Journal of Intelligent Manufacturing, 16, 287–301, 2005

A STEP-based manufacturing information system to share flexible manufacturing resources data

OMAR LÓPEZ-ORTEGA* and MORAMAY RAMÍREZ

Centro de Investigación en Tecnologías de Información y Sistemas, Instituto de Ciencias Básicas e Ingeniería, Universidad Autónoma del Estado de Hidalgo, Carretera Pachuca – Tulancingo Km. 4.5, Pachuca, Hidalgo, México
E-mail: olopez27@prodigy.net.mx

Received August 2003 and accepted August 2004

The paper describes the design and implementation of a Manufacturing Information System as a solution to share and exchange manufacturing data. The Manufacturing Information System possesses relevant features for data sharing and exchanging, that are compliant to the standard ISO 10303, known as STEP[1]. It is known that manufacturing planning and execution activities require data from machine-tools, robots, ASRSs[2], AGVs[3] and, on the other hand, computer-based applications (CAM[4], CAPP[5] or PAC[6]) use proprietary formats to store flexible manufacturing resources data. The way the proprietary formats are built has lead to incompatibilities and interoperability problems among the applications. It is argued that by solving these problems a qualitative functional synergy will result during manufacturing planning and execution activities. The suggested solution is using a STEP-based information system, a server and a prototype client application. These three programs are implemented in Java classes to facilitate data sharing and exchanging of flexible manufacturing resources.

Keywords: STEP standard, EXPRESS models, manufacturing information system, flexible manufacturing resources, data sharing and exchanging

1. Introduction

Manufacturing integration requires the achievement of complete interoperability among CAX applications[7] that are used in manufacturing planning and execution activities. Authors like Valdew Singh (1997) argue that information islands exist within manufacturing systems since CAX applications are not fully capable of sharing data. The current incompatibi1ity problems among CAX applications can be solved through several approaches, however, peer-to-peer format translation and an open information system seem to be preferred as last date generation solutions. Many format translators must be built so that CAX applications can share and exchange data; this approach is very expensive and not entirely functional. On the other hand CAX applications may benefit from a Manufacturing Information System (MIS) to make the manufacturing activities less decoupled. However, in order to build such a MIS which CAX applications can access, it is required a robust design of a data model and an appropriate implementation to support the requirements of such applications.

The present paper contains the description of the design and implementation of a MIS that is

*Author for correspondence.

used to store, share and exchange data of flexible manufacturing resources i.e. CNC[8] machine-tools, robots, ASRSs or AGVs (Table 1). Since manufacturing systems heavily depend on CAX applications that extensively employ data of the mentioned resources, it is compulsory to have a MIS that serves as a facilitator for data sharing and exchanging. As far as we know there is not a general data model to represent flexible manufacturing resources, therefore this particular information system is lacking in nowadays manufacturing systems. The importance of having this MIS architecture following principles of the STEP standard is explained as a necessary condition in this subject area.

The authors elaborate on the relevance of the proposed data model for flexible manufacturing resources, and the resultant MIS. EXPRESS was chosen to model flexible manufacturing resources because this modeling language brings coherence regarding the ISO[9] initiatives related to manufacturing data modeling. The proposed model was expanded and then implemented, in this way the MIS acts as a common database for specific instances of flexible manufacturing resources to be stored in. Moreover, to facilitate access in a distributed environment, along with the MIS a client–server architecture was programmed to attend information requests. The Java language was used to implement the MIS, the server and a client application. This approach is found to be a sound solution to improve manufacturing integration.

2. Importance of a step-based manufacturing information system

A manufacturing information system, containing data from the machine-tools, robots, etc., and accessible to CAX applications, is a compelling solution to enhance interoperability because, as it was stated lines above, such applications present problems to share and exchange manufacturing data, particularly of flexible manufacturing resources (Table 2). These problems can be visualized with two different scenarios.

Let us consider a situation when data exchange among several applications are important. When a CAPP application creates a process plan it ideally must match the machining operations with an appropriate machine-tool. To do so the capabilities and capacity of the machine-tools in the shop floor must be identified so that the process plan can be created. Next, to elaborate the production schedule, a PAC application has to know the status of the selected machine-tools before sending them the processing instruction to perform. In this sense communication must be established, via a messaging application, with the machine-tools by using the local area network. As can be seen several applications need to share data about processing resources. However, *data sharing* is not fluently done due to the different syntax and semantics for representing flexible manufacturing resources.

Current incompatibilities also inhibit *Data exchanging*. To exemplify consider the semantics

Table 1. Flexible manufacturing resources

Flexible manufacturing resource	*Acronym*	*Description*
Automated guided vehicle	AGV	Battery powered, automatically steered vehicles that follow defined pathways in the floor. They are used to move unit loads between load and unload stations.[16]
Automatic storage and retrieval system	ASRS	A storage system that performs storage and retrieval operations with speed and accuracy under a defined degree of automation.[17]
Computer numerical control	CNC	Numerical control machine-tools whose operation is based on a dedicated computer.[18]
Robot	None	General-purpose, programmable machine possessing certain anthropomorphic characteristics, the most obvious of them is the mechanical arm.[19]

Table 2. Computer-based applications

Computer-based application	*Acronym*	*Usage*
Computer-aided design	CAD	Part and product design Tools and Fixtures definition (Shen, 2001).
Computer-aided process planning	CAPP	Creation of process plans (Shen, 2001).
Computer-aided manufacturing	CAM	Production programming Production planning Machining control Assembly control (Shen, 2001).
Production activity control	PAC	To control and evaluate production activities. It interfaces with resources like AGV, ASRS, CNC machine-tools, and with applications like process planning. One of its roles is to download process instructions generated by CAPP applications and transfer them to processing resources in the shop floor (Browne *et al*., 1992).

and syntax used to represent a *Robot*. The *arm* of a *Robot* is semantically and syntactically represented in two different ways in CATIA and KISMET[10]. CATIA calls it *solid* while KISMET calls it *mechanism* (SPRIT, 1994; Fahim and Choi, 1998). With these two applications it is possible to store all the relevant information about a *Robot arm* i.e. its length or mass. However, if it is wanted to exchange information between them a translator must be used, taking the risk of losing important data.

To eliminate barriers and help to build a bridge among the "information islands" it is necessary a robust and general data model to represent the flexible manufacturing resources upon which the manufacturing information system is implemented. As the STEP standard is the current state-of the art solution to facilitate data sharing and exchanging we consider essential to respect its basic principles. The MIS for flexible manufacturing resources (see Fig. 1), which is the main contribution of the paper, possesses the following features:

(i) It has been modeled with the EXPRESS language. We chose the EXPRESS modeling language because it is a formal language for specification of technical data, it has a clear object-oriented foundation, it allows data inheritance, it permits the creation of user-defined types and the definition of complex entities. The STEP-based information system is created once the EXPRESS data model has been compiled.

(ii) Data is stored in a neutral format. The STEP standard imposes a neutral storage format, readable and understandable for every computer-based application regardless the system that actually creates the technical data. This format is described in the part 21 of STEP (ISO, 1998a).

(iii) Access and manipulation is done through normalized functions. To access and manipulate the resultant information system a specific set of functions, called Standard Data Access Interface (SDAI) functions, are used. The full, abstract definition of the SDAI functions is found in the part 22 of STEP (ISO, 1998b). Also, specific descriptions for implementing them in general purpose languages are also covered in STEP: Part 24 describe their implementation for the C++ language (ISO, 1998c), and part 27 for Java (ISO, 1998d).

Interested readers in knowing more details about the neutral format, the EXPRESS language and the SDAI functions, may consult the article of Guy Pierra (Pierra, 2000).

3. Manufacturing data models

The significant work concerning data sharing is being done by the International Standards Organization which has launched a number of standardized data models. Data models for flexible

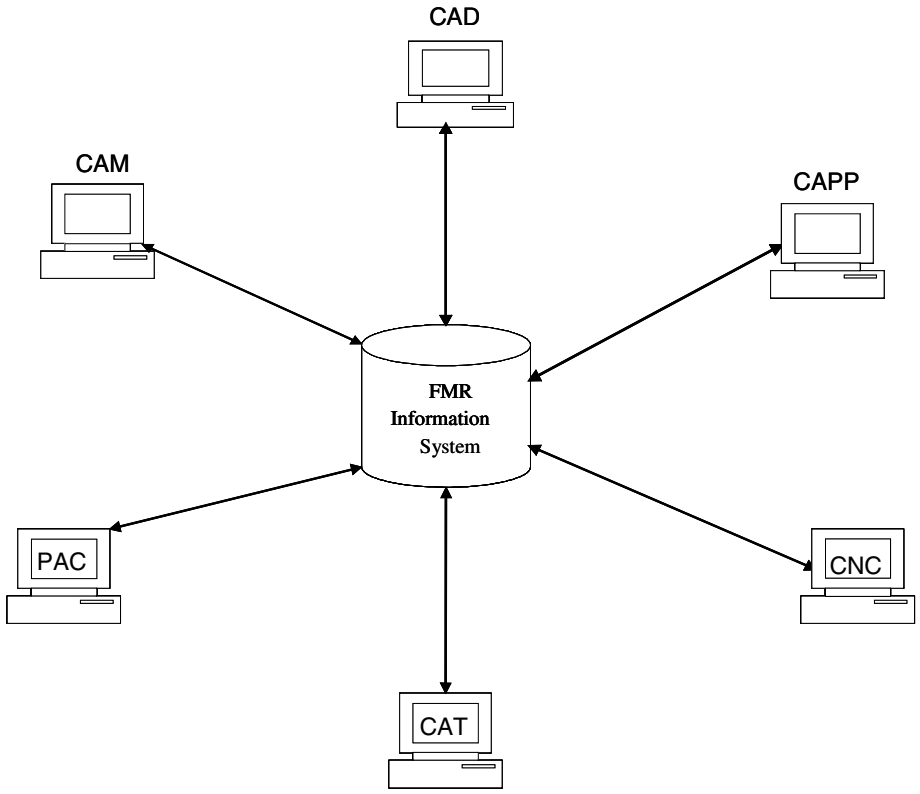

Fig. 1. Applications access data from a flexible manufacturing resources information system.

manufacturing resources (mainly object-oriented models) are also found in the scientific literature. An important current effort to achieve integration in manufacturing systems is conducted by the Object Management Group (OMG), which intends to develop CORBA-compliant applications. The objectives of these initiatives are succinctly presented next. Some notions regarding classification of the flexible manufacturing resources were useful in the final development of our EXPRESS models.

3.1 *ISO data models*

Table 3 presents those standards that contain data models for manufacturing entities. Although every standard has specific usage, all of them use EXPRESS as a modeling language and permit building neutral data repositories according to the format described in ISO 10303:21.

3.2 *Data models for flexible manufacturing resources*

Object-oriented models for flexible manufacturing resources have been reported in the scientific literature. In (Smith and Joshi, 1995) Flexible Manufacturing Resources are classified according to their function: (1) *Processing*, (2) *Storage*, (3) *Handling* and (4) *Transportation* resources. Similarly, an object-oriented model for an entire manufacturing cell is proposed in (Kcha and Park, 1996), where instances represent ASRSs, AGVs and CNC turning machines; data transfer is done over a TCP/IP network. Another model, using CORBAs IDL, is reported in (Suárez, 1998). The mentioned reference contains a *Work Station* class which is inherited by a *Cell* class which in turn is also inherited by a *Resource* class. In (Wataya *et al.*, 1997) it is compared a manufacturing cell with a biological cell due to the capability of each manufacturing cell to keep record of its own information.

Table 3. STEP-compliant standards

ISO standard	*Acronym*	*Objectives*	*Usage*
15531	MANDATE	To create normalized data models for improving manufacturing management information exchange (ISO, 1998e)	Manufacturing management and control
13584	PLIB	To develop data models for improving data exchanging of standard components via CAD files or common data repositories (ISO, 1999a)	Components and suppliers management
14649	CNC – STEP	To develop standardized data models for numerical control part programs (ISO, 1999b)	Numerical control part program creation and implementation
13399	None	To create normalized data models for cutting tools ISO (1999c)	Tools as entities to be integrated by CAD/CAM/CNC systems via neutral files or common databases

Some authors have modeled specific flexible manufacturing resources. For example, an Automated Guided Vehicle (AGV) is modeled in (Kwok and Norris, 1993). The *AGV* classes which constitute the model are built by decomposing the AGV in its basic components. The whole model is implemented in an object-oriented database management system called Gemstone. Another interesting proposal is the use of the EXPRESS language to construct flexible manufacturing resources data models (Chaxel *et al*., 1997). Finally, an UML model which is used to integrate physical properties of the resources has been reported in Schafer (1999).

3.3 *The CORBA[11] approach to manufacturing integration*

The Object Management Group (OMG) has launched the *Manufacturing White Paper* (OMG, 1996) which describes the road to follow so that manufacturing real objects can be represented by CORBA objects. The manufacturing real objects are modeled with the Interface Definition Language (IDL) of the Object Management Group. The data structure is represented by variables of the IDL. However, the OMG recognizes the achievements STEP and related standards on having modeled manufacturing entities. As a consequence the OMG also proposed to create interoperability mechanisms between STEP-compliant applications and CORBA applications.

4. The data model of flexible manufacturing resources

Flexible manufacturing resources are actually complex entities that provide data to the manufacturing system. In order to build a complete, robust information system some conceptual and technical issues were solved. For example it is proposed an efficient data organization for every flexible manufacturing resource. It is also proposed a mechanism to acquire and modify data by using a TCP/IP network, once the required information has been found in the MIS.

The proposed data model was constructed by defining a *Resource* as an abstraction that comprises the different types of flexible manufacturing resources known to date. It is also worth noting that the *Resource* has relations with the external world as seen in Fig. 2. One of them regards the fact that every *Resource* is actually a data source for the CAX applications; however, each application has particular data needs. That is to say, a CAPP requires different information of a Resource than that required by a PAC. Therefore, the MIS must only allow an application to view those data entities of a *Resource* that are strictly needed. This relation of the *Resource* with the *Application* is done via an *Application View*. It is said that a *Resource* presents different *Application Views* to the applications.

Another important aspect refers to the data organization for every *Resource*. Each *Resource* possesses particular information i.e. particular capacity and capabilities; but at the same time it belongs to one

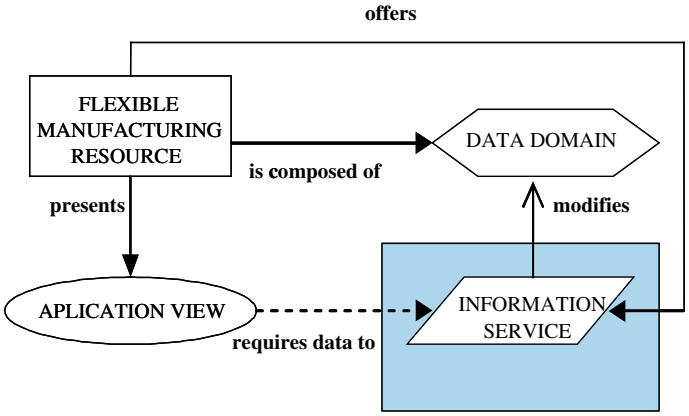

Fig. 2. Domains, Views, Services and their relation with the Resource.

category of *Resources* regarding their function i.e. a turning center and a machining center are both *processing resources*. Then it is necessary to create a useful concept to organize the data so that it reflects the particularities of the *Resource* and at the same time let it be maintained in the hierarchy of *Resources*. An appropriate concept that represents this fact is the *Domain*. A *Domain* is then the set of entities that compose a *Resource*. So a *Resource* is composed of one or more *Domains* that actually contain data. It is possible then to determine *General Domains*, *Intermediate Domains* and *Specific Domains* according to the position of the *Resource* in the hierarchy.

The way CAX applications access data of the resultant MIS is not a trivial issue, let alone when thinking of distributed applications over a network. A mechanism was implemented so that the applications could access the MIS. *Information Services* are proposed as the mechanism to modify the data structure of the *Resource* by demanding data to one or more *Resource Domains*. These *Information Services* take advantage of the network in order to connect applications to the MIS so data can be acquired and, in most of the cases, modified according to the dynamic changes of the manufacturing system.

Figure 2 is divided in two sections. The light-gray section corresponds to the information system. The MIS is composed of entities that represent the *Resource*, the *Domain*s of the *Resource*, and the *Applications View*. The dark-gray section represents the dynamic part of the proposed architecture. *Application*s via *Information Service*s demand or modify data of the *Resource* by modifying its *Domain*s (the *Domain* that the application is allowed to see).

4.1 *Construction of the data model for flexible manufacturing resources*

The proposed solution to build the EXPRESS model up reflects the considerations described lines above. Figure 3 is the resultant EXPRESS-G data model for flexible manufacturing resources. In the model an abstract entity *Resource* is defined. This entity constitutes the top of the resources hierarchy so that the *Flexible Resource* entity inherits data from the *Resource* entity. The next level of this hierarchy has the following entities: *Processing* resource, *Handling* resource[12], *Transportation* resource[13] and *Storage* resource[14]. The last level of this hierarchy has entities named after the specific resources belonging to one of the last categorization: *Turning Center* and *Machining Center* entities belong to the *Processing Resource* entity; the *Robot* entity is a type of *Handling* resource, whereas the *AGV* is a child entity of the *Transportation* resource, and finally the *ASRS* entity inherits data from the *Storage* entity.

On the other hand, the applications view concept, that was explained lines above, is represented by the *View* entity. The relationship between the *View* entity and the *Resource* entity stores the name of the application demanding information of a flexible manufacturing resource.

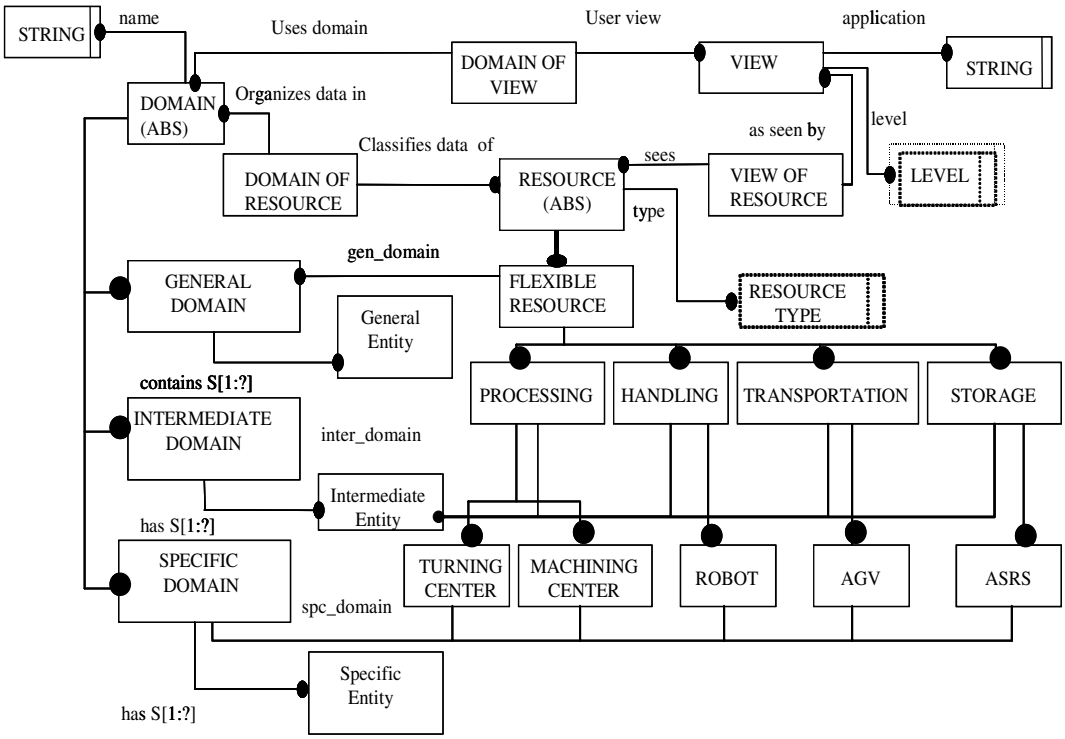

Fig. 3. The EXPRESS-G meta-model.

The information content of the flexible manufacturing resources that has common usage or similar attributes is grouped into a *Domain* entity. A *Domain* classifies data of one or more *Resources*, and a *Resource* has its data classified in one or more *Domains*. Each of the *Domains* is composed of entities that actually compose the logical or physical structure of each resource. More details about the model can be found in López-Ortega (2001, 2002).

4.2 *Validation of the EXPRESS meta-model*

The first stage of the validation process is done by creating instances of the entities that make up the data model. At this respect specific data was taken from an EMCO CNC turning machine, TC 100, which is part of the Manufacturing System Laboratory at Universidad Autónoma del Estado de Hidalgo. EXPRESS-I is the language used to *instantiate* the EXPRESS model. Figure 4 contains an extract of the resultant EXPRESS-I model. The complete EXPRESS-I model contains data taken from the next flexible manufacturing resources: One SCARA and two vertical robots from ESHED, a machining center, a turning center and one ASRS from EMCO systems. The second stage of the validation process was done by taking data into action, as it is described in the following section.

5. Implementation of the manufacturing information system

In order to exploit the proposed data-model, three Java programs were developed: The STEP-based information system, an information server and a client application (see Fig. 5). The whole suite of Java classes is comprised by the following modules:

Module *1*

This module is actually the implementation, in the Java language, of the STEP-based MIS. The Java classes that reproduce the data model of the flexible manufacturing resources were created by

```
TurningCenterUAEH = RESOURCE {type_of_resource -> "processing"}
TurningCenterUAEH = FLEXIBLE_RESOURCE {gdf ->@gdfisi_tuaeh; gdm -> @gdmate_tuaeh; gdi -> @gdiden_tueh;
gds -> @gdsiscoor_tuaeh;}
TurningCenterUAEH = PROCESSING_RESOURCE {idposm -> @idpos_tuaeh; idproc -> @idproc_tuaeh; iddim ->
@iddim_tuaeh; idelmo ->
@idelmo_tuaeh; iddes -> @iddes_tuaeh; idcon -> @idcon_tuaeh; idherr -> @idherr_tuaeh;}
TurningCenterUAEH = TURNING {edhetor -> @edhetor_tuaeh; max_part_dim ->@inter_domain_partTuaeh;
distance_between_centers ->
@distbetcentrTuaeh;}
gdiden_tuaeh = DOMAIN {domain_name -> "idDomain"; domain_type -> "general";}
idproc_tuaeh = DOMAIN {domain_name -> "processDomain"; domain_type -> "intermediate";}
sdhetor_tuaeh = DOMAIN {domain_name -> "turningTools"; domain_type -> "specific";}
v1tuaeh = VIEW {application -> "Process Planning"; level -> "Manufacturing Activity Planning",}
v2tuaeh = VIEW {application -> "Production Activity Control"; level -> "Manufacturing Activity Execution";}
resDom3_tuaeh = RESOURCE DOMAIN {classifies data of -> @TurningCenterUAEH; organizes data in -> @
gdiden_tuaeh;}
gdiden_tuaeh = IDENTIFICATION GENERAL DOMAIN {ide -> @identity_tuaeh;}
idproc_tuaeh = PROCESS INTERMEDIATE DOMAIN {machiningProcess -> @processes_tuaeh;}
identity_tuaeh = Identifier {resource number -> "1";
resource_name -> "turning center – UAEH"; resource_state -> "non – busy"; vendor_name -> "EMCO"; serial_number ->
"uaehmansyslabtc100";
model -> "htc100",}
processes_tuaeh = Machining_Process {resource_name -> "turning center UAEH"; process_capability ->{"reaming, boring,
drilling, facing";};}
```

Fig. 4. Extract of the EXPRESS-I model.

compiling the EXPRESS model, process that is also useful to check for possible errors in the model. Specific instances of flexible manufacturing resources are stored in this STEP-based MIS. The set of Java classes that constitute the MIS was created with the CASE tool ST Developer 1.8 from STEP Tools Inc.

Figure 6 shows the information structure of a Turning Center, implemented in a Java class named TornoC (TornoC stands for Turning Center). This class contains a number of variables that store data regarding any given Turning Center. For example, the TornoC member variable Diproc is an instance of an object called

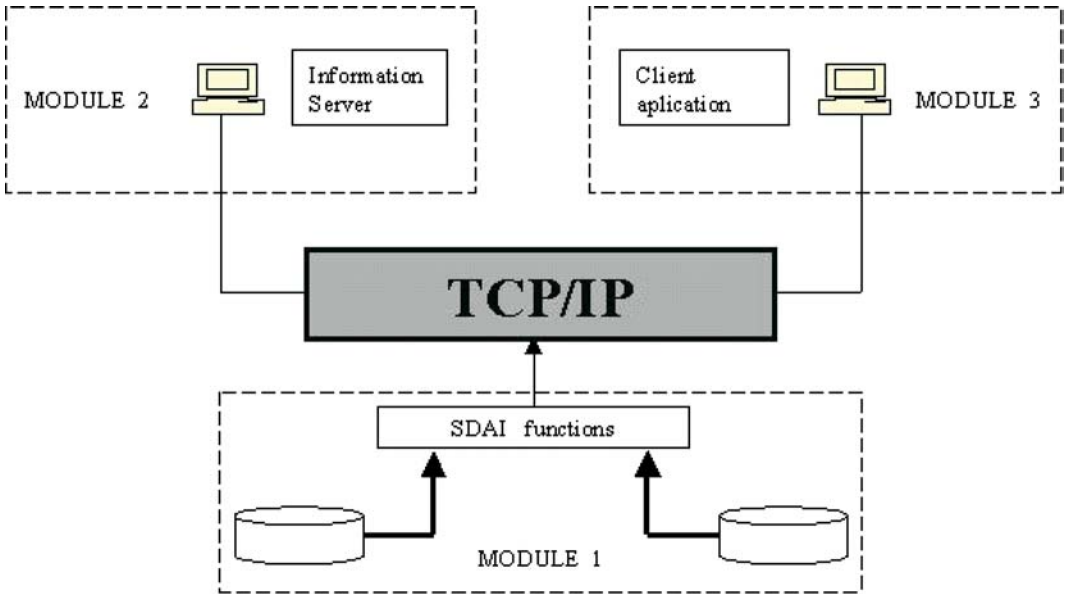

Fig. 5. The three modules: MIS, server and client application.

```
package SDAI.Recurso_fabricacionS;
import SDAI.lang.*;
import java.io.*;
import java.util.*;

public class TornoC extends App_instC implements Torno {public TornoC() {}

public TornoC(Model_contents mc) {mc.addInstance(this);}
public final SDAI.Recurso_fabricacionS.Process_Intermediate_Domain getDiproc()
{return Diproc;}
public final void
setDiproc(SDAI.Recurso_fabricacionS.Process_Intermediate_Domain v) {Diproc = v;}
public final SDAI.Recurso_fabricacionS.Tool_Intermediate_Domain
getDiherr() {return Diherr;}
public final void
setDiherr(SDAI.Recurso fabricacionS.Tool_Intermediate_Domain v) {Diherr = v;}
public final SDAI.Recurso_fabricacionS.Position_Intermediate_Domain getDiposm()
{return Diposm;}
public final void
setDiposm(SDAI.Recurso_fabricacionS.Position_Intermediate_Domain v) {Diposm = v;}
public final SDAI.Recurso_fabricacionS.Fixture_Intermediate_Domain getDisuj()
{return Disuj;}
public final void
setDisuj (SDAI.Recurso_fabricacionS.Fixture_Intermediate_Domain v) {Disuj = v;}
public final int getTipo_de_recurso() { return Resource_Type;}
public final void setTipo_de_recurso(int v) {Resource_Type = v;}

//instance objects of the different DOMAINS of a Turning Center:
protected SDAI.Recurso_fabricacionS.Process_Intermediate_Domain Diproc = null;
protected SDAI.Recurso_fabricacionS.Tool_Intermediate_Domain Diherr= null;
protected SDAI.Recurso_fabricacionS.Position_Intermediate_Domain Diposm = null;
protected SDAI.Recurso_fabricacionS.Fixture_Intermediate_Domain Disuj = null;
protected int Resource_Type = java.lang.Integer.MIN_VALUE;
protected SDAI.Recurso_fabricacionS.Performance_Intermediate_Domain Dides = null;
protected SDAI.Recurso_fabricacionS.Machine_Dimension
Part_Max_Dim = null;
protected SDAI.Recurso_fabricacionS.TurningTools_Specific_Domain Dehetor = null;
protected SDAI.Recurso_fabricacionS.Siscoor_General_Domain Dgs = null;
protected SDAI.Recurso_fabricacionS.Movement_Elements_Intermediate_Domain Dielmo = null;
protected SDAI.Recurso_fabricacionS.Identification_General_Domain Dgi = null;
```

Fig. 6. Java code extract of the MIS – The Turning Center.

Process_Intermediate_Domain which, according to the model, is a type of *Domain*. In the same way the rest of the member variables of the class under study instantiate several *Domain* objects, where the information is actually stored. It can be seen that a Turning Center is composed of different types of *Domain*s. For example, a *General_Domain* is the represented by the object Siscoor_General_Domain. Similarly the object Movemement_Elements_Intermediate_Domain is one of the *Intermediate_Domain*s of a Turning Center. This processing resource also contains *Specific_Domain*s, implemented with objects like TurningTools_Specific_Domain. The Java implementation of the MIS fully corresponds with the developed EXPRESS model (Fig. 3).

Module *2*

This module is made up with Java classes that constitute the information server. This information server reads and writes data in the implemented STEP-based MIS. The information server has also the necessary code to connect to a local area network under the TCP/IP protocol, in order to send and receive data of a specific flexible manufacturing resource. This capability is obtained by using the java.net package of the Java language in conjunction with the SDAI.lang package of the ST Developer tool. Figure 7 shows an example of the Java code of the information server and its connection with the MIS.

```
import SDAI.DeltornoS.*; __
import Java.net.*;
ímport Java.io.*;

public class Server extends Thread
{       public final static int PORT= 89;
        protected ServerSocket socket;
        public Server(int port);
        try (socket = new ServerSocket(port);
        this.start;
public void run()
{       Socket socketl = socket.accept(); Connection c = new
Connection(socketl) ;}
}
class Connection extends Thread4
{       protected Socket client;
        calcOffsetZ z, x;
        double Z-offsetValue. X-offsetValue;
        Offset offset-ins, offsetl;
        Position position-ins;
        Reference-axis axis-ins;
        Tool tool-ins;}
public Connection(Socket client-socket)
{ in = new DataInputStream(client.getInputStream()};}
  this.start() ;
public void run()
{ line=in.readLine(); calcOffsetZ(line); calcOffsetX(line);}
```

Fig. 7. Java code extract of the information sever.

Module *3*

This last module is the client application. This application searches the server, vía an IP address, to read or write data of flexible manufacturing resources. If the server is running the application connects to it, otherwise an error message is presented. Figure 8 shows how the SDAI functions are used to access the STEP-based MIS in order to implement a given service (i.e. the tool offset determination for a turning center). It is necessary to *open* a Session and specify the *type* of Repository that contains the instances of the flexible manufacturing resources. In this case a file that complies with the specification of part 21 of the STEP standard is used as a data repository. Once the type of Repository is defined, a Model is accessed to *get* the Contents of the Model. The example shows how data of a Reference_axis for a Turning Center is obtained and used to calculate the offset of a Tool.

The application has the capabilities to search the most appropriate resource to execute a task, according to dimensional and technological requirements of a given part. The application requests data of every processing resource instantiated on the common database, receives this data on the client port, and processes it. If the appropriate processing resource is found, it sends a message so that the chosen resource is asked for information. The complete suite was implemented in a client–server architecture so that data from flexible manufacturing resources can be accessed by remote applications. Figure 9 shows a screen-shot of the application.

6. Comparison of the proposed data model with related work

Both, the set of manufacturing resources and the corresponding model presented in this paper have a close relationship with ISO 15531, known as MANDATE (MANufacturing management DATa Exchange). However, the objectives of MANDATE are slightly different from the objectives of the model presented in this paper. MANDATE models a *Resource* entity around which the database is built. In MANDATEs sense, a *Resource* is any element or device within the manufacturing system: fixtures, machines, workers, tools, etc. According to Walter Eversheim, one

```
Class Connection extends Thread
(
        protected socket client;
        offsetZ z;
        offsetX X;
        double Z_value;
        double X_value;
        Position position_ins;
        Reference_axis axis_ins;
        Tool tool_ins;
        public void run()
        (
                OffsetZ(line);
                OffsetX (line);
        }
        public void OffsetZ(String line)
        (       Session session = SessionC.OpenSession();
                Repository repo = session.OpenRepository("file");
                Model model = repo.AccessModel(line, Access_type.READ_WRITE);
                Enumeration axis = model.getContents.getEntity
        extent("reference_axis");
                Enumeration tools = model.getContents.getEntity_extent("tools");
                Enumeration positions =
        model.getContents.getEntity_extent("position");
                enumeration offsets = model.getContents()
                .getEntity_extent("offset");
                tools_ins = (Tool)tools.nextElement();
                axis_ins = (Reference_axis)axis.nextElement();
                offset_ins = (Offset)offsets.nextElement();
                position ins = (Position) axis_ins.getPosition_on_Part();
                PositionZ1_on_Part=(double) position_ins.getposition_z();
                PositionZNext_on_Part=(double) position_ins.getposition_z();
                while (tools.hasMoreElements () {
                Z_value = PositionXNext_on_Part - PositionZ1_on_part;
                }}
model.SaveChanges();
session.ClseSession();}
```

Fig. 8. Use of the SDAI functions to access the MIS and implement services.

of the MANDATE project leaders, the model is general enough to store data from any resource within the manufacturing system (Eversheim *et al.*, 1998).

The related work also contemplates CORBA objects which are developed under OMG patronage and by independent researchers. The main objective of these objects is also to enhance interoperability among the applications that are compliant with the OMG standard. The OMG also recognizes, though, that interoperability can be improved if communication with STEP-based applications can be achieved. The set of entities that model flexible manufacturing resources presented in this paper can be of great interest because none of the STEP-based standards explicitly contemplate entities like the ones modeled in this work. Indeed, data from machine-tools, robots, ASRS, is needed in a more detailed fashion and by many applications and activities, as it was noted in the related work. It was seen that data from flexible manufacturing resources is required by CAPP, PAC, tool-setting, network message passing applications, among others.

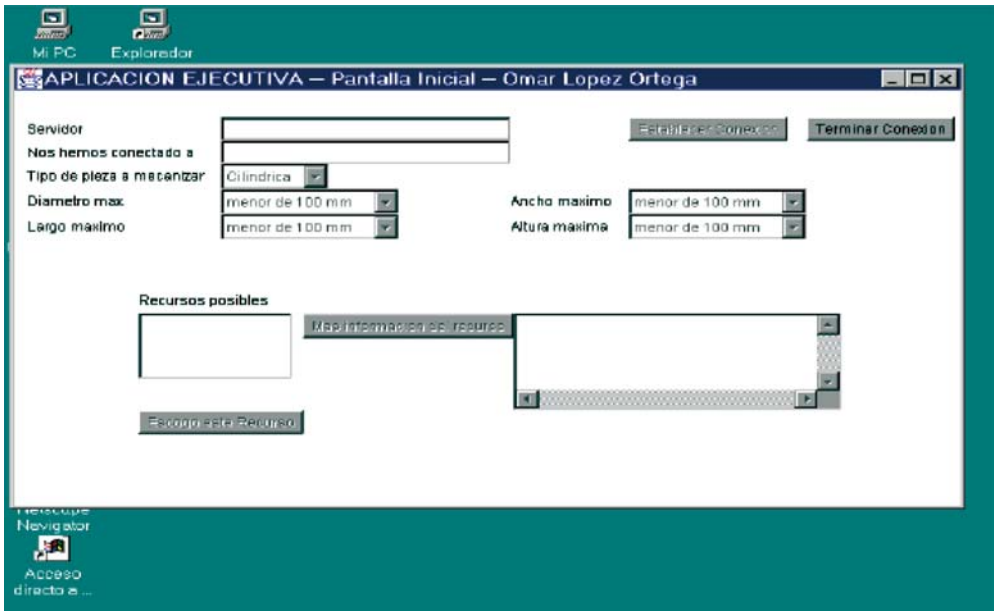

Fig. 9. GUI of the client application.

The model proposed in this paper is intended to store specific information of machine-tools, robots, AGVs, etc. The model is intended to store physical and logical data of the contemplated flexible manufacturing resources. It is useful to delineate the resources as they are designed and used, including processing capabilities for CNC machine-tools, mathematical and kinematical data for robots, physical structure and routes to follow as in the case of transportation resources, storing capabilities for ASRSs, etc. The authors of this paper argue that a more detailed description of the flexible manufacturing resources is actually obtained and a finer granularity of information can be stored. Java applications were developed to prove the model usage.

7. Future work – STEP-based extended enterprises

The ISO standards 10303 (STEP), 15531 (MANDATE), 13584 (PLIB), 14649 (CNC-STEP), along with the EXPRESS model for flexible manufacturing resources described in this paper, can be used to construct an information architecture to be used by Extended Enterprises. An information architecture based on these standards constitutes a solution to improve data sharing among the participants of the Extended Enterprise. However, it is important to analyze the data models of each of the standards, define the role of the participants both as data sources and data sinks, and develop a model that relates them.

By analyzing ISO 15531 it is possible to argue that their data models can be used to manage the flow of information related with suppliers, external production systems or resources used in the Extended Enterprise. ISO 15531 describes data models for production capacity and permits data sharing of manufacturing systems management. On the other hand, ISO 13584 proposes data models for sharing and exchanging information related to the parts supplier, the final user, and the relation among the components of a given product by means of several Basic Semantic Units. The current research work consists of developing an integrated information architecture whose preliminary design is already done. The integrated architecture is being developed by studying a set of common transactions given in the context of the Extended Enterprise. A data repository is also being modeled which will also be accessed through the SDAI functions. An application called Extended Enterprise Mediator is being created to

coordinate the data flow of the participants of the Extended Enterprise, see Fig. 10.

8. Conclusions

STEP-based information systems constitute a promising path to improve MIS. However, despite the intense information demanded from flexible manufacturing resources in several manufacturing activities, i.e. manufacturing planning and execution, no explicit data model had been developed to represent their complex information structure. Data models have been reported in the scientific literature; such models, though valuable, are not entirely coupled within the manufacturing context. The EXPRESS data model and its corresponding information system we described in this paper constitute a technological solution to store, share and exchange data of flexible manufacturing resources. The EXPRESS model is useful as a specification language for defining data and the constraints that must be respected by the computer-based applications in order to acquire information of flexible manufacturing resources. It is important to remark that the solution we achieved is coherent with STEP-based information systems, so that applications can interoperate and achieve a higher degree of integration.

The EXPRESS data that we proposed is built on three core concepts: Resource, View and Domain. Around these three concepts information regarding flexible manufacturing resources was stored regardless a particular system used to process it. As a result of having expanded and instantiated the proposed model, it was proved that:

1. The whole information content of the flexible manufacturing resources was stored.
2. No information distortion was noted.
3. Physical and logical coherence among the entities of the real-world flexible manufacturing resources were maintained with the entities that make up the EXPRESS model.
4. When accessing the data repository (implemented in several hundred Java classes) the ACID[15] properties of the transactions were satisfied.
5. Data independence was obtained. In this sense the data relies on the model not on the computer-based application.

We are aware that new concepts can be added to the model due to its modular design. This modularity also permits a partial implementation of the model, according to the user's needs. In order to fully comply with STEP it is still necessary to develop proper interfaces with the existing models. This is also possible and achievable as the core features of STEP are respected. However, to our knowledge, this is the first attempt to systematize data and usage of the flexible manufacturing

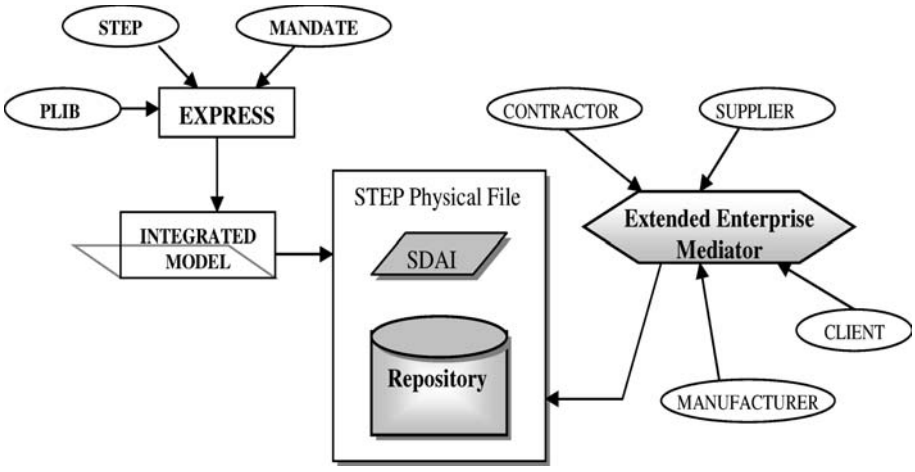

Fig. 10. The proposed STEP-based architecture for the Extended Enterprise.

resources, taking into account their manufacturing planning and execution role.

Although the research project was initially intended to accomplish integration within a manufacturing enterprise, it is possible to continue with the same research subject in order to create open information systems for the extended enterprise. As the EXPRESS models presented in PLIB and MANDATE are intended to demolish the information barriers among CAX applications, an appropriate framework must be designed to use them for exchanging manufacturing data along the product life-cycle. Future work will be done in this direction.

Acknowledgments

The research of Omar López-Ortega was partially supported by a SEP-PROMEP grant. Miss Francisca Tanca kindly advised style corrections.

Notes

1. STandard for the Exchange of Product data.
2. Automatic Storage and Retrieval System. See Table 1 for further information.
3. Automatic Guided Vehicle. See also Table 1.
4. Computer-Aided Manufacturing. See Table 2 for further details.
5. Computer-Aided Process Planning. See also Table 2.
6. Production Activity Control. See Table 2.
7. CAX is a general term for a given computer-aided activity such as Design, Manufacturing or Process Planning. X is replaced by one specific manufacturing activity.
8. Computerized numerical- control machines such as turning centers or machining centers. See Table 1.
9. International Standards Organization.
10. CATIA and KISMET are two Computer-Aided Design applications that also possess modules for robot movement simulation.
11. Common Object Request Broker Architecture. Middleware used to allow interoperability among heterogeneous software systems.
12. Material handling is the movement, protection and control of materials throughout the manufacturing system. Robots are a type of material handling resources.
13. Automatic equipment used to move materials inside a factory, warehouse or other facilities. AGVs are material transportation resources.
14. The function of a storage resource is to store materials and permit access to them when required.
15. Atomicity, Consistency, Isolation and Durability. Basic properties every information system must possess.
16. Groover (2001, p. 282).
17. Groover (2001, p. 329).
18. Groover (2001, p. 128).
19. Groover (2001, p. 211).

References

Browne, J., Harhen, J. and Shivnan, J. (1992) *Production Management Systems: A CIM Perspective*, London: Addison-Wesley 47–48.

Chaxel, F., Bajic, E. and Richards, J. (1997) Mobile database nodes for manufacturing information management: a STEP based approach. *International Journal of Advanced Manufacturing Technology*, **13**, 125–133.

ESPRIT Research Group. (1994) *NIRO – Neutral Interfaces in Design, Simulation and Programming for Robots, ESPRIT Research Projects*, Germany: Springer-Verlag.

Eversheim, W. *et al*. (1998) Resource Information Management in Autonomous Production Cells and Virtual Enterprises. In *Proceedings of the European PDT Days*, Vol. 1, pp. 149–156, United Kingdom.

Fahim, A. and Choi, K. (1998) The Uniset approach for the programming of flexible manufacturing cells. *Robotics and Computer Integrated Manufacturing*, **14**, 69–78.

Groover, Mikell P. (2001) *Automation, Production Systems and Computer Integrated Manufacturing, 2nd ed*, USA: Prentice-Hall.

ISO – International Standards Organization (1998a) *Industrial Automation Systems and Integration – Product Data Representation and Exchange – Part 21 Clear-Text Encode Exchange Structure*. Geneva, Switzerland..

ISO – International Standards Organization (1998b) *Industrial Automation Systems and Integration – Product Data Representation and Exchange – Part 22: Implementation Methods: Standard Data Access Interface*, Geneva, Switzerland.

ISO – International Standards Organization (1998c) *Industrial Automation Systems and Integration – Product Data Representation and Exchange – Part 24:*

Implementation Methods: C++ binding for the Standard Data Access Interface, Geneva, Switzerland.

ISO – International Standards Organization (1998d) *Industrial Automation Systems and Integration – Product Data Representation and exchange – Part 27: Implementation Methods: Java binding for the Standard Data Access Interface*, Geneva, Switzerland.

ISO – International Standards Organization (1998e) *Industrial Automation Systems and Integration – Industrial Manufacturing Management Data – Part 1: Overview and Fundamental Principles*, Geneva, Switzerland.

ISO – International Standards Organization (1999a) *Industrial Automation Systems and Integration – Parts Library – Part 10: Overview and Fundamental Principles*, Geneva, Switzerland.

ISO – International Standards Organization (1999b) *Industrial Automation Systems and Integration – Physical Device Control: Data Models for Computerized Controllers. Part 1: Overview and Fundamental Principles*, Geneva, Switzerland.

ISO – International Standards Organization (1999c) *Industrial Automation Systems and Integration – Cutting Data and Exchange. Part 1: Overview and Fundamental Principles*, Geneva, Switzerland.

Kcha, S. and Park, J. (1996) An object-oriented model for FMS control. *Journal of Intelligent Manufacturing*, **7**, 387–391.

Kwok, A. and Norris, D. (1993) Intelligent agent systems for manufacturing applications. *Journal of Intelligent Manufacturing*, **4**, 285–293.

López-Ortega, O. (2001) A Java application based on an EXPRESS model for sharing flexible manufacturing resources data. In *Proceedings of the Eighth IEEE International Conference on Emerging Technologies and Factory Automation*, Vol. 2, pp. 3–11, ESINSA, France, October 15–18, IEEE.

López-Ortega, O. (2002) Design and implementation of an open manufacturing information system to enhance data sharing and exchanging among applications. In *Proceedings of the 2002 IEEE International Symposium on Industrial Electronics*, Vol. 1, pp. 245–253, IEEE.

OMG – The Object Management Group – Manufacturing Domain Task Force. (1996) *Manufacturing Enterprise Systems: A white paper*, Framingham, MA, USA.

Pierra, G. (2000) Représentation et échange de données techniques. *Mec. Ind.,*, **1**, 397–414 (in French).

Schafer, C. and López-Ortega, O. (1999) An object-oriented robot model and its integration into flexible manufacturing systems. *Lecture Notes in Computer Science*, Vol. 1611, pp. 820–828, Springer-Verlag, Germany.

Shen, W., Norrie, D. and Barthes, J.-P. (2001) *Multi-Agent Systems for Concurrent Engineering and Manufacturing*, London: Taylor and Francis 18.

Singh, V. (1997) *The CIM debacle. Methodologies to facilitate software interoperability*, Singapore: Springer-Verlag.

Smith, Jeffrey S. and Joshi, S. (1995) A shop floor controller class for computer integrating manufacturing. *International Journal of Computer Integrated Manufacturing*, **8**, 327–339.

Suárez, O. (1998) Standard-based framework for the development of manufacturing control systems. *International Journal of Computer Integrated Manufacturing*, **11**, 401–415.

Wataya. H. *et al.* (1997) Integration of control and information systems by open autonomous decentralized system architecture and application to distributed manufacturing systems. *IEEE transactions on industrial electronics*, pp. 147–154.