J Syst Eng (1995)5:48–59


Journal of Systems Engineering

# Specification for a Process Planning Enabling Platform

V. Hetem, K. Carr, M. Lucenti, O. Ruiz, X. Zhu, P.M. Ferreira and S.C.-Y. Lu

Department of Mechanical and Industrial Engineering, University of Illinois at Urbana-Champaign, Urbana, IL 61801, USA

*The objective of this research is the specification of an enabling platform for process planning system development. This work was done under a contract issued by Computer Aided Manufacturing-International to the University of Illinois. An analysis framework is developed that integrates inter-related process planning activities by transforming inputs and controls through mechanisms into outputs. An object-oriented approach is used to define the object model and thereby facilitate the configuration of a process planning system by providing standardised data objects and fundamental library routines. The proposed software structure is layered with specific modules designed to support integration and data exchange. The development domain is machining and includes part understanding, process selection and ordering, equipment specification, setup and operation planning, manufacturability analysis, and plan evaluation and documentation.*

**Keywords:** Enabling platform; Process planning; Specification

## 1. Introduction

Computer-Aided Process Planning (CAPP) is a critical link between Computer-Aided Design (CAD) and Computer-Aided Manufacturing (CAM). In spite of roughly three decades of research in this area, success in the development of a viable process planning system has been elusive. The need for enhanced process planning abilities is emphasised by the developments in concurrent product and process design, and design for manufacturability. To address the multi-perspective problem-solving approach required of the overall task of process planning, most researchers have typically addressed arbitarily decomposed sub-problems within process planning. For example, a great deal of work has been done in feature extraction defined over specific geometry domains (rotational parts, prismatic parts, etc.), and process domains (turning, three-axis machining, etc.). Because researchers have typically focused only on a particular aspect of process planning, there has been very little integration of efforts within this field. Due to the vast diversity of application domains in sub-problems contained within the area of process planning, it is very unlikely that a process planning system addressing all needs will be developed in the near future. However, a step towards the achievement of an integrated and flexible process planning system is the development of an enabling platform for process planning.

An enabling platform is a software environment which bridges the gap between the support provided by an operating system and that required by specialised software applications. A Process Planning Enabling Platform (PPEP), much like an expert system shell, would provide its users with a base level of functional support. The user then needs to add only the domain specific information (i.e. procedural knowledge, declarative knowledge, and relevant informational data). Thus, the software shell will enhance CAPP firstly by significantly increasing the flexibility with which a process planner can approach a problem, and secondly by allowing for integration between the various software.

The tasks within the process planning domain are industry or facility specific. This model allows planners from different process planning domains to specify a system to suit their requirements. The PPEP supports these differing requirements by

*Received 19 August 1993*

*Correspondence and offprint requests to*: V. Hetem, Department of Manufacturing, Bradley University, Peoria, IL 61625, USA

facilitating configuration of process planning systems through standardised data objects and library routines. The system will have to process geometric information, data, and knowledge, and will therefore address the following needs:

- Standard data objects and extensibility of this set
- The development of procedures
- The development of association relationships
- A research and development environment
- Integration of data and knowledge processing technologies

## 2. PPEP Development

### 2.1. Functional Specification

Process planning is well defined, but we concur with Eshel [1] that it is essential first to characterise process planning. Therefore, we propose the following parameters for the specification of a process planning system:

- *Process domain* which characterises the set of processes or technologies considered by the process planning system. Within a particular technology (such as machining), a system might consider only a sub-set of processes that are possible (such as turning).
- *Part domain* which characterises the geometries, materials and other characteristics of the parts. Though the part domain is limited by the process domain, process planning systems often only consider a subset of geometries producible by the processes in the process domain (such as axis-symmetric machined parts).
- *Activity domain* which characterises the activities, such as degree of automation, size of production facility and maturity of technology involved.
- *Specificity* which refers to the constraints with respect to a specific production faciltiy. A feasible process plan assumes access to unconstrained resources. A realisable process plan is one which can be executed with a given set of resources (i.e. a particular facility).
- *Resolution* which refers to the level of detail, such as low levels as in variant systems (details common to a class of parts), and high levels as in generative systems (details specific to a particular part).
- *Determinacy* which refers to the level of flexibility of the process plans. A process plan can be either a specific sequence of operations (deterministic), or flexible based on a precedence relationship between operations.

### 2.2. Domain Specification

The determination of the activity domain of a process planning system is not only related to what kinds of problems should be addressed, but also how the system should be set up. Eary and Johnson [2] compiled a list of activities for process planning which can be summarised as follows:

- Determine basic manufacturing processes.
- Determine the order of operations to manufacture a part.
- Determine tooling and gauging.
- Determine the equipment.
- Determine necessary part revisions originated by manufacturability analysis.
- Examine functioning of tooling and equipment.
- Estimation duration and cost of operations.
- Determine necessary part changes originated by time and cost analysis.

This list of process planning activities is considered in the development of the PPEP with the following qualifiers:

- It was compiled before the emergence of computer-aided manufacturing.
- Activities such as part understanding and inspection planning are implied.
- The sequential structure is not always feasible.
- It is based on the output required of a process plan.

The PPEP is based on an activity- or task-oriented view of process planning, resulting in the following six inter-related groups of activities [3]:

- *Part understanding*, which involves a description of a part in terms of patterns of basic geometric entities (surfaces, edges and vertices) that are of manufacturing relevance (features), as well as the non-geometric specifications and tolerances.
- *Process selection and ordering*, which is responsible for making associations between processes and the extracted features, as well as any partial ordering of these processes.
- *Machine-tool selection*, which identifies machine-tools from an available set based on the previous process selection.
- *Setup planning*, which is based on the feasible set of candidate processes and machine tools for

each feature, by refining this feasible space into groups of operations that can be machined in a single setup, as well as orientation and location schemes.

- *Operation planning*, which is the detailed planning of each operation, considering setup, tooling, operations parameters, and cut paths (machine tool programs).
- *Evaluations and documentation*, which is the evaluation (simulation) of the consolidated plan elements, and configuration control establishment.

Each of these activities is further decomposed into tasks which in turn are decomposed into sub-tasks, see reference [3] for details. What is evident is that a relatively large domain of process planning can be decomposed as per the activities mentioned above. The precise mechanisms for any activity or task might differ with domain, however similarities indicate that at some level of abstraction, general modelling tools can be utilised. The topic of this paper is therefore the identification of these modeling tools and their organisation into a system that provides the developer of a process planning system a level of support that enables configuration and customising. This system is what we call a Process Planning Enabling Platform.

### 2.3. Activity Specification

To develop and analyse the data flow and the relationships among the different activities, system analysis diagrams are used to illustrate the inputs, controls, uses, and mechanisms, to produce an output [3]. Discussion on mechanisms have been purposely avoided (in spite of recognising the interdependency of the input requirements and the mechanisms used to perform an activity) because they constitute the *how* rather than the *what* of process planning and are beyond the scope of the intended development platform. The system is described in terms of cascading activities (Fig. 1), but each activity can be further developed where necessary into secondary hierarchies of tasks and sub-tasks etc. (Fig. 2).

### 2.4. Software Specification

The PPEP software system [4] will support process planning development through application specific modules not provided by an operating system. The data exchange between the PPEP software modulus will be based on interfacing standards such as Product Data Exchange Specification (PDES) [5], and Application Interface Specification (AIS) [6]. The following categories of support will be required of the PPEP:

- *Geometric reasoning library*. A repertoire of routines to perform various geometric manipulations and tests which are often required by the activities in process planning.
- *Plan management system*. This system will allow for variant process planning by extracting, comparing, and facilitating modifications of existing process plans.
- *Plan consistency maintenance library*. A library of routines which allow segments of plans to be pieced together so that consistency between segments is maintained or inconsistencies are pointed out and/or resolved.
- *Plan space configuration*. A utility to allow the application developer to configure the domain over which the planning is to be done. This domain might involve the processes to be considered, the body of knowledge to be used, the machines to be considered, and other such application-specific definitions.
- *Empirical modelling facilties*. A great deal of data are obtained from the shop floor. This module essentially contains a set of routines to identify relationships between variables, possibly perform some amount of inference on the vast quantity of data, and develop qualitative associations between variables.
- *Operator interface and documentation facilities*.
- *Simulation and modelling facilities*. These facilities provide the user with various simulation and modelling capabilities, such as process simulation (to observe the values of the process variables and outputs), graphical simulations (for visualisations), economic estimations, dynamic modelling, and other such facilities.
- *Graphs, editing, file management*. This set of utilities allows the user conveniences that might not be available in standard operating systems.

### 2.5. System Specification

The multitude of relationships between tasks within process planning and the resulting inter-dependencies have been identified in Section 2.2. Consequently, the specification of the PPEP focuses on the support required to perform these tasks, and the control structure needed to integrate these tasks, such as the basic system components of data objects, utilities, and support libraries and routines.

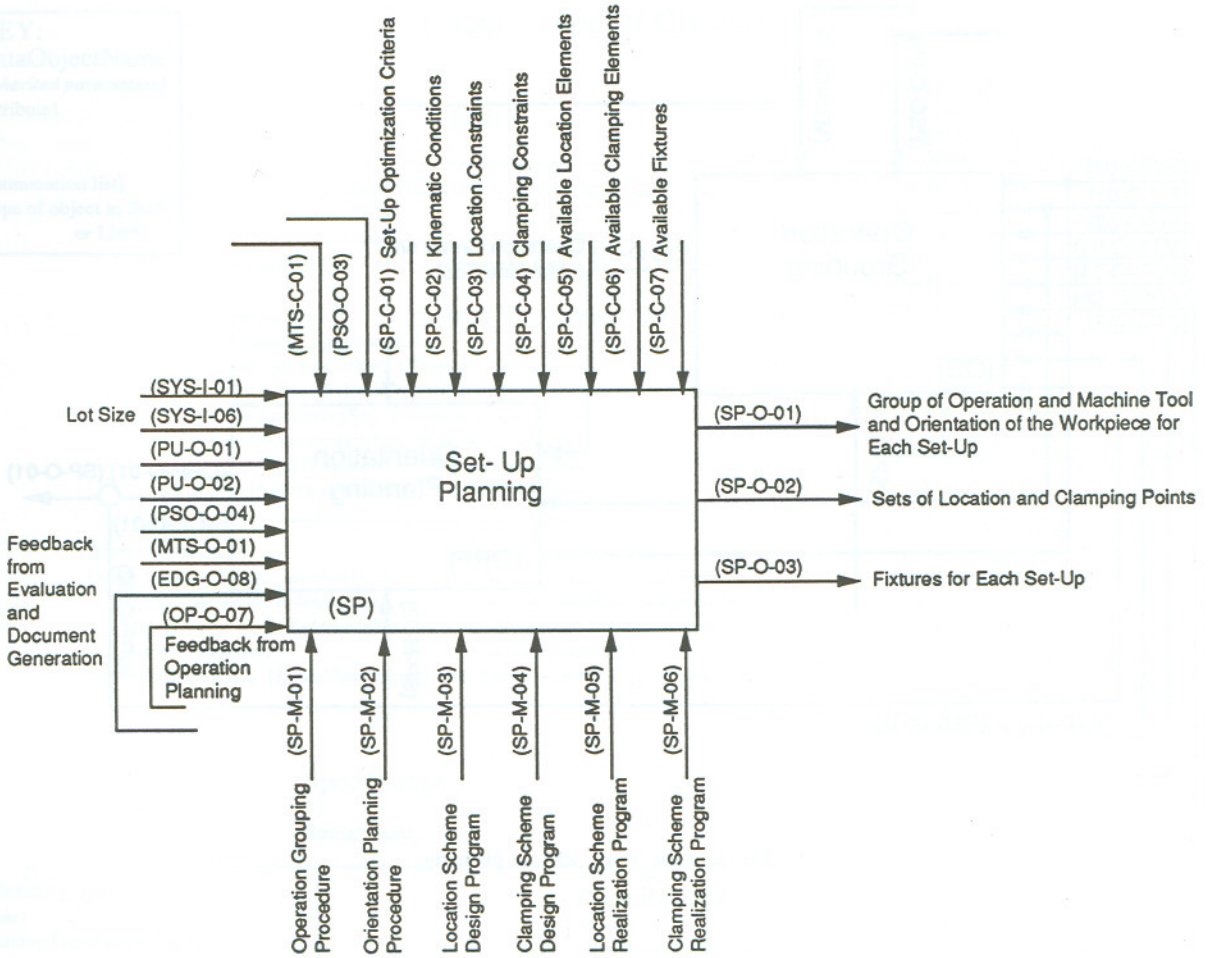


**Fig. 1.** Activity: setup planning.

## 2.6. Data Object Specification

The PPEP was developed using an object-oriented approach. These data definitions include the structure of the data objects and a specification of the various operations which can be performed on the objects. The data objects type within the PPEP are; fundamental, part, resource, process, plan, and domain.

### *2.6.1. Fundamental Data Objects*

Each object class is recursively sub-divided into smaller data components until the level of the fundamental data objects. For example, a planar face consists of some type of face identifier, the definition of the associated plane, a plane is identifier, a point on the plane, and a vector which is perpendicular to the plan.

The following are the most common data types: Boolean, Integer, Real, String, Vector3, Point3, Matrix3, Vector4, Point4, and Matrix4. Fundamental data types include a number of generic container classes, such as MinMax, List, Set, Bag, and Group.

### *2.6.2. Part Data Objects*

These data objects contain the information pertaining to the part. Process planning typically incorporates features, consequently the PPEP is able to maintain a feature-based description of the part. Since features are an abstract concept with different meanings in different domains, PPEP provides the process planner with a methodology by which features can be defined. The part data objects in PPEP are shown in a Type-Of-Hierarchy in Fig. 3.

### *2.6.3. Resource Data Objects*

The resource domain consists of both the physical resources and informational resources which are available to the process planner. Some examples of physical resources include machines tools, cutting

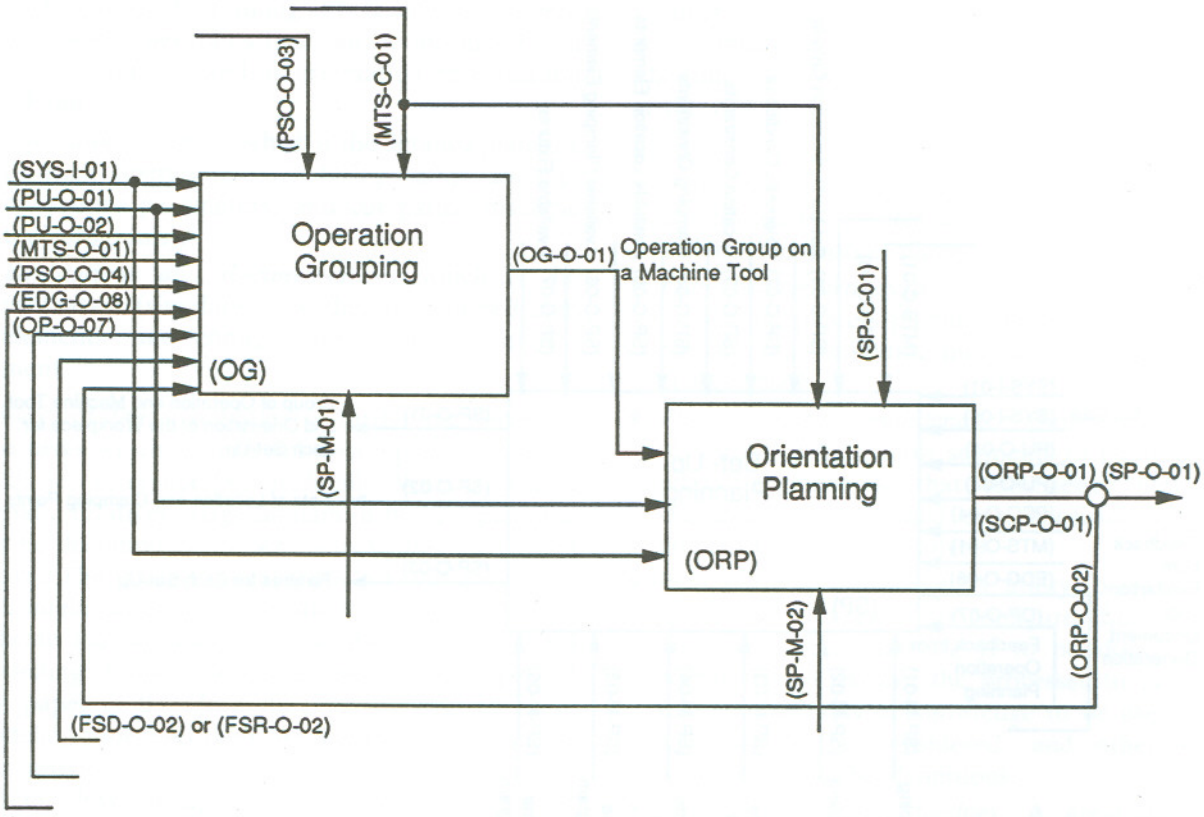


**Fig. 2.** Task: setup scheme planning.

tools, and fixtures. Informational resource includes lists of predetermined process plans. An example of a data object of the resource Machine Tool follows, as well as the Type-of-Hierarchy diagram in Fig. 4 depicting the resources structure:

*Definition*: a machine tool is a device that creates relative motion between a tool (or processing entity) and the part. It establishes the coordinate system that is the bases for the orientation between the machine tool, the work holding device (fixture), and the part.

*Object*: machine tool

*Attributes*:

| | |
|---|---|
| type: | (e.g. mill, drill, cmm, . . .) |
| facility: | (e.g. fabrication, heat treatment, assembly, . . .) |
| cell: | (e.g. fms #1234, transfer line #4321, group xyz . . .) |
| process association: | (e.g. hole making, cut off, contour milling, . . .) |
| configuration: | (e.g. axis type movement type travel resolution positioning acuracy repeatability clamp) |
| capability: | (e.g. degrees of freedom simultaneous axis control interpolation types volumetric accuracy) |
| fixture set: | (e.g. cube 800-wide × 800-deep × 500-high vice 150-gap chuck 3-jaw 250-diameter) |
| tool set: | (e.g. holder code: (e.g. PCLNR 1616H 12-M) insert code: (e.g. TNMM 16 03 04 S1P)) |
| options: | (e.g. pallet shuttling, chip remover coolant, . . .) |
| format detail: | (e.g. N3G2X +3.3Z +3.3I3 .3K3 .3T2M2) |
| post processor: | (e.g. GE-8M) |

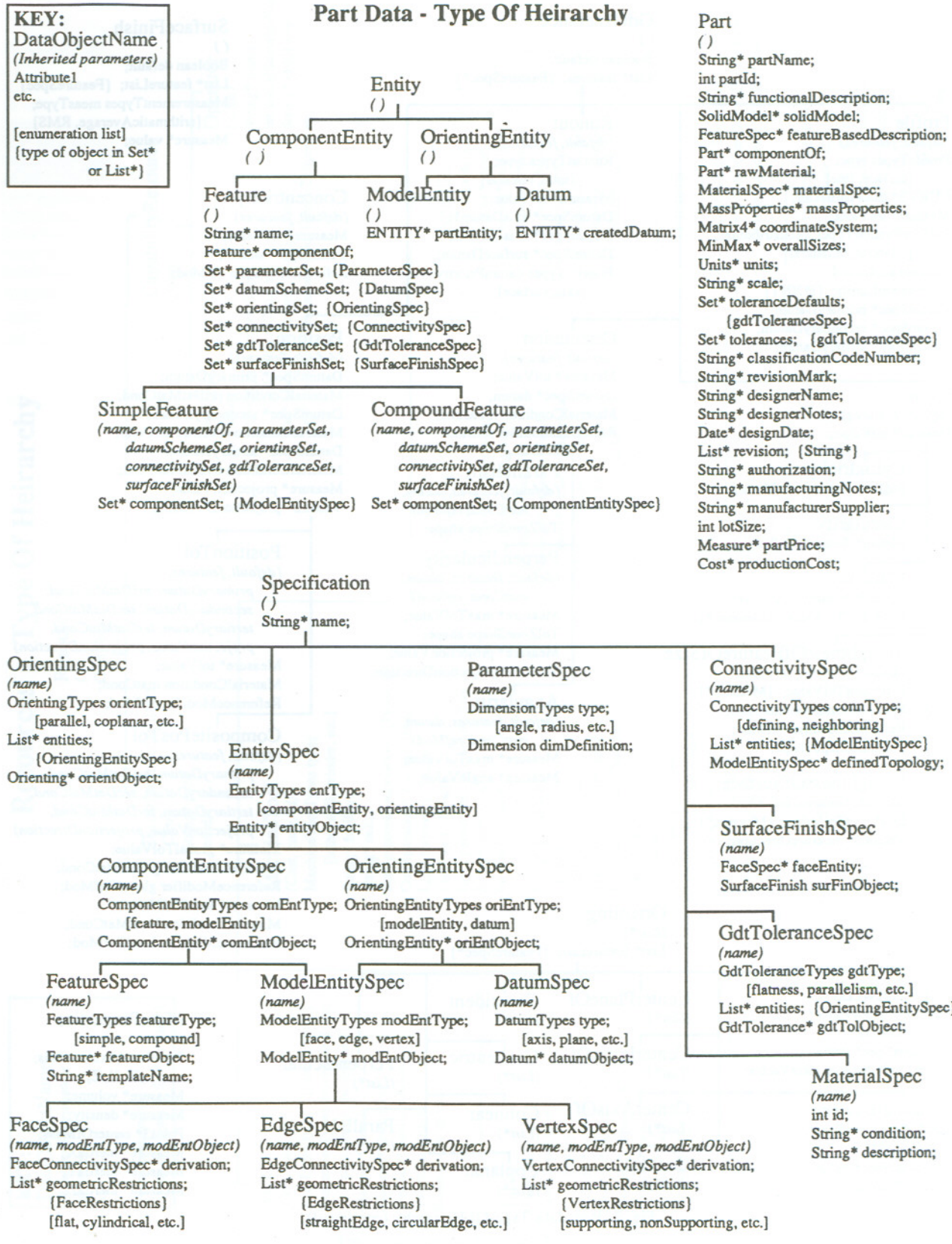


**Fig. 3.** Part data Type-Of-Hierarchy.

## Part Data - Type Of Heirarchy (continue)

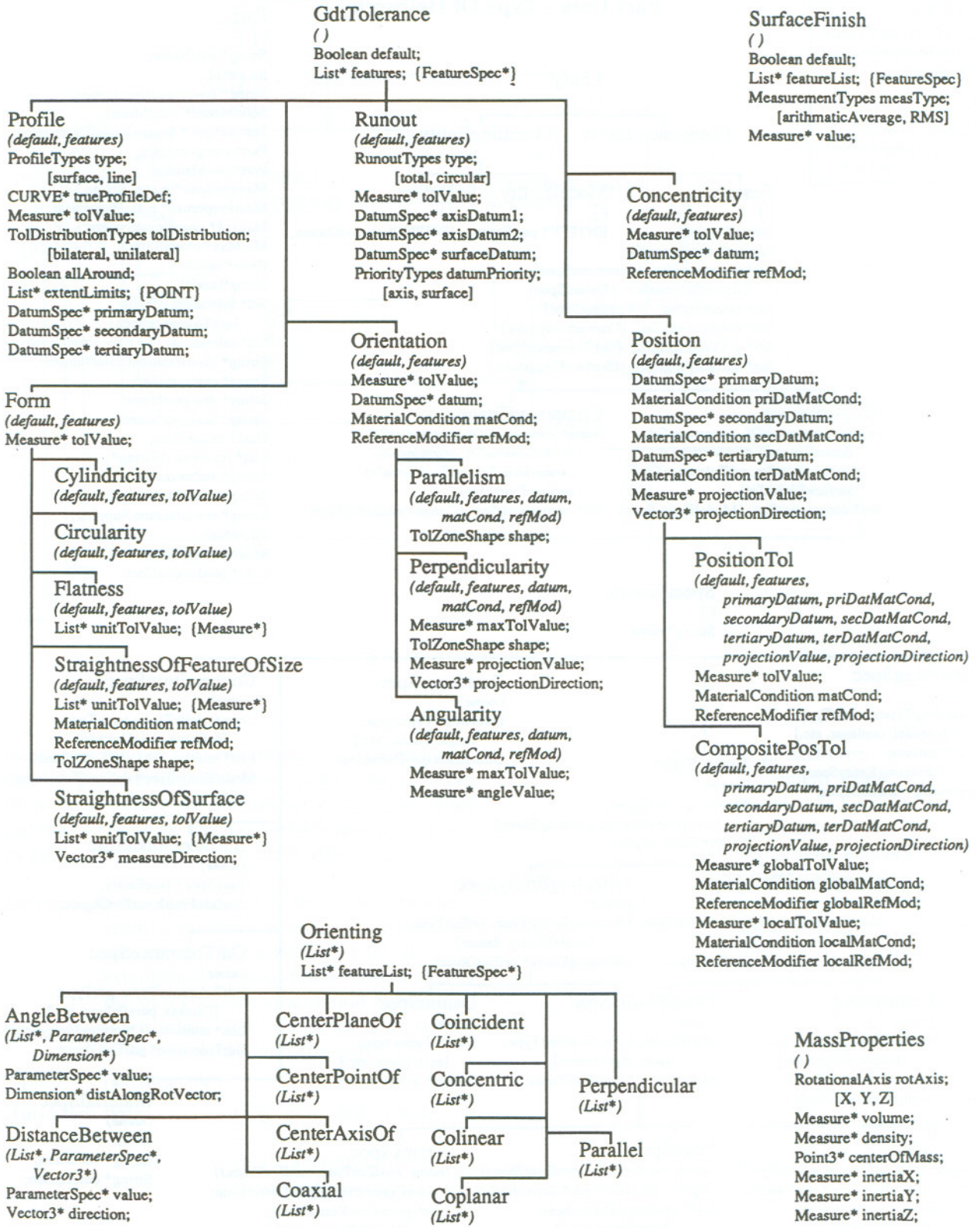


Fig. 3. Part data Type-Of-Hierarchy (continued).

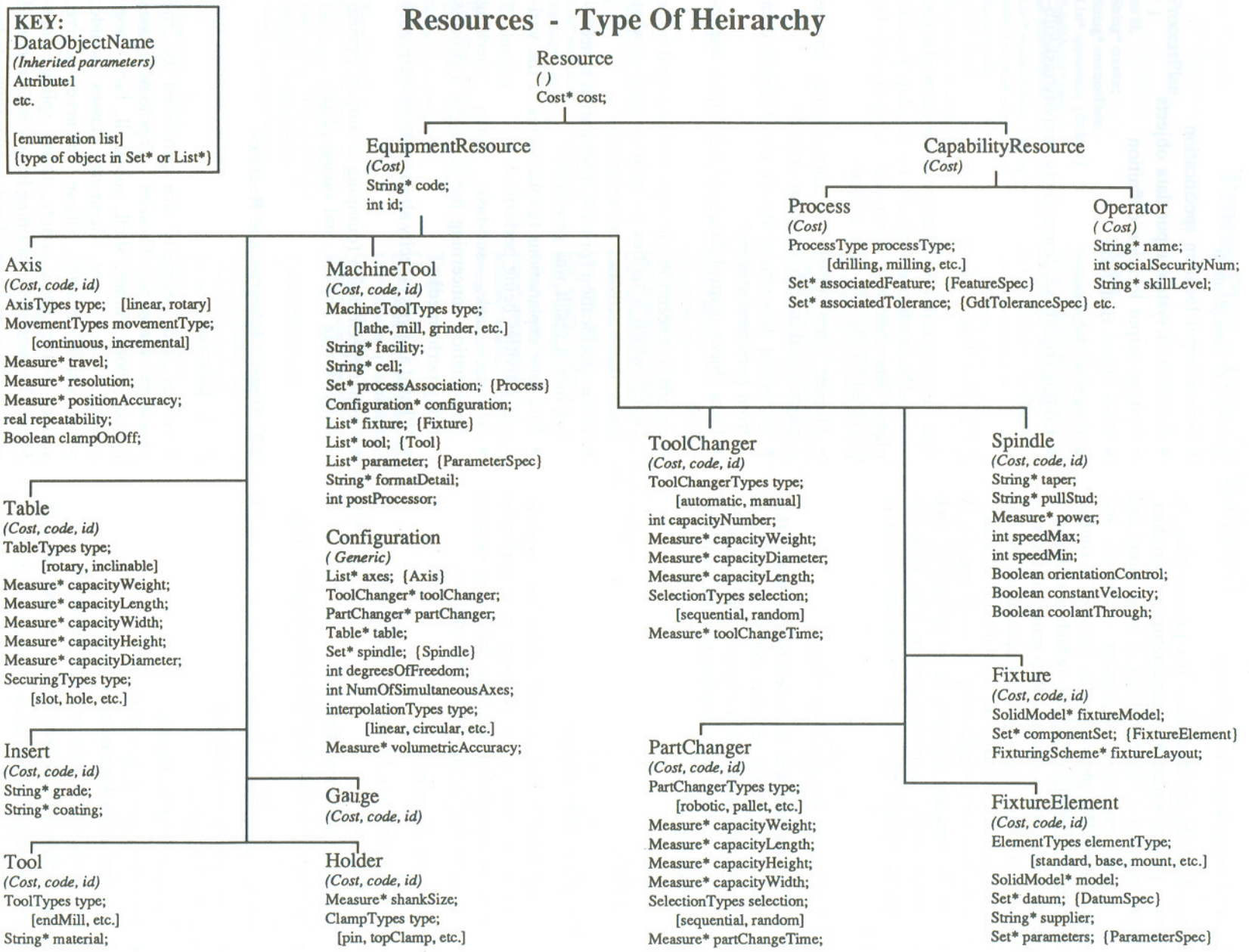


Fig. 4. Resources Type-Of-Hierarchy.

*2.6.4. Process Plan Data Objects*

These data objects document the process plan. A process plan is a plan for manufacturing a product, which includes a sequential set of operations and the related methods, equipment, material, and information. The plan is in the form of a routing comprising a group of setups, which consists of a group of operations comprising a group of tasks. A setup is an efficient grouping of operations on one machine and one fixture. An operation is a process element done at a single setup or work station. A task is the smallest element in an operation, such as a single tool to part engagement. The Type-Of-Hierarchy for the plan data objects are shown in Fig. 5.

*2.6.5. Domain Specific Data Objects*

This type of data was included to emphasise that the user of the PPEP has the ability to modify or define any additional data objects which may be required to perform process planning in a particular manufacturing domain.

**2.7. Utilities**

*2.7.1. Build-Time Utilities*

The supervisory system will allow the user to configure the system to perform in a particular process planning domain. The following support will be provided by the build-time utilities:

*Activity specification*

- Activity structure (activity/task/sub-task specification)
- I/O data object specification
- Mechanism association (inference engine, object code, etc.)
- Control data linking (rule-base, data-base, etc.)
- Intra-activity connectivity
- Aspect definition
- Run-time characteristics
- Help/explanation specification
- Run-time data logging specification

*Connectivity/system structural specifications*

- Association of I/O slots of activity/task/sub-task
- Relationship specification (direction, method, message definition)
- Definition of run-time characteristics (e.g. over-write, supervisor permission required, change only if different, etc.)

*System I/O specification*

- System input location specification
- Creation of system input data objects
- System output location definition
- Data-object definition
- Function definitions
- Editing (rule-base, connectivity, activity/task/sub-task)

*2.7.2. Run-Time Utilities*

The run-time utilities will allow the process planner to use the configured system to develop process plans. The following functional support will be provided by the run-time utilities:

- Manual data object creation/editing (create objects and modify exisiting objects)
- Importing data-object files (retrieve object data from persistent storage)
- Data object, control-data, and mechanism inspection
- Selective execution of activities/tasks/sub-tasks (select which activities to perform and in what order those activities are executed)
- Trace specification (provide information concerning how a result was generated)
- Run-time explanations (explanations of the various activities being performed)
- Activity, task, sub-task status checking (information concerning the status of a process being performed)
- What-if analysis (specify alternatives in temporary mode)
- Output specification (dumping of output, persistently storing objects, and saving plans)

**2.8. Support Libraries and Routines**

*2.8.1. Libraries*

Topology and geometry are processed by the geometric reasoning libraries. Topology classes include; body, lump, shell, sub-shell, face, loop, co-edge, vertex, wire. Geometry classes include; point, curve, straight, ellipse, intcurve, pcurve, surface, plane, cone, sphere, torus, spline, transform. The geometric reasoning library is a structured set of routines and procedures, which supports two types of activities:

- Queries about a particular configuration of the world whose results are booleans.

# ProcessPlan - Type Of Heirarchy

**KEY:**
DataObjectName
*(Inherited parameters)*
Attribute1
etc.

[enumeration list]
{type of object in Set* or List*}

### ProcessPlan
( )
int id;
String* creator;
String* creationDate;
List* approvals; {String*}
SolidModel* part;
int partRevision;
List* replacements; {String*}
MFGAnalysis* mfgAnalysis;
Route* route;
Units* units;
SolidModel* rawMaterial;
int productionVolume;

### Route
( )
Set* scheduleGroup; {Schedule}
String* mfgBillOfMaterials;
SolidModel* rawMaterial;
Set* setUpGroup; {SetUp}
RouteTypes routeType;
[series, parallel, etc.]

### SetUp
( )
IntermediateModel* partModelStatus;
MachineTool* machineTool;
Ste* operationGroup; {Operation}
Transformation* partOrientation;
Fixture* fixtureDesign;
Transformation* fixtureOrientation;
FixtureScheme* fixturingScheme;

### Schedule
()
int batchId;
String* startDate;
String* dueDate;
int quantity;

### FixturingScheme
( )
LocationScheme* locating;
ClampingScheme* clamping;

### LocationScheme
( )
*Set* surfaces;*
*{FixturingSurface}*

### ClampingScheme
( )
*Set* surfaces;*
*{FixturingSurface}*

### Operation
( )
SetUp* setUp;
String* description;
Set* taskGroup; {Task}
List* featuresAssociation; {FeatureSpec}
List* toleranceAssociation; {GdtToleranceSpec}
IntermediateModel* partModelStatus;
List* tool; {Tool}
List* guages; {Guage}
String* ncProgram;
String* qualityPlan;
String* operatorInstruction;
Measure* estimatedTimeStandard;
String* operatorGrade;

### FixturingSurface
( )
FACE* face;
List* position; {ContactPoint}
ContactTypes contactType;
[point, line, etc.]

### ContactPoint
( )
Point3* position;
Vector3* direction;
Measure* force;

### MFGAnalysis
( )
List* simulation; {Simulation}
List* evaluation; {Evaluation}
Metrics* metrics;

### Evaluation
( )
List* input; {ParameterSpec}
List* output; {ParameterSpec}
String* evaluationModel;
String* evaluationResults;

### Simulation
( )
List* input; {ParameterSpec}
List* output; {ParameterSpec}
String* simulationModel;
String* simulationResults;

### Metrics
( )
Measure* leadTime;
Cost* cost;
Measure* risk;
etc....

### Task
( )
List* parameter; {ParameterSpec}
List* geoMod; {GeometricModification}
List* nonGeoMod; {nonGeometricModification}
TaskInstruction* taskInstruction;

### TaskInstruction
( )
List* machineSetting; {MachineSetting}
String* magazineLoading;
String* offsetLoading;
String* fixtureLoading;
String* partLoading;
String* ncProgram;
String* operatorInstructions;
String* qualityPlan;

### MachineSetting
( )
String* name;
Measure* value;

### IntermediateModel
( )
SolidModel* modelStatus;
Set* performedOperation;
{Operation}
Set* performedTask; {Task}

### GeometricModification
( )
List* operands; {SolidModel}
BooleanOperations operator;

### NongeometricModification
( )
List* modifDescription; {String}

**Fig. 5.** Plan Type-Of-Hierarchy.

- Constructive routines that build objects which represent operations or concepts on the existing world, for example, projection of objects on other objects, ray tracing, convex hull construction, intersections.

The library is conceived to provide geometric objects (as extensions to standard programming entities), which are accompanied by operations which modify and administer them, find and test for relations among them, etc. To provide this set of objects and the functionalities among them, several layers of software are added on top of the standard utilities that a solid modelling package offers. The usual set of utilities that a geometric solid modelling package offers covers mainly boolean and constructive operations, based on a representation scheme, for example Boundary Representation (B-rep) of solids, Constructive Solid Geometry (CSG), etc. In general, the front end to the user includes manipulation of the objects in the world, but there is no interface provided which allows the testing of relationships between localised parts of the objects present, or to construct or manipulate the intermediate objects. In this sense, the geometric reasoning library is designed to allow access to the capabilities of the available software while at the same time providing functionality by an added layer of routines.

In the world there are topological objects such as vertices, edges, co-edges, faces, shells, bodies, etc., and geometrical objects such as points, planes, straights, curves, and mathematical entities such as matrices, homogeneous transformations, vectors, etc. A library of geometrical objects could also provide the manipulation of them, plus queries and constructions. Manipulations of objects are mostly done by using a set of linear transformations which represent shearings, translations, rotations, scalings, etc. In general, these operations are provided by a standard solid modeller, whereas another set of functions is rarely supplied, such as:

- Intersections of non-solid objects
- Intersection of objects with different dimensionality (e.g. edge versus face)
- Projections of objects on other objects (e.g. projection of faces on shells in some direction)
- Tests for co-linearity, parallelism, perpendicularity, etc.,
- Distances between objects following different criteria, etc.

Some examples of routines provided in the geometric reasoning module along with their functionality are:

- parallel (face, plane) → boolean
- parallel (face, curve) → boolean
- colinear (curve, edge) → boolean
- contained (curve, face) → boolean
- contained (curve, plane → boolean
- coincide (curve, curve) → boolean
- contained (vertex, face) → boolean
- contained (vertex, plane) → boolean
- project (curve, plane, direction) → curve
- project (curve, face, direction) → {edge}
- project (point, shell, direction) → {point}
- distance (point, curve) → real X vector
- distance (curve, curve) → real X vector
- intersection (face, face) → {edge}∪{face}
- intersection (edge, edge) → {point}∪{edge}
- mk_pl_perl_pl_ln (curve, plane) → plane
- (makes plane perpendicular to plane, containing a curve)
- mk_pl_bisect_pl_pl (plane, plane) plane
  (makes the plane which bisects angle between two planes)
- mk_face_pl_pls (plane, {plane}) → face
  (makes the face which is the convex intersection of {plane}, defined on a plane)

In the preceding examples, the functionality of a routine depends on its arguments rather than its name. Functionality can involve the projection of a curve onto a face which is a set of edges, or the intersection of two faces could be a set of edges if they are not co-planar, but if they are co-planar it could be a set of (non-connected) faces as in the intersection of two glove-shaped polygons. The fact that the functionality of a routine depends on its arguments rather than its name makes it desirable to use object-oriented programming and an overload of functions.

The library is being implemented in C++, in such a way that the user program has access to the high level library routines, as well as the objects in the world, which are administered by a solid modelling package.

### 2.8.2. *Routines*

The definition of support routines is divided into low-level geometric queries and constructions which work on top of the AIS interface to the solid modeller and include all AIS routines, and the routines which are more specific to process planning. A set of functionally described examples follow:

- *Axis-symmetry*: analyse any object and determine if the object has one or more axes of symmetry.
- *Monotonicity*: determine if an axis-symmetric object is monotonic and the direction of monotonicity.
- *Polyhedral*: determine if all of the object faces are strictly planar.
- *Convex hull*: generate the convex hull from the solid model of an object.
- *Coincidence*: determine whether a pair of topological, datum, or feature entities are coincident to one another.
- *Tolerance categorisation*: determine tolerance characteristics – smallest, type, etc.
- *Datum relationships*: determine precedence relationships between data of an object.
- *Fixture location scheme and part accessibility*: determine suitability of a pre-defined location scheme and the accessibility of non-fixturing surfaces.
- *Boxing algorithms*: determine the smallest box which completely contains an object.
- *Amount of material removed*: determine the amount of material removed in order to manufacture a feature.
- *Process-feature association*: formulate a process-feature association given a set of features and a database containing the process capabilities of a manufacturing facility.

## 3. Conclusion

The Enabling Platform is a significant step towards the development of computer-aided process planning systems. A layered software structure has been developed and the modules which will have to be incorporated have been identified. The specification of data objects has been initiated, and the hierarchical representation scheme has been used for product data. The geometric and topological representation of the object will be based on the AIS standard. A methodology for defining features so that a more abstract representation of the object is available for later phases of process planning has also been developed. Feasibility work has been completed in the development of the overall organisation of the software modules. This initial specification can be expanded to include the experience gained during system development and maintenance.

## References

1. Eshel G. Automatic generation of process outlines for forming and machining processes. PhD dissertation, Purdue University, W. Lafayette, IN, 1986
2. Eary DF, Johnson JE. Process engineering for manufacturing. Prentice-Hall, Englewood Cliffs, NJ, 1962
3. Ferreira PM, Lu SC-Y, Xhu X. A conceptual model for process planning. Preliminary report to CAM-I, 1250 E. Copeland Road, Arlington, Texas, August 1989
4. Ferreira PM, Lu SC-Y, Carr K, Hetem V, Lucenti M, Ruiz O, Zhu X. Specification for a manufacturing planning enabling platform. Final report to CAM-I, 1250 E. Copeland Road, Arlington, Texas, March 1992
5. Product data exchange specification. Manufacturing Technology Committee/Project (ISO TC184/SC4/WG3/P11), Document 4.1.3.2.1 (working paper), Chairman/Leader Greg A. Paul, c/o General Dynamics, Box 748, MZ 6480, Fort Worth, TX 76101, January 1991
6. Integrity Systems. Application Interface Specification (AIS) formalization. Final report, R-88-GMP-02, 2265 Manzana Way, San Diego, CA 92139, May 1990