

CAD/CAPP Integration using Feature Ontology

Christel Dartigues *, Parisa Ghodous **, Michael Gruninger ***, Denis Pallez**, Ram Sriram***

*I3S UNSA-CNRS - 2000, route des lucioles, Les Algorithmes - bât. Euclide B, BP.121, 06903 Sophia Antipolis
– Cedex – France
dartigue@unice.fr

**LIRIS, University Claude Bernard of Lyon, 43, Boulevard du 11 novembre 1918
69622 Villeurbanne Cedex, France
{ghodous, dpallez }@liris.cnrs.fr

***NIST, 100 Bureau Drive, Gaithersburg, MD 20899-3460, USA
sriram@nist.gov, gruninger@mie.utoronto.ca

Abstract:

In a collaborative computer-supported engineering environment, the interoperation of various applications will need a representation that goes beyond the current geometry-based representation, which is inadequate for capturing semantic information. The primary purpose of this paper is to discuss a semantically-based information exchange protocol that will facilitate seamless interoperability among current and next generation computer-aided design systems (CAD) and between CAD and other systems that use product data. We describe an ontological approach to integrating computer-aided design (CAD) and computer-aided process planning (CAPP). Two commercial software applications are used to demonstrate our approach. The approach involves the development of a shared ontology and domain specific ontologies in the KIF (Knowledge Interchange Format) language. Domain specific ontologies--which are feature-based—are developed after a detailed analysis of the CAD and the CAPP software. Mapping between the domain ontologies and the shared ontology is achieved by several mapping rules. The approach is validated by using a variety of parts.

Keywords: CAD; CAPP; Interoperability; Ontologies; Design; Process Planning; Knowledge Interchange Format.

1. Introduction

The early part of this millennium has witnessed the emergence of an Internet-based engineering marketplace, where engineers, designers, and manufacturers from small and large companies are collaborating through the Internet to participate in various product development and marketing activities [1, 2, 3, 4]. This will be further enhanced by the next generation manufacturing environment, which will consist of a network of engineering applications, where state of the art multi-media tools and techniques will enhance closer collaboration between geographically distributed applications, virtual reality tools will allow visualization and simulation in a synthetic environment, and information exchange standards will facilitate seamless interoperation of heterogeneous applications. The interoperation of various applications will need a representation that goes beyond the current geometry-based representation, which is inadequate for capturing semantic information. The primary purpose of this paper is to discuss a semantically-based information exchange protocol that will facilitate seamless interoperability

among current and next generation computer-aided design systems (CAD) and between CAD and other systems that use product data. Our focus will be on design/process planning integration during the later design stages. We will then present in this paper an approach using a neutral format based on a feature ontology. Our work is then decomposed in three main phases that will be further explain in the rest of this paper, as shown in Figure 1:

- The analysis of the two domains studied: detailed design and process planning,
- The creation of the ontology and
- The definition and implementation of mapping rules.

In the next section we provide a brief overview of design/process planning integration. This will be followed by a discussion of representative standards for interoperating design and process planning. The need for ontological approaches is presented followed by descriptions of ontologies in the design and process planning domains and a common ontology. Rules for mapping from and to the common ontology are described. Finally, we illustrate our approach with an example.

```

;A feature is the super type of: volume feature, transition
;feature and feature pattern.
(forall (?a)
  (implies (feature ?a)
    (or (volume_feature ?a)
      (transition_feature ?a)
      (feature_pattern ?a))))

;A volume feature is the subtype of a feature.
(forall (?a)
  (implies (volume_feature ?a)
    (feature ?a)))
...
;Volume feature attributes are: associated volume and
;cutting section type.
(forall (?a)
  (implies (volume_feature ?a)
    (exists (?b ?c)
      (and (associated_volume ?b)
        (cutting_section_type ?c)
        (optional_attribute ?b ?a)
        (optional_attribute ?c ?a)
        (is_composed_of ?a ?b)
        (is_composed_of ?a ?c)))))
...

;A subtractive volume feature is a volume feature whose
;volume is subtracted from a pre-existing volume
(forall (?a)
  (implies (subtractive_volume_feature ?a)
    (and (volume_feature ?a)
      (exists (?b ?c)
        (and (associated_volume ?b)
          (is_composed_of ?a ?b)
          (pre_existing_volume ?c)
          (subtracted ?b ?c))))))
...
;A void is a subtype of a subtractive volume feature
(forall (?a)
  (implies (void ?a)
    (subtractive_volume_feature ?a)))
...

```

Figure 1: Global process for data exchange using ontologies

2. Design/Process Planning Integration: An Overview

Engineering a product involves several stages with considerable iterations, starting with planning products, generating product specifications, performing preliminary and detailed design, developing process plans, building product facilities, manufacturing product, managing workflow, and finally marketing and maintaining products [5]. In this paper we focus on an important aspect of the above cycle: design and process-planning integration. We believe that it is important to integrate design and process planning at various levels of abstraction, as errors made during early design stages could have a significant impact on the overall product quality and costs [5-8].

Engineering design involves mapping a specified function (or functional specifications) onto a (description of a) realizable physical structure – the design artifact. Over the past several decades considerable research has been done in developing various design product and process models [9]. We will not delve into a detailed description of the design process, much as we feel a need for the adequate representations for process knowledge. The reader is referred to [5] for a formal description of a

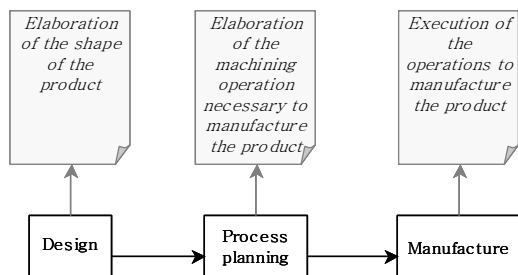


Figure 2: Role of design, process planning and manufacturing

design process model. At this stage our primary concern is on the product or artifact representation. For this we use the NIST CORE product model presented in [10].

Process planning is an intermediate phase between design and manufacture [11, 12]. More precisely, it links these two decisive phases of product development [13]. It depends on choices made in design and determines precisely actions that will be achieved during manufacture (Figure 2). Different definitions have been given for process planning [7, 14, 15, 16]. We use the following definition in this paper: process planning is the phase that, from information generated during preliminary design (product geometry for instance), determines necessary operations and actions to transform a raw part in a finished or semi-finished part, the necessary human and material resources to manufacture the product, as well as the product development cost evaluation.

A wide variety of manufacturing processes are available for the actual artifact production. In the current work we focus on the machining processes for part production, in particular material cutting processes. Figure 3 provides a representation of this process: the cutting tool comes against the surface, creating a chip that will be removed from the part.

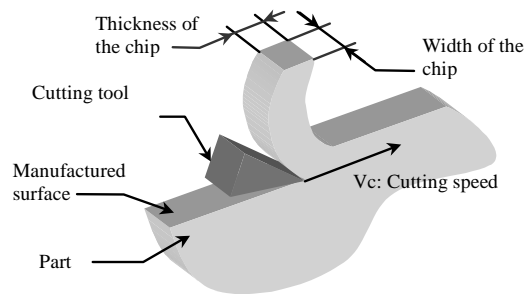


Figure 3: Representation of the material cutting process

The interactions between design and process planning occurs at various stages, from conceptual to detailed design/process planning as shown in Figure 4 [17].

Current interfaces between design and process planning are defined during the detailed design stage. This is primarily achieved through use of geometric features. However, there is considerable difference in the methods and terminology used: features are used to design a product (design by feature) [8, 13, 18, 19] while in process planning features are extracted from the product (design recognition or extraction) [13, 18, 19, 20, 21], and a consistent feature terminology does not exist for the two domains. These differences are illustrated in the software used by designers and process planners:

- CAD software, such as Pro-Engineer and SolidWorks, offer a limited number of features to users. The objective is here to have a compact set of parametric features, which can help designers to intuitively find more suitable features.
- CAPP software, such as PART, utilizes a feature extraction algorithm and contains a large number of features. The objective is not here to have a limited set of significant features but to have a very large number of features, which can improve the efficiency of the feature extraction algorithm.

These different viewpoints of designers and process planners on features makes data exchange a tedious task. Although features are considered differently in design and process planning, they represent a natural link between these two domains. Hence, features provide a valuable mechanism for information exchange. Next we review the current standards in design and process planning interoperability and discuss extensions needed for feature-based interoperability.

3. Standards for Interoperability

We illustrate the interoperability issue between CAD systems by considering a potential information exchange scenario during the design of the Boeing 777. For Boeing to incorporate Rolls Royce engines

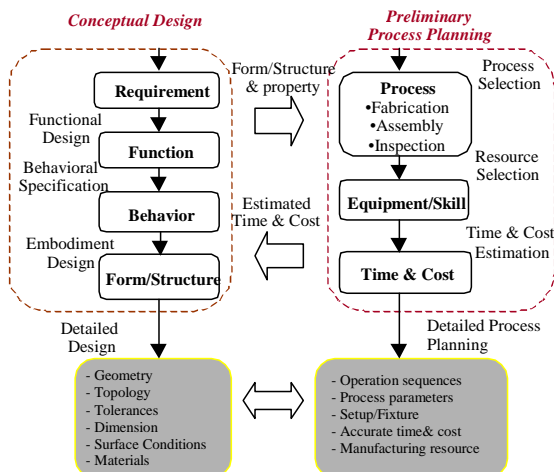


Figure 4: Design and process planning interfaces

into the design, the data format has to be converted from Computer Vision's CADDs (used by Rolls Royce) to Dassault's CATIA. Similarly, for Rolls Royce to understand changes made by Boeing engineers, the data need to be converted from CATIA to CADDs. Hence, we need at least 2 translators. For three systems this grows to 6 translators and for n systems we need $n(n-1)$ translators. Hence, there is a need to design, build, and maintain $n(n-1)$ translators. A solution to this problem is to use a neutral format and make all the CAD applications output this format. Doing so will reduce the number of translators to $2*n$, i.e., for each CAD system we will need two translators -- one from the CAD system to the neutral format and the other from the neutral format to the CAD.

A standard of primary interest to design is ISO 10303, also known informally as STEP [3, 34, 23] (Standard for the Exchange of Product model data) and developed by the International Organization for Standardization Technical Committee 184/ Subcommittee SC4 (ISO TC 184/SC4). Its intention is to enable the exchange of product model data between different modules of a product realization system, or the sharing of that data by different modules through the use of a common database [24]. The first parts of STEP to achieve International Standard status were published in 1994, but many other parts have since been published or are under development and will eventually be added to the standard. Recent updates (and other relevant details) can be found at the following websites: <http://www.nist.gov/sc4>, www.tc-184-sc4.org, and <http://www.iso.ch/iso/en/ISOOnline.frontpage>.

ISO 10303 (STEP) consists of many parts and can be viewed as consisting of several layers. The top layer consists of a set of application protocols or APs, which address specific product classes and life-cycle stages (e.g., mechanical, electronic, ships, automotive, design, process planning). These APs specify the actual data exchange, and are constructed from a lower layer set of modules called integrated resources, which are common for all disciplines. The language for modeling various STEP entities and their relationships is called EXPRESS. Other parts specify standard mechanisms for the actual transfer of data, the conformance testing methodology, and various test suites.

The STEP AP most relevant to traditional CAD systems is called AP 203 and is entitled "Configuration Controlled 3D Designs of Mechanical Parts and Assemblies." This protocol defines the data exchange of geometric entities and configuration control of products. AP 203 defines several levels of implementation -- called conformance classes -- which deal with increasing levels of sophistication.

The primary emphasis of STEP AP 203 is on shape description plus product configuration data. Facilities have been provided for capturing, in standard format, the following representations: 2D

drawings, 3D wireframes, surface models, and solid models. This reflects the state of CAD technology as it was when the STEP development effort commenced in the mid-1980s. However, CAD technology has progressed since that time, and most major CAD systems now provide facilities for parametric, variational (including constraints), and/or feature-based design. In addition many of these systems have facilities to record design histories. These systems generate additional information, beyond the pure shape descriptions created by older systems, and STEP AP 203 currently provides no means for capturing and transmitting this additional information. The short term parametrics effort (which comes under Working Group 12 of ISO TC 184/SC4) is addressing this problem.

Considerable research has been performed on mapping CAD data onto process planning systems. However, this work has met with limited success, such as the one reported by [25]. One problem with the current standards is the lack of integration between CAD data output and process planning input. For example, the primary focus of STEP AP 203 is the interoperability between geometry-centered CAD systems, while the focus of STEP AP 224 (Mechanical product definition for process plans using machining features) has been on input to process planning systems with a primary focus on representation of machine features. The idea of features has been in vogue for some time and the literature is abound with definitions of features [15, 19, 26-32]. For example, Shah et al. suggest that features “are primitive or low level designs with their attributes, qualifiers and restrictions which affect functionality and/or manufacturability. Features can describe form (size and shape), precision (tolerances and finishing), or materials (type, grade, properties and treatment), and vary with product and manufacturing process”.

To achieve truly collaborative design and engineering, exchange representations of both design and process information must support multiple levels of abstraction. To adequately achieve this we will need a more formal method for representing features, such as the ontological approach described in the next section. Our approach has some similarities to the one presented in [33], but our overall methodology is different.

4. Ontological Approach to Interoperability

In all types of communication, the ability to share information is often hindered because the meaning of information can be drastically affected by the context in which it is viewed and interpreted. This is especially true in manufacturing, because of the growing complexity of manufacturing

information and the increasing need to exchange this information among various software applications. Different representations of the same information may be based on different assumptions about the world, and use differing concepts and terminology -- and conversely, the same terms may be used in different contexts to mean different things. Often, the loosely defined natural-language definitions associated with the terms will be too ambiguous to make the differences evident, or will not provide enough information to resolve the differences.

To address these challenges, various groups within industry, academia, and government have been developing sharable and reusable models known as ontologies [3]. All ontologies consist of a vocabulary along with some specification of the meaning or semantics of the terminology within the vocabulary. In doing so, ontologies support interoperability by providing a common vocabulary with a shared semantics. Rather than develop point-to-point translators for every pair of applications, one simply needs to write one translator between the application's terminology and the common ontology. Similarly, ontologies support reusability by providing a shared understanding of generic concepts that span across multiple projects, tasks and environments.

The various ontologies that have been developed can be distinguished by their degree of formality in the specification of meaning. With informal ontologies, the definitions are expressed loosely in natural language. Semi-formal ontologies, such as taxonomies, provide weak constraints for the interpretation of the terminology. Formal ontologies use languages based on mathematical logic. Informal and semi-formal ontologies can serve as a framework for shared understanding among people, but they are often insufficient to support interoperability, since any ambiguity can lead to inconsistent interpretations and hence hinder integration.

Another source of semantic heterogeneity lies in the languages used to represent the ontologies. There have been several efforts within academia and industry to develop common languages that can be used as the basis for ontologies to support semantic integration; the most expressive is the Common Logic project, which combines the Knowledge Interchange Format [34-36] and Conceptual Graphs (CG) [37] languages. Common Logic includes a core language that has the expressiveness of first-order logic; its syntax and semantics are those of traditional first-order logic. Some other languages has been based on Logic, such as PSL [38]. Most recently, this has been extended to include extensions that allow sorted formulae for the specification of class hierarchies, and the specification of the meta theory of KIF within the language itself.

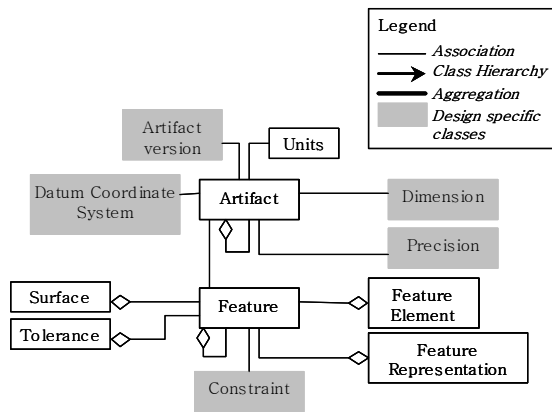


Figure 5: Design specific classes

Our objective in this paper consists in developing and implementing an approach for data exchange between designers and process planners. To realize this, we decided to develop a feature ontology. This ontology will represent all the common knowledge between designers and process planners as well as specific knowledge of both experts. We will use this ontology as depicted as follow: a designer creates an artifact shape model using a CAD software (such as Pro/Engineer); this model is then transform, using mapping rules (see Section 8), into instances of the shared ontology. These instances of the shared ontology are then transform, using other mapping rules, into a representation interpretable by CAPP software (such as Pro/Engineer). Features are then used not to realize features extraction or design by features: features represent for us a common knowledge that will be the base of our shared ontology for date exchange. In the next parts of this paper, we will present the design specific parts, the process planning specific parts and the design and process planning common parts of our ontology. We will continue with the description of the mapping rules used to translate data and we will finished by an example.

5. Design Feature Ontology

Our ultimate goal is to develop a comprehensive feature model that can be used through the entire design life cycle. However, for our prototype we restricted the NIST CPM's extensions to the

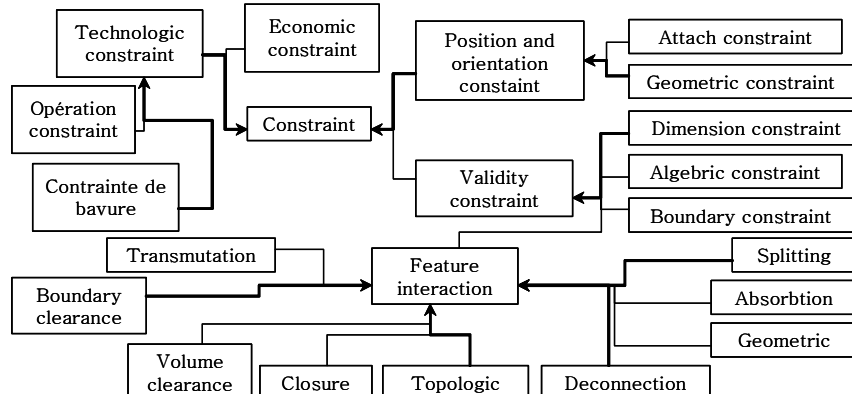


Figure 6: Constraint classification

information generated by commercial CAD systems. To identify these concepts, we first performed an extensive analysis to understand various designers' needs. This analysis phase involved:

- The extraction of designer know-how--which is implicit--in order to formalize designer's knowledge; and
- The analysis of different CAD software such as Pro-Engineer and SolidWorks: we used them to create various parts in order to better understand the design process.

Based on this analysis we concluded that the NIST CPM had most of the necessary classes to represent detailed design data. We added a few classes in order to increase the coverage to CAD software, such as: the datum coordinate system in which the artifact is defined, the dimensions associated to an artifact, the precision of the dimensions of an artifact, the different versions of an artifact and the constraints associated to each feature. Figure 5 represents these concepts.

We also defined different kind of constraints as shown in Figure 6. The initial categories that we considered are position and orientation constraints, which can be further classified into attachment and geometric constraints. Attachment constraints specify how a feature instance is attached to the global model by coupling some of the feature faces with the pre-existing faces. Geometric constraints specify geometric relations such as parallelism of two faces or distance between two faces. Validity constraints correspond to another constraint category defined in our ontology. These validity constraints can be further classified into:

- dimension constraints, which specify the authorized set of values for each feature parameter. e.g., radius parameter of a crossing hole can be limited to values between 1 and 10 millimeters;
- algebraic constraints, which are used when feature shapes are geometrically constrained with explicit relations (these relations can be simple equalities between two parameters or, in general, algebraic expressions implying two or more of two parameters or constants);
- boundary constraints, which specify if feature faces is on the boundary or not on the boundary of

```

:A constraint is the super type of: technologic constraint,
;economic constraint, validity constraint and position and
;orientation constraint.
(forall (?a)
  (implies (constraint ?a)
    (or (technologic_constraint ?a)
        (economic_constraint ?a)
        (validity_constraint ?a)
        (position_orientation_constraint ?a))))

:A technologic constraint is the subtype of a constraint.
(forall (?a)
  (implies (technologic_constraint ?a)
    (constraint ?a)))

:A validity constraint is the subtype of a constraint.
(forall (?a)
  (implies (validity_constraint ?a)
    (constraint ?a)))
...

```

Figure 7: KIF statements for constraint classification

the conceived object; and

- feature interaction constraints, which are used to indicate that a particular type of interaction is or is not allowed for a feature instance.

The above extensions suffice to illustrate our approach. Additional classes will be needed for a wider coverage. KIF representations of a representative set are shown in Figure 7.

6. Process Planning Ontology

Our feature ontology is also representative of the process planning viewpoint. We followed a similar approach used for design: we asked process planners to describe how they work, what kind of information they need, what are the different phases of their work, etc. We also studied a CAPP software: PART. This analysis of process planning turned out to be a more difficult task than obtaining the design features. While designers have a consistent notion of what design is, process planners seem to be in less agreement on the terminology in their domain. Based on our discussions, we decided to use the concepts presented in Figure 8.

In this figure, an artifact is associated with a manufacturing model. This model is used to create a

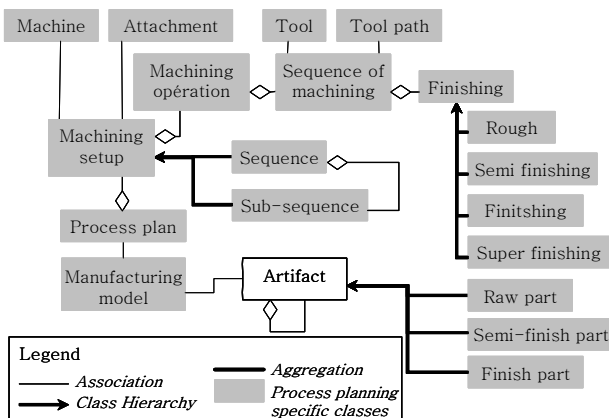


Figure 8: Process planning specific classes

process plan. The input of this process plan is a raw part and the output is a semi-finished or finished part. A process plan identifies the machining operations that are necessary to manufacture an artifact. Hence, a process plan is composed of machining setups, which contains all the machining operations that are realized with the same machine and without changing the attachments. For each machining setup, there is a set of machining operations. Each machining operation is then realized with the same machine and attachments. Each machining operation is composed of a set of machining sequences, which corresponds to a transformation of a part that is achieved with the help of a material removal tool moving according to a tool path. Finally, a machining operation modifies a surface in accordance to a required finish: raw, semi-finish, finish or super-finish. KIF representations of a representative set are shown in Figure 9.

7. Common Feature Ontology

The last part of our ontology corresponds to the common concepts between design and process planning and is composed of numerous classes and relationships. We base our ontology on the NIST Core Product Model (CPM). We used this model in order to take into account general concepts, initially present in this model, and we added more specific concepts allowing feature representation. (Figure 10) represents the main classes and relationships composing the Core Product Model and its extensions in this work, where the extensions are shown as darkened boxes (ideally, the NIST CPM should be a package in UML and our extensions should be in a separate package). The descriptions of key entities in the NIST CPM are as follows (taken from [10]).

An **Artifact** represents a distinct entity in a design, whether that entity is a component, product, subassembly or assembly. The **Artifact's** attributes refer to the **Specification** responsible for the Artifact and the **Form, Function, and Behavior** comprising the Artifact. The **Function** represents what the

```

:A manufacturing model uses a process plan
(defrelation use (?a ?b):=
  (and (manufacturing_model ?a)
        (process_plan ?b)))

;Attributes of a process plan are: a set of machining setup,
;an associated manufacturing model and specifications
(forall (?a)
  (implies (process_plan ?a)
    (exists (?b ?c)
      (and (associated_manufacturing_model ?b)
            (specification ?c)
            (machining_setup ?l)
            (is_composed_of ?a ?b)
            (is_composed_of ?a ?c)
            (is_composed_of ?a ?l))))))
...

```

Figure 9: KIF statements for feature decomposition

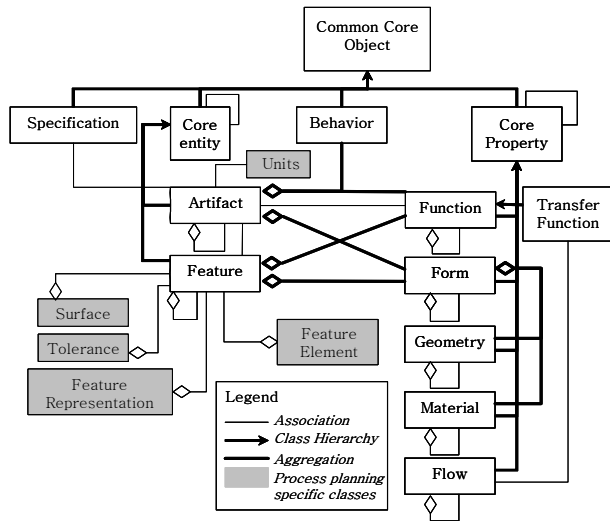


Figure 10: Main class diagram of the Core Product Model and extensions

Artifact is supposed to do. The **Artifact** satisfies the engineering requirements largely through its **Functions**. The term function is often used synonymously with the term *intended behavior*. The **Form** of the **Artifact** can be viewed as the proposed design solution for the design problem specified by the **Functions**. More precisely, the physical characteristics of an **Artifact** are represented in terms of its **Geometry** and **Material** properties.

Another important class of the Core Product Model is the **Feature**. An **Artifact** is composed of a set of features, where a feature is a subset of the form of an object that has some function assigned to it. We can have several types of features: analysis features, design features, manufacturing features, interface or port features, etc., Compound features can be generated from primitive features. The notion of a feature is further elaborated in the work presented here.

We modified the NIST Core Product Model (CPM) by adding some concepts that are common to design and process planning, are both necessary for designers and process planners, and are considered in CAD and CAPP software. Examples of these include:

- The surfaces composing any feature;

- The tolerances associated to any feature (such as the perpendicularity between two surfaces) (a more complete treatment of tolerances and assemblies is provided in [39]); and

- The units used to represent any artifact.

Our main objective is to find a common feature representation between design and process planning. To do so, we extended NIST CPM to address the following:

- The way each feature is represented, such as a B-Rep representation, a CSG representation, a swept representation, etc. (Feature Representation concept); and
- The elements composing each feature, such as a bottom side, an intermediary face, etc. (Feature Element concept).

We also characterized a complete feature decomposition which is based on the feature categories proposed in the part 48 of STEP [40]. Figure 11 illustrates this decomposition. Features are classified into:

- Volume features, which are viewed as a volume added to or subtracted from pre-existing volume;
- Transition features, which are viewed as separating or blending two or more surface elements; and
- Pattern features, which are viewed as a consisting of a number of identical sub features arranged in a mathematical pattern.

Volume features can be subtractive or additive, and transition feature can be corner or flat transitions. A more detailed description of this decomposition can be seen at [40, 41].

The KIF version of representative entities is shown below (Figure 12).

8. Mapping Rules For Case Study

Once the feature ontology in various domains is defined, the next step is to define the mapping rules that will transform specific files onto instances of our common ontology. For our case study, we choose the following software: Pro-Engineer, which is used by CAD experts, and PART, which is used

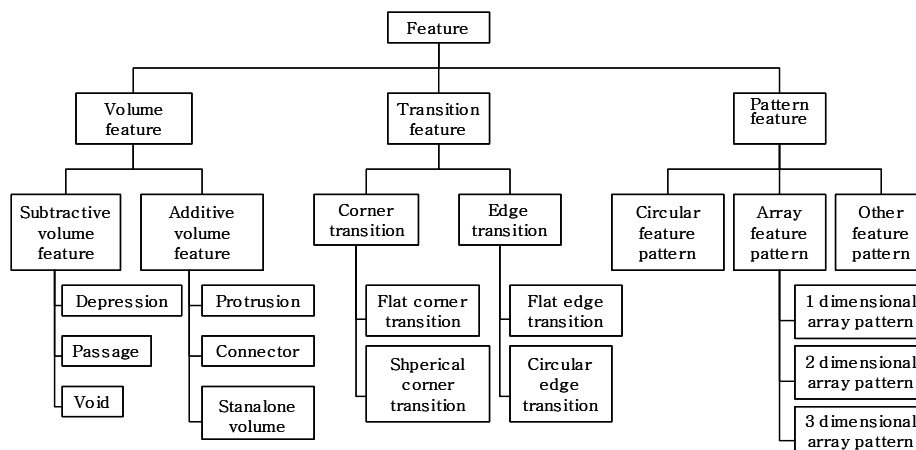


Figure 11: Feature decomposition

```

:A feature is the super type of: volume feature, transition
;feature and feature pattern.
(forall (?a)
  (implies (feature ?a)
    (or (volume_feature ?a)
        (transition_feature ?a)
        (feature_pattern ?a))))

:A volume feature is the subtype of a feature.
(forall (?a)
  (implies (volume_feature ?a)
    (feature ?a)))
...
:Volume feature attributes are: associated volume and
;cutting section type.
(forall (?a)
  (implies (volume_feature ?a)
    (exists (?b ?c)
      (and (associated_volume ?b)
            (cutting_section_type ?c)
            (optional_attribute ?b ?a)
            (optional_attribute ?c ?a)
            (is_composed_of ?a ?b)
            (is_composed_of ?a ?c))))))
...

:A subtractive volume feature is a volume feature whose
;volume is subtracted from a pre-existing volume
(forall (?a)
  (implies (subtractive_volume_feature ?a)
    (and (volume_feature ?a)
        (exists (?b ?c)
          (and (associated_volume ?b)
                (is_composed_of ?a ?b)
                (pre_existing_volume ?c)
                (subtracted ?b ?c))))))
...
:A void is a subtype of a subtractive volume feature
(forall (?a)
  (implies (void ?a)
    (subtractive_volume_feature ?a)))
...

```

Figure 12: KIF statements for feature decomposition

by CAPP experts. The methodology that we followed is described in Figure 13.

We first analyzed the existing export and import formats of Pro-Engineer and PART. Then, we selected one format for each of them: a proprietary format for Pro-Engineer, Neutral File Format, and ACIS format for PART. Once the formats have been chosen, we analyzed the representation of different artifacts in the two formats. The objective is to extract all the important concepts represented in each file in order to correlate them with the domain ontology entities. Once this is done our approach utilizes two algorithms: one to translate a file generated by a CAD software into a set of instances of the feature ontology and one to translate this generated file into a file that can be interpreted and processed by a CAPP software. The inputs to the first algorithm are:

- The file containing the entire description of the ontology, which is expressed in KIF, and
- The file generated by the CAD software (Pro/Engineer in this case), which represents the geometry and topology of the part that has to be manufactured.
- The inputs to the second algorithm are:
- The file containing the entire description of the ontology, which is the common ontology expressed in KIF, and
- The file generated by the first algorithm.

As we previously stated, the only assumption made during the elaboration of the ontology and the mapping rules was that we only considered parts that do not have any assembly; solving this problem for simple machining parts containing only features by itself is a difficult task. Taking into account more complex parts containing for example assemblies

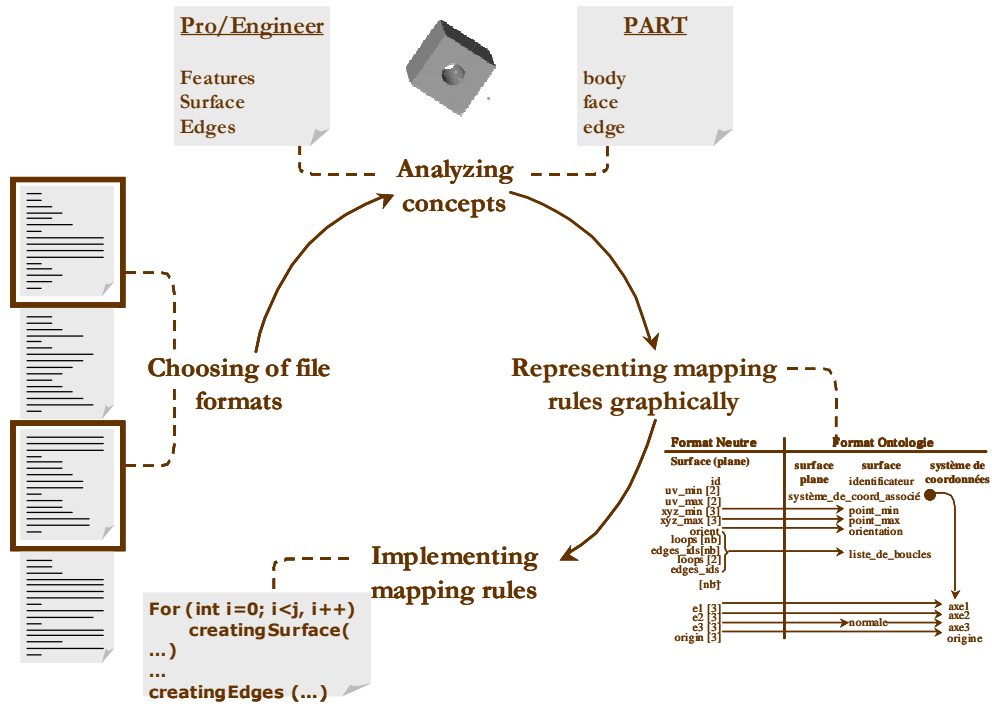
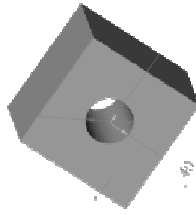


Figure 13: Methodology for the elaboration of the validation prototype

Pro/Engineer file

```
#- HOST
#- VERS 0 0
...
1 dimensions [8]
2 dimensions
3 name d0
...
1 features [5]
2 features
...
2 features
# Protrusion
3 id 47
3 user_name NULL
...
1 surfaces [8]
2 surfaces
3 id 50
3 uv_min [2]
...
1 edges [18]
2 edges
3 id 51
...
```



PART file

```
700 0 1 0
24 PART 6.1.000 07-JUN-2002 13 ...
-0 body $-1 -1 $-1 $1 $-1 $2 #
-1 lump $-1 -1 $-1 $-1 $2 $0 #
-2 shell $-1 -1 $-1 $-1 $3 $-1 $1 #
-3 face $-1 -1 $-1 $4 $5 $2 $-1 $6 reversed single #
-4 face $-1 -1 $-1 $7 $8 $2 $-1 $9 forward single #
-5 loop $-1 -1 $-1 $10 $11 $3 #
-6 cone-surface $-1 -1 $-1 -52.5 -25 129 0 0 1 -13 011 0 111 0 113 forward 1111 #
-7 face $-1 -1 $-1 $12 $13 $2 $-1 $14 reversed single #
-8 loop $-1 -1 $-1 $15 $16 $4 #
```

Figure 14: Data declaration in Pro-Engineer and PART files for a simple artifact

would implies to modify both the ontology and the mapping rules. If the two algorithms that we developed do not provide a “correct mapping” then:

- The mapping rules implemented in the algorithms are not correct; and/or
- The ontology is not correct, in which case, we have to modify the ontology and then modify the mapping rules in order to take into account these changes.

For a simple artifact such as a box with one hole (Figure 14), the file generated by Pro-Engineer is hierarchically structured: it contains the dimensions characterizing the artifact, the features used to build it, the surfaces determining the features and the edges composing the surfaces. PART files are totally different: information is stored with no specific order, and data contained in such files relates to geometric and topologic information. This kind of file format doesn’t explicitly provide information about features composing an artifact.

Using different instances of Pro-Engineer and PART files, we extracted a list of entities or concepts and their attributes in these files. Example concepts are: plane surface, cylindrical surface, straight curve, linear curve, edge, point, vertex, etc. Once this analysis is done, we elaborated the mapping rules between a Pro-Engineer file and a file containing instances of our ontology and between this generated file and a PART file. The objective of these rules is to identify in our domain ontologies the entities that are equivalent to the concepts that we identified in Pro-Engineer and PART files. Initially we expressed these mapping rules graphically. In this graphical representation, we display the relationships between the attributes of the entities represented in Pro-Engineer or PART file and attributes of the entities of our ontology. Figure 15 shows the graphical representation of one such mapping rule. This mapping rule shows the correspondence between a plane surface expressed

in a neutral file generated by Pro-Engineer and the equivalent concepts in our ontology.

Once this step is finished, we obtained two sets of mapping rules. These mapping rules are expressed graphically. The next step consists in implementing these rules in order to be able to translate a CAD file into a CAPP file via our ontology. As we have previously stated, our method involves starting from a file generated by Pro/Engineer, applying a first set of mapping rules in order to generate a neutral file, and applying our second set of mapping rules on this neutral file in order to obtain a file interpretable by PART. A description of our mapping algorithm is shown in Figure 16.

Starting from a CAD file generated by Pro/Engineer, we create all the features. For each feature, we have to extract from our ontology all the attributes that we have identified for a feature (for example the list of surfaces, the list of dimensions, etc.). For each of these attributes we search, still in our ontology, the nature of the attribute, which can be either simple (i.e., integer, string, boolean) or

Neutral format	Ontology format		
Surface (plane)	plane surface	surface	coordinate system
id	identificateur		
uv_min [2]	associated_coordinate	stem	sy
uv_max [2]			
xyz_min [3]	point_min		
xyz_max [3]	point_max		
orient	orientation		
loops [nb]			
edges_ids[nb]			
loops [2]	loop_list		
edges_ids [nb]			
...			
surface_type 34			
surface (plane)			
e1 [3]			axis1
e2 [3]			axis2
e3 [3]	normal		axis3
origin [3]			origin

Figure 15: Correspondence for plane surface between a Neutral file and our ontology

Create a feature instance

Find all feature attributes

For each feature attribute do

Find the type of the attribute

Create an instance of the attribute

If the attribute is a simple one (integer, string, etc.)

Find its value in the initial file (Pro-Engineer or PART file)

Else find its sub-attributes

For each sub-attribute do

...

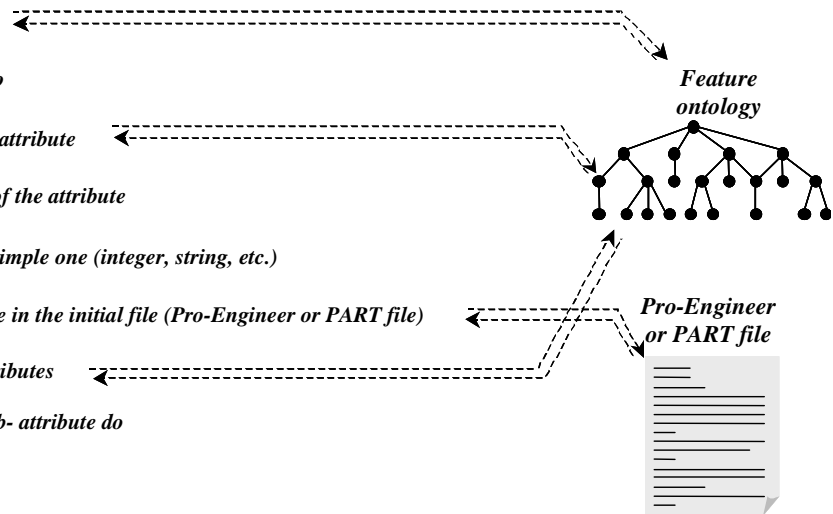


Figure 16: General algorithm for data exchange complex (i.e., the attribute is composed of sub-attributes). If the attribute is a simple one, we extract in the initial file the associated value and we add a new instance in the neutral file. If the attribute is more complex, we consider each sub-attribute until all concepts appearing in the initial file have been instantiated. The advantage of this algorithm is that if we decide to change the attributes of one of the concepts of the ontology –for example if we delete one attribute of the concept feature- the algorithm will not have to be change because the number of attributes of a concept is calculated each time the algorithm is running.

We also implemented a user interface for our prototype. This interface allows us to visualize the different artifacts that we considered, the Pro-

Engineer file, the file generated by our prototype and containing instances of our ontology, the file in which the ontology is coded and also a graphical representation of the artifacts that we considered. Figure 17 shows a typical screen of our prototype, which is implemented at University Claude Bernard of Lyon, FRANCE.

We tested our methodology and our prototype with different examples. For our initial prototype we considered only simple parts (see Figure 18), with great success. Our plans are to extend this work for complex artifacts (e.g., assemblies).

The result of our approach is shown in the Figure 19.

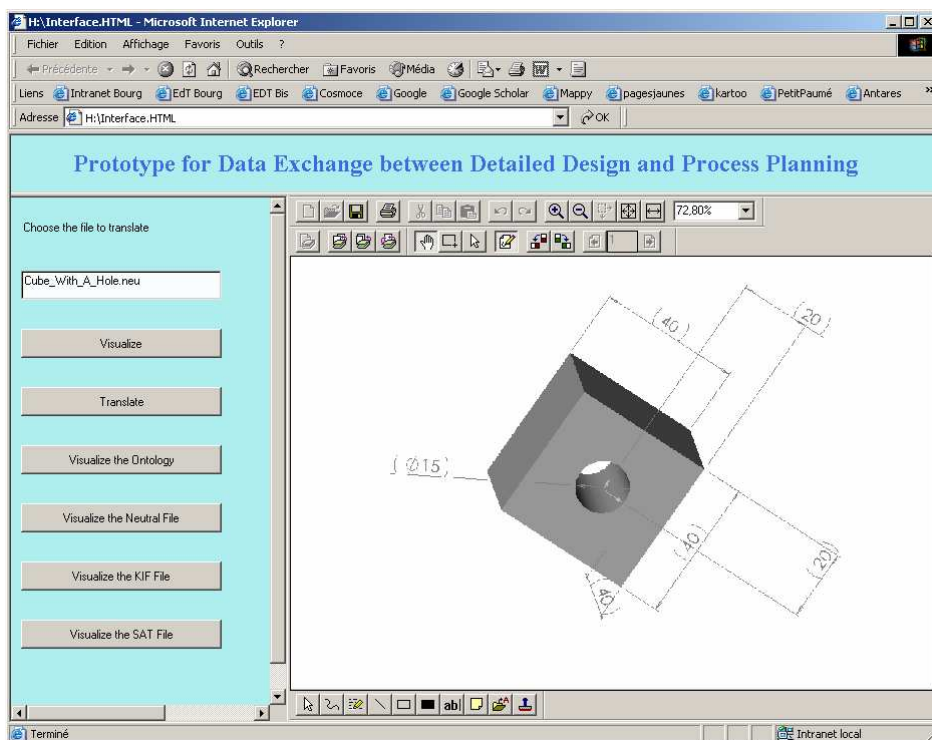


Figure 17: Screen of the prototype to visualize a file

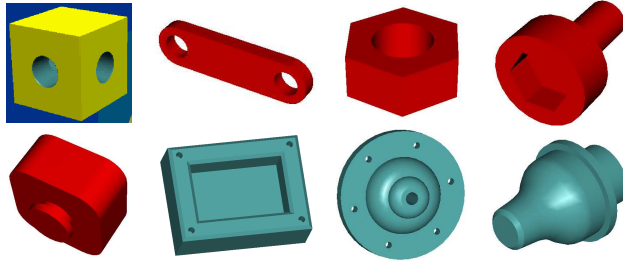


Figure 18: Some part examples

Summary

In this paper we have described an ontological approach to integrating computer-aided design (CAD) and computer-aided process planning (CAPP). Two commercial software applications were used to demonstrate our approach. The approach involved the development of a shared ontology and domain specific ontologies in the KIF (Knowledge Interchange Format) language. Domain specific ontologies--which were feature-based--were developed after a detailed analysis of the CAD and the CAPP software. Mapping between the domain ontologies and the shared ontology was achieved by several mapping rules. The approach was validated

by using a variety of parts.

Disclaimer

No approval or endorsement of any commercial product by the National Institute of Standards and Technology or by University Claude Bernard of Lyon is intended or implied. Certain commercial equipments, instruments, or materials are identified in this report in order to facilitate better understanding. Such identification does not imply recommendations or endorsement by the National Institute of Standards and Technology or by University Bernard of Lyon, nor does it imply the materials or equipment identified are necessarily the best available for the purpose.

Bibliography

1. Eynard, B., Liénard, S., Charles, S. and Odinot, A. (2005). Web-based Collaborative Engineering Support System: Applications in Mechanical Design and Structural Analysis, CERA Journal, Volume 13, Number 2, June 2005.
2. Jia, H.Z., Fuh, J.Y.H., Nee A.Y.C. and Zhang, Y.F. (2002). Web-based Multi-functional Scheduling System for a Distributed

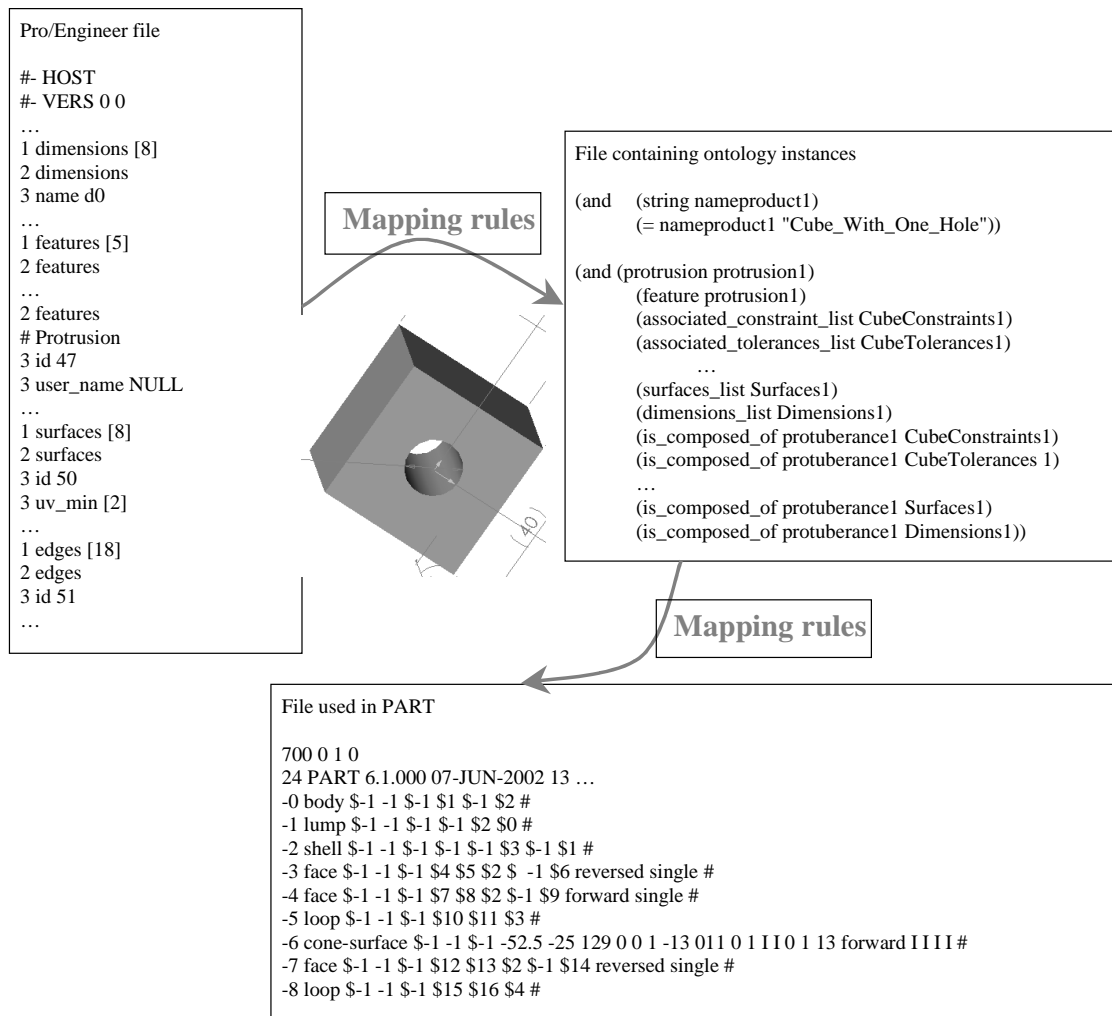


Figure 19: Examples of the different generated files

- Manufacturing Environment, CERA Journal, Volume 10, Number 1, March 2002.
3. Han, S., Choi, Y. Yoo, S. and Park, N. (2002). Collaborative Engineering Design Based on an Intelligent STEP Database, CERA Journal, Volume 10, Nuber 3, September 2002.
 4. Huang, G. Q. and Mak, K. L. (2002). Agent-based Collaboration Between Distributed Web Applications: Case Study on "Collaborative Design for X" Using CyberCO, CERA Journal, Volume 10, Number 4, December 2002.
 5. Sriram, R.D. (2002). Distributed and integrated collaborative engineering design. Sarven Publishers.
 6. Baker, B.A., Fish, R.D. and Cohen E. (2000). Using a multiple concurrent design views interface to enhance design complexity management. ASME Design Engineering Technical Conferences (DETC), Baltimore, Maryland, USA.
 7. Feng, S.C. and Song E.Y. (2002). Preliminary Design and Manufacturing Planning Integration Using Intelligent Agents. Proceedings of the Seventh International Conference on CSCW in Design, Rio de Janeiro, Brazil.
 8. Jan de Kraker, K. (1998). Feature Conversion for Concurrent Engineering, in Computer Science, Delft University of Technology: Delft, The Netherlands.
 9. Dym, C. (1994). Engineering Design: A Synthesis of Views, edition C.U. Press.
 10. Fenves, S.J. (2001). A core product model for representing design information, National Institute of Standards and Technologies (NIST): Gaithersburg, MD, USA.
 11. Algeo, M.E.A., Feng, S.C. and Ray S.R. (1994). A State-of-the-Art Survey on Product Design and Process Planning Integration Mechanisms, National Institute of Standards and Technology (NIST).
 12. Ball, M., Baras, J., Lin, E., Minis, I., Nau, D. and Karne, R. (1998). Integrated Product and Process Design Tool for Microwave Modules. in Integrating Product Design and Production: Designing for Time-to-Market, NSF Design and Manufacturing Grantees Conference.
 13. Han, J.H. and Requicha A.A.G. (1995). Integration of feature based design and feature recognition, Proceedings of ASME 15th International Conference in Engineering Conference, Boston, MA, USA.
 14. Kramer, T.R., (1987). Process plan expression, generation, and enhancement for the vertical workstation milling machine in the automated manufacturing research facility at the National Bureau of Standards, National Bureau of Standards: Gaithersburg, MD, USA.
 15. Regli, W.C. and Pratt M. (1996). What are Feature Interactions?, Proceedings the 1996 ASME Design Engineering Technical Conference and Computers in Engineering Conference, Irvine, California, USA.
 16. Houten, F.J.A.M.v., Erve, A.H.v.t. and K. H.J.J. (1989), PART a Feature Based CAPP System, 21st CIRP International Seminar on Manufacturing Systems, Stockholm, Sweden.
 17. Feng, S.C. and Song E.Y. (2000). Information Modeling on Conceptual Design Integrated with Process Planning. in Proceedings of Symposia for Design For Manufacturability, , the 2000 International Mechanical Engineering Congress and Exposition (IMECE). 2000. Orlando, FL, USA.
 18. Han, J.-h., Pratt, M. and Regli, W.C. (2000). Manufacturing Feature Recognition from Solid Models: A Status Report, Transactions on Robotics and Automation, Volume 16, Issue 6, p. 782-796.
 19. Han, J.H. (1996). Survey of Feature Research, Department of Computer Science and Institute for Robotics and Intelligent Systems, University of Southern California, Los Angeles, CA, USA.
 20. De Martino, T., Falcidieno, B. and Hassinger, S. (1998). Design and engineering process integration through a multiple view intermediate modeller in a distributed object-oriented system environment, Computer-Aided Design, Volume 30, Issue 6, p. 437-452.
 21. De Martino, T., Falcidieno, B., Giannini, F., Hassinger, S. and Ovtcharova J. (1994). Feature-based modelling by integrating design and recognition approaches, Computer-Aided Design, Volume 26, Issue 8, p. 646-653.
 22. Junhwan, K. and Soonhung, H. (2004). Manipulating Geometry in a STEP DB from Commercial CAD Systems, CERA Journal, Volume 12, Number 1, March 2004.
 23. Stephen, C.F.C., Tharam, D. and Vincent T.Y.N. (2003). Exchanging STEP Data Through XML-based Mediators, CERA Journal, Volume 11, Number 1, March 2003.
 24. Balakrishna, A., Suresh Babu, R., Nageswara Rao, D., Ranga Raju, D. and Kolli, S. (2006). Integration of CAD/CAM/CAE in Product Development System Using STEP/XML, CERA Journal, Volume 14, Number 2, June 2006.
 25. Dereli, T. and Filiz, H. (2002) A note on the use of STEP for interfacing design to process planning, Computer-Aided Design, Volume 34, p. 1075-1085.
 26. Bidarra, R. and Bronsvoort, W.F. (2000). Semantic feature modelling, Computer-Aided Design, Volume 32, p. 201-225.
 27. Bronsvoort, W.F., Bidarra, R. and Noort, A. (2001). Semantic and multiple-view feature modelling: towards more meaningful product modelling, Geometric Modelling, Theoretical and Computational Basis towards Advanced CAD Application, F. Kimura, Editor, Kluwer Academic Publishers, Boston. p. 384.
 28. Mäntylä, M., Nau, D. and Shah, J.J. (1996). Challenges in Feature-Based Manufacturing Research, Communication of the ACM, Volume 39, Issue 2, p. 77-85.
 29. Pratt, M. and Srinivasan, V. (2003). Towards a Neutral Specification of Geometric Features.
 30. Salomons, O.W., Houten, F.J.A.M.v. and Kals, H.J.J. (1993). Review of research in feature based design, Journal of Manufacturing Systems, Volume 12, Issue 2, p. 113-132.
 31. Shah, J.J. and Mäntylä, M. (1995). Parametric and feature-based CAD/CAM, ed. J.W.s. Inc., New-York, USA.
 32. Shah, J.J. and Rogers, M.T. (1988). Feature Based Modeling Shell: Design and

Implementation, Proceedings of the ASME Conference on Computers in Engineering.

33. Collet, C., Huhns, M.N. and Shen, W. (1991). Ressource Integration Using a Large Knowledge Base in Carnot, IEEE Computer, p. 55-62.

34. Hayes, E.E. and Menzel, C.P. (2001). A Semantics for Knowledge Interchange Format, Working Notes of the IJCAI-2001 Workshop on the IEEE Standard Upper Ontology, Seattle, Washington, USA.

35. Genesereth, M. and Fikes, R. (2001). Knowledge Interchange Format, Version 3.0 Reference Manual, Computer Science Department, Stanford University, Stanford, CA, USA.

36. Genesereth, M.R. (2000). Knowledge Interchange Format, American National Standard (dpANS).

37. Sowa, J.F., (2000). Glossary.

38. Cutting-Decelle, A. F., Young, R. I. M., Anumba, C. J., Baldwin A.N. and Bouchlaghem, N. M. (2003). The Application of PSL to Product Design Across Construction and Manufacturing, CERA Journal, Volume 11, Number 1, March 2003.

39. Sudarsan, R., Han, Y., Feng, SC, Roy, U., Wang, F., Sriram, RD and Lyons, KW (2003). Object-Oriented Representation of Electro-Mechanical Assemblies Using UML, National Institute of Standards and Technologies (NIST), Gaithersburg, MD, USA.

40. ISO10303-48 (1992). STEP Product Data Representation and Exchange, Integrated Generic Ressources: Form Feature, International Organisation for Standardization, Subcommittee 4, Part 48, NIST.

41. Dartigues, C. (2003). Product data exchange in a collaborative environment, Computer science, LIRIS - University of Lyon I, Lyon, France, p. 179.

Denis PALLEZ is currently an associate professor in the department of computer sciences at the University of Lyon, France, in the LIRIS laboratory (Lyon Research Center for Images and Intelligent Information Systems). He obtained his PhD in computer sciences in January 2000 at the University of Metz (France). He also spent four months as an invited researcher at the institute of technology at Montreal (ETS). His research interest is in the area of conceptual and functional design, shape synthesis, collaborative modelling and more recently, Interactive Evolutionary Computation.

