RSD-TR-19-86

Tool Management Systems Design For Flexible Manufacturing Systems (FMS)

by

Paul G. Ranky
Visiting Associate Professor

Department of Mechanical Engineering
and Applied Mechancics

October 1986

CENTER FOR RESEARCH ON INTEGRATED MANUFACTURING

Robot Systems Division

COLLEGE OF ENGINEERING

THE UNIVERSITY OF MICHIGAN

ANN ARBOR, MICHIGAN 48109-1109

TABLE OF CONTENTS

ABSTRACT

Considering the fact that Flexible Manufacturing Systems (FMS) should be able to accommodate a variety of different parts in random order, tool management at cell level and tool transportation, tool data management, tooling data collection, tool maintenance, and manual and/or robotized tool assembly. Tooling information in FMS is used by several subsystems, including: production planning, process control, part programming (CAM), tool preset and maintenance, robotized and/or manual tool assembly, stock control and materials storage.

The paper *summarizes the major tasks to be solved* when designing tool management systems for FMS, as well as *(1.) gives a solution for describing the data structure of a tool data base integrated with a generic tool description method and (2.) shows a sample transaction of the way the FMS real-time control system can access and use this data base.*

1. Introduction

The design of a tool management system incorporates a vast amount of analysis and system development work. It must be done by a team of process and production engineers as well as data processing staff, headed by an experienced team leader who understands not only the data management problems, but also the operation control aspects of FMS systems.

When designing the FMS tool management system, one should consider the following steps:

1. Collect all current and possible future user and system requirements
2. Analyze the system (i.e. the data processing and the FMS hardware and software constraints)
3. Design an appropriate data structure and data base for describing tools and then
4. Specify, and design programs that are capable of accessing this data base as well as communicating with the real-time production planning and control system of the FMS.

2. User and System Requirements

Probably the most important question to be answered before starting to design an FMS tool management system and a data base is *"Who"* is going to use the data, *"When"* and *"For what"* purposes in the particular system?

Tooling data in FMS, typically going to be used by several subsystems, as well as human beings. These are as follows:

- The production planning subsystem
- Process control
- Part programming
- Tool preset and tool maintenance
- Tool assembly (manual or robotized)
- Stock control and materials storage

For example the *production planning system* has to be informed in real-time about the availability of tools in stock, as well as about the current contents of the tool magazines of the machine tools (in the case of Flexible Assembly Systems, the robot hands in the End-of-Arm-Tool magazines) otherwise it will not be able to generate a proper scheduling program.

It must be noted that the real-time aspect is important because tools are changed in the magazines of cells not only because they wear, but also because different part programs may need different sets of tools. (The actual tool changing operation is done in most cases by robots. The tool magazine loading/unloading procedure is performed mostly by human operators, but sometimes by robots, as in some Cross and Trecker systems, for example.)

Both the *process control* and the *production planning systems* have to update any changes and act in real-time otherwise the operation of the system can be disrupted (**Figure 1**).

The *tool preset station* must be able to inform the process control system about tool preset and offset data, preferably via a digital tool preset unit linked directly to the data processing network of the FMS. (Note that the final adjustments, i.e. the x, y, z tool correction: length and diameter values, are often done at the cell prior to the first operation using that tool, as well as between operations using the real-time operated tool check and point station at the machine in order to check tool adjustments and tool wear.

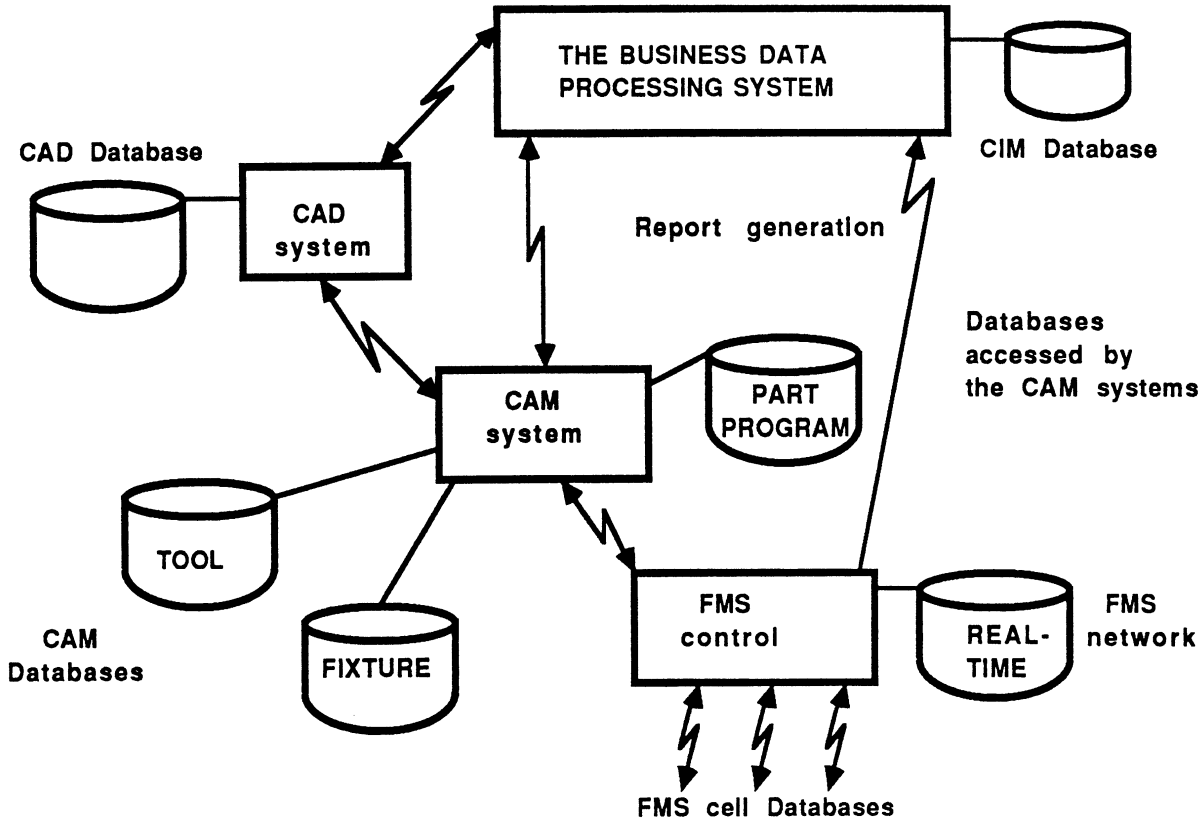

Figure 1a The relationship between subsystems interacting with each other as well as with their human users in the FMS data processing network.

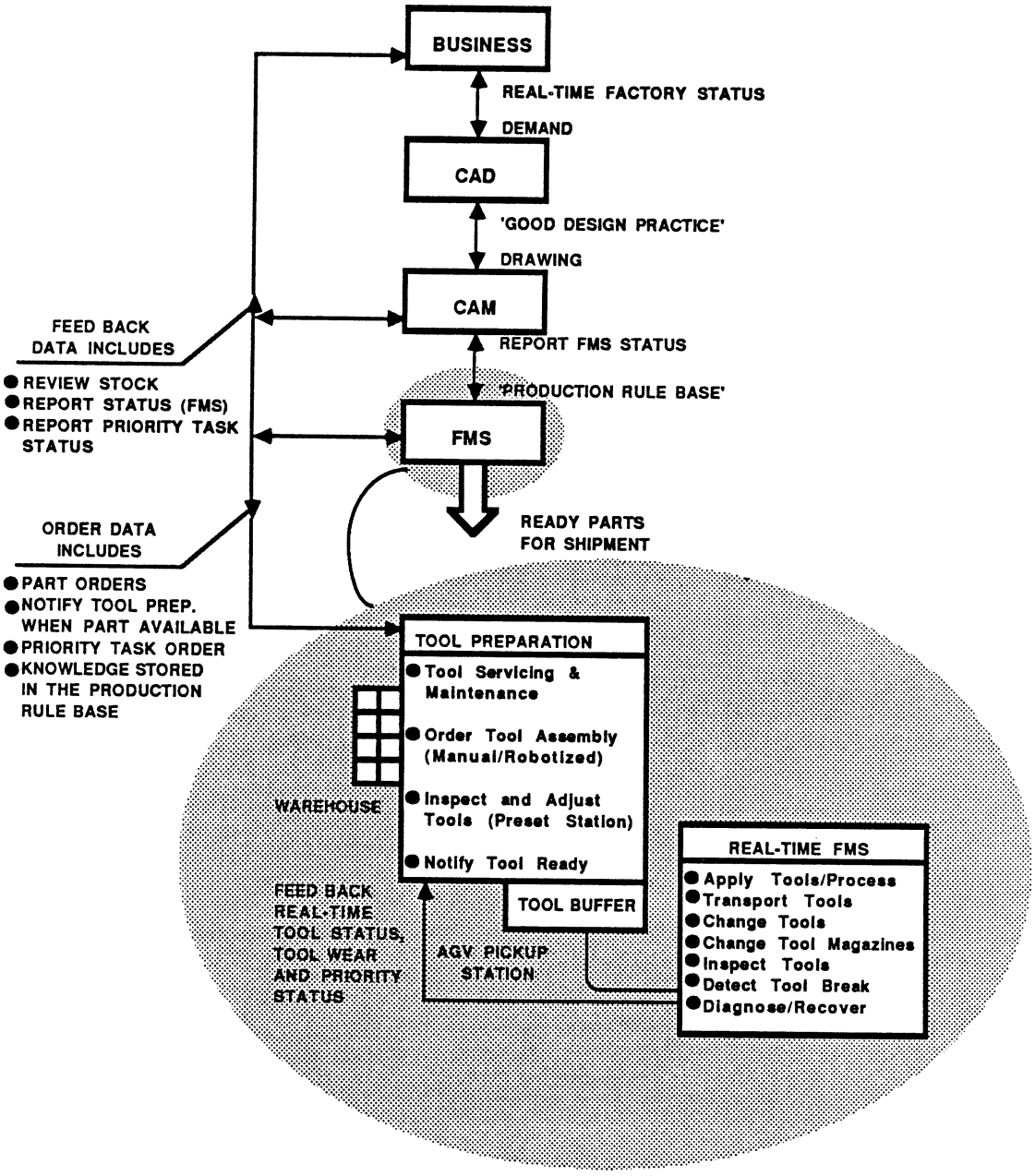

Figure 1b The links between the Tool Preparation subsystem and the rest of the CIM environment.

Finally, when writing *FMS part programs,* one has to know the actual sizes of tools and their characteristics and behavior in different cutting conditions. The tool geometry data is used when checking for tool collision by graphics simulation. (**Figures 2,3 and 4** illustrate a graphics tool path simulation using the McAuto Unigraphics System).

3. System Constraints

If data types are kept separately and accessed by independent programs which generate and access independent files, one program will "not a know" in time when another program updates a file and eventually *panic situation will occur.* because out of date data is is going to be used by one of the FMS control programs.

Data Base Management Systems are considered to be the essential core of the FMS tool management system since they:

- Provide logical as well as physical data independence. (*Logical data independence* means that new fields of records may be added to the system without rewriting the application programs. *Physical data independence* means that changes can be made to some element of data on disk, or on any type of data storage media, leaving the application programs untouched.)
- Ensure a standard software interface for the database users and provide fast information retrieval

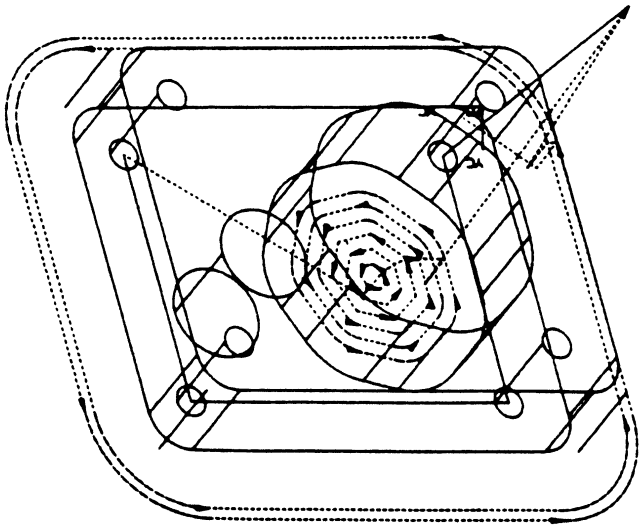

Figure 2 Three dimensional tool path simulation using an automatically generated by a UNIAPT program in the CAM system on the basis of a design transferred from the CAD system [2].

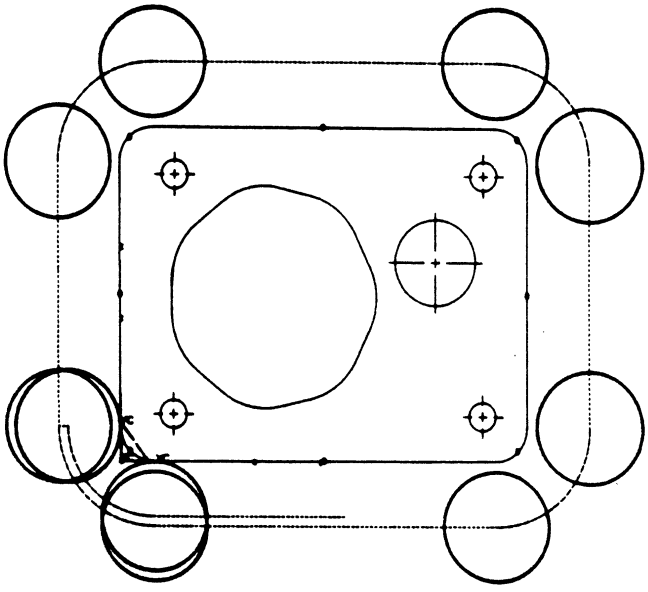

Figure 3 The automatically generated three dimensional tool path indicating the outside contouring, the drilling and the inside contouring operations. The shown simulations help to delay and test part programs before sending them to the FMS network [2].

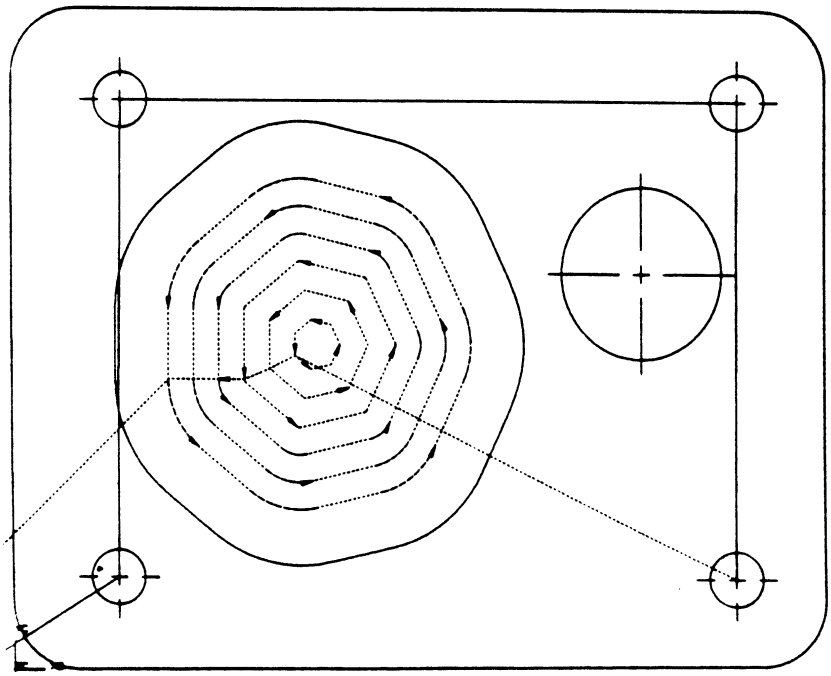

Figure 4 Top view of the generated tool path [2].

- Ensure that data is compatible for all subsystems, reducing data access time and application program development and maintenance costs
- Minimize data redundancy, but do not eliminate data redundancy because of security and reliability reasons.

To summarize, Data Base Management Systems enable the data base to be interfaced with several different application programs written in different languages, running in different processors of the FMS network and operating systems.

To provide the required flexibility and high level of local intelligence for the tool management system, as well as for the other subsystems of the FMS, it is necessary to apply distributed processing theory both for communications, as well as for Data Base Management purposes.

The most important aspects of distributed processing and data management from this point of view are:

- The possibility of real-time communication and data update in the tool store, at the tool assembly station, at the machines or cells using the tools and in general between all subsystems accessing this facility within the FMS data processing network.
- Flexible and user friendly operator interface at all terminals where the tool management system's users must access the distributed system
- "Well designed hardware and software architecture", preferably based on intelligent nodes linked together by a Local Area Network (LAN)

When following these principles, the man-machine and the machine-machine communication systems will be more flexible and compared to "non-distributed systems" the system to be created will be more reliable too.

4. Tool Management System Data Structure Design

This section demonstrates the concept of a structured tool description method, developed by the author, using a Relational Data Base Management System. The method is *"generic"* thus can be used and/or adapted relatively easily in more complex applications too.

Analysis of the *tooling data flow* within any FMS shows that the FMS part programmer and/or the part program generator needs both geometric and tool

material data to be able to select the most appropriate tools for the required operations. Because an average sized FMS with five to eight machines linked to DNC (Distributed Numerical Control) needs thousand or more tools, depending on the number and the variety of components it has to machine, a tool data base must be implemented to serve the part programmers as well as the real-time FMS system.

The proposed general-purpose *"generic"* FMS tool description method provides a four level structure (i.e. "tooltype," "element and edge," "subassembly," and "onetool"), each level of which represents a tool data collection. The levels are logically linked together (i.e. are logically "assembled") via the data structure and the software tool of the data base. (**Figure 5**).

The levels of this structure are as follows:

1. *Real-time data called ONETOOL,* representing the real-time data collection of tools, including the actual code number of the particular tool, the actual value of the monitored tool wear, the measured and/or adjusted tool diameter and length (if it is possible to adjust the tool) and finally the tool length and/or diameter compensation data (**Figure 5i**). Since no human being is capable of keeping track of thousands of tool data in real-time, this data collection is of extreme importance for the accurate and safe operation of the FMS. (To summarize each of these files describes a single tool as available for dispatching into the FMS and/or in the margarine of a machine.)

2. *Each tool is linked to one of a number of available tool type files, named TOOLTYPE 1...n.* TOOLTYPE files do not necessarily represent physically existing tools on stock or in the magazines of the machines. They are tools that could be assembled from subassemblies, elements and tool tips (edges) each of them described in different data sets, or files (**Figure 5c-5e**). The TOOLTYPE file logically links together the tool as an assembly of different tool geometry types, elements, sub-assemblies and tool edges, each of them described in the TOOLTYPE1...n, in the ELEMENT, in the SUBASSEMBLY and in the EDGE1...n files. Providing the user with several records as building blocks, different tools can be created logically, simply by using standard data base transaction

3. *Subassemblies,* represent units of the tool, that must be handled together, because this is the way they are purchased and/or assembled. This data set is stored in a file called SUBASSEMBLY.

4. Finally, *elements* enable this data base to be used for stock control and automated order processing. This data set is stored in a file called ELEMENT. A distinguished set of elements are called EDGE1...n, and they describe different inserts and the tool tip geometry actually involved in the cutting process (**Figure 5g-5h**).

This concept of handling data sets is similar to a LEGO kit. It allows total flexibility for each implementation, while providing a "generic" data structure (**Figure 5i**).

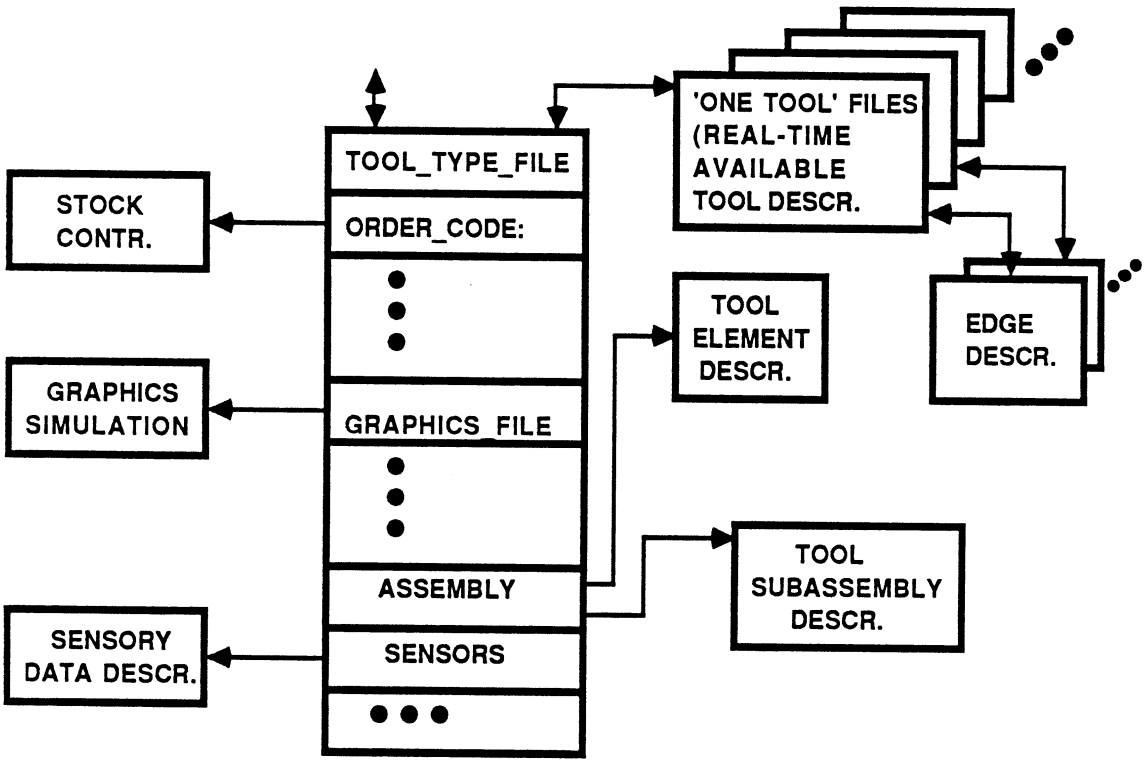

Figure 5a The data structure and some sample files of the FMS tool data base.

ORDERCODE	ALPHA, MAX	20 CHARACTERS	PRIMARY KEY
NAME	ALPHA, MAX	20 CHARACTERS	SECONDARY KEY
SUPPLIER	ALPHA, MAX	40 CHARACTERS	SECONDARY KEY
FILEUPDATE	DATE		SECONDARY KEY
RECEIVED	LONGMATH		SECONDARY KEY
SALES	LONGMATH		SECONDARY KEY
STOCK	LONGMATH		SECONDARY KEY
ONHAND	LONGMATH		SECONDARY KEY
ORDER	LONGMATH		SECONDARY KEY
MUSTORDER	BOOLEAN		DATA FIELD
COST	LONGMATH		DATA FIELD
ORDERCOST	LONGMATH		DATA FIELD
COMMENT	ALPHA, MAX	60 CHARACTERS	DATA FIELD

Figure 5b Typical customer order processing file [1].

1)	TOOLIDENT	ALPHA, MAX	20 CHARACTERS	PRIMARY KEY
2)	ALPHANAME	ALPHA, MAX	20 CHARACTERS	SECONDARY KEY
3)	DIANOMINAL	LONGMATH		SECONDARY KEY
4)	LNOMINAL	LONGMATH		SECONDARY KEY
5)	LWORKMAX	LONGMATH		SECONDARY KEY
6)	RCDEPTH	LONGMATH		SECONDARY KEY
7)	TCDEPTH	LONGMATH		SECONDARY KEY
8)	KAPPA	LONGMATH		SECONDARY KEY
9)	ZFACE	LONGMATH		SECONDARY KEY
10)	REVDIR	BOOLEAN		DATA FIELD

Figure 5c Typical tool type geometry description file (see also Figure 5d) where the fields describe the:

(1) Tool identifier

(2) Alphanumeric name of the tool

(3) Nominal tool diameter in mm

(4) Nominal tool length in mm

(5) Maximum working length of the tool

(6) Maximum useful cutting depth of tool tip as measured radially from the cutter body centre line

(7) Same as above but measured tangentially (Note, both fields 6 and 7 are sometimes required)

(8) Major cutting edge angle as measured on the assembled and adjusted tool (Note, this data type is very important for the FMS part programmer)

(9) Number of teeth measured on face

(10) Direction of revolution (0-right, 1-left)

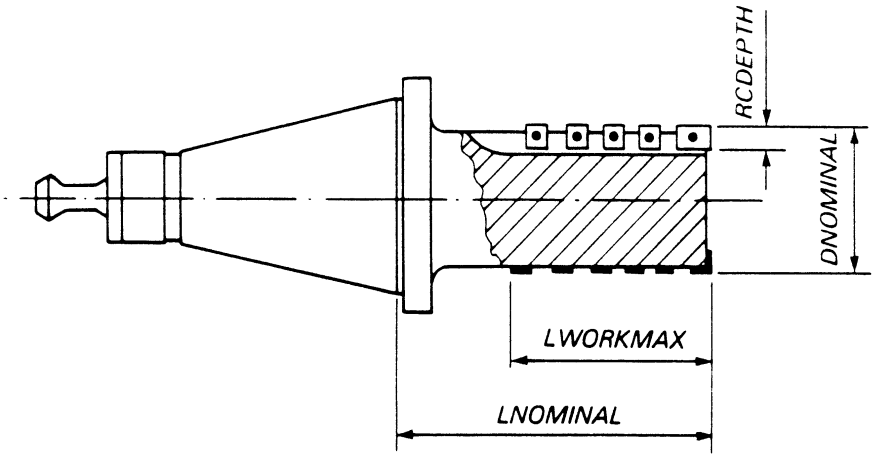

Figure 5d The drawing of a typical fixed diameter tool as described in Figure 5c [1]

TOOLIDENT	ALPHA, MAX	20 CHARACTERS	PRIMARY KEY
ALPHANAME	ALPHA, MAX	20 CHARACTERS	SECONDARY KEY
DIAMINIMUM	LONGMATH		SECONDARY KEY
DIAMAXIMUM	LONGMATH		SECONDARY KEY
LMINIMUM	LONGMATH		SECONDARY KEY
LMAXIMUM	LONGMATH		SECONDARY KEY
LWORKMAX	LONGMATH		SECONDMARY KEY
RCDEPTH	LONGMATH		SECONDARY KEY
TCDEPTH	LONGMATH		SECONDARY KEY
KAPPA	LONGMATH		SECONDARY KEY
REVDIR	BOOLEAN		DATA FIELD

Figure 5e Typical description of an adjustable tool as shown in Figure 5g [1]

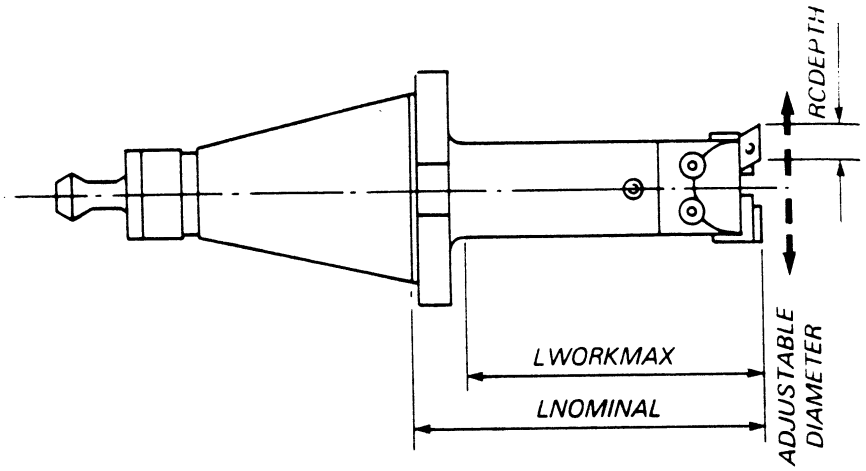

Figure 5f A typical adjustable tool

1)	EDGEIDENT	ALPHA, MAX	20 CHARACTERS
2)	MANUFACT	ALPHA, MAX	20 CHARACTERS
3)	SHAPE	ALPHA, MAX	1 CHARACTERS
4)	CLEARANCEA	ALPHA, MAX	1 CHARACTERS
5)	TOLERANCES	ALPHA, MAX	1 CHARACTERS
6)	CHIPBRCODE	ALPHA, MAX	1 CHARACTERS
7)	C_EDGE_L	ALPHA, MAX	1 CHARACTERS
8)	THICKNESS	ALPHA, MAX	1 CHARACTERS
9)	PARCLANGLE	ALPHA, MAX	2 CHARACTERS
10)	FREECODE	ALPHA, MAX	1 CHARACTERS
11)	FEED_DIR	ALPHA, MAX	1 CHARACTERS

Figure 5g The standard (ISO) changeable insert description shown here as one of the tool EDGE description files of the tool data base. The file contains the following data types:

1. Identifier
2. Manufacturer
3. ISO insert shape code
4. Main cutting edge clearance angle code (ISO)
5. Tolerance code(ISO)
6. Chipbreaker and clamp type ISO code
7. Cutting edge length ISO code
8. Thickness ISO code
9. Parallel land, clearance
10. Free char field (not used yet)
11. Feed direction code

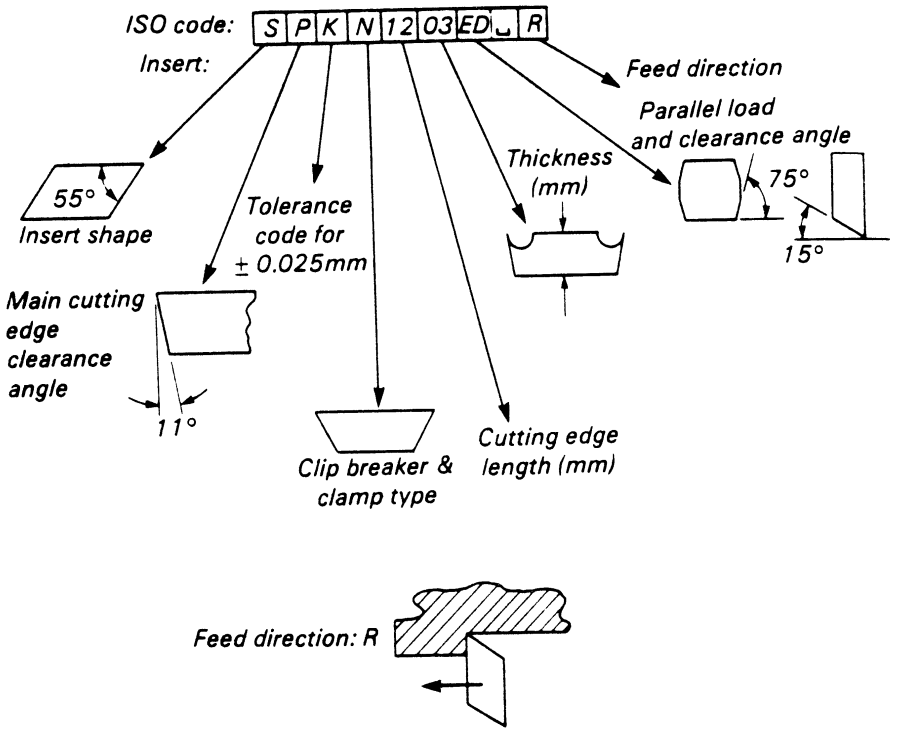

Figure 5h Example of an ISO insert code description

Following the above outlined principles and the structured tool description method new tools can be described and added to the system and necessary changed can be made as the system grows up-to the physical limits of the particular data processing hardware and software without any complications.

5. Real-Time Application Case Study

This case study represents an application possibility relating to the FMS tool management system and data base outlined in the previous sections of the paper.

Let us assume, that we have an FMS consisting of a number of machines, or cells, each of them capable of handling one tool magazine at a time, containing several tools (typically 40 to 120 tools). The contents of the tool magazines can be changed, of course, and they are interchangeable between cells. The question we would like to ask from our FMS tool management system is simple and practical: *Which machine(s), or cell(s) have the appropriate tool mix in their tool magazines to process a given part program (i.e. NC program)?*

This question could be asked by an FMS production manager, or by the FMS scheduling program itself, before finalizing the sequence in which the certain part, or parts will be processed in the system. Since tool loading into tool magazines, as well as changing entire tool magazines takes time and costs money, the above problem is important to be solved.

In this example we shall interact with the FMS machine, or cell description files, thus let us explain first the MACHINE table in our relational data base.

TOOL GEOMETRY (TOOLTYPE1 FILE)		
TOOL IDENTIFIER (TOOLIDENT)	=	3459841
ALPHANUMERIC NAME (ALPHANAME	=	SIDE AND FACEMILL
NOMINAL DIAMETER (DIANOMINAL)	=	50.00 MM
NOMINAL LENGTH (LNOMINAL)	=	125.00 MM
MAX. WORKING LENGTH (LWORKMAX)	=	75.00 MM
CUTTING DEPTH/RADIAL (RCDEPTH)	=	8.00 MM
KAPPA ANGLE (KAPPA)	=	
NUMBER OF TEETH ON FACE (ZFACE)	=	6.00
DIRECTION OF REVOLUTION (REVDIR)	=	0

EDGE DESCRIPTION (EDGE1 FILE)		
TOOL EDGE IDENTIFIER (EDGEIDENT)	=	4301
MANUFACTURER (MANUFACT)	=	COMPANY X
ORDERCODE (ORDERCODE)	=	SW-892:2R8WT523
ISO MATERIAL CODE (MATERCODE)	=	K20
ISO FORMCODE (FORMCODE)	=	S
CLEARANCE ANGLE CODE (ALPHACODE)	=	P
INSERT SIZE TOLERANCE (TOLERANCEC)	=	L
CHIPBREAKER ISO CODE (CHIPBRCODE)	=	X
INSERT SIZE ISO CODE (INSERTSIZE)	=	12
INSERT THIKNESS CODE (THICKNESSC)	=	04
TIPRADIUS ISO CODE (TIPRADIUSC)	=	AD
COATING CODE (COATINGC)	=	

REAL-TIME DATA (ONETOOL FILE)		
TOOL IDENTIFIER (ONETIDENT)	=	T03
CORRECTION X (XCORR)	=	0
CORRECTION Y (YCORR)	=	0
CORRECTION Z (ZCORR)	=	120.00
TIP RADIUS CORRECTION (TIPRADCORR)	=	0
MEASURED WORKING LENGTH (LWORK)	=	75.20
MEASURED WORKING DIAMETER (DIAWORK)	=	49.98
TOOL LIFE IN PERCENTAGE (TOOLLIFE)	=	80

Figure 5i Sample data of a tool as stored in the tool database

(**Figure 6**).

This table consists of the following fields:

1. MACHINE_ID, a 16 character long alphanumeric name, containing the machine identifier,

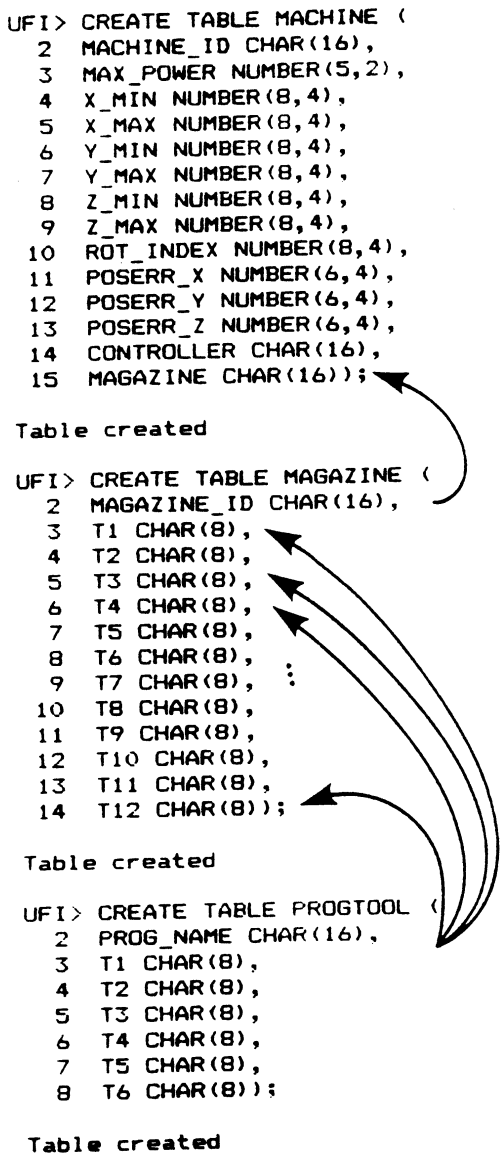

Figure 6 **The data structure of the "structure", the tool "magazines" and the programmed tools, "progtool" files (as retrieved from the variable route FMS job control file (see in Figure 7). The arrows represent a "one-to-many-to one" relationship in this data structure [2].**

2. MAX_POWER, maximum power of the machine in KWatts, stored as a real number,
3. X_MIN to Z_MAX describe the motion range limits of the machine's table and main spindle,
4. ROT_INDEX stores the size of increment the machine's table is capable of rotating
5. POSERRX_Y...Z contains the positioning error regarding to the relevant axis. (This is important information when automatically selecting equipment for certain operations).
6. CONTROLLER identifies the type of CNC control is used at the particular cell. This field points to a larger data set, giving more information about the controller. This data set is useful for the DNC system, as well as for part programming, when the appropriate post-processor and programming strategies must be selected for the cell.
7. Finally, MAGAZINE stores the tool magazine identifier, currently being utilized on the cell.

Note, that because of the nature of Relational Data Base Management Systems, this table does not define, or rigidly fix any further relationships between its data, thus any field can be related to any field of any other table by means of a query, created interactively, or batch by another program.

The other two tables we set up demonstrate the information we store in the tool magazines, and the tools in PROGTOOL.

```
BEGIN FMS_Part_program/Code:ABCO8

    BEGIN Pallet_program/Code:No. 1, Pallet-code:P1;
              Execute_operation/Code:No. 1, Tool_file:P1No. 1T01;
              Execute_operation/Code:No. 2, Tool_file:P1No. 2T02;

              If <Condition true> THEN
                                        BEGIN
                  Execute_operation/Code:No.3, Tool-file:P1No.3T03;
                  Execute_operation/Code:No.4, Tool-file:P1No.4T04;
                                        END (* PALLET PROGRAM END *)
                                     ELSE
                                        BEGIN
                  Execute_operation/Code:No. A3, Tool_file:P1No.A3T13;
                  Execute_operation/Code:No. A4, Tool_file:P1No.A4T14;

                     IF <Condition true> THEN
                     Execute_operation/Code:No.A5, Tool_file:P1No.A6T26;
                                           ELSE
                                              BEGIN
                           Execute_operation/Code:No.A6, Tool_file:P1No.A6T16;
                           Execute_operation/Code:No.A7, Tool_file:P1No.A7T17;
                                              END; (* 2nd ELSE *)
                                        END; (* 1st ELSE *)
    END; (* PALLET PROGRAM No.1 *)

    BEGIN Pallet_program/Code:No.2,Pallet_code:P12;
         (* PALLET PROGRAM No.2 DATA STRUCTURE DESCRIPTION *)
    END; (* PALLET PROGRAM No.2 *)

    BEGIN Pallet_program/Code:No. 3,Pallet_code:P33;
         (* PALLET PROGRAM No. 3 DATA STRUCTURE DESCRIPTION *)
    END; (* PALLET PROGRAM No.3 *)

         (* FURTHER PALLET PROGRAMS CAN BE ADDED HERE... *)

END. (* FMS PART PROGRAM ABCO8 *)
```

Figure 7 **The required list of tools is obtained from the "variable route" FMS job control file (or production rule base). This file is generated by the CAM system, after having accessed the tool database several times when selecting tools for certain operation. (This production rule base describes what the FMS should do with the part, including all possible combinations of manufacturing and the required resources). [2].**

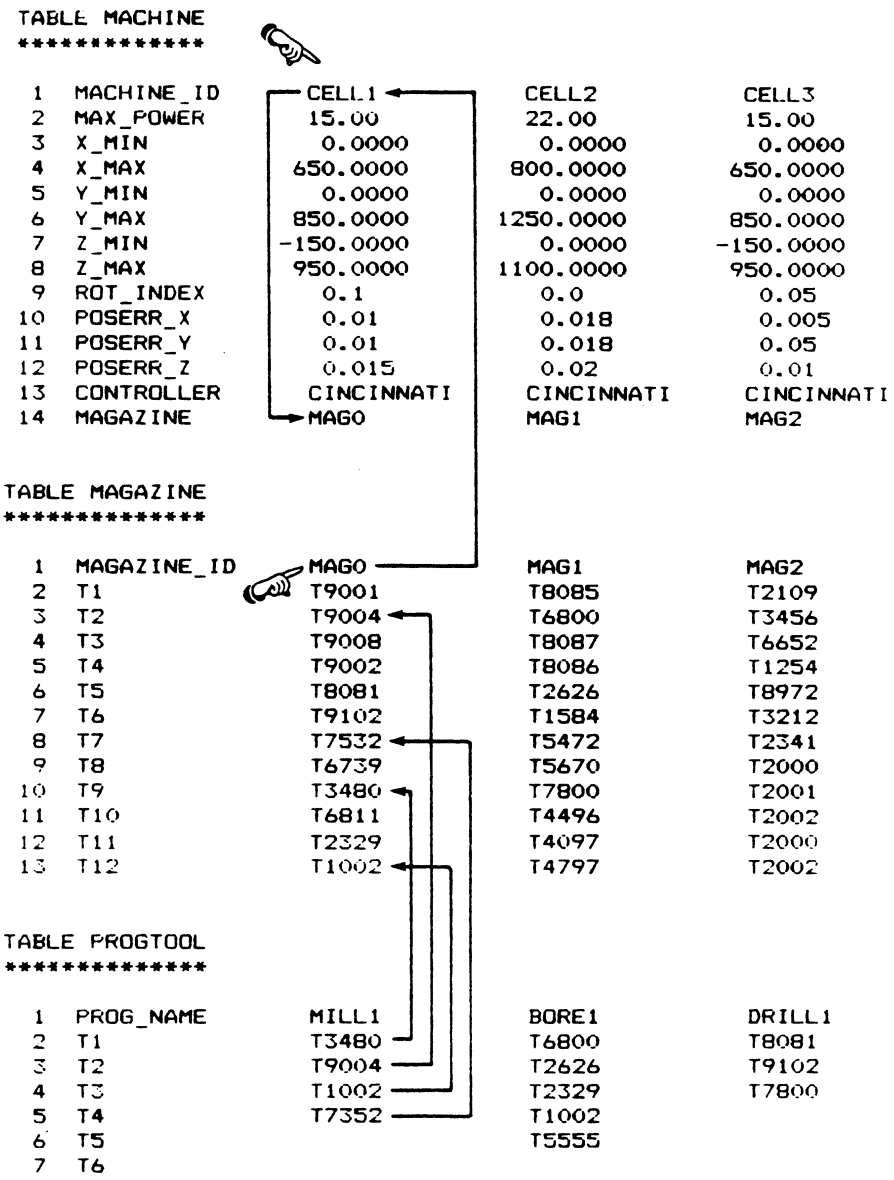

Figure 8 **Sample data loaded into the data structure containing machine, tool magazine contents and tool requirement data, extracted from the FMS part program.**

```
SELECT MACHINE_ID FROM MACHINE WHERE MAGAZINE=(
SELECT MAGAZINE_ID FROM MAGAZINE WHERE
T1  =  (SELECT T1 FROM PROGTOOL WHERE PROGNAME-'MILL1') OR
T2  =  (SELECT T1 FROM PROGTOOL WHERE PROGNAME-'MILL1') OR
T3  =  (SELECT T1 FROM PROGTOOL WHERE PROGNAME-'MILL1') OR
T4  =  (SELECT T1 FROM PROGTOOL WHERE PROGNAME-'MILL1') OR
T5  =  (SELECT T1 FROM PROGTOOL WHERE PROGNAME-'MILL1') OR
T6  =  (SELECT T1 FROM PROGTOOL WHERE PROGNAME-'MILL1') OR
T7  =  (SELECT T1 FROM PROGTOOL WHERE PROGNAME-'MILL1') OR
T8  =  (SELECT T1 FROM PROGTOOL WHERE PROGNAME-'MILL1') OR
T9  =  (SELECT T1 FROM PROGTOOL WHERE PROGNAME-'MILL1') OR
T10 =  (SELECT T1 FROM PROGTOOL WHERE PROGNAME-'MILL1') OR
T11 =  (SELECT T1 FROM PROGTOOL WHERE PROGNAME-'MILL1') OR
T12 =  (SELECT T1 FROM PROGTOOL WHERE PROGNAME-'MILL1'));
```

Figure 9 The query routine for searching the database for the appropriate tools in the magarines of the machines in the FMS.

The MAGAZINE file identifies 12 locations for simplicity, in practice there are usually 40 to several hundred tools in a magazine, particularly if one uses the modules, block tooling system from Sandrite, Hertel, or other manufacturers.

The tools are listed in the PROGTOOL file. It contains tool codes used in different part programs we need to machine the part on a particular cell. These tool codes can be gained from the production rule base, describing different set-ups, the fixturing and tooling requirements and the precedence rules of the

operations required. It also describes the alternatives of different operations. This production rule base, shown in **Figure 7** is the output of the CAM system and is issued for each component to be manufactured on the FMS.

Now that we have created all required tables, let us demonstrate a real-time query.

1. First we need to find the matching magazine to any inquiry given in our PROGTOOL table,
2. Then we must find the appropriate cell containing this magazine.

At this point let us explain something more about the ORACLE Data Base Management System as used in this case study. It offers a non-procedural, "English-like" query language, called SQL. (This language interface is also used by programmers when writing query programs in high level languages. It can be used to setup tables interactively, edit them, delete them, etc.).

We demonstrate the way it can be used for searching data. In our application we have used interactive searching, because it can be followed easier by readers of this paper.

To select the appropriate magazine we need to input some data (see **Figures 6 and 8**) and describe the "one-to-many" type query, using the SQL language. The result of this search is "MAGO" in this example, as we have expected. (**Figure 8**).

Now we must find the cell, this magazine is attached to. The query routine for this transaction is listed if **Figure 9**. (Note that this is a FOR cycle like

search, very similar to the previous one.)

The result of this search is CELL1, since magazine MAGO is mounted on CELL1. (See **Figure 8**).

6. Summary

Tool management is an important part of the FMS control software. Without understanding all important needs, such as FMS contro, NC past programming, tool preparation, etc., as well as the particular FMS hardware/software constraints one cannot design a usable tool management system. The paper described a generic method that can be implemented with minor modifications in a large variety of different FMSs.

7. References

[1] *Paul G. Ranky*: The Design and Operation of FMS, Flexible Manufacturing Systems), IFS Publications (Ltd) and North-Holland, 1983. 348 p.

[2] *Paul G. Ranky*: Computer Integrated Manufacturing, An Introduction with case studies, Prentice Hall International, 1985. 528p

[3] *M. Albert*: Unmanned turning: what is needed?, Modern Machine Shop, Vol. 55, No. 11, pp. 62-68. (1982)

[4] *T.R. Crossley*: Towards the Automated Factory, Proc. of PEMEC 79, Birmingham, UK.

[5] *S. Inaba*: Experience and effect of FMS in machine factory, IEEE Control Systems Magazine, Vol. 2, No. 2, pp. 3-9 (1982)

UNIVERSITY OF MICHIGAN
3 9015 03695 1856

[6] *Paul G. Ranky.* Tool Management Tasks in FMS, NAMRC XIV., North American Manufacturing Research Conference, Minnesota, 1986.