SRC TR 88-5

An Approach to Integrating Manufacturing Systems

by

G. Harhalakis, L. Mark, A. Johri,
and B. Cochrane

An Approach to Integrating Manufacturing Systems

G. Harhalakis[1], L. Mark[1], A. Johri[1], B. Cochrane[2]

Systems Research Center
and Departments of Mechanical Engineering
and Computer Science

University of Maryland, College Park, MD 20742.

[1] Supported in part by the NSF under grant CDR-85-00108
[2] Supported by the NSF under grant CDR-85-00108

AN APPROACH TO INTEGRATING MANUFACTURING SYSTEMS

By

G. HARHALAKIS[1], L. MARK[2], A. JOHRI[1], B. COCHRANE[2]

[1]Department of Mechanical Engineering, University of Maryland

[2]Department of Computer Science, University of Maryland

ABSTRACT. Computer Integrated Manufacturing is often misconstrued as simply the integration of CAD and CAM. In fact it is much more. It is the systems approach of tying together the various automation tools available today, so as to enable the control of an entire manufacturing operation. This includes the business functions as well. The generally accepted way to go about this is to develop a database management system, with the required capabilities. It is stressed here that Manufacturing Resource Planning - II (MRP-II), has the best inherent features, for the tying together of the various manufacturing functions. Initial work deals with Computer Aided Design (CAD) and MRP-II integration, the integration being centered around parts specifications, product structures, and engineering changes. A model for this integration is presented along with the rules of interaction between the systems. The model is based on an interoperability system, and uses a formal language named 'Update Dependencies' which has been defined for specifying the various operations in and between MRP-II and CAD. This has been used to test the model design specification. Future work includes the introduction of Computer Aided Process Planning (CAPP), to the present model.

INTRODUCTION. Feeling the pressure from foreign and domestic competitors, many US companies have had to incorporate some of the large number of technologies

available today, which automate individual functions of a manufacturing facility. Some of these technologies being, Computer Aided Design (CAD), Computer Aided Manufacturing (CAM), Manufacturing Resource Planning (MRP-II), Flexible Manufacturing (FMS), Group technology (GT), and Computer Aided Process Planning, to name just a few. There is no doubt that these technologies when implemented correctly, do help in promoting efficiency, within the specified area. It has however been visualised that, if some of these could be interfaced with one another, the resulting efficiency and productivity, would increase manifold. This interfacing, or the rule based automated communication, within related areas of a modern manufacturing facility, is what is known today as Computer Integrated Manufacturing. As the concept of CIM came much later, there was a proliferation of heterogenous hardware, and software, which optimised each function at a very narrow area of focus. As is also common these days, due to lack of any set standards, scores of different pieces of software became available for each functional area, each formatting its database differently. So the situation as it stands today is that the different functional areas have very little data commonality, and different companies have different software for automating the same areas. These two factors by themselves make up the major part of the barrier towards CIM. The amount of capital invested in existing software and hardware neccesitate modular and piecemeal integration implementation. Since the same areas have different existing software, with different data formats, one CIM implementation effort which works irrespective of the kind of software used, is not feasible. Instead, rules for integration will have to be developed for generic systems, along with the complete implementation models for such systems. This will serve as the general means for CIM. This generic CIM model, with minor modifications, depending upon existing software in a manu-

facturing plant, will result in an actual working CIM facilty.

It is proposed that MRP-II should serve as the coordinating and controlling medium of all of these technologies. This is because MRP-II alone has the position of posessing data commonality, with most of the computer aided technologies available today. So any kind of integration effort should begin with MRP-II as the hub, of an integrated system [1].

Although the concept of CIM has been loosely defined, neither a thorough system architecture, nor a requiste integration vehicle is yet available. There are many problems and issues which must be resolved, before CIM is possible, and the question of database architecture is of the utmost importance [2]. There are two approaches to designing a database architecture. The first is a 'general database', which is a single database, maintaining all the system data, and is accessible by all the system functions. The second is a system of multiple data bases, each function having its own database, with interoperability capabilities, for effective communication with other databases. The latter approach is supported for a variety of reasons, the primary one being that too much capital has already been invested in existing software and hardware. It would be economically unfeasible, designing the entire CIM system from scratch. Where possible, the integration efforts should aim at utilizing existing systems.

This paper incorporates a brief review of the model description, and the principles of multidatabase interoperability as developed in our previous work [3,4]. Details of the update dependencies language are presented, along with the demonstration of certain functions, including product structures.

SPECIFICATIONS FOR THE INTEGRATED SYSTEM. The first area being considered for integration with MRP-II is Computer aided Design (CAD). The data commonality

between the two systems is well established: product definition. CAD facilitates the creation and design of parts and assemblies, where assemblies are really just arrangements of component parts. MRP II plays the role of cataloguing each part and assembly by part number and description, and defining the product structures.

More specifically, the elements common to MRP IIand CAD addressed by the integration as follows:

-Part Specifications.
-Bills of Material.
-Engineering changes.

The part data maintained by each system is shown in Figure 1. General part data is maintained for each part and is retrieved by part number; in addition to this data, the effectivity start and end dates and status code (different for each revision) are maintained seperately under revision data.

For Each Part Number

CAD	MRP II
Part Number	Part Number
Drawing Number	Drawing Number
Drawing Size	Drawing Size
Description	Description
CAD Unit of Measure	BOM Unit of Measure
	MRP (Purchasing) Unit of Measure
	UOM Conversion Factor
	Source Code
	Cost
	Leadtime
Supersedes Part Number	Supersedes Part Number
Superseded by Part Number	Superseded by Part Number

For Each Revision Level

CAD	MRP II
Part Number	Part Number
Revision Level	Revision Level
Effectivity Start Date	Effectivity Start Date
Effectivity End Date	Effectivity End Date
CAD Status Code	MRP II Status Code
Drawing File Name	

Fig. 1 Part Specification Data Maintained By Each System

Product structures are represented and recorded by the individual parent-component relationships comprising them. The product structure related data maintained by each system are shown in Figure 2.

CAD	MRP II
Parent Part Number	Parent Part Number
Parent Revision Level	Parent Revision Level
Item Number	Item Number
Component Part Number	Component Part Number
Quantity Per Assembly	Quantity Per Assembly

Fig. 2 Product Structure Data Maintained by Each System

The working of the model can be represented by examining the status codes associated with each part and revision. Each of the functions have their own status codes, as shown below.

CAD PART STATUS

W - "Working"; not a completed drawing, used prior to approval, and not transmittable to MRP II.

R - "Released"; an active part.

H - "Hold"; under review, pending for approval, possibly with a new revision level. Part should not be used by either system.

O - "Obsolete"

MRP-II PART STATUS

R - "Released"; active part.

H - "Hold"; not to be used by MRP II.

The basic interrelationships of the two systems are represented with the help of status code diagrams, showing the flow of information and the status of each part in both systems during a given activity. In the following, the basic operations using Part Master Records and Product Structures between CAD and MRP II are described using status diagrams.

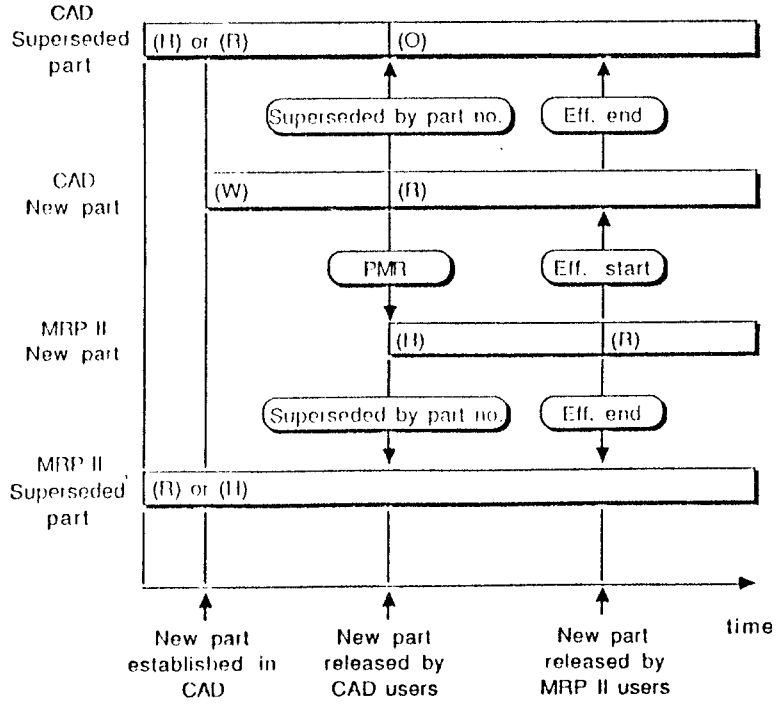

Fig. 3 Status diagram for the Creation of a New Part

PART MASTER RECORDS

In Figure 3, as shown above a new part is first created by a CAD user as a working drawing with status code "W". At this point, no information about the part exists in MRP II. Upon completion and approval within CAD, the part is released by a CAD user with a CAD status code "R".

The release of the new part within CAD results in the update of a skeletal Part Master Record (PMR) in MRP II using the CAD Part data. Because the PMR is not complete, and to give manufacturing time to plan for the procurement or the manufacture of the part (e.g. to search for vendors, or to develop routings), the part is given a status of "H" in MRP II. When an MRP II user completes the PMR, the part is ready to released within MRP II. The provision is made for MRP users to place a hold on the part at any time, without affecting CAD. This may be necessary due to machine breakdowns, or vendor problems, or other manufacturing related problems. MRP can place a hold on a part without affecting CAD. Once held, MRP can rerelease the part at any time.

If the new part is meant to supersede another, the status of the superseded part is changed in CAD to obsolete with a status code 'O', regardless of whether the old part previously had an "R" status or an "H" status. This occurs the moment the new part gets a released status in MRP II. In MRP II, the changeover to the superseding part is performed automatically, by virtue of the effectivity start date of any higher level assemblies calling for the new part as part of a revision change.

Similar diagrams have been developed for part obsolescence, deletion, and changes of revision codes.

PRODUCT STRUCTURES

The basic interoperability functions required to maintain consistent product structures or Bill of Material information have also been developed. Typically CAD will be the source of the first product structure for a given new assembly, as well as the origin of engineering changes requiring the modification of an assembly.

If the addition of a component calls for either a revision change or a new part number at the assembly level [5], the new revision or part should be created first, and the addition made to the new Bill of Material, regardless of whether the addition is being made via CAD or MRP II. If the new revision or part is not created prior to the operation, the system still allows the user to create it as a part of the operation.

To add a component relationship via CAD, all fields in the record must be specified:

- Parent Part Number
- Parent Revision Level
- Item Number
- Component Part Number
- Quantity Per Assembly

The indicated revision of the parent part, or assembly, may have a CAD status of either "working", "hold", or "released"; it cannot have an "obsolete" status, however. Typically, designers will create the product structure of an assembly before it is finalized, i.e., while it has "working" status. If the assembly revision has either a "hold" or "released" status in CAD, the system also requires that the part/revision combination exists in MRP II. These requirements allow the product structure of an assembly to be compiled prior to

its release from CAD, but prevent the creation of a product structure for an assembly that has been deleted from MRP II but not from CAD.

The component part must have a revision level with a status of either "hold" or "released" in both CAD and MRP II. Thus, parts cannot be used as components in product structures prior to the release of their first revision by CAD users and the subsequent creation of a Part Master Record in MRP II. Again, the system does not allow the use of parts deleted from MRP II (but not necessarily from) CAD to be used as components in product structures.

In addition, the system checks to make sure the item number assigned to the component part has not already been used by another relation in the same structure. The system does, however, allow the same component to be recorded in two or more different relationship records, allowing users to break up the quantity of a component into two or more items to better represent the manufacturing sequence of operations of the assembly.

Finally, the system ensures that the resulting relationship does not cause a "loop," i.e., a use of the assembly as a component of itself. To do so, the component part number must not be the same as the assembly part number at any level of the Bill of Material structure; consequently the system searches through all levels of the product structure of the component part (if applicable) for occurrences of the parent part number. If any are found, the relationship cannot be added.

If the database checks are successful, the relationship record is added to the CAD system. If any one of the database checks fails, a message describing the failure is printed, and the operation fails.

The timing of the transfer of the component relations from CAD to MRP II is dependent on the CAD status of the part/revision combination. The two possible

cases are shown in Figure 4. In the first case, shown in Figure 4a, the revision of the assembly being constructed begins with a working status. This is perhaps the most common situation, as designers typically construct the Bill of Material for an assembly before it is released. As is the case with the data associated with the revision itself (and part data as well, if the revision in question is the first one for the assembly), the component relationship data remains local to CAD until the part is finalized and released. When it is released, the component relationship data is added to the MRP II database at the same time as the revision data, which is assigned an MRP II status of "hold". When MRP II users have determined and entered any information not provided by the designers of the assembly, such as the effectivity dates, the revision is released to MRP II. It is then the effectivity dates of the new revision which determine the new relationship becomes effective.

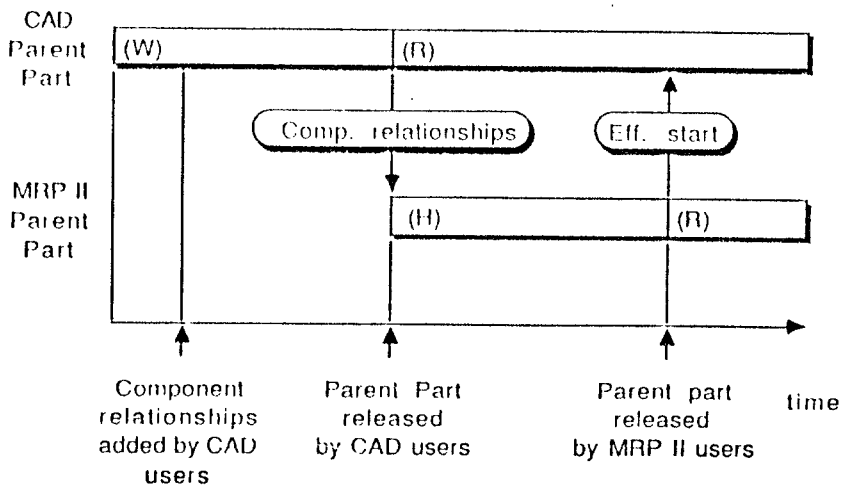

Fig. (4a)

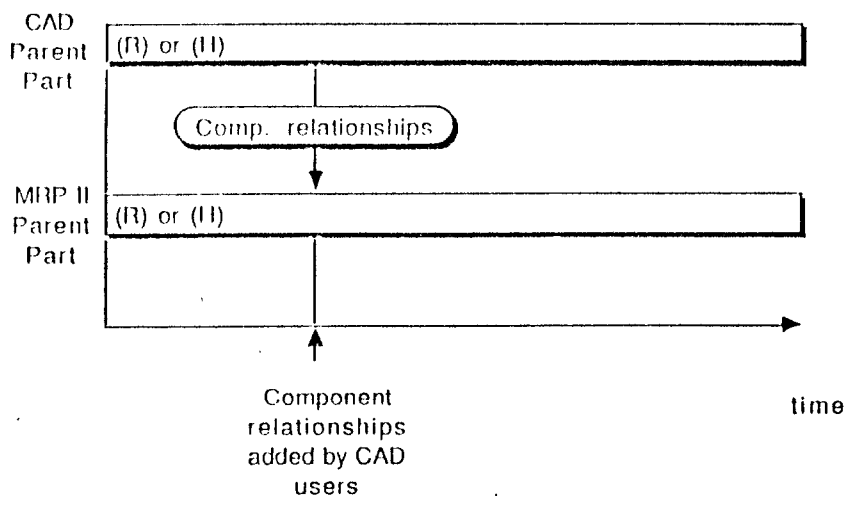

Fig. (4b)

In the second case, depicted in Figure 4b, the revision of the assembly being formed has already been finalized and has either a "hold" or "released" status in CAD. Since parent parts such as these exist in MRP II as well as CAD, the component relationships are immediately transferred to MRP II as well. If the parent part has a "hold" status in CAD, meaning that it is under review, the system allows the construction of product structures for such parts, since they may again be made active at some later date. As mentioned above, the parent part/revision combination must exist in both systems for a relationship to be defined. Because the design is already finalized, the relationship is transferred to MRP II immediately. Further, because a hold on a part/revision in CAD automatically implies a hold on the same part/revision in MRP II, the revision of the assembly being formed will also have a "hold" status in MRP II.

Being on hold in MRP II, the assembly part/revision may or may not have effectivity dates, depending on whether it had a "released" status prior to being put on hold, or was just transferred from CAD. In either case, by the time the assembly revision is released or rereleased in MRP II, it will have effectivity dates for determining the validity of its relationship records. If the parent part has a "released" status in CAD, the assembly revision may have an MRP II status of either "hold" or "released". If it has a "released" status in MRP II, then the effectivity dates of the revision should have been defined. If it has a "hold" status, then the effectivity dates may or may not have been defined, as described above, but will be defined prior to the release or rerelease of the revision.

Similar diagrams have also been developed for deleting component relationships, substituting components in relationships, and modifying component quantities.

MULTI DATABASE INTEROPERABILIY. To implement and demonstrate the integrated MRP II/CAD system, the concept of database interoperability is being utilized. This session presents this concept, which is based on the Update Dependency language currently under development in the department of Computer Science at the University of Maryland [6] as a tool for achieving interoperability. The language is being applied to the current problem as a means to critically analyze the design of the integrated system as well as to analyze the effectiveness of the language in specifying such a system. Therefore, a rule set is constructed for the integrated system, called update and retrieval dependencies, which controls inter-database consistancy through inter-database operation calls. This rule set is used here to enforce the functionality of the

integrated MRP II/CAD system.

In the following sections, the syntax and semantics of update dependencies are more formally presented.

SYNTAX

A compound update operation is defined by an update dependency with the following form:

```
 <op>
----> <c1>,
      <op1,1>,
      <op1,2>,
      .
      .
      <op1,n1>.

----> <c2>,
      <op2,1>,
      <op2,2>,
      .
      .
      <op2,n2>.

      .
----> .
      .
```

where <op> is the compound update operation being defined, <opi,j> is either an implied compound update operation or an implied primitive operation, and <ci> is a condition on the database state.

A compound update operation <opi> has the following form:

<operation name>(<relation name>(<tuple spec>))

where the <tuple spec> is a tuple variable for the relation with the name <relation name> and consists of a list of <domain variable>s. The <tuple spec> in <op> is the formal parameter for <op>. All the <domain variable>s in the <tuple spec> of <op> are assumed to be universally quantified. All <domain

variables>s in the <tuple spec>s of <opi,j>, that are not bound to a universally quantified <domain variable> in <op>, are assumed to be existentially quantified. All <domain variable>s are in upper case; nothing else is.

The implied primitive operators are: 'add' for adding a new tuple in a relation, 'remove' for eliminating one, 'write' and 'read' for retrieving data by and from the user, 'new' for creating a unique new surrogate, and 'break' for temporarily stopping the system to do some retrieval before giving the control back to the system. The implied primitive operations <opi,j> have the following forms:

- add (<relation name>(<tuple spec>))
- remove (<relation name>(<tuple spec>))
- write ('<any text>'), or write (<domain variable>)
- read (<domain variable>)
- new (<relation name>(<tuple spec>))
- break

The <relation name> used in the operation 'new' must be the name of a unary relation defined over a non-lexical domain. The conditions <cond> are expressions of predicates. The connectives used informing the expressions are 'and' ($\wedge$) and 'not' ($\neg$). The predicates are of the form <relation name>(<tuplespec>) to determine whether or not a given tuple is in a given relation; or of the form 'nonvar(X)' or 'var (X)' to decide whether or not a <domain variable>, X, has been instantiated; or of the form X <comp>Y, where <comp> is a comparison operator.

Conditions, or retrieval dependencies, can also be used to retrieve data from the system.

SEMANTICS

A compound update operation succeeds if, for at least one of the alternatives in its update dependency, the condition evaluates to be true and all the implied operations succeed. It fails otherwise.

When a compound update operation is invoked, its formal parameters are bound to the actual parameters. The scope of a variable is one update dependency. Existentially quantified variables are bound to values selected by the database system or to values supplied by the interacting user on request from the database system. Evaluation of conditions, replacement of implied compound update operations, and execution of implied primitive operations is left-to-right and depth-first for each invoked update dependency. For the evaluation of conditions we assume a closed world interpretation.

The non-deterministic choice of a replacement for an implied compound update operation is done by backtracking, selecting in order of appearance the update dependencies with matching lefthand sides. If no match is found, the operation fails.

An implied compound update operation matches the left-hand side of an update dependency if:

- the operation names are the same, and
- the relation names are the same, and
- all the domain components match. Domain components match

if they are the same constant or if one or both of them is a variable matches a constant it is instantiated to that value. If two variables match they share value.

The semantics of the primitive operations are:

- add(r(t)); its effect is r:=r$\cup\{t\}$; it always succeeds; all components of 't' are constants.

- remove(r(t)); its effect is r:=r$\setminus\{t\}$ where all components of 't' are constants. It always succeeds.

- write('text'); it writes the 'text' on the user's screen. It always succeeds.

- write(X); writes the value of 'X' on the user's screen. It always succeeds.

- read (X); reads the value supplied by the user and binds it to 'X'. It always succeeds (if the user answers).

- new(r(D)); produces a new unique surrogate, from the nonlexical domain over which 'r' is defined and binds the value of the variable 'D' to this surrogate. It always succeeds.

- break; suspends the current execution and makes a new copy of the interpreter available to the user, who can use it to retrieve the information he needs to answer a question from an operation.

The list of primitive operations is minimal for illustrating the concept, however it can easily be extended. It is emphasized that primitive operations are not available to the users (i.e. they cannot invoke primitive operations).

The execution of 'add' and 'remove' operations, which are done by the system in an attempt to make a compound update operation succeed, will be undone in reverse order during backtracking. This implies, that a (user invoked) com-

pound update operation that fails will leave the database unchanged.

Update dependencies for a number of high level control abstractions which include while, repeat, do-for, if-then-else, and case statement, can be generated automatically. Therefore we are going to revise the language to include these abstractions, and make it more user friendly.

DEMONSTRATION. An interactive session for the demonstration of some of the basic functions of the MRP II/CAD system is shown in Figures 5-10. Initially, both the MRP II and CAD databases are empty. The CAD and MRP II part and revision records contain the information specified in Figure 1 and the product structure records contain the information specified in figure 2.

In Figure 5, a new part is inserted into the CAD database; the user is prompted for the data required to establish a record for the new part and one for its first revision level. The remaining fields, if not specified by the user, are given the value "unknown", as shown in the subsequent listing of the CAD part and revision records. The absence of the MRP II part and revision records verifies that no information has been transferred to MRP II yet.

```
ud> insert(cadpart(Pnum,Dnum,Dsize,Des,Buom,
           Spnum,Sbnum)).

Part Number?
|: 12345.
Description?
|: deluxe_widget.
Unit of Measure?
|: each.
New Revision Level?
|: 1.
Drawing File Name?
|: 'widget.prt'.
Revision Has Been Added
Part Has Been Added

ud> listing(cadpart).

cadpart(12345,unknown,unknown,deluxe_widget,
        each,unknown,unknown).

ud> listing(cadrevs).

cadrevs(12345,1,unknown,unknown,w,'widget.prt').

ud> listing(mrppmr).

ud> listing(mrprevs).
```

Figure 5. Adding a New Part to CAD

```
ud> releasework(cadrevs(Pnum,Rev,Estart,Eend,
                Cstat,Dfname)).

Part Number?
|: 12345.
Revision Level?
|: 1.

Revision Has Been Released

ud> listing(cadrevs).

cadrevs(12345,1,unknown,unknown,r,'widget.prt').

ud> listing(mrppmr).

mrppmr(12345,unknown,unknown,deluxe_widget,
       each,unknown,unknown,unknown,
       unknown,unknown,unknown,unknown).

ud> listing(mrprevs).

mrprevs(12345,1,unknown,unknown,h).
```

Figure 6. Releasing a New Part.

```
ud> insert(cadrevs(Pnum,Rev,Estart,Eend,Cstat,
          Dfname)).

Part Number?
|: 12345.
New Revision Level?
|: 2.
Drawing File Name?
|: 'widget2.prt'.

Revision Has Been Added

ud> listing(cadrevs).

cadrevs(12345,1,unknown,unknown,r,'widget.prt').
cadrevs(12345,2,unknown,unknown,w,'widget2.prt').

ud> listing(mrprevs).

mrprevs(12345,1,unknown,unknown,h).
```

Figure 7. Adding a new revision to CAD

```
ud> insert(cadpart(Pnum,Dnum,Dsize,Des,Buom,
           Spnum,Sbnum)).

Part Number?
|: 12345.
Description?
|: hammer.
Unit of Measure?
|: each.

Part Number Already Exists

ud> releasework(cadrevs(Pnum,Rev,Estart,Eend,
                Cstat,Dfname)).

Part Number?
|: 12345.
Revision Level?
|: 1.

Part Does Not Have Working Status
```

Figure 8. Preventing users from Compromising Database Integrity

In Figure 6, the first revision of the part just created is released, and the listing of records after the part was released shows the updating that has occurred in both CAD and MRP II. In CAD, the status of the revision data is changed from working to released; in MRP II, a part master record is created for the new part, as well as a revision record with a status of "hold" for the first revision of the part. Any data not included in the CAD record is given the value "unknown".

In Figure 7, the addition of a new revision level to the part is shown, and the listings after this addition show the revision record with a working status in CAD, but not recorded at all in MRP II.

The system is designed to prevent the user from inadvertently compromising the integrity and consistency of the two databases. Figure 8 shows two examples of this. In the first example, the user attempts to enter a new part using a part number that already exists. The system prints out a message, indicating

this, and stops the operation. In the second example, a user tries to release a CAD revision that is already released. Again the system prevents this action and prints out a message.

```
ud> listing(cadpart).

cadpart(12345,unknown,unknown,assemblyA,each,unknown,unknown).
cadpart(12346,unknown,unknown,component1,each,unknown,unknown).
cadpart(12347,unknown,unknown,component2,each,unknown,unknown).

ud> listing(cadrev).

cadrev(12345,1,unknown,unknown,w,file1).
cadrev(12346,1,unknown,unknown,r,file2).
cadrev(12347,1,unknown,unknown,r,file2).

ud> insert(cadcomponet).

Does this change require a new assembly part number? no.
Does this change require a new assembly revision level? no.
Parent part number? 12345.
Parent revision level? 1.
Item Number? 1.
Component part number? 12346.
Quantity per assembly? 1.
in insert cadcomp 1Component relationship has been added to CAD

ud> listing(cadcomponent).

cadcomponent(12345,1,1,12346,1).

ud> listing(mrpcomponent).
```

Figure 9. Adding Component Relationship to CAD

```
ud> substitutepartcad.

Current part number? 12346.
Part number to substitute? 12347.
in subpartcad, 1Relationship has been deleted from CAD
in insert cadcomp 1Component relationship has been added to CAD
Part substitution has been completed

ud> listing(cadcomponent).

cadcomponent(12345,1,1,12347,1).
```

Figure 10. Substituting Component in Relationship

In Figure 9, the listings show that one parent part and two component parts already exist in the CAD database; the parent part has a "working" status and the component parts are released. A new component relationship is inserted into

the CAD database. The user is prompted to enter the new assembly part number and new assembly revision level, and every other field in the relationship. At this moment, no component relationship exists in MRP II database. To establish component relationships in MRPII, one has to release the parent part.

In Figure 10, if a component in a relationship is to be replaced by another part, the substitute function can easily be used but the new component must have a "released" status. The listing shows that the old component has already been replaced by the new component.

DISCUSSION. Due to our desire for generality, our first CAD/MRP II integration system is a simplification of the activities and data exchange involved in a typical manufacturing environment. At present stage, the functions of this model are tested on one computer without using actual CAD and MRP II systems. This forces the user to interact directly with the interoperability system. As we can see, when our research progresses, the system model will become more and more sophisticated and practical. We are currently investigating the possibility to use a more formal modelling technique, such as petri nets. This will facilitate the analysis of the model, and some simulations to be performed, prior to programming.

Update dependencies provide a convenient formalism for organizing, expressing, and communicating algorithms, and have a declarative representation that allows the designer to express algorithms in a natural manner. This natural representation is appropriate for reviews and presentation since it is a concise statement of the user's notions. Furthermore, this formalism eliminates the tedious and error prone phase of translating algorithm into code.

Although Update Dependencies is a language that supports high level

thinking, there are a few disadvantages. Because update dependencies form an interpreted language, applications written in this language will run slower than those written in a compiled language. While update dependencies allow an explicit expression of the algorithms, it is necessary to enumerate implicit cases, such as tests for key attribute instantiation. Thus to ensure consistency, the update dependencies must be completely defined. However, the need to express such complex dependency operations merits the use of such a language.

IMPLEMENTATION STRATEGY. The strategy for the implementation of the model has been such, that the testing of the specifications of the relationships between MRP-II and CAD could be done early in the research.

Definition of the update dependency language has therefore been the first step in the work involved. As explained earlier this language allows for specification of operations in and between MRP II and CAD.

The second step of the implementation strategy has been to create an interpreter for the Update Dependencies language. the first version of the interpreter has been implemented in Prolog. Both the interpreter and the specifications of the functional relationship between the MRP II and CAD applications under one instance of the interpreter have been tested.

The third step is to integrate a remote procedure call facility into the interpreter. This will allow for the running of functional copies of the MRP and CAD system under separate instances of the interpreter on the same machine.

The fourth step is to move the two interpreters to different machines by generalizing the remote procedure call facility to allow calls over a network.

An important aspect of this implementation strategy is that it allows early testing of the specification of the functional relationship between the

MRP and the CAD system. Consequently, step three and four should not imply any changes in this specification, i.e. the distribution of information in the system should be transparent to the user.

While this implementation strategy does allow us to test the specification of the functional dependencies within each system and between the two systems, it does not provide an integration of two actual systems.

Though it is clearly desirable to integrate existing pieces of software rather than developing new systems from scratch, it is not the purpose of this research to provide a "bridge-box" that will allow users to hook up any two particular MRP II and CAD systems. What is provided is a functional description of how such "bridge-box" for two given systems could be specified. Update Dependencies can then be used to implement it.

To actually build a "bridge-box" for two given systems, one would proceed as follows. First, the set of operations available to the users in each system should be identified. These operations should continue to be available to the users and the user interface should, to the extent possible, be kept unchanged. Second, the software procedures that support these operations should be identified. Third, through calls to the interpreter, one would insert a set of update dependencies between the operations available to the users and the software procedures supporting them. These update dependencies would capture calls of operations made by the users and would issue calls of the software procedures supporting them, i.e. the calls of the software components would be implied operations in the update dependencies. This approach would allow the enforcement of consistency both within and between the two systems, by reusing the software components already available.

It is important to realize that the above approach keeps the user inter-

faces stable, reuses the software components that are already there, and avoids redesigning of existing databases, however, it is also important to realize, that a certain amount of additional programming cannot be avoided.

In addition to the work being done, parallel efforts are made to develop the rules as well as the model specifications for the integration of Computer aided Process Planning (CAPP) to the integrated system as it now stands. The CAPP functions will be integrated to the routings module of MRP-II, since data commonality between the 'process plan sheet' output of the former and production routings section of MRP, exists. The relevant information in the 'process plan sheet' part of the CAPP output and the MRP-II routings module are shown below. The elements of data commonality are clearly seen.

CAPP	MRP-II
Part Number	Part Number
Part Name	Part Name
Operation Number	Operation Number
Operation Description	Operation Description
Workcenter I.D.	Workcenter I.D.
Stndard time	Begin Date
Cycle Time	End Date
Set Up Time	Operation class Code
Manual Operating Time	Set Up Time
N.C. Running Time	Running Time

The entire scope of the model will now change as it has to encompass three systems rather than two. Issues to be addressed include the consolidation of the integration rules to an absolute minimum and the optimization of information flow between CAD, CAPP, and MRP-II.

CONCLUSION. The future of the manufacturing industry, especially in the United States, largely depends upon the successful development and implementation of

CIM systems. The result is increased efficiency in manufacturing which means well-controlled production, high quality, and competitiveness.

The work presented here, with MRP II as the nucleus of the integrated system and CAD as its first "satellite", is a wellposed approach through several implementation stages. The information flow between these two systems based on the commonality of part specification, product structure, and engineering changes, is proceeded through the "Multi-database interoperability" system associated with an AI production system. A set of logical rules has been modelled and transformed for the generation and maintenance of part master records and product structures initiated in the product engineering/design division of every typical manufacturing organization. These rules have been translated into update dependencies by using a form of a rule-based expert system, and is being tested for inconsistencies.

Although a "bridge-box" allowing user to hook up any two MRP II and CAD systems is not provided here, we do provide the knowledge needed to specify a "bridge-box" for two given systems, which can be used to integrate existing software packages. The future implementation stages include the extension of the model over two databases on the same computer, and later on two computers using remote operation calls.

Work is also continuing in the effort to integrate the second satellite, 'CAPP', into the integrated system, as it now stands. The rules for the integration are being developed, and implementation is in progress.

ACKNOWLEDGEMENTS The National Science Foundation is acknowledged for partial funding of this work (Grant No. DMC-85-04922), and the System Research Center of the University of Maryland for funding the database interoperability part of this research.

REFERENCES

(1) "MRP II Providing a Natural 'Hub' for Computer Integrated Manufacturing Systems," Kenneth A. Fox, Industrial Engineering,

vol. 16, no. 10, pp.44-50, October, 1984.

(2) "The CIMS Database: Goals, Problems, Case Studies, and Proposed Approaches Outlined," Michael Melkanoff, Industrial Engineering, vol. 16, no. 11, pp. 78-92, November, 1984.

(3) "An Integration of Manufacturing Resource Planning (MRP II) systems and Computer Aided Design (CAD) Based on Update Dependencies," G. Harhalakis, L. Mark, M. Bohse and B. Cochrane, accepted for the International conference on Data and Knowledge Systems for Manufacturing and Engineering, Hartford, Connecticut, October, 1987.

(4) "Integration of Manufacturing Resource Planning (MRP II) systems with Computer Aided Design (CAD)," G. Harhalakis, L. Mark, and M. Bohse, Proceedings of the 13th Conference on Production Research and Technology, organized by the National Science Foundation, Gainsville, FL, November, 1986. Research and Technology Program, 1986

(5) "Engineering Change for Made-To-Order Products: How an MRP II System Should Handle Them," G. Harhalakis, Engineering Management International , vol. 4, no. 1, pp. 19-36, October, 1986.

(6) "Operational Specification of Update Dependencies," Leo Mark and Roussopoulos Nick, Department of Computer Science, University of Maryland.