A Federated Approach to Tool Integration

Malek Bounab and Claude Godart

CRIN/CNRS BP 239, 54506 Vandœuvre-lès-Nancy, France
E-mail: {bounab, godart}@loria.fr

Abstract. Due to enormous pressures from national and international marketplaces, Computer Integrated Manufacturing (CIM) has become a tremendously important area for both research and development. However, the current state of the art is still characterized by islands of automation. In order to connect these islands, appropriate frameworks have to be developed to integrate heterogeneous Computer Aided Design (CAD) tools. We present in this paper a federated approach to tool integration in distributed and heterogeneous environments making tools evolve in an autonomous way. We have experimented this approach by integrating PROPEL[1] and SPEX[2] CAD tools in the DMMS (Design Management and Manufacturing System) environment backed by a common PCTE[3] repository.

1 Introduction

CIM environments have known, these recent years, a tremendous expansion due to enormous pressures from national and international marketplaces. The purpose of CIM environments is to integrate all processes carried out within an enterprise. However, these environments are still characterized by heterogeneous islands of automation composed of monolithic CAD tools. Therefore, integrating tools becomes a key issue to offer homogeneous environments allowing (i) an uniform underlying communication system for events notification and messages interchanging between tools *(control integration)* (ii) an uniform repository in which all data are stored and shared *(data integration)* (iii) an uniform "look and fell" which enable users to swith between different tools easily *(presentation integration)* and (iv) an uniform and coordinated processing of activities with other tools *(process integration)*.

We focus in this paper on the data integration dimension. A standard approach consists in integrating all data into a large centralized database (like in GANDALF [1], CPS [2] and PECAN [3]). Though, tools involved in CIM environments are generally spread over different locations. On the other hand, a common representation doesn't always matches tools data modeling requirements

[1] PROPEL is a product of ITMI
[2] SPEX is a product of TNI, CRAN and SPIE-TRINDEL
[3] PCTE is an ISO standard providing an open repository for software development. We use in our experiment an implementation of PCTE 1.5 named Emeraude V12.5, product of GIE EMERAUDE

and it is often necessary to use different data models. Heterogeneous distributed databases (HDDB) seem to fulfill both previous requirements since they allow tools, scattered on different locations, to use data models which best fit their data modeling (eg: DATAPLEX [4] and IMDAS [5]). However, tools have to access a global schema, representing all data managed by HDDB, even for their own data, implying a loss of autonomy and performance.

This paper addresses the problem of tool integration in distributed and heterogeneous CIM environments where tool autonomy is an important requirement. We propose a federated approach built on the architecture of a federated database [6] preserving tools autonomy regarding the distribution and the heterogeneity of data. We present in section 2 the integration architecture and methodology followed to build *tool federations*. This methodology is experimented in the DMMS (Design Management and Manufacturing System) environment backed by a common PCTE repository. We focus in this paper particularly on data integration during the *design* step. Different integration steps leading to tool interoperability are described in section 3 and a production scenario showing tools cooperation for mechanical parts design concludes this section. The final section presents some lessons learned from our experiment and from the use of PCTE, initially designed for software engineering requirements, in a manufacturing environment.

2 Integration Architecture and Methodology

The DMMS environment, actually developed jointly by the Computer Science Research Center of Nancy (CRIN) and the Automation Research Center of Nancy (CRAN) [7, 8, 9] (figure 1), aims to establish a semantic link between different concurrent manufacturing functions.

DMMS intends to couple mechanical and automation working fields for designing and managing a product. It is composed of four working stations achieving (i) the design function, provided by automation and the mechanical working stations, which cooperate to design a product, (ii) the manufacturing function, provided by the flexible cell, which aims to execute the manufacturing of designed product, (iii) the maintenance function, provided by the maintenance working station, which insure the maintenance of a product during all its life cycle and (iv) the management of exchanged data between different working stations by the management working station. Each working station is composed of a set of tools integrated on top of a local repository. The different set of tools cooperate through a common PCTE repository.

The federated approach to tool integration is based on the reference architecture defined in [6] for database interoperability and consists in defining a set of schema levels which ensure, on the one hand, a tool autonomous access to its data and, on the other hand, a federated access to other tool's data via a canonical data model.

The data managed by tool and the relationships between them are defined by a *reverse engineering* process [10]. An abstract data representation is built using a conceptual data model which is, in our case, the ERA model [11].

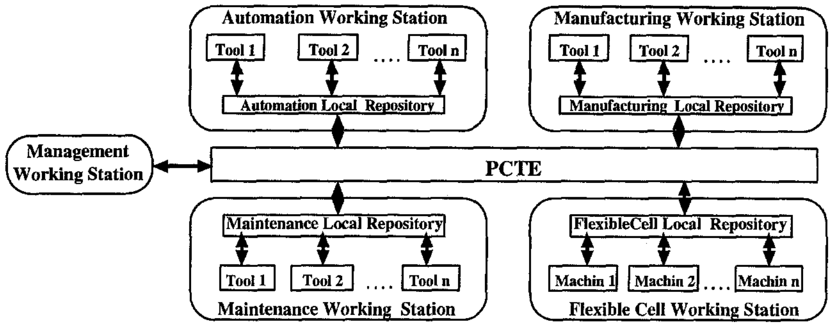

Automation Working Station
Tool 1
Tool 2
....
Tool n
Automation Local Repository
Manufacturing Working Station
Tool 1
Tool 2
....
Tool n
Manufacturing Local Repository
Management Working Station
PCTE
Maintenance Local Repository
Tool 1
Tool 2
....
Tool n
Maintenance Working Station
FlexibleCell Local Repository
Machin 1
Machin 2
....
Machin n
Flexible Cell Working Station

Fig. 1. DMMS ARCHITECTURE

Local accesses are done through local schemas expressed in the tool integration repository data model. This allows tools to access their own data locally and in an autonomous way without managing other data. We can then choose the most suitable local data model which best fit tools data modeling constraints and requirements.

Local schemas are then translated into a common data model which provides a coherent view of many distinct tools data. Different views, defined on these translated schemas, are integrated into a global federated schema representing all information which may be shared between tools. Finally, a view on these schemas allow tools interaction and cooperation. This view definition is visible to all federated tools in addition to their own local schemas.

To summerize, we give hereafter different steps of our methodology achieving tool integration in a heterogeneous and distributed environment.

- **step 1:** Define different tool federations by grouping tools according to a given criteria (eg: their application field).
- **step 2:** Define *conceptual schemas* by a reverse engineering process from tools managed data. The objective of this step is to define an abstract representation of tool data without considering any physical repository.
- **step 3:** Translate each conceptual schema to tool integration repository data model. The obtained schemas, expressed in the native data model of the database, are called *local schemas.*
- **step 4:** Translate each local schema to the canonical data model which unifies data representations through tool federation. The obtained schemas are called *translated schemas.*
- **step 5:** Build *exported schemas* by defining views on translated schemas. Exported schemas are requested because not all translated data have to be viewed by the federation. These views are defined for specific class of users and applications.
- **step 6:** Exported schemas are then integrated into *federated schemas.* The schema integration [12] provides data consistency between different exported schemas since common entities are merged into one unique entity after re-

solving name and structure conflicts.
At this level, auxiliary schemas may also be integrated to the federated schema. These schemas may contain information not directly related to tools but, for example, to the current project.
- **step 7:** To a given application or group of users, not all information belonging to the federated schema have to be viewed by tools. Therefore, we have to define a specific view on federated schemas called *external schemas*.

The number of these steps is not fixed. For example, if local and canonical data models are identical, a translated level is not required. The exported level is also not necessary if we want to integrate into the federated schema all translated schemas. In fact, the number of federation levels depends mainly on the federation components.

3 Experimenting the Tool Integration Methodology

We focus here on the design function which is mainly achieved by the mechanical and the automation working stations. This function deals with the relationship between the mechanical and automation skills which cooperate to design a product. It is a relatively long step since both two skills have to frequently reconsider their previous design. This is due to the set of physical requirements and constraints which must be considered during the manufacturing step.

The tool integration experiment involves the integration of PROPEL and SPEX CAD tools. These tools, respectively representative of mechanical and automation skills, have to cooperate widely to achieve mechanical and automation part design. MATISSE[4] database models mechanical local data while PCTE models those of the automation working station.

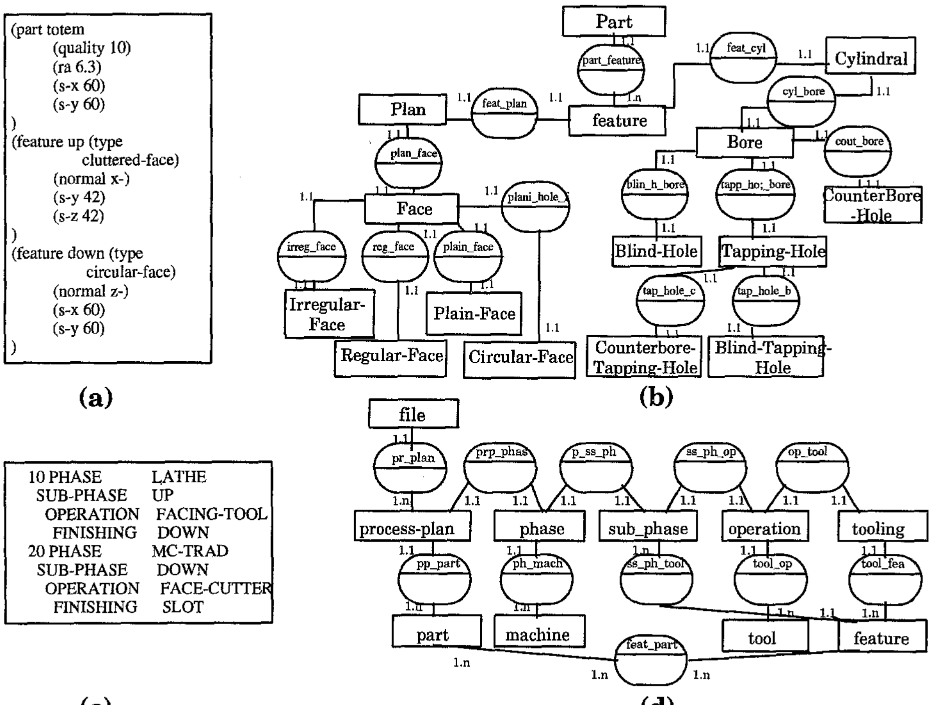
3.1 Reverse Engineering of the Tools

As we integrate only two tools belonging each to a different skill, each tool federation is reduced to only one tool. As a consequence, the first step of our methodology in not required. The second step consists in building conceptual schemas of the tools by reverse engineering from their data.

PROPEL Mechanical Design Tool
PROPEL [13] is an expert system generating process plans from part and workshop descriptions. The part is described by geometrical features (faces, slots...) and relationships between them (coaxiality, parallelism...). The workshop is described by a set of machines (lathes, milling machines ...), manufacturing tools present in the workshop (drills, face-cutters...) and the relationships between them. Part and manufacturing process plan descriptions are textual files which are either created by a tool or a user. In the following we present PROPEL and SPEX tools and the modeling of their data.

[4] MATISSE is an object oriented database produced by Intellitic International SNC.

Part and Manufacturing Process Plan Description and Modeling
The second step of the integration methodology consists in extracting data contained in the part and the process plan descriptions, by reverse engineering, in order to model data handled by PROPEL. The description of a part in PROPEL (figure 2.a) contains basic features composing it and the physical and geometrical characteristics of each feature. PROPEL being closed, we have considered only input and output data corresponding respectively to the part and the manufacturing process plan descriptions. The *part schema* (figure 2.b) includes all



```
(part totem
      (quality 10)
      (ra 6.3)
      (s-x 60)
      (s-y 60)
)
(feature up (type
            cluttered-face)
      (normal x-)
      (s-y 42)
      (s-z 42)
)
(feature down (type
             circular-face)
      (normal z-)
      (s-x 60)
      (s-y 60)
)
```

(a)

```
10 PHASE        LATHE
  SUB-PHASE     UP
    OPERATION   FACING-TOOL
      FINISHING DOWN
20 PHASE        MC-TRAD
  SUB-PHASE     DOWN
    OPERATION   FACE-CUTTER
      FINISHING SLOT
```

(c)

Fig. 2. Part and Process Plan Description and Modeling (step 2)

the features and parameters composing and describing a part. We have followed a *top-down* approach to break down a part into a set of features managed by PROPEL. Therefore, the schema describing a part contains all features which may occur in its description.

The manufacturing process plan generated by PROPEL (figure 2.c) represents operations to be processed in order to achieve a part manufacturing. These operations are structured in phases corresponding to machines involved in part manufacturing. From the process plan description generated by PROPEL, we build, by reverse engineering, the process plan schema depicted by figure 2.d. Each part (represented by the entity *part*) has one associated process plan (*process-plan* entity) linked to a *phase* entity. Each *phase* is linked to a set of *sub-phases* which are, in their turn, linked to *operations*. Each *operation* is composed of *toolings* processed on *features*. We associate, for each *phase*, a *machine* and, for each *sub-phase*, a *feature.*

SPEX Automation Design Tool

SPEX [14] is an environment allowing to specify, to design and to prototype automation systems. The objectives of SPEX are to allow designers to build reusable behaviors and to validate a functional organization during the analysis phase. SPEX manages two kind of automation components which are the *functional box* (FB) and the *functional diagram* (FD).

A functional box is a "behavioral" unity producing one or many output values from a set of input values and parameters and may be a graphcet, a ladder, a logical schema or a C program. Several behavioral boxes may be instantiated from a "generic" functional box. Each FB has its own identifier, a behavior and a set of variables (input, output and parameters). A functional diagram is a collection of FBs and/or FDs interconnected via their input/output interfaces and representing a composite behavior.

Building a SPEX Automation Design Application

We use a top down approach to define a SPEX application . That means that all FDs are initially empty and are completed by the automation engineer while needed information are received from the mechanical engineer. SPEX automation design application considers four basic activities occuring in a product design. These activities are the TRANSPORT activity which handles transport of products between machines during the manufacturing step, the TRANSFORM activity which considers the effective manufacturing of a product, the STORE/UNSTORE activities which ensure product storage and unstorage before its use and finally, the CONTROL activity which controls data flow between the above activities.

The information given by the mechanical engineer are considered in the transforming activity while the controling, transporting and storing/unstoring activities are defined by the automation engineer, the robotics engineer and the production manager. These activities are represented by CONTROL, TRANSPORT, STORE/UNSTORE and TRANSFORM FDs . The FD corresponding to TRANSFORM activity is built from information given by the process plan generation step. Each sub-phase of the process plan corresponds to a generic empty FD and to each of these FDs is associated a set of manufacturing tools used in this sub-phase. To each tool, is associated an empty FD which input/output variables and parameters are set by the automation engineer.

Automation Design Application Schema

The automation design application schema is deduced from the SPEX automation design application and is completed later by the automation engineer which has to define each FB behavior and connect different FBs and FDs making up the application. The schema built from this application comprises *Tool*, *Station*, *Control*, *Transform*, *Transport* and *Store/Unstore* object types (figure 3). These object types are all linked to *I_ Var*, *O_ Var* and *Behavior* object types corresponding respectively to their input variables, output variables and behaviors.

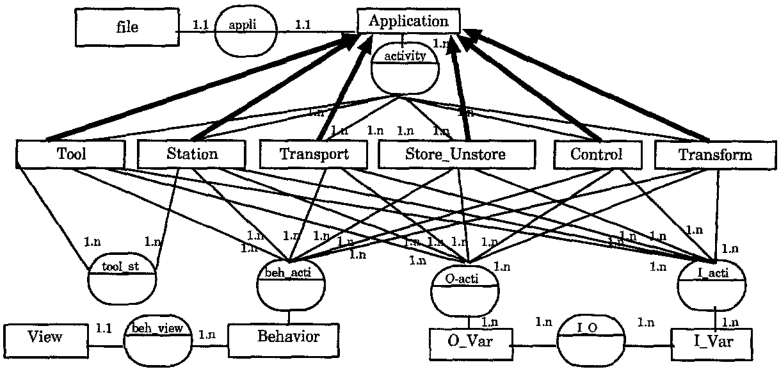


Fig. 3. The Automation Design Application Schema (step 2)

3.2 Tool Federation on top of PCTE

PCTE has been initially designed for software engineering needs. It is used in our architecture as a canonical data model in order to be experimented and evaluated for manufacturing requirements. It defines a repository in which data are stored and shared between tools. The choice of a standardized repository as a canonical data model is a key element of our architecture since it provides an open framework which homogenize different tool data representations making their integration and interoperation easier. Tools interoperability is achieved following the steps 3 to 7 of our integration methodology and is applied to the integration of SPEX and PROPEL tools by defining the five schema levels.

The Local Level

Data managed by SPEX and PROPEL were modeled, by a reverse engineering process, using an extended ERA data model independently from the tool integration repositories. We translate hereafter the conceptual schemas representing the data managed by the tools into their integration repository.

PROPEL integration repository is MATISSE. It is an object oriented database using an ERA like data model. Part and process plan conceptual schemas depicted respectively by figures 2.b and 2.d are translated to MATISSE data model. Each entity type is translated into an object class, each relationship into a link class and each attribute type into an attribute class. Cardinalities are kept unchanged. The resulting schemas are depicted by figures 4.a and 4.b.

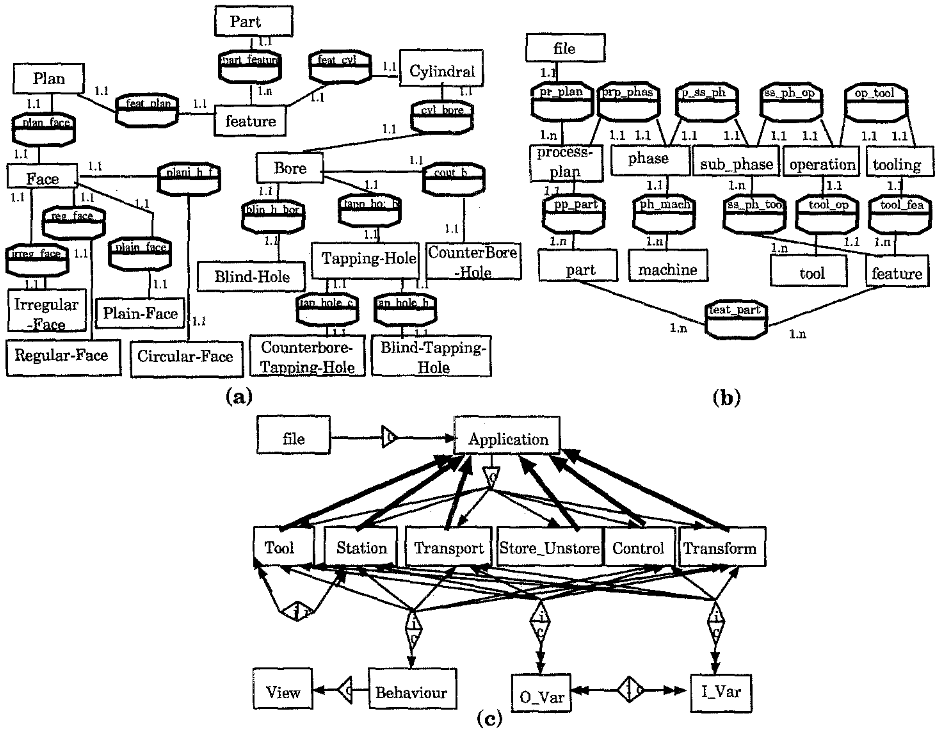


Fig. 4. PROPEL and SPEX local level (step 3)

SPEX automation design schema is translated into PCTE data model (figure 4.c). Each entity type is translated into an object type and each relationship type into a link type. The link category and attribute keys are defined by the designer. A formal definition of the mapping rules between the ERA and both MATISSE and PCTE data models is given in [8].

The Translated Level

We focus here on translating local schemas into the canonical data model corresponding to the fourth step of our integration approach. This translation is applied only to PROPEL schemas since the local schema (MATISSE) and the canonical data model (PCTE) are different. The correspondent translation rules transform each object class, link class and attribute class in MATISSE data model respectively into an object type, a link type and an attribute type in PCTE data model. Translated schemas of the part and the process plan are represented respectively by figures 5.a and 5.b. The cardinality of PCTE links doesn't traduce exactly the same semantics as MATISSE relationship cardinality since a cardinality *one* in PCTE represents a (0,1) or (1,1) relationship in MATISSE and a cardinality *many* in PCTE represents (0,n) or (1,n) in MATISSE. For a formal definition of the mapping rules between MATISSE and PCTE refer to [8].

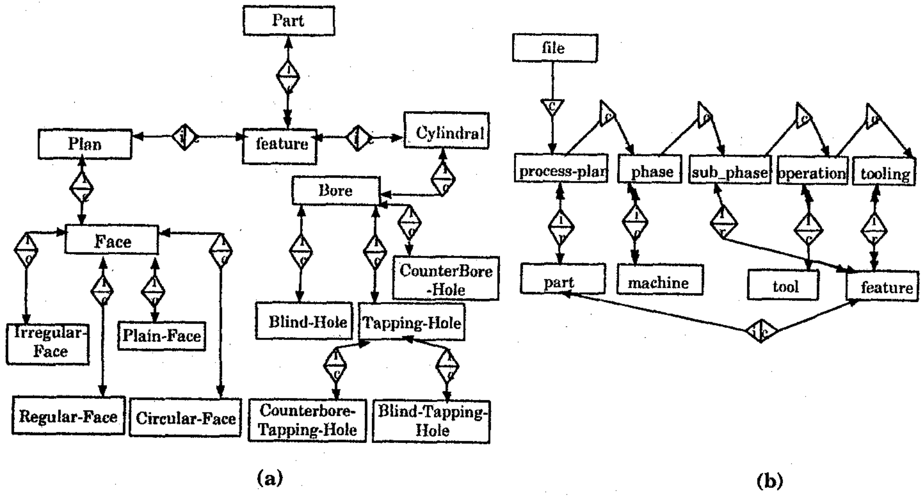


Fig. 5. Part and Process Plan Schemas in PCTE (step 4)

The Exported Level

The definition of the exported level from the translated level is carried out by defining a view on the translated schemas because not all translated schemas have to be seen and used by the federation. PCTE provides view definition thanks to the working schema (WS) mechanism. This mechanism stands for a filter since it allows to make visible only instances of object types it includes. The instances of object types which are not contained in the WS are not accessible and therefore invisible to the federated tools. Therefore, the exported schema should contain all tool schemas which have to be used by other tools of the federation. The inclusion of schemas and their order in the WS is managed by the user.

In our experiment, only the process plan and the automation design application schemas are included into the WS while the part schema remains local to PROPEL. The federation can therefore view instances of object types composing the process-plan and automation-design-application schemas.

The Federated Level

The sixth step consists in defining the federated level by integrating exported schemas into a federated schema. The integration process, as defined in [12], consists in solving, first, name and structure conflicts before defining common object type through which tools can cooperate. Finally, schema integration is processed by importing common object types and other related information which may be used by tools.

In PCTE, schema integration is achieved using importing mechanism provided by PCTE. From object and link types of exported schemas, we import object types common to schemas which have to be integrated. Importing these object types into the federated schemas allows tools to see, at the federated

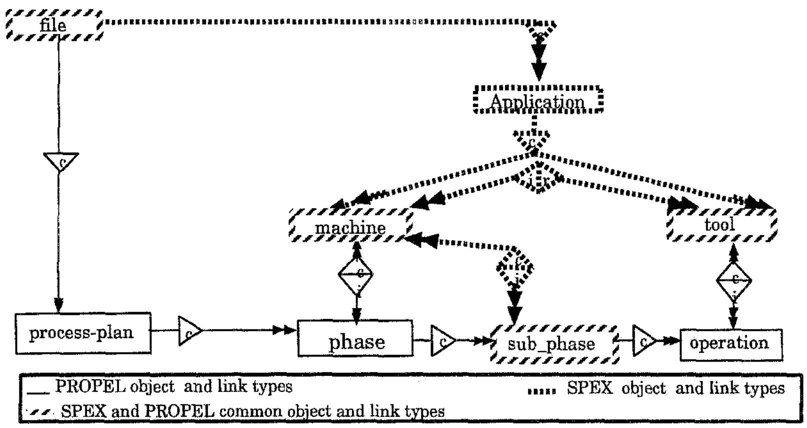

file
Application
machine
tool
process-plan
phase
sub_phase
operation
PROPEL object and link types
SPEX object and link types
SPEX and PROPEL common object and link types

Fig. 6. The Federated Schema (step 6)

level, object instances created at the local level by other tools. In our case (figure 6), we integrate exported part and automation design application schemas. Object types *Process_plan*, *phase*, *sub_phase* and *operation* and their links are imported from the process plan schema. Object type *Application* is imported from automation design application schema. Therefore, the integrated schemas share object types *machine* (called *Station* in automation design application schema), *tool* and *sub_phase*.

The External Level

The external schema, representing the seventh step, is obtained by defining a view on the federated schema. This view may correspond to a given application or a class of users which do not need to view all the information represented by the federated schema. The external schema allows data exchange between SPEX and PROPEL tools through their common object and link types (Figure 8). We have integrated into the federated schema, in addition to the information of

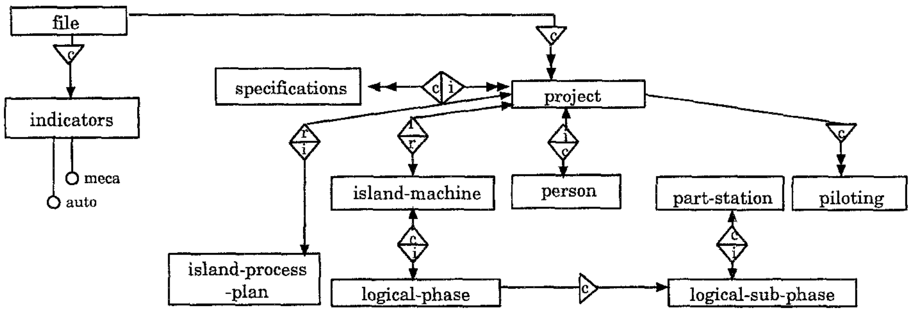


Fig. 7. The Auxiliary Management Schema

the process plan and automation design application schema, some management information related to the current project grouped in an auxiliary management schema (figure7). This schema represents the part, the manufacturing requirements (object type *requirements* containing a graphical representation of the currently manufactured part) and persons allowed to work on the project (object type *person*). We have also added to this schema a logical representation of the project which provides a link between logical and physical aspects of a project (object type *part_ station*), logical phase (object type *logical_ phase*) and sub-phase (object type *logical_ sub_ phase*). Object types *part* (called *project* in the management schema), *process-plan* (called *island-process-plan* in the management schema) and *application* (called *piloting* in the management schema) are common object types allowing the integration of the process plan, the automation design application and the management schemas.

External schema allows, thanks to the PCTE types importation mechanism to view object instances created at the local level. It is therefore possible, for any tool, to manipulate an object type instance of the external schema. Tools can access shared object type instances through which they can interoperate in an autonomous way by creating locally their instances and without any knowledge and management of other tool objects. The only information which should be known is the external schema. This one must always be included into tools WS to allow a local access to their objects and a federated access to other tool objects.

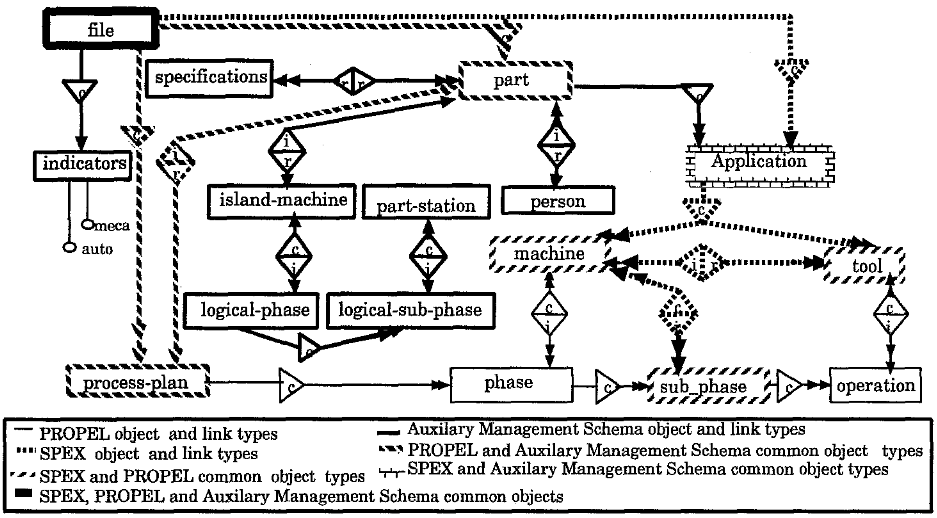

file
specifications
part
Application
indicators
island-machine
part-station
person
meca
auto
machine
tool
logical-phase
logical-sub-phase
process-plan
phase
sub_phase
operation
PROPEL object and link types
SPEX object and link types
SPEX and PROPEL common object types
SPEX, PROPEL and Auxilary Management Schema common objects
Auxilary Management Schema object and link types
PROPEL and Auxilary Management Schema common object types
SPEX and Auxilary Management Schema common object types

Fig. 8. The External Schema (step 7)

3.3 The Implemented Production Scenario

The objective of the production scenario depicted by figure 9 is to achieve parts design. The production scenario consists of two main steps which are the mechanical and the automation design steps managed by the management working station. This one allows the mechanical working station to begin its processing (1). From the information given by the part description file (2) (features and their physical and geometrical characteristics), we instantiate the MATISSE and PCTE part schemas (3) respectively at the local and the translated levels. PROPEL generates then, from the part description file and workshop machine and tool descriptions, the manufacturing process plan file (4). We process this file to extract information allowing the instantiation of MATISSE and PCTE process plan schemas (5) belonging respectively to the local and the translated levels. The end of the mechanical design step is notified to the management station through the external schema (6). This notification is considered by the management working station (7) which can now allow the automation station to start the automation design step (8).

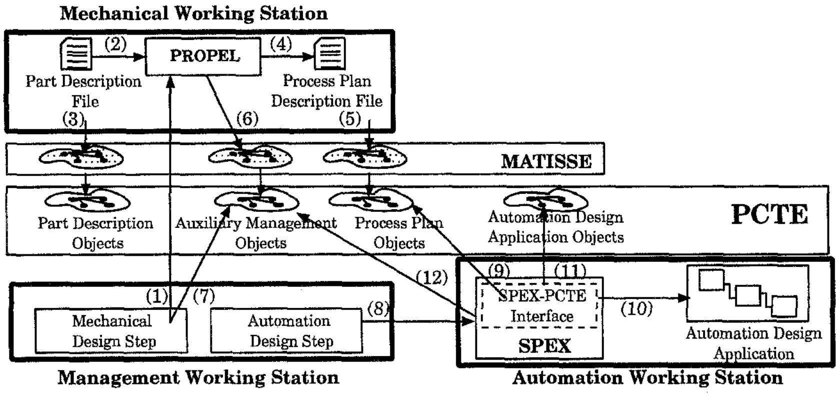

Mechanical Working Station
(2)
PROPEL
(4)
Part Description File
Process Plan Description File
(3)
(6)
(5)
MATISSE
PCTE
Part Description Objects
Auxiliary Management Objects
Process Plan Objects
Automation Design Application Objects
(12)
(9)
(11)
SPEX-PCTE Interface
SPEX
(10)
Automation Design Application
(1)
(7)
(8)
Mechanical Design Step
Automation Design Step
Management Working Station
Automation Working Station

Fig. 9. The Production Scenario

Automation design step takes results from mechanical design step (9) to build, on the one hand, a SPEX automation design application (10) and, on the other hand, a PCTE automation design application schema at the local level. We have extended SPEX in such a way that it can create, access and manipulate PCTE object type instances. This extension, which we have named SPEX-PCTE interface, generates a SPEX automation design application by counting *sub-phase* object type instances belonging to the PCTE process-plan schema at the external level instantiated during the mechanical design step. Considering that each process plan sub-phase corresponds physically to a transformation site, SPEX-PCTE module creates as much FDs as sub-phases. Created FDs are linked to each other using their input/output variables. Finally, a schema representing a PCTE automation design application is generated at the local level. This schema is compiled to be instantiated (11). The end of this step is notified to the management working station (12). All notifications are done using *meca* and *auto* notification attributes belonging to *indicators* object type in the external schema. Thanks to the importation and the WS mechanisms of PCTE, object instances created at the translated level are visible at the external level since both the process plan and the application design application schemas are included into the WS.

4 Lessons Learned and Future Work

This paper presents a federated approach to integrate tools in an heterogeneous environment. We have experimented this approach by integrating two different, but complementary tools whose integration was never anticipated.

The federated approach presents some lacks mainly related to the incompatibility between the different data models. We have to choose, on the one hand, a local data model which models the best tool data and, on the other hand, a canonical data model which has to be semantically rich enough to support different product modeling. These choices must limit the semantic losts when translating schemas from local to canonical data model. For our experiment, we got a favorable situation since both PCTE and MATISSE data models are ERA like models and therefore relatively homogeneous. This has made easier the translation between the two models and has limited semantic losts. From this point of view, a first conclusion is that the PCTE data model, close to the ERA is an acceptable canonic model, even if little semantic constraint types are handled by PCTE (we proposed in [15] an extension of the PCTE, inspired from the NIAM data model to enhance semantic constraint management).

Our second conclusion is that PCTE, due to its schema integration mechanism, provides an active support to federated schema building and mutually to federated database access. It is generally accepted that tool interaction depends on the object type granularity considered in the exchange schema. A coarse granularity limits the number of managed object types but can penalize tools interactions due to a non enough detailed view of managed objects. On the other side, fine granularity allows an efficient interaction between tools by enhancing schema semantics but increases the set of object types we have to deal with. PCTE is better adapted to manage coarse grain objects. However, it was sufficient in the context of our experimentation. In fact, it seems us that it is more important to decrease the granularity of the services extracted from tools. The redesign of tools, in order to extract internal services, even if it is an expansive task, can take benefits of services provided by PCTE to allow a better visibility of tool intermediate results and as a consequence to increase interactions and positive synergetic effects.

As perspectives, we plan to extend PCTE data model, initially designed for software engineering requirements, to cover other CIM requirements such as fine granularity object management and type instance management while evolving schemas. In addition, PCTE manages concurrent accesses using locking mechanisms and transactions. These mechanisms are efficient for short time transactions but are particularly unsuitable for long term design processes which can take several hours and even several days. We plan to integrate to DMMS architecture long term transactions such as those implemented in COO [16].

References

1. D. Notkin. The GANDALF Project. *Journal of Systems and Software*, 5(2):91–106, May 1985.
2. T. Reps and T. Taitelbaum. The Synthesizer Generator. In P. Henderson, editor, *Proc. ACM SIGSOFT/SIGPLAN Software Engineering Symposium on Practical Software Development Environments*, pages 42–48, May 1984.
3. S.P Reiss. PECAN: Program Development System that Support Multiple Views. *IEEE Transactions on Software Engineering*, SE-11(2):276–285, Mar. 1985.
4. C.W. Chung. DATAPLEX: an access to heterogeneous distributed databases. *Communications of the ACM*, 33(1), Jan. 1990.
5. V. Krishnamurthy, S.Y.W. Su, H. Lam, M. Mitchell, and E. Barkmeyer. A distributed database architecture for an integrated manufacturing facility. In Computer Society of IEEE, editor, *Proceedings of the International Conference on Data and Knowledge Systems for Manufacturing and Engineering*, pages 4–13, 1987.
6. A.P. Sheth and J.A. Larson. Federated Database Systems for Managing Distributed, Heterogeneous and Autonomous Databases. *ACM CS*, 22(3), Sep. 1990.
7. G. Morel and P. Lhoste. *Prototyping a Concurrent Engineering Architecture*, volume 1 of *TSI Press Series*, pages 163–167, 1994.
8. M. Bounab. Tool Integration in Heterogeneous Environments: Experimentation in a Manufacturing Framework. Phd Thesis, National Polytechnical Institute of Lorraine, Oct. 1994. (in french).
9. M. Lombard-Gregori. Contribution to discrete part manufacturing engineering : prototyping a concurrent engineering architecture for manufacturing integrated systems. Phd Thesis, University of Nancy I, Feb. 1994. (in french).
10. E.J. Chikofsky and J.H. Cross II. Reverse Engineering and Design Recovery: A Taxonomy. *IEEE Software*, pages 13–17, Janvier 1990.
11. P.P. Chen. The Entity-Relationship model: Toward an Unified View of Data. *ACM Transactions on Database Systems*, 1(1):9–36, Mar. 1976.
12. C. Batini, M. Lenzerini, and S.B Navathe. A Comparative Analysis of Methodologies for Databases Schema Integration. *ACM Computer Survey*, 18(4), 1986.
13. J.P. Tsang. Planification par combinaison de plans: application à la génération de gammes. PhD Thesis, National Polytechnical Institute of Grenoble, 1987. (in french).
14. H. Panetto, P. Lhoste, G. Morel, and M. Roesch. SPEX : Du Génie Logiciel pour le Génie Automatique. In *4th Int. Workshop: Software Engineering & its Applications*, pages 211–221, Toulouse (France), Dec. 1991.
15. M. Bounab, J. C. Derniame, C. Godart, and G. Morel. DMMS: A PCTE Based Manufacturing Environment. In *Proc. PCTE'93 Int. Conference*, pages 431–449, Nov. 1993.
16. C. Godart. COO: A Transaction Model to Support COOperating Software Developers COOrdination. In I. Sommerville and M. Paul, editors, *Proceedings 4th European Software Engineering Conference*, pages 361–379, Garmisch (Austria), Sept. 1993. Springer Verlag. Lecture Notes in Computer Science, Nř 717.