Attention has recently focused on improving the information and knowledge flow between design and manufacturing. Interfacing Computer Aided Design (CAD) with Computer Aided Manufacturing (CAM) systems has helped this information flow while also improving both processes. Once CAD and CAM systems were introduced the obvious next step to (1) decrease a product's time-to-market and its costs, while (2) increasing the product's quality, has been to utilize knowledge based system technology during both the design and the manufacturing phases of a product. Unfortunately, the knowledge based systems that we have developed have been designed as stand-alone components. These have been built using a variety of implementation philosophies, styles of reasoning and are deployed over distributed computing environments. The companies with large investments in design and manufacturing automation, as well as in expert system technology have come to realize that achieving the two goals stated above will require the integration of such diverse systems in ways that will allow their coordination and cooperation. This special issue of the Journal of Intelligent Manufacturing has brought together reports of recent research whose goal was this type of integration.

1. Knowledge level incompatibility

We have spent the last ten years developing knowledge based systems for a variety of independent tasks that are performed during a product's design and manufacturing phases. In order to implement such systems we introduce rule based programming languages such as Prolog (Comerauer, 1978), and OPSS (Forgy, 1984), frame-based languages such as Flavors (Cannon, 1982), and CLOS (Bobrow, 1988), and expert system development environments such as KEE (Intellicorp, 1988) and KnowledgeCraft (CGI, 1988). Even though other types of computer software have moved towards compatibility, and interconnectivity, knowledge based systems have not.

The lack of compatibility and interconnectivity inhibits expert systems from cooperating because they cannot communicate at the knowledge level. Newell (1981) has given the following definition of the knowledge level: 'There exists a distinct computer system level, lying immediately above the symbol level, which is characterized by knowledge as a medium and the principle of rationality as the law of behavior.' Communication at this level is performed directly using the knowledge structures (plans, actions, goals, models) that are manipulated by the expert systems. At this level, each expert system uses its knowledge in order to attain its local goals or goals that are established by other systems with which it cooperates. Unfortunately, communication between expert systems is currently performed only through low level data structures that are semantically incompatible. That is, communication between expert systems at such low levels requires that the operational model of each system's processing is known to the other systems.

2. Integrating expert systems

Why should we want to integrate expert systems at the knowledge level? First, because we want to be able to solve problems at the organization level rather than at the individual level. Consider the following scenario that has been paraphrased from Huhns (1990b): an automotive parts manufacturer installed an expert system at each machining operation island in a parts' plant in order to monitor the parts produced. When the expert system that monitors a particular machine detects that the machine has produced an unacceptable number of defective parts, it signals an operator to shut the machine down for maintenance or repair. Unfortunately, the monitoring expert system does not coordinate the signaling action with any of the other expert systems in the plant. As a result of shutting down the machine, parts (the machine's output had it been...
The expert systems that monitor these latter machines shut them down when they detect that there are no materials for the machines to process. However, if the expert systems in the plant could be coordinated then they could develop plans of how to maintain the operation of the plant when one of the machines is taken out of operation. Solving problems at the organization level is also the goal of concurrent engineering. This emerging discipline aims to provide tools and representations to allow the simultaneous execution of several engineering processes, while allowing the sharing of the results of these processes either with human agents or with other processes.

The second reason for integrating expert systems at the knowledge level is in order to solve problems across organizational boundaries. For example, the servicing of a product becomes easier when the service engineers have access to the necessary design and manufacturing information. Furthermore, problems that are found in a particular version of the product by a diagnostic expert system can be effectively communicated to the appropriate expert system that are used during the design and manufacturing phases of future versions of the product. In this way one guarantees that these future versions will not have any of the problems that have already been identified. The initial steps towards such cooperation have already been taken by the definition of data exchange formats between CAD and CAM systems. Unfortunately, data exchange formats have not been followed by the specification of knowledge exchange formats. Furthermore, the level of the data that is being exchanged using the defined formats is too low to be useful by expert systems.

3. Achieving the necessary integration

In order to achieve our goal of integrating expert systems at the knowledge level we need to:

1. Provide generic expert system development environments that allow the creation of cooperating knowledge based systems. Examples of such generic environments are described in Huhns 1990a, Gasser 1987.

2. Define methods for semantically integrating and coordinating existing knowledge based systems in order to make possible their cooperation.

Providing generic expert system development environments does not automatically allow us to meet our goal. These development environments must be capable of encoding the operation models of an organization. The encoded models are then utilized by the expert systems that are implemented using the development environments. The encoded models must include the objects that are utilized during a product’s lifecycle by each department in the organization and the processes that operate on these objects. Multiple views of the represented objects, some department specific and others organization-wide, must also be possible considering the operation models of the organization to which they are applied. In particular, they must include the objects that are utilized during a product’s lifecycle and the processes that operate on these objects. For example, a personal computer contains, by design, the integrated circuits of the types $I_{d1}$, $I_{d2}$, and $I_{d3}$. In order for the computer to be manufactured these circuits need to be purchased from a particular vendor. The purchased circuits (say $I_{m1}$, $I_{m2}$, and $I_{m3}$), in addition to the characteristics for which they were chosen by the design organization, also have characteristics that are particular to their manufacture. Therefore, for each personal computer component we have at least two types of objects: those used by the designers, and those used by the manufacturing engineers. These objects can be unified into a single hierarchy. However, they can also be organized into hierarchies based on their characteristics that are of interest only to one department (design or manufacturing). The expert systems that are used by the two departments can be integrated through the encoded operation model and the unified hierarchy while continuing to function in the environment for which they were implemented. The number of all these objects is potentially large thus prohibiting their constant storage in a computer’s main memory. Therefore, it is imperative that such objects are stored in appropriate databases and are retrieved as necessary.

In order semantically to integrate and coordinate existing expert systems we need to define ways for communicating knowledge. This is not only an issue of translating between the different formats in which knowledge is expressed in the various systems. The component systems need to be able to interpret this knowledge and act based on this interpretation. For example, two expert systems $E_1$ and $E_2$ are using the term 'system'. $E_1$ detects and corrects problems in the VMS operating system. $E_2$ detects and corrects hardware problems in VAX CPUs. In $E_1$ the term 'system' refers to operating systems and is represented as a CLOS object. In $E_2$, it refers to computer CPUs and is represented as an OPS5 working memory element. Assume that $E_1$ needs to coordinate its problem-solving efforts with those of $E_2$. For this reason it communicates to $E_2$ the following request: 'Can you help me solve a memory problem with the system that I monitor?' In this case, a mere translation of the term VMS from a CLOS object to an OPS5 working memory element will not suffice. Instead the two systems must communicate using either a shared vocabulary or two distinct vocabularies which have been formed out of the same lexical and semantic core. The following issues are important while addressing semantic integration: knowledge representation, system architectures, and control.

The papers included in this issue address one or more of the four issues stated above (knowledge representation,