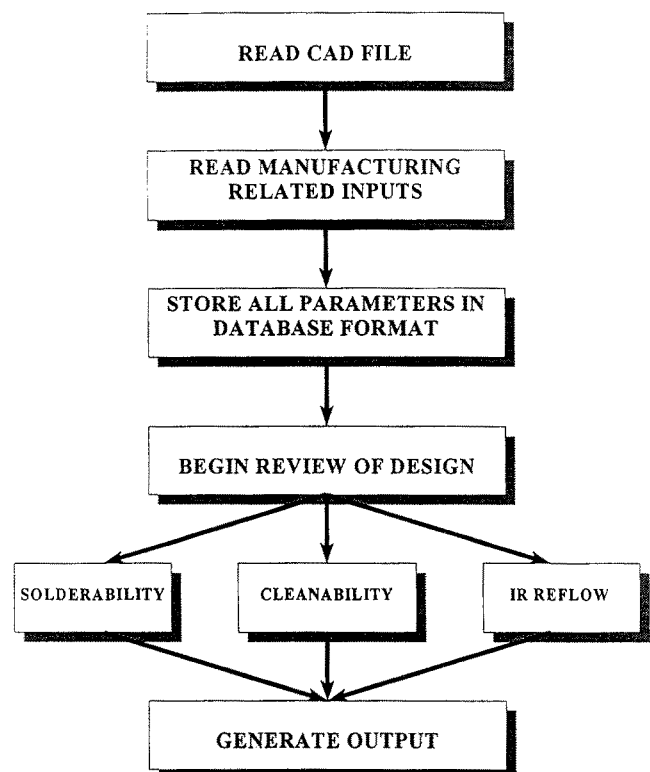


Fig. 2. Sequence of operations within the system.

to be checked from a solderability as well as reflow perspective. As far as the spacing and location considerations are concerned, it becomes important to ensure that the components located in the design are cleanable and placeable within the limitations of the relevant equipment. In this research the manufacturability review of a PCB's design is performed under three main categories – solderability, reflow and cleanability. Under each of the three main categories three subcategories are available which analyse the design with respect to potential manufacturing problems.

4.1 Solderability Issues

The solderability of a component to a PCB is dependent on several process parameters as well as some design factors. Good solderability is attained when the surfaces to be soldered are free from oxides and good wettability is ensured. Therefore, the material of the components and the board material are important factors that affect process yields. Another factor that affects the wettability of a surface with solder is the geometry of the surface itself. Consequently, it becomes important to know the interactions between the solder paste and the lead geometries of the component. The initial PCB design is reviewed with respect to three major issues that affect solderability, namely lead material and geometry, and land material.

4.2 Component Spacing or Location

Several issues need to be considered when laying out and spacing components on a surface mount PCB including soldering yield, placement, cleaning, inspection, repair and testability. These points are often overlooked when designers select SMT mainly for its real estate advantages. The location of SMCs on a PCB determines the IR reflow profile and affects process yields. If components that absorb heat (owing to colour and/or mass) are concentrated on one segment of the PCB, the PCB is non-uniformly heated. This can result in lower process yields, increased times to develop IR oven thermal profile, and reduced long term PCB reliability. The most difficult parts to clean, repair and rework are the “J” leaded components and the small outline integrated circuits (SOICs) [15]. It must be ensured that adequate clearances between components are provided to facilitate clear inspection. Another important consideration is the orientation of the components as it may influence placement accuracy.

As far as selection of packaging styles is concerned, the following guidelines can be used [15]:

For passive components, specify the 1206 style package as they are widely used and are the recognised standard.

Avoid the use of MELF type packages which are commonly used as diodes. This is because the cylindrical barrels are difficult to work with and require special tooling on the placement machines.

Specify nickel and or tin/lead barriers on the chip terminations. Do not use silver unless the solder itself contains silver.

Ensure that leaded devices have solder coated leads.

Choose gull wing parts of “J” leaded components as they effectively absorb the mismatch in the coefficient of thermal expansion. However, gull wing parts tend to create coplanarity problems and may not be suitable if packaging density is the objective as they occupy more real estate.

4.3 System Components and Interrelationships

The design advisor environment consists primarily of three components: a data repository (or several databases), an expert system rule base stored in a database format along with an inference engine, and user interfaces. All the three systems rely on object-oriented representation. The databases contain the relevant data on the components and the solder pastes. Input information related to the processing equipment and capabilities is required from the user which is then stored in a file (floor.dbf) for the rules to operate upon. The design advisor environment consists of knowledge bases which are a representation of the design guidelines in the form of rules. The knowledge itself is stored in databases. The user interface allows the user to browse and modify the databases as well as to invoke the expert system and view its results.

Prior to a detailed discussion of the modules, the technique used to develop the system is addressed. While an expert system based approach is used, the feature that distinguishes this system from other expert systems is the use of OOP techniques. The OOP approach reduces the time required for the development of the knowledge-based system since initially developed objects are re-used and modified rather than each module being developed individually. Also, given the nature of the problem and the domain to be dealt with, it is inevitable that changes will have to be made repeatedly and continually within the system. This is due to the rapidly evolving nature of the field and the need for knowledge update. The use of an OOP approach provides for easier knowledge base update without negatively impacting the performance of related software modules. Similar research done in the area of computer-aided specification of integrated circuits [16] indicates the success of implementation of such a scheme. The OOP strategy with its integration of the data and related procedures into a single module called objects facilitates a distribution of “intelligence” within the framework of the expert system [17].

Certain objects in the system are practically passive (e.g. the component database) in the sense that they merely hold information or data but do not act on them. On the other hand, active objects (for example the solderability module), act on the other objects (like the component database) and decide intelligently on future courses of action. Therefore, the use of an OOP structure in expert system development allows the user to change the structure of any object at any stage of software development as long as the response of the object to messages is not altered. This approach, in contrast to conventional expert systems, allows for the knowledge bases, databases and update facilities to be integrated into objects at different levels.

4.4 Component Database

A critical component of the design advisor is the component database. It contains information on components that could be placed on the PCBs. Information stored in the form of objects includes the component number, type, number of leads, maximum and minimum dimensions of the components' length, width and height from the substrate, pitch, termination/lead width and length, lead metallisations, placement location method preferred, the method of presentation (bins, trays, tape, etc.), and the maximum temperature of exposure allowed while soldering along with the time of exposure. The parameters stored in the component database primarily define the manufacturing and physical characteristics of the components. The termination material and dimensions (along with the fillet angles and radii) determine the solderability of the component leads. The temperature of exposure and the time of exposure permitted for the components determine the suitability of the components to withstand the reflow profile. The dimensions of the components along with their thermal mass helps determine if the design will absorb heat uniformly during the solder reflow process. The power dissipation factor assists in determining the PCB's temperature profile during its actual use (post-manufacturing performance). The cost of the components is used in determining the material cost of the design.

The component database helps reduce the descriptive input requested and concomitantly reduces human involvement in system operation. All databases are controlled through a centralised database management utility which allows the addition, deletion, modification of data records and fields in any database structured in the dBase IV format. From the database management utility, any file in dBase IV format can be created and/or viewed. The record number and the field number and name of any data item is conveniently displayed to perform any of the operations mentioned above. The component database contains information on both active and passive components. The passive devices considered include monolithic capacitors, tantalum capacitors, resistors, and resistor networks. Information on active devices such as transistors and integrated circuits (SOICs, plastic leaded chip carriers (PLCCs), and quad flat packs (QFPs)) is stored in the component database. Both gull wing and J-leaded devices are included in the component database. Components with lead pitches of 50 mils and 25 mils have been considered. TAB components with pitches as low as 10 mils are also included within the system.

4.5 Expert System Knowledge Sources

The knowledge sources created in this research provide for expert review of the PCB design. They use the databases constructed as part of the DFM environment to access relevant information about components and pastes and use the design and the floor file to access information regarding the initial conditions of the system. The design advisor consists of three modules namely the solderability module, reflow module and the cleanliness module. This modularisation allows the system to be easily expanded and maintained. Each of the modules

(mentioned below) provides the design with "advice". This "advice" along with a design cost evaluation advisor (designed and developed in associated research by Alexander [18] assist in deciding if a refinement effort is needed. If a need to improve the PCB is identified, the design advisor is then used to evaluate the redesign. If, on the other hand, the design is accepted, the designer would transmit the design to the manufacturing floor for production.

4.5.1 Solderability Module

The solderability module reviews a PCB's design based on the following three criteria:

Lead Material. The design advisor system reads in information on all the components on the PCB from the design file. It then accesses the component database to determine the lead material of the component. Next, it determines if the component is easily solderable taking into account the solder paste being used in the shopfloor. Generally solder plated components are preferred over solder coated components as the plated components show greater uniformity in the coating thickness. However, it is important to ensure that plated components are not exposed to the reflow cycle before they are soldered as most of the tin is consumed in the formation of the intermetallic layer. This module then makes suggestions on the basis of the compatibility between the paste and the lead material.

Lead Geometry. Here three major classes of components are checked with respect to the solder paste being used. The solderability of the QFPs and the small outline J-leaded (SOJ) components depends on the ceramic plate test (CPT) values of the solder paste being used. These values are obtained from the solder paste database. For other leaded components especially the PLCCs, the solderability is determined on the basis of a combination of the wetting balance test values of the solder paste (found in the solder paste database) and the fillet angle and fillet radius values of the component under consideration. The fillet angle and the fillet radius determine if the contact angle between the paste and the leads is acceptable or not. A contact angle of 90° or less is considered acceptable to ensure proper wetting.

Component Mix. This category checks if the thermal mass moment of the components on the board is within acceptable limits. This is done to limit the uneven heating of the board. The calculation of the moment is done by using the values of the location and thermal mass of the components on the board. The performance of the calculation of the moment is done through the use of a function called "MOMENT()" in the dynamic linked library called "math.dll". A detailed description of a dynamic linked library can be found in the next section (Fact). The "MOMENT()" function accesses the pcb.dbf file which contains the list of components present in the design. It then accesses the thermal mass values of the components on the PCB and calculates the mass-moment about the x- and y-axes by the following equations:

$$\text{mom_x} = \sum (\text{thermal_mass}) * ((\text{x_location}) - (\text{x_mid_pt}))$$

$$\text{mom_y} = \sum (\text{thermal_mass}) * ((\text{y_location}) - (\text{x_mid_pt}))$$

where

mom_x is the mass moment about the x-axis;

mom_y is the mass moment about the y-axis;

thermal_mass is the thermal mass values of the components on the design;

x_location is the x-coordinate of the component on the design;

y_location is the y-coordinate of the component on the design;

x_mid_pt is the x-coordinate of the mid point of the PCB;

y_mid_pt is the y-coordinate of the mid point of the PCB.

4.5.2 IR Reflow Module

This module checks the PCB's design using the following three criteria:

Coefficient of Thermal Expansion (CTE) Mismatch. Here the suitability of the component to be reflow soldered is determined by the difference in the CTE between the substrate material of the PCB, the component lead and the solder paste. If it is too large, the stresses induced in the component lead make it unsolderable. The calculation involved here is performed by the "CTE(D,D,D)" function in the "math.dll" library. The three Ds in the parentheses following the function name imply that the function accepts 3 parameters all of which are of type "double" (or floating point type). The function accepts the board_CTE, comp_CTE, and the solder_CTE as its parameters. It then determines the difference in the maximum and minimum values of the CTE passed to it and returns the value to the inference engine.

Component Density. The total number of interconnections per square inch determines the solderability and the heat absorbing/dissipating capability of the PCB. This factor is measured here and on that basis the ability of the design to be IR reflow soldered is determined. The calculation is performed by the function called "DENSITY()" located in the "math.dll" library. This function accesses the pcb.dbf (design data file), to get the names of the components on the design file and then accesses the "lead_no" field in the component database. It then sums up the lead count of the components in the design file, divides it by the board area and returns a density value to the inference engine.

Lead-Pad Dimension. Here the pad to lead dimensional compatibility of the components on the board is checked. This function is performed by the "PAD_LEAD(S)" function in the "math.dll" library. This function accepts the comp_name as a parameter (the S stands for variable type string), checks the lead to pad coverage and determines if it is acceptable or not. It then returns either a "PASS" or "FAIL" string to the inference engine.

4.5.3 Cleanability Module

Even before this module is accessed, the system checks the final application of the design. If the application is determined to be commercial, then the entire module is bypassed as commercial application boards (typically) are not cleaned. If, however, the final application of the board is determined to be industrial or commercial, the module checks the cleanability of the board based on the following parameters.

Component Height. The component clearance from the board surface determines the cleanability of the board. On the basis of the type of the cleaning equipment used, the cleanability of the board under the components is determined. Surface mount PCB assemblies populated with components such as QFPs, SOICs, and PLCCs present a geometry that can be very resistive to the cleaning process. The difficulty of removing flux residues increases dramatically as the component off-contact distance decreases. Though passive devices such as resistors and capacitors have practically no off-contact distance, it is relatively easy to clean them as they have smaller surface areas.

Package-Solvent. The chemical compatibility of the plastic components and the chlorinated solvents used in cleaning is determined. This is another factor that is used in determining the cleanability of the board.

Solvent-Flux. The ability of the solvent and the cleaning process to clean the residues of a given type of flux without damaging the components on the board is an important factor in determining the cleanability of the board. For instance, if the cleaning process is kept constant, i.e. the same solvent machine and process are used for cleaning assemblies with active and mildly active fluxes, it will be almost impossible for PCB assemblies soldered with active fluxes to reach the same or higher levels of cleanliness than other assemblies soldered with milder fluxes.

Fig. 3 shows the interaction of the different components within the design advisor. As described above, the rules require information on the type of components and the pastes used in the manufacture of the PCB, equipment used in the shopfloor, the type of flux, the type of solvent used, etc. In order to facilitate this, the user is required to select from a list of alternatives of pastes, solvents, equipments, etc. to determine the process conditions on the shopfloor. Given below is a list of the substrate materials, flux, solvent and cleaning methods considered by the design advisor.

The flux types considered are: organic acid flux; rosin; rosin mildly activated (RMA); rosin activated (RA); synthetic activated (SA). The solvents considered can be classified into the following categories:

Chlorofluorocarbons:

Carbontetrachloride (CCl₄); Trichloroethylene (TCE); Perchloroethylene (PCE); 1,1,1-Trichloroethane; Dichloromethane (DCM); Trichlorofluoromethane (R-11); Dichlorodifluoromethane (R-112); Trichlorotrifluoroethane (R-113).

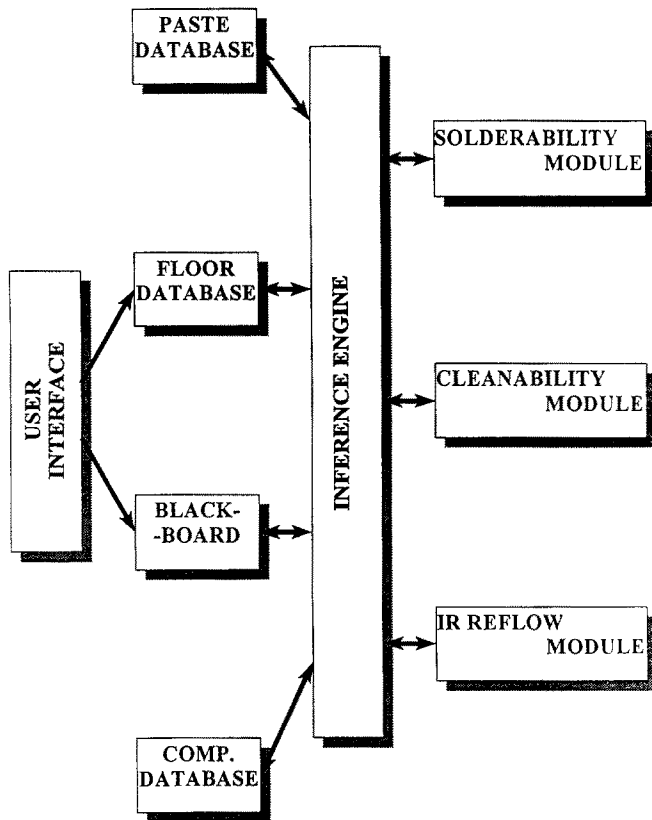


Fig. 3. Interaction between system components.

Alcohols:

Methyl alcohol; ethyl alcohol; isopropyl alcohol.

In addition to the above two categories, acetone and aqueous solvents are also considered within the system.

The cleaning methods considered are vapour solvent cleaning, liquid solvent cleaning, spray solvent cleaning, water cleaning and ultrasonic cleaning.

Sixteen solder pastes with the following alloy compositions are considered in this research. They are: Sn63Pb37 which is a tin-lead eutectic alloy; Sn62Pb36Ag2 which is a tin-lead-silver, ternary eutectic, and Sn63Pb36.65Sb0.35 which is a tin-lead-antimony alloy. The difference between solder pastes of similar alloy compositions is mainly in the difference in the metal weight percentage in the pastes. Different metal weight percentages cause variations in the slump and tack values of the pastes. The solder paste database stores the paste parameters which determine the suitability of the solder paste in the reflow and manufacturing process. The key parameters that are stored are the viscosity, slump, tack, CPT values (determines the solderability of ceramic leadless components), wetting balance test values, etc. Although most of the factors are dependent on one another (for instance the slump value may be dependent on the metal weight percentage and viscosity, it becomes necessary to store all of the parameters as the relationships between various parameters have not been studied and defined.

4.6 User Interface

The user interface program of the system has essentially two main menu items. The first menu item called "File" is used to either read in the .dxf file (Open) or to exit the system (Exit). Once the "Open" command is executed, the program provides the user with the available list of .dxf files. The user is required to select the appropriate ".dxf" file for analysis. Once the file has been read in, the user would have to select the Start-Execute command from the menu provided in the window. The command will provide the user with a series of listbox-dialog boxes. The user is required to make the necessary selections of the manufacturing inputs from the listboxes. The sequence of operations required from the user is shown in Fig. 4. At the end of all the input selections, the user is asked to determine the basis for design analysis and is provided with three alternatives – solderability, reflow and cleanability. The user's inputs are then stored in the appropriate database files. The design information from the .dxf file is stored in the pcb.dbf file. The fields in this file are the name, x_loc, y_loc, and orient of the board. The manufacturing inputs provided by the user are stored in the floor.dbf file. The fields of this file are name, paste, board, flux, solvent, equipment. As indicated by the names, the selections made by the user are stored in their respective fields of the database. The rules stored in the design advisor's knowledge bases are now used for design review.

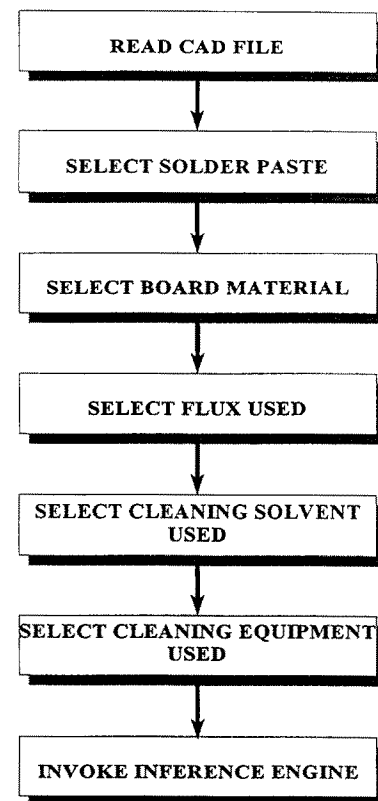


Fig. 4. Steps executed by the user interface.

4.7 Update Mechanism and Inference Engine

The knowledge update mechanism is an integral part of the design advisor. The update mechanism can be used as a stand alone system to build rules and create a knowledge base. This method was used to create the three modules of the design advisor DFM system. The main window of this program has six menu items (described below).

File: This section allows the user to perform the standard file management operations such as opening (Open Rule), creating (New Rule), saving (Save Rule As) files (in this case-rules) and exiting the system (Exit System). The entire program is in the multiple document interface (MDI) format. The MDI feature under the MS-windows operating system allows opening of multiple files at the same time. Under this system all the files share the same command menus and the window (file) that is on the topmost layer responds to the command executed by the user. The new rule command generates a window that has the basic framework of an entirely new rule with just one mode labelled "NEW_RULE". From this node either through the edit command or through the drop down menus, children of the node can be generated.

Edit: The edit command menu allows the addition (Add Children), deletion (Delete Branch), modification (Change Attribute), and assignment (Logical Operation) of the rules. The same operations can be performed through the use of drop down menus that will automatically pop-up on the clicking of the left mouse button in the vicinity of any rule node. The "Add Children" command generates a dialogue which requests the user to specify the name of the child node, the weightage associated with it and the logical mode of the child, i.e., if it is an "and", "or", or "not".

Each kind of node is automatically provided with a colour coded connector with the parent. The "or" nodes possess a green connector, the "and" node a black connector and the "not" node a red connector. Only tree structures can be created using this method and hence a network structure is eliminated. The "Delete Branch" command not only deletes the specified node but also all its associated children. Before it does this, it issues a warning to the user to confirm the delete operation. The "Change Attribute" command allows the user to change the attributes of the node by either changing its name, weight or logical characteristic. In addition to the three attributes (mentioned above), each node has another feature called the "explanation" feature. This is the feature that allows the generation of the output once a rule passes on firing. The text that is written in the explanation section of the node attributes is copied exactly onto the output screen as soon as the rule is passed.

The "Logical Operation" feature of the edit command allows the comparison (greater than, less than, equal to, not equal to) of a variable (or fact) with a certain value or string specified by the user. This feature in effect can be considered to be the heart of the rule formation operation. The generation or declaration of a variable (or fact) is explained in the next menu item description (Fact). However, before proceeding on a discussion of the "Fact" command, it is important to note that a "Logical Operation" command is to be executed only on the leaf nodes of a rule-tree. This is because all the

interior nodes of a tree are merely abstractions of their child nodes and if the child or all children node pass (depending on "or" or "not") then the parent node automatically passes.

Another important user interface feature that is provided is the colour coding of the nodes themselves. All blue-coloured nodes are necessarily parent nodes of some child node while a red coloured node indicates a leaf node on which no logical operation has been performed. In this case, either further children are to be added to that node or a logical operation is to be assigned to that node to complete the rule. All grey coloured nodes are leaf nodes on which a logical operation has been performed and therefore indicate completeness of the rule branch.

Fact: This command menu primarily performs the registration and declaration function of all variables that are to be used in the creation of a rule module. A variable or fact can be considered to be a parameter that is used to transfer values or strings from databases or functions into the rule module. In order to use a variable, the first step is to declare one. The "Fact-database" command allows the user to do exactly that. A dialogue box is generated requesting the user to specify the name of the variable, the type of the variable (long, double or string), the database and field that the variable would have to search in order to perform the logical operation. The databases that are available for the modules to search from are automatically listed in the dialogue box. Once the database is selected, the available tag names of the database file and the available fields in the database are listed for the user to easily browse (or scroll) through. The user then has to determine the field of the database through which the variable should search for the string specified in the "Logic Operation" dialogue box (please refer to section on "Edit" command).

There is an important feature in the "Fact-database" dialogue box called the "search" function. The search function determines the basis for searching a database for a particular record before a comparison of the specific field with a string or a value is made. The parameter that is usually entered in this area is another variable that points to the primary key of the database. The search function drastically reduces the search space as the inference engine knows precisely which record to look for in any database while attempting to match a variable with a value or a string. A detailed discussion of this process can be found in [14].

The second feature associated with the "Fact" command menu is to declare functional modules and use the to perform calculations during the firing of the rules. Since the inference engine by itself is not capable of performing any calculations, all the functions and calculations are performed by "dynamic linked libraries (dll)". A dll is typically a C program with a number of functions in it. However, instead of compiling this program into an executable form, Microsoft Windows allows the compilation of the program into a dll. A unique feature of a dll is that it allows the linking of the program (or the library) into any other program at run time instead of at compile time. Owing to this feature, all that a calling program has to know is the function prototype of any function, i.e. the number and type of parameters that will be passed onto a function within the dll. The "Fact-register function" command

requests the user to register the functions that will be used during the creation of the rule module. Here the user has to specify the name of the function, the dll that has to be used to access the function, and a prototype of the function, i.e. its input parameter types and its output parameter type. The “Fact-functional module” is quite similar to the “Fact-database” feature. Here, instead of declaring a variable that would be used to search a database, a variable that would be used to store the return value of any function that has been registered using the “Fact-register function” feature is declared.

Run: The “Run” option of the command menus performs an execution operation of the rule. It invokes the inference engine to begin operating on the rules and generates the output when a rule module is processed.

Option: The option feature of the update program allows simple database management functions. The “Option-Maintain” command allows the “packing, reindexing and linking” of the rule-databases. The “packing” function physically removes the records that have been marked for deletion from the database. For instance, when a certain branch of the rule is deleted, the relevant records in the rule-database are not deleted but only marked for deletion. It is necessary to “pack” the database in order to physically remove the records from the file. Similarly, once a modification to the record numbers in a database has been made either through deletion, addition of records, it is necessary to reindex the index files to facilitate correct search through the database. The “Option-Default” command requests the user to specify the starting database file on the basis of which the inference engine begins operating. Here the user is required to specify the rule module’s name, the starting (or initial) database file name (.dbf) and the name of the tag from the index file (the basis for sorting the file) of the initial database file.

Window: The final feature of this utility known as the “Window” option allows the user to perform standard window viewing manipulations. It allows viewing the window in the form of icons, in a cascaded form or tiled form. In addition, it allows the closure of all windows if required and also displays the status of all the open windows and indicates if a certain window is in the active window status or not.

4.8 Rule Structures

This section describes the structure of the rule files used within the system. Table 1 shows the important rule structure namely the framework of the .rbs (rule) datafile. The distinguishing characteristics of a rule (.rbs) are the address (a distinguishing number unique to the record), the name (of the rule), the level of abstraction of the rule (the smaller the number the greater the abstraction), the logical mode of the rule (“and”, “or”, or “not”), the weightage of the rule, the record number of the parent rule, the number of children

that it has, etc. The “pass” field refers to the string that is to be generated if the rule passes. The “operator” field refers to the operation that the rule has to be checked for when it is fired (<, >, = or <>). The “value” field keeps track of the string or value with which the comparison operation is to be performed. The field named “variable” keeps track of the search variable that is to be used while performing the search through the databases to access data (see Section 4.4. – Fact). The boolean field keeps track of the pass or fail status of the rule after it has been executed by the inference engine. The last three fields – “mode_date”, mod_time” and “creator” – refer to the time, date of creation of a rule, and the person who created the rule.

5. Validation

The design advisor was tested for several scenarios. The outputs generated were checked and validated at each stage by visually inspecting the databases in which they were stored and validated by verifying that this is indeed the output that would normally be expected. Validation provides a form of inherent verification of not just the body of knowledge represented within the system, but also of the inference mechanism that manipulates this knowledge to obtain solutions to problems at hand. Test scenarios were run to check all the different components of the design advisor. Such testing was done periodically during all stages of the system development to ensure that the system being developed was performing the functions that it was intended to do.

The validation of the rule modules was performed by testing each rule that was created using the update mechanism for each of the different design files and checking if the output was consistent with the rules that were created. Before a test of the rule logic was done, tests were conducted to determine if the functional modules located in the math.dll were performing the calculations correctly. Table 2 lists the return values of the functions MOMENT() and DENSITY() for the designs pcb0, pcb1, pcb2 and pcb3. A comparison of the values of the moment of pcb0 and pcb1 indicates that for the same set of components a better mass distribution of the components across the board lowers the value of the x and y moment. Therefore the design pcb1 passes the criteria of acceptable thermal mass distribution. A comparison of the

Table 2. Results of the functions “density and “moment” on the different PCBs.

	pcb0	pcb1	pcb2	pcb3
Density	2.55	2.55	2.25	4.675
Moment	Fail	Pass	Pass	Fail

Table 1. Structure of the .rbs file.

ADDRESS	NAME	LEVEL	LOGIC	PAR_REC	POINT	CHI_NO	PASS
VARIAB	OPER	VALUE	BOOL	MOD_D	TIME	CREAT	

board density measured in terms of "number of leads per square inch" indicates that pcb0 and pcb1 have the same values, which indeed should be the case. The functions CTE(D,D,D) and PAD_LEAD(S) were also tested by passing relevant parameters to them and then checking if the output was consistent.

The rule modules were tested by using the designs pcb2 and pcb3 as inputs. The shopfloor parameters that were used to test the pcb2 and pcb3 are listed in Table 3. The output generated by the design advisor for pcb2 and pcb3 were then manually verified. For example, for pcb2 the cleanability module cautioned the designer against using plastic components (PLCC44 and PLCC52) since the cleaning solvent used in the shopfloor (perchloroethylene) is a chlorinated solvent. In addition the cleanability module also determined that the solvent used may be too strong for the flux being used (rosin). In the case of pcb3, since the solvent used is R-112 and the flux used is rosin activated, the rules of the cleanability module determined that the solvent may not be able to clean the board adequately. In addition, the designer was informed that the transistor (SOT89) may cause problems owing to a combination of its low profile and the cleanability method used (spray solvent).

For both designs (pcb2 and pcb3) the reflow module detected a lead-to-pad coverage problem for the J leaded components (SOJ16, SOJ20). In addition, the rules check for a potential CTE mismatch problem for the chip components (RC1210, C6032) and the leadless ceramic chip carriers (LCCC24). Table 3 indicates that the density value of pcb3 is above the threshold criterion value (= 4.00). Therefore the rules automatically detect a density problem in pcb3. The output from the solderability module indicated that since pcb3 failed the moment criterion (Table 3), the thermal mass distribution was not acceptable. This module also determined that the high wetting balance value of the paste used in pcb2 (paste 3) tended to create solder wicking problems among the PLCCs, SOICs and the LCCCs.

6. Conclusion

The design advisor system described above uses an expert system framework. An OOP based methodology is identified for the integrated DFM environment. The primary advantage of using this approach is that it allows for a stepwise, modular development of the system. It also allows for a quicker development of the system owing to the properties of inheritance and polymorphism associated with OOP. Strategically, a system based on the above mentioned architecture is easier to update and maintain.

Table 3. Shopfloor conditions used to manufacture PCB2 and PCB3.

	Solder Paste	Substrate	Flux	Solvent	Clean-method
pcb2	PASTE3	Organic_base	Rosin	Perchloro-ethylene	Liquid solvent
pcb3	PASTE5	Ceramic_base	RA	R-112	Spray solvent

This design advisor was created with input from design and process engineers. Industry based experts helped format the inputs and outputs used. The knowledge content available in the rule modules of this research has not been tailored to any one specific shopfloor or set of machines. It is generic in nature. The PCB assembly and manufacturability problem has been analysed mainly in terms of the physics of the manufacturing process. However, the domain knowledge used could easily be tailored to the needs of any specific set of machines through the use of the update mechanism and the database management utility by domain experts possessing little programming knowledge.

This research serves to demonstrate how knowledge-based systems can be applied to the DFM function to improve the design to manufacture cycle time. The partitioning of the system's rules into separate databases has been done to provided for the features of updateability and expandability. This partitioning also helps in modular software development. The rule-editor has been critical in providing a user-friendly manipulation of the data and the rules within the system. The system utilises the concept of dynamic linked libraries (dlls) to perform the required calculations during the firing of rules. Another important feature of the entire system is that since the rules are not hard-coded into the program, it is not necessary to recompile the rules once they have been updated. The inference engine directly accesses the relevant databases (as specified in the rules), performs pattern-matching or numerical evaluation (as needed) and returns the explanation component of the rules as the output once the rule succeeds. The knowledge update system developed and used provides a new level of knowledge representation and may be used to develop rule-based troubleshooting systems in any domain.

Acknowledgements

This research was funded in whole or in part by the New York State Science and Technology Foundation, the National Science Foundation (NSF grant EEC-9108850), and the industrial sponsors of the Integrated Electronics Engineering Center (IEEC).

References

1. K. Srihari and C. R. Emerson, "Knowledge Based Manufacturing Problem Diagnosis in PCB Assembly", *Proceedings - International Electronic Packaging Society Conference*, Austin, Texas, pp. 245-257, 1992.
2. D. R. Wilson and K. K. Hutchinson, "Integrating Design with Manufacturing", *Proceedings - NEPCON West*, March, Anaheim, CA, pp. 446-455, 1990.
3. C. Lea, *A Scientific Guide to Surface Mount Technology*, Electrochemical Publications, Ayr, Scotland, 1988.
4. R. Prasad and R. Aspandiar, "A Comparison of IR and Vapor Phase Reflow Soldering for Surface Mount Assemblies", *Proceedings - NEPCON West*, Anaheim, CA, pp. 421-437, February 1988.
5. R. Prasad, *Surface Mount Technology*, Van Nostrand Reinhold, New York, 1989.
6. C. F. Coombs Jr, *Printed Circuits Handbook*, McGraw Hill, New York, 1988.
7. T. J. Byers, *Printed Circuit Board Design with Microcomputers*, McGraw Hill, New York, 1991.

8. G. L. Ginsberg, *Printed Circuits Design*, McGraw Hill, New York, 1991.
9. D. D. Bedworth, M. R. Henderson and P. M. Wolfe, "Computer Integrated Design and Manufacturing", McGraw Hill, New York, 1991.
10. T. Gupta and B. K. Ghosh, "A survey of expert systems in manufacturing and process planning", *Computers in Industry*, **11**(2), pp. 195–204, 1988.
11. G. F. Luger and W. A. Stubblefield, *Artificial Intelligence and The Design of Expert Systems*, Benjamin Cummings Publishing, Redwood City, California, 1989.
12. D. W. Rolston, "Principles of Artificial Intelligence and Expert Systems Development", McGraw Hill, New York, 1988.
13. S. R. T. Kumara, S. Joshi, R. L. Kashyap, C. L. Moodie and T. C. Chang, "Expert systems in industrial engineering", *International Journal of Production Research*, **24**(5), pp. 1107–1125, 1986.
14. Y. L. Hwang, An adaptable knowledge base environment for design for manufacturability by object-oriented approach, *Master's Thesis*, State University Of New York, Binghamton, New York, 1993.
15. R. J. Rapczynski, "Improving manufacturability with good SMT design practice", *Surface Mount Technology*, August 1989, pp. 26–29.
16. K. D. Mueller-Glaser and J. Bortolazzi, "An approach to computer aided specification", *IEE Journal of Solid State Circuits*, **25**(2), pp. 335–345, 1992.
17. R. Taylor, *Object Oriented Information Systems – Planning and Implementation*, John Wiley and Sons, New York, 1992.
18. M. Alexander, "Manufacturing cost analysis of surface mount PCB assemblies", *Master's Thesis*, State University of New York, Binghamton, New York, 1993.

Glossary

Chip Component	Typically stands for resistors, capacitors, diodes, and other passive components. It could include integrated circuits also.	CTE mismatch	expansion of material as the temperature changes (increases). The absolute difference in the thermal expansion of two materials.
LCCC	The leadless ceramic chip carrier is a ceramic package with leads around and down the sides of the package.	Flux	A chemically and physically active material that promotes wetting of solder and helps in the formation of a solder joint.
Component	A single part or a combination of parts that together performs a specific design function.	Fine pitch component	A component with a lead pitch less than 0.65 mm lead pitch.
Component lead	The wires or conductors that extend from a component. They provide mechanical and physical connections.	Integrated circuits	Electronic components assembled together on a single substrate to perform a specific electrical function. Typical surface mount packages that contain integrated circuits include quad flat packs, plastic leaded chip carriers, and small outline devices.
Coplanarity	Used to denote the relationship of leads or pads (on a board) to each other in the horizontal plane.	Ionic contaminants	Residues on the printed circuit board after the assembly process including flux, finger prints, etc.
CTE	The coefficient of thermal expansion (CTE) represents in linear terms the	Land	A conductive pattern etched on an unpopulated board. Typically used for the connection of components.
		MELF	A metal electrode face bonded chip carrier with metalized connections.
		Printed circuit board	A board with metal interconnecting traces.
		Surface mount component	A component that is designed to be mounted and attached (typically by soldering) onto the surface of a printed circuit board in contrast to the traditional insertion mounting method.
		Solder	An alloy of different metals typically with a melting point less than 200°C.
		Solder paste	Solder particles, flux, and other binders are combined together to form a paste. The size of the solder particles and the volume of the constituents are tightly controlled.
		Stencil	A thin sheet of metal with a circuit pattern etched or cut into it. Typically used for depositing solder paste.
		Via	A plated through hole used to connect two or more layers of a board.
		Wetting	The formation of a smooth, uniform, and unbroken film of solder to a metal surface.